

Abstract Music Player



Egileak: Ander Areitio, Leire Barainka eta Borja Turiel

AURKIBIDEA

AURKEZPENA.....	1
Deskribapena.....	1
Helburua.....	2
Justifikazioa.....	2
Kokapena.....	3
DISEINUA.....	4
Erabilitako tresnak.....	4
Orokorra.....	4
Zerbitzaria.....	7
Aplikazioa.....	13
Justifikazioa.....	15
EXEKUZIOA.....	19
Proiektuaren exekuzioaren azalpena.....	19
Egoera Ideala.....	19
Oraingo egoera.....	21
Falta dena.....	26
Hobekuntzak.....	27
Arazoak.....	29
Zerbitzarian.....	29
Aplikazioan.....	30
Tarteko arazoak.....	31
ERANSKINAK.....	33

AURKEZPENA

Deskribapena

Orain dela bi urte, kurtsoko proiektu finala bezala [Abstract](#) izeneko musika munduaren inguruko web-orri bat garatu genuen; non gustuko genuen musika entzun, eta baita musika berria ezagutzeko aukera zegoen.

Web-gune hori oinarritzat hartuta, zerbitzari bat jarri diogu atzetik, eta zerbitzari bera erabiltzen duen aplikazio bat egin dugu, “Abstract Music Player”, non, web-an geneukan ideia nagusia plasmatu dugun, oraingoan edonon eskuragarri eta erabilgarri egongo den plataforma baten. Aplikazioak copyright gabeko musika bideratuta egongo da.

Gure zerbitzariak hainbat gauza edukiko ditu gure aplikazioa funtzionaraziko dutenak. Lehenengo gauza gure datu basea da, non erabiltzaileen datuak nahiz abestiak, albumak eta artisten datuak gordeko diren. Honetaz gain, Android aplikazioa datu basearekin konektatzeko API bat edukiko dugu, honek izango dituen eskakizunak erabiliz datuak berreskuratu ahal izango ditugularik. Azkenik, zerbitzariak orain dela bi urte egin genuen web orria edukiko du, orritik naiz aplikaziotik datu berdinak lortu ahal izateko.

Musika entzuteko aukeraz gain, entzuten zauden araberako gomendioak egiten ditu aplikazioak. Horretarako, abesti bakoitza egile eta album, genero eta subgeneroaren arabera sailkatuta dago; horrela aplikazioak gomendio ezberdinak emateko gai izango da, entzuten hari zaren abestiaren antzekoak diren album edo abesti ezberdinak erakutsiz. Hori lortzeko, artisten musika mota oso zabala denez (abesti batetik bestera musika mota aldatu daiteke), kasu honetan musika genero orokorra hartuta egingo ditu beste artisten gomendioak.

Albumak eta abestiak, osteria, genero espezifikoago batekoak direnez, subgeneroaren arabera sailkatu ahal izango ditugu.

Diseinu simple bat erabiltzea erabaki genuen hasieratik. Elementu gehiegiz erabiltzeak diseinuaz azkarrago aspertzeko arriskua zekarren, eta baita aplikazioaren errendimenduan edota konplexutasunean ondorioak izan zitzakeen. Kolore oinarria beltza eta gorria izango dira aplikazioan zehar eta mugikorrak izango diren elementuak menua eta erreproduktorea izango dira, denbora guztian eskuragarriak izateko.

Azkenik, aplikazioa ingelesez egitea aukeratu dugu, orain dela bi urte bezala. Erabaki hau proiektuak ahalik eta eremu handiena hartzeko hartu genuen, aplikazioa kode irekia izanda zenbat eta publiko handiagoa izan, garapena hobea izango dela uste dugulako.

Helburua

Musika gomendioak beste aplikazio batzuk baino askoz espezifikoagoak egiten dituen aplikazio bat egitea da gure helburua. Albumak subgeneroetan sailkatuz, aukera gutxiago baina askoz zehatzagoak lortzen dira, honela momentuan pantailan dagoenaren arabera egiten dira gomendioak. Hala eta guztiz ere, artisten atalean betiko sistema erabiltzen dugu, gomendio zehatzagoak lortuz.

Artista bat ikustean, musika gama antzekoa egiten duten beste artistak erakusten dira; albumen kasuan, subgenero berdineko beste albumak erakusten dira, baita abestiekin ere.

Laburbilduz, sortu nahi dugun aplikazioa erabiltzaile bakoitzera egokitzea nahi dugu, pertsona bakoitzaren gustuak errespetatuz, honela gustuko den musika entzuten jarraitzeko.

Justifikazioa

Merkatuan hainbat musika erreproduktore daude, musika gomendioak egiten dituztenak. Gomendio hauek, ordea, ez dira erabiltzaileak nahi izaten duten bezain espezifikoak, eta hau gogaikarria izaten da gehienontzat. Horregatik agertu zen Abstract ideia. Ahalik eta zehatzentegiteko musika gomendioak eta beti izateko antzekoak diren abestiak eskuragarri.

Software librea informazio teknologien etorkizuna dela uste dugu, seguruagoa delako, erabiltzailearen askatasunak errespetatzen dituelako eta erabiltzailearen pribatutasuna mantentzen duelako. Gaur egun pribatutasunari buruz hainbat egonezin daude, Facebook eta antzerako plataformaei (eta hauei lotutako aplikazioak -Spotify, bat aipatzeko Facebooketik hartzen du bere erabiltzaileen informazioa-) buruz irten diren berrieikin jendea konturatzen ari da pribatutasunaren garrantziarekin, eta hau software librearekin baino ezin denez lortu, software librea erabiltzea aukeratu dugu, eta gure aplikazioa software libre den lizenzia batekin lizentziatzea.

Open Source izanda, gure aplikazioaren kodea ikusteko ahalmena emango diogu jendeari, hauek irakurtzeko, auditatzeko eta nahi izanez gero izan ahal dituen arazoak konpontzeko. Ezaugarri hauek aprobetxatu ahal izango ditugu aplikazioaren segurtasuna, errendimendua... denboran zehar modu hobeago batean kudeatzeko.

Kokapena

Aplikazioaren erabilerari dagokionez, internetera sarbide dagoen lekuetan egongo da aplikazioa erabiltzeko prest: kalean, etxeen... Internetera konexioa duen gailu batekin Abstract Music Player eskuragarri egongo da erabiltzaileentzako. Edozein momentutan egongo da prest gustoko den musika errepruduzitzeko.

Aplikazioaren garapenaren aldetik, Iurreta Institutuko AGPr-ko proiektua da, eta gehienbat gure etxeetan egin dugu praktika eta beste lan batzuen artean. Gero ostiraletan Iurreta Institutuan egin dugu lan, daukagun orduetan maisuen laguntza aprobetxatzen.

DISEINUA

Erabilitako tresnak

Orokorra

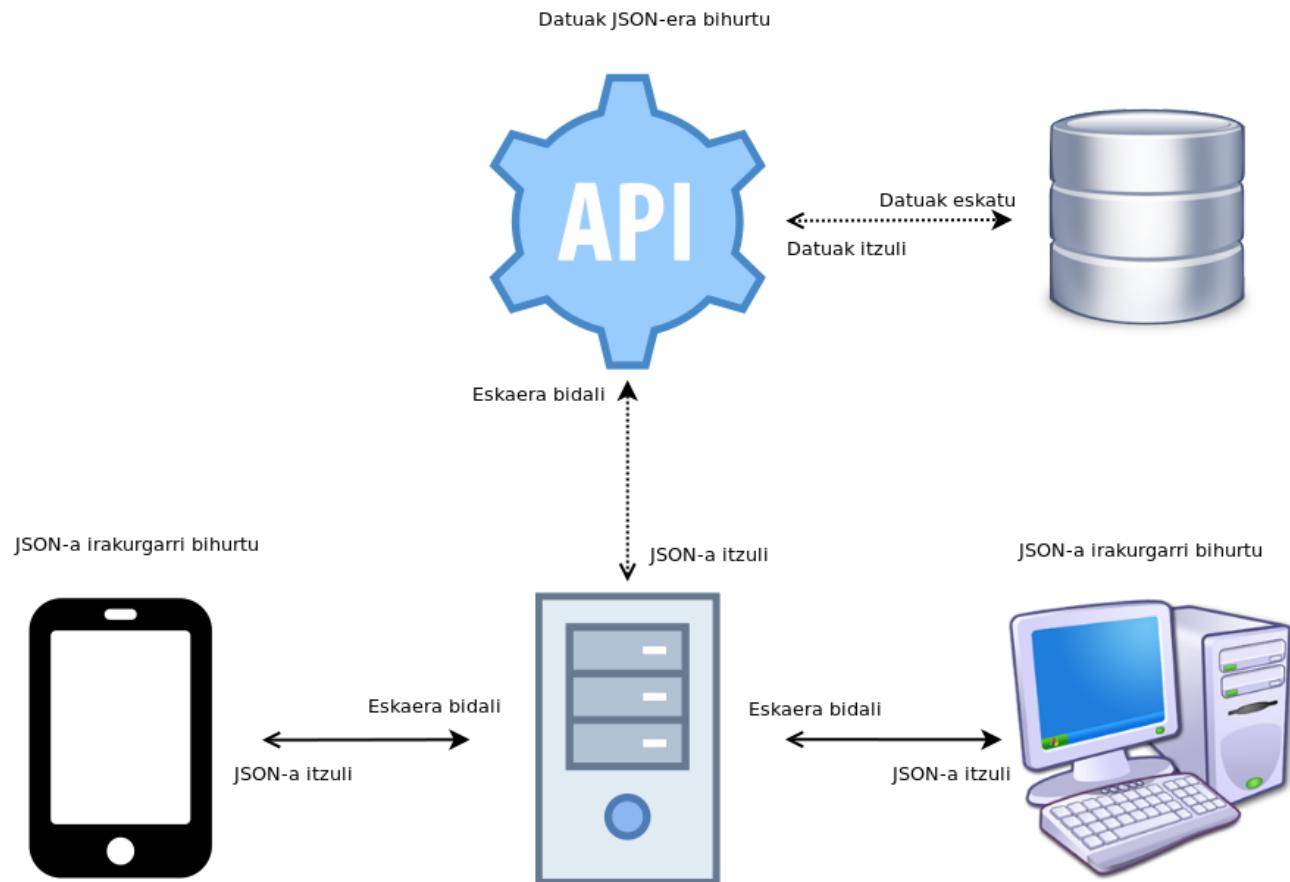
Hardwarea

Eskolan dagoen zerbitzaria da aipagarriena (HP MicroServer Gen8). Honek aplikazioari zerbitzua emango dio, eta bertan datu basea egongo da implementatuta bertan.

Aplikazioa instalatzeko erabiliko den mugikorrek Android 6.0 baino bertsio berriagoa behar du.

Azpiegitura

Internetera konektatuta dagoen mugikor batetik (edo web orrialdetik), Abstract Music Player aplikazioa erabilita, eskera bat egiten dio zerbitzariari (zerbitzari hau lurretako sarean dago kokatuta). Zerbitzarian kokatuta dagoen APIak eskaera interpretatuz datuak eskatuko dizkio Datu Baseari, eta APIak berak JSON-era bihurtuko ditu eskaera egin duen gailura bidaltzeko. Azkenik, gailuan bertan JSONa parseatu egingo da eta datuak prozesatuko dira behar den bezala (abestiak erakutsi, logatzeko baimena eman...).

Diagrama*Komunikazio Diagrama*

Exekuzio atalean azalduko da diagrama pausuz pausu, aplikazio guztiaren funtzionalitatearekin batera.

Ingurunea

Erabili izan ditugun erreminta gehienak kode irekikoak direnez erabaki dugu Android erabiltzea, hau ere kode irekiko sistema eragilea delako. Azken xehetasun bezala esan behar da Android bertsioa 6.0 edo berriagoa izan behar dela.

Internetetik eta ingurunean genituen pertsonen mugikorrik begiratuz, mugikor gehienak bertsio hau erabiltzen dutela ikusi genuen. Gainera, gaur egun erositako edozein mugikor gutxienez bertsio hau instalatuta dakar. Hau kontuan izanda bertsio honek garapen nahikoa izan duenez, konpatibilitatearekin arazo gutxiago izango genituela pentsatu genuen, eta honen ondorioz Android Marshmallow erabiltza erabaki genuen.

Aplikazio esparrua

Esparru pertsonalean erabiliko den aplikazioa da. Edonork erabiltzeko dago prestatuta eta pentsatuta; edonon eta edonoiz erabiltzeko prest dago, murrizketa edo inposiziorik gabe, baldintza soilak erregistratuta egon eta internetera konexioa izatea besterik ez direlarik.

Zerbitzaria

Apache



HTTP protokoloak erabiltzen duten zerbitzari bat edo gehiago hosteatzeko erabiltzen den kode irekiko softwarea da Apache.

Iurretan dagoen zerbitzarian instalatuta eta prestatuta, Apache erabiltzen dugu gure zerbitzaria sarean jartzeko, edonondik erabilgarri dago modifikazioren bat egiteko eta aplikazioarekin konexioak egiteko. Gainera API-a ere hosteatzen du. API honek egingo ditu datu transferentzi lanak, aplikazio - datu basea, eta alderantziz.

MariaDB



Datu baseak kudeatzeko erabiltzen den sistema da, MySQLko sortzaileak berak sortutako sistema (abantaila berdinak ditu, hobekuntza batzuekin; adibidez datu basearen optimizazioa eta errendimenduaren hobekuntza), gure beste erremintak bezala, kode irekiko softwarea da.

Datu kudeaketa sistema honetan erabaki dugu gure datu basea kokatzea.

PHP

API-a idazteko, bi lengoaien artean erabaki behar genuen, Python eta PHP, eta PHP aurretiaz erabili izan dugun lengoai bat denez, trebetasun gehiago daukagu. Horregatik aukeratu dugu kode irekiko lengoai hau gure API-a idazteko.

PHPMyAdmin

Kode irekiko softwarea, SQL datu baseen kudeaketa errazten du, interfaze grafiko bat erabilita begietara errazagoa izateko. Sistema osoa antolatuta izan dugunean erabili dugu triggerak amaitzeko eta frogatzeko gehien bat, honen egitura kontrola hobea izateko.

MySQL Workbench



Datu baseak era grafikoan maneiatzeko erabiltzen den erreminta, software garapena, datu baseen administrazioa, diseinua, sorrera eta mantentzeak osatzen dute software hau. MySQL-ren administrazioarekin laguntzen digun programa da.

SSH

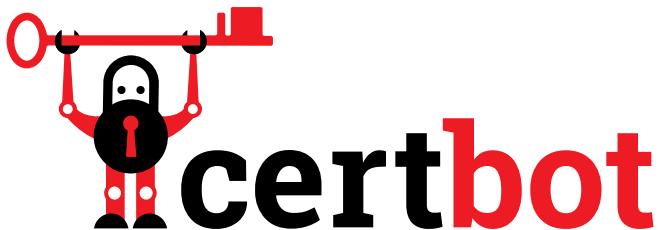


Urruneko administrazio protokoloa, Interneten bidez zerbitzariak kontrolatzeko eta aldatzeko erabiltzen da. Zerbitzaritik eta zerbitzarira doan komunikazioa enkriptatzen du. Erabiltzaile bat autentikatzeko mekanismoa eskaintzen du, bezeroaren ostalariaren sarrerak transferitu eta irteera atzera bezeroari berriro bidaltzen. Linux oinarriko SE-a erabiltzen dugunok erabili izan dugu.

Fail2Ban



Intrusioei aurre egiteko erabiltzen de aplikazioa, indarrez sartzen ahalegintzen diren urruneko konexioak penalizatu edo blokeatu egiten ditu, suebakia aldatzen. Zerbitzari bat eramateko oso egokia den aplikazioa.

Cerbot(Let's Encrypt)

Let's Encrypt ziurtagiri autoritatea da, dohaineko eta irekia, Linux Fundazioak bultzatuta, SSL ziurtagiriak dohainekoak eta automatikoak sortzeko aukera ematen du gure webguneetarako. Bere helburua Interneteko trafikoa segurua izatea sustatzea da.

ACME (Authentic Certificate Management Environment) protokoloa erabiltzen du, domeinua balidatzen duena, eta zihurtagirien eskaera.

Debian

GNU erreminta, Linux nukleoa, eta beste dohaineko software garrantzitsua, Debian GNU/Linux izeneko distribuzioa sortzen dute. Distribuzio hau paketeekin dago osatuta. Pakete bakoitza exekutagarriak ditu, script-ak, dokumentazioa eta konfigurazioko informazioa, eta arduradun bat, pakete hori eguneratuta izateaz arduratzen dena, honen errore txostenen segimendua egiten du eta hauek egileei bidaltzea dizkie, hauek erroreak konpontzeko ahalik eta azkarren.

Debian-en dagoen softwarearen kalitatea, software horrekiko ardurak eta egonkortasunak bihurtzen du Debian sistema eragilea zerbitzari munduan errespetatua eta erabilia.

Gainera, Debian-ekin Unattended-upgrades ere erabili dugu. Honek zerbitzaria eguneratuta mantentzen du azkenengo segurtasun (eta beste) eguneraketekin, -dena automatikoki.

Hau zerbitzarian edukitzeko, monitorizatzeko ahalmena izan behar da *apt-listchanges* paketea instalaturik, eta hau konfiguratuz eguneraketen informazioa eMail-era bidaltzeko aukera dago.

Putty



PuTTY SSH eta Telnet-eko bezeroa da, urruneko zerbitzariei konektatzeko erabiltzen da, saioa hasi eta komandoak exekutatzen uzten digu. Dohakoa da eta kode irekikoa, portablea da, simplea eta erraz maneiatzeko. Beti dago eguneratzen. Windows-etik konektatzeko zerbitzarira erabili izan dugu.

Borgbackup



Borg bezala ezagututako erreminta, software honekin deduplicating teknika erabilita segurtasun kopiak sortzeko gaitasuna du. Teknika honekin, segurtasun kopia hauek datu berriekin edo aldatu egin diren datuekin sortzen dira eta honek, egunero egiteko software egokia egiten du.

Postman

Postman API ezberdinak maneiatzeko erabiltzen den erreminta bat da. Erreminta honi esker, API eskaerak eta erantzunak frogatu daitezke, bidaltzen diren parametroak kontrolatu...

Aplikazioa

Java



Objektuei zuzendutako programazio lengoaia, asko zabalduta dago eta gero eta importantzia gehiago du internet esparruan zein informatikan orokorrean. Javak duen ezaugarri nagusienetarikoa ingurunearekiko independentea dela da, hau da, Java-n idatzitako programak edozein ordenagailutan ibiltzeko daude diseinatuta (Java-ren makina erabilita, sistema eragile bakoitzeko, programa eta sistemarekiko zubia eginez).

Bi urte hauetan gehien erabili dugun programazio lengoaia izan da, eta honen ondorioz, hobekien idazten duguna.

Android Studio



Android aplikazioen garapena egiteko erabiltzen den IDE ofiziala. Bere ezaugarrien artean hauek daude, beste batzuekin batera: frogak egiteko momentuan emuladoreak hainbat Android bertsioekin, Instant Run aldaketat ezartzeko konpilatu barik eta ingurune bateratu bat Android-eko hainbat dispositibotan egiteko lan.

Aplikazioa programatzeko erabili dugun erreminta nagusia da. Klaseen antolamendu erraza ahalbidetzen du, diseinuaren edizio ona, app instalazio azkarra eta emuladorea erabiltzeko aukera mugikor edota kablerik ez izatekotan.

Eclipse



Plugin-en bidez handitzen den garapen ingurunea da gehienbat Eclipse. Ez du lengoia espezifikorik garatzeko, baizik eta IDE orokor bat da, nahiz eta gehienbat Java-rako erabiltzen den komunitatean. Erraztasuna ematen du sintaxia zaintzeko, kodea idazteko laguntza ematen du, idatzitakoari buruzko gomendioak zein koloreekin objektuak ezberdintzeko...

Android-en frogak egiteko denbora asko galtzen denez (aplikazioa behin eta berriro instalatu, mugikorrean kargatzeari itxaron....) eclipse erabili izan dugu hauek egiteko, batez ere zerbitzariarekiko komunikazioa errazagoa izateko (eclipsek erantzuna jasotako momentuan erakustea ahalbidetzen du).

JSON



JavaScript Object Notation-en akronimoa. Testu planoan oinarritutako estandarra, informazio trukaketak egiteko, horregatik erabiltzen da datuak erakutsi edo bidali behar diren hainbat sistemetan. Abantaila bat da hizkuntzarekiko independentzia izatea, hau da, metodo hau erabiltzen duten zerbitzuek ez dute hizkuntz bera hitz egin behar.

Justifikazioa

Erabili izan ditugun erreminta asko etxekotasunagatik aukeratu ditugu. Denbora askotan erabili izan ditugun tresnak izan direnez, ez dugu pentsatu egokia denik aldatzea eta beste erremintarik erabiltzea.

Esate baterako, [Eclipse](#) bi urteetan gehien erabili izan dugun IDE-a izan da, eta frogak egiteko erosoa da guretzat, badakigulako nola erabili eta konpondu berarekin.

[Android Studio](#) lan egitea aukeratu dugu guretzako ezagunena delako. Gainera, Android-en garatzeko erreminta ofiziala da, eta dohainik. Azken ezaugarri hau proiektuan izan dugun filosofiarekin bat datorrenez, eta jada azaldutako arrazoiei gehituz, IDE hau erabiltzea erabakirik onena izango litzatekeela pentsatu genuen.

[Java](#) eta [PHP](#)-n kasuetan, lau urteetan zehar ikasi eta erabili izan ditugun bi lengoaiak dira. Orain dela lau urte hasi ginen [PHP](#) ikasten, eta, nahiz eta azkenengo bi urteetan asko erabili

ez izan, trebeagoak gara beste lengoai batzuekin konparatuz, Gainera, PHP-rako pakete instalatzaire bat erabili dugu, 'Composer' izenekoa. Hau JWT-ko liburutegia instalatzeko erabili dugu. Honen abantaila nagusia gure kasuan, liburutegia eguneraututa izateko erraztasuna da, berak egiten duelako gehienetan, gur komando bat sartu behar dugu.

[Java](#) orain dela bi urte ikasten hasi, eta bi urte hauetan zehar asko landu eta hobetu dugu lengoai honekin.

[Apache](#) ere aurretiaz erabili izan dugun softwarea da. Orain dela bi urteko proiektuetan erabili genuen web-orrialdeak sortzeko eta sareak muntatzeko, eta asko landu genuen bi urte horietan zehar.

[MariaDB](#) software berria da gurentzat, baina ez ezezaguna, azkenean MySQL-ren antzerakoa da, fundatzaile bera dute eta, hobekuntzak ditu MySQL-rekin konparatuz, aurretiaz esandakoak ditugu, baita bi motore erabili (PBXT eta FederatedX) ere. Ez dago diferentzia handirik bien artean, baina mariadb mysql-ren sortzailearen bertsioa denez, software librea denez eta oracle-ren izenpean ez dagoenez, mariadb aukeratu dugu mysql-ren aurrean. Gainera, Linux distribuzio gehienetan, mariadb-k hartu du mysql-ren tokia, eta distribuzioek gomendatzen duten bertsioa da gaur egun.

[SSH](#) eta [PuTTY](#), biak erabili izan ditugu helburu berdinarekin, gure etxeetatik zerbitzarira sarbidea izateko, urrunetik egiteko aldaketak, eguneraketak eta kontrola. Esan dugun bezala, [SSH](#) Unix oinarriko sistema eragileetan erabili ditugu (bi ordenagailuk Arch dute sistema eragile moduan), eta [PuTTY](#) Windows sistematan erabili dugu.

[Certbot](#) zerbitzaria sistema seguru bat egiteko erabili izan dugu. Let's Encrypt era erraz batean jartzen lagundu digu eta Let's Encrypt-en aldean, Certbot jadanik badator cron batekin; zeinek, ziurtagiria automatikoki berriztatzen du honen epea amaitu aurretik.

[Debian](#) aukeratu dugu gure zerbitzariko sistema eragilea izateko etxekotasuna gaitik. Lau urte hauetan ikasi dugun gehiena Ubuntu-ri buruz izan da, eta hau Debian-etik eratorria denez, ikasi duguna Debian-era eraman ahal dugu. Beste zerbitzarentzat gomendatutako sistema eragileen aurrean, CentOS adibidez, etxekotasunetik gainera, abantaila bezala du bertsioen arteko eguneraketak baimentzen dituela ofizalki, sistema eguneratzea erraza egiten duena, backup gabe eta berriro instalatu egin gabe. Ubuntun aurrean, honek daukan komunitatea handienetarikoa da, sisteman banatzen diren softwarearekiko sostengu ofiziala du.

[Fail2Ban](#) SSH-a seguruagoa egiteko metodo bat da, SSH-a LAN pasahitz sistema batekin lan egiteko gaituta dagoenez, pasahitz hori hiru bider txarto bidalita petizioari uko egiteko denbora jakin bat pasatu arte.

[PHPMyAdmin](#)-k errazago egiten du datuak tratatzea pantailaratzean. Sistema osoa muntatuta izan eta gero erabili genuen frogak egiteko, datuak sartzeko, eta eskaera konprobaketak egiteko. Aurretik erabilia dugu softwarea, eta trebetasuna lortu dugu lau urtetan.

[MySQL Workbench](#) proiektuaren hasieran erabili genuen bakarrik (zerbitzariaren konfigurazioa finkatuta eta prest egon zen arte). Datu baseak behar zituen aldaketak eta gehiketak egiteko erabili zen (batez ere triggeren programazioa). Erreminta hau instalatzeko denbora oso gutxi behar da, eta konfiguratuta dator, beraz datu basea maneiatzzen ahalik eta

azkarren hasteko aukerarik onena izan zen. Hala eta guztiz ere, behin phpmyadmin instalatuta eta konfiguratuta zegoenean, Workbench-a alde batera hutsi genuen, proiektuaren filosofiarekin bat jarraitzeko.

[JSON](#)-en bi liburutegi erabili izan dugu eclipsen eta PHP-n.

- Eclipse: Zerbitzaritik datozen erantzun guztiak JSON formatua dute. Erantzun hauek ulergarriak egiteko eta pantailatik erakusteko, abestia jotszeko partseatu egin behar ziren. Horretarako Maven biltegi erabili ahal den org.json liburutegia erabili dugu. Behin zerbitzariaren erantzuna string baten dugunean, funtzi bat erabiliz JSONObject edo JSONArray bihurtzen dugu, beharraren arabera, erantzun honek dituen datuak askoz ere maneiagarriagoak egiteko (datuak ateratzeko objektu batetik esaterako, liburutegiak berak duen beste funtzi bat erabili dugu). Honek, nahiz eta hasieran lan gehiago eman liburutegiaren funtzionamendua ulertzeko, lana asko erraztu digu.
- PHP: JWT erabiltzen dugu token moduan segurtasunagatik, erabiltzailearen datuak denbora guztian bidali gabe, token-a bidaltzen dugu datuak lortzeko. Erreminta hau aukeratu dugu gure lana sinpletu egiten duelako, eta ez dugulako hainbeste konprobaketa egin behar eskuz, JWT-ak defektuz egiten duelako automatikoki.

EXEKUZIOA

Proiektuaren exekuzioaren azalpena

Proiektuaren exekuzioa azaltzerakoan, txukunago eta ulertzeko errazago egiteko, lau ataletan banatzea erabaki izan dugu; Lehena, proiektua egiterakoan izan genuen ideia nagusia eta ideala, bigarrena, gaur egun dagoen egoeran, hirugarrena denbora faltagatik ezin izan ditugun gauzak, eta azkenik proiektua garatzen genuen bitartean bururatu zitzaizkigun hobekuntzak, posiblak izango zirenak denbora gehiago balego.

Egoera Idealak

Zerbitzaria

Hasieratik ideia nagusia zerbitzaria ondo muntatzea zen. Abstract Music web-orria egin genuenean (orain dela bi urte) makina berdinean muntatu genuen web zerbitzaria LAMP erabiliz, hau da, guk ez genuen konfigurazio handirik egin. Oraingo proiektuan hastean, zerbitzaria ondo muntatu nahi genuen, internetetik eskuragarri egongo zen zerbitzari bat eta, arrazoi honegatik, segurtasun handiago zuen zerbitzari bat muntatu nahi genuen.

Lehengo momentutik pentsatu genuen token-ak erabiltzea, Android aplikazioei reverse engineering egitea oso erraza delako, eta modu honetan aplikaziotik erabiltzailearen datuak ateratzeko arriskua dagoelako. Hauen ideia denborarekin iraungitzea zen, horrela momentu batean norbaitek token hori lortzen badu, ezin izango luke informazio asko lortu zeren eta token hori iraungitako zen eta ezin izango zen erabili.

Web zerbitzariaren abiadura handitzeko, hasierako ideia cache bat martxan jartzea zen, abesti zein argazkiak modu azkarrago batean bidali ahal izateko, eta cache hau datu basean ere gehitzea datu baseari egindako eskakizunak azkarragoak izateko.

Honetaz gain, MariaDB datu basea konfiguratu nahi genuen, datu basearen memoria zein prozesadore nahiko erabiltzeagatik ezaguna delako datu asko dituenean, gure kasuan datu asko ez ditu izango, behintzat momentuz, baina konfigurazio hau orain egiteak proiektua handitzerakotan prest egotea baimenduko liguke.

Honelako zerbitzari bat martxan jarri ondoren, hemen dauden konfigurazioen eta datuen segurtasun kopia bat egitea izango zen egokiena, horretarako borgbackup softwarea erabiltzea aukerako genuke.

Web orriaren aldetik, orain dela bi urteko web orria apur bat ukitzea zen ideia, PHP bera erabili beharrean TWIG bezalako framework batekin berridaztea eta modu egokian jartzea.

Azkenik, gure aplikazioa software librea denez, norbaitek erabili nahi badu, zerbitzaria martxan jartzeko erraztasuna ezaugarri handi bat da, horretarako Red Hat enpresak sortutako Ansible softwarea erabiltzea pentsatu genuen, horrela script batekin zerbitzaria martxan jartzeko modu erraza izateko.

Aplikazioa

Hasierako ideia aplikazioaren aldetik musika gomendioetan oinarritzea zen. Aplikazioak erregistroa egiteko aukera emango luke, non erabiltzaile izena, pasahitza eta eMaila sartu beharko genuke. Behin erregistratuta erabiltzaile berri honekin logatu beharko ginateke aplikazioaren pantaila nagusira sartzeko.

Bertan genero nagusiak agertuko ziren atal nagusi moduan (azaldu dugun bezala, musika genero eta subgeneroka sailkatuta dago gomendioak zehatzagoak izateko), eta bat aukeratzerakoan genero horren artista edo subgenero bat aukeratu beharko genuke. Bi kasuetan artistaren edo subgeneroaren albumak agertuko ziren, eta bat aukeratzearekin aplikazioak albumaren abestiak errepruduzituko lituzke. Hala ere, album horren abesti bakar bat entzun nahiko bagenu, aukera hori berdin izango genuke.

Azpiko aldean, errepruduktorearen kontrolak izango lituzkeen barra bat, errepruduktorearen pantaila sartu barik hurrengo edo aurreko abestira joateko edota musika pausatzeko.

Albo batean, menu hedagarri bat edukiko genuke, non aplikazioaren konfigurazioa, erabiltzailearen datuak aldatzeko aukera... izango genuke, eta baita bilaketak egiteko ere. Erabiltzailearen atalean honen eMaila, pasahitza edo erabiltzaile irudia aldatzeko aukera izango genuke.

Azkenik, errepruduktorearen atalean artistaren eta albumaren informazioa lortzeko aukera izango genuke, eta baita hurrengo abestia ausazko bat izateko edo errepruduzitzen gauden lista bukatzerakoan berriz hasteko aukera.

Oraingo egoera

Zerbitzaria

Zerbitzariaren uneko egoera ez dago oso urrun pentsatu genuenarengandik, nahiz eta pentsatuta genituen hainbat gauza ezin izan ditugu egin denbora faltagatik. Gure zerbitzaria martxan jarri dugu eta internetetik eskuragarri dago, web zerbitzaria, datu basea eta PHP guk instalatu eta konfiguratu ditugu LAMP erabili beharrean, eta asko zentratu gara segurtasun

aldean, nahiz eta ez garen heldu nahi genuen puntura. Datu baseko datuak aldatu egin ditugu, Creative Commons lizentzia duten abestiengatik aldatu ditugu abestiak, eta horrekin diskoak eta artistak. Web orrien aldean, API-a egiten zentratu gara aplikaziorako garrantzitsuena delako, datu basetik datuak ateratzeko eta aplikazioari bidaltzeko. JWT (JSON Web Token)-a implementatuta dugu, API eta aplikazioaren arteko komunikazioa modu erraz batean egiteko, API-ak token-ak sortu eta balidatzen ditu modu egoki batean.

Aplikazioa

Nahiz eta denboraren ondorioz hasierako ideia guztiak ezin izan ditugun bete, funtzionalitate asko implementatzeko aukera izan dugu. Gure aplikazioak, erabiltzaileak erregistratzeko eta logatzeko aukera ematen du hauek beharrezkoak dituzten konprobaketak eginez (bi erabiltzailek izen berdina ez izateaz, esaterako). Gainera, logatzerakoan, token sistema erabiliz erabiltzailearen token-a gordetzen dugu, eta nahiz eta aplikazioa itxi, irekitzerako orduan, token-a oraindik badugu, ez da beharrezkoa izango berriz datuak sartzea. Token hau borratzeko egin behar den bakarra gure saioa ixtea da.

Behin barruan irekitzen zaigun menuak generoak, subgeneroak, artistak, albumak eta abestiak listatzen ditu, guk filtroa bete ahala entzun nahi dugun musika motaren datuekin. Behin album edo abesti bat filtroan edukita, hau errepruduzitzen aukera dugu, errepruduzitzen hari denean, honen pantailan sartu gaitezke, eta bertatik ausazko erreprrodukzioa edo errepikatzeko aukera aktibatu edo desaktibatu. Bertan ere abestiaren, albumaren eta egilearen izenak ikusi ditzakegu.

Komunikazioa

Aplikazioa mugitzen duen gailuan (beti ere internetera konexioa badu) botoi, testu... bat sakatzerakoan petizio bat sortzen da, beharrezkoak diren parametroak eta helbidea erabiliz. Helbidea eskaeraren arabera ezberdina izango da, baina eskaeraren parametroak eskaeraren motaren arabera ezberdinduko dira. GET eskaera bat bada, parametroetan eskaera mota ezarriko da, eta baita (gordeta badugu) erabiltzailearen token-a bidaliko da. POST eskaera bat bada, ordea, POST mota ezarriko da parametroetan, baina ez da token-ik bidaliko. Bi eskaeretan, ordea, eskaeraren iturri bezala "AbstractMusicApp" agertuko da, zerbitzaritik jakiteko aplikaziotik egin dela eskaera.

Behin gailuan eskaera sortuta, zerbitzariarekin konexioa irekiko da, eskaeran sortutako helbidea erabiliz. GET eskaera bada, ez da daturik idatziko, helbidean bertan zein den eskaera jakin daiteke eta parametroak konexioarekin bidaltzen dira. POST eskaeretan, ordea, datuak "idatzi" daitezke behin konexioa irekita (Batez ere login edo erregistro berri bat egin behar denean, esaterako, seguruagoa delako bidali behar diren datuak sentikorrak direlako).

Bezeroak API-ari eskakizun bat egiterakoan, API-ak URL-a konprobatzen du. URL-an datorrenaren arabera, erantzun bat edo beste bat bueltatuko du.

Eskaera gehienak GET motakoak dira. Bezeroak datuak eskatzen ditu zerbitzarira token bat bidaliz. Behin eskaera heltzen denean zerbitzariak konprobatzen du(token-aren sinadura ondo dagoen, barruko datuak ondo dauden...)

Token-a frogatu ondoren, bi array dituen JSON array bat bueltatzen du. Token-a baliozkoa bada, lehenengo array-ean id moduan 1 bat eta token-a egokia delako mezu bat bueltatzen

du, eta bigarren array-ean bezeroak eskatutako datua. Token-a ez bada baliozkoa, aldiz, lehengo array-ean id moduan 0 bat bueltatzen du, eta mezu moduan token-aren arazoa, baina, kasu honetan token-ak ez duenez balio, bigarren array-a hutsik bueltatuko du zerbitzariak.

POST motako eskaera bi daude, "login" eta "register", bat erabiltzailea erregistratzeko eta bestea erabiltzailea aplikazioan sartzeko. POST eskaeren kasuan, array bakarra bueltatzen du, id batekin, 0 edo 1 bueltatzen duena, eta deskribapen bat aplikazioan erakusteko, login-aren kasuan, gainera, token-a bera bueltatzen du.

Behin bezeroak eskaera bat egiten duenean eta datuak bueltatzeko garaia denean, API-a datu basera konektatzen da eta datuak lortzen ditu, datu hauek JSON array batera transformatu, eta datuak bezeroari bueltatzen dizkio JSON formatuan

Erantzuna gailura heltzen denean bueltan, konprobatzen den lehenengo gauza lehenengo arrayak ekartzen dituen baloreak dira. Honek 1 bat badakar, erantzuna ondo bueltatzen dela dionez. 1 bat ez den beste edozer badator, honekin datuen string-a erakutsiko da, bertan errorearen zergatia datorrelako. 1 bat badator, ordea, erantzuna prozesatuko da, kasu bakoitzean beharrezkoak diren pausuak jarraituz (Abesti zerrenda bada bi arraytan banatu, bata id-ekin eta bestea izenekin, abesti baten datuak badira dagokien tokietan jarriz...).

Aplikazioaren pantailak

Aplikazioa zabaltzen denean, [login](#) pantaila agertzen da. Bertan, token bat gordeta badugu, hau erabiliz, logatzen saiatzen da. Lortzen ez badu, pantaila normala ikusi ahalko dugu, eta hemendik datuak sartuz logatzeko edo erregistro pantailara joateko aukera dago.

Erregistroa egiteko pantailan gure erabiltzaile izena, pasahitza eta eMail-a sartu beharko ditugu, eta ezaugarri batzuk betetzen baditzte (erabiltzaile izenak 5 karaktere baino gehiago izan behar ditu, eMail-ak formatu konkretu bat -edozer@zerbait.edozer-, pasahitzak 8 karaktere izan behar ditu gutxienez...). Ezaugarri hauek betetzen badira, mezu bat erakutsiko digute, erregistroa ondo bete dela esanez, eta login pantailara joango gara, non, gure erabiltzaile berriaren datuak sartuta, pantaila nagusira bidaliko gaituzte.

Pantaila nagusian ikusiko dugun lehenengo gauza generoaren filtroa izango da. Sakatu ostean, azpian agertuko den listan genero guztiak ikusiko ditugu. Bat aukeratuz subgeneroaren eta artistaren filtroa agertuko dira, biak generoka sailkatzen direlako. Bietako bat aukeratuta, bestea desagertu egingo da, eta albuma aukeratzeko filtroa agertuko da. Behin filtro hori beteta, musika entzuten hasteko aukera izango dugu, baina baita disco horren abesti bakar bat hartzeko ere. Nahiz eta momentuz implementatuta ez egon, listaren azpian gomendioak egiteko testua dago, non filtroa bete ahala antzerako musika agertuko den. Azkenik, beheko aldean bi botoi daude: Ezkerrekoa (Play Selection) filtroan dagoen musika jotzen hasiko da, eta erreproduktorearen pantailara joango da.

Erreproduktorearen pantaila ez da beste edozein erreproduktorerena baino askoz zailagoa ulertzeko. Aurreko abestira joateko, hurrengo abestira joateko edo abestia gelditzeko/berrabiarazteko aukerak ematen ditu, eta baita abestiaren edozein puntura joateko aukera azpian agertzen den barra erabiliz. Azkenik, aukeratutako abesti zerrenda bukatzean berriz hasteko botoi bat dugu, eta baita zerrendako abestiak ausazko ordenan entzuteko.

Azkenik, aplikazioa ixten dugunean, honek dituen parametro batzuk gordetzen dizkigu. Hauen artean gure erabiltzailearen token-a (berriz irekitzean gure datuak ez eskatzeko guk ez badugu gure saioa ixten), aukeratuta dugun musika zerrenda eta entzuten hari garen abestia (hurrengoan entzuten hari zinen abestiak entzuten jarraitzen hauek berriz bilatu barik) eta baita errepiaken eta ausazko botoien egoera.

* Pantailen itxura hobeto ulertzeko eranskinetan dauden argazkiak ikusi daitezke

Falta dena

Zerbitzaria

MariaDB martxan jarri dugu eta datu berriak sartu ditugu, baina ez dugu MariaDB konfiguratzeko denborarik izan, hau, gure kasuan, ez da oso garrantzitsua momentuz, datu asko ez ditugulako, baina etorkizunera begira konfigurazioa egin behar genuke, datu gehiago sartzen diren heinean

Web orrialdea ez dugu berridatzi, denbora hau API-a egiteko behar izan dugulako. Etorkizunera begira hau egin beharko genuke, web orria hobetzeko.

Ansible-ko script bat egiteko ez dugu denborarik izan, gomendagarria izango zen zerbitzaria berregin nahi badugu modu azkarrago batean egiteko, edo, software librea denez, norbaitek gure aplikazioa erabili nahi badu, zerbitzaria martxan jartzeko modu azkarrago bat izateko.

Apache-ren segurtasuna handitzea aldaketa egoki bat izango litzateke, mod_security eta horrelako moduluen bitartez kontutan izan ez ditugun gauzak segurtatzeko.

Aplikazioa

Proiektuan aurrera egin ahala ideia batzuk alde batera hutsi behar izan ditugu, beti ere garrantzi gutxien zuten funtzionalitateak izateaz ziurtatuz. Ideia nagusia erabiltzaile sistema eta musika entzuteko aukera izanda, eta hauek lortuta, atal nagusitik erabiltzaileen datuak editatzeko aukera eta aplikazioaren konfigurazioa editatzeko aukera ez ditugu implementatu, denbora faltagatik. Bi aukera hauek menu hedagarri baten zeuden, baina hau ere kanpoan gelditu da. Aplikazioaren leihon nagusi honetan, baita ere, musika kontrolatzeko barra bat egon beharko litzateke, baina hau implementatzeko denbora gehiegiz beharko genuke, informazioa bilatzeko zein ideia aurrera eramateko.

Musika gomendioak egiteko atala badago, baina ez du bere funtzioa betetzen momentuz. Hala ere, proiektuak aurrera jarraitzen badu hau izango da gure lehentasuna. Erreproduktorearen atalean album eta artistaren informazioaren atalera sartzeko aukera badago, baina bertan ez dago informaziorik.

Hobekuntzak

Zerbitzaria

API-ak gaur egun api.php izenari erantzuten dio, hau ez da modu egokiena hau egiteko. Hau hobetzeko api.php fitxategia /api edo horrelako modu batean deitzea izango litzateke, horretarako PHP router bat egin beharko litzateke, honek url-a /api bada api.php fitxategira bidaltzeko.

Gaur egun API-aren metodo guztiak fitxategi berdinean daude. Orain dauden metodoekin nahiko maneiagarria da, ez direlako metodo asko, baina etorkizunera begira hau objektuetara bideratu beharko zen, metodo bakoitzarengatik eskakizun bat izanik.

Token-eten doazen datuak ez dira oso garrantzitsuak, pasahitza adibidez ez dagoelako token-aren barruan, baina, hala ere, token-a JWT-era bihurtu baino lehen datu hauetan enkriptatu egin beharko ziren segurtasuna hobetzeko.

Token-ak denborarekin iraungi beharko lirateke, norbaitek token hori lortzen badu denbora luze batean erabili ahal ez izateko.

Aplikazioa

Proiektuan lanean egon garen bitartean ideia berriak bururatu zaizkigu, edota proiektuan dauden funtzioko edo atal desberdinak hobekuntzak ere. Esaterako, musika hari ezberdin baten kargatu beharko litzateke, momentuan hari nagusian guztia egiten denez aplikazioak askotan dena kargatzeko denbora gehiegi behar duelako.

Gomendioak momentu honetan elementu bakar bat erakusteko prest dago. Ideia hobea izango litzateke leihoa berri baten gomendio zerrenda bat egitea, edo filtroaren zerrendan bertan gomendio zerrenda kargatzea, label berri baten zehaztuz noiz zauden filtroen zerrenda ikusten edo gomendioen zerrenda ikusten.

Erreproduktorean momentuan ez dira albumen karatulak agertzen, zerbitzarian zein aplikazioan ez delako implementatu. Azkenik, erroreetan azaldu bezala, zerbitzariaren erantzuna konprobatzeko metoda hobetu beharko genuke, nahiz eta guztiz txarto ez egon

erantzuna irakurri aurretik erantzuna ondo edo txarto datorren aukera badugu hau erabili beharko genukeelako.

Arazoak

Zerbitzarian

Zerbitzarian izan ditugun arazoak, eta aurkitutako konponbideak:

- MySQL saioa, batzuetan, era seguruan hasten da, eta ez ditu errenkadak eguneratzen edota ezabatzen, Primary Key-ak erabili gabe. Horretarako `[SET SQL_SAFE_UPDATES = 0]` exekutatu izan dugu.
- Egiaztu egin behar da jasotzen ditugun datuak ez dutela ‘’’’’ edo ‘;’ karaktere antzerakoak, datu baseak ez interpretatzeko gaizki karaktere hauek.
- AES-ekin pasahitza enkriptatzerakoan, pasahitz eremua VARCHAR bada, hau desenkriptatzean null itzultzen du, string-a ez delako bera. Horregatik, pasahitz eremua LONGLOB-era aldatu dugu, gorde ahal den balore binario handiena.
- Gailu bat bakarrik ahal da konektatuta egon zerbitzarira denbora berberean bezero bakoitzeko, datu basean dugun last_login zutabearengatik. Hau hobetzeko, pentsatu izan dugun konponketa posiblea bezero bakoitzeko gailu erregistro bat izatea da, eta honekin token bakarra gailuko sortzea eta hori erabiltzea.
- API-aren metodo bakoitzean datu baseari conexioa egiten zitzaion. Hau, hasteko, ez da oso segurua, eta gainera datu basean, erabiltzailea edo pasahitz aldaketaren bat balego, aldaketa hau metodo guztietan egin beharko zen. Konpontzeko, klase berri bat sortu zen, honek datu basera conexioa egiten du, eta behin konektatuta bueltatzen dio API-ari, horrela edozein datu aldatzen bada, behin bakarrik aldatu behar dugu eta leku guztietan gertatuko da aldaketa.

- PHP-k defektuz ez du JWT-a (JSON Web Token) sortzeko aukera ematen. Hau konpontzeko, JWT-a sortzeko PHP-rako dagoen liburutegi ofiziala erabiltzea erabaki izan dugu. Liburutegi honekin JSON Web Token-ak sortu eta partseatu ahal ditugu, eta hauen balioa zein iraunkortasuna konprobatu, guri lana errazten. [Erabilitako erremintetan izendatuta]

Aplikazioan

Aplikazioan izan ditugun arazoak, eta aurkitutako konponbideak:

- MediaPlayer-entzako kontroladore bat sortzen ahalegindu ginen, baina horretarako kontroladorea exekutatzen zen activity-aren testuinguru pasatu behar zenez, ezin izan genuen egin. Hori konpontzeko, MediaPlayerFactory bat sortu dugu.
- HTTPS-a implementatu genuenean zerbitzarian, ziurtagiri guztiak ez genituenez, aplikazioan kodea aldatu genuen konprobaketa hauek saltatzeko, kode hau eranskinetan joango da. Esan behar da gaur egun kode zati hori ez dagoela jarrita.
- Aplikazioan sartu izan dugun letra iturburuak ez dira bistaratzen aplikazioa mugikorrean instalatu eta gero. Konponketa posible bat dugu eskutan, baina oraindik ez dugu implementatu.
- Hasieran playlista arraylist baten gordetzen zen, baina hauek ezin ziren aplikazioaren SharedPreferences atalean gorde. Interneten serializazio klase bat topatu genuen eta honekin string bezala gordetzen ahalegindu ginen, baina ez zen ondo gordetzen. Bukaeran, arraylist hori set objektu batera bihurtu genuen, SET-ak SharedPreferences atalean gorde ahal direlako.

- Berriz ere, aplikazio aldetik, kontu handia izan dugu karaktereekin. Aplikazioan beran egin izan dugu beste kontrol bat egin dugu eMail-ekin, erabiltzaileekin, pasahitz hauek maiuskula, minuskula, gidoi edo gidoi baxuekin bakarrik egin ahal izango dira, zerbitzarian bezala, ez da utziko karaktere berezirik sartzen, datu basearekin arazorik ez izateko.
- JSON-ekin frogak egiteko eclipse erabili izan dugu, baina arazo bat genuen, eclipsek ez du json.org-eko import zuzena baimentzen. Beraz interneten topatu izan dugun liburutegi bat erabili dugu eta eskuz importatu egin behar izan dugu.
- Aplikazioaren garapena egoera aurreratu batera heltzean, try eta throw-ak bata bestearekin talka egiten hasi ziren beste funtzi batzuetako try eta throw-ekin, adibide bat aipatzeko, funtzi bateko catch-ak saltatzen bazuen, hurrengo zetorren funtzioa ez zen exekutatzen, beraz denak berrikusi genituen.

Tarteko arazoak

Zerbitzari eta aplikazio artean edo kanpotik izan ditugun arazoak, eta aurkitutako konponbideak:

- Get eskaerak hasterakoan, honen erantzuna irakurri baino lehen honen kodea irakurtzen genuen (erantzuna ondo edo txarto zatorren jakiteko). Hau egiterakoan konexio berri bat irekitzen zen, eta konexioa jada irekita zegoenez erroreak ematen zituen. Konpontzeko, mezuaren kodea irakurri beharrean (konexio berri bat irekiz) mezia irakurtzen dugu, eta hutsik badator mezia ona ez dela ulertzen dugu (mezia

ona izango da beti edukia itzultzen badu, nahiz eta eduki hori ez izan eskatutakoa, token-a ez bada zuzena esaterakoan).

- Aplikazioan egiten duen konprobaketak zerbitzariko erantzunei dagokionez, erroreak ematen ditu zuzenak partseatzen direnean, beraz konprobaketa hauek kendu egin dira aplikaziotik momentuz. Konponketa baten bila hasiko gara denbora dagoenean.

ERANSKINAK

Ziurtagiri konprobaketa saltatzeko kodea

```
try {

    HttpsURLConnection.setDefaultHostnameVerifier(new HostnameVerifier() {
        public boolean verify(String hostname, SSLSession session) {
            return true;
        }
    });
    SSLContext context = SSLContext.getInstance("TLS");
    context.init(null, new X509TrustManager[]{new X509TrustManager() {
        public void checkClientTrusted(X509Certificate[] chain,
                                       String authType) throws CertificateException
    }
        public void checkServerTrusted(X509Certificate[] chain,
                                       String authType) throws CertificateException
    }
        public X509Certificate[] getAcceptedIssuers() {
            return new X509Certificate[0];
        }
    }}, new SecureRandom());
    HttpsURLConnection.setDefaultSSLSocketFactory(
        context.getSocketFactory());
} catch (Exception e) { // should never happen
    e.printStackTrace();
}
```

HTTP-n POST egiteko erabiltzen genuen kodea

```
//POST on http

public void sendPetition() throws IOException {

    //Allow usage of some sentences
    StrictMode.ThreadPolicy policy = new
    StrictMode.ThreadPolicy.Builder().permitAll().build();
    StrictMode.setThreadPolicy(policy);

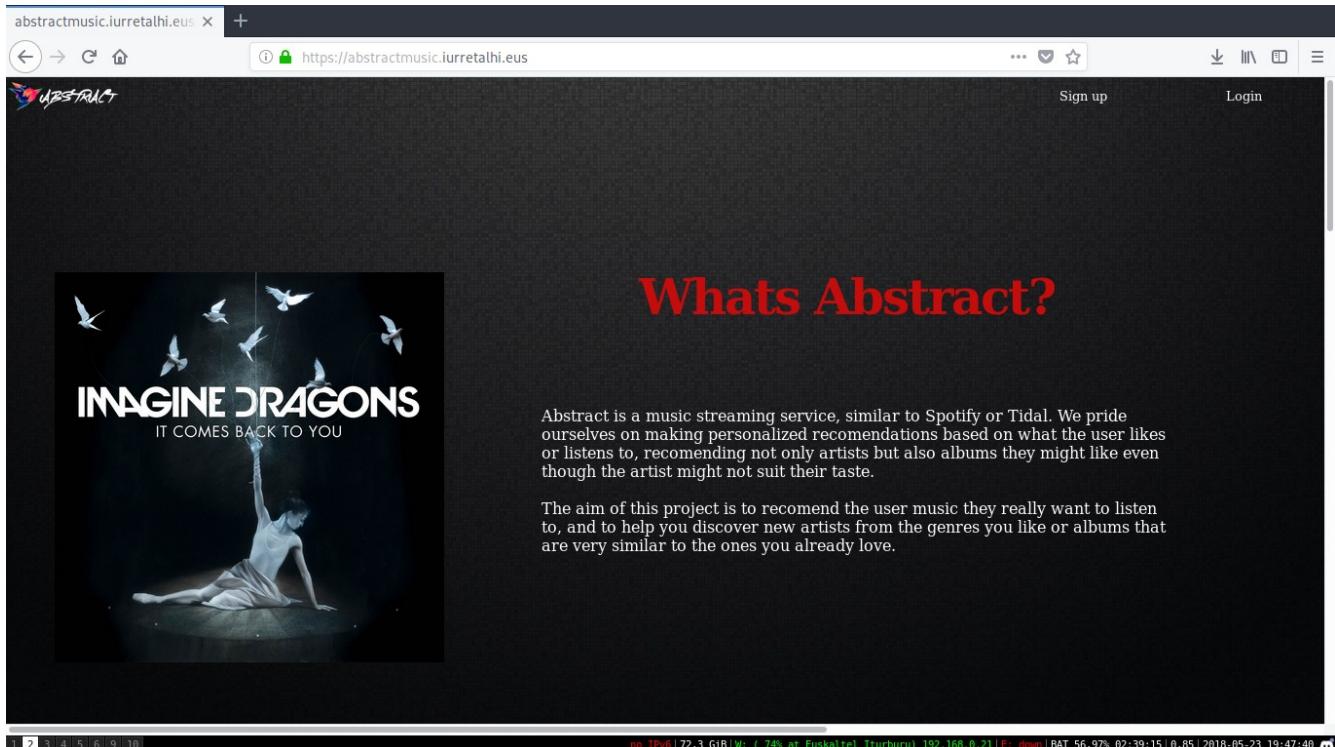
    //URL build
    URL url = new URL("http://abstractmusic.iurretalhi.eus/api.php/registeruser?
%22");
    URLConnection con = url.openConnection();
    HttpURLConnection http = (HttpURLConnection)con;
    http.setRequestMethod("POST"); // PUT is another valid option
    http.setDoOutput(true);
    http.setDoInput(true);

    //Body build
    String query = "username='prueba' & password='prueba' &
email='prueba@prueba.com'" ;
    byte[] out = query.toString().getBytes(StandardCharsets.UTF_8);
    int length = out.length;

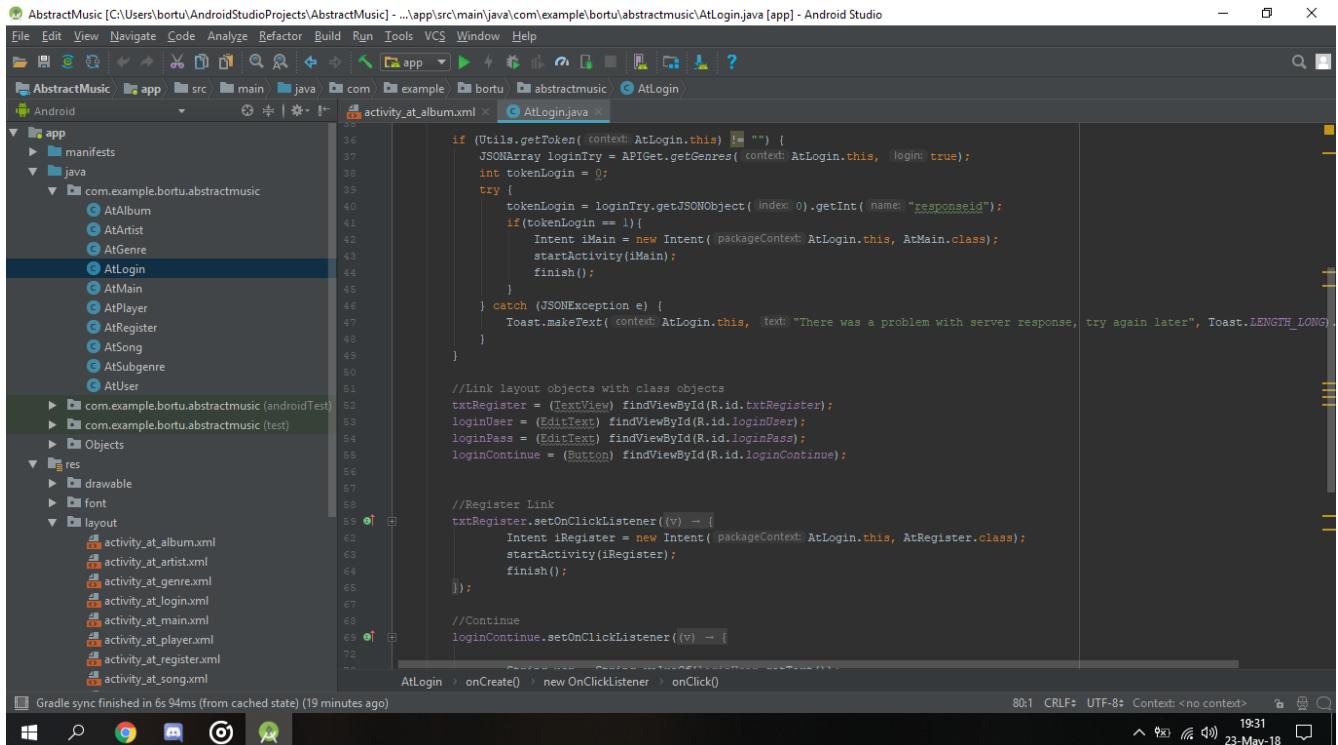
    //Body write
    http.setFixedLengthStreamingMode(length);
    http.setRequestProperty("Content-Type", "application/x-www-form-urlencoded;
charset=UTF-8");
    http.connect();
    try(OutputStream os = http.getOutputStream()) {
        os.write(out);
    }

    //Read response
    DataInputStream input = new DataInputStream(con.getInputStream());
    for (int c = input.read(); c != -1; c = input.read())
        System.out.print((char) c);
    input.close();

    http.disconnect();
}
```

Abstract web-gunea

Android Studio



```

if (Utils.getToken( context: AtLogin.this ) != "") {
    JSONArray loginTry = APIGet.getGenres( context: AtLogin.this, login: true );
    int tokenLogin = 0;
    try {
        tokenLogin = loginTry.getJSONObject( index: 0 ).getInt( name: "responseid" );
        if( tokenLogin == 1 ) {
            Intent iMain = new Intent( packageContext: AtLogin.this, AtMain.class );
            startActivity( iMain );
            finish();
        }
    } catch (JSONException e) {
        Toast.makeText( context: AtLogin.this, text: "There was a problem with server response, try again later", Toast.LENGTH_LONG );
    }
}

//Link layout objects with class objects
txtRegister = (TextView) findViewById( R.id.txtRegister );
loginUser = (EditText) findViewById( R.id.loginUser );
loginPass = (EditText) findViewById( R.id.loginPass );
loginContinue = (Button) findViewById( R.id.loginContinue );

//Register Link
txtRegister.setOnClickListener( (v) -> {
    Intent iRegister = new Intent( packageContext: AtLogin.this, AtRegister.class );
    startActivity( iRegister );
    finish();
});

//Continue
loginContinue.setOnClickListener( (v) -> {
}
)

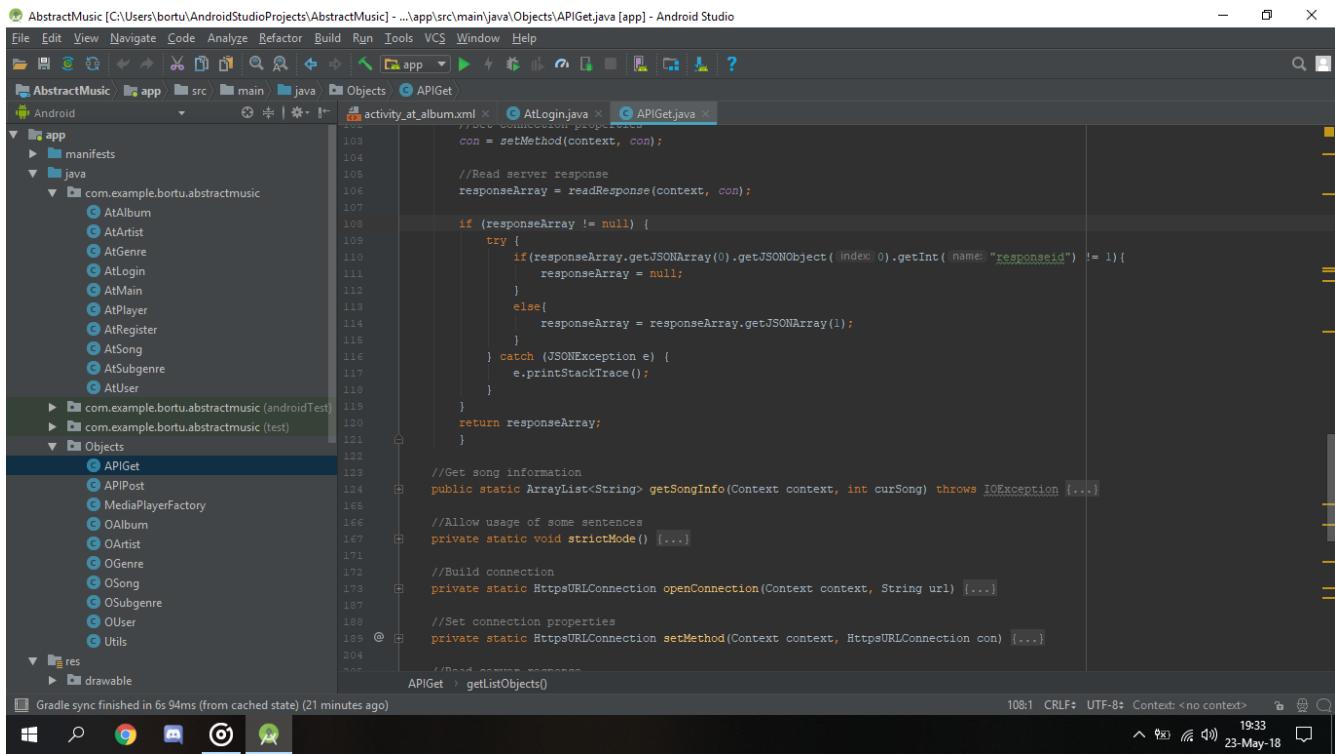
```

AtLogin > onCreate() > new OnClickListener > onClick()

Gradle sync finished in 6s 94ms (from cached state) (19 minutes ago)

80:1 CRLF: UTF-8: Context: <no context> 19:31 23-May-18

JSON liburutegia



The screenshot shows the Android Studio interface with the project 'AbstractMusic' open. The code editor displays the file 'APIGet.java' under the package 'com.example.bortu.abstractmusic'. The code implements a static method to get song information from a JSON response. The code includes error handling for JSONException and IOException, and uses HttpURLConnection to make the request.

```
103     con = setMethod(context, con);
104
105     //Read server response
106     responseArray = readResponse(context, con);
107
108     if (responseArray != null) {
109         try {
110             if(responseArray.getJSONArray(0).getJSONObject(index).getInt("responseid") != 1) {
111                 responseArray = null;
112             }
113             else{
114                 responseArray = responseArray.getJSONArray(1);
115             }
116         } catch (JSONException e) {
117             e.printStackTrace();
118         }
119     }
120     return responseArray;
121 }
122
123 //Get song information
124 public static ArrayList<String> getSongInfo(Context context, int curSong) throws IOException {
125
126     //Allow usage of some sentences
127     private static void strictMode() {...}
128
129     //Build connection
130     private staticHttpsURLConnection openConnection(Context context, String url) {...}
131
132     //Set connection properties
133     private staticHttpsURLConnection setMethod(Context context,HttpsURLConnection con) {...}
134
135     //Parse response
136     APIGet > getListObjects()
137 }
```

Gradle sync finished in 6s 94ms (from cached state) (21 minutes ago)

108:1 CRLF: UTF-8: Context: <no context> 19:33
23-May-18

Phpmyadmin

The screenshot shows the phpMyAdmin interface for the 'abstract' database. The left sidebar lists various tables: albums, albums_artists, albums_songs, artists, bans, bugs, bugs_users, genres, genres_artists, genres_independent_art, genres_subgenres, independent_albums, independent_albums_inc, independent_artists, independent_songs, playlists, playlists_independent_s, playlists_songs, reproductions, reproductions_independ, reproductions_songs, songs, and subgenres. The 'songs' table is selected in the main area. The table structure includes columns for id, name, and path. The data shows 11 rows of song names. A footer bar at the bottom displays system information.

	id	name	path
1	1	Once Upon A Time In Waverland	NULL
2	2	King Of San Felipe	NULL
3	3	Surf Me Baby	NULL
4	4	Ding O Stomp	NULL
5	5	Just A Party	NULL
6	6	Coconut	NULL
7	7	Calavera	NULL
8	8	Twist Of The Beast	NULL
9	9	Vanilla Caos	NULL
10	10	Leaving San Felipe	NULL
11	11	Small Town USA	NULL

no IPv6 | 72,3 GiB | W: (7% at Euskaltel Iturburu) 192.168.0.21 | E: down | BAT 60,78% 02:31:55 | 1,14 | 2018-05-23 19:38:00

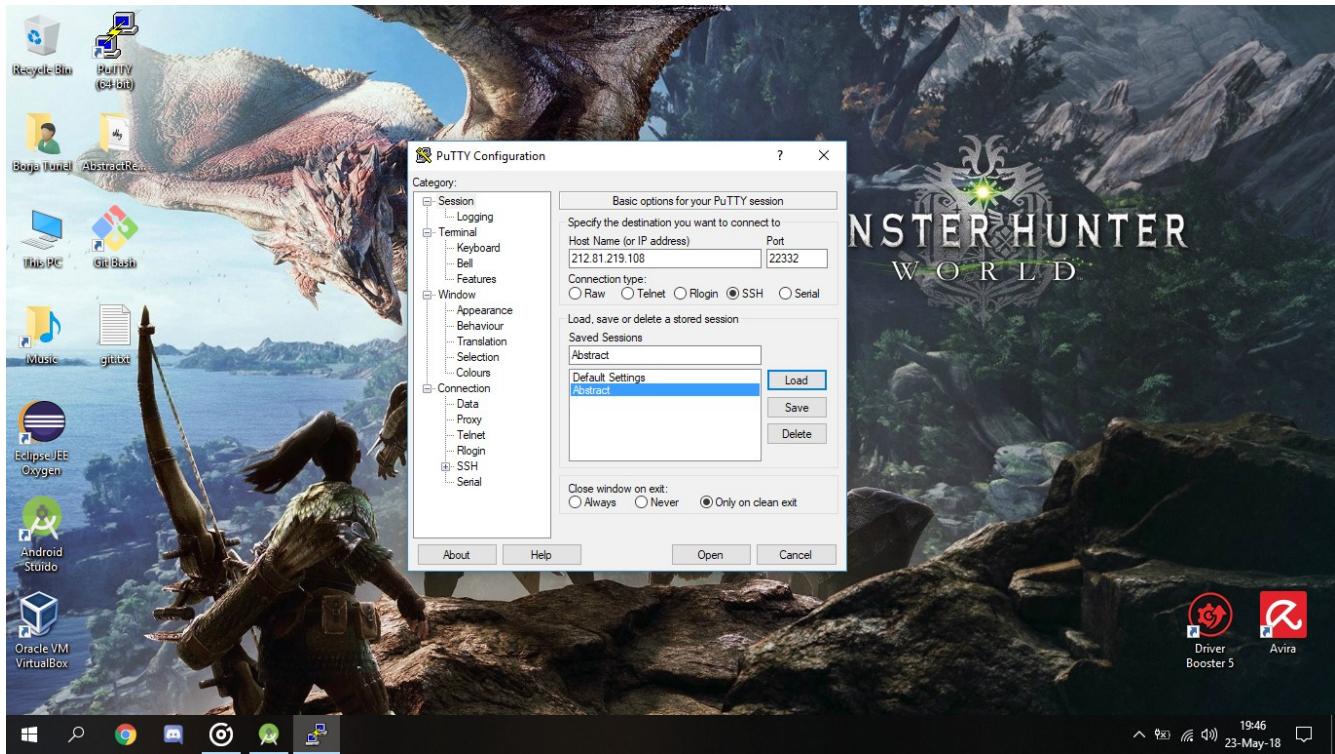
Postman

The screenshot shows the Postman application interface. At the top, there's a navigation bar with File, Edit, View, Help, New, Import, Runner, and a workspace dropdown set to "My Workspace". Below the bar, there are tabs for Collections, History, and Abstract (which contains 1 request). A search bar labeled "Filter" is present. In the center, a request card is displayed for "http://abstractmusic.urretalhi.eus/api.php/login". The method is set to GET, and the URL is https://abstractmusic.urretalhi.eus/api.php/artistsbygenre. The "Authorization" tab is selected, showing "Inherit auth from parent". The response status is 200 OK, time is 627 ms, and size is 1.22 KB. The response body is shown in Pretty, Raw, Preview, and JSON formats. The JSON response is:

```
1 [ [ { 2   "responseid": "1", 3     "description": "Key matches, sending response" 4   }, 5   { 6     "id": "1", 7       "name": "Buckethead", 8       "description": null, 9       "photo": null 10  } 11 ] 12 }
```

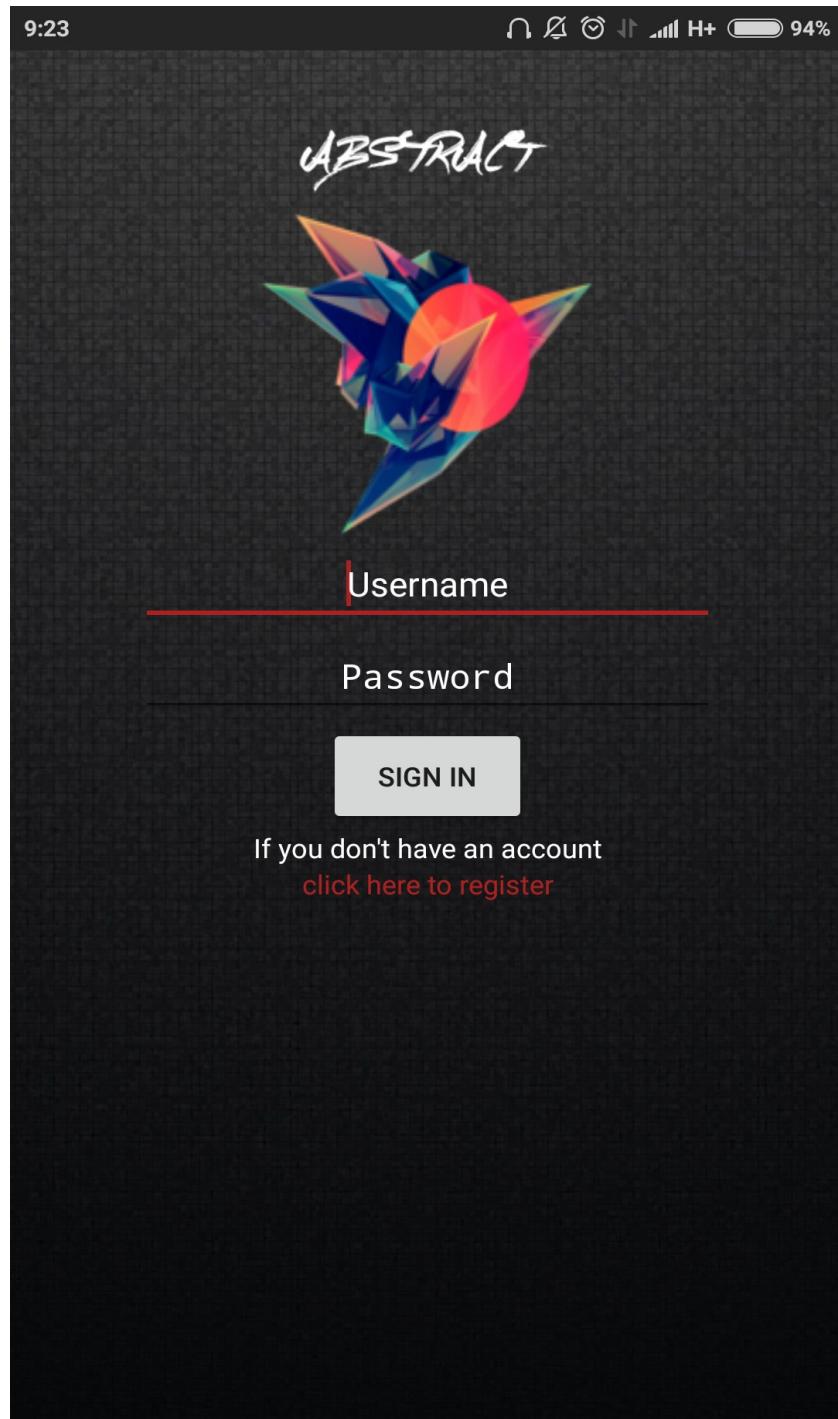
At the bottom, there are navigation icons for back, forward, search, and other functions, along with a status bar showing network information like IPv6, battery level, and system date.

Putty

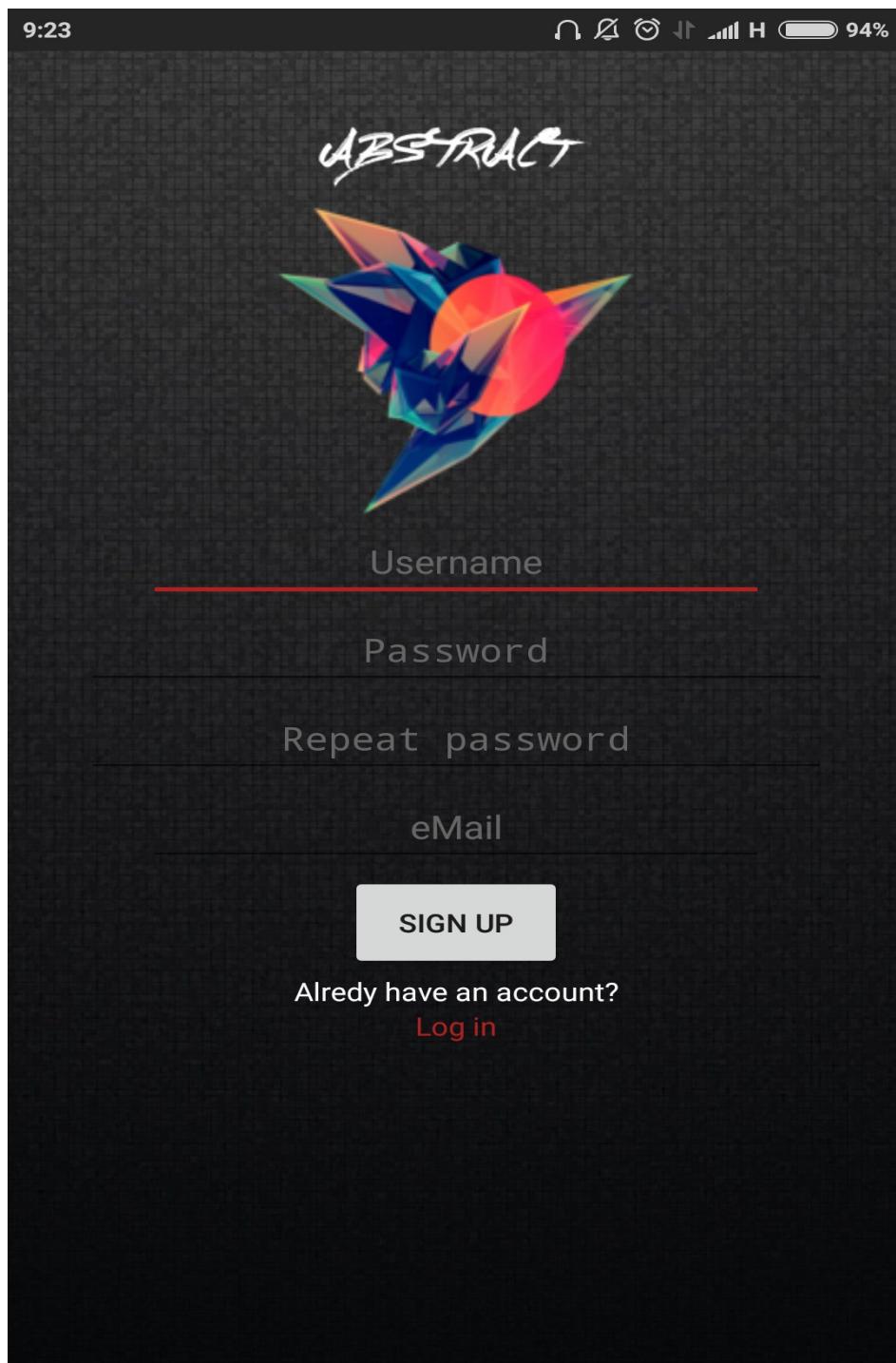


Aplikazioak dituen pantaila ezberdinen diseinua

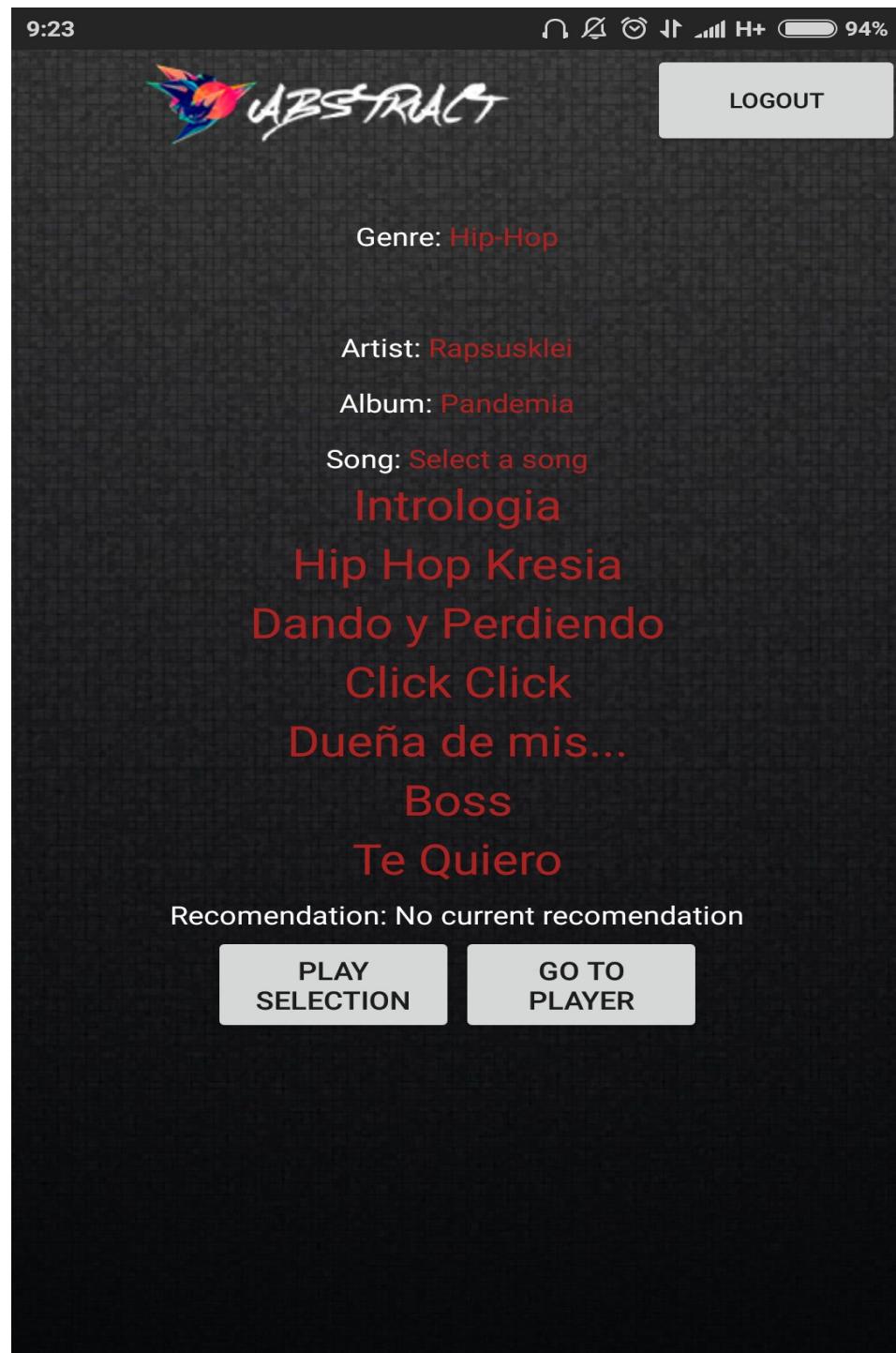
Login



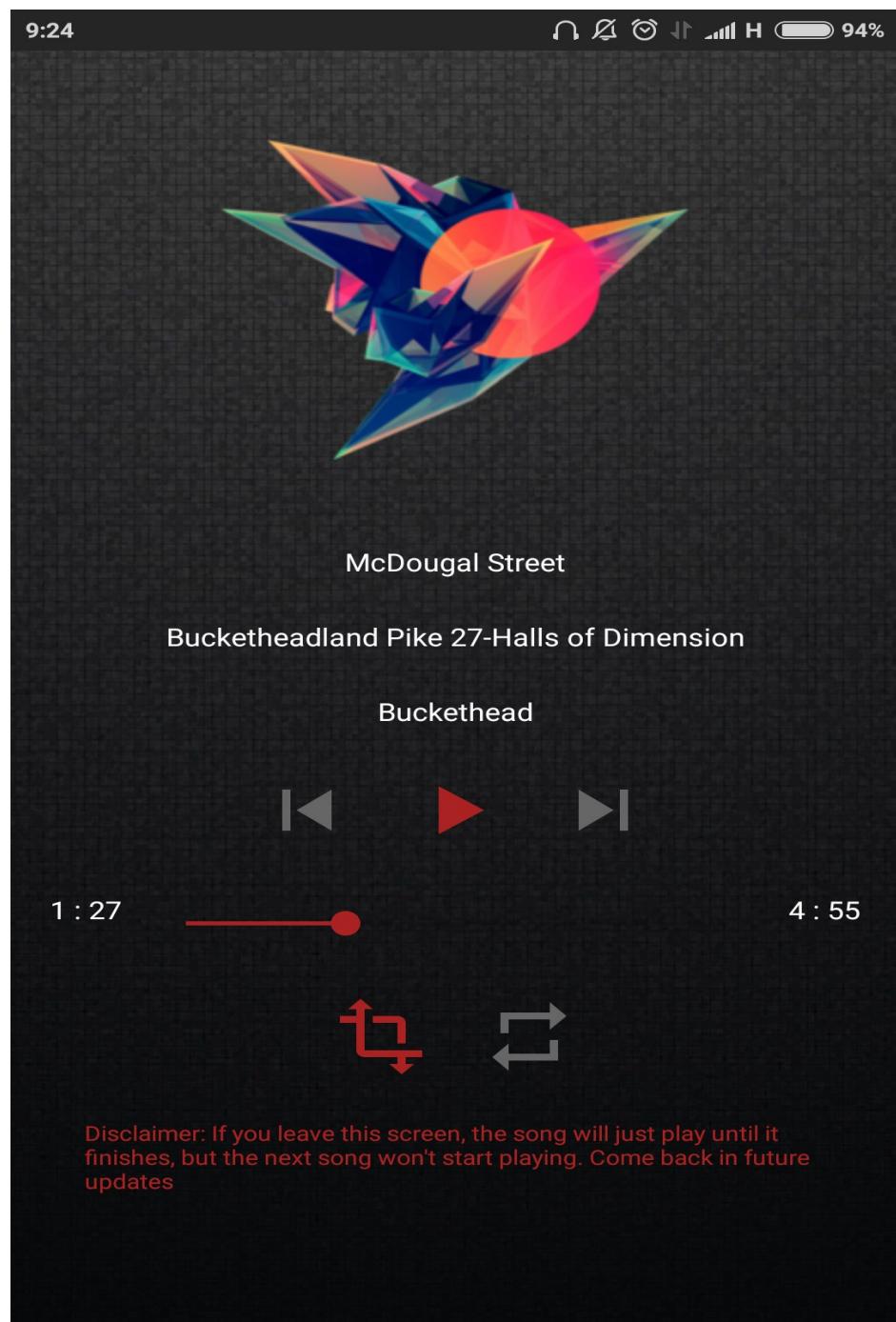
Register



Menu



Player



API-ra egin ahal diren eskaera (eta hauen erantzun) posibleak

API-ari eskaerak egiteko honen helbidea erabiliz eta hurrengo eskaerak eginez lortu daitezke datuak:

/songs -> Bi array dituen array bat bueltatzen du, bat token-aren erantzunarekin, eta bestea abestien informazioarekin

/albums -> Bi array dituen array bat bueltatzen du, bat token-aren erantzunarekin, eta bestea diskoen informazioarekin

/artists -> Bi array dituen array bat bueltatzen du, bat token-aren erantzunarekin, eta bestea artisten informazioarekin

/genres -> Bi array dituen array bat bueltatzen du, bat token-aren erantzunarekin, eta bestea generoen informazioarekin

/playlists -> Bi array dituen array bat bueltatzen du, bat token-aren erantzunarekin, eta bestea erreprroduzko listen informazioarekin

/subgenres -> Bi array dituen array bat bueltatzen du, bat token-aren erantzunarekin, eta bestea subgeneroen informazioarekin

/subgenresbygenre -> Bi array dituen array bat bueltatzen du, bat token-aren erantzunarekin, eta bestea subgeneroen informazioarekin. Kasu honetan, generoen ID bat bidaltzen ez bada, bigarren array-a hutsik joango da.

/artistsbyalbum -> Bi array dituen array bat bueltatzen du, bat token-aren erantzunarekin, eta bestea artisten informazioarekin. Kasu honetan, albumen ID bat bidaltzen ez bada, bigarren array-a hutsik joango da.

/albumsbyartist -> Bi array dituen array bat bueltatzen du, bat token-aren erantzunarekin, eta bestea diskoen informazioarekin. Kasu honetan, artistaren ID bat bidaltzen ez bada, bigarren array-a hutsik joango da.

/songsbyalbum -> Bi array dituen array bat bueltatzen du, bat token-aren erantzunarekin, eta bestea abestien informazioarekin. Kasu honetan, diskaren ID bat bidaltzen ez bada, bigarren array-a hutsik joango da.

/songsbyartist -> Bi array dituen array bat bueltatzen du, bat token-aren erantzunarekin, eta bestea abestien informazioarekin. Kasu honetan, artistaren ID bat bidaltzen ez bada, bigarren array-a hutsik joango da.

/songsbsubgenre -> Bi array dituen array bat bueltatzen du, bat token-aren erantzunarekin, eta bestea abestien informazioarekin. Kasu honetan, subgeneroaren ID bat bidaltzen ez bada, bigarren array-a hutsik joango da.

/artistsbygenre -> Bi array dituen array bat bueltatzen du, bat token-aren erantzunarekin, eta bestea artisten informazioarekin. Kasu honetan, generoaren ID bat bidaltzen ez bada, bigarren array-a hutsik joango da.

/registeruser -> Erabiltzailearen izena, email-a eta pasahitza bidaltzen zaizkio, datu hauek konprobatu egiten ditu, eta ondoren datu basean gordetzen du datua. Azkenik, bezeroari array bat bueltatzen dio erantzun moduan.

/login -> Erabiltzailearen izena eta pasahitza pasatzen zaizkio, datu hauek konprobatu egiten ditu, datuak ondo badaude, token-a bueltazen du, bestela ez du tokenik bueltatzen.

/playsong -> Erabiltzaileak token-a pasatzen dio, eta API-ak erabiltzaileak eskatutako abestia pasatzen dio bezeroari. Abestiaren ID-a pasatzen ez bazaio, ez dio abestirik bidaliko

/songinformation -> Bi array dituen array bat bueltatzen du, bat token-aren erantzunarekin, eta bestea abestiaren datu guztiekin, artista eta disko-a barne. Kasu honetan, eskatutakoaren ID bat bidaltzen ez bada, bigarren array-a hutsik joango da.

/albumsbsubgenre -> Bi array dituen array bat bueltatzen du, bat token-aren erantzunarekin, eta bestea diskoen informazioarekin. Kasu honetan, generoaren ID bat bidaltzen ez bada, bigarren array-a hutsik joango da.

GIT

Zerbitzarian erabilitako erreminten konfiguraziona edota aplikazioaren kodea ikusi nahi bada, dena hurrengo helbidean kontsultatu daiteke:

https://github.com/lurretaLHII-infor/2018_abstractmusicplayer