

Ida Kilpeläinen – Jeremias Korhonen – Toni Naumanen –
Mikko Puustinen

Remind Me

Ohjelmistotuotantoprojekti 2

Sisällys

1	Tavoite	1
1.1	Alkuperäinen tavoite	1
1.2	Kuinka tavoite on muuttunut?	1
2	Ominaisuudet	1
2.1	Kirjautumisruutu	2
2.2	Asiakas	3
2.2.1	Kotinäkymä	3
2.2.2	Kalenterinäkymä	4
2.2.3	Terveysteni -näkymä	6
2.2.4	Apua -näkymä	8
2.3	Ylläpitäjä	9
2.3.1	Lisäysnäkymä	9
2.3.2	Muokkausnäkymä	10
2.4	Työntekijä	11
2.4.1	Varauksen muokkaaminen ajanvaraukset -näkyssä	11
2.4.2	Reseptien hallinnointi ajanvaraukset -näkyssä	12
3	Ohjelmointi	12
3.1	Temp -tiedostot	13
3.2	DAOManager	14
3.3	Enum	15
3.4	Lokalisointi	16
3.5	Maven	16
3.6	Testaus	16
3.7	Koodin kommentointi ja ymmärtäminen	17
4	Haasteet	18

1 Tavoite

1.1 Alkuperäinen tavoite

Tavoitteena oli rakentaa hyvinvointi- ja terveyssovellus muistisairaille asiakkaille. Asiakas voi sovelluksesta nähdä tulevat ajanvaraukset, päivän sään ja suosituksen vaatetuksesta, viestikentän, mihin työntekijä (lääkäri, hoitaja tai asiakaspalvelija) voi lähettää asiakkaalle viestejä. Asiakas voi myös kirjoittaa ylös veriarvonsa. Työntekijä voi nähdä asiakkaan veriarvot ja kertoa hänelle etänä ohjeita ja toimenpiteitä viestikenttään. Sovelluksen ideana on vähentää tarpeettomia lääkäri-/hoitajakäyntejä, arvioida paremmin avun tarvetta ja etänä antaa asiakkaalle suoraan ohjeita. Lyhyesti sanottuna sovellus helpottaa asiakkaan ja omaisten elämää.

1.2 Kuinka tavoite on muuttunut?

Projektiin saatiin lisää vaatimuksia, jolloin myös työtehtävät lisääntyivät. Tämä, ja ajanpuute, tarkoittivat sitä, että ihan kaikkia alkuperäisessä tavoitteissa mainitsemia ominaisuuksia ei pystytty toteuttamaan. Vaatimuksiin kuului mm. asiakkaan kuvan lisääminen sovellukseen ja kalenteritoiminnon laajentamisen. Projektista jätettiin pois vaatetuksen suositukset sekä työntekijän näkymän toimintoja ajanpuutteen vuoksi.

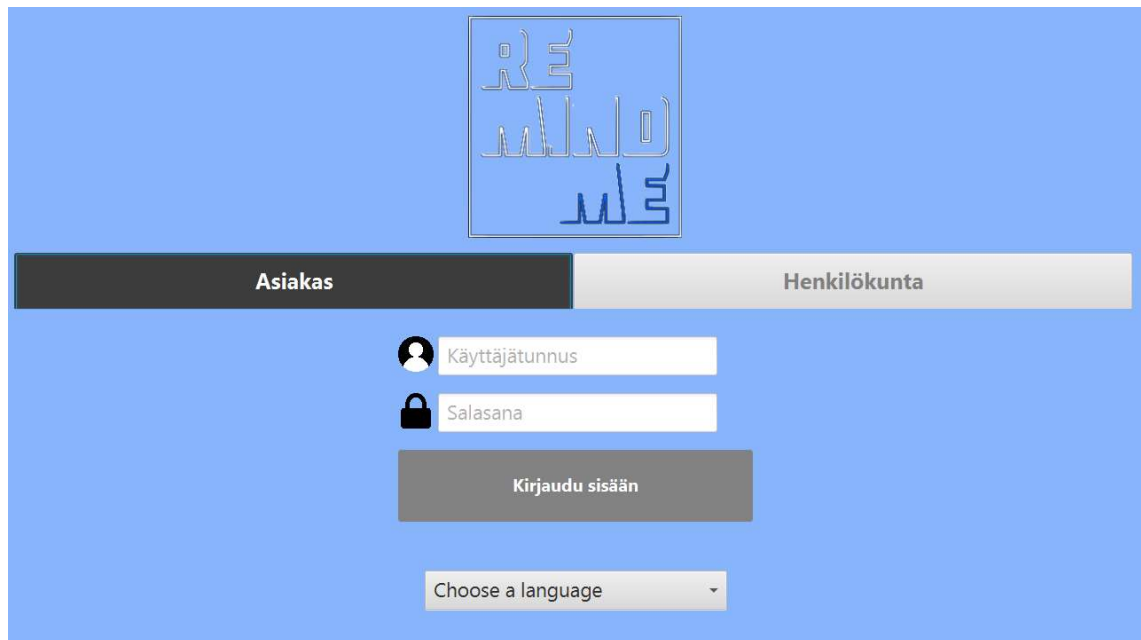
2 Ominaisuudet

Asiakas kirjautuessaan sisään näkee tervehdysruudun, päivän sään, tulevat ajat ja viestikentän, mikä näyttää mahdolliset ilmoitukset ja viestit henkilökunnalta. Asiakkaan näkymässä on tarjolla Koti-näkymän lisäksi myös Kalenteri-, Terveysteni- ja Apua-näkymät.

Henkilökunta voi lisätä ajanvarauksia, reseptejä ja lähettää viestejä halutulle asiakkaalleen. Ylläpitäjä voi lisätä henkilöitä tietokantaan ja muuttaa jo olemassa olevien henkilöiden tietoja tai käyttöoikeuksia.

2.1 Kirjautumisruutu

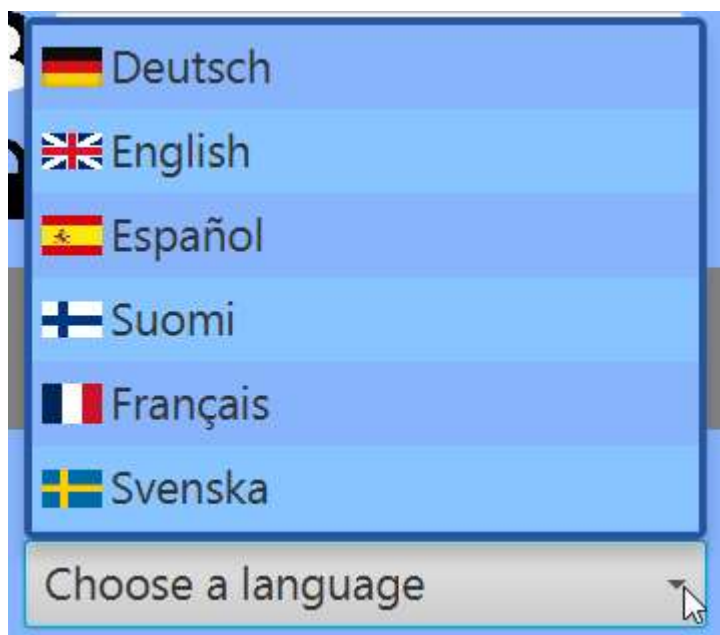
Kirjautumisruudussa on erikseen välilehdet asiakkaan ja henkilökunnan kirjautumiselle (Kuva 1).



The image shows a login interface with a light blue background. At the top center is a logo consisting of stylized letters 'RE' and 'W' in a square frame. Below the logo are two tabs: 'Asiakas' (highlighted in dark grey) and 'Henkilökunta' (light grey). Under the 'Asiakas' tab, there are two input fields: 'Käyttäjätunnus' (with a person icon) and 'Salasana' (with a lock icon). Below these is a dark grey button labeled 'Kirjaudu sisään'. At the bottom is a dropdown menu labeled 'Choose a language'.

Kuva 1. Kirjautumisruutu.

Asiakas voi kirjautumisruudussa vaihtaa haluamansa kielen (Kuva 2). Tällä hetkellä tuettuja kieliä ovat suomi, englanti, espanja, ruotsi, saksa ja ranska.



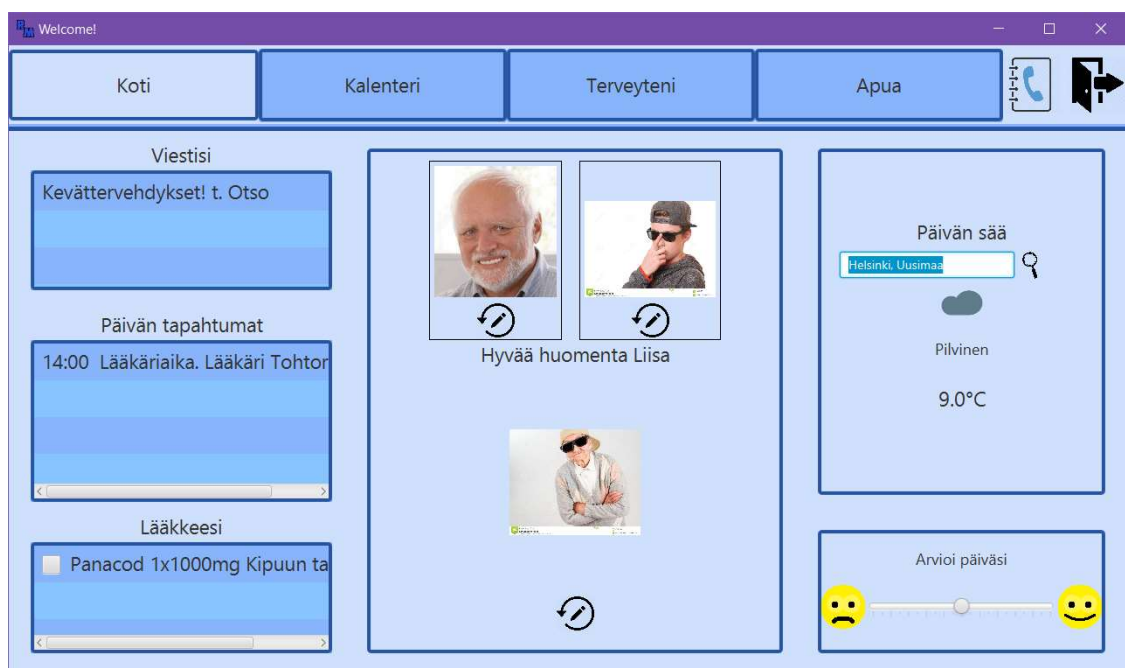
Kuva 2. Kielen valitseminen.

2.2 Asiakas

Asiakkaalla on neljä erillistä näkymää, Koti-, Kalenteri-, Terveysteni-, ja Apua-näkymä. Sovelluksen avautuessa ensimmäisenä avautuu Koti-näkymä. Asiakas voi navigoida sovelluksen sisällä aina näkyvillä olevan ylävalikon avulla.

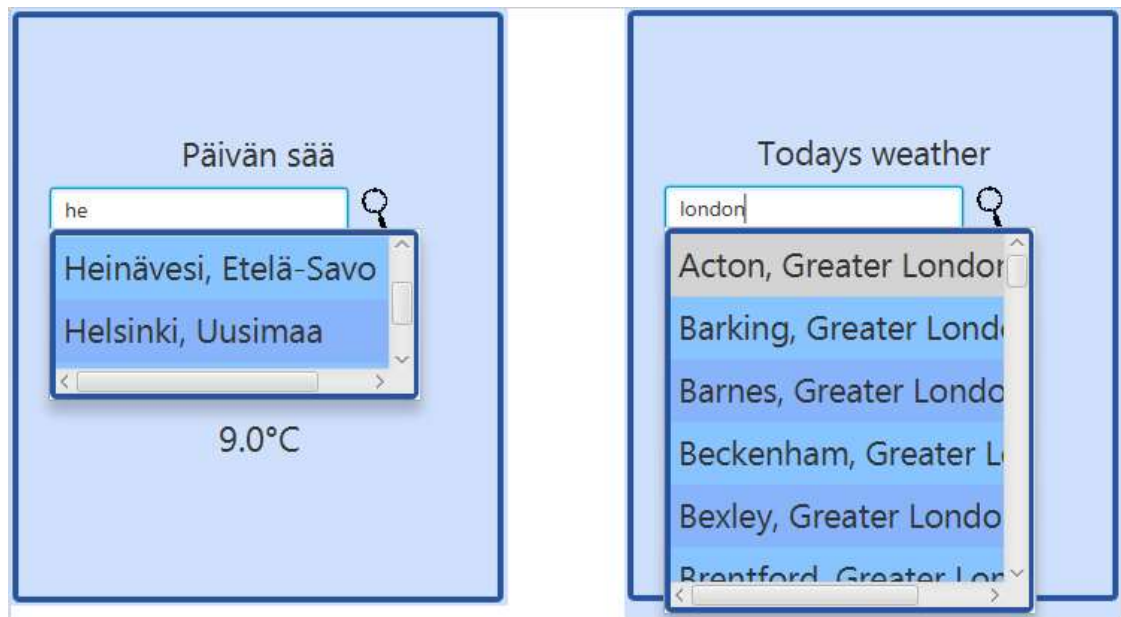
2.2.1 Kotinäkymä

Kotinäkymä avautuu asiakkaan kirjautuessa sisään. Näkymästä löytyy asiakkaalle lähetetyt viestit, päivän tapahtumat ja päivän sää. Asiakas voi myös lisätä oman, sekä läheistensä kuvia kotisivulle. (Kuva 3.)



Kuva 3. Asiakkaan etusivu.

Kotinäkymässä asiakas voi myös merkata lääkkeit otetuksi ja vaihtaa jo asettamansa kuvat. Päivän sään voi valita paikkakunnan mukaan, apuna on luettelo kaikista Suomen kunnista, jos kielenä on suomi. Kielen ollessa englanti, sovellus ehdottaa paikkakuntia Isosta-Britanniasta (Kuva 4).



Kuva 4. Sää, paikkakunnat.

2.2.2 Kalenterinäkymä

Kalenterissa on kuukausi-, viikko-, ja päivänäkymät. Kuukausinäkymässä ajanvaraukset näkyvät vaaleammalla sinisellä (Kuva 5). Tätä painamalla pääsee sen päivän päivänäkymään. Kuukautta voi vaihtaa nuolia klikkaamalla.

Koti

Kalenteri

Terveysteni

Apua





Kuuk...

<

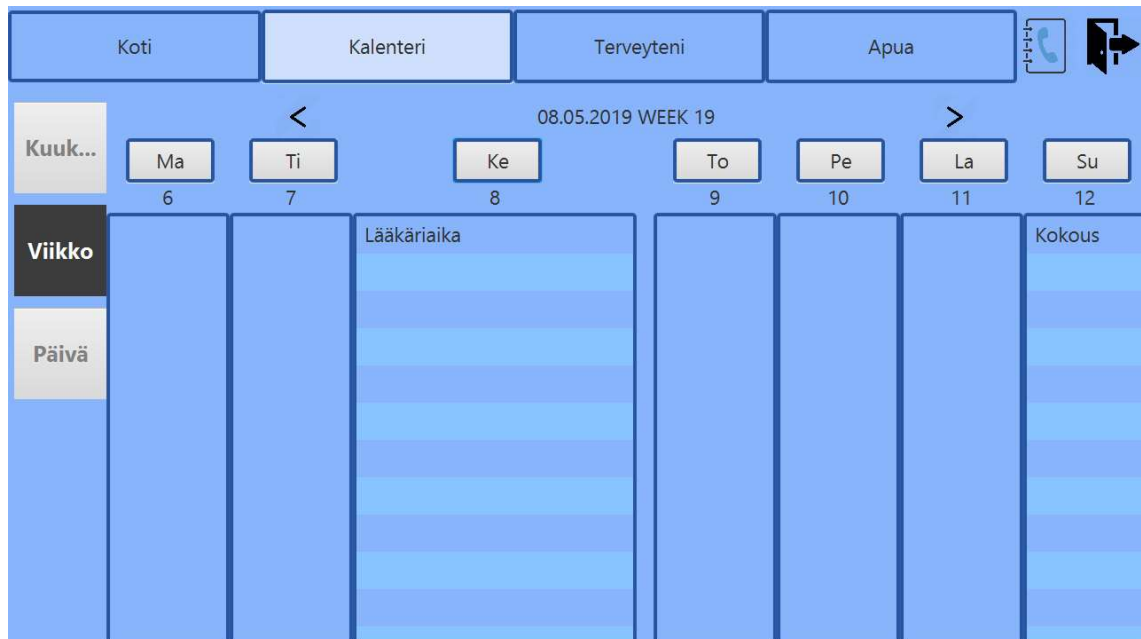
MAY 2019

>

Ma	Ti	Ke	To	Pe	La	Su
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

Kuva 5. Kalenteri, kuukausi.

Viikkonäkymään tulee asiakkaan sen viikon ajanvaraukset sekä muut tapahtumat. Päivävalikon nappia painaessa kenttä levenee (Kuva 6).



Kuva 6. Kalenteri, viikkonäkymä, keskiviikko painettu.

Päiväkalenterissa ajanvaraukset näkyvät listassa (Kuva 7). Listanäkymä vierittyy oikeaan kohtaan sen mukaan, mikä sen hetkinen kellonaika on.



Kuva 7. Kalenteri, äivänäkymä.

Päivää voi vaihtaa päivämäärän molemmilla puolilla olevia nuolia klikkaamalla.

2.2.3 Terveysteni -näky


Terveysteni -näkyssä on asiakkaan terveyteen liittyviä asioita, kuten reseptit ja verenpaine- ja verensokerimittausten kirjaaminen. Alla olevassa kuvassa näkyy asiakkaan kaikki tietokannassa olevat reseptit ja asiakas voi tarvittaessa lähettää reseptin uusintapyyntöä nappia painamalla (Kuva 8.)


Koti

Kalenteri

Terveysteni

Apua





Reseptit

Verenpaine

Verensokeri

Valitse resepti nähdäksesi lisätietoja

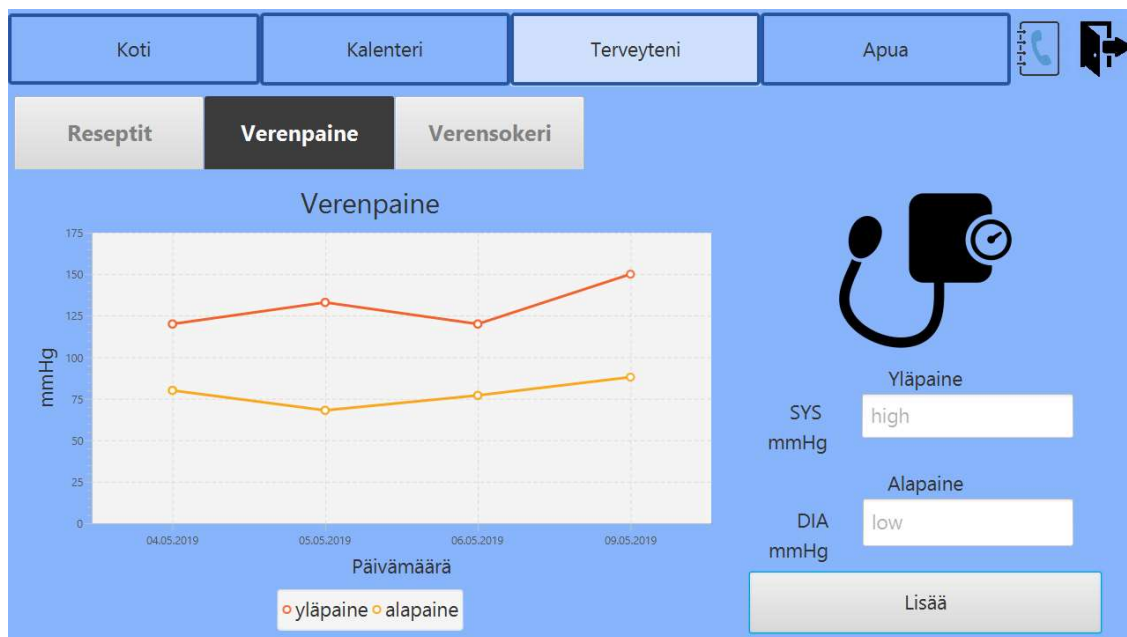
08.05.2019 - 08.05.2020 Kipuun tarvittaessa. Lääkäri Tohtori, Otso

Nimi	Annostus	Ohje	Otto aika	Uusi
Panacod	1x1000mg	Kipuun tarvittaessa	tarvittaessa	Uusi resepti

Kuva 8. Asiakkaan reseptit.

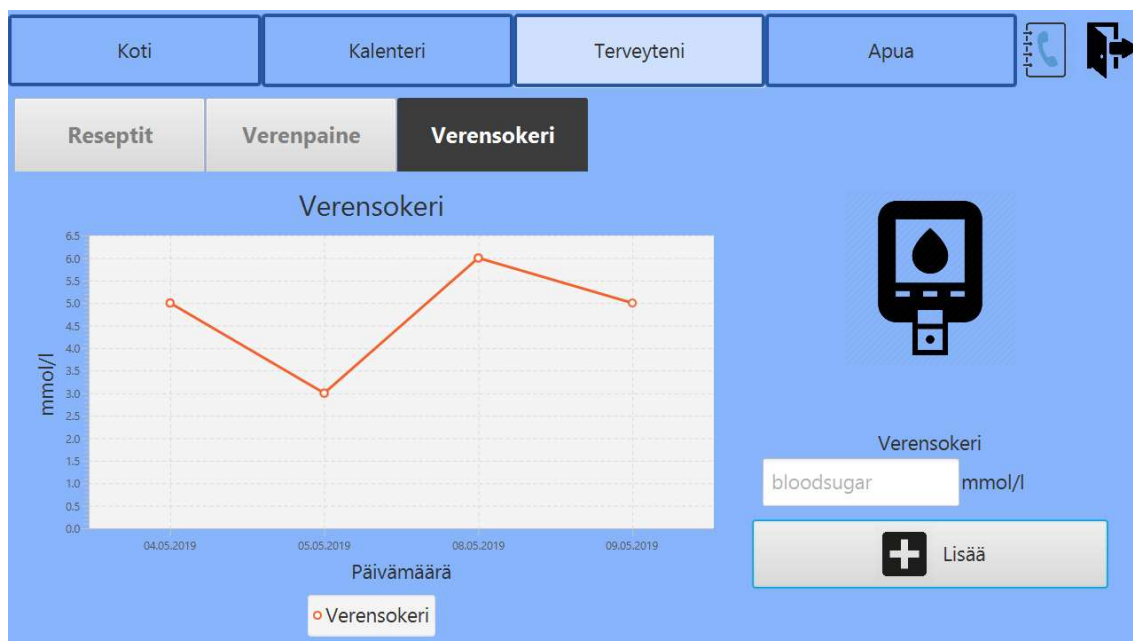
Asiakas voi lisätä verenpaine-arvoja tietokantaan, jonka avulla verenpaine-arvojen historiasta koostetaan diagrammi, jonka voi nähdä alla olevasta kuvasta (Kuva 9). Diagrammi päivittyy heti arvon lisäämisen jälkeen. Demossa unohdettiin kirjoittaa verenpaine-arvo temp-tiedostoon, jolloin päivitys ei onnistunut, nyt tämä on toteutettu.

```
daom.writeBloodValueToFileDuringSession(customer);
```

Kuva 9. Verenpainearvojen lisääminen.

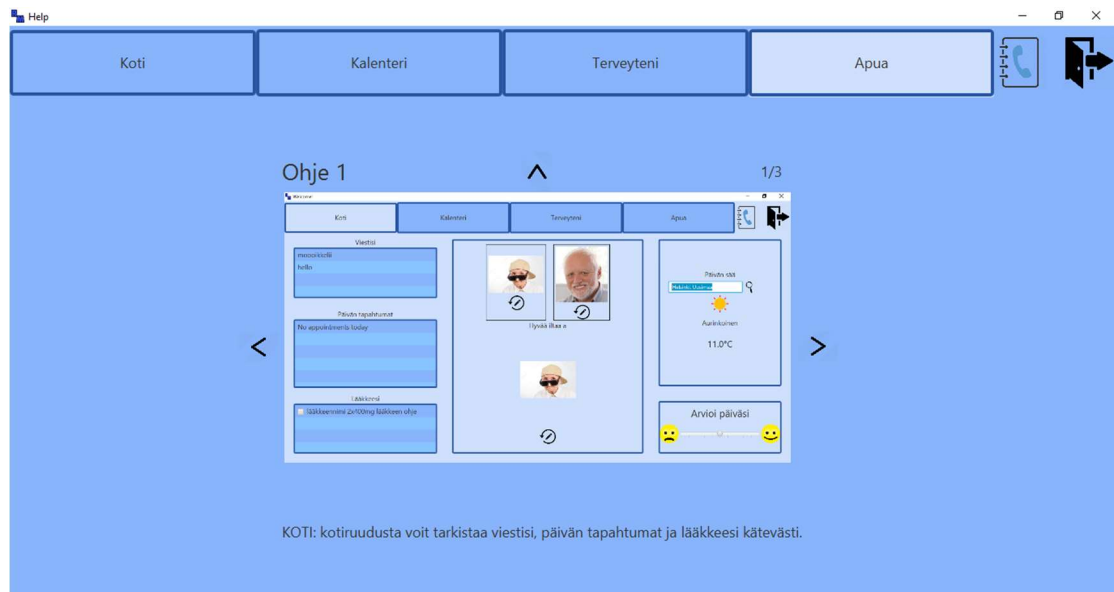
Verensokeriarvojen lisääminen ja näyttäminen toimii samalla tavalla kuin verenpainearvojen (Kuva 10).



Kuva 10. Verensokeriarvojen lisääminen.

2.2.4 Apua -näkymä

Apua -näkymän tarkoituksena on tarjota sovellusta käyttävälle asiakkaalle vinkkejä sovelluksen käyttöön ja kertoa eri näkymien tarkoituksista (Kuva 11). Näkymä on yksinkertainen ja se tarjoaa vain kuvaesityksen, jota voi selata käyttämällä kuvan sivuilla olevia nuolia. Kuvan alla kerrotaan mikä näkymä on kyseessä ja mitä kyseinen näkymä tarjoaa asiakkaalle.



Kuva 11. Apua -näkymä.

2.3 Ylläpitäjä

Ylläpitäjän näkymä sisältää näkymät asiakas- ja henkilökuntatilien hallintaan.

2.3.1 Lisäysnäkymä

Lisää työntekijä	Lisää asiakas	Muokkaa työntekijää	Muokkaa asiakasta
Etunimi: esim. Pekka		Puhelinnumero: esim. +358 401234567	
Sukunimi: esim. Korhonen		Henkilötunnus: esim. DDMMYY-XXXX	
Osoite: esim. Kotikatu 5		Sähköposti: esim. esimerkki@email.co	
Salasana: esim. salasana123		ICE-kontakti: esim. +358 401234567	
<button>Lisää asiakas</button>			

Kuva 12. Lisää asiakas -näkymä.

Lisää asiakas- ja lisää työntekijä -näkymissä voidaan lisätä uusia asiakkaita ja työntekijöitä (Kuva 12).

(18)

2.3.2 Muokkausnäkymä

Kuva 13. Työntekijän muokkaaminen.

Työntekijän muokkausnäkyssä (Kuva 13) on mahdollista hakea tietokannasta työntekijää ja muokata tämän tietoja. Kun näkymä avataan, ohjelma hakee tietokannasta kaikki työntekijät, jotta hakukentän automaattinen täydennys saadaan toimimaan sulavasti. Näkyssä on myös lista kaikista asiakkaista sekä lista haettuun työntekijään jo liitettyistä asiakkaista. Kaikkien asiakkaiden listasta voidaan haettuun työntekijään liittää uusia asiakkaita Add-painikkeella. Molemmat asiakaslistat täytetään samalla, kun työntekijää haetaan eriaikaisten tietokantakyselyiden vähentämiseksi.

(18)

2.4 Työntekijä

Työntekijän näkymässä voi hallita asiakkaiden ajanvarauksia, reseptejä ja viestintää.

2.4.1 Varauksen muokkaaminen ajanvaraukset -näkymässä

Kuva 14. Ajanvarauksen muokkaaminen.

Ajanvaraukset-näkymä (Kuva 14) toimii siten, että työntekijä valitsee ensiksi asiakaslistasta asiakkaan, jonka lataa järjestelmään. Tällöin ladatun asiakkaan ajanvaraukset menevät varausten listanäkymään muokattavaksi/poistettavaksi.

(18)

2.4.2 Reseptien hallinnointi ajanvaraukset -näkyssä

Kuva 15. Reseptin uusiminen.

Reseptinäkömä (Kuva 15) on avoinna vain työntekijälle, kenen valtuus on tohtori. Toimii samalla tavalla kuin ajanvaraus eli lataat ensiksi asiakkaan (tämä toimenpide tarvitsee vain kerran tehdä) ja hänen mahdolliset reseptinsä menevät reseptien listanäkymään muokattavaksi. Lataa/uusi -näppäimellä reseptin tiedot ilmestyvät oikeisiin kenttiin. Jos reseptiä ei ole valittu, uusi resepti luodaan, mihin voit haluamasi tiedot lisätä.

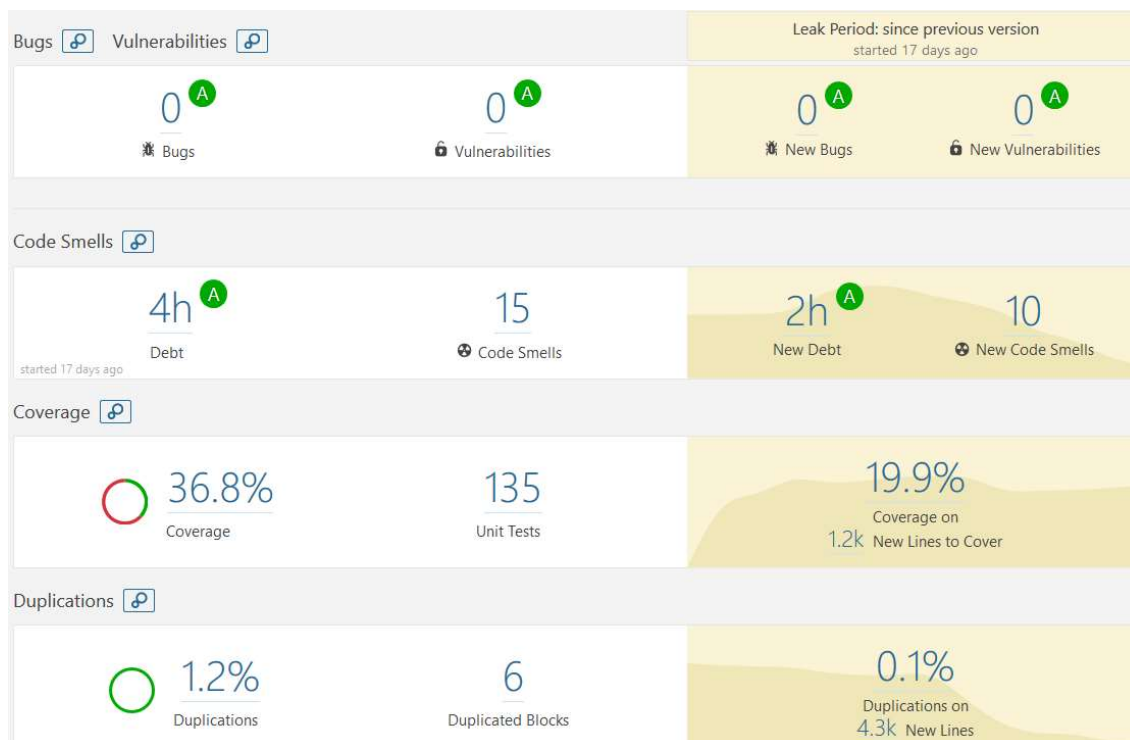
Viestinäkömässä on yksinkertainen tekstilaatikko, missä voit haluamallesi asiakkaalle lähettää viestin.

3 Ohjelmointi

Sovellus on ohjelmoitu Java-kielellä (JDK 1.8) Maven-projektina ja ulkoasu JavaFX:llä. Tietokannassa on taulut asiakkaiden ja henkilökunnan tietoihin, ajanvarauksiin, resepteihin, viesteihin, sairauksiin, kuviin (tallennus Blob-tyyppinä) sekä asiakkaan ja henkilökunnan yhdistämisiin tarkoitettu customersStaff-taulu (AsiakasID ja HenkilökuntaID). Yhteydet tietokantaan on helpotettu käyttäen Hibernate-rajapintaa. Sovelluksen käyttäjien salasanat ovat salattu SCryptUtil-kirjastolla (hashing).

(18)

Sovelluksen testit ovat tehty JUnit5-rajapinnalla ja automatisoitu Jenkins-serverillä. Jenkinsiin liitetty SonarQube-alusta auttoi meitä korjaamaan yleisiä vikoja koodissamme (Kuva 16).



Kuva 16. SonarQube analyysi.

3.1 Temp -tiedostot

Käyttökokemuksen parantamiseksi kaikki asiakkaan tiedot kirjoitetaan väliaikaisesti temp-tiedostoihin kirjautumisen yhteydessä. Näin vähennetään turhia interaktioita tietokannan kanssa ja nopeutetaan sovelluksen toimintaa. Alla olevassa kuvassa (Kuva 17) esi-merkki asiakkaan kuvien kirjoittamisesta tiedostoon.

(18)

```

public void writeImagesToFile(Customer customer) {
    userImages = userImageDAO.readCustomerUserImages(customer);
    for (int j = 0; j < userImages.length; j++) {
        imageNames[j] = userImages[j].getImageName();
        try{
            files[j] = File.createTempFile("userimages", null);
        } catch (IOException e1) {}
        try (FileOutputStream os = new FileOutputStream(files[j])) {
            BufferedImage img = ImageIO.read(new ByteArrayInputStream(userImages[j].getImage()));
            ImageIO.write(img, "png", files[j]);
            files[j].deleteOnExit();
        } catch (IOException e) {
            Log.severe("An error occurred while opening the file.");
        }
    }
}
}

```

Kuva 17. Kuvat temp -tiedostoon.

Tiedostojen hallinnoimiseen luotiin kaksi omaa luokkaa. Yksi kuvien kirjoittamiseen ja lukemiseen, toinen muille tiedoille. Molemmat luokat ovat singletonia. Tiedostot poistetaan sovelluksen sulkemisen yhteydessä.

3.2 DAOManager

Data access object -luokkien hallinnointiin luotiin DAOManager-luokka, joka mahdollistaa DAO -metodien käyttämisen controller -tasolla (Kuva 18). Tällöin DAO-luokkia ei tarvitse tuoda (import) controller -luokissa.

```

public CustomerDAO getCustomerDAO() {
    if (this.customerDAO == null) {
        this.customerDAO = new CustomerDAO(s);
    }
    return this.customerDAO;
}

```

Kuva 18. Asiakas DAO:n kutsumetodi.

DAOManagerissa on myös metodit temp -tiedostojen kirjoittamista ja lukemista varten. Alla olevassa kuvassa (Kuva 19) koodiesimerkki DAOManagerin metodista, jota kutsutaan kirjautumisen yhteydessä.

(18)

```

public void writeAllCustomerInformation(Customer customer) {
    info.writeAppointmentsToFile(customer);
    info.writePrescriptionsToFile(customer);
    info.writeBloodvaluesToFile(customer);
    i.writeImagesToFile(customer);
}

```

Kuva 19. Tietojen kirjoittaminen tiedostoon.

3.3 Enum

Projektissa käytetään Javan Enumia (Kuva 20). Tämä vähentää merkkijonojen kirjoittamista koodissa esimerkiksi FXML -tiedostoa ladattaessa:

```
FXMLLoader.Load(getClass().getResource(FxmlEnum.LOGIN.getFxml()), bundle);
```

```

public enum FxmlEnum {

    /**
     * fxml files for admin
     */
    ADDSTAFF("/fxml/AddStaffView.fxml"),
    ADDCUSTOMER("/fxml/AddCustomerView.fxml"),
    EDITSTAFF("/fxml/EditStaffView.fxml"),
    EDITCUSTOMER("/fxml/EditCustomerView.fxml"),

    /**
     * fxml files for login
     */
    LOGIN("/fxml/LoginView.fxml"),

    /**
     * fxml files for customer
     */
    CUSTOMERHOME("/fxml/CustomerHomeView.fxml"),
    CUSTOMERCALENDAR("/fxml/CalendarView.fxml"),
    CUSTOMERHELP("/fxml/CustomerHelpView.fxml"),
    CUSTOMERHEALTH("/fxml/CustomerHealthView.fxml"),

    /**
     * fxml files for staff
     */
    STAFFHOME("/fxml/StaffHomeView.fxml"),
    STAFFAPPOINTMENT("/fxml/StaffAppointmentView.fxml");
}

```

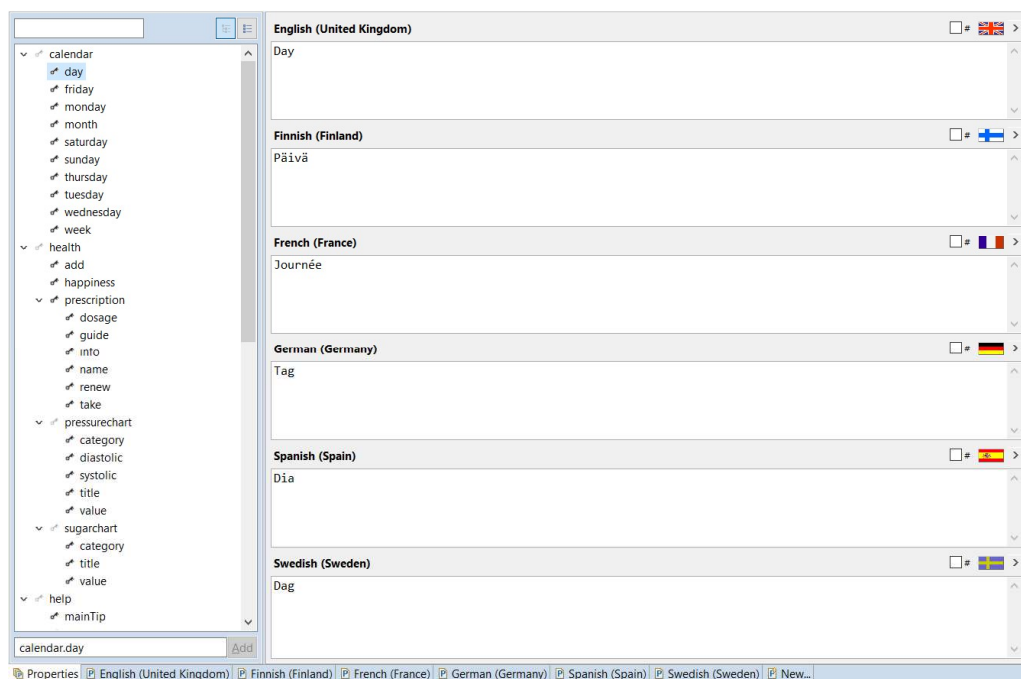
Kuva 20. FXML enum.

Tiedostonimet sisältävä enum helpottaa myös refaktorointia, sillä tiedostopolkua ei tarvitse muuttaa koodissa muualla kuin enumissa.

(18)

3.4 Lokalisointi

Lokalisoinnin toteuttamiseksi käytettiin properties -tiedostoja. FXML -tiedostojen tekstikentät vaihdettiin muotoon %key.specified, joille määritettiin arvo properties -tiedostossa, alla olevassa kuvassa (Kuva 21) esimerkki ResourceBundle editor -ohjelmasta, jolla pystyy määrittämään merkkijonojen arvon vaivattomasti.



Kuva 21. ResourceBundle editor.

3.5 Maven

Käännös- ja koontityökalu, jolla voidaan hoitaa paremmin riippuvuuksia projektien välillä ja kolmannen osapuolen kirjastoihin. Pom.xml -tiedostoon sai tarvittavat kirjasto/lisäosariippuvuudet, mitä hyödynnettiin sovelluskehityksessä. Tällä saadaan projekti loppujen lopuksi paketoitua levityskuntoon.

3.6 Testaus

Kaikki testit

Package	Kesto	Epäonnistui (muutos)	Ohitettiin (muutos)	Pass (muutos)	Yhteensä (muutos)
controller	2 sec	0	0	28 +2	28 +2
dao	0.9 sec	0	0	14	14
model	70 ms	0	0	65	65
view	32 sec	0	0	28	28

(18)

Sovelluksen testejä on yhteensä noin 140. Näissä testataan tietokantayhteyksiä, luokkia, kontrollereita ja näkymiä. Näkymät testattiin TestFX-rajapinnan avulla, alla esimerkki graafisen käyttöliittymän testimetodista (Kuva 22).

```
@Test
public void addCustomerNameTest() {
    TextField fname = lookup("#firstname").query();
    TextField surname = lookup("#surname").query();
    clickOn("#firstname");
    write("firstname");
    clickOn("#surname");
    write("surname");
    verifyThat(fname.getText(), Matchers.is("firstname"));
    verifyThat(surname.getText(), Matchers.is("surname"));
}
```

Kuva 22. TestFX testi.

Graafisen käyttöliittymän testaus piti implementoida käyttäen headless-tilaa, jotta Jenkins pystyi suorittamaan testit. Kaikille data access object -luokkien ja entity -luokkien metodeille on toteutettu testit. Alla esimerkki DAO -luokan create -metodin testistä. (Kuva 23).

```
@Test
public void testCreate() {
    UserImage image = new UserImage();
    byte[] bFile = new byte[10];
    image.setCustomer(customer);
    image.setImage(bFile);
    image.setImageName("test_image");
    image.setImageID(customer.getCustomerID()+1);
    assertTrue(iDAO.create(image), "create(UserImage): Failed to create image");
}
```

Kuva 23. DAO testi.

3.7 Koodin kommentointi ja ymmärtäminen

Kaikki metodit on nimetty siten, että ulkopuolinen kävijä ymmärtää heti, mitä metodi tekee. Metodien ja luokkien alussa on kommentoitu tarkemmin kyseisen tarkoituksesta.

4 Haasteet

Projektikehityksen suurimpana haasteena oli hyvän suunnitelman laatiminen. Koodaaminen oli aluksi helppoa, mutta ilman selkeitä kaavioita, päämäärää ja visiota, koodia jouduttiin muuttamaan jatkuvasti suunnitelmien muuttuessa.

Tietyt toiminnallisuudet kuten offline-tila ja aikamääräiset ilmoitukset olivat rajoitetun ajan ja taitojen vuoksi mahdottomia toteuttaa.

Ulkoasun parantaminen on aikaa vievää. Asiakasryhmää ajatellen fonttikokoa suurennettiin kaikissa elementeissä, mutta tämä myös aiheutti ongelmia näkyvyyden osalta. Jokaista näkymän elementtiä olisi joutunut manuaalisesti muokkaamaan .css -tiedostossa parhaan lopputuloksen saamiseksi.

Testejä olisi voinut tehdä enemmän controller -luokille, mutta ongelmia tuli siinä, kun piti käyttää samaa tietokannan taulua, DAO-testejen kanssa. Joidenkin model -tason luokkien id:nä käytetään juoksevaa lukua, jolloin esimerkiksi DAO-luokkien read(id) -metodi ei toimi oikein. Taulu pitäisi luoda uudestaan jokaisen testiluokan jälkeen.

SonarQuben kanssa oli paljon kummallisia ongelmia, kuten yhteys- ja muistiongelmia. Tämän korjaamiseen käytettiin todella paljon aikaa (vaihdettiin pom.xml:ssä plugin versioita useaan otteeseen yms.).