eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

Kudeaketa eta informazio sistemen informatika ingeniaritzako gradua

Gradu Amaierako Lana

GanttProject, proiektuak kudeatzeko aplikaziorako akatsen konponketa eta funtzionalitate berrien garapena

Urtzi Puente Gómez

Juanan Pereira Varelak
zuzendua

Bilbo, 2020ko uztailaren 3a

Laburpena

Proiektu honen bidez jada sortuta dagoen kode irekiko aplikazio batekin lan egin nahi da. Horretarako GanttProject aplikazioa aukeratu da, kode irekiko proiektuak kudeatzeko aplikazioa. Proiektua aurrera eramateko hiru ataletan banatu da:

Lehenengo atala, aplikazioarentzako funtzionalitate berria garatzea eta aplikazioaren garatzaileek onartuta aplikazioan ezartzea.

Bigarren atala, aplikazioaren garatzaileek edo erabiltzaileek aurkitu duten arazo edo akatsen bat identifikatzea, arazoa aztertzea eta konponketa inplementatzea.

Hirugarren atala, kodearen refaktorizazioa egitea, erabiltzen ez diren klase eta metodoak aurkitzea eta kodetik kentzea.

GanttProject Java programazio lengoaian eginda dago gehienbat eta Swing eta JavaFX liburutegiak erabiltzen ditu, hala ere, Kotlin eta HTML lengoaiak ere erabiltzen ditu. Horrez gain, aplikazioa muntatzeko Gradle erreminta erabiltzen da, aplikazioak eraikitzeko automatizazio-sistema.

Proiektu honen garapenerako hautatu diren tresna, modulu eta liburutegi guztiak kode irekikoak dira.

Aurkibidea

Irudien aurkibidea

Taulen aurkibidea

1. Sarrera

Gaur egun, gero eta gehiago dira aholkularitzaz eta informazio sistemen kudeaketaz arduratzen diren enpresak. Izan ere, erakunde batek entzute handikoa izan nahi badu, interneten eta ordenagailu edo mugikorreko aplikazioetan presentzia izatea beharrezkoa da.

Honekin jarraituz, gehienetan erakunde hauek ez dira beraien aplikazio edo web-orrien kudeaketaz arduratzen, baizik eta beste erakunde bati esleitzen diote lan hori, aholkularitza edo informazio sistemen kudeaketarako erakundeak, hain zuzen ere.

Adibidez, Bilboko Iberdrola dorrean kokaturiko Accenture enpresak Iberdrolaren web-orri eta mugikor eta tableten aplikazioaren kudeaketaz arduratzen da. Bertan aplikazioak edo web-orriak izan ditzakeen akatsak konpontzea eta Iberdrolak eskatutako funtzionalitate berriak garatzea dute helburu.

Era berean, smartphone,tablet edo ordenagailu bat erabiltzen duten pertsona kopurua urtero goraka doa, gailu hauentzako egokituta dauden aplikazioen beharra sustatuz.

Hori guztia kontuan hartuta proiektu hau aurrera eramatea erabaki da. Gradu Amaierako Lan(GAL) honen helburua GanttProject aplikazioarekin lan egitea da, zehazki, aplikazioak dituen akatsen bat konpontzea, funtzionalitate berria aurkitzea eta garatzea eta kode refaktorizazioa egitea lortu nahi da.

1.1. Proiektuaren jatorria

Proiektu honen proposamena Juanan Pereirarena izan zen, Euskal Herriko Unibertsitateko (UPV/EHU) irakaslea. Web aplikazioekin eta informazio sistemekin zerikusia duten irakasgaiak lantzen ditu, beste askoren artean.

Juananekin bere bulegoan GAL proposamenak ikusten eta ebaluatzen geundela, bi aukera nagusi aurkitu genituen.

Alde batetik, GAL-ekin eta hauen dokumentazioarekin lotutako Chatbot batekin lan egitea. Chatbot honek GAL baten memoriaren inguruko laguntza eskaintzen zuen, hala nola, memoriaren ebaluazioa egitea.

Beste aldetik, kode irekiko aplikazio batekin lan egitea. Aukera honekin, aplikazioa aztertu eta dituen akatsak konpontzeko edo funtzionalitate berriak garatzeko aukerak genituen.

Gauzak horrela, bigarren aukerarekin lan egiten hastea erabaki nuen, eta horretarako, hainbat aplikazioen artean GanttProject hautatu genuen. Horrez gain, proiektua aurretik aipatu den hiru zatitan banatzea adostu genuen: akatsen konponketa, funtzionalitate baten garapena eta kode refaktorizazioa.

1.2. Proiektuaren deskribapen eta kokapena

GanttProject [1] proiektuak programatzeko, planifikatzeko eta kudeatzeko mahaigaineko eta kode irekiko aplikazioa da. Aplikazioa 2011-ko abuztuaren 14an sortu zuen bardsoftware izeneko taldeak. Talde hau 4 pertsonek osatzen dute, hala ere, aplikazioak kode irekia duenez, beste erabiltzaile batzuk aplikazioarentzako hobekuntzak garatu ditzakete, gero onartuak izan behar direnak. Kodea eta Issues atala Github-en eskuratu daiteke.

Aurreko ataletan azaldu den moduan, proiektua GanttProject aplikazioarentzako akatsen zuzenketan eta funtzionalitate berrien garapenean datza. Honen helburua GanttProject-en garatzaileei laguntza ematea da, beraien aplikazioan erabiltzaileek aurkitu dituzten akatsak konponduz, funtzionalitate berriak garatuz edota kodearen refaktorizazioa eginez.

Honen bidez, erabiltzaileek aplikazioarekin lan egitean dituzten arazoak ekidin nahi dira, baita beraien aplikazioaren erabilera erraztea funtzionalitate berrien garapenarekin. Horrez gain, kodearen garbiketa bat egitea ere lortu nahi da, erabiltzen ez diren inplementazio zatiak ezabatuz.

[1] `https://github.com/bardsoftware/ganttproject`

1.3. Proiektuaren aukeraketaren zergatia

Lau dira proiektua hau hautatzearen arrazoiak:

1.3.1. Proiektuaren ideiarekiko konplizitatea

Juananen proposamen guztiak entzun eta gero, argi neukan zein izango zen nire proiektua. Alde batetik, gustukoa dudan programazioa hobetzen eta honekin lotutako tresna eta teknika berriak ezagutzen laguntzen du. Bestetik, ikasleok GAL-a aurrera eramateko eta kudeatzeko erabiltzen dugun tresnarekin lan egitea.

1.3.2. Ikasitakoarekin jarraitu

GAL-an erabiltzen diren erreminta gehienak graduan zehar ezagutu izan dira. Proiektu honen bidez, tresna horiekin sakondu egin dut baita beste tresna batzuk ezagutu eta erabili ditut.

1.3.3. Benetan erabiltzen den aplikazioa

Proiektu guztiek dute helburu berdina, benetan aplikatuak eta erabiliak izatea. Proiektu hau garatzen hasi baino lehen banekien nik garatutako funtzionalitate berriak eta akatsen konponketak eta kode refaktorizazioak erabiltzaile asko dituen aplikazio batean egongo zirela, beraz, ez zegoen motibazio falta.

1.3.4. Graduko praktikekin lotuta

Laugarren mailako kurtsoan zehar, Accenture enpresan egon nintzen praktiketan. Bertan Iberdrolaren mahaigaineko aplikazioarekin lan egiten egon nintzen; zehazki, Iberdrolak bezero bezala beraien aplikazioarentzako hobekuntzak, funtzionalitate berriak edo akatsen konponketa egitea eskatzen zuen eta horretan murgildu nintzen nire praktika denboraldian. Beraz, proiektua eta enpresan egiten nituen atazak oso lotuta daude, biak helburu berdina dute: benetan erabiltzen den aplikazio batentzako funtzionalitate berrien garapena eta akatsen konponketa.

2. Hasierako planteamendua

Atal honek, aurreko sarrerarekin batera, Proiektuaren Helburuen Dokumentua (PHD) osatzen du. Dokumentu honen bidez proiektuaren irismena eta helburuak azaltzen dira. Horretaz aparte, erabilitako erremintak, planifikazio tenporala, ebaluazio ekonomikoa, erabilitako arkitektura eta arriskuen analisia azalduko dira.

2.1. Helburuak

Proiektuaren helburuak azaltzeko bi ataletan banatuko ditugu: aipatutako azpiproiektuen helburuak eta helburu orokorrak.

2.1.1. Azpiproiektuen helburuak

Azpiproiektu bakoitzak bete beharreko helburuak hurrengoak dira:

- **Aplikazioaren akatsen bat konpontzea**: Erabiltzaileek aplikazioarekin lan egitean izaten dituzten arazoak konpontzea, beraien esperientzia hobetuz.
- **Funtzionalitate berria garatzea**: Aplikazioaren erabiltzaileentzat erabilgarria izan daitekeen funtzionalitate berriaren garapena, aplikazioaren erabilera errazteko.
- **Kodea refaktorizatzea**: Kodeak erabili gabe dituen klase nahiz metodoak ezabatzea, kode garbiago bat lortzeko eta aplikazioaren tamaina txikitzeko.

2.1.2. Helburu orokorrak

Atal honetan hiru azpiproiektuek bete behar dituzten helburuak eta dokumentazioarekin lotutako helburuak azalduko dira:

- **Kalitatezko dokumentazioa**: Proiektu honetan benetako aplikazio baten gainean lan egiten denez, dokumentazio ondo egituratu bat behar da. Horrez gain, gai honen barruan gehiago sakondu ahal denez, lagungarria izan daiteke sakontzen hasi baino lehen dokumentazioari begirada bat botatzea.
- **Aplikazioaren garatzaileei laguntza ematea**: Azpiproiektu bakoitzaren bidez aplikazioarentzako hobekuntzak lortu behar dira. Ondorioz, aplikazioaren garatzaileek beraien aplikazioarentzako funtzionalitate berriak edota akatsen konponketak jasoko dituzte.

2.2. Erabilitako tresnak

Atal honetan proiektuaren garapenerako erabili diren tresnen azalpen labur bat emango da eta zertarako erabili den argituko da:

- **Cacoo**: Mota desberdinetako diagramak egiteko online tresna da. Adibidez, LDE diagrama egiteko erabili da proiektuan zehar
- **GanttProject**: Proiektuak kudeatzeko laguntza eskaintzen duen softwarea da. Honi esker erraz planifikatu ahal izan da garapenean zehar egin beharreko ataza bakoitzaren denbora.
- **Git**:Kode irekiko sistema da, bertsioen kudeaketarako erabiltzen dena. Github, Gitlab edota Bitbucket sistemetan integratua aurkitu dezakegu.
- **Gradle**: Aplikazioak eraikitzeko kode irekiko automatizazio-sistema da. Proiektuan zehar GanttProject aplikazioa eraikitzeko eta konpilatzeko erabili izan da.
- **IntelliJ IDEA**: Programa informatikoak garatzeko JetBrainsek eskaintzen duen garapen ingurumena da.
- **Java**: Objektuei bideratutako programazioa-lengoaia da. Proiektuaren garapena aurrera eramateko erabili den lengoaia da.
- **Overleaf**: Latex-en bitartez PDF-ak sortzeko online tresna. Honen bitartez, denbora errealean ikus daiteke kode bitartez idatzita dagoena.
- **Sdkman**: SDK bertsioen kudeaketarako tresna. Java eta Gradle bertsioak kudeatzeko erabili da.
- **Skype**: Komunikaziorako Microsoft-ek eskaintzen duen tresna. Bilera birtualak egin ahal izateko erabili da.

2.3. Arkitektura

Atal honetan, GanttProject aplikazioaren arkitektura azalduko da; horretarako, honek dituen pakete guztiak adieraziko dira eta bakoitzaren betebeharra labur azalduko da.

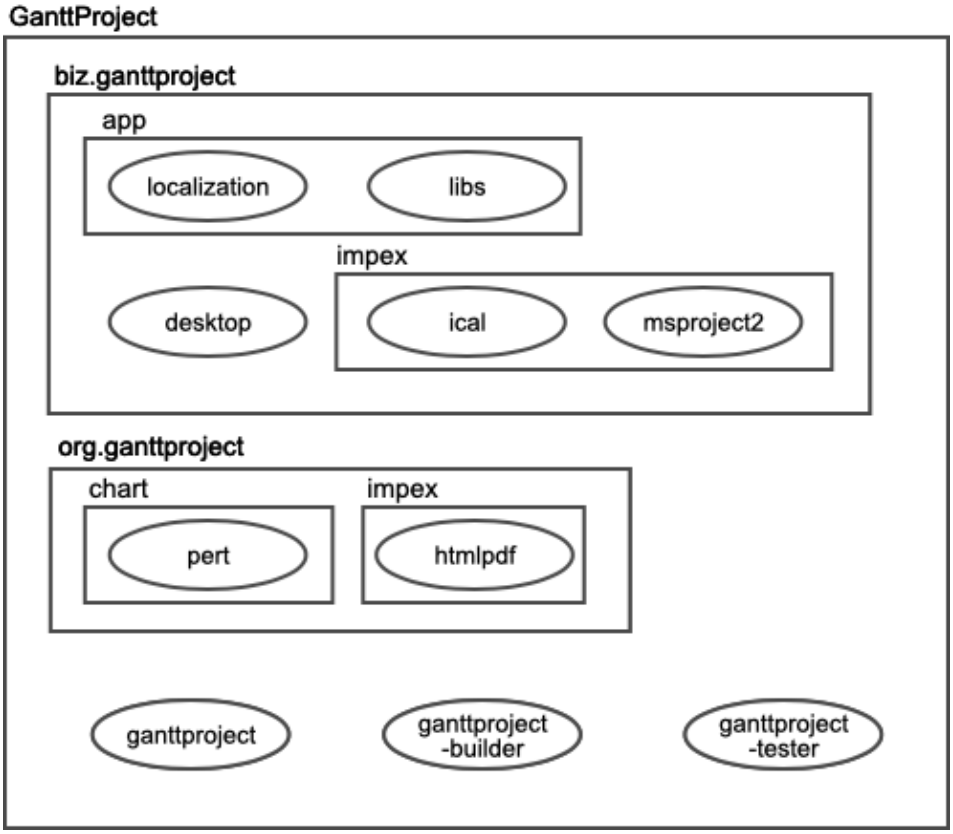

Figura 2.1: *GanttProject aplikazioaren arkitektura*

- **biz.ganttproject.app.libs**: Aplikazioak erabiltzen dituen liburutegiak gordetzeko paketea.
- **biz.ganttproject.app.localization**: Aplikazioak eskaintzen dituen hizkuntzak kudeatzen dituen paketea da. Honen barruan properties fitxategiak gordetzen dira, hizkuntza bakoitzaren itzulpenekin.
- **biz.ganttproject.core**: Aplikazioaren negozio logika osatzen duten klaseak kudeatzeen dituen paketea.

- **biz.ganttproject.impex.ical**: ICS motako fitxategiak kudeatzeko erabiltzen den paketea.
- **biz.ganttproject.impex.msproject2**: ProjectFile fitxategiak kudeatzeko behar den paketea.
- **ganttproject**: Aplikazioaren negozio logika atala osatzen duen paketea.
- **ganttproject-builder**: Aplikazioa eraikitzeko beharrezkoak diren fitxategiak dituen paketea.
- **ganttproject-tester**: Aplikazioaren testak kudeatzen dituen paketea da.
- **org.ganttproject.chart.pert**: PERT diagrama motak kudeatzeko paketea.
- **org.ganttproject.impex.htmlpdf**: HTML eta PDF formatuak kudeatzeko erabiltzen den paketea. Adibidez, PDF fitxategiak inportatzeko edo esportatzeko.

2.4. Irismena

Proiektu honen irismena azaltzeko LDE diagrama erabiliko da, proiektuan landuko diren bloke guztiak biltzen dituena. Aurkitu diren atal nagusiak hauek dira: Ikasketa, Antolaketa, 1.Azpiproiektua, 2.Azpiproiektua, 3.Azpiproiektua eta Dokumentazioa.

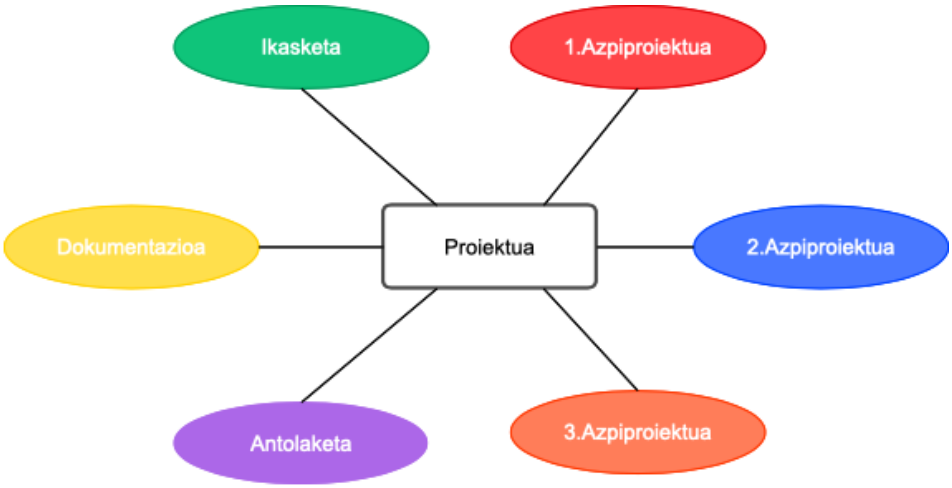

Figura 2.2: Proiektuaren LDE diagrama

2.2 Irudian aipatutako 6 atalak ikus daitezke. Jarraian bloke bakoitzaren azalpen labur bat emango da.

- **Ikasketa**: Proiektua garatzeko erabiliko diren tresna eta erreminten ikasketa prozesua biltzen duen atala.
- **Antolaketa**: Proiektua nola eramango den aurrera azaltzen duen atala.
- **1.Azpiproiektua**: Lehenengo azpiproiektuaren garapena biltzen duen atala.
- **2.Azpiproiektua**: Bigarren azpiproiektuaren garapena biltzen duen atala.
- **3.Azpiproiektua**: Hirugarren azpiproiektuaren garapena biltzen duen atala.

- **Dokumentazioa**: Proiektuari buruzko informazioaren idazketarako atala.

Jarraian, bloke bakoitzaren planifikazioa azalduko da, bakoitzari dagokion irudi eta taulak erabiliz.

2.4.1. Ikasketa

Atal honetan, proiektua aurrera eraman ahal izateko beharrezko tresnen ikasketa prozesuaren xehetasunak irudikatuko dira.

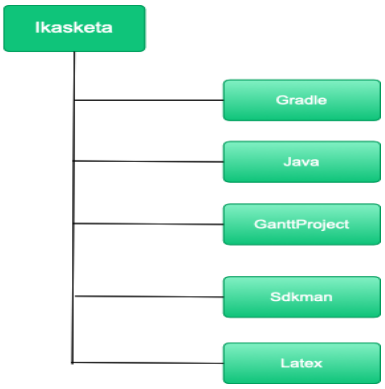

Figura 2.3: Ikasketa blokearen LDE diagrama

2.2 irudian, ikasketa blokearen LDE diagrama ikus daiteke, blokeak dituen zeregin guztiak zerrendatuta. Ondoren agertzen diren tauletan zeregin xehetasunak azalduko dira. Ataza batzuk antzeko deskribapena dute, beraz, taula bakar batean bateratuko dira.

Java eta Gradle ikasketa
Lan blokea: Ikasketa. **Ustezko iraupena**: 15 ordu.
Deskribapena: Java programazio-lengoaian eta Gradle sisteman sakondu beraien erabilera ikasiz. **Irteera/Entregagaiak**: Ez dago. **Beharrezko baliabideak**: Interneterako konexioa.

Tabla 2.1: Java eta Gradle ikasketa

GanttProject aplikazioaren ikasketa eta konfigurazioa
Lan blokea: Ikasketa. **Ustezko iraupena**: 60 ordu.
Deskribapena: GanttProject aplikazioaren kodea ulertzea, aplikazioaren konfigurazioa eta hau eraikitzea. **Irteera/Entregagaiak**: Garapenerako ingurumena prestatuta. **Beharrezko baliabideak**: Interneterako konexioa. IntelliJ IDEA. Java. Gradle. Sdkman

Tabla 2.2: GanttProject aplikazioaren ikasketa eta konfigurazioa

Sdkman ikasketa
Lan blokea: Ikasketa. **Ustezko iraupena**: 10 ordu.
Deskribapena: Sdkman erreminta instalatzea eta erabiltzen ikastea. **Irteera/Entregagaiak**: Ez dago. **Beharrezko baliabideak**: Interneterako konexioa.

Tabla 2.3: Sdkman ikasketa

Latex ikasketa
Lan blokea: Ikasketa. **Ustezko iraupena**: 25 ordu.
Deskribapena: Proiektuaren dokumentazioa egiteko erabili den programazio-lengoaia ikastea. **Irteera/Entregagaiak**: Ez dago. **Beharrezko baliabideak**: Overleaf web-orria, Latex-en idazteko.

Tabla 2.4: Latex ikasketa

2.4.2. Antolaketa

Atal honetan proiektuaren garapen zuzena lortzeko beharrezko zereginen xehetasunak azalduko dira.

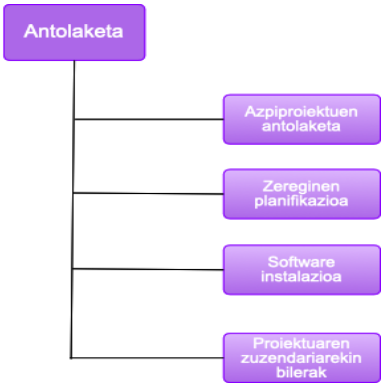

Figura 2.4: Antolaketa blokearen LDE diagrama

2.4 irudian, antolaketa blokearen LDE diagrama ikus daiteke. Ondoren agertzen diren tauletan zereginen xehetasunak azalduko dira.

Azpiproiektuen antolaketa
Lan blokea: Antolaketa. **Ustezko iraupena**: 6 ordu.
Deskribapena: Zeregin honetan proiektua hiru azpiproiektuetan banatu da eta azpiproiektu bakoitza zertan datzan zehaztu da. **Irteera/Entregagaiak**: Hiru azpiproiektuak zehazten dituen dokumentua. **Beharrezko baliabideak**: Interneterako konexioa. Koadernoa eta boligrafoa.

Tabla 2.5: Azpiproiektuen antolaketa

Zereginen planifikazioa
Lan blokea: Antolaketa. **Ustezko iraupena**: 6 ordu.
Deskribapena: Proiektua garatzeko beharrezko zeregin guztiak identifikatu, deskribapena egin eta planifikatu. **Irteera/Entregagaiak**: LDE diagrama eta xehetasunak biltzen dituen dokumentua. **Beharrezko baliabideak**: Cacoo LDE diagrama sortzeko eta koaderno eta boligrafoa dokumentua osatzeko

Tabla 2.6: Zereginen planifikazioa

Software instalazioa
Lan blokea: Antolaketa. **Ustezko iraupena**: 15 ordu.
Deskribapena: Proiektua garatzeko beharrezko tresna guztiak instalatzea eta konfiguratzea. **Irteera/Entregagaiak**: Ez dago. **Beharrezko baliabideak**: Interneterako konexioa.

Tabla 2.7: Software instalazioa

Proiektuaren zuzendariarekin bilerak
Lan blokea: Antolaketa. **Ustezko iraupena**: 15 ordu.
Deskribapena: Proiektuaren zuzendariarekin periodikoki egindako bilerak, online nahiz presentzialak. Bertan proiektuarekin izandako arazoak konpontzea izan da helburu nagusia. **Irteera/Entregagaiak**: Bilera bakoitzaren oharrak orri batean. **Beharrezko baliabideak**: Skype.

Tabla 2.8: Proiektuaren zuzendariarekin bilerak

2.4.3. 1.Azpiproiektua

Atal honetan 1.azpiproiektuaren garapenerako zereginen xehetasunak azaltzen dira.

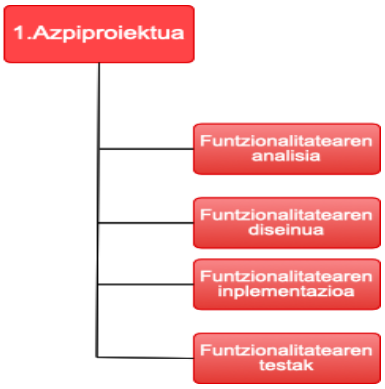

Figura 2.5: 1.Azpiproiektua blokearen LDE diagrama

2.5 irudian, 1.Azpiproiektua blokearen LDE diagrama ikus daiteke. Ondoren, garapenerako zereginen xehetasunak azalduko dira taulak erabiliz.

Funtzionalitatearen analisia
Lan blokea: 1.Azpiproiektua. **Ustezko iraupena**: 20 ordu.
Deskribapena: Aplikazioaren Github-eko web-orrian funtzionalitate berria bilatu eta funtzionalitatea aztertu . **Irteera/Entregagaiak**: Funtzionalitatearen deskribapena duen dokumentua. **Beharrezko baliabideak**: Github. Intellij IDEA.

Tabla 2.9: Funtzionalitatearen analisia

Funtzionalitatearen diseinua
Lan blokea: 1.Azpiproiektua. **Ustezko iraupena**: 20 ordu.
Deskribapena: Funtzionalitatean eragiten duten klaseekin klase-diagrama sortu eta inplementaziorako sekuentzia-diagrama osatu. **Irteera/Entregagaiak**: Klase-diagrama eta sekuentzia-diagrama. **Beharrezko baliabideak**: IntelliJ IDEA. Cacoo.

Tabla 2.10: Funtzionalitatearen diseinua

Funtzionalitatearen inplementazioa
Lan blokea: 1.Azpiproiektua. **Ustezko iraupena**: 10 ordu.
Deskribapena: Funtzionalitate berria inplementatzea. **Irteera/Entregagaiak**: Ez dago. **Beharrezko baliabideak**: IntelliJ IDEA.

Tabla 2.11: Funtzionalitatearen inplementazioa

Funtzionalitatearen testa
Lan blokea: 1.Azpiproiektua. **Ustezko iraupena**: 10 ordu.
Deskribapena: Inplementatutako funtzionalitate berriaren testak egitea. **Irteera/Entregagaiak**: Ez dago. **Beharrezko baliabideak**: IntelliJ IDEA. Gradle.

Tabla 2.12: Funtzionalitatearen inplementazioa

2.4.4. 2.Azpiproiektua

Atal honetan 2.azpiproiektuaren garapenerako zereginen xehetasunak azaltzen dira.

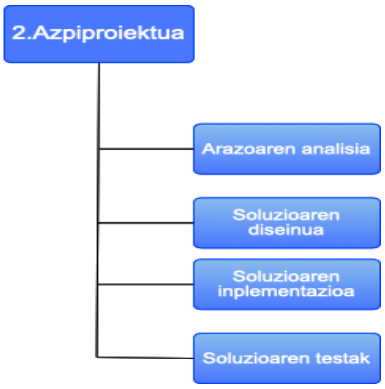

Figura 2.6: 2.Azpiproiektua blokearen LDE diagrama

2.6 irudian, 2.Azpiproiektua blokearen LDE diagrama ikus daiteke. Ondoren, garapenerako zereginen xehetasunak azalduko dira taulak erabiliz.

Arazoaren analisia
Lan blokea: 2.Azpiproiektua. **Ustezko iraupena**: 40 ordu.
Deskribapena: Aplikazioaren Github-eko web-orrian erabiltzaileek aurkitutako arazoa identifikatzea eta aztertzea. Behin hori eginda soluzio bat lortu. **Irteera/Entregagaiak**: Arazoarentzako soluzioa azaltzen duen dokumentua. **Beharrezko baliabideak**: Github. Intellij IDEA.

Tabla 2.13: Arazoaren analisia

Soluzioaren diseinua
Lan blokea: 2.Azpiproiektua. **Ustezko iraupena**: 20 ordu.
Deskribapena: Aurkitutako soluzioan eragiten duten klaseekin klase-diagrama sortu eta inplementaziorako sekuentzia-diagrama osatu. **Irteera/Entregagaiak**: Klase-diagrama eta sekuentzia-diagrama. **Beharrezko baliabideak**: IntelliJ IDEA. Cacoo.

Tabla 2.14: Soluzioaren diseinua

Soluzioaren inplementazioa
Lan blokea: 2.Azpiproiektua. **Ustezko iraupena**: 30 ordu.
Deskribapena: Aurkitutako soluzioaren inplementazioa egitea. **Irteera/Entregagaiak**: Ez dago. **Beharrezko baliabideak**: IntelliJ IDEA.

Tabla 2.15: Soluzioaren inplementazioa

Soluzioaren testa
Lan blokea: 2.Azpiproiektua. **Ustezko iraupena**: 10 ordu.
Deskribapena: Inplementatutako soluzioaren testak egitea. **Irteera/Entregagaiak**: Ez dago. **Beharrezko baliabideak**: IntelliJ IDEA. Gradle.

Tabla 2.16: Soluzioaren inplementazioa

2.4.5. 3.Azpiproiektua

Atal honetan 3.azpiproiektuaren garapenerako zereginen xehetasunak azaltzen dira.

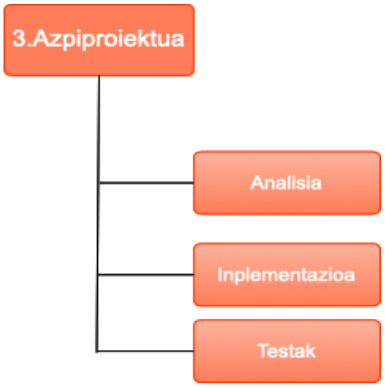

Figura 2.7: 3.Azpiproiektua blokearen LDE diagrama

2.7 irudian, 3.Azpiproiektua blokearen LDE diagrama ikus daiteke. Ondoren, garapenerako zereginen xehetasunak azalduko dira taulak erabiliz.

Analisia
Lan blokea: 3.Azpiproiektua. **Ustezko iraupena**: 22 ordu.
Deskribapena: Aplikazioak erabiltzen ez dituen klase eta metodoen identifikazioa IntelliJ-k eskaintzen dituen tresnen bidez, eta tresna honen emaitzen analisia. **Irteera/Entregagaiak**: Erabiltzen ez diren klase eta metodoen lista. **Beharrezko baliabideak**: IntelliJ IDEA.

Tabla 2.17: Analisia

Inplementazioa
Lan blokea: 3.Azpiproiektua. **Ustezko iraupena**: 8 ordu.
Deskribapena: Analisian lortutako metodo eta klaseak kontuan izanda kode refaktorizazioa egitea. **Irteera/Entregagaiak**: Erabiltzen ez diren klase eta metodoen lista. **Beharrezko baliabideak**: IntelliJ IDEA.

Tabla 2.18: Inplementazioa

Testak
Lan blokea: 3.Azpiproiektua. **Ustezko iraupena**: 10 ordu.
Deskribapena: Inplementazioan egindako aldaketak probatzea. **Irteera/Entregagaiak**: Ez dago. **Beharrezko baliabideak**: IntelliJ IDEA.

Tabla 2.19: Testak

2.4.6. Dokumentazioa

Azken atalean, dokumentazioarekin lotuta dauden xehetasunak azaltzen dira.

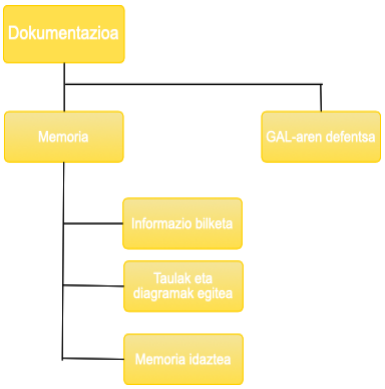

Figura 2.8: Dokumentazioa blokearen LDE diagrama

2.8 irudian, antolaketa blokearen LDE diagrama ikus daiteke. Ondoren agertzen diren tauletan zereginen xehetasunak azalduko dira.

Informazio bilketa
Lan blokea: Dokumentazioa. **Ustezko iraupena**: 8 ordu.
Deskribapena: Proiektuan erabiltzen diren tresna eta erremintei buruzko informazioa bilatu eta bildu. **Irteera/Entregagaiak**: Bibliografia. **Beharrezko baliabideak**: Interneterako konexioa.

Tabla 2.20: Informazio bilketa

Taulak eta diagramak egitea
Lan blokea: Dokumentazioa. **Ustezko iraupena**: 10 ordu.
Deskribapena: Proiektuaren memorian zehar erabiltzen diren taula eta diagramak sortzea. **Irteera/Entregagaiak**: Memoriako irudi eta taulak. **Beharrezko baliabideak**: Interneterako konexioa.

Tabla 2.21: Taulak eta diagramak egitea

Memoria idaztea
Lan blokea: Dokumentazioa. **Ustezko iraupena**: 80 ordu.
Deskribapena: Proiektuaren memoria idaztea. **Irteera/Entregagaiak**: Memoria. **Beharrezko baliabideak**: Overleaf.

Tabla 2.22: Memoria idaztea

GAL-aren defentsa
Lan blokea: Dokumentazioa. **Ustezko iraupena**: 10 ordu.
Deskribapena: Proiektuaren defentsa egiteko behar diren diapositibak sortzea. **Irteera/Entregagaiak**: Defentsarako diapositibak. **Beharrezko baliabideak**: Google Aurkezpenak.

Tabla 2.23: GAL-aren defentsa

2.4.7. Planifikazioaren laburpena

ID	*Atazaren izena*	*Aitzindariak*	*Ustezko iraupena*
	GanttProject		460 ordu
1	***Ikasketa***		110 ordu
1.1	Java eta Gradle ikasketa		15 ordu
1.2	GanttProject aplikazioaren ikasketa eta konfigurazioa	1.1,1.3	60 ordu
1.3	Sdkman ikasketa		10 ordu
1.4	Latex ikasketa		25 ordu
2	***Antolaketa***		42 ordu
2.1	Azpiproiektuen antolaketa		6 ordu
2.2	Zereginen planifikazioa	2.1	6 ordu
2.3	Software instalazioa		15 ordu
2.4	Proiektuaren zuzendariarekin bilerak		15 ordu
3	***1.Azpiproiektua***		60 ordu
3.1	Funtzionalitatearen analisia	1.2,2.2,2.3	20 ordu
3.2	Funtzionalitatearen diseinua	3.1	20 ordu
3.3	Funtzionalitatearen inplementazioa	3.2	10 ordu
3.4	Funtzionalitatearen testa	3.3	10 ordu
4	***2.Azpiproiektua***		100 ordu
4.1	Arazoaren analisia	3.4	40 ordu
4.2	Soluzioaren diseinua	4.1	20 ordu
4.3	Soluzioaren inplementazioa	4.2	30 ordu
4.4	Soluzioaren testa	4.3	10 ordu
5	***3.Azpiproiektua***		40 ordu
5.1	Analisia	4.4	20 ordu
5.2	Inplementazioa	5.1	10 ordu
5.3	Testak	5.2	10 ordu
6	***Dokumentazioa***		108 ordu
6.1	Informazio bilketa	5.3	8 ordu
6.2	Taulak eta diagramak egitea	5.3	10 ordu
6.3	Memoria idaztea	1.4,6.1,6.2	80 ordu
6.4	GAL-aren defentsa	6.3	10 ordu

Tabla 2.24: Zereginen iraupen eta aitzindariak(1)

2.5. Denboraren planifikazioa

Behin atazen zerrenda eginda dagoela, proiektuan zehar zeregin bakoitza nola burutuko den planifikatu behar da. Horretarako, 2.9 irudian dagoen Gantt diagrama osatu da. Diagrama honetan, aurreko atalean aipatu diren bloke guztiak ikusi daitezke.

Proiektuan langile batek egunero 3 ordu egin dituela simulatu nahi da, hots, aste bakoitzeko 21 ordu produktibo. Proiektua 2019ko irailaren 23an jarri zen martxan eta 2020ko uztailaren 3rako bukatuta egotea planifikatuta dago. Denbora tarte horretan 40 aste eta 5 egun daude, beraz, hurrengo kalkuluak egin dira:

$$Aste\ kopurua * Ordu\ kopurua + 5\ egun = 40 * 21 + 15 = 855\ ordu \quad (2.1)$$

Lortutako 855 ordu horietatik asteburuetako orduak kenduko ditugu, asteburuetan ez baita proiektuan lan egingo. Guztira asteburuetan lan egingo ez den ordu kopurua hurrengoa da:

$$Aste\ kopurua * Ordu\ kopurua\ asteburuko = 40 * 6 = 240\ ordu \quad (2.2)$$

Beraz, planifikatutako benetako ordu kopurua horrela geratuko da:

$$Ordu\ kopurua\ guztira - Asteburuko\ ordu\ kopurua = 855 - 240 = 615\ ordu \quad (2.3)$$

Proiektuan lan egiteko izango dudan ordu kopuruaren estimazioa aztertuz, proiektua amaitzeko denbora dagoela esan daiteke. Ordu kopuruaren estimazioa 615 da eta proiektuaren ustezko iraupena 460. Kontuan izan behar da, proiektua egiten den bitartean laugarren mailako azterketak edo bestelako entregak egon daitezkeela, eta horiek estimatutako ordu kopuruan eragina izan dezaketela. Baina,argi ikusten denez, beste zeregin horietarako denbora malgutasuna daukagu.

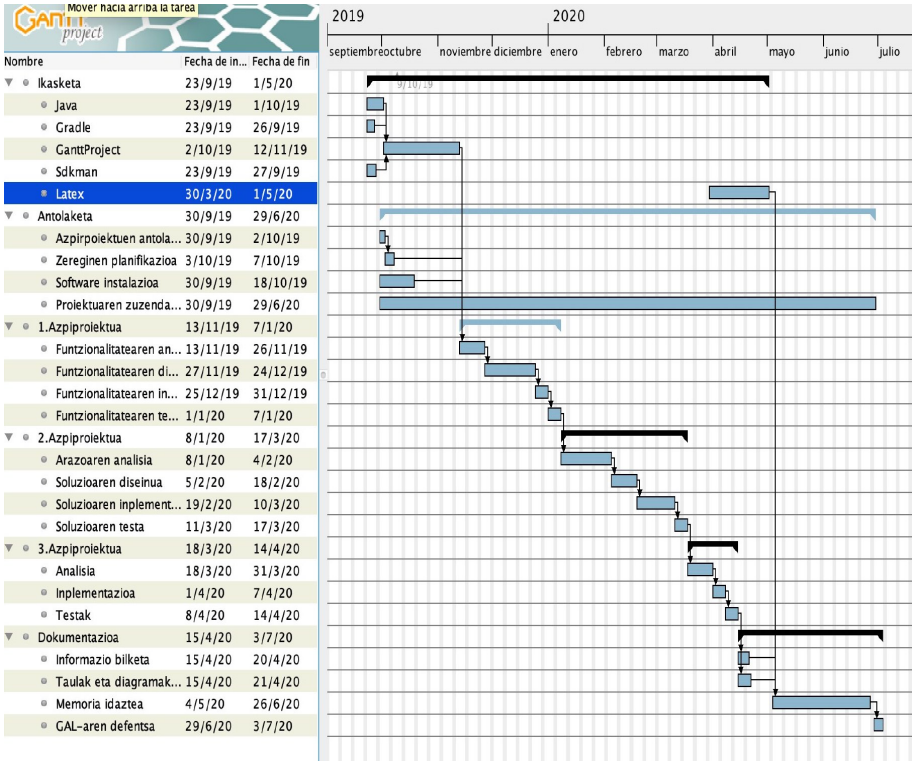

Figura 2.9: Gantt diagrama

2.6. Ebaluazio ekonomikoa

Hasierako planteamenduarekin bukatzeko, proiektuaren ebaluazio ekonomikoa egingo da. Nahiz eta proiektu honek ordainsaririk ez izan, interesgarria da honelako proiektu baten gastuak kalkulatzea. Honen bidez, antzeko proiektu batek jasoko lukeen mozkina ezagutu daiteke.

2.6.1. Eskulana

Proiektuan egindako lana Analista-Programatzaile bezala sailkatuta dago, beraz, sailkapen honetarako aholkularitzako eta merkatu-azterketako enpresen eta iritzi publikoaren estatuko 2018ko hitzarmen kolektiboak adierazten duen soldata minimoa erabili da, 2018ko hitzarmenerako 2018ko martxoaren 6an BOE-an argitaratutako tauletan oinarrituta. Dokumentu honetan, hilabetero 1481,22 € irabazten direla adierazten da (Soldaten taulak [Enplegu eta Gizarte Segurantzaren Ministerioa, 2018]).

Lehendabizi, kontuan izanda astero 40 ordu lan egiten direla, ordu bakoitzero irabazten dena adierazten da hurrengo ekuazioan :

$$Soldata_{orduro} = \frac{Soldata_{hilabetero}}{Ordu\ kopurua_{astero} \cdot Asteak_{hilabetero}} = \tag{2.4}$$

$$= \frac{1481.22\,€}{40 \cdot 4} = \frac{1481.22\,€}{160} = 9.26\,€ \tag{2.5}$$

Proiektuaren ustezko iraupena 460 ordukoa da. Horietatik, ikasketara bideratu diren orduak, 110 ordu, kendu behar dira, hauek ez baitira kostua kalkulatzeko erabiliko. Hala ere, ikasketa ataletik GanttProject aplikazioaren ikasketa eta konfigurazioa kontuan hartuko da. Hurrengo ekuazioan gastu bezala erabiliko diren ordu kopurua kalkulatuko da:

$$Orduak\ guztira - (Ikasketa\ orduak - GanttProject\ zeregina) = \tag{2.6}$$

$$= 460 - (110 - 60) = 410\ ordu \tag{2.7}$$

Orain, eskulanari dagokion gastu ekonomiko totala kalkulatu dezakegu:

$$Soldata = Lanordu\ kopurua \cdot Soldata_{orduro} = 410 \cdot 9.26\,€ = 3796.60\,€ \tag{2.8}$$

Beraz, eskulanari dagokion kostu totala 3796.60 €koa da.

2.6.2. Software gastua

Proiektu honetan erabili diren tresna guztiak doakoak edo kode irekikoak direnez, proiektuaren software gastua 0 €koa da.

2.6.3. Hardware gastua

Atal honetako gastua proiektua egiteko erabili den ordenagailu eramangarriaren amortizazioan oinarrituta dago. Ordenagailua Apple markakoa da, bere bizi iraupenaren estimazioa 5 urtekoa da eta prezioa 1 €500koa da.

Beraz, amortizazioa hilabetero hurrengoa da:

$$Amortizazioa_{hilabetero} = \frac{Prezioa}{Estimatutako\ bizi\ iraupena} = \frac{1500}{60\,\text{hilabete}} = 25\,€ \tag{2.9}$$

Aurreko emaitza kontuan izanda amortizazio totala kalkulatuko da:

$$Amortizazioa_{totala} = Amortizazioa_{hilabetero} \cdot Hilabete\ kopurua = 25 \cdot 9 = 225\,€ \tag{2.10}$$

2.6.4. Zeharkako gastuak

Atal honetan proiektuaren garapenarekin lotuta dauden kostu ez zuzenak kalkulatuko dira, Internet eta elektrizitatea, alegia.

Aurretik aipatu den moduan, guztira 460 ordu lan egingo dira. Suposatuko dugu ordenagailua piztuta egon dela ordu guzti horiek.

Ordenagailu eramangarri baten orduko batezbesteko kontsumoa 0,045 €-koa da ([GAndroid, 2014]), beraz, hurrengo emaitza lortzen da :

$$Elektrizitate\ gastuak = Ordu\ kopurua \cdot Orduko\ kontsumoa = 460 \cdot 0,045 = 20.70\,€ \tag{2.11}$$

Bestalde, Internet-ari dagokionez, kostu totala lan egingo diren 9 hilabeteen kontratazioaren prezioa izango da. Nire kasuan hilabeteroko Internet kostua 45 €-koa da, beraz, hau da lortzen den emaitza:

$$Internet\ gastuak = Hilabete\ kopurua \cdot Hilabeteroko\ kontratazio\ prezioa = 9 \cdot 45 = 405\,€ \tag{2.12}$$

Azkenik, zeharkako gastu guztien batura egingo da:

$$Zeharkako\ gastuak = Elektrizitate\ gastuak\ +\ Internet\ gastuak\ = \tag{2.13}$$

$$= 20,70 + 405 = 425.70 € \tag{2.14}$$

Guztira atal honen batura 425,70 €-koa da.

2.6.5. Proiektuaren kostu totala

2.25 taulan aurreko ataletan kalkulatu diren gastu guztiak agertzen dira, baita proiektuaren kostu totala:

Gastu mota	***Gastua (€)***
Eskulana	3796,60 €
Software gastua	0 €
Hardware gastua	225 €
Zeharkako gastuak	425,70 €
Guztira	4447,30 €

Tabla 2.25: Gastuen laburpena

2.7. Arriskuen analisia

Proiektu guztietan zehar ustekabeko atzerapenak gertatzen dira, horregatik, horien analisia egiteak izango dituen ondorioak minimizatzen lagundu dezake.

Jarraian proiektu honetarako kontuan izan diren arriskuak zerrendatzen dira, bakoitzarentzako diseinatutako prebentzio plana eta soluzioa adierazi, arriskua ekiditeko eta ondorioak minimizatzeko. Horrez gain, arrisku bakoitzaren probabilitatea eta inpaktua ere adierazki dira.

2.7.1. Kodearen galera

Gorde ez diren kode zatiak galtzea da, hauek lanorduen galera bat dakarte. Elektrizitate edo Interneten akatsa, garatzailearen despistea eta ordenagailuaren akatsa izan daitezke arriskuaren sortzaileak.

Prebentzio plana

- Online errepositorio bat erabiltzea kodea gordetzeko, kasu honetan, Github.
- Egunero gutxienez behin kodea errepositoriora igo, nahiz eta kodea amaitu gabe egon.
- Kodearen kopia bat memoria batean gordetzea.

Soluzioa

- Errepositorioan gordetako azken bertsioa jaitsi.
- Errepositorioarekin arazoak badaude, memorian gorde den kopia erabili.

Probabilitatea

Baxua: 20 %

Inpaktua

0,20 x 1 egun = 0,6 ordu .

2.7.2. Ordenagailua matxuratzea

Proiektua garatzeko erabiltzen den ordenagailu eramangarriak jasan ditzakeen kalteak.

Prebentzio plana

- Ordenagailua eguneratua mantentzea.
- Ordenagailuaren berrikuspen periodikoa egitea.
- Ordenagailua soilik proiektuarekin lotutako zereginetarako erabiltzea gainkarga ekiditeko.
- Bigarren ordenagailu bat prest edukitzea.

Soluzioa

- Prest dagoen bigarren ordenagailuarekin proiektua garatzen jarraitzea, lehenengo hau konpontzen den bitartean.
- Biak kaltetuta badaude, lehenengo ordenagailua lehenbailehen konpontzen saiatzea.

Probabilitatea

Baxua: 10 %

Inpaktua

0,10 x 1 egun = 0,3 ordu .

2.7.3. Gaixotzea edo bestelako kalteren bat jasatea

Proiektuan zehar garatzailea kalteren bat jasan dezake, baita gaixotu daiteke. Kontuan izanda 2020ko martxoan eta apirilean egon den pandemia, honen probabilitatea altuagoa izango da.

Prebentzio plana

- Proiektuarekin lotutakoak:
  - Posizio egokia mantentzea lan egiten den bitartean.
  - Ordenagailuarekiko distantzia egokia mantentzea.
  - 10 minutuko atsedenak hartzea 90 minuturo.
- Bestelakoak:
  - Lo egiteko gomendatzen diren ordu kopuruak errespetatzea.
  - Kirola egitea, honen arriskuak kontuan izanda.
  - Pandemiarekin lotuta, Osasunaren ministerioak ezartzen dituen gomendioak betetzea.

Soluzioa

- Ahal bada medikuarenera joatea gaixotasunaren iraupena estimatzeko.
- Gaixotasuna larria bada, proiektua atzeratzea berreskuratu ahal izateko.
- Gaixotasuna ez bada larria, proiektuarekin jarraitu atseden gehiago hartuz.

Probabilitatea

Bitartekoa: 50 %

Inpaktua

0,50 x 1 egun = 1,5 ordu .

2.7.4. Interneterako konexioa galtzea

Lan egiten den bitartean Interneteko konexioa galtzearen ondorioz galdutako denbora. Arazo honek errepositoriora konektatzea, informazio bilaketa eta bestelako zereginak egitea ekiditen du.

Prebentzio plana

- Abiadura nahikoa duen Interneterako konexioa kontratatzea.
- Kable bidezko konexioa erabiltzea WiFi bidezkoaren aurretik.
- Proiektuaren lanordu tartean honekin zerikusia ez duten Interneten erabilerak saihestea, adibidez, tamaina handiko aplikazio baten deskarga.

Soluzioa

- Mugikorraren bidez ordenagailua Internetera konektatu.
- Unibertsitate edo liburutegiren batera joatea, ikasleentzako gela batean lanarekin jarraitu ahal izateko.

Probabilitatea

Baxua: 25 %

Inpaktua

0,25 x 1 egun = 0,75 ordu .

2.7.5. Elektrizitatearen galera

Lan egiten den bitartean etxeko elektrizitatea galtzea, honen ondorioz, ordenagailuaren bateria amaitu bezain laster ezin izango da proiektuarekin jarraitu

Prebentzio plana

- Ordenagailu eramangarriaren bateria beti kargatuta izatea.
- Bigarren ordenagailu bat prest izatea bateria kargatuarekin.

Soluzioa

- Bi ordenagailuekin lan egitea bateria daukaten bitartean.
- Unibertsitate edo liburutegiren batera joatea, ikasleentzako gela batean lanarekin jarraitu ahal izateko.

Probabilitatea

Baxua: 10 %

Inpaktua

0,10 x 1 egun = 0,0,3 ordu .

3. 1.Azpiproiektua

Atal honetan 1.Azpiproiekuaren garapena azalduko da, honekin lotuta egin diren zeregin guztiak kontuan hartuz.

Aurretik aipatu denez, azpiproiektu honen helburua aplikazioaren erabiltzaileentzat erabilgarria den funtzionalitate berria garatzea da. Honen bidez, erabiltzaileen aplikazioaren erabilera erraztu nahi da.

Azpiproiektu honen azalpena aurrera eramateko 6 azpiatal nagusi erabiliko dira: Funtzionalitatearen aukeraketa, analisia, diseinua, inplementazioa, testa eta ondorioak.

3.1. Funtzionalitatearen aukeraketa

Funtzionalitate berria garatu baino lehen, zein izango den garatuko den funtzionalitatea erabaki behar da. Erabaki hau hartzeko orduan bi aukera ezberdin agertzen dira.

Alde batetik, aplikazioa erabiltzea eta erabileran aurkitutako zailtasunen aurrean, zailtasun hori errazten duen funtzionalitate baten garapena hautatzea.

Beste aldetik, GanttProject aplikazioak Github-en duen errepositorioan bilatzea. Github-en aplikazioak Issues[1] izeneko atal bat dauka, bertan erabiltzaileek nahiz garatzaileek aplikazioa erabiltzean aurkitu dituzten akatsak azaltzen dituzte baita erabilera errazteko garatu daitezkeen funtzionalitate berriak ere.

Aukera biak kontuan izanda, azpiproiektu hau aurrera eramateko bigarren aukera erabili da. GanttProject-eko erabiltzaileek proposatutako funtzionalitate berria garatzea aukeratu da.

Aurreko erabakiaren ondoren, aipatutako Issues atalean murgilduta hainbat funtzionalitate berri zerrendatu dira. Funtzionalitate hauek Proiektuaren zuzendariarekin batera aztertu dira eta azkenean garatuko den funtzionalitatea erabaki da.

[1] `https://github.com/bardsoftware/ganttproject/issues`

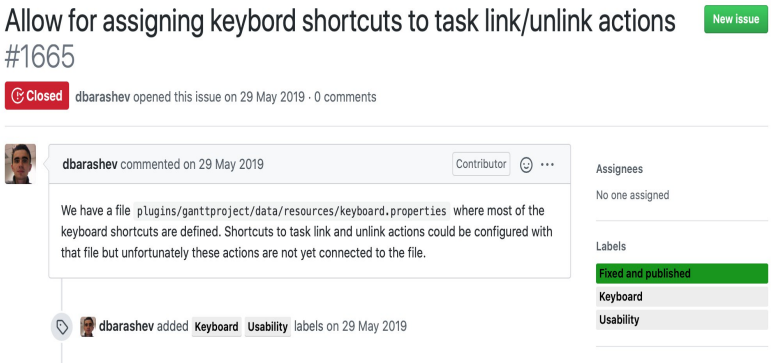

Figura 3.1: Funtzionalitatearen proposamena Github-en

3.1 irudian hautatu den funtzionalitatea Github-eko Issues atalean ikus daiteke. Bertan, funtzionalitatea proposatu duen erabiltzaileak gomendio edo azalpen txiki bat eman dezake. Kasu honetan, erabilgarria izan daitekeen keyboard.properties fitxategiaren path-a adierazten du.

3.2. Funtzionalitatearen analisia

Atal honetan, funtzionalitatearen azalpena eta funtzionalitatea garatzeko honekin lotuta dauden klase eta fitxategien identifikazioa egingo da.

GanttProject aplikazioak proiektu baten Gantt diagrama diseinatzeko aukera ematen du. Bertan, proiektuaren zeregin guztiak zerrendatzen dira eta zereginen artean loturak sortu daitezke. Adibidez, A eta B zereginak izanda, A B-ren aitzindaria dela adierazi dezakegu lotura baten bidez. Horretarako, aplikazioak ematen duen aukera zeregin baten propietateetan sartzea eta bertan lotura gehitzea da, 3.1 irudian ikus daitekeenez .

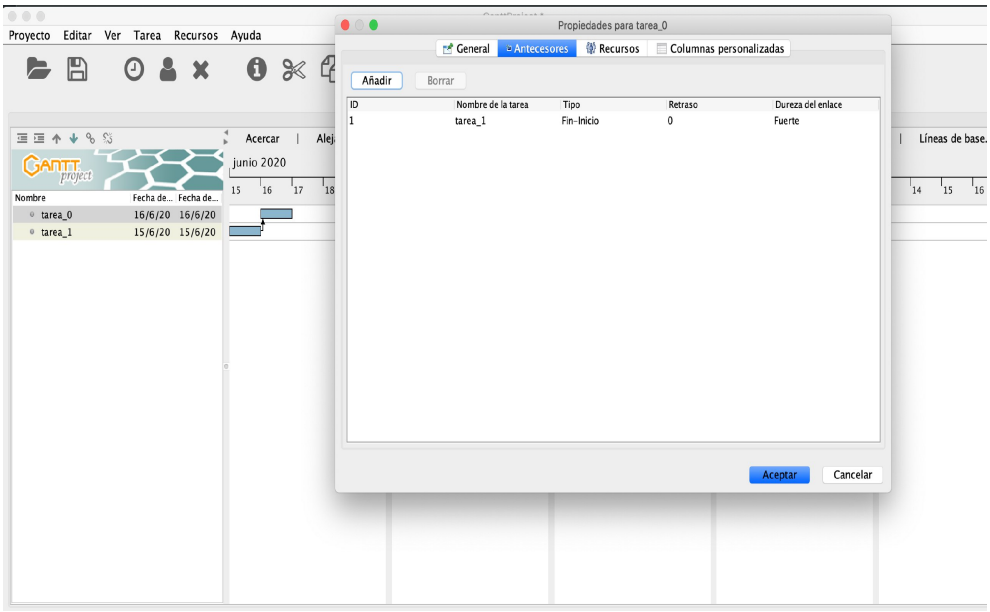

Figura 3.2: Lotura sortzeko beharrezko pausuak

Beraz, lehenengo azpiproiektuaren bitartez erabiltzaileari loturak era azkar eta errazago batean sortzeko aukera eman nahi da. Garatuko den funtzionalitateren bidez laster teklak erabiliz loturak sortzea da helburua. Kasu honetan, erabiltzaileak lotu nahi dituen bi zereginak aukeratuta izanda Ctrl + L komandoa sakatuta lotura sortuko du. Horrez gain, era berean lotura ezabatzeko Ctrl + U komandoa eskainiko da.

Behin funtzionalitatea azalduta, aipatutako garapena aurrera eramateko aldatu beharreko klase eta fitxategiak identifikatu behar ditugu.

Lehendabizi, Github-eko Issue-a sortu duen erabiltzaileak ematen duen

azalpenaren bidez, keyboard.properties fitxategia kontuan izan behar dugu. Fitxategi honetan, hainbat eragiketa burutzeko aplikazioak eskaintzen dituen laster tekla edo komandoak aurkitu ditzakegu. 3.3 irudian hainbat adibide ikus ditzakegu. Hau honela izanda, fitxategi honetan aurretik aipatutako komandoak sartu beharko direla ondorioztatzen da.

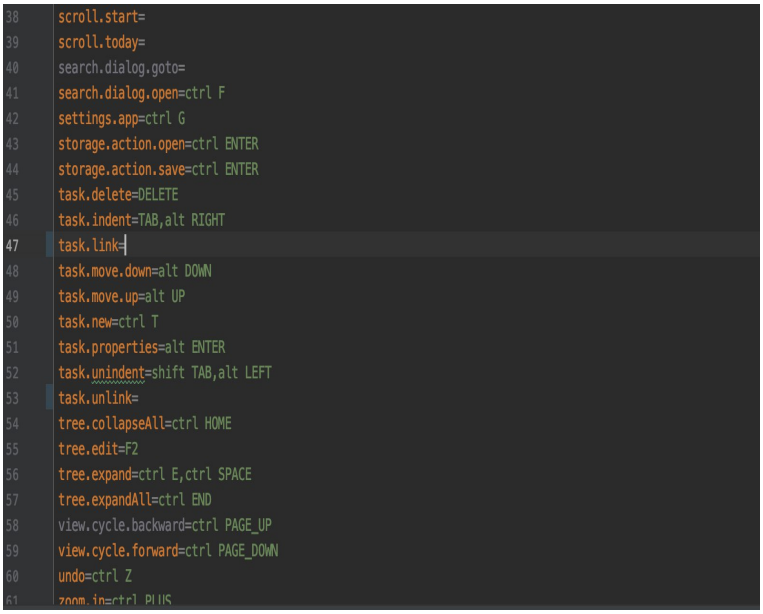

Figura 3.3: Keyboard.properties fitxategiaren edukia

Orain, kodearen azterketa egin behar da, zeintzuk diren eraldatu behar diren klaseak jakiteko. Azterketa hau egiteko, IntelliJ IDEA-k eskaintzen digun debugger-a erabiliko da. Tresna honen bidez, aplikazioak burutzen dituen ekintzen prozesua ikus dezakegu, eta horrela prozesuan parte hartzen duten klaseak identifikatu daitezke.

Aipatutako tresnaren bitartez, funtzionalitate berrian eragina izango duten TreeTableContainer, GanttTree2, TaskLinkAction, TaskUnlinkAction eta TreeTableClass klaseak identifikatu dira.

3.3. Diseinua

Atal honetan, aukeratutako funtzionalitatea garatzeko klase-diagramaren eta sekuentzia-diagramaren diseinua irudikatu eta azalduko da.

3.3.1. Klase-diagrama

Klase-diagrama sistema baten egitura azaltzeko erabiltzen den diagrama mota da. Ondoren, 1.Azpiproiektua aurrera eramateko diseinatu den klase-diagrama irudikatuko da eta agertzen den objektu bakoitzaren deskribapena emango da. Klase-diagrama sortzeko, aurreko atalean identifikatu diren klaseak erabiliko dira.

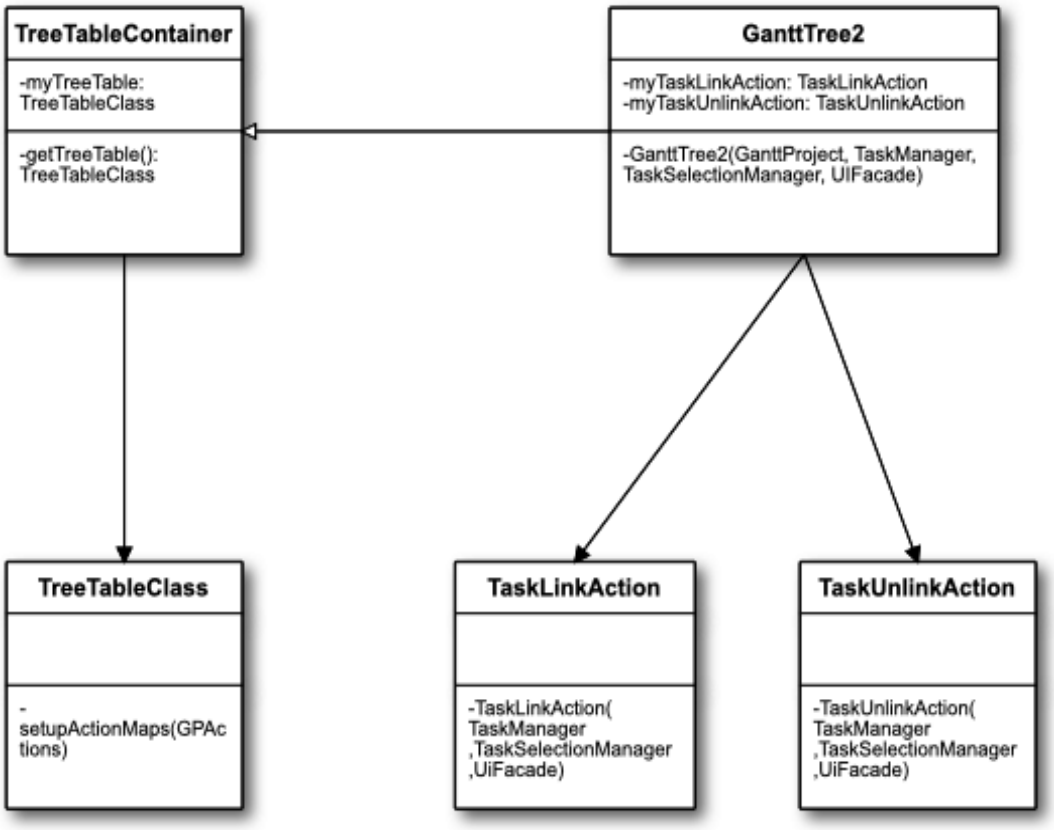

Figura 3.4: 1.Azpiproiektuaren klase-diagrama

- **TreeTableContainer**: Ekintza guztiak kudeatzeko erabiltzen den klase abstraktua.

- **GanttTree2**: TreeTableContainer klase abstraktua inplementatzen duen klasea da. Hainbat ekintza burutzen ditu, baina azpiproiektuaren garapenarekin lotuta dagoena ekintzen esleipena da.
- **TreeTableClass**: Ekintza guztiak esleitzen dituen klasea da.
- **TaskLinkAction**: Bi zereginen artean lotura sortzeko ekintza kudatzen duen klasea da.
- **TaskUnlinkAction**: Bi zereginen artean sortuta dagoen lotura ezabatzeko ekintza kudeatzen duen klasea da.

3.3.2. Sekuentzia-diagrama

Sekuentzia-diagrama sistema baten objektuek haien artean dituzten interakzioak grafikoki adierazten dituen diagrama da.3.5 irudian funtzionalitate berriak bete behar dituen ekintzak ikus daitezke.

Sekuentzia diagraman GanttTree2 objektuaren instantzia bat sortzerakoan funtzionalitate berria inplementatzeko egin beharreko ekintzak adierazten dira.

GanttTree2-ren instantzia sortzen denean, TaskLinkAction eta TaskUnlinkAction objektuen instantzia bat sortu behar da, honen bidez, lotura sortzeko eta lotura ezabatzeko ekintzaren kudeatzaileak izango ditugu.

Behin hau eginda, sortuta dagoen TreeTableClass objektuaren instantziaren setupActionMaps() metodoaren bidez, aurretik sortu ditugun ekintzen instantziak esleituko ditugu.

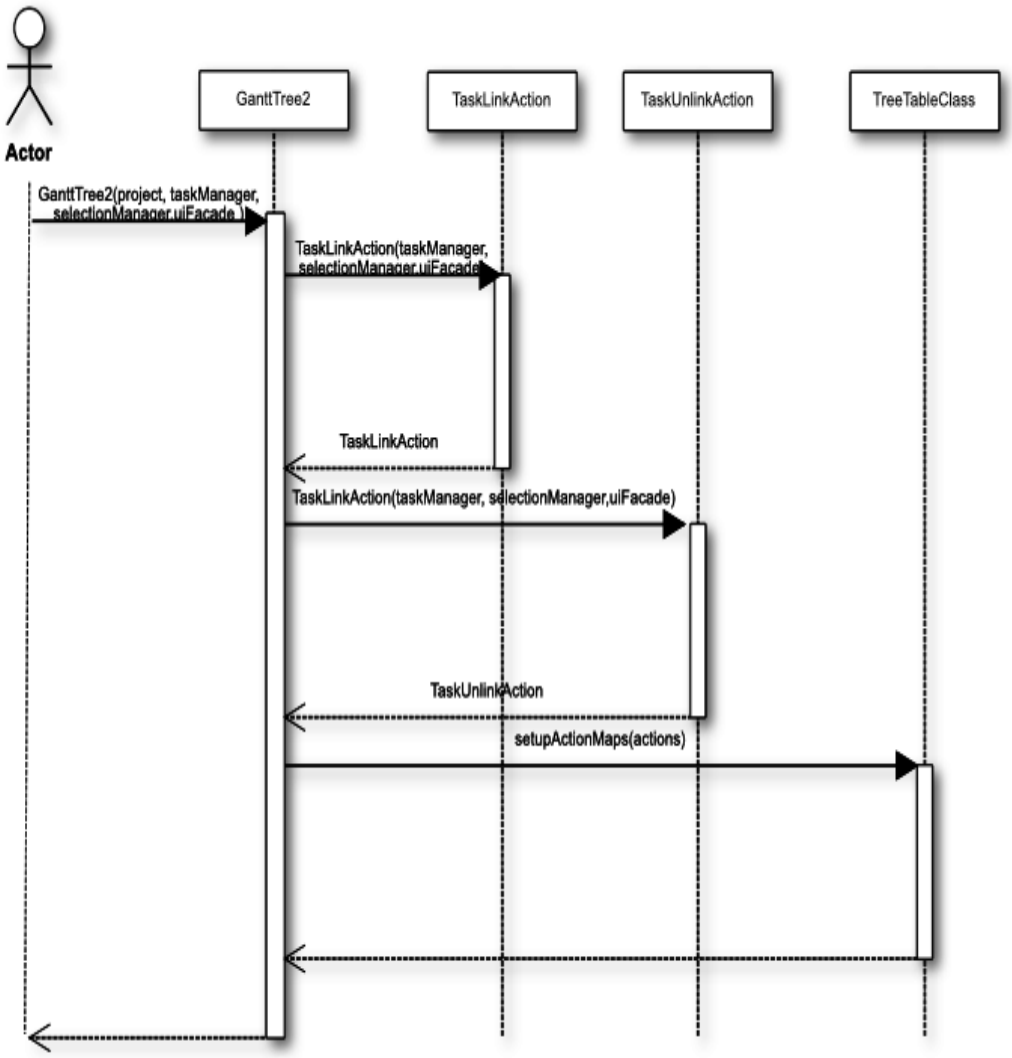

Figura 3.5: 1.Azpiproiektuaren sekuentzia-diagrama

3.4. Inplementazioa

Atal honetan, funtzionalitatearen inplementazioari buruzko xehetasunak azalduko dira.

Alde batetik, keyboard.properties fitxategian sartutako aldaketak daude. Kasu honetan, jada definituta zeuden task.link eta task.unlink ekintzak erabiliko dira. Ekintza hauei Ctrl + L eta Ctrl + U komandoak esleituko zaizkie, 3.6 irudian ikus daitekeen moduan.

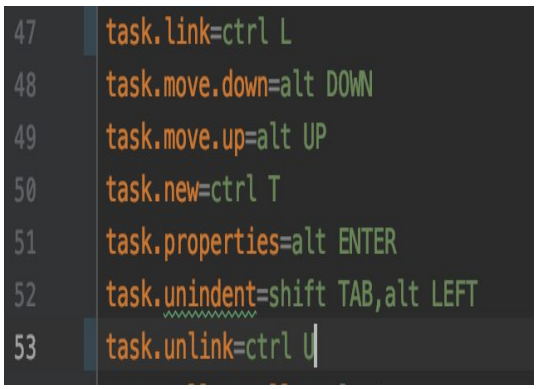

Figura 3.6: Keyboard.properties eraldatua

Beste aldetik, GanttTree2 klasearen eraikitzailean egin beharreko aldaketa daukagu. Aplikazioaren instalazioa eta konfigurazioa amaitzean 3.7 irudian ikusten dugun moduan zegoen metodoa inplementatuta.

```
161     // Create Actions
162     GPAction propertiesAction = new TaskPropertiesAction(project.getProject(), selectionManager, uiFacade);
163     GPAction deleteAction = new TaskDeleteAction(taskManager, selectionManager, uiFacade, tree: this);
164     GPAction newAction = new TaskNewAction(project.getProject(), uiFacade);
165
166     setArtefactActions(newAction, propertiesAction, deleteAction);
167
168     myIndentAction = new TaskIndentAction(taskManager, selectionManager, uiFacade, tree: this);
169     myUnindentAction = new TaskUnindentAction(taskManager, selectionManager, uiFacade, tree: this);
170     myMoveUpAction = new TaskMoveUpAction(taskManager, selectionManager, uiFacade, tree: this);
171     myMoveDownAction = new TaskMoveDownAction(taskManager, selectionManager, uiFacade, tree: this);
172     getTreeTable().setupActionMaps(myMoveUpAction, myMoveDownAction,
173             myIndentAction, myUnindentAction, newAction, myProject.getCutAction(), myProject.getCopyAction(),
174             myProject.getPasteAction(), propertiesAction, deleteAction);
175 }
```

Figura 3.7: GanttTree2 eraikitzailea hasieran

3.8 irudian GanttTree2 eraikitzailea ikus daiteke inplementatutako alda-

ketekin. Ikus daitekeenez, TaskLinkAction eta TaskUnlinkAction objektuen instantzia bana sortzen da.

Behin ekintza guztien instantziak izanda, GanttTree2-k duen TreeTableClass-en instantzia hartzen da getter baten bidez. Gero, azken honek duen setupActionMaps() metodoari egiten saio deia aurreko ekintza guztiak esleitzeko. Honen bidez, aipatutako bi komandoak edozein momentutan erabiltzeko prest egongo dira.

```
161     // Create Actions
162     GPAction propertiesAction = new TaskPropertiesAction(project.getProject(), selectionManager, uiFacade);
163     GPAction deleteAction = new TaskDeleteAction(taskManager, selectionManager, uiFacade,  tree: this);
164     GPAction newAction = new TaskNewAction(project.getProject(), uiFacade);
165
166     setArtefactActions(newAction, propertiesAction, deleteAction);
167     myLinkTasksAction = new TaskLinkAction(taskManager, selectionManager, uiFacade);
168     myUnlinkTasksAction = new TaskUnlinkAction(taskManager, selectionManager, uiFacade);
169     myIndentAction = new TaskIndentAction(taskManager, selectionManager, uiFacade,  tree: this);
170     myUnindentAction = new TaskUnindentAction(taskManager, selectionManager, uiFacade,  tree: this);
171     myMoveUpAction = new TaskMoveUpAction(taskManager, selectionManager, uiFacade,  tree: this);
172     myMoveDownAction = new TaskMoveDownAction(taskManager, selectionManager, uiFacade,  tree: this);
173     getTreeTable().setupActionMaps(myLinkTasksAction, myUnlinkTasksAction, myMoveUpAction, myMoveDownAction,
174             myIndentAction, myUnindentAction, newAction, myProject.getCutAction(), myProject.getCopyAction(),
175             myProject.getPasteAction(), propertiesAction, deleteAction);
```

Figura 3.8: GanttTree2 eraikitzailea eraldatua

3.5. Testa

Atal honetan, 1.Azpiproiektuan garatutako funtzionalitate berria nola testatu den azalduko da.

Alde batetik, Gradle tresna erabili da egindako aldaketak testatzeko. Tresna honen bitartez aplikazioa konpilatzen da eta eraikitzen da. Konpilazioa eta eraikitzea ondo joan badira(ikusi 3.9 irudia), inplementatutako aldaketak izan ditzakeen akats lexikoak baztertu daitezke. Beraz, erreminta honen bidez, garatutako funtzionalitatea konpilatzen duen edo ez jakingo da.

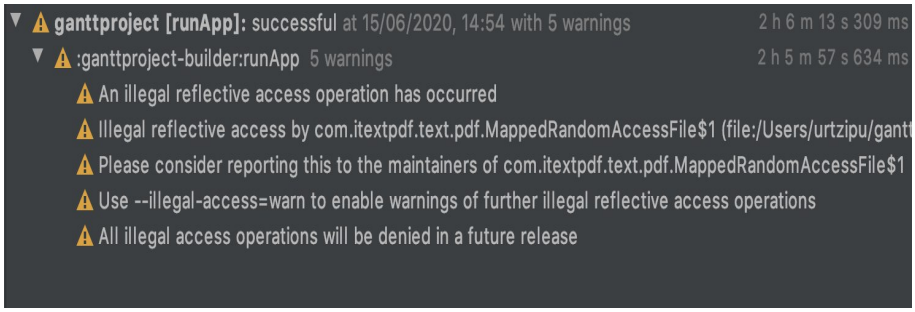

Figura 3.9: Gradle tresnak ematen duen emaitza

Behin aplikazioa konpilatzen dela jakinda, aplikazioan sartutako funtzionalitatearen eskuzko probak egingo dira. Hauen bidez, egindako aldaketak hasiera batean espero zena egiten dutela testatuko da.

Kasu honetan, eskuzko probak egiteko hainbat zeregin desberdin sortu dira, eta Ctrl + L eta Ctrl + U komandoak erabiliz hain artean loturak era zuzenean sortu eta ezabatu dira.

3.6. Ondorioak

Azken atal honetan, 1.Azpiproiektua bukatzean aurkitu diren ondorioak azalduko dira.

Behin funtzionalitatea inplementatuta eta testatuta , GanttProject-eko errepositoriora igo dira aldaketak. Bertan aplikazioaren garatzaileek aldaketak ikus eta aztertu ditzakete, eta zuzen iruditzen bazaie aplikazio ofizialean sartu dezakete funtzionalitatea.

Aldaketak beraiekin partekatzeko Pull Request bat egin behar da, eta garatzaileek egindako Pull Request-a aztertzen dutenean feedback-a bidaltzen dute.

Inplementatutako funtzionalitatearen kasuan, Pull Request-a aplikazioaren garatzaile batek onartu du, eta 3.10 irudian ikus daitekeen mezua jaso da.

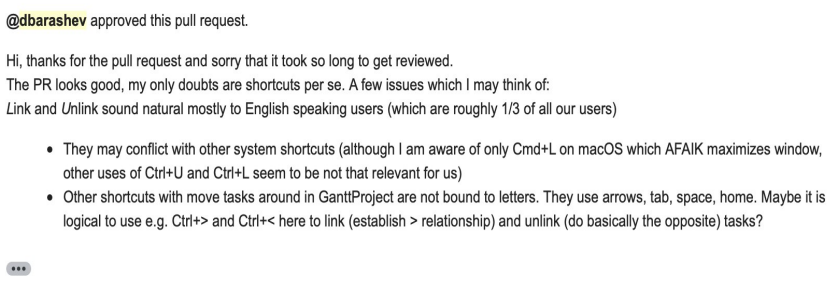

Figura 3.10: Aplikazioaren garatzailearen erantzuna

Mezuan garatzaileak aldaketa bat proposatzen du, Ctrl + L eta Ctrl + U erabili ordez, Ctrl + > eta Ctrl + < erabiltzea. Izan ere, L eta U letrak Link eta Unlink hitzetatik datoz, eta berak mezuan aipatzen duen bezala, soilik erabiltzaileen heren bat ingelesez hitz egiten du. Gainera, macOS edo bestelako sistema eragileek horrelako komandoak erabiltzen dituzte ere. Beraz, garatzailearekin kontaktuan jarrita proposatutako aldaketa hori egitea adostu genuen. Azkenean, aldaketa hori garatzaileak berak egin zuen. Hurrengo estekan emaitza ikus daiteke Github-en: `https://github.com/bardsoftware/ganttproject/pull/1697` .

4. 2.Azpiproiektua

Atal honetan 2.Azpiproiektuaren garapena eta horretarako egindako zereginen azalpena egingo da.

Aurretik esan denez, azpiproiektu honen helburua aplikazioak duen akatsen bat zuzentzea da. Horretarako, erabiltzaileek aplikazioarekin lan egitean aurkitzen dituzten akatsen artean bat hautatuko da, eta bere soluzioa garatuko da.

2.Azpiproiektuaren garapenari buruzko azalpena aurrera eramateko 6 azpiatal nagusitan bananduko da: Akatsaren aukeraketa, akatsaren analisia, soluzioaren diseinua, soluzioaren inplementazioa, testa eta ondorioak.

4.1. Akatsaren aukeraketa

Akatsaren zuzenketaren garapena egin baino lehen, zein akatsarekin lan egingo den definitu behar da. Horretarako, 1.Azpiproiektuan aipatutako Github-eko Issues atalera bueltatuko gara.

Github-eko errepositorioan egonda, hainbat akats desberdin aztertu dira eta zerrenda bat osatu da. Zerrenda hori egin eta gero, proiektuaren zuzendariarekin bilera egin da, bertan akats bakoitzaren azterketa egin da bien artean. Azterketa horretan, akats bakoitza garatzeko beharrezko denbora eta, batez ere, zailtasuna estimatu dira. Izan ere, ez da akats oso erraz bat zuzendu nahi, ezta akats konplexuena ere.

Beraz, bilera amaituta 4.1 irudian ikus daitekeen Issue-a garatuko dela ondorioztatu da.

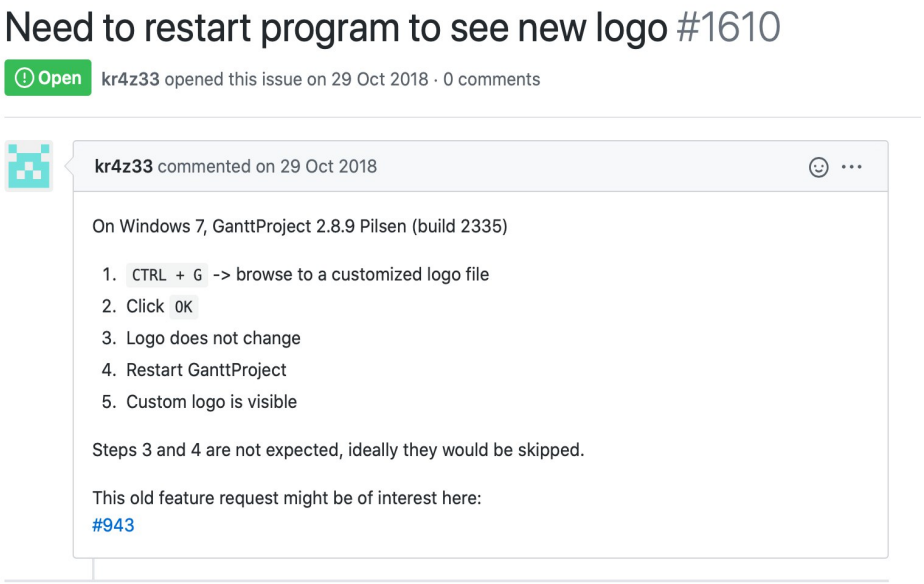

Figura 4.1: 2.Azpiproiektuan konponduko den akatsa Github-en

Aurreko azpiproiektuan bezala, akatsaren berri eman duen erabiltzaileak azalpen txiki bat uzten du. Kasu honetan, aplikazioak egiten duen prozesua zerrendatzen du, baita aplikazioak era zuzen batean egin beharko lukeen prozesua ere.

4.2. Akatsaren analisia

Atal honetan, aurreko atalean aukeratutako akatsaren azalpena eta akats horretan eragina duten klaseen identifikazioa egingo da.

Aurretik aipatu den moduan, GanttProject proiektuak kudeatzeko aplikazioa da. Aplikazio honen barnean Ctrl + G komandoa erabiltzen badugu proiektuaren konfigurazioa kudeatzeko pantaila bat irekitzen da, 4.1 irudian ikus daiteke. Bertan, proiektuaren ikurra aldatzeko aukera ematen da, beste askoren artean.

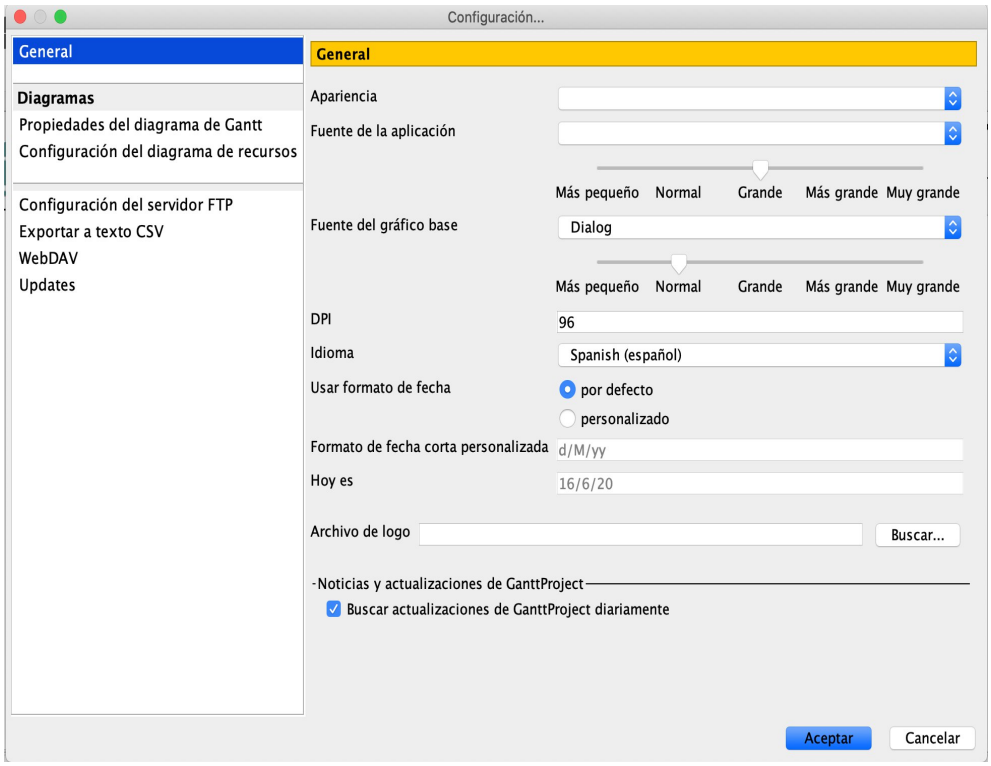

Figura 4.2: GanttProject aplikazioak proiektuen konfigurazioa egiteko eskaintzen duen pantaila

Akats honen ondorioz, ikur berri bat jartzean eta pantaila nagusira buelltatzean proiektuaren ikurra ez dela aldatu ikus daiteke,4.3 irudian agertzen den bezala. Baina, aplikazioa ixten bada eta berriro irekitzen bada ikurra aldatu dela konprobatu daiteke,4.4 irudian agertzen dena ikusten da .

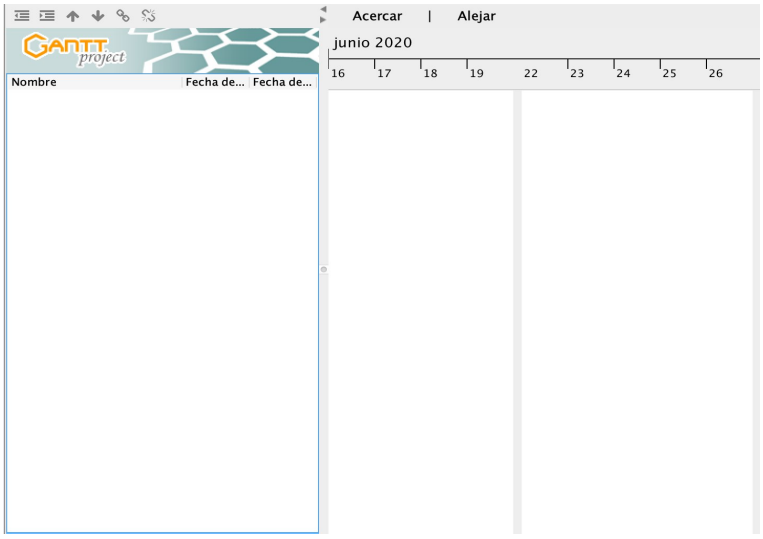

Figura 4.3: GanttProject proiektu bat ikurra aldatuta daukana berrabiarazi gabe

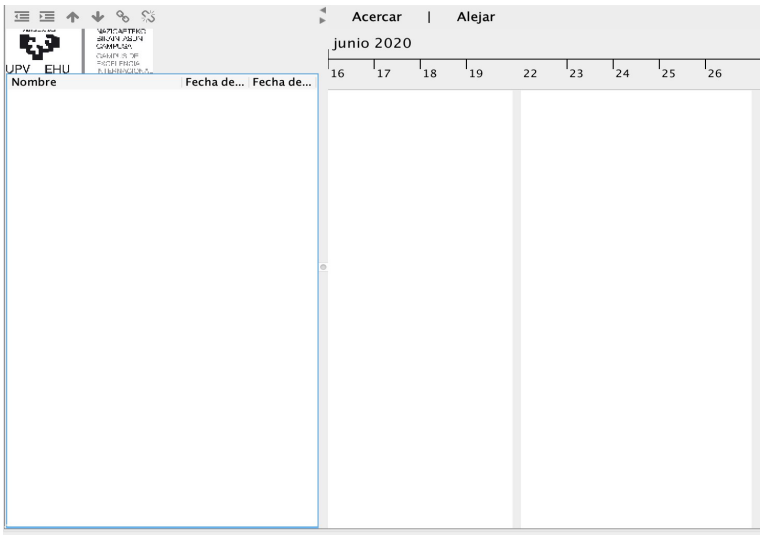

Figura 4.4: GanttProject proiektu bat, ikurra aldatuta daukana, aplikazioa berrabiarazi ostean

Beraz, hori izango da gure azpiproiektu honen garapenaren helburua, proiektu baten ikurra aldatzen bada konfigurazio pantaila ixtean zuzenean agertzea, aplikazioa berrabiarazi gabe.

Behin hau eginda, kodearen azterketa egin beharra dago garapenean eragina duten klaseak identifikatzeko. Azterketa hau egiteko berriz ere aurretik aipatu dugun debugger-a erabiliko da. Tresna honen bitartez hurrengo klaseak eragina dutela ikusi da:

- **UIFacadeImpl**
- **GanttTree2**
- **GanttProject**
- **ChartTabContentPanel**
- **UIFacade**

4.3. Soluzioaren diseinua

Atal honetan akatsaren soluzioa garatzeko klase-diagramaren eta sekuentzia-diagramaren diseinua irudikatu eta azalduko da.

4.3.1. Klase-diagrama

Aurretik esan denez, klase-diagrama sistema baten egitura azaltzeko erabiltzen da. 4.6 irudian 2.Azpiproiektuaren garapena aurrera eramateko diseinatu den klase-diagrama ikus daiteke. Horretarako, aurreko atalean identifikatu diren klaseak erabili dira. Ondoren, diagramaren objektu bakoitzaren deskribapena emango da.

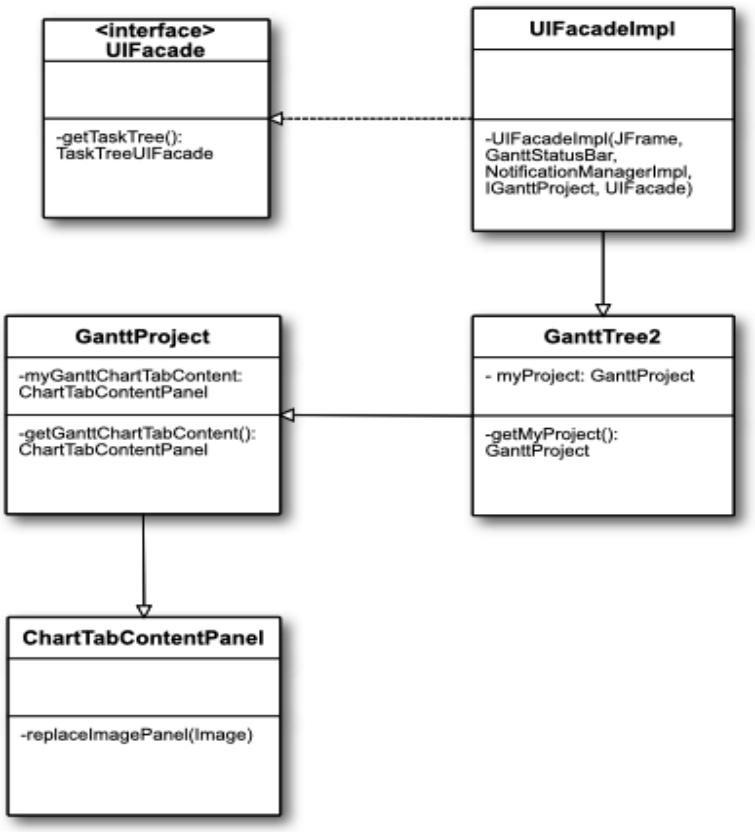

Figura 4.5: 2.Azpiproiektuaren klase-diagrama

- **UIFacadeImpl**: UIFacade klasea inplementatzen du, pantaila nagusiaren objektuak kudeatzeaz arduratzen da: hizkuntza, letraren tamaina, ikurra, eta abar.
- **GanttTree2**: TreeTableContainer klase abstraktua inplementatzen duen klasea da. Hainbat ekintza burutzen ditu, baina azpiproiektuaren garapenarekin lotuta dagoena GanttProject objektuaren instantzia lortzea ikurra aldatzeko da.
- **GanttProject**: Aplikazioaren Frame nagusia kudeatzen duen klasea, honen bidez, ikurra barnean duen panela lortuko da.
- **ChartTabContentPanel**: Ikurraren laukia kudeatzeaz arduratzen den klasea da, honen bidez ikurra pantailaratuko da.
- **UIFacade**: Pantaila nagusia sortzeko erabiltzen den interfazea da.

4.3.2. Sekuentzia-diagrama

Aurretik aipatu den moduan, sekuentzia-diagrama sistema baten objektuek hain artean egiten dituzten ekintzak grafiko batean adierazten duen diagrama da. 4.6 irudian akatsa konpontzeko garatutako inplementazioak bete behar dituen interakzio guztiak ikus daitezke.

Sekuentzia-diagraman UIFacade objektuaren instantzia berri bat sortzen den bakoitzean ikurraren aldaketa zuzena egiteko egin behar dituen ekintzak adierazten dira.

UIFacade objektuaren instantzia berria sortzen denean, beste ekintza askoren artean, ikurraren panelari listener bat esleitzen dio. Horren bidez, ikurra aldatzen denean egin beharreko prozesua adierazten da.

Ikurra aldatzen bada, GanttTree2 objektuaren instantzia lortu behar da getter baten bidez. Instantzia honek berdina egingo du GanttProject objektuarekin, eta azken honek ere, prozesua errepikatuko du ChartTabContentPanel objektuarekin.

Azken objektuaren instantziaren bidez replaceImagePanel() metodoari deia egingo zaio, erabiltzaileak ezarri duen ikur berria pasatuz. Honek, ikur berria esleituko du.

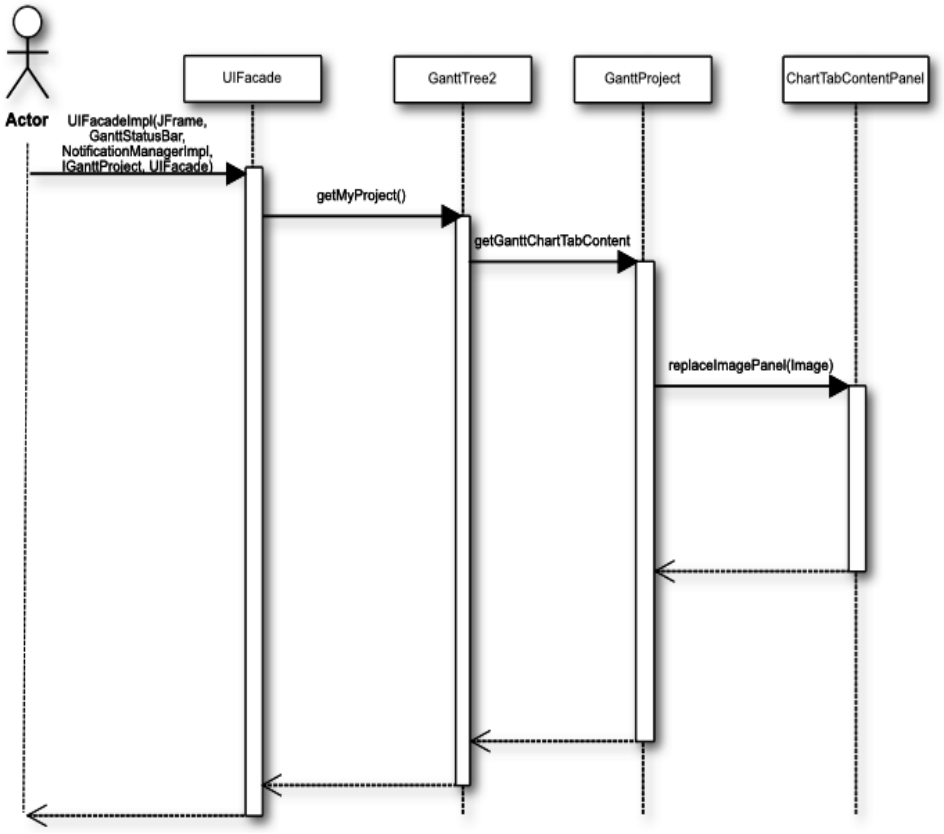

Figura 4.6: 2.Azpiproiektuaren klase-diagrama

4.4. Soluzioaren inplementazioa

Atal honetan, akatsarentzako diseinatutako soluzioaren inplementazioari buruzko azalpena eta xehetasunak emango dira.

2.Azpiproiektu honen garapenean inplementatutako akatsaren soluzioa ikusi baino lehen, metodoa hasiera batean nola inplementatuta zegoen ikus dezakegu 4.7 irudian. Bertan, ikurrarentzat fitxategi lehenetsi bat esleitzen zaiola ikus dezakegu, eta ez du inolako listener-rik aldaketa gertatzekotan argazkia eguneratzeko.

```
myLogoOption = new DefaultFileOption( id: "ui.logo");

myLogoOptions = new GPOptionGroup( id: "ui2", myLogoOption);

myLogoOptions.setTitled(false);

addOptions(myOptions);
addOptions(myLogoOptions);
```

Figura 4.7: UIFacadeImpl eraikitzailea hasieran

Beraz, gure garapenaren lehenengo pausua ikurraren aukerarentzako listener bat sortzea izango da, zehazki, ChangeValueListener motakoa. Honen bidez, ikurraren fitxategia aldatzen denean ChangeValueEvent motako event bat sortuko da. Event hau sortzen denean, listener-aren barruan inplementatu den changeValue() metodoa exekutatuko da.

```
myLogoOption.addChangeValueListener(new ChangeValueListener(){
  @Override
  public void changeValue(ChangeValueEvent event) {
    // Update date sample
    //if (dateFormatSwitchOption.isChecked()) {
    // ... update default date format option
        String selected = myLogoOption.getValue();
        shortDateFormatOption.setSelectedLocale(selected);
    Image newLogo =getLogo();
    //setIconImage(newLogo);
    //refresh;
  }
});
```

Figura 4.8: UIFacadeImpl eraikitzailea hasieran

4.8 irudian listener-aren inplementazioa ikus daiteke. Momentuz, listener-ak soilik irudi berria hartzen du getter baten bidez, baina ez du ikurra aplikazioan aldatzen.

Irudia aplikazioan agertzeko listener-aren bitartez ChartTabContentPanel klaseak gordetzen duen myImagePanel elementua beharrezkoa da. Elementu honek setIcon() metodoa eskaintzen du ikurra aldatzeko.

Beraz, hurrengo pausua ChartTabContentPanel klasearen barnean aurreko elementua erabiliz ikurra aldatzen duen metodo bat sortu behar da. 4.9 irudian inplementatutako replaceImagePanel() metodoa ikus daiteke.

```
62      }
63    });
64  }
65  void replaceImagePanel(Image image){
66    myImagePanel.setIcon(new ImageIcon(image.getScaledInstance(-1, 50, Image.SCALE_DEFAULT)));
67  }
68
69  JComponent createContentComponent() {
70    JPanel tabContentPanel = new JPanel(new BorderLayout());
71    final JPanel left = new JPanel(new BorderLayout());
72    final Box treeHeader = Box.createVerticalBox();
```

Figura 4.9: ChartTabContentPanel klasearen replaceImagePanel() metodoa

Behin ikurra aldatzeko metodoa eginda eta ikurra aldatu behar denean aldaketa gertatzeko listener-a inplementatuta izanda, listener-aren barruan

replaceImagePanel() metodoari deia egin behar zaio.

Dei hau UIFacadeImpl klasetik egin ahal izateko, GanttTree2 klasean GanttProject instantzia eskuratzeko getter-a inplementatu behar da, baita GanttProject klasean ChartTabContentPanel instantzia eskuratzeko ere. 4.10 irudian GanttTree2 klasean GanttProject-entzako inplementatutako getter-a ikus daiteke eta 4.11 irudian GanttProject klasean inplementatutako ChartTabContentPanel-entzako getter-a.

```
175             myProject.getPasteAction(), propertiesAction, deleteAction);
176   }
177   public GanttProject getMyProject() {
178     return myProject;
179   }
180
181   @Override
182   protected void init() {
183     getTreeTable().initTreeTable();
```

Figura 4.10: GanttProject-entzako getter-a

```
192   private FXSearchUi mySearchUi;
193
194   public GanttChartTabContentPanel getGanttChartTabContent() {
195     return myGanttChartTabContent;
196   }
197
198   public GanttProject(boolean isOnlyViewer) {
199     System.err.println("Creating main frame...");
```

Figura 4.11: ChartTabContentPanel-entzako getter-a

Behin aurreko inplementazio pauso guztiak emanda, akatsaren konponketarekin bukatzeko listener-aren barnean inplementatzen den changeValue() metodoaren inplementazioa egingo da.

```
238     myLogoOption = new DefaultFileOption("ui.logo");
239     myLogoOption.addChangeValueListener(new ChangeValueListener(){
240       @Override
241       public void changeValue(ChangeValueEvent event) {
242
243         if (event.getOldValue()!=null && event.getOldValue().equals(event.getNewValue())){
244           Image newLogo =getLogo();
245
246
247           ((GanttTree2)myFallbackDelegate.getTaskTree()).
248                   getMyProject()
249                   .getGanttChartTabContent().
250                   replaceImagePanel(newLogo);
251         }
252
253       }
254     });
255
256     myLogoOptions = new GPOptionGroup("ui2", myLogoOption);
257     myLogoOptions.setTitled(false);
258     addOptions(myOptions);
259     addOptions(myLogoOptions);
260   }
```

Figura 4.12: Listener-aren inplementazio finala

4.10 irudian listener-ak izan duen inplementazio finala ikus daiteke, bertan aurreko getter guztiak eta inplementatutako replaceImagePanel() erabili dira. Inplementazio honekin irudia era zuzenean aldatzea lortu da, aplikazioa berrabiarazi barik.

4.5. Testa

Atal honetan, 2.Azpiproiektuan akatsarentzako emandako konponbidea nola probatu den azalduko da.

Azpiproiektu hau testatzeko 1.Azpiproiektuarekin bezala bi proba nagusi egon dira: Gradle tresnaren konpilaketa eta eskuzko probak.

Alde batetik, Gradle tresnarekin aplikazioa konpilatu eta eraiki egingo da. Honen bidez, egin ahal izan diren akats lexikoak aurkitu daitezke. Proba hau ondo badoa, aplikazioa erabili dezakegu eta eskuzko testa egitera pasatuko da.

Eskuzko proben bidez, aplikazioak espero den funtzionamendua jarraitzen duen edo ez testatuko da. Kasu honetarako, proiektuaren ikurra hainbat aldiz aldatu da, eta aplikazioa berrabiarazi barik ikur berria duela konprobatu behar da.

Eskuzko probei esker, konponketak zuen akats bat aurkitu zen. Ikur berria ezartzean, ikurraren dimentsioak aldatzen ziren eta ikurraren zati bat handiagotuta ikusten zen.

Akats hau replaceImagePanel() metodoak setIcon() metodoari pasatzen dizkion parametroak aldatuz konpondu da. Aldaketa hau egin eta gero, aplikazioak bi testak zuzen pasatu ditu.

4.6. Ondorioak

Azken atal honetan, 2.Azpiproiektua bukatzearen ondorioak azalduko dira.

Behin akatsaren konponketa garatuta eta testatuta dagoela, GanttProject-eko errepositoriora igo dira egindako aldaketak. Pull Request baten bidez, aplikazioaren garatzaileekin partekatzen da inplementazioa eta beraiek aztertzen dute.

Inplementatutako soluzioaren kasuan, Pull Request-a aplikazioaren garatzaile batek aztertu eta gero ez du onartu. Bidaltzen duen erantzunean getter berri askoren erabilera dela eta ez duela onartzen aipatzen du, 4.13 irudian garatzailearen erantzuna ikus daiteke .

In ganttproject/src/net/sourceforge/ganttproject/UIFacadeImpl.java:

```
> @@ -247,6 +236,23 @@ public void changeValue(ChangeValueEvent event) {
      myOptions.setTitled(false);

      myLogoOption = new DefaultFileOption("ui.logo");
+     myLogoOption.addChangeValueListener(new ChangeValueListener(){
+       @Override
+       public void changeValue(ChangeValueEvent event) {
+
+         if (event.getOldValue()!=null && event.getOldValue().equals(event.getNewValue())){
+           Image newLogo =getLogo();
+
+
+           ((GanttTree2)myFallbackDelegate.getTaskTree()).
```

I don't think that we want to expose that many internal fields and do all these casts to concrete classes and call chains.

Figura 4.13: Garatzailearen erantzuna

Garatzaileak jarritako arazoa azpiproiektu honen garapenean zehar kontuan hartu zen gai bat da. Behin soluzioa inplementatuta zegoela getter gutxiagoren erabilera egitea saiatzea izan zen zeregin bat. Hainbat saiakera egin eta gero, getter hauek beharrezkoak zirela ondorioztatu zen.

5. 3.Azpiproiektua

Atal honetan 3.Azpiproiektuaren garapena eta egindako zereginen azalpena egingo da.

Aurretik azaldu denez, azpiproiektu honen helburua kode refaktorizazioa egitea da, hau da, aplikazioak dituen erabili gabeko metodo eta klaseen identifikazioa eta ezabaketa. Honen bidez, kodearen garbiketa eta aplikazioaren tamaina txikitzea lortu nahi da.

Azpiproiektu honen garapena azaltzeko 3 azpiatal nagusi daude: Kodearen analisi eta inplementazioa, testak eta ondorioak.

5.1. Kodearen analisi eta inplementazioa

Atal honetan, kodearen refaktorizazioa egiteko kodearen analisiari eta inplementazioari buruzko xehetasunak emango dira.

Kodearen erabili gabeko metodo eta klaseak ezabatu ahal izateko aurretik hauen identifikazioa egin behar da. Horretarako, IntelliJ IDEA-k eskaintzen duen tresna erabiliko da. Tresna hau Run Inspection By Name aukeran sakatuta erabili daiteke.

Behin tresna irekita, 5.1 irudian ikus daitekeenez hainbat aukera ematen dira kodearen analisia egiteko. Proiektu honetan, Unused declaration aukera erabiliko da, zehazki, Java Declaration redundancy aukera.

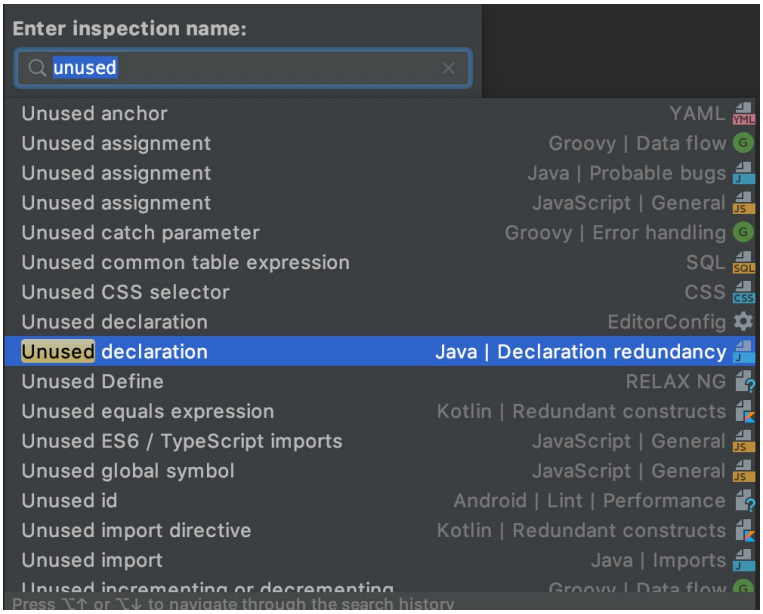

Figura 5.1: Run Inspection By Name tresnak ematen dituen aukerak

Behin aipatutako aukera hautatuta, analisiarentzako konfigurazioa erabaki behar da. 5.2 irudian tresnak ematen dizkigun konfigurazio aukerak ikus daitezke.

Konfigurazio pantaila horretan, lehenik eta behin, zein fitxategietan egin nahi den analisia adierazi behar da, kasu honetan proiektu osoan egin da bilaketa.

Gero, zein dira aurkitu nahi diren elementuak eta publikoak ala pribatuak diren adierazi behar da. Proiektu honetarako elementu guztiak aukeratu dira eta publiko aukerarekin lan egin da.

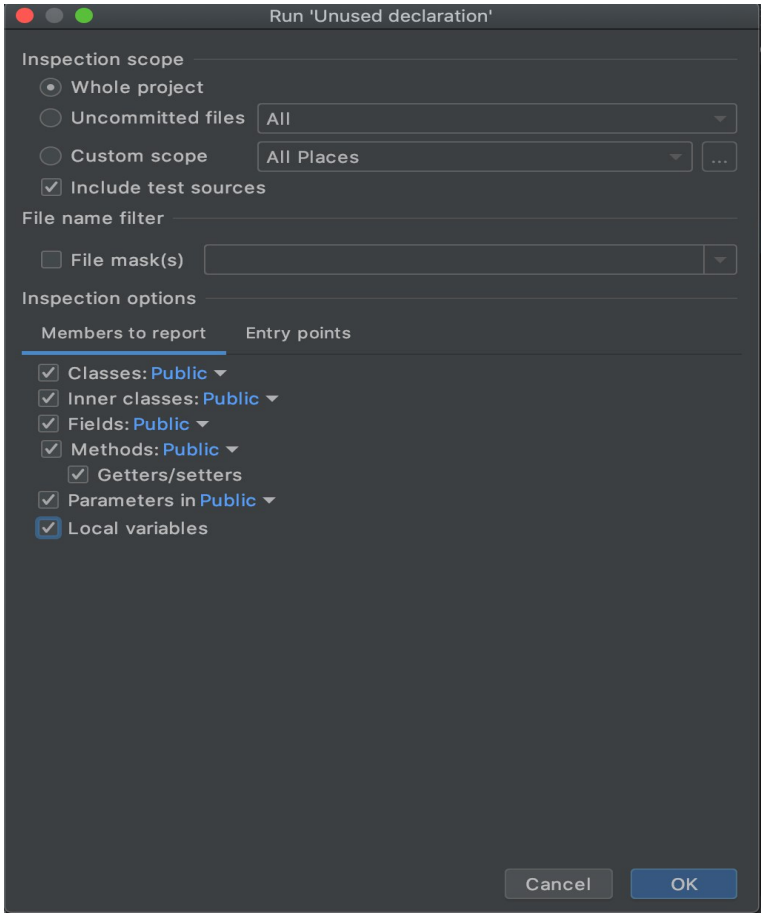

Figura 5.2: Analisiarentzako konfigurazio pantaila

Konfigurazioa egin eta gero, aplikazioa analisia egiten hasiko da eta amaitzean 5.3 irudian ikus daitekeen emaitza lortuko da. Emaitzan ikus daitekeenez klase asko daude erabiltzen ez diren metodoak, parametro edo atributuak dituztenak.

Zerrenda hau lortu denean, inplementazioaren atala hasten da. Metodo eta klaseak ezabatu baino lehen, proiektuan eragina izan dezaketen edo ez kontuan izan behar da. Gerta daiteke metodo bat erabilia ez izatea baina metodoa ezabatzearen ondorioz aplikazioak ez konpilatzea.

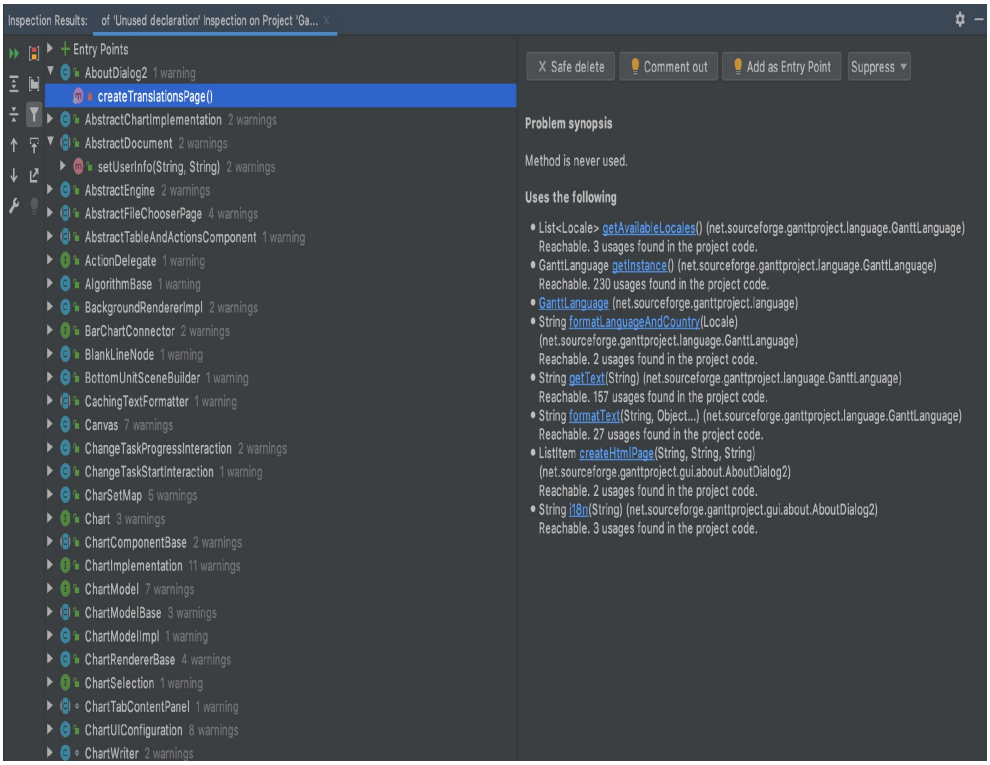

Figura 5.3: Analisiaren emaitza

Hau guztia kontuan izanda, ezabatu daitezkeen lau klase aurkitu eta ezabatu dira: DateUtils, ImporterFactory, PanelBorder eta SearchUiImpl. Horretaz aparte, ChartModelImpl klaseak erabili gabe zituen hiru metodo ere kodetik kendu dira: setExplicitlyHiddenTasks(), getTaskContainment() eta isExplicitlyHidden().

5.2. Testak

Atal honetan 3.Azpiproiektuaren kodearen refaktorizazioa nola testatu den azalduko da.

Azpiproiektu hau testatzeko, aurreko bi azpiproiektuekin bezala, Gradle bidezko konpilaketa eta eskuzko probak egin dira.

Alde batetik, Gradle tresna erabiliz aplikazioa konpilatuko da. Honen bidez, metodo edo klaseen ezabaketa zuzena izan den edo ez detektatu daiteke. Oker ezabatutako klase edo metodoren bat aurkitzekotan Gradle konpilaketak erroreren bat bueltatuko du.

Gradle-n bidez ezabatutako metodo batekin egon den arazoa identifikatu da. Ezabatutako metodoa ez da inoiz erabiltzen, baina metodoa duen klaseak interfaze bat inplementatzen du eta interfazeak metodo hori duenez, klaseak metodoaren inplementazioa izatea beharrezkoa da.

Eskuzko proben bitartez, aplikazioak funtzionamendu zuzena mantentzen duela testatu da, horretarako hainbat eragiketa desberdin probatu dira: zereginak sortu, loturak sortu, konfigurazioa aldatu, etab.

5.3. Ondorioak

Azken atal honetan, 3.Azpiproiektuaren ondorioak azalduko dira.

Behin kodearen refaktorizazioa eginda eta testatuta dagoela, Github-eko errepositoriora igo dira aldaketak. Aurretik aipatu den Pull Request bat erabiliz, inplementatutako kode aldaketak aplikazioaren garatzaileekin partekatu dira haiek aztertzeko.

Pull Request-a garatzaile batek aztertu eta gero, aldaketak onartu eta errepositoriora igo ditu. 5.3 irudian garatzaileak bidaltzen duen erantzuna ikus daiteke, aldaketak onartuz eta egindako lana eskertuz.

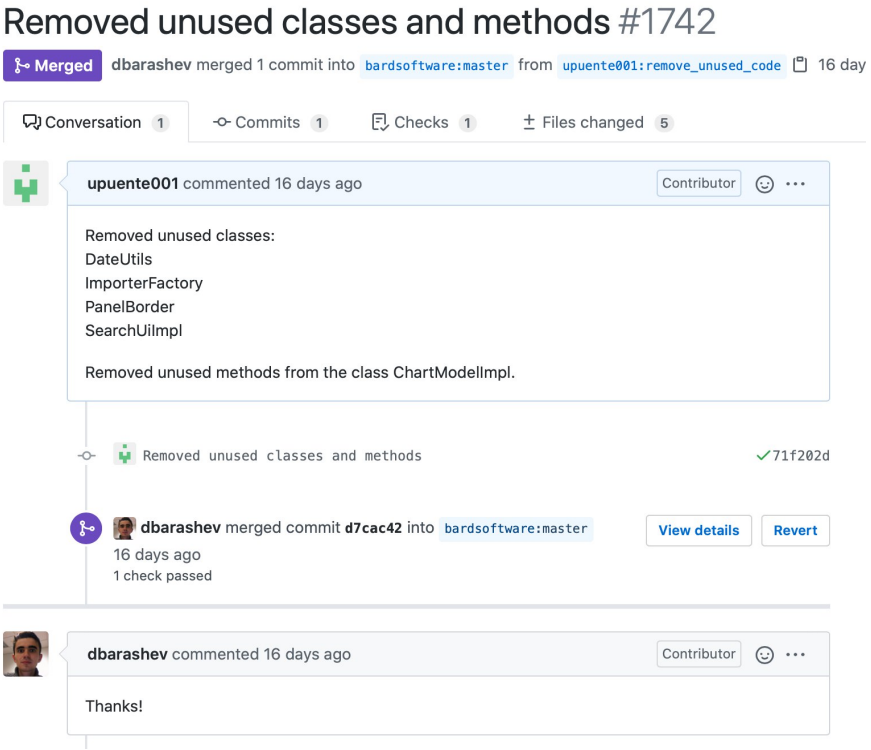

Figura 5.4: 3.Azpiproiektuko garatzailearen erantzuna

6. Ondorioak

Azken atal honetan, proiektuaren ondorioak aipatu eta azalduko dira. Bertan egindako planifikazioaren estimazio eta benetan gertatutakoaren arteko konparaketa bat egingo da. Horrez gain, eskerrak emateko atal bat ere idatziko da.

6.1. Planifikazio estimatuaren eta errealaren arteko konparaketa

Ondoren, proiektuaren irismenean egin behar izandako aldaketak azalduko dira, baita honen ondorioz egin den berrebaluazio ekonomikoa ere.

6.1.1. Proiektuaren irismenean egindako aldaketak

Proiektuaren irismena egitean sortutako bloke nagusi guztiak mantendu egin dira, baina bloke hauen barnean hainbat aldaketa egin dira. Ondoren bloke bakoitza banaka aztertuko da:

6.1.1.1. Ikasketa

Bloke honetan ataza berri bat gehitu da, izan ere, proiektuan zehar Github-ekin lan asko egin da. Alde batetik, proiektuan egindako aldaketak gorde ahal izateko erabili da. Bestalde, proiektu honetan burututako azpiproiektuak garatzaileekin partekatzeko eta hauek aldaketak aplikazioan sartzeko beharrezkoa izan da. Beraz, Git eta bere komandoak ikasteko ataza berri bat sortzea erabaki da.

Git eta bere komandoen ikasketa
Lan blokea: Ikasketa. **Ustezko iraupena**: 6 ordu.
Deskribapena: Git erreminta eta honek eskaintzen dituen komandoen ikasketa. **Irteera/Entregagaiak**: Ez dago. **Beharrezko baliabideak**: Interneterako konexioa. Github.

Tabla 6.1: Git eta bere komandoen ikasketa

6.1.1.2. Antolaketa

Bloke honetan ez da inolako aldaketarik egin. Ez da ataza berririk gehitu eta denbora erreala eta estimatuaren arteko diferentzia oso txikia izan da.

6.1.1.3. 1.Azpiproiektua

Bloke honetan bi ataza gehitu dira. Alde batetik, 1.Azpiproiektua garatzeko funtzionalitate berri baten inplementazioa egiteko Github-en beste erabiltzaileek proposatzen zituzten funtzionalitateak aztertu, zerrendatu eta baten aukeraketa egin behar izan da.

Funtzionalitate desberdinak zerrendatu eta aztertu
Lan blokea: 1.Azpiproiektua. **Ustezko iraupena**: 4 ordu.
Deskribapena: Github-en dauden funtzionalitate berrien proposamenak aztertu, zerrendatu eta bat aukeratu. **Irteera/Entregagaiak**: Ez dago. **Beharrezko baliabideak**: Github.

Tabla 6.2: Funtzionalitate desberdinak zerrendatu eta aztertu

Bestalde, behin inplementazioa bukatuta eta testatuta dagoela, aldaketak Github-era igo eta garatzaileekin partekatu dira. Honetarako ere, ataza berri bat sortzea erabaki da.

Funtzionalitatea garatzaileekin partekatu
Lan blokea: 1.Azpiproiektua. **Ustezko iraupena**: 2 ordu.
Deskribapena: Inplementatu eta testatu den funtzionalitatea Github-era igo eta garatzaileekin partekatu. **Irteera/Entregagaiak**: Ez dago. **Beharrezko baliabideak**: Github.

Tabla 6.3: Funtzionalitatea garatzaileekin partekatu

6.1.1.4. 2.Azpiproiektua

Bloke honetan, aurrekoan bezala, bi ataza berri gehitu dira. Alde batetik, 2.Azpiproiektuan garatzen den akats baten konponketa egin ahal izateko, Github-en beste erabiltzaileek edo garatzaileek proposatzen dituzten akats desberdinen konponketa aztertu, zerrendatu eta horietako bat aukeratu.

Akats desberdinak aztertu eta zerrendatu
Lan blokea: 2.Azpiproiektua. **Ustezko iraupena**: 4 ordu.
Deskribapena: Github-en dauden akats konponketak aztertu, zerrendatu eta bat aukeratu. **Irteera/Entregagaiak**: Ez dago. **Beharrezko baliabideak**: Github.

Tabla 6.4: Akats desberdinak aztertu eta zerrendatu

Bestalde, aurreko atalean aipatu den bezala, inplementatu eta testatu den soluzioa garatzaileekin Github-en partekatzeko ataza berria sortu da.

Akatsaren konponketa garatzaileekin partekatu
Lan blokea: 2.Azpiproiektua. **Ustezko iraupena**: 2 ordu.
Deskribapena: Inplementatu den soluzioa Github-era igo eta garatzaileekin partekatu. **Irteera/Entregagaiak**: Ez dago. **Beharrezko baliabideak**: Github.

Tabla 6.5: Akatsaren konponketa garatzaileekin partekatu

6.1.1.5. 3.Azpiproiektua

Bloke honetan proiektuan zehar ataza bat gehitu da. Aurreko kasuetan bezala, inplementatu eta testatu den kode refaktorizazioa Github-era igo behar da garatzaileek aztertu ahal izateko. Horretarako, aurretik planifikatuta ez zegoen ataza berri bat beharrezkoa izan da.

Kode refaktorizazioa garatzaileekin partekatu
Lan blokea: 3.Azpiproiektua. **Ustezko iraupena**: 2 ordu.
Deskribapena: Inplementatu eta testatu den kode refaktorizazioa Github-era igo eta garatzaileekin partekatu. **Irteera/Entregagaiak**: Ez dago. **Beharrezko baliabideak**: Github.

Tabla 6.6: Kode refaktorizazioa garatzaileekin partekatu

6.1.1.6. Dokumentazioa

Azken bloke honetan, planifikatuta ez zegoen beste ataza bat gehitu behar izan da, izan ere, behin dokumentazioa eginda dagoela, dokumentazioaren berrikuspen bat egin behar da. Honen bidez, dokumentazioak izan ditzakeen akatsak aurkituko dira.

Dokumentazioaren berrikuspena
Lan blokea: Dokumentazioa. **Ustezko iraupena**: 5 ordu.
Deskribapena: Proiektuaren defentsa egiteko behar diren diapositibak sortzea. **Irteera/Entregagaiak**: Ez dago. **Beharrezko baliabideak**: Overleaf.

Tabla 6.7: Dokumentazioaren berrikuspena

6.1.2. Planifikazio erreala

Jarraian proiektuaren zeregin guztiak egin ondoren zereginen planifikazioaren benetako emaitza azalduko da, ustezko emaitzarekin konparatuta. Horretarako, zeregin bakoitzaren ustezko iraupena eta benetako iraupena bananduko dira.

ID	***Atazaren izena***	***Benetako iraupena***	***Ustezko iraupena***
	GanttProject	505 ordu	460 ordu
1	***Ikasketa***	119 ordu	110 ordu
1.1	Java eta Gradle ikasketa	12 ordu	15 ordu
1.2	GanttProject aplikazioaren ikasketa eta konfigurazioa	70 ordu	60 ordu
1.3	Sdkman ikasketa	6 ordu	10 ordu
1.4	Latex ikasketa	25 ordu	25 ordu
1.5	Github eta bere komandoen ikasketa	6 ordu	
2	***Antolaketa***	41 ordu	42 ordu
2.1	Azpiproiektuen antolaketa	8 ordu	6 ordu
2.2	Zereginen planifikazioa	6 ordu	6 ordu
2.3	Software instalazioa	12 ordu	15 ordu
2.4	Proiektuaren zuzendariarekin bilerak	15 ordu	15 ordu
3	***1.Azpiproiektua***	62 ordu	60 ordu
3.1	Funtzionalitatearen analisia	20 ordu	20 ordu
3.2	Funtzionalitatearen diseinua	24 ordu	20 ordu
3.3	Funtzionalitatearen inplementazioa	8 ordu	10 ordu
3.4	Funtzionalitatearen testa	10 ordu	10 ordu
3.5	Funtzionalitate desberdinak zerrendatu eta aztertu	4 ordu	
3.6	Funtzionalitatea garatzaileekin partekatu	2 ordu	

Tabla 6.8: Zereginen iraupenak(1)

ID	*Atazaren izena*	*Benetako iraupena*	*Ustezko iraupena*
4	***2.Azpiproiektua***	116 ordu	100 ordu
4.1	Arazoaren analisia	30 ordu	40 ordu
4.2	Soluzioaren diseinua	20 ordu	20 ordu
4.3	Soluzioaren inplementazioa	50 ordu	30 ordu
4.4	Soluzioaren testa	10 ordu	10 ordu
4.5	Akats desberdinak zerrendatu eta aztertu	4 ordu	
4.6	Akatsaren konponketa garatzaileekin partekatu	2 ordu	
5	***3.Azpiproiektua***	42 ordu	40 ordu
5.1	Analisia	20 ordu	20 ordu
5.2	Inplementazioa	10 ordu	10 ordu
5.3	Testak	10 ordu	10 ordu
5.4	Kode refaktorizazioa garatzaileekin partekatu	2 ordu	
6	***Dokumentazioa***	125 ordu	108 ordu
6.1	Informazio bilketa	8 ordu	8 ordu
6.2	Taulak eta diagramak egitea	12 ordu	10 ordu
6.3	Memoria idaztea	90 ordu	80 ordu
6.4	GAL-aren defentsa	10 ordu	10 ordu
6.5	Dokumentazioaren berrikuspena	5 ordu	

Tabla 6.9: Zereginen iraupenak(2)

6.1.3. Berrebaluazio ekonomikoa

Atal honetan, 2.6 atalean egindako ebaluazio ekonomikoaren estimazioa benetakoarekin konparatuko da.

6.8 taulan ikus daitekeenez, proiektua garatzeko guztira 505 ordu egin dira lan. Berrebaluazioa egiteko orduan, estimazioa egin denean bezala, ikasketari bideratutako orduak kenduko dira. Kalkulu hori egiten bada:

$$Orduak\ guztira\ -\ (Ikasketa\ orduak\ -\ GanttProject\ zeregina) = \quad (6.1)$$

$$= 505\ -\ (119\ -\ 70) = 456\ ordu \quad (6.2)$$

2.6 atalean jada kalkulatuta dagoen moduan, programatzaile batek lan-ordu bakoitzeko 9,26 € irabazten ditu.

Beraz, proiektuaren benetako kostua kalkulatzeko hurrengo ekuazioa erabiltzen da:

$$Orduak\ guztira\ *\ Kostua\ orduro\ = 456 * 9,26 = 4222,56\,e \quad (6.3)$$

Hau horrela izanda ebaluazio ekonomikoaren taula eguneratua hurrengoa da:

Gastua	***Ustezko diru kantitate***	***Diru kantitate erreala***
Eskulana	3796,60 €	4222,56 €
Software gastua	0 €	0 €
Hardware gastua	225 €	225 €
Zeharkako gastuak	425,70 €	425,70 €
Guztira	4447,30 €	4873,26 €

Tabla 6.10: Ebaluazio ekonomiko finala

Taulan ikus daitekeen moduan, proiektuaren kostu erreala 4873,26 €-koa izan da, estimatutakoa baino 425,96 € handiagoa.

6.2. Azpiproiektuen ondorioen laburpena

Atal honetan,proiektuaren laburpen gisa, azpiproiektu bakoitzean garatutako kode zatiak onartuak izan diren edo ez eta zergatik adieraziko dira taula baten bidez. Horrez gain, azpiproiektu bakoitzaren emaitzaren Github esteka adieraziko da.

Azpiproiektua	***Zeregina***	***Onartua?***	***Zergatik?***	***Emaitzaren esteka***
1	Funtzionalitate berria	Bai		`https://github.com/bardsoftware/ganttproject/pull/1697`
2	Akats konponketa	Ez	Getter askoren erabilera	`https://github.com/bardsoftware/ganttproject/pull/1723`
3	Kode refaktorizazioa	Bai		`https://github.com/bardsoftware/ganttproject/pull/1742`

Tabla 6.11: Ebaluazio ekonomiko finala

6.3. Esker emateak

Memoriari bukaera emateko, proiektuaren garapenean era zuzenean laguntza eskaini didan pertsonari eskerrak ematea falta zait.

Proiektuaren tutore eta zuzendari bezala eta Bilboko Ingeniaritza Eskolan informatika sailean irakasle moduan lan egiten duen Juanan Pereira eskertu nahi nuke, proiektu hau aurrera eramateko aurkitu ditudan zailtasunak ulertzen eta hauei soluzioa aurkitzen lagundu dit. Horretaz aparte, bere bulegora joateko beharra nuenean beti egon da prest, eta Skype nahiz Telegram bitartez ere beti prest egon da behar izan ditudan arazoekin laguntzeko. Azkenik, laguntza eskatu diodan bakoitzean erantzuna emateko bizkortasuna ere eskertu nahiko nioke.

7. Glosategia

- **Debugger**: Programazioan, aplikazioak testatzeko eta arazteko erabiltzen den tresna da. Tresna honen bidez, aplikazioak dituen erroreak aurkitu eta konpondu daitezke. Horretaz aparte, tresna honen bidez kodearen exekuzioa puntu desberdinetan geldi daiteke honen azterketa egin ahal izateko.
- **Errepositorio**: Aplikazioaren kodeak gordetzeko eta banatzeko biltegia da.
- **Event**: Java programazio lengoaian, erabiltzaileak ekintza bat burutzen duenean sortzen den instantzia edo objektua. Adibidez, erabiltzaileak botoi baten gainean click egiten duenean ActionEvent bat sortzen da.
- **Feedback**: Proiektu honen kasuan, garatzaileek gure proposamenari emandako erantzuna deritzo.
- **Frame**: Java programazio lengoaian, objektuak biltzen dituen leihoa da. Objektu hauek botoiak, irudiak, testu koadroak, eta abar izan daitezke.
- **Getter**: Programazioan, objektu baten instantzia edo atributu baten balioa lortzeko erabiltzen den metodoa.
- **Integratutako garapen ingurunea**: Software garapena errazteko zerbitzuak eskaintzen dituen aplikazio informatikoa da. Normalean, kode editore bat, debugger bat eta eraikitzaile edo konpilatzaile automatiko batek osatzen dute.
- **Kode refaktorizazioa**: Kodea mantentzearen atal bat da, ez dira akatsak konpontzen ezta funtzionalitate berriak gehitzen. Honen helburua, erabiltzen ez den kodea ezabatzea eta kodearen ulermena erraztea da.
- **Konpilazioa**: Programatzaileek idatzitako kodea makina kodera itzultzeko prozesua da.

- **Listener**: Programazioan, programatzailea ekintza batez jakinarazteko erabiltzen den metodoa da. Metodo honi, Event bat pasatzen zaio gertaerari buruzko informazioa ematen diona.
- **Panel**: Java programazio lengoaian, osagaien edukiontzi bat da. Elementu honen barruan osagai desberdinak sar daitezke, adibidez, botoi bat.
- **Pull Request**: Egindako kode aldaketak finalak izan baino lehen, garatzaileek aldaketa horiek aztertzeko eta onartzeko egiten den eskaera. Kode aldaketak lokalean egiten dira eta gero urruneko proiektu batera pasatzeko erabiltzen da.

7. Webgrafia

[Enplegu eta Gizarte Segurantzaren Ministerioa, 2018] Enplegu eta Gizarte Segurantzaren Ministerioa (2018). Boletín oficial del estado - xvii convenio colectivo estatal de empresas de consultoría y estudios de mercado y de la opinión pública. Ikusi: https://www.boe.es/boe/dias/2018/03/06/pdfs/BOE-A-2018-3156.pdf.

[GAndroid, 2014] GAndroid (2014). ¿cuál es el consumo de un ordenador pc o portá- til? Ikusi: https://www.galaxyandroid.es/cual-es-el-consumo-de-un-ordenador-pc-o-portatil/.