Universidad del País Vasco Euskal Herriko Unibertsitatea

Informatika Fakultatea

Informatika Ingeniaritzako Gradua

▪ Gradu Amaierako Proiektua ▪

Software Ingeniaritza

GipuzkoaBidaian: Gipuzkoako eremuan garraio publikoz puntu batetik bestera bidaiatzeko Android APPa

---

Julen Apodaca Ibarguren

2014 - ekaina


informatika
fakultatea
facultad de
informática

Laburpena

Dokumentu honek gradu amaierako proiekturako garatu dudan Android aplikazioa, garraio publikoko bidaia bilatzailea, jorratzen du.

Azken urteotan Android sistema erabiltzen duten gailu kopurua ikaragarri handitu da, ez hori bakarrik, gero eta jende kopuru gehiagok smartphone bat dauka. Estatistika horiek ikusita erabilgarria izango zen aplikazio bat garatu nahi nuen. Merkatuan bidaia bilatzaile gehiegi ez zeudela ikusita neurea garatzea erabaki nuen, momentuan zeudenez desberdinduz.

Proiektu honen helburua puntu batetik bestera garraio publikoan bidaiatzeko dauden posibilitateak lortzea da.Horrez gain, aplikazioa Android bertsio guztientzat erabilgarria izatea ere xedea dut. APPak GipuzkoaBidaian du izena, soilik eremu horretan funtzionatzen duelako.

Hauek dira GipuzkoaBidaia-ren ezaugarri nagusiak:

- Bilaketa modu bat baino gehiago aukeran egongo da: azkarra eta zehatza.
- Aplikazioa osoa Internet gabe erabiltzeko aukera izango da.
- Erabiltzaileak bilaketa erradioa alda dezake.
- Garraio publiko batean edo bietan egin ahal izango dira bilaketak.
- Ailegatze ordu estimazio bat jar daiteke.
- Bidaia zuzenak soilik egitea aukeratu ahal izango da.
- Aplikazioa Android 2.2-tik aurrera dutenek erabili ahal izango dute.

Gaien Aurkibidea

Irudien zerrenda

IRUDIAK

Taulen zerrenda

TAULAK

Kode-zatien zerrenda

Kode-zatiak

1

Sarrera eta hasierako definizioak

Atal honetan proiektuaren sarrera azalduko da. Lehenengo bidaia bilatzaileei buruz hitz egingo da, nolakoak diren orokorrean eta GipuzkoaBidaian-ekin zertan desberdintzen diren. Ondoren, Android sistemaren azterketa bat egingo da.

Proiektuaren Helburuen dokumentuaren egitura
1. Zer da bidaia bilatzaile bat? 2. Android Sistema

1.1. Zer da bidaia bidatzaile bat?

Bere izenak adierazten duen bezala bidaiak bilatzeko sistema bat da. Bidaia bat bi urratsez osatzen da, oinezko zatiez eta garraioko zatiez. GipuzkoaBidaian aplikazioak soilik garraio publikoak erabiltzen ditu, hori baita bere helburua. Hala ere, oinezko ibilbideen informazioa ere ematen du.

Bidaia bat izan daiteke:

1. Jatorri puntu batetik geltoki batera oinez ibiltzea.
2. Hor garraio publiko batean igo.
3. Bidaiatu jaitsi behar den geltokira arte.
4. Irten.
5. Azkenik helmugaraino ibili.

Gerta liteke helmugara iristeko garraio bakarra nahikoa ez izatea, kasu horretan, azken urratsean helmugara mugitu ordez beste geltokira joango litzateke. Beste garraio batean igo, bidaiatu, jaitsi eta helmugara arte ibili.

Beste garraiobide bat hartzearen prozesu hau behin eta berriz errepikatu liteke, baina GipuzkoaBidaian-en gehienez 2 aldiz gertatuko da, Gipuzkoan zehar mugitzeko ez delako gehiago behar.

Mota hauetako aplikazioen hasieran erabiltzaileak bilaketaren parametroak definitzen ditu:

- Jatorria zein den. Hau bi eratan egiteko aukera dago, helbidea eskuz idatziz edo mapan batetik zuzenenean aukeratuz.
- Helmuga zein den. Jatorrian bezala egin daiteke.
- Finkatu bidaiaren iritsiera ordu zehatz batean izango den hala ez.
- Helduera ordua zehaztu.
- Bilaketa ze garraioekin egingo den: autobusa eta trena dira.

Bilaketa egin eta gero, emaitzak zerrenda batean agertuko dira, hauetako bakoitzak bidaia osoaren iraupena, distantzia, garraio publikoko konpainien izenak, hartu behar diren lineen izenak eta garraio kopurua adierazten du. Ez hori bakarrik, emaitzak ordenatzeko aukera ere badago.

Bat aukeratuta, bere informazio garrantzitsuena pantailaratuko da, alde batetik bidaiaren urratsen informazioa, oinezko zatietan jarraibideak eta garraio zatietan linearen informazioa eta geltokiak. Bestetik mapa batean bidaiaren ibilbide osoa irudikatuko da.

1.1.1. Gaurko merkatua

Proiektua egiten hasi baino lehen merkatuan zeuden antzeko produktuak bilatu nituen, orduan ikusi nuen ez zeudela mota honetako hainbeste aplikazio. Hauek probatu nituen, beraien alde onak kontuan hartuz eta txarrak ez egiteko apuntatuz.

Ez hori bakarrik, konpetentziak aztertu aurretik garbi nuen GipuzkoaBidaian-ek bidaia posible guztiak (ezarritako limiteen barruan) bilatzea izango zuela helburu, ez soilik azkarrena edo motzena.

1.1.1.1. Google Maps

Google Maps Interneteko webgunea izateaz gain, gailu hau mugikorrentzako aplikazio bat da. Proiektua hasi zenetik bukatu arte aplikazioa eguneratu dute, funtzio berriak gehituz eta daudenak hobetuz. Esan behar da APP-a ikaragarria dela, denetatik daukala eta gainera bilaketak oso azkar egiten dituela. Hasieratik garbi neukan Google-ren produktua baino hobeagoa egitea zaila izango zela, dena den nire helburua ez da izan Google Maps baino hobeago den produktu bat sortzea.

Hala eta guztiz ere, gustatzen ez zaidan zerbait du: oinezko distantzia aukeratu ahal izatea. Gerta liteke niretzako onargarria den distantzia bat sistemarentzat egia ez izatea, adibidez, aplikazioak esaten du tarte bat garraio publikoz egin behar dudala, nik, berriz, oinez egingo nuke.

Arazo hau konpontzeko distantzia aukeragarria izatea inplementatu dut, era horretan erabiltzaileak oinez ibiltzeko prest dagoen distantziaren arabera aldatzeko aukera du

1.1.1.2. GipuzkoaTransit

Aplikazio honen itxura eta GipuzkoaBidaian-ena oso antzekoak dira. Bere funtzionalitateak ere oso antzekoak dira, biek egin dezakete:

- Garraio mota aukeratu.
- Oinezko distantzia aldatu.
- Bidaia zuzenak soilik egiteko aukera eman.
- Bidaia baten informazioa testuan eta mapa batean bistaratu.

Hala ere, badaude hainbat xehetasun zuzenegiak ez direnak eta GipuzkoaBidaian-en zuzenduta daudenak:

- Jatorria eta helmuga aukeratzeko mapa oso kaxkarra da, asko kostatzen zaio kargatzea eta ez da batere erosoa. GipuzkoaBidaian-ek ordez, Google Maps-ekoa erabiltzen du, pribatua izan arren askoz indartsuagoa baita beste edozein baino.
- Jatorria eta helbidea eskuz idazten badira, testu eremuak idatzitakoaren araberako posibilitateak erakutsi arte itxaron behar da, bestela ez dio kasurik egiten idatzitakoari. Hau akats larria da, erabiltzaileak ez du zergatik jakin behar idazten duenetik itxaron behar duela aukerak agertu arte. Hau konpontzeko GipuzkoaBidaian-ek antzeko gauza egiten du, bilatzeko botoia ukitzean sistemak

testu eremuak hartu eta horien arabera helbide aukerak ematen dio erabiltzaileari, horrela ziurtatzen da beti zerbait aukeratuko dela.

- Emaitzak ordenatu ahal ez izatea.
- Bidaia baten informazioa bistaratzean ez da erraz ikusten mapan ikusteko aukera ere dagoela (edo jarraibideak ikustea mapako interfazean egonda).

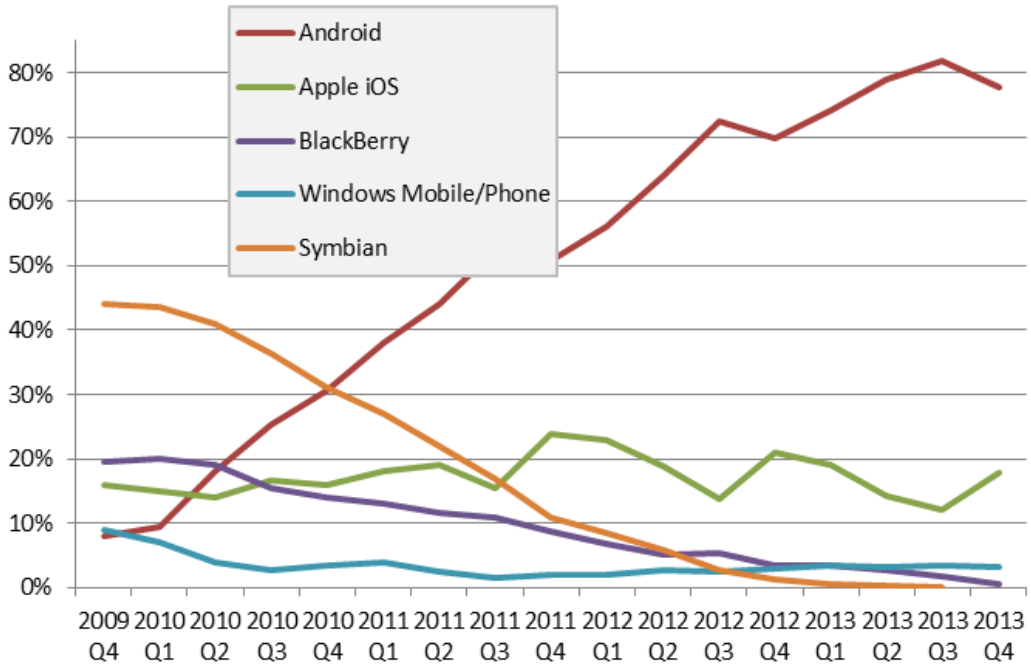
1.1.1.3. Moveuskadi

GipuzkoaTransit garatu duen konpainia bera da sortu duena, hortaz funtzionalitate eta interfaze guztiek egitura bera jarraitzen dute.

Hala ere, ez dut lortu aplikazio hau funtzionatu dezala. GipuzkoaTransit ondo funtzionatu duen arren, honek erroreak besterik ez ditu ematen. Adibidez, Bilbotik Donostirako edo Lasartetik Donostirako bidaiak ez ditu ondo bilatzen. Ondorioz, GipuzkoaBidaian sortzerakoan ez da aplikazio hau kontuan hartuko.

1.2. Android sistema

GipuzkoaBidaian Android sistema eragilea duten gailuentzako garatua izan da, erabaki hau bi arrazoiengatik hartu da gehien bat, dagoeneko ezagutzaren bat edo beste nituelako eta etorkizun gehien duen sistema delako. Hala ere, mugikorrentzako programatzen ikasi nahi dudalako ere egin dut.



1.1. irudia: Munduan saldutako smartphonen portzentajea sistema eragilearen arabera (iturria: "El gran libro de Android 3º edición, 2013")

1.2.1. Android-en garrantzia

Android merkatuan atera zenetik paregabeko hazkuntza izan du gaur egunera arte. 1.1. irudian ikusi daiteke Gratnet Group enpresak egindako ikerketa, non argi ikusten den azken urteotan Android-ek %75 baino gehiagoko merkatu-kuota lortu duela. Besteak, Symbian adibidez, hasieran kuota oso altua izan arren, azkeneko urteotan ia gailu bakar batek ere ez du erabiltzen. Apple iOS-ek ordea, antzeko maila mantendu du denbora osoan.

Estatu mailan antzeko joera dago, Android sistema eragileak espainiar merkatua menperatzen jarraitu du, horrela adierazten du Kantal Worldpanel kontsultariak 1.1. taulan.

Sistema eragileen kuota Smartphone berrietan	Abendua-Otsaila 2012	Abendua-Otsaila 2013
Android	%70,9	%92,1
iOS	%5,9	%4,4
Symbian	%10,5	%1,1
RIM	%10,2	%1,0
Windows	%2,5	%0,9
Besteak	%0,0	%0,5
Guztira *Smartphoneak*	%100	%100

1.1. taula: Espainiako sistema eragileen kuota smartphone berrietan (iturria: http://www.kantarworldpanel.com/es/Noticias/Android-ya-est-en-9-de-cada-10-nuevos-smartphones)

1.2.2. Hasierako egoera

Proiektua hasi baino lehen Android-i buruz nituen ezagutzak oso mugatuak ziren, unibertsitatean hirugarren kurtsoko irakasgai batean soilik pare bat gauza erakutsi zizkiguten. Momentu horretan gustua hartu nion eta hurrengo lauhilekoan beste irakasgai batentzat praktika bat egin nuen.

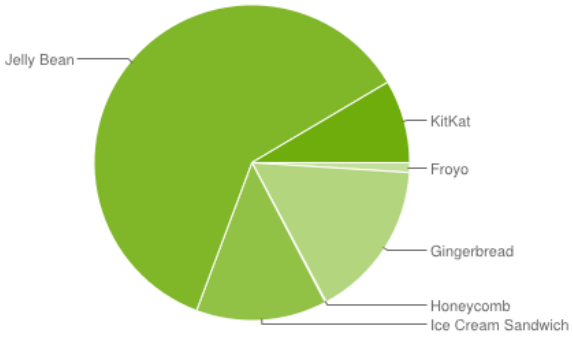
Hala ere, interes handia nuen Android munduan eta mugikorrentzat programatzen ikasi nahi nuen, horregatik proiektu hau egitea erabaki nuen.

Hasiera oso motela izan zen dena berria zelako, nahiz eta programazio lengoaia Java izan. Android sistemaren arkitektura ulertzea, interfazeak nola funtzionatzen duten, direktorioen egitura ulertzea eta abar asko kostatu zitzaidan. Autoikaskuntza prozesu bat izan dela esan daiteke.

1.2.3. Helburuak

Proiektua programatzen hasi baino lehen erabaki oso garrantzitsu bat hartu behar izan zen, aplikazioaren API maila minimoa. Erabaki honek APP-aren garapena mugatuko zuen, ez baita berdina 8 eta 18 mailarentzat programatzea.

Bertsio	Izengoiti	API	Banaketa
2.2	Froyo	8	%1.0
2.3.3 - 2.3.7	Gingerbread	10	%16.2
3.2	Honeycomb	13	%0.1
4.0.3 - 4.0.4	Ice Cream Sandwich	15	%13.4
4.1.x	Jelly Bean	16	%33.5
4.2.x		17	%18.8
4.3		18	%8.5
4.4	KitKat	19	%8.5



1.2. taula eta 1.2. irudia: 2014-ko Maiatza arte API mailen erabilpen portzentajeak (iturria http://developer.android.com/about/dashboards/index.html).

GipuzkoaBidaian aplikazioak dituen interfazeetarako ez da beharrezkoa API maila altua jartzea, maila baxuko bateragarritasun liburutegiak daudenez ia elementu guztiak erabil daitezke. Errazagoa da azken bertsioetarako programatzea, baina orduan mugikorren hamarrenak ezingo lukete aplikazioa erabili. Gainera, programatzaile bezala nire helburua da ahalik eta erabiltzaile kopuru handienak aplikazioa erabili ahal izatea, nahiz eta horrek lan karga handiagoa suposatu.

1.2. taularen eta 1.2. irudiaren informazioaren arabera soilik %1 Android 2.2 bertsioa erabiltzen dute, hain kopuru txikia denez maila minimoa 2.3.3 izango da, hau da, API 8.

2

Proiektuaren Helburuen Dokumentua

Kapitulu honetan proiektuaren planifikazioa azaltzen da. Proiektuaren irismena, denboraren planifikazioa, kalitatea, komunikazio plana eta arriskuen kudeaketa plana.

Proiektuaren Helburuen dokumentuaren egitura
1. Proiektuaren irismena 2. Denboraren planifikazioa 3. Kalitatea 4. Komunikazio plana 5. Arriskuen kudeaketa plana

2.1. Proiektuaren irismena

Atal honetan proiektuaren irismena deskribatuko da, hau da, proiektuaren betekizunak azalduko dira, LDE diagrama batek egin beharreko lana deskonposatuko du, proiektuko emangarriak zerrendatuko dira eta azkenik mugarriak identifikatuko dira.

Proiektuaren helburua bidaiak garraio publikoz egiteko bilatzailea garatzea da. Aplikazioak nagusiki hiru atal izango ditu:

- Bilaketaren parametro zehazpena:
    - Helbideak mapatik lortu.
    - Bestelako datuak eskuz sartu.
- Bidaia bilaketa:
    - Google Directions API-arekin.
    - Open Data eta Dbus.
    - Lokala.
- Emaitzen informazioa erakutsi:
    - Emaitzen zerrenda ikusi eta ordenatu ahal izatea.
    - Bidaia baten jarraibideak eta informazio garrantzitsuena erakutsi.
    - Ibilbidea mapa batean erakutsi.

GipuzkoaBidaian hiru hizkuntzatan egongo da: euskaraz, gaztelaniaz eta ingelesez. Hau automatikoki aukeratuko da gailuaren hizkuntzaren arabera. Ezezaguna izango balitz edukia ingelesez erakutsiko litzateke.

Aplikazioa Android sistemarentzat garatuko da eta itxura hurrengoa izango da:

- Orokorrean: pantailen atzealdeak kolore finko bat izango du. Android bertsio berrietan botoiak eta testu eremuak erdi-gardenak izango dira, bertsio zaharretan ordea, opakuak.
- Hasierako interfazea:
    - Jatorria eta helmuga idazteko aukera emango da.
    - Helbideak mapa baten bidez lortzeko interfazera joateko aukera, jatorriarentzat eta helmugarentzat.
    - Horren azpian bilaketa orduaren arabera egiteko aukera emango da eta berarekin uneko ordua idazteko posibilitatea.
    - Behean garraio publikoko aukerak, bakoitzak irudi bat izango du, batek autobus bat eta besteak tren bat.
    - Azpialdean bidaia bilaketa hasteko posibilitatea emango da.
- Helbideak mapan lortzeko interfazea: soilik bi elementu edukiko ditu, mapa bat, ia toki osoa okupatzen, eta helbidea gordetzeko aukera. Puntu batean ukituz gero, zooma egingo da eta ikono bat agertuko da leku horretan, informazioa lortu dela adieraziz.
- Doikuntzak pantaila: honen itxura Android doikuntzetarako duena izango da, bi elementu egongo dira: oinezko distantzia maximoa eta bilaketa modua.

- Bilaketa emaitza interfazea: goialdean emaitza zein eratan ordenatua dagoen adieraziko da eta ordenatzeko botoi bat. Ondoren, lista bat bilaketaren emaitzekin. Zerrendaren elementu bakoitzak bidaiaren informazio orokorrena erakutsiko du: distantzia totala, iraupen totala, hartuko diren lineak, bere konpainiak eta garraio kopurua.
- Bidaia baten emaitza bistaratzeko interfazea:
    - Jarraibideak erakusten duen atala: hasieran testu eremu bat egongo da jatorri helbidea adieraziz eta bukaeran berdina baina helmugarekin. Tartean bidaiaren urratsak idatziko dira: oinezko zatiak eta garraioko zatiak.
    - Mapa atala: hemen mapa bat egongo da besterik ez. Ibilbidea adieraziko da eta puntu garrantzitsuenak seinalatuko dira (jatorria, helmuga eta oinezko eta garraioko zatien hasiera).

2.1.1. Betekizunak

Aplikazioaren garapena proiektuaren atal garrantzitsuena izan arren, ez da ahaztu behar beste hainbeste zeregin daudela bere inguruan: planifikazioa, teknologien azterketa, kudeaketa, proiektuaren memoria eta defentsa. Hau hobeto ulertzeko 2.1 taulan atal bakoitza zehaztuta agertzen da.

1. Proiektuaren planifikazioa
1.1. Proiektuaren irismena
1.2. Aplikazioaren nondik norakoa
1.3. Ataza eta betekizunak listatu
1.4. Aurreikusitako denbora kalkulatu
1.5. Atazen egutegi posiblea
1.6. Arriskuen kudeaketa
1.6.1. Identifikazioa
1.6.2. Ekiditeko plana
1.7. Kalitate plana
1.8. Komunikazio plana
2. Teknologiaren azterketa
2.1. Lan gunea prestatu
2.2. Garraio publikoen zerbitzarien erabilpena aztertu
2.3. Garraio publikoen webguneen azterketa
2.4. Google Maps API
2.4.1. Funtzionamendua aztertu
2.4.2. Google-ek eskaintzen dituen API-en azterketa
2.5. Datu Basea inplementatu
2.6. Garraio konpainien azterketa
2.6.1. Laguntza eskaera
3. Aplikazioa
3.1. Izena asmatu eta logotipoa diseinatu
3.2. Interfazeak
3.2.1. Interfazeak diseinatu
3.2.2. Interfazeak sortu
3.3. Helbideak lortu maparen bidez
3.4. Bidaiak bilatu

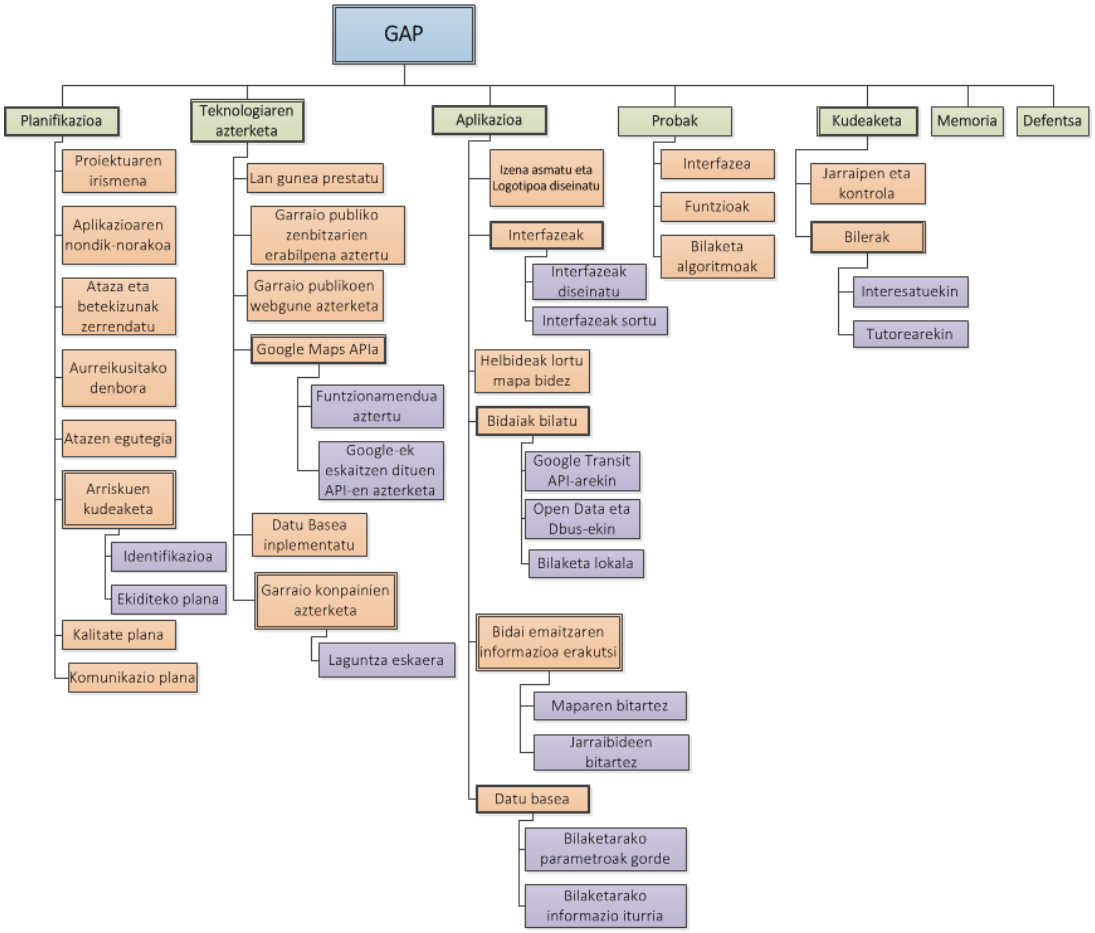
3.4.1. Open Data eta Dbus-ekin
3.4.2. Google Transit API-arekin
3.4.3. Bilaketa lokala
3.5. Bidaia emaitzaren informazioa erakutsi
3.5.1. Jarraibideen bitartez
3.5.2. Maparen bitartez
3.6. "Datu basea"
3.6.1. Bilaketarako informazio iturria
3.6.2. Bilaketarako parametroak gorde
4. Probak
4.1. Interfazea
4.2. Funtzioak
4.3. Bilaketa algoritmoa
5. Kudeaketa
5.1. Jarraipen eta kontrola
5.2. Bilerak
5.2.1. Tutorearekin
5.2.2. Interesatuekin
6. Proiektuaren memoria garatu
7. Defentsa

2.1. taula: Proiektuaren atazak

2.1.2. LDE diagrama

2.1. irudian proiektuaren lan deskonposaketa irudikatzen dira. Hauek dira elementu nagusien azalpenak:

- GAP: Gradu Amaierako Proiektua.
- Planifikazioa: hasieran proiektuan zehar bete behar diren helburuak definitzen dira, hala nola, irismena, kalitate maila, atazen kronograma eta abar.
    - Proiektuaren irismena: proiektuak zertan datzan azalpena, zein den ideia, aplikazioaren helburua eta abar.
    - Aplikazioaren nondik norakoa: GipuzkoaBidaian-en funtzionalitateak, izango duen itxura eta nola funtzionatuko duen.
    - Atazak eta betekizunak zerrendatu.
    - Aurreikusitako denbora: proiektuaren atal bakoitza egiteko beharrezkoa izango den denboraren estimazioa.
    - Atazen egutegia: proiektua kurtsoan zehar nola banatuko da, ataza bakoitza noiz hasi eta bukatu beharko den zehaztu.
    - Arriskuen kudeaketa: proiektua garatzen den heinean, hau trabatu edo bertan behera utzi dezaketen arrazoiak identifikatu behar dira eta kontrako neurriak hartu ezer gertatu ez dadin.
    - Kalitate plana: aplikazioa, memoria eta defentsarako gardenkiek izan behar duten kalitate maila minimoa, egokia eta hoberena definitu behar dira.
    - Komunikazio plana: proiektuaren interesatuekin nola komunikatu definitu behar da.



2.1. irudia: Proiektuaren lan deskonposaketa irudikatzen duen LDE diagrama

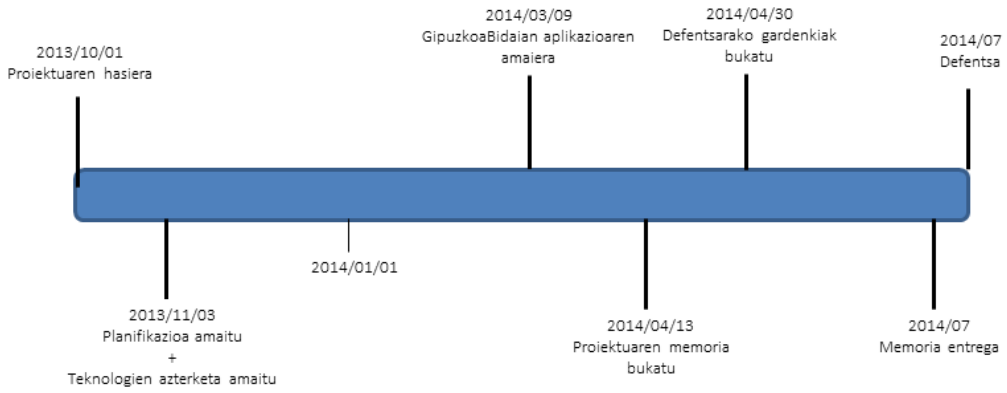
- **Teknologien azterketa**: aplikazioa garatzen hasi baino lehen erabiliko diren tresnak erabaki behar dira, informazio iturria lortu eta existitzen diren Interneteko zerbitzuak ezagutu.
    - **Lan gunea prestatu**: Android-entzat programatzeko hainbat programazio ingurune existitzen direnez, lanean hasi baino lehen hauen analisia egin behar da egokiena aukeratzeko.
    - **Garraio publikoko zerbitzarien erabilpena aztertu**: garraio publikoen informazioa eskaintzen duten zerbitzariak daudela jakinda, eskuratu daitekeen informazioa analizatu behar da.
    - **Garraio publikoen webgune azterketa**: webguneetatik informazioa lortzeko posibilitatea egon daitekeenez, aztertu behar diren tokiak zerrendatu behar dira eta sakonki begiratu.
    - **Google Maps APIa**: Google-ek hainbat web zerbitzu eskaintzen ditu, hauen azterketa egin behar da erabakitzeko zeintzuk eta nola erabiliko diren aplikazioan.
    - **Datu basea inplementatu**: aplikazioak erabiliko duen informazioa biltegiratu daitekeen aztertu behar da, zer den egokia eta zer ez.
    - **Garraio konpainien azterketa**: beraien kontaktuan jarri laguntza eske.

- Aplikazioa: GipuzkoaBidaian aplikazioaren garapena. Sei ataletan banatu daiteke:
    - Izena asmatu eta logotipoa diseinatu.
    - Interfazea: diseinatu eta sortu. Hasieran, pantaila guztiak paperean marraztu, hauen arteko trantsizioak definitu eta azkenik kodera pasa.
    - Helbidea lortu maparen bidez: aplikazioaren zati garrantzitsu bat da. Lehenik mapa aplikazioan nola jartzen den ikasi behar da.
    - Bidaiak bilatu: aplikazioaren mamia. Hiru eratan egin daiteke, Google-eko zerbitzuak erabiliz, konpainien informazioa zuzenean erabiliz edo mugikorrean biltegiratuta dauden datuak erabiliz.
    - Bidaia emaitzaren informazioa erakutsi: lehenik zer erakutsiko den erabaki behar da, non eta nola. Bi eratan izango da, testuz eta mapaz. Azkenik zati hau inplementatu.
    - Datu basea: biltegiratzea egokia den informazioa aztertu behar da, datu basearen UML-a sortu eta inplementatu.
- Probak: aplikazioa garatzen den heinean hainbat proba egin behar dira denak ondo funtzionatzen duela ziurtatzeko, ez hori bakarrik terminal desberdinetan ondo ikusten dela ere egiaztatu behar da.
- Kudeaketa: proiektuan zehar egindako lanaren kudeaketa.
    - Jarraipen eta kontrola: egindako ataza bakoitzaren kudeaketa, izan ditudan arazoen kontrola eta behar izan den denbora.
    - Bilerak: proiektuan zehar hainbat bilera egingo dira, batzuk berriemaileak eta besteak aplikazioaren inguruko erabakiak hartzeko. Bi pertsona motarekin izango da, tutorearekin eta interesatuekin. Interesatuak jarri arren agian ez da egongo beste hirugarren pertsonarik, hala ere, suposatuko da norbait egongo dela.
- Memoria: Dokumentu hau garatu.
- Defentsa: Defentsarako gardenkiak prestatu. Baita entseguak egin ere.

2.1.3. Emangarriak

Proiektuan zehar bi emangarri mota sortuko dira:

- Aplikazioarekin zerikusia dutenak: produktua bukatzerakoan hiru emangarri sortuak izango dira:
    - GipuzkoaBidaian APK, hau da, instalatzailea.
    - Fitxategia instalatzeko eskuliburua.
    - Aplikazioaren eskuliburua.
- Proiektuarekin orokorrean zerikusia dutenak:
    - Proiektua garatzeko plana.
    - Proiektuaren memoria.
    - Proiektuaren defentsarako gardenkiak.



2.2. irudia: Proiektuaren mugarriak

2.1.4. Mugarriak

2.2. irudian proiektuaren mugarriak diagrama batean irudikatuta daude. Estimazio horiek hasieran egindakoak dira eta ez dute islatzen bukaeran izandakoa.

2.2. Denboraren planifikazioa

Atal honetan proiektuan landuko den denboraren planifikazioa egingo da.

2.2.1. Atazak

2.1. irudiaren LDE-an proiektuaren ataza guztiak agertzen dira. Atal bakoitza xehatuta agertzen da, baina horrek ez du esan nahi azpi-atal guztien denbora estimazioa egin dela. “Aplikazioa” atalean ordea, hain garrantzitsua denez, hainbeste denbora beharko denez osatzeko, azpi-atal guztien estimazioa egitea beharrezkoa da.

2.2.2. Atazen estimazioa

Atal honetan proiektuaren atal bakoitza burutzeko estimatzen den denbora adierazten da:

Planifikazioa: **30 ordu**. Proiektuaren hasieran egindako estimazioa.

Teknologiaren azterketa:

- Lan gunea prestatu: 7 ordu.
- Garraio publiko zerbitzarien erabilpena aztertu: 5 ordu.
- Garraio publikoen webguneen azterketa: 6 ordu.

- Google Maps API: 9 ordu.
- Datu basea inplementatu:6 ordu.
- Garraio konpainien azterketa: 2 ordu.

Guztira estimatzen da 32 ordu lan izango direla. Atal honek bere garrantzia du, hemen hartzen diren erabakien arabera ondorengo lana arinagoa edo astunagoa izango da.

Aplikazioa:

- Izena asmatu eta logotipoa diseinatu: 3 o.
- Interfazeak:
  - Diseinatu: 6 ordu.
  - Sortu: 20 ordu.
- Helbideak lortu mapa bidez: 4 o.
- Bidaien bilaketa:
  - Google Transit API-arekin: 20 ordu.
  - Open Data eta Dbus-ekin: 30 ordu.
  - Bilaketa lokala: 30 ordu.
- Bidaia emaitzaren informazioa erakutsi:
  - Jarraibideen bitartez: 20 ordu.
  - Maparen bitartez: 20 ordu.
- Datu basea:
  - Bilaketarako parametroak gorde: 3 ordu.
  - Bilaketarako informazio iturria: 10 ordu

Guztira estimatzen da 166 ordu izango direla. Aplikazioa garatzea lan gogorrena izango da dudarik gabe.

Probak:

- Interfaze probak: 10 ordu.
- Funtzioen probak: 5 ordu.
- Bilaketa algoritmoa: 15 ordu.

Guztira estimatzen da 30 ordu izango direla. Funtzioen probak egiteko ez da beharrezkoa izango denbora gehiegi, aplikazioa garatzen den bitartean hauek behin eta berriro probatu behar izango direlako.

Kudeaketa:

- Jarraipena eta kontrola: 15 ordu.
- Bilerak:
  - Interesatuekin: 5 ordu.
  - Tutorearekin: 15 ordu.

Guztira estimatzen da 30 ordu izango direla. Jarraipena eta kontrola proiektuaren prozesu osoan pixkanaka-pixkanaka egingo diren atazak dira, ondorioz, ez da beharrezkoa izango denbora asko egotea.

Memoria: **50 ordu**. Proiektuan zehar xehetasunak dokumentatzen badira azkeneko lana ez da handiegia izango.

Defentsara: **20 ordu**.

Ondorioz, proiektu osoa egiteko **358 ordu** estimatzen dira. 2.2. taulan estimazioak laburtuta ikus daitezke.

Ataza	Estimazioa
Planifikazioa	30
Teknologien azterketa	32
Aplikazioa garatu	166
Probak	30
Kudeaketa	30
Memoria	50
Defentsa	20
Guztira	358

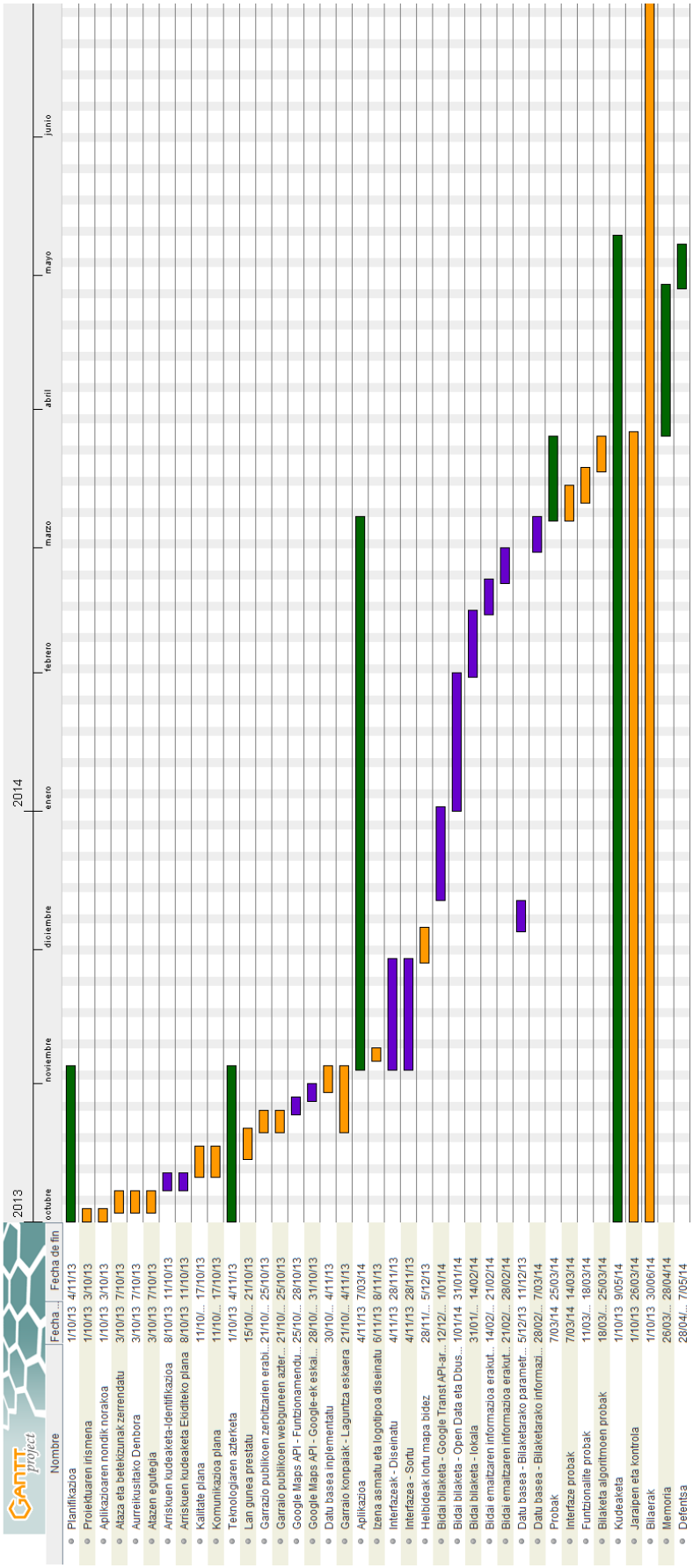
2.2. taula: Proiektuaren denbora estimazioa orokorrak

2.2.3. Atazen planifikazioa

2.3. irudiak atazen antolaketa irudikatzen du Gantt diagrama baten bitartez. Ataza bakoitzaren datak erabakitzeko eragin dezaketen hurrengo egoerak kontuan hartu dira:

- Unibertsitatea: proiektua garatzen den aldi berean klasea dagoenez, denbora gutxi edukiko diren garaiak egongo dira, adibidez azterketen garaia. Ezinezkoa da kurtso osoan edukiko den lan karga estimatzea, baina azterketen egutegia kontuan hartu daiteke jakiteko noiz izango diren ofizialak.
- Lana: kurtsoan zehar noizbehinka lanean arituko naizenez, aurreikusi behar dut ez dudala beti denbora osoa libre izango.
- Pertsonalak: badaude okupatuta izango diren data batzuk hasieratik ezagutzen direnak, hala nola aste santuko bigarren astean ia ezinezkoa izango da lan egitea.

Proiektua kurtso hasieran hasiko denez denborarekin bukatzea estimatzen da, egoera ideala proiektua maiatzaren hasieran bukatzea da. Hau lortzea oso zaila izango da, baina estimazio honekin, edozein atzerapen gertatuta ere denbora marjina edukiko da garaiz bukatzeko.



2.3. irudia: Proiektuaren Gantt diagrama

2.3. Kalitatearen kudeaketa

Gaur egun garatzen diren produktuak gero eta kalitate hobeagokoak izan behar dira, bezeroak ez dira edozerekin konformatzen, horregatik helburu garrantzitsuenetakoa kalitatezko produktu bat sortzea izan behar da.

Proiektua kalitatezkoa izateko lehendabizi maila minimoa eta egokia ezarri behar dira. Ez hori bakarrik, hauek betetzeko plan bat garatu behar da.

2.3.1. Kalitate maila planifikatu

Proiektuaren irismenean dagoeneko adierazten da zeintzuk diren aplikazioaren kalitate maila minimoak, alegia, funtzionatzeko behar dituen elementu-funtzionalitate minimoak. Proiektuan kalitate kontrola behar duten atal guztiak hurrengoak dira:

- Komunikazioa: proiektuaren interesatuen arteko komunikazioan maila egoki bat egon behar da arazorik ez egoteko.
- Produktua: aplikazioa erabilgarria izateko minimo batzuk bete behar dira.
- Memoria.
- Defentsarako gardenkiak

2.3.1.1. Kalitate maila minimoa

Atal honetan minimo onargarriak zehaztuko dira.

- Komunikazioa: tutorea informatuta mantendu. Proiektuan zerbait garrantzitsua egiten den bakoitzean tutoreari idatzi edo berarekin bildu. Ez da bilera kopuru minimo bat ezarriko.
- Produktua: irismena atalean adierazitakoa bete, hots, bilaketak egin (zuzenak eta konposatuak), garraio mota aukeratu ahal izatea eta bilaketa orduaren arabera egin ahal izatea. Ez hori bakarrik aplikazioaren kodea txukuna eduki behar da ulergarria izateko.
- Memoria: kalitatezko memoria bat egiteko gutxienez “GAP eredua” dokumentuan adierazten dena bete behar da (ez zehatz-mehatz, erreferentzia bat besterik ez da). Gainera hasi baino lehen edukiaren eskema bat sortu eta tutoreari erakutsiko zaio, berak ontzat eman arte ez da hasiko atal hau. Memoriaren atal nagusiaren orri kopurua gutxi gora behera 80 izan behar da.
- Defentsarako gardenkiak: aurrekoaren antzekoa, hasi baino lehen tutoreari oniritzia eskatuko zaio. Diapositiba kopurua ez da finkoa, azaldu nahi diren ideien arabera, pare bat bakoitzean.

2.3.1.1. Kalitate maila egokia

Atal honetan kalitate maila egokia zein den definituko da.

- Komunikazioa: tutorea informatuta mantentzeaz gain egiten dena erakutsiko zaio berak jakiteko proiektua aurreratzen ari dela eta bere iritzia jakiteko. Atal baten dudak izanez gero tutoreari galdetuko zaizkio bide zuzenetik joateko.
- Produktua: aplikazioa maila egokia edukitzeko hainbat funtzionalitate berri eduki behar ditu, hala nola, bidaia zuzenak soilik bilatzeko aukera izatea, aplikazioa Internet gabe erabili ahal izatea, oinezko distantziak zehaztu ahal izatea eta abar. Lantokia hobetzeko asmoz, unibertsitateko proiektuko gelaren sarbidea eskatuko da, hor askoz hobeto lan egingo da beste edozein lekutan baino. Kodea ere txukuna idazteaz aparte funtzio bakoitza komentatuko da, horrela ez da inoiz ahaztuko zer egiten duen.
- Memoria: kalitate egokia izateko, orri kopurua 80 eta 100 bitartean izan behar da, eranskinak aparte. Ez hori bakarrik, edukiak kalitatezkoa izan behar da. Gainera memoriaren idazkera errazteko formatu egokia duen txantiloi bat sortuko da.
- Defentsarako gardenkiak: Gardenkiek ez dute eduki gehiegirik eduki behar, askoz hobeagoa da testu gutxi jartzea eta asko azaltzea. Hortaz gardenki bakoitzean 2-3 ideia nagusi izango dira.

2.3.2. Kalitatearen kontrola

Kalitate mailak definitzea bakarrik ez da nahikoa, hau betetzen dela ziurtatu behar da. Horretarako neurri batzuk hartuko dira:

- Produktuan: APP-a garatzen den heinean, proiektuarekin zerikusirik ez duten pertsonen iritzia eskatuko da erabilgarria dela ziurtatzeko.
- Memorian: tutorearen iritzia eskatuko da honen sorkuntzan zehar. Ulergarria dela bermatzeko hirugarren pertsonei ere eskatuko zaie irakurtzeko.
- Defentsarako gardenkietan: Memoriaren kontrol bera erabiliko da.

2.4. Komunikazio plana

Komunikazioa proiektu kudeaketaren atal oso garrantzitsua da, interesatuak nortzuk diren behin definituta, proiektuan duten interes maila zehaztu behar da eta komunikazio plan bat garatu.

2.4.1. Interesatuen identifikazioa

Atal hau aldatzen joan da, proiektuaren hasieran soilik bi ziren, proiektuaren egilea eta tutorea. Aurrera egiten zuela interesatu berriak gehitu ziren, Dbus konpainia eta Open Data ataria.

2.4.1.1. Proiektuaren egilea

- Interes maila: Oso altua.
- Kontaktua: japodaca002@ikasle.ehu.es.
- Deskribapena: Proiektuaren autorea. GipuzkoaBidaian berak asmatutako ideia denez bere interes maila oso altua da, ez hori bakarrik, proiektua gainditu behar duen pertsona denez, ondo egiten interesatuena da.

2.4.1.2. Tutorea

- Interes maila: Altua.
- Kontaktua: alfredo@ehu.es.
- Deskribapena: Proiektuaren zuzendaria. Galdera teknikoak ez erantzun arren, proiektuetan esperientzia duen pertsona da, horregatik kontuan hartuko da proiektua ondo bideratzeko.

2.4.1.3. Dbus konpainia

- Interes maila: Altua.
- Kontaktua: info@dbus.es.
- Deskribapena: Dbus konpainiaren informazio iturri hornitzailea. Dbus esan arren Javier Vallejo, Zerbitzu eta Planifikazio Zuzendaria, izan da kontaktuan egon den pertsona. Hau izan zen laguntza eskaerari erantzun zion konpainia bakarra, "D Eranskina - Garraio publiko konpainien webguneen azterketa" kapituluan hobeto azaltzen den bezala.

2.4.1.3. Open Data ataria

- Interes maila: Altua.
- Kontaktua: http://opendata.euskadi.net/w79-contgen/eu/o53VisualizadorWar/o53BuzonEuskadi.jsp?por=200&lenguaje=2
- Deskribapena: Informazio iturria. Open Datak garraio publikoen informazioa gordetzen du eta publikoari eskaintzen diote. Beraiekin kontaktuan jartzean, jatorrizko informazioa okerra zela eta, aplikazioan interesa zutela adierazi zuten eta momentu horretatik aurrera kontaktua mantendu zen.

2.4.2. Informazioaren banaketa

Interesatu guztiekin komunikazio bide nagusi berdina erabiliko da, mezu elektronikoak. Open Data-ko kasuan era bakarra dago, webguneko formularioa betetzean bidaltzen den mezuaz.

Zerbait garrantzitsua izanez gero beste bideak erabiliko dira, tutorearekin zuzenean hitz egingo da bere bulegoan eta Dbus-en kasuan Javierri telefonoz deituko zaio edo bilera bat antolatuko da.

2.5. Arriskuen kudeaketa plana

Atal honetan proiektua kaltetu dezaketen egoerak identifikatu eta prebentzio planak sortuko dira.

Arazoak edozein lekutik etorri daitezke, gainera askotan ezin dira ekidin, horregatik ahal diren egoera txar posible guztiak suposatu behar dira.

2.5.1. Konputagailua hondatzea

Azalpena: ordenagailuak ez dira sendoegiak, are gehiago eramangarriak, horregatik arduraz erabili behar dira.

Gertatzeko probabilitatea:txikia.

Eragina: oso handia. Ordenagailuan dagoen informazio zati bat edo dena galdu daiteke. Kasu txarrenean ordenagailua erabilgaitz geratu daiteke informazio guztia galduz eta lantoki berria bilatu behar izanez.

Jatorri posiblea: hainbeste lan edukita gerta liteke eramangarria bat-batean hondatzea, piezaren bat galdatzen delako edo kolperen bat jasotzen duelako (leku batetik bestera mugitzeagatik).

Prebentzio plana: ezer gertatu ez dadin ordenagailua arduraz erabili eta eramango da, adibidez gehiegi ez berotzeko haizagailuak dituen oinarri bat erabiliko da aireztatuta mantentzeko. Garraiatzean, kolperik ez jasotzeko, zorro baten barruan eramango da. Informazioa galtzen bada “Informazio galera” ataleko prebentzio plana jarraituko da. Ordenagailua erabiltezin geratzen bada ez da arazorik egongo, fakultateko teknikariei eskatuko zaie mahai gaineko bat jartzeko proiektu gelan eta hor proiektua jarraituko da.

2.5.2. Informazio iturria aldaketak egitea

Azalpena: Dbus edo Open Data-k eskaintzen duten informazioaren egitura aldatzen da.

Gertatzeko probabilitatea: txikia.

Eragina: oso handia. Informazioa eskaintzeko era aldatzen bada proiektuan eragina izango du, hainbat atal aldatu beharko liratekeelako. Iturriak bat-batean funtzionatzeari utziko balio edo informazio hori eskaintzeari uzten badiote, orduan, proiektuan izango lukeen eragina oso handia izango litzateke.

Jatorri posiblea: kanpoko arazoa denez ezinezko da jakitea zerk eragin dezakeen.

Prebentzio plana: eskaintzen duten informazioaren egitura edo era aldatzen badute proiektuaren kodea aldatu beharko litzateke, hori ezin da ekidin, ez hori bakarrik, datu iturri desberdinak bilatu beharko lirateke. Hortaz, aplikazioaren kodeak banaketa logiko bat izango du, zati bat aldatu behar bada besteak ez eragiteko. Era honetan, datu iturri berria aurkitzean, aldatu beharrekoa minimoa izango litzateke.

2.5.3. Informazio galera

Azalpena: proiektuaren fitxategi edo dokumentu bat galtzen den kasua.

Gertatzeko probabilitatea: altua.

Eragina: altua. Desagertutako fitxategia garrantzitsua bada, proiektu osoari eragin diezaioke.

Jatorri posiblea: nahigabe zerbait ezabatzen bada edo konputagailua kaltetzen bada egoera honetara ailegatu daiteke.

Prebentzio plana: proiektu osoaren segurtasun kopiak izatea. Hauek periodikoki egingo dira eta zerbait garrantzitsua egitean. Hiru motakoak izango dira: aplikazioaren kodea, proiektuaren dokumentazioa eta memoria (defentsaren gardenkiak barne). Informazioa ZIP fitxategi batean biltegiratuko da, edozein lekutan irakurri daitekeelako. Artxiboaren Izenean gordetzen dena adieraziko da (workspace, PFG edo memoria) eta segidan kopia egin deneko data (urtea_hilabetea_eguna). Fitxategi hauek hainbat lekutan biltegiratuko dira, baten bat hondatzen bada ondorioak nuluak izateko: ordenagailuan, USB memoria orokorrean, segurtasun kopiak soilik dituen beste USB memoria batean eta Google Drive-en.

2.5.4. Aurreikuspenak ez betetzea

Azalpena: lana korapilatzen bada eta estimatutako datan bukatzea ezinezkoa den kasua.

Gertatzeko probabilitatea: handia. Sistema berri baterako programatzeak ezusteko zailtasunak ekar ditzake, hortaz, seguruenik lan gehiago egin beharko da. Gehiagozko ordu kopurua sartu arren, estimatutako datak betetzea izango da helburu nagusia.

Eragina: txikia. Proiektua goiz hasi da atzerapenek arazorik ez emateko.

Jatorri posiblea: programatzerakoan zailtasunak izatea eta bizitzako ustekabekoak, adibidez gaixorik jartzea.

Prebentzio plana: prebentzio plana dagoeneko martxan dago, proiektua ahalik eta goizen hastea.

2.5.5. Memoria eta defentsaren daten aldaketa

Azalpena: memoria edo defentsaren entrega daten aldaketak.

Gertatzeko probabilitatea: txikia.

Eragina: txikia.

Jatorri posiblea: ezinezkoa zait hori jakitea, unibertsitatea baita datak jartzen dituena.

Prebentzio plana: aldaketa atzerapena bada orduan ez dago arazorik. Data aurreratzen bada orduan egun gutxiago izango dira bukatzeko, hala ere, denbora tarte nahikoarekin bukatzea estimatzen da, arazoa hauek ere ekiditeko.

3

Tresnak eta teknologiak

Atal honetan proiektua garatzeko erabilitako tresnak eta teknologiak zeintzuk diren azalduko da. Kasu berezi batzuetan ere zergatik software bat eta ez bestea erabili den argituko da.

Tresnak eta teknologien egitura
1. Erabilitako hardwarea 2. Garapen-inguruak 3. Web-tresnak eta teknologiak 4. Emuladoreak 5. Kasu probak 6. Testu-editoreak 7. Irudi-editoreak

3.1. Erabilitako hardwarea

Proiektua garatzeko erabili diren hardwareak ordenagailu eramangarria eta hainbat sakelako telefono izan dira. Horrekin batera hainbat USB ere erabili dira datuak gordetzeko. Proiektuaren izaeragatik ez da beharrezkoa izan beste ekiporik, lan guztia leku berean kudeatu da.

3.2. Garapen-inguruak

Aplikazioa sortzeko Eclipse garapen-ingurua erabili da, beste hainbat aukera zeuden, adibidez Android Studio, baina azkenean ezaguna zena erabiltzea aukeratu zen. Eclipse ez da besteak baino askoz hobeagoa, Interneten esaten duten bezala, hoberena da norbera erosoago sentitzen den softwarea erabiltzea.

3.2.1. Eclipse

Eclipse kode irekiko software plataforma da, zehazki IDE (integrated development environment edo garapen ingurune integratu) bat da. Eclipse ez da bakarrik erabiltzen Java aplikazioak garatzeko, beste hainbat lengoaietako programak ere (Ada, C++, Haskell, PHP eta beste hainbeste) idatz daitezke.

Bere abantaila nagusia da *Plugin*-ak gehitu zaizkiola funtzionalitateak handitzeko, adibidez, Android sistemarentzako programatzeko. Hasiera batean, Eclipse softwarea ez da gai Android proiektuak sortzeko, ADT (Android Development Tools) *Plugin*-a instalatu behar da.

Android garatzaileen webgune ofizialean [1] garapen-ingurune prestatuak zuzenean deskargatzeko aukera dago, honi esker Eclipse-a oso azkar prest egon da erabiltzeko. Proiektu osoan ez da arazo gehiagorik egon eta ez da beharrezkoa izan programa aldatzea, hortaz aukeratutakoa erabaki ona izan dela esan daiteke.

3.2.2. Android SDK

Android SDK ez da garapen inguru bat, Eclipse-ko tresna bat da. API liburutegiak eta beharrezko garapen tresnak hornitzen ditu Android-erako APPak sortzeko, probatzeko eta arazteko. Oso garrantzitsua da, bera gabe ezinezkoa izango litzateke aplikazioak garatzea.

[1] http://developer.android.com/sdk/index.html

Aplikazioak Android bertsio batean edo bestean programatzeko beharrezko liburutegiak SDK-tik deskargatu behar dira, ez hori bakarrik, gehigarri guztiak ere hortik deskargatzen dira. Honek arazo bat sortzen du, proiektua betikoa ez den ordenagailuan garatu nahi bada, ekipo berriaren SDK-ren konfigurazioa berdina izan beharko da. Arazo hau arintzeko deskargatu diren elementu guztiak dokumentatu dira, era honetan konputagailu berria erabili beharko balitz ez litzateke denbora xahutuko lan ingurunea prestatzen. Zorionez ez da inoiz beharrezkoa izan ekipoz aldatzea.

3.3. Web-tresnak eta teknologiak

Proiektuaren garapenean ia denbora guztian Internetera konektatuta egotea beharrezkoa izan da, informazioa bilatzeko, datuak eskuratzeko edo web zerbitzuak erabiltzeko. Honetarako hainbat tresna erabili dira, Google Chrome nabigatzailea, Firefox-en debbuger-a eta FileZilla. Google-eko hainbat API erabili dira informazioa eskuratzeko, JSON eta XML formatuetan.

3.3.1. Google Chrome nabigatzailea

Google konpainiaren nabigatzailea, besteekin konparatuz azkarrena denez Interneteko kontsultak eta deskargak egiteko erabili da.

3.3.2. Firefox nabigatzailea: debugger-a

Proiektuaren hasieran, garraio konpainien webguneak aztertu ziren informazioa hortik lor zitekeen ikusteko. Hori egin ahal izateko Firefox nabigatzailearen Firebug Plugin-a erabili da. Tresna honek debugger-a izateaz gain beste hainbat gauzetarako balio du, hauetako bat DOM egiturak aztertzea da.

Gehigarri horri esker posiblea izan zen jakitea zer lortu zitekeen webgune bakoitzetik eta datu iturriari buruz erabakiak hartzea. Memoriaren “D Eranskina: Garraio publiko konpainien webguneen azterketa” kapituluan azaltzen da konpainiei egindako azterketa.

3.3.3. FileZilla

FileZilla kode irekiko eta software libreko FTP bezero multiplataforma da, GNU Lizentzia Publiko Orokorrean lizentziatuta dago. FTP, SFTP eta FTP-SSL (FTPS) gaineko protokoloak onartzen ditu.

Programa hau erabili da Open Data eta Dbus zerbitzarietara konektatzeko eta informazioa deskargatzeko. GipuzkoaBidaian garraio publikoen informazioa FTP-z lortzen duenez hasieran ziurtatu behar zen zerbitzariek protokolo hori onartzen zutela eta lan egiteko arazorik ez zegoela.

3.3.4. JSON

JSON, JavaScript Object Notation-en akronimoa, datu elkartrukerako formatu arina da. Gizakiarentzat ulergarria den testua erabiltzen du datu objektuak transmititzeko, hauek atributu-balio bikoteak dira. Zerbitzari eta bezeroen artean datuak bidaltzeko gehienbat erabiltzen da.

Google-eko API-ek informazioa bi eratan eskaintzen dute, XML-ez eta JSON-ez. Bigarrena erabiltzea erabaki da azkarragoa delako.

3.3.5. XML

XML, ingelesezko *e**X**tensible **M**arkup **L**anguage*, irakurgarriak diren datuak gordetzeko lengoaia da. SGML lengoaiatik dator eta HTML-ren egitura bera erabiltzen du. World Wide Weberako estandarrak, World Wide Web Consortium (W3C) garatua.

Formatu hau erabiltzen da aplikazioaren parametroak gordetzeko, cache memorian gordetzen da eta bilaketarako parametroak gordetzen ditu. Android-ek ere bere XML propioak sortzen ditu aplikazioen datuak aldi baterako gordetzeko.

3.4. Emuladoreak

Android-erako programak ordenagailuan probatzeko software berezi bat behar da, Eclipse ez da gai bera bakarrik egiteko, ADT Plugin-a Eclipse Device Manager tresnatik gehitu behar zaio. Gehigarri hau smartphonen emuladoreak sortzeko erabiltzen da, era honetan sortutako programak hor instalatu eta proba daitezke.

Programak probatzeko beste hainbat metodo desberdin daude, bata aplikazioaren APK, instalatzailea, sortzea eta norberaren mugikorrean instalatzea da. Metodo hau erabili da APP-aren bateragarritasuna frogatzeko. Bestea Android emuladore independentea erabiltzea da, Genymotion.

3.4.1. Android Virtual Device

Android Tresna Birtuala, AVD, Eclipsen lehenetsita dagoen emuladorea (ADT gehigarria gehitzean) da. Bere ezaugarri interesgarriena da oso moldagarria dela, merkatuko ia edozein mugikor emulatu dezake eta edozein pantaila tamaina sortu dezake. Hala eta guztiz ere, arazo bat du, motelegia da, edozein aplikazioa exekutatzea asko kostatzen zaio.

AVD ez erabiltzearen arrazoia ez da abiadura bakarrik izan, GipuzkoaBidaian-ek mapak erabiltzen ditu, eta horretarako Google-eko zerbitzuak instalatuta egon behar dira smartphonean. Arazoa da emuladoreak ez dituela, eskuz instalatu behar zaizkiola.

Interneten begiratu ondoren konturatu nintzen ez zela gomendagarria mapak AVD-an erabiltzea, hortaz alternatiba bat bilatu nuen, Genymotion.

3.4.2. Genymotion

Genymotion Android-eko emuladore bat da, Oracle VM VirtualBox softwarearen makina birtual bezala funtzionatzen du. Bere ezaugarri garrantzitsuena azkartasuna da, emuladore hau martxan askoz azkarrago jartzen da, eta zer esanik ez aplikazioez. AVD-rekin konparatuz, aurrezten den denbora hainbeste da, non hau erabiltzea aukera paregabea den.

Hala ere, ez da gomendatzen software hau erabiltzea proiektu txiki eta bakarra egiteko. Genymotion instalatzeko, lehenik, Oracle VM VirtualBox ere instalatu behar da, eta ez da prozesu azkarra. Ez da zaila softwareak instalatzea, arazoa da prozesu motela dela. Ez hori bakarrik, erabiltzen diren emuladoreak ere deskargatu behar dira, AVD-n ez bezala non emuladoreak momentuan sortzen dira, ez deskargatzen, Genymotion-ekin ordea, emuladore bakoitza deskargatu behar da.

Dena dela, proiektu honetan software hori erabiltzea erabaki zen aipatu ez den arrazoi batengatik, hasieran, Genymotion-en emuladoreak Google-ren zerbitzuak instalatuta zituzten, mapak zituzten aplikazioak onartuz.

2013ko Azaroan Genymotion-ek argitaratu zuen[2] bertsio berria kaleratuko zutela hurrengo hilabetean hainbat eguneraketarekin. Zoritxarrez ez ziren berri guztiak onak, “Google apps” ezaugarria, Google-ko zerbitzuak, bere emuladoreetatik kenduko zituztela adierazi zuten.

Honek arazo larria suposatu zuen proiektuarentzat, azken batean Google-ek eskaintzen dituen zerbitzuak beharrezkoa dira.

Zorionez dena ez zen hain txarra, eguneraketa emuladore bertsio zaharretan ez zuen eraginik izan. Hots, “Google apps” ezaugarriak zituzten makina birtualak oraindik funtzionatzen zuten, baina hauetako bat ezabatuz gero ezingo zen berreskuratu, Internetetik kendu baitzituzten bertsio zahar guztiak.

[2] https://plus.google.com/+GenymotionEmulator/posts/jNF8Kwu5p1c

3.5. Kasu probetarako

Aplikazioaren kodeak ondo funtzionatze duela frogatzeko komenigarria da tresna bereziren bat erabiltzea, kasu honetan, JUnit framework-a erabili da.

3.5.1. JUnit

JUnit probak egiteko framework bat da, aplikazioaren zati zehatza exekutatzeko erabiltzen da. Bere erabilpena oso sinplea da, klase baten objektua sortzen da eta ondoren probatu nahi den funtzioari deitzen zaio. Egindako probak emaitza bat itzuliko du, hau balio finko batekin konparatzen da jakiteko funtzioa ondo funtzionatzen al duen. Honekin lortu nahi dena da etorkizunean, edozein aldaketa eginda ere, emaitza ez dela aldatuko.

JUnit Java-ko programekin probak egiteko pentsatua egon arren, Android-eko aplikazioak ere probatu ditzake arazorik gabe, emuladoreak erabiltzeko kapaz da eta.

Dokumentatu diren proba guztiak bidaia bilaketenak dira, hainbat emaitza desberdineko probak sortu dira eta hauek erabili dira kodea aldatu den bakoitzean.

3.6. Testu-editoreak

Proiektuaren garapenean hainbat gauza idatzi behar izan dira, aurkitutako arazoak, egindako lana, memoria, eta abar. Hau guztia egiteko Word programa erabili da.

3.6.1. Microsoft Word 2010

Word testu editore bat da, Microsoft konpainiarena. Erabiltzeko bere erraztasunagatik eta ia edozein ordenagailutan aurkitu daitekeelako erabili da behar den guztia idazteko.

3.7. Irudi-editoreak

GipuzkoaBidaian-ek hainbat irudi erabiltzen ditu, eta hauek ezaugarri zehatz batzuk eduki behar dituzte, arazoa da ez dela beti posible irudi guztiak tamaina zuzenekoak edo kolore egokikoak aurkitzea, horregatik hauek editatu behar dira.

3.7.1. GIMP

GIMP, *GNU Imagen Manipulation Program* edo irudiak manipulatzeko GNUren programa, irudien ediziorako programa da. GNU General Public License lizentziapean argitaratzen da.

Software honekin aplikazioaren logoa sortu da, atzealdeko irudia editatu da eta hainbat ikonoen tamaina aldatu da. Argitu behar da erabilitako irudi guztiak lizentzia librekoak direla eta aldatzeko eskubidea dagoela.

4

Garapena

Kapitulu honetan aplikazioaren xehetasun guztiak azalduko dira. Eskakizun bilketarekin hasiko da, ondoren jarraitutako diseinua azalduko da, inplementazioan aplikazioaren funtzionamendua esplikatuko da eta azkenik, funtzionamendua ziurtatzeko egin diren probak azalduko dira.

Proiektuaren Helburu-dokumentuaren egitura
1. Eskakizunen bilaketa 2. Diseinua 3. Inplementazioa 4. Proba kasuak

4.1. Eskakizunen bilketa

4.1.1. Aktoreak

GipuzkoaBidaian aplikazioan aktore mota bakarra dago, erabiltzailea. Erregistratzeko aukerarik ez dagoenez bezero mota bakarra egongo da. Aplikazioa independentea denez, hau da, behin informazioa deskargatuta bera bakarrik datuak modelatzeaz arduratzen denez, ez du behar hirugarren laguntzarik eta ez dago beste aktore motarik.

4.1. irudia: GipuzkoaBidaian APP-aren aktoreak

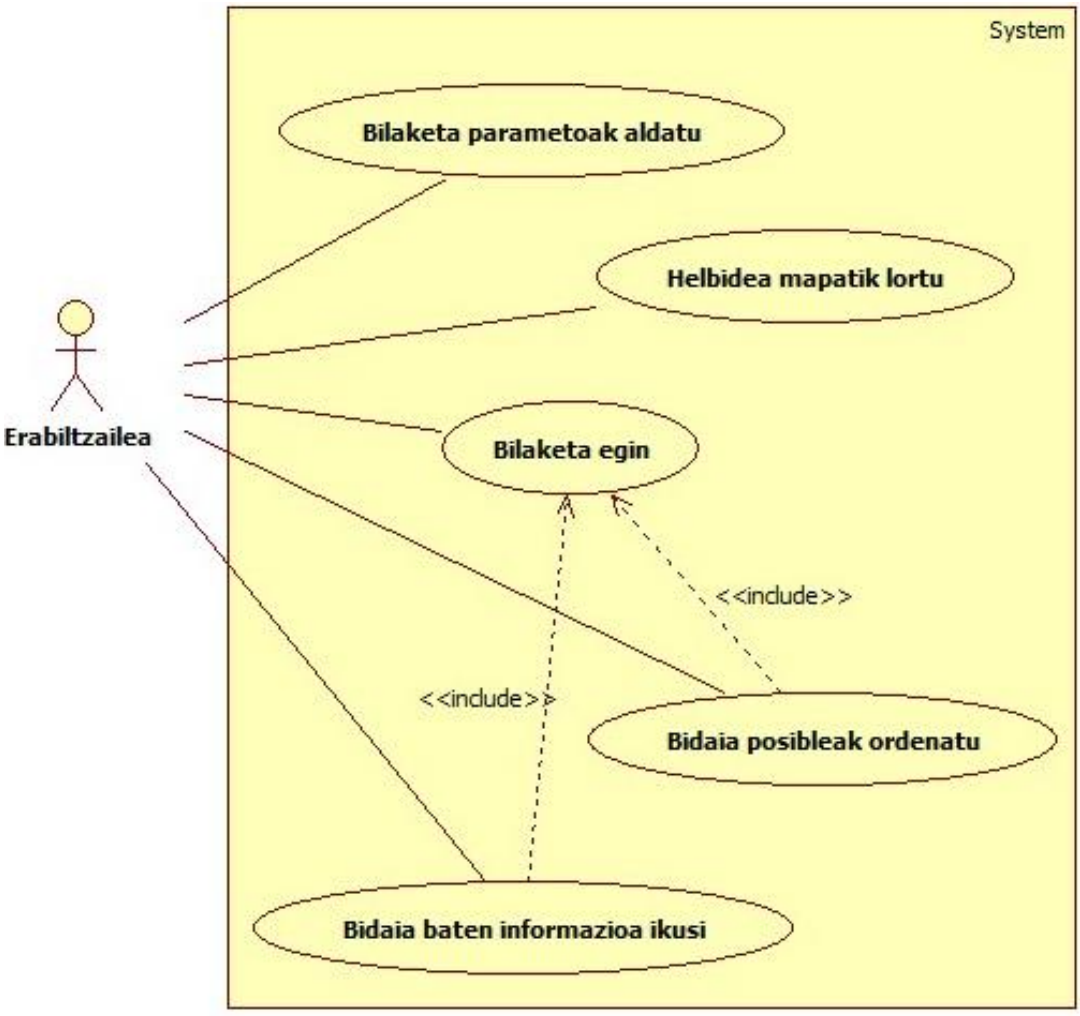
Aplikazioa era optimoan eginda egongo balitz, zerbitzari bitartekari bat erabiliz ("C Eranskina: GipuzkoaBidaian-en funtzionamendua zerbitzari baten bidez" kapituluan azaltzen den bezala), orduan aktore berri bat egongo litzateke, "Zerbitzari sistema" edo antzeko zerbait.

4.1.2. Erabilpen kasuak

Aplikazio honek ez du erabilpen kasu asko, erabiltzaileak egin ditzakeen gauza kopurua oso mugatuta dago. APP-aren helburua bakarra da, bi puntuen arteko garraio publikoko bidaiak lortzea. Ondorioz erabiltzaileak egin dezake:

- Aplikazioaren parametroak aldatu.
- Bilaketa egiteko jatorria eta helmuga helbideak mapa baten bidez lortu.
- Bidaiak bilatu.
- Lortutako emaitzak ordenatu.
- Lortutako emaitza baten informazioa ikusi.

Lan handiena, garrantzitsuena, sistemak egiten du bidaiak bilatzerakoan. Hor erabiltzaileak ez du ezer egiten, soilik bilaketaren parametroak ezarri eta martxan jarri.



4.2. irudia: Erabilpen kasuen diagrama

4.1.2.1. Erabilpen kasuen eredua

Aktore bakarra dagoela kontuan izanda eta erabilpen kasu gehiegi ez daudela ikusita, 4.2. irudiko diagrama ez da handiegia ateratzen.

Erabiltzaileak bidaiak ordenatu nahi baditu edo baten informazioa ikusi nahi badu, lehenik bilaketa egin beharko du.

4.1.2.2. Erabilpen kasuen gertaera fluxuak

Erabilpen kasuak zerrendatzeaz gain zer egiten dute ere azaldu behar da, ondoren azaltzen da bakoitza zehaztasunez:

BILAKETA PARAMETROAK ALDATU

Aktoreak: erabiltzailea.

Laburpena: doikuntza interfazean bilaketan erabiliko diren parametroak finka daitezke:

- Oinezko distantzia maximoa: jatorri eta helmuga kokapenetatik zein distantziarako geltokiak bilatu nahi diren. Zenbaki hau erlatiboa da, ez baita berdina bi puntuen arteko distantzia eta puntu batetik bestera joateko oinez egin behar den distantzia.
- Bilaketa modua: zehatza, azkarra eta lokala. Lehenengoa eta hirugarrena Open Data eta Dbus-ek eskaintzen duten informazioarekin burutzen dira. Bilaketa mota hauekin ibilbide konbinazio guztiak bilatuko dira. Bi hauen desberdintasuna da bat informazioa eguneratua lortzen saiatzen dela eta besteak mugikorrean gordeta dagoenarekin lan egiten duela. Beste bilaketa modua Google Directions API-a erabileran oinarritzen da, honek soilik bidaia bat bilatzen du, baina lortutakoa aukera hoberena da.
- Zuzeneko bidaia: nahi izanez gero, erabiltzaileak zuzeneko bidaiak soilik bila ditzake. Aukera hau bilaketa modua zehatza edo lokala denean bakarrik beteko da.

Aurrebaldintzak: ez du.

Postbaldintzak: ez du.

Gertaera fluxua normala:

1. Erabiltzaileak doikuntzetara joateko botoia sakatuko du.
2. Sistemak uneko parametroak idazten ditu.
3. Erabiltzaileak distantzia maximo berria idazten du, bilaketa modua aukeratuko du eta bidaia zuzena nahi duen aukeratuko du.
4. Sistemak datuak hartzen ditu eta informazio hau gordetzen duen fitxategia eguneratzen du.
5. Sistemak aurreko interfazeari itzultzen da datu berriekin.

Gertaera fluxu alternatiboa:

3- Emandako oinezko distantzia limiteetatik at baldin badago, erabiltzaileari datua gaizki dagoela abisatuko dio aplikazioak eta ez du gehiago ezer egingo.

HELBIDEA LORTU

Aktoreak: erabiltzailea.

Laburpena: erabiltzaileak jatorria eta helmuga helbideak lortu ahal izango ditu mapatik zuzenean. Biak ezin dira lortu aldi berean, bakoitza banaka lortu behar da. Mapan Gipuzkoako eremua markatuta egongo da errazagoa izateko honen identifikazioa.

Aurrebaldintzak:

1. Mugikorrak gutxienez OpenGL 2 liburutegia izatea.

2. Google Services instalatuta edukitzea.

Postbaldintzak: ez du.

Gertaera fluxua normala:

1. Erabiltzaileak kokapena lortzeko mapara joateko botoia sakatzen du.
2. Sistemak mapa erakusten dio eta Gipuzkoa erakutsiko du mapan.
3. Erabiltzaileak maparen puntu bat ukitzen du.
4. Sistemak ikono bat jarriko du erabiltzaileak ukitu duen koordenadetan.
5. Erabiltzaileak ikonoa ukitzen du.
6. Sistemak erakusten dio koordenada hori eta dagokion helbidea.
7. Erabiltzaileak kokapena lortzeko botoia sakatzen du.
8. Sistemak bere helbide aldagaiak eguneratzen ditu eta aurreko interfazera bueltatzen dio datu berriekin.

Gertaera fluxu alternatiboa:

2- Mapa kargatzen ez bada erabiltzaileari abisatzen zaio eta ez da ezer gehiago egiten.

2- Interfazea kargatzean GPS-a aktibatuta dago. Mapa erabiltzailearen posizioan kokatuko da.

3- Erabiltzaileak ukitu duen puntuaren helbidea ezin da lortu. Sistema saiatuko da koordenadaren helbidea eskuz lortzen. Hau ere ezin bada, orduan erabiltzaileari abisatuko dio eta ez du ezer egingo

BILAKETA EGIN

Aktoreak: erabiltzailea.

Laburpena: erabiltzaileak bi helbideen arteko garraio publiko bidezko ibilbideak lortuko ditu, horretarako bilaketa parametroak zehaztuko ditu:

- Jatorri helbidea: hau bi eratan finkatu daiteke, mapa bidez edo idatziz. Lehenengo eran eginez gero, HELBIDEA LORTU erabilpen kasura joango da eta behin informazioa lortuta, hau testu eremuan agertuko da.
- Helmuga helbidea: hau bi eratan finka daiteke, mapa bidez edo idatziz. Lehenengo eran eginez gero, HELBIDEA LORTU erabilpen kasura joango da eta behin informazioa lortuta, hau testu eremuan agertuko da.
- Orduaren arabera bilaketa egin: bilatutako ibilbideak orduaren arabera izateko. Kontuan izan behar da bidaia zuzeneko kasuetan bilatze denbora handitzen dela, bidaia konposatuetan ordea murrizten da.

- Ordua finkatzeko bere testu eremua ukitu behar da, orduan erloju motako interfaze txiki bat agertuko da. Behin nahi diren ordua eta minutuak ikusita OK botoia sakatu eta aurreko testu eremua eguneratuko da.
- Garraio mota aukeratu: erabiltzaileak autobus eta/edo tren artean aukeratu ahal izango du. Honetarako botoiak uki ditzake edo bere laukia aktibatu/desaktibatu.

Aurrebaldintzak:

1. Bilaketa zehatza egin nahi izanez gero, SD txartela izatea smartphonean.

Postbaldintzak: ez du.

Gertaera fluxu normala:

1. Erabiltzaileak jatorri helbidea, helmuga helbidea, ordua eta garraio mota(k) finkatzen ditu.
2. Sistemak jatorri helbidea hartu eta benetako helbide posible lista erakusten du.
3. Erabiltzaileak jatorri bat aukeratzen du.
4. Sistemak idatzitako helmuga helbidetik benetako helbide posible lista erakusten du.
5. Erabiltzaileak helmuga bat aukeratzen du.
6. Sistemak bilaketa parametroaren arabera informazio eskaera egiten dio zerbitzari bati, besteari edo inori. Beharrezkoa izanez gero garraioen informazio fitxategiak irakurtzen ditu bidaia posible guztiak izateko. Azkenik informazioa prestatzen du erakusteko eta erabiltzaileari itzultzen dio.
7. Erabiltzailea automatikoki hurrengo interfazera bidaltzen da.

Gertaera fluxu alternatiboa:

1- Parametroren bat bete ez bada abisatu eta ez egin ezer.

2- Jatorri helbidea mapa bidez definitu bada, orduan ez zaio eskatuko erabiltzaileari helbideak zehazteko.

2- Erakusten den zerrenda elementu bakarrekoa bada orduan ez zaio erabiltzaileari ezer erakutsiko, zuzenean hori hartuko da.

2- Idatzitako jatorria okerra bada, hau da, izen hori duen helbiderik ez bada existitzen, erabiltzaileari abisatuko zaio eta ez da ezer egiten.

3- Sistemak aukeran ematen dituen helbide guztien artean erabiltzaileak ez badu gustukorik aurkitzen, ezeztatzeko botoia sakatuko du eta sistemak ez du jatorria gordeko.

4- Helmuga helbidea mapa bidez definitu bada orduan ez zaio eskatuko erabiltzaileari helbideak zehazteko.

4- Erakusten den zerrenda elementu bakarrekoa bada orduan ez zaio erabiltzaileari ezer erakutsiko, zuzenean hori hartuko da.

4- Idatzitako helmuga okerra bada, hau da, izen hori duen helbiderik ez bada existitzen, erabiltzaileari abisatuko zaio eta ez da ezer egiten.

5- Sistemak aukeran ematen dituen helbide guztien artean erabiltzaileak ez badu gustukorik aurkitzen, ezeztatzeko botoia sakatuko du eta sistemak ez du ezer gehiago egingo.

6- Bilaketa parametroren bat falta bada erabiltzaileari abisatuko zaio zein den eta ez da ezer egingo.

6- Google-eko zerbitzuak erabiltzekotan, hauek ez badute funtzionatzen abisatu erabiltzaileari eta ez egin ezer.

6- Open Data eta Dbus zerbitzariak erabiltzekotan, hauek ez badute funtzionatzen, orduan, bilaketa datu lokalekin egingo da.

6- Garraio konpainia baten informazioa ez badago mugikorrean gordeta, ez da hori kontuan hartuko bilaketarako.

6- Adierazitako parametroekin ez bada bidaiarik aurkitzen abisatu erabiltzailea eta ez egin ezer.

BIDAI POSIBLEAK ORDENATU

Aktoreak: erabiltzailea.

Laburpena: behin bilaketa eginda sistemak erabiltzaileari lortutako emaitzak erakutsiko dizkio. Hauek lista batean agertuko dira hainbat datu erakutsiz:

- Distantzia estimazioa: jatorri puntutik helmugara ailegatzeko ibili behar den distantzia osoa. Zenbaki hau oinezko eta garraiokoen arteko batura izango da.
- Estimatutako denbora: bidaia osoa egiteko beharrezkoa izango den denbora estimazioa. Datu hau estimazio bat da, pertsona guztiak ez direlako abiadura berdinean ibiltzen eta garraioen ibilbide denbora datu teorikoa delako.
- Konpainia(k): jatorritik helmugara joateko erabiliko diren garraio publiko konpainiak.
- Linea(k): emaitza bakoitza linea bat edo hainbatez osatuko da. Zuzeneko bidaietan soilik bat erabiliko da, baina konposatuetan gehiago izango dira. Ez dira egongo bi emaitza linea bera izango dutenak.
- Garraio kopurua: bidaia osoa egiteko zenbat garraiotan igo behar den adierazten du.

Erabiltzaileak ordenatzeko botoia ukituz gero aukera hauek izango ditu: denboraren arabera, distantziaren arabera eta garraio kopuruaren arabera. Hauetako bat aukeratu eta adosteko botoia sakatu ondoren lista ordenatuko da.

Aurrebaldintzak:

1. BILAKETA EGIN erabilpen kasua exekutatu izana.

Postbaldintzak: ez du.

Gertaera fluxua normala:

1. Erabiltzaileak ordenatzeko botoia sakatu.
2. Sistemak elkarrizketa kutxa irekiko du ordenatzeko hiru erarekin.
3. Erabiltzaileak bat aukeratu eta adosteko botoia sakatzen du.
4. Sistemak emaitza lista ordenatzen du eta erakusten du. Ez hori bakarrik, interfazeko goiko aldean ordenazio mota idatziko da.

Gertaera fluxu alternatiboa:

2- Dagoeneko emaitzak ordenatuak izan badira, kutxa irekitzean erabili den ordenazio mota aukeratua egongo da.

3- Erabiltzaileak ezeztatzeko botoia sakatzen badu ez da ezer gertatuko.

4- Sistemak detektatzen badu emaitza bakarra dagoela ez du ezer egingo.

BIDAIA BATEN INFORMAZIOA IKUSI

Aktoreak: erabiltzailea.

Laburpena: emaitza bat aukeratzean honen informazio guztia ikusi ahal izango da:

- Irteera eta iriste helbideak agertuko dira interfazearen goialdean eta behealdean.
- Bidaia osatzen duen zati bakoitza agertuko da ordenatua: Oinez, garraioz, oinez (azkeneko “garraioz-oinez” hainbat aldiz errepika daiteke, bidaia konposatua denean adibidez). Gerta liteke ontzi aldaketa bat geltoki berean izatea eta bi garraioko zatien artean oinezko zatirik ez egotea.
- Zati bakoitzaren hasieran estimatutako distantzia eta denbora agertuko dira.
- Oinezko kasuan: puntu batetik bestera ailegatzeko jarraibideak agertuko dira.
- Garraioaren kasuan:
    - Garraio mota.
    - Egingo duen ibilbidearen izena.
    - Irteera geltokia.
    - Geldialdi kopurua.
    - Iristera geltokia.
    - Garraioaren konpainia.
    - Bilaketa denboraren arabera izanez gero irteera eta iristera orduak ere agertuko dira.

Aurrebaldintzak:

1. BILAKETA EGIN erabilpen kasua exekutatu izana.

Postbaldintzak: ez du.

Gertaera fluxua normala:

1. Erabiltzaileak bidaia zerrendako emaitzen artean bat aukeratzen du.
2. Sistemak informazioa egokitzen du eta erakusten du.
3. Erabiltzaileak gora eta behera mugitzen du pantaila informazio osoa ikusteko.
4. Edukia mugitzen den heinean sistemak informazio berria erakusten du oraindik hau erakutsi ez bada.
5. Erabiltzaileak ibilbidea mapan marraztuta ikusteko erlaitza ukitzen du.
6. Sistemak ibilbide osoa mapan marraztuko du.

Gertaera fluxu alternatiboa:

2- Tableta bat erabiltzekotan informazioaz gain ibilbide osoa ere mapan marraztuta agertuko da.

5- Tableta bat erabiltzekotan hau ezingo da egin.

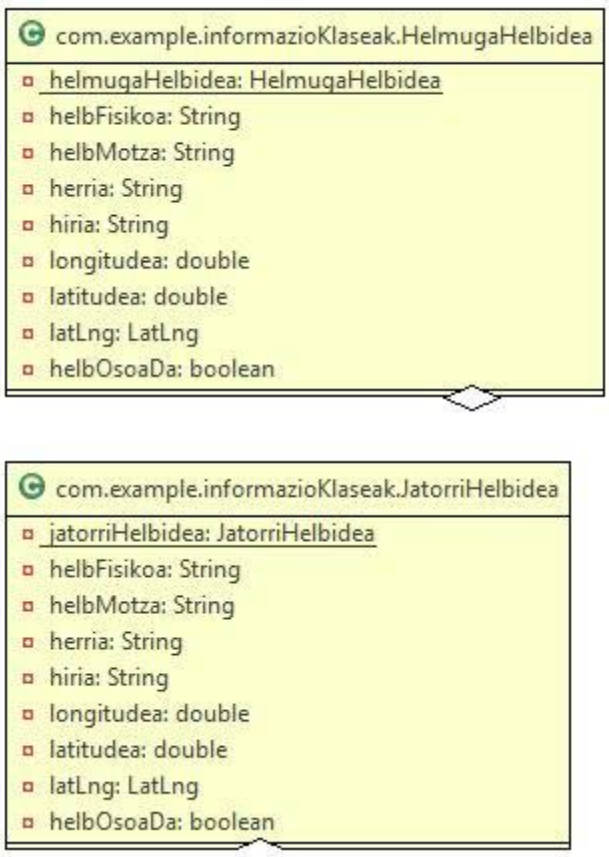
4.1.3. Domeinuaren eredua

Honek erakusten ditu sisteman erabiltzen diren objektu garrantzitsuenak eta beraien arteko erlazioak. Aplikazio honek bidaia bilaketei buruz soilik datza, hortaz ez du objektu asko, klase gehienak eragiketak egiteko baitira.

4.1.3.1. Helbide fisikoak irudikatzen duten objektuak

4.3. irudian ikus daitekeen moduan, GipuzkoaBidaian-en helbideak irudikatzen dituzten bi klase daude: *JatorriHelbidea* eta *HelmugaHelbidea*. Hauek Singleton patroia jarraitzen dute, ondorioz objektuak izango balitz erabiltzen dira. Biak klase bakar batean bildu daitezke, baina hainbat pantailatan, klaseetan, erabili behar direnez informazioa transmititzeko erosoagoa da separatuta edukitzea. Biak egitura bera dute, baina bakoitza independenteki erabiltzen da, bakoitzak bere informazioa gordetzen du.

- **JatorriHelbidea**: bere izenak adierazten duen bezala bilaketa egiteko jatorri helbidea adierazten du. Klase honek hainbat datu dauzka, baina garrantzitsuenak *latLng* eta *helbFisikoa* dira. Lehenengoa geltokiak bilatzeko erabiltzen da eta bigarrena erabiltzaileari erakusten zaio.
- **HelmugaHelbidea**: *JatorriHelbidea* bezala baina helmugako informazioarekin.

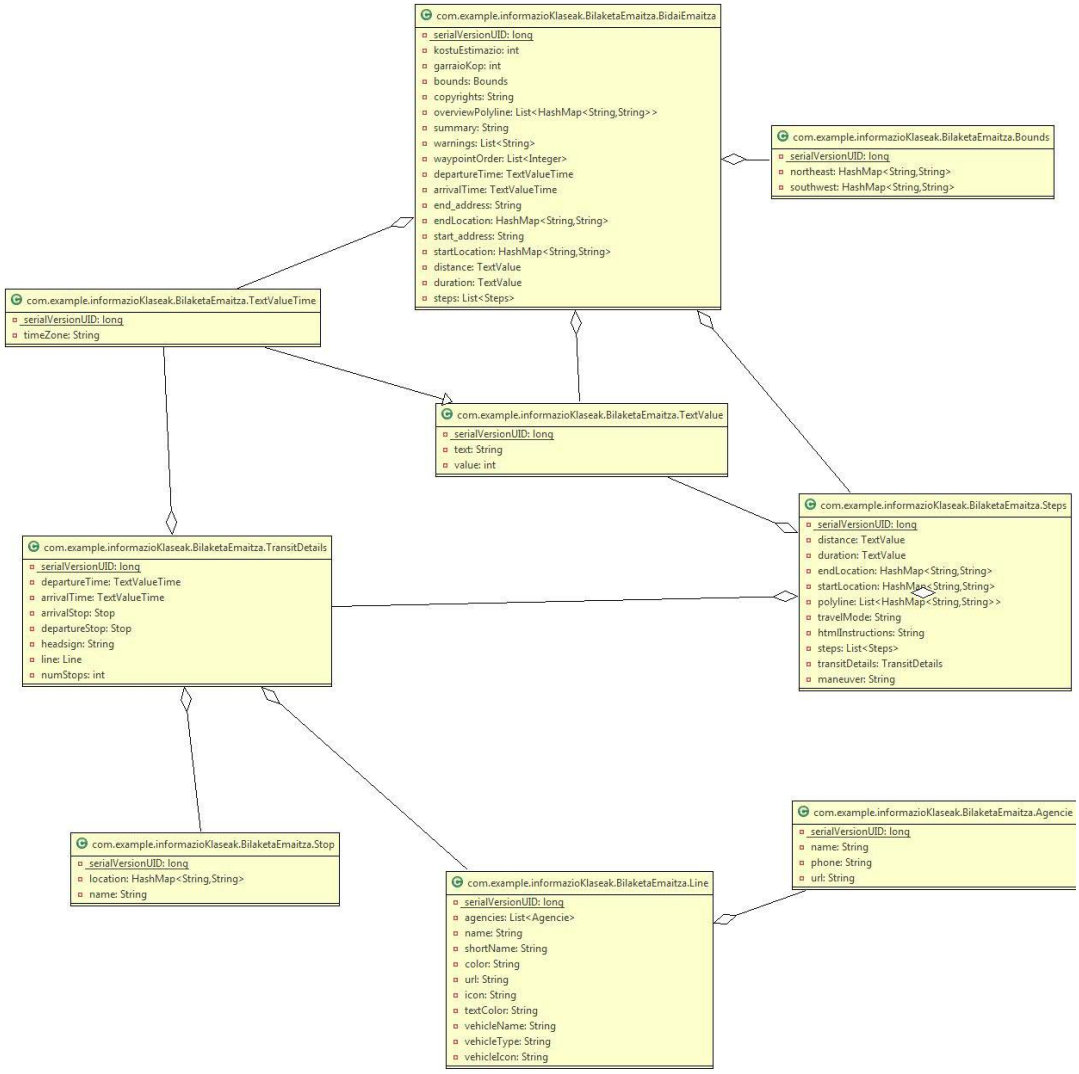


4.3. irudia: Helbideen domeinu eredua

4.1.3.2. Bidaia bat irudikatzen duten objektuak

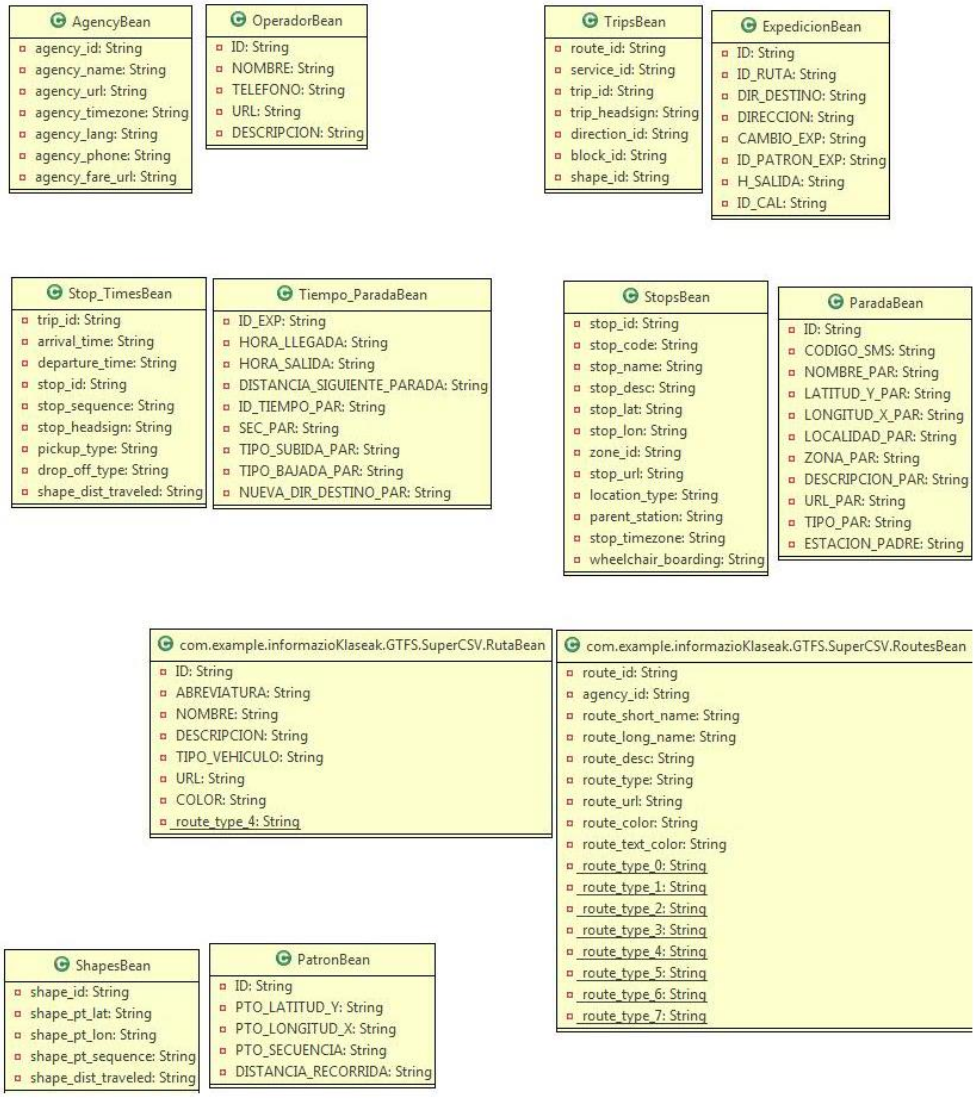
Behin bilaketa eginda, informazioa egokitu behar da hurrengo interfazera bidaltzeko eta pantailan erakutsi ahal izateko. Bidaia batek osatzen duen informazioa hainbat elementu desberdinez osatuta dago, 4.4. irudian erakusten den bezala:

- **BidaiEmaitza** (Bidaia): puntu batetik bestera joateko informazio osoa, hartu behar den edo diren garraioen informazioa, oinez egin behar diren zatien informazioa eta bidaia osoa deskribatzen duten datuek klase hau osatzen dute. Informazio hau beharrezkoa da.
- **Agencie**: bidaia batean, garraiozko zatia egiten duen konpainiaren informazioa. Informazio hau beharrezkoa da.
- **Bounds**: bidaia mapan non egongo den adierazten du ipar-ekialde eta hego-mendebalde puntuak adieraziz. Informazio hau ez da beharrezkoa.
- **Line**: garraioaren informazioa adierazten duen klasea. Informazio hau beharrezkoa da.



4.4. irudia: Bidaia irudikatzeko erabiltzen diren klaseak

- **Steps**: bidaia batean mugimendua adierazten duen zatia da, hau da, oinez edo garraioz egin behar den zatiaren informazioa. Hemen distantzia, denbora, ibilbidea eta mota horretakoak datuak adierazten diren dira. Informazio hau beharrezkoa da.
- **TextValue**: distantzia eta iraupena adierazten duen klasea da. Sistemaren egituragatik informazioa zenbakitan eta testuan eduki behar da. Informazio hau beharrezkoa da.
- **TextValueTime**: aurrekoaren berdina, kasu honetan ordea ordu eremua ere adierazten da. Informazio hau beharrezkoa da.
- **TransitDetails**: bidaia batean garraioz egin behar den zatiaren informazioa. Informazio hau beharrezkoa da.



4.5. irudia: Datu iturri klaseen domeinu eredua

4.1.3.3. Datu iturri klaseak

Objektu hauek zertxobait bereziak dira, garraioarekin zerikusia duten mundu errealeko objektuak irudikatzen dituzte, hala nola konpainia bat, linea bat, geltoki bat eta beste hainbeste. Objektu hauek datu baseen eremuak bezala funtzionatzen dute, ondorioz, 4.5. irudian ikusi daitekeenez, ez dute beraien arteko erlaziorik.

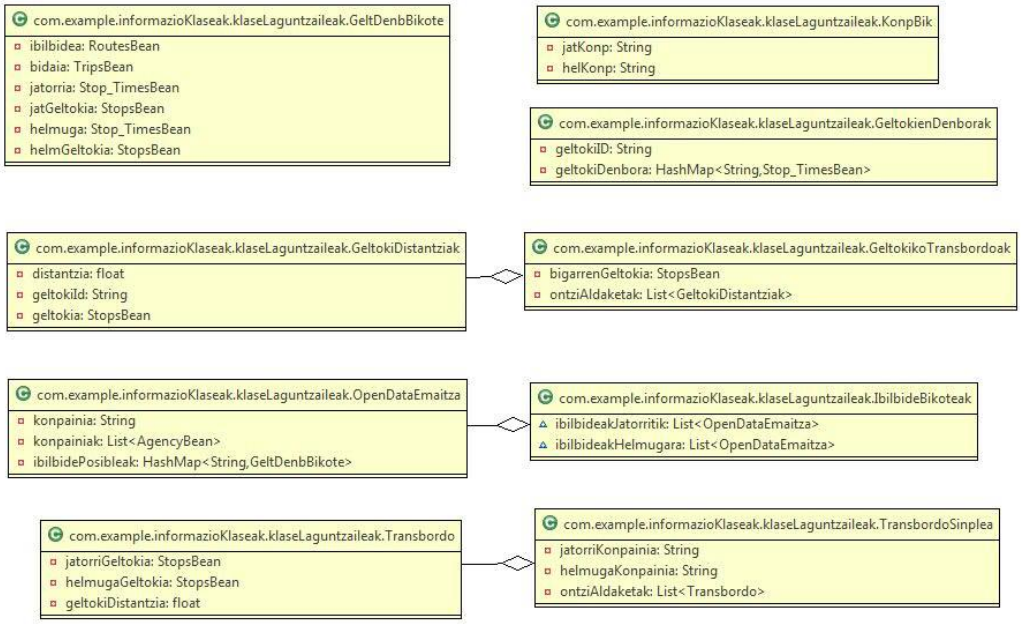
- **AgencyBean**: prozesatu gabeko garraio publiko konpainia baten informazioa.
- **OperadorBean**: Dbus konpainiaren informazioa gordetzen du.
- **RoutesBean**: prozesatu gabeko linea baten informazioa gordetzen du.
- **RutaBean**: Dbus konpainiaren linea bat gordetzen du.
- **ShapesBean**: ibilbide bat marrazten duen lerroaren puntu baten informazioa gordetzen du.

- PatronBean: Dbus konpainiaren ibilbide bat marrazten duen lerroaren puntu baten informazioa gordetzen du.
- Stop_TimesBean: geltoki baten informazio osagarria gordetzen du.
- Tiempo_ParadaBean: Stop_TimesBean bezalakoa baina Dbus-en informazioarekin soilik.
- StopsBean: geltoki baten informazioa du.
- ParadaBean: Dbus-en geltoki baten informazioa du.
- TripsBean: linea batek egiten duen ibilbidearen informazioa (kontuan izan behar da linea batek hainbat ibilbide desberdin egin ditzakeela).
- ExpedicionBean: Dbus konpainiaren linea batek egiten duen ibilbidearen informazioa.

4.1.3.4. Klase laguntzaileak

"*com.example.informazioKlaseak.klaseLaguntzaileak*" paketean biltzen diren klaseak bilaketa gauzatzerakoan laguntzeko sortuak dira. Funtzio batetik bestera datu zehatz batzuk bidali edo jaso behar direnerako, informazioa batzeko ordenatuagoa izateko eta horrelako gauzatarako erabiltzen dira. Adibidez, geltoki bikote lista gordetzeko klaseak edo ontzi aldaketaren arabera konpainiak biltzen dituen klaseak. 4.6. irudiak erakusten du klase hauen egitura.

- GeltDenbBikote: garraio batek egingo duen ibilbidea hainbat parametroren arabera definituta dago, klase honetan datu garrantzitsuenak definitzen dira. *ibilbidea* aldagaiak bidaiaren kodea eta informazio orokorra zeintzuk izango diren definitzen ditu. *bidaia* aldagaiak ibilbide honek zein bidaiari dagokion zehazten du. *jatorria* eta *helmuga* aldagaiek garraioa ze ordutan aterako den jatorritik eta helmugara ze ordutan ailegatuko den zehazten dute. Ez hori bakarrik, geltokien arteko distantzia ere zehazten du. *jatGeltokia* eta *helmGeltokia* aldagaiek geltokien informazioa, kokapena, izena... zehazten dute.
- GeltokiDistantziak: zehaztutako puntu batetik geltokira dagoen distantzia gordetzen da klase honetan. Distantziaz aparte, geltokiaren identifikatzailea eta geltoki beraren gehigarrizko informazio (irteera eta helmuga ordua, distantzia, eta abar) gordetzen da. Klase honen helburua distantzia batera dauden "geltokiak" lortzea eta distantziaren arabera automatikoki ordenatzea da.
- GeltokienDenborak: klase honek simulatzen du mundu errealeko geltoki batean aurki daitezkeen lineen denboren informazioa. Hemen geltokiaren identifikatzailea eta hortik pasatzen diren ibilbide guztiak gordetzen dira.
- GeltokikoTransbordoak: klase hau ontzi aldaketak helmugako geltokiaren arabera biltzeko erabiltzen da. Ontzi aldaketa desberdinak izan arren helmugako geltokia berdina izan dezakete, bidaia denboraren arabera edo ez izanda. Ondorioz, exekuzioa azkarragoa izateko, ontzi aldaketaren bigarren geltokiaren arabera biltzen dira.



4.6. irudia: Klase laguntzaileen domeinuaren eredua

- **IbilbideBikoteak**: jatorritik helmugara joateko bidaia zuzenekoan ezin denean bi bidaia egin behar dira. Klase honek jatorritik ontzi aldaketara eta hortik helmugara izango diren garraiozko ibilbide posible guztiak gordetzen ditu
- **KonpBik**: bidaiek jatorri bat eta helmuga bat dute, hauek geltoki batekin definitzen dira. Geltokiak zehazteko konpainiaren izenak erabiliko dira, hauek *Konfig* klasean definituta daude.
- **OpenDataEmaitza**: honen helburua da Open Data eta Dbus-ekin egindako bilaketen emaitzaren informazioa gordeta eta juntatuta izatea. Hemen puntu batetik bestera, garraio publikoz egiten den bidaiaren informazio osoa gordetzen da, ze konpainiarekin egiten da eta ibilbide posible guztiak. Gogoratu behar da jatorri eta helmuga artean ez dagoela bidaia bat soilik, baizik eta hainbat bidaia posible.
- **Transbordo**: mundu errealeko bi geltokien arteko distantzia gordetzeko klasea. Bere *geltokiDistantzia* aldagaia lerro zuzenean egindako kalkulua da, hau da, mapa bat hartuta bi puntuen arteko distantzia da. Ez da oinez ibiliko litzatekeen benetako distantzia.
- **TransbordoSinplea**: klase honek ontzi aldaketa egiteko informazioa gordetzen du. Jatorritik geltokira izango den, ontzi aldaketa ze bi geltokiren artean egingo den eta ontzi aldaketaren bigarren geltokitik helmugara doan konpainiaren izena gordetzen ditu.

4.2. Diseinua

4.2.1 Sistemaren arkitektura

GipuzkoaBidaian aplikazioaren arkitektura oso sinplea da, zerbitzari bitartekari propiorik ez duenez erabiltzen, kanpoko datu iturrietara besterik ez da konektatzen, 4.7. irudiak erakusten duen moduan.

Egitura honen abantaila da aplikazioaren muina osoa batera dagoela, era honetan, noizbait datu iturriek aldaketaren bat egiten badute, aktualizazio bat argitaratuz arazoa konpontzen da. Ez hori bakarrik, kanpoko zerbitzariak konexio azkarreko sareetan daudenez, adibidez, Google-eko zerbitzuak erabiltzean, informazioa oso denbora laburrean jasotzen da.

Zoritxarrez, arkitektura honek hainbat desabantaila ditu, aplikazioak era optimoan funtziona dezan zerbitzari bitartekari bat erabili beharko litzateke, informazioa gordetzeko, prozesatzeko eta bidaia eskaerei erantzuteko. Era horretan, aplikazioak edukiko lukeen lan karga askoz arinagoa izango litzateke.

Soluzio hau proiektuaren irismenetik kanpo geratzen da. Proiektuaren hasieran ezarri zen aplikazio bat sortuko zela, zerbitzari propiorik gabe. Ondorioz, soluzio hau inplementatzeak lan gehiegi suposatu lukeenez, xehetasunez azaltzen duen eranskina sortu dut, “C Eranskina: GipuzkoaBidaian-en funtzionamendua zerbitzari baten bidez”.

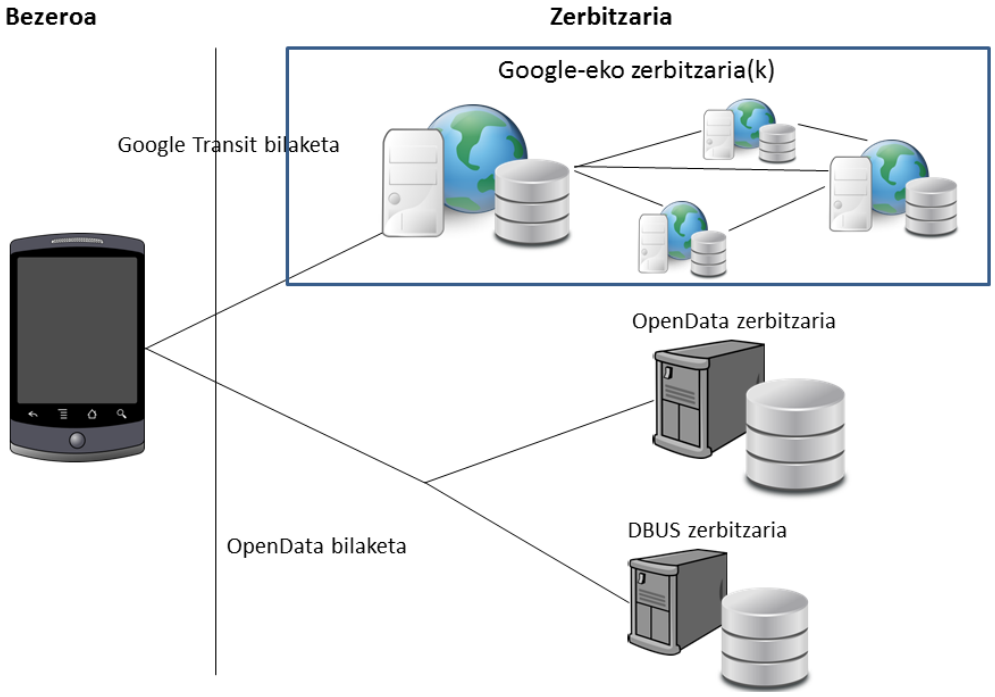
4.2.2. Aplikazioa kanpoko zerbitzuen menpe

GipuzkoaBidaian Internet gabe lan egiteko prest badago ere, kanpoko zerbitzuez baliatzen da ondo funtzionatu ahal izateko. Dena dela, aplikazioa erabiltzen den lehenengo aldian, “Bilaketa zehatza” aukerarekin, Interneteko konexioa eduki behar da.

Ez hori bakarrik, APP-a kanpoko zerbitzuez baliatzen diren funtzionalitatea asko ditu, horregatik oso garrantzitsua da Interneteko konexioa beti izatea.

4.2.2.1. Mapa erakusteko

Mugikor batean mapa bat ondo erakutsi ahal izateko lehenengo erabileran Interneteko konexioa beharrezkoa da. Behin aldi batean erabilita, cache memorian gordeta geratuko da ikusitako eremua eta horrela hurrengo erabilpenetan ez da beharrezkoa izango informazio guztia berriro deskargatzea.



4.7. irudia: Bezero eta zerbitzarien arteko erlazioa

4.2.2.2. Helbidea lortu koordenadetatik

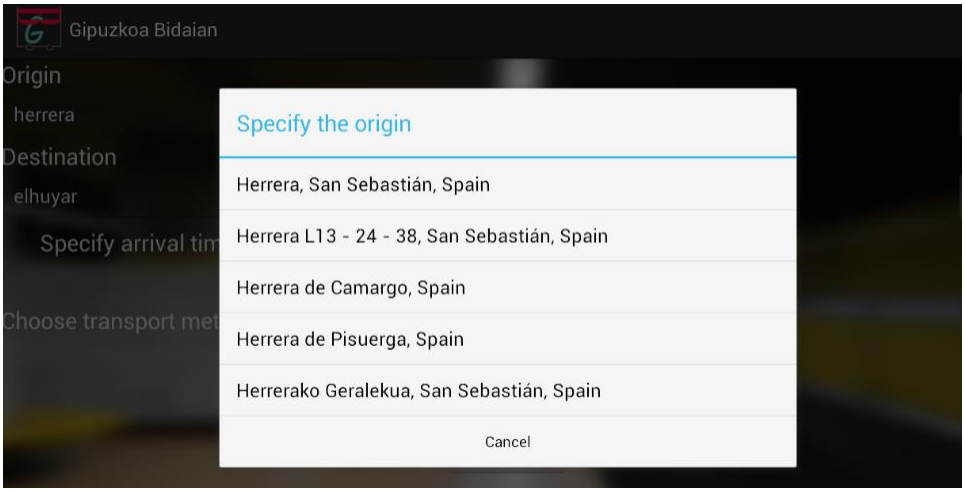
Google-ek API bat du zeinean latitudea eta longitudea bidaliz, dagokion helbidea, hiria, herria eta kode postala bueltatzen dituen. Ez hori bakarrik, Android-ek, "*android.location*" paketean, "*Geocoder*" izeneko klasea dauka eta lan hori automatikoki egiten du. Informazio eskakizuna egiten du, datuak jaso, prozesatu eta beharrezkoa soilik itzultzen du.

Arazoa da klase hori batzuetan ez dabilela. Arazoari aurre egiteko soluzio bat prestatu da: erroreren bat gertatzean hau detektatzen da eta informazioa beste funtzio baten bitartez lortzen da. Funtzio honek Google-eko API-ari zuzenean deitzeko URL-a prestatzen du eta beste klase bati helbidea bidaltzen dio. Informazioa Internetik zuzenean eskuratzen da, hau JSON formatuan egongo da prozesatu gabe, hortaz, hau irakurtzen da eta ondoren aplikazioaren sistemara egokitzen da. Azkenik, beharrezko datuak hartzen dira eta hasierako funtzioari itzultzen zaizkio.

Internetek funtzionatzen ez badu, ezinezkoa izango da hau egitea, ezingo litzatekeelako helbide erreala lortu. Orduan koordenatuekin soilik lan egin beharko litzateke.

4.2.2.3. Helbidea lortu helbide batetik

Erabiltzaileak testu hutsunean helbide bat idatzi arren, hau ez da beti zuzena izaten. Egiaztapen hau Google Maps-ek eskaintzen duen API baten bitartez egiten da, mundu mailan funtzionatzen duen zerbitzu bat. 4.8. irudian ikusten da zer gertatzen den jatorri anbiguo bat jartzean: izen hori daukaten hainbat posibilitate daudela eta zuzena aukeratu behar dela.



4.8. irudia: Jatorri helbidea zehaztu

Hala ere, Google-eko sistemak azkarrak dira eta soilik estatu mailako aukerak erakutsiko ditu, atzerriko izen bat jartzen ez den bitartean noski.

Aplikazioak helbide posibleak erakusterakoan aukera bakarra dagoela detektatzen badu, hots, idatzi dena helbide bakarra bati dagokiola, zuzenean hori aukeratzen da.

4.2.2.4. Google-eko bidaiak

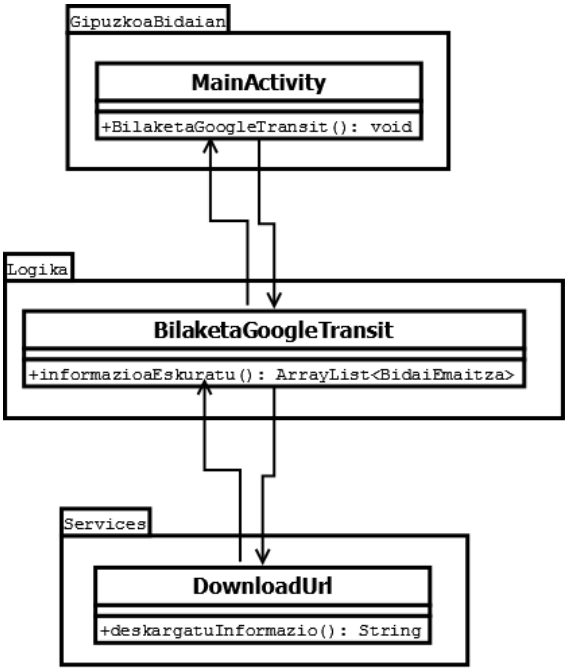
Bilaketa mota hau Google Directions API-aren erabileran oinarritzen da. Kanpoko zerbitzu honek lan karga handiena eramaten du, hau da, aplikazioak informazio eskakizuna egiten dio, emaitzak jaso eta erakutsi besterik ez du egiten.

```
https://maps.googleapis.com/maps/api/directions/json?origin=43.28468104405296,-2.1793729439377785&destination=43.2740300347191,-1.9710668921470642&sensor=false&mode=transit&departure_time=1385928496&language=es&region=es
```

4.1. taula: Google Direction API dei adibidea

Hau egiteko klase nagusiak erabiltzaileak zehaztu dituen parametroekin URL bat prestatzen da, 4.1. taulako adibideak erakusten duen bezala:

- Web helbide nagusia: "https://maps.googleapis.com/maps/api/directions/"
- Irteera modua: informazioa jaso nahi den formatua, Google-ek bi aukera ematen ditu, XML eta JSON. Beti bigarrena aukeratuko da.
- Jatorria: latitudea eta longitudea adierazi behar dira.
- Helmuga: latitudea eta longitudea adierazi behar dira.
- Sentsorea: aukera hau GPS-aren bidez bilaketa zehatzagoa izateko eskaintzen da, aktibatuta edo desaktibatuta egon daiteke. Beti desgaitua egongo da normalean jendeak GPSa ez duelako piztuta eta bilaketa azkartzen delako.



4.9. irudia: Bilaketa azkarra informazio eskuraketa

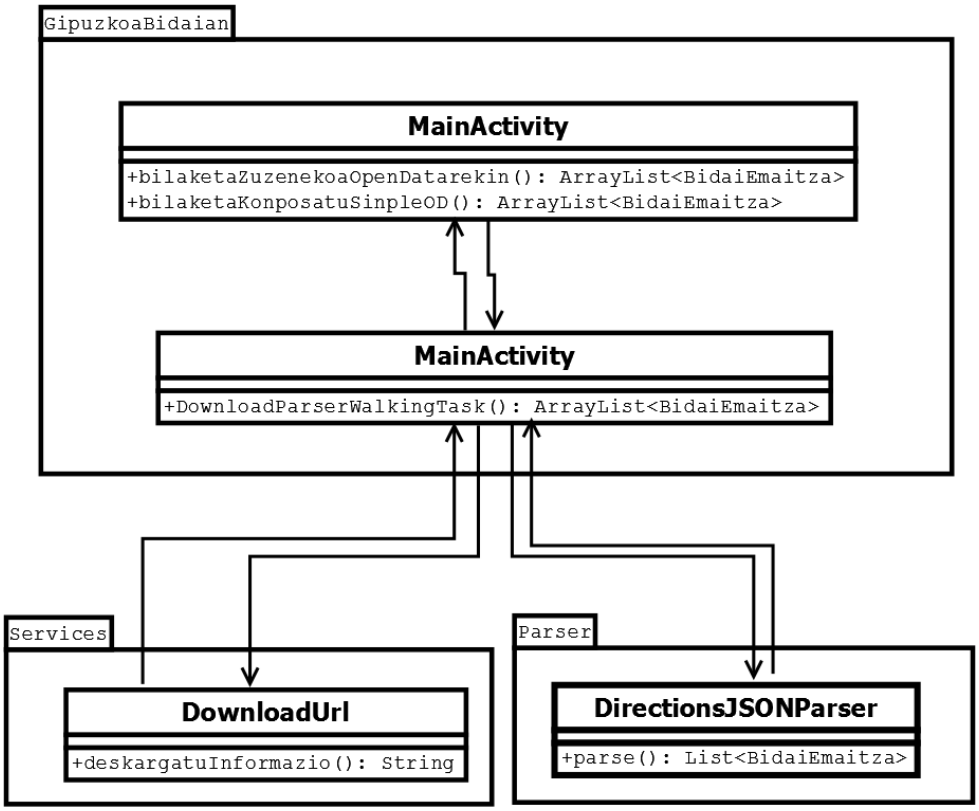
- Modua: API honek ez du soilik balio garraio publikoko bidaiak bilatzeko (TRANSIT) , oinez (WALK) eta autoz (CAR) ere bilatu daiteke. Ondorioz, lehenengo, aukera adierazi behar zaio.
- Ailegatze ordua: helmugara ze ordutan iritsi nahi den adierazi behar zaio. Datu hau beharrezkoa da, GipuzkoaBidaian-en ordea, hautazkoa denez, erabiltzaileak ez duenean ordua zehazten unekoa hartzen da.
- Hizkuntza: Google-eko zerbitzuak hainbat hizkuntzatan daudenez, aplikazioak detektatzen du unekoa eta horren arabera emaitzak irakurriko dira.
- Eskualdea: datu hau aukerakoa da, hala eta guztiz ere beti automatikoki definituko da, bilaketa azkartzen baita.

Informazio hau beste klase bati bidaltzen dio. Honek ez du zuzenean informazioa eskuratzen, horretarako hirugarren klase bat dago, zeinek Interneteko informazioa eskuratzeaz arduratzen dena. 4.9. irudiak irudikatzen du informazio ibilbide hau.

Erabilitako egitura hiru geruzetako arkitekturan oinarritzen da, non *MainActivity* klaseak soilik informazioa eskatzen eta erakusten duen (errealitatean hurrengo interfazea arduratzen da erakusteaz, baina ideia nagusia ez da aldatzen), esan liteke "*GipuzkoaBidaian*" paketeak aurkezpen geruza simulatzen duela.

BilaketaGoogleTransit bilaketa parametroak jasotzen ditu eta URLa prestatzen du, behin informazioa edukita hau irakurtzen du, prozesatzen du eta prest uzten dio hasierakoari itzultzeko. Klase hau dagoen paketeak, "*Logika*", negozio logika geruza simulatzen du, non informazioaren prozesamendua eta eskuraketa (ez iturritik) suertatzen diren.

Azkenik Datu base geruza dator. Hemen, *DownloadUrl* klaseak kanporako deiak egiten ditu eta erantzunak bueltatu. Zati honek "zuzenean" datu iturriarekin lan egiten du.



4.10. irudia: Oinezko informazioa lortzeko prozesua

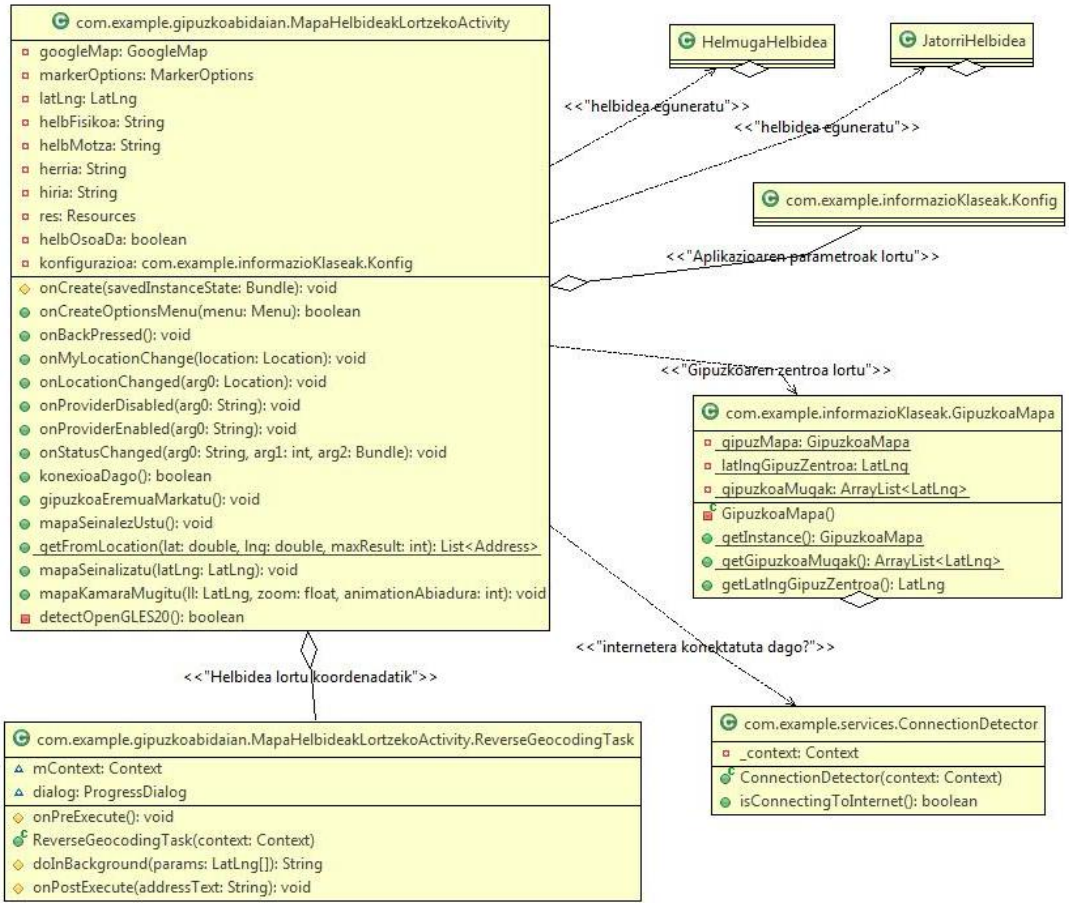
Arkitektura hau jarraituz, noizbait Google-ek bere zerbitzua aldatzen badu ez dut kode guztia aldatu beharko, soilik dagokion zatia, denbora eta arazoak aurreztuz.

4.2.2.5. Oinezko zatiak lortu

Open Data-ko edo Dbus-eko datuak erabiltzen direnean garraio publikoko bidaiak besterik ez dira lortzen . Ondorioz, informazio zati bat falta da geltokietara joateko, oinezko urratsak. Hauek Google Directions API-a erabiltzen lortzen dira, garraio publikoko bilaketak egiteko erabilitako web zerbitzu bera.

Oinezko informazioa lortzeko era sinpleagoa da bidaia osoak lortzekoa baino. Jaso eta prozesatu behar den informazioa askoz laburragoa denez, ez da prozesu konplikatuegia, hala ere, 4.10. irudian ikus daitekeen moduan, informazioak hiru pausu dauzka.

Informazioa behar duten funtzioak, bi kasu hauetan, *DownloadParserWalkingTask* funtzioari deitzen diote. Honek *DownloadUrl* klaseari deitzen dio Internetera konektatzeko eta beharrezkoa deskargatzeko. Ondoren, tratatu gabeko informazioa bueltatzen da eta bera arduratzen da aztertzaile klasera, *DirectionsJSONParser*, deitzeaz. Datuak egokitu eta gero, *BidaiEmaitza* klaseko objektuak sortu eta gero, hau klase nagusira bueltatzen da. Azkenik, informazioa eskatu zutenengana bueltatzen da eta aplikazioaren exekuzioa jarraitzen du.



4.11. irudia: Helbideak mapatik lortzeko klase diagrama

4.2.3. Klaseen diagrama

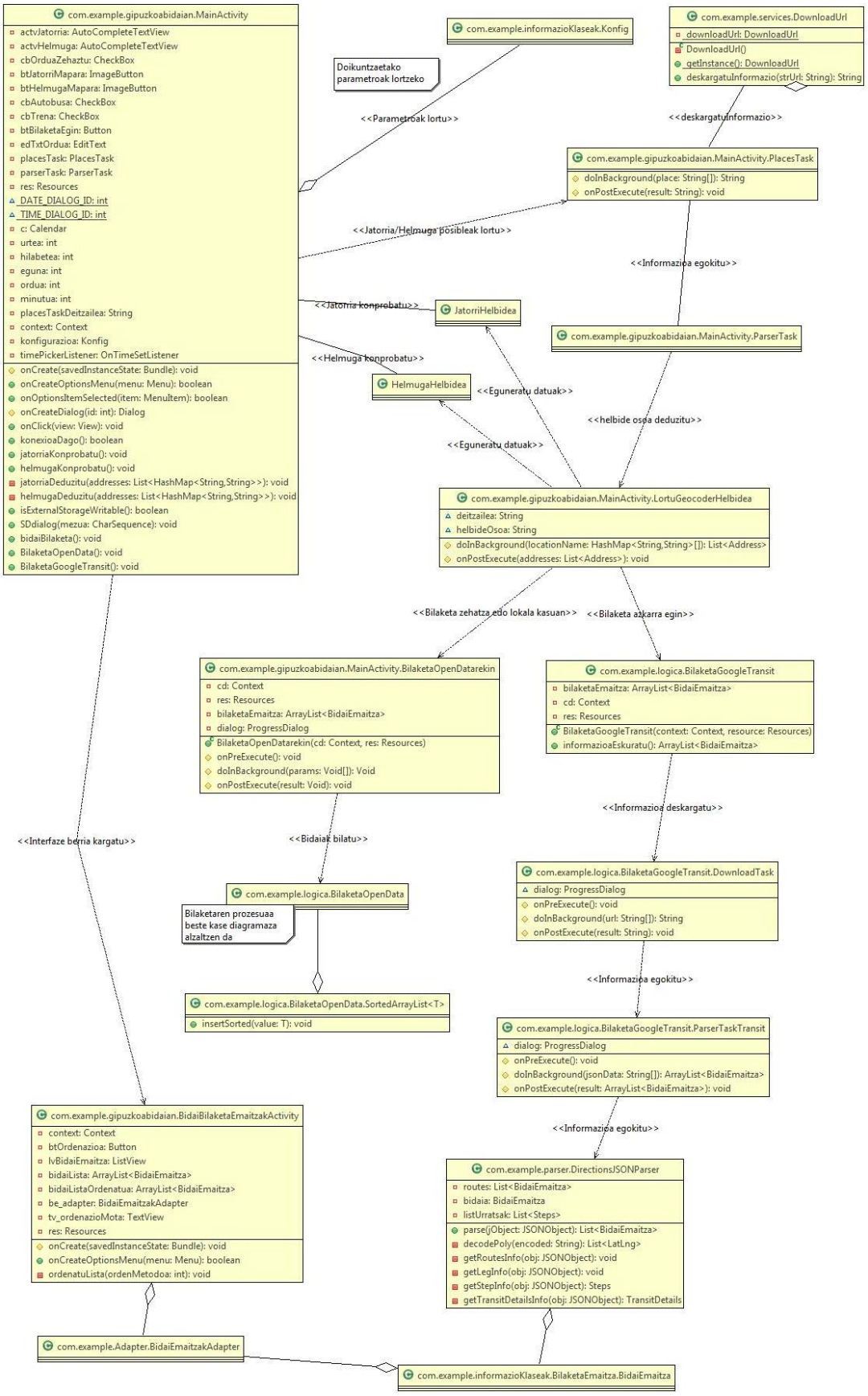
Erabilpen kasuak ezagututa, hauen klase diagramak azaltzea falta da. Horretarako parte hartze duten klaseak bilatuko dira, garrantzitsuenak azaldu eta beraien arteko erlazioak definituko dira.

Ez dira erabilpen kasu guztiak aztertuko, soilik interesgarrienak, hots, helbideak mapatik lortu eta bidaien bilaketa.

4.2.3.1. Helbidea mapatik lortu

Helbidea mapatik lortzeko prozesuan hainbat klase parte hartzen dute, 4.11. irudiko diagraman ikusi daitekeen moduan.

Klase garrantzitsuena *MapaHalbideakLortzekoActivity* da, hor prozesu guztiak kudeatzen dira. *FragmentActivity*-tik hedatzen duenez, sorkuntzan *onCreate* funtzioari automatikoki deitzen zaio, honek pantailan ikusten dena prestatzen du eta interfazearen elementu bakoitzaren jokaera definitzen du.



4.12. irudia: Bidaien bilaketaren zati baten klase diagrama

Mapa ukitzean *mapaSeinalizatu* funtzioari deitzen zaio (maparen maneiatzailearen bidez), latitudea eta longitudea hartu, *ReverseGeocodingTask* klasea exekutatu eta helbidea lortzen da. Informazioa gordetzeko botoia sakatzerakoan *HelmugaHelbidea* edo *JatorriHelbidea* klasea eguneratzen da eta *MainActivity*-ra bueltatzen da.

ReverseGeocodingTask klasea *AsyncTask* klasetik hedatzen denez atzeko mailan exekutatuko dela esan nahi du. Funtzio honek jasotzen duen koordenada helbide batean transformatzen du Androideko *Geocoder* klasearen bitartez.

4.2.3.2. Bidai bilaketa

Bidaia bilatzeko prozesua hain handia da ezin dela diagrama bakar batean erakutsi, ulergarria izateko bi zatitan banatu behar da, bata bilaketa egiten duen funtzioari deitu, informazioa jaso eta bestea interfazeari deitzen diona eta bestea bilaketa bera.

Bilaketaren inguruan

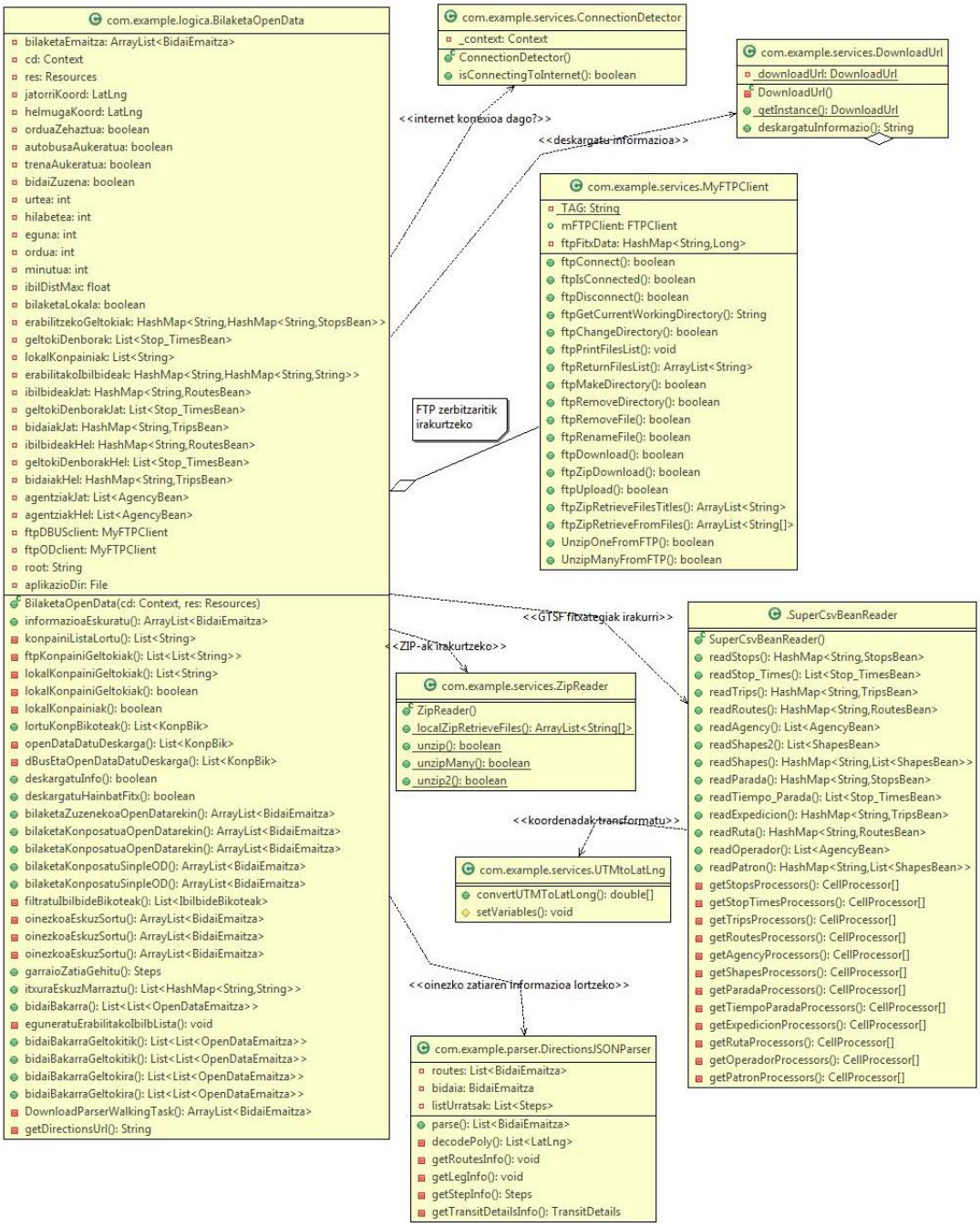
4.12 irudiaren diagraman ikus daiteke bilaketaren inguruan parte hartzen duten klaseak, baina ez bilaketa gauzatzeko.

MainActivity klasea garrantzitsuena da, hortik prozesu osoa kudeatzen da. Erabiltzaileak bilatzeko botoia sakatzean jatorri helbidea begiratuko da *jatorriaKonprobatu* metodoarekin, zuzena bada helmugarekin jarraituko da, bestela *PlacesTask* klaseari deituko zaio. Honek Google-ko Autocomplete API-ari deituko dio idatzitako helbidea aztertzeko. Jasotako emaitza *ParserTask* klaseari bidaliko, honek egokituko du ulergarria den helbide zerrenda batean eta azken aurreko urratsari bidaliko dio, *jatorriaDeduzitu*. Urrats honetan erabiltzaileari helbide lista bat agertuko zaio pantailan, bat aukeratu eta horren informazioa gordeko da. Azkenik, aukeratutako helbidea jakinda, faltatzen den informazioa eskuratuko da, koordenadak, *lortuGeocoderHelbidea* klasearen bitartez.

Jatorriaren prozesua egin ondoren helmugakoarekin egingo da, urratsa berdinak jarraituz. Behin bien helbideak izanda *konfigurazioa.xml* fitxategia eguneratuko da, bilatutako helbide listari bi berriak gehituko zaizkio, dagoeneko ez badaude.

Helbideak izanda sistemak begiratzen du zein bilaketa egin behar duen, azkarra bada *BilaketaGoogleTransit* metodoari deituko dio, zehatza bada ordea, *BilaketaOpenData* metodoari deituko dio. Bigarren kasuan, sistemak aurretik egiaztatzen du mugikorrak SD txartela duela *isExternalStorageWritable* funtzioarekin.

Edozein bilaketa metodo aukeratuta, behin bidaien emaitza jasota *BidaiBilaketaEmaitzakActivity* klaseari deituko dio hauek pantailan erakusteko.



4.13. irudia: Bilaketa zehatzaren klase diagrama (funtzio eta metodo guztiei parametroak kendu zaizkie irudia hobeto ikusteko)

Bidai bilaketa

Bilaketa azkarreko kasuan ez dago ezer azaltzeko, ia lan gehiena Google-eko zerbitzariek egiten dutelako. Hemen bilaketa zehatzean parte hartzen duten klaseak azalduko dira. Hauetako garrantzitsuenak ikus daitezke 4.13. irudiko diagraman. Gehiegi direnez ez dira denak erakusten, gainera Domeinuaren Ereduan atalean klase batzuk agertzen dira.

BilaketaOpenData klasea bilaketa prozesuaren ardatz nagusia da, *informazioaEskuratu* funtzioa bilaketa egiten duen arduraduna da. Honek beharrezko funtzio guztiei deitzen die eta bidaia emaitza lista itzultzen du.

Lehenik *konpainiaListaLortu* funtzioari deitzen zaio, honek *Konfig* klasean dagoen garraio publikoen konpainia lista lortzen du. Ondoren, *ftpKonpainiGeltokiak* funtzioari deitzen zaio, honek , bilaketa lokala ez bada, FTP zerbitzarira konektatzen da geltokien zerrenda duten fitxategiak deskargatzeko, horretarako *MyFTPClient* klaseaz baliatzen da.

Geltoki guztiak izanda gertuenak bilatzen dira eta bilaketan parte hartzen duten konpainiak lortzen dira. Behin hauek izanda falta den informazioa eskuratu behar da, lehenengo Internetetik deskargatzen da *MyFTPClient* klasearekin eta ondoren deskonprimatzen da *ZipReader* klasea erabiliz.

Jatorriko informazioa Dbus konpainiaren zerbitzarietatik etortzen bada orduan hau egokitu behar da, horretarako *Bean* motako klaseak arduratzen dira. Arazoa da geltokien koordenadak UTM formatuan daudela eta aplikazioak LatLng-an behar dituela. Horretarako *UTMtoLatLng* deitzen zaio, bere funtzioa *convertUTMToLatLong* beharrezko kalkulu matematikoak egiten baititu.

Konpainien informazio guztia prest egonda (deskargatuta eta memoria biltegiratuta), *SuperCsvBeanReader* klasearekin fitxategiak irakurtzen dira eta *Bean* motako klaseetara egokitzen dira.

Hurrengo urratsa bilaketa gauzatzea da eta *OpenDataEmaitza* motako objektuetan gordetzea. Hau egiteko ez da beharrezkoa beste inori deitzea, bera bakarrik egiten du dena. Arazoa da datuak ez daudela itzultzeko prest, hauek *BidaiEmaitza* klasera moldatu behar dira. Horretarako emaitza bakoitza lortzean hau hartzen da eta forma egokia ematen zaio.

Oinezko zatiak oraindik falta dira, horretarako *DirectionsJSONParser klaseari* deitzen zaio, informazioa jasotzeko eta gordetzeko.

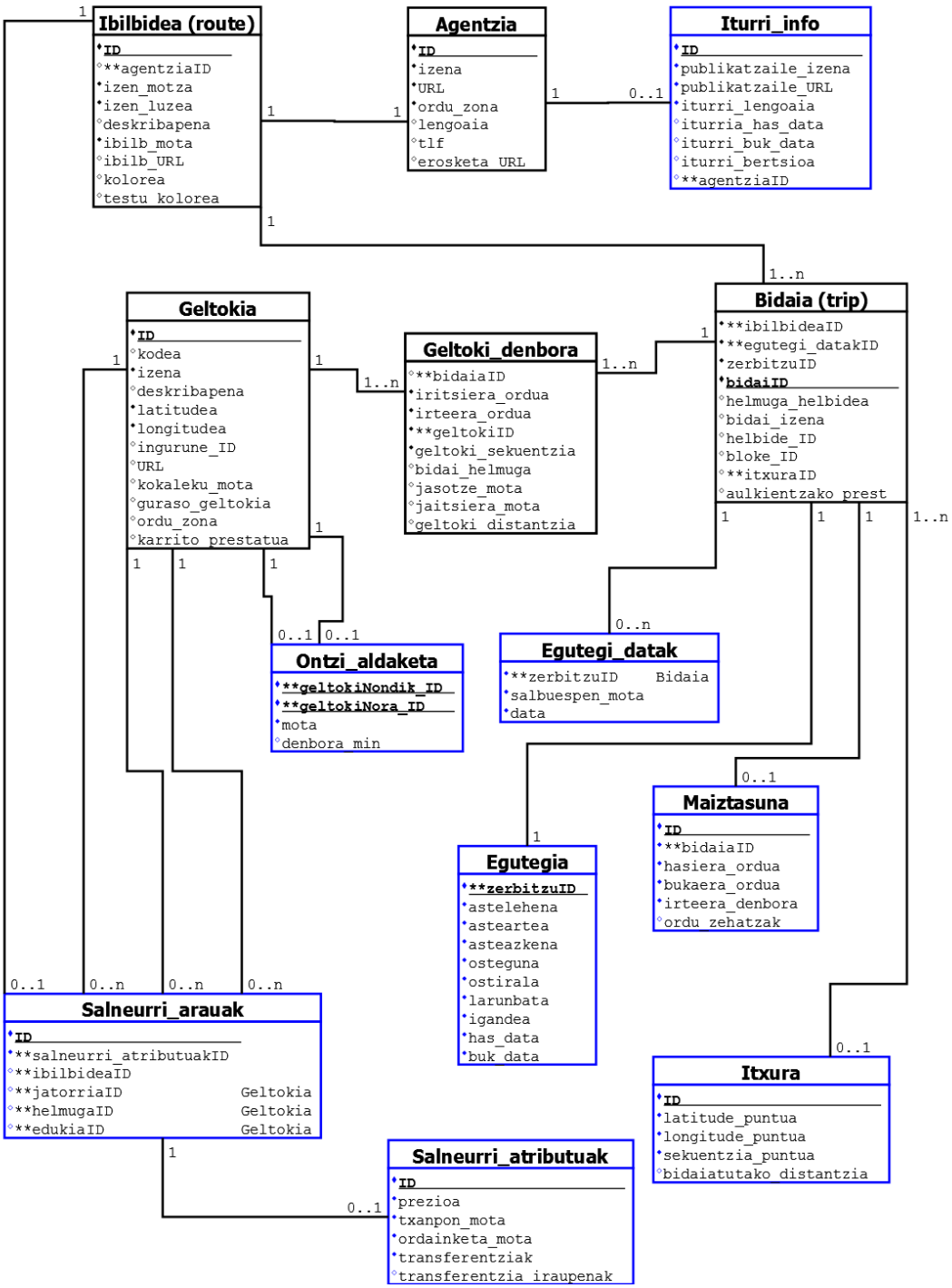
Azkenean *BidaiEmaitza* klaseko objektu lista bat lortzen da eta *MainActivity*-ri bueltatzen zaio aplikazioaren exekuzioarekin jarraitzeko.

4.2.4. Datu basearen diseinua

GipuzkoaBidaian-ek ez du MySQL motako datu baserik erabiltzen, informazio guztia fitxategietan gordetzen da. Hau arrazoi sinple batengatik egiten da, ez delako errentagarria datu base bat edukitzea, lan gehiegi suposatuko luke bere kudeaketak. Hala ere, egongo balitz 4.14. irudiaren egitura izango luke (kolorezko taulak hautazkoak dira eta bi izartxo dituzten eremuak gako arrotzak).

Aplikazioak, informazioa kanpoko zerbitzarietatik lortzen du eta mugikorrean bertan biltegiratzen du, zehatz-mehatz SD txartelean. Bidaiak bilatzean memoriako fitxategiak irakurtzen dira, prozesatzen dira, bilaketa egiten da eta emaitza erakutsi. Datu base bat izanez gero, bilaketak han zuzenean egingo lirateke, fitxategi originalak berriro irakurri beharrik gabe. Metodo horrek denbora aurreztuko luke bilaketetan, hori garbi dago, baina soilik hor.

Ideia horren alde txarra datu basearen sorkuntzan dator, aldi gehiegitan eguneratu beharko litzatekeelako. Mota hauetako aplikazioak ez daude pentsatuta normalean erabiltzeko. Ondorioz, bilaketak ahalik eta azkarrenak izan behar dira.



4.14. irudia: Bidaia bilaketarako datu base EE_UML diagrama

Datu base bat edukiz gero hau sortu edo eguneratu beharko litzateke bilaketa bat egitean, hau da, fitxategiak deskargatu ondoren. Xehetasun oso garrantzitsu bat da azkarragoa dela fitxategiak deskargatzea eta irakurtzea, Internetetik zuzenean irakurtzea baino (gertaera hau nire aldetik egiaztatua izan da). Ez hori bakarrik, Open Data-ko informazioa fitxategi konprimituetan dago, irakurketa moteltzen duelarik.

Aplikazioak, deskargatzeko, irakurtzeko eta prozesatzeko behar duen denborari, formatua eman eta datu basean gordetzeko denbora gehitzen badiogu, bilaketaren denbora dezente handituko da. Egia da hurrengo erabileran murriztu egingo litzatekeela, baina, eguneratu edo zerbait gehitu behar litzatekeen aldi guztietan arazo honen aurrean egongo ginateke. Gainera, jatorriko fitxategiak erregularki eguneratzen direla kontuan hartuz, datu basearen eguneratzea eta ondorioz bilaketen mantsotzea gehiegitan gertatuko litzateke.

Ez hori bakarrik, informazio berria deskargatzen den heinean datu basea handitzen joango litzateke, eremuak eguneratzeko beharrezko denbora handituz. Kontuan hartu behar da Open Data-tik datorren informazioa batera biltegiratzeko prest datorrela, ez baitago oinarrizko gakoen errepikapenik.

Ondorioz, datu basea ez erabiltzea erabaki zen, aurretik aipatutako mantsotzeagatik eta baita datu baseak har dezakeen tamainagatik ere. Ez memorian okupatuko lukeenagatik, SD txartelean biltegiratuko litzatekeelako, baizik eta bilaketa arrunt bat egiteko irakurri beharko litzatekeen informazio kopuruagatik.

4.2.5. Sekuentzia diagramak

Atal honetan erabilpen kasuen sekuentzia diagramak azalduko dira, erabiltzaileak ekintza bat egiten duenean zer gertatzen den sisteman azalduko da.

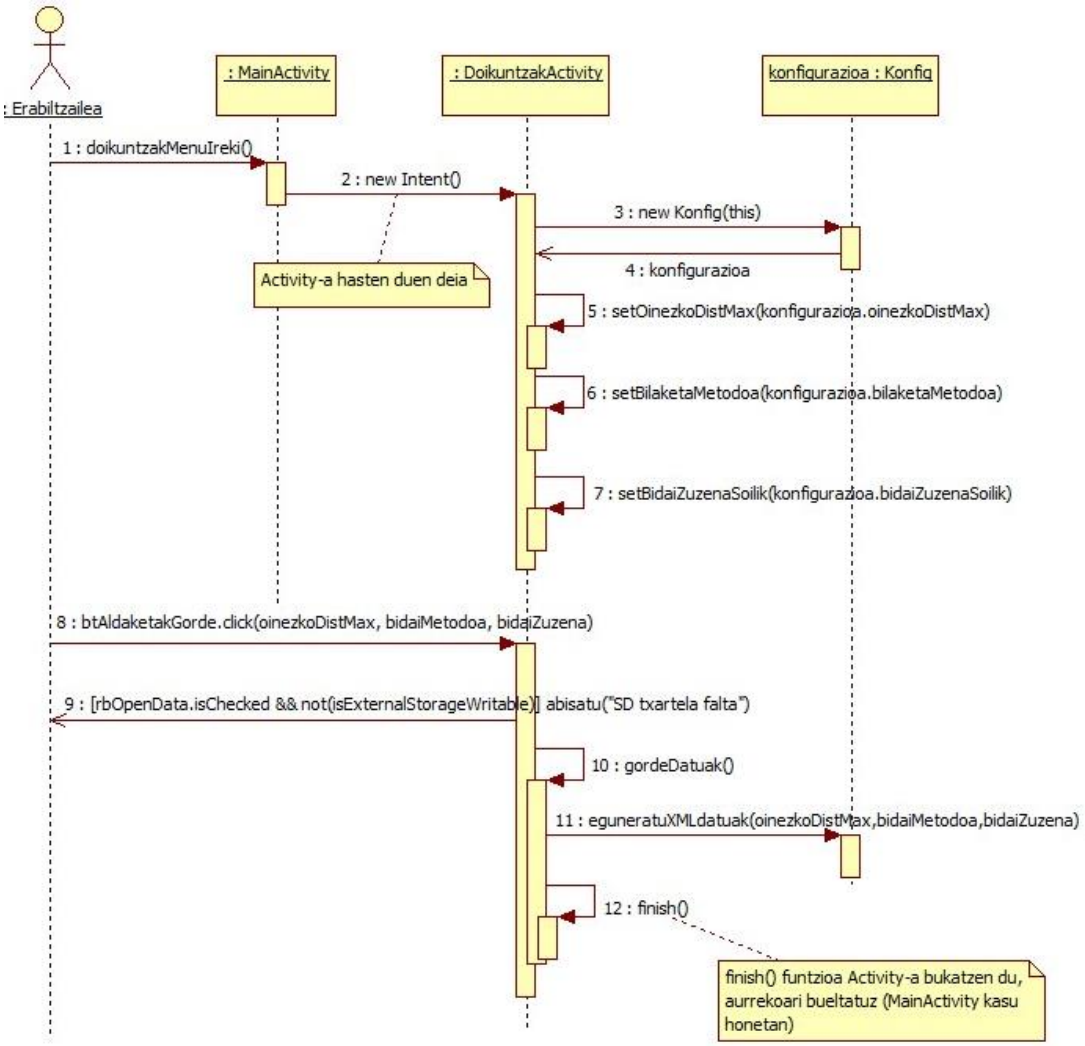
4.2.5.1. Bilaketa parametroak aldatu

4.15. irudiaren diagramak irudikatzen duen bezala, erabiltzaileak doikuntzetara joateko botoia sakatzean, *Doikuntza* klasea kargatuko da *Konfig* klaseko uneko datuekin eta interfazea beteko da (distantzia maximoa, bilaketa mota aukera eta bidaia zuzena egingo den ala ez).

Erabiltzaileak datu hauek aldatu eta gero informazioa gordetzeko botoia sakatuko du. Bilaketa mota zehatza den (Open Datakoa) eta mugikorrak SD txartelik duen begiratzen du lehenengo sistemak (hainbeste informazio deskargatu behar da, ez dela batere gomendagarria barruko memorian gordetzea), horrela ez bada mezu bat idatziko du pantailan eta ez du ezer gehiago egingo.

Dena ondo badago orduan *Konfig* klasearen *EguneratuXML* funtzioari deituko dio, honek informazio berria hartu eta *konfigurazioXML* fitxategia eguneratzen du.

Argitu behar da aplikazioaren bilaketa parametroak XML fitxategia batean gordetzen direla, horrela aplikazioa ixten denean ez da uneko konfigurazioa galduko.

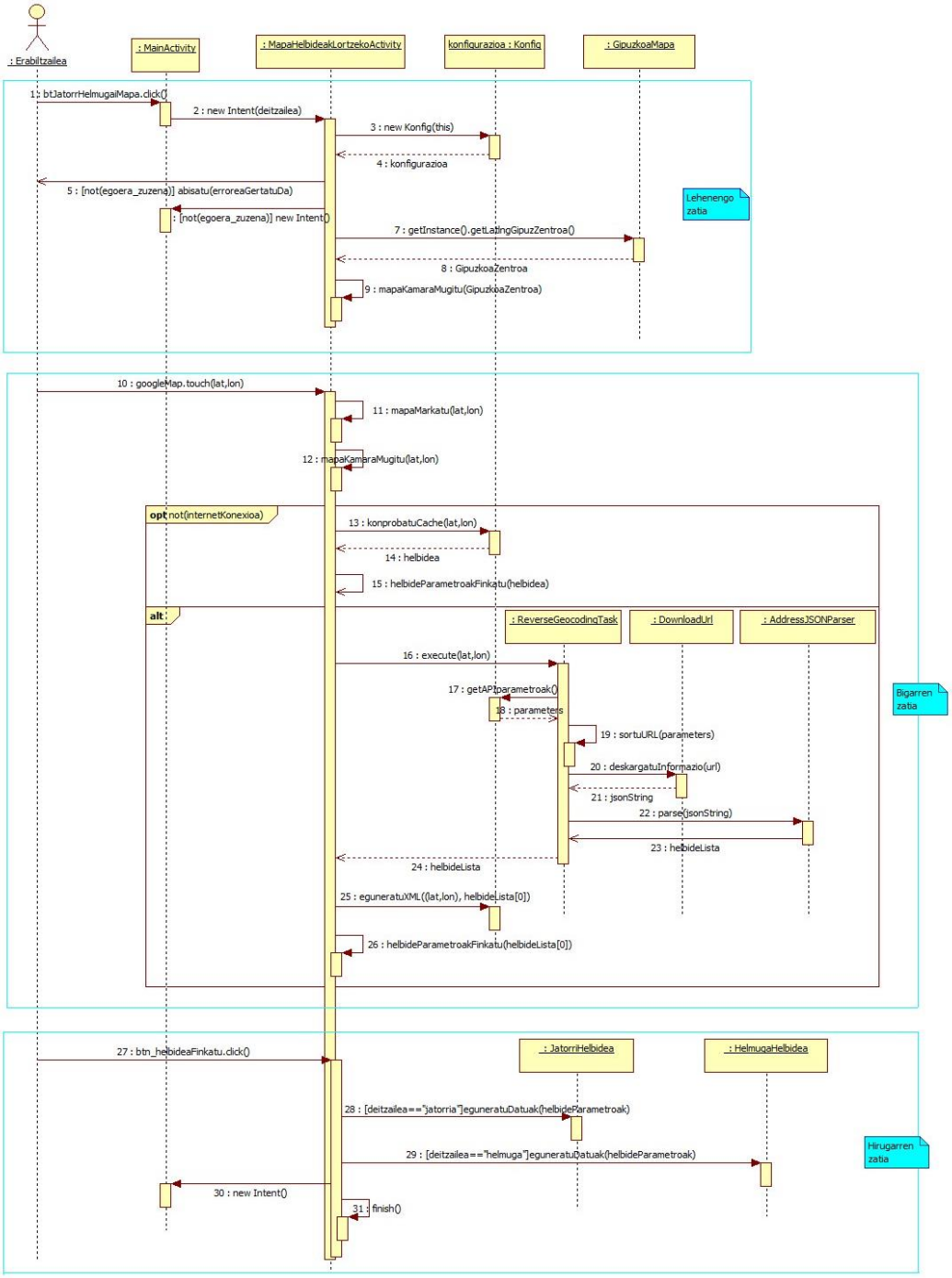


4.15. irudia: "Bilaketa parametrok aldatu" sekuentzia diagrama

4.2.5.2. Helbidea mapatik lortu

Kasu hau berezi xamarra da, sekuentzia diagrama hiru zatitan banatzen da, 4.16. irudian hiru laukiak adierazten duten bezala. Lehenengoa erabiltzaileak interfaze berria irekitzen duenerako da, bigarrena mapa ukitzen den unekoa eta hirugarrena helbidea lortzeko botoia sakatzen denekoa. Hirurak batuz helbidea mapatik lortzeko prozesua azaltzen da.

- **Lehenengo zatia**: Erabiltzaileak jatorria edo helmuga helbidea mapatik lortzeko botoia sakatzen du eta interfaze berria irekitzen da. *Activity*-a (interfazea) kargatu baino lehen egiaztapen batzuk egin behar ditu: mugikorrak OpenGL ES 20 onartzen duela eta Google Play Services ondo funtzionatzen duela. Horrela bada mapa kargatuko da, Gipuzkoako eremua seinalatuko du eta kamara mugituko du.



4.16. irudia: Helbidea mapa bidez lortzeko sekuentzia diagrama

- **Bigarren zatia**: Erabiltzaileak mapa ukitzean, puntu horren latitudea eta longitudea lortuko dira eta mapa hara mugituko da. Ondoren Interneteko konexioa dagoen ziurtatuko da, ez badago cachean gordetako koordenadetan bilatuko da. Cache memorian, gordetzen diren datuen artean, lortutako koordenada guztiak daude, bakoitzari dagokion helbidearekin. Era honetan Interneteko konexiorik gabe ere lan egin daiteke. Kontrako kasuan *ReverseGeocoding* klasea exekutatuko da atzeko planoan. Hemen *Geocoder*

klasea baliatuz (Androideko klasea), latitude eta longitudeari dagozkien helbidea zuzenean lortu daiteke. Arazoa da honek askotan huts egiten duela. Kasu horretan, gauza berdina egingo da baina eskuz. URLa prestatuko da eta *DownloadUrl* klasearen *deskargatuInfo* funtzioari deituko zaio informazioa lortzeko. Hau jaso ondoren *AddressJSONParser* klasearen *parse* funtzioari deituko zaio helbide lista lortzeko. Behin listaren lehenengo helbidea edukita *konfigurazioXML* fitxategia eguneratuko da koordenada eta helbide berriarekin eta helbide parametro orokorrak eguneratuko dira.

- Hirugarren zatia: Helbidea lortzeko botoia sakatzean jatorria edo helmuga helbidearen informazioa gordetzen duen klasea eguneratuko da. Azkenik klase nagusia, *MainActivity*, martxan jarriko da eta hau itxiko da *finish* funtzioarekin.

4.2.5.3. Bidaia bilaketa egin

Erabilpen kasu hau garrantzitsuena da, aplikazioaren muina da azken batean. Prozesu guztiaren sekuentzia diagrama zehatza egitea luzeegia izango litzatekeenez, 4.17. irudian ideia orokorra erakusten. Atal garrantzitsuenak Inplementazio atalean berriro eta zehatzago azalduko dira.

Sekuentzia diagrama hainbat zatitan banatzen da, lehenik jatorria eta helmuga helbideak zuzenak diren begiratzen da. Maparen bidez lortu badira, exekuzioarekin jarraituko da, kontrako kasuan Google-eko API bati deitzen zaio helbidea lortzeko. Behin informazio guztia izanda, Negozio Logikan dagoen klase bati deituko dio bilaketa parametro guztiekin, hau Google Direction zerbitzutik bilaketa egiteko izango da edo Open Data eta Dbus zerbitzarien informazioarekin lan egiteko izango da.

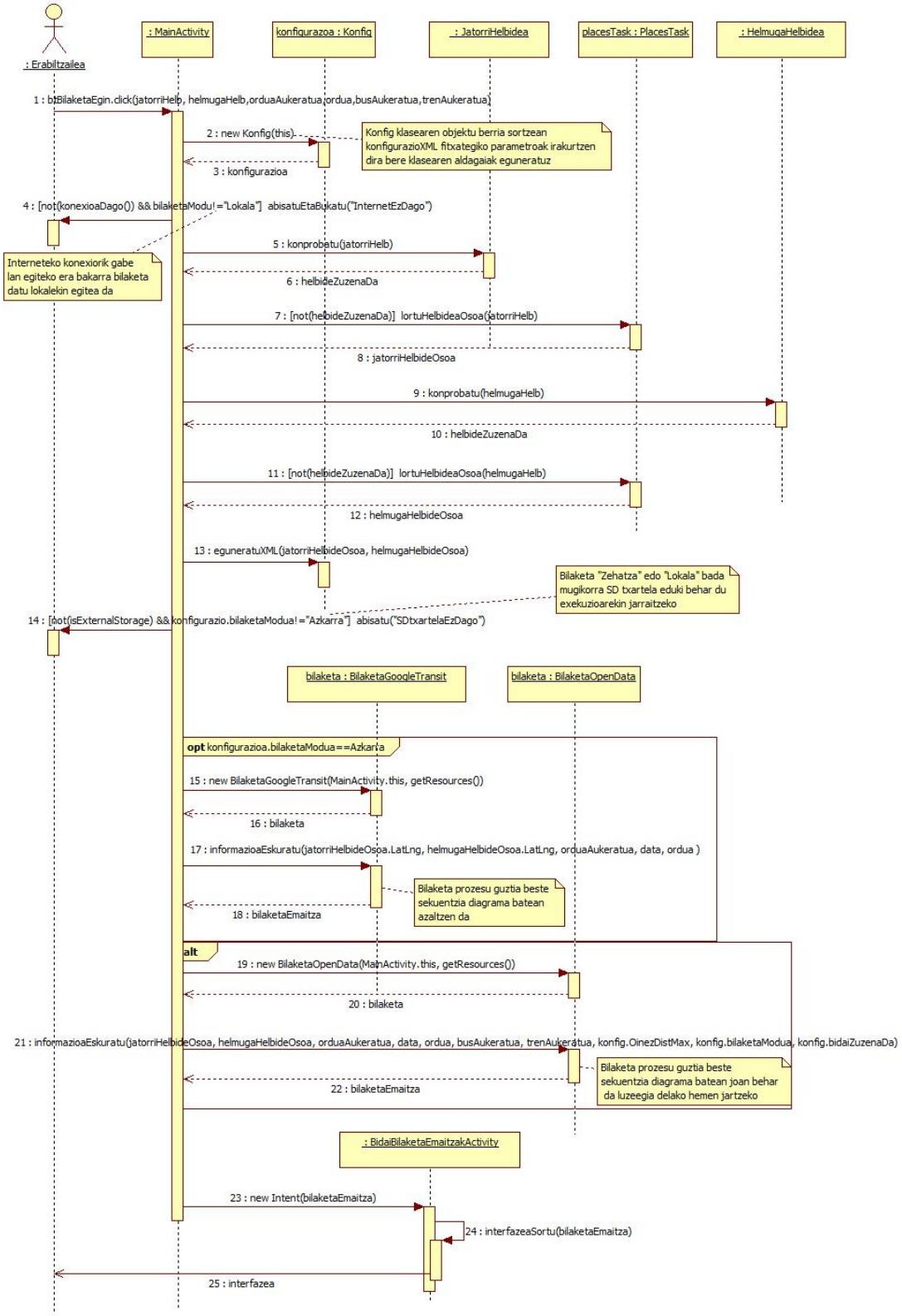
Google-ren API-a erabiltzen bada prozesua oso motza izango da, informazioa eskaera egiten da, datuak jaso, *BidaiEmaitza* klasera egokitu eta listo.

Beste kasuetan prozesua askoz luzeagoa da. Lehenik informazioa zuzena daukaten konpainien lista lortzen da, bakoitzaren geltoki lista izanda kalkulatzen da zeintzuk igarotzen dira jatorritik eta/edo helmugatik. Orduan, bidaiak egiteko konpaini bikoteak izanda, falta den informazioa deskargatzen da. Ondoren, bidaia zuzenak eta konposatuak (bi garraio publiko hartu behar direnean) kalkulatzen dira, *BidaiEmaitza* klasera egokitzen dira eta listo.

Azkenik, bidaia lista prest izanda, *Activity*-a martxan jarriko da informazio horrekin eta interfazeak kargatzean datuak pantailan erakutsiko ditu.

4.2.5.4. Bidaiak ordenatu

Bidaia bilaketa egin erabilpen kasua exekutatu eta gero emaitza zerrenda pantailan ikusiko da. Hau ordenatzeko erabiltzaileak ordenazio botoia sakatuko du eta elkarrizketa kutxa bat agertuko da, non hiru ordenazio aukera egongo diren (soilik bat aukeratu ahal izanez), “OK” daukan botoi bat eta beste bat kutxa ezeztatzeko.



4.17.irudia: Bidaia bilaketa egin sekuentzia diagrama

Adosteko botoia sakatuz gero, sistemak *BidaiEmaitza* klaseko objektu lista ordenatu berria sortuko du (gogoratu behar da bilaketaren bidaia emaitza bakoitza klase horren objektu bat dela).

Edozein ordenazioa izanda ere, jarraituko dituen urratsak berdinak izango dira, uneko listaren elementu guztietatik iteratuko da eta bakoitzagatik lista ordenatu berria ere iteratuko da. Ordenazioaren arabera objektuen parametro bat edo bestea konparatuko da dagokion posizioan sartzeko.

Elementu guztiak lista berrian gehitu direlarik, interfazearen *ListView*-a (Android objektu bat, zerrendak sortzeko) informazio berriarekin beteko da eta testu eremu batean adieraziko da zein izan den ordenazio metodoa.

4.18. irudiaren sekuentzia diagraman ordenazio konparaketa begizta barruan dago leku gutxiago okupatzeko. Hala ere, prozesuaren ideia ez da aldatzen.

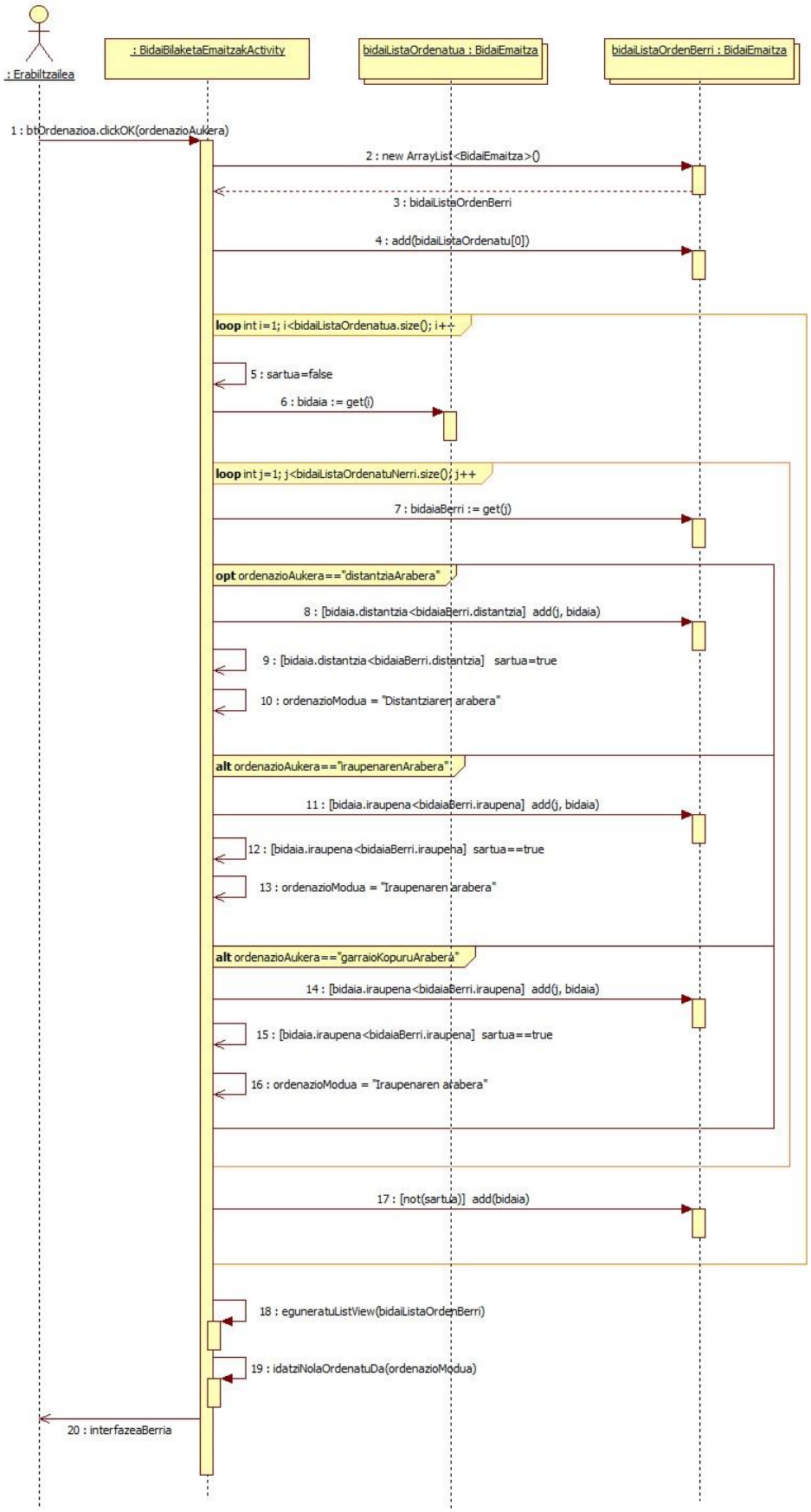
4.2.5.5. Bidaia baten informazioa ikusi

"**Bidaia bilaketa egin**" erabilpen kasua exekutatu eta gero, emaitza lista pantailan ikusiko da. Hauetako bat aukeratzean *BidaiInfoActivity* klaseari deituko zaio aukeratutako bidaia, *BidaiaEmaitza* klaseko objektu bat, parametro bezala bidaliz.

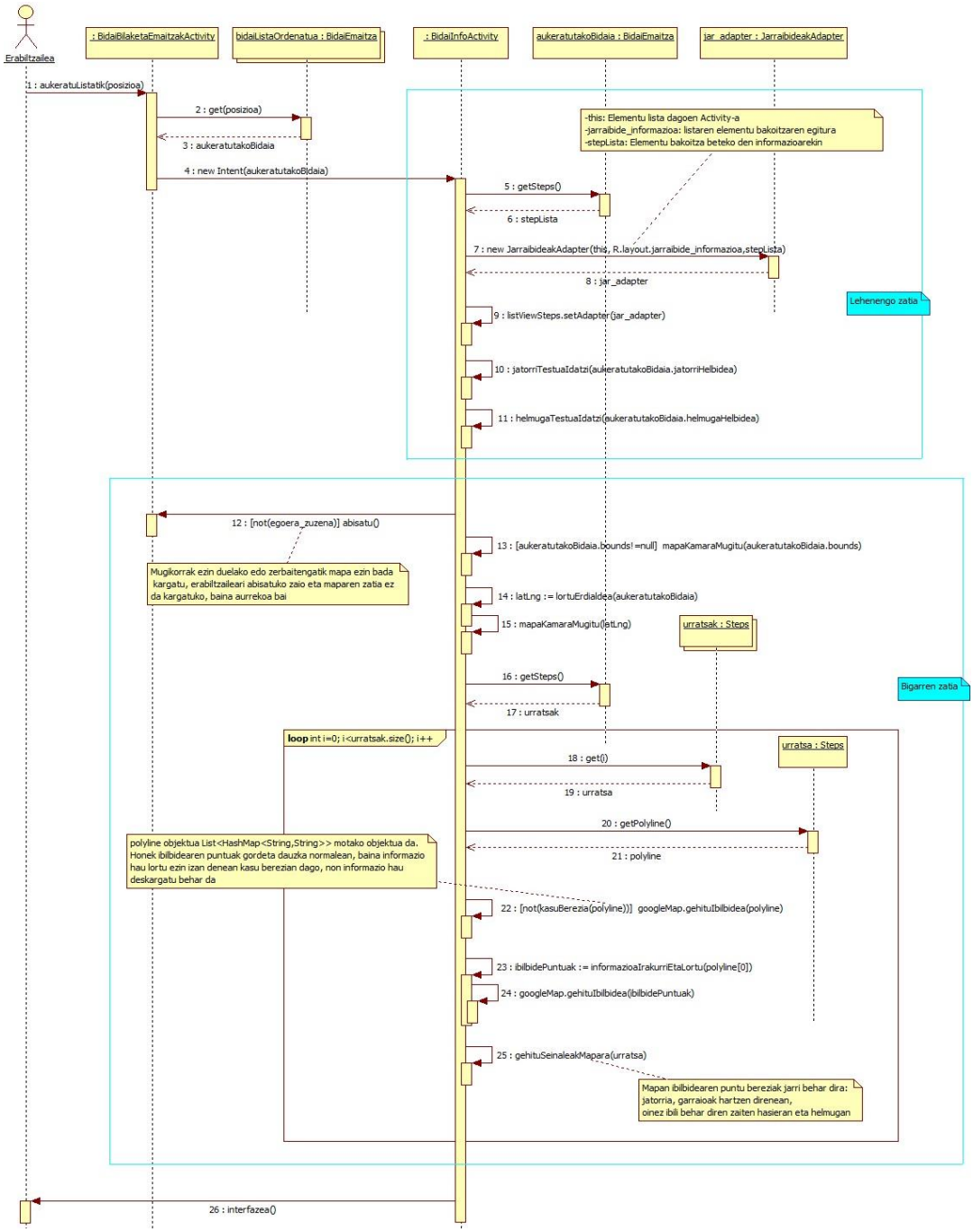
BidaiInfoActivity klaseak *Activity*-a hedatzen du, honek esan nahi du pantailan ikusiko den edukiaz arduratuko dela. Interfaze hau bi ataletan, "*TabHost*" objekturekin, erakusteko prestatuta dago, bidaiaren jarraibideak (hasieran ikusten dena) eta mapa bat ibilbidearekin eta puntu garrantzitsuak marraztuta. Hortaz, 4.19. irudian ikusten den bezala, sekuentzia diagrama bi zatitan bana daiteke.

Lehenengo zatian adierazten da nola jatorria eta helmuga testu eremu batzuetan idatziko diren eta bidaiaren urratsak lista batean. Hauetako bakoitza *Steps* klaseko objektu bat da, 4.20. irudian ikus daitekeen bezala *BidaiEmtaitza*-ren egitura oso antzekoa dauka. Urrats bakoitzarekin beste zerrenda berria sortuko da, oinezko zatia kasuan jarraibideak adieraziz eta garraioko kasuan datu garrantzitsuenak erakutsiz (Irteera eta ailegatze geltokiak, denbora estimazioa, geltoki kopurua …) .

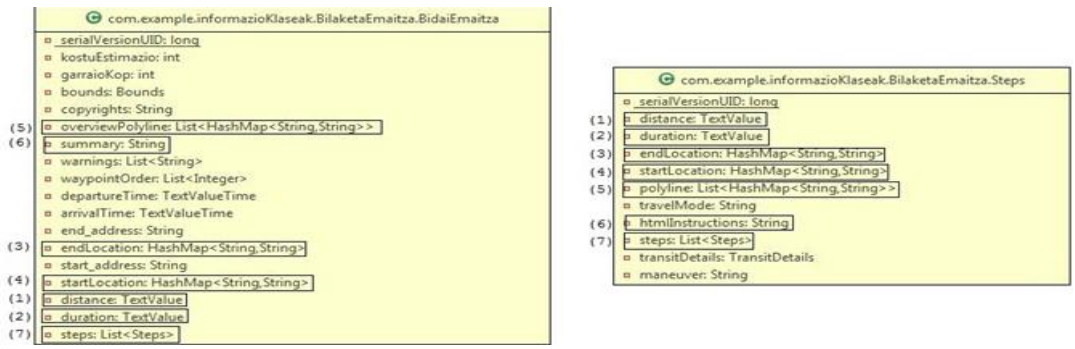
Bigarrenean adierazten da nola beteko den mapa puntuz eta lerroz. Lehenik mapa ibilbidearen erdialdeko puntura mugitzen da, hau dagoeneko kalkulatuta dator urratsaren *Bounds* objektuari esker, baina batzuetan datu hori ez dago eta eskuz lortu behar da. Ondoren, sistema urratsez-urrats joango da ibilbidea marrazten eta seinaleztatzen. Hau ere bi eratan egin daiteke, automatikoki informazioa eskura dagoelako edo *Shapes* fitxategia irakurriz eta lortuz. Bigarren aukerak ez luke inoiz huts egin behar, GTFS egitura dela eta informazio hau eskura egon beharko loke. Zoritxarrez ez da horrela beti gertatzen, konpainia batzuk informazio guztia ez dute zuzen eta ez da lortzen ibilbiderik, arazo hau hobeto azaltzen da "D Eranskina: Garraio publiko konpainien webguneen azterketa" atalean.



4.18. irudia: Bidaia emaitzak ordenatzeko sekuentzia diagrama



4.19. irudia: Bidaia baten informazioa ikusteko sekuentzia diagrama



4.20. irudia: *BidaiaEmaitza* eta *Steps* klaseen antzekotasunak

4.3. Inplementazioa

Atal honetan kode aplikazioaren zati garrantzitsuenak azalduko dira, nola funtzionatzen duten, izandako arazoak, erabilitako soluzioak, etab.

4.3.1 Bilaketa prestaketa

Erabiltzaileak bilatzeko botoia sakatzen duenean, sistemak sarrerako informazioa konprobatzen eta prestatzen du lehenik eta behin. Behin eremu guztiak beteta daudela ziurtatuta jatorria eta helmuga konprobatzen ditu, hauek bereziak dira, bi eratan definitu daitezkeelako.

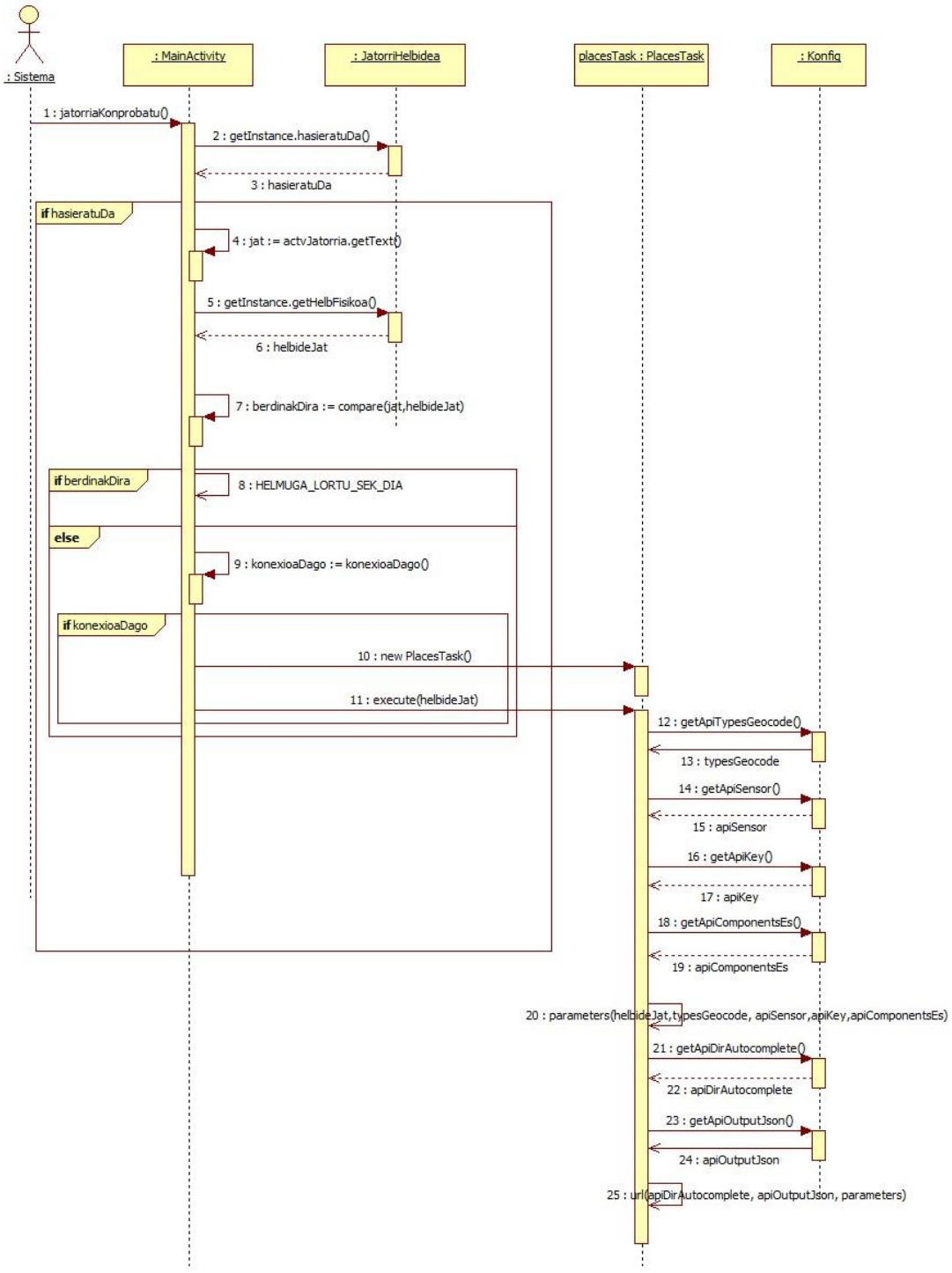
Jatorria edo helmuga maparen bidez definitu bada, testu eremuan formatu egokia duen helbide bat agertuko da (suposatuz Interneteko konexioa dagoela edo zehaztutako koordenada cache memorian dagoela), bestela erabiltzaileak idatzitakoa egongo da.

Lehenengo jatorria begiratzen da, sistemak *JatorriHelbidea* klasea hasieratu dela ziurtatzen du, ondoren, testu eremuen edukia klasearen edukiarekin konparatzen du, berdina bada esan nahi du informazio hori dagoeneko eskuratu dela mapa bidez eta helmugarekin jarraituko da. Kontrako kasuan, helbidearen informazioa bilatu behar da, baina hori ez da hain sinplea, helbide batetik bere koordenadak lortzea erraza den arren, lehenengo ziurtatu behar da idatzitakoa guk nahi duguna dela. Prozesu hau ulergarriagoa izateko 4.21. irudiko sekuentzia diagrama sortu da. Diagramak diseinu atalean egon beharko luke, baina erabilpen kasu oso bat irudikatzen ez duenez, zati txiki bat baizik, hemen txertatzea erabaki da. Prozesu osoaren kodea “F Eranskina: Inplementazioaren kode zatiak“ eranskinean aurkitu daiteke.

Horretarako sistema Google-en auto-betetze API-az baliatzea da, helbide zati bat bidaliz, hori duten posibilitate guztiak itzultzen ditu. Zerbitzu hau testu eremuetan idazten den heinean idatzitakoaren helbide posibleak erakusteko prestatua dago printzipioz.

Ondoren, erabiltzaileak listako elementu bat aukeratuko du. Hau edukita Google-en beste API bat erabiliz helbidea koordenada batean bihurtuko da.

Jatorriarekin bukatu eta ondoren prozesu bera jarraitzen da helmugarekin. Behin informazio guztia izanda bilaketa egingo den metodoari deitzen zaio.



4.21. irudia: Jatorria lortzeko sekuentzia diagrama

4.3.2. Bilaketa prozesuaren egitura

Lehenik zehaztu behar da hemen Open Data eta Dbus zerbitzariekin (ondorioz bilaketa lokalarekin) egiten den bilaketari buruz hitz egingo dela. Google-eko zerbitzuarekin jasotako informazioa soilik egokitu behar denez ("4.3.4. Bidaien informazio egokitzapena" atalean azaltzen da) ez da hemen kontuan hartzen.

Bilaketa prozesuak bi atal nagusi ditu: datuen eskuraketa eta datuen irakurketa. Hauek sakonki azalduko dira atal honetan. "4.3.3. Bilaketa algoritmoa" atalak bigarrenenaren azalpenak osatzen ditu.

4.3.2.1. Datuen eskuraketa

Datuak eskuratu ahal izateko lehenik jakin behar da zein konpainiekin lan egingo den, horretarako hurrengo urratsak jarraitzen dira:

1. Aurretik aukeratu den garraio moten arabera konpainia zerrenda lortu. Hau egiteko memorian hiru konpainia lista daude, autobusekoak bakarrik dutena, trenekoak bakarrik dutena eta guztiak dituena. Metodo hau erabiltzen da ez dagoelako era eraginkor bat jakiteko konpainia bat autobusez edo trenez osatuta dagoen (ez dago biak aldi berean dituenik).
2. Behin hasierako konpainia zerrenda edukita, bakoitzaren geltoki lista fitxategia deskargatuko da. 4.1. kode-zatian, 13-17 lerroetan ikusten den bezala, deskarga hasi baino lehen dokumentu hori dagoeneko ez dela existitzen ziurtatzen da. Egonez gero bere aldaketa data konparatzen da zerbitzarian dagoenarekin eta hau deskargatuko da berriagoa bada.

```
1.  public boolean UnzipOneFromFTP(String filePath, String zipFile, File
    outputDest) throws IOException {
2.    ZipInputStream zis = null;
3.    File outputFile = new File(outputDest, zipFile);
4.    try {
5.      zis = new ZipInputStream(mFTPClient.retrieveFileStream(filePath));
6.      ZipEntry ze;
7.
8.      while ((ze = zis.getNextEntry()) != null) {
9.        String filename = ze.getName();
10.
11.       if (filename.compareTo(zipFile) == 0) {
12.
13.         if (outputFile.exists() && outputFile.lastModified() >= ze.getTime()){
14.             zis.close();
15.             mFTPClient.completePendingCommand();
16.             return true;
17.         }
18.
19.         if (ze.isDirectory()) {
20.           outputFile.mkdirs();
21.         } else {
22.           byte buffer[] = new byte[2048];
23.           FileOutputStream fileOutputStream = new FileOutputStream(outputFile);
24.           BufferedOutputStream bufferedOutputStream = new
    BufferedOutputStream(fileOutputStream, 2048);
25.           int count;
26.
27.           while ((count = zis.read(buffer, 0, 2048)) != -1) {
```

```
28.             bufferedOutputStream.write(buffer, 0, count);
29.           }
30.
31.           bufferedOutputStream.flush();
32.           bufferedOutputStream.close();
33.         }
34.       }
35.
36.     }
37.     zis.close();
38.     mFTPClient.completePendingCommand();
39.     return true;
40.   } catch (Exception e) {
41.     e.printStackTrace();
42.     if(zis!=null){  zis.close();  }
43.     mFTPClient.completePendingCommand();
44.   }
45.
46.   return false;
47. }
```

4.1. kode-zatia: Fitxategiak zerbitzaritik deskonprimatzeko eta deskargatzeko kodea

3. Gerta liteke zerbitzaria erorita egotea edo fitxategiren bat zuzen ez egotea. Honelako arazoa duten konpainiak lista berezi batean gordetzen dira, beraiekin ez dela ezer gehiago deskargatu behar eta datu lokalekin lan egingo dela gogoratzeko.
4. Geltokien informazioa duten fitxategiak edukita, hauek irakurriko dira jatorritik eta helmugatik igarotzen diren konpainiak lortzeko. Atal hau oso garrantzitsua da, aurrerago konpainien informazio osoa deskargatu behar denez, ez da komenigarria erabiliko ez diren datuak deskargatzea, denbora galera baita.
5. Ondoren, bidaiak egiteko bikoteak lortu behar dira, hauek konpainia desberdinez edo berdinez osatuta egon daitezke. Konpainia bat jatorritik eta helmugatik igarotzen bada, orduan, ez da logikoa beste konpainiarekin ontzi aldaketa egiten saiatzea. Hauek desberdinak direnean konbinazio posible guztiak lortzeko bikoteak sortzen dira, hau da, konpainia bat beste hiru desberdinekin konbinatu daiteke adibidez.
6. Hurrengo urratsa Dbus datu iturria sartzea da. Bikoteen konpainiaren bat Dbus bada, orduan, ez da Open Datako informazioa hartuko baizik eta Dbus-ekoa. Ez hori bakarrik jatorria eta helmuga Donosti-Donosti bada, orduan, kasu berezian sartuko gara, hau “4.3.6.2. Donosti-Donosti bidaiak” atalean azaltzen da.
7. Azkenik, bilaketarako konpainiak zeintzuk izango diren jakinda, falta den informazioa deskargatuko da: ibilbideak, lineak, mapan marrazteko puntuak, agentzien informazioa eta geltokien informazio gehigarria. Ezer deskargatu baino lehen lokalean informazioa daukatenak zein diren begiratzen da (3. urratsean azaldutakoa).

```
//konpZuzen aldagaiak konpainia zerrenda du bere barruan
for(int i=0;i<konpZuzen.size();i++){
   try{
      //Geltoki listak berriro erabiliko direnez aurrerago,  geltokiak aldagia globalean gordeko dira
      geltokiak = superReader.readStops(aplikazioDir.getAbsolutePath() + "/" +
                                 konpZuzen.get(i) + "/" + Konfig.getOpenDataStops());

      Boolean jatAurkitua=false; // Geltoki guztiak ez analizatzeko aldagaia
      Boolean helAurkitua=false; // Geltoki guztiak ez analizatzeko aldagaia.

      float[] distantziaEmaitzak = new float[1];
```

```
        for(String keyStop: geltokiak.keySet()){
            //Behin koinzidentzia aurkitua ez da berriro begiratu beharko
            if(jatAurkitua==false){
                unekoLat = Double.parseDouble(geltokiak.get(keyStop).getStop_lat());
                unekoLng = Double.parseDouble(geltokiak.get(keyStop).getStop_lon());
                Location.distanceBetween(jatorriKoord.latitude, jatorriKoord.longitude,
                                         unekoLat, unekoLng, distantziaEmaitzak);
                if(distantziaEmaitzak[0] < ibilDistMax){
                    jatorriKonpainiak.add(konpZuzen.get(i));
                    jatAurkitua=true;
                    erabilitzekoGeltokiak.put(konpZuzen.get(i), geltokiak);
                }
            }

            if(helAurkitua==false){
                unekoLat = Double.parseDouble(geltokiak.get(keyStop).getStop_lat());
                unekoLng = Double.parseDouble(geltokiak.get(keyStop).getStop_lon());
                Location.distanceBetween(helmugaKoord.latitude, helmugaKoord.longitude,
                                         unekoLat, unekoLng, distantziaEmaitzak);
                if(distantziaEmaitzak[0] < ibilDistMax){
                    helmugaKonpainiak.add(konpZuzen.get(i));
                    helAurkitua=true;
                    erabilitzekoGeltokiak.put(konpZuzen.get(i), geltokiak);
                }
            }

            if(jatAurkitua && helAurkitua){
                break; //Dagoeneko jatorria eta helmuga aurkitu badira ez jarraitu
            }
        }//END->geltokiak.keySet
    } catch(Exception e){
        //Stops.txt fitxategiak errororen bat baldin badu konpainia hori saltatu
        e.printStackTrace();
    }
}
```

4.2. kode-zatia: Gertueneko geltokiak lortzeko kodea

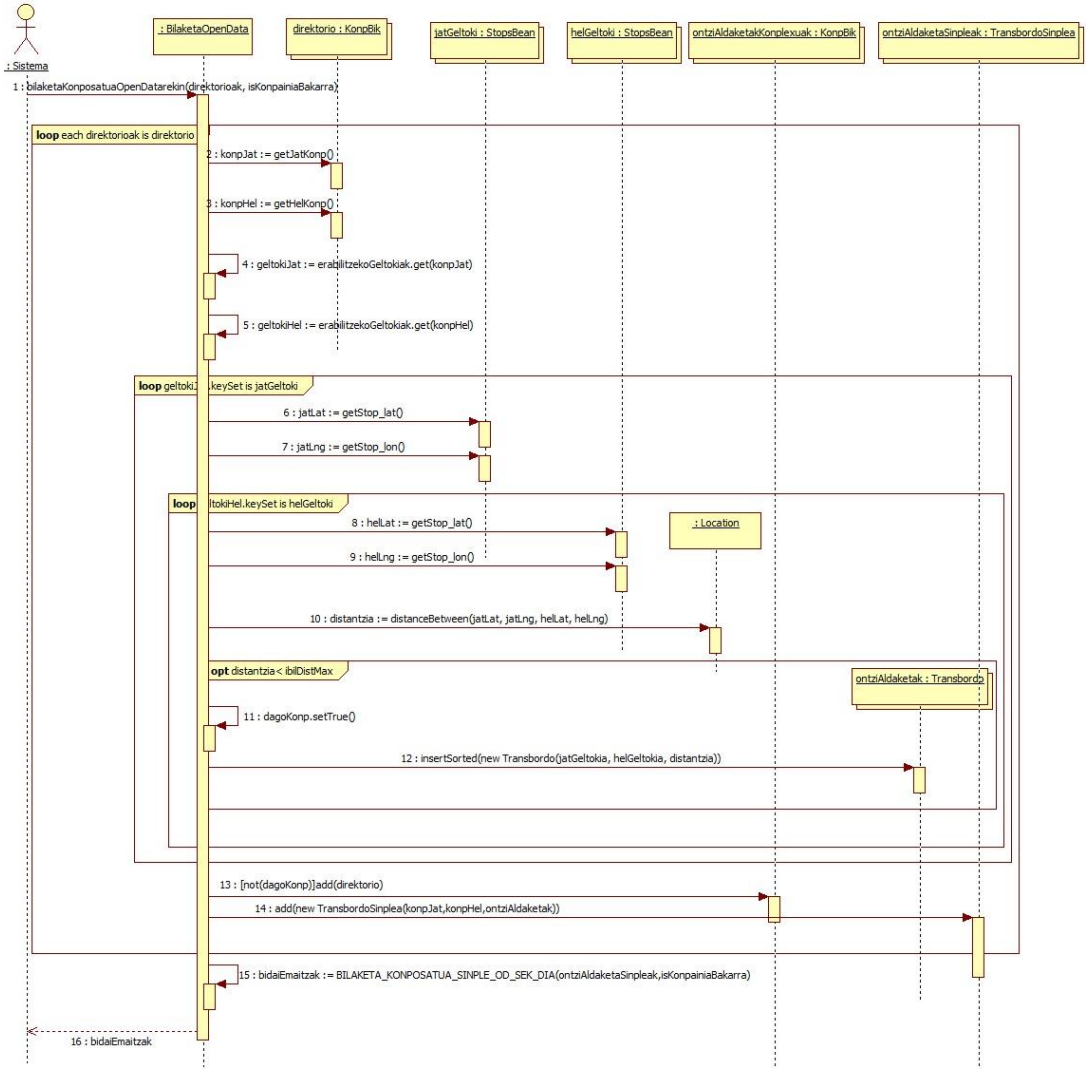
4.3.2.2. Datuen irakurketa

Bidaiak bilatzeko erabiliko diren konpainiak eta hori egiteko beharrezkoak diren fitxategiak deskargatuta izanda, hurrengo urratsa bilaketa gauzatzea da. Bi bidaia mota bilatuko dira, konpainia bakar batekin egingo direnak eta bi konpainiekin egingo direnak.

Konpainia bakarrarekin

Lehenengo urratsa konpainiaren fitxategiak irakurtzea eta aldagaietan gordetzea da. Orduan, informazio osoa prest dagoenean, bilaketa gauzatzen da. "4.3.3. Bilaketaren algoritmoa" atalean azaltzen den bezala, sistema prestatuta dago konpainia batekin jatorri eta helmugaren arteko bidaiak bilatzeko.

Bidaiak osatzen duten informazioa edukita hurrengo pausua hau erakusteko prestatzea da. Hau egiteko lortu den emaitza bakoitza hartzen da eta *BidaiEmaitza* klaseko objektu bat osatzen da. 4.4. irudiak erakusten duen bezala, klase horren egitura prestatuta dago informazioa pantailan erakutsi ahal izateko.



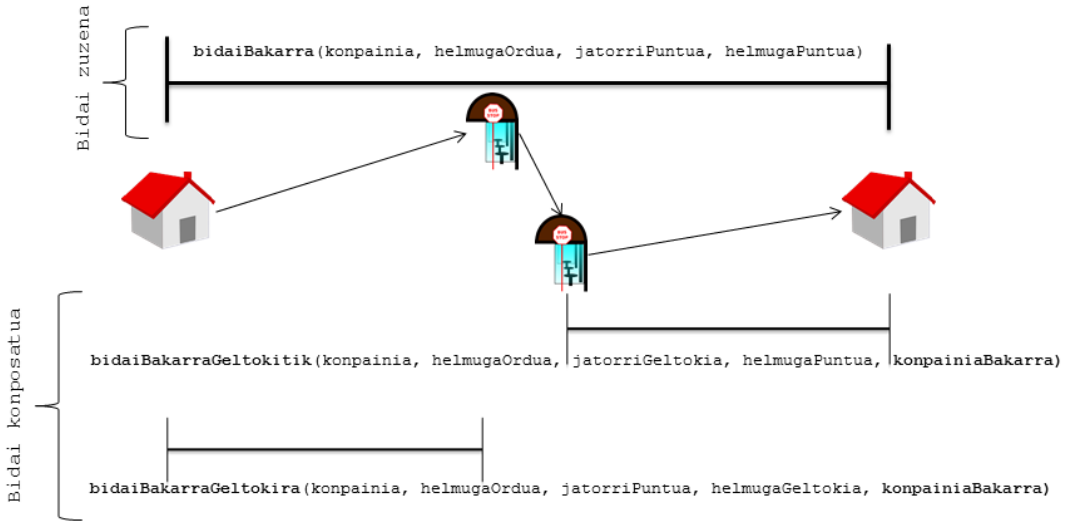
4.22. irudia: Ontzi aldaketak lortzeko sekuentzia diagrama

Puntu honetara ailegatuta bidaia zuzenak lortu dira, hortaz hurrengoa bidaia konposatuak lortzea da. Hau egiteko era oso sinplea da, bi konpainiekin bilaketa egingo duen funtzioari deitzen zaio, baina bi konpainia desberdin parametro bezala pasatu ordez bakarra pasatuko zaio. Honek arazo bat du, ezer kontrolatzen ez bada lortuko diren emaitzatan dagoeneko erabilitako lineak agertuko dira. Arazoa ekiditeko bidaia zuzen bat lortzen den bakoitzean honen identifikatzailea gordetzen da lista batean, horrela bidaia konposatuak lortzerakoan filtrazio bat egiten da.

Azkenik *BidaiEmaitza* objektuak sortuko dira lortutako emaitzekin eta bidaia zuzenak dituen zerrendari gehituko zaizkio.

Bi konpainiekin

Egin behar den lehenengo gauza ontzi aldaketak bilatzea da. Jatorritik eta helmugatik igarotzen diren konpainia bikoteak ezagutzeak ez du ziurtatzen beraien artean bidaia oso bat egitea posible denik, horregatik ontzi aldaketa posibleak bilatu behar dira.



4.23. irudia: Bidaia zuzena eta konposatuaren arteko desberdintasuna

4.22. irudiko sekuentzia diagraman ikus daiteke aplikazioak jarraitzen duen prozesua ontzi aldaketak bilatzeko (diagrama honek beste atalean egon beharko luke, baina azalpen hau ulergarriagoa egiteko hemen jarri da. Kode osoa "F Eranskina: Inplementazioaren kode zatiak" eranskinean aurkitu daiteke). Lehenik jatorri eta helmuga konpainien geltokiak lortzen dira. Bi hauen konbinazio guztien arteko distantziak kalkulatuko dira eta oinezko distantzia maximoa baino gutxiago bada, orduan ontzi aldaketak gordetzen dituen zerrenda batean sartzen dira (jatorriko geltokia, helmugakoa eta distantzia). Xehetasun garrantzitsu bat da listan sartzen diren heinean distantziaren arabera ordenatuko direla, aurrerago elementu errepikatuak errazago ezabatzeko.

Bidaia zuzenak eta konposatuak ez dira desberdinegiak, lehenengoa hiru urratsez osatuta dago: jatorritik geltoki batera oinez joan, geltoki horretatik beste batera garraio publiko batean joan eta azkenik oinez berriro mugitu helmugaraino. Konposatua ordea bost urratsez: jatorritik oinez joan behar da lehenengo geltokiraino, garraioan igo, jaitsi eta beste konpainiaren geltokiraino ibili, berriro garraio publikoan bidaiatu, jaitsi eta oinez joan helmugaraino. Ondorioz, bidaia konposatuak bi bidaia zuzen bezala kontsideratu daitezke, desberdintasun batekin, hauen jatorri edo helmuga geltokia finkoa da.

Ontzi aldaketa bakoitzaren bidaiak bilatzeko, lehenengo helmugara doazen bidaiak lortzen dira eta horren arabera jatorritik igarotzen direnak. Hau egiteko bidaia bakarra bilatzeko funtzioaren bertsio azkarragoak erabiliko dira. 4.23. irudiak erakusten duen bezala, bidaia konposatuak aurkitzeko funtzioa bakarrak bilatzeko erabiltzen dena da desberdintasun batekin, parametro bat koordenada izatetik geltoki bat izatera pasatzen da.

Azkenik *BidaiEmaitza* klaseko objektuak sortzen dira lortutako bidaiekin. Zati hau konpainia bakarrarekin egingo balitz ia berdina litzateke, desberdintasun bakarra da urrats gehiago kontua hartu behar direla. Ontzi aldaketan, geltoki batetik bestera joateko oinezko ibilbidea lortu behar da, horretarako Google Directions API-a erabiliko da.

4.3.3. Bilaketaren algoritmoa

Konpainiaren datu iturriak irakurri eta informazioa aldagaietan gorde ondoren, sistema prest dago bilaketa gauzatzeko. 4.3. kode-zatian bilaketaren prozesuaren kodea dago eta iruzkinek urratsez urrats azaltzen dute egiten dena.

```
//jatorritik gertu dagoen geltoki bakoitzari bere helmuga posibleak bilatuko dizkiot
for(int i=0;i<jatorritikGeltGertu.size();i++){

   //lehenik lortu geltokiari dagokion geltoki denbora (stop_Times) guztiak  (lehen lortu ditudanak)
   HashMap<String, Stop_TimesBean> jatGeltDenbk =
                                          new HashMap<String,Stop_TimesBean>();
   jatGeltDenbk = geltokiIdGertuen.get(jatorritikGeltGertu.get(i).getGeltokiIid());

   //ondoren geltoki denbora guztietatik pasatuko naiz begiratzeko helmugan badagoen ibilbide bera
     (TRIP_ID bera duena) egiten duen geltokirik
   for(String keyJatGD: jatGeltDenbk.keySet()){

      //lortzen dut Stop_TimesBean objektu bat (TRIP_ID, ARRIVAL_TIME, DEPARTURE_TIME....)
      Stop_TimesBean jatGD = jatGeltDenbk.get(keyJatGD);

      //STOP_TIMES-eko elementuari dagokion ibilbidearen identifikatzailea, ROUTE_ID,
       (beti 1 egongo da)  lortu
      routeID = bidaiak.get(jatGD.getTrip_id()).getRoute_id();

      //orain helmugako geltoki guztietatik pasatuko naiz ibilbide koinzidentziak bilatzeko
      for(int j=0;j<helmugatikGeltGertu.size();j++){

         //lehenik geltokiari dagokion geltoki denbora (stop_Times) guztiak lortu
         HashMap<String, Stop_TimesBean> helGeltDenbk =
                                           new HashMap<String,Stop_TimesBean>();
         helGeltDenbk = geltokiIdGertuen.get(
                                    helmugatikGeltGertu.get(j).getGeltokiIid());

         //lortzen dut Stop_TimesBean objektu bat hau existitzen baldin bada (ez badago, esan nahi
          du ez dela ibilbide zuzena)
         Stop_TimesBean helGD = null;
         helGD = helGeltDenbk.get(keyJatGD);

         //jatorria beti egongo da, baina helmuga dagoen kasuetarako bere informazioa gordeko dut
         if(helGD!=null){

            //noski, behar ditugun bidaiak jatorritik helmugarako bidaiak dira, hortaz
              STOP_SEQUENCE bidez hau kontrolatzen da
            if(Integer.parseInt(jatGD.getStop_sequence())<
                              Integer.parseInt(helGD.getStop_sequence())){

               float unekoDistOsoa = jatorritikGeltGertu.get(i).getDistantzia() +
                                     helmugatikGeltGertu.get(j).getDistantzia();

               //hau esan nahi du oraindik ibilbide horretako bidairik ez dela gorde
               if(routeGeltkGertuen.get(routeID)==null){

                  bidaiBikoteak.put(routeID,
                                    new GeltDenbBikote(ibilbideak.get(routeID),
                                    bidaiak.get(jatGD.getTrip_id()), jatGD,
                                    helGD, jatorritikGeltGertu.get(i).getGeltokia(),
                                    helmugatikGeltGertu.get(j).getGeltokia()));
                  routeGeltkGertuen.put(routeID, unekoDistOsoa);
               }
               //uneko bikote geltoki distantzia orain arte gordeta dagoen distantzia baino
                 txikiago bada orduan bai eguneratu dela balioa
               else if(unekoDistOsoa < routeGeltkGertuen.get(routeID)){
                  bidaiBikoteak.put(routeID,
                                    new GeltDenbBikote(ibilbideak.get(routeID),
                                    bidaiak.get(jatGD.getTrip_id()), jatGD,
```

```
                                    helGD, jatorritikGeltGertu.get(i).getGeltokia(),
                                    helmugatikGeltGertu.get(j).getGeltokia()));
                    routeGeltkGertuen.put(routeID, unekoDistOsoa);
                }
                eguneratuErabilitakoIbilbLista(konpainia,routeID);
            }
        }
    }
}
```

4.3. kode-zatia: Bidaiak lortzen dituen kode zatia

Bidaia bilaketa berez dena, lehen azaldu den bezala, konpainia bakarrarekin egiten da beti. Bilaketa egiten duen oinarrizko kodean jatorri geltoki zerrenda batetik helmugako beste batera dauden bidaia posible guztiak bilatzen dira, 4.3. kode-zatiak funtzio nagusi hori irudikatzen du. Kode horrek ez du balio bilaketa mota guztientzat, ondorioz honetaz aparte beste hainbat bertsio daude:

- Bilaketa normala orduaren araberakoa.
- Geltoki finko batetik hasi behar diren bidaien bilaketa.
- Geltoki finko batetik hasi behar diren bidaien bilaketa orduaren araberakoa.
- Geltoki finko batean bukatu behar duten bidaia bilaketa.
- Geltoki finko batean bukatu behar duten bidaia bilaketa orduaren araberakoa.

Bost bertsio hauek oso antzekoak dira 4.3. kode-zatikoarekin konparatuz, adibidez, bilaketa orduaren arabera izatean geltoki denbora bat hartzean ziurtatu behar da finkatutako ordua baino lehenago dela. Emaitza bat eguneratzen denean azkarrena dela konprobatzeaz gain adierazitako ordutik hurbilena dela ziurtatu behar da.

Ibilbideak aurkitzen direnetik formatu egokian dauden arte hiru aldagaietatik pasa behar dute, 4.4. kode-zatian agertzen direnetatik.

Ibilbideak lortzen diren heinean *bidaiBikoteak* listan txertatzen dira. Zerrenda hau HashMap motako objektu bat da eta indize bezala linearen identifikatzailea erabiltzen du, ez baitira nahi emaitza errepikaturik.

Behin ibilbide guztiak lortuta *OpenDataEmaitza* klaseko objektu berria sortuko da, *konpainiakBidaiekin*. Honek hiru parametro izango ditu: konpainiaren izena, agentzia horren *Agency* fitxategian agertzen diren izenak (gertatu daiteke konpainia baten bi izen izatea edo hainbat konpainien batura izatea) eta azkenik *bidaiBikoteak* aldagaia.

Bukatzeko aldagai hori *bidaiEmaitzak* ArrayList-ean sartzen da.

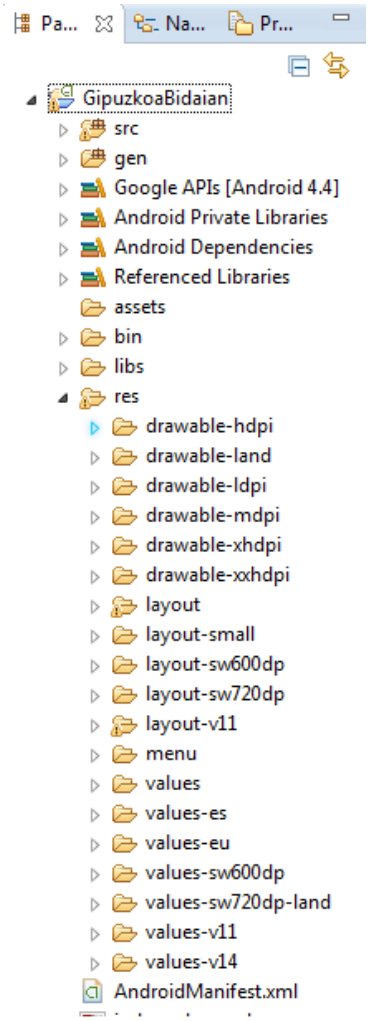
```
//GARRANTZITSUA: Hau prestatuta dago edozein ordutarako bidaiak bilatzeko, hau da, HashMap denez
ROUTE_ID bati dagokion edozein bidaia (Trip) gordeko dio, azken batean ez du axola irteera ordua
HashMap <String, GeltDenbBikote> bidaiBikoteak = new
                                                HashMap <String, GeltDenbBikote>();

//garrio konpainiak eta bidaiak (bidaia zuzeneko kasuetan elementu bakarrekoa, bidai konposatuko kasueta
hartuko diren garraio desperdin kopuruaren arabera)
List <OpenDataEmaitza> konpainiakBidaiekin = new ArrayList <OpenDataEmaitza>();

//Hau da bukaerako emaitza, hemen gordeko da bidai zuzenak (garraio konpainia eta bidai posiblea) eta bidai
konposatuak (garraio konpainiak eta bakoitzak egin ditzakeen bidai posibleak)
List<List <OpenDataEmaitza>> bidaiEmaitzak =
                                            new ArrayList <List <OpenDataEmaitza>>();
```

4.4. kode-zatia: Emaitzak gordetzen dituzten aldagaien erazagupenak

4.3.4. Bidaien informazio egokitzapena

Bidaia guztiak behin lortuta informazioa egokitu behar da ondoren erakutsi ahal izateko, hau egiten da 4.4. kode-zatiko *bidaiEmaitzak* aldagaia irakurtzen eta *BidaiEmaitza* klaseko objektu batean sartzen. *BidaiEmaitza* klasearen aldagaien azalpena hurrengoa da:

- kostuEstimazioa: bidaia osoaren kostu estimazioa. Aldagai hau ez da oraingoz erabiltzen, hasieran pentsatuta neukan bidaiaren kostua kalkulatzea, baina azkenean ezinezkoa izan da lortzea.
- garraioKop: bidaia osoa egiteko zenbat garraioetan igo behar den kopurua. *Steps* klasearen *getTravelMode* parametroak nola mugituko den adierazten du, hortaz steps zerrendan "*WALKING*" ez diren elementuak zenbatuz lortzen da.
- bounds: bidaia osoa okupatzen den eremuaren mapa. Parametro hau soilik lortzen da Google Directions API-a erabiltzean, bestelakoetan hutsik dago.
- copyrigths: datuen iturria adierazten du. Parametro hau soilik lortzen da Google Directions API-a erabiltzean, bestelakoan hutsik egongo da.
- overviewPolyline: Jatorritik helmugarainoko ibilbidea osatzen duen koordenada zerrenda.
- summary: bidaiaren deskribapen motza. Parametro hau beti hutsik egongo da.
- warnings: oharpen zerrenda. Parametro hau beti hutsik egongo da.
- waypointOrder: mugarri zerrenda. Parametro hau beti hutsik egongo da.
- departureTime: irteera ordua. Oinezko lehen tartea lortu ondoren, Google-eko API-a erabili denez, bere irteera ordua eta bidaia osoarena berdinak dira.
- arrivalTime: ailegatze ordua. Aurrekoa bezala baina oinezko azken tartearekin.
- end_address: helmuga helbidea. Bi eratan lortu daiteke, hasieran erabiltzaileak adierazi duena edo azken oinezko tartearen helmuga helbidea. Tarte hau Google API-a erabiliz lortzen da, azken geltokitik helmugarainoko bilaketan.
- start_address: jatorri helbidea. Bi eratan lor daiteke, hasieran erabiltzaileak adierazi duena edo lehenengo oinezko tartearen irteera helbidea. Tarte hau Google API-a erabiliz lortzen da, jatorritik lehenengo geltokirainoko bilaketan.
- startLocation: jatorri koordenada. Aurrekoa bezala baina helbidea ordez koordenadekin.
- endLocation: helmuga koordenada. "startLocation" bezala.
- distance: jatorritik helmugarainoko distantzia. Parametro hau kalkulatu behar da oinezko distantziak (Google-ko API-arekin lortuak) eta garraiokoak batuz. Garraioko distantzia lortzen da ibilbidean zehar zeharkatzen diren geltokien arteko distantziak batuz (*Stops* fitxategian daudenak).
- duration: jatorritik helmugara ailegatzeko behar den denbora. Aurrekoaren antzekoa, baina kasu honetan garraiozkoa lortzen da ailegatze orduari irteerakoa kenduz.
- steps: oinez eta garraioz egiten diren urratsen zerrenda. Dagoeneko esan den bezala, *Steps* klasearen parametroak *BidaiEmaitza* klasearen oso antzekoak dira. Hala ere, hainbat datu gehiago dauzka, hala nola, *travelMode*, *htmlInstructions* eta beste batzuk.



4.24. irudia: GipuzkoaBidaian proiektuaren direktorioen zuhaitza

4.3.5. Diseinu direktorioen egitura

Android-rako diseinatzea ez da batere erraza, merkatuan sistema hori erabiltzen duten hainbeste gailuak daude, non denentzat berdin ikusten den interfaze bat garatzea lan gogorra den. Arazo hau konpontzeko aplikazioaren "*res*" direktorioan, diseinuari buruzko kode guztia doan lekuan, karpeten izenak egitura bat jarraitzen dute. 4.24. irudian ikus daitezke Eclipse-ko GipuzkoaBidaian proiektuaren direktorioak.

Diseinu direktorioen izenen oinarria "*layout*" da, honen ondoren jartzen denak ze gailu mota irakurriko duten identifikatzen du. Adibidez:

- **layout**: Gailu guztiak hemengo XML fitxategiak irakurriko dituzte interfazeak sortzeko.
- **layout-small**: Pantaila txikia duten gailuak "*layout*"-tik hartuko dute XML guztiak hor daudenak izan ezik.
- **layout-sw600dp**: "*sw*" eta "*dp*" artean dagoen zenbakiak adierazten du zein izan behar den pantailaren zabalera minimoa karpeta hau kontuan hartzeko. 600

zenbakiarekin adierazten da 7” edo handiagoko tabletak direktorio honen araberako interfazea sortuko dutela.

- layout-v11: Izenean “-vN”, non N zenbaki bat da, adierazten du ze Android bertsio minimoa eduki behar den karpeta hori irakurtzeko. Hau erabili oi da bertsio zaharrak eta berriak desberdintzeko. GipuzkoaBidaian aplikazioan adibidez, testu eremuak ez dira berdin ikusten bertsio zahar eta berri batean, XML fitxategiak desberdinak direlako.

4.3.6. Arazoak eta soluzioak

Atal honetan programazioarekin zerikusia duten arazoak azaltzen dira, zer gertatu den eta nola konpondu da.

4.3.6.1. Gertueneko geltokiak lortzea

Aplikazioaren garapenaren hasieran zailtasun bat aurkitu nuen, nola lortu adierazitako puntutik igarotzen diren lineak. GTFS fitxategien egitura aztertu eta gero konklusio batera iritsi nintzen, bi era daude:

- Gertueneko geltokiak lortu eta ondoren dagozkien lineak.
- Linea guztiak lortu eta gertutik igarotzen direnekin geratu.

Azkenean, lehenengo metodoarekin aurrera jarraitzea erabaki nuen.

Behin geltokien fitxategi guztiak deskargatuta daudela, *Stops.txt*, irakurtzen dira. Hauen datuen artean geltoki bakoitzaren koordenadak daude, hortaz, jatorri eta helmugako puntuak izanda geltoki bakoitza dagoen distantziara kalkula daiteke. Zorionez, Javak *Location* izeneko klasea dauka, honek automatikoki bi koordenaden arteko distantzia kalkulatzen duen funtzio bat du (bestela eskuz egin izango nuke).

Orduan beste arazo bat agertu zen, zenbateko distantzia gertu dela kontsideratu daiteke? Erabiltzaile bakoitza desberdina da, gazte bati motza iruditzen zaion distantzia agure bati luzeegia iruditu liezaioke. Hau konpontzeko erabiltzaileak distantzia maximoa aukeratu dezala erabaki zen. Ez da soluzio logikoena, baina ez dago beste erarik.

4.3.6.2. Donosti-Donosti bidaiak

Donostian bezeroak jasotzeko nork duen eskumena arazo bat da garraio publiko konpainientzat. Dbus-eko langilearekin bildu nintzenean kontatu zidan beraiek zirela Donostian zehar bidaiatzeko aginpidea zeukatenak, autobusekoa. Esan zidanaren arabera beste konpainiak hiritik igaro daitezke, bezeroak bertan uzteko eta kanpora eramateko. Arazoa da konpainia hauek hainbeste geltoki dituztela Donostian zehar, hiriko bezeroak jasotzen dituztela eta bertan uzteko. Honek diru galera handiak eragiten zietela azaldu zidan.

Hori guztia entzun ondoren promestu nion nire aplikazioak hirian bertan bidaia bat bilatu behar bazuen autobusarekin, soilik Dbus-ekin izango zela.

4.25. irudia: Bidaia informazioa bistaratu, ezkerrean mugikor batean eta eskuinean tableta batean

4.3.6.3. Tabletentzako bertsioa egin

Interfaze gehienetan ez dago desberdintasun handirik aplikazioa nola ikusten den mugikor eta tableta batean. Hala ere, badago atal bat Android-ek ematen dituen gaitasunak aprobetxatu ditzakeena bidaia baten informazioa bistaratzen denean.

Bertsio normalean *Tabhost* objektu batek (erlaitzak sortzen dituen objektuak) bi ataletan, jarraibideak eta mapa, desberdintzen du ikusten dena. Orduan, tableta bat askoz zabalagoa bada, zergatik ez erakutsi bi gauzak aldi berean? 4.25. irudian ikus daitekeen bezala hori lortu zen.

```
<LinearLayout >

  <RelativeLayout  >

       <LinearLayout >
              <TextView />
              <TextView />
        </LinearLayout>

        <ListView >
        </ListView>

        <LinearLayout >
              <TextView />
              <TextView />
        </LinearLayout>

  </RelativeLayout>

  <RelativeLayout >
       <fragment />
  </RelativeLayout>
```

```
<RelativeLayout  >
    <TabHost >
        <LinearLayout >

            <TabWidget >
            </TabWidget>

            <FrameLayout >

                <RelativeLayout >

                   <LinearLayout>
                        <TextView/>
                        <TextView />
                   </LinearLayout>

                   <ListView >
                   </ListView>

                   <LinearLayout >
                         <TextView />
                         <TextView />
                   </LinearLayout>

                 </RelativeLayout>

                 <RelativeLayout >
                          <fragment />
                 </RelativeLayout>
```

```
</LinearLayout>
```

```
            </FrameLayout>
        </LinearLayout>
    </TabHost>
</RelativeLayout>
```

4.5. kode-zatia: *bilaketa_emaitza.xml*, tabletentzako bertsioa ezkerrean eta orokorra eskuinean

Hau egiteko bi aldaketak egin behar izan ziren kodean, bat izan zen *Activity*-a kargatzen duen kode zatian, garrantzirik gabekoa, eta bestea XML-an. Aurreko atal batean azaldu dudan bezala, tabletentzako diseinu fitxategi propioak egin daitezke, hortaz egin dudan bakarra izan da mugikorrentzako berdina sortu baina *Tabhost*-a gabe.

4.5. kodean ikus daiteke nola tabletentzako bertsioak hainbat elementu gutxiago dauzkan, bertsio orokorrak elementu guztiak RelativeLayout barruan kokatzen dira eta aldi berean *TabHost*-en barruan. Ondoren, *TabWidget*-a definitzen da, erlaitzak izango litzatekeena, eta azkenik *RelativeLayout* barruan geratzen den kodea. Azken hau jartzen da bistaratzen ari den kodea dinamikoki alda daitekeelako *TabWidget*-a ukitzean.

4.4 Proba kasuak

Aplikazio bat garatzen den heinean hainbat proba egin behar dira funtzionatzen duela ziurtatzeko. Probak bi eratakoak izan behar dira: funtzionalak eta diseinukoak. Lehenengoak egiten dira konprobatzeko funtzioak dei zuzenak egiten dituztela, datu zuzenak itzultzen dutela, etab. Diseinukoak garrantzi handia dute aplikazio honetan, Android sistemarako bideratutako aplikazioa denez proba asko egin behar dira ikusteko interfazeak gailu desberdinetan badabilela.

4.4.1. Proba funtzionalak

Aplikazioaren funtzionalitate nagusia bidaien bilaketa denez, zati hori ondo funtzionatzen duela ziurtatu behar izan da aplikazioaren garapenean. Hasieran bidaia zuzenak soilik bilatzen ziren, hortaz hainbat proba egin ziren ziurtatzeko ondo bilatzen zuela. Azkenean bidaia konposatuak egitea lortu zen eta beste hainbat proba egin ziren.

Probak bi zatitan banatzen dira:

- Bilaketa bat egin, emaitza lortu eta hau Interneten egiaztatu zuzena dela. Konpaini guztiak ezagutzen ez ditudanez, eta are gutxiago konpaini bakoitzaren linea guztiak, horrela egin behar izan da.
- Behin proba batzuk eginda JUnit-a erabili zen proba horiek automatikoki egiteko. Honela kode aldaketa bat egiten zen bakoitzean ziurtatu zitekeen emaitza ez zela aldatzen.

```
package com.example.gipuzkoabidaian.test;
```

```
//import guztiak falta dira

@SuppressWarnings("unused")
public class BidaiakTest extends
ActivityInstrumentationTestCase2<MainActivity> {

	private MainActivity activity;

	public BidaiakTest9() {
		super(MainActivity.class);
	}

	protected void setUp() throws Exception {
		super.setUp();
	    activity = getActivity();
	}

	protected void tearDown() throws Exception { }

	public void testBilaketa() {
		//Funtzio honen barruan probak egiteko kodea idazten da
	}
}
```

4.6. kode-zatia: JUnit proben eredua

JUnit-eko probak egiteko 4.6. kode eredua erabili da. Hor ikus daiteke *BidaiakTest* klaseak *MainActivity*-tik hedatzen deela, azken batean bilaketen funtzioei klase horretatik deitzen zaie. Proba kasu bakoitzarentzako aldatzen den kode bakarra *testBilaketa* funtzio barruan dago.

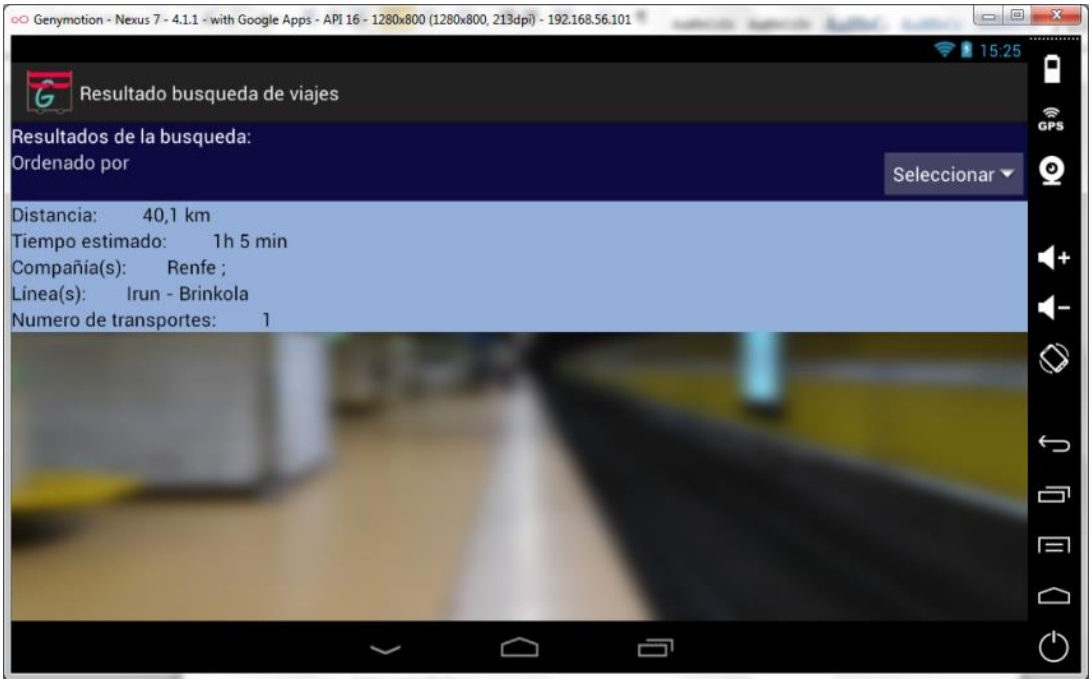
4.4.1.1. Google Directions proba

- **Deskribapena**: Google-eko web zerbitzua erabiliko da bi helbideen arteko bidaia hoberena lortzeko (soilik emaitza bat itzultzen duenez suposatzen da bidaia hoberena dela bere irizpidearen arabera).
- **Bilaketa modua**: azkarra.
- **Jatorri helbidea**: San Frantzisko Pasealekua Ibilbidea, 21, Tolosa (43.1349045,-2.0771048).
- **Helmuga helbidea**: Jose Eguino Kalea, 18, Irún (43.3376933,-1.796974).
- **Orduaren arabera**: ez.
- **Garraio moduak**: ez da kontuan hartzen.
- **Oinezko distantzia maximoa**: ez da kontuan hartzen.
- **Bidaia zuzenak soilik?**: ez da kontuan hartzen.
- **Espero den emaitza**: ezin da emaitza zehatz bat espero Google-eko bilaketan, ezinezkoa baitzait ezagutzea beraien bilaketa algoritmoa. Hala ere, jakinda soilik Renfe konpainiak geltoki bat daukala bai jatorrian eta bai helmugan, ziur aski berarekin izango da bidaia emaitza.

- **Lortutako emaitza**: 4.26. irudian ikus daiteke lortutakoa espero zena dela.
- **JUnit kodea**:

```
Resources res = activity.getResources();
LatLng origin= new LatLng(43.1349045,-2.0771048);
LatLng dest = new LatLng(43.3376933,-1.796974);
boolean cbOrduaZehaztu = true;
int urtea = 2014;
int hilabetea=4;
int eguna=9;
int ordua=15;
int minutua=24;

BilaketaGoogleTransit bilaketa = new BilaketaGoogleTransit(activity,
                                                           res);
ArrayList<BidaiEmaitza> bilaketaEmaitza =  new
ArrayList<BidaiEmaitza>();
bilaketaEmaitza = bilaketa.informazioaEskuratu(origin, dest,
                  cbOrduaZehaztu, urtea, hilabetea, eguna, ordua,
                  minutua);

assertEquals("Emaitza kopurua bakarrekoa izan bearko luke",
              1, bilaketaEmaitza.size());
assertEquals("Distantzia 52km izan beharko luke",
             "40.1 km",
bilaketaEmaitza.get(0).getDistance().getText());
assertEquals("Bidaiaren iraupena ordu ta gutxi izan behar da",
             "1 hour 5 mins",
             bilaketaEmaitza.get(0).getDuration().getText());
```

4.7. kode-zatia: Google Directions proba



4.26. irudia: Google Directions-rekin egindako bilaketaren emaitza

4.4.1.2. Bilaketa normala

- **Deskribapena**: kasu honetan, bilaketa arrunt bat egingo da, Open Data eta Dbus zerbitzarien informazioarekin. Honek balioko du batez ere Google-eko emaitzarekin desberdintasunak ikusteko.
- **Bilaketa modua**: bilaketa zehatza (Open Data).
- **Jatorria helbidea**: San Frantzisko Pasealekua Ibilbidea, Tolosa (43.1349045,-2.0771048).
- **Helmuga helbidea**: Jose Eguino Kalea, Irun (43.3376933,-1.796974).
- **Orduaren arabera?**: ez.
- **Garraio moduak**: autobusa eta trena.
- **Oinezko distantzia maximoa**: 500 metro.
- **Bidaia zuzenak soilik?**: ez.
- **Espero den emaitza**: Soilik bi konpainia pasatzen dira jatorritik, Renfe eta TSST. Lehenengoak badauka linea bat Irunera, besteak ordea ez, soilik bi lineak daukate geltokia hor , “Tolosa-Donostia” eta “Asteasu-Donostia”. Azken linea Tolosatik Asteasura doa, ondorioz, ontzi aldaketa baten bidez Irunera ailegatzea ezinezkoa du. Helmugako helbidetik hainbat konpainia desberdin igarotzen dira, hauen artean ontzi aldaketa hauek egitea espero da: Euskotren, Lurraldebus Ekialdebus, La Burundesa S.A. eta Lurraldebus Pesa (denak ontzi aldaketa Donostian eginez).
- **Lortutako emaitza**: (4.27. irudian ikus daiteke) Bidaia zuzena soilik Renferekin egin daiteke. Bi konpainiekin bidaiatuz gero Lurraldebus Tsst hartu daiteke jatorrian, Donostian jaitsi eta han Euskotren, Lurraldebus Ekialdebus (bost linea desberdin dauzkana aukeratzeko), La Burundesa S.A. edo Lurraldebus Pesa. Google Directions emaitzarekin konparatuz gero ikusi daiteke nire emaitzen artean berarena dagoela (Renfe konpainiarekin bidaiatzeko aukera), aldi berean aukera hoberena dela.
- **JUnit kodea**:

```
Resources res = activity.getResources();
LatLng origin= new LatLng(43.1349045,-2.0771048);
LatLng dest = new LatLng(43.3376933,-1.796974);
boolean cbOrduaZehaztu = false;
int urtea = null;
int hilabetea=null;
int eguna=null;
int ordua=null;
int minutua=null;
boolean trenBilaketa=true;
boolean autobusBilaketa=true;
float ibiliDistMax=500;
boolean bilaketaLokala=true; //Azkarrago egiteko
boolean bidaiZuzena=false;

ArrayList<BidaiEmaitza> bilaketaEmaitza =  new
ArrayList<BidaiEmaitza>();
BilaketaOpenData bilaketa = new BilaketaOpenData(activity, res);
bilaketaEmaitza = bilaketa.informazioaEskuratu(origin, dest,
```

```
                    trenBilaketa, autobusBilaketa, cbOrduaZehaztu,
                    urtea, hilabetea, eguna, ordua, minutua,
                    ibiliDistMax, bilaketaLokala, bidaiZuzena);
assertEquals("Emaitza kopurua bakarrekoa izan bearko luke",
             7, bilaketaEmaitza.size());

String konpainiak = "";
konpainiak = bilaketaEmaitza.get(0).getSteps().
            get(1).getTransitDetails().getLine().
            getAgencies(0).getName();
assertEquals("Lehenengo konpainia Renfe izan behar da",
             "Renfe", konpainiak);


konpainiak = "";
konpainiak =
bilaketaEmaitza.get(1).getSteps().get(1).getTransitDetails().
                getLine().getAgencies(0).getName()+"_";
konpainiak = konpainiak + bilaketaEmaitza.get(1).getSteps().
             get(3).getTransitDetails().getLine().
             getAgencies(0).getName();
assertEquals("Bigarren konpainiak LurraldebusTsst eta Euskotren izan
              behar dira", "Lurraldebus Tsst_Euskotren", konpainiak);


konpainiak = "";
konpainiak =
bilaketaEmaitza.get(2).getSteps().get(1).getTransitDetails().
                getLine().getAgencies(0).getName()+"_";
konpainiak = konpainiak + bilaketaEmaitza.get(2).getSteps().
             get(3).getTransitDetails().getLine().
             getAgencies(0).getName();
assertEquals("Bigarren konpainiak LurraldebusTsst eta La Burundesa S.A.
izan behar dira", "Lurraldebus Tsst_La Burundesa S.A.", konpainiak);


konpainiak = "";
konpainiak =
bilaketaEmaitza.get(3).getSteps().get(1).getTransitDetails().
                getLine().getAgencies(0).getName()+"_";
konpainiak = konpainiak +
bilaketaEmaitza.get(3).getSteps().get(3).getTransitDetails().
                getLine().getAgencies(0).getName();
assertEquals("Hirugarren LurraldebusTsst eta Lurraldebus Edialdebus
izan behar dira", "Lurraldebus Tsst_Lurraldebus Ekialdebus",
konpainiak);

//3-5 posizioko emaitzak bigarrenaren konpainia bikote bera dira, hortaz ez da jarriko.


konpainiak = "";
konpainiak =
bilaketaEmaitza.get(7).getSteps().get(1).getTransitDetails().
                getLine().getAgencies(0).getName()+"_";
konpainiak = konpainiak + bilaketaEmaitza.get(6).getSteps().get(3).
             getTransitDetails().getLine().getAgencies(0).getName();
assertEquals("Bostgarren konpainiak LurraldebusTsst eta Lurraldebus
Pesa izan behar dira", "Lurraldebus Tsst_Lurraldebus Pesa",
konpainiak);
```

4.8. kode-zatia: Bilaketa normala proba

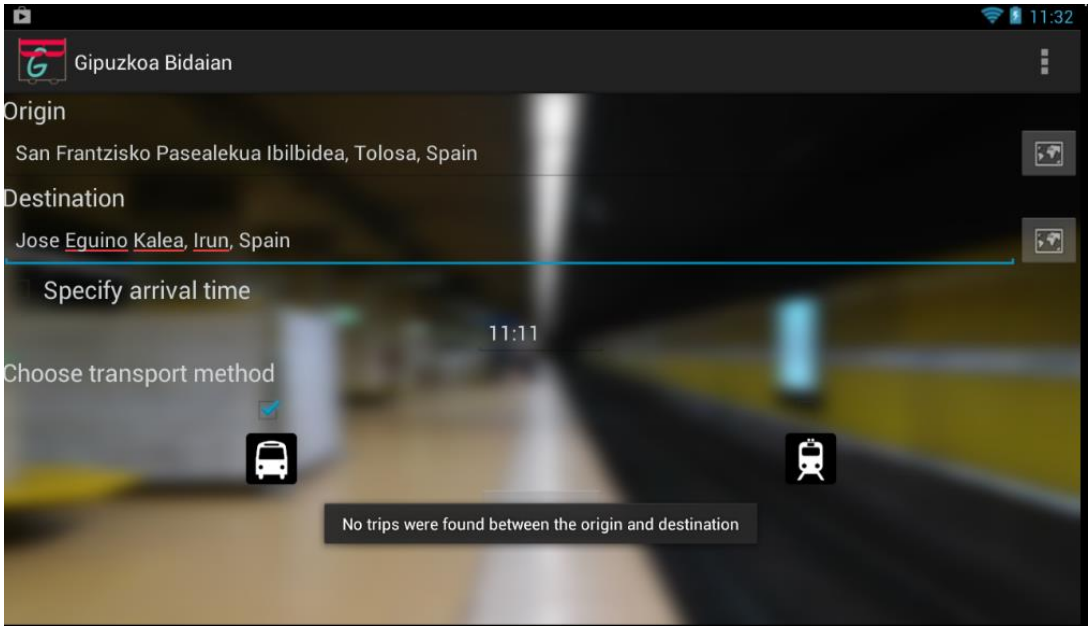
4.27. irudia: Bilaketa arrunt baten emaitza

4.4.1.3. Emaitzarik gabeko bilaketa autobusarekin soilik

- **Deskribapena**: aurreko probak ikusita garbi dago Tolosarik Irunera zuzenean soilik tren bidez joan daitekeela, hortaz bilaketa parametroetan *autobus* aukera bakarra uzten bada ez luke bidaiarik lortu behar.
- **Bilaketa modua**: bilaketa zehatza (Open Data).
- **Jatorria helbidea**: San Frantzisko Pasealekua Ibilbidea, Tolosa (43.1349045,-2.0771048).
- **Helmuga helbidea**: Jose Eguino Kalea, Irun(43.3376933,-1.796974) .
- **Orduaren arabera?**: ez .
- **Garraio moduak**: autobusa.
- **Oinezko distantzia maximoa**: 500 metro.
- **Bidaia zuzenak soilik?**: bai
- **Espero den emaitza**: ez da espero inongo emaitzarik.
- **Lortutako emaitza**: ez da lortu emaitzarik, 4.28. irudiak erakusten duen bezala.
- **JUnit kodea**:

```
Resources res = activity.getResources();
LatLng origin= new LatLng(43.1349045,-2.0771048);
LatLng dest = new LatLng(43.3376933,-1.796974);
boolean cbOrduaZehaztu = false;
int urtea = null;
int hilabetea=null;
int eguna=null;
int ordua=null;
int minutua=null;
boolean trenBilaketa=false;
boolean autobusBilaketa=true;
float ibiliDistMax=500;
boolean bilaketaLokala=true; //Azkarrago egiteko
boolean bidaiZuzena=true;

ArrayList<BidaiEmaitza> bilaketaEmaitza =  new
ArrayList<BidaiEmaitza>();
BilaketaOpenData bilaketa = new BilaketaOpenData(activity, res);
bilaketaEmaitza = bilaketa.informazioaEskuratu(origin, dest,
                  trenBilaketa, autobusBilaketa, cbOrduaZehaztu,
                  urtea, hilabetea, eguna, ordua, minutua,
                  ibiliDistMax, bilaketaLokala, bidaiZuzena);
assertEquals("Ez da emaitzarik espero", 0, bilaketaEmaitza.size());
```

4.9. kode-zatia: Emaitzarik gabeko proba



4.28. irudia: Emaitzarik ez duen bilaketa

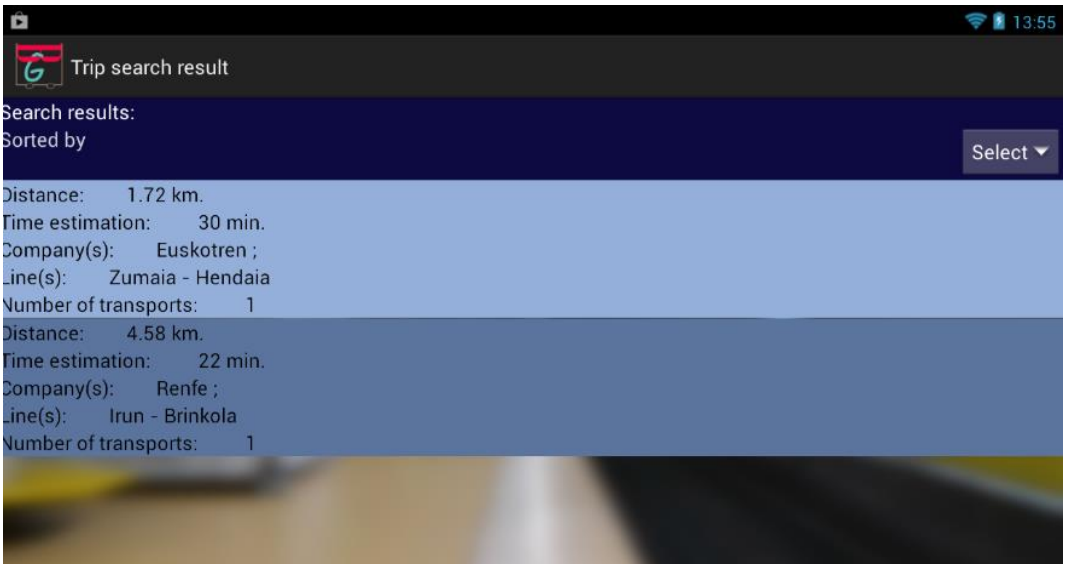
4.4.1.4. Bilaketa soilik trenarekin

- **Deskribapena**: jatorrian eta helmugan bi motako garraioak igarotzen diren lekuen arteko bidaiak bilatuko dira. Soilik trenez mugitzeko esango zaionez aplikazioari bi emaitza lortuz.
- **Bilaketa modua**: bilaketa zehatza (Open Data).
- **Jatorria helbidea**: Calle Viteri, 39, Renteria (43.31411661302957,-1.9026128947734833).
- **Helmuga helbidea**: De Larratxo Ibilbidea 94, San Sebastian (43.31654409279899,-1.933806985616684).
- **Orduaren arabera?**: ez.
- **Garraio moduak**: trena.
- **Oinezko distantzia maximoa**: 1000.
- **Bidaia zuzenak soilik?**: ez.
- **Espero den emaitza**: nahiz eta jatorritik Lurraldebus Urbano de Errenteria, Euskotren, Renfe eta Lurraldebus Ekialdebus konpainien lineak pasa eta helmugatik Dbus, Lurraldebus Ekialdebus, Euskotren eta Renfe igaro, soilik treneko aukerak kontuan hartzen direnez bi emaitza lortu beharko lirateke.
- **Lortutako emaitza**: Euskotren eta Renfe. Espero nuen bezala beste konpainiaren bidaiak ez dira aintzat hartu, 4.29. irudian ikusi daitekeen bezala.

- JUnit kodea:

```
Resources res = activity.getResources();
LatLng origin= new LatLng(43.31411661302957,-1.9026128947734833);
LatLng dest = new LatLng(43.31654409279899,-1.9338069856166 84);
boolean cbOrduaZehaztu = false;
int urtea = null;
int hilabetea=null;
int eguna=null;
int ordua=null;
boolean trenBilaketa=true;
boolean autobusBilaketa=false;
float ibiliDistMax=1000;
boolean bilaketaLokala=true; //Azkarrago egiteko
boolean bidaiZuzena=false;

ArrayList<BidaiEmaitza> bilaketaEmaitza =  new
ArrayList<BidaiEmaitza>();
BilaketaOpenData bilaketa = new BilaketaOpenData(activity, res);
bilaketaEmaitza = bilaketa.informazioaEskuratu(origin, dest,
                  trenBilaketa, autobusBilaketa, cbOrduaZehaztu,
                  urtea, hilabetea, eguna, ordua, minutua,
                  ibiliDistMax, bilaketaLokala, bidaiZuzena);
assertEquals("Bi emaitza espero dira", 2, bilaketaEmaitza.size());

String konpainiak = "";
konpainiak = bilaketaEmaitza.get(0).getSteps().get(1).
             getTransitDetails().getLine().getAgencies(0).getName();
assertEquals("Lehenengo konpainia Euskotren izan behar da",
             "Euskotren", konpainiak);

konpainiak = "";
konpainiak = bilaketaEmaitza.get(1).getSteps().get(1).
             getTransitDetails().getLine().getAgencies(0).getName();
assertEquals("Bigarren konpainia Renfe izan behar da",
             "Renfe", konpainiak);
```

4.10. kode-zatia: "Bilaketa trenarekin soilik" proba



4.29. irudia: Tren bidai bilaketaren emaitza

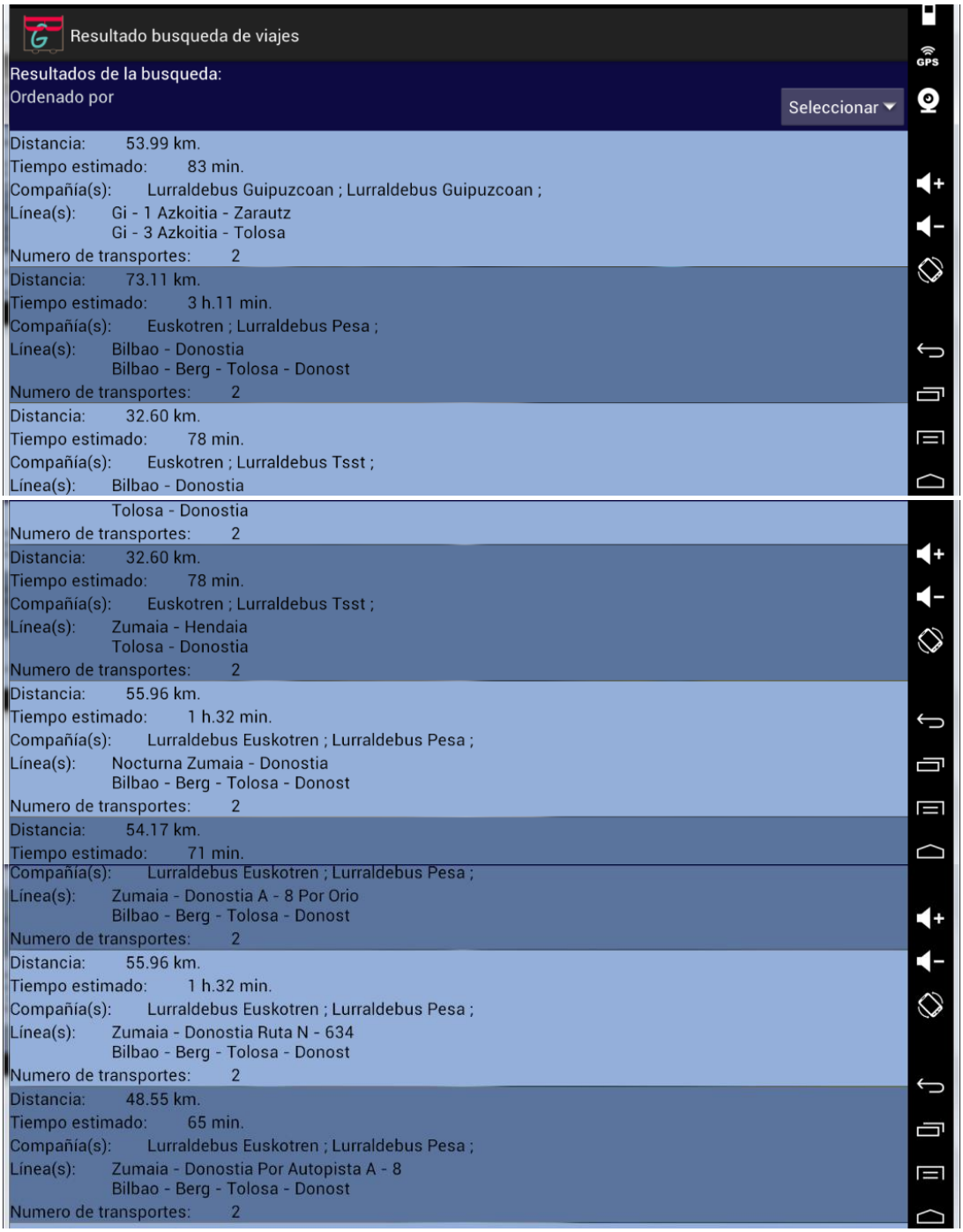
4.4.1.5. Bidaia zuzenak ez dituen bilaketak

- **Deskribapena**: bilaketa bat gauzatuko da zeinean jatorriaren eta helmugaren artean ez dagoen bidaia zuzenik, soilik ontzi aldaketa egin behar den bidaiak lortuko dira. Bi proba egingo dira, lehenengoa normal eta bigarrena bidaia zuzenak soilik bilatuz (emaitzarik ez lortzeko).
- **Bilaketa modua**: bilaketa zehatza (Open Data).
- **Jatorria helbidea**: Mitxelena Kalea, 23 Zarauz (43.283366, -2.166334).
- **Helmuga helbidea**: Calle Pablo Gorosabel, 4, Tolosa (43.13704554464196, -2.0740442723035812).
- **Orduaren arabera?**: ez
- **Garraio moduak**: autobusa eta trena
- **Oinezko distantzia maximoa**: 500 metro.
- **Bidaia zuzenak soilik?**: ez/bai.
- **Espero den emaitza**: Jatorritik igarotzen diren konpainiak Lurraldebus Guipuzcoana, Euskotren eta Lurraldebus Euskotren dira. Helmugatik ordea Lurraldebus Guipuzcoana, Lurraldebus Pesa, Lurraldebus Tolosaldea, Lurraldebus Tsst eta Renfe Donostia-San Sebastian. Ondorioz, posible izango zen bidaia zuzen bakarra lehenengo konpainiarekin izango litzateke, baina hori ez luke gertatu behar. Konpainia horrek hiru linea dauzka, Azkoitia-Zarautz, Azkoitia-Donostia eta Azkoitia-Tolosa. Izenek adierazten duten bezala, ez da bat ere igarotzen aldi berean jatorritik eta helmugatik, ondorioz, itxaron daitekeen emaitza bakarra ontzi aldaketa bidez da.
- **Lortutako emaitza**: (4.30. irudia)Espero zen bezala hainbat bidaiaren emaitzak lortu dira eta denak ontzi aldaketa behar duten bidaiak izan dira. Lurraldebus Guipuzcoana kasuan adibidez, lehenengo GI-1 (Azkoitia-Zarautz) linea hartu beharko litzateke eta ondoren GI-3 (Azkoitia-Tolosa). Bigarren proba egitean mezu bat atera da esanez ez dagoela jatorri eta helmuga horien artean bidaia zuzenik (4.31. irudia).
- **JUnit kodea (1. Proba)**:

```
Resources res = activity.getResources();
LatLng origin= new LatLng(43.283366, -2.166334);
LatLng dest = new LatLng(43.13704554464196, -2.0740442723035812);
boolean cbOrduaZehaztu = false;
int urtea = null;
int hilabetea=null;
int eguna=null;
int ordua=null;
boolean trenBilaketa=true;
boolean autobusBilaketa=true;
float ibiliDistMax=500;
boolean bilaketaLokala=true; //Azkarrago egiteko
boolean bidaiZuzena=false;

ArrayList<BidaiEmaitza> bilaketaEmaitza =  new ArrayList<BidaiEmaitza>();
BilaketaOpenData bilaketa = new BilaketaOpenData(activity, res);
```

```
bilaketaEmaitza = bilaketa.informazioaEskuratu(origin, dest, trenBilaketa,
                autobusBilaketa, cbOrduaZehaztu, urtea, hilabetea, eguna,
                ordua, minutua, ibiliDistMax, bilaketaLokala, bidaiZuzena);
assertEquals("Hamabi emaitza espero dira", 12, bilaketaEmaitza.size());
```

4.11. kode-zatia: Bidaia zuzenak ez dituen bilaketaren proba

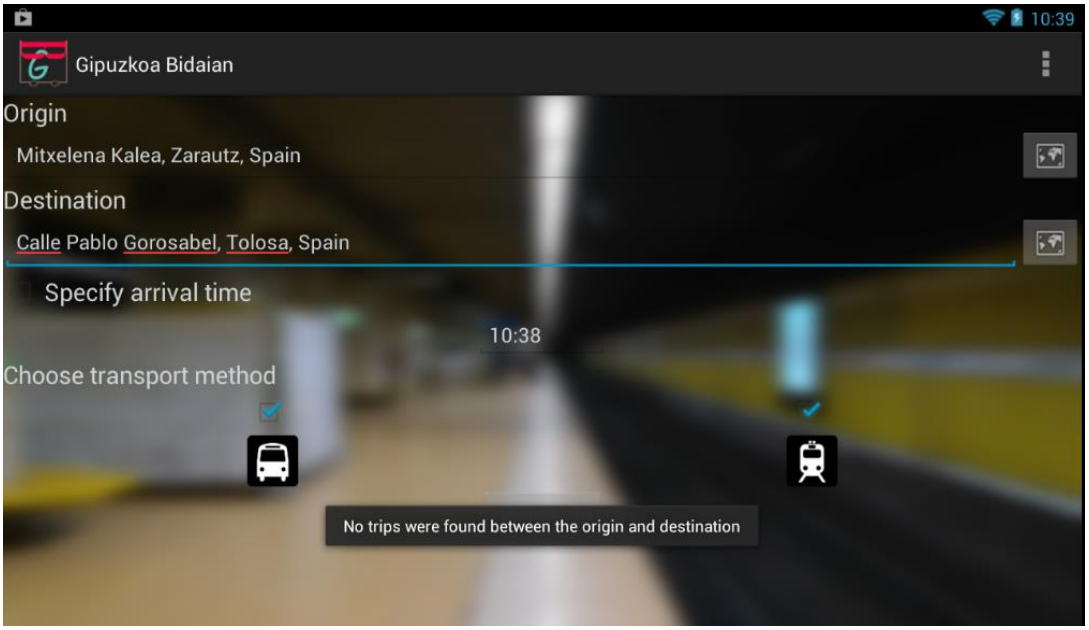


4.30.irudia: Bidaia konposatuak soilik lortutako emaitza

- JUnit kodea (2. Proba):

```
Resources res = activity.getResources();
LatLng origin= new LatLng(43.283366, -2.166334);
LatLng dest = new LatLng(43.13704554464196, -2.0740442723035812);
boolean cbOrduaZehaztu = false;
int urtea = null;
int hilabetea=null;
int eguna=null;
int ordua=null;
boolean trenBilaketa=true;
boolean autobusBilaketa=true;
float ibiliDistMax=500;
boolean bilaketaLokala=true; //Azkarrago egiteko
boolean bidaiZuzena=true;

ArrayList<BidaiEmaitza> bilaketaEmaitza =  new ArrayList<BidaiEmaitza>();
BilaketaOpenData bilaketa = new BilaketaOpenData(activity, res);
bilaketaEmaitza = bilaketa.informazioaEskuratu(origin, dest, trenBilaketa,
                autobusBilaketa, cbOrduaZehaztu, urtea, hilabetea, eguna,
                ordua, minutua, ibiliDistMax, bilaketaLokala, bidaiZuzena);
assertEquals("Ez da emaitzarik espero", 0, bilaketaEmaitza.size());
```

4.12. kode-zatia: Bidaia zuzenak ez dituen bilaketaren proba emaitzarik ez lortzeko



4.31. irudia: Bidaia zuzenak soilik bilatu eta emaitzarik ez lortu

4.4.1.6. Distantziaren araberako bilaketa

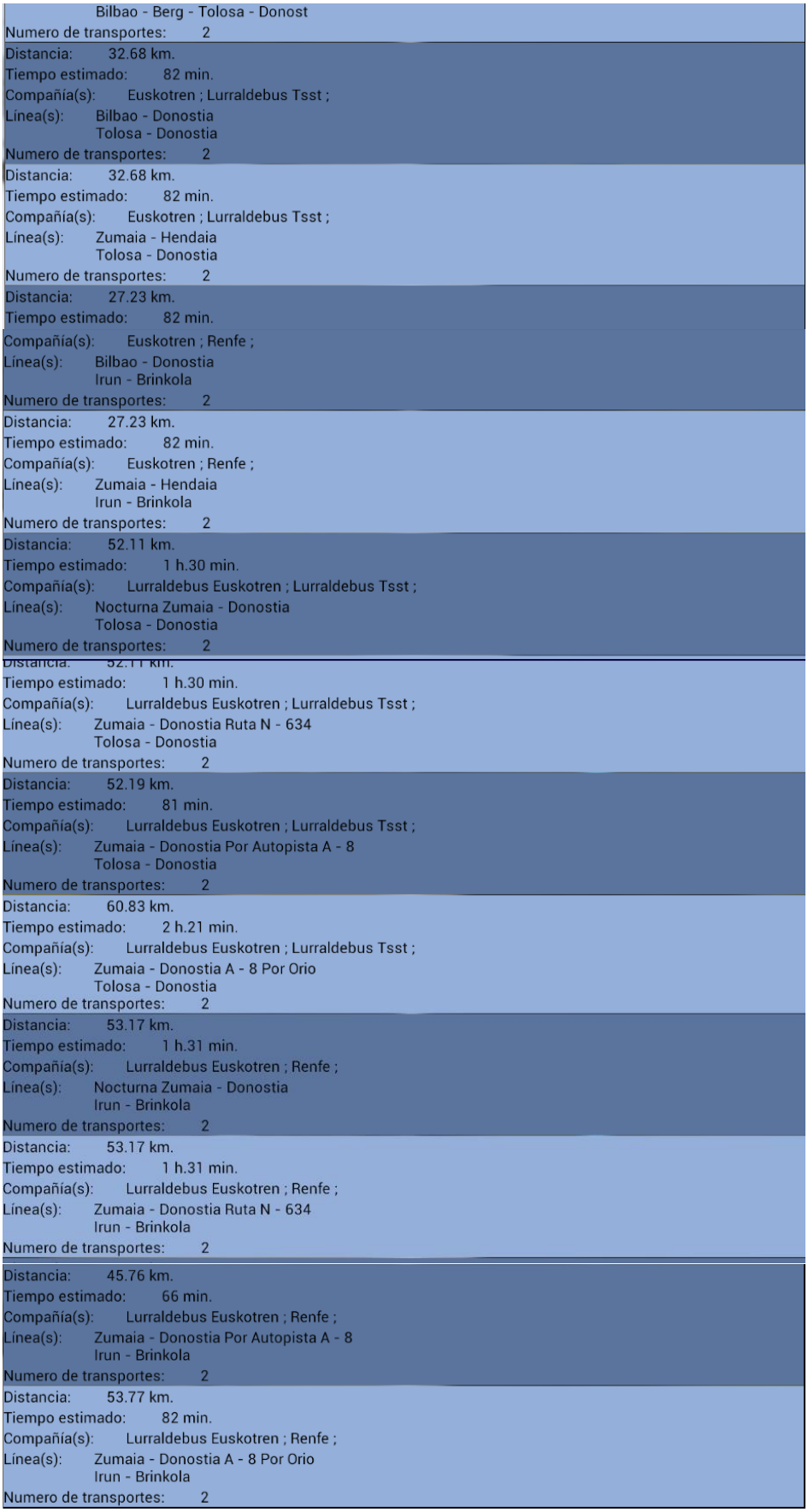
- **Deskribapena**: proba honetan frogatuko da oinezko distantzia handituz gero, emaitzak lortzeko posibilitateak handitzen direla. Horretarako aurreko probaren parametro berdinak erabiliko dira geltoki bilaketa erradioa kilometro batez handituz, era honetan lehen kontuan hartu ez diren geltoki batzuk limitearen barruan sartuko dira bidaia emaitza posibilitateak handituz.
- **Bilaketa modua**: bilaketa zehatza (Open Data).
- **Jatorria helbidea**: Mitxelena Kalea, 23 Zarauz (43.18327911106921, -2.166339084506035).
- **Helmuga helbidea**: Calle Pablo Gorosabel, 4, Tolosa (43.13704554464196, -2.0740442723035812).
- **Orduaren arabera?**: ez.
- **Garraio moduak**: autobusa eta trena.
- **Oinezko distantzia maximoa**: 1500 metro
- **Bidaia zuzenak soilik?**: ez.
- **Espero den emaitza**: hainbat emaitza espero dira, lehen esan den bezala, hainbat konpainia desberdin igarotzen dira bi puntuetatik. Arazoa da gehienak ontzi aldaketa Donostian egiten dutela eta ontzi aldaketaren bi geltokiak urrunegi egon daitezkeela, baina frogapen honetan oinezko distantzia handiagoa denez ontzi aldaketa kopurua igotzea espero da.

- **Lortutako emaitza**: 4.30. irudia eta 4.32. irudia konparatuz gero, ikus daiteke emaitza kopura ez dela berdina, are gehiago, handitu da. Orain Lurraldebus Pesa konpainia jatorritik eta helmugatik igarotzen diren bidaia konbinazioak egin ditzake berak bakarrik, ondorioz, ez da beste konpainiekin nahastuko. Baina aldi berean, hartu daitezkeen geltoki kopurua handitu denez, garraio konbinazio berrika agertu dira: Euskotren - Renfe eta Lurraldebus Eusktren - Renfe.

- **JUnit kodea**:

```
Resources res = activity.getResources();
LatLng origin= new LatLng(43.283366, -2.166334);
LatLng dest = new LatLng(43.13704554464196, -2.0740442723035812);
boolean cbOrduaZehaztu = false;
int urtea = null;
int hilabetea=null;
int eguna=null;
int ordua=null;
boolean trenBilaketa=true;
boolean autobusBilaketa=true;
float ibiliDistMax=1500;
boolean bilaketaLokala=true; //Azkarrago egiteko
boolean bidaiZuzena=false;

rrayList<BidaiEmaitza> bilaketaEmaitza =  new
ArrayList<BidaiEmaitza>();
BilaketaOpenData bilaketa = new BilaketaOpenData(activity, res);
bilaketaEmaitza = bilaketa.informazioaEskuratu(origin, dest,
                  trenBilaketa, autobusBilaketa, cbOrduaZehaztu,
                  urtea, hilabetea, eguna, ordua, minutua,
                  ibiliDistMax, bilaketaLokala, bidaiZuzena);
boolean txikiago = bilaketaEmaitza.size()>12;
assertEquals("Hamabi emaitza baino gehiago espero dira",
             true, txikiago);
```

4.13. kode-zatia: Distantziaren arabera emaitza desberdina lortuko duen proba



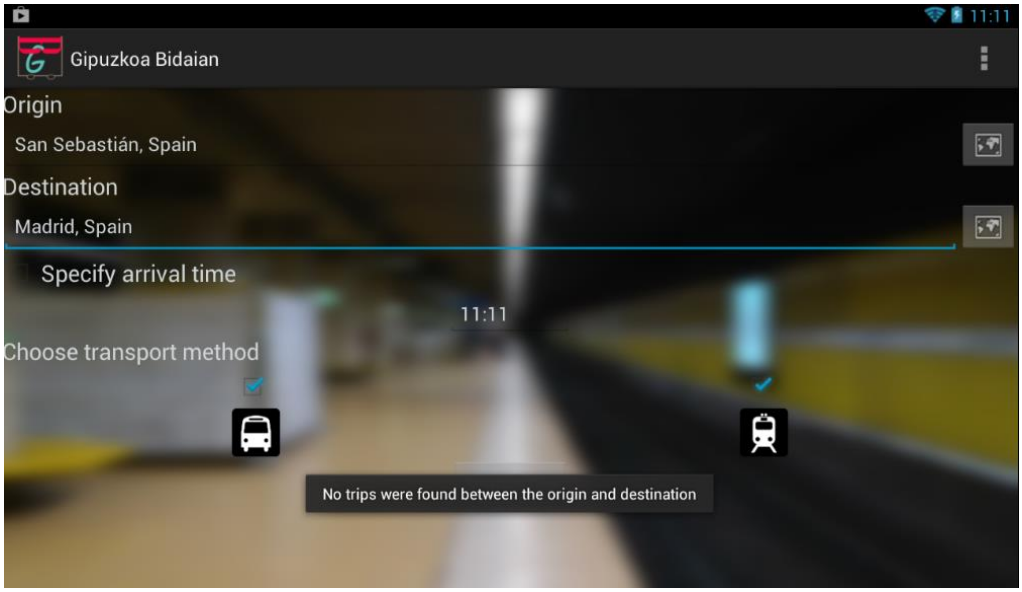
4.32. irudia: Distantzia handitzean bilaketaren emaitza

4.4.1.7. Limitez kanpoko saiakera Open Data-rekin

- **Deskribapena**: proba honetan frogatuko da, bilaketa zehatza erabiliz gero, GipuzkoaBidaian soilik Gipuzkoan bidaiak bilatu ahal direla.
- **Bilaketa modua**: bilaketa zehatza (Open Data).
- **Jatorria helbidea**: San Sebastian, Spain (43.318334,-1.9812313).
- **Helmuga helbidea**: Madrid, Spain (40.4167754,-3.7037902).
- **Orduaren arabera?**: ez.
- **Garraio moduak**: autobusa eta trena.
- **Oinezko distantzia maximoa**: 1500 metro
- **Bidaia zuzenak soilik?**: ez.
- **Espero den emaitza**: ez dut inolako emaitzarik espero.
- **Lortutako emaitza**: ez dut emaitzarik lortu, Open Data-ko datuetan ez baitago Madrileko garraiobideen informaziorik (4.33. irudia).
- **JUnit kodea**:

```
Resources res = activity.getResources();
LatLng origin= new LatLng(43.318334,-1.9812313;
LatLng dest = new LatLng(40.4167754,-3.7037902);
boolean cbOrduaZehaztu = false;
int urtea = null;
int hilabetea=null;
int eguna=null;
int ordua=null;
boolean trenBilaketa=true;
boolean autobusBilaketa=true;
float ibiliDistMax=1500;
boolean bilaketaLokala=true; //Azkarrago egiteko
boolean bidaiZuzena=false;

ArrayList<BidaiEmaitza> bilaketaEmaitza =  new
ArrayList<BidaiEmaitza>();
BilaketaOpenData bilaketa = new BilaketaOpenData(activity, res);
bilaketaEmaitza = bilaketa.informazioaEskuratu(origin, dest,
                  trenBilaketa, autobusBilaketa, cbOrduaZehaztu,
                  urtea, hilabetea, eguna, ordua, minutua,
                  ibiliDistMax, bilaketaLokala, bidaiZuzena);
assertEquals("Ez da emaitzik espero",0, bilaketaEmaitza.size());
```

4.14. kode-zatia: Limitez kanpoko proba Open Data-rekin



4.33. irudia: Limitetik kanpoko bilaketaren emaitza

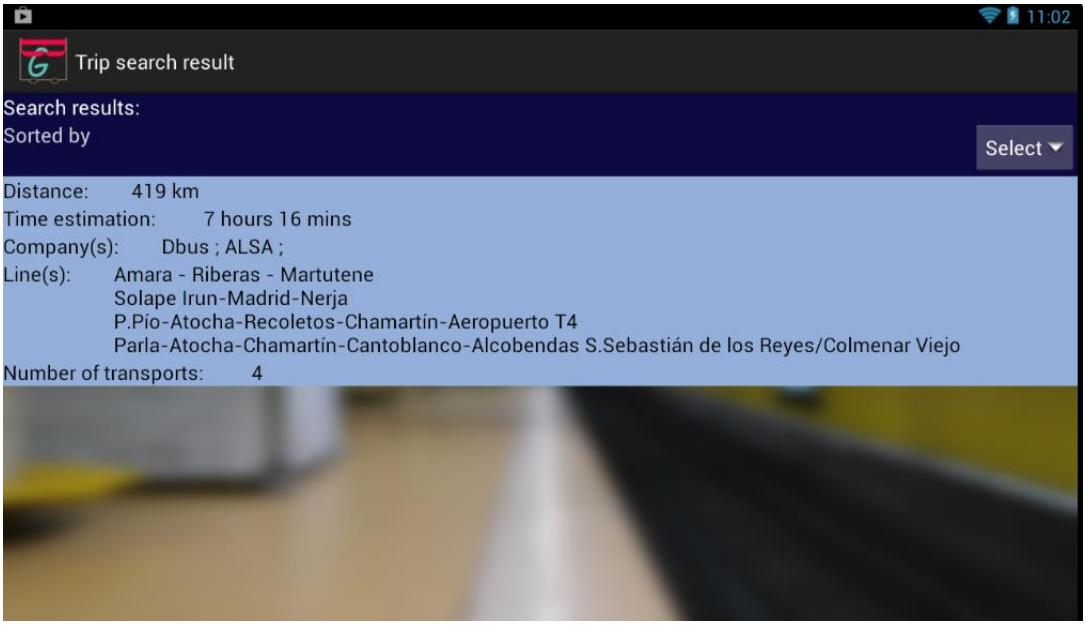
4.4.1.8. Limitez kanpoko saiakera Google Directions

- **Deskribapena**: Proba honetan frogatuko da Google-ek eskaintzen duen web zerbitzuak ez duela limiterik.
- **Bilaketa modua**: Bilaketa azkarra (Google Directions).
- **Jatorria helbidea**: San Sebastian, Spain (43.318334,-1.9812313).
- **Helmuga helbidea**: Madrid, Spain (40.4167754,-3.7037902).
- **Orduaren arabera?**: ez da kontuan hartzen, unekoa hartu da (10:58).
- **Garraio moduak**: ez da kontuan hartzen.
- **Oinezko distantzia maximoa**: ez da kontuan hartzen.
- **Bidaia zuzenak soilik?**: ez da kontuan hartzen.
- **Espero den emaitza**: ezinezkoa zait jakitea, hala ere, emaitzaren bat lortzea espero dut.
- **Lortutako emaitza**: 4.34. irudia ikus daiteke bilaketa ondo egin duela.
- **JUnit kodea**:

```
Resources res = activity.getResources();
LatLng origin= new LatLng(43.318334,-1.9812313);
LatLng dest = new LatLng(40.4167754,-3.7037902);
boolean cbOrduaZehaztu = false;
int urtea = 2014;
int hilabetea=4;
int eguna=9;
int ordua=10;
int minutua=58;
```

```
BilaketaGoogleTransit bilaketa = new BilaketaGoogleTransit(activity,
res);
ArrayList<BidaiEmaitza> bilaketaEmaitza =  new
ArrayList<BidaiEmaitza>();
bilaketaEmaitza = bilaketa.informazioaEskuratu(origin, dest,
                  cbOrduaZehaztu, urtea, hilabetea, eguna,
                  ordua, minutua);
boolean dago = bilaketaEmaitza.size()>0;
assertEquals("Emaita bat lortzea espero da", true, dago);
```

4.15. kode-zatia: Limitez kanpoko proba Google Directions-rekin



4.34. irudia: Google Directions-rekin limitez kanpoko saiakeraren emaitza

4.4.2 Diseinu probak

Hemen ezaugarri desberdineko smartphonen aurrean aplikazioaren erantzun desberdinak aztertuko dira.

Arazoa da mapako interfazea ezingo dela nahi adina probatu, emuladoreak ez dutelako Google-eko zerbitzuak instalatuta. Hala ere, ez dago kezkatu beharrik, ezaugarri bateko edo besteko pantailan berdina ikusten baita.

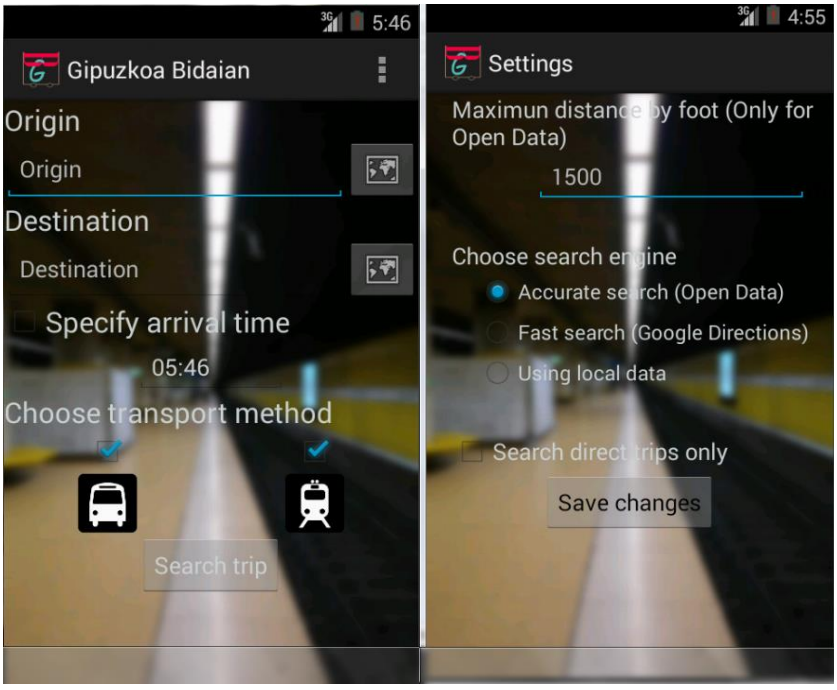
Proba bakoitzean hurrengoa kontuan hartuko da:

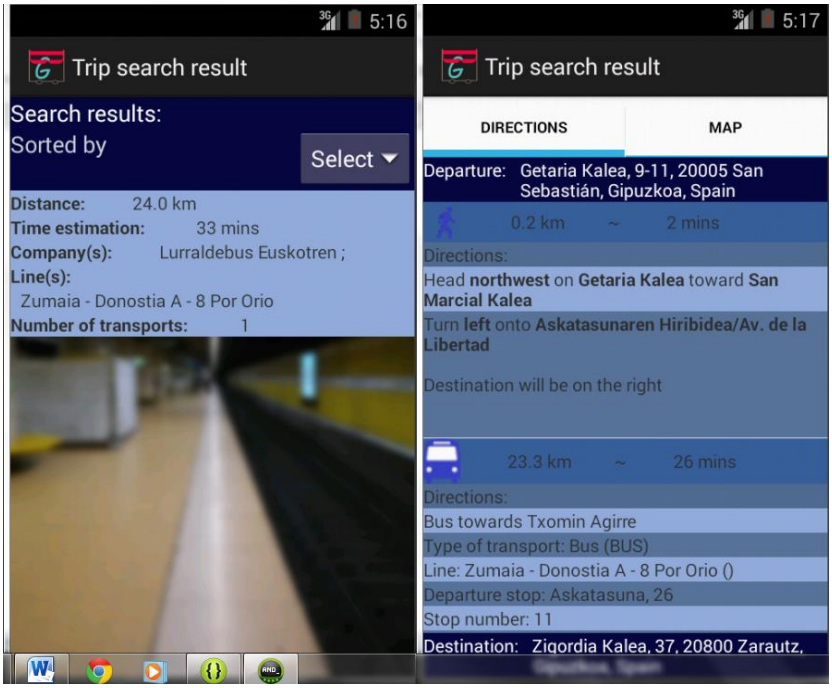
- Smartphone bakoitza duen API maila.
- Pantailaren tamaina dimentsiotan eta ahal bada hazbestekotan.
- Mugikorraren Android bertsioa.
- Zein hizkuntzatan frogan egingo da.

- Emuladorea edo benetako telefonoa den adieraziko da.
- Azalpen bat kasu bakoitzean zer den bereizgarri, hau da, proba bakoitzean zer aldatuko den besteekin konparatuz.

4.4.2.1. Nexus One – API 19

- **API maila**: 19.
- **Pantaila tamaina**: 480x800.
- **Android**: 4.4.
- **Hizkuntza**: ingelesez.
- **Makina**: emuladorea.
- **Azalpena**: diseinurako erabili den makina nagusia da. Papereko zirriborroak kodera eramateko emuladore hau erabiltzea erabaki zen Android garatzaileen webgunean azkeneko bertsioarekin programatzea gomendatzen dutelako.
- **Irudiak**:

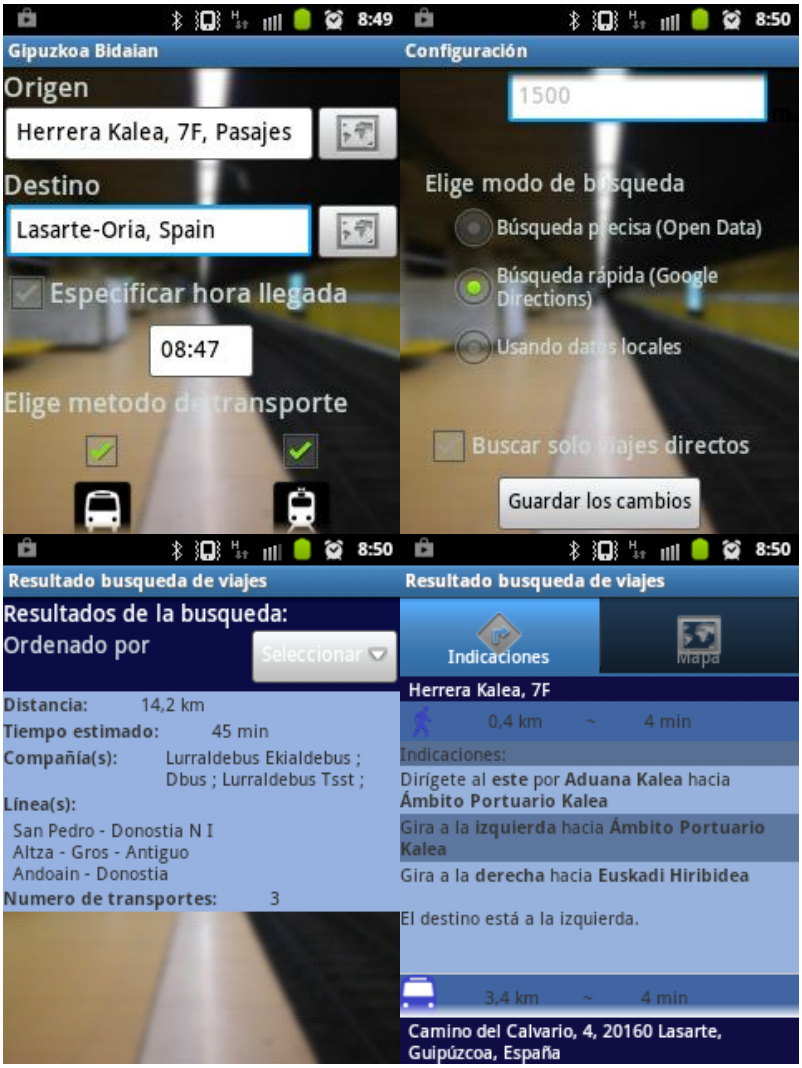




4.35. irudia: Aplikazioa Nexus One mugikorrean instalatuta. Pantaila nagusia (goi ezker), doikuntzak (goi eskuin), bilaketaren emaitza (behe ezker) eta bidaiaren informazio (behe eskuin)

4.4.2.2. Samsung Galaxy Mini – API 8

- **API maila**: 8.
- **Pantaila tamaina**: 240 x 320 (3.14”).
- **Android**: 2.3.6.
- **Hizkuntza**: gaztelaniaz.
- **Mota**: benetako telefonoa.
- **Azalpena**: smartphone hau erabiltzearen zergatia bateragarritasuna zuzena dela frogatzea da. Aplikazioa garatzen zen heinean emuladore berdinarekin probak egiten ziren (denbora aurrezteko) funtzionalitateak zuzenak zela ikusteko. Baina horrek ez du esan nahi aplikazioa ez dela beste makina batean probatu azkeneko momentura arte. Garapenean zehar, batez ere pantailak diseinatzerakoan, hainbat mugikor desberdin erabili ziren bateragarritasuna ziurtatzeko. Kontuan izan behar da Android bertsio berriak eta zaharrak oso desberdinak direla, ez hori bakarrik, telefonoen tamaina anitz dago merkatuan, eta ez da berdina batentzat edo besteentzat diseinatzea.
- **Irudiak**:



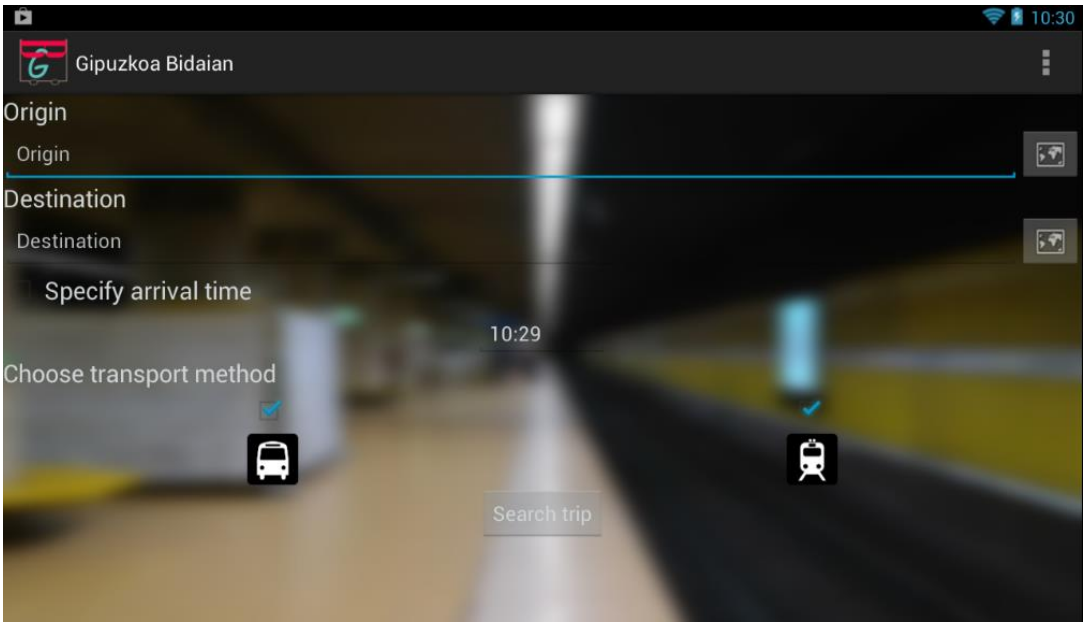
4.36. irudia: Aplikazioa Samsung Galaxy Mini mugikorrean instalatuta. Pantaila nagusia (goi ezker), doikuntzak (goi eskuin), bilaketaren emaitza (behe ezker) eta bidaiaren informazio (behe eskuin)

4.4.2.3. Nexus 7 (Tableta) – API 16

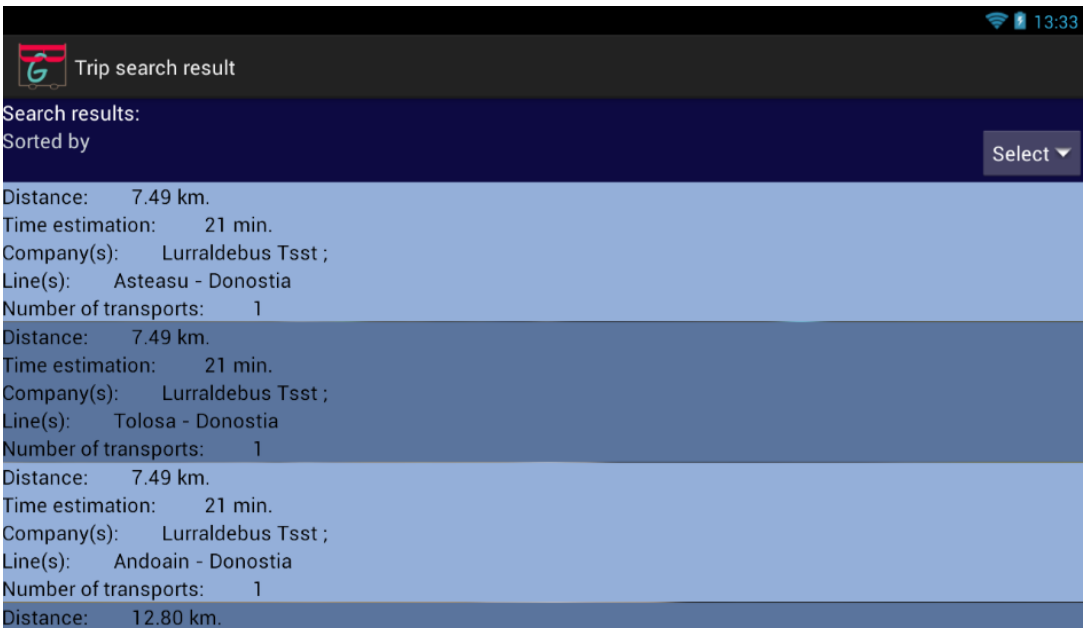
- **API maila**: 16.
- **Pantaila tamaina**: 1280x800 (4.0”).
- **Android**: 4.1.1.
- **Hizkuntza**: ingelesez.
- **Azalpena**: proba hau bereizgarrienetakoa da, ez baita berdina smartphone batentzat edo tableta batzentzat diseinatzea. Azken hauen pantailak askoz zabalagoak dira, ondorioz, “lerro” batean sar daitekeen informazioa handiagoa da. GipuzkoaBidaian-en interfaze gehienek ez dute horretarako kode bereizgarririk, ondo ikusten baita kode lehenetsiarekin. Badaude bi pantaila desberdin ikusten direnak ordea, hauek egokitu izan dira tabletetan hobeto ikusteko. Bata da emaitzak ikusten direnean,

emaitza bakoitzaren lineak alboan daude, espazio arazoa ez baitago. Bigarrena, eta garrantzitsuena, Bidaia baten informazio zehatza begiratzean da, orain jarraibideak eta mapa aldi berean ikusten dira, aplikazioa erakargarriago bihurtuz.

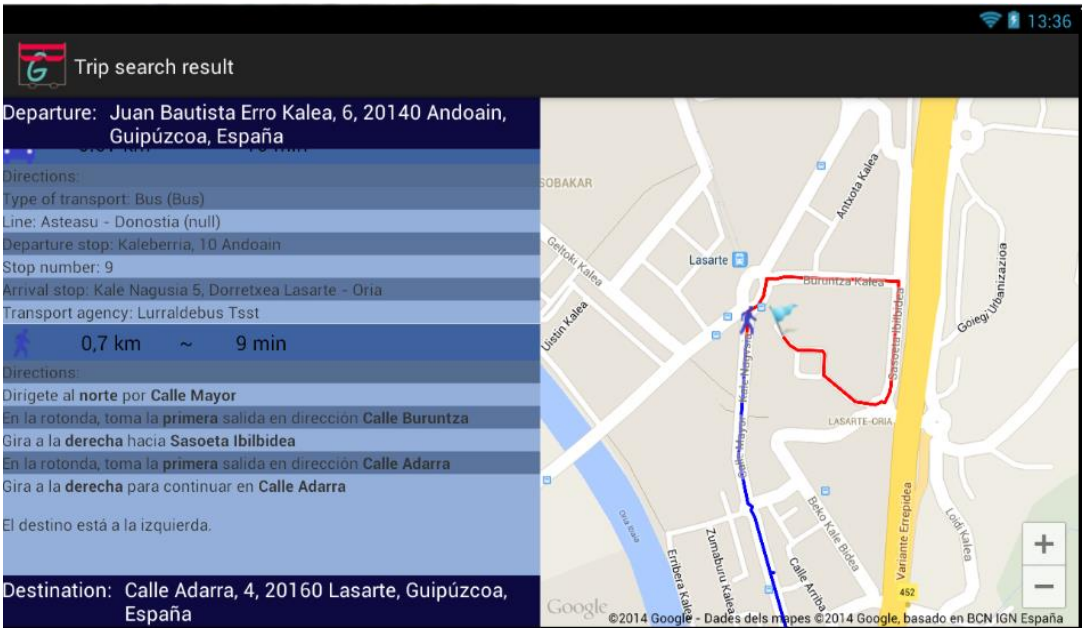
- Irudiak:



4.37. irudia: Aplikazioa Nexus 7 tabletan instalatuta. Pantaila nagusia



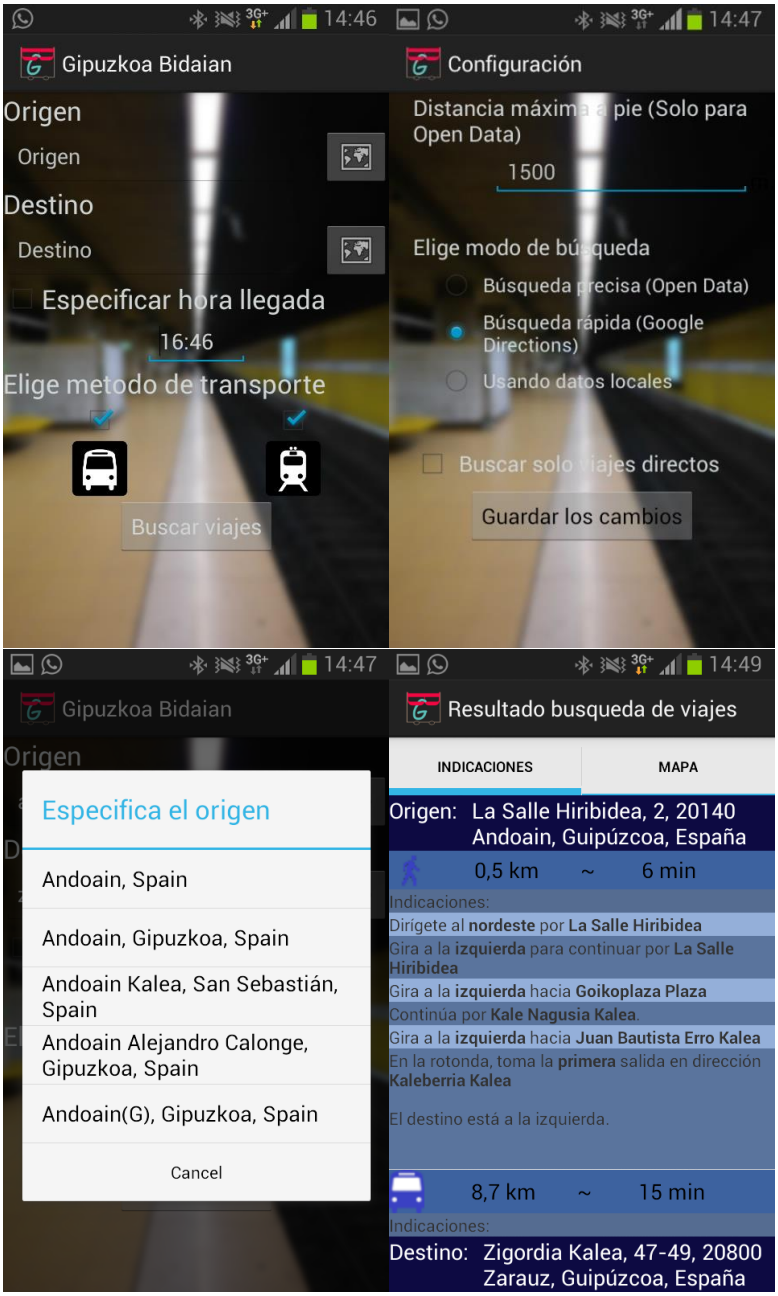
4.38. irudia: Aplikazioa Nexus 7 tabletan instalatuta. Bilaketaren emaitzak



4.39. irudia: Aplikazioa Nexus 7 tabletan instalatuta. Emaitza bate informazio osoa

4.4.2.4. Samsung Galaxy 3 Mini – API 16

- **API** maila: 16.
- **Pantaila tamaina**: 480x800 (4.0”).
- **Android**: 4.1.2.
- **Hizkuntza**: gaztelaniaz.
- **Makina**: benetako mugikorra
- **Azalpena**: proba honen helburua aplikazioa beste edozein terminaletan dabilela frogatzea da. Horretarako Android bertsio desberdina duen makina desberdina erabili da. Egia esan proba honek ez du garrantzi gehiegirik, soilik ziurtatu nahi izan da ondo funtzionatzen duela. Eta hala izan da, aplikazioa arazorik gabe ikusten da.
- **Irudiak**:

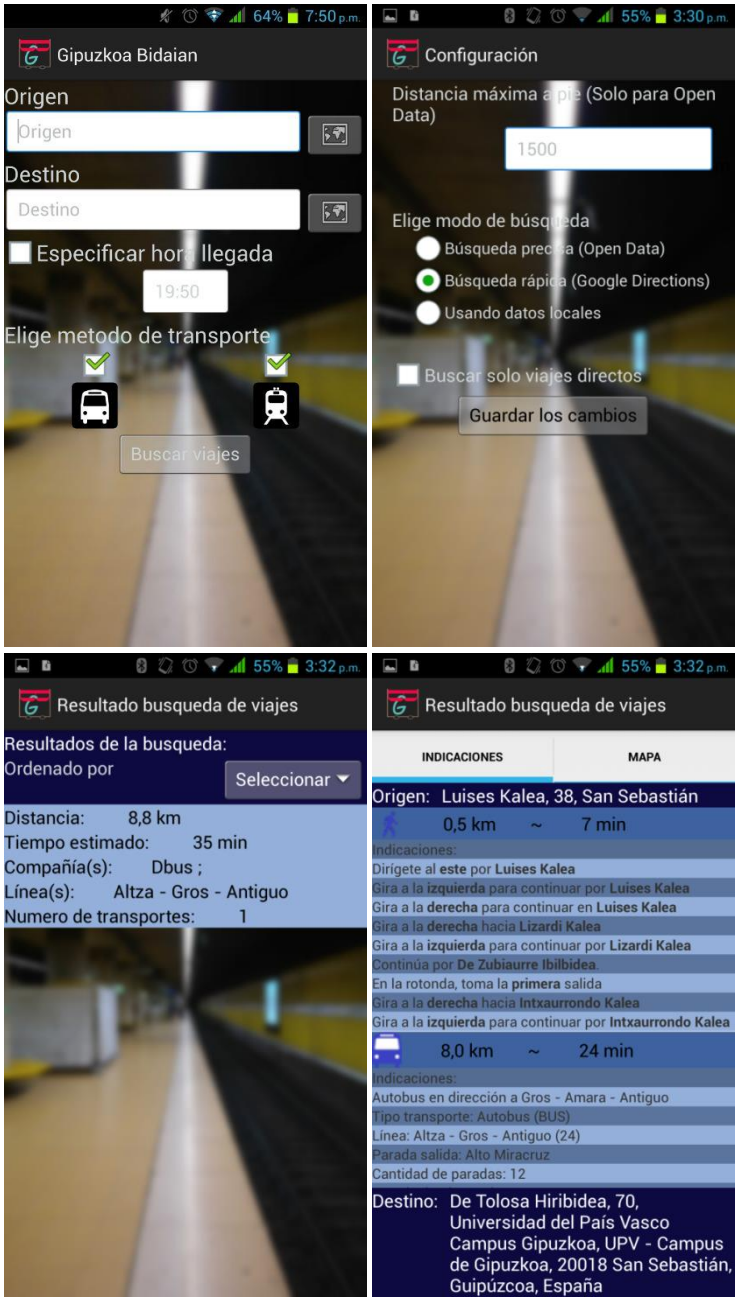


4.40. irudia: Aplikazioa Samsung Galaxy 3 Mini mugikorrean instalatuta. Pantaila nagusia (goi ezker), doikuntzak (goi eskuin), jatorria zehazteko(behe ezker) eta bidaiaren informazio (behe eskuin)

4.4.2.5. Star U9501 – API 17

- **API** maila: 17.
- **Pantaila tamaina**: 1280*720, 5.0”.
- **Android**: 4.2.2.
- **Hizkuntza**: gaztelaniaz.
- **Makina**: benetako mugikorra.

- **Azalpena**: proba honen helburua izan da aplikazioa probatzea mugikor ez arrunt batean, zera da, marka ezezaguneko smartphone batean. Teorikoki ez luke desberdintasunik egon behar, APP-ak Android-aren arabera funtzionatzen dute, hala ere gomendagarria da mugikor berezi hauetan funtzionatzen duela egiaztatzea.
- **Irudiak**:



4.41. irudia: Aplikazioa Star U9501 mugikorrean instalatuta. Pantaila nagusia (goi ezker), doikuntzak (goi eskuin), bilaketaren emaitzak (behe ezker) eta bidaiaren informazio (behe eskuin)

5

Jarraipena eta kontrola

Kapitulu honetan proiektuaren jarraipen txostena azaltzen da. Hasieran, planifikatu zen denbora eta egindakoa alderatuko da, beharrezkoa denean azalduz. Ondoren, kalitatearen, komunikazioaren eta arriskuaren kudeaketa azalduko da.

Jarraipena eta kontrola kapituluaren egitura
1. Burutako lana 2. Komunikazioak 3. Kalitatea 4. Arriskuak

5.1. Burututako lana

Proiektuaren hasieran 358 ordu lan egingo zirela estimatu zen, bukaeran gehiago izan da. 5.1. taulan ikusi daiteke ataza bakoitzean zenbat ordu estimatu ziren, zenbat burutu dira eta desbideratzea zenbatekoa izan den.

Ataza	Estimatutako orduak	Burututakoa	Desbideratzea
Planifikazioa	30	20	% -33,33
Teknologien azterketa	32	17	% -46,87
Aplikazioa garatu	166	265	% 59,64
Probak	30	25	% -16,7
Kudeaketa	30	20	% -33,33
Memoria	50	105	% 110
Defentsa	20	11	% -45
Guztira	**358**	**463**	**% 29,33**

5.1. taula: Proiektuaren desbideratzeak

Desbideratze nabarmenenak aplikazioa garatzerakoan eta memoria idazterakoan izan dira.

Ordu kopuruan desberdintasuna egon arren, noiz burutuko zen planifikazioa mantendu da kasu gehienetan. Kasu batzuetan ezin izan da bete arazoak egon direlako atazekin eta besteetan uneko egoeragatik.

5.1.1 "Aplikazioa garatzea" atazaren desbideratzearen arrazoiak

Denbora gehien suposatu duen ataza aplikazioa garatzearena izan da. Proiektuaren hasieratik jakina zen ez zela lan erraza izango, baina ez hainbeste. 5.1. taulan irudikatzen da aplikazioa garatzeko burutu diren azpi-atazak, ez dira zehazki LDE diagramak agertzen direnak baizik eta lan egiterakoan garatzen ari ziren zatiak.

Hauetako batzuk burutzeko hasieran pentsatutakoa baino denbora gehiago beharrezkoa izan da, 5.2. taulan letra lodiz idatziak dauden orduek adierazten dute. Hala eta guztiz ere, orokorrean ordu gehiago sartzea beharrezkoa izan zen, proiektua pentsatzen zena baino zailagoa baitzen.

Azpi ataza	Burututako orduak
Datu basearen azterketa	9
Interfazea	12
Mapa	**14**
Doikuntzak eta konfigurazio klasea	8
Bidaien emaitza formatua egin	**27**
Bilaketa azkarra - GoogleTransit bilaketak	16
Bilaketa zehatza - Bilaketaren algoritmoa	18
Bilaketa zehatza - FTP konexioa eta irakurketak	**30**
Bilaketa zehatza (hasiera +bidai zuzenak)	**50**
Bilaketa zehatza - Bidaia konposatuak	32
Zuzenketak eta hobekuntzak	49
Guztira	265

5.2. taula: Aplikazioa garatzerakoan lan egindako zatiak

5.1.1.1 Mapa

Mapekin lan egitea ez da batere erraza izan, Android sistemarako ez dagoenez mapa den objektu propiorik, testu eremuak bezala, kanpoko liburutegiak erabiliz lortu behar da. Hasieratik Google Maps erabiliko zela pentsatua zegoen, hobeagoa delako ezagutzen duen objektu bat eskaintzea.

Orduan, lehenengo arazoa agertu zen, mapa gehitzea ez zen lan erraza, hainbat kode lerro gehitu behar baitira. Interneten asko bilatu behar izan zen informazio zuzena aurkitzeko, ez hori bakarrik, kokatu nahi zen lekuaren arabera objektuaren kodea aldatzen zen. Gainera Android 4.4 bertsioa baino zaharragoa diren terminaletan erabiltzeko bateragarritasun liburutegia instalatu behar zen.

Zoritxarrez arazoak ez ziren hor bukatu, mapa ikusi ahal izateko smartphonak Google-eko zerbitzuak, Google Services, instalatuta eduki behar ditu, eta Eclipse-ko makina birtualak ez dakarte aldez aurretik finkatuta. Hau konpontzeko bi aukera zeudela ikasi nuen, edo Google Services eskuz instalatu edo dagoeneko gehituta zekarren emuladore berria aurkitzea. Bigarren aukerarekin jarraitu zen.

Emuladore berria prest edukitzea ez zen lan arina izan, Genymotion eta Oracle Vm BirtualBox instalatu behar zirelako horretarako.

Hala ere, denbora galera honek merezi izan zuen, emuladore berria zaharra baino askoz azkarragoa delako.

5.1.1.2. Bidaia emaitzen formatua

Hasierako planifikazioan ez zen kontuan hartu datuak informazioa iturritik lortzean formatu bat eman beharko zitzaiela. Arazo hau konpontzeko "4.1.3.3. Datu iturri objektuak" ataleko klaseak sortu ziren, hauek fitxategien egitura simulatzen dute, horrela datuak gordetzea prozesu azkarra izango zen.

Dbus-eko artxiboak egitura desberdina izan arren oso antzekoak direnez, egindako aldaketa bakarra, informazioa gordetzen duten klaseetan, funtzio berri bat gehitzea izan zen. 5.1. kode-zatian ikusi daiteke funtzioak egiten duen bakarra da Dbus-eko datuak Open Data-kora (GTFS formatua) egokitzea dela.

```
public RoutesBean adaptazioOpenData(){
      RoutesBean obj;
      obj = new RoutesBean(this.ID, "1",
                  this.ABREVIATURA, this.NOMBRE,
                  this.DESCRIPCION, this.TIPO_VEHICULO,
                  this.URL, this.COLOR,
                  "000000");
      return obj;
}
```

5.1. kode-zatia: *RutaBean* klaseko egokitzapen funtzioa

Bidaiak bilatzerakoan antzekoa gertatu zen, emaitzak pantailan erakusten dituen klaseari datuak formatu zehatz batean bidali behar zaizkio, hauek Google Directions-etik edo Open Data eta Dbus-etik lortu daitezke eta. Hau hasierako planifikazioan ez nuen kontuan izan, horregatik soberako lan denbora dut.

Asko pentsatu eta aztertu ondoren, soluzio sinple eta eraginkorrarenarekin jarraitzea erabaki zen, Google Directions APIak informazioa bidaltzeko erabiltzen duen formatu bera erabili, hau da, "4.1.3.1 Bidaia bat irudikatzen duten objektuak" atalean azaltzen diren klaseak.

5.1.1.3. FTP konexioa eta fitxategi irakurketak

Aplikazioaren zati honek hainbat arazo eman zuen bere hasieratik, FTP zerbitzari batera konektatzea eta edukia deskargatzea erraza bada ere, eraginkorra egitea oso zaila da. Open Data-ko zerbitzarian konpainien informazioa ZIP fitxategietan biltegiratuta dago, hortaz, bi aukera zeuden, deskargatzea eta deskonprimatzea edo zuzenean barruko artxiboak lortu.

Informazio gehiagorik gabe ezinezkoa zen jakitea zenbat denbora behar zuen metodo bakoitzak informazioa eskuratzeko, horregatik erabaki bat hartu baino biekin probak egin ziren.

Interneten informatu ondoren, kodea prestatu zen: *MyFTPClient* klasea sortu zen FTP zerbitzarietara konektatu ahal izateko eta *ZipReader* fitxategi konprimatuekin lan egiteko.

Azkenik, hainbat saiakera egin eta gero, denak funtzionatu zuen, baina denbora asko behar izan zen horretarako, batez ere online zuzenean deskonprimatzeko eta deskargatzeko. Hala ere, lan honek merezi izan zuen, bi metodoen abantailak eta desabantailak aurkitu ziren eta biak erabiltzea erabaki zen, bakoitzak eduki jakin bat lortzeko.

Tamalez arazoak ez ziren hor bukatu, fitxategiak irakurtzeaz gain edukia lortu behar zen eta nolabait egokitu. GTFS formatukoak direnez, fitxategi mota bakoitzarentzako klase bat sortu zen, "4.1.3.3. Datu iturri objektuak" atalekoak.

Edukia CSV formatuan idatzita dago, 5.2. kode-zatian ikus daitekeen modua, horregatik irakurketa eta objektuetan gordeketa oso motela zen. Hau konpontzeko Interneten soluzio bat aurkitu zen, *Super CSV* paketea aurkitu zen.

```
route_id,agency_id,route_short_name,route_long_name,route_desc,
route_type,route_url,route_color,route_text_color
1,1,Bi_Be,Bilbao - Bermeo,"",2,,,""
2,1,De_Le,Deustu - Lezama,"",2,,,""
3,1,Ba-At,Basurto - Atxuri,Tranvía De Bilbao,0,,,""
4,1,Za-Pt,Zamudio - Parque Tecnológico,"",3,,,""
5,1,Zu_He,Zumaia - Hendaia,"",2,,,""
6,1,Bi_Ss,Bilbao - Donostia,"",2,,,""
```

5.2. kode-zatia: *Routes.txt* fitxategiaren adibidea

Super CSV-ko klaseak CSV egiturako fitxategiak irakurtzeko prestatua daude, egin behar den bakarra da *Bean* motako klaseak prestatzea edukia gordetzeko. Behin kodea prestu probatu zen eta harrigarriki oso azkar irakurtzen zituen fitxategiak eta edukia gordetzen zuen.

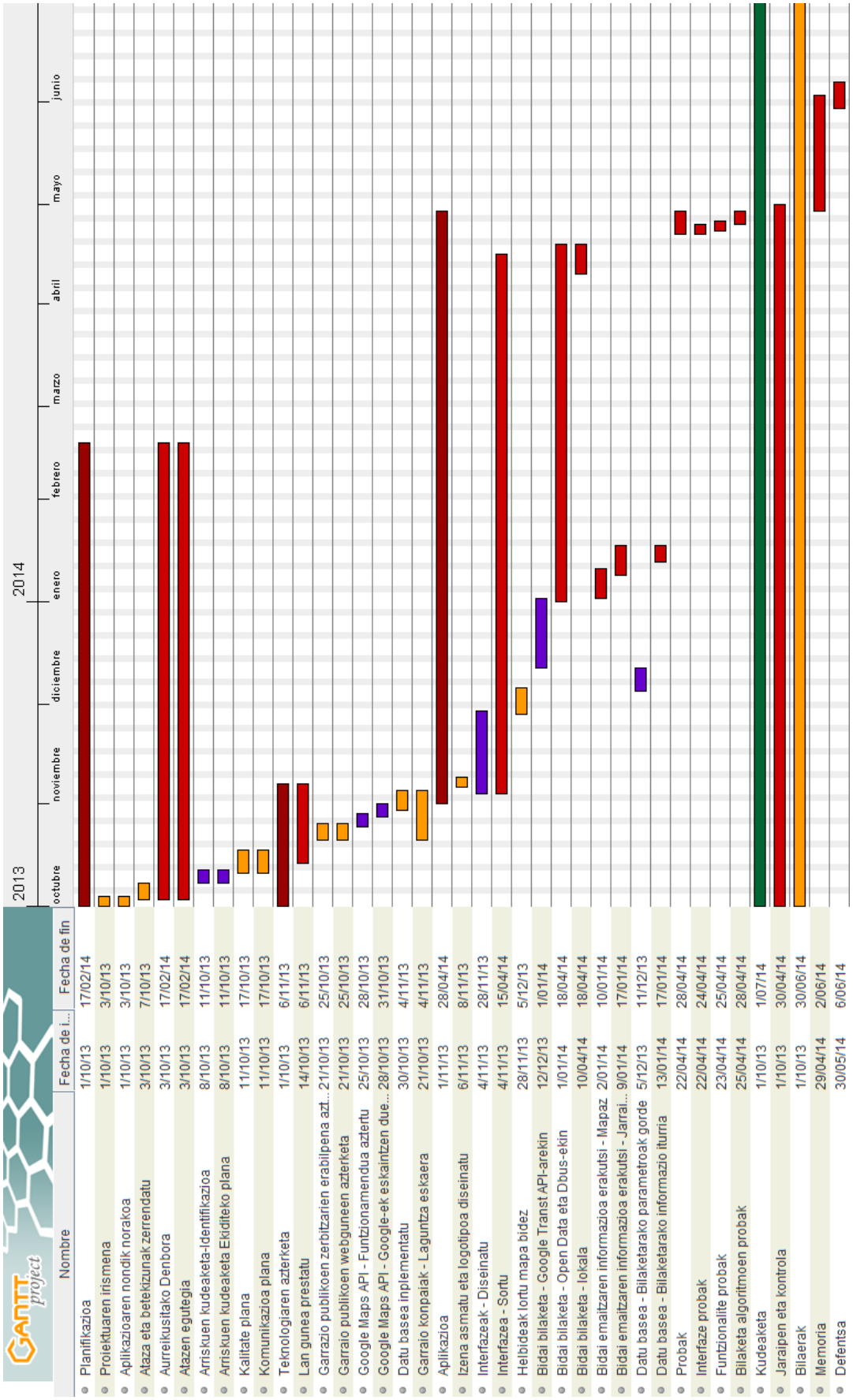
Hala eta guztiz ere, deskargak eta irakurketak ez ziren azkarregiak, horregatik bilaketa nola egingo zer birplanteatu behar izan zen, ahalik eta denbora gutxien gastatzeko. 5.2. taulako "Bilaketa zehatza - Bilaketaren algoritmoa" denboran islatzen da behar izandako aparteko denbora hori.

5.1.1.4. Bilaketa zehatza (hasiera +bidaia zuzenak)

Atal hau garatzean ez zen arazo handirik gertatu, txiki asko izan ziren ordea. Hasieran pentsatutakoa baino zailagoa izaten bukatu zen, horregatik lan egindako ordu kopurua asko igo zen.

Moteltzearen arrazoi handiena hasieratik programa oso eraginkorra sortu nahi zela izan da. Informazio fitxategiak lortzea zein motela zen ikustean, programa ahalik eta eraginkorra egitea erabaki zen, hau da, bidaien bilaketak ahalik eta azkarren egin behar ziren. Honek hainbat kode zati hainbat aldiz idaztea, aldaketak asko egitea eragin zuen, denbora kontsumituz.

Laburtuz, ez da arazorik aurkitu, lana uste zena baino zailagoa izan da eta ondorioz ordu kopurua igo da.



5.1. Irudia: Burututako lanaren Gantt diagrama

5.1.2 Atazak noiz egin diren

Proiektuan zehar denbora desbideratzeak egon arren ezarritako data limiteak betetzen saiatu da, kasu batzuetan ordea ezinezkoa izan da.

2014ko urtarrilean praktikak egiteko aukera eskaini zidaten, otsailetik apirilera arte. Nire zeregina bakarra zen, Android aplikazio bat sortzea. Gauza berriak ikasteko aukera paregabea zenez onartu egin nuen.

Hasieran planifikatuta zegoen aplikazioa martxoaren bukaeran bukatzea. Memoriaren entregara arte denbora marjina handia zegoenez onartu nuen, gainera gauza berriak ikasten banituen, ondoren, nire aplikazioarekin jarraitzean, hauek aprobetxatu ahal izango nituen produktuaren kalitatea hobetzeko eta azkarrago lan egiteko.

5.1. irudiko diagraman gorriz ikus daitezke aldaketak jaso zituzten atazak. Planifikazioa luzatu zen, praktikak hasi baino lehen geratzen zen lana birplanifikatzea beharrezkoa izan zen.

Bidaien bilaketa atalaren amaiera atzeratu behar izan zen. Aplikazioak otsailerako bidaia zuzenak egiten zituen, baina oraindik falta zen konposatuak egitea, hauek apirilean bukatu ziren.

Interfazearen garapena luzatu zen kalitate handiagoko bat sortzeko. Praktiketan egindako aplikazioan garrantzi handia eman behar nion itxurari, hortaz interfazeei buruz asko ikasi nuen. Ezaguera berriak baliatuz GipuzkoaBidaian hobetu zen.

Ez dago esan beharrik baina probak, memoria eta defentsa atzeratu behar izan ziren.

5.2. Komunikazioak

Proiektuaren hasieran planifikatu zen bezala, gehien erabilitako komunikabidea mezu elektronikoa izan da. Hasieran, ez zen finkatu mezu kopuru minimo bat, ez baitzen beharrezkoa. Hortaz, espero zena bete da.

Tutorearekin beharrezkoa zenean bilerak egin dira, batzuetan dudak galdetzeko eta gehienetan proiektuari buruzko berriak emateko.

Beste interesatuei dagokionez, Dbus konpainia, proiektuaren egoeraz informatuta mantendu zaie. Horretarako erabili den komunikabidea mezu elektronikoa eta bilerak izan dira. Lehenengoa berri txikiez informatzeko erabili da eta bigarrena gauza garrantzitsuetarako, adibidez, paperak firmatzeko eta aplikazioa erakusteko.

Open Data webgunearekin komunikazioa soilik mezuz izan da, hasieran pentsatu zen bezala.

Ondorioz, esan daiteke proiektuaren hasieran planifikatutakoa bete dela, kalitatezko komunikazioak lortuz.

5.3. Kalitatea

Proiektuaren hasieran lau atal identifikatu ziren kalitate kontrola eduki behar zutenak, hauek baitira kalitatezko produktua lortzeko bete behar direnak.

5.3.1. Komunikazioa

Dagoeneko aurreko atalean azaldu da nola izan diren interesatuekiko komunikazioak. Hasierako minimoak bete direnez kalitatezko komunikazioa egon dela esan daiteke.

5.3.2. Produktua

Proiektuaren irismenean zehaztutako minimoak bete dira, ez hori bakarrik, hainbat arlotan hasieran definitutakoa hobetzea lortu da:

- **Bilaketa lokalak**: informazio iturria SD memoria txartelean gordetzen denez Internet erabili gabeko bilaketak egin ahal izango zirela planifikatu zen. Hau hobetzeko asmotan bilaketa osoak lokalki egin ahal izatea inplementatu da. Erakusten den informazioa kalitate berdinekoa ez izan arren larrialdi kasuetarako oso ondo etortzen da.
- **Interfazea**: praktiketan ikasitakoa aprobetxatuz aplikazioaren itxura hobetu zen, adibidez, atzealdea kolore batekoa izan ordez, tren geltoki baten irudi bat jarri zen. Gainera lausotuta jarri zen itxura hobetzeko.
- **Tableta eta mugikorra desberdindu**: Bi gailuen pantailak oso desberdinak direnez, interfaze batzuk bi aldiz inplementatu dira, aplikazioaren elementu guztiak ondo ikusteko edozein lekutan. Ez hori bakarrik, bidaia baten informazioa erakutsi behar denean, tableta bat erabiltzen ari dela ikustean, jarraibideen informazioa eta mapa aldi berean ikusten dira, erosoagoa izanez.
- **Erabiltzailearen aukerak**: Bezeroari malgutasun handiena ematekotan, bidaia zuzenen bilaketak soilik egiteko aukera ematen zaio.

Hala eta guztiz ere, dena ez da hasieran estimatutako kalitatekoa, aplikazioak egin beharrezko guztia egin arren, lortutako efizientzia maila ez da espero zena. Bilaketak egin ahal izateko informazio fitxategiak deskargatu behar direnez, banda-zabaleraren kontsumoa oso altua izan daiteke. WIFI-a erabiliz hau ez da arazo bat, are gehiago, aplikazioa mugatu daiteke soilik konexio horrekin funtzionatzera. Baina hori ez da bere arazo bakarra, bilaketa bakoitzean fitxategiak irakurri behar direnez, itxaronaldia luzatzen da, kasu batzuetan gehiegizkoa izanda.

Ondorioz, aplikazioak kalitate minimoa betetzen duela esan daiteke, baina ez maila altukoa dela. GipuzkoaBidaian-ek era optimoan funtzionatzeko bilaketa prozesua zerbitzari bitartekari batean egin beharko litzateke, gehigarrien “C Eranskina: GipuzkoaBidaian-en

funtzionamendua zerbitzari baten bidez" kapituluan zehaztasunez azaltzen da nola izango litzatekeen.

Azkenean aplikazioa Google Play-era ez igotzea erabaki da, publikoari zerbait eskaintzekotan kalitate onenekoa izan behar da. Horrez gain, ez daukadala kontu bat hori egin ahal izateko.

5.3.3. Memoria

Hasieran estimatu zen memoria kalitatezkoa izateko gutxienez 80 orri behar zituela, memoria honek kopuru hori gainditu du eta asko (totala gehiegi izan gabe). Gainera hirugarren pertsona baten laguntza izan da memoria irakurtzeko, ulergarria dela eta lexikoa zuzena dela ziurtatzeko.

Ez hori bakarrik, dokumentua tutorearen laguntzarekin osatu da, hortaz, kalitatezko memoria dela esan daiteke.

5.3.4. Defentsarako gardenkiak

Memoria kasuaren oso antzekoa. Hasieran ezarritakoa guztia bete da eta gainera hirugarren pertsona batek irakurri du ulergarritasuna bermatzeko.

5.4. Arriskuak

Proiektuaren hasieran kaltegarriak izan litezkeen hainbat egoera identifikatu ziren, hauei aurre egiteko plan batzuk zehaztu ziren. Zorionez, dena ondo bukatu da, kasu batzuetan ezer txarrik gertatu ez delako eta besteetan hartutako neurriak eraginkorrak izan direlako.

5.4.1. Konputagailua hondatzea

Hasieran hartutako neurriei esker ordenagailuari ez zaio ezer gertatu. Hainbat alditan gehiegi berotzeagatik Windows-eko pantaila urdina atera da, baina hala ere ez da ezer txarragorik gertatu.

5.4.2. Informazio iturrian aldaketak egitea

Momentu guztian informazio iturriak ez dira aldatu, hortaz ez da arazorik egon honi dagokionez.

5.4.3. Informazio galera

Noizbait fitxategiaren bat edo kode zati bat nahi gabe ezabatu izan da, baina hainbat segurtasun kopia egin direnez proiektu osoan ez da ondorio txarrik izan. Prebentzio planak oso ondo funtzionatu duela esan daiteke.

5.4.4. Aurreikuspenak ez betetzea

Praktikak eta aplikazioaren zaitasunak izan direla medio, proiekturen bukaera atzeratu behar izan da. Hala ere, hasieran denbora marjina handia zehaztu zenez ez da arazorik egon, proiektua bukatuta oraindik marjina dezentea geratu da.

5.4.5. Memoria eta defentsaren daten aldaketa

Hasieran jarri zen bezala, proiektua uztailaren hasieran entregatu behar da, hortaz ez da arazorik egon.

6

Ondorioak

Kapitulu honetan proiektuaren ondorioak azalduko dira, aplikazioa nola garatu den laburpena, zer ikasi den proiektu honekin, APP-a merkatura ateratzeko aukerak eta etorkizunean aplikazioak nola jarraituko lukeen.

Ondorioak kapituluaren egitura
1. Ondorioak 2. Ikasitako lezioak 3. Merkaturatze-perspektibak 4. Etorkizuneko lan-lerroak

6.1. Ondorioak

Proiektu honetan lan asko egin da, horri esker finkatutako helburuak neurri batean betetzea lortu da. Bi puntuen arteko garraio publikoko bidaiak bilatzen dituen APP-a sortzea lortu da, proiektuaren helburu nagusia betez.

Laburbilduz, aplikazioaren garapenaren hasieran interfazea diseinatu eta inplementatu zen. Momentu horretan Android-i buruzko neuzkan ezagutzak oso mugatuak ziren, batez ere, diseinuari buruz, gai hori ez baitzen jorratzen unibertsitatean.

Hurrengo urratsa bilaketak egitea zen, lehenengo Google Transit-ekin egin zen. Bilaketa modu horrekin lan gehiena eginda etortzen zen. Emaitzak jaso, gorde eta informazioa erakutsi besterik ez zen egin behar. Google-ek ematen duen emaitzaren formatua hain ongi egituratua dagonez, hori bera jarraitzea erabaki zen informazioa transmititzeko. Horregatik bilaketen emaitzak gordeko zituzten klaseak, "*BilaketaEmaitza*" paketean dauden klase guztiak sortu ziren.

Ondoren, lan garrantzitsuena eta zailena egin zen, bidaien bilaketa Open Data eta Dbus zerbitzarien informazioa erabiliz. Horretarako, lehenik, fitxategiak eskuratzeko eta hauek irakurtzeko kodea idatzi zen. Behin informazioa nola irakurtzen zen jakinda, bilaketak nola egingo ziren erabaki zen. Informazio fitxategien edukiek GTFS egitura dute, hori kontuan izanik bi geltokien arteko bidaia posibleak lortzeko era asmatu zen.

Bilaketak egiteko lehenengo jatorri eta helmugako geltokiak hartzen dira, bakoitzetik igarotzen diren bidaia guztiak (*TripsBean*) lortzen dira eta bakoitzari dagokion ibilbidea (*RoutesBean*) lortzen da. Orduan, jatorriko informazioa eta helmugakoa konparatzen dira ibilbide kointzidentziak bilatzeko. Bilaketa orduaren arabera izanez gero, helmugako geltokiaren ordua ere kontrolatzen da (linea bat zein ordutan pasatzen den hortik), ea finkatutako orduan baino goizago den.

Bilaketak egiteko beste era batzuk daude, adibidez, konpainia baten ibilbide guztiak lortuz hastea (bere lineak), ondoren bakoitzak egiten duen bidaia guztiak eskuratu eta azkenik bakoitzari dagokion geltokien artean jatorritik eta helmugatik igarotzen direnak lortu. Metodo hau ez da erabili motelagoa delako, erabiltzen ez den informazio gehiegi irakurtzen baita.

Azkenik bilaketen emaitza egokitzeko kodea sortu zen pantailan erakusteko.

GipuzkoaBidaian Android sistema erabiltzen duten gailuentzat sortutako aplikazioa da, gaur egun asko erabiltzen delako eta interes handia daukadalako teknologia horretan. Nire etorkizun profesionalari begiratuz, gai honi buruz ikastea ideia paregabea da, gero eta gehiago erabiltzen den zerbait delako.

Kontuan hartuta unibertsitatean ez dela gai honi buruz sakonki irakasten, proiektu hau garatu ahal izatea nire formazioarentzat lagungarria izan dela esan daiteke. Aplikazioa garatu ahal izateko ikasi behar ditudan kontzeptu berriak nire kabuz egin dut. Horrek proiektuaren garapena moteldu du puntu batzuetan, baina asko ikasteko lagundu dit.

Ez hori bakarrik, eskala handiko egiten dudan lehenengo proiektua denez eta gainera ia ezezaguna den teknologia batekin, esan behar dut proiektuaren emaitzak positiboak izan direla, ikasi nahi dudan teknologia bati buruz asko sakondu dut, proiektu handi bat egitearen zailtasunak ezagutu ditut eta mugikorren merkatuari buruz asko ikasi dut.

6.2. Ikasitako lezioak

Proiektuan zehar nire etorkizunerako baliagarri diren lezio batzuk ikasi ditut, ondoren zerrendatzen dira:

6.2.1. Proiektu handi baten planifikazioaren garrantzia

Unibertsitatean, proiektuen kudeaketa ikasgaian, planifikazioaren garrantzia ikasi nuen, baina honelako proiektu handi bat egin arte ez naiz konturatu zein beharrezkoa den. Hasieran, proiektuaren atazak ondo zehazteak, noizko eginak egon behar ziren definitzen asko lagundu dit.

Atazen datak zehaztea oso lagungarria izan zait, hainbat ataletan arazoak izan arren, hasieran estimatutako ordu kopurua betetzea ezinezkoa izan arre, datak betetzea lortu dut, eta hori esker proiektua arazorik gabe bukatzea lortu dut.

6.2.2. Arriskuak garaiz identifikatzearen garrantzia

Proiektuaren hasieran arriskuak ondo definitu behar dira, are gehiago proiektu handiei buruz hitz egiterakoan. Ez hori bakarrik, prebentzio plan bat edukitzea ere funtsezkoa da etorkizuneko arazo asko ekiditeko.

Horri esker proiektuaren garapena arazorik gabe joan da, ez da denbora galdu zuzenketetan eta denborarekin bukatu ahal izan da.

6.2.3. Diseinuaren garrantzia aplikazioetan

Hasieratik banekien APP-ak diseinu erakargarri bat eduki behar dutela, publikoa eroso sentitu behar dela erabiltzean, baina ez nuen uste hainbesterako izango zenik. Android erabiltzen duten hainbat gailu existitzen dira, hainbeste pantaila tamaina daude, Android bertsio zaharrak eta berriak hain desberdinak dira, oso zaila dela gailu guztientzat interfaze berdina eta ona egitea.

Mugikorrentzako aplikazio bat garatu nahi denean lehenik ondo finkatu behar da zein publikotara bideratuta dagoen, hau da, APP-ak onartuko duen Android bertsioa. Horren arabera diseinu bat edo bestea egingo da, lan gehiago edo gutxiago suposatuko duelarik aplikazioa sortzeak.

Ikasitako guztiarekin badakit hurrengo Android-rako proiektua egiteko urrats berezi batzuk jarraitu beharko ditudala:

1. Definitu Android bertsio minimoa.
2. Momentuko modak aztertu (jakiteko erabiltzaileak zertara ohituta dauden).
3. Marraztu paperean.
4. Zirriborroak erakutsi.
5. Garatu eta erakutsi.

6.2.4. Zerbitzari - Mugikor erlazioa

Mugikorrentzat sortutako aplikazioak prozesamenduko lan gutxi egin behar zutela ikasi dut proiektu honetan. Garapenean zehar konturatu nintzen aplikazioak web-orriak bezala direla, hauek lan minimoa egin behar dute ahal denean. Argi dago ez dela beti posible izango, baina ahal denean lortu behar da.

Erabiltzaileak gailu mugikor batean APP bat erabiltzean ez dira zain gelditukо, dagoeneko ohituta daude erantzun azkarreko aplikazioak erabiltzera. Jendeak gero eta denbora gutxiago du bere gauzetarako, horregatik aplikazioak azkarrak eta intuitiboak izan behar dira, denbora ez xahutzeko erabileran.

GipuzkoaBidaian motako aplikazioa, non lan gehiena informazioaren prozesamenduan eta fitxategien irakurketan dago, soilik emaitzak erakusteaz arduratu beharko litzateke azkarra izateko, lan gogorra hirugarren makina bati delegatuz.

Etorkizunean mota honetako proiektu bat egitekotan, hasi baino lehen, ondo definituko nuke egitura, jakiteko produktua sortzearen esfortzua mereziko lukeen ala ez. Zerbitzari bat erabili beharko balitz, lan gehiago izateaz gain diru inbertsioa egin beharko litzateke.

6.3. Merkaturatze-perspektibak

Hasiera batean pentsatu zen aplikazioa bukatzerakoan Google Play Store-ra igotzea, horretarako izen erakargarri bat asmatu nuen aplikaziorako, GipuzkoaBidaian, eta logoa sortu nuen.

Aplikazioa bukatzean erabaki nuen momentuz ezin zela kaleratu. Funtzionatzen duen arren, ez da izan beharko litzatekeen bezain efizientea. Are gehiago, aplikazioa bukatu

nuenerako, Google Maps aplikazioa eguneratu izan zen (edo hainbat aldiz egin zuen, ez dut kontatu), eta hauetako batean bere bidaia bilatzailea asko hobetu zuten.

Ondorioz, Internetera oraingoz ez igotzea erabaki dut. Aplikazio bat publikatzeko, ez du soilik funtzionatu behar, bezeroarentzat perfektu egon behar da. GipuzkoaBidaian polita eta erabilgarria izan arren ez luke arrakastarik izango, hortaz esperientzia bezala oso baliagarria izan zait, baina ez publikoaren eskura jartzeko.

6.4. Etorkizuneko lan-lerroak

Aurreko atalean azaldu den bezala, aplikazioa ez dago prest publikori eskaintzeko, lan asko gelditzen da oraindik. Horretarako, lehentasun handieneko betebeharra, bilaketa prozesua aldatzea izango litzateke. Horrek ez du esan nahi gaizki dagoela, "C Eranskina: GipuzkoaBidaian-en funtzionamendua zerbitzari baten bidez" kapituluan azaltzen den bezala, lan hori zerbitzari bitartekari batean egin beharko litzateke. Hortaz, aplikazioa etorkizunean hobetzekotan, ildo horretatik jarraitu beharko litzateke.

Interfazean ere lan egin beharko litzateke. Bidaiei buruz erakusten den informazio kopurua handitu beharko litzateke, adibidez, mapa ibilbidearekin erakustean jarraibideak erakuts daitezke. Baita ibilbidearen puntu batean ukitzean informazioa erakustea ere.

Aplikazioari nahi adina hobekuntza egin diezaioke, baina beti kontrolatuz bere tamaina handiegia ez dela. Gogoratu behar da mugikorren aplikazioak arinak izan behar direla, ez soilik hauen memoria ez delako hain handia, baizik eta gainkargatu egin daitezkeelako.

Bibliografia

Argitalpenak

[1] Jesus Tomas Girones: El gran libro de Android 3º edición, 2013.

[2] Jesus Tomas Girones: El gran libro de Android Avanzado 1º edición, 2014.

Webguneak

[3] Android Dvelopers webgune ofiziala:
http://developer.android.com/

[4] Open Data:
http://opendata.euskadi.net/w79-home/es/

[5] Open Data FTP zerbitzaria (Konpainien fitxategien iturria):
ftp://ftp.geo.euskadi.net/cartografia/Transporte/Moveuskadi/

[6] Google Developers. GTFS formatua:
https://developers.google.com/transit/gtfs/reference

[7] Google Developres. Google Maps API Android-en:
https://developers.google.com/maps/documentation/android

[8] Google Developers. Google Directions API:
https://developers.google.com/maps/documentation/directions/

[9] Genymotion. Android emuladorea:
http://www.genymotion.com/

[10] Google Developers. Autocomplete API-a:
https://developers.google.com/places/documentation/autocomplete?hl=es

[11] stackoverflow (programazio dudetarako):
http://stackoverflow.com/

[12] stackoverflow (Android-i buruzko zalatzetarako):
http://android.stackexchange.com/

[13] Apache Commons Compress (erabilitako deskromprimatzailearen liburutegia:
http://commons.apache.org/proper/commons-compress/

[14] Super CSV (CSV fitxategiak irakurtzeko liburutegia):
http://supercsv.sourceforge.net/index.html

[15] Google Plus. Genymotion-en mezu publikoa (Googe Services arazoari buruz):
https://plus.google.com/+GenymotionEmulator/posts/jNF8Kwu5p1c

[16] Google Developers. Geocoding API (koordenata batetik helbidea lortzeko eta alderantziz):
https://developers.google.com/maps/documentation/geocoding/

[17] Apache Commons. Apache Commons Net liburutegia (FTP bezeroa erabiltzeko):
http://commons.apache.org/proper/commons-net/

[18] University of Wisconsin Green Bay. Converting UTM to Latitude and Longitude:
http://www.uwgb.edu/dutchs/usefuldata/utmformulas.htm

[18] Kantar Worldpanel, kontsumitzaileen ezagutza eta ulermen adituak:
http://www.kantarworldpanel.com

Akronimoak

- ADT: Android Development Tools
- API: Application Programming Interface
- APK: Application Package File
- APP: Application Program
- AVD: Android Virtual Device
- CSV: Comma-Separated Values
- DOM: Document Object Model
- FTP: File Transfer Protocol
- GIMP: GNU Imagen Manipulation Program
- GNU: *GNU's Not Unix!*
- GPS: Global Positioning System
- GTFS: General Transit Feed Specification
- HTML: HyperText Markup Language
- HTTP: Hypertext Transfer Protocol
- JSON: JavaScript Object Notation
- JSP: JavaServer Pages
- LDE: Lanaren Deskonposaketa Eskema
- PHP: PHP Hypertext Pre-processor
- RIM: Research In Motion
- SD: Secure Digital
- SDK: Software Development Kit
- SFTP: Secure File Transfer Protocol (SSH File Transfer Protocol)
- SGML: Standard Generalized Markup Language
- SSL: Secure Sockets Layer
- SQL: Structured Query Language
- UML: Unified Modeling Language
- URL: Uniform Resource Locator
- USB: Universal Serial Bus
- UTM: Universal Transverse Mercator
- VM: Virtual Machine
- W3C: World Wide Web Consortium
- XML: eXtensible Markup Language

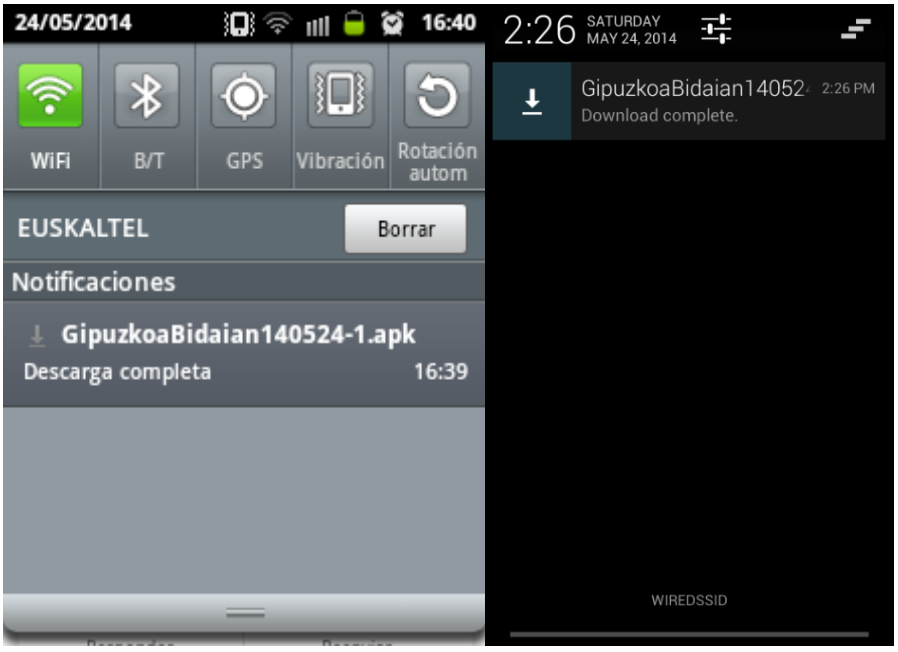
Eranskinak

A Eranskina: GipuzkoaBidaian instalatzeko eskuliburua

Android aplikazioak bi eratan instalatu daitezke smartphone eta tabletetan, aplikazio espezializatu baten bitartez edo eskuz.

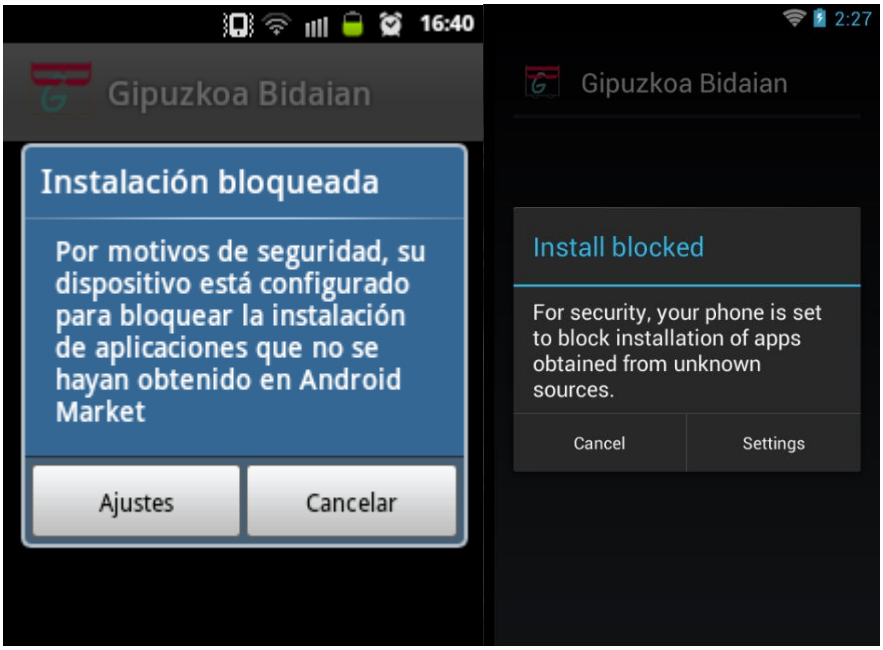
Lehenengo eran egiteko kontu berezi bat eduki behar da. Adibidez, Google Play Developer Console webgunean Google Play Store-ko aplikazioak kudeatzen dira. Gmail-eko posta elektronikoarekin edozein sar daiteke, baina aplikazioak argitaratzeko urtero kuota bat ordaindu behar da. Ondorioz, metodo hau ez da erabiliko instalazioa azaltzeko.

Bigarren era konplikatuagoa da, baina bestea bezain eraginkorra. Lehenik aplikazioaren instalatzailea, APK-a, deskargatu behar da, adibidez Google Drive-etik. Aplikazioa jaisterakoan sistemak abisatuko du fitxategia mugikorraren memorian dagoela, A.1. irudian ikus daitekeen bezala.



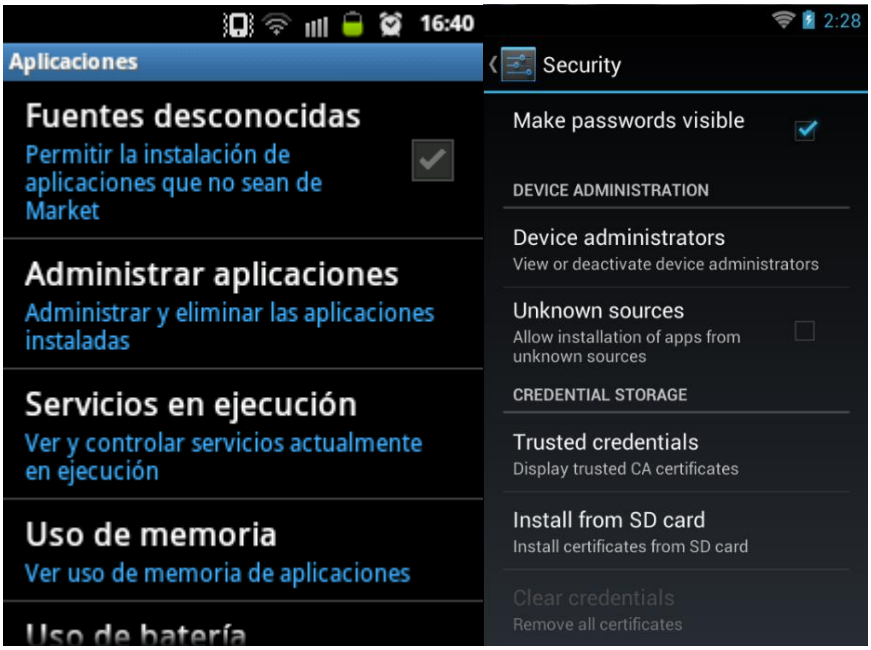
A.1. irudia: APK deskargatu den mezua, ezkerrean Android 2.3 bertsioa eta eskuinean 4.2

Deskarga osoa egin dela adierazten duen abisua ukitzean instalazioari eraman beharko luke, baina normalean sistemak jatorri ezezagunetik datozen aplikazioen blokeoa aktibatuta dauka, horrela izanez gero A.2. irudiko mezua agertuko da pantailan.



A.2. irudia: Segurtasun mezua, ezkerrean Android 2.3 bertsioa eta eskuinean 4.2

Doikuntzak botoia sakatuz doikuntza menura irekiko da, non iturri ezezaguneko aplikazioak onartzeko egiaztapen laukia dagoen. Gailuaren Android bertsioaren arabera hau leku batean edo bestean aurkituko da, A.3. irudiak erakusten duen moduan. 2.3 bertsioan adibidez, “Aplikazioak” atalean dago, 4.2 atalean ordea, “Segurtasun” atalean.

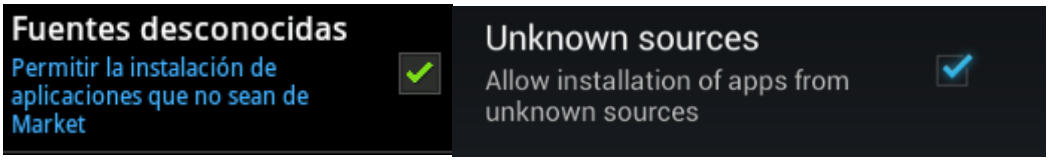


A.3. irudia: Iturri ezezaguneko aplikazioak onartzeko menua, ezkerrean Android 2.3 bertsioa eta eskuinean 4.2

Egiaztapen laukia sakatzean arriskuaz abisatzen duen mezu bat agertuko da, A.4. irudian agertzen dena. Onartzeko ematen zaio eta listo.

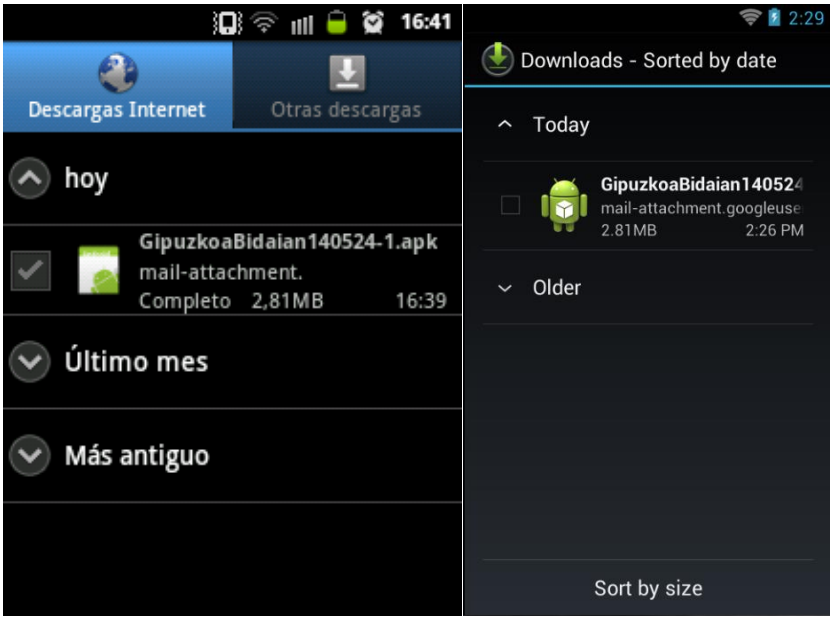
A.4. irudia: Arriskuaz abisatzen duen menua, ezkerrean Android 2.3 bertsioa eta eskuinean 4.2

Momentu horretan egiaztapen laukia aktibatuko da.



A.5. irudia: Egiaztapen laukia aktibatua, ezkerrean Android 2.3 bertsioa eta eskuinean 4.2

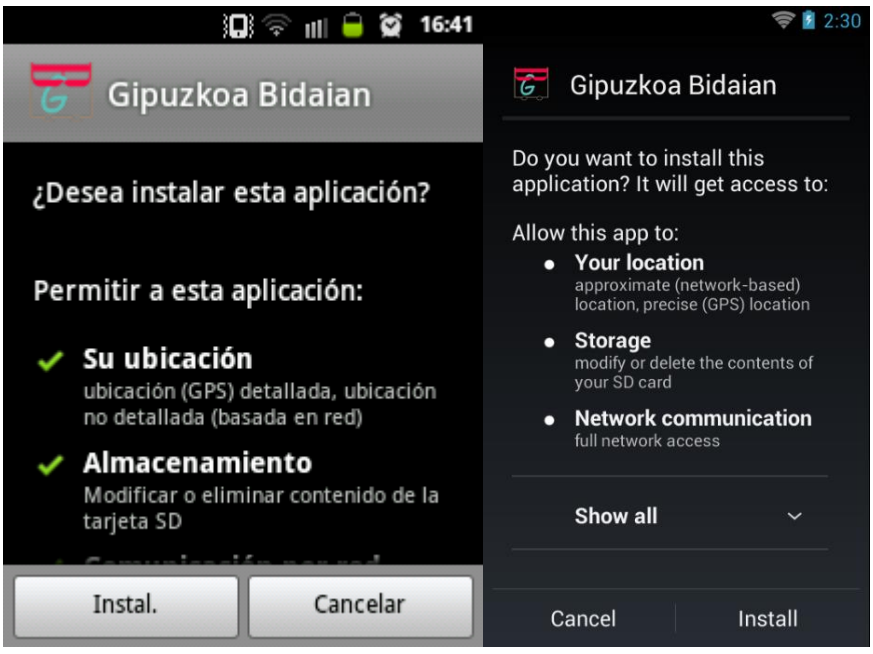
Orduan aplikazioa instalatzeko prest dago. Hau lehen deskargatu denez “Deskargak” menura joanez gero instalatzailea aurkituko da. A.6. irudian ikus daiteke deskargatutako elementuen zerrenda.



A.6. irudia: Deskargak menua, ezkerrean Android 2.3 bertsioa eta eskuinean 4.2.

APK fitxategia ukituz instalazioko lehenengo urratsa agertu da. A.7. irudian ikus daitekeen bezala, sistemak erakusten du aplikazioari onartuko zaizkion baimenak. GipuzkoaBidaian-ek hurrengoa behar du:

- **Erabiltzailearen kokapena**. Helbideak mapatik lortzeko beharrezkoa da.
- **Biltegiratzea**. Garraio publiko konpainien informazioa SD txartelean gordetzen da.
- **Sare komunikazioa**. Aplikazioa Internetera konektatu behar da.
- **Garapen tresna**. Mapak erabiltzeko Google-en zerbitzuen konfigurazioa irakurri behar da.

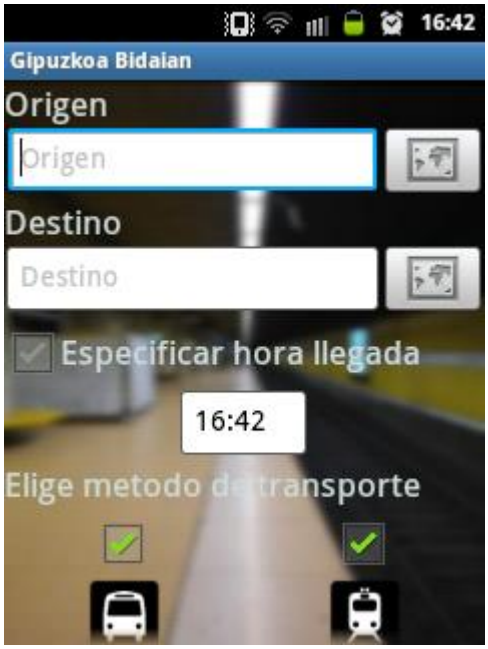


A.7. irudia: Instalazioaren lehenengo urratsa, ezkerrean Android 2.3 bertsioa eta eskuinean 4.2

Instalatzeko sakatzean GipuzkoaBidaian instalatuko du, aplikazioa txikia denez ez zaio denbora gehiegi kostatuko. Behin instalatuta aplikazioaren ikonoa ikusiko da menu nagusian. Prozesu hau Android bertsio berrian edo zaharrean eginda, ikusiko dena A.8. irudiaren oso antzekoa da.

A.8. irudia: Goi ezkerretik, instalazio prozesua, bukaera eta aplikazioaren ikonoa menu nagusian

Azkenik, GipuzkoaBidaian-en ikonoa sakatzen da exekutatzeko eta ziurtatzeko ondo instalatu dela. A.9. irudian ikus daiteke aplikazioaren lehenengo sarrera.

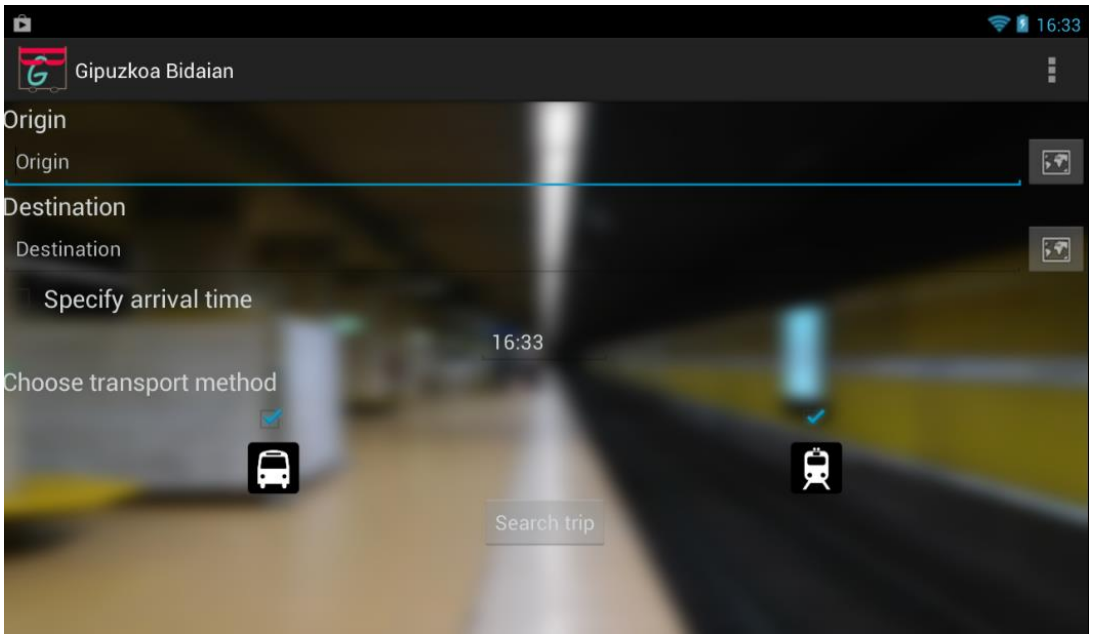


A.9. irudia: Aplikazioaren lehenengo exekuzioa

B Eranskina: GipuzkoaBidaian erabilpen eskuliburua

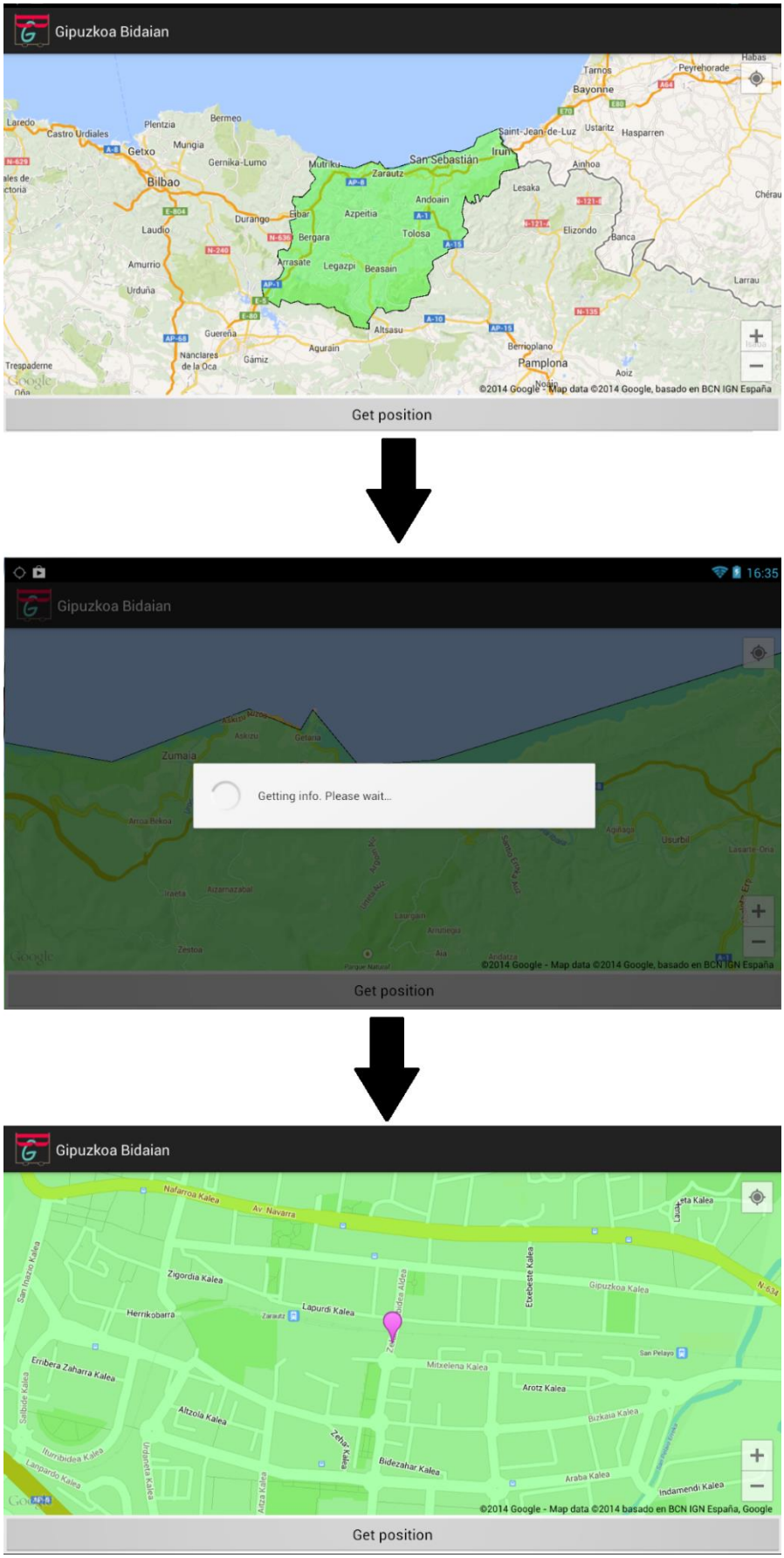
Kapitulu honetan aplikazio nola erabiltzen den azalduko da, horretarako exekuzio baten irudiak erakutsiko dira, ulergarriagoa egiteko. Eskuliburua sortzeko makina birtuala erabili da, zehazki, aplikazioa tableta batean exekutatu da. Aplikazioa ingelesez exekutatu da, sistemaren lehenetsitako hizkuntza delako.

GipuzkoaBidaian exekutatzean B.1. irudiko pantaila agertuko da, jatorri eta helmuga testu eremuak hutsik agertuko dira (bakoitzak bete behar denaren aztarna “idatzita” izango du). Erabiltzen den lehenengo aldia denez ordua zehazteko egiaztapen laukia ez da egongo aukeratua eta bi garraio motak aukeratuta agertuko dira.



B.1. irudia: GipuzkoaBidaian exekutatzea agertzen den lehenengo pantaila

Jatorriko eta helmugako helbidea zehaztea beharrezkoa da bilaketa bat egiteko. Erabiltzaileek kale guztien izenak ezagutzen ez dituztenez mapa batez bidez lortzeko aukera dute. Adibidez, jatorria lortzeko, lehenengo testu eremuaren alboan dagoen botoia sakatu behar da B.2. irudiko lehen pantailara ailegatzeko.



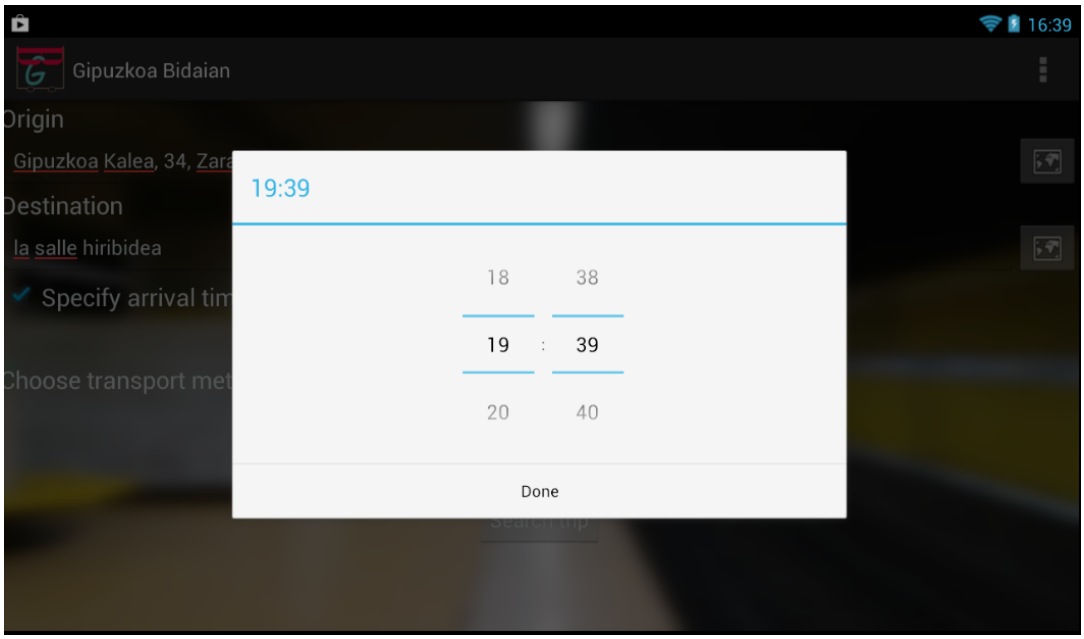
B.2. irudia: Mapa kargatu, mapa ukitu eta aukeratutako puntuan seinalea jarri

Orduan mapa bat kargatuko da pantailan Gipuzkoako eremua nabarmenduz. GPS-a aktibatuta egongo balitz, mapa uneko posiziora mugituko litzateke eta zooma egingo luke. Erabiltzaileak mapatik nabigatu eta nahi duen helbidea aurkitu ondoren pantaila ukituko du. Momentu horretan sistemak koordenada hartuko du eta helbide batean itzuliko du, hori egiten duen bitartean mezu bat agertuko zaio erabiltzaileari itxaroteko abisatuz. Azkenik, mapan seinale bat agertuko da, erabiltzaileak zuzena dela uste badu botoia sakatzen du eta aurreko interfazera bueltatuko da. Prozesu hau B.2. irudiak irudikatzen du hiru pausutan.

Hasierako interfazera itzultzean jatorriaren testu eremua beteta egongo da helbide berriarekin. Orain suposatuko da erabiltzaileak helmugako helbidea ezagutzen duela, hortaz, testu eremua beteko du.

Ondoren, bilaketaren parametro orokorrenak zehaztuko dira. Nahi izanez gero bilaketa ailegatze ordu baten arabera egin daiteke, horretarako egiaztapen laukia aktibatu behar da eta ordua duen testu eremua sakatu. Momentu horretan ordua eta minuta aukeratzeko leihoa agertuko da, era horretan ezinezkoa izango da gaizki idaztea.

B.3. irudian ikus daiteke azaldutakoa.



B.3. irudia: Ordua aukeratzeko leihoa eta atzealdean jatorria zehaztua eta helmuga idatzia

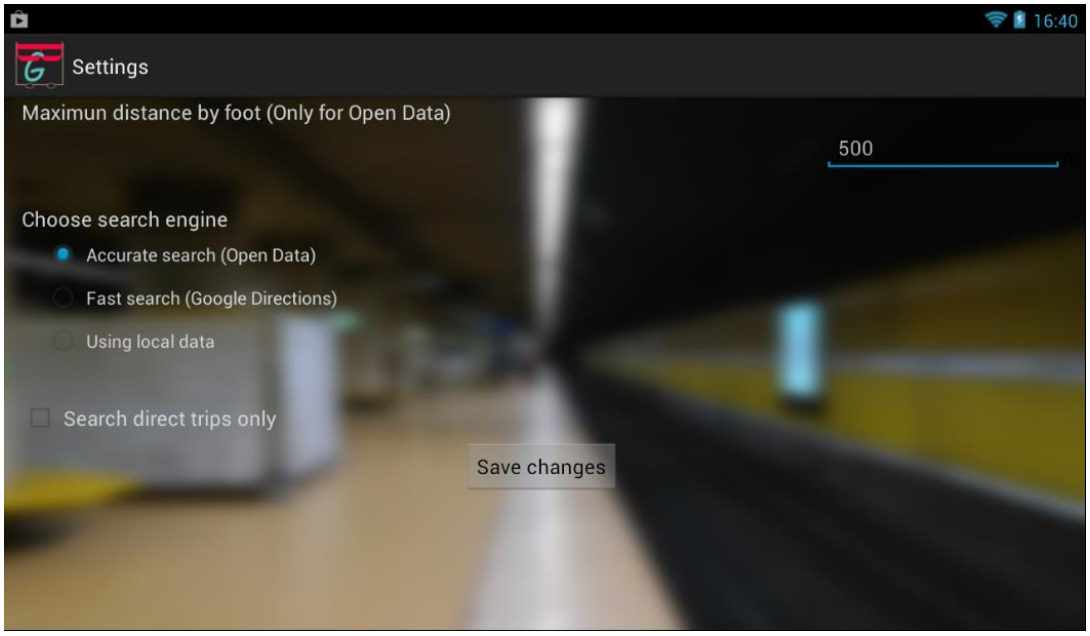
Nahi izanez gero bilaketarako parametro gehiago zehaztu daitezke, horretarako *Menu* botoia sakatu behar da eta *Settings* (Doikuntzak / Ajustes) aukeratu. Orduan interfaze berria agertuko da, non hurrengoa konfiguratu daitekeen:

- Oinezko distantzia maximoa (oinezko tarte bakoitzaren distantzia maximo erlatiboa).
- Bilaketa modua:
    - Zehatza: Open Data eta Dbus zerbitzarien informazioa erabiliz.
    - Azkarra: Google Directions API-a erabiliz.

- Lokala: SD txartelean biltegiratuta dagoen Open Data eta Dbus zerbitzarien informazioa erabiliz.

- Soilik bidaia zuzenak bilatu behar diren ala ez.

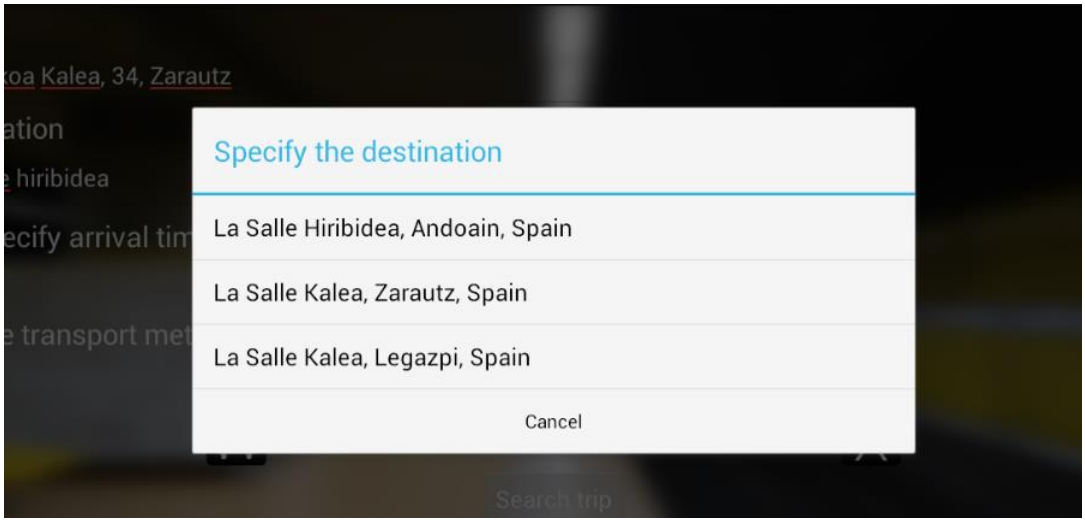
Lehenengo aldian oinezko distantzia 1500 izango da, bilaketa modua zehatza eta egiaztapen laukia ez da aukeratua egongo. Hirugarren bilaketa modua aukeratzeko, bilaketa zehatz bat egin behar da lehenik gutxienez informazioa biltegiratzeko.



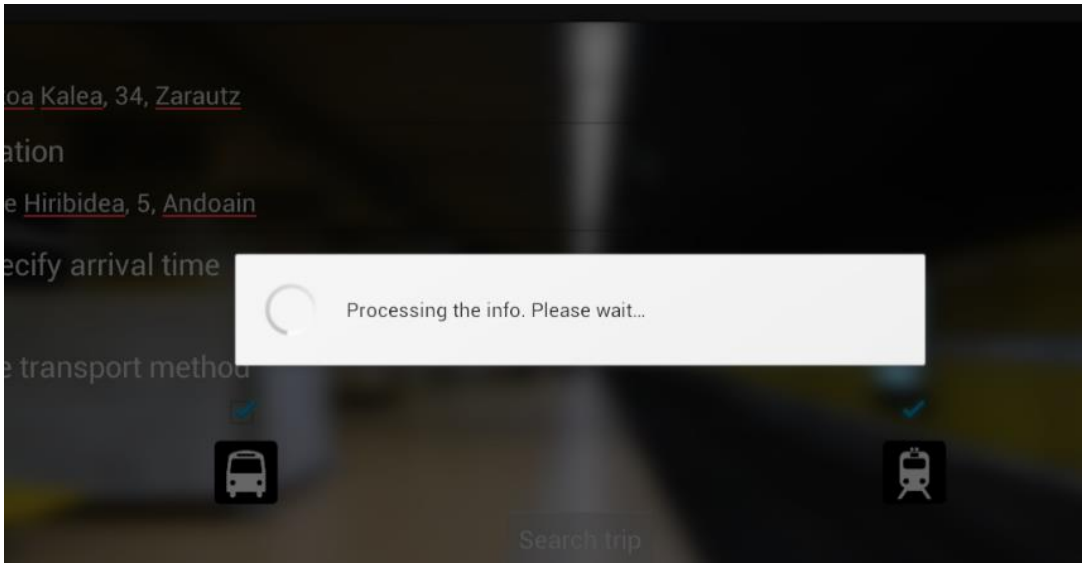
B.4. irudia: Doikuntzak interfazea

Erabiltzaileak parametroak zehazten ditu, B.4. irudian agertzen den bezala adibidez, eta gordetzeko botoia sakatzen du, hasierako pantailara bueltatuz.

Bilaketa egiteko parametro guztiak prest daudenean B.1. irudiko “Search trips” botoia sakatzen da, orduan sistema konturatuko da helmugako helbidea ez dela guztiz definitu. Idatzitako helbidea oso zehatza ez bada leiho berri bat agertuko da hainbat posibilitateekin, B.5. irudian agertzen den bezala. Erabiltzaileak bat aukeratuko du eta bilaketarekin hasiko da, B.6. irudiko mezua erakutsiz.



B.5. irudia: Helmuga zehazteko leihoa

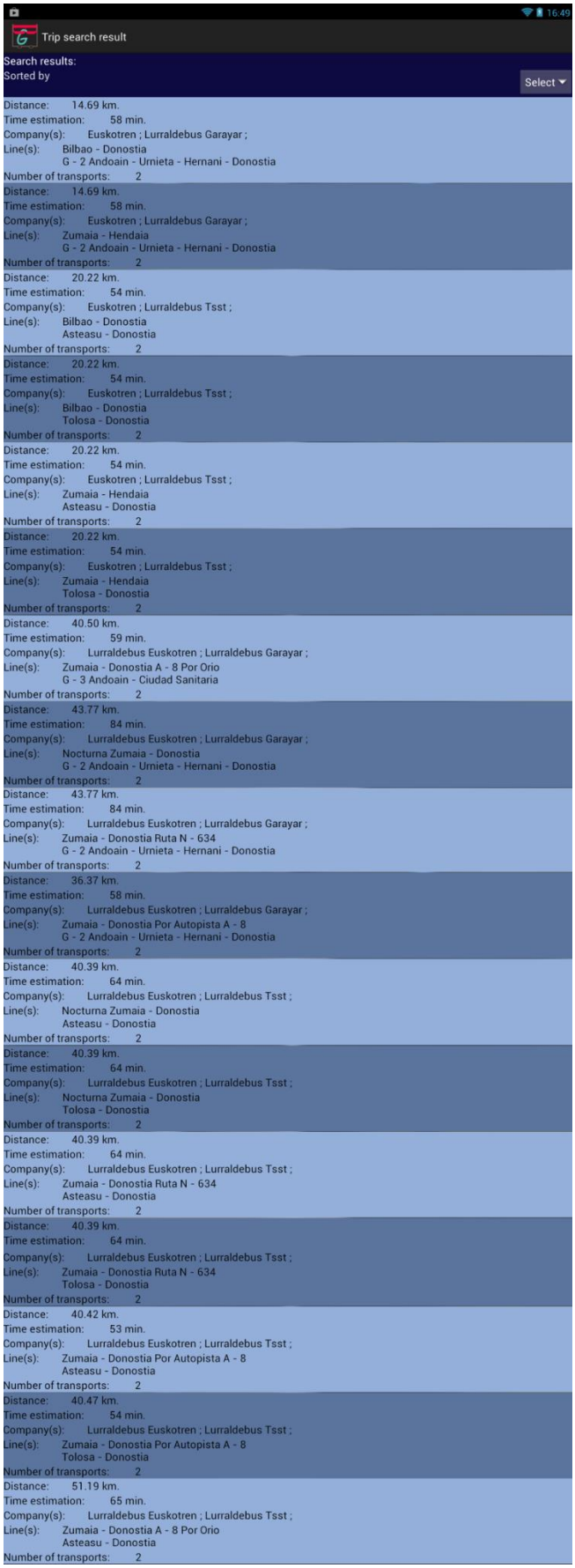


B.6. irudia: Bilaketa egitean agertzen den mezua

Bilaketa bukatzean hurrengo pantaila kargatuko da emaitzekin. Bidaia zuzenak soilik bilatzeko ez zaionez esan emaitza asko lortu ditu, B.7. irudian ikus daitezke exekuzio honetan agertu diren emaitza guztiak lista batean.

Zerrendako elementu bakoitzak bost datu erakusten ditu:

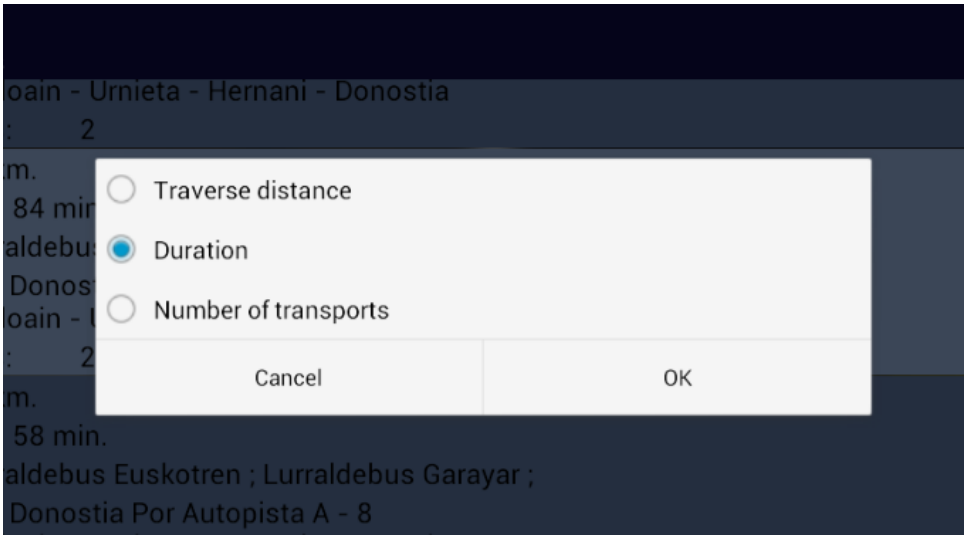
- Bidaiaren distantzia totala.
- Bidaiaren iraupen totala.
- Bidaia osatzen duten konpainiak.
- Bidaia egiteko hartu behar diren lineen izenak.
- Garraio kopurua.



B.7. irudia: Bilaketa egitean agertzen den mezua

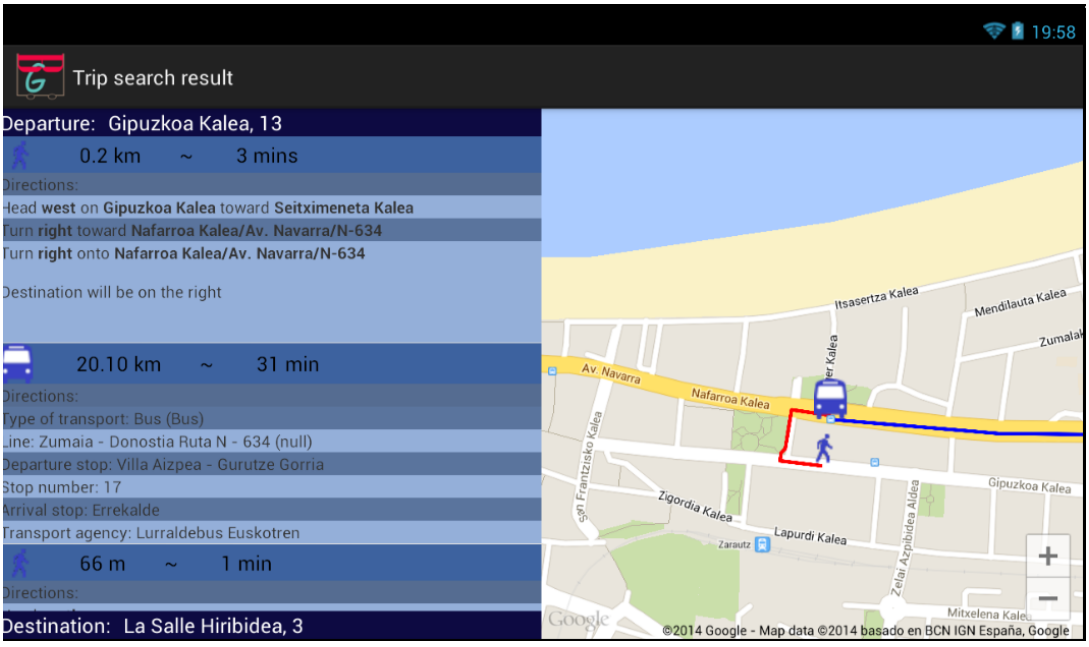
Hainbeste emaitza edukita erabiltzailea nahas daiteke, horregatik emaitzak ordenatzeko aukera dago. Horretarako interfazean dagoen botoi bakarra sakatzen da. Orduan leiho berri bat agertuko da ordenatzeko hiru aukerarekin, B.8. irudian bezala:

- Distantziaren arabera.
- Denboraren arabera.
- Garraio kopuruaren arabera.



B.8. irudia: emaitzak ordenatzeko leihoa

Azkenik, emaitzetako bat aukeratzean horren informazio osoa ikusiko da interfaze berri batean. B.9. irudian ikus daiteke emaitza baten adibidea.



B.9. irudia: emaitzak baten informazioa

C Eranskina: GipuzkoaBidaian-en funtzionamendua zerbitzari baten bidez

Garatutako aplikazioa independenteki funtzionatzeko prestatu da. Honek bere abantailak eta desabantailak ditu. Hemen azalduko da bere desabantaila nagusiaren konponbidea.

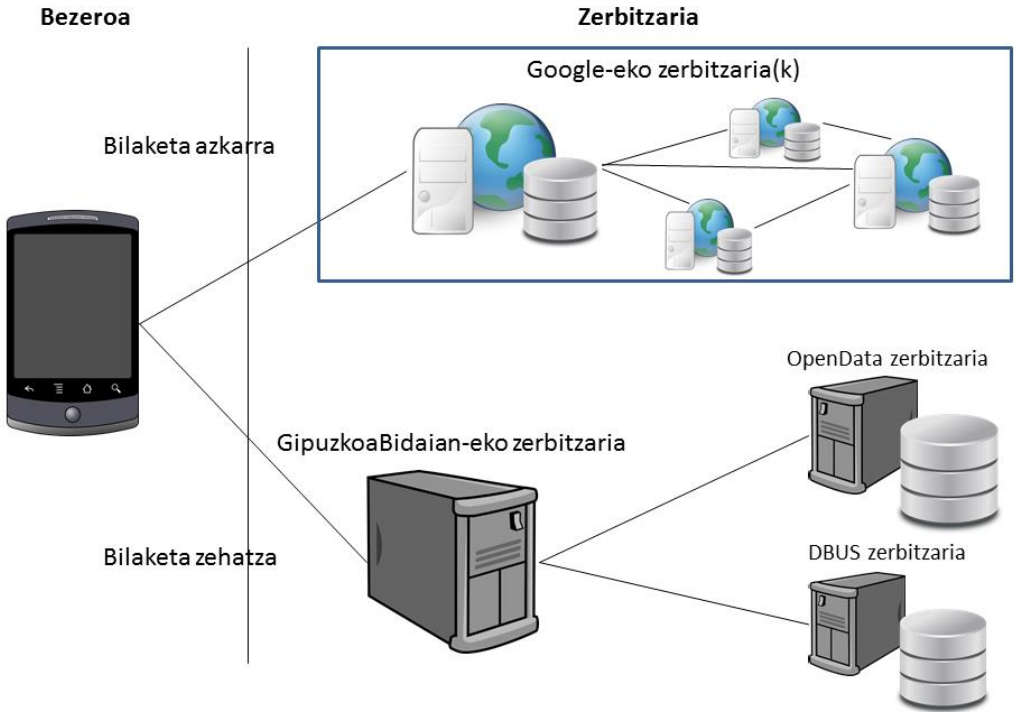
GipuzkoaBidaian-ek bitartekaririk erabiltzen ez duenez, informazioaren lorpena eta prozesamendua drastikoki moteltzen da. Mugikorrak ahalmen handikoak izan arren, oraindik ez dira kapaz datu kopuru handiegiak prozesatzeko. Hortaz, lan karga nagusia makina laguntzaile batera eraman behar da mugikorrena gutxitzeko eta azkarragoa izateko.

Soluzioa oso sinplea da, bidaien bilaketa hirugarren batek egingo du. Smartphoneak zerbitzari bitartekari bati bilaketaren parametroak bidaltzen dizkio, honek jaso eta barruan izango duen datu base batean bilaketa gauzatzen du (lehenik Open Data eta Dbus zerbitzariei konektatu delako informazioa lortzeko), ondoren emaitza bidaltzen dio mugikor eskatzaileari eta azkenik informazioa egokitzen da eta pantailan erakusten da.

Metodo hau jarraituz denbora gehien kontsumitzen duen ekintza, bilaketa, potentzia handiko zerbitzari batek egingo luke, mugikorrak bakarrik eskaeraz eta emaitza erakusteaz arduratuz. Era honetan bilaketa bat gauzatzeko beharrezkoa den denbora asko murriztuko litzateke.

Konponbide hau Open Data eta Dbus zerbitzariekin lan egiten den kasurako da, C.1. irudian ikusten den moduan. Datu eskuraketa eta prozesamendua oraintxe bertan Google-ekin lan egiten den kasuan bezala izango litzateke. Hau lortzeko hiru geruzetako estrategia jarraituko litzateke, ondoren hobeto azalduko dena.

Aplikazioaren klase nagusiak, *MainActivity*, soilik informazioa eskatu eta jasoko luke. Emaitza zuzena dela egiaztatzean erakusteko egokituko luke (ezin baita zuzenean erakutsi zerbitzaritik lortutakoa) eta informazioa pantailan erakusteaz arduratzen den klaseari informazioa bidaliko lioke.



C.1. irudia: Bezeroen eta zerbitzarien arteko erlazio zuzena

C.1. Zerbitzariaren egitura eta funtzionamendua

Zerbitzari bat erabiliz gero, egitura asko aldatuko balitz ere, informazioa prozesatzeko era oso antzekoa izango litzateke. Aplikazioa orain egina dagoen ez bezala, non prozesu guztiak "aldi berean" exekutatzen diren, zerbitzariaren lana hiru ataletan banatuko litzateke:

- Informazio eskuraketa: Open Data eta Dbus-eko zerbitzarietara nola eta noiz konektatu informazio guztia lortzeko
- Informazio eskuraketa eta bilketa: datu berriak lortzean datu basea eguneratu.
- Informazioaren prozesamendua: bilaketa egitea.

C.1.1. Bezero - Zerbitzari konexioa

2013. urtean merkatuan mugikorrentzako 28 bat sistema eragile desberdin zeuden. Nahiz eta hauetako gehienak ezezagunak izan eta ez diren erabiltzen, hor daude eta kontuan hartu behar dira. Zerbitzari batek eskaintzen duen zerbitzua kalitatezkoa izan behar da eta edozein publikori eskuragarri egon behar da. Honek arazo bat planteatzen du, zein lengoaiatan programatu zerbitzaria? Aplikazioa nola konektatu ahal izango da eta nola jasoko du informazioa?

Kurtso honetan, praktiketan egon naizen denboran, sare sozial bat sortzen ari ginen eta antzeko arazoa izan genuen. Hasierako urratsak egin eta gero (erabilpen kasuen analisia, ereduen diagramak, etab.) zerbitzaria zein lengoaiatan idatziko zen aukeratu behar zen. Garbi genuen, mugikorrentzako zerbitzua bigarren maiakoa izanda, zerbitzaria nagusiki webgunera bideratuta egongo zela. Hala ere, zerbitzariak zerbitzu bera eskaini behar zuen. Ezin genuen sistema bakoitzarentzat inplementazio desberdin bat garatu, hori sistema ez eraginkorra sortzea izango litzatekeelako.

Leku askotan eta jende askori galdetu genion eta gehienek antzekoa erantzun ziguten, ez dagoela sistema bat bestea baino askoz hobeagoa denik. Ondorioz, konklusio batera iritsi ginen: zerbitzaria, programatzaileak hobeto kontrolatzen duen lengoaian egon behar zuen, ezezagun den lengoaia batekin lan egiteak denbora atzerapena ekar zezakeelako, bai sistema bukatzeko epean eta baita bere errendimenduan ere. Ez hori bakarrik, programatzaileek erosoago lan egiten badute produktibitatea handitzen da.

Esperientziaren horren ondoren, zerbitzaria, segur aski, Java lengoaian egitea erabakiko nuke, gehien ezagutzen dudan lengoaia delako eta gustukoen dudalako.

Dena den, arazo bat izango nuke, aplikazioa beste sistema eragile batean eginez gero, konektatu ahal izango litzateke zerbitzarira? Datuak forma egokian jaso ahal izango lirateke? Android-en kasuan ez zen arazorik egongo, biak Java-n programatuta egingo lirateke eta.

Zailtasun honen aurrean konponbide orokor bat erabiliko nuke, PHP. Lengoaia honetako zerbitzari bat izanda aplikazioak egin beharko luketen konexioa oso sinplea izango litzateke, http dei bat beharrezko parametroak bidaliz.

Datuak jasotzeko beste hainbeste, PHP-k JSON edo XML formatua erabiliko luke (oso generikoak dira eta errazak ondoren interpretatzeko). Java-ko zerbitzariak ordea, bere formatuan propioan bueltatuko lituzke datuak, ondorengo lana zailduz.

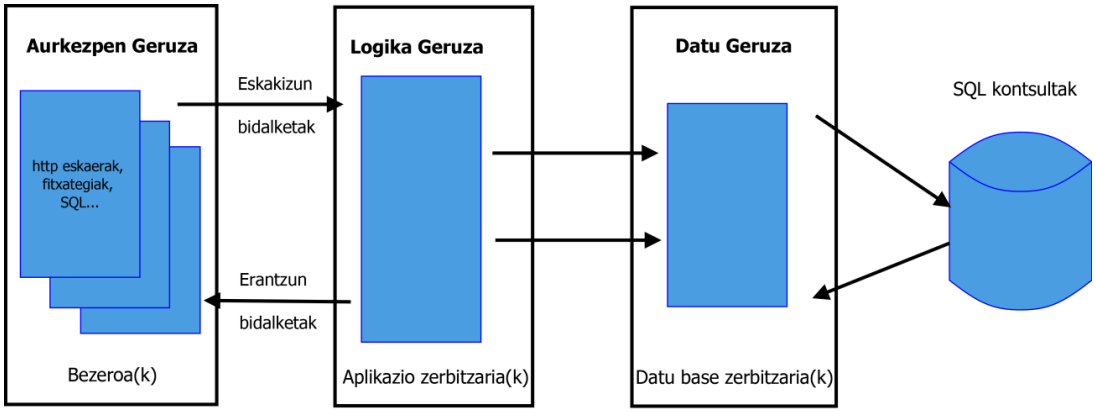
Zerbitzaritik informazioa JSON/XML formatuan jasotzeak, zerbitzuak sistema desberdinetan erabiltzeak ahalbidetzen du. GipuzkoaBidaian smartphonentzat bideratuta egon arren, ideia hau garatuz, aplikazioa edozein plataformarako garatu ahal izango litzateke.

Funtsean, zerbitzaria edozein lengoaian, Java edo PHP, inplementatzea ideia ona izango litzateke, biak bere abantailak dauzkatelako. Hori dela eta hirugarren aukera desberdina jarraituko litzateke, biak alde batean erabiltzea. Ondoren hobeto azaltzen da nola izango litzatekeen zerbitzariaren egitura berezi hau.

C.1.2. Bezero - Zerbitzari egitura

GipuzkoaBidaian-en sistema osoa hiru geruzetako arkitekturan oinarrituko litzateke, C.2. irudikoa bezala, eskaintzen dituen abantailak aukera hoberena bihurtzen baitute.

Bere izenak adierazten duen bezala, hiru zatitan banatzen da: aurkezpen geruza, logika geruza eta datu geruza. Hauetako bakoitzak independenteki lan egiten du, nahiz eta beraien artean “konektatuta” egon.



C.2. irudia: Hiru geruzetako arkitektura

Aurkezpen geruza erabiltzaileak ikusten duen zatia da. Hemendik zerbitzariaren informazioa eskaintzen, jasotzen eta erakusten da. Zati honek normalki ez du pisu gehiegirik, datu prozesamendu garrantzitsuenak ez baitira hemen egiten. Bezeroa edozein motakoa izan liteke: ordenagailuko aplikazio bat, nabigatzaile bat, smarthpone bat, tableta bat, eta abar. Hala ere, oraingoz Android aplikazioa bakarrik izango da.

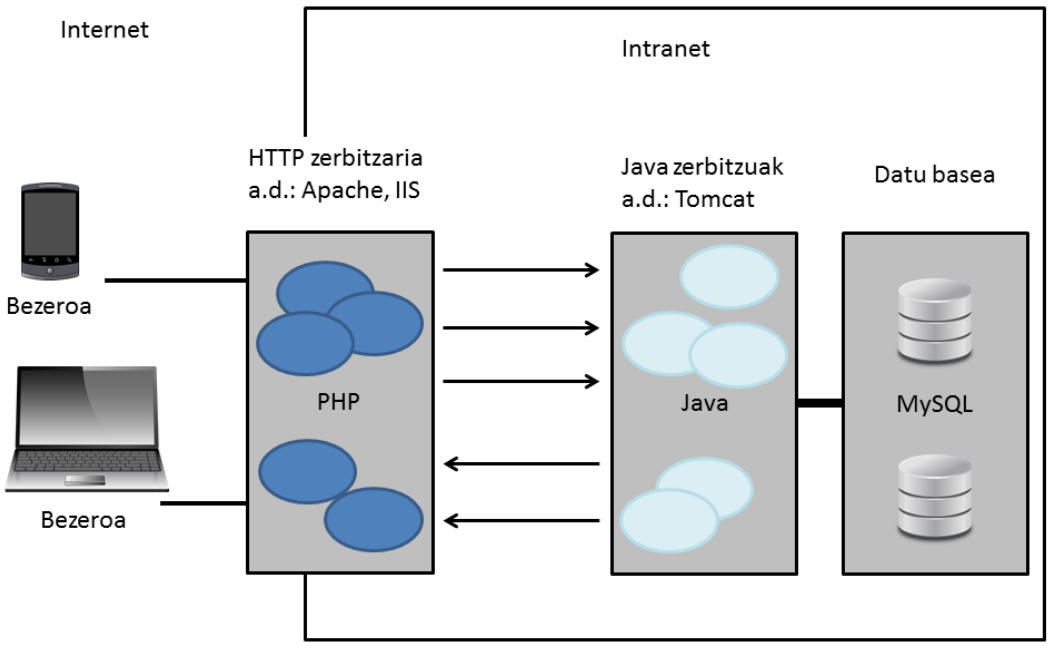
Logika geruza, negozio geruza ere deitzen zaio, kanpotik iristen diren eskaerak tratatzen ditu. Honek behar diren datuak begiratzen ditu eta hurrengo geruzari beharrezko eskakizunak igortzen dizkio informazioa prozesatzen duen bitartean. Azkenik, emaitza bezeroari bidaltzen dio. Zerbitzariaren zati hau garrantzitsuenetakoa da, hemen eskaerak eta informazio prozesamenduak egiten baitira.

Lehen aipatutako lengoaien arazoak hemen eragiten du. Geruza bat izateak ez du esan nahi sistema bakarrak inplementatzen duenik, aldi berean lanean dauden hainbat zerbitzari desberdin egon daitezke. Hori da jarraituko litzatekeen ideia.

Logika geruzak bi atal dituenez, eskaeren harrerarena eta informazioaren prozesamenduarena, bi zerbitzari desberdin egongo lirateke. Lehenengoa PHP-n oinarrituta eta bigarrena Java-n, bien arteko zubi bat lortuz. Egitura hau bi ekipok osatuko lukete, hauek elkarrekin lanean egongo lirateke C.3. irudiak erakusten duen bezala.

Lehenengo makina, http zerbitzaria izango litzateke, soilik eskaerak jasoz eta informazio eskakizunaz arduratuko litzateke. Bigarrenak, Java-ko zerbitzuaz arduratuko litzateke eta lehen aipatutako lanak izango lituzke.

Datu geruza informazio iturriekin lan egiten duen zatia da. Informazio eskaerak jasotzen ditu, datu baseari informazioa eskatzen dio eta jasotako datuak bueltan bidaltzen ditu. Geruza honen lanaz arduratzen den sistema ere Java lengoaian inplementatuko litzateke. Logika eta Datu geruzak programazio lengoaia bera izan arren (Lehenengoaren Java zatia bakarrik kontuan hartuz), ez du esan nahi makina berean egon behar direnik, ahal izanez gero bi makina independente erabiliko lirateke.



C.3. irudia: PHP/Java zubia

Orokorrean datu geruza honek ez du lan karga handirik, logika geruza eta datu basearen zubi bezala jokatzen baitu. GipuzkoaBidaian-en kasuan, ordea, kontrakoa izango litzateke, Open Data-ko eta Dbus-eko fitxategi iturriak ia egunero freskatzen direnez, datu baseko informazioa ere eguneratua mantendu beharko litzateke. Lan horretaz datu geruza arduratuko litzateke.

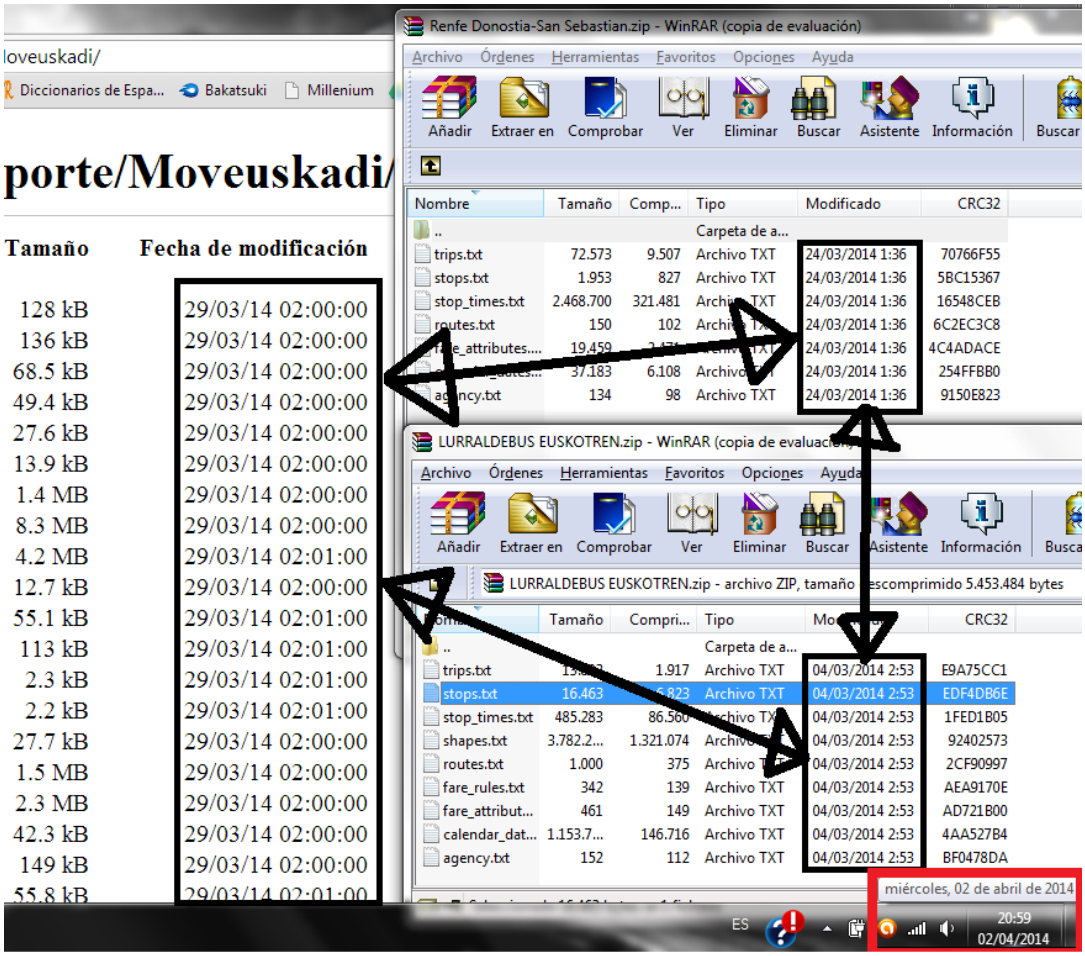
Hau lortzeko eguneraketaren kontrola eramateko, taula esklusibo bat sortuko litzateke datu basean, C.4. irudikoa bezala. Hau hiru eremuz osatuta egongo litzateke:

- Identifikatzaile bat.
- Konpainiaren izena.
- Eguneraketa data.

Informazio iturrian sartuz gero, artxibo konprimitu guztien gaurkotze data berdina dela ikus daiteke, C.5. irudiko ezkerreko laukiak erakusten duen bezala. Hala eta guztiz ere, horrek ez du esan nahi barruko fitxategi guztiak egun berean eguneratu direnik. Haietako baten barruan sartuz gero, ikus daiteke aldaketa parametroa ez datorrela bat webgunean azaltzen denarekin, C.5. irudiko eskuineko laukiak erakusten duen bezala.

Eguneraketak	
♦id	String
•konpainia	String
•eguneraketaData	Date

C.4. irudia: Eguneraketa taula



C.5. irudia: Open Data datu iturriko data diferentziak

Open Data webguneak egunero eguneratzen ditu Interneteko fitxategiak, barruko edukia berria ez izan arren. Izan ere, konpainia bakoitza arduratzen da eduki horretaz.

Beraien hitzetan, garraio publikoko konpainiek GTFS formatuko testu fitxategiak bidaltzen dizkiete eta hauek egiten duten bakarra Internetera igotzea da. Honek barruko eta kanpoko datak ia beti desberdinak direnaren zergatia azaltzen du.

C.3.1. Bezero - Zerbitzari egituraren abantailak

- **Eskalagarritasuna**: ("Zerbitzariaren eskalagarritasun modeloa" atalean hobeto azalduta)
    - Bezero eta zerbitzari kopurua handiago daitezke banan-banan.
    - Zerbitzaria hainbat makinetan antola daiteke.
    - Datu baseak ez du konexio bat ezarri behar bezero bakoitzarengatik.
- **Kontrolaren zentralizazioa**: zerbitzariak datuen sarbideak, baliabideak eta segurtasuna kontrolatzen dituenez, akastun edo ez baimendutako programa batek

zailagoa izango du sistema kaltetzeko. Zentralizazioak ere datuen eguneraketei eta beste baliabideei laguntzen die.

- Mantenu erreza: funtzioak eta ardurak hainbat ordenagailutan banatuko liratekeenez, ordezkapenak, konponketak, eguneraketak eta baita zerbitzariaren lekualdaketak ere posibleak izango lirateke bezeroa ezertaz konturatu gabe.
- Berrerabilpena: kode zatiak berrerabiltzeko erraztasuna handituko litzateke.
- Datu seguruak: erdiko geruzak ziurta dezake datu basean datu onargarriak idatziko direla soilik.
- Segurtasun hobekuntzak:
    - Bezeroak ez duenez datu basera sarbide zuzena, Datu Geruza seguruagoa da.
    - Negozio Geruza, orokorrean, seguruagoa da, zerbitzari zentral seguru batera kokatuta baitago.

C.3.2. Bezero - Zerbitzari egituraren desabantailak

- Konplexutasuna eta esfortzuan areagotu:
    - Orokorrean 3 geruzako arkitektura zailagoa da 2 geruzakoa (oinarrizko modeloa, sinpleena) baino.
    - Komunikazio puntuak bikoizten dira, lana eta esfortzua handituz.

C.2. Informazioaren eskuraketa eta biltegiratzea

Dagoeneko azaldu den bezala, bilaketa bat egiterakoan, aplikazioa kanpoko zerbitzarien FTP-ra konektatzen da eta hortik konpainien fitxategiak deskargatzen ditu. Behin hauek biltegiratuta, datu base modura erabiltzen dira, hortik zuzenean bidaiak bilatzeko.

Zerbitzari bitartekari bat erabiliz gero informazio lortzearen prozesua guztiz desberdina izango litzateke, kanpoko zerbitzarien FTP-ra egunero konektatuko litzatekeen prozesu bat martxan egongo litzatekeelako egindako eguneraketei adi egoteko.

Informazio fitxategiak deskargatuko lirateke zerbitzariaren datu basea sortzeko edo eguneratzeko. Hori egin eta gero fitxategiak ez lirateke ezabatuko, gutxi okupatzen baitute (ordenagailu batentzat) eta badaezpada ondo etortzen direlako.

Datu basearen egiturak GTFS formatuaren ideia bera jarraituko luke, 4.2.4. atalean azaldutakoa bezala. Ondorioz, fitxategietatik datuetara bihurtzeko prozesua sinplea eta azkarra izango litzateke.

Smartphonekin ez bezala, non memoria ahalmena askoz mugatuagoa den, ordenagailu batean datu guztiak gordetzeak ez du arazorik suposatzen, GipuzkoaBidaian-ek erabiltzen duen informazio kopurua ez baita hain handia:

Open Data-ko zerbitzarietatik eskuratuta:

- Fitxategi konprimituek 33 bat mega okupatzen dute.
- Informazio testu fitxategiek 140 bat mega okupatzen dute.
- Datu baseak, fitxategien eduki bera izango zuenez, 140 bat mega okupatuko luke.

Dbus konpainiaren kasuan:

- Informazio testu fitxategiek 26 bat mega okupatzen dute.
- Datu baseak, fitxategien eduki antzekoa izango zuenez, 26 bat mega okupatuko luke.

Zerbitzariaren informazioa eguneratua mantendu beharko litzateke, horretarako kanpoko hornitzaileak bere eguneraketak argitaratu bezain laster datu basea eguneratu beharko litzateke. Momentu horretan arazo bat suertatzen da, erabiltzaileak ezingo lukeela datu basean sartu hau berritzen den bitartean, bestela erdizko informazioa lortzeko arriskua egongo litzateke. Arazo hau konpontzeko bi datu base egongo lirateke, era honetan batekin lan egiten den bitartean bestea eguneratzen egongo litzateke eta behin prest egotean lehenengora ordezkatuko litzateke.

Beste arazo posibleak murrizteko helburuarekin, FTP-ak goizaldean begiratuko luke eguneraketarik al dagoen, zerbitzariaren aktibitatea minimoa izango litzatekeen momentuan. Aplikazio hau eremu itxi batean erabiltzeko prestatua dagoenez, Gipuzkoan soilik, ez litzateke beharrezkoa izango ordu diferentziaz kezkatzea.

C.2.1. Open Data eta Dbus datuak batera biltzeko arazoa

Lehenik gogoratu behar da bi iturri daudela: OpenData eta Dbus. Biak eskaintzen dituzten datuak GTFS formatuan egon arren ez dira berdinak, ondorioz, ezin dira batera erabili.

Open Data-ko informazioa leku beretik etortzen denez eremu bakoitzaren gako nagusia ez da inoiz errepikatuko, hainbat konpainien datuak izan arren. Etorkizunean konpainia berriren bat gehitu behar bada, ez da arazorik egongo, izan ere, momentu horretan erabiltzen den azken gakoa begiratzen da eta hortik aurrera datuak txertatzen dira. Berdina gertatzen da konpainiaren batek datu berriak gehitu nahi dituenean (adibidez geltoki berriren bat).

Esan liteke eremuen identifikatzailea automatikoki handitzen dela. Adibidez, Euskotren konpainiaren agentziaren datuen artean, bere agentzia identifikatzailea “1” da, BilboBus-ekoa “150”, Autocares del Zadorra S.L konpainiarena “179” etab. Konpainia berriak sartzen

diren heinean, balio handiagoak hartuko dituzte, errepikapenik ez egoteko. Gauza bera gertatzen da beste datuekin, geltokien identifikatzaileekin eta abar.

Dbus konpainiak eskaintzen duen informazioa ordea era propioan eginda dago, neurri batean GTFS formatua jarraitu arren ez da Open Data-koa bezain zorrotza. Ez hori bakarrik, erabiltzen dituen "identifikatzaileak" ezin dira Open Datako-ekin batu, errepikatzen baitira. Hala ere, horrek ez luke arazorik suposatuko, oraintxe bertan sistema kapaz da Dbus-eko informazioa Open Data-kora egokitzeko, hortaz egin beharko litzatekeen gauza bakarra identifikatzaileak aldatzea izango litzateke edo datu base propioa sortu berarentzat.

C.3. Informazioaren prozesamendua

GTFS formatuaren egitura datu basearen oso antzekoa denez, bidaiak bilatzeko prozesua dagoeneko egina dagoenaren oso antzekoa egin liteke. Kodean aldatu beharko litzatekeena batez ere informazioa nondik lortzen diren lerroak izango lirateke.

Gertuko geltokiak lortzea bi eratara egin daiteke:

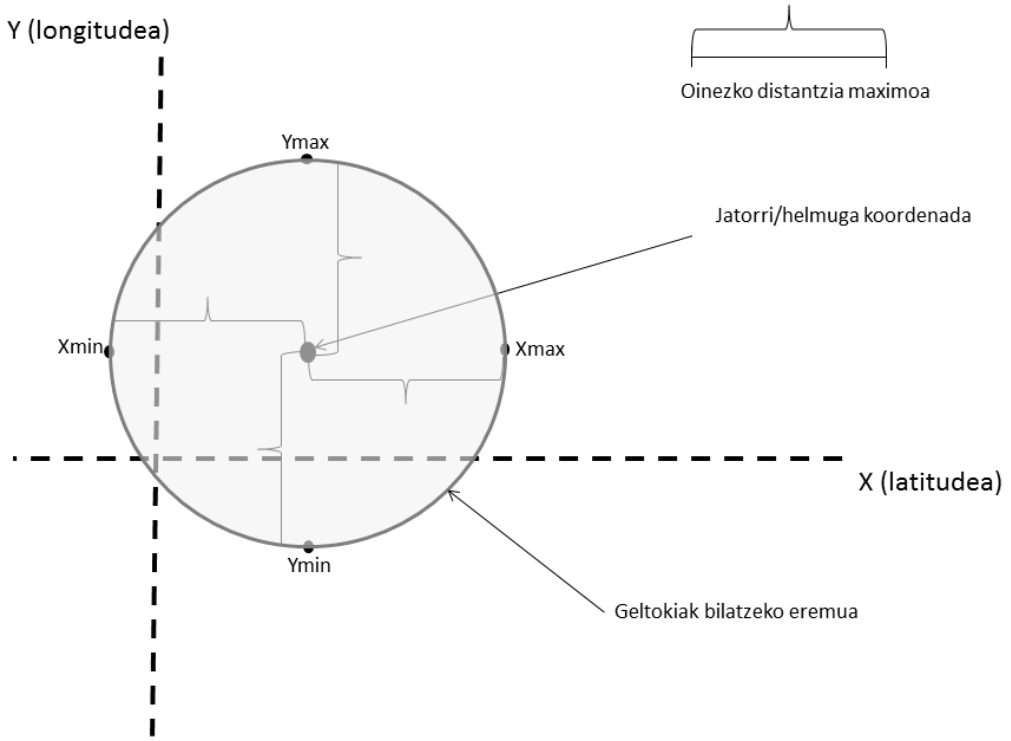
- Orain egiten den bezala, geltoki guztiak irakurri, bakoitzaren distantziak kalkulatu (jatorriarekin eta helmugarekin), distantzia maximoa baino handiago direnak baztertu eta zuzenak direnak jatorri eta helmuga listetan gorde.
- Bilaketa egin baino lehen, distantzia erradio bat ezarriko nuke, hau da, C.6. irudiak azaltzen duen bezala, koordenada eta *D* distantzia maximoa jakinda, bilaketa eremua ezar daiteke.

Adibidez, *Ymax* jatorrizko koordenadatik D distantziara iparraldean dagoen puntua da. *Ymin* ordea hegoaldean dagoena. Bi puntu hauek izanda, jatorrizko longitudea *Ymax* baino txikiago eta *Ymin* baino handiago izango beharko litzateke.

Hau egitea beharrezkoa izango litzateke, datu basean bilaketa sinpleak bakarrik egin daitezkeelako (ezin zaio eskatu distantzia kalkulatzea, izan ere, berak datuak ematen ditu prozesatu gabe).

Behin geltokiak izanda hurrengo prozesua egina dagoenaren oso antzekoa izango litzateke:

1. Jatorriko geltoki bakoitza hartu.
2. Berari dagokion *Stop_times*-ak lortu.
3. *Stop_times* bakoitzaren ibilbide identifikatzailea lortu.
4. Helmugako geltoki bakoitza hartu.
5. *Stop_times*-en bilatu helmuga geltoki identifikatzailea eta lehen lortutako *Stop_times*-eko identifikatzailea duen emaitzarik dagoen.
    a. Ez badago hurrengo helmuga geltokiarekin probatu.



C.6. irudia: Geltokiak bilatzeko eremua, oinezko distantzia maximoaren arabera

6. Baldin badago, geltoki sekuentzia zuzena dela konprobatu. Hala bada, falta diren datuak lortu (identifikatzaileekin) eta gorde edo informazioa eguneratu (dagoeneko ibilbide bera egiten duen emaitzarik baldin badago, geltokiak gertuen dauzkana mantenduko da).

C.4. Lokalean lan egiteko posibilitateak

Zerbitzari bitartekari bat erabiltzeko emergentzia plan bat izan behar da sistemak funtzionatzeari uzten dion kasurako. Aplikazioa Internet gabe erabiltzea ez da gomendagarria, informazioaren baliotasuna frogatu ezingo litzatekeelako eta bilaketaren emaitzak mugatuak izango liratekeelako.

Bidaiak offline bilatzeko era bat baino gehiago dago, baina lehenik xehetasun bat bereizi behar da, ez da berdina garraioen informazioa lokalki eskuratzea edota oinezkoena. Hauek dira bi erak:

- (A metodoa) Bilaketen emaitzak biltegiratu, oinezko eta garraiozko zatiak desberdinduz.
- (B metodoa) Garraio publiko konpainien informazioa duten fitxategiak eduki hortik zuzenean bilaketak egiteko.

C.4.1. Bilaketen emaitzak biltegiratuz

A metodoarekin ez litzateke bilaketarik egin behar, informazio guztia dagoeneko prozesatuta eta erakusteko prest egongo litzatekeelako. Arazoa da, dagoeneko egindako bilaketekin funtzionatuko lukeela soilik, hau da, gordeta dauden bilaketa parametroak eta unekoak desberdinak balira, metodo honen eraginkortasuna asko gutxituko litzateke.

Hau lortzeko, bilaketa baten emaitza lortzen den bakoitzean hainbat fitxategietan gorde beharko litzateke, izan ere bidaia bat oinezko eta garraioko zatiez osatzen da, eta datu hauek desberdindu behar dira.

Horretarako bi karpeta berri egongo lirateke aplikazioaren direktorio pribatuan. Informazio guzti hau SD txartelean biltegiratuko litzateke, smartphonearen barneko memoria askoz txikiagoa baita.

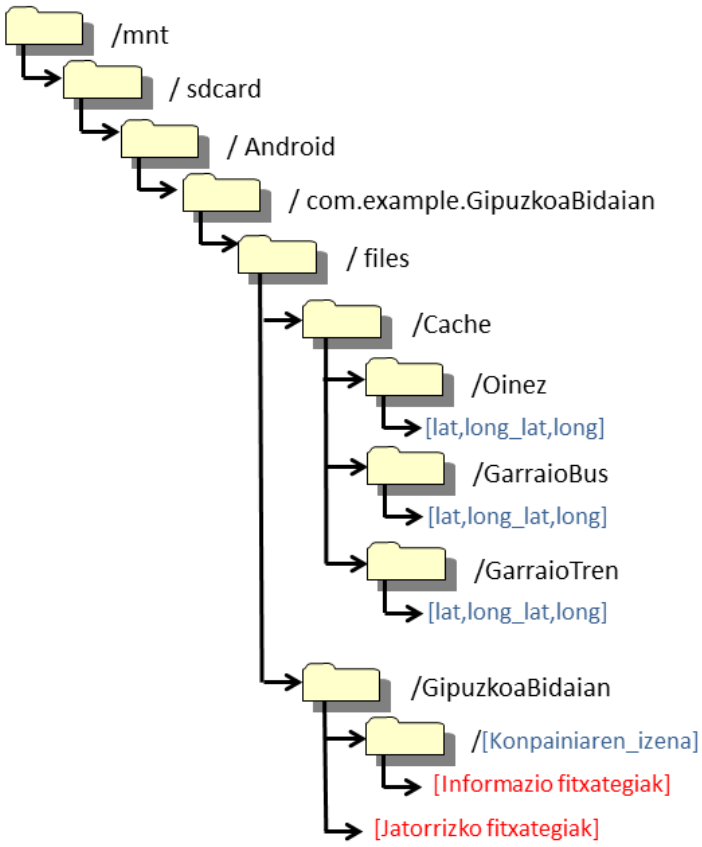
Aplikazio bat instalatzen denean Android sistemak dagozkion karpetan automatikoki sortzen ditu. Hala eta guztiz ere, berriak sortu beharko lirateke bilaketen emaitzak gordetzeko eta jatorrizko informazio iturri fitxategiak biltegiratzeko, C.7. irudiak erakusten duen bezala:

- *Cache*: Bilaketen emaitzak gordetzen diren lekua, A metodoarentzat.
  - *Oinez*: oinezko ibilbideak gordetzen dira.
  - *GarraioTren*: tren garraio publikoko ibilbideak gordetzen dira.
  - *GarraioBus*: autobus garraio publikoko ibilbideak gordetzen dira.
- *GipuzkoaBidaian*: garraio publiko konpainien fitxategi originalak gordetzen diren lekua, B metodorako. Direktorio honen barruan beste hainbat karpeta egongo dira, garraiobide konpainia bakoitzaren bat eta bakoitzak bere izena izanda.

GarraioTren eta *GarraioBus* direktorioen karpeten izenak egitura bera dute. Hau jatorri eta helmuga koordenadaz osatzen da honelako itxura edukiz: [“zenbakia” + “,” + “zenbakia” + ”_” + “zenbakia” + “,” + “zenbakia”]. Adibidez, karpeta bat “*43.273767,-2.021503_43.316676,-1.934079*” deituko litzateke. *Oinez* karpeta kasuan, fitxategiek ere izen formatu hori izango lukete.

Garraiozko ibilbideak geltoki batetik bestera izan arren, ez da berdina ordu zehatz batean edo edozeinetan egitea. Beraz, fitxategien izenak ezingo lirateke direktorearen berdinak izan, bi koordenaden artean bidaia desberdinak egon daitezke eta. Arazo hau konpontzeko fitxategi bakoitzak bilaketan adierazitako ordua izango luke izentzat: [“ordua” + “:” + “minutua”]. Bilaketa ordu gabe egin bada, orduan izen adierazgarria izango luke: [“ordu_gabe”].

GarraioBus eta *GarraioTren* direktorioen karpeten izenen koordenadak *Oinez* direktorioan ere aurkituko lirateke, ez latitude-longitude bikote bera, baina bai bakoitza independenteki, azken batean, oinezko bidaia bat geltoki batean hasten edota bukatzen da. Egitura hau honela antolatuko litzateke informazio bilaketak azkarrak eta zehatzak egiteko.



C.7. irudia: Aplikazioaren zuhaitz direktorioa

Kontuan izan beharko litzatekeen beste xehetasun bat da bilaketa bat egiterakoan, oinezko ibilbideek ezingo luketela erabiltzaileak zehaztutako distantzia maximoa gainditu. Ondorioz, aurretik egindako bilaketa baten emaitza berrerabiltzea ez litzateke beti posible izango. Suposatuz DIST_MAX oinezko distantzia maximoa dela, jatorri puntua JAT dela eta HELM helmuga puntua dela, bidaiak bilatzeko jarraitu beharreko urratsak hurrengoak izango lirateke:

1. *JAT* jatorria duen oinezko ibilbideen fitxategi lista lortu, alegia, "*JAT_[edozer]*" izena duten fitxategiak.
2. Artxibo bakoitzaren distantzia osoa begiratu eta *DIST_MAX* baino gehiago direnak baztertu. Hemen malgua izan daiteke eta akats tarte bat definitu, hots, emaitza aldatuko ez lukeen aparteko distantzia. Lista definitiboa "*jatorrikOinez*" deituko litzateke.
3. HEL helmuga duen oinezko ibilbideen fitxategi lista lortu, alegia, "*[edozer]_HEL*" izena duten fitxategiak.
4. Bigarren pausua errepikatu baina lortutako datu berriekin. Lista definitibo hau "*helmugakOinez*" deituko litzateke.
5. "*jatorrikOinez*" listako elementu bakoitzagatik bere helmuga eskuratzen da, "*helm*".

6. "*helm*" bakoitzagatik, garraiobideen direktorioetan (batean edo bietan, erabiltzailean garraio motak aukeratu dituen arabera) izen horrekin hasten diren karpetak eskuratzen dira.
7. Aurreko urratsean lortu den direktorio bakoitzaren barruan bilatzen da erabiltzaileak zehaztutako orduarekin bat etortzen den fitxategia. Garraio fitxategi lista hau "*garraioakIbilb*" deituko da.
8. "*garraioakIbilb*" listako elementu bakoitzaren helmuga hartuko da eta "*helmugakOinez*" fitxategien jatorriarekin konparatuko da berdintasunak bilatzeko.

Bilaketa metodo hau erabiltzeko, lehenik zerbitzarira ezin dela konektatu edo Internet konexiorik ez dagoela konprobatu beharko litzateke. Ondoren, aurreko bilaketa urratsak burutuko lirateke ibilbidea osatuko dituzten fitxategiak lortuz. Azkenik, datu guztiak lortu eta erakutsiko lirateke.

C.4.2. Fitxategi originalak biltegiratzen

Metodo honek mugikorrentzat karga handiagoa suposatuko luke, dena dela, garraio publiko askotariko bilaketak egitea ahalbidetuko luke. Honen ideia GipuzkoaBidaian originalaren funtzionamendu bera jarraitzean oinarritzen da, hau da, lokalean dauden fitxategi originalak erabiltzea bilaketak egiteko.

Erabiltzaileak aurreko bilaketa baten, emaitza deskargatzen duen aldi berean, erabilitako informazio artxiboak deskargatuko lirateke. Honek arazo bat suposatzen du, sare trafiko handia. Arazo hau konpondutzat eman liteke wifia bakarrik dagoenean erabiliz.

Erabiltzaile batek normalean antzeko jatorri eta helmugak adierazten dituenez, deskargatutako informazio kopurua ez litzateke neurriz kanpo izango. Gainera, ez litzateke beti deskargatu behar, soilik lokalean daudenak zaharkituta daudenean. Ez hori bakarrik, aldi batean soilik deskargatzeko aukera eman liteke, zaharkituegiak geratu arren, azken batean funtzionatuko luketelako.

C.5. Zerbitzariaren eskalagarritasun modeloa

Zerbitzari sistema bat eskalatzeko bi modu nagusi daude:

- Eskalatze bertikala: makina handiago baten erabileran oinarritzen da.
- Eskalatze horizontala: lan karga banatzen duen protokolo batekin, makina gehiago edukitzean oinarritzen da.

Bakoitzak bere abantailak eta desabantailak izan arren ez da bat bakarrik aukeratu behar, diseinu misto bat jarrai daiteke:

- Aurkezpen eta logika geruzak eskalatze horizontal batekin kudeatuko dira. Bezero gehiegiren eskaerak jasotzen duen egoerara iritsiko balitz, makina berriak instalatuko lirateke eskaeren tratamendua osatzeko. Sistemak prest egongo beharko luke hasieratik hainbat makina desberdinetan erabili ahal izateko, hau da, nahiz eta hasieran dena ekipo berean egin, programatzerakoan, posibilitate hau kontuan hartu beharko litzateke kodea sortzerakoan. Hau egiten ez bada, arazo larria izango litzateke etorkizunean, izan ere, aplikazioa martxan egonda sistemaren egitura aldatu beharko litzateke.
- Datu Geruza osoa, datu basea eta bere kudeatzailea, eskalatze bertikal batekin kudeatuko da. Hainbat datu base, zerbitzari bakarra izango balitz bezala, ez da egokia, ekonomikoki garestia da, konputazionalki zaila da eta eraso informatikoak jasateko probabilitate handiagoak ditu.

C.6. Arriskuak eta prebentzio plana

GipuzkoaBidaian-ek zerbitzari bitartekari bat erabiliko balu arrisku gehiagotara ikusgai egongo litzateke, gerta litekeen arazo lista asko handituko litzatekeelako. Atal honetan proiektua arriskuan jar dezaketen elementuak identifikatu dira: hauetako bakoitzak zertan datza azalduko da, gertatzeko aukera, izango lukeen eragina, zergatik gerta liteke eta ekiditeko edo efektuak txikiagotzeko estrategiak zehaztuko dira.

C.6.1 Zerbitzaria hondatzea

Azalpena: sistema osoa kudeatzen duen eraikina (sistema martxan jartzen duten ekipo guztiak leku berean daudela suposatuz) hondatzea. Sistema geografikoki bananduta badago zati bat suntsitzen bada.

Gertatzeko probabilitatea: baxua (oso zaila da mota honetako arazoen probabilitatea kalkulatzea).

Jatorri posiblea: adibidez, hondamendi natural bat edo istripu bat. Hala ere, gaur egungo eraikinak prest daude ustekabe hauek eusteko, hortaz, gertatzeko probabilitateak ez dira handiegiak. Istripuak ordea, edozein motakoak izan daitezke, hortaz ezinezkoa da jatorria asmatzea.

Eragina: oso handia. Horrelako zerbait gertatuz gero aplikazioa soilik offline erabili ahal izango zen, informazioaren baliotasuna gutxituz.

Prebentzio plana: hau ekiditeko bi plan dauzkat.

- Zerbitzari osoaren kopia bat beste leku batean izan. Era honetan, zerbitzua ematen duen sistema osoa edo zati bat erortzen bada, beste leku batean dagoen kopia

martxan jarriko litzateke aurrekoa ordezkatuz. Soluzio honek arazo bat du, garestia dela, izan ere, bi leku desberdinetako makinak martxan eduki behar lirateke denbora osoan. Gainera, sistemak prest egon beharko luke zerbitzaria dinamikoki aldatzeko.

- Zerbitzaria lainoan izatea (Cloud Computing). Metodo hau erabiliz arrisku hau ez litzateke existituko.

C.6.2. Datu iturria erortzea

Azalpena: zerbitzariak erabiltzen dituen datu iturriren bat zerbitzurik ez ematea.

Gertatzeko probabilitatea: kalkulaezina, nire esku ez dagoen zerbait da eta ondorioz ezin dut jakin.

Eragina: oso handia. Hasieran, erabiltzaileak ez luke diferentziarik nabarituko, baina denbora pasatzen den heinean, arazoa areagotzen joango litzateke, informazio zaharkituta geratuko litzatekeelako.

Jatorri posiblea: ezin da jakin, beste erakundeek kudeatzen duten zerbait da eta.

Prebentzio plana:

- Datu Iturri desberdinak bilatu, batak funtzionatzen ez badu sistemak bestetik informazioa hartzeko. Arazoa da datu iturri berria aurkituz gero, segur aski informazioaren formatua ez litzatekeela berdina izango, sistemaren konplexutasuna handituz.
- Zerbitzaria honelako arazoa detektatzeko prest egotea, gertatzeko momentuan iturriarekin kontaktuan jartzeko arazoaz abisatzeko. Era honetan, datu iturria konpontzen den arte zerbitzariak gordeta duen informazioarekin lan egingo luke, bukaerako erabiltzaileak ez konturatuz.

C.6.3. Datu basea betetzea

Azalpena: ordenagailuaren disko gogorra betetzea.

Gertatzeko probabilitatea: oso txikia. Gaur egungo hardwarearekin hau gertatzea ia ezinezkoa da.

Eragina: oso handia. Informazio berria ezin bada gehitu, bidaien informazioa ezingo da bermatu.

Jatorri posiblea: kudeaketa txar baten eta kontrolik ezaren ondorioz gerta litekeen arazoa.

Prebentzio plana: bi era daude hau ekiditeko:

- Zerbitzarian programa bat instalatu non aldizka, zerbitzariaren egoeraz abisatuko lukeen: trafikoa, ordenagailuen egoera, memoriaren egoera, etab.

- Zerbitzaria lainoan izatea (Cloud Computing). Metodo hau erabiliz, memoria guztia betetzea ezinezkoa izango litzateke, memoria limitearen kontzeptua ez bailitzateke existituko.

C.6.4. Kanpoko erasoak

Azalpena: norbaitek zerbitzaria erasotzen badu sistemari kalte egiteko edo informazioa lortzeko.

Gertatzeko probabilitatea: txikia. Sistemak informazio pertsonala erabiltzen ez duenez ez du interesik erasotzaileentzat. Gainera, aplikazioa Gipuzkoa mailan erabiltzeko pentsatua dagoenez ez du publiko handiegirik izango, erasoen helburua izateko posibilitateak murriztuz.

Eragina: oso handia, birus bat datuen artean sartuz gero mugikorrak infektatzeko arriskua egongo litzateke.

Jatorri posiblea: edozeinek. Erasotzailearen profila bi motakoa izan daiteke: informazioa ostu nahi duena edo sistema izorratu nahi duena.

Prebentzio plana: sistema eragile seguru bat erabiltzea, suhesi indartsu bat erabiltzea eta kodea edozertarako prest egotea.

C.6.5. Kalitatezko informazio ez eskaintzea

Azalpena: erabiltzaileari erakusten zaion informazioa kalitatezkoa ez izatea.

Gertatzeko probabilitatea: txikia. Informazioa hornitzen duten erakundeak ospe handikoak dira eta horrelako zerbait gertatzeko probabilitateak txikiak dira.

Eragina: oso handia, erakusten den informazioa okerra izan daiteke.

Jatorri posiblea: jasotako datuak zuzenak ez izatea, hau edozein arrazoigatik gerta daiteke. Hirugarren pertsonek kudeatutako gauza denez ezinezkoa da jakitea zerk eragiten duen.

Prebentzio plana: bukaerako erabiltzaileri informazioa zuzena ez dela abisatzeko ahalmena emango zitzaion, era honetan kexa bat jaso bezain laster informazio okerra dagokion garraio publikoko konpainiari abisatuko litzaioke, arazoa ahal den lasterren konpontzeko.

D Eranskina: Garraio publikoko konpainien webguneen azterketa

Eranskin honetan, datu iturri desberdinak aurkitzeko helburuarekin, garraio konpainien webguneei egindako azterketa azaltzen da. Bestetik, laguntza lortzeko konpainiekin harremanetan jartzeko egindako saiakerak ere azalduko dira, azkenean Dbus-ekin bakarrik izan zen posible harremana ezartzea

Lehenik eta behin gogoratu behar da web-orrialdeen analisia proiektuaren hasieran egin zela.

D.1. Webguneetatik datuak lortu

Garraio konpainien webguneetan nabigatu eta atal guztiak begiratu ondoren, konklusio batera ailegatu nintzen, orokorrean hortik lor daitekeen informazio bakarra da zenbat denbora gelditzen den garraio bat geltoki batera ailegatzeko.

Gainera geltokien informazioa ezin da lortu, normalean web-orriek udalerrien lista edo geltokien izenak bakarrik erakusten dituzte, baina hori ez da nahikoa lan egiteko, datu gehiago behar dira. Ondorioz beste informazio iturriak erabiltzea erabaki zen.

Ondoren azalduko da garraio konpaien webguneei egin zitzaien analisia: zein informazio eskaintzen duten, nola lor zitekeen informazio hori eta zer aprobetxatu ahal izango litzatekeen.

D.1.1. Renfe

Konpainia honek estatu osoan lan egiten du, probintzia desberdinen arteko bidaiak eta bakoitzean bertan bidaiak egiten ditu. Hortaz bi atalak aztertu behar izan dira.

Renfe-k web-orrialde dauka bat non hiri guztiak agertzen diren, bi aukeratuz eta nahi den ordua zehaztuz bilaketa bat egiten du, D.1 taulako hiperestekak orri horretara eramaten du.

Webguneaz balia daiteke eskuz bilaketa bat egiteko. Kode-iturria aztertuz ikus daiteke agertzen diren geltokiak D.1. kode-zatiko elementuaren barruan daudela. Barruko edukia lortzeko D.2. kode-zatian agertzen den XPath-a erabili beharko litzateke.

```
http://horarios.renfe.com/HIRRenfeWeb/estaciones.do?&ID=s
```

D.1. taula: Espainiako Renfe-ko geltoki guztiak

```
<li id="cuadrado_amarillo">
   …
</li>
```

D.1. kode-zatia: Geltokiak dauden elementua

```
//*[@id="cuadrado_amarillo"]
/html/body/form/div/div[2]/table/tbody/tr[2]/td/li
```

D.2. kode-zatia: Geltoki guztiak lortzeko XPath-ekin, bi era

```
<a class=”linkgrise”
href="javascript:buscarDestinosDirectos('geltoki-kodea', 's',
document.estaciones.DF.value, document.estaciones.MF.value,
document.estaciones.AF.value)">
       geltoki-izena
</a>
```

D.3. kode-zatia: Geltokiaren kodea eta izena hiperestekan

```
http://horarios.renfe.com/HIRRenfeWeb/buscar.do?&O=
[jatorri_geltoki_kodea]&D=[helmuga_geltoki_kodea]&ID=s&
DF=[eguna]&MF=[hilabetea]&AF=[urtea]
```

D.2. taula: Bilaketa egiteko URL helbidea

Elementu bakoitzaren barrualdea desberdina da, D.3. kode-zatian ikus daiteke nola den elementu bakoitza eta non aurkitzen den geltokiaren izena eta kodea.

Bilaketa D.2. taulako URL helbidearekin egiten da, gorriz agertzen diren zatiak bilaketa egiteko parametroak dira:

- **Jatorri geltokia**: bost digituko zenbaki bat, D.3. kode -zatiko geltoki-kodea da.
- **Helmuga geltokia**: bost digituko zenbaki bat, D.3. kode -zatiko geltoki-kodea da.
- **Eguna**: zenbaki bat.
- **Hilabetea**: zenbaki bat.
- **Urtea**: zenbaki bat.

D.3. taulan adibide bat ikusi daiteke:

```
http://horarios.renfe.com/HIRRenfeWeb/buscar.do?&O=11505&D=11600&ID=
s&DF=10&MF=10&AF=2013
```

D.3. taula: Bilaketa egiteko URL helbidearen adibide bat

Behin deia eginda HTML orri bat lortzen da bilaketaren emaitzekin, hortik informazioa lortzen da eta bidaien informazioa lortzen da.

Aldiriko trenen kasuan desberdina da, geltokien lista eta bilaketa egiteko akzioa orri beran daude, D.4. taulako helbidean.

http://www.renfe.com/viajeros/cercanias/sansebastian/
http://horarios.renfe.com/cer/hjcer300.jsp?NUCLEO=61&CP=NO&I=s#

D.4. taula: Donostiako aldiriko trenak bilatzeko webguneak

Kode iturria begiratzen bada ikus daiteke geltoki guztiak dituzten bi lista daudela. D.4. kode-zatian jatorriko geltokien zerrendaren zati bat irudikatzen da.

```
<select class="caja_texto1" onclick="borrar_buffer()"
onblur="borrar_buffer()" onkeypress="buscar_op(this)" name="o">
      <option value="?" selected="">Seleccione Estación<option>
      <option value="11409">Alegia de Oria</option>
      <option value="11505">Andoain<option>
      <option value="11504">Andoain-Centro<option>
      <option value="11502">Anoeta<option>
      …
      …
      …
      <option value="11506">Urnieta<option>
      <option value="11518">Ventas<option>
      <option value="11400">Zumarraga<option>
</select>
```

D.4. kode-zatia: Geltokien lista

Web-orrian, bilaketa egiteko botoiaren kodea aztertuz, funtzio bati deitzen diola ikus daiteke, D.5. kode-zatiak deskribatzen du. Funtzio horri parametro bat bidaltzen zaio, JSP fitxategi bat, eta zoritxarrez hori simulatzea ezinezkoa da.

```
<a title="Buscar" onclick="eje_cual('hjcer310.jsp')" href="#">
      …
</a>
```

D.5. kode-zatia: Bilaketa egiteko botoiaren kodea

D.6. kode-zatian ikus daiteke “eje_cual” funtzioaren (D.5. kode-zatikoa) kode iturria .

```
function eje_cual(program) {
   if(validar()=='s') {
      document.f1.action='/cer/'+program ;
      document.f1.submit();
   }
   else {
      return(false);
   }
}
```

D.6. kode-zatia: "eje_cual" funtzioa

Laburtuz, Renfe-ko webgunean bidaien bilaketak egin daitezke, arazoa da aldiriko trenekin lan egiteko bilaketak egiten duen funtzioari deitzea ezinezkoa dela (agian ikerketa sakon baten ondoren posible dela ikusten da, baina oraingoz ez da horrela). Hala ere, arazo handiena da hemen egin daitekeen bakarra irteera eta iritsiera orduak kalkulatzea dela, ez da bidaiei buruz informazio gehiagorik. Ez da geltokiei buruz informazio zehaztua ematen, bere erabilpena ezinezkoa egiten da.

D.1.2. Euskotren

Bere webgunetik bi atal bereizi behar dira, trenaren ibilbidea erakusten duen zatia eta geltoki batetik ze ordutan pasako den jakiteko zatia.

Geltokien lista eta ordutegia bilaketa D.5. taulako URL helbidean aurki daiteke. Kode iturria begiratuz gero geltokien lista aurki daiteke, D.7. kode-zatiak nola zerrenda osatuta dagoen, elementu bakoitzak geltokiaren izena eta identifikatzaile bat du.

http://www.euskotren.es/es/horarios

D.5. taula: Bidaia bilaketa egitek URL helbidea

```
<select id="edit-origen" class="form-select required error"
name="origen">
      <option value=""></option>
      <option value="AO">AIA-ORIO</option>
      <option value="AZ">ALTZOLA-ELGOIBAR</option>
      <option value="AM">AMAÑA</option>
      <option value="SS">AMARA</option>
      <option value="A">AMOREBIETA</option>
      <option value="AG">AMOREBIETA GERALEKUA</option>
      <option value="AT">ANOETA</option>
      …
      …
      …
      <option value="ZM">ZAMUDIO</option>
      <option value="PT">ZAMUDIOKO PARKE TEKNOLOGIKOA</option>
      <option value="ZZ">ZARAUTZ</option>
      <option value="ZT">ZUGAZTIETA-MUXIKA</option>
      <option value="ZG">ZUHATZU-GALDAKAO</option>
      <option value="ZI">ZUMAIA</option>
```

```
      <option value="ZK">ZUMALAKARREGI</option>
</select>
```

D.7. kode-zatia: Geltokien zerrenda, bakoitzaren kodearekin eta izenarekin

Bilaketa formulario baten bidez egiten da, hortaz, hau simulatu daiteke eta parametroak eskuz bete. Behin bilaketa eginda emaitza lortzen da eta XPath-ekin bidaia posibleen ordutegiak lor daitezke.

Trenak egiten duen ibilbidea jakiteko, geltokiak barne, beste era bat dago. Euskotreneko webguneak esteka batzuk dauzka Google Maps-era, non trenak egiten duen ibilbidea eta geltokiak ikus daitezkeen zuzenean. Ez hori bakarrik, orri horietan esteka bat dago fitxategi batera, non geltoki lista, helbideak, koordenadak eta ibilbidea marrazteko kodea dagoen.

Euskotren Gizpuzkoak aldean bi linea bakarrik dauzkanez ez da ezer bilatu behar, informazioa D.6. taulako URL helbideetan aurki daiteke.

(Lasarte/Oria-Hendaia linea): https://maps.google.es/maps/ms?ie=UTF8&hl=es&t=h&source=embed&msa=0&output=kml&msid=201319501371444821422.0004a2481810172cb8851
(Bilbao-San Sebastián/Donostia linea): https://maps.google.es/maps/ms?ie=UTF8&hl=es&t=h&source=embed&msa=0&output=nl&msid=201319501371444821422.0004a25ffe8f94719fbc8

D.6. taula: Euskotren lineen informazioaren URL helbideak

Google Maps-eko webgunean informazio asko dago, baina soilik zati bat da interesgarria GipuzkoaBidaian-entzat. D.8. kode-zatian aprobetxa daitezkeen zatiak ikus daitezke beltzaranaz.

```
  …
    <Placemark>
      <name>Hendaia</name>
      <description><![CDATA[<div dir="ltr">Boulevard de Gaulle,
s/n<br>64700 Hendaye, France<br></div>]]></description>
      <styleUrl>#style3</styleUrl>
      <Point>
        <coordinates>-1.782076,43.352528,0.000000</coordinates>
      </Point>
    </Placemark>
    <Placemark>
      <name>Topo</name>
      <description><![CDATA[]]></description>
      <styleUrl>#style5</styleUrl>
      <LineString>
        <tessellate>1</tessellate>
        <coordinates>
          -2.020140,43.270779,0.000000
          -2.021310,43.273281,0.000000
          -2.021390,43.273609,0.000000
  …
```

D.8. kode-zatia: Google Maps-eko kode iturriko zati bat

Datuak lortzeko (geltoki izena, helbidea eta koordenadak) XPath erabili behar da:

```
//Placemark [name/text()!="Topo" and name/text()!="bilbao-donosti"]
/ (name/text() | description/text() | Point/coordinates/text())
```

D.9. kode-zatia: Informazioa lortzeko XPath-a

Eta ibilbidearen koordenadak lortzeko:

```
//Placemark [name/text() ="Topo"  or name/text() = "bilbao-donosti"]
/ LineString/coordinates/text()
```

D.10. kode-zatia: Ibilbidearen koordenadak lortzeko XPath-a

Laburtuz, Euskotreneko webgunetik geltokien informazio asko lor dezaket eta ordutegiak ere lor ditzaket, hortaz ezer hobeagoa aurkitzen ez den bitartean hau datu iturri aukera posible bat da.

D.1.3. DBUS

Konpainia honen bidaiak soilik Donostian zehar dira, hortaz, bestean ez bezala hemen ez da herriz-herri bilatu behar.

Bere webguneak planifikatzaile bat du, D.7. taulako URL helbidean. Hemen, jatorrizko helbide bat eta helmugako bat adieraziz sistemak bidaia posible guztiak bilatzen ditu. D.11. kode-zatian kode iturriaren geltoki zerrendaren zati bat erakusten da, bakoitzaren izena eta kodea zehaztuz.

http://www.dbus.es/es/usuarios/planificador-rutas

D.7. taula: Dbus-eko planifikatzailearen URL-a

```
<select name="street[name_es]" id="calle_origen">
      <option value="">Seleccione una calle</option>
      <option value="8020">Abalotz, Autovia de</option>
      <option value="10">Adarra, Plazoleta del</option>
      <option value="7240">Aduna, Plaza de</option>
      <option value="7020">Agerre, Camino de</option>
      <option value="30">Agirre Miramon, Calle de</option>
      <option value="40">Agiti, Camino de</option>
      <option value="6000">Agorrene, Paseo de</option>
      …
      …
      <option value="1025">Zuhaizti, Plaza de</option>
      <option value="7010">Zuloaga, Plaza de</option>
      <option value="5210">Zumalakarregi, Avenida de</option>
      <option value="5230">Zurriola, Avenida de</option>
      <option value="1035">Zurriola, Paseo de</option>
      <option value="1225">Zurriola, Puente de</option>
      <option value="6930">Zuzenene, Camino de</option>
</select>
```

D.11. kode-zatia: Geltoki zerrenda, izena eta kodearen

Bilaketa egiteko botoi bat sakatu behar da eta orduan bilaketa egingo duen web-orriari GET metodoaren bidez egingo deituko zaio. D.8. taulan URL bilaketaren adibide bat adierazten da (beltzaranez dauden zatiak bilaketa egiteko parametroak dira).

"http://rutas.dbus.es/es/desktop/rutas?origen_id=**39991**&origen_str=**+Calle+de+Agirre+Miramon**&destino_id=**29148**&destino_str=**+Plaza+de+Adun**a&distancia=**300**&dia=**Laboral**&commit=**Obtener+rutas**"

D.8. taula: Bilaketa URL adibidea

Ez hori bakarrik, Dbus-ek web zerbitzu bat dauka, non geltoki bat adieraziz igaroko diren garraioak esaten dituen, noiz pasatuko diren zehazki.

D.9. taulako webgunean konpainiaren linea guztiak agertzen dira, bat aukeratuz web-orrialde berria kargatuko da bere geltoki guztiekin. D.10. taulan URL helbidearen egitura adierazten da, beltzaranez dauden zatiak linea eta geltokia zehazten dute.

http://www.dbus.es/es/usuarios/lineas-horarios/

D.9. taula: Geltokien informazioaren webgunea

http://www.dbus.es/es/usuarios/lineas-horarios /[***linea_zenbakia***]-[***linea_deskripzioa***]/horarios-de-llegada /[***geltoki_kodea***]-[***geltoki_izena***]

D.10. taula: Linea baten eta geltoki baten informazioa erakusten duen URL-a

Geltoki baten web-orrialdean egonda autobusa zer ordutan igaroko den lor daiteke, horretarako D.12. kode-zatiko XPath erabiltzen da.

html/body/div/div[3]/form/div[3]/ul/li
//*[@id="pass_times_form"]//*[@id="derecha"]/ul

D.12. kode-zatia: Geltoki batetik autobusa igarotzen den ordua lortzeko XPath bi erak

Laburtuz, Dbus-eko webguneak planifikatzaile bat izan arren ez du eskaintzen informazio nahikoa, geltokien izena agertzen da baina ez koordenadak. Beraz, ahal izanez gero ez da erabiliko.

D.1.4. Lurraldebus

Lurraldebus konpainiak autobus konpainia desberdin asko biltzen ditu, hortaz, hauen datu guztiak bere webgunean aurki daitezke. Autobusak zein herritatik pasatzen diren jakiteko eta bilaketak egiteko D.11. taulako helbidera jo behar da.

http://www.lurraldebus.net/Hors.aspx

D.11. taula: Euskotren webgunean bidaiak bilatzeko URL-a

Kontrakoa adierazten duen ikerketa sakonagoa egin arte, Android-ek ezin du webgunearen zerbitzuaz baliatu bilaketak zuzenean egiteko.

Linea bakoitzaren informazioa jakitea ordea posible da. D.12. taulako web-orrian linea guztiak agertzen dira eta bat aukeratuz mapan ikusi ahal izango da honen ibilbidea eta geltokiak. Ez hori bakarrik autobusa non dagoen momentu guztian ikus daiteke mapan.

http://www.lurraldebus.net/BuOl.aspx

D.12. taula: Lineak mapan ikusteko URL-a

Linea baten informazioa eskuratzeko GET dei bat egin behar da D.13. taulako helbidera, linea baten identifikatzailea parametro bezala bidaliz. Datu hau D.12. taulako webgunetik lor daiteke, hemen linea guztiak dituen zerrenda bat dago. Gainera bakoitzak bere kode identifikatzailea dauka. XPath erabiliz, D.13. kode-zatikoa, lineen kodeak lor daitezke.

http://www.lurraldebus.net/WS_Lbw/Service.asmx/DefLinea?lineaId=***[zenbakia]***

D.13. taula: Linea baten informazioa lortzeko URL-a

```
//*[@id="lbLins"]/option[text()=”linearen_izena”]/@value
```

D.13. kode-zatia: Linea baten identifikatzailea lortzeko XPath-a

Behin deia eginda XML formatuko objektu bat itzuliko du informazio guztiarekin. D.14. kode-zatian adibide bat adierazten da.

```
<?xml version="1.0" encoding="utf-8"?>
<InfoLinea xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="JsonData">
  <lat>43.254626935617047</lat>
  <lon>-2.1307258170384622</lon>
  <zoom>13</zoom>
  <paradas>
    <Parada>
      <lat>43.274324230599795</lat>
      <lon>-2.1255037403163719</lon>
      <cod>1014</cod>
      <des>AIA ORIO GELTOKIA</des>
    </Parada>
    <Parada>
      <lat>43.267922981782107</lat>
      <lon>-2.1197669592995951</lon>
      <cod>6114</cod>
      <des>SANTIAGO 2</des>
    </Parada>
    <Parada>
      <lat>43.246909179940658</lat>
      <lon>-2.1178154210602562</lon>
      <cod>68</cod>
      <des>MAÑARIN-ZELAI</des>
    </Parada>
    <Parada>
      <lat>43.241076258796781</lat>
      <lon>-2.1192544432264961</lon>
```

```
      <cod>66</cod>
      <des>ERREKONDO</des>
    </Parada>
    <Parada>
      <lat>43.232821752715374</lat>
      <lon>-2.137341193257968</lon>
      <cod>63</cod>
      <des>ILARRAMENDI</des>
    </Parada>
    <Parada>
      <lat>43.241272014935049</lat>
      <lon>-2.14626257342894</lon>
      <cod>1020</cod>
      <des>ETXENAGUSIA</des>
    </Parada>
    <Parada>
      <lat>43.237483467804978</lat>
      <lon>-2.1477934070735847</lon>
      <cod>1027</cod>
      <des>AIA</des>
    </Parada>
    <Parada>
      <lat>43.273130162165181</lat>
      <lon>-2.1378318344999729</lon>
      <cod>440</cod>
      <des>ALTO ORIO 2</des>
    </Parada>
    <Parada>
      <lat>43.276651734584071</lat>
      <lon>-2.1248911790662968</lon>
      <cod>439</cod>
      <des>ORIO</des>
    </Parada>
    <Parada>
      <lat>43.241076258796781</lat>
      <lon>-2.1192544432264961</lon>
      <cod>66</cod>
      <des>ERREKONDO</des>
    </Parada>
    <Parada>
      <lat>43.246909179940658</lat>
      <lon>-2.1178154210602562</lon>
      <cod>68</cod>
      <des>MAÑARIN-ZELAI</des>
    </Parada>
    <Parada>
      <lat>43.267922981782107</lat>
      <lon>-2.1197669592995951</lon>
      <cod>6114</cod>
      <des>SANTIAGO 2</des>
    </Parada>
    <Parada>
      <lat>43.274324230599795</lat>
      <lon>-2.1255037403163719</lon>
      <cod>1014</cod>
      <des>AIA ORIO GELTOKIA</des>
    </Parada>
    <Parada>
      <lat>43.237483467804978</lat>
      <lon>-2.1477934070735847</lon>
```

```
      <cod>1027</cod>
      <des>AIA</des>
    </Parada>
    <Parada>
      <lat>43.241272014935049</lat>
      <lon>-2.14626257342894</lon>
      <cod>1020</cod>
      <des>ETXENAGUSIA</des>
    </Parada>
    <Parada>
      <lat>43.232821752715374</lat>
      <lon>-2.137341193257968</lon>
      <cod>63</cod>
      <des>ILARRAMENDI</des>
    </Parada>
    <Parada>
      <lat>43.273130162165181</lat>
      <lon>-2.1378318344999729</lon>
      <cod>440</cod>
      <des>ALTO ORIO 2</des>
    </Parada>
    <Parada>
      <lat>43.2767427133143</lat>
      <lon>-2.1250131156935819</lon>
      <cod>761</cod>
      <des>ORIOKO GELTOKIA (EuskoTren)</des>
    </Parada>
  </paradas>
</InfoLinea>
```

D.14. kode-zatia: 360 linearen informazioa

Lurraldebus-eko webgunetik autobus bat geltoki batetik noiz iritsiko den jakin daiteke, horretarako GET motako deia egin behar da, D.14. taulan ikusten den bezala bi parametro adierazi behar dira, geltokiaren eta linearen identifikatzaileak.

http://www.lurraldebus.net/WS_Lbw/Service.asmx/Estima?codParada=***[zenbakia]***&idLinea=***[zenbakia]***

D.14. taula: Geltoki batekiko ordutegiak

Ez hori bakarrik D.15. taulako webgunean hainbat zerbitzu daude, hauek erraz erabil daitezke hurrengo informazioa lortzeko:

- Linea guztien zerrenda.
- Lurraldebus-en barnean lan egiten duen konpainien zerrenda eta bakoitzaren lineak.
- Udalerrien zerrenda.
- Linea bakoitzaren geltoki guztien zerrenda.
- Udalerri bakoitzaren geltoki guztien zerrenda.
- Geltoki batetik igarotzen diren lineak eta helduera ordutegiak.

http://www.lurraldebus.net/WS_Lbw/Service.asmx

D.15. taula: Lurraldebus webgunearen online zerbitzuak biltzen dituen web-orrialdea

Oso garrantzitsua da aipatzea proiektuaren hasieran ez zela D.13. taularen webgunea existitzen, antzeko zerbait zeukaten baina ez hainbeste zentralizatua.

Laburtuz, garraio publiko konpainien webgune guztietatik Lurraldebus-ekoa dudarik gabe hoberena da, ia beharrezko datu guztiak eskaintzen ditu eta erabiltzeko ez da zaila. Beste datu iturriak funtzionatuko ez balute hau izango litzateke soluzio paregabea.

D.1.5. Beste konpainiak

Beste konpainien webguneetan nabigatu ondoren ez da aurkitu Internetetik zuzenean datuak hartzeko aukera. Ondorioz, hauen informazioa Open Data-k eskaintzen duen zerbitzutik lortu behar da. Hauek dira arazo hori duten konpainiak:

- Adnor S.L_(Alsa)
- Hermanos Berrotaran Daguer CB
- La Burundesa S.A_(autobuseslaunion)
- La Unión_(autobuseslaunion)

Gipuzkoan beste garraio konpainia asko daudela dirudien arren, hauek LURRALDEBUS enpresaren barnean sartzen dira, ondorioz linea horien informazio guztia aurki daiteke lehen azaldutako atalean.

D.2. Konpainiekin harremanetan

Garraio publiko konpainiekin harremanetan jarri nintzen jakiteko erabili nezakeen web zerbitzu edo API-ren bat bazeukaten. Horrez gain, kasu batzuetan laguntza eskatu nien arrazoi desberdinengatik.

Urriaren 8an mezu elektroniko bat bidali nien, D.16. taulakoa, garraio publikoko konpainia nagusiei, (Dbus, Euskotren, Renfe eta Lurraldebus) galdetuz nik erabil nezakeen web zerbitzu edo API-ren bat bazeukaten.

```
Buenas tardes,

  me llamo Julen Apodaca y soy un estudiante de Grado de
Informática de la Universidad de País Vasco. Estoy desarrollando una
aplicación para móviles con sistema operativo Android sobre los
transportes públicos y me gustaría saber si tienen algún servicio
que yo podría utilizar.

  El asunto en cuestión es que me gustaría saber si tienen algún
servicio web y/o API que se podría utilizar para conseguir una lista
de paradas y/o conseguir la hora de llegada del transporte una
parada indicada. Cualquier cosa que me podáis ofrecer será de gran
```

```
ayuda para el avance del proyecto.

  Perdón por las molestias y gracias de antemano,

         Julen Apodaca.

  PD: Si este mensaje debería ir dirigido a otro correo electrónico
por favor indíquenme a cual, se lo agradeceré.
```

D.16. taula: Garraio publiko konpainiei bidalitako mezu elektronikoa

Renfek erantzun zidan, Urriak 9an, esanez bidalitako mezua beste departamentu batera bidaliko zutela, hala ere, ez zidaten berriro erantzun.

Dbus-ek Urriak 11an, ostiralean, erantzun zidan. Esan zidaten gai hori tratatzeko hobeto zuzenean arduradunarekin hitz egin nezala, telefono bat eta izen bat bidali zidaten berarekin kontaktuan jartzeko. Astelehen goizean, Urriak 14an, deitu nion eta hurrengo egunean bilduko ginela geratu ginen.

Eduki nuen bileran teknikariak esan zidan ez zutela arazorik izango datuen sarbidea emateko. Horretarako erabilpen kontratu bat sinatu beharko nuen. Horretaz aparte, bileran kontatu zidan konpetentziaren aplikazioek dituzten ahultasunak, hau asko eskertu nion.

Hurrengo egunetan kontaktuan jarraitu genuen kontratu sinatua emateko eta beste hainbat gauzaz hitz egiteko.

Urriak 14an Gipuzkoa Transit aplikazioaren garatzaileei mezu elektroniko bat bidali nien laguntza eskatuz, D.17. taula.

```
  Buenos días,

    soy un estudiante de la facultad de Informática de la UPV,
estoy planeando desarrollar una aplicación para móviles con sistema
Android parecida a Gizpuzkoa Transit como proyecto de fin de grado.

     He visto y probado vuestra aplicación, a ser posible me
gustaría poder conocer mejor vuestra aplicación, como conseguís
encontrar las paradas más cercanas, como calculáis las rutas, que
servicios web o API utilizáis...

    Cualquier información será muy agradecida.

    Un cordial saludo,
      Julen Apodaca.
```

D.17. taula: Gipuzkoa Transit garatzaileei bidalitako mezu elektronikoa

Egun berean TGG-rekin kontaktatu nuen baita ere laguntza eskatuz, D.18. taula.

```
  Buenos días,

    soy un estudiante de la facultad de Informática de la UPV,
estoy planeando desarrollar una aplicación para móviles con sistema
Android parecida a Gizpuzkoa Transit como proyecto de fin de grado.

     Me gustaría saber si dicha aplicación utiliza algún servicio
```

```
web o API vuestro, de ser así me gustaría saber si podría yo
utilizarlas para mi aplicación.

     Cualquier información me podáis proporcionar será muy
agradecida.

    Un cordial saludo,
      Julen Apodaca.

  PD: Si este mensaje debería de ser enviado a otro destinatario
por favor indíquenme a cual.
```

D.18. taula: TGG-ri bidalitako mezu elektronikoa

Hurrengo egunean, Urriak 15ean, erantzun zidaten esanez beraiek ez dutela zerbitzurik ematen, zorionez informazioa non bilatu nezakeen gomendatu zidaten: Moveuskadi eta Open Data.

Open Data webgunean behar nuen informazioa aurkitu nituen, hau erabili ahal izateko formulario bat bete behar nuen, baina arazo bat egon zen, datuak eskuraezinak ziren. Orduan mezu elektroniko bat bidali nien beraiei eta MoveEuskadi webguneari, D.19. taulakoa, (suposatzen zen datuak MoveEuskadi webgunean zeudela).

```
Buenas noches,

  me llamo Julen Apodaca y soy un estudiante de la facultad de
Informática de la UPV, estoy planeando desarrollar un proyecto para
el cual necesito unos datos de "opendata.euskadi", más precisamente
los de "Red de transporte público de Euskadi: horarios, paradas,
tiempo, coste y otros datos para planificar rutas".

  En dicha página me dicen que para utilizar los datos he de
rellenar un formulario, y para ello me redirige a la página
"http://moveuskadi.com/?page_id=587". El problema es que esa página
vuestra no muestra nada.

  Me gustaría saber si el formulario para uso de datos está
realmente en esa dirección, y si es así si es simplemente un
problema temporal.

  Perdonen las molestias,
                    Julen Apodaca.
```

D.19. taula: Open Data-ri bidalitako mezu elektronikoa

Urriak 18an erantzun zidaten esanez webgunean aldaketak egiten ari zirela. Baita ere esan zidaten datu publikoak eskaintzeko era aldatu zutela, formularioak bete beharrean, datu guztiak eskuragarri ziren FTP-ren bidez edozeinentzat.

D.2.1. Euskotren-eko arazoa

2014ko Otsailak 4ean, bi mezu idatzi nituen Euskotren-eko arazoari konponbide bat bilatzeko. Informazio okerra egotearen arrazoia nirea ez izan arren, pentsatu nuen beharrezkoa zela egileei abisatzea. Beraiekin kontaktatzeko bi lekutatik egin nuen, bata izan zen Open Data-ko webgunean beraiekin kontaktuan jartzeko web-orrialdetik, D.20. taulakoa, eta bestea MoveEuskadi webguneak beraiekin harremanetan jartzeko ematen duten webgunetik, D.21. taulakoa.

http://opendata.euskadi.net/w79-contgen/es/o53VisualizadorWar/o53BuzonEuskadi.jsp?por=200&lenguaje=1

D.20. taula: Open Data webgunean beraiekin harremanetan jartzeko orria

http://moveuskadi.com/contacto

D.21. taula: Open Data webgunean beraiekin harremanetan jartzeko orria

Biei idatzi nien bi arrazoi nagusiengatik: batetik, biak datuen arduradunak direlako, eta bestetik, askotan mota honetako mezuak ez direlako erantzuten. Hortaz, bi leku desberdinetara bidaliz gero erantzuna jasotzeko probabilitateak igoko ziren.

Hau da idatzi dudan mezua:

```
Buenas noches,

  os escribo para comentaros que he encontrado unos fallos en los
datos que proporcionáis sobre la compañía Euskotren (Tren).

  Estoy utilizando los datos que ofrece Open Data para desarrollar
una aplicación como proyecto de mi carrera, haciendo pruebas y
comprobando datos me he dado cuenta de un par de detalles erróneos
sobre los datos de Euskotren:

  -Los datos de las paradas (stops.txt) están incompletos, se puede
observar que en ellos no existe la estación de Intxaurrondo. Ese
fichero fue modificado por última vez el 20/05/2013, fecha para la
cual dicha estación ya existía, pues está se puso en funcionamiento
al público el 04/10/2012.

  -En el archivo trips.txt se hace una referencia al archivo
shapes.txt, el cual tiene la información necesaria para marcar la
ruta en un mapa. El problema es que me he encontrado casos en los
que dicha información no corresponde con la ruta que debiera ser.
Por ejemplo, la información que se supone que muestra el tramo
Lasarte-Irun en realidad muestra un tramo (ni siquiera uno entero)
de Bizkaia.

  No sé si esto os concierne o debería de escribir un email a los
responsables mismos de la fuente de datos. De ser lo último os
agradecería que me dieseis un correo electrónico o contacto para
poder comentarles directamente los fallos encontrados.

  Gracias por vuestro tiempo,
            Julen Apodaca
```

D.22. taula: Euskotren informazioaren arazoaren abisu mezua

Egun bat beranduago Open Data webgunekoek erantzun zidaten. Bidali zidaten mezuan esaten zuten Moveuskadi webgunea arduratzen zela datu horien kudeaketaz eta nire mezua beraien birzuzendu zietela.

Aste bat pasa ondoren Moveuskadi-k erantzun zidan. Mezuaren esanetara, konpainia bakoitzak informazioa automatikoki sortzen duen zerbitzu pribatu bat dauka eta beraiei bidaltzen dizkiete fitxategiak. Azkenik hauek automatikoki eguneratzen dute webgunea. Arazoa da Euskotren konpainiarekin desberdina dela, informazioa eskuz idatzi behar dute eta Moveuskadi-ri bidali hauek eskuragarri jartzeko. Esan zidaten errorea azalduko zietela eta nire laguntza eskertzen zutela.

Erantzuna jaso bezain laster Open Data webgunean sartu nintzen eskerrak emateko. Hala eta guztiz ere, arazoa ez dute oraindik konpondu.

E Eranskina: Zerbitzaritik jasotako informazioaren egitura

GipuzkoaBidaian zerbitzari propio batekin funtzionatuko balu, berarengandik jasoko lukeen informazioaren egitura Google Directions API-aren oso antzekoa izango litzateke. Kapitulu honetan zerbitzaria bezeroari bidaliko lioken datuen egitura erakutsiko da XML Schema eta JSON Schema bitartez.

XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:mv="http://www.gipuzkoaBidaian.org"
  elementFormDefault="qualified"
  targetNamespace="http://www.gipuzkoaBidaian.org">

  <xs:element name="bidai_emaitza">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="mv:emaitza_egoera" minOccurs="1" maxOccurs="1"/>
        <xs:element ref="mv:emailtzaLista" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="emaitza_egoera" type="mv:emaitza_egoeraT"/>
    <xs:simpleType name="emaitza_egoeraT">
        <xs:restriction base="xs:string">
            <xs:enumeration value="OK"/>
            <xs:enumeration value="URL_ERROR"/>
            <xs:enumeration value="NO_DATA"/>
            <xs:enumeration value="ERROR"/>
        </xs:restriction>
    </xs:simpleType>


    <xs:element name="emailtzaLista" type="mv:emailtzaListaT"/>
    <xs:complexType name="emailtzaListaT">
        <xs:sequence>
          <xs:element name="garraioKop" type="xs:int"/>
          <xs:element ref="mv:bounds" minOccurs="0"/>
          <xs:element name="copyrights" minOccurs="0" type="xs:string"/>
          <xs:element ref="mv:overviewPolyline" minOccurs="0"
maxOccurs="unbounded"/>
          <xs:element name="summary" minOccurs="0" type="xs:string"/>
          <xs:element ref="mv:warnings" minOccurs="0"
```

```
maxOccurs="unbounded"/>
          <xs:element ref="mv:waypointOrders" maxOccurs="unbounded"/>
          <xs:element ref="mv:departureTime"/>
          <xs:element ref="mv:arrivalTime" />
          <xs:element name="end_address" type="xs:string"/>
          <xs:element ref="mv:endLocation"/>
          <xs:element name="start_address" type="xs:string"/>
          <xs:element ref="mv:startLocation" />
          <xs:element ref="mv:distance"/>
          <xs:element ref="mv:duration"/>
          <xs:element ref="mv:steps" maxOccurs="unbounded"/>
          <xs:element ref="mv:via_waypoint" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:element name="bounds" type="mv:boundsT"/>
      <xs:complexType name="boundsT">
        <xs:sequence>
          <xs:element ref="mv:northeast"/>
          <xs:element ref="mv:southwest"/>
        </xs:sequence>
    </xs:complexType>

    <xs:element name="endLocation" type="mv:koordenadaT"/>
    <xs:element name="startLocation" type="mv:koordenadaT"/>
    <xs:element name="northeast" type="mv:koordenadaT"/>
    <xs:element name="southwest" type="mv:koordenadaT"/>
    <xs:element name="location" type="mv:koordenadaT"/>
     <xs:complexType name="koordenadaT">
        <xs:sequence>
          <xs:element ref="mv:koordenadaLat"/>
          <xs:element ref="mv:koordenadaLon"/>
        </xs:sequence>
    </xs:complexType>

    <xs:element name="koordenadaLat" type="koordenadaLatT"/>
      <xs:complexType name="koordenadaLatT">
        <xs:sequence>
          <xs:element name="lat" type="xs:string" fixed="lat"/>
          <xs:element name="koordenatua"
type="lurrekoKoordenatuLatitudeaT"/>
        </xs:sequence>
    </xs:complexType>

    <xs:element name="koordenadaLon" type="koordenadaLonT"/>
    <xs:complexType name="koordenadaLonT">
        <xs:sequence>
          <xs:element name="lng" type="xs:string" fixed="lng"/>
          <xs:element name="koordenatua"
type="lurrekoKoordenatuLongitudeaT"/>
        </xs:sequence>
    </xs:complexType>

    <xs:element name="koordenada_mota" type="mvv:koordenada_motaT"/>
    <xs:simpleType name="koordenada_motaT">
        <xs:restriction base="xs:string">
            <xs:enumeration value="lat"/>
            <xs:enumeration value="lng"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="lurrekoKoordenatuLatitudeaT">
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="-90"/>
            <xs:maxInclusive value="+90"/>
        </xs:restriction>
```

```
    </xs:simpleType>

    <xs:simpleType name="lurrekoKoordenatuLongitudeaT">
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="-180"/>
            <xs:maxInclusive value="+180"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:element name="overviewPolyline" type="mv:polylineT"/>
    <xs:element name="polyline" type="mv:polylineT"/>
      <xs:complexType name="polylineT">
        <xs:sequence>
          <xs:element name="polylineCoordHashMap" type="mv:koordenadaT"/>
        </xs:sequence>
    </xs:complexType>


    <xs:element name="warnings" type="xs:string"/>
    <xs:element name="waypointOrders" type="xs:int"/>


    <xs:element name="departureTime" type="mv:TextValueTimeT"/>
    <xs:element name="arrivalTime" type="mv:TextValueTimeT"/>
      <xs:complexType name="TextValueTimeT">
        <xs:sequence>
            <xs:element name="orduTestua" type="xs:string"/>
            <xs:element name="orduBalioa" type="xs:int"/>
            <xs:element name="orduEremua" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>


    <xs:element name="distance" type="mv:TextValueT"/>
    <xs:element name="duration" type="mv:TextValueT"/>
      <xs:complexType name="TextValueT">
        <xs:sequence>
            <xs:element name="orduTestua" type="xs:string"/>
            <xs:element name="orduBalioa" type="xs:int"/>
        </xs:sequence>
    </xs:complexType>


    <!-- HEMENDIK AURRERA STEPS ZATIA -->
    <xs:element name="steps" type="mv:stepsT"/>
      <xs:complexType name="stepsT">
        <xs:sequence>
          <xs:element ref="mv:distance"/>
          <xs:element ref="mv:duration"/>
          <xs:element name="endLocation" type="mv:koordenadaT"/>
          <xs:element name="startLocation" type="mv:koordenadaT"/>
          <xs:element ref="mv:polyline" maxOccurs="unbounded"/>
          <xs:element name="travelMode" type="xs:string"/>
          <xs:element name="htmlInstructions" type="xs:string"/>
          <xs:element ref="mv:steps" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="mv:transitDetails" minOccurs="0"/>
          <xs:element name="maneuver" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>


    <xs:element name="transitDetails" type="mv:transitDetailsT"/>
      <xs:complexType name="transitDetailsT">
        <xs:sequence>
          <xs:element ref="mv:departureTime"/>
          <xs:element ref="mv:arrivalTime"/>
          <xs:element ref="mv:arrivalStop" />
```

```
          <xs:element ref="mv:departureStop" />
          <xs:element name="headsign" type="xs:string"/>
          <xs:element ref="mv:line"/>
          <xs:element name="numStops" type="xs:int"/>
        </xs:sequence>
    </xs:complexType>

    <xs:element name="arrivalStop" type="mv:stopT"/>
    <xs:element name="departureStop" type="mv:stopT"/>
      <xs:complexType name="stopT">
        <xs:sequence>
          <xs:element ref="mv:location"/>
          <xs:element name="name" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>

    <xs:element name="line" type="mv:lineT"/>
      <xs:complexType name="lineT">
        <xs:sequence>
          <xs:element ref="mv:agencies" maxOccurs="unbounded"/>
          <xs:element name="name" type="xs:string"/>
          <xs:element name="shortName" type="xs:string"/>
          <xs:element name="color" type="xs:string" />
          <xs:element name="url" type="xs:string" />
          <xs:element name="icon" type="xs:string" />
          <xs:element name="textColor" type="xs:string"/>
          <xs:element name="vehicleName" type="xs:string"/>
          <xs:element name="vehicleType" type="xs:string"/>
          <xs:element name="vehicleIcon" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>

    <xs:element name="agencies" type="mv:agencieT"/>
      <xs:complexType name="agencieT">
        <xs:sequence>
          <xs:element name="name" type="xs:string"/>
          <xs:element name="phone" type="xs:string"/>
          <xs:element name="url" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
  </xs:schema>
```

E.1. kode-zatia: XML Schema

JSON Schema

E.2. kode-zatiaren luzea murrizteko “GipuzkoaBidaian” ordez “GB” idatzi da.

```
{
  "type":"object",
  "$schema": "http://json-schema.org/draft-03/schema",
  "id": "http://GB.net",
  "required":true,
  "properties":{
    "emailtzaLista": {
      "type":"array",
      "minitems": "1",
```

```
      "maxitems": "1",
      "id": "http://GB.net/emailtzaLista",
      "required":true,
      "items":
        {
          "type":"object",
          "id": "http://GB.net/emailtzaLista/0",
          "required":true,
          "properties":{
            "bounds": {
              "type":"object",
              "id": "http://GB.net/emailtzaLista/0/bounds",
              "required":false,
              "properties":{
                "northeast": {
                  "type":"object",
                  "id": "http://GB.net/emailtzaLista/0/bounds/northeast",
                  "required":true,
                  "properties":{
                    "lat": {
                      "type":"number",
                      "id":
"http://GB.net/emailtzaLista/0/bounds/northeast/lat",
                      "required":true
                    },
                    "lng": {
                      "type":"number",
                      "id":
"http://GB.net/emailtzaLista/0/bounds/northeast/lng",
                      "required":true
                    }
                  }
                },
                "southwest": {
                  "type":"object",
                  "id": "http://GB.net/emailtzaLista/0/bounds/southwest",
                  "required":true,
                  "properties":{
                    "lat": {
                      "type":"number",
                      "id":
"http://GB.net/emailtzaLista/0/bounds/southwest/lat",
                      "required":true
                    },
                    "lng": {
                      "type":"number",
                      "id":
"http://GB.net/emailtzaLista/0/bounds/southwest/lng",
                      "required":true
                    }
                  }
                }
              }
            },
            "copyrights": {
              "type":"string",
              "id": "http://GB.net/emailtzaLista/0/copyrights",
              "required":false
            },
            "garraioKop": {
              "type":"string",
              "id": "http://GB.net/emailtzaLista/0/garraioKop",
              "required":true
            },
            "legs": {
              "type":"array",
              "id": "http://GB.net/emailtzaLista/0/legs",
```

```
                "required":true,
                "items":
                  {
                    "type":"object",
                    "id": "http://GB.net/emailtzaLista/0/legs/0",
                    "required":true,
                    "properties":{
                      "arrival_time": {
                        "type":"object",
                        "id":
"http://GB.net/emailtzaLista/0/legs/0/arrival_time",
                        "required":true,
                        "properties":{
                          "text": {
                            "type":"string",
                            "id":
"http://GB.net/emailtzaLista/0/legs/0/arrival_time/text",
                            "required":true
                          },
                          "time_zone": {
                            "type":"string",
                            "id":
"http://GB.net/emailtzaLista/0/legs/0/arrival_time/time_zone",
                            "required":false
                          },
                          "value": {
                            "type":"number",
                            "id":
"http://GB.net/emailtzaLista/0/legs/0/arrival_time/value",
                            "required":true
                          }
                        }
                      },
                      "departure_time": {
                        "type":"object",
                        "id":
"http://GB.net/emailtzaLista/0/legs/0/departure_time",
                        "required":true,
                        "properties":{
                          "text": {
                            "type":"string",
                            "id":
"http://GB.net/emailtzaLista/0/legs/0/departure_time/text",
                            "required":true
                          },
                          "time_zone": {
                            "type":"string",
                            "id":
"http://GB.net/emailtzaLista/0/legs/0/departure_time/time_zone",
                            "required":false
                          },
                          "value": {
                            "type":"number",
                            "id":
"http://GB.net/emailtzaLista/0/legs/0/departure_time/value",
                            "required":true
                          }
                        }
                      },
                      "distance": {
                        "type":"object",
                        "id":
"http://GB.net/emailtzaLista/0/legs/0/distance",
                        "required":true,
                        "properties":{
                          "text": {
                            "type":"string",
```

```
                            "id":
"http://GB.net/emailtzaLista/0/legs/0/distance/text",
                            "required":true
                          },
                          "value": {
                            "type":"number",
                            "id":
"http://GB.net/emailtzaLista/0/legs/0/distance/value",
                            "required":true
                          }
                        }
                      },
                      "duration": {
                        "type":"object",
                        "id":
"http://GB.net/emailtzaLista/0/legs/0/duration",
                        "required":true,
                        "properties":{
                          "text": {
                            "type":"string",
                            "id":
"http://GB.net/emailtzaLista/0/legs/0/duration/text",
                            "required":true
                          },
                          "value": {
                            "type":"number",
                            "id":
"http://GB.net/emailtzaLista/0/legs/0/duration/value",
                            "required":true
                          }
                        }
                      },
                      "end_address": {
                        "type":"string",
                        "id":
"http://GB.net/emailtzaLista/0/legs/0/end_address",
                        "required":true
                      },
                      "end_location": {
                        "type":"object",
                        "id":
"http://GB.net/emailtzaLista/0/legs/0/end_location",
                        "required":true,
                        "properties":{
                          "lat": {
                            "type":"number",
                            "id":
"http://GB.net/emailtzaLista/0/legs/0/end_location/lat",
                            "required":true
                          },
                          "lng": {
                            "type":"number",
                            "id":
"http://GB.net/emailtzaLista/0/legs/0/end_location/lng",
                            "required":true
                          }
                        }
                      },
                      "start_address": {
                        "type":"string",
                        "id":
"http://GB.net/emailtzaLista/0/legs/0/start_address",
                        "required":true
                      },
                      "start_location": {
                        "type":"object",
                        "id":
```

```
"http://GB.net/emailtzaLista/0/legs/0/start_location",
                      "required":true,
                      "properties":{
                        "lat": {
                          "type":"number",
                          "id":
"http://GB.net/emailtzaLista/0/legs/0/start_location/lat",
                          "required":true
                        },
                        "lng": {
                          "type":"number",
                          "id":
"http://GB.net/emailtzaLista/0/legs/0/start_location/lng",
                          "required":true
                        }
                      }
                    },
                    "steps": {
                      "type":"array",
                      "id": "http://GB.net/emailtzaLista/0/legs/0/steps",
                      "required":true,
                      "items":
                        {
                          "type":"object",
                          "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0",
                          "required":true,
                          "properties":{
                            "distance": {
                              "type":"object",
                              "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/distance",
                              "required":true,
                              "properties":{
                                "text": {
                                  "type":"string",
                                  "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/distance/text",
                                  "required":true
                                },
                                "value": {
                                  "type":"number",
                                  "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/distance/value",
                                  "required":true
                                }
                              }
                            },
                            "duration": {
                              "type":"object",
                              "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/duration",
                              "required":true,
                              "properties":{
                                "text": {
                                  "type":"string",
                                  "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/duration/text",
                                  "required":true
                                },
                                "value": {
                                  "type":"number",
                                  "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/duration/value",
                                  "required":true
                                }
                              }
```

```
                            },
                            "end_location": {
                              "type":"object",
                              "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/end_location",
                              "required":true,
                              "properties":{
                                "lat": {
                                  "type":"number",
                                  "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/end_location/lat",
                                  "required":true
                                },
                                "lng": {
                                  "type":"number",
                                  "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/end_location/lng",
                                  "required":true
                                }
                              }
                            },
                            "html_instructions": {
                              "type":"string",
                              "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/html_instructions",
                              "required":false
                            },
                            "polyline": {
                              "type":"array",
                              "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/polyline",
                              "required":true,
                              "properties":{
                                "lat": {
                                  "type":"number",
                                  "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/polyline/lat",
                                  "required":true
                                },
                                "lng": {
                                  "type":"number",
                                  "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/polyline/lng",
                                  "required":true
                                }
                              }
                            },
                            "start_location": {
                              "type":"object",
                              "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/start_location",
                              "required":true,
                              "properties":{
                                "lat": {
                                  "type":"number",
                                  "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/start_location/lat",
                                  "required":true
                                },
                                "lng": {
                                  "type":"number",
                                  "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/start_location/lng",
                                  "required":true
                                }
                              }
                            },
```

```
                            "steps": {
                              "type":"array",
                              "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/steps",
                              "required":false,
                              "items":
                                {
                                  "type":"object",
                                  "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/steps/0",
                                  "required":true,
                                  "properties":{
                                    "distance": {
                                      "type":"object",
                                      "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/steps/0/distance",
                                      "required":true,
                                      "properties":{
                                        "text": {
                                          "type":"string",
                                          "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/steps/0/distance/text",
                                          "required":true
                                        },
                                        "value": {
                                          "type":"number",
                                          "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/steps/0/distance/value",
                                          "required":true
                                        }
                                      }
                                    },
                                    "duration": {
                                      "type":"object",
                                      "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/steps/0/duration",
                                      "required":true,
                                      "properties":{
                                        "text": {
                                          "type":"string",
                                          "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/steps/0/duration/text",
                                          "required":true
                                        },
                                        "value": {
                                          "type":"number",
                                          "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/steps/0/duration/value",
                                          "required":true
                                        }
                                      }
                                    },
                                    "end_location": {
                                      "type":"object",
                                      "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/steps/0/end_location",
                                      "required":true,
                                      "properties":{
                                        "lat": {
                                          "type":"number",
                                          "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/steps/0/end_location/lat",
                                          "required":true
                                        },
                                        "lng": {
                                          "type":"number",
                                          "id":
```

```
"http://GB.net/emailtzaLista/0/legs/0/steps/0/steps/0/end_location/lng",
                                        "required":true
                                      }
                                    }
                                  },
                                  "html_instructions": {
                                    "type":"string",
                                    "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/steps/0/html_instructions",
                                    "required":false
                                  },
                                  "polyline": {
                                    "type":"array",
                                    "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/steps/0/polyline",
                                    "required":true,
                                    "properties":{
                                      "lat": {
                                        "type":"number",
                                        "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/steps/0/polyline/lat",
                                        "required":true
                                      },
                                      "lng": {
                                        "type":"number",
                                        "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/steps/0/polyline/lng",
                                        "required":true
                                      }
                                    }
                                  },
                                  "start_location": {
                                    "type":"object",
                                    "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/steps/0/start_location",
                                    "required":true,
                                    "properties":{
                                      "lat": {
                                        "type":"number",
                                        "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/steps/0/start_location/lat",
                                        "required":true
                                      },
                                      "lng": {
                                        "type":"number",
                                        "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/steps/0/start_location/lng",
                                        "required":true
                                      }
                                    }
                                  },
                                  "travel_mode": {
                                    "type":"string",
                                    "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/steps/0/travel_mode",
                                    "required":true
                                  }
                                }
                              }
                            },
                            "travel_mode": {
                              "type":"string",
                              "id":
"http://GB.net/emailtzaLista/0/legs/0/steps/0/travel_mode",
                              "required":true
                            }
                          }
```

```
                        }
                    },
                    "via_waypoint": {
                      "type":"array",
                      "id":
"http://GB.net/emailtzaLista/0/legs/0/via_waypoint",
                      "required":false
                    }
                  }
                }
            },
            "overview_polyline": {
              "type":"array",
              "id": "http://GB.net/emailtzaLista/0/overview_polyline",
              "required":true,
              "properties":{
                "lat": {
                  "type":"number",
                  "id":
"http://GB.net/emailtzaLista/0/overview_polyline/lat",
                  "required":true
                },
                "lng": {
                  "type":"number",
                  "id":
"http://GB.net/emailtzaLista/0/overview_polyline/lng",
                  "required":true
                }
              }
            },
            "summary": {
              "type":"string",
              "id": "http://GB.net/emailtzaLista/0/summary",
              "required":false
            },
            "warnings": {
              "type":"array",
              "id": "http://GB.net/emailtzaLista/0/warnings",
              "required":false,
              "items":
                {
                  "type":"string",
                  "id": "http://GB.net/emailtzaLista/0/warnings/0",
                  "required":true
                }
            },
            "waypoint_order": {
              "type":"array",
              "id": "http://GB.net/emailtzaLista/0/waypoint_order",
              "required":false
            }
          }
        }
    },
    "emaitza_egoera": {
      "type":"string",
      "id": "http://GB.net/emaitza_egoera",
      "required":true
    }
  }
}
```

E.2. kode-zatia: JSON Schema

F Eranskina: Inplementazioaren kode zatiak

Eranskin honetan 4. kapituluan, Garapena, sartu ez diren kode-zatiak erakutsiko dira. Kode zati hauek egiten dutena dagoeneko azaldu da dagokion kapituluan sekuentzia diagrama batekin, leku gutxiago okupatzeko eta ulergarriagoa izateko. Orain ordea, aplikazioan erabili den kodea erakutsiko da informazio gehigarri bezala.

Bilaketa prestaketa – jatorria lortu

F.1. kode-zatiak jatorria konprobatzen duen lehenengo metodoa irudikatzen du. Hemen lehenengo konprobaketak egiten dira eta dena ondo dagoela ziurtatzean helmuga konprobatzeko metodoari deitzen dio exekuzioarekin jarraitzeko.

```
public void jatorriaKonprobatu(){
   if(JatorriHelbidea.getInstance().hasieratuDa()){

if(actvJatorria.getText().toString().compareTo(JatorriHelbidea.getIn
stance().getHelbFisikoa())!=0){
             if(konexioaDago()){
              placesTaskDeitzailea="JatorriHelbidea";
              placesTask = new PlacesTask();
              placesTask.execute(actvJatorria.getText()
                                  .toString());
             }
             else {
                Toast.makeText(getBaseContext(),
                 res.getText(R.string.abisuInternetEzKokapenaMapaz),
                 Toast.LENGTH_LONG).show();
             }
          }
          else{
          helmugaKonprobatu();
          }
    }
    else{
       if(konexioaDago()){
              placesTaskDeitzailea="JatorriHelbidea";
              placesTask = new PlacesTask();
              placesTask.execute(actvJatorria.getText().toString());
       }
       else {
             Toast.makeText(getBaseContext(),
```

```
                    res.getText(R.string.abisuInternetEzKokapenaMapaz),
                    Toast.LENGTH_LONG).show();
            }
        }
}
```

F.1. kode-zatia: Jatorria lortu

Mugikorrak edo tabletak Internetera konexioa duela ziurtatzeko F.2. kode-zatiko funtzioari deituko dio.

```
//atal hau beste paketeetan egongo diren fitxategiak erabiliko ditu
public boolean konexioaDago(){
        Boolean isInternetPresent = false;

        // Konexioa detektatzeko klasearen aldagaia
        ConnectionDetector cd;

        // Konexio detektatzaile klasearen instantzia
        cd = new ConnectionDetector(getApplicationContext());

        // Internetaren egoera lortu eta gorde
        isInternetPresent = cd.isConnectingToInternet();

        return isInternetPresent;
}
```

F.2. kode-zatia: Internet konexioa konprobatu

Helbide fisiko bat koordenada batean bihurtzeko, lehenengo urratsa F.3. kode-zatiko klaseari deitzea da, helbidea parametro bezala bidaliz. Honek hainbat helbide posibleak lortuko ditu (normalean erabiltzaileak idazten duena ez da oso zehatza) eta hurrengo klaseari bidalizko dizkio, F.4. kode-zatikoari. Honek Internetetik jasotako informazioari formatu egokia ematen dio eta ondoren dagokion metodoari deitzen dio.

```
// GooglePlaces AutoComplete Web Zerbitzutik lortu helbide posible guztak
private class PlacesTask extends AsyncTask<String, Void, String>{

   @Override
   protected String doInBackground(String... place) {
        String data = "";
        String input="";

        try {
            input = "input=" + URLEncoder.encode(place[0], "utf-8");
        } catch (UnsupportedEncodingException e1) {
            e1.printStackTrace();
        }

        // Web zerbitzurako parametroak zehaztu
        String parameters = input+"&"+Konfig.getApiTypesGeocode()
                +"&"+Konfig.getApiSensor()+"&"+Konfig.getApiKey()
                +"&"+Konfig.getApiComponentsEs();

        // Web zerbitzurako url-a sortu
        String url = Konfig.getApiDirAutocomplete()
```

```
                    +Konfig.getApiOutputJson()+"?"+parameters;
        try{
            // Deskargatu informazioa internetik
            data = DownloadUrl.getInstance()
                              .deskargatuInformazio(url);
        }catch(Exception e){
            Log.d("Background Task",e.toString());
        }
        return data;
    }

    @Override
    protected void onPostExecute(String result) {
        super.onPostExecute(result);

        // Sortu ParserTask
        parserTask = new ParserTask();

        // Jarraitu bilaketaren exekuzioarekin lortutako  informazioarekin
        parserTask.execute(result);
    }
}
```

F.3. kode-zatia: URL-aren prestaketa eta informazioaren deskarga

```
private class ParserTask extends AsyncTask<String, Integer,
List<HashMap<String,String>>>{

    JSONObject jObject;

    @Override
    protected List<HashMap<String, String>> doInBackground(String...
jsonData) {

        List<HashMap<String, String>> places = null;

        PlaceJSONParser placeJsonParser = new PlaceJSONParser();

        try{
            jObject = new JSONObject(jsonData[0]);

            places = placeJsonParser.parse(jObject);

        }catch(Exception e){
            Log.d("Exception",e.toString());
        }
        return places;
    }

    @Override
    protected void onPostExecute(List<HashMap<String, String>>
result)
    {
      // Hau sortu da testu eremuan aukera lista erakusteko naiz eta oraingoz ez inplementatu

if(placesTaskDeitzailea.compareTo("JatorriAutCompTextView")==0){
          String[] from = new String[] { "description"};
          int[] to = new int[] { android.R.id.text1 };
          // Sortu SimpleAdapter-a nire AutoCompleteTextView eremuarentzat
```

```
            SimpleAdapter adapter = new SimpleAdapter(
                        getBaseContext(),
                        result,
                        android.R.layout.simple_list_item_1,
                        from,
                        to);
             // Adapter-a zehaztu
             actvJatorria.setAdapter(adapter);
            //aukeratutakoaren arabera JatorriHelbidea klasearen elementuak zehaztu
        }

        // Hau sortu da testu eremuan aukera lista erakusteko naiz eta oraingoz ez inplementatu

if(placesTaskDeitzailea.compareTo("HelmugaAutCompTextView")==0){
            String[] from = new String[] { "description"};
            int[] to = new int[] { android.R.id.text1 };
            // Sortu SimpleAdapter-a nire AutoCompleteTextView eremuarentzat
            SimpleAdapter adapter = new SimpleAdapter(
                        getBaseContext(),
                        result,
                        android.R.layout.simple_list_item_1,
                        from,
                        to);
            // Adapter-a zehaztu
            actvHelmuga.setAdapter(adapter);
            //aukeratutakoaren arabera HelmugaHelbidea klasearen elementuak zehaztu
        }

        if(placesTaskDeitzailea.compareTo("JatorriHelbidea")==0){
            jatorriaDeduzitu(result);
        }

        if(placesTaskDeitzailea.compareTo("HelmugaHelbidea")==0){
            helmugaDeduzitu(result);
        }
    }
}
```

F.4. kode-zatia: ParserTask klasearen kodea

Helbide posibleak lortzean pantailan erakutsi behar dira, erabiltzaileak nahi duena aukeratzeko. Honetaz arduratzen den metodoa "*jatorriaDeduzitu*" da, F.5. kode-zatian bere barruko kodea azaltzen da. Metodo honetatik azkeneko urratsari deitzen zaio, F.6 kode-zatikoa, aukeratutako helbidearen datu guztiak lortzen dituena eta gordetzen dituenak.

```
private void jatorriaDeduzitu(final List<HashMap<String, String>>
addresses){
        final ArrayList<String> helbidePosibleLista =
                                    new ArrayList<String>();
        final HashMap<String, String> helbidea =
                                        new HashMap<String, String>();
        helbidea.put("deitzailea", "Jatorria");

        if(addresses.size()==1){

        actvJatorria.setText(addresses.get(0).get("description"));
              helbidea.put("helbidea",
                            addresses.get(0).get("description"));
```

```
            new lortuGeocoderHelbidea().execute(helbidea);
        }
        else{
            // Prestatu erakutsiko den helbide zerrenda
            for(int i=0;i<addresses.size();i++){
                    helbidePosibleLista.add(addresses.get(i)
                                            .get("description"));
            }

            final CharSequence[] helbList =
                        helbidePosibleLista.toArray (
                        new CharSequence[helbidePosibleLista.size()]);

            AlertDialog.Builder builder3 =
                            new AlertDialog.Builder (
                                MainActivity.this);
            builder3
             .setTitle(res.getText
                      (R.string.alertDialog_zehaztu_jatorria))
             .setItems(helbList,
                       new DialogInterface.OnClickListener() {

                    @Override
                    public void onClick(DialogInterface dialog,
                                        int which) {
                          actvJatorria.setText(addresses
                                               .get(which)
                                               .get("description"));
                          helbidea.put("helbidea",
                                       addresses.get(which)
                                                .get("description"));
                          //aukeratutako helbidearen informazio osoa lortu
                          new LortuGeocoderHelbidea()
                              .execute(helbidea);
                    }
              });

            final AlertDialog alert = builder3.create();
            builder3.setNegativeButton("Cancel",
                                new DialogInterface.OnClickListener()
{
                    public void onClick(DialogInterface dialog,
                                        int which) {
                          alert.cancel();
                          helmugaKonprobatu();   //exekuzioarekin jarraituko da

                    }
            });

            //Helbide aukerak erakutsi
            builder3.show();
        }
}
```

F.5. kode-zatia: Helbide zerrenda erakutsi

```
// Klase honen bitartez helbide fisiko bat [Jatorria/Helmuga]Helbidea klasean bihurtuko dut, hau da, helbide inkompleto batetik latitudea, longitudea, hiria, herria eta helbide ona lortuko dut
private class LortuGeocoderHelbidea extends
AsyncTask<HashMap<String, String>, Void, List<Address>>{
	String deitzailea;
	String helbideOsoa;

	@Override
	protected List<Address> doInBackground(HashMap<String,
                                               String>...
                                               locationName)
	{
		// Creating an instance of Geocoder class
		Geocoder geocoder = new Geocoder(getBaseContext());
		List<Address> addresses = null;
		deitzailea="";
		try {
			// Hiru elementuak: haukeratutako helbidea, bere informazio osoa eta deitzailea
			helbideOsoa = locationName[0].get("helbidea");
			addresses = geocoder.getFromLocationName(
                                locationName[0].get("helbidea"), 1);
			deitzailea = locationName[0].get("deitzailea");
		} catch (IOException e) {
			e.printStackTrace();
		}
		return addresses;
	}

	@Override
	protected void onPostExecute(List<Address> addresses) {
		if(addresses==null || addresses.size()==0){
			//arrazoi batengatik ezin izan bada helbidearen latitudea eta longitudea lortu
			Toast.makeText(getBaseContext(),
                                res.getText(R.string.
                                            abisuKokapenaEzAurktu),
                                Toast.LENGTH_SHORT).show();
		}
		else{
			Address address = (Address) addresses.get(0);
			String herria = ""; //hau nulua izan daiteke
			String hiria = "";  //hau nulua izan daiteke
			String helbMotza =
                                address.getMaxAddressLineIndex() > 0 ?
                                        address.getAddressLine(0) : "";

			LatLng latLng = new LatLng(address.getLatitude(),
                                                address.getLongitude());

			if(address.getCountryName()!=null) {
                                herria = address.getCountryName();
			}

			if(address.getLocality()!=null){
                                hiria = address.getLocality();
			}


			if(deitzailea.compareTo("Jatorria")==0){
                                JatorriHelbidea.getInstance().
                                                setHelbFisikoa(helbideOsoa);
```

```
                        JatorriHelbidea.getInstance().
                                        setLatLng(latLng);
                        JatorriHelbidea.getInstance().
                                      setLatitudea(latLng.latitude);
                        JatorriHelbidea.getInstance().
                                    setLongitudea(latLng.longitude);
                        JatorriHelbidea.getInstance().
                                        setHelbMotza(helbMotza);
                        JatorriHelbidea.getInstance().
                                        setHerria(herria);
                        JatorriHelbidea.getInstance().
                                        setHiria(hiria);

                        //behin jatorria aukeratuta helmuga finkatu behar da
                        helmugaKonprobatu();   //exekuzioarekin jarraitu
                    }

                    if(deitzailea.compareTo("Helmuga")==0){
                        HelmugaHelbidea.getInstance().
                                        setHelbFisikoa(helbideOsoa);
                        HelmugaHelbidea.getInstance().
                                        setLatLng(latLng);
                        HelmugaHelbidea.getInstance().
                                      setLatitudea(latLng.latitude);
                        HelmugaHelbidea.getInstance().
                                    setLongitudea(latLng.longitude);
                        HelmugaHelbidea.getInstance().
                                        setHelbMotza(helbMotza);
                        HelmugaHelbidea.getInstance().
                                        setHerria(herria);
                        HelmugaHelbidea.getInstance().
                                        setHiria(hiria);

                        //honera iritsi bada orduan jatorria dagoeneko finkatu da (hala ez,
                          agian  ezeztatzeko botoia sakatu du)
                        //  ondorioz begiratzen da jatorria eta helmuga hasieratu direla eta
                          zuzenak direla bilaketaren hurrengo urratsarekin jarraitu ahal
                          izateko

      if(JatorriHelbidea.getInstance().hasieratuDa()  &&
HelmugaHelbidea.getInstance().hasieratuDa()){

      if(actvJatorria.getText().toString().compareTo(JatorriHelbidea
.getInstance().getHelbFisikoa())==0  &&

      actvHelmuga.getText().toString().compareTo(HelmugaHelbidea.get
Instance().getHelbFisikoa())==0){
                                bidaiBilaketa();//helbideak izanda bilaketa hasi
                                }
                        }
                    }
                } //if(addresses!=null or addresses.size()!=0) kasua
      } //onPostExecute funtzioa
}
```

F.6. kode-zatia: Helbide osoa lortu

F.7. kode-zatian Internetik informazioa deskargatzeko erabiltzen den klasearen kodea dago. Klase hau beste klaseetatik erabiltzen da informazioa deskargatzeko, Singleton patroia erabiltzen duenez ez da beraren objektu bat sortu behar erabiltzeko.

```
package com.example.services;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

import android.util.Log;

public class DownloadUrl {

      private static DownloadUrl downloadUrl;

      private DownloadUrl(){ }

      public static DownloadUrl getInstance(){
            if (downloadUrl==null) {
                  downloadUrl= new DownloadUrl();
          }
          return downloadUrl;
      }

      /**
       * Adierazitako "strUrl"-era konektatzen da eta han lortutako informazioa
       * deskargatzen du. Klase hau erabiltzen oi da web zerbitzuetaz baliatzeko eta eskaitzen
       * duten datuak deskargatzeko.
       * Adibidez: Google-eko zerbitzuei deituz XML edo JSON formatuko informazioa lortu
       * daiteke, hau String batean gordeko da ondoren formatua emateki.
       * @param strUrl: String -> Informazioa deskargatu behar den helbide iturria.
       * @return String -> Lortutako informazioa String batean.
       * @throws IOException -> Sarrera irteera motako errore bat gertatzean hau igoko du klase
          deitzaileari
       */
    public String deskargatuInformazio(String strUrl) throws
IOException{
        String data = "";
        InputStream iStream = null;
        HttpURLConnection urlConnection = null;
        try{
            URL url = new URL(strUrl);

            // HTTP konexioa sortu URL-arekin komunikatzeko
            urlConnection = (HttpURLConnection) url
                            .openConnection();

            // URL-era konektatu
            urlConnection.connect();

            // Datuak irakurri URL-etik
            iStream = urlConnection.getInputStream();

            BufferedReader br = new BufferedReader(
                                    new InputStreamReader(iStream));
```

```
            StringBuffer sb = new StringBuffer();

            String line = "";
            while( ( line = br.readLine()) != null){
                sb.append(line); //Gorde lerro bekoitza
            }
                  //Bihurtu deskargatutako informazio String batean
            data = sb.toString();

            br.close();

        }catch(Exception e){
            Log.d("Errorea informazioa deskargatzean",
                  e.toString());
        }finally{
            iStream.close();
            urlConnection.disconnect();
        }
        return data;
    }
}
```

F.7. kode-zatia: DownloadUrl klasea

Bilaketa prozesuaren egitura – ontzi aldaketak lortu

Atal honetan 4. kapituluko 4.22. irudiaren sekuentzia diagramari dagokion kodea erakutsiko da. Ontzi aldaketak bilatzeko prozesua hurrengoa da: lehenik jakin behar da jatorritik eta helmugatik igarotzen diren konpainiak (datu hau funtzioari bidaltzen zaio, *direktorioak)*. Hauen bikoteak izanda bakoitzaren geltokiak lortzen dira eta distantzia maximoa baino gertuago al dauden begiratzen da. Horrela bada, horrek esan nahi du bi konpaini horiek bidaia osoa egin dezaketela.

```
/**
 * Konpainia bikoteak analizatuko dira, jatorritik igarotzen direnak eta helmugatik igarotzen direnen
geltokiak konparatuko dira gertu daudela ikusteko, hau da, ontzi aldaketa
 * egin daitekeela ikusteko (erabiltzaileak ezarritako distantzia maximoa baino gertuago daudela).
*/
public ArrayList<BidaiEmaitza> bilaketaKonposatuaOpenDatarekin(
                                        List<KonpBik> direktorioak,
                                        boolean konpainiaBakarra){
      ArrayList<BidaiEmaitza> bidaiEmaitzak =
                                    new ArrayList<BidaiEmaitza>();
//bidaia zuzena egin dezaketenentzat
      List <TransbordoSinplea> ontziAldaketaSinpleak =
                                    new ArrayList<TransbordoSinplea>();
 //hemen gordeko dira aurreko listan gordetzen ez diren bikoteak
      List<KonpBik> ontziAldaketakKonplexuak =
                                    new ArrayList<KonpBik>();

      // Konpainia bikote guztiak begiratuko dira
      for(int i=0;i<direktorioak.size();i++){
            boolean dagoKonp = false; //Aldagai honen bitartez kontrolatzen da
```

```
konpainia bikoteak distantzia minimoa baino gutxiagoko ontzi aldaketa al duten
                String konpJat, konpHel;   //Jatorritik eta helmugatik igarotzen diren
konpainiak
                konpJat = direktorioak.get(i).getJatKonp();
                konpHel = direktorioak.get(i).getHelKonp();

                HashMap<String, StopsBean> geltokiJat, geltokiHel; //Uneko
bikoteko jatorri eta helmugako konpainiaren geltokiak

                //konpainien geltokien zerrenda duen aldagaitik (erabiltzekoGeltokiak) lortu unekoaren
geltoki lista eta gorde HashMap-ean
                geltokiJat = erabilitzekoGeltokiak.get(
                                konpJat.substring(0, konpJat.length()-4));
                geltokiHel = erabilitzekoGeltokiak.get(
                                konpHel.substring(0, konpHel.length()-4));
                SortedArrayList <Transbordo> ontziAldaketak =
                                        new SortedArrayList<Transbordo>();

                //lortu geltoki identifikatzaile guztiak eta joan gordetzen jatKey aldagaian
                for(String jatKey: geltokiJat.keySet()){

                        //lortu geltokiaren latitudea eta longitudea
                        double jatLat = Double.parseDouble(
                                        geltokiJat.get(jatKey).getStop_lat());
                        double jatLng = Double.parseDouble(
                                        geltokiJat.get(jatKey).getStop_lon());

                        //berdina egin helmugarekin
                        for(String helKey: geltokiHel.keySet()){

                                float[] distantzia = new float[1];

                                double helLat = Double.parseDouble(
                                        geltokiHel.get(helKey).getStop_lat());
                                double helLng = Double.parseDouble(
                                        geltokiHel.get(helKey).getStop_lon());

                                //kalkulatu uneko "jatorriaren" eta "helmugaren" geltokien arteko
                                  distantzia
                                Location.distanceBetween(jatLat,
                                                         jatLng,
                                                         helLat,
                                                         helLng,
                                                         distantzia);
                                //ibilDistMax: Erabiltzaileak ezarritako oinezko distantzia maximoa
                                if(distantzia[0] < ibilDistMax){
                                        dagoKonp = true;  //hemen sartuz gero ontzi
aldaketa sinplea egin daitekeela ziurtatzen da
                                        ontziAldaketak.insertSorted(
                                         new Transbordo(geltokiJat.get(jatKey),
                                                        geltokiHel.get(helKey),
                                                        distantzia[0]));
                                }
                        }
                }

                if(!dagoKonp){    //hemen sartzen bada bikote hau ontzi aldaketa sinplea ez duela
esan nahi du. Orduan konplexuen listan sartuko da
                        //oraingoz aldagai hau ez da erabiliko, baina hiru (edo gehiago) garraiobideko
bidaiak egin nahi izan  ezkero aldagai hau erabili beharko litzateke
```

```
                    ontziAldaketakKonplexuak.add(direktorioak.get(i));
                }
                else{
                    //ontzi aldaketak gordetzen dituen listari elementu berria gehitu
                    ontziAldaketaSinpleak.add(new TransbordoSinplea(
                                                konpJat,
                                                konpHel,
                                                ontziAldaketak));
                }
            }

            // jakinda ze konpainiak bidai osoa egin dezakete,  egin bidaia bilaketa
            bidaiEmaitzak = bilaketaKonposatuSinpleOD(
                                                    ontziAldaketaSinpleak,
                                                    konpainiaBakarra);
            return bidaiEmaitzak;
        }
```

F.8. kode-zatia: Ontzi aldaketa bilatzen duen funtzioa