Sosiaali- ja terveyspalveluiden chat

Sosiaali- ja terveyspalveluiden chat on web-sovellus terveydenhuollon ammattilaisten ja asiakkaiden väliseen keskusteluun. Ohjelman avulla on tarkoitus osittain korvata esimerkiksi puhelimessa tapahtuvaa asiakaspalvelua ja neuvontaa.

Ohjelma tarjoaa chat-yhteyden, jossa toisena osapuolena on terveydenhuollon ammattilainen. Ammattilainen ottaa vastaan asiakkaiden chat-pyyntöjä, jotka on asetettu jonoon. Kun ammattilainen on ottanut asiakkaan jonosta, kummallekin osapuolelle avautuu selaimessa chat-ikkuna, jossa he voivat keskustella kahden kesken. Asiakas voi liittyä jonoon syöttämällä aloitussivun kenttiin nimimerkin ja aloitusviestin, kirjautumista ei vaadita. Terveydenhuollon ammattilaiselta vaaditaan järjestelmään kirjautuminen tunnuksilla.

Ammattilaisen näkymässä voi vastaanottaa keskusteluja jonosta ja keskustella useamman asiakkaan kanssa samanaikaisesti. Keskustelut tallennetaan tietokantaan ja ammattilaisen käyttöliittymään kuuluu myös mahdollisuus selata vanhoja keskusteluja historianäkymässä. Ammattilaisen näkymään kuuluu paikalla/pois-nappi, jolla hän voi viestittää käytettävyydestään. Jos yhtään ammattilaista ei ole “paikalla”-tilassa, suljetaan chat uusilta asiakkailta. Avatut keskustelut ja jonossa entuudestaan olevat asiakkaat voi kuitenkin käsitellä normaalisti.

Ammattilaisen ja asiakkaan näkymän lisäksi sovellukseen kuuluu ylläpitäjän näkymä, jossa järjestelmään voidaan lisätä uusia tunnuksia ammattilaisille sekä poistaa niitä tarvittaessa. Ylläpitäjällä on myös mahdollisuus tyhjentää palvelin viesteistä yhdellä painikkeella sen takia, että varsinkin kehitysvaiheessa palvelimelle voi tallentua paljon ylimääräisiä viestejä, joita ei haluta näyttää ohjelmaa esiteltäessä.

Käyttäjäryhmät

Sovelluksen käyttäjät voi jakaa kolmeen ryhmään: asiakkaisiin, ammattilaisiin ja ylläpitäjiin.

Asiakkaat: Järvenpään sosiaali- ja terveyspalveluiden asiakkaat, jotka haluavat keskustella esimerkiksi päihdeongelmasta tai lohjenneesta hampaasta.

Ammattilaiset: asiakkaita chatissa palvelevat terveydenhuollon ammattilaiset.

Ylläpitäjät: rekisteröityneitä käyttäjiä, joilla on mahdollisuus lisätä tai poistaa ammattilaisia sekä suorittaa muita ylläpidollisia tehtäviä.

Projekti netissä

Projektin lähdekoodi ja dokumentaatio löytyy osoitteesta https://github.com/PauliNiva/Sotechat

Esittelyversio pyörii Herokussa ja sitä pääsee kokeilemaan eri käyttäjien rooleissa klikkaamalla seuraavia linkkejä:

Hammashoidon asiakkaan linkki: http://sotechat.herokuapp.com/from/?source=hammashoito

Terveydenhuollon asiakkaan linkki:
http://sotechat.herokuapp.com/from/?source=terveydenhuolto

Ammattilaisen linkki: http://sotechat.herokuapp.com/pro/

Tunnukset hoitajalle:
Käyttäjätunnus: hoitaja
Salasana: salasana

Tunnukset adminille:
Käyttäjätunnus: admin
Salasana: 0000

Arkkitehtuuri

Ohjelmiston voi ajatella jakautuvan selain- ja palvelinpuoleen. Kommunikaatio selaimen ja palvelimen välillä tapahtuu sekä HTTP- että WebSocket-liikenteellä.

Tietoliikenne

Kun käyttäjä menee ensi kertaa selaimellaan sivulle, palvelin lähettää HTML-sivut, selainpuolen toiminnallisuuden luovat JavaScript-tiedostot ja muut kirjastot. Kaikki staattiset resurssit ovat julkisia. Muiden pyyntöjen osalta oikeuksien tarkistus ja pyynnön validointi tapahtuu palvelinpuolella.

Yksinkertaiset selaimen pyynnöt on toteutettu HTTP GET- POST- ja DELETE -pyyntöinä, kun taas monimutkaisempi viestintä - kuten aukinaisessa chatissä molempiin suuntiin liikkuvat viestit - on toteutettu STOMP over WebSocket-protokollalla. Useimmiten pyynnön sisältö on

JSON-muodossa, mutta joissain tapauksissa kaikki tarvittava tieto on pyynnön polussa. Lisää tietoa protokollasta löytyy Protokollakuvaus-dokumentista.

Kuva Protokollakuvaus-dokumentista. Helpomminkin tarkasteltava versio löytyy projektin dokumenteista, mutta sen avaaminen vaatii taulukkolaskentaohjelmaa.

Polku	Tyyppi	Oikeudet	Tarkoitus	Pyynnön sisältö (Clientiltä serverille)	Vastauksen sisältö (Serveriltä clientille)	Lisätietoa
/toServer	STOMP over WebSocket	All	Websocket-yhteyden muodostus	STOMP-protokollan mukainen CONNECT-kehys. Kehyksessä tieto WebSocket-yhteyden versiosta, heart-beat eli kuinka usein client testaa yhteyden palvelimelle, sekä WebSocket-istunnon id.	CONNECT-kehys, jonka sisällä tieto siitä, onnistuiko yhteyden muodostaminen vai ei.	
/getusers	GET	ADMIN	Kaikkien käyttäjien listaus	Normaali GET-pyntö	JSON-lista muotoa [{"userId":"admin","username":"pauli", "loginName":"admin","conversationsOfPerson":[]}, {"userId":"666","username":"Hoitaja", "loginName":"hoitaja","conversationsOfPerson":[]}]	
/newuser	POST	ADMIN	Uuden käyttäjän luonti	Parametrit Base64 enkoodattuna HTTP POST RequestBodyssa, eli upotettuna URLiin	JSON muotoa {"status":"OK"} tai {"error":"reason"}	Parametrien upotus URLiin tyyliin http://example.com/page?parameter=value &also=another - tässä tapauksessa
/resetpassword/{id}	POST	ADMIN	Annetun käyttäjän salasanan vaihto	Käyttäjän userId URL-osoitteen prässä, uusi salasana base64-koodattuna	JSON muotoa {"status":"OK"} tai {"error":"reason"}	Paluuarvo kertoo, onko salasanan vaihto onnistunut vai ei.
/delete/{id}	DELETE	ADMIN	Annetun käyttäjän poisto	Poistettavan käyttäjän userId URL-osoitteen perässä.	JSON muotoa {"status":"OK"} tai {"error":"reason"}	Paluuarvo kertoo, onko käyttäjän poisto onnistunut.
/tuhoaHistoria	POST	ADMIN	Palvelimen tilan nollaaminen demoamis-käyttöä varten	Normaali POST-pyntö	JSON muotoa {"status":"OK"} tai {"error":"reason"}	Paluuarvo kertoo, onko palvelimella olevien keskustelujen poistaminen onnistun
/from/	GET	All	Asiakkaan ohjaaminen oikean kategoriaan.	HTTP-pyynnön perässä olevan source-kentän arvo. Arvo tulee sen perusteell mistä osoitteesta Sosiaali- ja terveyspalveluiden chattiin on tultu.	Uudelleenohjauspyntö osoitteeseen / eli juureen	
/toServer/chat/{channelId}	STOMP over WebSocket	Sessio*	Validoi, muokkaa ja reitittää chattiin kirjoitettuja viesteja	STOMP-protokollan mukainen SEND-kehys, jonka sisältönä JSON, joka on muotoa: {"userId":"123","channelId":"abcd123","content":"Heil"} ja Principal-oli jos viestin lähettäjä on kirjautunut käyttäjä. Lisäksi mukana kulkee tieto WebSocket-osoitteesta, joka on muotoa /toServer/chat/{channelId}. WebSock istunnon id, sekä heartbeat(kts. ensimmäinen rivi).	Vastaus tulee osoitteeseen /toClient/chat/{channelId}, kts. allaoleva rivi	
/toClient/chat/{channelId}	STOMP over WebSocket	Sessio*	Viestien reitti clientin suuntaan	Clientiltä serverille lähtevät viestejä koskevat pyynnöt tehdään websocket-osoitteeseen /toServer/chat/{channelId} kts. yllaoleva riv.	STOMP-protokollan SEND-kehyksen sisällä lähetetään JSON, joka on muotoa: {"messageId":"321","username":"Matti","channelId":"abcd123", "timeStamp":"2016-07-02T19:01:21.981+03:00","content":"Hei vaan!"}. Lisäksi lähetetään muut tiedot, jotka löytyvät yllä olevaita rivitä.	
/toServer/heartBeat	POST	All	HTTP-Session elossa pitamiseksi	Normaali POST-pyntö	JSON muotoa "{"heartbeat":"server-alive"}"	
/getLogs/{channelId}	GET	Ammattilainen*	Palauttaa halutun keskustelun viestit	ChannelId löytyy GET-pyynnön polusta	JSON-lista muotoa [{"messageId":"321","username":"Matti","channelId":"abcd123", "timeStamp":"2016-07-02T19:01:21.981+03:00","content":"Hei vaan!"}, {"messageId":"313","username":"Pertti","channelId":"b-bd112", "timeStamp":"2016-07-02T19:45:21.981+03:00","content":"Moi!"}]	
/listMyConversations/	GET	Ammattilainen*	Hakea listaus kaikista omista historiallisista keskusteluista	Normaali GET-pyntö	JSON-lista muotoa [{"channelId":"abcd123", "date":"2016-07-02T19:50:21.981+03:00","person":"Matti", "category":"Hammashoito"},{"channelId":"abcd123", "date":"2016-07-02T20:00:21.981+03:00","person":"Pertti", "category":"Hammashoito"}]	
/auth	GET	All	HTTPBasicAuthentication eli kirjautuminen ja sen tarkistus	Normaali GET-pyntö	Kirjautuneelle käyttäjälle vastaus JSON:ina, joka on muotoa {"details": {"remoteAddress":"0:0:0:0:0:0:0:1","sessionId":"abcdefg"},"authorities": [{"authority":"ROLE_USER"}],"authenticated":true,"principal":"Hoitaja", "credentials":null,"name":"Hoitaja"}	Kirjautunut käyttäjä saa tästä osoitteena paluuarvona Javan Principal-olion, joka on muunnettu JSON-muotoon.
/pro	GET	All	Näyttää ammattilaispuolen näkymä	Normaali GET-pyntö	proCP.html-tiedosto	
/	GET	All	Näyttää asiakaspuolen näkymä	Normaali GET-pyntö	index.html-tiedosto	
/login	GET	All	Uudelleen ohjaa /pro polkuun	Normaali GET-pyntö	Uudelleenohjauspyntö osoitteeseen /pro	
/logout	POST	All	Tuhoaa session ja kirjaa rekisteröityneen käyttäjän ulos	Normaali POST-pyntö	Uudelleenohjauspyntö osoitteeseen / eli juureen	
/userState	GET	All	Palauttaa sessioon liittyvät asiakkaan tilatiedot	Normaali GET-pyntö	JSON muotoa {"state":"start","username":"Pertti","userId":"ikjl112", "category":"mielenterveys","channelId":"asgfsdslk123"}	Paluuarvona saadurssa JSON:issa state-kentän arvoja voivat "start":in lisäksi olla "chat" ja "inqueue". Arvo riippuu siitä missä näkymässä asiakas on. Category-kentän arvo puolestaan riippuu mitä sivulta asiakas on tullut ja voisi olla myös esim. hammashoito.
/proState	GET	Ammattilainen	Palauttaa ammattilaissession liittyvät tilatiedot	Normaali GET-pyntö	JSON muotoa {"state":"start","username":"Hoitaja","userId":"asdf112", "QBCC":"QBCC","online":"true","channelIds":["asgfsdslk123", "jkl565"]}	JSON-kentä QBCC on WebSocket-osoite, jonka ammattilainen voi tilata. Tilaamalla ammattilainen saa kaikki jonon tilapäivitykset. ChannelIds kentässä on listattu kaikki kanavat, joka ammattilainen on tilannut.
/joinQueue	POST	All	Liittää käyttäjän jonoon halutulla nimimerkillä ja aloitusviestillä	HTTP POST -pyyntö, jonka sisältönä JSON muotoa: {"username":"Matti" ,"startMessage":"Jaiikoja kolottaa."}	Vastaus pyntöön on JSON muotoa: {"content":"<LIITTYMINEN ONNISTUI>"} tai muotoa: {"content":"<LIITTYMINEN EPÄONNISTUI>"}	
/toServer/queue/{channelId}	STOMP over WebSocket	Sessio*	Jonossa odottamiskanava. Asiakas kuuntelee ja odottaa jonosta nosto	Ammattilainen lähettää STOMP-protkollan mukaisen SEND-kehyksen, jonka mukana kulkee tyhjä JSON.	Vastaus lähetetään osoitteeseen /toClient/queue/{channelId}. Kts. allaoleva rivi.	
/toClient/queue/{channelId}	STOMP over WebSocket	Sessio*	Jonotuksen tilatietojen lähetys clientin suuntaan	String channelId, String assignee - kenelle kanava popattu	STOMP-protokollan SEND-kehyksessä lähetetään channelId:n mukaiselle kanavalle kaulueille henkilölle JSON muotoa {"channelAssignedTo": "Hoitaja Anne"}, missä kenttä channelAssignedTo saa arvokseen sen hoitajan käyttäjänimen, kuka on nostanut asiakkaan jonosta.	
/leave/{channelId}	POST	All	Vastaanottaa kanavalta poistumispyynnön	POST-pyyntö jossa channelId on yksilöity polussa	WebSocket-osoitteeseen /toClient/chat/{channelId} lähetetään STOMP-protokollan SEND-kehyksen sisällä JSON muotoa{"notice":"chat closed"}	
/setStatus/	POST	Ammattilainen	Vaihtaa ammattilaisen tilan halutuksi. online tai offline	Parametri HTTP POST RequestBodyssa: avain "online", arvo joko "true" tai "false"	WebSocket-osoitteeseen /toClient/chat/{channelId} lähetetään JSON muotoa {"join":"Hoitaja"}, missä "join"-kentän arvo on ammattilaisen käyttäjänimi.	
/toClient/QBCC	STOMP over WebSocket	Ammattilainen	Lähettää ammattilaisille jonon tilatietoja	STOMP-protokollan mukainen SUBSCRIBE-kehys. Kehyksessä on tieto WebSocket-osoitteesta, jorta tilataan viestejä, eli osoitteesta /toClient/QBCC. Lisäksi kehyksessä on heart-beat eli kuinka usein client testaa yhteyttä palvelimelle, WebSocket-istunnon id, sekä Principal-olio, jos kanavan tilaa kirjautunut käyttäjä.	SUBSCRIBE-kehyksessä lähetetään muiden tietojen lisäksi vastaus, joka on JSON muotoa {"jono": [{"channelId": "xyz", "category": "1", "username": "Ra"}]}	Vastaus JSON:issa lähetetään siis kaikille QBCC-kanavan tilanneille käyttäjille vastauksena jonon tilanne. Jonon alkioista ilmoitettavat tiedot ovat jonottajan käyttäjänimi, millä kategorialla jonottaja on sivulle tullut, sekä minkä kanavan jonottaja on sivulle tullessaan tilannut.
		Sessio* = Riippuen onko oikeus parametrina annettuun kanavaan				
		Ammattilainen* = Riippuen onko oikeus tiettyihin historioihin				

Palvelinpuoli

Controller-luokat reitittävät *saapuvat* pyynnöt ja *vastaukset* pyyntöihin. Itse pyyntöjen käsittelystä vastaavat Service- ja Data-luokat.

Keskeisiä controllereita:

- ChatController: chattiin saapuvien viestien reititys.

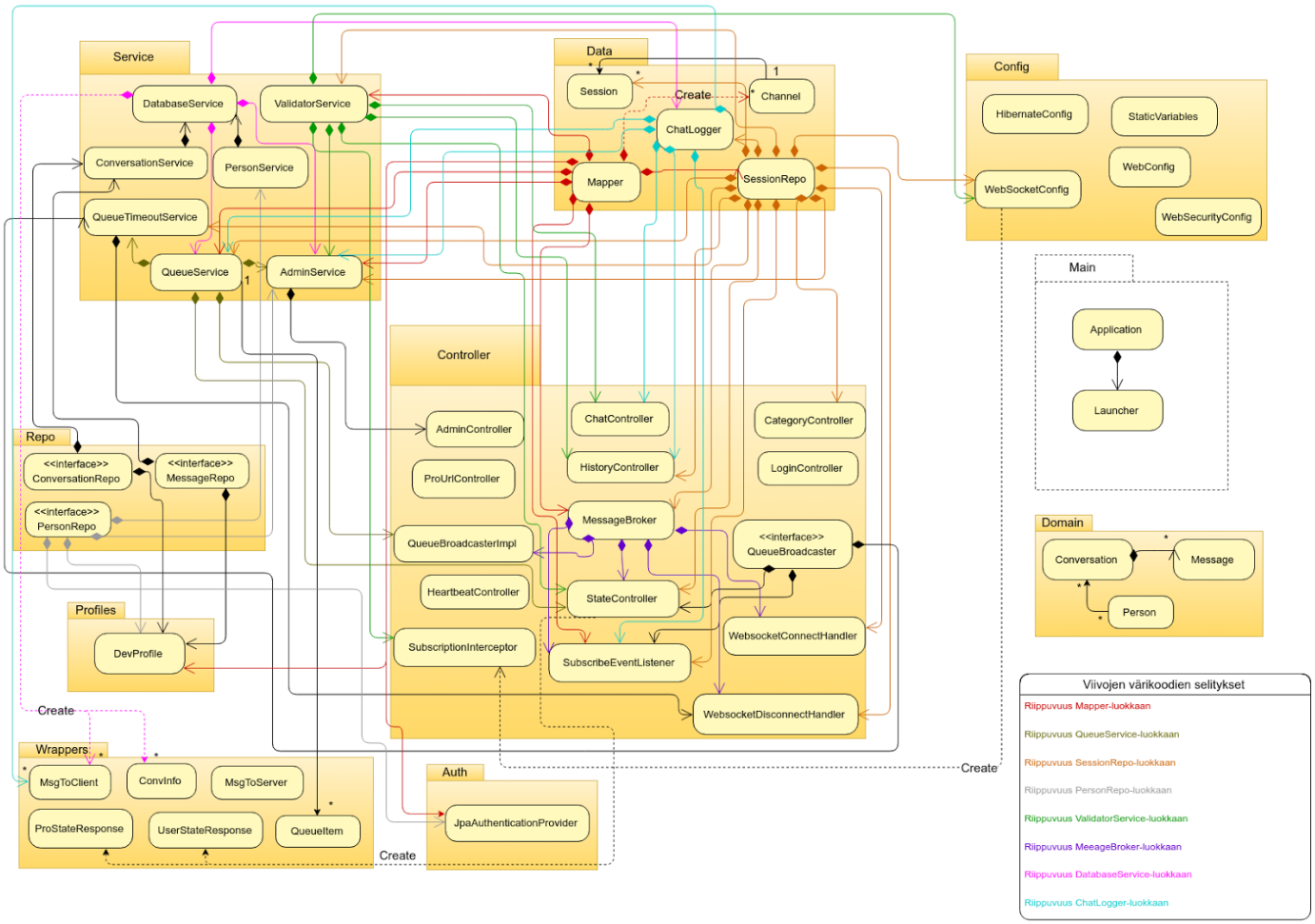
- StateController: tilaan liittyvien pyyntöjen reititys.
- AdminController: ylläpitäjän pyyntöjen reititys.
- MessageBroker: chat-viestien ja tila-viestien lähettäminen.

Muita keskeisiä luokkia:

- ValidatorService hyväksyy saapuvan pyynnön käsiteltäväksi tai hylkää sen.
- AdminService käsittelee ylläpitäjän pyynnöt (lisää ammattilainen, resetoi palvelin, yms.).
- QueueService: käsittelee jonoa, jossa asiakkaat odottavat pääsyä chattiin.
- SessionRepo pitää kirjaa Session-olioista, jotka kuvaavat tällä hetkellä aktiivisia käyttäjiä. Sovellus ei toteuta REST-API:a vaan palvelin muistaa sessioon liittyviä tietoja, jotta tietyt käyttötapaukset voidaan toteuttaa (esimerkiksi ammattilaisen sisäänkirjautuminen uudelleen tietokoneen kaaduttua ilman, että menetetään auki olevia keskusteluja). Kun käyttäjään menetetään yhteys, TimeoutService hoitaa Sessionin sulkemisen hetken kuluttua.
- Mapper pitää kirjaa Channel-olioista, jotka kuvaavat keskustelukanavia. Tätä käytetään esimerkiksi silloin, kun halutaan tietää onko kanava aktiivinen tai saako tietty henkilö nähdä kanavan viestejä.
- ChatLogger muistaa chattiin kirjoitetut viestit. Tuoreimmat viestit ovat muistissa nopeasti haettavassa muodossa, vanhempia joudutaan hakemaan DatabaseServicen avulla tietokannasta.
- JpaAuthenticationProvider toteuttaa rekisteröityneen käyttäjän kirjautumisen varmentamisen.

Esimerkki pyynnön elämänkaaresta palvelimen sisällä:

1. Palvelimelle saapuu HTTP POST -pyyntö “lähdin kanavalta”.
2. Pyyntö koskee tilan muutosta, joten StateControllerin metodi leaveChat käsittelee sen.
3. StateController vahvistaa ValidatorServiceltä, että pyyntö näyttää aidolta.
4. StateController pyytää SessionRepoa käsittelemään session poistamisen kanavalta (eli suorittamaan varsinaisen työn, joka liittyy pyyntöön).
5. StateController pyytää MessageBrokeria lähettämään tiedotteen kanavalle.



Luokkakaavio palvelinpuolen luokista. Kuva löytyy myös erillisenä tiedostona projektin dokumenteista.

Selainpuoli

Selainpuolen käyttöliittymä on toteutettu Single-page application -periaatteella AngularJS-kirjastolla. Käyttöliittymä on jaettu ammattilaisen ja asiakkaan puoliin. Palvelimelta kysytään käyttäjän tila ja käyttäjälle näytetään eri näkymä tilasta riippuen. Palvelin tunnistaa käyttäjän selaimeen tallennetun session cookien avulla. Jonoa ja keskustelua varten muodostetaan WebSocket-yhteys palvelimeen. Selaimen puolella yhteydestä huolehtivat Sock.js- ja STOMP.js-kirjastot. Käyttöliittymän eri näkymillä on omat HTML-tiedostot. Kontrollerien tehtävä on päivittää näkymiä, reagoida niiden tapahtumiin ja välittää tai kysyä tapahtumia palveluilta.

Asiakkaan saapuessa sivulle ladataan staattiset JavaScript-tiedostot palvelimelta. UserStateService pyytää tilatiedot palvelimelta. Jos chat on suljettu, asiakkaalle ladataan ja näytetään “suljettu”-näkymä. Jos chat on auki, näytetään asiakkaalle aloitusnäkymä, jossa voi

valita nimimerkin ja aloitusviestin jonoon liittymistä varten. Tiedot lähetetään HTTP POST -pyyntönä ja vastauksen saavuttua pyydetään tilatietoa palvelimelta. Jos tila on vaihtunut tilaan “Jono”, vaihdetaan näkymä vastaavaan. Jonotusnäkymässä muodostetaan ensimmäisen kerran asiakkaalle WebSocket-yhteys connectToServer-palvelussa. Asiakas tilaa hänelle osoitetun jonotuskanavan (eli pyytää palvelinta lähettämään sinne tulevat viestit myös hänelle). Viestin (“ammattilainen liittynyt kanavalle”) saapuessa pyydetään palvelimelta tilatietojen päivitys. Jos tilaksi on vaihtunut “chat”, katkaistaan jonokanavan tilaus ja aloitetaan näkymän vaihto keskustelunäkymään. WebSocket-yhteyttä ei katkaista välissä. Seuraavaksi tilataan varsinainen keskustelukanava ja aloitetaan sen kuuntelu. Käyttäjälle näytetään kanavalle tulleet viestit ja hän voi lähettää sinne viestejä. Asiakkaan lopettaessa keskustelun lopetus-napilla lähetetään POST-pyyntö kanavalta poistumiseksi. Vastauksen tultua katkaistaan tilaus. Jos asiakas lataa sivun uudelleen, pyydetään taas palvelimelta tilaa, joka on poistettu latauksen yhteydessä, ja näytetään asiakkaalle uutta istuntoa vastaava tila.

Ammattilaisen sivulle tultaessa ladataan ammattilaisen sivun staattiset Javascript-tiedostot. Palvelimelle lähetetään GET-kysely siitä, liittyykö tähän istuntoon kirjautuminen. Koska käyttäjä on uusi, eikä häneen liity kirjautumista, näytetään hänelle kirjautumisnäkymä. Käyttäjä lähettää kirjautumistiedot samaan osoitteeseen GET-pyynnöllä HTTPBasicAuthentication-protokollaa käyttäen. Jos tunnukset ovat palvelimen mielestä oikeat, aloitetaan näkymän vaihto.

Jos käyttäjä on ammattilainen, vaihdetaan näkymään ammattilaisen hallintanäkymä. ProStateService pyytää GET-pyynnöllä tiedot kirjautuneesta ammattilaisesta. ConnectToServer-palvelu muodostaa WebSocket-yhteyden ja proQueue-kontrolleri tilaa jonon seurantakanavan. Palvelin lähettää viestin jonon tilasta ja se tallennetaan queuePro-palveluun. Kontrolleri ja palvelu huolehtivat yhdessä jonon tilan päivityksistä ja jonosta nostamisesta. Kun ammattilainen aloittaa asiakkaan noston jonosta, hän tilaa saman jonotuskanavan, jolla asiakas jonottaa. Palvelin lähettää kanavalle viestin siitä, kenelle ammattilaiselle kanava on annettu. Jos se vastaa ProStateServicessä olevia tietoja, avataan ammattilaiselle uusi välilehti, jossa kyseinen chat-keskustelu voi alkaa. Ammattilainen siis tilaa asiakasta vastaavan keskustelukanavan. Ammattilainen voi lopettaa keskustelun eli sulkea välilehden napista, joka lähettää “leave channel” -POST-pyynnön palvelimelle. Hänellä on siis auki yksi tilaus jokaista välilehteä kohti. Ammattilainen voi vaihtaa tilakseen Paikalla tai Poissa. Kun hän vaihtaa tilaansa, lähetetään POST-pyyntö, jonka parametrissa kerrotaan uusi paikallaolotila.

Ammattilainen voi mennä myös katsomaan keskusteluhistoriaansa. Historianäkymästä huolehtivat ProHistory-palvelu ja histories-kontrollerit. Listaus historiallisista keskusteluista haetaan GET-pyynnöllä näkymän vaihtuessa historianäkymään. Avoimien kanavien tilaukset katkeavat historianäkymään mentäessä ja ne muodostetaan uudelleen käyttäjien hallintaan palattaessa. Kun ammattilainen valitsee listauksesta yhden historiallisen keskustelun, pyydetään GET-pyynnöllä palvelimelta keskusteluun liittyvät viestit. Ammattilainen voi kirjautua ulos hallintanäkymässä olevasta napista, jolloin katkaistaan tilaukset ja WebSocket-yhteys. Uloskirjautuminen myös muuttaa ammattilaisen paikallaolotilaksi “poissa”.

Jos käyttäjä on ylläpitäjä, näytetään käyttäjien hallintanäkymä. Käyttäjälista haetaan GET-pyynnöllä. Ylläpitäjä voi lisätä uuden käyttäjän, vaihtaa käyttäjän salasanan tai poistaa käyttäjän. Lisäys ja salasanan vaihto tapahtuu POST-pyynnöllä, poistaminen DELETE-pyynnöllä. Ylläpitäjän näkymästä vastaa admin-palvelu ja admin-kontrolleri.

Yhteisenä kaikille kuuluu heartbeat-palvelu, joka lähettää tyhjän POST-pyynnön palvelimelle 16 minuutin välein. Tällä pidetään HTTP-istunto elossa. Milloin tahansa, kun minkä tahansa tyyppinen käyttäjä päivittää sivun, haetaan tälle tilatiedot palvelimelta ja ladataan vastaava näkymä uudestaan sekä muodostetaan tarvittavat yhteydet. Kirjautuneen käyttäjän eli ammattilaisen tai ylläpitäjän mennessä asiakkaan sivulle näytetään tälle virheilmoitus, koska ammattilainen ei voi olla samassa istunnossa asiakas.

Käytetyt teknologiat

Selainpuolen toiminnallisuus on toteutettu AngularJS:llä, palvelinpuolen toiminnallisuus Javan Spring -sovelluskehyksellä. Kommunikointi tapahtuu sekä HTTP- että WebSocket-protokollaa hyödyntäen.

Selainpuolella:

- HTML5: Käyttäjänäkymien rungot
- CSS: Käyttäjänäkymien tyylittely
- JavaScript: Toiminnallisuus selainpuolella
- AngularJS: Sovelluskehys single-page -applikaatioiden tekemiseen JavaScriptillä
- SockJS: Kirjasto WebSocket-toiminnallisuuden tarjoamiseen myös vanhoihin selaimiin
- STOMPJS: Sanomien vaihdon toteuttamiseen STOMP-protokollalla
- ScrollGlue AngularJS-direktiivi vierityspalkin lukitsemiseen. Esim. chatissa mahdollistaa sen, että käyttäjä näkee aina uudet viestit keskusteluikkunan alareunassa.
- FocusIF: AngularJS-direktiivi fokuksen hankkimiseksi HTML-elementistä tietyissä olosuhteissa. Esim. kun chatin käyttäjä siirtyy keskustelunäkymään, osaa AngularJS automaattisesti valita kirjoituspalkin keskustelunäkymän alareunasta, jolloin käyttäjä voi alkaa saman tien kirjoittaa. Käyttäjän ei siis itse tarvitse valita hiiren painalluksella kirjoituspalkkia.
- Bootstrap: CSS-kirjasto, joka tarjoaa paljon toiminnallisuutta näkymien ulkoasun tyylittelyyn

Palvelimen puolella:

- Java: Olio-ohjelmointikieli
- Maven: Rakennustyökalu projektin riippuvuuksien hallintaan
- Spring: Sovelluskehys web-sovellusten rakentamiseen Javalla
- Spring Boot: Sovelluskehys Spring-sovelluskehyksen konfiguroinnin helpottamiseksi

- Tomcat: Tarjoaa web-palvelimen infrastruktuurin osana Spring Boot:ia
- Spring Security: Osa Spring-sovelluskehystä. Mahdollistaa ammattilaiskäyttäjien varmentamisen ja käyttäjäoikeuksien rajaamisen
- Spring Session: Osa Spring-sovelluskehystä. Mahdollistaa käyttäjien tunnistamisen istuntoevästeiden avulla
- Joda-time: Java-kirjasto aikojen ja päivämäärien käsittelyyn
- Gson: Java-kirjasto, jonka tarkoitus on helpottaa JSON-muotoisten sanomien käsittelyä
- JUnit: Sovelluskehys Java-puolen yksikkötestaukseen

Tietokantatoteutus:

- SQL: Kyselykieli tietokantaoperaatioiden suorittamiseen
- JDBC: Rajapinta, joka mahdollistaa tietokantayhteydet Java-sovelluksesta
- HikariCP: Yhteyskonsortio JDBC-tietokantayhteyksien nopeuttamiseksi
- Java Persistence API: Rajapinta SQL-tietokantaoperaatioiden suorittamiseen
- Hibernate: Sovelluskehys, joka toteuttaa yllä mainitun JPA-rajapinnan
- H2: Muistissa oleva SQL-tietokanta.
- PostgreSQL: SQL-tietokanta, jonne talletetut tiedot talletetaan levylle pitkäkestoiseen muistiin.

Testaus:

- Selenium, ChromeDriver: Tarjoaa toiminnallisuuden verkkosivujen html-elementtien käsittelyyn, kuten nappien painamiseen ja syötteiden antamiseen. Apuväline integraatiotestauksessa.
- Mockito: Sovelluskehys testaamisen helpottamiseksi. Mahdollistaa luokan riippuvuuksien väärentämisen.
- Jacoco: Apuväline testien rivi- ja haarautumakattavuusraporttien tuottamiseen.
- PITest: Tekee koodiin pieniä muutoksia ja tutkii käyttäytyvätkö testit tällöin oikealla tavalla. PITest on siis apuväline nk. mutaatiotestaukseen.
- CheckStyle: Työkalu koodin ulkoasun ylläpitoon. Vertaa ohjelmakoodia ennalta määriteltyihin tyylisääntöihin ja huomauuttaa, jos koodi ei ole näiden sääntöjen mukainen.
- Jasmine: Sovelluskehys JavaScriptin yksikkötestaamiseen.

Rajapinnat:

- QueueBroadcaster: Rajapinta jonon tilanteesta tiedottavalle luokalle QueueBroadcasterImpl
- ConversationRepo: Rajapinta keskustelujen tallentamiseksi tietokantaan
- MessageRepo: Rajapinta viestien tallentamiseksi tietokantaan
- PersonRepo: Rajapinta ammattilaiskäyttäjien tunnusten tallentamiseksi tietokantaan

Tietokannan rakenne: