LAPPEENRANNAN TEKNILLINEN YLIOPISTO

Yhdistetty harjoitustyö

CT60A4301 & CT60A2410

0452088 Jaakko Tuuri

1.6.2016

Kurssien CT60A4301 Tietokannat ja CT60A2410 Olio-ohjelmointi yhdistetyn harjoitustyön raportti, joka pitää sisällään raportin kummankin kurssin osuudesta erillisissä osissa.

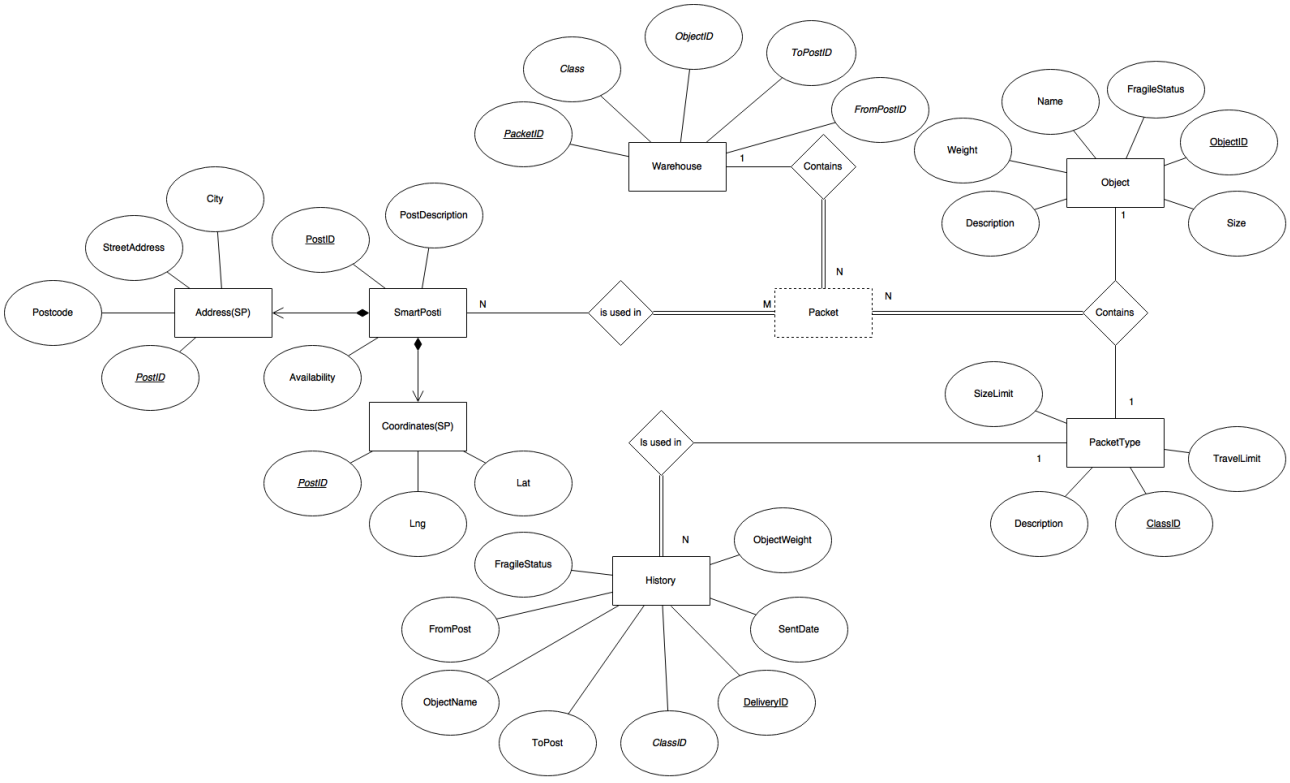
Sisällys

Tehtävänannon kuvaus ja työn rajaukset

Tehtäväksi annettiin kehittää T.I.M.O.T.E.I-ohjelma (toiminnaltaan Itellaa muistuttava ohjelmisto, tietokantaa edellyttävä integraatio), jonka tarkoitus olisi toimia jonkinlaisena pakettien luonti ja lähetys – ohjelmana. Ohjelman tulisi ladata Itellan sivuilta SmartPost-pisteiden tiedot XML-dokumenttina, käyttää saatua dataa muun muassa esittääkseen SmartPostien sijainnin kartalla, sekä antaa käyttäjän luoda esineitä sekä niitä sisältäviä paketteja. Paketteja olisi tarkoitus mahdollista asettaa varastoon, sekä lähettää eteenpäin ohjelmassa SmartPostista toiseen. Tietojen varastoinnin oli tarkoitus tapahtua SQL relaatiotietokantaa hyödyntäen.

Perustasolla ohjelman tulisi antaa käyttäjän lisätä SmartPosteja kartalle käyttöliittymän avulla, sekä luoda uusia esineitä/paketteja, sekä mahdollistaa pakettien varastointi. Pakettien lähettäminen SmartPostien välillä tulisi myös olla mahdollista. Lisäksi kartalle jo piirrettyjen SmartPostien ja/tai reittien tulisi olla poistettavissa käyttäjän toimesta. Näistä lisää kappaleessa ”Ohjelman kuvaus”.

Tietokannan käsitemalli ja eheyssäännöt



Tietokanta oli tarkoitus toteuttaa siten, että sitä käytetään harjoitustyössä tehtävän Java-ohjelman kautta. Sen olisi tarkoitus pitää sisällään Itellan internetsivujen kautta saatu SmartPost-data, eli SmartPost-pisteiden GPS-koordinaatit sekä muut tiedot eri tauluissa. Tietokantaan oli tarkoitus myös tarkoitus tehdä muita tauluja pitämään sisällään ohjelman muita tietoja, kuten esimerkiksi luodut paketit, esineet ja varasto. Käyttäjän tulisi olla mahdollista myös poistaa tietokannassa olevia tietoja niin halutessaan.

Tietokannan tulisi olla tehty relaatiomallia noudattaen, sekä sen täytyisi koostua noin 7-10 käsitteestä. Lisäksi, tietokannassa ei saisi olla tietorakenteita, koska ne olisivat relaatiomallin vastaisia. Tietokantaan tulisi myös tehdä sitä käytettäessä vähintään viisi kyselyä tai näkymää. Kyselyissä tulisi myös käyttää JOIN-

komentoa. Tietokanta täytyy tehdä siten, että se on käytettävissä yhtä hyvin sekä Java-ohjelman kautta, että SQLite3-ohjelmalla. Kun tietokantaa alettiin luoda ER-kaaviosta, määritettiin sen jokaiselle taululle eheyssäännöksi se, että mikään attribuutti ei saa sisältää ”tyhjää” missään taulussa, vaan että jokainen tietue sisältää jokaisessa kohdassa jotakin(eli asetettiin NOT NULL –komento jokaiselle attribuutille). Tällöin vältytään siltä, että tietoa ei löydy ohjelman lukiessa tietokannan sisältöä. AddressSP- ja CoordinatesSP-tauluille asetettiin eheyssäännöksi se, että kun SmartPost-taulusta poistetaan tietue, poistuu vastaavat myös näistä tauluista (eli asetettiin ON DELETE CASCADE –komento tauluihin).

Ohjelman kuvaus

Perustason vaatimukset ohjelmalle(13 pst):

Näistä täytyy olla toteutettuna jokainen.

-Pääluokka, jonka kautta on mahdollista liittää yhteen alku- ja päätepiste sekä haluttu paketti varastosta.

-Luokka, joka pitää kirjaa SmartPost-olioista (Toteutetaan tietokantaa käyttäen)

-Luokka, joka lukee XML-muotoista dataa ja syöttää sitä tietokantaan

-SmartPost-luokka, joka sisältää automaattikohtaisen datan sekä osaluokan, johon on tallennettu automaatin geograafinen paikka (Toteutetaan tietokantaa käyttäen)

-Varastoluokka, johon on mahdollista tallentaa jo luotuja paketteja (Toteutetaan tietokantaa käyttäen)

-Pakettiluokka, josta on periytetty useampia pakettiluokkia aikaisemman määritelmän mukaan (Toteutetaan tietokantaa käyttäen)

-Esineluokkia vähintään 4 kappaletta, joilla on erilaisia ominaisuuksia (Toteutetaan tietokantaa käyttäen)

Laajennetun tason vaatimukset ohjelmalle(26 pst):

Kaikki aikaisemmat vaatimukset tulee olla toteutettuna, sekä vähintään kaksi tämän tason vaatimusta.

-Pitää kirjaa pakettien määrästä, lähtö- ja saapumispaikoista, matkan pituuksista ja paketeissa olevista esineistä erillisessä ikkunassa / välilehdessä tai muussa hyvin suunnitellussa UI-komponentissa (Toteutetaan tietokantaa käyttäen)

-Ohjelmaa lopettaessa se kirjoittaa kuitin tiedostoon istunnon aikaisesta kirjanpidosta ja varaston tilanteesta, eli suoltaa login siitä mitä käyttäjä on tehnyt (Toteutetaan tietokantaa käyttäen)

-Ohjelman on käytettävä pakettien lisäämiseen ja statistiikan esittämiseen joko uusia ikkunoita tai välilehtiä

-Jokin oma hieno feature

Laajimman vaatimustason vaatimukset ohjelmalle(40 pst):

Kaikki aikaisemmat vaatimukset tulee olla toteutettuna, sekä vähintään kaksi tämän tason vaatimusta.

-GUI ei käytä vain pelkkiä standardikomponentteja, vaan niitä on muokattu visuaalisesti. Tähän ei riitä, että painikkeiden fonttia on muutettu, vaan käyttöliittymää on muokattava visuaalisesti erilaiseksi

-Pelkän logikirjanpidon lisäksi ohjelma lataa kirjanpidon käynnistyessään. Tällöin käyttäjä voi valita aloittaako hän tyhjältä pöydältä vai otetaanko käyttöön ladattu varastotilanne. Käyttäjällä on koko ajan tarjolla tilastot, jotka käsittävät ohjelman koko käyttöajan (Toteutetaan tietokantaa käyttäen)

-Ohjelman generoimia tilastoja on mahdollista selata kalenterin avulla. Käyttäjä voi valita kalenterista tietyn päivän ja tarkastella, mitkä paketit ovat liikkuneet mistä minne sinä päivänä, ovatko ne saapuneet ehjinä perille jne. (Toteutetaan tietokantaa käyttäen)

-Jokin todella hieno oma uusi feature

Ohjelmallinen toteutus ja lista toiminnallisuuksista

Tietokannan rakentaminen käsitemallista valmiiksi tietokannaksi oli lyhyt prosessi, joka hoitui kirjoittamalla SQL-tietokannan luomiseksi tarvittavat komennot erilliseen tiedostoon, ja testaamalla sitä SQLite3:lla ennen käyttämistä varsinaisessa ohjelmassa.

Ohjelma on toteutettu vaatimuksia noudattaen. Ohjelmassa on kaikki perustasolla vaaditut toiminnallisuudet, sekä perustason vaaditut ominaisuudet. Käyttäjän on mahdollista lisätä SmartPosteja kartalle, mutta myös poistaa kaikki karttamerkinnät päävalikon painikkeita käyttäen. Uudesta ikkunasta (joka aukeaa päävalikon kautta) käyttäjän on lisäksi mahdollista luoda uusia paketteja sekä esineitä lähetettäväksi päävalikon kautta. Tällöin piirretään kartalle reitti ja alku- sekä päätepisteet. Perustason vaatimuksien lisäksi ohjelma pitää sisällään muidenkin vaatimustasojen kuvailemia toiminnallisuuksia.

Laajennetuista vaatimuksista ohjelma pitää kirjaa luoduista lähetyksistä varastossa, jonka sisällön näkee pääikkunan pudotusvalikosta. Ohjelma myös tulostaa lokin sekä historian lähetetyistä paketeista erilliseen tiedostoon ohjelman sammutuksen yhteydessä. Lisäksi, ohjelma näyttää lähetettyjen pakettien tietoja historia-näkymässä. Uusien pakettien luonti sekä esineiden luonti taasen tapahtuu erillisessä ikkunassa, joka avataan päävalikosta. Omana erikoisena ominaisuutena ohjelmassa on oma, kustomoitu latausikoni (eli pyörivä doge ja ”Loading…” teksti).

Laajimman vaatimustason ominaisuuksista ohjelmassa on kustomoitu GUI, johon kuuluu muun muassa kuvankäsittelyohjelmalla tehty ohjelman nimi, sekä tausta. Ohjelma antaa myös käyttäjälle mahdollisuuden valita käyttääkö hän nykyistä varastoa, vai haluaako hän aloittaa uuden sisäänkirjautumisen yhteydessä.

Ohjelman toteutus tietokantojen käytön suhteen on tehty siten, että yksi luokka pitää huolen muiden luokkien pääsystä tietokantaan. Tämä luokka (Database_manager) pitää sisällään eri toiminnallisuuksia sisältäviä metodeja, joita kutsuttaessa palautetaan haluttu tieto tietovarastona toimivan olion sisälle laitettuna. Eli, varsinaiset tietokannan lukemiset tai sinne laittamiset tapahtuvat kyseisen luokan sisällä. Mikään muu luokka ei käytä tietokantoja, paitsi poikkeuksena BG(BackGround)-luokat, jotka on tehty siten, että ne pyörivät kutsumisen jälkeen taustalla. Ne pitävät tarvitsemansa tietokantatoiminnallisuuden sisällään, eivätkä siten kutsu Database_manager-luokkaa. Niiden tehtävänä on pääasiassa hoitaa tietokannan suuremmat, yksittäiset muutokset, kuten SmartPost-datan päivitys sekä tietokannan luominen.

Database_manager-luokan lisäksi on kaksi muuta ”yleisesti usein käytettyä” luokkaa. Ohjelman käyttöliittymä pitää sisällään useita ikkunoita, joten täytyi olla jonkinlainen mahdollisuus saada ne muokkailemaan toistensa sisältöä. Tämän vuoksi kehitettiin InterfaceController-luokka, joka pitää sisällään eri ikkunoiden sisältöä, jolloin kyseisen luokan kutsumisen jälkeen voidaan päästä käsiksi toisen ikkunan elementteihin. Viimeinen yleisesti käytetty luokka on LogWriter-luokka. Tämä luokka tehtiin kirjaamaan

jokainen ohjelman toimenpide erilliseen lokiin. Eli, jos jotain meni pieleen, voitiin se kirjata lokiin josta käyttäjä pystyi sen katsomaan.

Koska ohjelman tulisi pystyä hallitsemaan tietokannasta saatua dataa järkevästi ja yksinkertaisesti, talletetaan tietokannasta saadut tiedot olioihin, joita ohjelma voi käsitellä. TempStorageController-luokka loi uusia olioita, jotka pitivät sisällään haetun tiedon. Tällaisia ”tietovarastoluokkia” oli luotu yksi jokaiselle tarpeelle, kuten esineille, pakettityypeille, lähetyksille, SmartPosteille sekä yhden lähetyksen tarkoille tiedoille (mikäli haluttiin saada tiedot yksittäisestä lähetyksestä mahdollisimman yksinkertaisesti).

Kaikki muut ohjelman luokista ovat FXML_Controller-luokkia. Nämä luokat hallinnoivat käyttöliittymän ikkunoita, sekä niihin liittyviä toiminnallisuuksia. Esimerkiksi FXML_loginController-luokka hallitsee sisäänkirjautumisnäyttöä ja FXML_mainController hallitsee ”pääikkunaa”. Koska FXML_mainController hallitsee pääikkunaa, on tässä luokassa enemmän sisältöä kuin muissa. Lisäksi, tämän ikkunan sulkeutuessa sulkeutuvat kaikki muutkin ikkunat. Tämä luokka hoitaa myös automaattisen historia- ja lokitiedostojen kirjoittamisen ohjelman sulkeutuessa.

Lista ohjelman toiminnallisuuksista:

-Sisäänkirjautuminen salasanan ja käyttäjänimen avulla (ohjelmassa kirjaudutaan sisälle käyttäjänimi ”admin” ja salasana ”password”)

-mahdollisuus valita halutaanko käyttää uutta vai vanhaa tietokantaa

-mahdollisuus valita halutaanko käyttää uutta vai vanhaa varastoa

-ohjelman toiminnan kirjaaminen lokiin

-lähetettyjen lähetysten tietojen kirjaaminen historiaan

-SmartPostien lisääminen kartalle paikkakunnan mukaan pudotusvalikoiden ja napin avulla

-kaikkien karttamerkintöjen poistaminen nappia painamalla

-uusien esineiden luonti käyttöliittymästä

-uusien lähetysten luonti(sisältävät: esine, pakettityyppi, alku- sekä päätepiste) (uusi ikkuna)

-lähetysten(valmiiden pakettien) lähetys pudotusvalikon ja napin avulla

-valitun lähetyksen tietojen tarkistus (uusi ikkuna)

-varaston tyhjennys napin avulla

-tietokannan muokkaaminen asetuksista (uusi ikkuna)

-->vanhan SmartPost-datan korvaus uudella

-->tietokannan palautus oletustilaan

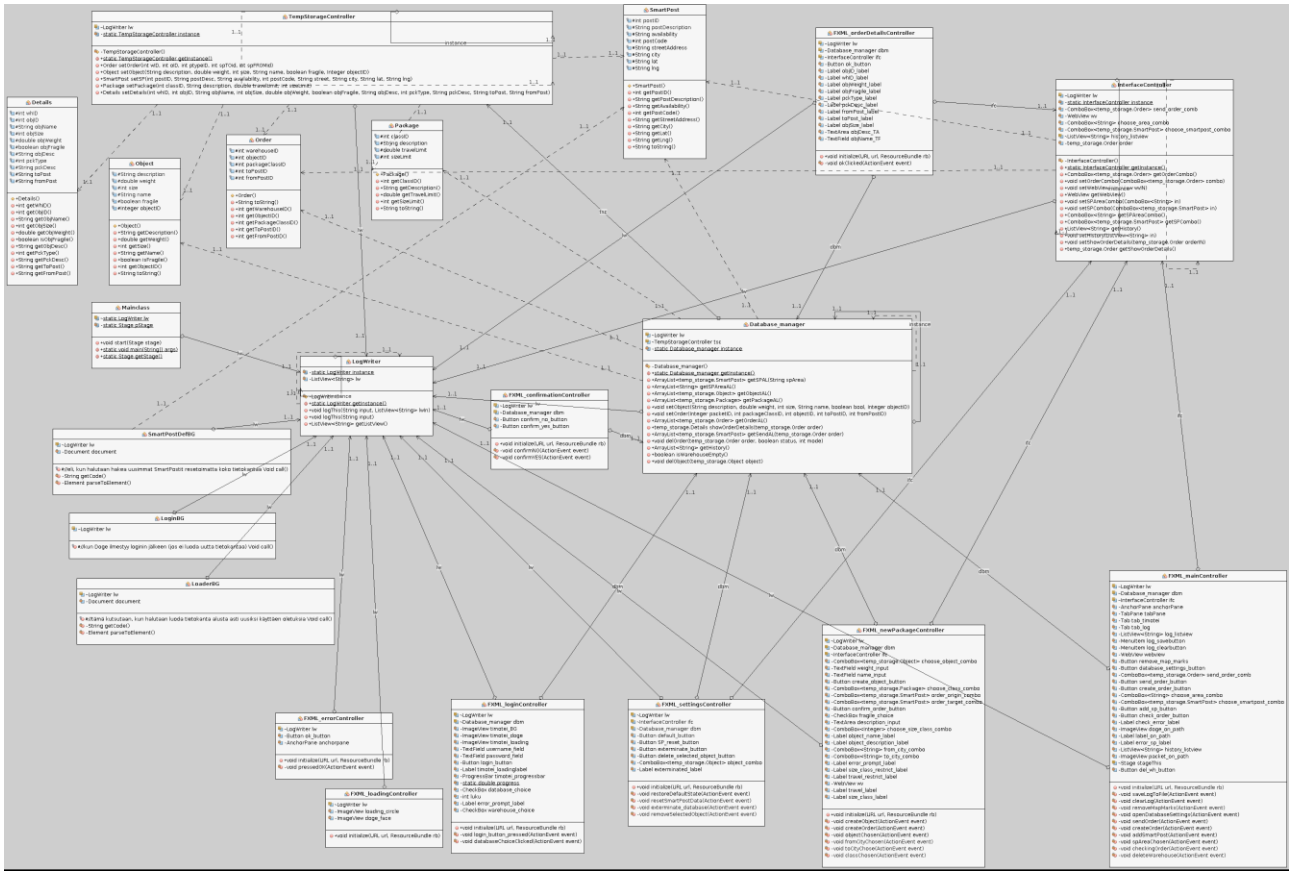
-->tietokannan täydellinen tuhoaminen

-->esineiden poistaminen pudotusvalikon ja napin avulla

-lokin ja historian kirjoittaminen tekstitiedostoihin, kun ohjelma lopetetaan

-lokin tyhjentäminen tai tallentaminen manuaalisesti käyttäjän niin halutessa

Ohjelman täydellinen luokkakaavio



Yhteenveto – ongelmat ja niiden ratkaisut

Ohjelman teon aikana muutettiin tietokannan taulut AddressSP ja CoordinatesSP siten, että niihin lisättiin ON DELETE CASCADE SmartPost-taulun pääavaimen viiteavaimeen. Tällöin tarvitsee poistaa SmartPost-tietue vain SmartPost-taulusta, jolloin vastaavat poistuvat myös muista tauluista. Muuten olisi tarvinnut poistaa tiettyyn SmartPostiin liittyvät tietueet jokaisesta taulusta erikseen.

Muita ohjelmaa tehtäessä vastaan tulleita ongelmia olivat muun muassa se, että miten ikkunat voisivat muokata toistensa sisältöä tai että miten onnistuisi eri luokkien tietojen kirjaaminen samaan lokiin. Näihin kehitettiin ratkaisuiksi se, että käytettäisiin luokkia, jotka toimivat ikään kuin ”siltoina” muiden luokkien välillä. Tällöin tietoa saatiin siirrettyä luokasta toiseen varastoimalla se erilliseen luokkaan, josta se voitaisiin vaivattomasti noutaa tarvittaessa.

Ongelmia tuotti myös se, että miten taustaprosessit saataisiin toimimaan siten, että ohjelmaa pystyisi käyttämään ilman, että se menisi suorituksen ajaksi jumiin. Lisäksi, ohjelman pitäisi näyttää suoritusprosentti esimerkiksi SmartPost-tietojen lataamiselle latauspalkin avulla. Tähän ratkaisuksi kehitettiin BG-luokat, jotka laajentavat Javan Task-luokkaa. Tämä mahdollisti niiden ajamisen taustalla, mutta myös niiden edistymisen seuraamisen.