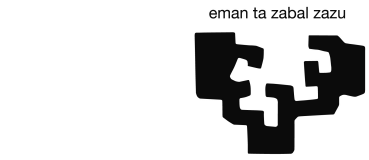


Informatika Ingeniaritzako Gradua

Konputagailuen Ingeniaritza

Gradu Amaierako Proiektua

KEApp: while-programen interpretea

Egilea

Elena Hernandez Colado

Zuzendaria

Montserrat Maritxalar Anglada

2018

Laburpena

KEApp mugikorreko aplikazioa da, "Konputazioaren Eredu Abstraktuak"(KEA) ikasgairako baliabide izatea bilatzen duena. KEA ikasgaian irakasten diren bi programen interpretea gauzatu da: while- eta makro-programena hain zuzen. Interpretea, erabiltzaileentzat eskuragarri egoteko, Android plataformarako mugikorreko aplikazioa gauzatu da, KEApp izenekoa. Aplikazioan, beste hainbat elementu sortu dira, alde batetik aplikazioa jaitsiko duten erabiltzaileek ikasketa pertsonala bultzatzeko asmoz, eta bestetik, interpretea modu erakargarri batean erabiltzeko aukera eskaintzeko. KEA ikasgaieko ikasleentzat bideratuta dago, hala nola, irakasleek parte-hartzea areagotzeko erabili dezaketen baliabide dinamikoa sortu nahi izan da. Erabiltzaileen ikasketa bideratzeko, *Gamification* metodologia erabili da, jokoen dinamikan oinarritzen dena.

Esker onak

Lehenik, eskerrak eman naiz dizkiot nire proiektuaren zuzendari, Montse Maritxalar, proiektu osoan zehar emandako animo, aholku eta orientazioagatik.

Bestetik, aplikazioaren *Alpha* bertsioa probatu duten ikasle zein irakasleei. Zuen iritzi eta aholkuak laguntza handikoak izan dira.

Eskerrak ere, nirekin batera ikasten hasi ziren ikaskideei, lagun on eta bizitzako aholkulari bihurtzeagatik.

Azkenik familiari, nire euskarri sendoena izateagatik.

Gaien aurkibidea

Eranskinak

Irudien aurkibidea

Taulen aurkibidea

1. KAPITULUA

Sarrera

Ordenagailu-programa baten bitartez ebazgarriak diren problemak identifikatzeko beharrezkoa den ezagutza barneratzea da “Konputazioaren Eredu Abstraktuak” (KEA) ikasgaiaren helburua. Ikasgaiak “Konputazioa” espezialitatearen oinarrizko ikasgai bat da eta informatika teorikoaren arloan kokatzen da. Problema konputagarri eta konputaezinen artean desberdintzen irakasten da. Problema bat konputagarria dela frogatzeko; hau da, tresna informatikoen bidez ebatz daitekeela frogatzeko, tresna horiek zeintzuk diren zehaztu beharko ditugu.

Gaur egun, *Turing Makina* baino eredu ahalmentsuagorik ez da aurkitu. Hortaz, eredu honekin problema bat ebatzi ezin bada, konputaezina dela esaten dugu. Ikasgaian, Turing Makinen baliokide diren while-programak erabiliko dira problemak konputagarrien edo konputaezinen artean kokatzeko.

Proiektu honetan, problemen konputagarritasunaren frogapen teorikoa alde batera utzita, while-programen interpretea sortuko da. Interpretearekin, KEA lantzen ari diren ikasleek programak idatzi eta exekutatu ahalko dituzte, while-programen funtzionamenduari buruzko ikasketa bultzatuz. Baina, nola lortu ikasleek interpretea modu erakargarri batean ikustea?

Gaur egun, *Smartphone* mugikorrak edonoren esku daude. Hauek, aplikazioak deskargatzeko eta internetera konektatzeko aukera eskaintzen dute. Atzean gelditu da mugikorrak telefono arruntekin zuten berdintasuna. Momentu oro gainean eramaten ditugu eta, distrakzio bat izan daitezkeen arren, gure abantailerako erabili daitezke. Ikasleek mugikorrekiko duten menpekotasuna aprobetxatuz, interpretea modu erakargarri batean erakusteko,

mugikorretarako aplikazio bat garatuko da. KEApp aplikazioan, while-programak sortzeko erremintetaz gain, arreta lortzeko beste elementu batzuk inplementatuko dira.

Ikasleen motibazioa eta parte-hartzea suspertzeko, *Gamification* metodologia erabili da. Metodologia honek jokoen elementuak (ranking-ak, puntu bidezko sistema, erronkak, etab.) erabiltzen ditu, aplikazioa ikasketarako baliabide xume baten ordez ikasketarako joko baten antzekoan bihurtuz. Gainera, ikasleen arteko lehia areagotzen du, hauen parte-hartzea bultzatuz.

1.1 Motibazioa

Proiektu hau pertsonalki proposatutako proiektua da. Ikasleen eta irakasleen onurarako baliabidea sortu nahi zen, KEA irakasgaiarekin erlazioa zuena eta datorren kurtsoetan praktikan erabili zitekeena.

Hasiera batean, while- eta makro-programen interpretea garatu nahi zen. KEA irakasgaian pisu handia duten edukia baitira. Gaur egunera arte, hauek exekutatzeko ingurunerik ez da sortu, eta, hortaz, hau izan zen lehendabizi proposatutako ideia.

Handik gutxira, mugikorrerako aplikazioa garatzea erabaki zen. Honen bitartez erakutsiko baitzen programak exekutatzeko ingurunea. Irakasgaian ikusten diren while- eta makro-programak exekutatzeko ez da ahalmen handiko gailurik behar, eta, hortaz, mugikorrean bertan exekutatzea ez du arazorik suposatzen.

Aplikazioa erabiliko zuten erabiltzaileen ezaugarriak kontuan hartuta, aplikazioa erakargarriagoa izateko *Gamification* metodologia (jokoen dinamika imitatzen duena) erabiltzea erabaki zen.

1.2 Proposatutako soluzioa

Konputazio zein software ingeniaritzako elementuak biltzen ditu proiektu honek. Elementu nagusi horiek bi dira: interpretea, *Java* programazio lengoaian garatu nahi zen, hau baita gehien landu dudan lengoaia, eta aplikazioa, *Android* sistemarako gauzatzea erabaki zena, hau baita gehien erabiltzen dena mugikorrentzako.

Proiektuaren helburu nagusiak bi dira: while-programen eta makro-programen (while-

programen hedapena direna) interpretea gauzatu, eta honen erabilera errazteko, *Android* sistemerako mugikorreko aplikazioa sortu.

Helburu nagusi horiez gain, aplikazioa KEA irakasgaiko ikasleen ikasketa pertsonala bideratzeko (*Gamification* metodologiaren bitartez) eta irakasleentzat baliabide osagarria izan nahi da.

2. KAPITULUA

Proiektuaren Helburuen Dokumentua

Proiektu honen helburu nagusia “Konputazio Eredu Abstraktuen” (KEA) ikasgairako baliabide izan daitekeen while-programen interpretea gauzatzea da, hau modu erakargarri batean erabiltzeko *Android* telefono mugikorretarako aplikazioa gauzatu da. Mugikorreko aplikazioa, ikasketarako erreminta izango dena, ikasleen ikasketa bultzatzea bilatzen du. Horretarako, *Gamification* metodoa erabili da, jokoen bitartez KEA ikasgaiko zenbait kontzeptu, teoriko zein praktiko, lantzeko.

Proiektuan bi atal nagusi desberdindu daitezke: alde batetik, while-programen interpretea, eta bestetik, mugikorrerako aplikazioa. Interpretea, while-programak ebazteko balio du, hauek KEA ikasgaian pisu handia dute eta programa bat konputagarria dela adierazteko balio dute. Hortaz, ikasgairako baliabidea izan daitekeen erreminta da. Erreminta baliotsua izan daitekeen arren, erabiltzaileek ez badute erraz eskuratzeko edo erabiltzeko aukerarik, bigarren planoan baztertzeko aukera gerta liteke. Hori saihestu nahian, eta ahalik eta erakargarrien izateko, proiektuaren bigarren azpiatal nagusia garatu da: mugikorreko aplikazioa. Bertan, hainbat erronka (galdetegi, ariketa) garatzeko aukera egongo da. *Gamification* kontzeptuaren bidez, ikasketa propioaz bultzatzeaz gain, konpetentzia (eta horrela parte-hartzea) bultzatzeko puntu-bidezko sistema erabiliko da, ahal den moduan joko baten antzeko metodologia jarraituz.

2.1 Hasierako erabakiak

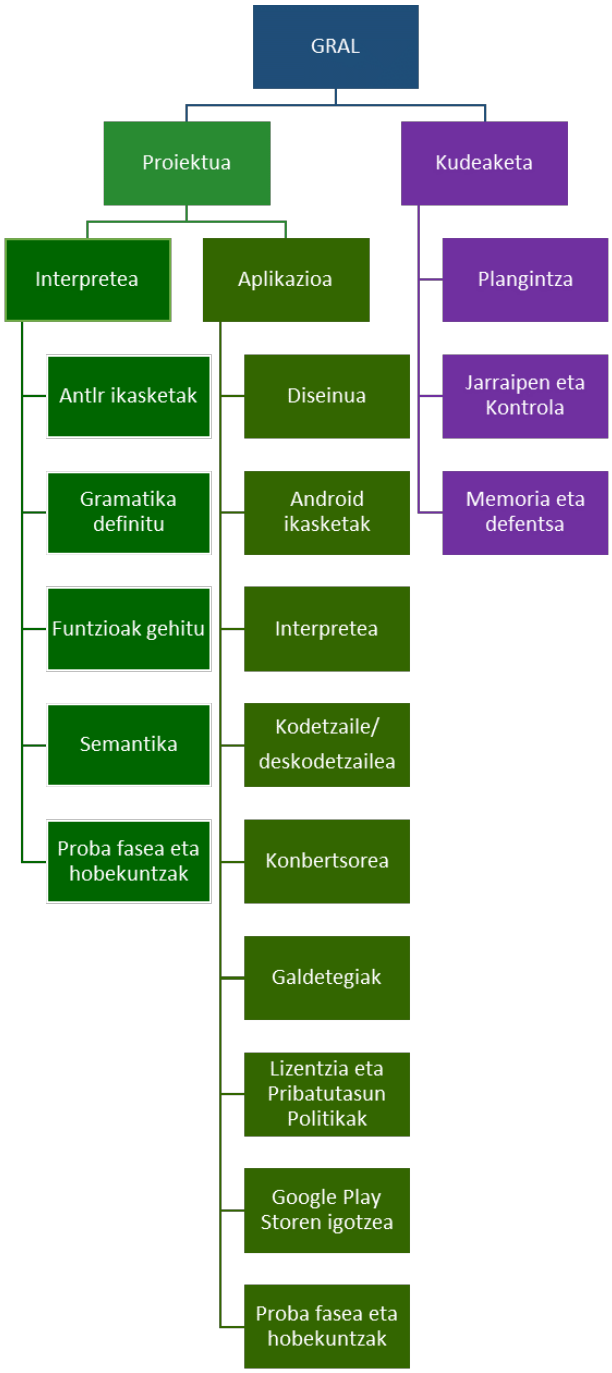
Proiektuaren helburu nagusia KEA irakasgairako erabilgarria den aplikazioa diseinatzea da. Aplikazio horretan while- eta makro-programen interpretea gauzatuko da erabiltzaileek KEA irakasgaian hain paper nagusia duten programak exekutatu ahal izateko.

Interpretea gauzatzeko, analizatzaile sintaktikoa sortzeko *Antlr* liburutegia erabili da, hau *Java* lengoaian garatu delako eta interpretea lengoaia horretan gauzatu nahi zelako.

Aplikazioa mugikorrerako egitea erabaki da, mugikorrek eskaintzen diguten ahalmenarekin nahikoa delako mota hauetako programak exekutatzeko. Gainera, ikasleentzat bideratutako aplikazioa izanik, *Gamification* metodologiaren ezaugarriak praktikan jarri dira hauen interesa eta parte-hartzea bultzatzeko asmoarekin.

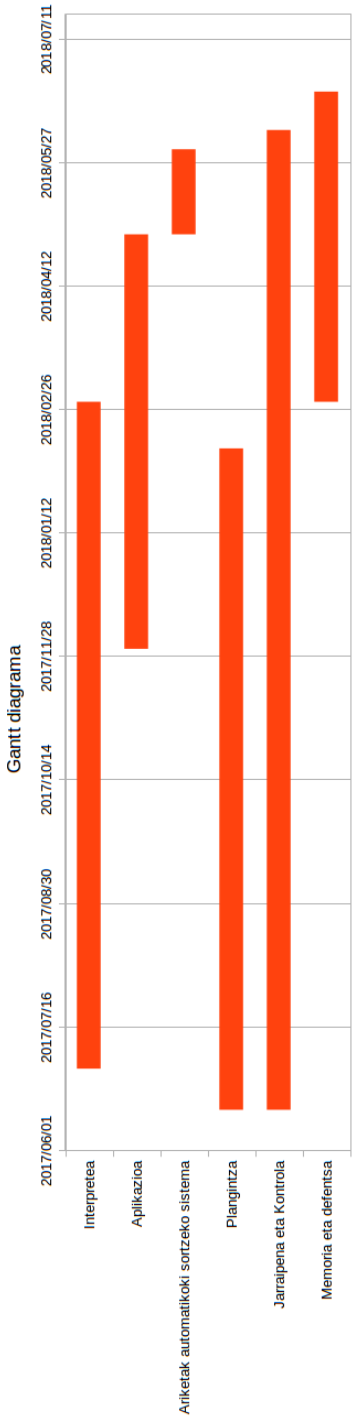
2.2 LDE

Lanaren Deskonposaketaren Egitura (LDE) (ikus 2.1 irudia) irudikatu da, proiektua osatzen duten atazak hobeto bereizteko.



2.1 Irudia: LDE diagrama

2.3 Gantt diagrama



2.2 Irudia: Gantt diagrama

2.4 Lan-paketeak

Gradu Amaierako Lana, bi ataletan banatu da, alde batetik proiektua eta bestetik, kudeaketa.

Proiektuan, bi ataza nagusi desberdindu dira: while-programen interpretea eta mugikorreko aplikazioa. Interpretea, KEA ikasgaian pisu handia duten while-programak garatzeko tresna da, eta mugikorreko aplikazioa erabiltzaileek tresna honen atzigarritasuna errazteko baliabidea.

Kudeaketan, proiektua garatzeko planifikazioa, jarraipen eta kontrol txostena eta memoria eta defentsa emangarriak daude.

Proiektuaren plangintza ahal den modu zehatzenean gauzatzeko, proiektuko atazak osatzen dituzten lan-paketeak definitu dira. Hauek lan-karga gutxiagoko atazez osatuta daude eta eginbeharreko guztien ideia sortzeko lagungarri gerta daitezke.

2.4.1 Lan-paketeen deskribapen zehatza

Atal honetan, lan-pakete bakoitzean zein ataza burutu diren eta hauetan garatutako lanaren deskribapena definitzen da.

PROIEKTUA

1. Interpretea
    - **Antlr ikasketak:** sintaxi analizatzailea Antlr liburutegia erabiliz garatzea erabaki da, hortaz ingurune horren ikasketak beharrezkoak izango dira.
    - **Gramatika:** while- eta makro- programen gramatika definitzea.
    - **Funtzioak gehitu:** Datu Mota Abstraktuak (DMAk) erabili ahal izateko funtzioak gehituko dira, hala nola bitarteko kodean erabili diren funtzio batzuk (hur(), aurre(), …).
    - **Semantika:** while- eta makro- programen semantika zehaztea, ahal eta idazketa errore gehienak erabiltzailean programa idatzi ahala erakusteko.
    - **Proba fasea eta hobekuntzak:** aplikazioan interpretea gehitu aurretik, interpretearen funtzionamendu egokia bermatzeko probak egingo dira.
2. Aplikazioa

- **Diseinua:** aplikazioan agertuko diren elementuen diseinua (GUI) eta funtzionalitate bakoitzaren portaera zehaztea.
- **Android ikasketak:** Android garapen ingurunean beharrezko ezagutza lortzea aplikazioa garatu ahal izateko.
- **Interpretea:** interpretea mugikorreko aplikazioan gehitzea.
- **Kodetzaile/deskodetzailea:** alfabetoa definituta, hainbat kate errepresentatzen duen hitzera kodeketa, eta alderantziz.
- **Konbertsorea:** alfabetoa definituta eta zenbaki bat emanda, edozein hitz edozein DMAra bihurtzeko aukera: bektore, pila edo programara.
- **Zerbitzaria:** erabiltzaileen datuak eta galdetegien informazioa gordetzeko, zerbitzari baten beharra dago. Zerbitzaria datu horiek gordetzeko egokitu behar da eta informazioa maneiatzeko datu baseak sortu.
- **Galdetegiak:** test motatako galderak eta hutsuneak betetzeko galderez osatutako galdetegi multzoa sortuko da. Galdetegiak betetzerakoan erabiltzaileak puntuak eskuratu ditzake.
- **Lizentzia eta Pribatutasun Politikak:** LOPD legediaren betekizunak eta erabiltzaileen datuen pribatutasuna bermatzeko segurtasun neurriak hartuko dira. Aplikazioaren Pribatutasun Politika dokumentua ere gauzatu beharko da Play Store-n igo aurretik.
- **Google Play Storen igotzea:** aplikazioa Google-en Play Store-ko plataformara igoko da, eta bertatik Android sistema duen mugikor guztietatik atzigarri egongo da. Hau egiteko, garatzaileak, lehenik kontu bat sortu beharko du plataforman eta ondoren formulario bat bete beharko du aplikazioaren zehaztapen guztiekin.
- **Proba fasea eta hobekuntzak:** behin aplikazioa Play Store-n igota dagoela, erabiltzaileei (ikasle, irakasleei) aplikazioa probatzeko eta baloratzeko testa egingo zaie. Behin hobekuntzarako ideiak jasota, ahal den moduan, hobekuntzak gauzatuko dira.

KUDEAKETA

1. Plangintza
2. Jarraipena eta kontrola
3. Memoria eta defentsa

2.4.2 Lan-paketeen iraupena

2.1 Taula: Lan-pakete bakoitza gauzatzeko estimatutako orduen taula

Lan-paketeak			**Iraupena (ordutan)**	**Emangarri bakoitzeko iraupena (ordutan)**
Proiektua	**Interpretea**	Antlr ikasketak	20	90
		Gramatika definitu	15	
		Funtzioak gehitu	25	
		Semantika	10	
		Proba fasea eta hobekuntzak	20	
	Aplikazioa	Diseinua	10	120
		Android ikasketak	20	
		Interpretea	10	
		Kodetzaile/deskodetzailea	10	
		Konbertsorea	10	
		Zerbitzaria	10	
		Galdetegiak	10	
		Lizentzia eta Pribatutasun Politika	10	
		Google Play Store-n igotzea	10	
		Proba fasea eta hobekuntzak	20	
Kudeaketa	**Plangintza**		25	105
	Jarraipena eta kontrola		20	
	Memoria eta defentsa		60	
GUZTIRA				315

2.2 Taula: Lan-paketeak amaitzeko datak

Lan-paketeak	Amaiera-data
Interpretea	Martxoak 5
Aplikazioa	Ekainak 1
Plangintza	Otsailak 12
Jarraipena eta kontrola	Ekainak 8
Memoria eta defentsa	Ekainak 15

Lan-pakete bakoitzaren iraupena ahal den modu zehatzenean kalkulatu ahal izateko, lan-pakete bakoitza osatzen duten atazen iraupena estimatu da (ikus 2.1 taula).

Lan-pakete bakoitzaren iraupenaren arabera bukatzeko datak zehaztu dira (ikus 2.2 taula).

2.5 Emangarriak

Bi motatako emangarri sortu dira proiektu honetan:

- **Proiektuarekin lotutakoak:**
  - Interpretea osatzeko behar izandako kode osoa.
  - Aplikazioa osatzen duten bezero eta zerbitzariaren kodea, baita exekutagarriak ere.
- **Proiektuaren kudeaketarekin lotutakoak:**
  - Proiektuaren memoria non proiektuaren deskribapenaz gain plangintza eta jarraipen eta kontroleko txostenak bilduko diren.
  - Proiektuaren defentsarako gardenkiak

2.6 Kalitatearen kudeaketa

Atal honetan proiektuaren kalitatea ebaluatzeko irizpideak zehaztu dira.

2.6.1 Kalitatearen plangintza

Proiektuak izan beharko duen kalitate maila zehazteko, bi maila hauek bereizi dira: Kalitate maila minimoa eta onargarria.

Kalitate maila minimoa

- **Interpretea:** KEA irakasgaian definitutako while- eta makro-programak eskaintzen dituzten funtzionalitateak inplementatzea.
- **Aplikazioa:**
  - Aplikazioaren funtzionamendua intuitiboa izatea. Erabiltzaileek, zailtasun handirik gabe, aplikazioa erabiltzea jakitea.
  - Gailu desberdinetarako moldatzea. Mugikorrean erabiliko den aplikazioa izanik, hau pantaila tamaina tamaina guztietarako modu egokian moldatzen dela bermatu beharko da.
- **Memoria:** : Memoria idazteko Informatika Fakultateak proposatzen duen formatuari jarraitu beharko zaio, horretarako Informatika Fakultatearen web orrian dagoen txantiloia erabiliz.
- **Gardenkiak:** Aurkezpenerako denbora mugatua denez, gardenki kopurua zehazteko kontuan hartu behar da.

Kalitate maila onargarria

- **Interpretea:** makro-programen gramatika hedatzea.
- **Aplikazioa:** aplikazioaren itxura eta edukia hobetu.
- **Memoria:** ez dago aldaketarik.
- **Gardenkiak:** ideiak modu labur eta ordenatu batean azaldu behar dira.

2.6.2 Kalitatearen kontrola

- **Interpretea:** aplikazioan integratuko da eta aplikazioaren kalitatea ebaluatzeko irizpidean kontuan hartuko da.

- **Aplikazioa:** produktuaren kalitatea ebaluatzeko, KEA irakasgaia irakatsi duten irakasleen eta kurtsatu duten ikasleen iritzia kontuan hartuko da. Hauek aplikazio mugikorra instalatu eta probatuko dute, ondoren iritzia jasotzeko.
- **Memoria:** memoriaren kalitatea proiektuaren zuzendariak ebaluatuko du entregatu aurretik.
- **Gardenkiak:** gardenkien kalitatea proiektuaren zuzendariak ebaluatuko du entregatu aurretik.

2.7 Interesatuak

- **KEA irakasgaiaren ikasleak:** ikasketa autonomorako baliabidea da. Honekin, ikasleek bere ezagutza maila neurtu dezakete, KEA ikasgaiko galdetegiak eginez eta while- eta makro-programak idatziz. Bestalde, aplikazioak eskaintzen dituen baliabideak (datu-moten arteko bihurgailuak eta hitzen kodeketa eta deskodeketarako erremintak) irakasgaiko ariketak errazteko balio dezakete.
- **KEA irakasgaiaren irakasleak:** irakasgaiko parte-hartzea areagotzeko baliabide didaktikoa da. Jolas bidezko *Gamification* metodologia erabiliz, ikasleen arreta lortzeko tresna izan daiteke. KEA irakasgaiaren edukiak irakasteko baliabide osagarria izan daiteke.

2.8 Arriskuak eta prebentzioa

Proiektuaren garapenean eragin handia izan dezaketen faktoreen deskribapena egin da, hauen prebentziorako neurriak hartu ahal izateko.

2.8.1 Arriskuak

1. **Informazio-galera:** proiektuaren kodea eta bestelako txostenen informazioa galtzeko arriskua dago.
2. **Aplikazioaren erabilera:** aplikazioan hainbat funtzionalitate eta erreminta desberdin inplementatu dira. KEA irakasgaiarekin erlazioa duten pertsonek, terminoak ulertzeko gai izango diren arren, aplikazioaren erabilera ez ulertzeko arriskua dago.

2.8.2 Prebentzioa

1. **Informazio-galera:** kodea GitHub-en igo da, ezbeharren bat gertatzekotan bertatik deskargatu ahal izateko. Bertsioen kudeaketarako Git erabili da, kode bertsioren batean erroreak badaude, aurreko bertsio batetik abiatzeko aukera eskaintzen du.

   Txostenak gordetzeko Google Drive plataforma erabili da. Plataforma hau online bidezkoa da eta hortaz arazoak eman ditzake, horregatik USB batean ere txostenak gordetzea erabaki da.

2. **Aplikazioaren erabilera:** aplikazioa modu intuitibo batean inplementatzen saiatuko da. Proba fasean, hainbat erabiltzaileekin esperimentatuko da aplikazioa Google Play plataforman ofizialki igo aurretik eta erabiltzaileek proposatutako iradokizun eta hobekuntzak aplikatuko dira, aplikazioaren nabigazioa eta erabilera errazteko helburuarekin.

2.9 Jarraipena eta kontrola

Proiektuaren kalitateari dagokionez, plangintzan zehaztutako kalitate irizpideak betetzea lortu da.

Proiektu honen irismena oso handia da, hobekuntza posible asko baititu. Aplikazioan batez ere, esperotako ordu kopurua baino gehiago sartu behar izan dira. Gehienbat aplikazioaren diseinua berregiten edota erabiltzaileekin esperimentuak egin eta gero hobekuntzak gehitzen.

Hasiera batean, galdetegien esparruan, ariketa automatikoak sortzeko sistema sinple bat diseinatzea pentsatu zen. Ariketa horietan, zenbait hutsune dituzten while- eta makroprogramak agertuko ziren, eta hauetan ausaz sortutako erantzun multzoa sortuko zen. Erantzunak modu koherente batean sortu beharko lirateke, erabiltzaileek erantzun zuzena hain erraz ez antzemateko. Baina, esperimentu fasea gauzatu zuten erabiltzaileen iritziak jasota, egin beharreko hobekuntza eta konponketa asko zeudela ikusita, horren ordez horiek gauzatzea erabaki zen. Aplikazioaren funtzionamendu orokor hobea ariketa automatikoak sortzeko sistema baino garrantzitsuagoa dela kontsideratu delako.

Proiektuan zehar ez da lan-orduen desbideraketa handirik egon (ikus 2.3 taula).

Jarraipen eta kontrolarekin ildotik, tutorearekin egindako bileren deskribapena ikusteko bilera-aktak gehitu dira dokumentu honen amaieran (ikus G. eranskina).

2.3 Taula: Proiektuaren plangintzan estimatutako orduen desbideraketa

Lan-paketeak			**Estimatutako denbora (ordutan)**	**Denbora erreala (ordutan)**	**Desbideraketa**
Proiektua	**Interpretea**	Antlr ikasketak	20	20	0
		Gramatika definitu	15	15	0
		Funtzioak gehitu	25	25	0
		Semantika	10	20	+10
		Proba fasea eta hobekuntzak	20	20	0
	Aplikazioa	Diseinua	10	15	+5
		Android ikasketak	20	15	-5
		Interpretea	10	15	+5
		Kodetzaile/deskodetzailea	10	5	-5
		Bihurgailua	10	5	-5
		Zerbitzaria	10	7	-3
		Galdetegiak	10	15	+5
		Lizentzia eta Pribatutasun Politika	10	5	-5
		Google Play Store-n igotzea	10	5	-5
		Proba fasea eta hobekuntzak	20	25	+5
Kudeaketa	**Plangintza**		25	25	0
	Jarraipena eta kontrola		20	10	-10
	Memoria eta defentsa		60	55	-5
GUZTIRA			**315**	**327**	**+12**

3. KAPITULUA

Interpretearen garapena

Interpretea programa informatiko bat da, beste programak analizatzeko eta exekutatzeko gaitasuna duena. Interpreteak ez dira konpiladoreekin nahastu behar. Interpreteak programaren itzulpena behar ahala gauzatzen dute eta normalean ez dute itzulpenaren emaitza gordetzen, konpiladoreak, aldiz, programa sistemako mihiztadura lengoaiara itzultzen dute.

Horrek suposatzen du, interpreteak konpiladoreak baino motelagoak izatea, interpreteek programa exekutatu ahala emaitza itzuli behar dute-eta. Bestalde, interpreteek eskaintzen duten abantaila nagusia moldagarritasun handiagoa da; hau da, idazketaren erroreak detektatzeko eta zuzentzeko aukera zabalagoa. Gainera, interpreteek ez dute makinaren idazketa lengoaiarekiko dependentziarik, konpiladoreek bezala, dependentzia bakarra interpretea beraren lengoaia da.

Desabantailak kontuan hartuta, interpretea konpiladorearen ordez gauzatzea erabaki da idatziko diren programak, while-programak, duten helburua konputagarritasuna frogatzea delako eta ez eraginkortasuna.

Konputazio Eredu Abstraktuak (KEA) ikasgaian ordenagailu bidez ebazgarriak diren problemak, konputagarriak, eta ebatzi ezin daitezkeenak, konputaezinak, bereizteko beharrezkoa den ezagutza irakastean datza. Konputazio espezialitateko ikasgai hau, informatika teorikoko arloan kokatzen da.

Problema bat konputagarria dela frogatzeko, Turing makina batean ebatzi daitekeela frogatu behar da, eredu hau baino ahalmen handiagorik duen eredurik ez baita ezagutzen

gaurko egunera arte. Ikasgaian, Turing makinaren idazketa errazteko asmoz, baliokideak diren while-programak erabiltzen dira.

Proiektuan, KEA ikasgaiko while-programen interpretea gauzatu da. Honekin, while-programak idazteko aukera izango dugu, baita exekutatzekoa ere.

Kontzeptuen definizioa:

- **Analizatzaile lexikoa:** (*lexer*, ingelesez) konpilazioaren lehen fasea osatzen du. Programa baten iturburu-kodea hartzen duen programa da, eta *token*-etan zatitzen duen. *Token*-ak elementu lexikoak dira, programazio lengoaian zentzua daukaten karaktere-segidak dira (else, if, :=, etab.). Behin tokenak lortuta, hauek analizatzaile sintaktikoari pasatuko zaizkio.

- **Analizatzaile sintaktikoa:** (*parser* ingelesez) konpilazioaren bigarren fasea osatzen du, analizatzaile lexikoaren ondoren. Tokenak eta gramatika baten definizioa erabiltzen ditu, token segidak programa baten egitura duten ala ez aztertzeko.

- **Semantika:** Analisi lexikoak *hitzak* ondo idatzita dauden ala ez aztertzen du. Analisi sintaktikoak, hitz horiek osatzen duten egitura aztertzen du. Analisi semantikoak, aldiz, egitura horren esanahiak zentzua duena ala ez aztertzen du. Adibidez, makro-programa baten gramatikak (analisi sintaktikoa) `eskod_1_3` onartuko luke, hau modu egokian idatzi baita, baina semantikoki gaizki idatzita dago, hitz bat elementu bakarrean deskodetzen bada, 3. elementua hartzeak ez duelako zentzurik.

- **Anbiguitatea:** Analisi sintaktikoan, bi token segida bi eratara interpretatu daitezkenean, anbiguitatea dagoela esaten da. Adibidez, `1+2+3` segida izatean, bi eratara interpretatu daiteke: `(1+2)+3` edo `1+(2+3)`. Anbiguetate hauek sahiesteko, Antlr-en bi aukera onar daitezkenean, gramatikan lehen agertzen dena hartuko da, eta erregela berdinekoak baldin badira, defektuz ezker-elkarketa egiten du, `(1+2)+3` aukera hartuz.

- **LL(*) gramatika:** testuingururik gabeko gramatika izanik, LL analizatzaile sintaktikoa beheranzko analizatzaile sintaktikoa da, behar diren token aurreikusten dituena. Analizatzaileak ezkerretik eskuinera tratatzen ditu sarrerako elementuak. Hortaz, anbiguitaterik balego, ezkerrerago dagoen elementua tratatuko litzateke lehenik.

3.1 Helburu zehatzak

While- eta makro-programen gramatika ahal den modu zehatzenean jarraitu da interpretea gauzatzeko, hauen funtzionamendua azaltzen den dokumentua[Jesús Ibáñez, 1998] [EHU, 2018] oinarri bezala hartuta. Hala ere, aldaketa batzuk egongo dira, karaktere batzuk mugikorreko teklatutik idaztea zaila baita. Dokumentu honen zehaztasunak jarraitu dira interpretea erabiliko duten bai ikasle bai irakasleentzat ulertzen errazago izateko eta erabilera deskribapen teorikotik ahal den hurbilen egoteko.

Interpretearen helburuetako bat, aplikazioaren erabiltzaileek while- eta makro-programen erabilera ulertzea da. Hau mugikorreko aplikazioan bateratuko da, bertatik modu erraz eta eroso batean erabili ahal izateko. Ildo horretatik jarraituz, makro-prograметan erabili daitezkeen hainbat funtzio gehitu dira, erabiltzaileek zuzenean erabili ditzaketenak aplikaziotik deituz gero. Funtzio hauek ikasgaian landutako ohiko funtzioak dira.

Bestalde, gaur egungo mugikorren ahalmena Turing makina baten baliokideak diren programak ebazteko nahikoa dela erakutsi nahi da. Nahiz eta KEA ikasgaian landutako programak laburrak izan, ahal den modu eraginkorrenean exekutatzen saiatuko da, mugikorraren ahalmen maximoa aprobetxatzeko.

3.2 While-programak eta kodeketa

KEA ikasgaian while-programak erabiltzen dira problema bat konputagarria dela frogatzeko. Problema bat konputagarria dela esan dezakegu Turing makina batean ebatzi badaiteke. Church-Turing tesian oinarrituta, while-programak Turing makinen baliokideak direla frogatuta geratu da, hauek Turing makinak duten ahalmen berdina izanez.

While-programen helburua problemen konputagarritasuna frogatzea denez, ahalik eta kontrol-egitura sinpleenak erabiltzen dira bertan. Izan ere, kontrol-egitura horiek while-programen barnean erabiltzeko, konputagarriak direla frogatu behar da, bestela problemaren konputagarritasuna frogatzerik ez zen posible izango.

Hortaz, while-programak programazio-lengoaia osoa, oinarrizko elementu guztiak eduki behar dituelako, eta minimoa, oinarrizkoa ez den elementurik egoterik nahi ez dugulako, izan behar dira. Elementu bat oinarrizkoa ez dela diogu baldin eta elementu hori kenduz, gainontzeko elementuekin adierazi badaiteke. Esaterako, *for* motako egitura ez da oinarrizkoa, begizta batekin ordezkatu daitekeelako, portaera bera simulatuz. Bestalde, *if*

motatako aginduak, oinarrizkoak dira, ezin delako gainontzeko kontrol-egiturekin simulatu.

Kontrol-egituren konputagarritasuna frogatzea, proiektu honetatik kanpo geratzen da.

3.2.1 Sintaxia eta semantika

Sintaxia oso sinplea da. Soilik oinarrizkoak diren kontrol-egiturez osatuta dagoelako.

While-programen gramatika dokumentu honen A.eranskinean dago.

While-programen sintaxi eta semantikaren ezaugarriak:

- **Alfabetoa**: edozein while-programa idatzi aurretik, bertan erabiliko den Σ alfabetoa definitu behar da. $\Sigma = (a, b)$ izanda, ezin izango ditugu beste horiek ez diren karaktererik erabili programa osoan, ezta aldagaiei karaktere-kate horiek ez diren baliorik ezarri.

- **Aldagaiak**: aldagaiak izendatzeko modu bakarra dago (X0, X1, X2, ...). Programaren amaieran X0-n dagoen balioa emaitzatzat itzuliko da. Aldagaien balioa alfabetoaren barneko ikurren kate bat da. Hau da while-programen datu-mota bakarra.

- **Konstanteak**: konstante bakarra egongo da, ε karakterez adierazita eta kate-hutsa adierazteko erabiliko da.

- **Esleipena**: **:=** ikurrak esleipenetarako erreserbatuta dauden karaktere-segida osatzen dute. Hiru motako esleipen daude:

  - Esleipen hutsa: aldagaiei konstante hutsa esleitzeko eragiketa.
  - $cons_s$: non s ikurrak alfabetoaren edozein ikur adierazten duen. s karakterea katearen eskuin aldean txertatzeko balio duen funtzioa da.
  - *cdr*: katearen lehen ikurra ezabatzeko balio duen funtzioa da. Karaktere hutsean aplikatuz gero, karaktere hutsa itzuliko luke.

- **Baldintzak**: baldintzak osatzeko *if* hitz erreserbatua erabili dezakegu. Baldintzarako oinarrizko predikatu bakarra erabili daiteke: $car_s\ ?$. $car_s?$: non s ikurrak alfabetoaren edozein ikur adierazten duen. Predikatua egiazkoa da baldin eta parametroa s ikurrez hasten bada.

- **Begiztak**: begiztetarako erabili daitekeen egitura bakarra *while* egitura da. Begiztetarako gelditze-irizpidea definitzeko erabili daitekeen oinarrizko predikatua *nonem?* da. *nonem?*: egiazkoa da katea hutsa ez den bitartean.

- **Funtzioak**: while-programetan ezin da funtzio berririk sortu.

- **Iruzkinak**: iruzkinak -- ikurren ondoren gehitu daitezke.

- **Puntuazio-ikurrak**: [,;()] programaren elementu desberdinak banatzeko erabiltzen diren ikurrak.

3.2.2 Erabilera-adibideak

- **Esleipenak**: Programa honek hutsa itzuliko du. Esleipen aukera guztiak biltzen dituen while-programa da.

```
1       X1 := hutsa ;
2       X2 := cons_a(X1);
3       X0 := cdr(X2);
4
```

- **Baldintzak**: X1 katea *a* karakterez hasten bada *a* itzuliko du, hutsa bestela.

```
1       if car_a?(X1) then
2           X0 := cons_a(X0);
3       end if;
4
```

- **Begiztak**: X1 aldagaiko katearen alderantzizkoa itzultzen duen while-programa da.

```
1       while nonem?(X1) loop
2           if car_a?(X1) then
3               X0 := cons_a(X0);
4           end if;
5           if car_b?(X1) then
6               X0 := cons_b(X0);
7           end if;
8           X1 := cdr(X1);
9       end loop;
10
```

3.3 Makro-programak eta kodeketa

Problema konplexuen konputagarritasuna aztertzean, while-programek eskaintzen duten sinpletasuna eragozpena izan liteke. While-programetan oinarrizkoak diren egiturak soilik erabili daitezke, horrek problemaren konputagarritasuna frogatzea errazten baitu. Baina problemen konplexutasuna igo ahala, geroz eta zailagoa gertatuko da while-programak eskaintzen dituen oinarrizko egiturekin kodea idaztea.

Hori ebazteko, makro-programak daude. Makro-programak, while-programetik eratorriak diren programak dira, baina oinarrizkoak ez diren egiturak erabili daitezke.

Makro-programen gramatika dokumentu honen B.orriko eranskinean dago.

3.3.1 Sintaxia eta semantika

Makro-programen sintaxi eta semantikaren ezaugarriak:

- **Alfabetoa**: edozein while-programa idatzi aurretik, bertan erabiliko den Σ alfabetoa definitu behar da. $\Sigma = (a, b)$ izanda, ezin izango ditugu beste horiek ez diren karaktererik erabili programa osoan, ezta aldagaiei karaktere-kate horiek ez diren baliorik ezarri.
- **Aldagaiak**: aldagaiak izendatzeko bi modu daude: while-programen antzera `X` karakterez hasi eta zenbaki batez jarraitutako aldagaiak (`X0`, `X1`, `X2`, ...) edo letra batekin hasten den eta ondoren _ (azpimarra), letra ala digituen karaktere-segidaz osatutako aldagaiak. Programaren amaieran `X0`-n dagoen balioa emaitzatzat itzuliko da. Aldagaien balioa, while-programetan bezala, alfabetoaren barneko ikurren kate bat da.
- **Konstanteak**: hutsa konstanteaz gain, alfabetoaren barneko karaktere kateak erabili daitezke balio konstante bezala, beste programazio lengoaietan egin daitekeen moduan, kakotsen artean (`"aaba"`, adibidez).
- **Baldintzak**: baldintzak osatzeko, *if* arruntez gain, *elsif* eta *else* klausulak erabili daitezke. Gainera, switch baten antzekoa den *case-when* sententziak erabili daitezke. Gainera, baldintzetarako edozein adierazpen erabili daitezke.
- **Begiztak**: begiztetarako *while* eta *for* egiturak daude. Begizten gelditze-irizpidea definitzeko edozein adierazpen erabili daiteke.

- **Esleipena**: **:=** ikurrak esleipenetarako erreserbatuta dauden karaktere segida osatzen dute.

- **Funtzioak**: makro-programetan, while-programetan ez bezala, funtzio berriak sortu daitezke. Gainera, aurretik definitutako hainbat funtzio dagoeneko inplementatu dira, makro-programak errazago idatzi ahal izateko.

- **Iruzkinak**: iruzkinak `--` ikurren ondoren lerro amierara arte gehitu daitezke.

- **Puntuazio-ikurrak**: **[,;()]** programaren elementu desberdinak banantzeko erabiltzen diren ikurrak.

- **Alderaketak: =, /=, <, >, <=, >=, and, or** gauzatu daitezkeen alderaketa guztiak dira. Desberdinketa gauzatzeko ere `not` erabili daiteke **/=** karaktereen ordez.

- **Eragiketa aritmetikoak: +,-,***, batuketa, kenketa eta biderketa eragiketak inplementatu dira. Honetarako zenbaki arrunten errepresentazio funtzioa erabili da, hitzetatik zenbakira pasa, eragiketa gauzatu, eta berriro hitzera bihurtzeko (honela *hur* eta *aurre* funtzioak baino eraginkortasun handiagoa lortu da). Zenbaki negatiboak ez daudenez definituta, eragiketa batek (kenketak, zehazki) zenbaki negatibo bat sortuko balu, *0*-ren baliokidea den hitz hutsa bueltatuko luke.

- **Erreserbatutako hitzak**: `hutsa`, `phutsa`, `while`, `for`, `loop`, `if`, `then`, `elsif`, `else`, `case`, `is`, `when`, `others`, `def begin` eta `not`, programari egitura emateko erreserbatutako hitzak dira.

- **Programa nagusia**: Makro-programetan funtzio berriak sortu daitezkenez, programa nagusia identifikatzeko *main* izeneko funtzioa sortu behar da. Programa exekutatzerakoan, hori izango da deituko den lehen funtzioa. Honelako egitura izan beharko du:

  ```
  def main begin ... end def;
  ```

3.3.2 Erabilera-adibideak

- **Esleipenak**: While-programetan azaldutako esleipen posibleez gain, orain edozein adierazpenaren balioen esleipena gauzatu daiteke. Gainera, aldagaiek izen esanguratsuak izan ditzakete.

```
def main begin
    aldagaiHandiena := handiena("a", "b");
    besteAldagaiBat := "abba" + aldagaiHandiena;
end def;
```

- **Baldintzak**: Baldintzetarako *if* sententzia hedatu da, orain *elsif* eta *else* motatako klausulak erabili ahal izateko, ondorengo adibidean erakusten den bezala.

  $B_1, B_2, \ldots B_n$ makrobaldintza eta $Q_1, Q_2, \ldots Q_n$ makroprogramak izanda:

```
def main begin
    if B1 then
        Q1
    elsif B2 then
        Q2
    elsif Bn then
        Qn
    else Qn+1
    end if;
end def;
```

  switch motatako sententzia ere inplementatu da. `V` adierazpen bat izanik, y_1, y_2, y_n balioen berdina den kasuetan dagokion programa exekutatuko da.

```
def main begin
    case V is
        when y1 => Q1
        when y2 => Q2
        when yn => Qn
        when others => Qn + 1
    end case;
end def;
```

- **Begiztak**: *while* sententziak orain edozein adierazpen erabili dezake baldintza moduan. Gainera, `for` makroa inplementatu da. `V` identifikadore bat izanik, Q_1 exekutatuko du, aldi bakoitzean V-k $[A, B)$ tarteko balioak hartuz, non A eta B edozein adierazpen izan daitezken.

  Esaterako: `for i in 1..10 loop`, `for i in "a".."ab" loop` edo `for X4 in X1..X2 loop`.

```
def main begin
    for V in A..B loop
        Q1
```

```
4         end loop;
5     end def;
```

- **Funtzioak**: *main* izeneko programa nagusitik, guk sortutako edozein funtziori dei egin diezaiokegu. Funtzioek errekurtsioa erabili dezakete, eta ez dira zertan erabili baino lehen definitu behar.

```
1     def main begin
2         --Programa nagusia
3         X0:=a_gehitu(X1);
4     end def;
5
6     def a_gehitu begin
7         X0:= cons_a(X1);
8     end def;
```

Makro-programen erabilera-adibide konplexuagoak, dokumentu honen C. eranskinean daude.

Analizatzaile-sintaktikoak, adierazpen eta sententziak nola desberdintzen dituen hobeto ikusteko, erabilera-adibide horien Sintaxi Zuhaitz Abstraktuak (SZA) erakusten dira D. eranskinean.

3.3.3 Datu-mota astraktuak

Makro-programetan, while-programetan bezala, dagoen datu-mota bakarra alfabetoaren barneko ikurrez osatutako kateak dira. Kasu honetan, Datu Mota Abstraktuen (DMA) erabilera ahalbidetzeko, zenbait funtzio inplementatu dira. Funtzio horiekin zenbaki arrunt, pila eta bektoreekin eragiketak gauzatu daiteke. Datu-mota hauen portaera definitzeko hainbat funtzio sortu behar izan dira. Gainera, lehen esan bezala, datu-mota bakarra alfabeto barneko ikurrez osatutako katea izanik, datu-mota bakoitzaren interpretazio-funtzioa beharko dugu.

3.3.3.1 Boolearrak

Mota boolearra, **B** = {true, false}, inplementatzeko, ondorengo interpretazio-funtzioa erabili dugu:

$$\Im_B(w) = \begin{cases} false & \text{baldin } w = \varepsilon \\ true & \text{bestela} \end{cases}$$

Mota boolearra, nagusiki, baldintzetan erabiltzen da, nahiz eta edozein adierazpenen moduan ere erabili daitekeen. Honela, ondorengo kodearen Q_1 soilik `X1`-en balioa hutsa ez bada exekutatuko da.

```
1  if X1 then
2      Q1
3  end if;
```

Datu mota honekin bereziki zentzua duten eragiketak ondorengoak dira:

$$=, not, and, or$$

3.3.3.2 Zenbaki arruntak

Zenbaki arrunten datu-mota osatzeko, $\mathbb{N} = (1,2,3,\ldots)$, ondorengo interpretazio-funtzioa erabili da:

Baldin $\Sigma = a_1, a_2, \ldots, a_n$, orduan $\Im_N : \Sigma^* \to N$ funtzioa honela definituko da:

$$\Im_N(x \cdot a_i) = n * \Im_N(x) + i$$

Datu mota honen inplementazioari esker, ondorengo eragiketak ahalbidetu dira:

$$<, >, <=, >=, ==, / =, hur(), aurre(), +, -, *$$

non *hur()* hurrengo eta *aurre()* aurreko zenbakiak itzultzen dituzten funtzioak diren (zenbaki aurruntetan behe-muga bat dagoenez, hitz hutsaren (0-ren baliokidea) aurreko hitza eskatzerakoan, hitz hutsa bera bueltatuko da).

3.3.3.3 Pilak

Pilak, balio anitz gordetzeko datu-egiturak dira, non lehen sartzen den osagaia ateratzen azkena izango den. Datu-motaren balioen multzoari **P** deituko diogu. Pila motako datuak **<** eta **]** ikur artean adieraziko ditugu. "<" ikurraren ondoren dagoen lehen osagaia pilaren tontorra izango da. Adibidez: `<a, bb]` pilaren tontorra `a` izango da. Pilen gainean

definitutako funtzioak ondorengoak dira:

$$pilaratu : \Sigma^* \times P \to P$$

$$pila\_hutsa\_da? : P \to B$$

$$tontorra : P \to \Sigma^*$$

$$despilaratu : P \to P$$

Interpretazio-funtzioa definitzeko, pila posible bakoitza hitz desberdin batez adierazi behar dugu, osagaien ordena eta alfabetoa kontuan hartuz kodetuz.

$$\Im_P(w) = \begin{cases} <] & \text{baldin } w = \varepsilon \\ < aurre(deskod_{2,1}(w))] & \text{baldin } w \neq \varepsilon \wedge deskod_{2,2}(x) = \varepsilon \\ pilaratu(deskod_{2,1}(w); \Im_P(deskod_{2,2}(w)) & \text{bestela} \end{cases}$$

3.3.3.4 Bektoreak

Bektoreak, pilen antzera, osagaiak ordenan gordetzen dituzten datu-egiturak dira. Bektoreen kasuan, piletan ez bezala, osagaiak indizearen bidez zuzenean atzitu daitezke. Indizeak zerotik zenbatzen hasiko dira. Datu-mota honen balioen multzoari **V** deituko diogu. Bektoreek, gutxienez osagai bat izan behar dute, eta parentesien arteko osagai-zerrendaren bidez adieraziko dira. Adibidez: *(a, b, aba)* bektorearen zerogarren osagaia *a* izango da eta *aba* bigarrena.

Bektoreen funtzionamendua simulatzeko oinarrizko funtzioak:

$$azken\_indizea : V \to N$$

$$atzitu : V \times N \to \Sigma^*$$

$$aldatu : V \times N \times \Sigma^* \to V$$

Interpretazio-funtzioan, bektorearen tamaina erabiliko dugu kodeketan.

$$\Im_V(w) = (x_0, x_1, x_2, \ldots, x_k), \text{ baldin eta } \begin{cases} k = \Im_N(deskod_{2,1}(w)), \text{ eta} \\ \forall i(0 \leq i \leq k \rightarrow x_i = deskod_{k+1,i+1}(deskod_{2,2}(w))) \end{cases}$$

3.4 Tresnak eta teknologiak

3.4.1 Antlr4

Liburutegi hau analizatzaile sintaktikoa sortzeko erabili da, zehazki *Antlr4.7* bertsioa erabili da. *Antlr* liburutegia [Parr, 2013] Java Makina Birtualean inplementatuta dago, eta proiektuan sortuko den interpretea ere, *Java* programazio lengoaian gauzatu da.

Antrl meta-programa bat da, hau programazio lengoaiak idazteko programa baita. Lengoaia baten gramatika abiapuntu bezala hartuta, testu bat lengoaia horren parte den ala ez determinatzeko LL(*) motako analitzatzaile lexikoa (ikusi definizioa 3. atalean). Analizatzaile horrekin, sarrerako testuak sortutako sintaxi zuhaitza bisitatu daiteke, kasu honetan interprete bat sortzeko.

Gainera, *Antlr* liburutegiak analisirako tarteko egiturak sortzeko erraztasunak ematen ditu, esaterako, sintaxi zuhaitz abstraktuak sortzeko.

3.4.2 IntelliJ IDEA

Interpretea gauzatzeko erabili den editorea *IntelliJ IDEA Ultimate Edition* [JetBrains, 2009a], *JetBrains* enpresak sortutako programazio-ingurunea da. *Java*-ren 7. bertsioa erabili da, 8. bertsioa existitzen den arren, aurreko bertsioak bateragarritasun handiagoa eskaintzen duelako beste zerbitzuekin, bereziki, *Android*-ekin.

3.5 Inplementazioa

Interpretea gauzatzeko, *Antl* programa erabili da while- eta makro-programak aztertzeko analizatzaile sintaktikoa sortzeko. Analizatzaileak, LL(*) gramatika erabiltzen du (elementuak ezkerretik eskuinera tratatzen dituen gramatika), sintaxi zuhaitz abstraktuak tratatzeko. Hauek, programen kodearen [Hernandez, 2018a] egitura adierazteko metodoa di-

ra. Programen adierazpen eta sententziak analizatzaileak nola identifikatzen dituen ikusteko erabiltzen da. Sintaxi zuhaitz abstraktuak tratatzeko, *Java* programazio lengoaiean sortutako klaseak erabili dira.

While- eta makro-programetan anbiguitate arazoak erraz konpondu dira. Anbiguitateak esanahi bereko adierazpenak daudela esan nahi du. Esaterako, ”a” + ”b” + ”c” motako aginduan, anbiguitatea dago, aldi berean ”a” + ”b” edo ”b” + ”c” eragiketak gauzatu ahal direlako. LL gramatikako analizatzailea izanda, ezkerretik hasiko da, eta hortaz, lehenengo ”a” + ”b” eragiketa gauzatuko litzateke.

Bestalde, eragiketen lehentasuna zehaztu behar izan da,adibidez, ”a” + ”b” ∗ ”c” motako aginduan, lehenik biderketa gauzatzeko. Izan ere, *Antlr*-ek anbiguotasuna kentzeko bi aukera posible direnean, lehenengoa hartzen du beti, hortaz, eragiketen lehentasuna gramatikan agertzen diren ordenaren arabera definitzen da

3.5.1 Kodearen egitura

Interpretearen iturburu kodean bi atal nagusi desberdindu daitezke: alde batetik *Antlr*-ekin osatutako gramatika dago (ikus A eta B eranskinak), eta bestetik gramatika erabiltzen duten sintaxi zuhaitz abstraktuaren klaseak (ikus 3.5.3, 3.5.4 eta 3.5.5 atalak).

Antlr (ANother Tool for Language Recognition) analizatzaile lexiko eta sintaktikoak sortzen dituen tresna da. Gramatika batetik abiatuz, horren elementuak bisitatzeko zuhaitzak sortu eta zuhaitz hori hainbat programazio lengoaia desberdinetik bisitatzea ahalbidetzen duen analizatzaile sintaktikoa sortu ditzake. *antlr*-en karpetan, bi gramatika desberdin daude, bat while-programen gramatika, eta bestea makro-programena. Nahiz eta makro-programak while-programen hedapena izan, kodea sinplifikatzeko bakoitzarentzat gramatika bana egitea erabaki da. Honela, while-programen gramatikak errore sintaktiko guztiak arrapa ditzake, eta soilik errore semantikoak kudeatu behar dira. Azken hauek Java-tik bilatzen dira, eta batik-bat alfabetotik kanpoko ikurrik erabiltzen ez dela begiratzea da.

Beste karpetak *java* izena du, eta lengoaia horretan gauzatutako klaseak daude. Klase horietako bakoitza Sintaxi Zuhaitz Abstraktuaren (SZA) (ikus D. eranskina) elementu bakoitza errepresentatzen du, eta dagokion elementuaren erroreen-tratamendua eta exekuzioaz arduratzen dira.

3.5.2 While- eta makro-programen tratamendua

Bi programa mota daudenez, pentsa dezakegu bakoitza alde batetik tratatzeko kodea idatzi behar dela, baina while-programak makro-programen azpimultzoa direnez, kode bera bi motatako programentzat berrerabili da.

Gramatika desberdinak sortuz, sintaxi zuhaitz abstraktua sortzeko inplementazioa berdin mantentzen da. Hau inplementatzeko, adierazpen eta sententzia bakoitzeko klase bat sortu behar izan da.

3.5.3 Sintaxi Zuhaitz Abstraktuak tratatzeko klaseak

Sekzio honetan, sintaxi zuhaitz abstraktuaren nodoen klaseak azalduko dira. While-programek soilik hauen azpimultzo bat erabiliko dute.

Klase bakoitzak, klase abstraktua ez bada, bi funtzio inplementatu behar ditu gutxienez: `verify`, errore semantikoak bilatzeko, eta `verifyAlf`, alfabetoaren kanpoko karaktereak erabiltzen ez direla bermatzeko. Bi funtzio hauek banatzearen arrazoi nagusia ondorengoa da: etorkizunean, while- edo makro-programa bat idatzi ahala, alfabetoa definitu baino lehen, errore batzuk aurkitu daitezke. Honela, alfabetoa programa idatzi ondoren finkatu daiteke, errore gehienak konpondu direnean.

Bi nodo mota nagusi daude: `Statement`, sententziak errepresentatzen dituztenak, eta `Adierazpena` izenekoak. Mota askotariko sententzia eta adierazpenak daude, eta bi klase abstraktu hauen azpi-klase zehatzak izango dira.

Sententzia mota guztiek lehen aipatutako bi funtzioez gain, `execute` funtzioa inplementatu behar dute. Funtzio honek sententzia exekutatzeko balio du, eta *SinboloTaula* hartzen du parametrotzat, aldagaien balioak irakurri eta aldatu ahal izateko. Gainera, ziklatzen duen programa bat sortuz gero, gelditu ahal izateko baliabideak ditu. Hau, *Java* lengoaiak eskaintzen duen *Thread*-en bitartez inplementatu da.

Adierazpenei dagokienez, erroreen-tratamendurako bi funtzioez gain, `getValue` funtzioa inplementatu behar dute. Honek ere *SinboloTaula* hartzen du parametrotzat. Funtzio honekin, adierazpenaren balioa lortzen da (karaktere kate motakoa).

SZA-ren nodo ez diren beste hainbat klase daude:

- `Alfabetoa`: alfabetoa osatzen duten karaktere desberdinen indizea lortzeko klasea, hau, *Utils* klasean erabilgarria izango da, kodetzeko funtzioan esaterako.

3.1 Taula: Klase eta azpi-klaseen egitura erakusteko taula

Klase nagusia	**Azpi-klaseak**	**Adibidea**
Programa		def main begin ... end def;
Statement	**CaseStmt**	case X1 is when "a"=> ...; end case;
	EsleipenStmt	X1:= aldagaia; X1:= "abab"; X1:= funtzioa(X0); ...
	ForStmt	for i in "a".."ab" loop ... end loop;
	ForZenbakiStmt	for i in 1..10 loop ... end loop;

	IfStmt	if X1="a" then
		...
		elsif X1="b" then
		...
		else
		...
		end if;

Funtzioa		def reverse begin ... end def;
Adierazpena	**AldagaiExpr**	aldagaiIzena
	AlderaketakExpr	X1 < X2
	CarExpr	car_a(X0)
	ConsExpr	cons_a(X0)
	DeskodExpr	deskod_2_2(X1)
	FuntzioExpr	reverse(X1)
	HitzaExpr	"ab"
	HutsaExpr	hutsa
	NotExpr	not car_a(X1)

- `Errorea`: Errore bat errepresentatzen du. Errorearen posizioa eta dagokion mezua gordetzen ditu.
- `Liburutegia`: aplikaziotik erabiltzaileek erabili ditzaketen funtzioen multzoa, bai eta funtzio horiek automatikoki SinboloTaula-n gehitzen dituen funtzio bat ere.
- `Posizioa`: errorearen posizioa. Zein lerroan eta zutabean errorea hasi eta bukatu den gordetzen du.
- `SinboloTaula`: sinbolo taula programen interpretazioan erabiltzen den egitura da. Bertan, definitu diren aldagaiak, funtzioak eta alfabetoa gordetzen dira. Sinbolo taulaz baliatuz, eta analizatzaile sintaktikoak prozesatutako elementuekin, exekutatzen dira programak. Programa batean sortutako elementu guztiak gordetzen dira bertan, programan aurrera egin ahala bertako aldagaiak edo funtzioak zein diren ezagutzeko.
- `MyMakroVisitor`: SZA sortzeko klasea. *Antlr*-ek sortutako klase baten azpi-klasea da. Kodearen elementuak bereizi ahala, elementu horiek bisitatzean SZA-ren klaseak sortuko ditu.
- `Utils`: liburutegiko zenbait funtzio sortzeko erabili den klasea. Erabiltzaileek zuzenean erabili ezin ditzaketen funtzioak daude. Hala nola, hitzak beste datu-mota batetara bihurtzekoak. Klase hau Android aplikazioak erabiltzen du, datu moten arteko bihurketak eta kodeketa/deskodeketa inplmenetatzeko.

3.5.4 Utils

Klase honetan, funtzioak laguntzaile batzuk biltzen dira. Erabiltzaileek funtzio hauek ezin izango dituzte zuzenean mugikorreko aplikaziotik erabili, barne-kalkuluak egiteko erabiliko baitira. Aldiz, funtzio hauek modu erabilgarriago batean eskainiko dira, erabilera zuzenerako.

Aplikazioan datu-moten arteko bihurgailu atal bat egongo da, non erabiltzaileek hasierako datu-mota batetik beste batera bihurketak gauzatu ahal izango dituen. Erabili diren datu-motak, karaktere kateak, zenbaki arruntak, pilak eta bektoreak izan dira. Bihurketak gauzatzeko, erabiltzaileak bihurtu nahi duen balioa karaktere katera bihurtuko da. Horrela datu-mota guztietatik guztietarako bihurketa funtzioak sortzea saihesten da; hau da, soilik beste datu motetatik karaktere katera, eta alderantziz bihurtzeko funtzioak inplementatu behar izan dira.

Horrez gain, kodetzeko eta deskodetzeko funtzioak inplementatu behar izan dira, bai datumoten arteko bihurketa-funtzioetan behar direlako eta aplikazioan kodetzeko eta deskodetzeko atala egongo delako.

- `kod_2`: bi funtzio desberdin daude, batak *BigInteger* datu motarentzat eta bestea *String*. Kodetzen den hitzaren motaren arabera erabili daitezke, hau zenbakizkoa ala karaktere segida den arabera. `kod_2` 2 hitz kodetzeko erabili den funtzioa da.
- `kod`: `kod_2` *n* aldiz deitzeko funtzioa da.
- `dekod_2`: bi funtzio desberdin daude, batak *BigInteger* datu motarentzat eta bestea *String*. Deskodetzen den hitzaren motaren arabera erabili daitezke, hau zenbakizkoa ala karaktere segida den arabera. `dekod_2` 2 hitz deskodetzeko erabili den funtzioa da.
- `dekod`: `dekod_2` *n* aldiz deitzeko funtzioa da.
- `hitzakZenbakira`: karaktere-segida motako hitzak zenbakizko hitzera bihurtzeko funtzioa.
- `zenbakiakHitzera`: zenbakizko hitzak karaktere-segida motako hitzera bihurtzeko funtzioa.
- `hitzetikPilera`: karaktere-segida motako hitzak pila datu-motera bihurtzeko funtzioa.
- `pilatikHitzera`: pila datu-motetik karaktere-segida motako hitzera bihurtzeko funtzioa.
- `hitzetikBektorera`: karaktere-segida motako hitzak bektore datu-motera bihurtzeko funtzioa.
- `bektoretikHitzera`: bektore datu-motetik karaktere-segida motako hitzera bihurtzeko funtzioa.
- `pilaratu`: pila datu-motean elementuak gehitzeko funtzioa.
- `despilaratu`: pilatik elementuak ateratzeko funtzioa.
- `tontorra`: pilan sartu den azken elementua itzultzeko funtzioa.

Funtzio hauetatik aparte, ondorengo atalean agertzen diren funtzioen inplementazioa *Utils* klasean dago, eta *Liburutegian* dagoena interfazea besterik ez da.

3.5.5 Liburutegia

Makro-programetan erabiltzaileak sortu ditzakeen funtzioez gain, zenbait funtzio inplementatu dira erabiltzaileek makro-programetan erabili ahal izateko. Interpreteak funtzio dei bat ikustean, ondoren definitutako funtzio izenak aurkituz gero, dagokion eragiketak gauzatzen dira, erabiltzaileak funtzioak sortu behar gabe. Funtzio hauek liburutegian gorde dira: $\Sigma = a, b$ izanda:

- `kateatu`: bi hitz emanda, hitz hauek bakar batean lotzen ditu.

  $$kateatu(x \cdot y) = xy$$

  Adb: kateatu("a", "b") = "ab"

- `hur`: hitz bat emanda, kodeketan hurrengoa den hitza itzuliko du.

  $$hur : \Sigma^* \rightarrow \Sigma^*$$

  Adb: hur("a") = "b"

- `aurre`: hitz bat emanda, kodeketan aurrekoa den hitza itzuliko du.

  $$aurre : \Sigma^* \rightarrow \Sigma^*$$

  Adb: aurre("b") = "a"

- `zero_da`: balio hutsa den ala ez adierazteko funtzio boolearra.

  $$zero\_da : \Sigma^* \rightarrow B$$

- `phutsa`: pila hutsa sortzen du.

  $$phutsa :\rightarrow P$$

- `pilaratu`: pilan balio berria sartzeko balio du.

  $$pilaratu : \Sigma^* \times P \rightarrow P$$

  Adb: pila = <"a", "b"] izanik, `pilaratu("c", pila) = <"c","a", "b"]`

- `despilaratu`: pilatik sartutako azken balioa kentzeko balio du.

$$despilaratu : \Sigma^* \times P \rightarrow P$$

- `tontorra`: pilan sartutako azken balioa itzultzen du.

$$tontorra : P \rightarrow \Sigma^*$$

- `p_hutsa_da`: pila hutsa den ala ez adierazten duen funtzio boolearra da. Kate hutsa itzuliko du pila hutsa ez badago.

$$pila\_hutsa\_da? : P \rightarrow B$$

- `lehen_bektore`: bektore hutsa sortzeko funtzioa.

$$lehen\_bektore :\rightarrow V$$

- `azken_indize`: bektorea emanda, horren azken indizea itzultzen du.

$$azken\_indizea : V \rightarrow N$$

- `atzitu`: bektorea eta atzitu nahi den elementuaren indizea pasatuz, elementua itzultzeko funtzioa.

$$atzitu : V \times N \rightarrow \Sigma^*$$

- `aldatu`: bektorea, aldatu nahi den elementuaren indizea eta elementu horren balio berria pasatuz, elementua eguneratzeko funtzioa.

$$aldatu : V \times N \times \Sigma^* \rightarrow V$$

- **Kodeketa eta deskodeketa funtzioak**:
  - `kod`: hitz-segidak hitz bakarrean konprimitzen dituen funtzioa. Objektu-moten arteko bihurketa gauzatzeko baliagarria izango den funtzioa. Deskodeketa funtzioarekin, behar izanez gero, jatorrizko hitzak berreskuratu daitezke.

$$kod^2 : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$$

	$\Sigma = \{a\}$	$\Sigma = \{a,b\}$	$\Sigma = \{a,b,c\}$
w_0	ε	ε	ε
w_1	a	a	a
w_2	aa	b	b
w_3	aaa	aa	c
w_4	aaaa	ab	aa
w_5	aaaaa	ba	ab
w_6	aaaaaa	bb	ac
w_7	aaaaaaa	aaa	ba
w_8	aaaaaaaa	aab	bb
w_9	aaaaaaaaa	aba	bc
w_{10}	aaaaaaaaaa	abb	ca

3.1 Irudia: Alfabetoaren ordena azaltzeko taula

	w_0	w_1	w_2	w_3	w_4	...	W_j
w_0	w_0	w_2	w_5	w_9	w_{14}	...	
w_1	w_1	w_4	w_8	w_{13}	w_{19}	...	
w_2	w_3	w_7	w_{12}	w_{18}	w_{25}	...	
w_3	w_6	w_{11}	w_{17}	w_{24}	w_{32}	...	
w_4	w_{10}	w_{16}	w_{23}	w_{31}	w_{40}	...	
w_5	w_{15}	w_{22}	w_{30}	w_{39}	w_{49}	...	
w_6	w_{21}	w_{29}	w_{38}	w_{48}	w_{59}	...	
...	...	...	...	...	...	...	
w_i							W_k

3.2 Irudia: kod^2ren funtzionamendua azaltzeko taula

Kodeketa funtzioaren funtzionamendurako, alfabetoaren gaineko hitzen ordena (ikus 3.1 irudia) definitu behar da. Ordinal txikiena 0 da eta beraz, $w_0 = \varepsilon$ eta $w_{i+1} = hur(w_i)$ bezala definitzen da.

Bi dimentsioko taula honetatik (ikus 3.2 irudia) abiatuz, non zutabe eta lerroak alfabetoaren hitz ordenatuekin etiketaturik dagoen, lerro eta zutabe bikote bakoitzari kod^2 funtzioaren definizio bat da.

(i+j). diagonalean dagoen (w_i, w_j) bikotearen w_k kodearen azpindizea kalkulatzeko:

$$k = \frac{(i+j)*(i+j+1)}{2} + j$$

Adb: $kod_2(w_{10}, w_{20}) = \frac{(10+20)*(10+20+1)}{2} + 20 = w_{485}$

Deskodeketa funtzioarekin i eta j-ren balioak berreskuratu ditzazkegu:

$$kod^2(deskod_{2,1}(w), deskod_{2,2}(w)) = w$$

Kodeketa funtzioak, hitz-multzo bat emanda, hitz horien kodea den hitza itzuliko du. kod^n, non n kodetu nahi diren hitzen kopurua diren.

$$kod^k(z_1, z_2, \cdots, z_k) = kod^2(z_1, kod^2(z_2, \cdots kod^2(z_{k-1}, z_k)\cdots))$$

– `deskod`: hitz bakarra hitz-segida batean deskonprimitzen duen funtzioa. Funtzio hau ere, datu-moten arteko bihurketak gauzatzeko baliagarria izango den funtzioa da.

$$deskod_{k,i} : \Sigma^* \to \Sigma^*$$

$$deskod_{k,i}(w) = \begin{cases} w & \text{baldin } i = 1 \wedge k = 1 \\ deskod_{2,1}(w) & \text{baldin } i = 1 \wedge k > 1 \\ deskod_{k-1,i-1}(deskod_{2,2}(w)) & \text{baldin } i > 1 \wedge k > 1 \end{cases}$$

Liburutegiaren eta DMA erabilerari buruzko adibideak C.eranskinean daude.

3.5.6 Erroreen kudeaketa

While-programak makro-programak baino aukera gutxiago eskaintzen ditu programatzerako orduan. Adibidez, while-programetan, `while` motako sententzia baten barruan soilik `nonem?` motako baldintza egon daiteke. Makro-programetan, ordea, `while` motako agindu baten barnean edozein adierazpen idatzi daiteke.

Murriztapen horiek hobeto tratatzeko, bi gramatika desberdin sortu behar izan dira. Baina programak exekutatzeko SZA-ren klaseak eta erroreak (ikus 3.2 taula) bientzako berdin mantendu dira. Desberdintasun nagusia, makro-programetan funtzioak sortu daitezkeela da, eta hortaz, programa nagusia desberdintzeko, `main` izeneko funtzioa existitu behar duela da.

3.2 Taula: Programetan detektatzen diren erroreen deskribapena

Errorea	**Mezua**
Alfabeto kanpoko karaktereak erabiltzea	Alfabeto barneko karaktereak erabili ("alfabeto kanpoko letra")
Programa nagusirik ez idatea makro-programetan	main izeneko funtzioa existitu behar da
Existitzen ez den funtzio bati deitzerakoan	"funtzioaren izena"izeneko funtziorik ez da existitzen
Gramatika ez errespetatzea	"n". lerroan errore sintaktikoa dago
Deskod funtzioaren erabilera okerra	Deskod funtzioa gaizki idatzi da: "indize"<= "tamaina"izan behar du
Alfabetoa idaztean, karaktere errepikatuak sartzea	Alfabetoan ezin dira karaktere errepikaturik sartu

4. KAPITULUA

Aplikazioaren garapena

Interpretea erabilgarriagoa izateko KEApp izeneko mugikorrerako aplikazioan integratu da. Ikasleen zein bestelako erabiltzaileen eskura egongo da Google Play plataforman, eta hortaz, *Android* sistema eragileko mugikorretan egongo da eskura.

Garapen teknologikoko garaian bizi gara eta ikasleek dituzten hezkuntza beharretara moldatzeko tresna izan daiteke sortutako aplikazioa. KEApp, interpretea bisualki erakusteko erreminta ez ezik KEA ikasgairako baliabidea izan daiteke. KEA ikasgaiko eduki teorikoak eta praktikoak lantzeko galdetegiak eta ariketak egongo dira bertan. Erabiltzaileek beraien gaitasunak neurtzeko eta trebatzeko joko moduko (*Gamification* metodologia erabiliz) aplikazioa da.

Aplikazioarekin ikasleen parte-hartzea eta motibazioa areagotzea bilatzen da, mugikorrek eskaintzen duten eramangarritasunez baliatuz, eskola barruan eta, batez ere, kanpo erabili ahal izateko.

Aplikazioaren nagusitasuna KEA irakasgaiak izango du, bertako edukiak lantzeko tresna baita. Eduki teorikoak galdetegien bitartez landuko dira.

Zehazki, KEA irakasgaiko gaia bakoitzeko edukiak lantzeko galdetegiak egingo dira. Hauek dira landuko diren arloak: konputagarritasunaren teoria, while- eta makro-programak, funtzio unibertsala eta Church-en Tesia, gelditze-problema eta konputaezintasuna eta multzo erabakigarriak, partzialki erabakigarriak eta erabakiezinak.

Eduki praktikoentzat, while- eta makro- programen erabilera landuko da, interpretea erabiliz. Interpretearekin while- eta makro-programak exekutatzeko aukera izango dute era-

biltzaileek.

4.1 Helburu zehatzak

Aplikazioaren garapenari dagokionez, ondorengo helburuak beteko dira, gutxienez:

- Erabiltzaileek aplikazioaren funtzionamendua modu intuitibo batean ulertzea.
- Mugikor desberdinetatik atzituko den aplikazioa izanik, hauen pantaila tamainara moldatzen den aplikazioa izatea.
- Behin aplikazioa amaituta, eskuraketa prozesua errazteko asmoz, Googleko Play Storean igoko da. Plataforma horretatik *Android* sistema erabiltzen duten erabiltzaileek aplikazioa mugikorrean deskargatu ahal izango dute.
- Ikasleen ikasketa pertsonala eta parte-hartzea bilatzen duen aplikazioa izanik, Gamifikazio metodologia erabiliko da. Hau da, klasetik kanpo ikasketa-prozesua bultzatzeko jokoetatik eratorriko ezaugarriak erabiliko dira aplikazioan, hau erabiltzaileentzat entretenigarria den tresna batean bihurtzeko.

4.2 Gamification

Gaur egungo ikasleak, mundu digitalean jaiotako ikasleak dira. Garapen teknologikoen eskutik hezi dira eta aplikazio teknologikoak erabiltzeko berezko dohaina dute. Hezkuntza arloan ikasleen ezaugarriak kontuan hartu behar dira ikasgaiaren irismena eta motibazioa definitzeko. Irakasleek, ikasleen ezaugarriei erantzuna emateko asmoz, irakaskuntza digitala bultzatzen saiatu dira. Metodologia desberdinak daude aplikazio teknologikoak hezkuntza munduan integratzeko; horietako teknologia ezagunena Gamifikazioa (*Gamification*, ingelesez) metodologia da.

4.2.1 Zer da eta zergatik erabili hezkuntzan

Gamifikazioa, lehia, motibazioa eta parte-hartzea indartzea areagotzen duen metodologia da [Kiryakova et al., 2014]. Metodo honek jokoen ezaugarriak erabiltzen ditu; hala nola, puntu bidezko sistema, erronkak, mailak, ranking-ak, etab.

Gamifikazioa ez da jokoetan oinarritutako ikasketarekin nahasi behar, bi gauza guztiz desberdinak baitira. Jokoetan oinarritutako ikasketak joko erreal bat erabiltzen du ezagutza eta trebetasunak irakasteko. Ikasketarako jokoak, hasiera eta behin-betiko amaiera duen unitate bereizgarriak dira. Ikasleek jokoa amaiera duela dakite eta "irabazte-egoera" batera iritsi daitezkeela. Gamifikazioarekin, bestalde, jokoen ezaugarri batzuk erabiltzen dira, soilik. Ikasleek ez dute joko oso bat jokatzen, hasieratik amaierara arte; erronkak irabazten, puntuak lortzen edota mailaz igotzen dira zenbait helburu lortzerakoan.

Metodo hau hainbat arlo desberdinetan erabili daiteke, bai marketing-ean bai hezkuntzan. Hezkuntzan geroz eta indar handiagoa hartzen ari den kontzeptua da, batik bat digitalizazioak eraginda. Izan ere, gaur egun ia edonon eta edonoiz daukagu eskura ordenagailu edo mugikorren bat, eta distrakzio bat izan daitekeen arren, gure helbururako erabili ditzakegu. Teknologiak eskaintzen dizkigun tresnetaz baliatuz, hezkuntza bultzatu nahi da. Gamifikazioarekin, jokoek dituzten ezaugarriekin, erabiltzailearen ikasketa-prozesua bideratzea, klasean ikasitako kontzeptuak indartzea eta ikasketarako motibazioa areagotzea bilatzen da, modu entretenigarri batean. Metodologia honen eraginkortasunari dagokionez, 2013. urtean Georgiako unibertsitateko John Dobson irakasleak egindako ikerketan [Dobson, 2013] galdetegi laburrek gelan ikasitako kontzeptuak errepasatzen laguntzen dutela frogatu zuen. Galdetegiak egin zituzten ikasleek galdetegiak egin ez zituztenek baino % 40 asmakuntza-tasa altuagoa izan zuten ondoren gauzatutako galdetegietan .

Kontzeptuen definizioa

Atal honetan azalduko diren zenbait terminoen azalpena:

- **Feedback**: erabiltzaileek erabili dezaketen tresna, bere garapenetik zenbat ikasi duten ikusteko. Esaterako, ikasleek egindako akatsak ikusiz ikasi dezakete.

- **Arauak**: aplikazioan erabiltzaileek egin dezaketen erronka eta faseen mugak. Ikasketa-prozesua bideratzeko asmoz, erabiltzaileak ikusi dezakeen ingurunea moldatuko da, bere garapenaren arabera. Adibidez, bigarren erronka egin ahal izateko lehen erronkako helburu guztiak lortu beharko dira.

- **Maila**: Ikasketa pertsonala neurtzeko erabilitako neurria. Puntu-sistema baten bitartez erabiltzaile bakoitzaren maila inkrementatuko da, honek helburuak bete ahala.

4.2.2 Hezkuntza arloan nola aplikatu

4 pausoko prozesua [Huang and Soman, 2013] jarraitu behar da Gamifikazio kontzeptua modu egokian aplikatu ahal izateko.

1. **Pausoa: Ingurunea eta audientziaren ezaugarriak identifikatu**

   Norentzat izango da aplikazioa? Erabiltzaileen eta aplikazioa erabiliko den inguruneko ezaugarriak zehaztuz, gamifikazio metodorako baliabide egokiak sortu ahal izango ditugu. Aplikazioa unibertsitateko ikasleentzat bideratuta dago. Inguruneari dagokionez, unibertsitatean ez ezik, etxean edo unibertsitatera joateko bidean (trenean, autobusean, etab.) erabiltzeko aukera eskaini nahi da. Izan ere, unibertsitatean ikasitako kontzeptuen menderakuntza bilatzen da gehienbat, hala nola parte-hartzea motibatzea. Horregatik, mugikorrek eskaintzen duten eramangarritasunaz baliatuko gara.

2. **Pausoa: Irakatsi nahi diren kontzeptuak definitu**

   Zein kontzepturen ikasketa indartu nahi ditugu? KEA irakasgaiari buruzko aplikazioa izango da, hortaz ikasgaiaren kontzeptu garrantzitsuenak aukeratuko dira. While eta makro-programek protagonismo nabarmena izango dute, eta gainera, gai bakoitzeko kontzeptu teorikoak landu nahi dira. Kontzeptu teorikoak indartzeko, test motatako galdetegiak egongo dira eta while- eta makro-programak lantzeko programak idazteko ariketak.

3. **Pausoa: Metodologiaren ibilbidearen gida** Nola bideratuko ditugu erabiltzaileak? Erabiltzaileak bideratzeko erronkak sortuko dira. Erronka horietako bakoitzean zenbait ariketa edota galdetegi bete behar izango dituzte hurrengo erronkara pasatu ahal izateko.

   Erronkak erabiltzaileen eta ingurunearen ezaugarrietara moldatzeko kontuan hartu behar diren kontzeptuak honakoak dira, batik bat: **Arreta**, erabiltzaile gazteen arreta mantentzea zaila gerta daiteke. Hortaz, erronken iraupena laburra izan beharko luke. **Motibazioa**, aplikazioan interesa ez galtzeko, KEA ikasgairako baliagarri izan daitezkeen erreminten garapena egin da. **Gaitasunak**, erronken zailtasuna erabiltzaileek lortutako gaitasunetatik urrun egongo balitz, parte-hartzea sustatu ordez, galtzeko arriskua egon daiteke. Ezaugarri hauek kontuan hartuta erabaki da: erronken zailtasuna ikasgaiaren garapenarekin batera inkrementatuko dela, 10-15 minutuko iraupena izango dituztela erronkek eta erronkez gain ikasgaiaren baliabide izan daitezkeen erreminten garapena gauzatuko dela.

4. **Pausoa: Gamifikazio kontzeptuaren elementuak gehitu**

   Nola bateratuko dugu Gamifikazio metodologiarekin? Behin erronkak eta faseak definituta, gamifikaziorako erabili daitezkeen baliabideak sortu behar dira. Hauek dira aplikazioan erabili diren baliabideak:

   - Puntuazio-sistema: puntu bidezko sistema garatuko da, erabiltzaile bakoitzak bere ikasketa-prozesua neurtu ahal izateko. Galdetegiak eta ariketak egin ahala, puntuak lortuko dituzte.
   - Mailak: erronkak bete ahala, mailaz igotzeko aukera izango dute erabiltzaileek. Mailaz igo ahala, aplikazioaren ingurune zabalagoa eskainiko zaie; hau da, baliabide gehiago izango dituzte eskura, lehen ikusi ezin zitzaketen eremuak erakutsiz.
   - Ranking-ak: erabiltzaileen arteko lehia bultzatzeko asmoz, modu publiko batean erakutsiko da puntu gehien dituzten lehenengo 10 erabiltzaileen izenak. Honekin parte-hartzea areagotzea lortu nahi da.
   - Galdetegiak eta ariketak: test moduko galderak egongo dira eduki teorikoak birpasatzeko eta ariketak eduki praktikoentzat. Akatsak erakutsiz, klasean ikasitako edukiak berrikustea bilatzen da, ikasketa indartzeko. Galdetegi edo ariketa bakoitzean lortutako asmatzeak erakutsiko dira, erabiltzaileek feedback-a jaso dezaten, eta bakoitzak bere ezagutza-maila neurtu dezan.
   - Erremintak: aplikazioa erabiltzeko motibazioa areagotzeko, klasean erabili ditzaketen tresnak garatu dira. Tresna hauek erronketatik kanpo dauden edukiak dira, hala nola datu-moten arteko bihurgailua.
   - Erronkak: erronka bakoitza helburu zehatz baten lorkuntzarekin bat dator. Helburu horiek lortuz, mailaz igotzea lortuko da. Erronka bakoitza 10-15 minutukoa izango da, gehienez, ikasleen arreta mantentzeko.
   - Faseak: ikasgaiarekin bat etortzeko eta erronken zailtasuna gradualki inkrementatzeko, 3 fase desberdindu dira. Bakoitza eduki konkretu batzuei dedikatua: 1.fasea: while-programa sinpleena, 2.fasea: makro-programa sinpleena eta 3.fasea: DMAk integratuta dituzten makro-programena.

Gamifikazioa ikasketarako tresna egokia bihur daitekeen metodologia da, erabiltzaileen eta hauen ingurunearen ezaugarrietara moldatuz gero.

Erabiltzaileen arreta, gaitasunak eta motibazioa mantenduz, ikasgelan landutako kontzeptuen berrikusketa bultzatzen da erronka laburren bidez. Hezkuntzarako metodo hau gaur

egun ia edonon aurki ditzakegun mugikorreko aplikazio batean garatuko da, unibertsitateko ikasleen motibazioa, ikasketa pertsonala eta parte-hartzea bultzatzeko asmoz. Ikasleen portaeran modu positibo batean islatuko dela aurreikusten da, baina aplikazioa erabiltzaileen ezaugarrietara moldatzeko noizbehinka egokitu beharko dela aurreikusten da, ikasleen beharrak aldatu ahala.

4.3 Tresnak eta aplikazioak

4.3.1 Android

Android mugikorretako sistema eragilerik erabiliena da. 2017. urtean saldutako mugikor guztietatik %85ak *Android* sistema eragilea zuen. Bigarren postuan iOS sistema dago, saldutako mugikorren %14.7arekin. *Android*-en nagusitasunak datorren urteetan mantentzea espero da. Hortaz, mugikorretan sistema erabiliena izateagatik aukeratu da aplikazioa garatzeko.

Plataforma honetarako aplikazioak garatzeko jatorrizko lengoaia Java da eta horregatik bai aplikazioa bai interpretea lengoaia horretan garatzea erabaki da.

Aplikazioa *Android 4.0.3* bertsio minimorako prestatuta dago, eta proiektua konpilatzeko, *Android 8.1* bertsioa erabili da.

4.3.1.1 Android Studio

Android aplikazioa *Android* Studio ingurunean garatuko da. *Android* Studio, JetBrainseko Intellij IDEA softwarean oinarrituta dago eta Apache 2.0 lizentziapean argitaratu da, dohainik. Linux-etik erabili daiteke eta bereziki *Android*-eko garapenera zuzenduta dagoelako aukeratu da. Hauek izan dira aplikazioan erabili diren liburutegiak:

- *Leonids:* galdetegi bat amaitzerako orduan, honetan asmatutako erantzun kopurua erakusterakoan sortzen den konfetia marrazteko liburutegia [1].
- *Ion:* galdetegietan erakusten diren argazkiak mugikorreko pantailaren tamainera moldatzeko eta errazago deskargatzeko liburutegia. [2].

[1] https://github.com/plattysoft/Leonids
[2] https://github.com/koush/ion/

- *Cardview:* orrialde nagusiko botoien eta galdetegi-zerrendaren diseinurako erabili diren elementuak. [3].

- *OkHttp: Android* eta *Java* aplikazioentzako HTTP/HTTPS bezeroa da. Honekin bezero eta zerbitzariaren artean datuak bidali daitezke [4].

- *ACRA:* aplikazioko erroreen informazioa duen fitxategia mezu bidez bidaltzeko liburutegia [5].

4.3.2 Raspberry Pi 3

Aplikazioak zenbait datu gorde beharko ditu: erabiltzaile izena, pasahitza, erabiltzaile bakoitzak egindako galdetegiak, puntuak, erronkak, etab. Datu horiek gordetzeko datu base bat behar da, eta horretarako nolbaiteko zerbitzari bat. Aplikazioaren garapen faserako Raspberry Pi 3 bat erabili da, bere prezio merkearen, eta sinpletasunarengatik.

Raspberry Pi kreditu-txartelen tamainadun eta zirkuitu-plaka bakarreko ordenagailu pertsonalen serie bat da. Bezero-zerbitzari datu transferentzia probatzeko erabili da, aplikazioaren funtzionamendu egokia mantentzen dela bermatuz.

Zerbitzari minimalista hau eskasa gerta daiteke erabiltzaile askoren aldibereko aktibitatea kudeatzeko. Aplikazioaren erabilera datorren kurtsoan erabili nahiko balitz, beste zerbitzari bat erabili beharko litzateke. Izan ere, Rapsberry Pi 3-a baliabide pertsonala da, eta etorkizunean ez da honen mantentze-lanik espero. Zerbitzari hobe baten beharra dago, non erabiltzaileen datuen kopiak gorde beharko diren, edozein arazorengatik datuak ezabatuko balira, kopietatik datuak berreskuratzeko.

4.3.3 PHP

PHP (PHP: Hypertext Preprocessor) interpretatutako programazio lengoaia bat da. Eskaintzen dituen abantaila nagusiak datu-base sistema ugarirekin funtzionatzeko aukera izatea eta sistema eragile gehienetarako eskuragarri izatea dira.

PHP-z zerbitzariaren datu-basea maneiatzeko fitxategiak idatzi dira.

[3] https://developer.android.com/reference/android/support/v7/widget/CardView
[4] https://github.com/square/okhttp
[5] https://github.com/ACRA/acra

4.3.3.1 PhpStorm

PHP lengoaiaz idazteko *PhpStorm* IDE-a [JetBrains, 2009b] erabili da. IDE hau, interpretea garatzeko erabili den *IntellIJ* IDE-aren garatzaile berak sortu du, *JetBrains*-ek hain zuzen.

4.3.4 MySQL

Datu-baseak kudeatzeko sistema ezagunena MySQL da. Bertan aplikazioak behar dituen datu guztiak gorde daitezke: bai datu-pertsonalak zein aplikazio barneko datuak (puntuazioa, galdetegien emaitzak, etab.). Datu basea sortu, diseinatu eta kudeatzeko *phpMyAdmin* erabili da, PHP-z idatzitako erreminta honek, datu-basearen kudeaketa web nabigatzaile baten bitartez egitea ahalbidetzen du.

4.3.5 Google Play Store

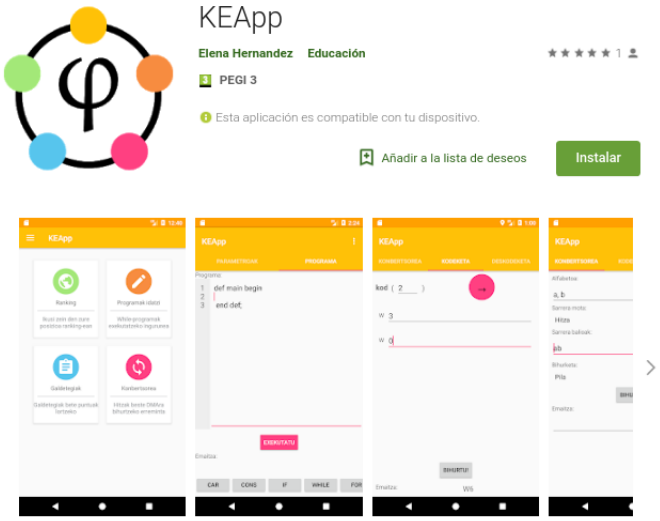
Google Play Store, *Android* sistema eragilea duten mugikorrentzako aplikazioen denda ofiziala da. Honek *Android*-entzat bereziki garatutako bai dohaineko bai ordainpeko aplikazioak deskargatzeko aukera ematen du.

KEApp aplikazioa dohainik argitaratu da plataforman (ikus 4.1 irudia), inolako publizitaterik gabe, ikasleek ordaindu behar ez izateko, eta motibazioa mantentzeko. Google Play plataformak eskaintzen duen edukiari buruzko analisia egin ondoren, aplikazioa edozein adineko (herrialde batzuetan, 3 urte baino gehiagoko) publikorentzat egokia dela zehaztu da.

Aplikazioa plataformara igotzeko, Google Play Console-n [Google, 2008] kontua sortu behar izan da. Hau Google Play-eko aplikazioak igotzeko eta kudeatzeko sortu den plataforma da. Bertatik, aplikazioari buruzko informazio guztia ikusi daiteke, bai zenbat pertsonek jaitsi duten, bai zenbatek desinstalatu duten.

4.3.6 Inkscape

Aplikazioaren logoa (ikus 4.2 irudia) sortzeko erabili den softwarea da. Irudi-bektorialak sortzeko erabiltzen da, eta irudien kalitatea tamainarako edozein dela izanda mantentzen dela bermatzeko aukeratu da.



4.1 Irudia: KEApp Google Play pataforman

4.2 Irudia: KEApp aplikazioaren logoa

4.4 Aplikazioaren erabilera eta inplementazioa

Atal honetan, aplikazioaren nabigazioa azalduko da, bai eta hau gauzatu ahal izateko sortu den kodearen azalpen orokorra.

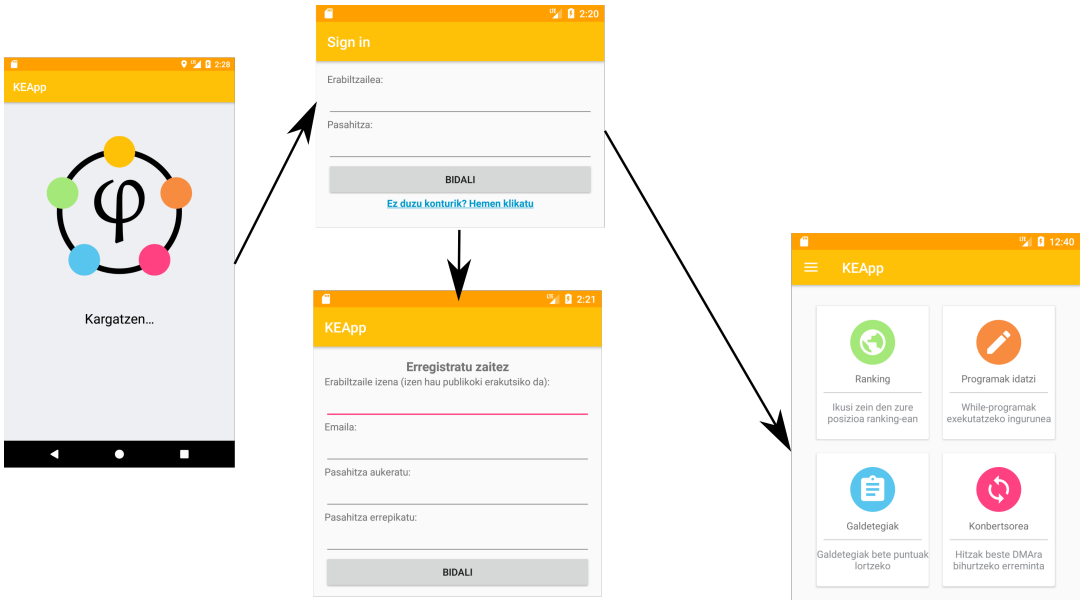
4.4.1 Diseinua

Diseinuak garrantzi handia izan du proiektuan, hau aplikazioaren funtzionamendua ahal den heinean sinplifikatzeko erabili delako.

4.4.1.1 Nabigazio-eskema

Aplikazioa intuitiboa izatea saiatu da. Interfazearen nabigazioa hobeto azaltzeko, atal bakoitza indibidualki azalduko da. Aplikazioaren nabigazio globala ikusi ahal izateko ikus F eranskina.

Aplikazioa erabiltzeko, kontu bat sortu behar da (ikus 4.3 irudia). Saioa hasteko pantailan *Ez duzu konturik? Hemen klikatu* testuan klikatu behar da. Ondoren erregistroa gauzatzeko pantaila agertuko da. Erregistroa gauzatuta, saioa hasi beharko da. Azkenik, orri-nagusia erakutsiko da, lau aukera nagusiekin: ranking-ak ikusi, datu-moten arteko bihurgailua erabili, galdetegiak bete edo programak exekutatu.

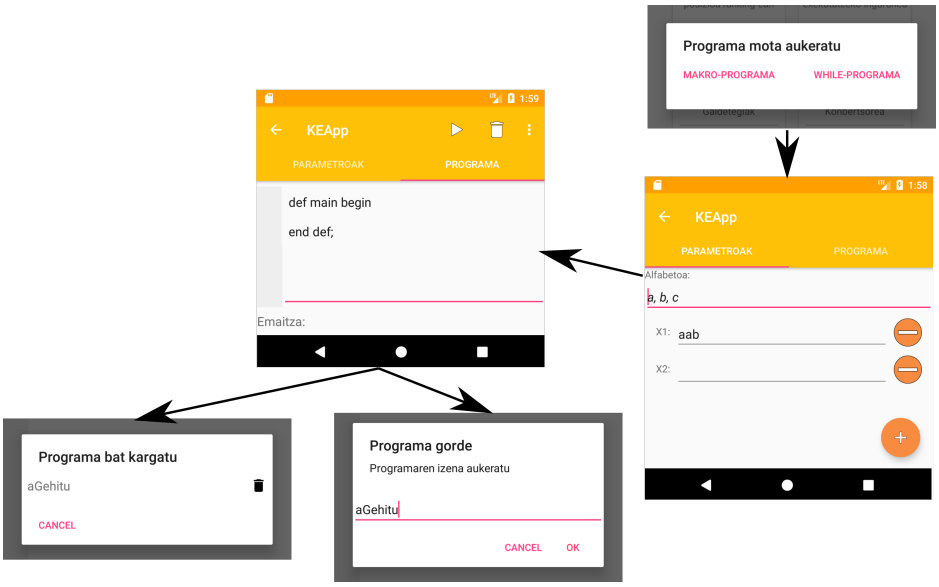


4.3 Irudia: Kontua sortzeko eta saioa hasteko prozesuen nabigazio-eskema

Ranking-a ikusteko aukera sakatuz (ikus 4.4 irudia), puntuazio altueneko 10 erabiltzaileen izenak eta puntuazioak erakutsiko dira, erabiltzaile bakoitzaren posizioaz gain. Erabiltzailearen posizioa lehen 10 postuetan ez badago, ezin izango du rankingean ikusi. Horregatik, erabiltzaile bakoitzak bere posizioa azpian ikusi ahalko du.

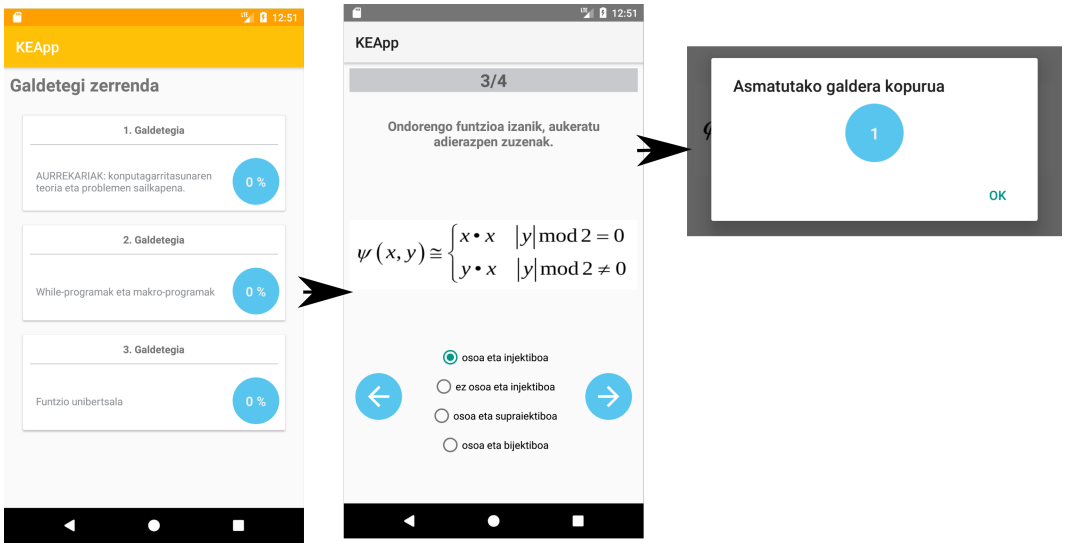
4.4 Irudia: Ranking-a erakusteko pantaila

Programak idazteko aukera sakatuz (ikus 4.5 irudia), programa-mota aukeratu beharko dugu: while- ala makro-programa. Ondoren, programan erabili beharreko parametroak sortzeko eta hasieratzeko pantaila erakutsiko da. Hortik, programak idazteko pantailara joan daiteke. Behin programak idazteko pantailan egonda, aurretik sortutako programaren bat kargatzeko aukera dugu, edo sortu dugun programa hurrengo baterako gordetzeko, goiburuko botoiak erabiliz.



4.5 Irudia: Programak idazteko nabigazio-eskema

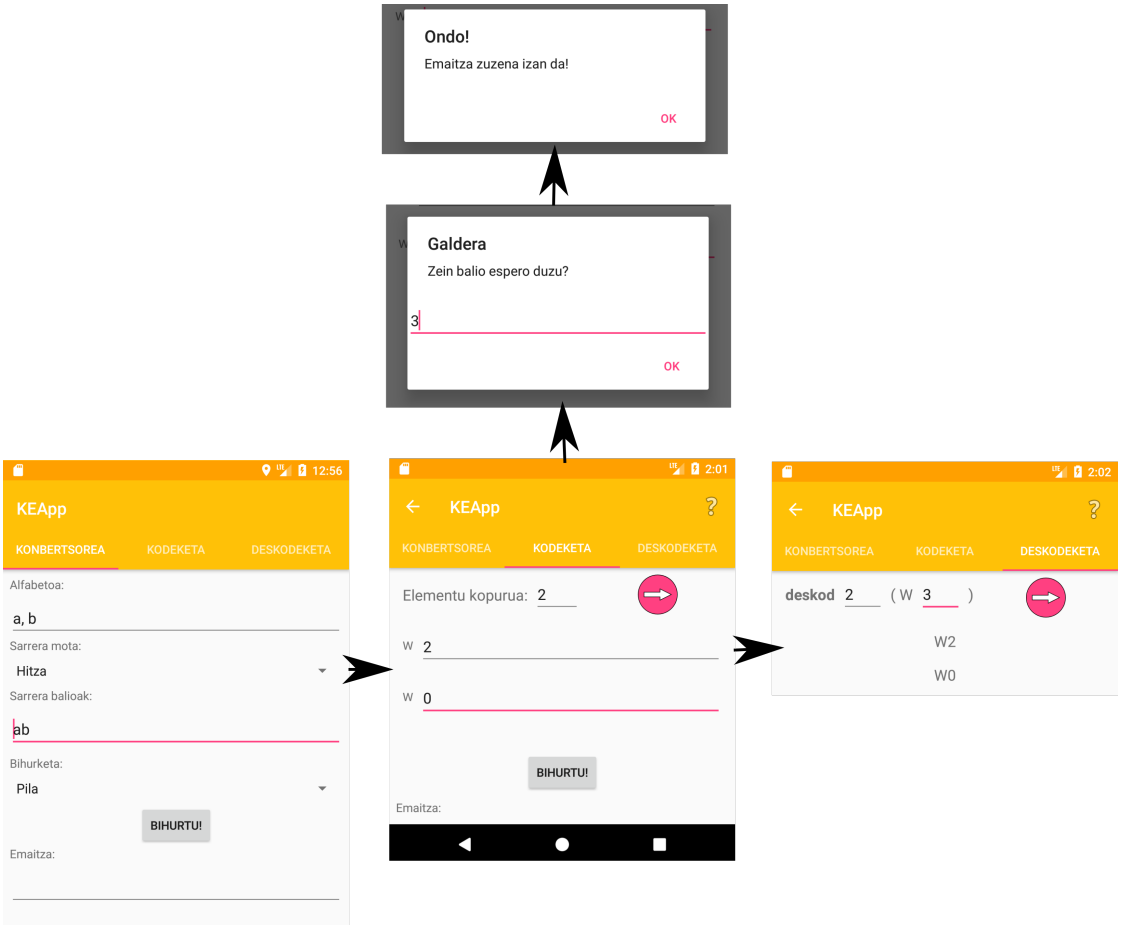
Galdetegiak betetzeko aukera sakatuz (ikus 4.6 irudia), galdetegi guztien zerrenda agertuko zaigu. Galdetegi bat aukeratuz, galdetegi horretako galderak agertuko dira. Galdera bakoitzak 4 erantzun posible dauzka, eta irudi bat izan dezake. Galdera guztiak erantzuterakoan, emaitzak bidaltzeko botoia agertuko da. Honekin, galdetegi horretan asmatutako galdera kopurua lortuko da.



4.6 Irudia: Galdetegiak betetzeko nabigazio-eskema

Bihurgailua erabiltzeko aukera sakatuz (ikus 4.7 irudia), datu-moten arteko bihurketak

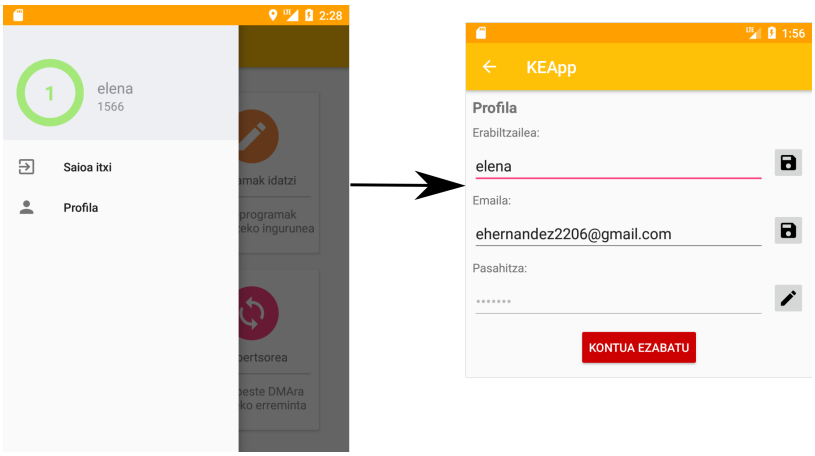
gauzatu ditzakegu. Gainera, zenbakizko hitzen kodeketa eta deskodeketa gauzatzeko aukera dago, *Kodeketa* eta *Deskodeketa* ataletan sakatuz gero.



4.7 Irudia: Bihurgailua erabiltzeko nabigazio-eskema

Lau aukera nagusi horiez gain, *RGPD* legea bermatzeko asmoz (ikus 4.4.5 atala), erabiltzaile bakoitzak bere profileko datuak ikusteko, moldatzeko eta ezabatzeko aukera izango du (ikus 4.8 irudia).

Orrialde-nagusitik ireki daitekeen albo-nabigazioan, erabiltzaileak lortutako puntuazioa eta maila ikusi dezake. Gainera, saioa ixteko aukera dute.



4.8 Irudia: Erabiltzaileek bere profileko datuak ikusteko, aldatzeko eta ezabatzeko nabigazio-eskema

4.4.1.2 Erabilpen-kasuak

Atal honetan erabiltzaile batek prozesu bat aurrera eramateko jarraitu beharreko pausoak azaltzen dira (ikus 4.9 irudia).

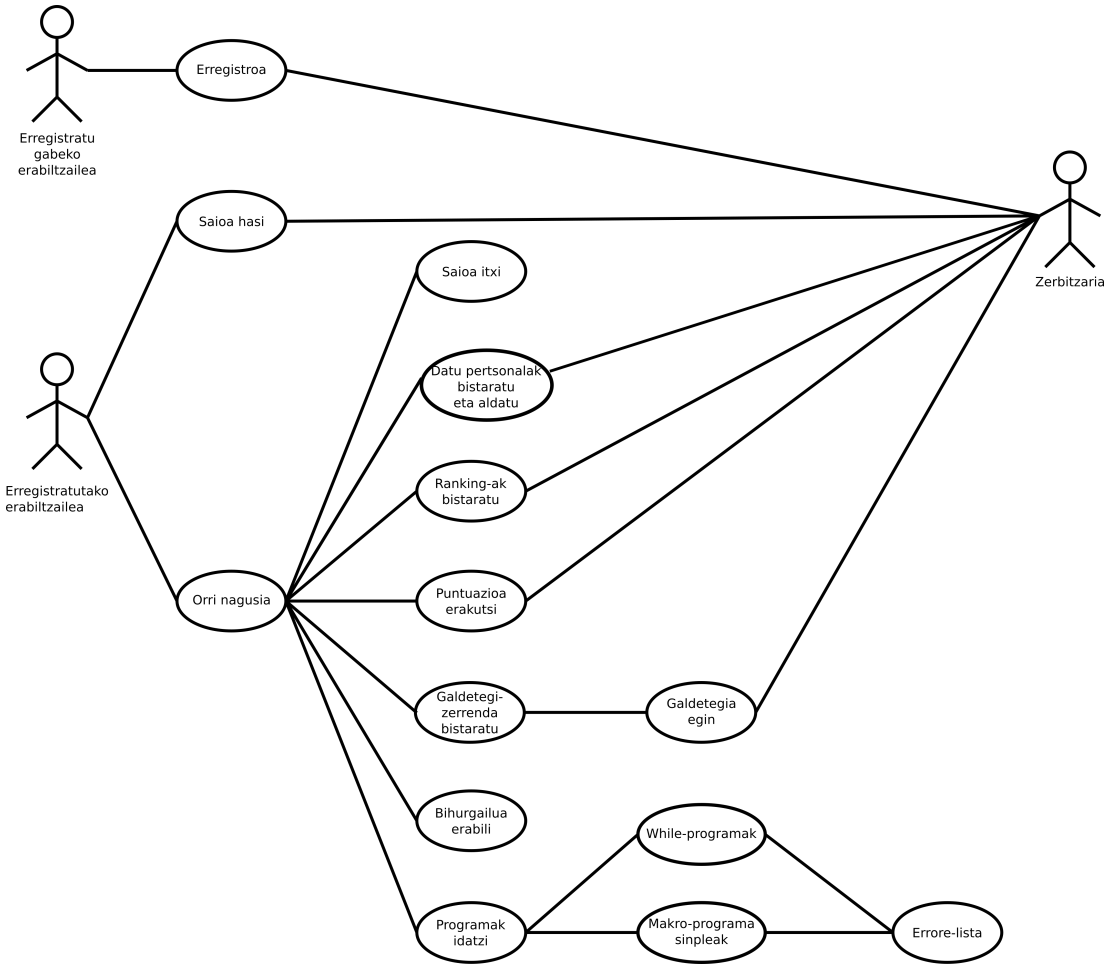
Aplikazioaren erabilpenean, hiru aktore desberdindu daitezke:

Erregistratu gabeko erabiltzaileak:

- **Erregistroa**: erabiltzaile bakoitzak erabiltzaile izena, email-a eta pasahitza aukeratu beharko ditu erregistratzeko. Ondoren, saioa hasteko aukera eskainiko zaio.

Erregistratutako erabiltzaileak:

- **Saioa hasi**: erabiltzaile izena eta pasahitza sartu behar ditu. Ondoren hasierako orria erakutsiko zaio.
- **Orri nagusia**: erabiltzaileak modu intuitiboan aplikazioak eskaintzen dituen erremintak eta bere kontua kudeatzeko aukera izango ditu.
- **Saioa itxi**: behin saioa hasita, saioa ixteko aukera eskainiko zaio.
- **Datu pertsonalak bistaratu**: erregistroan eskatutako datuak ikusteko aukera izango du, hala nola bere maila eta metatutako puntuak ikusi ahalko ditu.
- **Konfigurazioa aldatu**: aplikazioaren ezaugarriak aldatzeko.



4.9 Irudia: Erabilpen-kasuak

  - Profila aldatu: erabiltzailearen datu pertsonalak ikusteko eta aldatzeko. Gainera, kontua ezabatzeko aukera eskaintzen da.
  - Elementuen koloreak aldatu: kolore argiak ala ilunak. Fondoko koloreak eta hitzena aldatzeko aukera eskaintzen du.

- **Galdetegi-zerrenda bistaratu**: galdetegi guztiak erakutsiko dira, eta egindako galdetegiak badira, hauetan lortutako emaitzak.
  - Galdetegiak egin: bi motatako galdetegiak daude, test modukoak eta hutsuneak betetzekoak. Galdetegi bakoitza hainbat galderaz osatuta dago. Azken galderen emaitzak bidaltzeko botoia agertuko da, eta ondoren lortutako puntuak erakutsiko zaizkio.

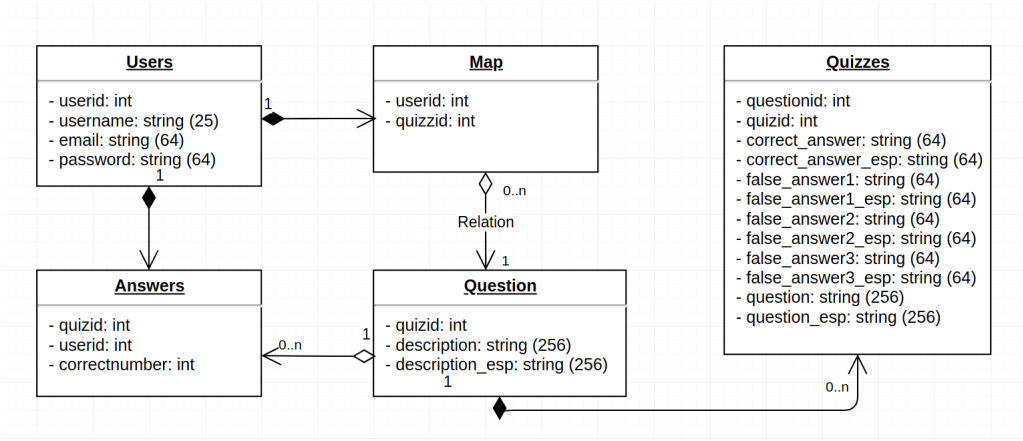
- **Bihurgailua erabili**: datu-mota desberdinen arteko bihurketak egiteko bihurgailua da.

- **Programak idatzi**: exekutatu nahi den programa mota aukeratu eta hau exekutatzeko aukera eskainiko zaio.
  - While-programa: hasieran while-programak exekutatzeko aukera eskainiko da.
  - Makro-programa sinpleak: hasieran makro-programak exekutatzeko aukera eskainiko da.
  - Errore-lista: Hauetako edozein programa motan erroreak egonez gero, errore-listan erakutsiko dira, errore gramatikalak badira, mezu bat agertuko da .

4.4.1.3 Datu-ereduak

Atal honetan zerbitzarian gordetzen diren ereduen eta hauen arteko erlazioak agertzen dira.

- **Users**: erabiltzaileen informazioa.
  - `userid`: erabiltzaile bakoitzaren identifikadorea.
  - `username`: erabiltzaileek aplikazioan ibiltzeko aukeratu dezaketen izena. Hau ranking-en atalean erakutsiko da, non beste erabiltzaileek ikusi ahal izango duten.



4.10 Irudia: Datu-ereduen eskema

  - `email`: erregistratzerakoan eskatzen den emaila. Erabiltzaile batek hainbat kontu sortzeko prebentzio neurria.
  - `password`: erabiltzaileek bere kontura sartzeko pasahitza. Pasahitza gutxienez 6 karaktereko luzera izan behar du. Segurtasun arrazoiak direla eta, pasahitza enkriptaturik gordeko da. Enkriptatzeko *bcrypt* algoritmoa erabili da.

- **Quizzes**: galdetegien informazioa.
  - `quizid`: galdetegien identifikadorea.
  - `description`: galdetegi bakoitzaren edukia deskribatzeko atala, euskaraz.
  - `description_esp`: galdetegi bakoitzaren edukia deskribatzeko atala, gaztelaraz.

- **Map**: erabiltzaile bakoitzak betetako galdetegien informazioa gordetzeko. Datuen arteko elkarketak errazteko sortu da.

- **Questions**: erantzunen informazioa.
  - `questionid`: erantzunen identifikadorea.
  - `quizid`: galdetegien identifikadorea. Galdera bakoitza zein galdetegiren barruan dagoen adierazteko.
  - `correct_answer`: emaitza zuzenaren testua, euskaraz.
  - `false_answer1, false_answer2, false_answer3`: emaitza okerren testua, euskaraz.
  - `question`: galderaren testua, euskaraz.

  - `correct_answer_esp`: emaitza zuzenaren testua, gazteleraz.
  - `false_answer1_esp, false_answer2_esp, false_answer3_esp`: emaitza okerren testua, gazteleraz.
  - `question_esp`: galderaren testua.

  Galdera bakoitzak, irudi bat izan dezake, irudi hauek datu-basean gorde ordez, zuzenean zerbitzarian gordetzen dira, fitxategi moduan. Irudiak, dagokien galderaren identifikadorearekin izendatu behar dira. Irudi hauek gehitzeko, zerbitzarian`qimages` karpetan gehitu behar dira. Adibidez, `qimages/12.png` fitxategia, 12 id-a duen galderaren irudia izango da.

- **Answers**: erabiltzaile bakoitzak zein galdetegi eta zenbat galdera asmatu dituen bertan jakiteko.
  - `quizid`: galdetegien identifikadorea.
  - `userid`: erabiltzaileen identifikadorea.
  - `correctnumber`: asmatutako erantzun kopurua.

Aplikazioan zerbitzaritik kanpo gordetzen diren beste datu batzuk daude.

Erabiltzaileek sortutako programak mugikorraren barne-memorian gordetzen dira. Bertatik kargatzeko aukera eskaintzen da eta zerbitzariarekiko dependentzia handiagotzea ekiditen da.

Bestalde, erabiltzaileen puntuazioa eta maila ez dira zuzenean datu-basean gordetzen. Informazio hau galdetegietan erantzundako galdera kopuruaren arabera kalkulatzen denez, zerbitzariari galdetuz lortzen da. Puntuazioa, zuzen erantzundako galdera kopuruaren araberakoa da. Galdetegi bat guztiz zuzen erantzuteak 1000 puntu ematen ditu. Galdera bakoitzaren puntuazioa 1000 / galdetegiaren galdera kopurua da; galdetegi batek geroz eta galdera gehiago izan, orduan eta puntu gutxiago balioko du galdera bakoitzak. Maila, erabiltzailearen puntuazio totalaren menpekoa da; maila = puntuazioa / 1000 izanik. Hortaz, galdetegi bat guztiz ondo erantzuteak maila bat igoko du, baina bi galdetegi erdi-ondo erantzuteak ere.

4.4.2 Kodearen egitura

Aurrera jarraitu baino lehen, zenbait kontzeptu definitu behar dira:

- *Activity*: erabiltzaileek mugikorraren pantailarekin elkar eragiteko aplikazioaren osagaiak dira. *Activity* bakoitzeko *layout* bat esleitzen zaio, hau da, pantailean erakutsiko diren elementuak eta hauen kokapenaren distribuzioaren diseinua.
- *Fragment*: *Activity* motako osagaiek izan ditzaketen elementuak dira. Sekzio modular bezala erabili daitezke, *Activity* bakar batetik hainbat *layout* atzitu ahal izateko.
- *Adapter*: objektuen zerrendak sortzeko beharrezko elementuak. *RecyclerView* motako elementuak erabiltzen dira, elementuen listan errendimendu handirik erabili gabe mugitzea ahalbidetzeko.

Kodea bi zati nagusitan banatu da: bezero aldean erabiltzen dena [Hernandez, 2018b] eta zerbitzari aldekoa [Hernandez, 2018c].

4.4.2.1 Bezeroa

Android aplikazioa garatzeko honako egitura jarraitu da:

- `SplashActivity`: aplikazioa irekitzerakoan kargatzen ari den animazioa erakusteko.
- `RegisterActivity`: aplikazioan kontua sortzeko.
- `LoginActivity`: saioa hasteko.
- `MainActivity`: orri nagusia erakusteko.
- `SettingsActivity`: aplikazioaren konfigurazio orokorra aldatzeko.
- `GalderaActivity`: galdera bakoitza erakusteko. Galdetegia bukatzerakoan, erantzunlista bidaliko dio zerbitzariari eta honek zenbat erantzun zuzen dauden erantzungo du.
- `GaldetegiZerrendaActivity`: galdetegi guztiak erakusteko. Galdetegi bat klikatzerakoan, honen galderak kargatuko ditu.
- `KodDekodActivity`: Bihurgailua erakusteko. Bertan hiru *fragment* daude:
  - `CalculatorFragment`: datu-mota batetik beste batera bihurtzeko leihoa.
  - `DeskodFragment`: zenbakizko deskodeketa gauzatzeko leioa.

  - `KodFragment`: zenbakizko kodeketa gauzatzeko leihoa.

- `KonfigInfoActivity`: profileko datuak erakusteko.

- `ProgActivity`: programak idazteko.

  - `ParamFragment`: programak sortu aurretik, honetan erabiliko den alfabetoa eta parametroen hasieraketa gauzatzeko leihoa.
  - `ProgFragment`: programak idazteko leihoa.

- `RankingActivity`: puntuazio altuena duten 10 erabiltzaileak eta erabiltzailearen posizioa erakusteko.

Proiektuan hainbat *adapter* erabili dira:

- **Kodeta eta deskodeketan**: kodeketan, sarrerako zenbakizko hitzak eskatzeko eta deskodeketan, irteerako hitzak erakusteko.
- **Errore lista**: programa bat exekutatzerakoan, honetan erroreak egon badira, hauen deskribapena eta posizioa erakusteko.
- **Galdetegietan**: galdetegi zerrenda erakusterakoan, galdetegi bakoitzaren informazioa erakusteko.
- **Parametroetan**: programak idazterako orduan, parametroen balioak hasieratzeko.
- **Programak gordetzeko eta kargatzeko**: erabiltzaileak gordetako programak zerrendaturik agertuko dira programak kargatzeko botoia sakatuz gero.

Kodearen antolakuntza errazteko asmoz, ondorengo klaseak sortu dira:

- `Galdera`: galdera bakoitzak dituen elementuen atzipena errazteko. Galdera bakoitza identifikadoreaz, 4 erantzun posiblez eta enuntziatuaz osatuta dago.
- `Quizz`: galdetegi bakoitzaren identifikadorea, erabiltzaileak egindako galdetegietan lortutako erantzun zuzenen kopurua, deskripzioa eta galderen kopuru totala gordetzen dira klasean.
- `RunPrograma`: while- ala makro-programak exekutatzeko. Programa mota bakoitzaren araberako analizatzaile sintaktikoa sortuko du.

- `ServerRequest`: zerbitzariari deia egiterako orduan, kodea laburtzeko asmoz sortutako klasea.
- `SharedPrefManager`: aplikazioaren saio guztian zehar mantenduko diren datuak gordetzeko. Erabiltzailearen datuak behin eta berriz zerbitzariari eskatu beharrean, hauek saioa hastean gordeko dira eta saioa mantendu bitartean gordeko dira.

4.4.2.2 Zerbitzaria

Zerbitzariak datu-baseko informazioarekin lan egiten du. Informazio hori eguneratu, eskatu eta ezabatzeko eskaerak PHP lengoaian exekutatzen dira, honek MySQL datu baseen kudeaketa sistemaren bitartez datu baseko informazioa moldatuz.

Aplikazioaren funtzionamendurako honako fitxategiak sortu dira:

- `register.php`: erabiltzaile berriek kontua sortzeko eskaera. Erabiltzaile izena, pasahitza eta email-a gordeko dira datu basean, baldin eta erabiltzaile izena eta email-a errepikaturik ez badaude.
- `login.php`: erabiltzaile izena eta pasahitza emanda, saioa hasteko eskaera. Saioa hasita, hainbat eskaeren artean mantentzeko, Cookiak (informazio atal txikiak) erabiliko dira erabiltzailea identifikatu ahal izateko. Honekin zerbitzariaren eskaera kopurua minimizatzen da, eta erabiltzailearen identifikazioa eskaera guztietan bermatzen da.
- `logout.php`: erregistratutako erabiltzaileek saioa ixteko erabiltzen den fitxategia.
- `deleteAccount.php`: erregistratutako erabiltzaileek aplikazioko kontua ezabatzeko fitxategia.
- `crash.php`: aplikazioaren proba fasean, *alpha-tester*-ren batek aplikazioan erroreak aurkituz gero, errorea detektatzeko mezua bidaltzen duen fitxategia.
- `userInfo.php`: Cookien bitartez erabiltzailearen puntuak , maila, eta erabiltzaile izena lortzen ditu.
- `myranking.php`: erabiltzailearen posizioa ranking-ean bueltatzen du.
- `ranking.php`: puntuazio altueneko 10 erabiltzaileen erabiltzaile izenak eta puntuazioa erakusteko balio du.

- `quizzes.php`: galdetegien zerrenda erakusteko balio du, eta hauetan asmatutako galderen portzentaia kalkulatzeko informazioa.
- `question.php`: galdetegi bakoitzari dagokion galderak erakusteko balio du.
- `check_answers.php`: erabiltzaileak erantzundako galderak okerrak diren ala ez egiaztatzeko kodea. Galdetegiak erantzuterakoan, lehendik lortutako puntuazioa hobetzekotan, puntuazio berria datu basean gordeko da.
- `changeEmail, changePassword, changeUsername`: profileko datuak modifikatzeko fitxategiak. Emaila, pasahitza eta erabiltzaile-izena aldatzeko aukera eskaintzen dute, hurrenez hurren.
- `check_password`: pasahitza aldatzerakoan, lehendik zegoen pasahitza eskatzen da, segurtasuna bermatzeko. Lehendik zegoen pasahitza erabiltzaileari eskatutakoaren berdina dela egiaztatzeko fitzategia da.
- `PrivacyPolicy.html`: Pribatutasun Politika erakusteko fitxategia.

4.4.3 Erroreen detekzioa

Aplikazioa, proba-fasean egon bitartean, ixtarazten duen erroreren bat gertatuz gero, aplikazioa itxi eta email-kontua irekiko da. Aurreidatzitako mezu bat bidaliko da, non mezu horretan gertatutako errorearen informazioa biltzen duen fitxategi bat atxikituko da. Erabiltzaileak mezua bidali ala ez erabaki dezake, bidaltzearekin errorea konpontzeko informazio guztia bidaliko da *keaaplikazio@gmail.com* kontura.

Mezua bidaltzeko, *ACRA* liburutegia erabili da. Aplikazioaren bertsio berrian, honen kodea ezabatu da, ez baita etorkizun baterako mantentze-lanik egingo. Hala ere, kodearekin batera dagoen *README.txt* fitxategian honi buruzko informazioa dago.

4.4.4 Aplikazioaren erabilera

Erregistroa: aplikazioan ibiltzeko, erabiltzaileak kontua sortu behar du. Horretarako, erabiltzaile izena, emaila eta pasahitza aukeratu beharko dira.

Saioa hasi: erregistroan idatzitako erabiltzaile izena eta pasahitza sartuz, saioa hasi ahalko du erabiltzaileak.

Behin aplikazioan kontua sortuta eta saioa hasita, erabiltzaileek lau aukera nagusi dituzte, hauetako bakoitza orri nagusian agertzen diren lau botoiei dagokie: ranking-ak, programak idatzi, galdetegiak eta bihurgailua.

- **Ranking-ak**: erabiltzaileen puntuen araberako ranking-a. Gamifikazio metodologiaren elementua dira ranking-ak, hauek erabiltzaileen arteko lehia sustatzeko erabiltzen dira, honekin parte-hartzea areagotzeko asmoarekin. Pantailan, puntuazio altuena duten 10 erabiltzaileen izenak eta hauen puntuazioa erakutsiko da. Azpian, erabiltzaileak bere posizioa ikusi ahal izango du, baina lehen 10-en artean ez badago, hau ezin izango dute beste erabiltzaileek ikusi.
- **Programak idatzi**: While- edo makro-programak idatzi daitezke. Bakoitzaren gramatika desberdina denez, lehenengo idatzi nahi dugun programa mota aukeratzeko mezua agertuko da. Ondoren, bi atal erakutsiko zaizkigu: lehenengoa parametroen egokitzapenerako, eta bigarrena, programen garapen eta exekuziorako.
  - Parametroen egokitzapena, while-programetan aldagaien izenak beti finko mantentzen direla baliatuz (X1, X2, ..., Xn), hauek modu ordenatu batean sortzeko eta hasieratzeko helburuarekin sortu da. Aldagai berriak sortzeko "+" ikurra sakatu behar da eta aurretik sortutakoak ezabatzeko "-" ikurra.

    Gainera, alfabetoa definitzeko esparrua bete behar da. Honekin, parametroei balioak ezartzerako orduan edo programa idazterakoan, alfabetotik kanpoko karaktereak idatziz, errore sintaktiko bat dagoela adieraziko da.

    Programean behar diren aldagaiak sortu eta hasieratu daitezke, beti ere alfabeto barneko karaktereak erabiliz.
  - Bigarren pantaila programen garapen eta exekuziorako bideratuta dago. Pantailaren goiko aldean lau botoi daude: behin programa idatzita exekutatzekoa, programa ezabatzekoa, idatzitako programa gordetzekoa eta aurretik sortutako programak kargatzekoa. Pantailaren erdian, programa idazteko atala dago eta exekutatzeko botoia sakatu ondoren, emaitza azpian erakutsiko da. Programa idazteko testuaren ezker aldean lerro zenbakiak agertuko dira, errore bat gertatzean, arazoa non gertatu den azkarrago aurkitzeko. Exekutatzen denbora asko tardatzen duen programa idatziz gero (ziklatzen duen programa esaterako), programa gelditzeko botoia erakutsiko da exekutatzeko botoiaren ordez.

    Erroreak programa exekutatzerako botoia sakatzean erakutsiko dira, errorelista moduan. Bertan errorea gertatutako posizioa eta errorearen azalpen orokorra agertuko da.

Mugikorrean programak idaztea zaila gertatu daitekeenez, teklatuaren gainean hainbat botoi sortu dira, while- eta makro-programen egitura erabilienak botoia sakatzearekin idazteko.

- **Galdetegiak**: erabiltzaileek puntuak irabazteko galdetegiak bete ditzakete. Galdetegien atala sakatzerakoan, galdetegien zerrenda erakutsiko da, non galdetegi bakoitzaren laburpena eta galdetegi horretan asmatutako galderen ehuneko maximoa erakutsiko den.

  Galdetegi bat aukeratzerakoan, galdetegi horretako galderak erakutsiko dira bananbanan. Galdera batzuk argazki bat dute, hauek mugikorraren pantaila tamainara moldatuko dira. Pantaila txikietan, argazkien definizioa baxua izango delakoan, argazkien gainean klikatzerakoan hauetan zoom-a egiteko aukera dago.

  Azkeneko galderara iristerakoan, erantzunak bidaltzeko botoia agertuko da. Hau sakatuz, asmatutako galdera kopurua erakutsiko da. Aurretik ondo erantzundako galdera kopurua baino handiagoa bada, galdetegiaren ondoan agertzen den ehunekoa eguneratuko da, puntuazio altuenarekin.

- **Bihurgailua**: bihurgailua hiru ataletan banatu da: bihurgailua, kodeketa eta deskodeketa izenekoetan.

  – Bihurgailua datu mota batetik beste batera bihurtzeko erreminta da. Lehenik eta behin alfabetoa definitu behar da. Ondoren, sarrerako datu mota aukeratu: hitza, zenbakia, pila ala bektorea. Jarraian, dagokion sarrerako balioa idatzi beharko da. Balio hau aukeratutako datu motaren araberakoak dira, zenbakia baldin bada, soilik zenbakiak onartuko dira, hitza baldin bada, alfabeto bareko karaktere segida bakarra eta pila eta bektoreen kasuan elementuak komaz banandu beharko dira. Azkenik, helburu datu mota aukeratu eta “bihurtu” botoia sakatu behar da emaitza ikusteko.

  – Kodeketan zenbakizko hitzekin soilik lan egiten da. Kodetu nahi diren elementu kopurua sartuz eta gezia sakatuz, elementu kopuru haina eremu agertuko dira. Hauetako bakoitzean kodetu beharreko balioa idatzi eta gero, “bihurtu” botoia sakatu behar da. Gamifikazio metodologiarekin bat egiteko, emaitza agertu aurretik, espero den emaitza zein den galdetuko da. Balioa zuzena nahiz okerra izan ez du desberdintasunik, emaitza zuzena agertuko bait da.

  – Deskodeketa atalean ere, soilik zenbakizko hitzekin lan egiten da. Deskodetu nahi den zenbakizko hitza eta zenbat elementutan deskodetu nahi den esanez, kopuru horren araberako hitzen lista erakutsiko du gezia sakatzean.

Bihurgailua osatzen duten 3 ataletan, laguntza botoia agertzen da, atal bakoitzaren funtzionamendua azalduz. Laguntza botoia sakatzean, argazki bat erakutsiko da, atal bakoitzaren funtzionamendua pausoz-pauso azalduz.

Lau funtzionalitate nagusiez gain, erabiltzailearen kontuari buruzko informazioa agertuko da orri nagusiaren menuan. Hor, erabiltzaileak dituen puntuak eta maila agertuko dira, gainera, konfigurazio ataleko aukerak erakutsiko dira: kontutik irteteko, profileko datuak ikusi eta aldatzeko, pribatutasun politika erakusteko eta kontuaren konfigurazioa aldatzeko aukerak.

Legediaren arabera, erabiltzaileek beren datu pertsonalak ezabatzeko aukera, eta datu horiek nola babesten diren jakiteko eskubidea dute. Hau profileko informazioa erakusten den atalean aldatu ahal izango da.

Bestalde, aplikazioaren estiloa aldatzeko aukera dago: gris tonalitatea erabiliz edo defektuz erakusten den koloreekin.

4.4.5 Lizentziak eta datuen babesa

Lizentzia:

GNU GPLv3 lizentziapean [6] argitaratu da KEApp aplikazioa. Lizentzia honek, kodea berrerabiltzeko eta moldatzeko baimena ematen du, hala nola erabilera komertzialerako erabili daiteke. Kasu guztietan, kodea berrerabiliko balitz, lizentzia berdinaren menpean argitaratu beharko litzateke.

Datuen babesa:

Orain arte, datuen babeserako *Ley Orgánica de Protección de Datos* (LOPD) legedia errespetatu behar zen. 2018ko maiatzaren 25etik aurrera, *Reglamento General de Protección de Datos* (RGPD) [7] legediak LOPD ordezkatuko du. Aplikazioan erabiltzen den datu pertsonal bakarra email helbidea da. Emaila, aplikazioari buruzko informazioa bidaltzeko erabili daiteke, baina ez publizitatea bidaltzeko, eta ez da hirugarren bitarteko bati eskainiko. Edonola dela ere, legedi berriak erabiltzaileek bere informazio pertsonala zein den, nola tratatzen den eta zeinek ikusi dezakeen jakiteko eskubidea duela dio, hala nola, edozein momentutan informazio pertsonala aldatzeko eskubidea. Informazio pertsonalaren tratamenduari buruzko informazioa Pribatutasun Politikan zehaztu da. Erabiltzaileei,

[6] https://choosealicense.com/licenses/gpl-3.0/
[7] http://www.agpd.es/portalwebAGPD/temas/reglamento/index-ides-idphp.php

Pribatutasun Politikan [8] ezartzen diren terminoak baimentzeko aukera eskaini behar zaie, erregistratzerakoan. Dokumentua, bai aplikazioaren barnetik bai aplikazioa deskargatzerakoan ikusgai egon behar du.

[8] http://elenah.duckdns.org/PrivacyPolicy.html

5. KAPITULUA

Esperimentuak

Proiektu honen helburuetako bat, ikasleen parte-hartzea areagotzeaz gain, KEA irakasgaiko edukiak lantzeko erreminta egokia sortzea da. Bereziki, bai while- bai makro-programak landu nahi dira, ikasleei hauek exekutatzeko tresna eskainiz.

Helburu horiek bete diren ala ez bermatzeko, esperimentazio-fase bat egitea erabaki da.

Fase honetan, parte-hartzaileen balorazioa jasotzeaz gain, proiektuaren inplementazioa (aplikazioarena zein interpretearena) hobetzea bilatzen zen.

5.1 Diseinua

Proiektu osoaren (interpretearen eta aplikazioaren) balorazioa lortzeko, ikasle eta irakasle multzo baten iritziak lortu ziren. Iritzi hauekin, zenbait hobekuntza egin ziren proiektuko inplementazioan.

5.1.1 Parte-hartzaileak

Esperimentazio-fasean 10 erabiltzailek hartu zuten parte: 5 ikasle eta 5 irakasle. Irakasleak, KEA ikasgaia irakatsi duten bai gaztelerako bai euskarako irakasleak izan dira, ikasgaiarekin erlazionatutako edukien balorazio egokia eskaini dezaten. Bestalde, ikasleak 4. mailakoak izan dira, 3. mailan KEA irakasgaia aukeratu zutenak, euskaraz edo gazteleraz.

Hauek ikasleen ikuspuntutik aplikazioaren intuitibotasuna eta erabilgarritasuna baloratzeko egokiak izan dira.

Bai ikasleek eta bai irakasleek, aplikazioan zenbait akats aurkitzen lagundu dute. Gainera, erabiltzaileek proposatutako hobekuntza batzuk inplementatu dira.

Parte-hartzaileekin hitz egin eta gero, hauen email helbidea eskatu zitzaien, beraiekin kontaktuan jartzeko. Bestalde, erabiltzaileek aplikazioari buruzko zalantzak ebazteko *keaaplikazioa@gmail.com* email kontua sortu zen. Kontu honetara aplikazioan aurkitutako akatsak zein iradokizunak bidal zitzaketen.

5.1.2 Garapena

Aplikazioaren proba-fasea baloratzeko, *Google Forms*-en inkesta bat sortu zen (ikus F. eranskina).

Aplikazioa ebaluatzerako orduan honako irizpide hauek kontuan hartu dira: erabilitako hizkuntzaren egokitasuna, mugikorreko pantailarako moldagarritasuna, aplikazioaren maneiu intuitiboa, edukia KEA irakasgairako erabilgarritasuna eta edukiaren egokitasuna.

Aplikazioa proba-fasean egonda, ez zen Google Play plataforman argitaratu nahi, oraindik amaitu gabeko lana kontsideratzen zen-eta. Hortaz, aplikazioa *alpha* bertsio bezala argitaratu zen. Honi esker, aplikazioa jaisteko baimena soilik email helbidearen bitartez baimendutako erabiltzaileei eman zitzaien.

Aplikazioaren helburua eta funtzionamendua azaltzen zuen email bat (ikus F. eranskina) bidali zitzaien erabiltzaile hauei. Mezu horretan, aplikazioa jaisteko baimena eskaintzen zien esteka bidali zitzaien. Esteka horrek Google Play plataformara bidaltzen zien, KEApp aplikazioa jaisteko orrialdera hain zuzen.

Behin aplikazioaren proba-fasea amaitutzat emanda, Google Play plataformako garatzaileen orrialdetik, aplikazioa *alpha* bertsiotik *beta* bertsiora aldatzeko aukera eskaintzen da. Honek oraindik hobekuntzak gauzatzeko bertsioan dagoela esan nahi du, baina *beta* fasean edozein erabiltzailek jaitsi dezake aplikazioa.

5.2 Erabiltzaileen balorazioa

Erabiltzaileen balorazioa lortzeko, *Google Forms* bidezko inkesta egin zitzaien. Atal honetan inkesta horren emaitzekin lortutako ondorioak azaltzen dira.

5.2.1 Aplikazioaren balorazioa

Parte-hartzaile guztietatik, soilik 5 pertsonen balorazioa jaso da. Inkestan (ikus G. eranskina) gehienbat aplikazioaren interfazeari buruzko galderak daude. Erabiltzaileen erantzunetatik ondorengoak ondorioztatu dira:

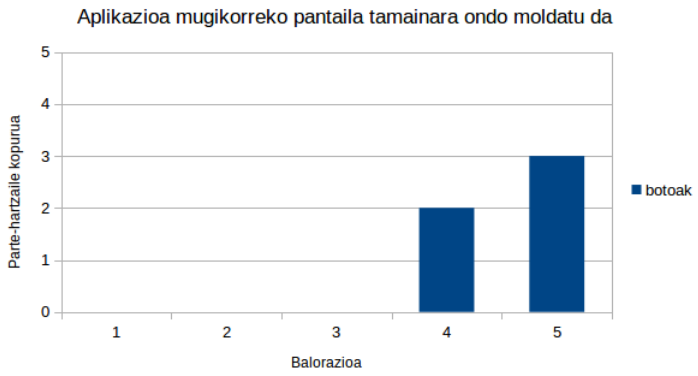
1. Aplikazioan erabilitako hizkuntza aproposa izan da. Erabiltzaile guztiek puntuazio altuenarekin baloratu zuten-eta.
2. Aplikazioa mugikorreko tamainara moldatzen zenbait arazo aurkitu ziren. Aplikazioa pantaila handiko mugikorrekin probatu zenez, hasiera batean aurkitu ez ziren interfazeko akatsak aurkitu ziren. Hala ere, lortutako balorazioa (ikus 5.1 irudia) oso positiboa izan da.
3. Aplikazioaren erabilera nahiko intuitiboa da baina hobetu daiteke (ikus 5.2 irudia). Zenbait erabiltzailek ez zuten zenbait erremintaren funtzionamendua ulertu eta *keaaplikazioa@gmail.com* email kontura idatzi zuten horiei buruz galdetuz.
4. Edukia KEA irakasgairako baliagarria da: galdera honek ere puntuazio altuena lortu zuen.
5. KEA irakasgaiaren edukiak modu egokian lantzen dira. Batek izan ezik, beste erabiltzaile guztiek puntuazio maximoarekin baloratu zuen. Beste erabiltzaileak 4-koarekin baloratu zuen. Hortaz hau ere positiboki baloratua izan den arloa da.
6. Orokorrean, aplikazioaren balorazioa positiboa izan da (ikus 5.3 irudia).

Esperimentazio fasea amaitu gabeko aplikazioaren bertsio batekin probatu dela kontuan izanda, emaitza oso positiboak lortu direla kontsideratzen da. Bertsio horretan hainbat akats eta findu beharreko atal batzuk zeuden oraindik.

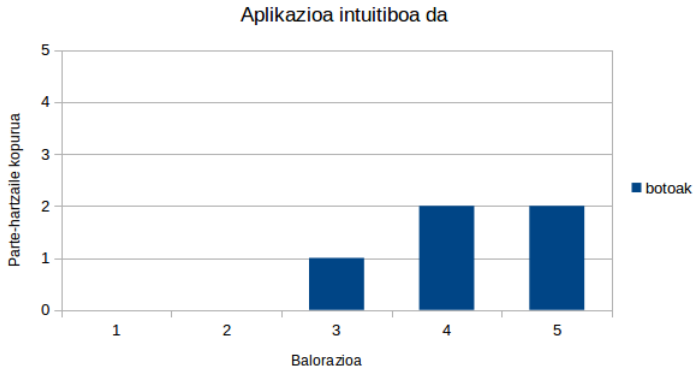
5.2.2 Hobekuntza posibleak

Aplikazioaren balorazioa nahiko positiboa izan da. *alpha* bertsioa hainbat aldiz hobetu da, eta ikasle/irakasleen iritzietatik abiatuz hainbat hobekuntza egin dira bai inplementazioan bai diseinuan.

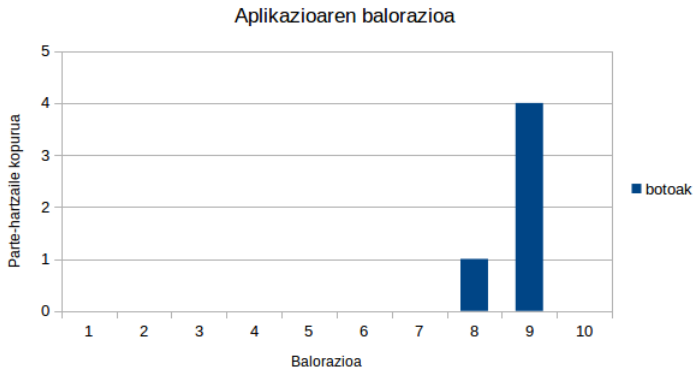
Inkestan, interfazeari buruzko galdera zehatzetaz gain, hobekuntzei buruzko iritziak jaso ziren. Hobekuntza posible horiek bai interpreteari buruz bai interfazeari buruzkoak izan ziren.



5.1 Irudia: Erabiltzaileen erantzunak, aplikazioa mugikorreko tamainara moldagarritasunari buruz



5.2 Irudia: Erabiltzaileen erantzunak, intuitibotasunari buruz



5.3 Irudia: Erabiltzaileek orokorrean aplikazioari emango lioketen balorazioa

5.2.3 Interpretea

Interpretearen erabilerari buruzko hobekuntza gutxi batzuk proposatu zituzten:

1. Hitz hutsak ezin zirela esleitu ikusi zen (hau esperotako konportamendua da).
2. While- eta makro-programek izan beharreko egiturari buruzko azalpen gehiago behar ziren. Makro-programek, while-programek ez bezala, `main` izeneko funtzioa izan behar dute nahitaez. Erabiltzaile batzuek funtzionamenduari buruzko galderak egin zituzten, funtzioa idaztea ahaztu zitzaielako.
3. Gaizki idatzitako programa bat idazterakoan, erroreen informazio gehiago eskaini.

5.2.4 Interfazea

Erabiltzaileek proposatutako hobekuntza gehienak interfazearekin erlazio zuzena zuten:

1. Bertsio zahar batean pantaila ezkutatzen zen programak idazterakoan.
2. Atzera egitean, behar ez den tokietara bueltatu daiteke, edo behin baino gehiagotan sakatu beharra dago atzera egiteko.
3. Aplikazioa atzeko planoan irekita uztean, saioa iraungi ondoren berriro irekiz gero, funtzionalitate asko ez dute funtzionatzen, eta saioa eskuz itxi eta ireki behar da.
4. Kodeketa/deskodeketarako erabilera adibideak agertzea, hauen funtzionamendua hobeto ulertzeko.
5. Idatzitako programak izen batekin gordetzeko aukera, aurrerantzean berriro idatzi behar ez izateko.
6. Aplikazioa atzeko planora eramatean, atal berdinean mantentzea. Honela programa baten zati bat beste aplikazio batetik kopiatu nahi bada, ez da berriro dena idatzi beharko.

5.3 Inplementazioan egindako hobekuntzak

Aurreko atalean, erabiltzaileek proposatutako hobekuntzen laburpena egin da. Hemen, proposatutako hobekuntzetatik inplementazioan egindako aldaketei buruz hitz egingo da.

5.3.1 Interpretea

1. While- eta makro-programak idazteko pantailan, lehenetsitako programak azalduko dira. While-programen kasuan, makro-programetan ez bezala, `main` izeneko funtziorik behar ez dela gogorarazteko balio dute.

2. Programetan aurkitutako erroreen lista ahal den zehatzena izaten saiatu da. Orain hainbat errore desberdin bereizten dira, eta beste errore berri batzuk erakustea ahalbidetu da (ikus 3.5.6 atala).

   Hala ere, honen inguruko lana zabala da, eta etorkizunean hobetu liteke, posizioaz gain zehazki zein karakterek sortu duen errorea adieraziz, eta horren ordez idatzi beharko zena adieraziz.

5.3.2 Interfazea

1. Programak idazterakoan pantaila ezkutatzeko arazoa konpondu da.

2. Galdetegi-zerrendetatik itzultzeko botoiaren funtzionamendua konpondu da. Honekin batera, atzerantz joateko botoiaren funtzionamendua berrikusi zen eta akats gehiago topatu ziren. Hauek ere, konpondu ziren.

3. Aplikazioa atzeko planora eramatean, saioa iraungiz gero, saioa hasteko pantaila erakusten da.

4. Bihurgailuan eta kodetzeko eta deskodetzeko atalean, laguntza botoi bat gehitu da. Botoi honek erreminta hauen funtzionamendua azaltzen du irudi baten bitartez, egin beharrekoa pausoz-pauso azalduz.

5. Programak gordetzeko eta kargatzeko aukera sortu da. Programak mugikorretik idaztea neketsua izan daitekeenez, behin idatzita hauek gordetzeko aukera eskaintzen da. Honela, edozein momentutan, gordetako programa kargatu besterik ez dago programa hori exekutatu ahal izateko.

6. Aplikazioa atzeko plano batera eramatean, saioa irekita mantenduz gero, idatzitako informazioa mantenduko du.

Gainera, aplikazioaren intuitibotasuna eta funtzionalitatea hobetzeko asmoz, ondorengo hobekuntzak gauzatu dira:

- Orrialde nagusian erreminta bakoitzaren laburpena agertzen da.
- Aplikazioaren diseinua mugikor tamaina bakoitzera moldatzeko aldaketak egin dira. Elementuak pantaila tamainaren arabera handitu edo txikituko dira. Gainera, mugikor txiki batzuetatik, nahiz eta pantaila tamainara moldatu, elementu batzuk txikiegi ikusten zirenez, hauen antolaketa berregin da. Bereziki programak idazteko pantailan, exekutatzeko eta gelditzeko botoiak goiko partera mugitu dira. Orain botoiak izan beharrean, menuko irudiak dira.

5.4 Aplikazioaren kudeaketa

Aplikazioa proba-fasean egonda, erroreak gertatzeko arriskua dago. Errore hauetako batzuk oso larriak badira, aplikazioa ixtea behartzen dute. Errore horiek lehenbailehen konpontzeko, email bidez notifikatzen ziren.

Alpha erabiltzaileei, aplikazioa ixtarazten dioten erroreak gertatuz gero, hauen email kontua ireki eta mezu bat sortzen zen *keaaplikazioa@gmail* helbidera zuzenduta. Mezu horretan, errorearen deskribapena zuen fitxategi bat atxikiturik bidaltzen zen, gertatutako erroreari buruzko informazioarekin (ikus 4.4.3 atala).

6. KAPITULUA

Ondorioak eta etorkizuna

6.1 Ondorioak

Proiektu honetan KEApp aplikazioa sortu da, Google Play Store plataformatik *Android* erabiltzen duen edozein mugikorretatik jaitsi daitekeena. While- eta makro-programen interpretea garatu da, eta aplikazioan integratu da.

Aplikazioak 4 erreminta nagusi eskaintzen ditu:

1. While- eta makro-programak exekutatzea. Interpreteaz baliatuz, bi programa mota hauek exekutatu daitezke, hauek sortzen duten emaitza ikusiz. Errore semantiko edo sintaktikoren bat egonez gero, programa exekutatzerakoan errorearen posizioa eta deskribapena azalduko dira.

2. *Gamification* metodologiaren elementuez baliatuz, erabiltzaile bakoitzak bere ranking-a ikusteko aukera. Erabiltzaile bakoitzak bere posizioa zein den edozein momentutan ikus dezake, eta bere posizioa publikoki erakutsiko da, baldin eta bere puntuazioa puntuazio altueneko lehen 10 postuetan badago.

3. Datu-moten arteko bihurgailua. KEA irakasgaian hainbat datu-mota desberdin lantzen dira: karaktere-segidako hitzak, zenbakizko hitzak, pilak eta bektoreak. Datu-mota batetik besterako bihurgailua sortu da. Gainera, atal honetan, irakasgaian landutako bi funtzio erabili daitezke aplikaziotik: kodeketa eta deskodeketa.

4. Galdetegiak betetzeko aukera. Galdetegiak betez, puntuazioa handitzea lortzen da, ranking-eko posizioa hobetuz.

Hasiera batean planteatu zen modura, interpreteaz baliatuz, hainbat erreminta sortu dira, erabiltzaileen arreta lortzeko. Gainera, jokoen dinamikaren bitartez puntu bidezko sistema sortu da, lehia eta parte-hartzea areagotzeko.

6.1.1 Lortutako emaitzak

Proiektuan lortu nahi ziren helburuak betetzea lortu da. Alde batetik, while- eta makro-programen interpretea gauzatzea lortu da, eta bestetik, interpretea integratzeko eta KEA ikasgaiko ikasketa bultzatzeko mugikorreko aplikazioa sortu da.

Kalitate-maila onargarriko proiektua lortu dela uste da. Erabiltzaileen iritziak jasoz, kalitatea hobetzen duten aldaketak egiteko denbora eman du. Aplikazioaren diseinua eta funtzionalitatea egokitu dira hau ahal den modu intuitiboenean erabili ahal izateko.

Etorkizunera begira, aplikazio honekin datorren kurtsoko erreminta bezala erabiltzea eta *Gamification* metodologiarekin KEA ikasgaiaren parte-hartzea areagotzea espero da.

Irismen handiko proiektua izanik, bertan egin daitezkeen hobekuntzen multzoa oso handia da. Nahiz eta proiektuan lortutako kalitatea onargarria izan, etorkizunera begira, asko hobetu liteke. Hala ere, lortutako balorazioak oso positiboak izan dira.

6.1.2 Ikasketa-pertsonala

Proiektu honekin, eskala handiko proiektuen plangintza eta kudeaketa landu da. Proposatutako lana izanik, honen irismena zenbateraino mugatu behar zen eta proiektuko ataza bakoitzak zenbateko iraupena izango zuen estimatzen ikasi da.

Proiektu honen edukia, KEA irakasgaiarekin erlazio zuzena du, eta, hortaz, irakasgai honetako hainbat kontzeptu landu dira, nagusiki while- eta makro-programak.

Programa hauek aplikaziotik exekutatu ahal izateko, interpretea gauzatu da. Interpretea gauzatzeko Konpilazioa irakasgaian ikasitako kontzeptuak beharrezkoak izan dira: analizatzaile-lexikoa, analizatzaile-sintaktikoa, gramatika, sintaxi zuhaitz abstraktuak, etab. proiektu honetan nagusitasuna izan duten kontzeptuak dira.

Bestalde, nahiz eta konputazio adarretik urrun egon, Software Ingeniaritza, Web Sistemak, eta Sare Zerbitzuak eta Aplikazioak irakasgaietako ezagutza behar izan da, aplikazioa garatzeko eta zerbitzariarekin komunikazioa modu eraginkorrean egitea ahalbidetzeko.

Teknologia berrien erabilerari buruzko ezagutza ere lortu da, bai *Antlr* liburutegiari buruz eta bai *Android* garapenari buruz.

Bestalde, garatutako proiektuaren aurkezpena nola gauzatu behar den ikasi da. Hau, esperimentuak egiterako orduan, aplikazioa probatzeko erabiltzaileekin kontaktuan jartzeko landu da, bai eta hauen balorazioa jasotzerako orduan.

6.2 Hobekuntzak eta etorkizunerako lana

Proiektu honetan landutako kontzeptu nagusiak bi izan dira, interpretea eta mugikorreko aplikazioa.

Interpreteari dagokionez, errore sintaktiko nahiz semantikoen tratamendu hobea sortu daiteke. Proiektuan, erroreen mota eta posizioa adierazten da baina errorea saihesteko edo konpontzeko gomendioak erakutsi litezke. Programa idatzi ahala, erroreak edo gomendioak emateko sistema sortu daiteke; honek aplikazioa erabiltzaileentzat ikasketa pertsonalerako erreminta intuitiboagoa bihurtuko luke.

Bestalde, makro-programen liburutegian funtzio gehiago gehitu daitezke, hala nola, *while-programa* datu-mota maneiatzeko.

Aplikazioan egin daitezkeen hobekuntza multzoa handia da.

Hasteko, gazteleraz itzuli liteke. Aplikazioa gaztelerara itzultzea erraza da, bertako `strings.xml` fitxategiko edukiak gaztelerara itzuli besterik ez dago. Bestalde, while- eta makro-programetan erabili daitezkeen funtzioek izen desberdinak dituzte, eta, hortaz, hauek ere itzuli beharko ziren. Datu-basea prest dago galdetegiak gazteleraz eta euskaraz sortzeko, datu horiek hizkuntza batean ala bestean aukeratzeko *PHP* fitxategien kodea moldatu beharko litzateke.

Eduki aldetik, erronken kopurua handitu daiteke. Erronka gehiago jarriz, erabiltzaileek kontzeptu gehiago berrikusteko aukera izango dute. Datu-basean galdetegi gehiago sortuz erraz konpondu daitekeen ataza da. Hala ere, ikasleen arreta galtzeko arriskua sortu daiteke gehiegizko erronkak sortuz gero. Edukia KEA irakasleen esku utziko da, datorren kurtsoan aplikazioa erabili nahiko balitz.

Mantentze-lana errazteko galdetegietan agertzen diren irudiak datu-basean gorde litezke. Gainera, hauetan zoom-a egiteko aukera sortu litezke, pantaila txikiko gailuetan ezin baitira irudiak tamaina egokian ikusi.

Itxurari dagokionez, animazio gehiago sortu litezke, aplikazioaren erabilera dinamikoagoa izateko. Animazio hauek erabiltzaileari aplikazioaren funtzionamendua azaltzeko erabili daitezke, *Tutorial* moduan, aplikazioan nabigatzeko eta erabiltzaileari oinarrizko tresnak aurkezteko.

Ariketa automatikoak sortzeko ideia landu daiteke. Galdetegietan programei buruzko ariketak modu automatiko batean sortu litezke. Ariketa hauetan, while-programa bat (automatikoki ala aurretik sortutakoa) zenbait hutsunerekin agertuko da eta erabiltzaileei hutsune bakoitza betetzeko hainbat aukera erakutsiko zaizkie, aukera horien artean erantzun egokia hautatu beharko luketelarik. Erantzun posible horietariko bakoitza modu automatikoan sortu daiteke, while- eta makro-programen gramatika erabiliz. Erantzunak modu koherente batean sortu litezke, esaterako, while-programa baten baldintza batean egonda, badakigu car_a moduko predikatua egon behar dela. Hortaz, horren antzeko erantzun posibleak sortu beharko lirateke, emaitza zuzena hain erraz ez antzemateko.

Kontua sortzeko eskatzen den email-a egiaztatu beharko litzateke. Hau egiteko, erregistratzerako orduan eskatzen den email helbidea aurrez idatzitako mezu bat bidali beharko litzateke, eta hor agertzen den esteka email helbidea egiaztatzerakoan soilik sortuko litzateke kontua.

Beste plataforma batzuetara hedatu liteke aplikazioa, *Android* sistemara ez mugatzeko. Beste mugikorreko sistemetarako edo ordenagailurako aplikazioa moldatuz.

6.3 Mantentze-lana

Hurrengo ikasturtean aplikazioa erabili nahiko balitz, galdetegien edukia osatzeko KEA ikasgaia irakasten duen irakasleren baten parte-hartzea beharrezkoa izango litzateke, honek edukiak kurtsoaren edukiaren arabera moldatu ahal izateko.

Bestalde, zerbitzari hobe baten beharra dago. Orain erabili den zerbitzaria, Raspberry Pi 3-a, baliabide pertsonala da eta ez da nahikoa hainbat erabiltzaileren arteko parte-hartzea kudeatzeko, ezta segurtasuna bermatzeko ere.

Mantentze-lanari dagokionez, behin kurtsoa bukatuta, aurreko kurtsoko erabiltzaileen kontuak ezabatu daitezke. Hau datu-basetik zuzenean egin daiteke. Ez da beharrezkoa hau

egitea, baina erabiltzaile-izen errepikaturik edo email kontu berdinik ezin denez sortu, kontuan hartzekoa da. Hala ere, ikasleek ikasgaia amaitu ondoren, aplikazioa erabili nahiko balute, hobe izango litzateke beraien kontuak ez ezabatzea. Azken finean, *RGPD* legearekin bat etortzeko, erabiltzaileek beren kontuak ezabatzea ahalbidetu da.

Eranskinak

A. ERANSKINA

While-programen gramatika

```
grammar While;

prog:   stmts+=statement*  EOF ;

statement: 'if' bald=CAR_X '?' '(' ald=ALDAGAIA ')' 'then' stmts+=statement+ 'end if' ';'  # if
         | 'while' 'nonem?' '(' ald=ALDAGAIA ')' 'loop' stmts+=statement+ 'end loop' ';'# while
         | ald=ALDAGAIA ':=' ad=adierazpena ';' # esleipen
         ;

adierazpena: 'hutsa'                          # hutsa
           | 'cdr' '(' ald=ALDAGAIA ')'       # cdr
           | cons=CONS_X '(' ald=ALDAGAIA ')' # cons
           ;

ALDAGAIA: 'X' [0-9]+ ;
CAR_X: 'car_' [a-zA-Z];
CONS_X: 'cons_' [a-zA-Z];
COMMENT: '--' [^\n\r]* -> skip;
WS : [ \t\r\n]+ -> skip;
```

B. ERANSKINA

Makro-programen gramatika

```
grammar Makroprograma;

prog: funtz+=func+ EOF;

func: 'def' ald=ALDAGAIA '?'? 'begin' stmts+=statement* 'end def' ';';

statement: 'if' bald=adierazpena 'then' stmts+=statement* elif+=elsif* ('else' falseStmts+=
    statement*)? 'end if' ';'  # if
        | 'while' bald=adierazpena 'loop' stmts+=statement* 'end loop' ';'
                        # while
        | 'for' ind=ALDAGAIA 'in' has=adierazpena  '..' buk=adierazpena 'loop' stmts+=statement*
     'end loop' ';'      # for
        | 'for' ind=ALDAGAIA 'in' has=ZENB '..' buk=ZENB 'loop' stmts+=statement* 'end loop' ';'
                        # forZenbaki
        | 'case' ad=adierazpena 'is' w+=when+ ('when' 'others' '=>' otherStmts+=statement+)? '
    end case' ';'          # caseMakroa
        | ald=ALDAGAIA ':=' ad=adierazpena ';'
                        # esleipen
        ;

elsif: 'elsif' bald=adierazpena 'then' stmts+=statement*;
when: 'when' ad=adierazpena '=>' stmts+=statement+;

adierazpena: ('hutsa' | 'phutsa')                                                  # hutsa
           | cons=CONSX '(' ad=adierazpena ')'                                     # cons
           | car=CARX '?' '(' ad=adierazpena ')'                                   # car
           | deskod=DESKOD '(' ad=adierazpena ')'                                  # deskod
           | hitz=HITZA                                                            # hitza
           | ald=ALDAGAIA                                                          # aldagaia
           | ald=ALDAGAIA '?'? '(' (arg+=adierazpena (',' arg+=adierazpena)* )? ')' # funtzioDeia
           | ad1=adierazpena erag='*' ad2=adierazpena                              # alderaketak
```

```
26            | ad1=adierazpena erag=('+'|'-') ad2=adierazpena                          # alderaketak
27            | ad1=adierazpena erag=('='|'>='|'<='|'>'|'<'|'/=') ad2=adierazpena      # alderaketak
28            | ad1=adierazpena erag='and' ad2=adierazpena                            # alderaketak
29            | ad1=adierazpena erag='or' ad2=adierazpena                             # alderaketak
30            | 'not' ad=adierazpena                                                # notAdierazpena
31            ;
32
33  HITZA: '"' [a-zA-Z]* '"';
34  CONSX: 'cons_' [a-z] ;
35  CARX: 'car_' [a-zA-Z] ;
36  DESKOD: 'deskod_' [0-9]+ '_' [0-9]+ ;
37  ALDAGAIA: [a-zA-Z][a-zA-Z0-9_]* ;
38  ZENB: [0-9]+ ;
39  COMMENT: '--'~[\n\r]* -> skip;
40  WS : [ \t\r\n]+ -> skip;
```

C. ERANSKINA

Programen adibideak

Interpreteak eskaintzen dituen funtzioen funtzionamendua adibideen bitartez azalduko da. Programen erabilera ulertzeko asmoz sortutako programa hauen portaera aztertu da, funtzioen funtzionamendu zuzena bermatzearekin batera. Adibide zehatz hauetan $\Sigma = a, b$ erabili da.

C.1 Kodeketa eta deskodeketa

```
1  def deskodKateatu begin
2      X2:=deskod_2_1(X1);
3      X3:=deskod_2_2(X1);
4      X0:=kateatu(X2, X3);
5  end def;
6
7  def main begin
8      X0:=deskodKateatu(kod("a", "bb"));
9  end def;
```

Programa honen emaitza “abb” karaktere-segida da. Kodeketa, deskodeketa eta kateaketaren funtzionamendua egokia dela egiaztatzeko balio du.

C.2 For

```
def main begin
    for i in "a".."ab" loop
        if car_a?(i) then
            X0:=cons_a(X0);
        else
            X0:=cons_b(X0);
        end if;
    end loop;
end def;
```

Programa honen emaitza “aba” karaktere-segida da. Programak [“a”, “ab”) tarteko elementuen lehen karakterea itzuliko du hitz bakar batean. Adibide honetan, (“a”, “b”, “aa”) elementuak iteratzen dira, eta hortaz horien lehen karakterea, azken elementutik lehenengora irakurrita, itzultzen duen hitza emaitzaren berdina dela ikusi daiteke. Programa honekin `for` makroaren funtzionamendua egiaztatzeaz ez ezik, `if` eta `cons`-en funtzionamendua egiaztatu da.

C.3 Switch

```
def main begin
    X1:="bb";
    case X1 is
        when "a"=> X0:=cons_a(X0);
        when "b"=> X0:=cons_b(X0);
        when others => X0:="ab";
    end case;
end def;
```

Programa honek a karakterea itzuliko du, $X1 =$ ”a” bada, b karakterea $X1 =$ ”b” denean eta bestela “ab”. Adibide zehatz honetan, $X1 =$ ”bb” denez, “ab” itzuliko du.

C.4 Hur eta aurre

```
def main begin
    hitza := kateatu("bb", "b");
    if car_a?(hitza) then
```

```
4           X0:=hur(hitza);
5       else
6           X0:=aurre(hitza);
7       end if;
8   end def;
```

Programa honetan, hasierako hitza *b* karakterez hasiz gero, hitzaren aurreko elementua itzuliko du, eta *a* karakterez hasiz gero hitzaren hurrengo elementua. Adibide zehatz honetan "bba" itzuliko du eta hitzaren balioa "abb" karaktere-segidera aldatuz gero "baa".

C.5 Eragiketa aritmetikoak

C.5.1 Batuketa

```
1   def main begin
2       X0:=hutsa;
3       for i in 1..9 loop
4           X0:=X0+i;
5       end loop;
6   end def;
```

Programaren emaitza "aabab" karaktere-segida da, hau zenbakira bihurtuz gero 36 dela ikusi dezakegu; hau da, 1-etik 8-ra arteko zenbakien segidaren batuketaren balio bera ($1+2+\cdots+8=36$). `For` makroaren azken bukaerako elementua ez da tratatuko.

Proba-programa honekin `hutsa` konstantea eta `for` makroak probatu ez ezik, eragiketa aritmetikoak probatu dira.

C.5.2 Kenketa

Kenketa eragiketan, kontuan izan behar da zenbaki negatiborik ez direla sortuko. Hortaz bi programa hauek 0 itzuliko dute:

```
1   def main begin
2       X0:="b"-"b";
3   end def;
4   def main begin
5       X0:="b"-"bbb";
6   end def;
```

Azkeneko honek, ordea, 1 itzuliko du:

```
def main begin
    X0:="b"-"a";
end def;
```

C.5.3 Biderketa

Biderketak, kenketa eta batuketa eragiketekiko lehentasuna du:

```
def main begin
    X0:="a"+ "b" * "b";
end def;
```

Programa honetan, lehenengo biderketa eta gero batuketa eginez, “bb” hitza lortuko dugu, hau 6 da zenbakizko hitzera bihurtuz. Batuketa eta ondoren biderketa eginez gero, 5 lortuko genuke.

C.6 Pilak

```
def nireTontorra begin
    X0:=deskod_2_1(X1);
end def;

def main begin
    pila:=pilaratu("ab", pilaratu("b", pilaratu("a", phutsa)));
    if nireTontorra(pila)=tontorra(pila) then
        X0:="a";
    else
        X0:="";
    end if;
    if not p_hutsa_da?(X0) then
            X0:="b";
        else
            X0:="bbb";
    end if;
end def;
```

Pila datu-motetan 4 predikatu desberdin erabiltzen dira: `phutsa`, pila hutsa sortzeko, `pilaratu`, elementuak pilan gehitzeko, `despilaratu`, elemetuak piletik kentzeko, eta `tontorra`, pilan sartu den azken elementua lortzeko.

Adibide zehatz honetan, berdinketa eragiketaz baliatuz, `nireTontorra` funtzioa `tontorra`-ren emaitza bera itzultzen duela erakutsi da. <ab, b, a] pila sortu da eta elementuaren tontorra itzultzen duen funtzioa sortu da. Bi funtzioek pilaren tontorra “ab” elementua dela esaten badute, $X0 = "a$” idatziko dute, eta ondoren, `p_hutsa_da?` funtzioa erabiliz, X0 hutsa ez denez, “b” karakterea itzuliko du.

C.7 Bektoreak

```
1  def main begin
2      X0:=lehen_bektore();        --bektore hutsa sortzeko
3      X0:=aldatu(X0, hutsa, "ab");--"ab" elementua lehen posizioan gehitzeko
4      X0:=atzitu(X0, "");         --lehen posizioaren elementua lortzeko
5  end def;
```

Bektore datu-motekin aritzeko 3 predikatu behar dira: `lehen_bektore`, bektore hutsa sortzeko, `atzitu`, bektorea eta indizea pasatuz, posizio horretako indizea itzultzeko, eta `aldatu`, bektorea, atzitu nahi den elementuaren indizea eta elementu berria pasatuz, indize horretan elementu berria aldatzen du. Posizio horretan elementurik egongo ez balitz, bektorea handituko litzateke (hitz hutsak sartuz), elementu hori sartu arte.

Adibide honetan, bektore huts bat sortu da eta 0.posizioan “ab” karaktere-segida sartu da. Ondoren, posizio horretako elementua itzuli da, funtzioen portaera zuzena bermatzeko.

C.8 Funtzioak

```
1  def reverse begin
2      while nonem?(X1) loop
3          if car_a?(X1) then
4              X0:=cons_a(X0);
5          else
6              X0:=cons_b(X0);
7          end if;
8          X1:=cdr(X1);
9      end loop;
10 end def;
11
12 def main begin
13     X0:=reverse("abb");
14 end def;
```

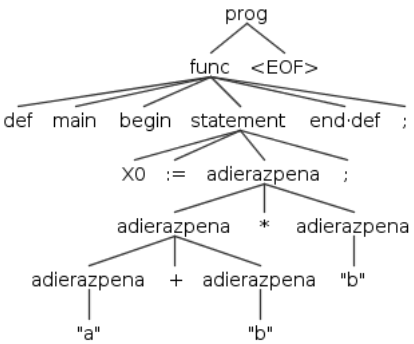
Funtzioak sortzeko, programa nagusia egon behar duela kontuan izan behar da. Kasu honetan, programa nagusitik alderantzizkoa kalkulatzen duen funtzioari deitzen zaio “abb” karaktere-segidarekin, hortaz, honek itzuliko duen emaitza “bba” da.

D. ERANSKINA

Sintaxi zuhaitz abstraktuak

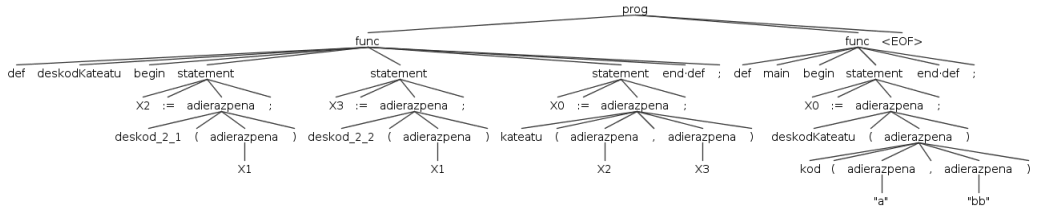
Sintaxi Zuhaitz Abstraktua (SZA), programen kodearen egitura adierazteko metodoa da. *Antlr* analizatzaile sintaktikoak sortutako zuhaitzak dira, C.eranskinean erakutsitako programetatik abiatuz.

Eragiketa aritmetikoak



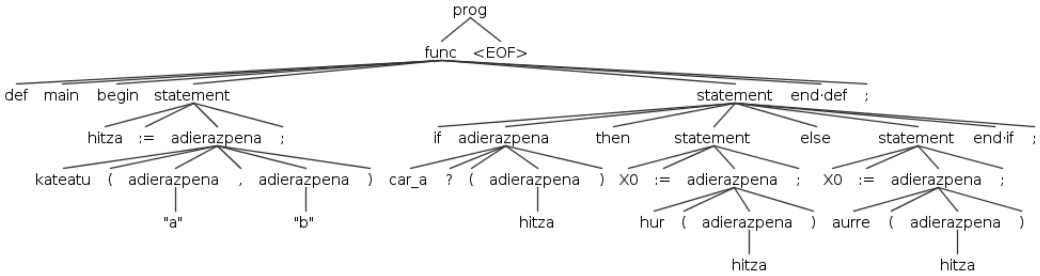
D.1 Irudia: Batuketa eragiketaren sintaxi zuhaitza

Kodetu eta deskodetu



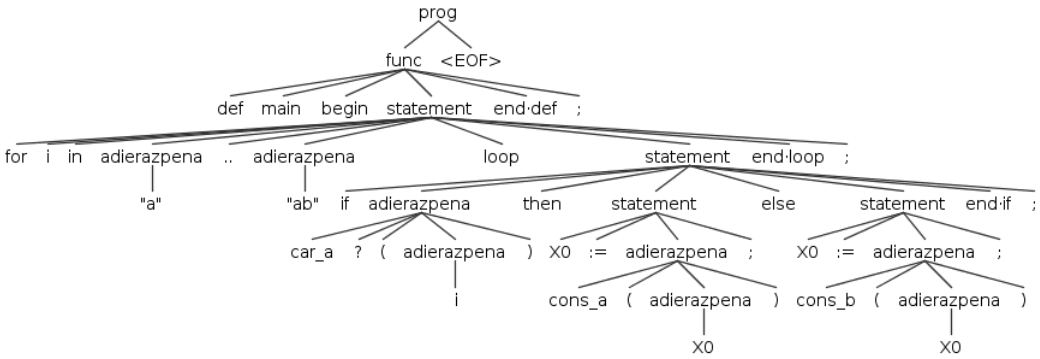
D.2 Irudia: Kodetu eta deskodetu funtzioen sintaxi zuhaitza

Hur eta aurre



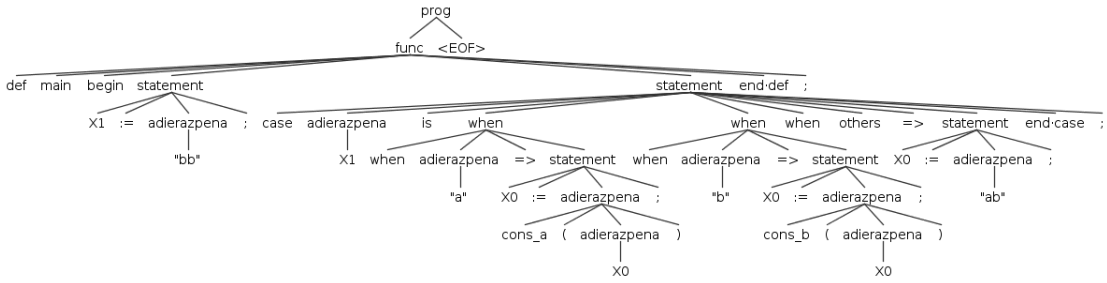
D.3 Irudia: "Hur"eta "aurre"funtzioen sintaxi zuhaitza

For



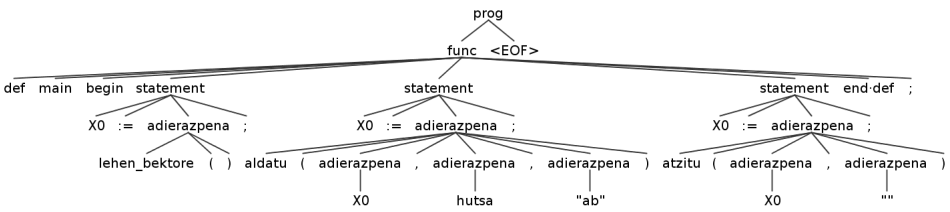
D.4 Irudia: For makroaren sintaxi zuhaitza

Switch



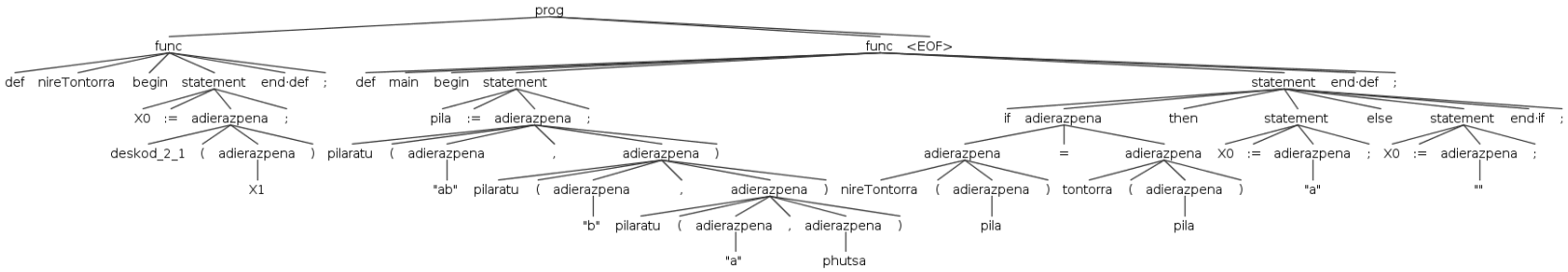
D.5 Irudia: Switch makroaren sintaxi zuhaitza

Bektoreak



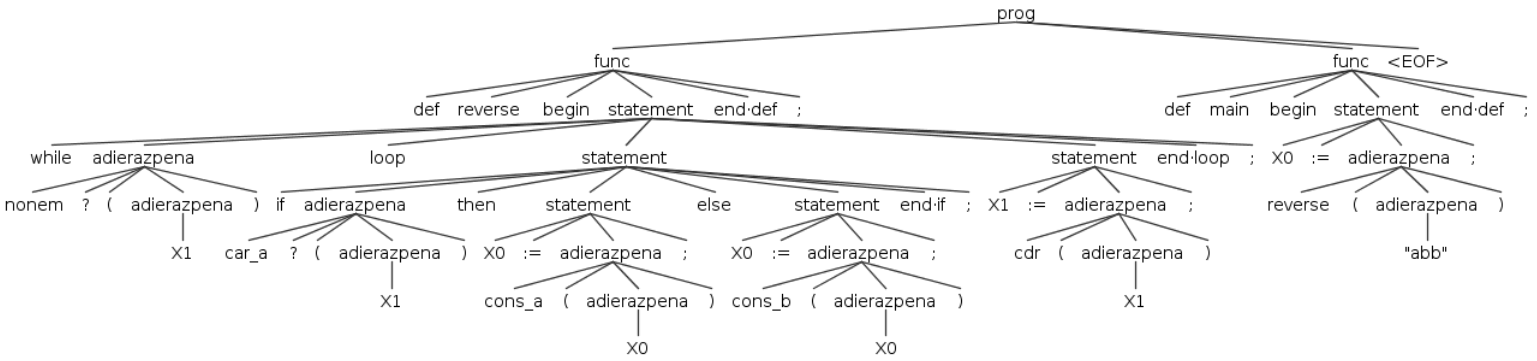
D.6 Irudia: Bektoreen sintaxi zuhaitza

Pilak



D.7 Irudia: Pilen eragiketen sintaxi zuhaitza

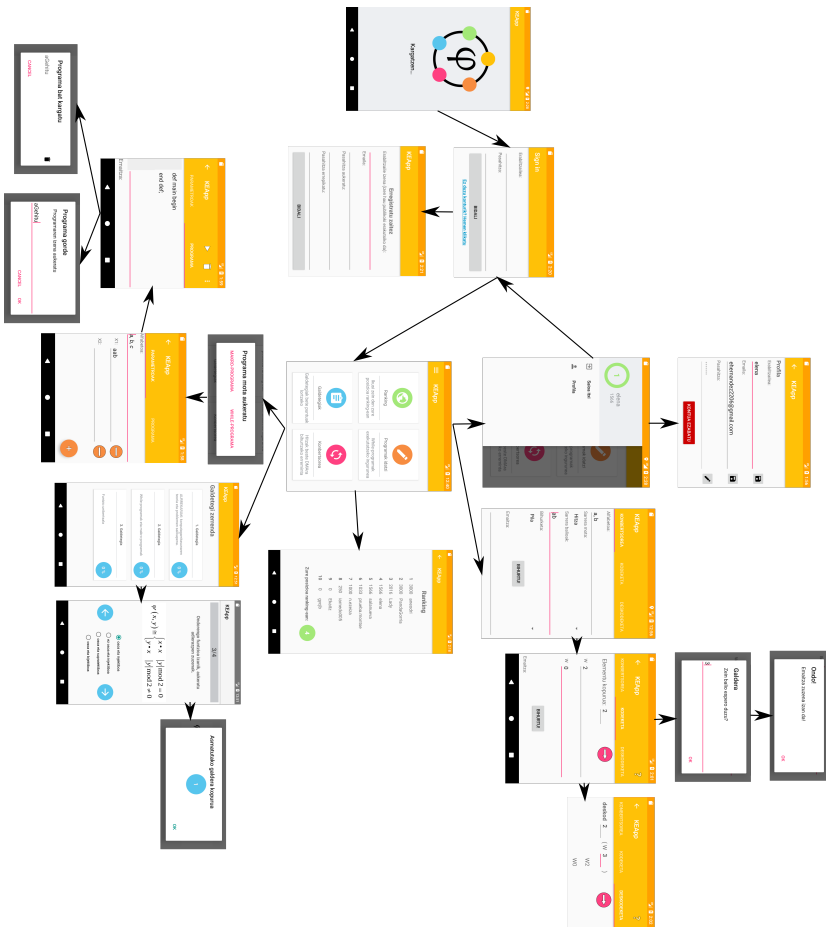
Funtzioak



D.8 Irudia: Alderantzizkoa kalkulatzen duen funtzioaren sintaxi zuhaitza

E. ERANSKINA

Aplikazioaren nabigazio-eskema



E.1 Irudia: KEApp aplikazioaren nabigazio-eskema

F. ERANSKINA

Inkesta eta lortutako emaitzak

Aplikazioa ebaluatzeko inkesta betetzeko eskatu zitzaien esperimentazio-fasea osatu zuten erabiltzaileei. Inkesta, gazteleraz [1] eta euskaraz[2] egin zen.

KEApp aplikazioa probatu zuten erabiltzaileek, ondorengo galderez osatutako formularioa bete zuten:

- Erabilitako hizkuntza egokia da (1-etik 5-erako balorazioa)
- Aplikazioa mugikorreko pantaila tamainara ondo moldatu da (1-etik 5-erako balorazioa)
- Aplikazioa intuitiboa (1-etik 5-erako balorazioa)
- Edukia KEA irakasgairako baliagarria da (1-etik 5-erako balorazioa)
- KEA irakasgaiaren edukiak modu egokian lantzen ditu (1-etik 5-erako balorazioa)
- Aurkitutako akatsak
- Hobekuntza posibleak
- Orokorrean, aplikazioaren balorazioa (1-etik 10-erako balorazioa)

[1] https://goo.gl/forms/dO2uOM1HwsOMlStA3
[2] https://goo.gl/forms/uuUyP6HLKMAWtzia2

F.1 Email-a

Aplikazioaren helburua eta funtzionamendua azaltzeko, ondorengo mezua bidali zitzaien, bai euskaraz [3], bai gazteleraz [4].

Mezuan aplikazioa jaisteko agertzen den esteka, soilik email baimenduentzako funtzionatuko du.

Aplikazioa probatu zuten erabiltzaileei bidalitako email-a euskaraz honakoa da:

> Kaixo,
>
> Mezu hau jaso baduzu, nire GRAL-erako sortzen ari naizen aplikazioaren alpha tester bat zarela esan nahi du. Alpha tester-rak aplikazioan akatsak bilatzen eta hobekuntzak proposatzen dituzten pertsonen multzo mugatua da. Guztien iritziekin hobekuntzak gauzatuko dira aplikazioaren behin-behineko bertsioa atera aurretik.
>
> Aplikazioa KEApp izena du eta hirugarren mailan erakusten den “Konputazio Eredu Abstraktuen” (KEA) edukiak lantzeko balio du. Irakasgaian problema konputagarriak eta konputaezinak bereizten erakusten da, horretarako while-programak erabiliz. Programa hauek Turing makinaren baliokideak dira eta honek gaur egunera arte ezagutzen den kapatizitate konputazional handiena du. Horregatik, problemaren bat ezin bada Turing makinaren bidez, edo baliokidea den while-programaren bidez, ebatzi, problema konputaezina dela diogu.
>
> Oraingoz makro-programak soilik exekutatu daitezke. Hauek while-programen bertsio hedatua dira, non goi-mailako egiturak inplementaturik dauden.
>
> While-programen sintaxi eta semantikari buruzko informazio gehiago:
>
> Bertsio laburra:
>
> https://drive.google.com/open?id=1S08Pjy8BwJwoTS2AY_WI8zbbkl6t9KWl
>
> Bertsio luzea:

[3] https://docs.google.com/document/d/14e-k6OVuKZUqs5VlcttsFRt9n6ynKTYSk4l2ciy58aE/edit?usp=sharing

[4] https://docs.google.com/document/d/1RaOKJbV4q0FL6B6-17zrF97SN5fdMe76BsCFZKe4gEQ/edit?usp=sharing

https://drive.google.com/open?id=1M8BwE_iToVSQAJB5CkbJR_yrV_nRAikh

Aplikazioak bi helburu nagusi ditu: alde batetik while-programak exekutatzeko sortu den interpretea erakustea, eta bestetik, KEA irakasgaiko ikasketa errazteko baliabidea izateko.

Mugikorrerako aplikazioa deskargatzeko esteka:

https://play.google.com/apps/testing/es.ehu.ehernandez035.kea

Aplikazioaren helburuak argituta eta aplikazioa ikusita, hau baloratzeko formularioa:

https://docs.google.com/forms/d/1wCy24A9Qh2P_JF7g9hwUFG4y0B6huts6eJUT3wXm9VU/edit

OHAR GARRANTZITSUAK:

Aplikazioaren gaztelerazko itzulpena garatzen ari da eta bertsio berri batean gehituko da.

Aplikazioak espero gabeko erroreren bat itzuliko balu, mezu bat bidaltzeko prest egongo da. Mezu horretan errorea eta errorea sorrarazi duenari buruzko informazioa atxikituko da, informazio honek errorea konpontzen lagunduko du. Hortaz, mezua aldaketarik gabe bidaltzea eskertuko da.

Elena Hernandez

keaaplikazioa@gmail.com

G. ERANSKINA

Bilera-aktak

G.1 2017/06/20

Ordua: 15:00 - 15:30

Bertaratuak: Montse Maritxalar, Elena Hernandez

Bertaratze ordua: 15:00

Gai-zerrenda:

1. Proiektua definitu

Gai bakoitzari buruz hartutako erabakiak:

1. Proiektuaren lehen proposamena gauzatu zen, proiektua garatzeko ideia nagusiak bilduz. Bertan landuko diren atal garrantzitsuei buruz hitz egin; hala nola, while-programen interpretea gauzatuko dela, mugikorreko aplikazioa garatu nahi dela (bai ikasleei, bai irakasleei zuzenduta dagoena) eta KEA irakasgaiari buruzko ezagutza-maila jakiteko eta ikasketak sustatzeko galdetegiak gauzatuko direla erabaki zen.

Hartutako eginkizunak:

Proiektuaren irismena zehaztu eta planifikazioa gauzatu.

G.2 2018/02/05

Ordua: 15:20 - 16:30

Bertaratuak: Montse Maritxalar, Elena Hernandez

Bertaratze ordua: 15:20

Gai-zerrenda:

1. Proiektuaren irismena murriztu
2. Atazak definitu
3. Planifikazioa zehaztu

Gai bakoitzari buruz hartutako erabakiak:

1. Proiektuko irismenaren inguruko hausnarketa egin da. Proiektuan gauzatu daitezkeen ideiak eztabaidatu eta murriztu egin dira proiektua entrega eperako bideragarria izateko. Horretarako, proiektuan gauzatu behar diren atazak definitu eta lehentasunmailak zehaztu dira, garrantziaren arabera.
2. Proiektua osatzen duten ataza nagusiak bereizi dira. Ataza horiek 3 dira: while-programen interpretea, mugikorreko aplikazioa eta ariketak automatikoki sortzeko sistema.

   While-programen interpreteak, hiru aukera eskainiko ditu: while-programa puruen interpretea, makroprogramena non bestelako funtzioak(hur(), aurre(), ...) inplementaturik dituen eta makroprogramena non Datu Mota Abstraktuak sortzeko funtzioak (pilaratu(), tontorra(), aldatu() ,...) inplementaturik dauden.

   Mugikorreko aplikazioan, Gamificaction moduan erronkak egongo dira. Erabiltzaileak aplikazioan izango duen atzipen eremua eta ariketen zailtasuna handituz joango da erronkak bete ahala.

   Ariketa automatikoak mota desberdinekoak izango dira. Hauetako batzuk, while-programetan erabiltzen diren funtzioak izango dira, non “ausaz” (aurretik definitutako eremuetan, adibidez, baldintza eremuetan) programaren zati bat ezabaturik agertuko den. Hutsuneak betetzeko, aukerak eskeiniko zaizkio (ausaz sortuak, baina koherentzia maila batekin) edo aukerik sortu gabe, hutsuneak betetzeko eskatuko zaio.

3. Planifikazioaren garapena zehaztu da, egin beharreko atazak definituz eta proiektuko zenbait xehetasun konkretatuz.

   Hala nola, ariketa automatikoak hainbat motatakoak, aurretik definitutakoaz gain, izango direla denbora izanez gero. Gainera, irakasle eta ikasleen laguntza eskatuko dela aplikazioaren proba egiteko eta hobekuntzak iradokitzeko.

   Aplikazioaren diseinua ahalik eta dinamikoen izaten saiatuko da, posible bada animazioak erabiliz, bisualki erakargarriagoa izateko.

Hartutako eginkizunak:

Bileren aktak gauzatuko dira eta planifikazioa berrantolatuko da bilera honetan zehaztutako atazen arabera.

G.3 2018/03/21

Ordua: 9:30 - 10:30

Bertaratuak: Montse Maritxalar, Elena Hernandez

Bertaratze ordua: 9:30

Gai-zerrenda:

1. Aplikazioaren garapena
2. Aplikazioaren proba fasea

Gai bakoitzari buruz hartutako erabakiak:

1. Aplikazioaren diseinua berrantolatu, *Gamification* kontzeptuari garrantzi handiagoa emanez.
2. Aplikazioa maiatzerako aurkezteko moduan egotea, ikasleren batzuek probatu ahal izateko. Proba horien ondoren beraien iruzkinak kontuan hartuko dira aplikazioan hobekuntzak garatzeko.

Hartutako eginkizunak:

Gamification-i buruz informazioa bilatu.

G.4 2018/03/26

Ordua: 12:30 - 13:00

Bertaratuak: Montse Maritxalar, Elena Hernandez

Bertaratze ordua: 12:30

Gai-zerrenda:

1. Memoria

Gai bakoitzari buruz hartutako erabakiak:

1. Memoriaren egitura zehaztu da: bi zati nagusitan banatzea erabaki da, lehenengo zatia, while-programen interpretearena eta bigarrena aplikazioarena izango dira.

   Hala ere, plangintzan bi atalak batera azalduko dira.

Hartutako eginkizunak:

Memoriaren egitura egin eta interpretearen atala garatzen hasi.

G.5 2018/04/07

Ordua: 11:30-12:00

Bertaratuak: Montse Maritxalar, Elena Hernandez

Bertaratze ordua: 11:30

Gai-zerrenda:

1. Aplikazioa Google Play Storera igotzea
2. Memoria

Gai bakoitzari buruz hartutako erabakiak:

1. Aplikazioa ikasle eta irakasleek probatzeko Google Play Storen igoko da. Hau datorren asterako prest egongo da KEA ikasgaia kurtso honetan kurtsatu duten ikasleek proba dezaten.

2. Memoriari buruzko zuzenketak.

Hartutako eginkizunak:

Memoriaren atal batzuk berrantolatu eta aplikazioa Play Storen igotzea datorren asterako. Ikasle eta irakasleek aplikazioa baloratzeko galdetegia sortu.

G.6 2018/05/14

Ordua: 11:00-11:30

Bertaratuak: Montse Maritxalar, Elena Hernandez

Bertaratze ordua: 11:00

Gai-zerrenda:

1. Aplikazioaren berrikuspena
2. Galdetegiak

Gai bakoitzari buruz hartutako erabakiak:

1. Aplikazioa ikasle eta irakasleek probatzeko Google Play Storen igo aurretik, honen berrikuspena egin da irakaslearen iritzia jasoz hobekuntzak gauzatzeko.
2. Galdetegien edukia osatzeko, KEA irakasgaia datorren kurtsoan emango duen irakaslearekin hitz egingo da.

Hartutako eginkizunak:

Aplikazioan hobekuntzak gauzatuko dira eta bihar Google Play plataformara igoko da. Alpha fasean egonda, irakasle batzuekin partekatuko da hauen iritziak eta hobekuntzak jasotzeko. Ondoren, hirugarren eta laugarren mailako ikasleekin partekatuko da hauen iritzia ere jasotzeko.

G.7 2018/06/12

Ordua: 16:00-16:45

Bertaratuak: Montse Maritxalar, Elena Hernandez

Bertaratze ordua: 16:00

Gai-zerrenda:

1. Memoriaren berrikuspena

Gai bakoitzari buruz hartutako erabakiak:

1. Esperimentazio-faseari indar handiagoa emango zaio dokumentuan. Erroreen-tratamenduari buruzko sekzio bat sortuko da aplikazioaren garapenari buruzko atalean. Eranskin bezala aplikazioaren balorazioari buruzko inkesta atxikituko da. Memoriaren egitura berdiseinatuko da.

Hartutako eginkizunak:

Memoriaren bertsio berria aste honetako ostiralerako entregatuko da, hau datorren astelehenerako bileran errepasatu ahal izateko.

G.8 2018/06/18

Ordua: 13:00-13:30

Bertaratuak: Montse Maritxalar, Elena Hernandez

Bertaratze ordua: 13:00

Gai-zerrenda:

1. Memoriaren berrikuspena

Gai bakoitzari buruz hartutako erabakiak:

1. Memoriaren azken xehetasunak konpontzeko bilera izan da. Egitura berrordenatzeko eta zenbait akats ortografiko konpontzeko.

Hartutako eginkizunak:

Memoriaren bertsio berria lehenbailehen idazteko.

Bibliografia

[Dobson, 2013] Dobson, J. L. (2013). Retrieval practice is an efficient method of enhancing the retention of anatomy and physiology information. *Advances in physiology education*, 37(2):184–191.

[EHU, 2018] EHU (2018). *While-programen informazio gehiago*. https://ocw.ehu.eus/course/view.php?id=8.

[Google, 2008] Google (2008). *Google Play Console: Android Developers*. https://developer.android.com/distribute/console/.

[Hernandez, 2018a] Hernandez, E. (2018a). *Interpretearen kodea*. https://github.com/ehernandez035/KEApp-Interpreter.

[Hernandez, 2018b] Hernandez, E. (2018b). *KEApp zerbitzari aldeko kodea*. https://github.com/ehernandez035/KEApp.

[Hernandez, 2018c] Hernandez, E. (2018c). *KEApp zerbitzari aldeko kodea*. https://github.com/ehernandez035/KEApp-Server.

[Huang and Soman, 2013] Huang, W. H.-Y. and Soman, D. (2013). Gamification of education. *Research Report Series: Behavioural Economics in Action, Rotman School of Management, University of Toronto*.

[Jesús Ibáñez, 1998] Jesús Ibáñez, Arantza Irastorza, A. S. (1998). While-programak: Konputagarritasun teoria oinarritzeko tresna. *ADDI*.

[JetBrains, 2009a] JetBrains (2009a). *IntellIJ IDEA: The Java IDE for Professional Developers*. https://www.jetbrains.com/idea/.

[JetBrains, 2009b] JetBrains (2009b). *PhpStorm: Lightning-smart PHP IDE*. https://www.jetbrains.com/phpstorm/.

[Kiryakova et al., 2014] Kiryakova, G., Angelova, N., and Yordanova, L. (2014). Gamification in education.

[Parr, 2013] Parr, T. (2013). *Antlr documentation*. http://www.antlr.org/.