Ohjelmistotuotantoprojekti
Syksy 2004

Toteutusdokumentti

“Viisas pitää yllä järjestystä, Nero hallitsee kaaoksen”

Sisällysluettelo

Liitteet

1 Johdanto

Nero on työhuoneiden varauksien hallintajärjestelmä. Nero on kehitetty Helsingin Yliopiston ohjelmistotuotantoprojektina. Tarkoituksena on helpottaa huoneiden varauksien hallintaa siten, että huoneet ovat mahdollisimman hyvin käytössä ja huoneiden asukkaat ovat lähekkäin muihin saman alan henkilöihin nähden. Ohjelmiston omistaa Helsingin Yliopisto ja ohjelman lisenssiksi tulee GPL tai LGPL.

1.1 Dokumentin tarkoitus

Tämä dokumentti on tarkoitettu selvittämään suunnitteludokumenttia suppeammin miten sovellus on toteutettu. Ohjelman tarkempi toiminta selviää luokkakaaviosta ja liitteenä olevasta Javadoc-dokumentaatiosta. Poikkeamat suunnitteludokumentin määrityksistä esitellään lyhyesti. Käyttöliittymän rakenne kuvataan tarkemmin, koska se jäi suunnitteludokumentissa vähälle.

2 Yleiskuvaus

Tässä luvussa esitellään lyhyesti käyttöliittymän ja siihen liittyvän tietokannan lopullinen muoto.

2.1 Käyttöliittymä

Järjestelmän äly on sijoitettu kokonaisuudessaan käyttöliittymän sisälle ja näin ollen vain käsiteltävä tieto toimitetaan tietokantaan muutosten tapahtuessa.

2.1.1 Käyttöympäristö

Ohjelmisto on toteutettu Sun Java 1.4.2 järjestelmälle ja se on testattu myös 1.5.0 versiolla. Ohjelma vaatii yhteyden tietokantaan, joka sijaitsee nykyisellään (2.12.2004) osoitteessa *kontti.helsinki.fi*.

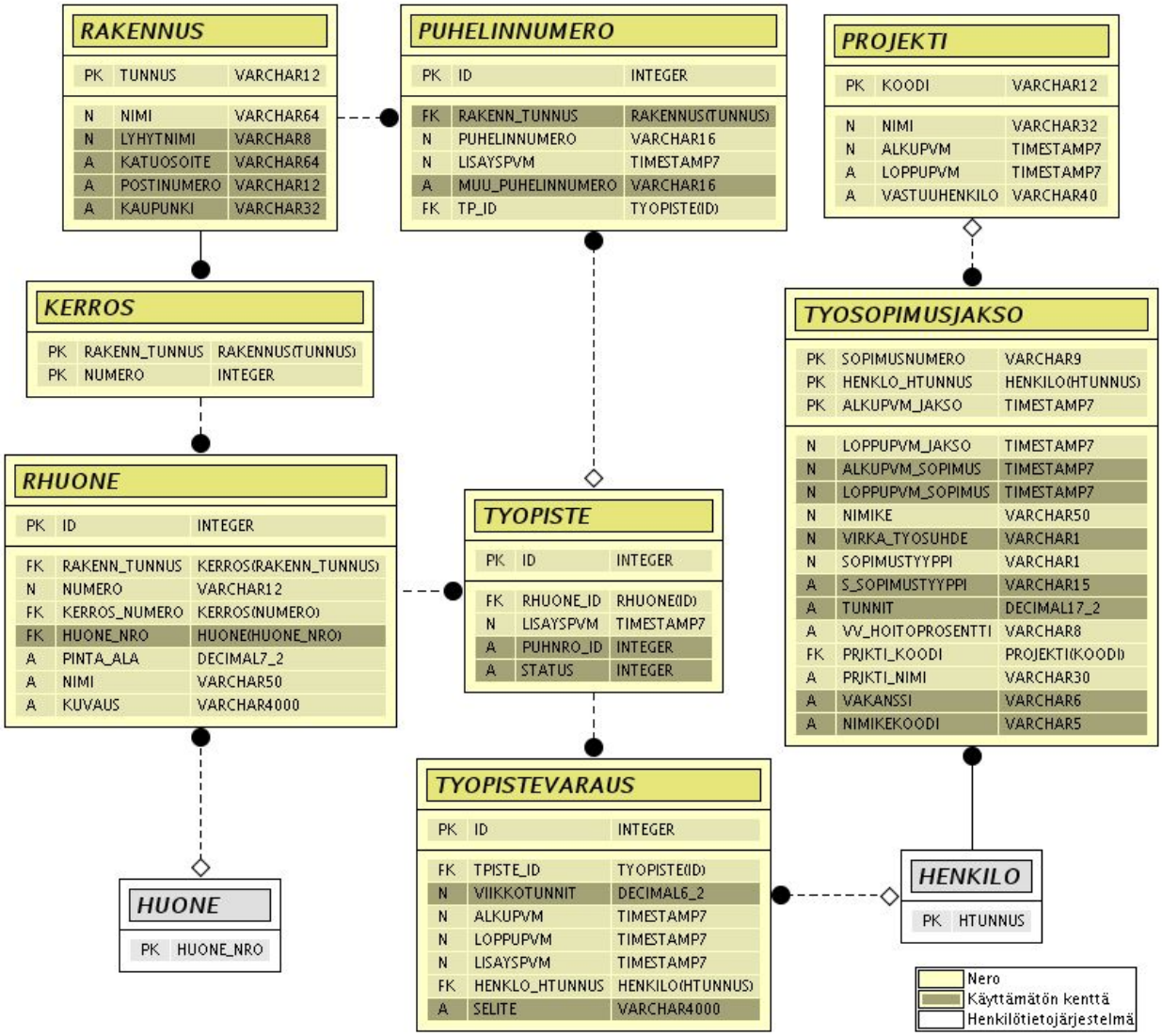
2.2 Tietokanta

Rooma-projektin tietokanta otettiin käyttöön minimaalisin muutoksin. Näin ollen Rooman kopiointiosajärjestelmä voitiin liittää suoraan Neroon. Tietokantana käytetään Oracle 8 Enterprise Edition Release 8.0.5.0.0. Koska kyselyt on tehty nimenomaan Oraclea ajatellen, ei voida taata, että ohjelma toimii muissa tietokantajärjestelmissä muutoksitta.

Tietokannan lopullinen rakenne on esitelty kuvassa 1 ja siitä käy myös ilmi mitkä kentät ovat täysin käyttämättömiä (Oracle 8:n rajoituksien vuoksi niitä ei poistettu). Oleellisimpia muutoksia Rooman kantaan verrattuna ovat TYOPISTE- ja PUHELINNUMERO-taulujen välisen viitteen kääntäminen toisin päin, sekä RHUONE -> HUONE -viittausten jättäminen null-arvoiksi.

Käskyt TYOPISTE- ja PUHELINNUMERO-taulujen muuntamiseen Rooma-järjestelmän tietokannasta ovat uuden TP_ID-sarakkeen luonti PUHELINNUMERO-tauluun (1), sen asettaminen viiteavaimeksi TYOPISTE-tauluun (2) sekä vanhan puhelinnumeroviitteen FK_TPISTE_PUHNRO poistaminen (3):

```
1. ALTER TABLE puhelinnumero ADD COLUMN tp_id NUMBER(20, 0);
2. ALTER TABLE puhelinnumero ADD CONSTRAINT fktp FOREIGN KEY (tp_id)
   REFERENCES tyopiste (id);
3. ALTER TABLE tyopiste DROP CONSTRAINT fk_tpiste_puhnro;
```



Kuva 1. Tietokannan rakenne

3 Käyttöliittymäkomponentit

Käyttöliittymäkomponenttien yhteydet luokkiin.

3.1 Hakuosio

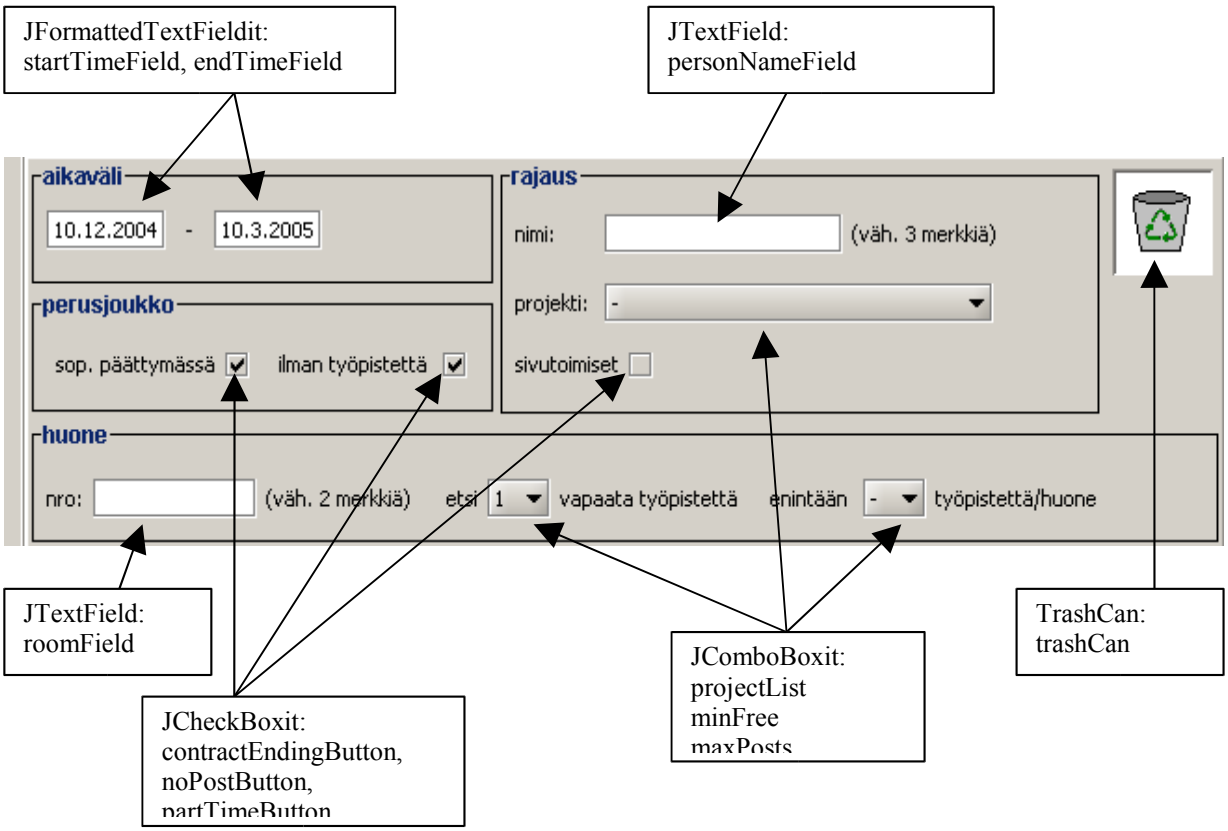
SearchPanel on JPanel luokan aliluokka, joka toteuttaa luokat PropertyChangeListener, ActionListener. Hakupaneelissa olevien kenttien muutokset välitetään sessiolle.

startTimeField ja endTimeField hyväksyvät arvoikseen vain päivämääriä. Kenttiä kuuntelee SearchPanel (PropertyChangeListener). Kenttien alkuarvot haetaan sessiolta.

contractEndingButtonin, noPostButtonin, partTimeButtonin alkuarvot kysytään myös sessio-oliolta. Kenttiä kuuntelee ButtonListener luokan (, joka toteuttaa ActionListenerin) ilmentymä. Kenttien alkuarvot haetaan sessiolta.

personNameFieldiä ja roomFieldiä kuuntelee FilterEventListener luokan(, joka toteuttaa DocumentListenerin ja ActionListenerin) ilmentymä. FilterEventListener välittää muuttuneen tiedon sessiolle vasta pienen viiveen jälkeen, jottei jokaisen merkin kohdalla suoritettaisi sessio-kutsua. Kentät ovat oletusarvoisesti tyhjinä.

projectListiä, minFreetä, maxPostsia kuuntelee SearchPanel (ActionListener). projectListin vaihtoehdot haetaan sessiolta. minFreen ja maxPostsin vaihtoehdot on kovakoodattu luokan sisälle. Valikoiden alkuarvot näkyvät kuvasta.



Kuva 2. Hakuosio

3.2 Puhelinnumeroikkuna

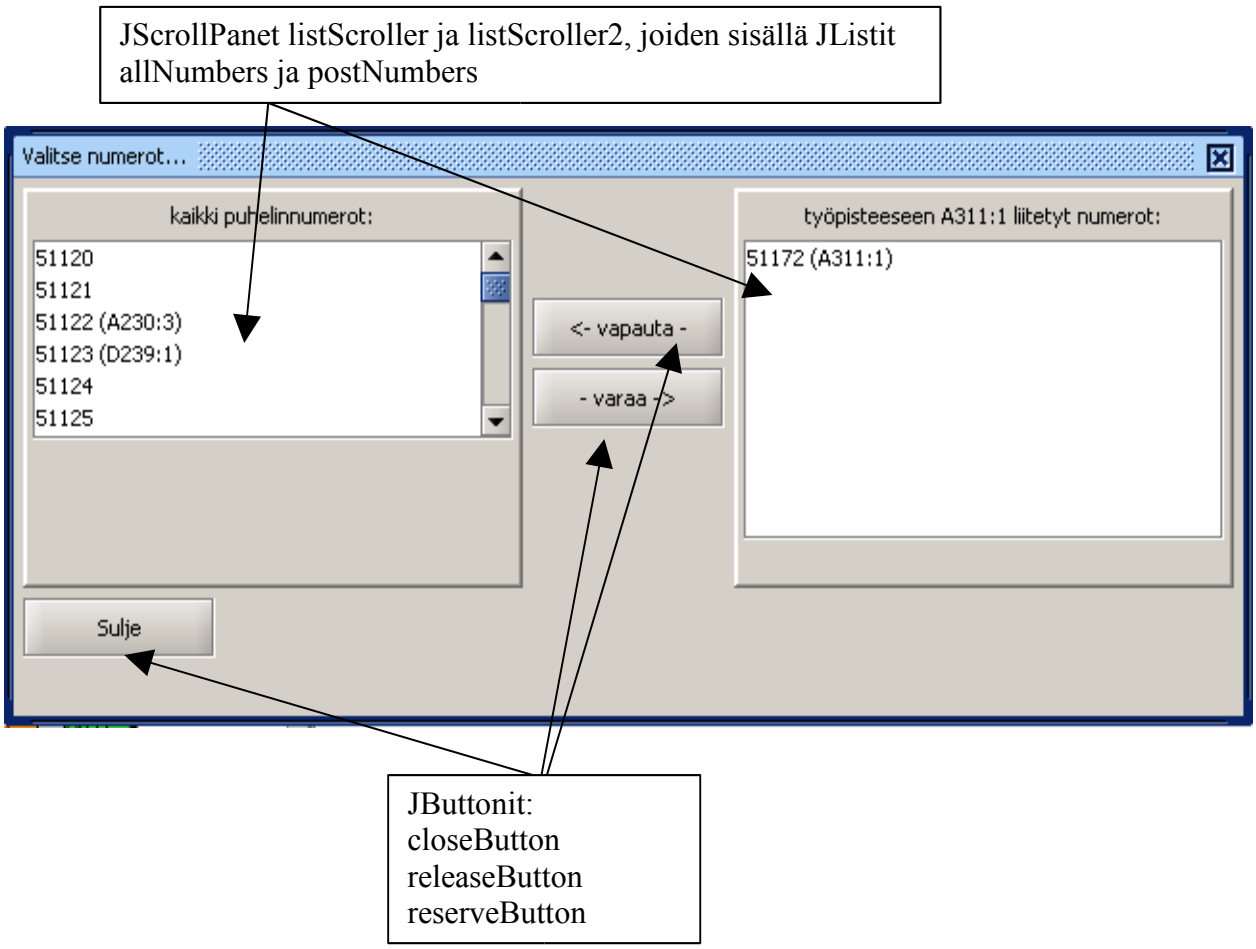
PhonenumberFrame on JDialogin aliluokka, joka toteuttaa rajapinnat ListSelectionListener ja ActionListener. Ensin mainittu kuuntelija ei ole käytössä. Puhelinnumeroiden varaukset ja vapautukset välitetään sessiolle.

allNumbers listan sisältämät puhelinnumerot saadaan sessiolta ja postNumbers listan sisältämät puhelinnumerot tulevat puolestaan käsiteltävänä olevalta työpisteoliolta.

reserveButton varaa allNumbers listalta valitun puhelinnumeron käsittelyssä olevalle työpisteelle. Jos ko. listalta ei ole valittuna mikään numero, ei myöskään tapahdu mitään. reserveButtonia kuuntelee PhonenumberFrame (ActionListener).

releaseButton vapauttaa postNumbers listalta valitun puhelinnumeron. Jos ko. listalta ei ole valittuna mikään numero, ei myöskään tapahdu mitään. releaseButtonia kuuntelee PhonenumberFrame (ActionListener).

closeButton sulkee ikkunan. closeButtonia kuuntelee PhonenumberFrame (ActionListener).

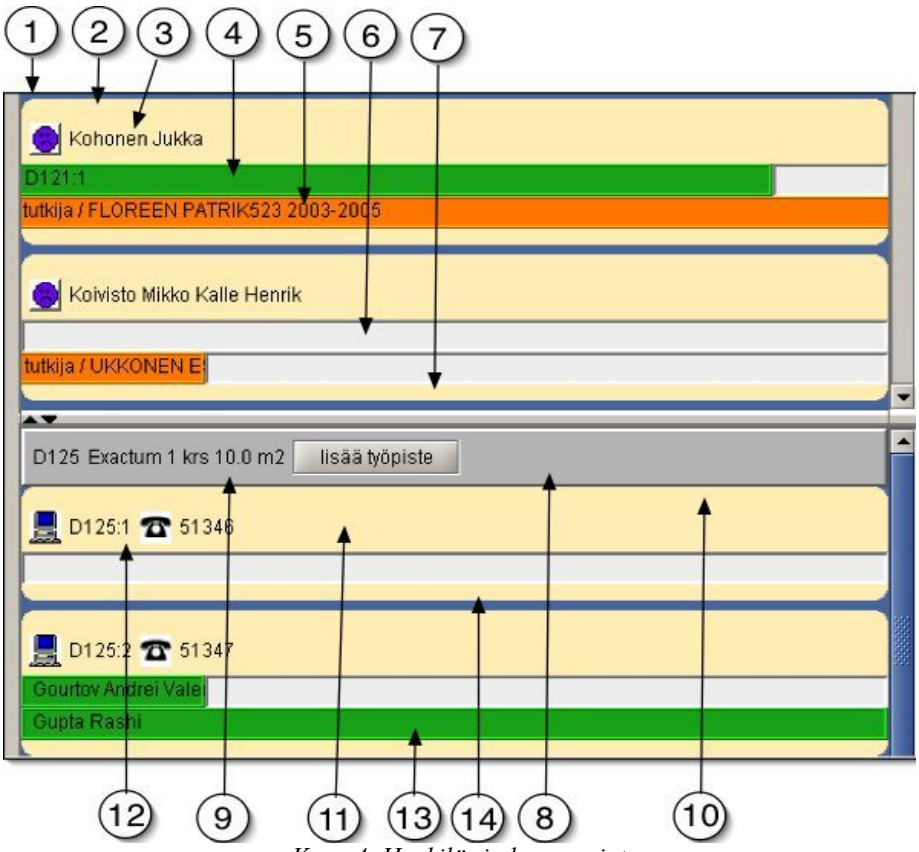


Kuva 3. Puhelinnumeroikkuna

3.3 Henkilö- ja huoneosiot

Luokat PersonScrollPane ja RoomScrollPane huolehtivat halkaistussa ikkunassa näkyvien listojen graafisten esityslistojen generoinnista. Ne käyttävät hyväkseen luokkia PersonsContracts, joka pitää sisällänsä yhden henkilön tiedot ja PostsReservations joka pitää sisällään yhden työpisteen varaustiedot. Molemmat edellä mainitut luokat käyttävät esityksensä rakentamiseen Row luokkaa, joka huolehtii aikajanakomponettien ajan päällekkäisyyden tarksitamisesta ja yhden rivin generoimsiesta. Row luokka käyttää esityksessänsä TimelineElement luokan aliluokkia UIContract, UIReservation, UIEmpty.

Drag&Drop toiminnot sekä hiiren klikkaukset käsittellään luokasssa DragMouseAdapter.



Kuva 4. Henkilö- ja huoneosiot

3.3.1 Henkilöosio

Henkilölistauksen pääluokkana toimii PersonScrollPane-luokka. Se huolehtii listan rakentamisesta ja päivittämisestä, sekä siitä, että esitettävien henkilöiden listaa voi skrollata.

Käytännössä RoomScrollPane-ilmentymä sisältää pääpaneelin (JPanel, kuva 4, kohta 1). Pääpaneeliin on liitetty esitettävien henkilöiden tiedot. Henkilön tiedot koostuvat henkilön pääpaneelista (JPanel). Henkilön pääpaneeli pitää sisällään pyöristetyn yläreunan (JPanel), (kuva 4, kohta 2), otsakerivin (JPanel), jossa on henkilön tilannetta kuvaava kuva (JLabel, kuva 4, kohta 3) ja henkilön nimi (JPanel, kuva 4, kohta 3). Henkilön varausjaksot, sopimusjaksot tai tyhjät aikajaksot liitetään pääpaneliin mukaan TimelineElement-luokan ilmentyminä (kuva 4, kohdat 4, 5 ja 6). Oikeasti tästä luokasta ei luoda ilmentymiä, vaan ilmentymät ovat sen aliluokkien ilmentymiä. Työpistevaraus (kuva 4, kohta 4) on UIReservation-luokan ilmentymä. Työsopimusjakso (kuva 4, kohta 5) on UIContract luokan-ilmentymä ja tyhjä aikajakso (kuva 4, kohta 6) on UIEmpty-luokan ilmentymä. Lopuksi pääpaneelissa on pyöristetty alareuna (JPanel, kuva 4, kohta 7).

3.3.2 Huoneosio

Huoneen tietojen esittämisestä huolehtii RoomScrollPane-luokka. Se huolehtii huoneen tietojen ja työpistelistan rakentamisesta sekä päivittämisestä. RoomScrollPane-luokka mahdollistaa myös skrollauksen.

Käytännössä RoomScrollPane-ilmentymä sisältää JPanel-luokan ilmentymän, johon se liittää mukaan huoneen otsaketietopalkin (JPanel, kuva 4, kohta 8).

Otsaketietopalkissa on huoneen tiedot (JLabel, kuva 4, kohta 9) ja työpisteen lisäämisnappi (JButton, kuva 4, kohta 9).

Työpisteen tiedot esitetään pakettina. Pääkomponentin muodostaa JPanel, johon on liitetty pyöristetty yläreuna(JPanel, kuva 4, kohta 10), työpisteen otsakepalkki (PostLabel).

PostLabel koostuu työpisteen ikonista (JLabel, kuva 4, kohta 11) ja työpisteen tiedoista (JLabel, kuva 4, kohta 11) sekä puhelimen ikonista (PhoneLabel, kuva 4, kohta 11) ja puhelinnumerosta (JLabel, kuva 4, kohta 11). Työpisteen varaukset koostuvat yhden rivin käsittävästä paneelista (RowPanel, kuva 4, kohta 12), joka pitää sisällänsä työpisteeseen liittyvät varaukset (UIReservation, kuva 4, kohta 13). Lopuksi pääpanelissa on pyöristetty alareuna (JPanel, kuva 4, kohta 14).

3.4 Karttaosio

Kartta on toteutettu Batik-kirjastoa käyttäen. Käyttöliittymän karttaosan piirtämiseen käytetään Batikin tarjoamaa JSVGCanvas-komponenttia, joka on tavallinen Swing-komponentti. Karttapohja on tehty SVG-muotoiseksi piirrokseksi (kartta.svg), ja interaktiivisuus on toteutettu Java-koodilla liittämällä SVG-dokumentin elementteihin tapahtumakuuntelijoita ja tekemällä ajonaikaisia muutoksia SVG-dokumentin rakenteeseen käyttäen Batikin tarjoamia DOM Level 2 -rajapinnan metodeita.

SVG-dokumenttiin liittyy CSS-muotoinen tyylimäärittelytiedosto (kartta.css), jossa asetetaan mm. värejä, viivojen paksuuksia ja tekstin ominaisuuksia. Ratkaisun tarkoituksena on mahdollisuuksien mukaan erottaa kartan looginen tietosisältö täsmällisistä tiedonesitystavoista. On siis helppoa esimerkiksi vaihtaa kartassa käytettyjä värejä koskematta SVG-karttapohjaan, Java-koodista puhumattakaan.

SVG-dokumentissa jokainen kerros on ryhmitelty omaksi <g>-elementikseen ja kerrokset sijaitsevat toistensa päällä. Kerrokset

sisältävät huoneita ja käytäviä, jotka on toteutettu yleensä <rect>-elementteinä (suorakaide) mutta toisinaan myös <polygon>-elementteinä. Java-koodi pitää huolen siitä, että kerrallaan vain yksi kerroksista on näkyvissä muiden ollessa piilotettuina (käyttäen display-attribuuttia).

Kerrosten lisäksi karttapohjaan kuuluu kerrostenvaihtonappulat numeroineen sekä merkkien selitykset ja muut ohjeet. Nappulat ovat <rect>-elementtejä. Joitakin kartan sisältämiä elementtejä sekä niiden ajonaikaisia class-attribuutin arvoja on esitetty kuvassa N.

Kuva 5. Karttaosio

Kerrokset, kerroksenvaihtonappulat ja huoneet on yksilöity antamalla kullekin niistä oma id-attribuutti tunnisteeksi. Tunnisteiden muodot on kuvattu taulukossa 1. Huoneiden osalta tunnisteena käytetään huoneen numeroa. Tästä aiheutuu rajoitteita, joista kerrotaan tarkemmin Muutosten tekeminen karttaan -kohdassa.

Tunnisteen muoto	**Käyttötarkoitus**	**Elementtityypit**	**Esimerkki**
floorN	kerros	g	floor1
buttonN	kerrosnappula	rect	button2
XNNN	huone	rect, polygon	C132
legend	merkkien selitteet	g	legend
floors	kerrosten ryhmä	g	floors
buttons	kerrosnappien ryhmä	g	buttons

Taulukko 1. Tunnisteiden muodot

Elementtien esitystapaa säätelevät niille annetut CSS-luokat eli class-attribuutit. Ohjelmakoodissa muutokset karttaan tehdään lähinnä lisäämällä tai poistamalla CSS-luokkia elementtien (huoneet, kerrosnappulat) class-attribuutista DOM-rajapinnan avulla. Käytössä olevat arvot on kuvattu taulukossa 2.

class-arvo	***Käyttötarkoitus***	***Elementtityypit***	***Esitystapa***
button	kerrosnappula	rect	musta ohut reuna
label	huoneen numero	text	kirjasinkoko 13
room	huone	rect, polygon	musta ohut reuna
project	projektin huone, kerrosnappula	rect, polygon	sininen paksu reuna
active	valittu huone, valitun kerroksen nappula	rect, polygon	magenta paksu reuna
filtered	hakuehtojen mukainen huone	rect, polygon	(ei merkitä)
occupied	varattu huone	rect, polygon	punainen tausta
partfree	osittain vapaa huone	rect, polygon	oranssi tausta
free	vapaa huone	rect, polygon	vihreä tausta
noposts	huone ilman työpisteitä	rect, polygon	vaaleanruskea tausta

Taulukko 2. class-arvot

Kartan toteutusta hankaloitti se, että Batik-operaatiot (SVG-elementtien tapahtumakäsittelijät ja muutokset SVG-dokumentin rakenteeseen) on tehtävä Batikin omassa säikeessä, kun taas muu ohjelma pyörii Swingin omassa säikeessä. Tämän vuoksi joudutaan usein hyppäämään säikeestä toiseen eli käytännössä suorittamaan

operaatioita Runnable-ilmentymien avulla. Javadoc-dokumentaatiosta käy ilmi kunkin metodin kohdalla mistä säikeestä sitä on kutsuttava.

3.4.1 Muutosten tekeminen

Kosmeettiset muutokset pohjapiirrokseen

SVG-karttapohjaan ja CSS-tiedostoon voi tehdä muutoksia, kunhan rakenne säilyy samankaltaisena. Kartan toiminta ei ole riippuvainen elementtien sijoittelusta (koordinaateista) joten esim. seiniä voi siirrellä vapaasti. Karttaan voi myös lisätä passiivisia elementtejä kuten selitetekstejä.

Huoneen lisäys

Karttapohjassa kaikki näkyvissä olevat huoneet on varustettu tunnisteella, mutta tietokannassa on vain ne huoneet, jotka ovat syksyllä 2004 laitoksen käytössä. Jos laitoksen käyttöön tulee uusia huoneita, luultavasti riittää lisätä vastaavat rivit tietokannan huonetauluun. Jos kuitenkaan ko. huoneita ei löydy kartalta, karttaan voidaan vapaasti piirtää uusia huoneita, kunhan ne varustetaan asianmukaisilla tunnisteilla. Jos huone on piirretty karttaan ja löytyy tietokannasta, Nero tunnistaa sen.

Kerroksen lisäys

Kartalle on piirretty kolme kerrosta (1, 2 ja 3). Uusia kerroksia voidaan tarvittaessa lisätä. Tällöin on huolehdittava seuraavista asioista:

- tietokannan kerrostauluun lisätään ko. kerrosta kuvaava rivi
- karttaan luodaan uusi kerrosta kuvaava g-elementti, jonka id on floor*N*, ja sen sisälle huoneita sekä tarvittaessa muuta grafiikkaa (käytäviä ym.)
- karttaan lisätään kerrokselle nappula, jonka id on button*N*, ja sille tunnistemerkki (text-elementti)
- Map.java:ssa lisätään kerroksen tunniste FLOORS-vakiokenttään. Huomaa, että FLOORS-merkkijonossa ensimmäisenä esiintyvä kerros on se, joka näytetään oletuksena ohjelman käynnistyessä. Muuten järjestys on vapaa.

Tietokannassa kerrosten tunnisteet ovat kokonaislukuja. Tämän vuoksi esim. K-kerroksen lisääminen voi olla ongelmallista muuttamatta tietokannan rakennetta. Rajoitus on peräisin Rooma-projektin tekemästä tietokannan suunnittelusta. Nero-projektin koodissa kerrosten tunnisteet ovat merkkejä tai merkkijonoja.

Karttakoodissa kerrosten tunnisteet ovat yksittäisiä merkkejä, joten ohjelmisto ei tue esimerkiksi tilannetta, jossa kerrostunnisteet menevät yli 9:n.

Rakennuksen lisäys

Kartta on tehty vain yhtä rakennusta, Exactumia, silmällä pitäen. Huoneiden tunnisteina käytetään niiden numeroa. Tästä voi aiheutua ongelmia, jos järjestelmää laajennetaan useamman rakennuksen kattavaksi, koska tällöin rakennusten huoneden tunnisteissa voi olla päällekkäisyyksiä.

Jos näin ei ole, voidaan rakennus lisätä tietokannan rakennustauluun ja lisätä siihen kerroksia kuten yllä on kuvattu.

Jos päällekkäisyyksiä tulee, täytyy joko huoneiden numerointia muuttaa (esim lisätä jokin etuliite uuden rakennuksen huoneen numeroihin) tai muokata karttakoodia ja SVG-dokumenttia siten, että ne huomioivat huoneen numeron lisäksi myös rakennuksen.

Vastaava tunnisteongelma tulee myös kerrosten käsittelystä. Kartta olettaa kerrosten tunnisteiden olevan sellaisinaan yksikäsitteisiä. Uuden rakennuksen kerroksissa täytyisi käyttää jotakin vaihtoehtoista nimeämistapaa, esim. A, B, C, tai sitten muuttaa kartan ohjelmakoodia kuten huoneiden kohdalla.

4 Dataluokat

Dataluokille esitellään muutokset suunnitteludokumenttiin nähden.

Kaikkiin luokkiin lisättiin toString()-metodit, ja lisäksi luokkiin tuli seuraavat muutokset:

Contract-luokkaan lisättiin kentät henkilön työnimikkeelle (title) sekä hänen viranhoitoprosentilleen (workingPercentage). Molemmille kentille lisättiin myös ne palauttavat metodit getTitle() ja getWorkingPercentage().

Person-luokkaan lisättiin metodit getStatus(), joka kertoo onko henkilöllä tutkittavalla aikavälillä työsopimuksia, joiden koko kestoksi ei myös ole työpistevarausta, ja getStatus()-metodin käyttämä metodi contractBetweenDates(), joka kertoo onko henkilöllä annettujen päivien välillä voimassaolevia työsopimuksia, joiden aikana henkilö ei ole kokonaan virkavapaalla.

PhoneNumber-luokkaan ei tullut muita lisäyksiä.

Post-luokkaan lisättiin postNumber-muuttuja, joka on huoneen järjestysnumero huoneen sisällä, sekä sen palauttava metodi. Lisäksi lisättiin metodit:

- getReservations(TimeSlice), joka palauttaa työpisteen varaukset annetulla aikavälillä
- getRoom, joka palauttaa huoneen johon työpiste kuuluu
- setPhoneNumbers(PhoneNumber[]), joka asettaa työpisteen puhelinnumerot

5 Tietokantaluokka

Tietokantaluokka muuttui suunnitteludokumentin liittenä olevasta Javadoc-luonnoksesta toteutusvaiheen aikana melko paljon. Oleellisimmat muutokset olivat puuttuneiden sisäisten muuttujien lisääminen, konstruktorin parametrien muutokset sekä puhelinnumeroiden käsittelyyn liittyvien metodien lisääminen. Uutta luokan sisäisessä toiminnassa on vieläkin laajempi lähes muuttumattomien tietojen säilyttäminen luokan sisäisissä muuttujissa.

5.1 Uudet sisäiset muuttujat

- private Map people: kaikki järjestelmän tuntemat henkilöt. Ladataan ohjelman käynnistyessä.
- private Map phoneNumbers: järjestelmän tuntemat puhelinnumerot. Ladataan ohjelman käynnistyessä ja päivitetään puhelinnumerotietoja muutettaessa.
- private Map posts: järjestelmän tuntemat työpisteet. Ladataan ohjelman käynnistyessä ja varaustietoja muutettaessa.
- private PreparedStatement prep*: tietokantaan tehtävien kyselyiden esikäännetyt SQL-kyselyt. Luodaan kun kyseinen kysely suoritetaan ensimmäisen kerran.
- private Map prepFilteredPeople: hajautusrakenne henkilöhakuun liittyville PreparedStatement-olioille. Tarpeellinen, koska henkilöhaun monimutkaiset kyselyt muodostetaan useasta eri osasta, joiden kombinaatioita on useita. Kyselyn täydentämätön muoto toimii hajautusrakenteen avaimena.

5.2 Muuttuneet metodit

- public NeroDatabase(Session session, String className, String connectionString, String username, String password): Konstruktori ottaa parametreinaan myös tietokantaluokan nimen, jonka tulisi löytyä classpathista, tietokantayhteyden JDBC-yhteyden määrittelyn sekä tietokannan käyttäjänimen ja salasanan.
- private Connection createConnection(String className, String connectionString, String username, String password): vastaavia muutoksia kuin konstruktoriin, eli tietokantaluokan nimen ja JDBC-yhteyden asetusten lisääminen parametreihin.
- public boolean updateReservation(Reservation reservation): poistettu TimeSlice-parametri, uusi aikaväli on asetettu Reservation-oliolle valmiiksi.

5.3 Poistetut metodit

- public Contract[] getContracts(String personID, TimeSlice timeSlice): Poistettu turhana, käytetään ainoastaan vastaavaa metodia joka saa parametreikseen Person- ja TimeSlice-oliot.
- public Post[] getPosts(Room room): Poistettu turhana. Huoneen Post-oliot on jo valmiiksi asetettu.
- public Reservation[] getReservations(String personID, TimeSlice timeSlice): Poistettu turhana, käytetään ainoastaan vastaavaa metodia joka saa parametreikseen Person- ja TimeSlice oliot.
- public void setConnection(java.sql.Connection connection): Poistettu turhana, tietokantayhteys avataan luokan ilmentymää luodessa.

5.4 Lisätyt metodit

- public PhoneNumber[] getAllPhoneNumbers(): Palauttaa kaikki järjestelmän tuntemat puhelinnumerot järjestettynä PhoneNumber []-taulukkona.
- private boolean filterPerson(Person person, TimeSlice timescale, java.sql.Date contractEndDate, boolean showEndingContracts, boolean withoutPost): Tarkistaa kuuluuko henkilö hakuehtojen mukaiseen listaan. Käytetään getFilteredPeople()-metodin apumetodina.
- public Room getRoom(String roomID): Palauttaa huoneolion huone-id:n perusteella. Etsii huoneen kyseisellä ID:llä sisäisestä tietorakenteesta. Käytetään aktiivista huonetta vaihdettaessa.
- private void loadPhoneNumbers(): Lataa järjestelmän tuntemat puhelinnumerot ja tallettaa ne sisäiseen tietorakenteeseen.
- public void updateObserved(int type): NeroObserver-kuuntelija. Asettaa tarvittaessa henkilötiedot haettaviksi uudelleen kannasta.

5.5 Muita huomioita

Tietokantaluokkaan jäi joitakin huomionarvoisia asioita, jotka on syytä huomioida. Nämä asiat on syytä ottaa huomioon ohjelmiston mahdollisessa jatkokehityksessä.

Päivitettäessä tietoja tietokantaan korvataan päivitykseen liittyvät sisäiset tietorakenteet kokonaan uusilla tiedoilla. Olisi tehokkaampaa päivittää sisäistä tietorakennetta vain muuttuneiden olioiden osalta. Tämä olisi kuutenkin paljon virhealttiimpaa, eivätkä kyseisten tietojen muutokset liity ohjelmiston yleisimpiin käyttötapauksiin.

Tietokantakyselyitä ja niiden parametreja ei ole testattu tarpeeksi kattavasti. Ongelmia saattaa ilmetä lähinnä getFilteredPeople()-

metodissa, joka yrittää ehkä tehdä liian montaa asiaa yhdellä kertaa. Kyseisen metodin suorittamat kyselyt ovat näinollen myös melko raskaita.

“Vain sivutoimiset” -hakuehto olisi mahdollista korvata hakuehdolla joka erittelee sivutoimiset tuntiopettajat (S), päätoimiset tuntiopettajat (P), dosentit (D) sekä laitoshenkilöt (L).

Huoneisiin liittyvissä kyselyissä huoneita ei liitetä eksplisiittisesti oikeaan rakennukseen.

Useat metodit antavat paluuarvoinaan taulukoita. Kävi kuitenkin ilmi, että sekä itse tietokantametodeissa että niiden paluuarvoja käyttävissä metodeissa olisi helpompi käsitellä jotakin Javan Collection-rajapinnan toteuttavaa tietorakennetta, kuten java.util.Vectoria.

Tietokantayhteyden virhetilanteista ei yritetä toipua. Jos tietokantayhteys katkeaa, niin ohjelmisto on käynnistettävä uudelleen.

6 Sovelluslogiikka

6.1 Sessio

Sessio-olio (Session.java) on Nero-sovelluksen ydin ja kuvaa käynnissä olevan ohjelman tilaa. Se säilyttää tiedot käyttöliittymässä näkyvistä hakuehdoista ja huolehtii siitä, että käyttöliittymä saa aina käyttöönsä ajantasaiset tiedot.

Session tehtävät ovat:

- ylläpitää tilatietoja hakuehdoista
- hakuehtojen muuttuessa kertoo tapahtumankuuntelijoille, että tiedot ovat muuttuneet
- toimii käyttöliittymän ja tietokannan välissä tarjoten käyttöliittymälle sekä dataolioille yksinkertaistetun näkymän tietokantaoperaatioihin (hakuihin ja päivityksiin)
- tarjoaa palvelun rekisteröityä kuuntelemaan tapahtumatyyppejä
- mahdollistaa tilatietojen (sanalliset viestit sekä tieto siitä, että jokin operaatio vie aikaa) välityksen käyttöliittymälle

Sessio luodaan heti ohjelman käynnistyksen yhteydessä. Se luo itselleen kuuntelijahallitsijan (NeroObserverManager) ja tarvitsee lisäksi toimiakseen tietokantaolion.

6.2 Tapahtumankuuntelijat ja kuuntelijahallitsija

Sovelluslogiikka on tehty niin, että se ei ota kantaa käyttöliittymän toteutukseen. Ainoa tapa välittää viestejä sessiolta käyttöliittymälle kulkee tapahtumakuuntelijoiden kautta: käyttöliittymäkomponentit rekisteröityvät kuuntelemaan tiettyjä tapahtumatyyppejä, jonka jälkeen sessio vihjaa niille ko. tietojen muutoksesta. Kuuntelu ja päivitys on toteutettu Observer-suunnittelumallin mukaisesti.

Kuuntelijoiden tulee toteuttaa rajapinta NeroObserver, joka sisältää updateObserved -metodin. Kuuntelijahallitsija NeroObserverManager toimii session apulaisena hoitaen kirjanpidon kuuntelijoista. Se ylläpitää listaa kunkin tapahtumatyypin kuuntelijoista ts. olioista, jotka ovat kiinnostuneet tietyntyyppisten tietojen muutoksista.

Tyypin nimi	***Muuttunut tieto***	***Kuuntelijat***
ROOMS	Huonetiedot (esim. työpisteet, puhnrot)	Map, RoomScrollPane, NeroDatabase
ACTIVE_ROOM	Aktiivinen huone	Map, RoomScrollPane
FILTER_PROJECT	Hakuehtojen projekti	Map
FILTER_PEOPLE	Hakuehtojen rajaama ihmisjoukko	PersonScrollPane
FILTER_ROOMS	Hakuehtojen rajaama huonejoukko	Map
TIMESCALE	Aikaväli (koko)	DateSlider, PersonScrollPane, RoomScrollPane, TimeSliceIndicatorPanel
TIMESCALESLICE	Osa-aikaväli	Map, TimeSliceIndicatorPanel
RESERVATIONS	Henkilöihin liittyvät varaustiedot	Map, PersonScrollPane, RoomScrollPane
STATUSBAR	Tilarivin teksti	Statusbar
TIMESCALESLICEUPDATING	Osa-aikaväliä ollaan muuttamassa (ei vielä asetettu)	TimeSliceIndicatorPanel
CURSORCHANGE	Kursorityyppi (tavallinen/odotuskursori)	NeroUI, PhoneNumberFrame

Taulukko 3. Tapahtumatyypit

Tapahtumatyyppien nimet ovat vakioita, joille on annettu kokonaislukuarvot NeroObserverTypes-luokan staattisina vakioina.