



Mohammed Al-Jewari, Sampo Savolainen, Johanna Toivanen, Jesper Ojala

Harmonia 2.0 -ohjelmisto

8.5.2023

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintäteknologian tutkinto-ohjelma

Projektin toteutusdokumentti

Sisällys

Harmonia 2.0 -ohjelmisto	3
Metropolia Ammattikorkeakoulu	3
Insinööri (AMK)	3
Tieto- ja viestintäteknologian tutkinto-ohjelma	3
Projektin toteutusdokumentti	3
1. Johdanto	5
2. Käyttöönotto	5
3. Tuotteen vaatimukset	5
4. Käyttäjäroolit ja käyttötapaukset	6
5. Ohjelmiston Tietorakenne sekä ominaisuudet	8
Websockets	9
6. Ohjelmiston Rakenne	10
Moduulit	10
Arkkitehtuuri	13
7. Testaus	15
8. Ohjelmiston toiminta	16
9. Lokalisaatio	20
10. Kehitysprosessi ja kehitysvaiheen tekniikat	20
Miksi SCRUM?	21
11. Jatkokehitysideat	21
12. Yhteenveto	22

1. Johdanto

Tämä dokumentti kuvaa Harmonia-nimisen viestintäsovelluksen vaatimuksia, tietomallia, arkkitehtuuria ja toimintaa, sekä sovelluksen kehitysprosessia. Tämä dokumentti on tarkoitettu kehittäjille, jotka haluavat ymmärtää kyseisen ohjelmiston toiminnan ja kehityksen perusteet.

Harmonia on viestintäsovellus, joka tarjoaa käyttäjilleen yksityisemmän ja yksinkertaisemman käyttöliittymän kuin muut suositut viestintäsovellukset. Sovelluksessa on sisäänrakennettuja lisäosia, jotka korvaavat muissa viestintäsovelluksissa käytettyjä bottiohjelmia. Harmonia on myös sosiaalisempi, ja käyttäjän mieltymyksiin mukautuvampi, sillä se ehdottaa servereitä/yhteisöjä perustuen käyttäjän mielenkiinnon kohteisiin, kuten suosikkipelien tai muiden samankaltaisten kiinnostusten mukaan.

2. Käyttöönotto

Nykyisessä tilassa ohjelmiston voi ottaa käyttöön joko kloonaamalla arkiston Github sivuiltamme ja rakentamalla se lähdetiedostoista tai lataamalla valmiiksi rakennetun jar/exe tiedoston arkiston README osiosta löytyvästä linkistä tai Releases osiosta:

<https://github.com/Jihau/Harmonia/releases/>

3. Tuotteen vaatimukset

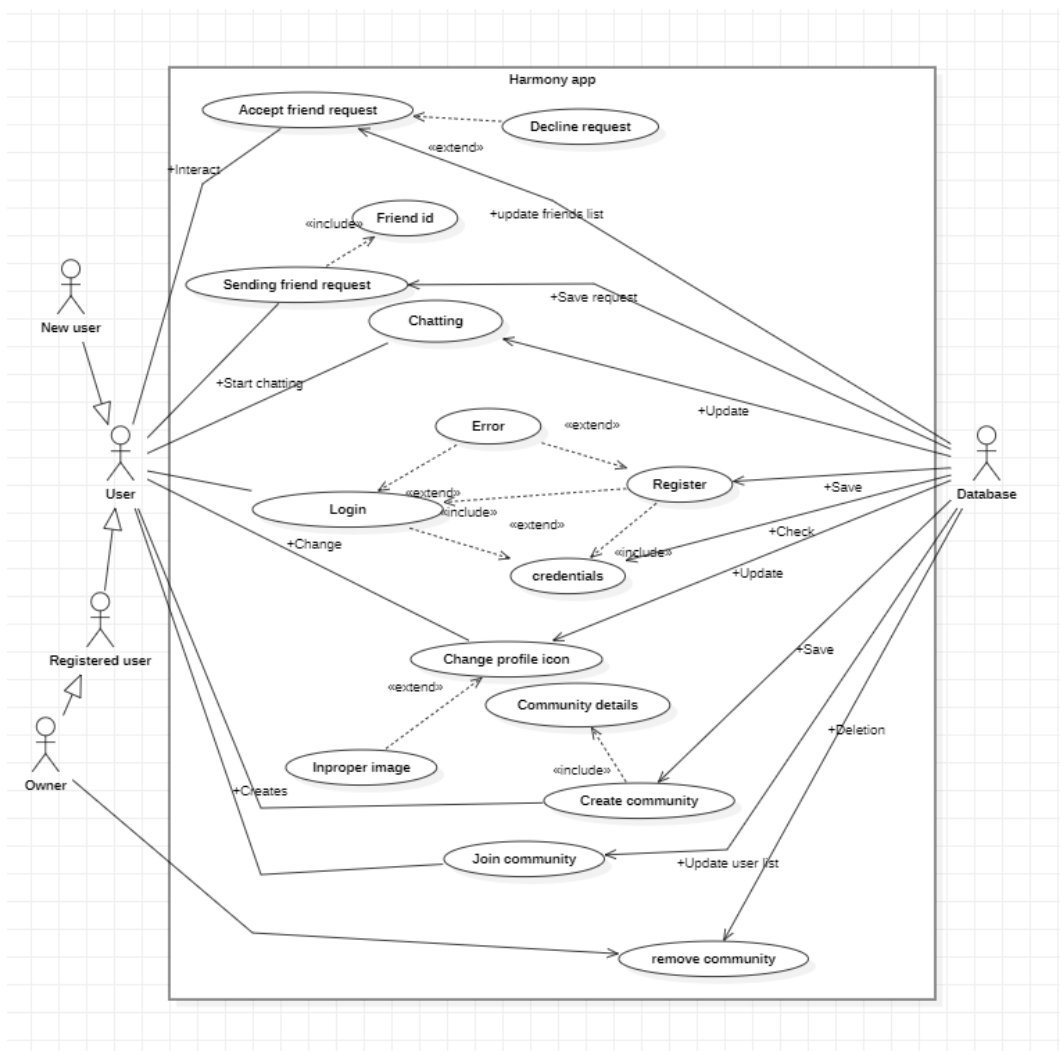
Harmonia nimisen sovelluksen tarve syntyi käyttäjien tarpeesta yksityisempään ja yksinkertaisempaan viestintäsovellukseen. Sovelluksen sisäänrakennetut lisäohjelmat korvaavat vaikeakäyttöiset botti-ohjelmat, ja mahdollistavat yhteisöjen löytymisen kiinnostuksen mukaan. Sovelluksen päätavoite on tarjota käyttäjille helppokäyttöinen sekä turvallinen viestintäympäristö.

4. Käyttäjäroolit ja käyttötapaukset

Käyttäjäroolit:

Harmonialla on **normaaleja käyttäjiä(user)**, jotka voivat lähettää viestejä ystäville, liittyä eri yhteisöihin osallistua chat-huoneisiin yhteisön sisällä sekä **palvelimen omistaja(Server owner)**, jolla on ylläpitäjän oikeudet, kuten kyky hallita palvelimen jäseniä ja chat-huoneita. Ownerilla on myös oikeudet myös hallita tekstikanavia.

Käyttötapaukset:

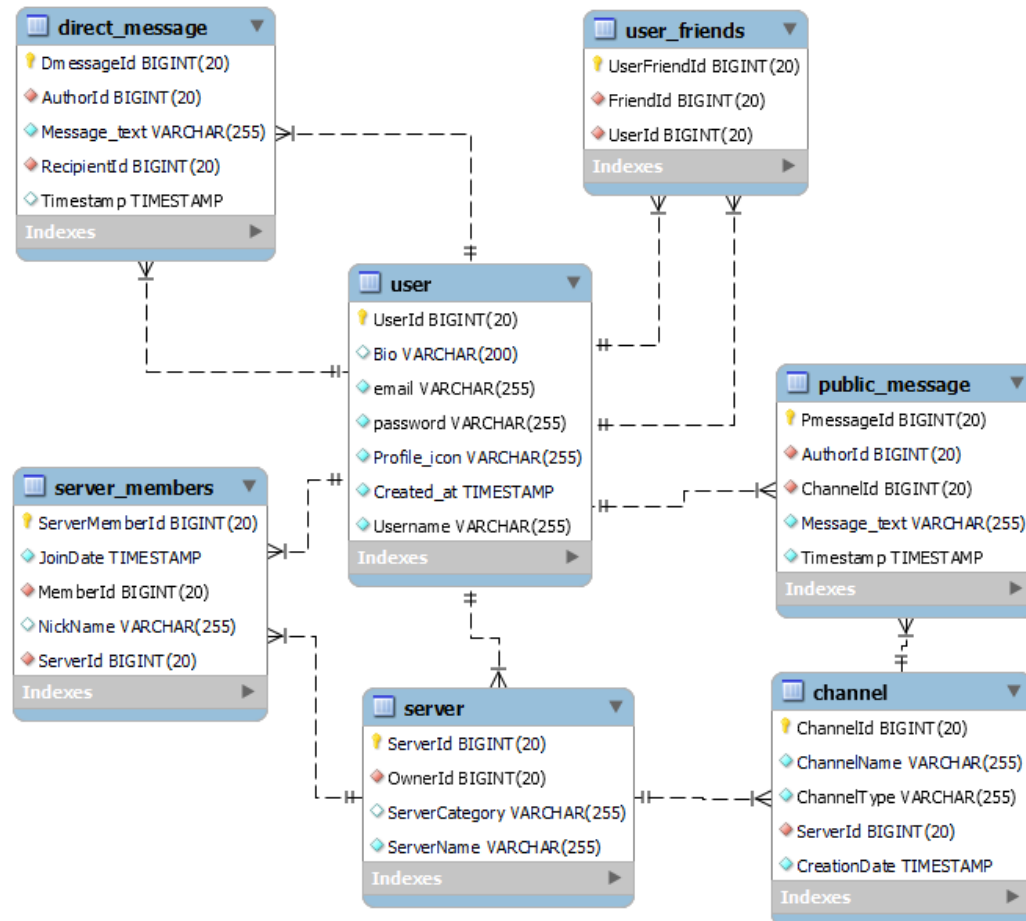


Kuva 1: Harmonian sovelluksen käyttötapauskaavio.

Käyttäjä tarinoita:

1. "As an online gamer, I need a good communication application to communicate with friends."
2. "As an online personality I want to be able to send news and updates to my fans/followers."
3. "As an employee I want to be able to send messages to my boss and co-workers to improve communication."
4. "As a new user I want to have easy and comprehensive search functions for communities so that I can find other people with the same interest as me."
5. "As a gamer dad, I want an easily accessible chatting application without having a big knowledge of IT stuff."
6. "As a manager I want to be able to send a message to multiple of my employees to save time"
7. "As a community admin I want to have tools to manage communities I own so that I can keep the communities I own in order and spam or conflict free."
8. "As a cyber security student I want to have the ability to delete my own messages, so that I can control my personal data and protect my privacy. This feature will allow me to have private conversations without the need to worry."

5. Ohjelmiston Tietorakenne sekä ominaisuudet



Kuva 2: Harmonia-sovelluksen tietorakenne koostuu seitsemästä taulusta.

Server -taulu sisältää tiedot palvelimesta, kuten sen omistajan ID:n, kategorian ja nimen. Tämä taulu on yhteydessä kahteen muuhun tauluun: server_members ja channel.

Server_members -taulu sisältää tiedot palvelimen (eli serverin) jäsenistä, kuten jäsenen ID:n, liittymispäivän, käyttäjänimen sekä nicknimen, jonka käyttäjä voi halutessaan valita. Tämä taulu on yhteydessä kolmeen muuhun tauluun: server, user ja direct_message.

User -taulu sisältää tiedot käyttäjästä, kuten sähköpostiosoitteen, salasanan, profiilikuvan, käyttäjänimen, sekä tunnuksen luontipäivän. Tämä taulu on yhteydessä kahteen muuhun tauluun: server_members ja direct_message.

Channel -taulu sisältää tiedot kanavasta, joka sijaitsee serverillä, kuten sen nimen, tyyppin ja luomispäivän. Tämä taulu on yhteydessä kahteen muuhun tauluun: server ja public_message.

Public_message -taulu sisältää tiedot julkisista viesteistä käyttäjien välillä, servereiden kanavilla, kuten viestin tekstin, lähettäjän ID:n ja lähetysajan. Tämä taulu on yhteydessä kahteen muuhun tauluun: channel ja user.

Direct_message -taulu sisältää tiedot yksityisistä viesteistä käyttäjien välillä, kuten viestin tekstin, lähettäjän ja vastaanottajan ID:t, sekä viestin lähetysajan. Tämä taulu on yhteydessä kahteen muuhun tauluun: server_members ja user.

User_friends -taulu sisältää tiedot käyttäjien ystäväistä. Tämä taulu on liitetty ainoastaan User-tauluun.

Käyttötapauksena voisimme kuvitella tilanteen, jossa käyttäjä haluaa lähettää yksityisviestejä toiselle käyttäjälle Harmonia-sovelluksen kautta. Käyttäjän tulee ensin kirjautua sisään sovellukseen ja etsiä haluamansa vastaanottajan käyttäjänimi. Kun vastaanottaja on löytynyt, käyttäjä luo kirjoittamalla viestikenttään uuden direct_message -tauluun rivin, jossa määritellään viestin teksti, lähettäjän ID, vastaanottajan ID ja lähetysaika. Näiden tietojen avulla viesti voidaan tallentaa tietokantaan. Vastaavasti kun käyttäjä haluaa lukea vastaanotetut yksityisviestit, sovellus hakee direct_message taulusta käyttäjän vastaanottamien viestien rivit ja näyttää ne käyttäjälle.

Websockets

Harmoniassa käytämme websockets-tekniikkaa päivittämään suorat viestit verkkosovelluksen osassa reaaliajassa aina kun uusi viesti saapuu. Ilman websocketsia käyttäjän olisi päivitettävä sivu manuaalisesti nähdäkseen, onko uusia viestejä, mutta websocketsin ansiosta viestit päivittyvät automaattisesti ilman viivettä. Tämä tekee viestien lähettämisestä käyttäjälle paljon kätevämpää ja tehokkaampaa.



Kuva 3: WebSocket rakenteen idea visuaalisesti esitettynä

6. Ohjelmiston Rakenne

Moduulit

Harmonian ohjelmisto on jaettu kahteen moduuliin: backend ja frontend. Moduulien jakaminen kahteen on hyödyllistä, koska se mahdollistaa kunkin moduulin erillisen kehittämisen ja testaamisen. Tämä tarkoittaa sitä, että jos jokin muutos on tehtävä yhteen moduuliin, muutoksen vaikutus muihin moduuleihin on todennäköisesti minimaalinen, mikä vähentää virheiden mahdollisuutta ja tekee kehittämisestä helpompaa.

Jos Harmonia-sovelluksen frontend haluttaisiin tulevaisuudessa siirtää JavaFX:ltä esimerkiksi Reactille, viimeistään siinä vaiheessa kun tuki JavaFX:ään päättyy (nykytiedoin 2025), modulaarinen lähestymistapa helpottaa tätä siirtymää. Koska moduulit ovat erillisiä, esimerkiksi Reactin käyttöönotto frontendin osalta voidaan tehdä ilman, että se vaikuttaa backendiin, mikä mahdollistaa joustavuuden, ja pienentää riskiä mahdollisille virheille.

Lisäksi tämä modulaarinen lähestymistapa mahdollistaa sen, että tiimimme kehittäjät voivat keskittyä yhteen moduuliin kerrallaan, mikä helpottaa tiimityötä.

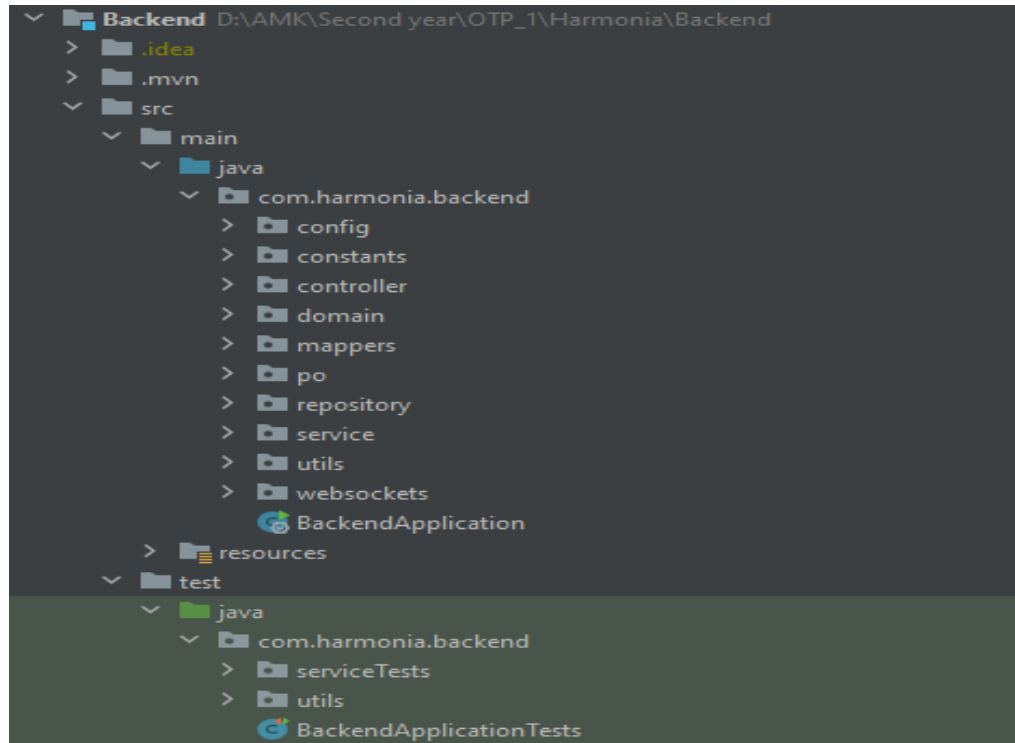
Backend

Backend käyttää Spring Frameworkia ja Javan ohjelmointikieltä RESTful-rajapintojen rakentamiseen käyttäjähallintaa, palvelinhallintaa ja viestintää varten.

Backend-moduulin (kuva 3.) arkkitehtuuri noudattaa Model-View-Controller(MVC) -mallia seuraavasti:

- Mallikerros (Model): Koostuu domain- ja repository-luokista.
 - Domain luokkien tarkoitus on mallintaa sovellusalueen käsitteitä ja käyttäytymistä java-koodin muodossa. Domain luokissa määritetään myös liiketoiminnan logiikka, validoidaan tietoa ja käyttötapauksia, jotka ohjaavat lopulta sovelluksen toimintaa. Näiden luokkien avulla voidaan helposti ymmärtää Harmonia-sovelluksen käyttäytymistä ja muutoksia, sillä ne ovat helposti ylläpidettäviä ja testattavia.
 - Repository-luokat vastaavat tiedon tallentamisesta ja hausta tietokannasta. Näiden luokkien avulla sovelluksen data- ja liiketoimintalogiikka voidaan erottaa toisistaan, mikä parantaa sovelluksen ylläpidettävyyttä, ja testattavuutta.
- Näkymäkerros (View): Koostuu controller-luokista.

- Controller-luokat käsittelevät HTTP-pyyntöjä ja vastauksia, ja käyttävät service-luokkia vastaavan liiketoimintalogiikan suorittamiseen välittämällä niille kyseiset HTTP-pyynnöt.
- Palvelu-, eli service-luokat ovat osa MVC-arkkitehtuuria, ja ne toteuttavat liiketoimintalogiikan, ja käyttävät repository-luokkia vuorovaikutuksessa tietokannan kanssa. Lopullinen tulos on RESTful-rajapinta, jonka avulla frontend voi kommunikoida sovelluksen backendin kanssa.



Kuva 4. Backend pakkaus rakenne.

Frontend

Frontend lähettää HTTP-pyyntöjä RESTful-rajapinnan kautta, jotka käsitellään backendin controller-luokissa. Harmonian frontendissä, käytämme Spring Frameworkin tarjoamaa RestTemplate-kirjastoa, joka tarjoaa helpon tavan lähettää HTTP-pyyntöjä RESTful rajapinnoille käyttäen Spring Frameworkin tukemia HTTP-metodeja. RestTemplate voidaan injektoida haluttuun luokkaan Spring-kontekstin avulla, mikä mahdollistaa sen käytön kaikkialla sovelluksen eri luokissa.

Kun pyyntö on lähetetty, RestTemplate odottaa vastausta RESTful-rajapinnalta. Lopulta backendin controller-luokka palauttaa vastauksen frontendille, joka voi käsitellä sen, muuttaa sen haluamaansa muotoon (esim. JSON->text) ja päivittää sen näkyväksi käyttöliittymään.

“Client”-luokat hyödyntävät FasterJackson kirjaston muuntajaa. FasterJackson muuntaa sille annetut objektit RestTemplaten ymmärtämään JSON formaattiin, ja viestien palautuessa Backend:iltä se muuntaa vastauksessa saadun JSON objektin takaisin ohjelman määrittämäksi olioksi.

Client luokkien lähettämät pyynnot ohjataan hyödyntämällä HarmoniaConstant luokassa löytyvillä URL-muuttujilla, sekä sinne tallennetaan kirjautuneen käyttäjän tiedot sessiota varten.

Client luokkien metodeja kutsutaan HarmoniaDataLoader luokan staattisilla metodeilla.

Tiedostojen helppolukuisuuden parantamiseksi, sekä ohjelman monikielisyyttä varten on luotu myös HarmoniaMessagesConstants luokka, johon tallennetaan käyttäjälle näytettäviä virheviestejä ja ilmoituksia.

Näkymien välillä liikkumisen tueksi loimme myös HarmoniaViewConstants johon on tallennettu tarvittavien FXML tiedostojen sijainnit, jottei niitä tarvitsisi Controller-luokissa toistaa.

Controller-luokat hoitavat näkymillä löytyvien käyttöliittymien toiminnallisuuden. Ne asettavat tarvittavien muuttujien arvot, piirtävät ruudulle listojen sisällön, sekä kuuntelevat käyttäjän pyyntöjä ja toimii niiden mukaan.

HarmoniaData luokkaan tallennetaan ohjelman toiminnalle tärkeitä muuttujia. Näihin kuuluu listat käyttäjistä, yksityisviesteistä, sekä tarvittaessa käyttäjän valitsema vastaanottaja, valittu yksityisviesti ja viestien lukumäärä.

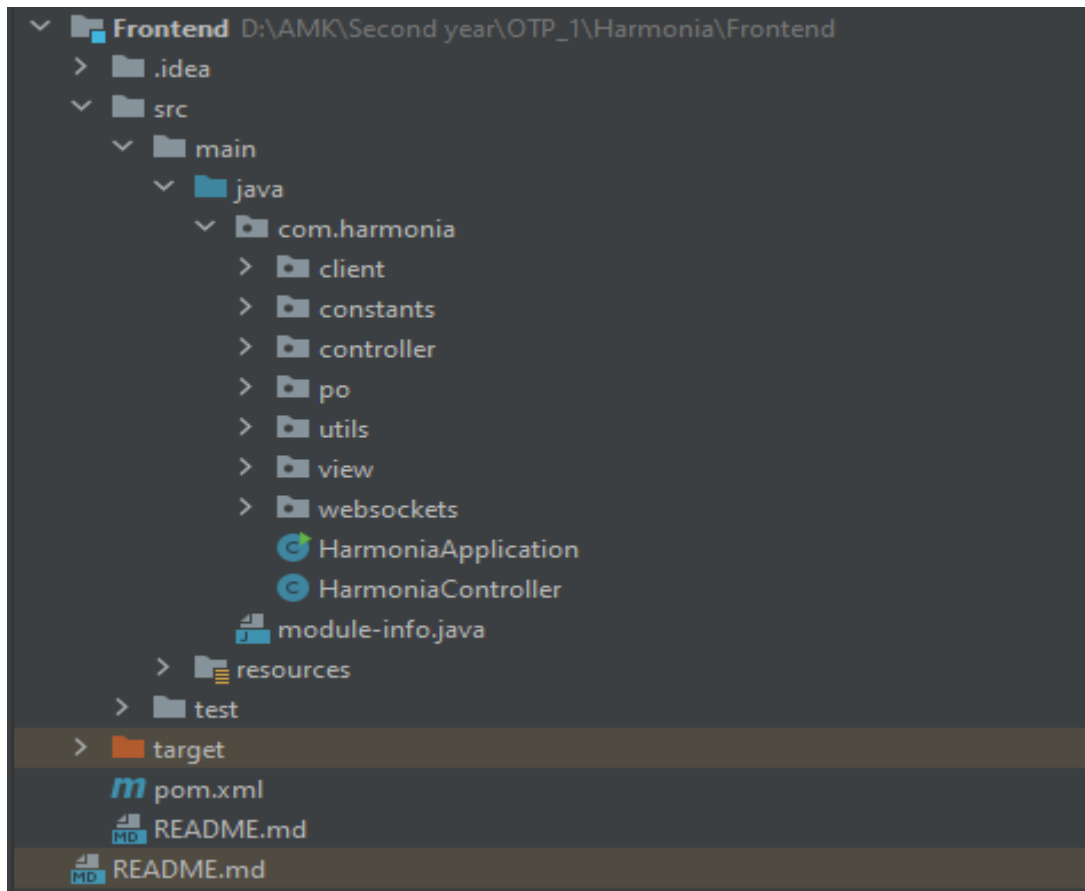
Spring frameworkin RestTemplate-kirjaston käyttö tekee sovelluksemme kehittämisestä helppoa ja nopeaa.

Frontend:iin kuuluu graafinen käyttöliittymä, joka tarjoaa käyttäjälle toiminnot kirjautumiselle, Yhteisöihin liittymiselle, sekä viestien lähettämiselle (Public Message ja Direct Message).

Käyttöliittymä on toteutettu JavaFX:llä ja controller-luokilla hyödyntäen MVC mallia.

HarmoniaUtils sisältää yleisessä käytössä olevien muuttujien generoivia metodeja, tätä luokkaa käytetään rakentamaan HttpHeaders muuttuja client-luokille, näyttämään error ja vahvistusviestejä käyttäjälle, siirtämään käyttäjä näkymästä toiseen, näiden toimintojen lisäksi luokalla on myös mahdollista simuloida käyttäjätietoja debug:austa varten.

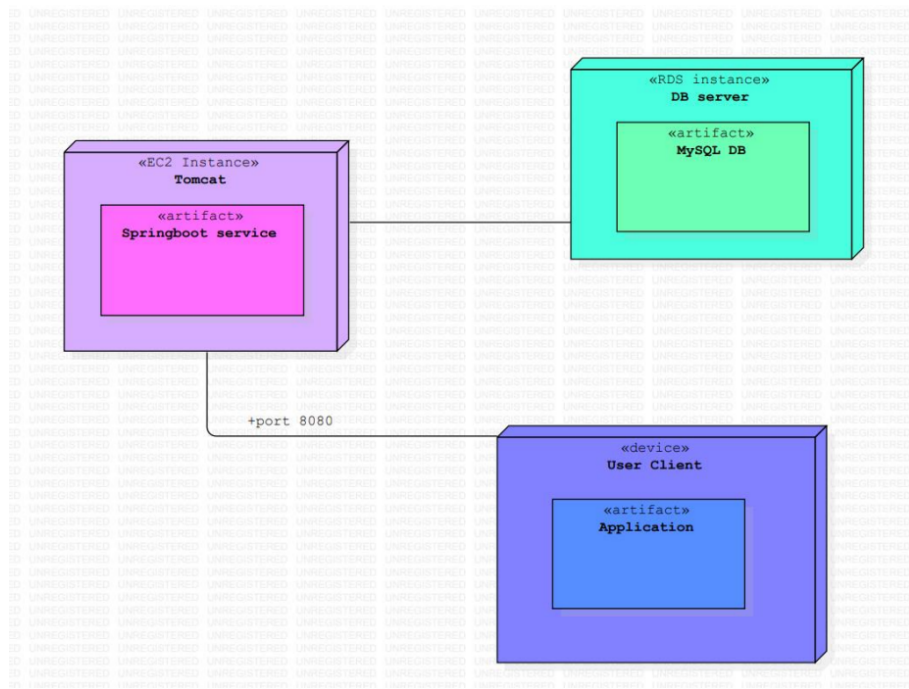
po luokat, eli "plain old java object"-luokat sisältävät muuttujia, jotka vastaavat tietokantaan tallennettua dataa. Näitä luokkia käytetään client luokissa pyyntöjä lähettäessä, sekä näkymissä käyttäjä- ja viestitietojen listaamisessa.



Kuva 5. Frontend pakkaus rakenne.

Arkkitehtuuri

Harmonian sijoittelukaavio (kuva 5.) koostuu kolmesta nodesta, eli solmusta, jotka toimivat yhdessä Spring Boot-palvelun tukemiseksi. Sovelluslogiikan, tietojen tallennuksen ja käyttöliittymän erottaminen eri solmuihin mahdollistaa helpon skaalautuvuuden ja ylläpidon palvelulle.



Kuva 5: Harmonian sovelluksen sijoittelukaavio.

- Ensimmäinen solmu on AWS:n (Amazon Web Services) ylläpitämä EC2-instanssi, joka ajaa Tomcat-palvelinta ja isännöi Spring Boot-palvelun artefaktia. Tämä node toimii sovelluksen keskeisenä yhteydenottopisteenä, käsitellen kaikki saapuvat pyynnöt ja tarjoillen tarvittavat vastaukset.
- Toinen solmu on AWS:n (Amazon Web Services) ylläpitämä RDS-instanssi, joka isännöi MySQL-tietokanta-artefaktia, jota Spring Boot-palvelu käyttää tietojen tallentamiseen ja noutamiseen. RDS-instanssi kommunikoi EC2-instanssin kanssa internetin välityksellä, tarjoten turvallisen ja skaalautuvan ratkaisun tietojen tallentamiseen ja noutamiseen.
- Kolmas solmu on käyttäjän asiakassovellus, eli Harmonia, joka edustaa loppukäyttäjän laitetta, joka käyttää Spring Boot-palvelua JavaFX:llä rakennetun sovelluksen kautta. Laite isännöi sovellus artefaktia, joka mahdollistaa käyttäjän suorittamaan erilaisia toimintoja sovelluksen sisällä. Tämä deployment-diagrammi kuvaa hajautettua järjestelmäarkkitehtuuria, joka tarjoaa mitattavuutta, turvallisuutta ja korkeaa käytettävyyttä Springboot-palvelulle.

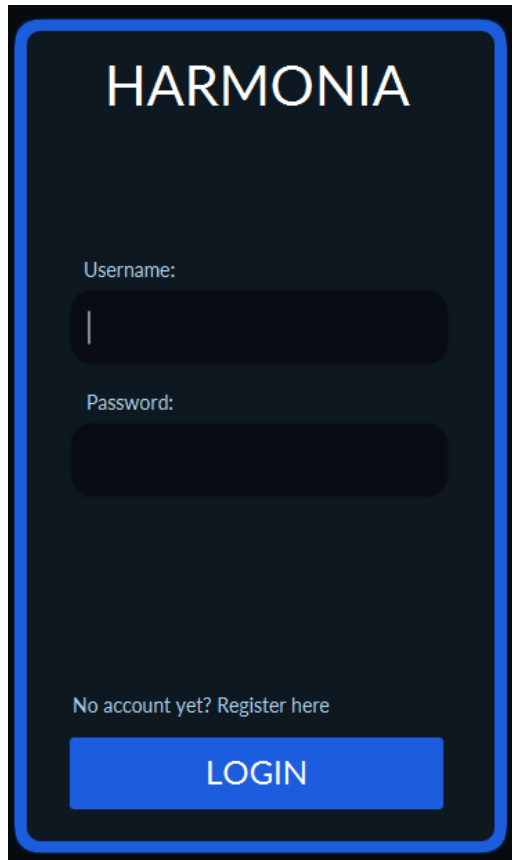
7. Testaus

Ohjelmistoa testataan automoidusti hyödyntäen jenkins palvelinta, jonka lisäksi ohjelmisto sisältää samat testit, joita voidaan ajaa manuaalisesti omalla koneella. Testit on kirjoitettu JUnit 5 kirjastolla ja ne hyödyntävät Mockito kirjastoa simuloimaan tietokannasta saatavaa dataa. Jenkins palvelin on asetettu ajamaan testit uudelleen kun muutoksia tehdään arkiston “main” -oksaan.

All Tests

Class	Duration	Fail	(diff)	Skip	(diff)	Pass	(diff)	Total	(diff)
ChannelClientTest	97 ms	0		0		4		4	
ContentfulClientTest	2.2 sec	0		0		2		2	
DMClientTest	55 ms	0		0		3		3	
EditUserClientTest	0.11 sec	0		0		2		2	
FriendClientTest	50 ms	0		0		3		3	
PublicMessageClientTest	3.7 sec	0		0		4		4	
ServerClientTest	69 ms	0		0		3		3	
ServerMemberClientTest	17 ms	0		0		1		1	
UserClientTest	90 ms	0		0		4		4	

Coverage: java in Frontend			
Element	Class, %	Method, %	Line, %
com	38% (17/44)	24% (59/241)	16% (165/985)
harmonia	38% (17/44)	24% (59/241)	16% (165/985)
client	100% (8/8)	69% (29/42)	65% (127/193)
constants	20% (1/5)	25% (1/4)	25% (6/24)
controller	0% (0/13)	0% (0/69)	0% (0/498)
po	77% (7/9)	30% (27/88)	28% (27/95)
utils	33% (1/3)	11% (2/18)	6% (5/75)
view	0% (0/2)	0% (0/4)	0% (0/9)
websockets	0% (0/2)	0% (0/4)	0% (0/13)
HarmoniaApplication	0% (0/1)	0% (0/3)	0% (0/16)
HarmoniaController	0% (0/1)	0% (0/9)	0% (0/62)

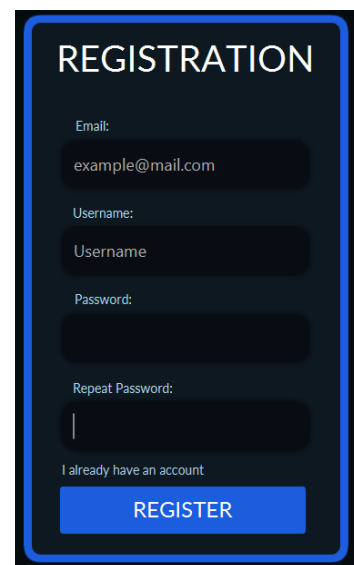


8. Ohjelmiston toiminta

1. Ohjelman käynnistyessä käyttäjälle näytetään kuvan (kuva 6) mukainen lomake. Lomakkeen tiedot täyttämällä käyttäjä pääsee kirjautumaan sisään. Mikäli käyttäjällä ei vielä ole tiliä, pääsee sen luomaan “No account yet? Register here” hyperlinkkiä painamalla.

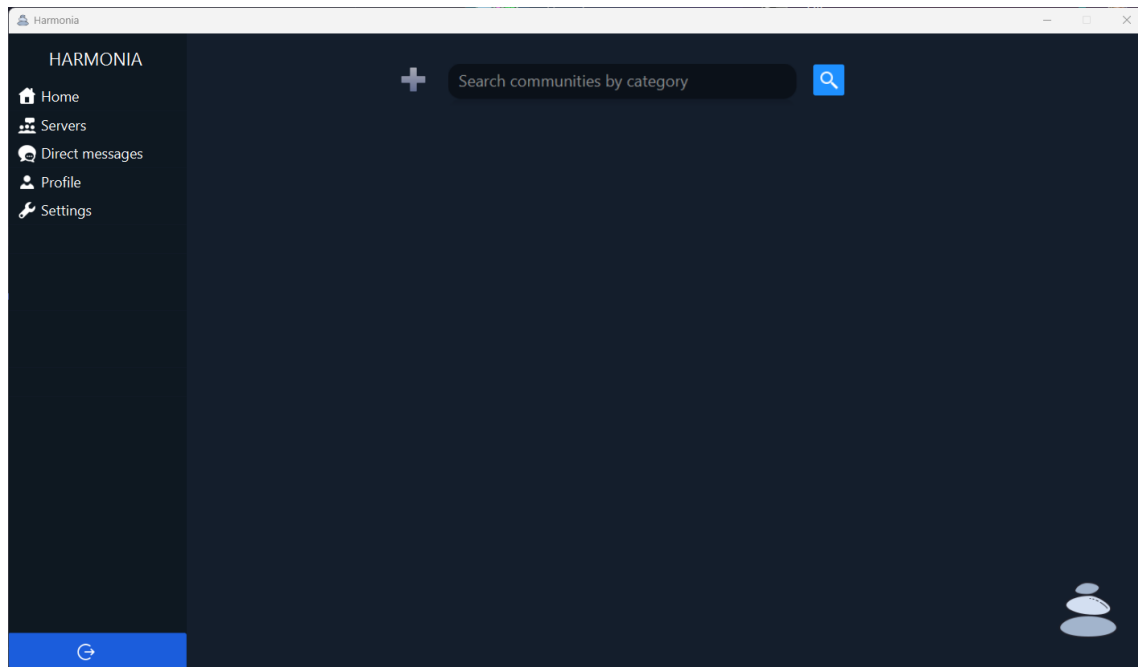
Kuva 6 : Harmonia sovelluksen kirjautumislomake

2. Rekisteröintilomakkeen (kuva 7) avulla käyttäjä voi luoda tilin. Lomakkeen “Email” kenttään syötetään haluama sähköpostiosoite. “Username” kenttään syötetään haluama käyttäjänimi, käyttäjänimi joka kenttään syötetään on käyttäjä kirjautumisessa käyttämä nimi. Nimi näkyy kaikille kunnes käyttäjä vaihtaa lempinimeään eli niin sanottua “display name”:a. “Password” kenttään syötetään haluttu salasana. Salasan syötön jälkeen käyttäjää pyydetään vielä toistamaan salasana varmistaakseen, että käyttäjä syötti haluamansa salasan.

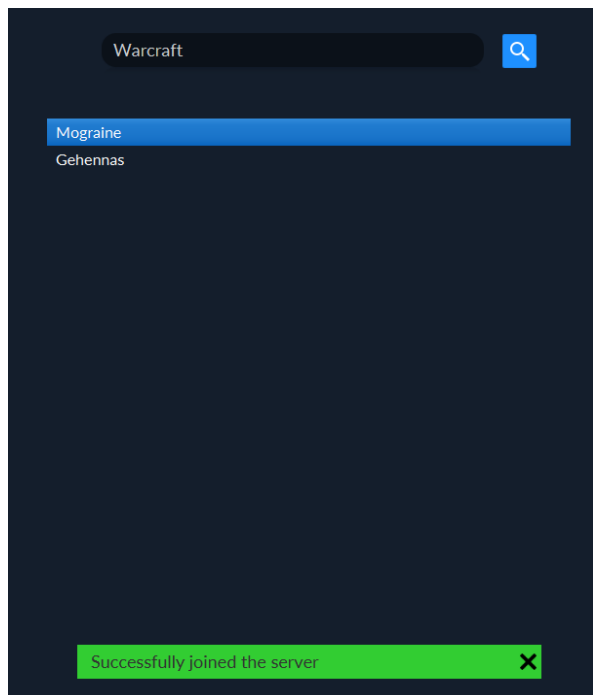


Kuva 7: Rekisteröintilomake

3. Kun käyttäjä on kirjautunut sisään, käyttäjälle aukeaa sovelluksen kotisivu (kuva 8) jonka avulla käyttäjän on mahdollista etsiä haluamiaan yhteisöjä eli servereitä näkymässä olevaan hakukenttään syöttämällä hakusanoja, kuten “gaming” “kalastus” tai esimerkiksi “jalkapallo”, jolloin ohjelma hakee yhteisöjen joukosta hakuun sopivat vaihtoehdot. Hakukentän vasemmalta puolelta on myös mahdollista luoda oma yhteisö.



Kuva 8: Harmonia sovelluksen “home” näkymä.



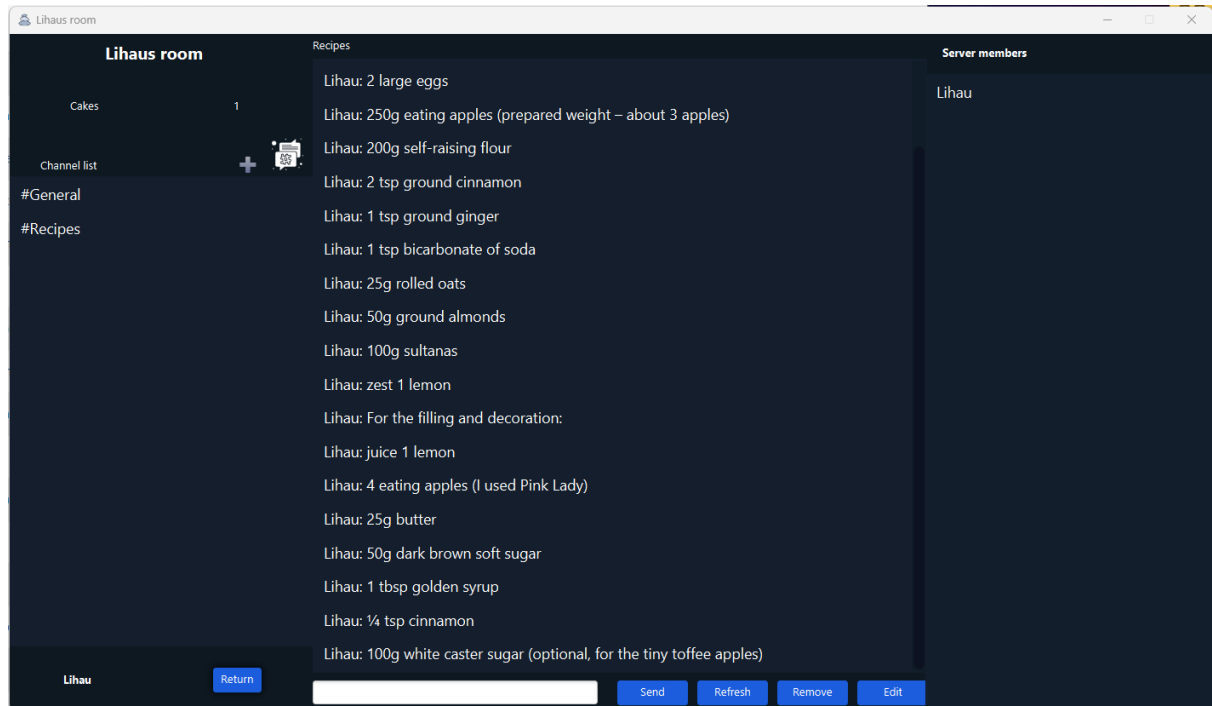
4. Hakupalkkiin syötetty haku “Warcraft” haki kaksi yhteisöä (Kuva 9) . Käyttäjä voi liittyä palvelimelle klikkaamalla palvelimen nimeä. Mikäli palvelimelle liittyminen onnistuu, käyttäjä saa ruudun alareunaan ilmoituksen onnistuneesta liittymisestä.

Kuva 9: Harmonia sovelluksen hakutoiminto

5. “Servers” näkymässä käyttäjä pääsee tarkastelemaan kaikkia yhteisöjä joihin on liittynyt.

Tuplaklikkaamalla yhteisön nimeä pääsee yhteisön palvelimelle.

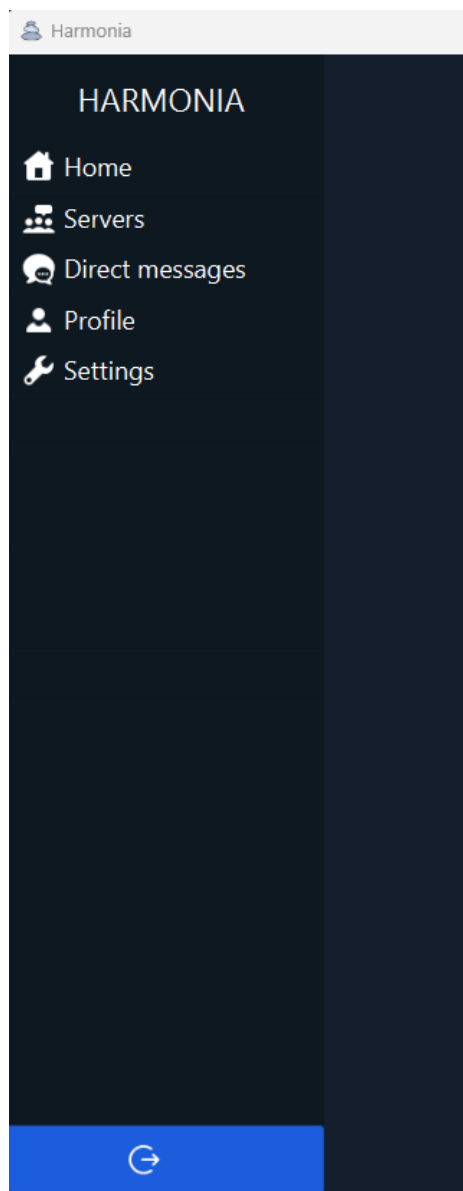
6. Yhteisön palvelin-näkymässä vasemmalla on tekstikanavat joita klikkaamalla pääsee keskustelemaan kanavan viestiketjuun. Palvelin-näkymän oikeassa reunassa on palvelimen jäsenlista. Kun tekstikanava on valittu palvelin-näkymän keskelle aukeaa ketju viesteistä sekä tekstikenttä johon voi kirjoittaa haluamiaan viestejä. (Kuva 10)



Kuva 10: Harmonia sovelluksen yhteisön palvelin-näkymä.

7. Profiili-näkymässä käyttäjä pääsee tarkastelemaan omia tietojaan sekä muokkaamaan omaa "About me" kenttää eli lyhyttä esittelyä itsestään.

8. "Settings" näkymässä käyttäjä voi vaihtaa profiilikuvaansa tai vaihtaa sovelluksen kieltä. Profiilikuvan vaihtamiseen tarvitaan halutun profiilikuvan url-osoite. "URL" kenttään tulee korvata valmiina oleva url osoite uudella halutun kuvan url osoitteella ja painettava save näppäintä tallentaaksesi muutokset.



9. Harmonia sovelluksen jokaisessa näkymässä on navigointipalkki (Kuva 11)

jonka avulla käyttäjän on mahdollista siirtyä eri näkymien välillä.

Painikkeet

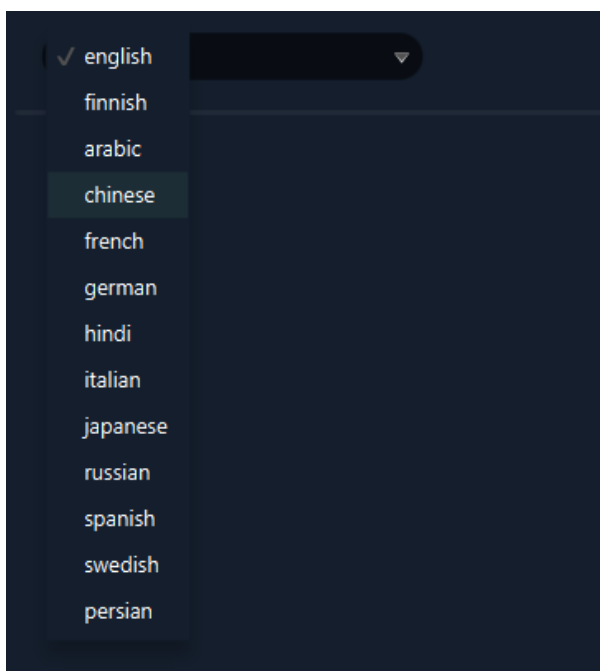
- “Home” painiketta painamalla käyttäjä pääsee sovelluksen alkunäkymään jossa voi hakea ja tarkastella yhteisöjä koti-näkymässä
- “Servers” painiketta painamalla pääsee tarkastelemaan kaikkia yhteisöjä joihin käyttäjä on liittynyt server-näkymässä.
- “Direct messages” painiketta painamalla pääsee yksityisviesti-näkymään
- “Profile” painikkeesta pääsee tarkastelemaan omaa profiiliaan profiili-näkymässä.
- “Settings” painikkeesta pääsee tilin asetus-näkymään
- Ympyrä+nuoli painikkeesta käyttäjä kirjaa itsensä ulos

Kuva 11: Harmonia sovelluksen navigointipalkki.

9. Lokalisaatio

Harmonia-sovelluksessa lokalisaatio on toteutettu käyttämällä Mapia ja Contentfulia. Mapia käytetään tallentamaan kieliparametreja ja niiden vastaavia Contentful-avaimia ja -tokeneita, ja lataamaan Contentful-merkinnät valitulla kielellä paikallistettujen sisältöjen noutamiseksi. Kielivalinta tapahtuu käyttäjän valitessa halutun kielen, ja vastaavat kieliparametrit ladataan resurssipaketista. Kielivalinta pudotusvalikko alustetaan käytettävissä olevilla kielillä, ja aikaisemmin valittu kieli valitaan automaattisesti. Kun käyttäjä vaihtaa kieltä, pudotusvalikko sitoutuu kielen muutoksen toimintoon asettamaan valitun kielen ja päivittämään sivun.

Suurin osa lokalisaatiosta on toteutettu käyttäen language.property -tiedostoja. Contentfulia käytetään vaihtoehtona perinteiselle tietokannan lokalisaatiolle, koska käyttäjien ja heidän viestiensä lokalisoiminen perinteisellä tavalla ei ole mahdollista.



Kuva 12: Tällä hetkellä Harmonia-ohjelmisto on saatavilla 13 eri kielelle.

10. Kehitysprosessi ja kehitysvaiheen tekniikat

Hyödynsimme koko projektin ajan SCRUM kehitysmenetelmää ja pidimme kirjaa valmiista, ja vielä toteutettavista toiminnoista Trello ja Nektion -verkkosivuilla.

Järjestimme päivittäin palaverin tiimimme kesken retrospektiivisiä kokouksia, jossa pidettiin katsaus edellisen päivän tapahtumiin, suunniteltiin kyseisen päivän tehtäviä jokaiselle, ja keskusteltiin mahdollisista esteistä, joiden vuoksi edellisen päivän

suunnitellut työt eivät toteutuneet. Käytimme kokouksissa kirjanpidon apuna Fig Jam-taulua valmistuneista tai kesken jääneistä töistä.

Aloitimme projektin suunnittelemalla tietokantarakenteemme, jonka jälkeen loimme Spring Boot-pohjaisen sovelluksen backend, johon tietokantamme viimein yhdistettiin.

Versionhallintatyökaluna käytimme Git-versionhallintaa ja arkistoimme koodimme githubin palveluun.

Miksi SCRUM?

Projektin aikana pyrimme seuraamaan SCRUM-kehitysmenetelmää, jotta ohjelmamme kehitys olisi nopeaa, sujuvaa ja tehokasta. Käyttämällä SCRUM-menetelmää, pystyimme välttämään konflikteja ja parantamaan ohjelmamme toimivuutta monella eri tavalla. Testaaminen oli myös keskeisenä osana projektiamme, sillä se auttoi meitä vahvistamaan, että ohjelmisto toimi odotetulla tavalla ja vastasi käyttäjätarinoiden asiakkaiden tarpeita. Testaukseen käytimme Junit testaus kirjastoa ja mockito kirjastoa simuloimaan tietokannan palauttamaa dataa.

Yksi tärkeimmistä syistä, oli kommunikaation edistäminen ja yhteistyö tiimimme välillä. Pystyimme keskittymään yhdessä tärkeisiin asioihin, kuten asettamiimme tavoitteisiin, prioriteetteihin sekä työnjakoon. Pystyimme myös välttämään tiimin välille syntyviä konflikteja ja epäselvyyksiä.

SCRUM -tekniikan ansioista pystyimme myös tunnistamaan, ennaltaehkäisemään ja käsittelemään mahdollisia ongelmia ja riskejä varhaisessa vaiheessa, mitkä olisivat voineet johtaa konflikteihin myöhemmässä kehitysvaiheessa. Lisäksi tekniikka kannusti meitä jatkuvaan integraatioon, jolloin sekä uudet ominaisuudet ja korjaukset lisättiin ohjelmaan säännöllisesti pieninä paloina. Tämä auttoi välttämään suuria ongelmia, jotka olisivat ilmenneet, kun suuri määrä koodia olisi yhdistetty kerrallaan.

11. Jatkokehitysideat

Tarkoituksena on jatkaa Harmonian kehittämistä sekä laajentaa kehitystiimiä. Aiomme ensimmäisenä vaihtaa Harmonian käyttöliittymän React-pohjaiseksi ja näin ollen tehdä siitä modernimman. Tämän myötä meillä on mahdollisuus lisätä teknologiaa, joka ei ole saatavilla JavaFX:lle. Lisäksi aiomme lisätä puhelu- sekä videopuhelu-ominaisuuden käyttäjille, sekä mahdollisuuden jakaa tiedostoja.

Tulevaisuudessa Harmoniasta tulee olemaan saatavilla kaksi erilaista versiota, joista toinen on suunnattu yrityskäyttöön ja toinen vapaa-ajan yhteisöpalveluksi.

12. Yhteenveto

Tämä dokumentti kuvaa viestintäsovellus Harmonian vaatimuksia, datamallia, arkkitehtuuria ja toimintaa, sekä sovelluksen kehitysprosessia. Harmonia on viestintäsovellus, joka tarjoaa käyttäjille yksityisemmän ja yksinkertaisemman käyttöliittymän kuin muut suositut viestintäsovellukset. Sovelluksessa on sisäänrakennettuja lisäosia, jotka korvaavat bot-ohjelmat, joita käytetään muissa viestintäsovelluksissa. Harmonia on myös sosiaalisempi ja käyttäjien mieltymyksiin paremmin sopeutuva, sillä se ehdottaa palvelimia/yhteisöjä heidän kiinnostuksenkohteidensa, kuten lempi pelien tai muiden vastaavien mielenkiintojen perusteella. Tarve Harmonia-sovellukselle syntyi käyttäjien tarpeesta saada yksityisempi ja yksinkertaisempi viestintäsovellus. Sovelluksen sisäänrakennetut lisäosat korvaavat vaikeasti käytettävät bot-ohjelmat ja mahdollistavat yhteisöjen löytämisen kiinnostusten perusteella. Sovelluksen tärkein tavoite on tarjota käyttäjille helppokäyttöinen ja turvallinen viestintäympäristö.

Harmonian ohjelmisto on jaettu kahteen moduuliin: Backendiin ja Frontendiin. Moduulien jakaminen on hyödyllistä, koska se mahdollistaa kunkin moduulin kehittämisen ja testaamisen erikseen. Tämä tarkoittaa sitä, että jos muutoksia tarvitaan yhteen moduuliin, vaikutus muihin moduuleihin on todennäköisesti vähäinen, mikä vähentää virheiden mahdollisuutta ja helpottaa kehitystä. Jos esimerkiksi Harmonian Frontend siirrettäisiin tulevaisuudessa JavaFX:stä Reactiin, kun JavaFX:n tuki päättyy (nykyisellään suunniteltu vuodelle 2025), modulaarinen lähestymistapa helpottaisi tätä siirtymää. Koska moduulit ovat erillisiä, Reactin esimerkiksi käyttöönotto Frontendiin voidaan tehdä vaikuttamatta Backendiin, mikä mahdollistaa joustavuuden ja vähentää mahdollisia virheriskejä. Lisäksi tämä modulaarinen lähestymistapa mahdollistaa kehittäjätiimimme keskittymisen yhteen moduuliin kerrallaan, mikä helpottaa tiimityötä. Backend käyttää Spring Frameworkia ja Javan ohjelmointikieltä käyttäjähallinnan, todennuksen ja valtuutuksen RESTful-liittymien rakentamiseen. Backend myös vuorovaikuttaa tietokannan kanssa Hibernate-kirjaston kautta, ja tietokantaa hallitaan PostgreSQL:n avulla. Frontend käyttää JavaFX:ää graafisen käyttöliittymän rakentamiseen ja kommunikoi backendin kanssa RESTful-rajapintojen kautta.

Sovelluksen kehitysprosessi sisältää suunnitteluvaiheen, toteutusvaiheen ja testausvaiheen, minkä jälkeen seuraa jatkuva käyttöönotto ja ylläpito. Suunnitteluvaiheeseen kuuluu vaatimusten määrittely ja tietomallin luominen, joka kuvaa tietojen entiteetit ja niiden väliset suhteet. Toteutusvaiheessa ohjelmisto rakennetaan suunnittelun pohjalta käyttäen ketteriä ohjelmistokehitysmenetelmiä. Testausvaiheessa tehdään yksikkötestaus,

integraatiotestaus ja järjestelmätestaus varmistaa, että ohjelmisto toimii oikein. Jatkuva käyttöönotto tarkoittaa käyttöönoton automatisointia nopeuttaakseen ja tehdäkseen sen luotettavammaksi. Ylläpito sisältää vikojen korjaamisen ja parannusten tekemisen ohjelmistoon ajan mittaan. Harmonian kehitystiimi käyttää myös versionhallintaa ohjelmiston lähdekoodin hallintaan ja tehokkaampaan yhteistyöhön. Lopuksi tiimi käyttää ketteriä projektinhallintamenetelmiä varmistaa, että ohjelmisto toimitetaan ajoissa ja vastaa käyttäjien tarpeita.