

Varausjärjestelmä

Matti Yli-Ojanperä
355153
Automaatio- ja systeemitekniikka
30.4.2018

Yleiskuvaus

Kuvitteelliselle yritykselle toteutettu varausjärjestelmä, joka yhdistää kalenteri- asiakas-, palvelu- ja resurssitietoja. Ohjelmaa käytetään graafisen käyttöliittymän kautta. Ohjelmaan liitetään tiedot yrityksen käytössä olevista resursseista sekä yrityksen tarjoamista palveluista. Käyttäjä voi muokata näitä tietoja vastaamaan yrityksen kulloistakin tilannetta.

Varausta tehdessä ohjelma tallentaa tiedot tehdyistä varauksista kalenteriin ja merkitsee valitut resurssit varatuiksi, jolloin konfliktitilanteessa (resurssi on jo varattu) varausta ei voida toteuttaa. Varaukseen ohjelma liittää tarvittavat tiedot asiakkaasta sekä asiakkaan valitsemat palvelut.

Ohjelma näyttää valitun päivän varaukset jaotellen jokaisen resurssin omaan sarakkeeseen. Lisäksi ohjelmalla voi etsiä tietyltä aikaväliltä varaukset sekä tarkastella asiakaskohtaisia varauksia. Ohjelma laskee myös varauksen pituuden ja siihen valittujen palveluiden perusteella hinnan.

Ohjelmalla voi lisätä, muokata ja poistaa varauksia, palveluita, asiakastietoja sekä resursseja. Ohjelma tallentaa keräämänsä tiedot SQLite reaaliaikotietokantaan.

Mielestäni työ on toteutettu vaikeana.

Käyttöohje

Ohjelma käynnistetään komentoriviltä python main.py komennolla. Ennen kuin voi lisätä varauksia pitää lisätä vähintään yksi resurssi ja palvelu.

Resurssin lisääminen: Manage resources → New → täytä tarvittavat tiedot → Ok

Palvelun lisääminen: Manage services → New → täytä tarvittavat tiedot → Ok

Varauksen lisääminen: New reservation → New → täytä tarvittavat tiedot → Ok

Huomioitavaa varauksen palveluita valittaessa: Pitämällä ctrl tai shift nappeja pohjassa voi valita useamman palvelun.

Asiakkaita voi lisätä varauksen lisäämisen yhteydessä tai samaan tyyliin kuin resurssien ja palveluiden lisäys.

Vasemmassa reunassa olevaa kalenteria klikkaamalla voi vaihtaa tarkasteltavan päivän. Kalenterissa on merkitty vihreällä ne päivät, jolloin on varauksia.

Mielestäni käyttöliittymä on hyvin selkeä ja intuitiivinen, että käytöstä selviää ilman ohjeita.

Ulkoiset kirjastot

Projektissa on käytetty PyQt5 kirjastoa graafista käyttöliittymää varten.

Ohjelman rakenne

Database-luokka

Yhteys sqlite tietokantaan. Säiliö table-luokille. Rajapinta tietokannan ja Model-luokkien välillä.

Table-luokat (ReservationTable, CustomerTable, ResourceTable, ServiceTable, ReservationServiceTable)

Vastaavan sqlite taulun käsittelyyn.

Model-luokat (Reservation, Customer, Resource, Service)

Yksittäisiä varauksia, resursseja, asiakkaita sekä palveluita.

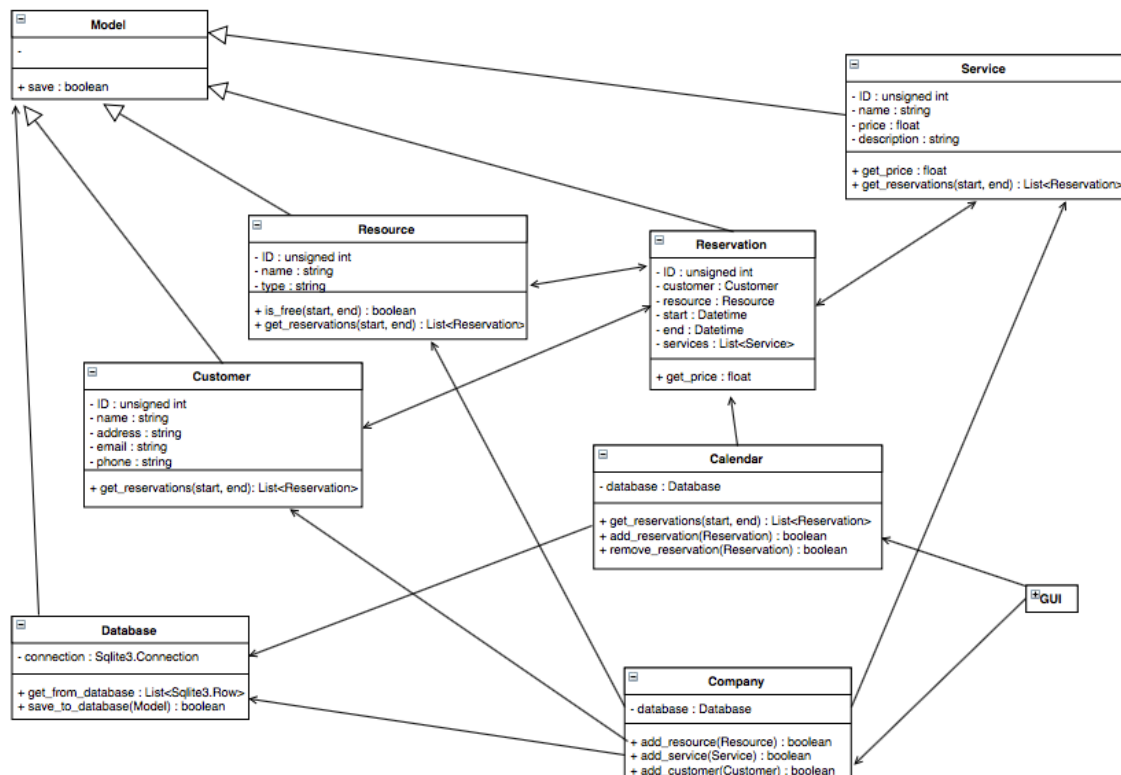
Dialog-luokat

Käyttöliittymän pop-up -ikkunat tietojen lisäämiseen ja hallintaan.

GraphicsItem-luokat

Valitun päivän varausten esittämiseen.

Toteutus ei eroa ajatuksellisesti juurikaan suunnitellusta luokkajakoista. Calendar ja Company luokat puuttuvat toteutuksesta koska ne on kaikki yhdistetty Database-luokkaan.



Kuva 1 Suunniteltu luokkajako

Algoritmit

Vaikka ohjeissa sanottiin, että jokaisessa työssä on aina algoritmeja niin en koe, että työssä olisi yhtäkään esittelemisen arvoista algoritmia.

Esitellään nyt kuitenkin pari:

Varauksen hinta lasketaan summaamalla varaukseen liitettyjen palveluiden hinnat.

Varaukset kesto lasketaan summaamalla varaukseen liitettyjen palveluiden kestot.

Tietorakenteet

Ohjelma käyttää sqllite tietokantaa tiedon varastointiin.

Ohjelma ei lataa käynnistyessään kaikkea dataa luokkiin, vaan tekee hakuja tietokantaan tarpeen mukaan.

Ohjelman käyttämät taulut:

Resources(ID, Name, Type)

Services(ID, Name, Price, Description)

Customers(ID, Name, Email)

Reservations(ID, CustomerID, ResourceID, Start, End)

ReservationServices(ID, ReservationID, ServiceID)

Tiedostot

Ohjelma käsittelee sqllite tietokantatiedostoa eli .db-tiedostoa.

Testaus

Suunnitelman mukaan database-luokkaa on testattu yksikkötestausen avulla. Kaikki suunnitellut testit menee läpi. Täyteen testikattavuuteen en pyrkinyt vaan siihen, että voin luottaa siihen, että backend-toimii halutulla tavalla, jolloin ongelmat on helpompi paikantaa GUI-puolella. Yksikkötestaus oli mielestäni erittäin hyvä ratkaisu, sillä tietokanta ei ole ihmiselle luettavassa muodossa, minkä vuoksi ilman testejä on vaikea arvailla onko kaikki kunnossa.

Gui-puolella testaus on suoritettu ohjelmaa käyttämällä ja kokeilemalla.

Ohjelman tunnetut puutteet ja viat

Käyttäjän syöttämää tekstiä ei validoida, mikä tarkoittaa sitä että esim. Nimet voivat olla pelkkiä välilyöntejä ja sähköposti voi myös olla mikä tahansa merkkijono.

Resurssien, palveluiden ja asiakkaiden nimien ei tarvitse olla uniikkeja, mikä vaikeuttaa näiden erottamista toisistaan käyttäjän näkökulmasta. Ohjelma tai tarkemmin tietokanta käyttää primary key:tä, mikä tarkoittaa sitä että ohjelmalla ei ole ongelmia samojen nimien kanssa.

Varauksen voi lisätä menneisyyteen. Tämä oli välillä tehty niin, että vain nykyhetkestä eteenpäin pystyy lisäämään varauksen mutta toteuttamani varauksien muokkaaminen olisi sitten tarkoittanut sitä että jos muokkaa mennyttä varausta niin sitä ei voi pitää menneisyydessä. Ajattelin että ehkä parempi että voi lisätä varauksen jälkikäteenkin.

Jokaisesta pop-up ikkunasta puuttuu otsikko.

Vasemman valikon nappeja painamalla voi avata äärettömän määrän pop-up-ikkunoita.

Jos poistaa resurssin tai asiakkaan, poistuu myös kaikki varaukset joihin nämä oli liitetty ilman sen kummempia kyselyitä. Tähän voisi lisätä käyttäjältä varmistuksen, että oletko varma haluatko jatkaa.

Jos poistaa palvelun, poistuu se palvelu kaikista varauksista, joissa se oli. Mutta varauksen kesto ei muutu.

Palveluiden, resurssien ja asiakkaiden poistaminen vaikuttaa myös historia tietojen keräykseen. Poistamisen sijaan parempi ratkaisu olisi lisätä ominaisuus, jolla voisi poistaa käytöstä palvelu, resurssi yms. Jolloin menneet varaukset pysyisivät ennellaan ja tuleviin varauksiin tulisi muutos. Ja samalla kun käyttäjä poistaa jonkin käytöstä saisi tämä listan muuttuneista varauksista, että voisi olla esim yhteydessä asiakkaaseen.

Parhaat ja heikoimmat kohdat

Yksi parhaista kohdista ohjelmassa sqliten käyttö ja sen abstraktoiminen Database luokan avulla. Backend:in puolella on mielestäni muutenkin oikein hyviä ja toimivia ratkaisuja kuten esim Model-luokan periminen. Gui:n puolella varausta lisätessä hinta ja kesto päivittyvät samalla kun valitaan palveluita. Lisäksi resursseista on valittavissa vain ne jotka ovat valittuna ajan kohtana vapaana.

Ehdottomasti heikoin kohta on yleinen kommenttien puute ohjelmakoodissa. Gui on mielestäni viimeistelemätön.

Poikkeamat suunnitelmasta

En poikennut juurikaan suunnitelmasta toteutuksen osalta. Sain melkein kaiken toteutettua mitä olin suunnitellutkin. Historiatietojen tarkastelu jäi puuttumaan, toki menneitä varauksia pystyy katselemaan. Sekä varausten etsiminen resurssien tai palveluiden perusteella.

Toteutunut työjärjestys ja aikataulu

Työ järjestys pysy pitkälti suunnitellun mukaisena, mutta aikataulullisesti viimeisille päiville/öille jäi paljon tehtävää.

Pysyin aikataulussa noin ensimmäiset neljä viikkoa, kun tein tasaisesti hommia. Sitten tuli elämässä vähän muuta ja koodailu jäi vähemmälle ja nyt sitten viimeisellä viikolla käytännössä yötä päivää koodanneena sain kirittyä.

Arvio lopputuloksesta

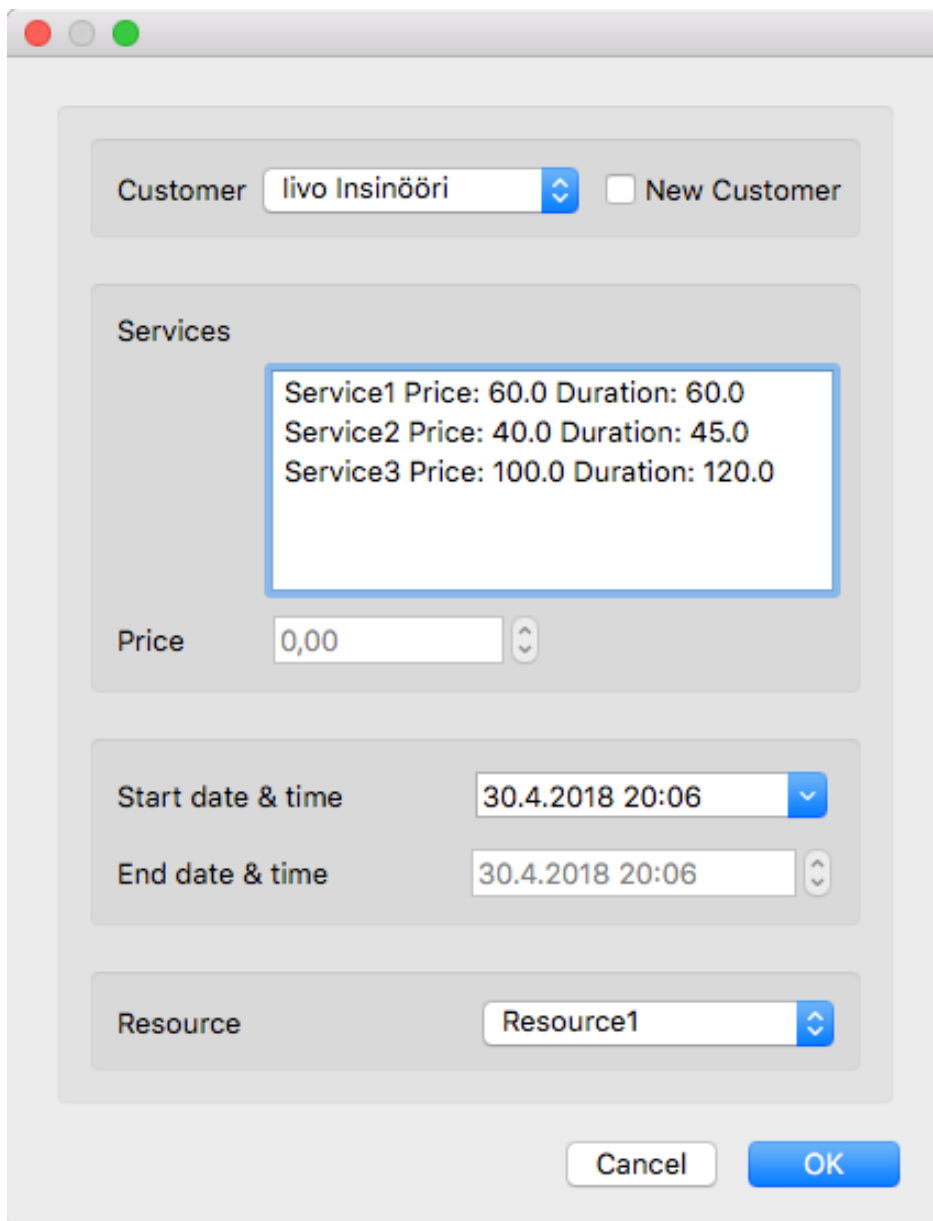
Mielestäni lopputulos on kurssin laajuuteen suhteutettuna hyvä. Ohjelma toimii niin kuin pitää, sitä on yksinkertainen käyttää ja se on koodattu niin, että sitä on helppo kehittää. Kommenttien puutetta perustelen sillä, että nimeäminen on pyritty tekemään niin, että funktiot ja luokat selittävät itse itsensä. Olen tyytyväinen luokkajakoon backend:in puolelta. Gui:n puolella sen sijaan koodissa on paljon siistimisen varaa ja varmasti PyQt kirjastoa olisi voinut käyttää tehokkaamminkin. PyQt oli tuntematon kirjasto ennen tätä kurssia ja tuntuu, että sain vain pintaraapaisun siitä. Joka kerta, kun käyttöliittymää muokkasi oppi jotain uutta ja joutui toteamaan, että näinhän tämä kannattaisikin tehdä. Sqlite käyttö oli ehdottomasti ohjelman jatkokehityksen kannalta tärkeä valinta. Ja se miten sen käytössä onnistui loi hyvän pohjan kehittää ohjelmaa vaikka kuinka paljon.

Viitteet

<http://doc.qt.io/qt-5/reference-overview.html>
<https://docs.python.org/3.6/library/>
<https://www.sqlite.org/docs.html>
<https://pysqlite.readthedocs.io/en/latest/sqlite3.html>
<http://zetcode.com/db/sqlitepythontutorial/>
<http://zetcode.com/gui/pyqt5/>

Liitteet

<https://version.aalto.fi/gitlab/yliojm1/varausjarjestelma>



A screenshot of a reservation system window. The window has a title bar with red, yellow, and green buttons. The main content area is divided into several sections. At the top, there is a 'Customer' section with a dropdown menu showing 'Iivo Insinööri' and a 'New Customer' checkbox. Below this is a 'Services' section with a list of services: 'Service1 Price: 60.0 Duration: 60.0', 'Service2 Price: 40.0 Duration: 45.0', and 'Service3 Price: 100.0 Duration: 120.0'. Below the services list is a 'Price' section with a text input field showing '0,00'. At the bottom, there is a 'Start date & time' section with a dropdown menu showing '30.4.2018 20:06' and an 'End date & time' section with a text input field showing '30.4.2018 20:06'. At the very bottom, there is a 'Resource' section with a dropdown menu showing 'Resource1'. At the bottom right of the window are 'Cancel' and 'OK' buttons.

Customer Iivo Insinööri ☐ New Customer

Services

Service1 Price: 60.0 Duration: 60.0
Service2 Price: 40.0 Duration: 45.0
Service3 Price: 100.0 Duration: 120.0

Price 0,00

Start date & time 30.4.2018 20:06

End date & time 30.4.2018 20:06

Resource Resource1

Cancel OK

Varausjärjestelmä

	Resource1	Resource2	Resource3	Resource4
16	Iivo Insinööri 2018-04-30 16:00 - 2018-04-30 19:45			
17				
18				
19				
20				

Add reservation

Manage reservations

Manage resources

Manage services

Manage customers

huhtikuuta

2018

ma

ti

ke

to

pe

la

su

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