Nero
Suunnitteludokumentti

Versio 1.0

Ohjelmistotuotantoprojekti Nero
Helsinki
HELSINGIN YLIOPISTO

Tietojenkäsittelytieteen laitos

Dokumentin versiohistoria

Versio	**Päivämäärä**	**Muutos**	**Tehnyt**
0.1	12.10.2004	Dokumenttipohja luotu. Lukujen otsikkojen ja sisällön hahmottelua.	Teemu Sjöblom

SISÄLLTÖ

1 JOHDANTO **1**
1.1 Dokumentin tarkoitus 1
1.2 Dokumentin rakenne 1
2 SUUNNITTELUN JA TOTEUTUKSEN RAJOITTEET **1**
2.1 Toteutuskielen rajoitteet 1
2.2 Tietokantarajoitteet 2
2.3 Ohjelmointikieli, -tyyli ja -sopimukset 2
2.4 Luokkakirjastot ja ajurit 2
3 JÄRJESTELMÄN YLEISKUVAUS **2**
3.1 Sovellusalueen kuvaus 2
3.2 Kehitys- ja toteutusympäristö 3
3.3 Laiteympäristö 3
3.4 Vaatimusten huomiointi 3
3.5 Käyttöliittymä 3
4 ARKKITEHTUURIN YLEISKUVAUS **3**
4.1 Valitut toteutusratkaisut 4
5 KÄYTTÖLIITTYMÄKOMPONENTIN OSAKOMPONENTIT **4**
5.1 Karttakomponentti 6
5.1.1 Kerroksenvaihtokomponentit 7
5.1.2 Kerrosnäkymäkomponentti 7
5.2 Hakukomponentti 8
5.2.1 Aikajakso 9
5.2.2 Projekti 9
5.2.3 Nimi 9
5.2.4 Työsopimus 10
5.2.5 Huone 10
5.2.6 Työpisteitä 11
5.2.7 Vapaita työpisteitä 11
5.2.8 Ilman työpistettä 11
5.2.9 Sivutoimisuus 12
5.3 Henkilö- ja työhuonekomponentit 12
5.3.1 Halkaistu ikkunakomponentti 13

5.3.2 Henkilölistakomponentti 14
5.3.3 Työpistelistakomponentti 14
5.3.4 Aikajanakomponentti 15
5.3.5 Nimikomponentti 15
5.3.6 Varausjaksokomponentti 15
5.3.7 Sopimuskomponentti 16
5.4 Janakomponentti 16
6 ONGELMAN MALLINTAMINEN 17
6.1 Session 19
6.2 Data-oliot 20
7 TIETOKANNAN MUUTOKSET 20
8 KOPIOINTIOSAJÄRJESTELMÄ 20
9 LÄHDEKOODIN PAKKAUKSET 22
10 KÄYTTÖÖNOTTOSUUNNITELMA 22

1 Johdanto

Nero on Helsingin yliopiston tietojenkäsittelytieteen laitoksen ohjelmistotuotantoprojekti (581260-4) kurssin mukainen ohjelmistotuotantoprojekti. Projektiin liittyvä materiaali on saatavissa ryhmän kotisivulta osoitteesta

http://www.cs.helsinki.fi/group/nero/

Nero-projektin tarkoitus on jatkaa Rooma-ryhmän kesällä tekemän järjestelmän kehittelyä ja parannella heidän aikaansaannostaan. Rooma-ryhmän alkuperäinen tavoite lyhyesti: Luoda järjestelmä, jolla Reijo Siven ja Juhani Haavisto voivat seurata laitoksen työhuoneiden käyttöastetta sekä helpottaa työhuonetilanteen hallintaa uusien työntekijöiden saapuessa sekä vanhojen lähtiessä tai vaihtaessa työhuonetta.

1.1 Dokumentin tarkoitus

Tämä dokumentti on tarkoitettu projektiryhmälle sisäiseksi ohjeeksi siitä, kuinka määrittelydokumentissa kuvattu järjestelmä tulee toteuttaa. Dokumentissa kuvataan projektin aikana tuotettava ohjelmisto sellaisella tarkkuudella, että sen suoraviivainen toteuttaminen on mahdollista yksinomaan tämän dokumentin pohjalta. Dokumentti on kirjoitettu määrittelydokumentin version 0.8 pohjalta.

1.2 Dokumentin rakenne

Luvussa 2 käydään läpi rajoitteita, joita projektin suunnittelulla ja toteutuksella on. Luku 3 kuvailee järjestelmän yleisellä tasolla sekä järjestelmän toteutus-, ja käyttöympäristöt laitteiden ja ohjelmistojen suhteen sekä järjestelmän liittymät näihin. Luvussa 4 kuvataan järjestelmän arkkitehtuuri yleisellä tasolla, luvussa 5 kuvataan järjestelmän käyttöliittymän osakomponentit. Luvussa 6 kuvataan ratkaistavan ongelman vaatimat ajonaikaiset tietorakenteet. Luvussa 7 kuvataan Rooma-järjestelmän tietokantaan tarvittavat muutokset. Luvussa 8 kuvataan kopiointiosajärjestelmään tarvittavat muutokset. Luvussa 9 kuvataan ohjelman pakkausrakenne. Luku 10 sisältää käyttöönottosuunnitelman.

2 Suunnittelun ja toteutuksen rajoitteet

Nero on aikaisemman ohjelmistotuotantoprojektin jatkoprojekti. Alkuperäisenä tavoitteena oli käyttää aikaisemman projektin tuotoksia mahdollisimman paljon hyväksi. Tarkemman suunnittelun ja tutkimisen pohjalta Rooman tuotokset todettiin projektiin sopimattomiksi kaikilta muilta osilta, paitsi tietokanta-arkkitehtuurin ja kopiointiosajärjestelmän osalta. Koska kuitenkin Nero-projekti käyttää valmista tietokantaa ja kopiointiosajärjestelmää, on käsitteellinen mallinnustapa sekä käytettävissä olevan datan muoto ennalta määrätty, joten Nero-projektissa ei voida toteuttaa ominaisuuksia, jotka vaatisivat suuria muutoksia tietokantaan.

2.1 Toteutuskielen rajoitteet

Toteutuksessa käytetään Java- ja PL/SQL-kieliä. PL/SQL on Rooma-projektin valitsema kieli henkilötietojen kopioimiseen henkilötietojärjestelmästä omaan

tietokantaansa. Koska Rooma-projektin tietokantaa käytetään lähes sellaisenaan, ei tähän kopiointikomponenttiin kosketa. Mahdolliset muutokset Rooman tietokannan rakenteeseen tehdään nykyisen rakenteen kanssa yhteensopivalla tavalla.

Java-kieli valittiin ohjelman toteutuskieleksi, koska se on ryhmän jäsenille entuudestaan tuttu ja Java-osaaminen oli parhaiten edustettuna ryhmässä.

2.2 Tietokantarajoitteet

Järjestelmä tulee käyttämään Rooma-projektissa toteutettua tietokantaa ja kopiointiosajärjestelmää. Tietokanta on toteutettu Oracle8 Enterprise Edition Releasen versio 8.0.5.0.0–tietokantaohjelmistolle. Ohjelmisto sijaitsee Helsingin yliopiston Kontti-palvelimella.

Kopiointiosajärjestelmä on itsenäinen osajärjestelmä, jolla kopioidaan yliopiston keskushallinnon tietovarastosta projektien tiedot sekä työntekijöiden henkilö- ja sopimustiedot henkilötietokantaan. Tietovarasto on Oracle-tietokanta, johon ladataan yliopiston eri osajärjestelmistä tietoa. Latauksia lukuunottamatta tietovarastoon ei tehdä muutoksia.

2.3 Ohjelmointikieli, -tyyli ja -sopimukset

Käytettävä ohjelmointikieli on Java. Käytettävä versio on 1.4.2. Ohjelmoinnissa tullaan pyrkimään hyvään ohjelmointitapaan; koodista tehdään luettavaa ja hyvin kommentoitua noudattamalla vakiintunutta Java-koodaustyyliä, käyttämällä sovelluskehittimen tarjoamia sisennys- ym. työkaluja ja siistimällä koodia aina, jos se ei näytä riittävän selkeältä. Tuotetun koodin uudelleenkäytettävyyteen ei erityisesti kiinnitetä huomiota tekemällä palveluita geneerisemmäksi tai laajemmin kuin ohjelman omat tarpeet vaativat. Javakoodissa käytetään kielenä englantia. Javadoc-dokumentaatio kirjoitetaan suomeksi.

2.4 Luokkakirjastot ja ajurit

Javan omien kirjastojen lisäksi käytetään käyttöliittymän karttakomponentissa Batik-luokkakirjastoa (http://xml.apache.org/batik/). Pohjapiirroksen esittämiseen käytetään SVG (scalable vector graphics) –formaattia (http://www.w3.org/TR/SVG/). Batik on julkaistu vapaan Apache Public License 1.1-lisenssin alla. JDBC-ajurina käytetään Oraclen ojdbc14.jar-pakettia joka on yhteensopiva JDK-versio 1.4:n kanssa. GPL ei kuitenkaan ole yhteensopiva APL1.1:n eikä Oraclen JDBC-ajurin lisenssin kanssa, joten jos ohjelmistoa halutaan levittää, sen lisenssiksi tulee valita joko sallivampi LGPL tai lisäpykälällä varustettu GPL, joka erikseen sallii em. kirjastojen käytön.

3 Järjestelmän yleiskuvaus

Tässä kappaleessa kuvataan toteutettava järjestelmä ja ympäristöt yleisellä tasolla.

3.1 Sovellusalueen kuvaus

Projektissa tuotetaan sovellus kiinteistön työhuoneiden varausten hallintaan. Järjestelmän avulla voi pitää kirjaa Exactumin tilojen varaustilanteesta, sekä tehdä,

poistaa ja muokata varauksia. Käytännössä tämä tarkoittaa työhuoneissa olevien työpisteiden sekä henkilöillä niihin olevien varausten lisäämistä ja poistamista. Järjestelmä tarjoaa käyttäjälle yleiskuvan varaustilanteesta kuvana, jonka pohjana on Exactumin tarkasteltavan kerroksen pohjapiirros. Lisäksi käyttäjä voi hakea laitoksen työntekijöitä muuttamalla hakuehtoja. Haun tulokset esitetään listana, jossa henkilöiden työsopimukset ja niihin liittyvät työpistevaraukset on esitetty graafisesti. Kartasta voi valita huoneen, josta esitetään sen työpisteiden tiedot niinikään listana, jossa työpisteisiin sijoitettujen henkilöiden varaukset on kuvattu graafisesti.

3.2 Kehitys- ja toteutusympäristö

Ohjelmisto toteutetaan Tietojenkäsittelytieteen laitoksen Java-ympäristöä vastaavassa ympäristössä Java SDK:n versiota 1.4.2 käyttäen. Tietokantapalvelinohjelmistona käytetään Oracle8 Enterprise Edition Release tuotetta. Tietokantaohjelmiston tuotespesifisiä välineitä ei käytetä, joten sovellus voidaan asentaa mille tahansa Java 1.4.2 ja Oracle - alustalle.

3.3 Laiteympäristö

Nero tulee toimimaan TKTL:n laiteympäristössä joka käsittää Windows- ja Linux-koneita. Ohjelma toimii edellä mainituilla koneilla olettaen, että niihin on asennettu Java Runtime Environment (JRE) 1.4.2 ja Batik-luokkakirjasto.

Tietokannat toimivat ATK-keskuksen palvelimilla ja niihin otetaan yhteys yliopiston sisäverkon kautta.

3.4 Vaatimusten huomiointi

Vaatimusmäärittelyssä dokumentoiduista vaatimuksista toteutetaan vaatimukset V1-V24. Vaatimukset V25-V29 jätetään toteuttamatta.

3.5 Käyttöliittymä

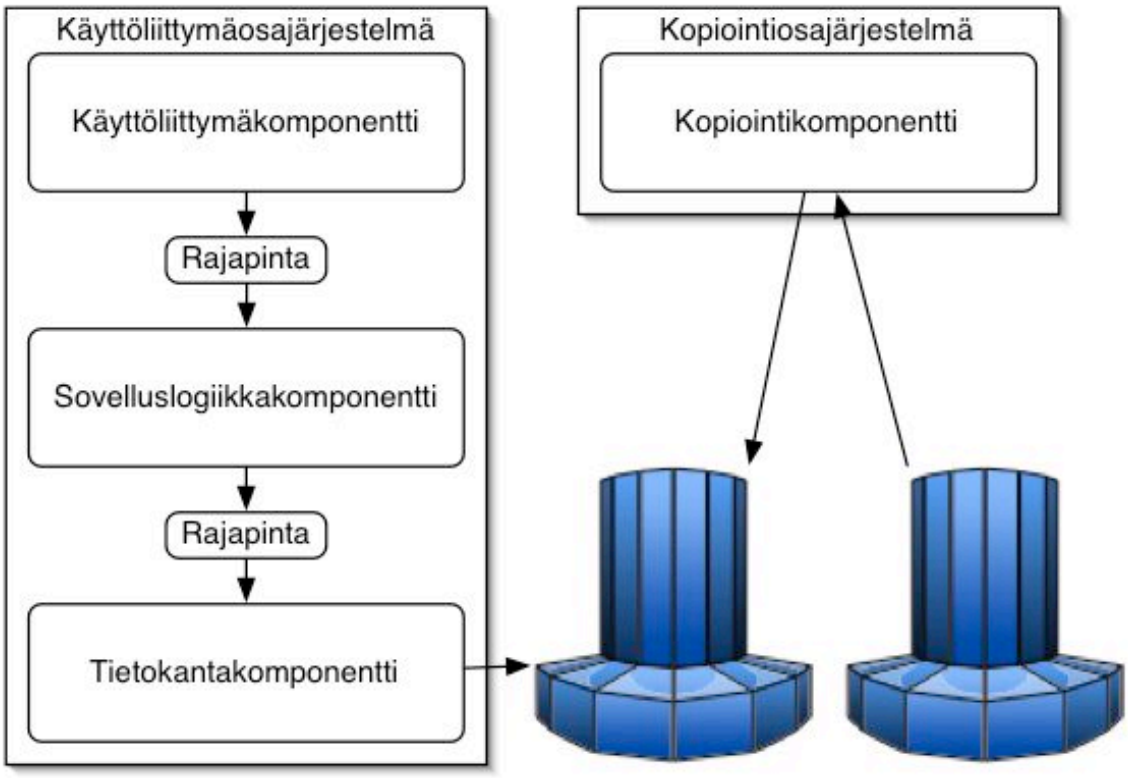
Käyttöliittymän ulkoasu toteutetaan vaatimusdokumentissa kuvassa kaksi määritellyssä muodossa.

4 Arkkitehtuurin yleiskuvaus

Toteutettavan järjestelmän yleisrakenne käy ilmi kuvasta 1. Järjestelmä jaetaan kahteen eri osajärjestelmään. Käyttöliittymäosajärjestelmään ja kopiointiosajärjestelmään.

Käyttöliittymäosajärjestelmä jakautuu kolmeen komponenttiin. Ylin komponentti on käyttöliittymäkomponentti. Käyttöliittymän tehtävänä on esittää sen hetkinen tila käyttäjälle ja reagoida käyttäjän toimenpiteisiin. Käyttöliittymän reagoidessa käyttäjän toimenpiteeseen se kutsuu tarvittaessa rajapinnan kautta sovelluslogiikkakomponentissa sijaitsevaa sovelluslogiikkaa. Sovelluslogiikka kutsuu tarvittaessa tietokantakomponenttia joka vuorostaan hoitaa interaktion tietokannan kanssa.

Kopiointiosajärjestelmä toteutetaan erillisenä järjestelmänä, jossa on vain kopiointikomponentti.



Kuva 1 Osajärjestelmät komponentteineen.

4.1 Valitut toteutusratkaisut

Käyttöliittymäkomponentti

Jotta käyttöliittymästä saataisiin joustava, monipuolinen ja havainnollinen se ohjelmoidaan kokonaan uusiksi Java-sovelluksena. Apuna käytetään Javan valmiita kirjastoja sekä Batik-kirjastoja.

Sovelluskomponentti

Rooma ryhmän sovelluskomponenttia ei voida käyttää hyväksi, johtuen sen erilaisesta tavasta hahmottaa ratkaistava ongelma. Konvertterikomponentin ohjelmoimisen sijaan ohjelmoidaan uusi sovelluslogiikkakomponentti.

Tietokantakomponentti

Rooma ryhmän tietokantakomponenttia ei voida käyttää hyväksi, koska sen sisältämät kyselyt eivät anna tietoa tarvittavassa muodossa. Konvertterikomponentin ohjelmoimisen sijaan ohjelmoidaan uusi tietokantakomponentti.

Kopiointikomponentti

Rooma-ryhmä toteutti kopiointikomponentin PL/SQL-kielellä. Komponenttia pyritään käyttämään sellaisenaan ja jos siihen tarvitaan muutoksia ne tehdään PL/SQL-kielellä.

5 Käyttöliittymäkomponentin osakomponentit

Käyttöliittymäkomponentti jaetaan viiteen pääkomponenttiin, nämä viisi pääkomponenttia, sekä pääkomponenttien keskinäiset vaikutussuhteet käyvät ilmi

kuvasta 2. Jokainen pääkomponentti voi vuorostaan jakautua useaan osakomponenttiin. Tämän kappaleen alikappaleissa kuvataan pääkomponenttien ja sen osakomponenttien rakenne tarkemmin. Komponentit esitetään seuraavanlaisella tavalla:

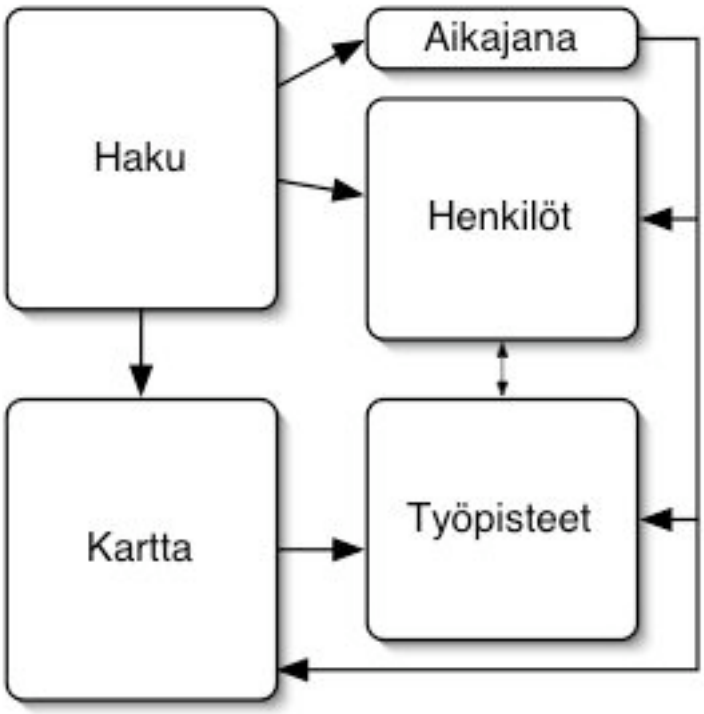
Komponentin nimi

Aluksi komponentista on lyhyt kuvaus.

Syötteet
Tässä kuvataan komponentin syötteet. Syötteitä voi olla toisten komponenttien aiheuttamat metodikutsut tai käyttäjän käyttöliittymässä tekemät toimenpiteet.

Komponentti tarvitsee/käyttää
Tässä kerrotaan se, mitä muiden komponenttien palveluita komponentti käyttää tai tarvitsee.

Komponentti toteutetaan
Tässä annetaan lyhyt kuvaus siitä, millä tavalla komponenttia lähdetään toteuttamaan. Esimerkiksi kerrotaan käytettävät luokat ja rajapinnat.



Kuva 2: Käyttöliittymäkomponentin pääkomponentit ja keskinäiset vaikutussuhteet.

5.1 Karttakomponentti

Karttakomponentti esittää rakennuksen yhden kerroksen huoneiden varaustilanteen visuaalisesti annetulla aikavälillä ja mahdollistaa esitettävän kerroksen vaihtamisen. Karttakomponentista voi valita klikkaamalla huoneen tarkempaa tarkastelua varten.

Karttakomponentti osakomponentteineen selviää kuvasta 3.

Syötteet

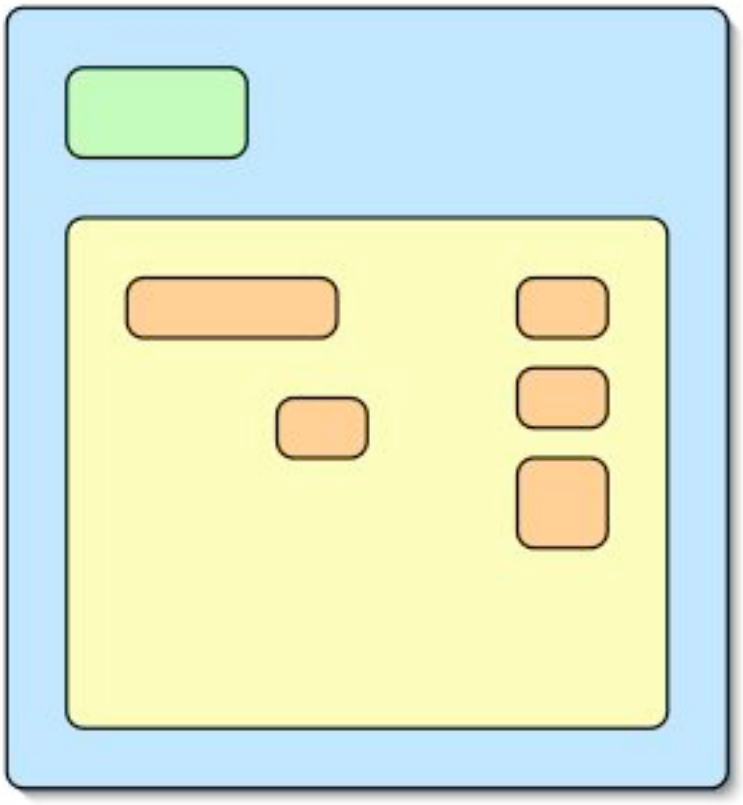
Komponentti saa syötteenä tapahtumat ROOMS (huonetiedot muuttuneet), FILTER_ROOMS (hakuehtojen mukainen huonelista muuttunut) ja ACTIVE_ROOM (aktiivinen huone vaihtunut).

Komponentti tarvitsee/käyttää

Tarvitsee sovelluslogiikalta palvelun, jolla rekisteröityä tapahtumakuuntelijaksi. Välittää tiedon tapahtumista edelleen kerrosnäkymäkomponentille.

Komponentti toteutetaan

Komponentti toteutetaan Batikin *JSVGCanvas*-luokan avulla. Komponentti toteuttaa *NeroObserver*-rajapinnan tapahtumien kuunteluun.



Kuva 3 Karttakomponentti osakomponentteineen.

5.1.1 Kerroksenvaihtokomponentit

Kerroksen vaihtokomponentti esittää rakennuksen yhden kerroksien tunnisteen visuaalisesti. Vaihtokomponentteja on useita, yksi kullekin kerrokselle. Kuvassa kerrosnäkymäkomponentit ovat vihreässä laatikossa. Käyttäjän valitsemaa kerrosta kuvaava tunniste muuttuu siten, että siitä käy ilmi ko. kerroksen olevan valittuna.

Syötteet

Komponentti saa syötteenä käyttäjän klikkauksen käyttäjän ensin painaessa hiiren napin pohjaan komponentin päällä ja sitten päästäessään napin takaisin ylös.

Komponentti tarvitsee

Komponentti tarvitsee kerrosnäkymäkomponentilta palvelun, jolla vaihdetaan näkyvissä olevaa kerrosta.

Komponentti toteutetaan

Komponentti toteutetaan Batikilla luotuina SVG-dokumentin elementteinä (rajapinnan *Element* toteuttava luokka, jonka ilmentymän Batik luo *createElement*-metodilla). Komponentti tarjoaa metodit, joilla kerrotaan tieto, että kerroksessa on/ei ole vapaata ja että kerroksessa on/ei ole valitun projektin huoneita.

5.1.2 Kerrosnäkymäkomponentti

Kerrosnäkymäkomponentti esittää kerroksen pohjapiirroksen, missä on uusimman tilanteen mukaiset asiat korostettuina. Pohjapiirroksesta korostetaan seuraavat asiat.

-Valitun projektin huoneet (esitetään reunuksen värin ja/tai paksuuden muutoksena)

-Hakuehtojen mukaisista huoneista vapausaste (esitetään pohjavärin muutoksena)

-Aktiivinen huone (esitetään reunuksen värin ja/tai paksuuden muutoksena)

Pohjapiiroksesta voi valita huoneen, jonka tiedot näytetään työpisteet-komponentissa. Kuvassa 3 kerrosnäkymäkomponentti on vaaleankeltainen ja sen sisältämät huoneet hieman tummempia kellanruskeita laatikoita.

Syötteet

Komponentti saa syötteenä käyttäjän klikkauksen käyttäjän ensin painaessa hiiren napin pohjaan komponentin esittämän kerroksen yhden huoneen päällä ja sitten päästäessään napin takaisin ylös.

Komponentti tarvitsee/käyttää

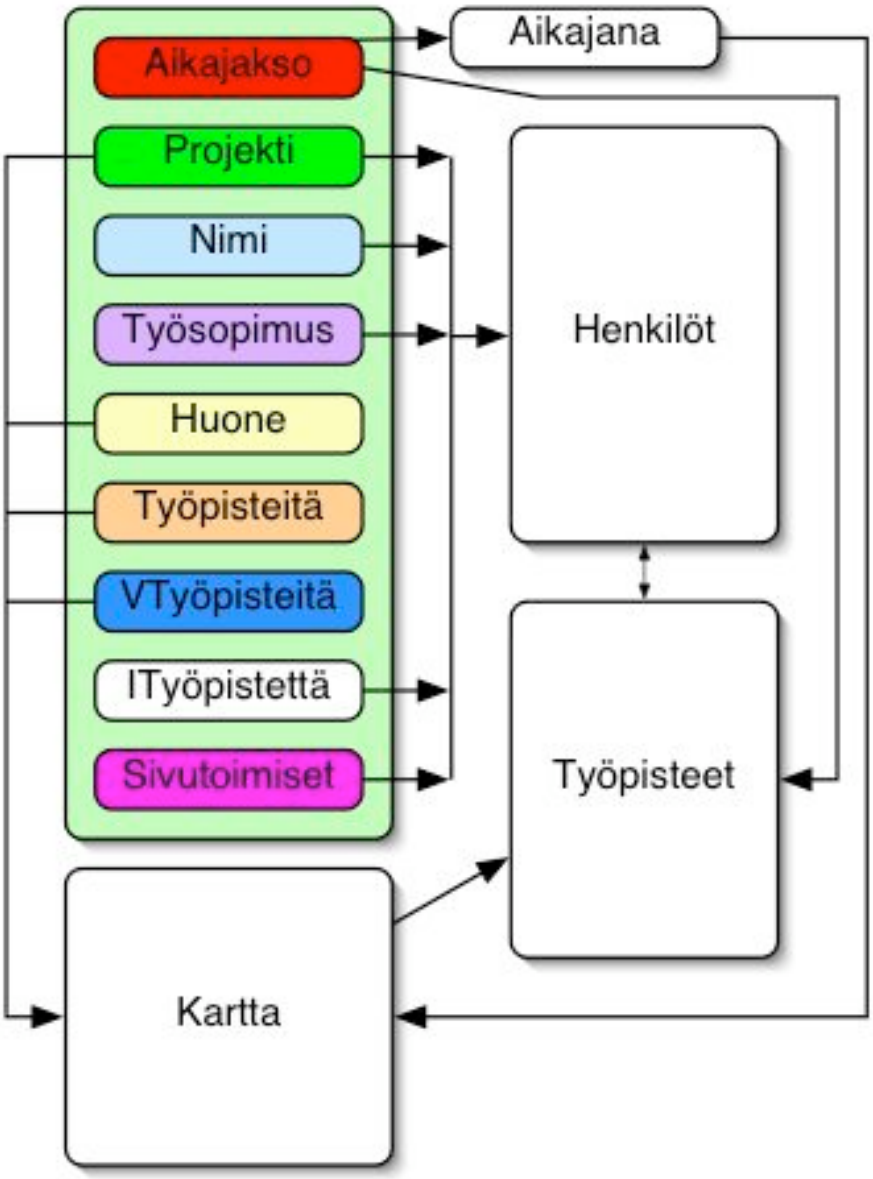
Komponetti hankkii listan kaikista huoneista sovelluslogiikan metodilla *getRooms()*. Komponetti hankkii listan hakuehtojen rajaamista huoneista sovelluslogiikan metodilla *getFilteredRooms()*. Komponetti hakkii listan projektin huoneista sovelluslogiikan metodilla *getProjectRooms()*. Komponentti tarvitsee palvelun, jolla vaihdetaan aktiivista huonetta. Komponentti tarvitsee kerroksenvaihtokomponenteilta palvelun, jolla asettaa kunkin kerroksen osalta onko ko. kerroksessa vapaita työpisteitä ja onko siellä valitun projektin huoneita.

Komponentti toteutetaan

Komponentti toteutetaan Batikin *SVGDocument*-luokan avulla. Komponentti tarjoaa metodin, jolla vaihdetaan näkyvissä olevaa kerrosta. Komponentti toteuttaa *NeroObserver*-rajapinnan, jota karttakomponentti käyttää tapahtumien raportointiin.

5.2 Hakukomponentti

Hakukomponentti sisältää komponentteja joiden avulla käyttäjä voi rajata näytettävää tietoa. Hakukomponentin rakenne käy ilmi kuvasta 4. Hakukomponentti toteutetaan *JPanel*-luokan avulla. Hakukomponentin sisällä on muita komponentteja, jotka kuvataan luvun aliluvuissa. Komponentit järjestetään hakukomponentin sisällä *BoxLayout*-asettelumanagerin avulla.



Kuva 4: Hakukomponentin rakenne vaikutussuhteineen.

5.2.1 Aikajakso

Aikajaksokomponentin avulla käyttäjä voi valita hakuja rajaavan aikajakson. Ohjelman käynnistyessä käytetään oletusarvoja, jotka ovat kuluva päivä ja siitä kuukauden päässä oleva päivämäärä. Oletusarvojen määritys tehdään sovelluslogiikassa. Aikajakso vaikuttaa kaikkiin hakuihin.

Syötteet

Komponentti saa syötteenä käyttäjän kirjoittamat kaksi päivämäärää, jotka ovat muotoa dd.mm.yy.

Komponentti toteutetaan

Komponentti toteutetaan *JPanel*-luokan avulla, jonka sisällä on *JTextField*- tai *JFormattedTextField*-luokan ilmentymiä. Lisäksi *DocumentListener*-rajapintaa käytetään.

Komponentti tarvitse/käyttää

Komponentti käyttää sovelluslogiikan *setFilterTimescale()*-metodia, jonka avulla komponentti muuttaa ohjelmassa tarkasteltavaa aikaväliä.

5.2.2 Projekti

Projektikomponentin avulla käyttäjä voi liittää hakuehtoihin määreen, jonka mukaan esitettävien henkilöiden tai huoneiden on kuuluttava valittuun projektiin. Valittavissa olevat projektit esitetään alasvetovalikkona. Komponentti säilyttää tehdyn valinnan. Ohjelman käynnistyessä yksikään projekti ei ole valittuna. Vaikuttaa näytettäviin henkilöihin. Ei vaikuta näytettäviin huoneisiin, vaan projektin huoneet näkyvät kartalla haettujen huoneiden lisäksi.

Syötteet

Komponentti saa syötteenä muutoksen valinnassa.

Komponentti toteutetaan

Komponentti toteutetaan *JComboBox*-luokan ja *ActionListener*-rajapinnan avulla.

Komponentti tarvitsee/käyttää

Komponentti käyttää sovelluslogiikan metodia *getProjects()*-saadakseen listan näytettävistä projekteista. Sovelluslogiikan *setFilterProject()*- metodilla komponentti asettaa projektin rajausehdoksi.

5.2.3 Nimi

Nimikomponentin avulla käyttäjä voi liittää hakuehtoihin määreen, jonka mukaan esitettävien henkilöiden etunimien tai sukunimien tulee sisältää tietty merkkijono. Komponentti säilyttää käyttäjän kirjoittaman nimen. Ohjelman käynnistyessä kenttä on tyhjä. Vaikuttaa näytettäviin henkilöihin.

Syötteet

Komponentti saa syötteenä merkkijonon, joka liitetään hakuehtoihin rajaaman näytettäviä henkilöitä. Syötteen on oltava vähintään kolme kirjainta pitkä, jotta se vaikuttaisi rajaavasti. Tätä lyhyempi syöte ei vielä rajaa henkilöitä.

Komponentti toteutetaan

Komponentti toteutetaan *JTextField*-luokan ja *DocumentListener*-rajapinnan avulla.

Komponentti tarvitsee/käyttää

Komponentti käyttää sovelluslogiikan *setFilterPersonName()*- metodia, sen avulla komponentti asettaa hakuja rajaavan henkilön nimen osamerkkijonon.

5.2.4 Työsopimus

Komponentin avulla haku rajoitetaan niihin henkilöihin, joilla on käsiteltävällä aikavälillä jokin työsopimus päättymässä. Aikaväli on säädetty aikajaksokomponentin kautta. Ohjelman käynnistyessä ehto on pois päältä. Vaikuttaa näytettäviin henkilöihin.

Syötteet

Komponentti saa syötteenä käyttäjän klikkauksen, joka vaihtaa hakuehdon tilan päinvastaiseksi.

Komponentti toteutetaan

JCheckBox-luokan ja *ActionListener*-rajapinnan avulla.

Komponentti tarvitsee/käyttää

Komponentti käyttää sovelluslogiikan *setFilterEndingContracts()*-metodia. Metodin avulla komponentti asettaa määreen, jonka avulla haut rajataan vain loppuviin työsopimuksiin.

5.2.5 Huone

Huonekomponentin avulla käyttäjä voi liittää hakuehtoihin määreen, jonka mukaan rajataan esitettäviä huoneita niiden nimen perusteella. Syöte voi olla kokonainen huoneen nimi tai huoneen nimen vähintään 2 merkkiä pitkä alkuosa, esim. D2. Ohjelman käynnistyessä kenttä on tyhjä. Vaikuttaa näytettäviin huoneisiin.

Syötteet

Komponentti saa syötteenä hakuehtoja rajaavan merkkijonon.

Komponentti toteutetaan

Komponentti toteutetaan *JTextField*- tai *JFormattedTextField*-luokan sekä *DocumentListener*-rajapinnan avulla.

Komponentti tarvitsee/käyttää

Komponentti käyttää sovelluslogiikan *setFilterRoomName()*- metodia. Metodin avulla komponentti asettaa hakuja rajaavan huoneen nimen.

5.2.6 Työpisteitä

Komponentin avulla haku voidaan rajoittaa koskemaan huoneita, joissa on maksimissaan haluttu määrä työpisteitä. Komponentti säilyttää syötetyn arvon. Ohjelman käynnistyessä kenttä on tyhjä. Vaikuttaa näytettäviin huoneisiin.

Syötteet

Komponentti saa syötteenä luvun, joka liitetään hakuehtoihin kuvaamaan kuinka monta työpistettä haetuissa huoneissa saa korkeintaan olla.

Komponentti toteutetaan

Komponentti toteutetaan *JTextField*- tai *JFormattedTextField*-luokan sekä *DocumentListener*-rajapinnan avulla.

Komponentti tarvitsee/käyttää

Komponentti käyttää sovelluslogiikan *setFilterMaxPosts()*- metodia. Metodin avulla komponentti asettaa hakuja rajaavan työpisteiden lukumäärän.

5.2.7 Vapaita työpisteitä

Komponentin avulla haku voidaan ohjata etsimään huoneita, joissa on useampia vapaita työpisteitä halutulla aikavälillä. Komponentti säilyttää syötetyn arvon. Ohjelman käynnistyessä kenttä on tyhjä. Vaikuttaa kartalla näkyvien huoneiden merkintätapaan siten, että huoneen vapaus määritellään tiukemmin: vain X vapaata työpistettä sisältävä huone esitetään vapaana. Ei kuitenkaan vaikuta kartalla esitettävien huoneiden joukkoon.

Syötteet

Komponentti saa syötteenä luvun, joka liitetään hakuehtoihin kuvaamaan kuinka monta vapaa työpistettä haetuissa huoneissa tulisi vähintään olla.

Komponentti toteutetaan

Komponentti toteutetaan *JTextField*- tai *JFormattedTextField*-luokan ja *DocumentListener*-rajapinnan avulla.

Komponentti tarvitsee/käyttää

Komponentti käyttää sovelluslogiikan *setFilterFreePosts()*-metodia vapaiden työpisteiden vähimmäismäärän asettamiseen.

5.2.8 Ilman työpistettä

Komponentin avulla haku rajoitetaan ainoastaan niihin henkilöihin, joilla on halutulla aikavälillä osa työsopimusjaksosta tai kokonainen työsopimusjakso vailla työpistevarausta. Ohjelman käynnistyessä ehto on päällä. Vaikuttaa näytettäviin henkilöihin.

Syötteet

Komponentti saa syötteenä käyttäjän klikkauksen, joka vaihtaa hakuehdon tilan päinvastaiseksi.

Komponentti toteutetaan
JCheckBox-luokan ja *ActionListener*-rajapinnan avulla.

Komponentti tarvitsee/käyttää
Komponentti käyttää sovelluslogiikan *setFilterWithoutPost()*-metodia. Metodin avulla komponentti asettaa määreen, jonka avulla haut rajataan vain sellaisiin henkilöihin, joilla ei ole työpistevarausta jonkin työsopimuksen koko ajalle.

5.2.9 Sivutoimisuus

Komponentin avulla haku rajoitetaan ainoastaan niihin henkilöihin ja henkilöiden huoneisiin, joilla on halutulla aikavälillä jonakin työsopimuksena sivutoimisen tuntiopettajan työsopimus. Ohjelman käynnistyessä ehto ei ole päällä. Vaikuttaa näytettäviin henkilöihin.

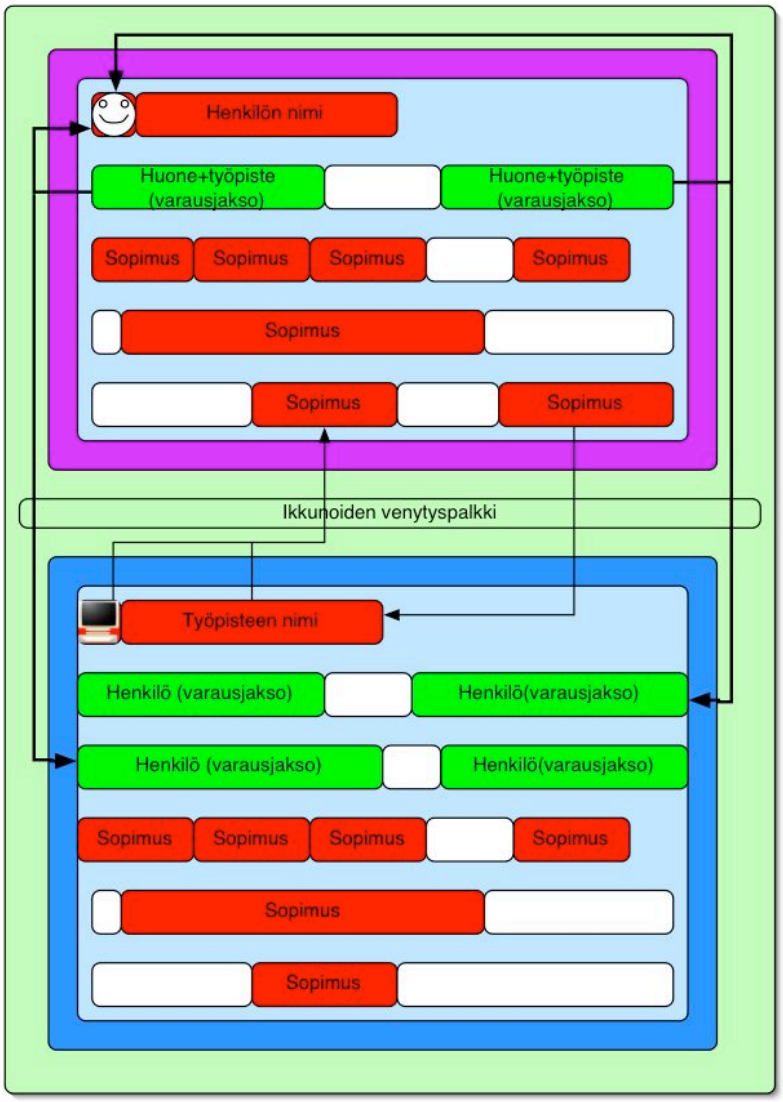
Syötteet
Komponentti saa syötteenä käyttäjän klikkauksen komponentin päällä, joka vaihtaa hakuehdon tilan päinvastaiseksi.

Komponentti toteutetaan
JCheckBox-luokan ja *ActionListener*-rajapinnan avulla.

Komponentti tarvitsee/käyttää
Komponentti käyttää sovelluslogiikan *setFilterPartTimeTeachers ()*-metodia. Metodin avulla komponentti asettaa määreen, jonka avulla haut rajataan vain sellaisiin henkilöihin joilla on sivutoimisen tuntiopettajan työsopimus ja työhuoneisiin, joissa kyseisillä henkilöillä on työpistevarauksia.

5.3 Henkilö- ja työhuone-komponentit

Henkilö- ja työhuonekomponentit ovat hyvin samankaltaiset. Molempien tehtävänä on visualisoida työpistevarauksia ja työsopimusjaksoja ajan suhteen. Henkilökomponentti näyttää hakukomponentin hakukriteereihin sopivat henkilöt. Työhuonekomponentti näyttää kartalta valitun työhuoneen työpisteet sekä niihin kohdistuvat varausjaksot. Henkilö ja työhuonekomponentit koostuvat kuvan 5 mukaisista osakomponenteista. Osakomponentit kuvataan luvun aliluvuissa. Komponentti skaalaa sopimuksia ja työpistevarauksia kuvaavat palkit riippuen tarkasteltavasta ajanjaksosta. Esimerkiksi tarkasteltaessa 60 päivän mittaista jaksoa ja komponentin leveyden ollessa 400 pikseliä saadaan 30 päivän mittaisen jakson leveydeksi ruudulla 200 pikseliä.



Kuva 5: Henkilö- ja työhuone-komponenttien rakenne.

5.3.1 Halkaistuikkuna-komponentti

Halkaistuikkuna-komponentti koostuu kahdesta näyttöalueesta joiden välissä on vetopalkki. Palkkia vetämällä ikkunoiden koon suhdetta komponentin sisällä voi muuttaa. Ylemmässä alueessa sijaitsee henkilölistakomponentti. Alemmassa alueessa sijaitsee työpistelistakomponentti.

Syötteet

Käyttäjä voi valita vetopalkin painamalla sen päällä hiiren napin pohjaan. Valinnan jälkeen käyttäjä voi hiirtä liikuttamalla vetää palkkia joko ylös- tai alaspäin, jolloin toisen ikkunan näyttöala pienenee ja toisen kasvaa. Hiiren napin vapauttaminen lukitsee palkin sen hetkiseen kohtaan.

Komponentti tarvitsee/käyttää

Komponentti ei tarvitse ulkoisia palveluita.

Komponentti toteutetaan

Komponentti toteutetaan Javan *JSplitPane*-luokan avulla.

5.3.2 Henkilölista-komponentti

Komponentti esittää hakuehtoihin sopivat henkilöt työsopimuksineen ja työpistevarauksineen listana, jossa henkilöt ovat allekkain. Jos listan henkilöt eivät mahdu niille varattuun tilaan komponentti tuo esille vetopalkit, josta listaa voi selata edestakaisin.

Syötteet

Komponentti saa syötteenä tiedon näyttämiensä tietojen mahdollisesta muutoksesta.

Komponentti tarvitse/käyttää

Komponentti käyttää sovelluslogiikan *getPeople()*-metodia, jonka avulla se saa esitettävät ihmiset. Komponentti luo jokaista ihmistä vastaavan aikajanakomponentin. Komponentti rekisteröityy kuuntelemaan esittämissään tiedoissa tapahtuneita muutoksia sovelluslogiikan *registerObserver()*-metodin avulla.

Komponentti toteutetaan

Komponentti toteutetaan *JScrollPane*-luokan aliluokkana.

5.3.3 Työpistelista-komponentti

Komponentti esittää valitun työhuoneen työpisteet työpistevarauksineen (sekä henkilöineen ja henkilön sopimuksineen) listana, jossa työpisteet ovat allekkain. Jos listan henkilöt eivät mahdu niille varattuun tilaan komponentti tuo esille vetopalkit, josta listaa voi selata edestakaisin. Työpisteeseen liittyvät puhelinnumerot ovat näkyvillä työpisteen tiedoissa. Puhelinnumeroa klikkaamalla sitä pääsee muuttamaan. Klikkaus avaa uuden ikkunan, jossa on napit puhelinnumeron poistamiseen ja lisäämiseen.

Syötteet

Komponentti saa syötteenä tiedon sisältämiensä tietojen mahdollisesta muutoksesta.

Komponentti tarvitsee / käyttää

Komponentti käyttää sovelluslogiikan metodia *getFilteredPeople()*, jonka avulla komponentti saa listan näytettävistä ihmisistä. Komponentti rekisteröityy kuuntelemaan sen esittämään tietoon tapahtuvia muutoksia sovelluslogiikan *registerObserver()*-metodin avulla.

Komponentti toteutetaan

Komponentti toteutetaan *JScrollPane*-luokan avulla.

5.3.4 Aikajana-komponentti

Aikajanakomponentti esittää joko henkilön tiedot, työsopimukset ja työpistevaraukset ajan suhteen tai työpisteen tiedot, varaukset sekä varauksiin liittyvät henkilöt ja näiden työsopimukset ajan suhteen.

Syötteet

Komponentti ei saa syötteitä.

Komponentti toteutetaan

JPanel-luokan avulla. Komponentti käyttää sisältämiensä komponenttien sijoitteluun *BoxLayout*-asettelumanageria. Komponentilla on oltava kaksi konstruktoria. Toisen avulla luodaan kuvaus henkilön tiedoista, toisen avulla työpisteen tiedoista.

5.3.5 Nimi-komponentti

Komponentti esittää henkilön tai työpisteen nimen. Komponentin koko riippuu nimen pituudesta, eikä sitä voi muuttaa.

Syötteet

Komponentillä ei ole syötteitä.

Komponentti tarvitsee/käyttää

Komponentti ei tarvitse ulkoisia palveluita.

Komponentti toteutetaan

Komponentti toteutetaan *JPanel*-luokan avulla.

5.3.6 Varausjakso-komponentti

Komponentti esittää varauksen keston suhteessa aikaan. Lisäksi komponentti sisältää varausjakson työpisteen ja huoneen nimen tai henkilön nimen. Varausjakson keston muutoksen yhteydessä komponentti muuttuu vastaamaan uutta kestoa.

Syötteet

Komponentti ottaa syötteenä varausjakson keston muutoksen. Kesto muutetaan painamalla hiiren vasemmanpuoleinen nappi pohjaan varausjakson jommassa kummassa päässä ja vetämällä tämä pää uudelle paikallensa. Varausjakson keston muutoksen hyväksyminen tapahtuu vapauttamalla alaspainettu nappi.

Komponentti tarvitsee/käyttää

Komponentti ei tarvitse ulkoisia palveluita.

Komponentti toteutetaan

Komponentti toteutetaan *Jpanel*-luokan ja *MouseListener*-rajapinnan avulla.

5.3.7 Sopimus-komponentti

Komponentti esittää kiinteän mittaisen sopimuskauden ajan suhteen. Komponetti ei muutu luomisen jälkeen. Komponentin päällä voi lukea jokin kuvaus sopimukseen liittyen.

Syötteet

Komponentti saa syöteenään käyttäjän D&D-operaatiosta johtuvan pudotuksen. Komponentin päälle voidaan pudottaa työpisteen nimikomponentti.

Komponentti toteutetaan

Komponentti toteutetaan *JPanel*-luokan aliluokkana. Komponentillä on metodi setPost(Post post), jonka avulla komponentti voi ottaa vastaan D&D:n pudotustoiminnon kun sen pääle pudotetaan työpisteen nimikomponentti. Komponentti käyttää *MouseAdapter*- ja *MouseMotionListener*-rajapintoja.

Työpistevarausten lisäykset, poistot ja muutokset tapahtuvat kutsumalla sovelluslogiikan vastaavia metodeja *createReservation(), deleteReservation() ja updateReservation()*.

Työpisteiden lisäys ja poisto –operaatiot käyttävät sovelluslogiikan metodeja *createPost()* ja *deletePost()*.

Puhelinnumerojen lisäys ja poisto käyttävät metodeja *addPhoneNumber()* ja *deletePhoneNumber()*.

5.4 Jana-komponentti

Komponentin ulkoinen rajapinta muodostuu sen graafisesta esityksestä käyttöliittymässä. Käyttäjä voi siirtää kahta säädintä aikajanan rajoissa. Säätimiä ei voi siirtää toistensa yli. Komponentti ilmaisee säätimien paikat (päivämäärät) aikajanalla.

Syötteet

Komponentti saa syötteenä muutoksen säätimien sijainnissa hiiren vedoilla tai metodien avulla. Komponentti saa syötteenä muutoksen aikavälissä metodien avulla.

Komponentti tarvitsee

Komponentti käyttää sovelluslogiikan *setTimeScaleSlice()*-metodia, jonka avulla se asettaa muuttuneen osa-aikavälin. Sovelluslogiikan *getFilterTimescale()*-metodin avulla komponentti saa tarkasteltavan aikavälin. Komponentti asettuu kuuntelemaan aikavälin muutoksia kutsumalla sovelluslogiikan *registerObserver()*-metodia.

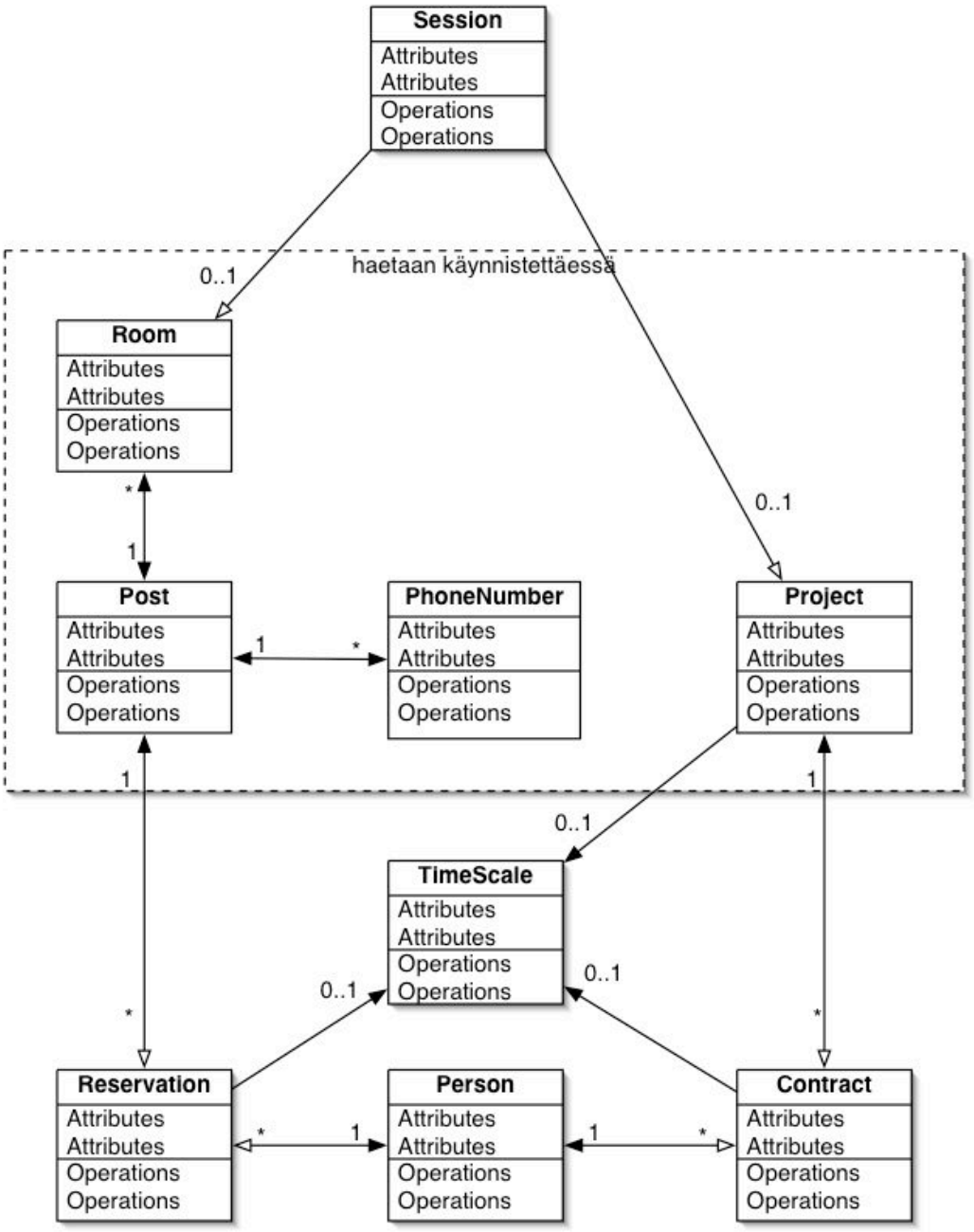
Komponentti toteutetaan

JComponent-luokan ja *Graphics2D*-rajapinnan avulla.
Komponentillä on seuraavat metodit:

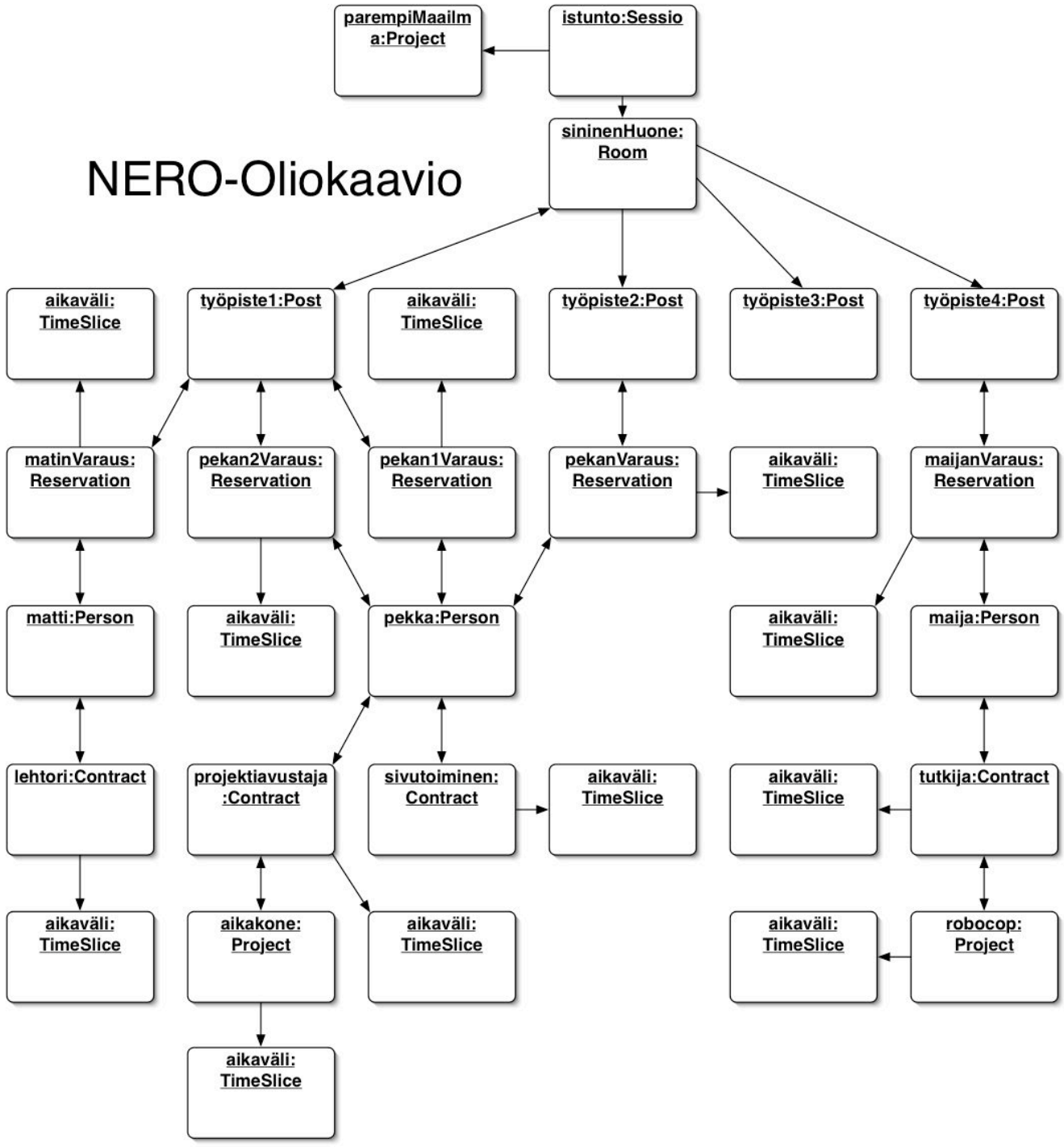
addChangeListener(), jonka avulla komponenttiin voidaan kiinnittää kuuntelija. Kuuntelijaa kutsutaan kun aikajanan säätimiä on muutettu. Kuuntelijalle kerrotaan mikä on uusi aikaväli.
removeChangeListener(), jolla poistetaan komponenttiin liitetyt kuuntelijat.
getMaximum(), joka palauttaa komponentin maksimiarvon.
setMaximum(), jolla muutetaan komponentin maksimiarvoa.
getMinimum(), joka palauttaa komponentin minimiarvon.
setMinimum(), jolla muutetaan komponentin minimiarvon.
getValueLeftIsAdjusting(), joka kertoo ollaanko juuri nyt siirtämässä vasenta säädintä.
getValueRightIsAdjusting(), joka kertoo ollaanko juuri nyt siirtämässä oikeaa säädintä.
getValueLeft(), joka palauttaa vasemman säätimen arvon.
getValueRight(), joka kertoo oikean säätimen arvon.

6 Ongelman mallintaminen

Ohjelman ajonaikainen tila mallinnetaan erilaisina olioina. Ohjelman käynnistyessä luodaan pääikkuna *JFrame*-luokan avulla. Käynnistämisen yhteydessä pääkomponentti luo *Session*-olion. *Session*-olio pitää tallessa ohjelman sen hetkisten sovelluslogiikkaan liittyvien muuttujien tilan, sekä tarjoaa sovelluslogiikan palvelut käyttöliittymän käytettäväksi. Kuvan 6 luokkakaavio esittää sovelluslogiikan keskeisimmät luokat. Kuvan 7 oliokaavio esittää ohjelman ajonaikaiset tietorakenteet sovelluslogiikan tasolla ja niiden keskinäiset suhteet.



Kuva 6 Sovelluslogiikan luokat.



Kuva 7 Oliokaavio ajonaikaisesta tietorakenteesta.

6.1 Session

Luokka pitää sisällänsä ohjelman sovelluslogiikkaan liittyvien muuttujien arvot ja tarjoaa käyttöliittymälogiikalle sen tarvitsemat sovelluslogiikan palvelut. Luokan rakenne käy ilmi sen Javadoc-kuvauksesta, joka on liitteessä 1.

6.2 Data-oliot

Data-oliot mallintavat yhden esineen tai käsitteen. Niitä käytetään tiedon tallentamiseen ja välittämiseen ajon aikana. Nerossa tulee olemaan seuraavat Data-oliot.

Luokka *Room* mallintaa huoneen. Luokan rakenne käy ilmi sen Javadoc-kuvauksesta, joka on liitteessä 2.

Luokka *Post* mallintaa työpisteen. Luokan rakenne käy ilmi sen Javadoc-kuvauksesta, joka on liitteessä 4.

Luokka *Project* mallintaa projektin. Luokan rakenne käy ilmi sen Javadoc-kuvauksesta, joka on liitteessä 4.

Luokka *Timescale* mallintaa aikavälin. Luokan rakenne käy ilmi sen Javadoc-kuvauksesta, joka on liitteessä 5.

Luokka *PhoneNumber* mallintaa puhelinnumeron. Luokan rakenne käy ilmi sen Javadoc-kuvauksesta, joka on liitteessä 6.

Luokka *Contract* mallintaa työsopimuksen. Luokan rakenne käy ilmi sen Javadoc-kuvauksesta, joka on liitteessä 7.

Luokka *Person* mallintaa henkilön. Luokan rakenne käy ilmi sen Javadoc-kuvauksesta, joka on liitteessä 8.

Luokka *Reservation* mallintaa työpisteen varausjakson. Luokan rakenne käy ilmi sen Javadoc-kuvauksesta, joka on liitteessä 9.

7 Tietokannan muutokset

Rooman tietokannan tauluihin joudutaan tekemään yksi muutos. PUHELINNUMERO-tauluun lisätään sarake TPISTE_ID, joka viittaa TYÖPISTE-tauluun, ja siihen liittyvä indeksi I_FK_PUHNRO_TPISTE. Rooman TYÖPISTE-taulun sarakkeita I_FK_TPISTE_PUHNRO ja sarake PUHNRO_ID ei käytetä Nerossa. Edellä mainittuja sarakkeita ei myöskään poisteta tauluista, jonka seurauksena voidaan käyttää Rooman kopiointiosajärjestelmää sellaisenaan.

8 Kopiointiosajärjestelmä

Kopiointiosajärjestelmään tehtiin korjaus osana Rooman korjausprojekti Viulua.

Korjaus etsii työsopimuksista nimen nimettömille projekteille, poistaen aiemmin ongelmana olleet dummy-projektit. Proceduuri lisaa_dummy muutettiin seuraavan näköiseksi.

```
PROCEDURE lisaa_dummy (p_koodi IN projekti.koodi%TYPE, nimi IN
siirto_sopimusjakso.s_projekti%TYPE) IS

BEGIN

  INSERT

    INTO projekti

         (koodi, nimi, alkupvm, loppupvm, vastuuhenkilo)
```

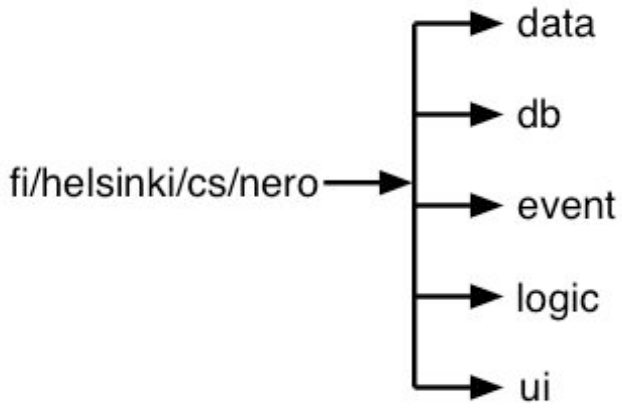
```
  VALUES (p_koodi
         ,nimi
         ,SYSDATE
         ,NULL
         ,'Ei tietoa'
         )
  ;
EXCEPTION
  -- Skipataan ORA-00001 exceptionit ("duplicate value on index"),
eli jos
  -- projekti on jo olemassa.
  WHEN DUP_VAL_ON_INDEX THEN
    NULL;
END lisaa_dummy;
```

Proseduuri tarkista_projektit muutettiin seuraavan näköiseksi.

```
PROCEDURE tarkista_projektit IS
  -- Hakee niiden projektien koodit, jotka ovat SIIRTO_SOPIMUSJAKSO-
taulussa,
  -- mutta eivät PROJEKTI-taulussa. MINUS-operaattori on kohtalaisen
rivakka.
CURSOR crs_projektit IS
SELECT DISTINCT(projekti) AS p_koodi, nvl(s_projekti, 'Nimetön') as
nimi
FROM siirto_sopimusjakso
WHERE projekti IN
       (SELECT projekti AS projekti_koodi
      FROM siirto_sopimusjakso
     WHERE projekti IS NOT NULL
   MINUS
    SELECT koodi AS projekti_koodi
      FROM projekti);
BEGIN
  FOR c IN crs_projektit LOOP
    kopiointi_projekti.lisaa_dummy (c.p_koodi, c.nimi);
  END LOOP;
END tarkista_projektit;
```

9 Lähdekoodin pakkaukset

Ohjelmiston lähdekoodin pakkausten rakenne on kuvan 8 mukainen. Juurena toimii *fi.cs.helsinki.nero*-hakemisto. Sen alihakemistoina on koodin sisällänsä pitävät hakemistot. Hakemisto *data* sisältää ajonaikaiseen tietorakenteeseen liittyvien olioiden luokat. Hakemisto *db* sisältää tietokantaoperaatiot hoitavat luokat. Hakemistossa *event* sijaitsee tapahtumaluokat, sekä niiden käsittelijä- ja kuuntelijaluokat. Hakemistossa *logic* sijaitsee ohjelman sovelluslogiikan luokat. Hakemisto *ui* pitää sisällänsä käyttöliittymän luokat.



Kuva 8 Neron lähdekoodien pakkaukset.

10 Käyttöönottosuunnitelma

Valmis ohjelmisto tullaan asentamaan asiakkaalle hyväksymistestauksen jälkeisenä päivänä 8.12.2004. Käyttöönottoon liittyy oleellisesti neljä eri vaihetta: uuden tietokannan alustus, Rooman avulla talletettujen tietojen siirtäminen Neron tietokantatauluihin, kopiointiosajärjestelmän uudelleenasennus sekä varsinaisen ohjelman asentaminen ja konfigurointi. Uusi tietokanta alustetaan käyttämällä asennuspaketissa mukana tulevia SQL-komentoja.

Koska Rooman tietokantarakenne säilyy lähes ennallaan, on Rooman avulla tehtyjen työpistevarausten siirto Neron tietokantaan helppoa. Tiedon oikeellisuus tulee kuitenkin tarkistaa, varsinkin muuttuneiden rakenteiden osalta. Virheellinen tieto voidaan korjata joko suoraan SQL-tulkilla, tai myöhemmin käyttäen varsinaista ohjelmistoa.

Rooman kopiointiosajärjestelmä poistetaan Rooman käyttöohjeessa kuvatulla tavalla ja tilalle asennetaan vastaavasti Neron kopiointiosajärjestelmä. Ohjelmisto asennetaan asiakkaan koneelle JAR-pakettina. Samalla asennetaan ohjelmiston vaatimat apukirjastot. Ohjelmisto konfiguroidaan käyttämään edellä mainittua Neron tietokantaa ja testataan, että yhteys tietokantaan toimii.

Edellä kuvatut vaiheet tullaan dokumentoimaan tarkemmin käyttöohjeessa, jonka on valmis asennuspäivään mennessä. Lisäksi ennen asennuspäivää on varmistettava, että kyseiset vaiheet voidaan viedä läpi käyttöohjeessa kuvatulla tavalla.

Liitteet

Liite 1: Session-luokan Javadoc-kuvaus.

Liite 2: NeroDatabase-luokan Javadoc-kuvaus.

Liite 3: Room-luokan Javadoc-kuvaus.

Liite 4: Post-luokan Javadoc-kuvaus.

Liite 5: Project-luokan Javadoc-kuvaus.

Liite 6: TimeSlice-luokan Javadoc-kuvaus.

Liite 7: PhoneNumber-luokan Javadoc-kuvaus.

Liite 8: Contract-luokan Javadoc-kuvaus.

Liite 9: Person-luokan Javadoc-kuvaus.

Liite 10: Reservation-luokan Javadoc-kuvaus.