****

**Xabier Lekunberri**

**Gontzal Pujana**

**Eneko Sampedro**

**Javier Sautua**

**Software Ingeniaritza**

**2016**



**Dokumentazio Teknikoa**



Pokésweeper

# EDUKIAK

[EDUKIAK 2](#_Toc451116812)

[SARRERA 3](#_Toc451116813)

[ANTOLAKUNTZA 3](#_Toc451116814)

[DISEINU OSOA 4](#_Toc451116815)

[Klase Diagrama 4](#_Toc451116816)

[Klaseen azalpena 4](#_Toc451116817)

[Sekuentzia Diagramak 10](#_Toc451116818)

[Sekuentzia diagramen azalpena 10](#_Toc451116819)

[GARAPENA/RETROSPEKTIBA 12](#_Toc451116820)

[Sprint 1 12](#_Toc451116821)

[Sprint 2 12](#_Toc451116822)

[Sprint 3 13](#_Toc451116823)

[EGINDAKO PROBAK 15](#_Toc451116824)

[ONDORIOAK 17](#_Toc451116825)

[ERANSKINAK 18](#_Toc451116826)

[Bibliografia eta estekak 18](#_Toc451116827)

# SARRERA

Kurtso honetan garatu beharreko proiektua Dragamina jokoa izan da. Gure kasuan, Dragamina joko honi “Pokésweeper” izena eman diogu, izan ere Pokémon jokoen gaiaren arabera garatu dugu. Pokésweeper jokalari bakarreko jokoa da, non jokalariak, minarik zapaldu gabe, tablero osoa zabaltzea lortu beharko du ahalik eta denbora gutxienean. Hau lortzeko 999 segundo izango ditu, eta 3 zailtasun mailen artean erabaki ahal izango du. Maila zenbat eta altuagoa izan, jokoaren zailtasuna hainbat eta handiagoa izango da.

# ANTOLAKUNTZA

Taldea antolatzeko modelo adaptatiboa erabili dugu, lider definitu baten beharra ez zegoelako gure taldean, nahiz eta momentu batzuetan norbaitek lana koordinatzen zuen zeregin bat aurrera eramateko, gure erlazio pertsonala nahiko hobea zelako.

Desadostasun bat geneukanean zeregin bat aurrera nola eramateari buruz, arazo hori konpontzeko modu desberdinak jartzen genituen, eta hitzarmenez hoberena iruditzen zitzaiguna hartzen genuen, honela denbora aurreztuz geldi ez egoteagatik .

Beraz, uste dugu modelo adaptatiboa hartzearen erabakia oso zuzena izan dela, arintasuna, produktibitatea eta efizientzia handiagoa lortu dugulako orokorrean.

Proiektuaren kudeaketa egiteko, *GitHub* plataforma eta *Scrum* metodoa konbinatzea pentsatu genuen. *GitHub*-ekin proiektuaren kontrol globala eraman dugu, eta *Scrum*-ekin zeregin bakoitzaren kontrola. GitHub erabiltzean ez dugu arazorik izan, jadanik beste proiektu batzuetan erabili dugulako. Scrum-ekin, aldiz, arazo batzuk izan genituen proiektuaren hasieran, *IceScrum* programa erabiltzerakoan, zerbitzariaren muntaketa egiteko eta bere erabilpena ulertzeko, bere dokumentazioa ez zelako ulergarria. Hori gertatuta ere, azkenean programa menderatu genuen eta sprint-ak bakoitza bere erabiltzaile istorioekin jarri genituen, proiektuaren kudeaketa errazagoa lortuz.

Bilerak egiteko orduan irakasgaiaren laborategiaz eta eskolan daukagun talde lanak egiteko geletaz baliatu gara. Ordu horietatik kanpo, TeamSpeak komunikazio programa erabili dugu berba egiteko, zalantzak argitzeko eta eztabaidatzeko. Programatzeko orduan TeamViewer softwarea erabilita, guztion artean ordenagailu bera erabili dugu, gauza konkretu batzuk egiteko.

#### IceScrum

IceScrum web bideratutako programak proiektu bat sprint-etan banatzen uzten du. Sprint bakoitzak erabiltzaile istorio batzuk ditu, eta erabiltzaile istorio horiek zeregin batzuk dituzte. Zeregin bakoitza 8 ordu baino gutxiago irauten duen lan bat izango da, bestela zeregin handiago bat bitan banatuko da. Honela lan metodo dinamiko bat bultzatzen da taldearen heinean.

# DISEINU OSOA

### Klase Diagrama

<https://github.com/Xabiier/PokeSweeper/tree/master/dokumentazioa/irudiak/klase%20diagramak>

### Klaseen azalpena

#### **Audioa**

Gure proiektuaren musika kudeatzeaz arduratzen den klasea da. 2 atributuz eta 3 metodoz osatuta dago.

//Atributoak

**private** **static** Media *md*: Musikaren artxiboa gordetzeko erabiltzen dugu.

**private** **static** MediaPlayer *player*: Gure musika erreproduzitzeko beharrezkoa dugun erreproduktorea da.

//Metodoak

**public** **void** play(): Musika martxan jartzen duen metodoa da.

**public** **static** **void** setVolume(**double** z): z parametroa hartuz (honen balioa 0tik 100era doa) musikaren bolumena aldatzen du.

**public** MediaPlayer getPlayer(): *player* atributuko erreproduktorea lortzeko balio du.

#### **BarruLaukia**

Lauki generiko bat errepresentatzen du. Herentziaren bitartez beste laukiak ateratzen dira klase honetatik.

//Atributoak

**protected** **int** errenkada: Tableroan laukiak daukan errenkada adierazten du.

**protected** **int** zutabea: Tableroan laukiak daukan zutabea adierazten du.

**protected** **int** egoera: Laukiaren egoera da. Lauki batek lau egoera izan ditzake: Zabaldu gabe, bandera jarrita, galdera ikurra eta laukia

**private** **int** ikonoZenb: Laukia 3x3-ko tablero batean egongo balitz zein zenbaki izango luken adierazteko. Honekin zein ikono-talde esleitu behar zaion dakigu.

//Beste metodoak

**public** **void** eskumakoBotoia(): Bandera eta galdera ikurra ikonoen artean aldatzen du.

**public** **void** ezkerrekoBotoia(): Metodo abstraktua da

**public** **void** mouseEntered(): Belarraren mugimendua abiarazten du.

**public** **void** mouseExited(): Belarra aurreko mugimendu gabeko ikonora bueltatzen du.

**public** **void** mousePressed(): Pika-ren ikonoa aldatzen du.

**public** **void** mouseReleased(): Pika egoera normalaren ikonora aldatzen du.

**protected** **boolean** koordenadaEgokiak(**int** pErrenkada, **int** pZutabe): Laukiaren koordenatuak egokiak diren ala ez begiratzen du.

**public** **int** getEgoera(): Laukiaren egoera itzuliko du.

**public** **int** getIkonoZenb(): Ikono zenbakia itzultzen du.

#### **BichilloLaukia**

Proiektuaren “easter egg”-a da. Gehiago jakin nahi baduzu aurki ezazu :P

//Metodoak

**public** **void** ezkerrekoBotoia(): Laukia zabalduko du, Pokemon baten irudia ezarriz.

**public** **void** eskumakoBotoia(): Ez du ezer egiten, BarruLaukiko metodoa gainidazteko da.

#### **DenborraKontadorea**

//Atributoak

**private** Timer timer; Erabiltzen den erlojua da.

//Metodoak

**protected** **void** kontadoreaEguneratu(): Timer-aren balio zatitzen du, kontadorearen UI-a eguneratzeko, eta denboraren arabera jokoa galdu den konprobatzen du.

**public** **void** denboraGelditu(): Denbora gelditzeko erabiltzen da.

**public** **void** denboraErreseteatu(): Kontadorea berriro 0-ra jartzeko erabiltzen da.

#### **Helbideak**

Jokoan zehar erabiliko diren irudi eta hauen helbideak kudeatuko dituen klasea da.

#### **IkonoKonbinaketa**

Klase honek bi irudi ipiniko ditu ImageIcon batean, bat aurrean eta bestea atzean ager daitezen.

//Atributoak

**private** **final** Icon atzekoa: Irudiaren atzealdean jarriko den ikonoa da.

**private** **final** Icon aurrekoa:  Irudiaren aurrealdean jarriko den ikonoa da.

//Metodoak

**public** **void** paintIcon(Component arg0, Graphics arg1, **int** arg2, **int** arg3): Bi irudiak marraztuko ditu, ezarritako ordenean.

#### **Kontadorea**

//Atributoak

**protected** **int** kont: Kontadorearen balio gordetzen duen atributua da.

//Metodoak

**protected** **void** kontadoreaEguneratu(): Metodo abstraktu bat da.

**public** **int** getKont(): Kontadorearen balioa itzultzen du.

#### **LaukiFaktory**

Factory patroia erabilita, jokoan zehar erabiliko ditugun lauki guztiak eratzeaz arduratuko den klasea da.

//Atributoak

**private** **static** LaukiFactory *nireFactory*: Singleton patroia.

**private** **int** errenkadaKop: Tableroaren errenkada kopurua gordetzen du.

**private** **int** zutabeKop: Tableroaren zutabe kopurua gordetzen du.

**private** **int** minaKop: Tableroaren mina kopurua gordetzen du.

**private** **int**[][] minak: Factory-ak ausaz beteko duen mina matrizea. Honen arabera, tableroko matrizearen posizio bakoitzari dagokion lauki mota itzultzen du.

//Metodoak

**public** BarruLaukia createLaukia(**int** pErrenkada, **int** pZutabe, **boolean** pBichilloak): minak matrizearen arabera lauki mota bat ala beste bat itzultzen du.

**public** **void** setTamaina(**int** pErrenkadaKop, **int** pZutabeKop, **int** pMinaKop): Tableroak duen tamaina eta mina kopurua ezartzen ditu, geroago minak matrizea sortzeko.

**private** **void** albokoakAbisatu(**int** pErrenkada, **int** pZutabe): Mina bat jartzen denean albokoen zenbakia unitate batean handitzen du, zenbaki lauki batean bihurtuz.

**private** **boolean** koordenadaEgokiak(**int** pErrenkada, **int** pZutabe): Koordenatuak egokiak diren konprobatzen du.

**private** **int** zenbakiaLortu(**int** pErrenkada, **int** pZutabe): Laukiari dagokion belar ikono zenbakia bueltatzen du.

#### **Login**

Erabiltzaileen logina kudeatuko duen (datuan .txt batean irakurri eta idatzi) klasea da.

//Atributoak

File fitxErabiltzaileak: Erabiltzaileak gordetzen dituen artxiboa da.

**private** **static** Login *nireErabiltzaileak*: Singleton patroia aplikatzeko atributua.

**private** String fitxIzena = "erabiltzaileak.txt": fitxErabiltzaileak atributuak izango duen fitxategiaren izena da, hasieran ez genuen jarri, baina JUnit-ak kudeatzeko beharrezkoa genuen.

//Metodoak

**public** **boolean** logeatu(String user, String pass): Pasatzen dizkiogun erabiltzaile eta pasahitza fitxategian batera dauden begiratzen du.

**private** **boolean** baDago(String user): Erabiltzailea fitxategian dagoen begiratzen du.

**public** **boolean** erabiltzaileaSortu(String user, String pass): Erabiltzailea fitxategian ez badago, erabiltzailea sortzen du.

#### **LurLaukia**

BarruLauki-tik heredatutako klasea da. Lauki normal bat da, hau da, mina edo zenbakirik ez dauka.

//Metodoak

**public** **void** ezkerrekoBotoia(): Laukiaren egoeraren arabera, posiblea bada, laukia zabaltzen du.

**private** **void** albokoakZabaldu(): Lur laukiak diren alboko laukiak zabaltzen ditu zenbaki laukietara heldu harte.

#### **MinaKontadorea**

Izenak ezaten duen bezala, mina kontadore bat da. Partidan geratzen diren mina kopurua adierazten du.

//Metodoak

**protected** **void** kontadoreaEguneratu(): Kontadorea tableroak jarritako banderen arabera eguneratzen du.

**public** **void** kontadoreaErreseteatu(): Kontadorea 0-ra jartzen du.

**public** **void** minaGehitu(): Kontadoreari +1 egiten dio.

**public** **void** minaKendu(): Kontadoreari -1 egiten dio.

**public** **void** minaKopuruaJarri(int pZenbakia): Kontadoreari mina kopurua ezartzen dio.

#### **MinaLaukia**

BarruLauki-tik heredatutako klasea da. Mina bat daukan laukia adierazten du.

//Beste metodoak

**public** **void** ezkerrekoBotoia(): Laukiaren egoeraren arabera, posiblea bada, laukia zabaltzen du.

**public** **void** bistaratu(): Egoeraren arabera mina ikono bat ala beste bat ezarriz bistaratzen du.

#### **Pika**

JButton bat izango da, jokoaren egoera irudien bidez azalduko duena eta honen berrerabiarazpena ahalbidetuko duena.

//Atributoak

**private** **static** Pika *nirePika*; Singleton patroia.

//Metodoa

**public** **void** ezkerrekoKlik(): Tableroa erreteatzen du eta Pika-ren ikonoa ikono normalera jartzen du.

**public** **void** eskumakoKlik(): Pokemon-ez betetako tablero bat sortzen du eta Pika-ren ikonoa ikono normalera jartzen du.

**public** **void** setPikaEgoera(String pEgoera): Egoeraren arabera Pika-ren ikonoa aldatzen du.

#### **PokeSweeper**

Jokoaren main klasea da. Hemendik kudeatzen dira bezte frame-ak.

//Atributoak

**private** **static** PokeSweeper *nirePokéSweeper*; Singleton patroia.

**final** JFXPanel jfx; Beharrezkoa da audioa funtzionatu ahal izateko.

**private** **int** sLekua = 10; Joko osoarentzako musika Slider-arentzako posizioa gordetzen du.

//Metodoak

**public** **void** hurrengoa(**int** pInt): Hurrengo Frame-a sortzen du, Splash, login ala jokoa.

**private** **void** splash(): Splash-a sortzen du.

**public** **void** login(): Logina sortzen du.

**private** **void** jokoa(): Jokoa eratzen du.

**private** **void** musikaJarri(): Musika hasieratu eta bukleak eratzen ditu.

**public** **void** sLekuaAldatu(**int** i): Musika Slider-arentzako metodoa da, gordeko den musika bolumenaren arabera Slider-aren posizioa aldatzen du..

**public** **int** getSLekua(): Gordetako slider-aren posizioa bueltatzen du.

#### **Ranking**

Rankig-a kudeatuko du datuan .txt batean irakurriz eta idatziz.

//Atributoak

File fitxRanking; Puntuazioak gordetzen dituen artxiboa da

**private** **static** Ranking *nireRanking*; Singleton patroia.

//Metodoak

**public** **void** idatzi(String pIzena, **int** pDenbora): Jokoaren maila lortu eta erabiltzailea maila horretan puntuaziorik ez badu, puntuazioa idatziko du.

**private** **boolean** baDago(String pIzena, **int** pDenbora, **int** pMaila): Erabiltzailea maila horretan puntuazioa badu eta orain lortutako puntuazioa hobea bada, puntuazioa eguneratzen du.

**public** String[] rankingLortu(**int** pMaila): Maila bateko 10 jokalari hoberenak bueltatzen ditu, errepikatutako erabiltzailerik gabe.

**private** **boolean** barruanDago(String[] izenak, String iz): 10 jokalari hoberenak lortzeko, jokalari horren puntuazio bat gorde dugun ala ez konprobatzen du.

#### **Tableroa**

Jokoaren tableroa izango, honen tamaina, egoera eta barnean izango dituen elementuak gordeko dituena.

//Atributoak

**private** **static** Tableroa *nireTableroa*; Singleton patroia

**private** BarruLaukia laukia[][]; Laukiz osatutako matrize bat

**private** ArrayList<MinaLaukia> minak; Mina laukiak gordetzen ditu, minak bistaratzeko orduan erabiltzen da.

**public static**  MinaKontadorea minaKontadorea; MIna kontadorea gordetzen du.

**public** **static** DenboraKontadorea denboraKontadorea; Denbora kontadorea gordetzen du.

**private** **int** errenkadaKop; Tableroaren errenkada kopurua

**private** **int** zutabeKop; Tableroaren zutabe kopurua

**private** **int** minaKop; Tableroaren mina kopurua

**private** **int** laukiFalta;  Zabaldu edo markatu gabe geratzen diren lauki kopurua.

//Metodoak

**public** **void** tableroaEraiki(**int** pErrenkada, **int** pZutabe, **int** pMinaKop, **boolean** pBichilloak): Pasatutako zutabe eta errenkada kopuruaren araberako tableroa sortuko du.

**public** **void** minakBistaratu(): Minak dauden laukiak ikusgarri bezala ipiniko ditu

**public** **int** getErrenkadaKop(): Errenakada kopurua itzuliko du

**public** **int** getZutabeKop(): Zutabe kopurua itzuliko du

**public** **int** getMinaKop(): Mina kopurua lortzeko

**public** **void** irabaziDu(): Partida irabazi den ala ez konprobatzen du.

**public** **void** minaGehitu(**MinaLaukia** pMinaLaukia); minak Array-ari mina lauki bat gehitzen dio.

#### **ZenbLaukia**

BarruLauki-tik heredatutako klasea da. Mina bat edo gehiagoren ondoan dagoen laukia da.

//Atributoak

**private** **final** **int** ingurukoMinaKop: Inguruan dituen mina kopurua gordetzen du.

//Metodoak

**public** **void** ezkerrekoBotoia(): Laukiaren egoeraren arabera, posiblea bada, laukia zabaltzen du.

#### **JokoaUI**

//Atributoak

**private** **static** JokoaUI *nirejokoa*; Singleton patroia.

**public** **static** **boolean** *bukatuta* = **false**; Partida bukatu den ala ez gordetzen du.

**private** **static** JPanel *goikoPanela*; Menuaren eta tableroaren arteko panela gordetzen du.

**private** **static** JPanel *behekoPanela*: Tableroa gordetzeko erabiltzen da.

**public** **static** KontadoreaUI *minaKontadorea*; Mina kontadorearen UI gordetzen du.

**public** **static** KontadoreaUI *denboraKontadorea*;  Denbora kontadorearen UI gordetzen du.

**public** **static** String *erabiltzailea*; Erabiltzailearen izena tableroan bistaratzeko erabiltzailearen izena gordetzen du.

**public** **static** **int** *maila*; Jokatzen ari garen zailtasun maila gordetzen du.

//Metodoak

**private** JPanel goikoPanelaEraiki(): Menuaren eta tableroaren arteko panela eraikitzen du.

**private** JPanel behekoPanelakEraiki(): Tableroa eta tableroa zentratzeko erabiltzen diren frameak sortzen ditu.

**private** JLabel getLorea(): Lore berri bat sortu eta itzuliko du

**public** **static** **void** galdu(): Jokoa galdutzat ezarriko du

**public** **static** **void** irabazi(): Jokoa irabazitzat ezarriko du

**private** **void** erreseteatu(**int** pErrenkada, **int** pZutabe, **int** pMinaKop, **boolean** pBichilloak): Egungo jokoa ezabatu, eta parametrotzat dituen aldagaiekin, beste joko bat eraikiko du

**public** **void** erreseteatu(**int** pMaila, **boolean** pBichilloak): Egungo tableroa ezabatu eta tablero berria eraikitzen du, kontadoreak hasieratuz.

**public** String getUserName(): Erabiltzailearen izena bultatzen du.

#### **pokesweeperui**

//Atributuak

**private** **static** PokéSweeperUI *nirePokéSweeperUI*: Singleton patroia.

//Metodoak

**public** **void** panelaAldatu(JPanel pPanela): Splash-a, login-a, eta jokua leiho berdinean agertzeko metodo honek JPanelak aldatzeaz.

### Sekuentzia Diagramak

<https://github.com/Xabiier/PokeSweeper/tree/master/dokumentazioa/irudiak/sekuentzia%20diagramak>

Sekuentzia eta klase diagramak oso handiak direnez eta dokumentu honetan sartzen ez direnez, gure Github-eko biltegian gorde ditugu.

### Sekuentzia diagramen azalpena

Exekutagarria zabaltzean, PokeSweeper klaseak lehenengo leihoa (Splash-a) zabalduko du. Leiho hau, jokoaren “Intro”-aren papera izango du. Bertan, jokoaren izena duen irudi bat zabalduko da eta, aldi berean, soinu bat exekutatuko da. Splash hau ixteko, beharrezkoa izango da erreproduzitutako soinua bukatzea eta, ondoren, teklatuko “enter”-a sakatzea.

Ondoren Login leihoa zabalduko da, joko osoan zehar erreproduzituko den musikarekin batera, non erabiltzaileak kontu berri bat sortu, logeatu eta lehenengo partida zein mailarekin hasi erabakitzeko aukera izango duen. Erabiltzaile berria bagara, erabiltzaile izena eta pasahitza sartu, eta erregistratu botoia sakatuko dugu. Login klaseak, sortutako erabiltzaileen .txt-an begiratuko du. Erabiltzailea aurretik sortua izan bada mezu bat aurkeztuko zaigu eta jokoan sartzeko beste erabiltzaile izen berri bat sartzea beharrezkoa izango da. Behin erabiltzaile eta pasahitza sortuak, eta maila aukeratuta dagoela, “Sartu” botoia sakatuta, Login klasea .txt-a begiratzera joango da. Erabiltzaile hori existitzen bada eta sartutako pasahitza erabiltzaile horrek duen pasahitzarekin bat badator, jokoa hasiko da.

Jokoan zehar, klase eta metodorik aipagarrienak:

·PokéSweeper hasieratu: Jokoaren assert guztiak kargatuko ditu memorian, audioa hasieratuko du, eta jokoaren Splash-a erakutsiko du.

·Tableroa eraiki: Errenkada, zutabe eta mina kopuruak ezarriko zaizkio. Mina kontadorea, mina balio horrekin hasieratuko du. Laukiak gordeko dituen matrizea eta minak egongo diren lekuak gordeko dituen arraylist bat hasieratuko ditu. Ondoren, tableroa eraikitzen hasteko, LaukiFactory-ari eskatuko dizkio behar dituen laukiak, eta hauen motak.

·Minak jarri: Ausaz koordenatu batzuk sortuko ditu, eta begiratu ea hauetan minarik ote dagoen. Baldin balego, beste koordenatu batzuk eratuko lituzke. Behin koordenatu on batzuk sortuta, mina ezarriko du lauki horretan, eta alboan dauden gelaxkei esango die mina bat dela.

·Laukiak sortu: Sartu zaizkion koordenatuen arabera, zenbaki bat lortuko du, laukiari irudi bat edo beste bat ezartzeko. Ondoren, sartutako parametroen arabera, lauki bat edo beste bat sortu du (BichilloLaukia, LurLaukia, MinaLaukia edo ZenbLaukia). MinaLaukiaren kasuan, mina bat gehitu dela esango dio tableroari.

·Ezkerreko klik (Mina bat duen lauki batean): Belarra den ikonoa desagerrarazi eta mina bat jarriko du, tableroaren minakBistaratu metodoari dei eginez eta mina guztiak bistaratzea eraginez.

·Ezkerreko klik (Mina bat EZ duen lauki batean): Belarra den ikonoa desagerrarazi eta lur ikono bat jarriko dio. Gero, jokoa irabazi duen konprobatuko du, eta honela ez bada eta gainera, lurLaukia bada, alboko laukiak zabalduko ditu, zenbakiak diren laukietara heldu arte. Alboko laukiak zabaltzea lauki bakoitzak duen ezkerrekoBotoia metodoaren bidez burutuko da.

·Minak bistaratu: Lauki bakoitzaren egoera begiratuko du, eta honen egoera 1 bada, voltorb\_s ikonoa aterako zaigu, bestela voltorb ikonoa. voltorb\_s ikonoak lauki horretan bandera bat jarrita zegoela adieraziko du, hau da, lauki horretan, jokatzen ari ginen bitartean, jarritako bandera baten bidez, bertan mina bat zegoela asmatu genuela; bestela, voltorb irudi arrunta agertuko da. Metodo honek partida galtzea ere eragingo du.

·Pikan ezkerreko klik: Tableroko mina kopurua 0 den begiratzen da. Kopuru hori 0 ez bada, JokoaUI-ko erreseteatu metodoari deitzen zaio. Hemen, tableroa interfazetik kendu egiten da eta berri bat sortzen da tamaina berdinarekin. Tablero berria pantailaratu ostean Pika-ren irudia normalera erreseteatzen da eta denbora kontadorea 0-ra jartzen da.

·Pika - Eskumako klik: Tableroa interfazetik kendu egiten da eta berri bat sortzen da tamaina berdinarekin, baina kasu honetan, gure “easter egg”-arekin. Tablero normal bat sortu beharrean “bichillo”-z osatutako bat egiten da. Hemen, ez daude mina, zenbaki edo lur lauki normalik, denak BichilloLaukiak dira.

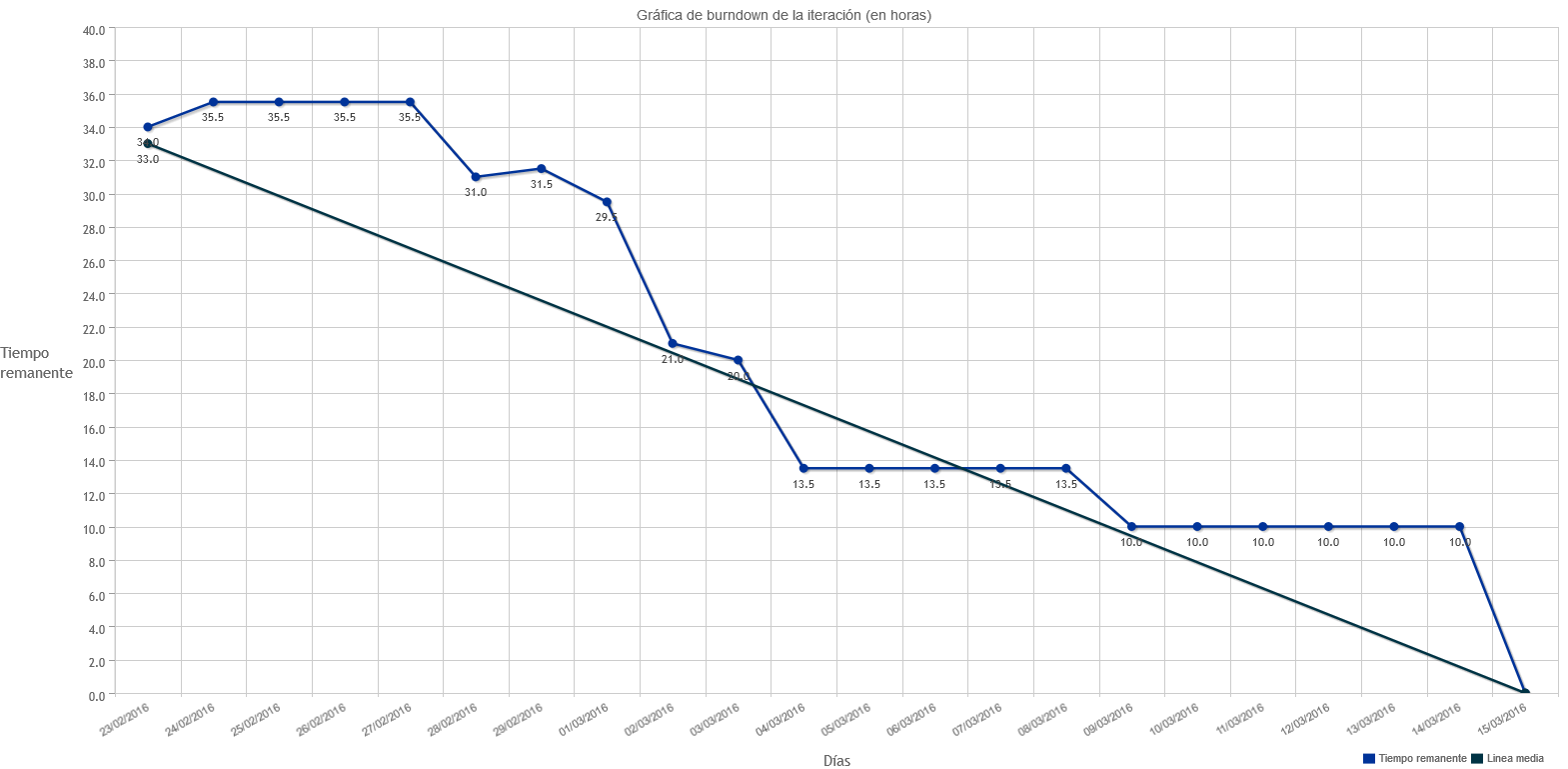
   ·Eskumako botoia (Laukia batean): Gelaxkak markatzeko botoia denez, laukia irekita dagoen begiratuko du. Egotekotan, ez luke ezer egingo. Bestela, bandera, galdera ikurra edo berezkoa den belar irudia ipiniko luke ikonotzat. Bandera bidez markatu ahal diren lauki kopurua, tableroaren mina kopuruaren berdina bada. Bandera guztiak agortuta, markatuko diren hurrengo laukiak galdera ikur batekin markatuko dira. Bandera egoeratik, galdera ikur egoerara pasatuko da. Eta azken egoera honetatik, markatu gabeko egoerara. Jarritako bandera bakoitzagatik mina kontadoreak -1 egingo du.

Puntuazioak begiratzerako orduan:

    Jokoan lortu diren puntuaziorik hoberenak erakusteaz Ranking klasea arduratuko da. Rankin klaseak, egin zaion maila zenbakiaren eskaeraren arabera, maila horretan puntuaziorik altuenak lortu dituzten 10 jokalarien datuak bueltatuko ditu. Puntuazio guztiak gordetzen diren .txt fitxategira bilatzera joango da. Bilaketa 10 aldiz burutuko da gehienez. Hasierako bueltan, erabiltzaileen izenak gordeko dituen array bat sortuko du eta ondoren, maila horretan puntuazio altuena duen jokalaria aukeratuko du, eta aldi berean, aurretik sortutako array-an gordeko du. Maila horretan puntuazioa duten jokalari gehiago badago, metodoa egikaritzen jarraituko du. Bigarren buelta honetan, puntuazio hoberena duen jokalaria aukeratzen jarraituko du, baina kasu honetan, erabiltzaileen array-aren bidez, aurretik aukeratu ez dugula zihurtatuko du.

# GARAPENA/RETROSPEKTIBA

### Sprint 1

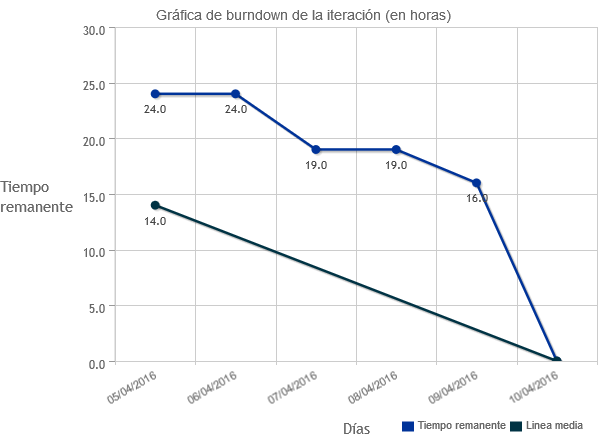


Helburuak: Tableroaren gelaxkak irekitzea eta aukeratutako mailako dragamina sortu eta pantailaratu.

Hartutako erabakiak:

* Lauki motak herentzia bidez kudeatuko dira.
* Laukiak JButton-ak izango dira.
* Lauki ezberdinak Factory baten bitartez sortuko dira.
* Zati logikoa eta interfaze grafikoa bananduak egongo dira.

### Sprint 2



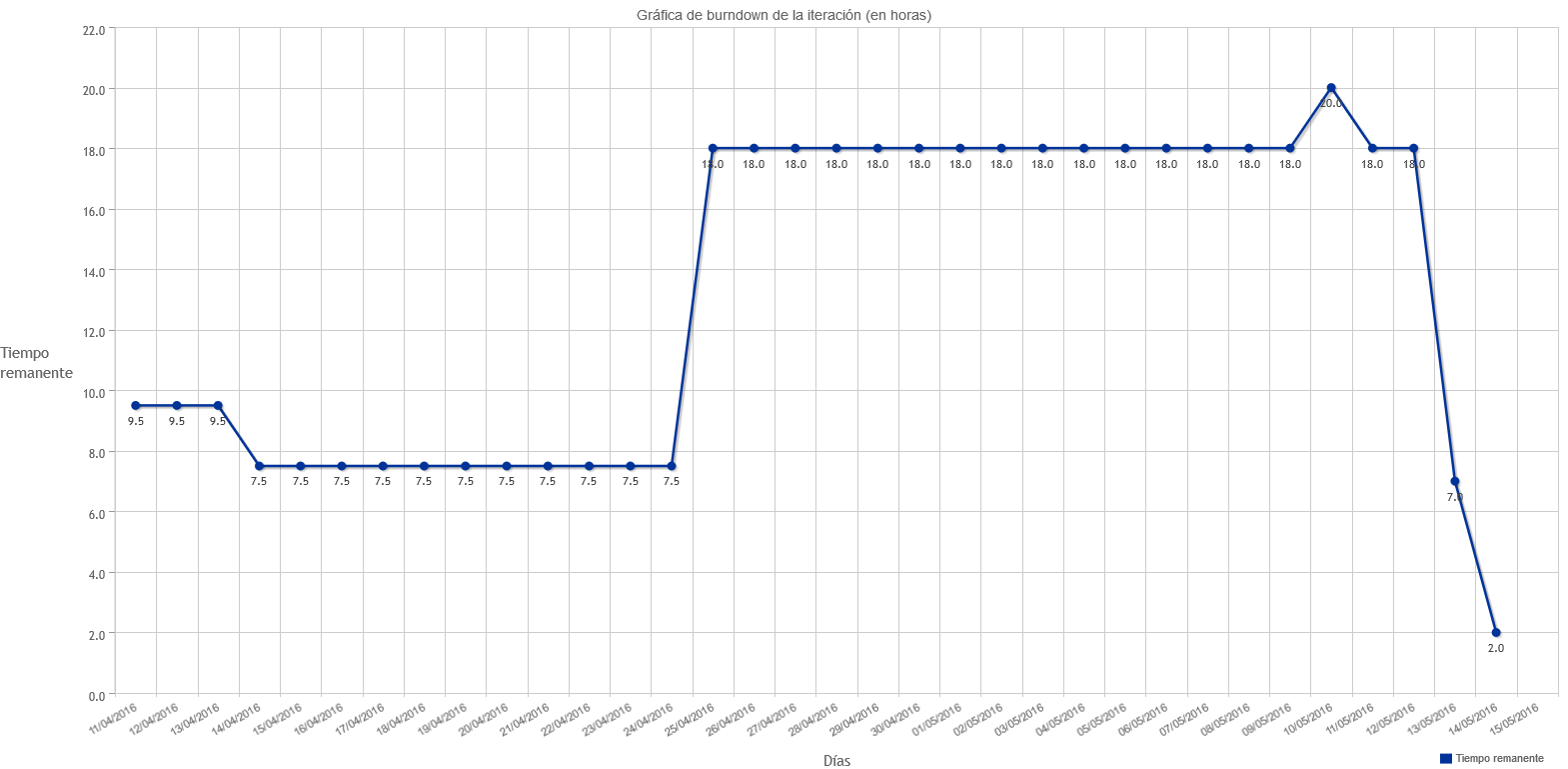
*\*Sprint-a aktibatzea ahaztu zitzaigun*

Helburuak: Gelaxkak markatu eta jokoaren zailtasuna aukeratu.

Hartutako erabakiak:

* Login bat egitea, jokalariak erregistratu daitezen.
* Login-a .txt batean gordeko da.

### Sprint 3



*\*Sprint-a eguneratzea ahaztu zitzaigun*

Helburuak: Jolasten jarraitzea (jokoa bukatzen denean), jokoa bertan behera utzi eta ranking bat pantailaratzea.

Hartutako erabakiak:

* Ranking-a .txt batean gordeko da.

# EGINDAKO PROBAK

1. JUnit bidezkoak: Hurrengoa konprobatu da.

* Tableroa klasea:
  + Tableroa era egokian osatzen dela, eskatutako tamainarekin eta mina kopuruarekin.
  + Minak era egokian bistaratzen direla.
  + Tableroak dituen errenkada kopurua eta zutabe kopurua era egokian ematen direla.
  + Jokoa irabazi dugun ala ez ondo konprobatzen duela.
* Login klasea:
  + Erabiltzaile berriak era egokian sortzen direla.
  + Existitzen den erabiltzaile eta pasahitz bat sartzerakoan logina kendu eta jokoa era egokian zabaltzen dela.
* LukiFactory klasea:
  + Eskatzen diogun laukia era egokian sortzen duela.
  + Eskatutako tamainaren arabera minak ausaz leku egokian kokatzen dituela.
* BarruLaukia klasea:
  + Ondo konprobatzen dela bere koordenatuak egokiak direla.
* BichilloLaukia klasea:
  + Ezkerreko botoia sakatzean Pokemon bat era egokian agertzen dela.
* Pika klasea:
  + Egoeraren arabera ikonoa era egokian aldatzen dela.
  + Eskumako eta ezkerreko klik egitean tableroa erreseteatzeko agindu egokia bidaltzen duela.
* MinaKontadorea klasea:
  + Tableroak dituen minak era egokian hartzen dituela.
  + Tableroan banderak ezartzean edo kentzean kontadorea era egokian eguneratzen dela.
  + Kontadorea era egokian erreseteatzen dela.
* DenboraKontadorea klasea:
  + Segunduak pasa ahala, kontadorea era egokoan eguneratzen dela.
  + Denbora bukatzerakoan jokoa galdu egiten dela.

1. Eskuz egindakoak: Hurrengoa konprobatu da.

* Audioa klasea:
  + Audioa era egokian erreproduzitzen dela eskatzen diogun momentu orotan.
  + Bolumena era egokian aldatzen dela hari lotutako slider-aren arabera eta aldi berean, slider-a eguneraturik geratzen dela.
  + Egindako bukleak era egokian erreproduzitzen direla.
* BarruLukia klasea:
  + Eskumako klik egiterakoan, laukiaren egoeraren arabera, dagokion ikonoa era egokian ezartzen duela eta eguneratu beharreko elementuetarako deia era egokian exekutatzen dela(mina kontadorea adibidez).
* Helbideak klasea:
  + Eskatutako ikono edo irudi guztiak era egokian hartu eta dagokien lekuan gordetzen dituela.
* IkonoKonbinaketa klasea:
  + Bi ikono konbinatu behar diren momentuan era egokian egiten dela, ikono bat irudiaren atzealdean eta beste ikonoa aurrealdean ezarriz.
* LukiFactory klasea:
  + Mina bat ezartzean alboko laukien zenbakia era egokian eguneratzen dela.
  + Luki bakoitzaren belar irudia, laukiaren posizioaren arabera ongi ezartzen duela.
* Login klasea:
  + Erabiltzailea guztiak gordetzeko erabiltzen den .txt fitxategia era egokian sortzen dela eta ez dela fitxategi hori, behin sortuta dagoenean, berriro sortzen.
  + Ezingo dela errepikatutako erabiltzailea berriro sortu.
* LurLaukia klasea:
  + Ezkerreko botoia sakatzean egoeraren arabera era egokian zabaltzen dela edo zabaltzen ez dela.
  + Lur lauki bat sakatzean tableroa era egokian zabaltzen dela, alboko lur laukiak zabalduz zenbaki lauki batera heldu arte.
* MinaLaukia klasea:
  + Laukiaren egoeraren arabera, lauki hori zabaltzea posible bada, mina agertzen dela jokoa galduz eta gainerako minak bistaratuz.
  + Minak bistaratzerakoan, laukiaren egoeraren arabera, mina lauki hori mina irudi batekin ala beste batekin era egokian zabaltzen dela.
* Pika klasea:
  + Ezkerreko klik egitean jokatzeko tableroa berria agertzen dela.
  + Eskumako klik egitean ezkutuko tableroa era egokian agertzen dela, Pokemon-ez osatutakoa.
* PokeSweeper klasea:
  + Splash, login eta tableroaren frame-ak aldatzeko deiei ondo erantzuten diela.
  + Musika erreprodukzioa era egokian betetzen duela.
* Ranking klasea:
  + Jokalarien puntuazioak gordeko diren .txt fitxategia ongi sortzen duela, errepikatu gabe.
  + Irabazi duen jokalari baten puntuazioa era egokian idazten duela .txt fitxategian.
  + Jokalari bat aurretik puntuazio bat badu maila batean, oraingo puntuazioa hobea bada, puntuazio hori era egokian eguneratzen duela.
  + Maila bakoitzeko 10 puntuazio hoberenak era egokian bueltatzen dituela.
* ZenbLaukia klasea:
  + Ezkerreko klik egiterakoan, laukiaren egoeraren arabera, era egokian bistaratu ala bistaratzen ez dela.

# ONDORIOAK

Jokoaren garapenean IceScrum-en bidez egin beharreko lana antolatu ahal izateak laguntza handia eman digu, hau da, gure denbora antolatzeko eta ordu libreak aprobetxatzeko gaitasuna eman digu. Aldi berean, Sourcetree-ren bidez bakoitzak egindako aldaketak momentuan bertan ikusteko aukera eman digu, etorkizun batean arazoak ekiditeko.

    Klaseen inplementazioari dagokionez, Factory patroiak lauki ezberdinen sorrera asko erraztu du, gainera, honen bidez, kodeari baldintza gutxi batzuk jarrita lauki mota gehiago sortzeko aukera izango dugu era oso sinple batean. Honekin batera, erabilitako patroien artean gehien nabarmenduetarikoa Model-View-Controller izan da. Patroi honen bidez, zati logikoa eta interfaze grafikoa banandu ditugu, irudiekin gerta daitezkeen arazoak kode nagusian izatea ekidituz. Hala, oso erraza litzateke, nahi izanez gero, ikonoak aldatzea eta jokoari giro ezberdin bat ematea.

    IceScrum-ari dagokionez, hainbat akats egin ditugu. Software hau erabiltzen genuen lehen aldia zenez hainbat alditan eguneratzea ahazten zitzaigun (adibidez, 3. sprint-ean) edo sprint bat hasterakoan martxan jartzea. Nahiz eta hutsegite hauek egin ditugun, proiektuak kudeatzeko tresna oso interesgarri bat iruditu zaigu. Dudarik gabe karrerako beste proiektuetan erabiliko dugu.

# ERANSKINAK

### Bibliografia eta estekak

Github biltegia: <https://github.com/Xabiier/PokeSweeper>

Java dokumentazioa: <https://docs.oracle.com/javase/7/docs/api/>, <https://docs.oracle.com/javase/8/docs/api/> eta <https://docs.oracle.com/javase/tutorial/>

Stacoverflow-en begiratutako galdera batzuk:

* <https://stackoverflow.com/questions/16299517/setting-multiple-icons-on-jbutton>
* <https://stackoverflow.com/questions/19321367/adding-action-events-to-jbuttons/19321426>
* <https://stackoverflow.com/questions/4699892/how-to-set-the-component-size-with-gridlayout-is-there-a-better-way>
* <https://stackoverflow.com/questions/527719/how-to-add-hyperlink-in-jlabel>
* <https://stackoverflow.com/questions/1832432/which-swing-layouts-do-you-recommend>
* <https://stackoverflow.com/questions/26305/how-can-i-play-sound-in-java>
* <https://stackoverflow.com/questions/14025718/javafx-toolkit-not-initialized-when-trying-to-play-an-mp3-file-through-mediap>
* <https://stackoverflow.com/questions/23498376/ahow-to-make-a-mp3-repeat-in-javafx>
* <https://stackoverflow.com/questions/27881730/javafx-sound-play-stopped-after-seconds>

Java swing tutorialak:

* <http://www.java2s.com/Tutorials/Java/Java_Swing/index.htm>
* <https://docs.oracle.com/javase/tutorial/uiswing/TOC.html>