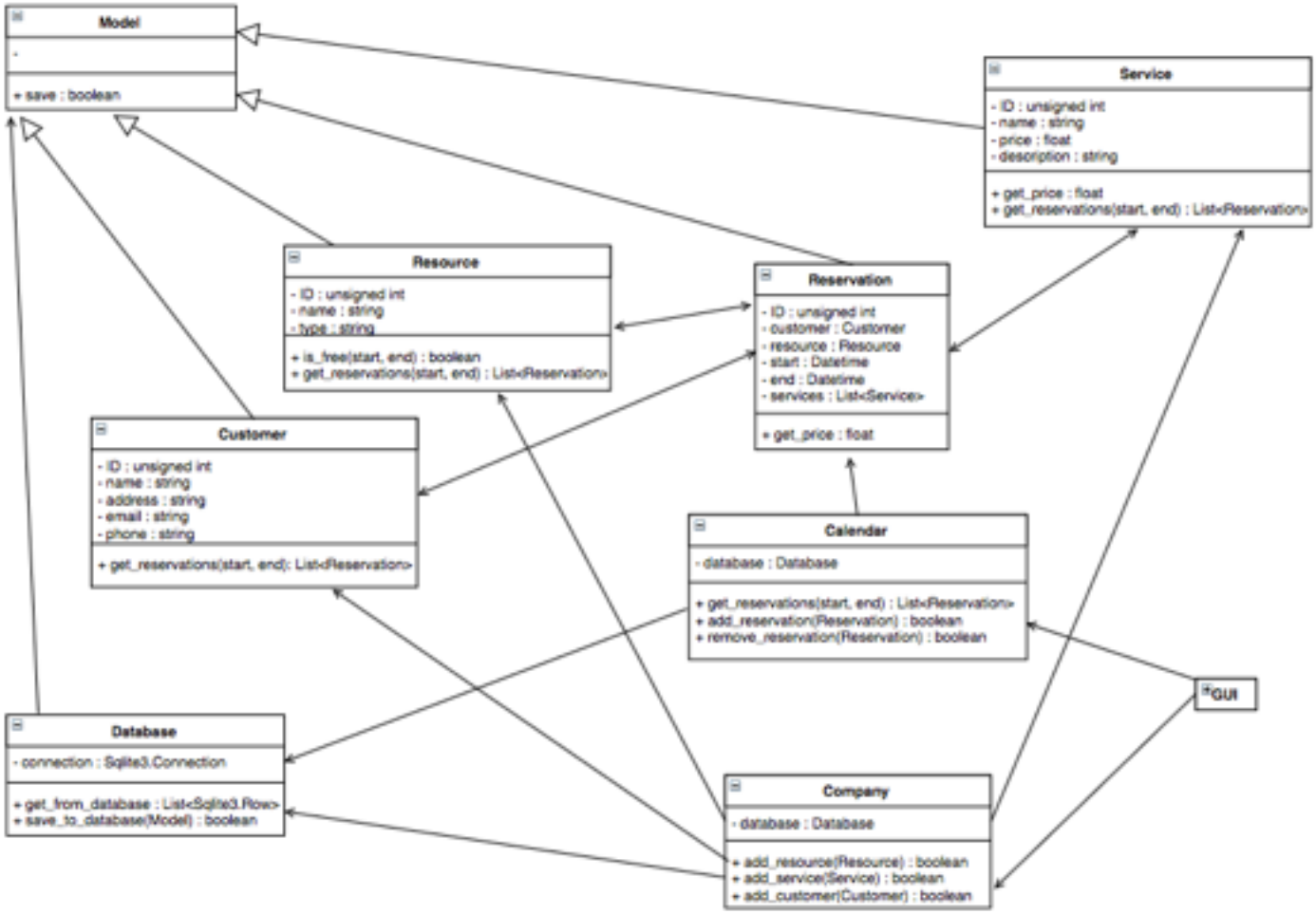
Tekninen suunnitelma

Ohjelman rakennesuunnitelma



Yllä on UML-kaavio ohjelman luokkajaosta. Kaaviossa on luokkien keskeisimmät metodit.

Graafiseen käyttöliittymään liittyvät luokat ovat hyvin suppeasti kuvattu, koska pohdin mahdollisuutta siihen, että Customer, Resource, Reservation ja Service-luokat perivät sopivat QtWidget:in, jolloin ei tarvitse erikseen luokkaa jokaisen graafiseen esitykseen. Tämän ratkaisen kun pääsen tutkimaan ja kokeilemaan kuinka paljon koodia graafinen esitys vaatii, jos koodia tulee paljon jaan graafisen puolen omaan luokkaansa. Yksi luokka joka puuttuu kaaviosta on Day-luokka jonka graafinen esitys on neliö ja niistä koostuu kalenteri-näkymä.

Customer, Resource, Reservation ja Service-luokat perivät Model luokan. Pohdin, että onnistunko tekemään Model-yläluokasta sellaisen, joka osaa tallentaa luokan tiedot tietokantaan, jolloin alaluokat voisi tallentaa tietokantaan kutsumalla save-metodia. Tämä voi kuitenkin osoittautua liian

vaikeaksi, jolloin teen Model-luokasta abstraktin ja save-metodin implentaatio tulee jokaiseen alaluokkaan.

Yhteys tietokantaan kulkee Database-luokan kautta. Calendar ja Company-luokat ovat ”säiliöitä” Customer, Resource, Reservation ja Service-luokkien objekteille.

Käyttötapauskuvaus

Varauksen lisääminen

1. Käyttäjä käynnistää ohjelman
2. Käyttäjä klikkaa hiirellä Uusi Varaus -nappia
    1. GUI-luokan luoman napin painallus aktivoi Calendar-luokan
    2. GUI-luokka tekee uuden pop up ikkunan
3. Käyttäjä syöttää varauksen tiedot uuden pop up ikkunan kenttiin
    1. Calendar-luokka luo uuden Reservation-luokan objektin
    2. Reservation-luokka tarkistaa, että Resource-luokan olio on vapaana
    3. Jos käyttäjän antama asiakas on uusi luodaan myös uusi Customer-luokan olio
    4. Tallennetaan uudet oliot tietokantaan, jolloin Database-luokka aktivoituu
4. Onnistuneesta varauksen lisäämisestä tulee tieto käyttäjälle
5. Varaus näkyy kalenterissa

Algoritmit

Ohjelman toimintaan ei liity juurikaan algoritmeja.
Varauksen hinta lasketaan summaamalla palveluiden hinnat yhteen sekä lisäämällä varauksen aika tunteina kerrottuna tuntihinnalla.

Tietokannasta haut suoritetaan sql-kyselykielellä.

Tietorakenteet

Ohjelman käyttämät tiedot tallennetaan sqlite3 -tietokantaan, mikä antaa hyvät mahdollisuudet laajaan tietojen keräykseen. Ohjelma hakee tietokannasta vain tarvitsemansa tiedot ja luo niistä oliot. Kaikki ohjelman käyttämät tietorakenteet ovat mahdollisuuksien mukaan luokkien sisällä.

Aikataulu

Viikko 9

Tiistai 27.2.2018 Suunnitelmademo

Reservation-luokan teko ilman customer, services ja resource attribuutteja
Abstrakti Model-luokka
Database-luokka niin että varaukset voi tallentaa/hakea
Gui sen verran että pystyy lisäämään varauksen
Unittestien teko Database-luokalle

Viikko 10

Resource-luokan toteutus sekä integrointi edelliseen työhön
Gui resurssien lisääminen
Gui kalenteri-näkymä
Unittestien teko Reservation ja Resource luokille

Viikko 11

Customer-luokan toteutus sekä integrointi edelliseen työhön
Unittestien teko Customer-luokalle

Viikko 12

Historia-tietojen haku/katselu
GUI hiomista paremmaksi

Viikko 13
    Testikattavuuden parantamista ja varmistus että keskivaikea toteutus ohjelmasta toimii
Viikko 14
    Service-luokan toteutus sekä integrointi edelliseen työhön
Viikko 15
    Hinnan laskemisen lisääminen
Viikko 16
    Testailua ja mahdollisesti graafisten tilastojen lisäys
Viikko 17
    Testailua ja bugien korjausta

Yksikkötestaussuunnitelma

Tärkein testattava luokka on Database, joka hoitaa kommunikoinnin tietokannan kanssa. Muita ohjelman kannalta tärkeitä on Resource-, Reservation- ja Customer-luokat.

Database-luokan get_from_database metodin testauksessa pitää katsoa, että metodi palauttaa kaikki rivit jotka pitääkin.

Resource-luokan is_free metodi on tärkeä osa päällekkäisten varausten estoa. Sen pitää palauttaa totuusarvo kertoen onko resurssi vapaa vai ei haluttuna ajankohtana. Testeissä erityisesti huomiota, että yhden varauksen loppuaika voi olla seuraavan aloitus. Lisäksi pitää huomioida, että varaukset voi olla eripituisia, joten tilanne jossa alku- ja loppuaikana resurssi on vapaa muttei koko aikaväliä.

Reservation-luokka get_price metodi palauttaa varauksen hinnan. Testeissä tärkeä tarkistaa, että metodi laskee kaikkien valittujen palveluiden hinnat yhteen sekä että varauksen kesto lasketaan oikein.

Customer-luokan get_reservations metodin pitää palauttaa kaikki kyseisen asiakkaan varaukset halutulta aikaväliltä. Testeissä pitää varmistaa, että metodi palauttaa oikeat varaukset. Testaus myös yhdellä, monella ja ei yhdelläkään tuloksella.

Kirjallisuusviitteet ja linkit

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/