

# **Loppudokumentti TryWithKids**

Ohjelmistotekniikka kevät 2019  
Satu Korhonen

# Sisällysluettelo

<b>SISÄLLYSLUETTELO</b>	<b>2</b>
<b>1. JOHDANTO</b>	<b>3</b>
<b>2. PÄIVITETTY VAATIMUSMÄÄRITTELY</b>	<b>3</b>
<b>3. KÄYTTÖOHJE</b>	<b>5</b>
3.1. Kehitysympäristö ja ohjelmiston rajoitteet	5
3.2. Asennusohje	5
3.3. Käyttöohje	6
3.4. Sananen turvallisuudesta käyttäjille	13
3.5. Ohjelmiston rajoitteet, virhetilanteet ja bugit ohjelmassa	14
<b>4. OHJELMISTON RAKENTEEN JA TOIMINNAN KUVAUS YLEISELLÄ TASOLLA</b>	<b>15</b>
<b>5. TARKEMPI KUVAUS SOVELLUSLOGIIKASTA</b>	<b>17</b>
<b>6. TESTAUSSELOSTUS</b>	<b>24</b>
6.1. Yksikkö- ja järjestelmätestaus	24
6.2. Järjestelmätestaus	27
6.3. Ongelmia testauksessa	29
<b>7. POHDINTA JA ITSEREFLEKTIO</b>	<b>30</b>
<b>LIITTEET</b>	<b>33</b>
Readme.md	33
Työpäiväkirja	36
Git ja Github	42
Testausta - viikko 2	46
Vertaisarvio 1	48
Vertaisarvio 2	49

## 1. Johdanto

Tämä on loppudokumentti projektille TryWithKids-ohjelmistolle. Ohjelmisto on desktop-ohjelmisto, joka on kirjoitettu javalla. Tietokantaratkaisuna on MongoDB ja ohjelmassa on kerrosarkkitehtuuri. Ohjelmiston tarkoituksena on toimia apuvälineenä opettajalle tai vanhemmille, jotka haluavat tehdä luonnontieteellisiä kokeita lasten kanssa. Ohjelmassa tulee valmiina neljä koetta ja vanhempi tai opettaja pystyy näitä kokeita lisää tekemään. Lisäksi vanhempi tai opettaja kykenee luomaan lapsille käyttäjätilejä ohjelmaan. Lisää ohjelmiston mahdollisuuksista ja ominaisuuksista löytyy luvusta 2, jossa on vaatimusmäärittely. Luvussa 3 löytyy asennusohjeet, käyttöohjeet sekä virheilmoitukset ja bugit. Luvusta 4 löytyy ohjelmiston kuvaus yleisellä tasolla ja luvusta 5 kuvaus tarkemmalla tasolla. Luku 6 keskittyy ohjelmiston testaukseen. Viimeinen luku 7 sisältää pohdintaa ja reflektiota ohjelman tekemisen ajalta ja sen jälkeen.

## 2. Päivitetty vaatimusmäärittely

### Lyhyt yleiskuvaus

Tarkoituksena on tehdä MongoDB-tietokantaa hyödyntävä Java Maven -ohjelmisto, johon opettaja tai vanhempi voi lisätä luonnontieteellisiä kokeita tehtäväksi lasten kanssa. Lisäksi vanhempi tai opettaja voi lisätä, nähdä ja poistaa lasten käyttäjätietoja. Hän voi myös lisätä, muokata, hakea ja poistaa luonnontieteellisiä kokeita. Lapset voivat nähdä omat tietonsa, selata kokeita, hakea kokeita ja lisätä niitä omalle listalleen sekä poistaa niitä sieltä.

### Luettelo ohjelmiston käyttäjärooleista

Ylläpitäjä (vanhempi tai opettaja), joka voi lisätä kokeita ohjelmistoon sekä selata, etsiä, muokata ja poistaa tietokannassa olevia kokeita. Ylläpitäjä voi lisäksi lisätä käyttäjiä, selata käyttäjien tietoja ja poistaa käyttäjiä.

Lapset/loppukäyttäjät voivat etsiä kokeita eri kriteerien (aihepiirit ja käytettävissä oleva aika) pohjalta ja tutustua niiden kuvauksiin sekä lisätä kokeita omalle listalleen. He voivat myös nähdä omat tietonsa.

Kumpikin käyttäjäryhmä voi tulostaa kokeita pdf-tiedostoiksi. Molemmat käyttäjäryhmät voivat vaihtaa oman salasansansa, lisätä kokeita omalle listalleen, nähdä oman listansa ja poistaa kokeita omalta listaltaan.

### Kuvaus ohjelmiston toiminnallisista vaatimuksista

- Kokeiden lisääminen: ylläpitäjä
- Kokeiden selaaminen: ylläpitäjä ja loppukäyttäjä
- Kokeiden poistaminen: ylläpitäjä
- Yksittäisten kokeiden yhdistäminen listaksi: ylläpitäjä, loppukäyttäjä
- Kokeiden muokkaus: ylläpitäjä
- Kokeiden hakutoiminto 2 kriteerin pohjalta: ylläpitäjä ja loppukäyttäjä
- Kirjautumistoiminto: ylläpitäjä ja loppukäyttäjä
- Käyttäjien lisääminen: ylläpitäjä
- Käyttäjien tietojen selaaminen: ylläpitäjä
- Salasanan vaihto: ylläpitäjä ja loppukäyttäjä

- Omien tietojen tarkastelu: ylläpitäjä ja loppukäyttäjä
- Koe-tietojen tulostaminen mielekkäästi muotoiltuna: ylläpitäjä ja loppukäyttäjä
- Listojen selailu: ylläpitäjä ja loppukäyttäjä

### **Lyhyt kuvaus etenemistavasta – kevyt projektisuunnitelma**

#### **Iteraatio 1:**

- Valita kokeisiin sopiva tietokantaratkaisu
- Tehdä mielekkäät luokat
- Kyky lisätä kokeita
- Kyky selata kokeita
- Kyky lukea kokeen ohjeet
- Yksinkertainen graafinen käyttöliittymä
- 4 koetta tietokantaan
- Toimintoihin liittyvät testit

#### **Iteraatio 2:**

- Kyky poistaa kokeita
- Kyky muokata kokeita
- Hakutoimintoja
- Toimintoihin liittyvät testit

#### **Iteraatio 3:**

- Kyky ylläpitäjälle lisätä käyttäjiä, kyky selata käyttäjien tietoja, poistaa käyttäjiä ja vaihtaa oma salasana.
- Toimintojen erottaminen käyttäjäroolin pohjalta
- Graafinen käyttöliittymä eri käyttäjäryhmille
- Kirjautumistoiminto
- Tulostustoiminto (graafisessa käyttöliittymässä)
- Loppukäyttäjälle kyky nähdä omat tietonsa ja vaihtaa salasanansa
- Kyky yhdistää kokeita listaksi
- Listojen selailu
- Kokeiden poistaminen listalta
- Toimintoihin liittyvät testit

#### **Iteraatio 4:**

- Testit valmiiksi
- Bugien korjausta
- Dokumentaatiota

#### **Iteraatio 5: viimeistely**

- Käyttöohjeet
- Testaus ja sen raportointi
- Loppudokumentti

### **Ohjelmiston rajoitukset**

Ohjelmistoa voi päivittää vain yksi ihminen kerrallaan. Ohjelmistoa voi käyttää vain yksi ihminen kerrallaan.

## 3. Käyttöohje

### 3.1. Kehitysympäristö ja ohjelmiston rajoitteet

Ohjelmisto on kirjoitettu Netbeans-ohjelman versiolla 8.2 ja javan versiolla 1.8.0\_181. Mongon java-ajuri on versiota 3.10.1. ja Morprian versio on 1.4.0. Tietokanta MongoDB:n versio on 4.0 Community Edition. Ohjelmiston teossa käytetyssä tietokoneessa on MacOS Mojave käyttöliittymä. Ohjelmistoa ei ole testattu kattavasti muilla käyttöliittymillä.

Ohjelmisto on myös testattu ja rakennettu niin, että sillä on kerrallaan vain yksi käyttäjä.

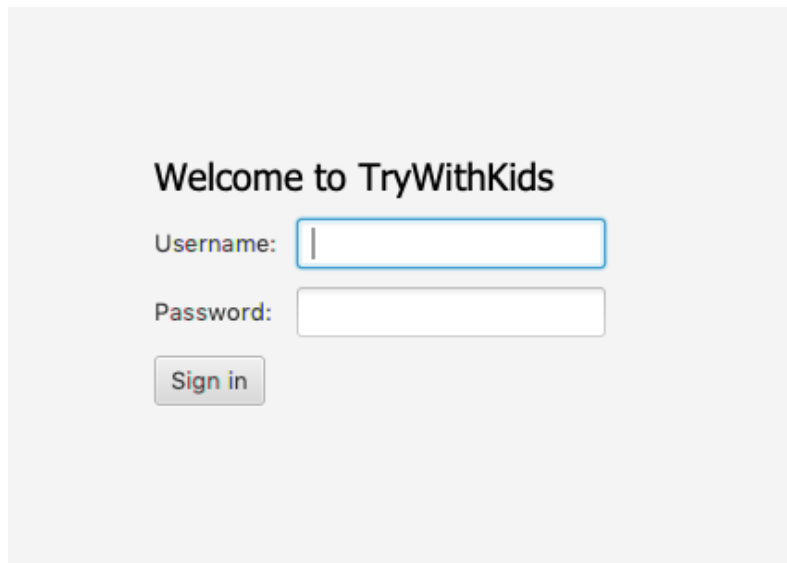
### 3.2. Asennusohje

1. Lataa koneellesi TryWithKids.zip -tiedosto.
2. Pura tiedosto koneellasi. Käyttöjärjestelmällä on yleensä keinot purkaa zip-tiedostot ilman erillisiä ohjelmien latauksia. Macintosh-koneella zip-tiedosto purkautuu automaattisesti, kun sitä yritetään avata (tuplaklikkaus kuvakkeen kohdalla).
3. Avaa terminal-ohjelmisto (macOs-käyttäjät) tai vastaava ohjelmisto muilla käyttöliittymillä.
4. Siirry TryWithKids-kansioon.
5. Siirry target-nimiseen kansioon
6. Kirjoita komentokehoitteeseen seuraava käsky: `java -jar TryWithKids-1.0-SNAPSHOT-shaded.jar` ja paina enteriä
7. Ohjelma käynnistyy.

Jos et osaa käyttää terminal-ohjelmistoa, hyvä ohje siellä toimimiseen ja liikkumiseen löytyy osoitteesta: <https://macpaw.com/how-to/use-terminal-on-mac>

### 3.3. Käyttöohje

Kun ohjelma käynnistyy, ensimmäisenä on edessä kirjautuminen:



The login screen has a light gray background. At the top, it says "Welcome to TryWithKids" in a bold, black font. Below this, there are two input fields: "Username:" followed by a text box with a blue border, and "Password:" followed by a text box with a gray border. Below the password field is a "Sign in" button with a gray background and black text.

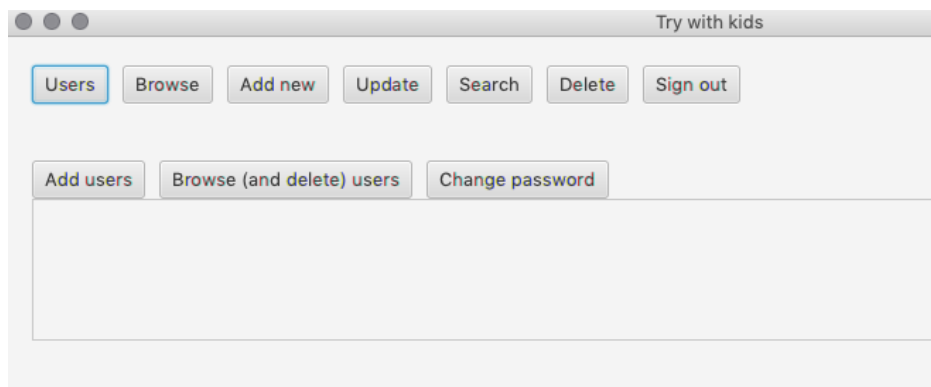
Kuva 1: Login-ikkuna

Ohjelmistossa on valmiina kaksi käyttäjäprofiilia, jolla ohjelman toimintaa voi testata. Näistä laajemmat oikeudet omaa ylläpitäjä, jonka kirjautumistiedot ovat:

Username : maintenance

Password: main\_auth123

Täytä nämä tiedot ja paina "sign in". Seuraavaksi näet tämän näkymän



The "Users" screen is titled "Try with kids" in the top right corner. It features a row of buttons: "Users" (highlighted with a blue border), "Browse", "Add new", "Update", "Search", "Delete", and "Sign out". Below this row are three more buttons: "Add users", "Browse (and delete) users", and "Change password". The bottom half of the screen is a large, empty white rectangular area.

Kuva 2: Users-näkymä

#### USERS

Tässä on automaattisesti auki Users-valikko, jossa voit lisätä käyttäjiä (add users), selata ja poistaa käyttäjiä (Browse (and delete) users) sekä vaihtaa salasanasasi (Change password). Kuhunkin näistä toiminnoista pääset painamalla nappia, jossa lukee haluamasi toiminto.

Lisää käyttäjät-napin painalluksesta aukeaa seuraava ikkuna:

Try with kids

Users Browse Add new Update Search Delete Sign out

Add users Browse (and delete) users Change password

Please fill all boxes below

Username:

Compulsory: please write your username

Password:

Compulsory: please choose a password. Must be over 8 characters long

Repeat password:

Compulsory: please rewrite password

Compulsory: Please select a role

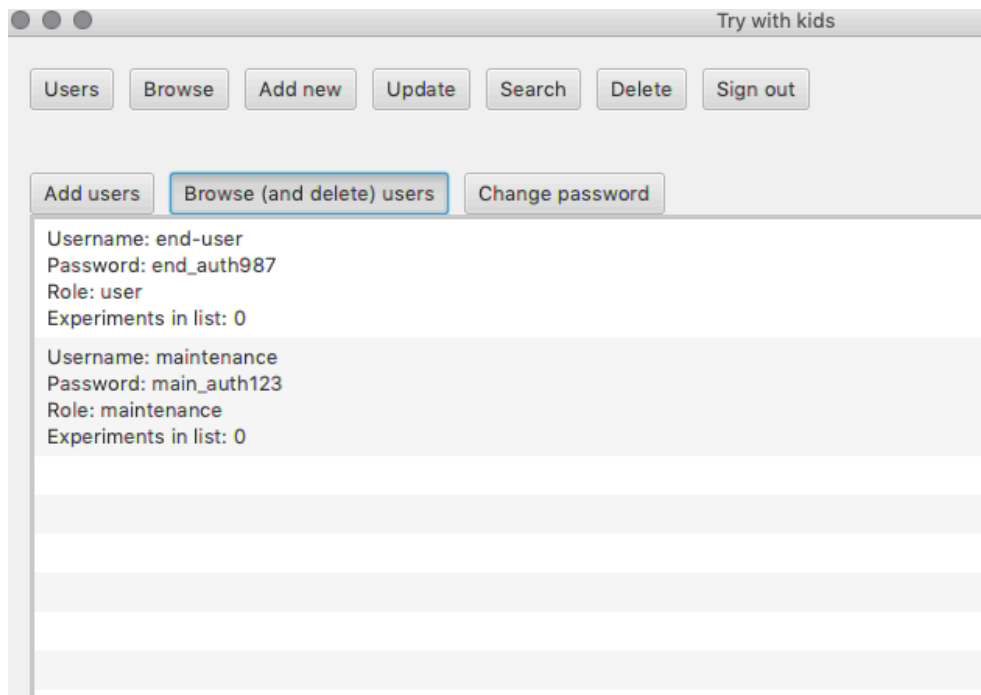
Maintenance User

SAVE

Kuva 3: Add Users-näkymä

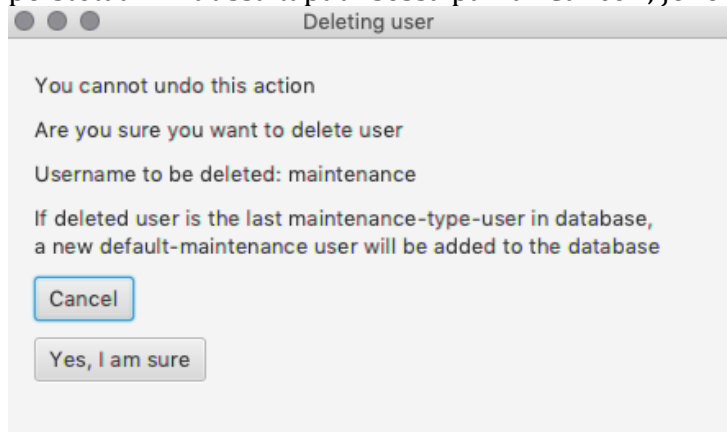
Tässä ensin kirjataan uudelle käyttäjälle käyttäjätunnus. Sen pitää olla uniikki eli tietokannassa ei voi olla kahta samannimistä käyttäjää, mutta siitä sinun käyttäjänä ei tarvitse huolehtia. Voit kurkata Browse (and delete) users välilehdeltä tietokannassa jo olevien käyttäjien nimet, tai voit kokeilla haluamaasi käyttäjätunnusta. Jos sellainen jo tietokannassa on, saat virheilmoituksen ja voit kokeilla toista käyttäjätunnusta. Käyttäjätunnuksen lisäksi sinun pitää kirjoittaa salasanasi sille osoitettuun kohtaan käyttäjätunnuksen alapuolella sekä kirjoittaa salasana uudelleen vielä. Tällä varmistetaan, että salasanan kirjoitusprosessissa ei tullut kirjoitusvirheitä. Lopuksi valitset, onko luotava käyttäjä ylläpitäjä (maintenance), jolla on laajat käyttöoikeudet luoda, muokata ja poistaa käyttäjiä ja kokeita, vai käyttäjä (user), joka saa selata kokeita, lisätä niitä listalleen, poistaa niitä listaltaan ja nähdä omat tietonsa ja vaihtaa salasanansa.

Kun nämä kohdat on valittu, voit painaa "SAVE". Tällöin käyttäjä tallentuu tietokantaan. Voit käydä katsomassa tiedot Browse (and delete) users-nappia painamalla. Tällöin avautuu seuraavanlainen ikkuna:



Kuva 4: Browse (and delete) users –näkymä

Tästä näkee kaikki käyttäjät, heidän käyttäjätunnuksensa, salasansa, roolinsa sekä kuinka monta koetta he ovat lisänneet omalle henkilökohtaiselle listalleen. Mikäli haluat poistaa käyttäjän listalta, klikkaa sen käyttäjän tietoja. Tällöin avautuu varmistus-ikkuna (alla), jossa on poistettavan käyttäjän tiedot. Mikäli haluat varmasti poistaa käyttäjän, paina "Yes, I am sure", jolloin käyttäjä poistetaan. Muussa tapauksessa paina "Cancel", jolloin mitään ei tapahdu.



Kuva 5: Varmistus-ikkuna käyttäjiä poistettaessa tietokannasta

Jokaisen listalla olevan käyttäjän voi poistaa. Ainoana huomiona on, että jos poistat viimeisen ylläpitäjän (maintenance) listalta, ohjelma lisää automaattisesti oletus-ylläpitäjän ohjelmaan. Tämän kirjautumistiedot löytyvät ylempää. Näin varmistetaan, että ohjelmassa on aina jotkut tiedot, joiden avulla ohjelman sisältöä voi muokata.

Oman salasanan vaihto onnistuu painamalla "Change password". Siinä ensin kirjoitetaan vanha salasana ensimmäiseen laatikkoon. Toiseen laatikkoon kirjoitetaan uusi salasana, jonka pitää olla yli 8 merkkiä pitkä. Tämä uusi salasana kirjoitetaan uudelleen kolmanteen laatikkoon. Tämän jälkeen painetaan



"SAVE". Mikäli vanha salasana täsmää siihen, mikä tietokannassa on, ja kaksi kertaa kirjoitettu uusi salasana on kirjoitettu samalla tavalla molemmilla kerroilla ja on yli 8 merkkiä, salasana tallentuu. Muussa tapauksessa saat virheviestin siihen kohtaan, jossa ongelma oli ja voit kokeilla uudelleen. Esimerkiksi alla vanha salasana ei täsmännyt aiempien tietojen kanssa eivätkä uudet salasanat olleet samanlaisia. Vain oman salasanan voi vaihtaa. Maintenance-tyypin käyttäjäkään ei pysty vaihtamaan toisen käyttäjän salasanaa.

Kuva 6: Salasanan vaihto -näkymä

## BROWSE

Browse-napin alla löytyy lista kaikista ohjelmassa olevista kokeista. Täällä voit valita kahdesta eri valikoimasta. Browse all-nappia painamalla sekä oletusarvoisesti heti näkyvillä on kaikki ohjelmassa olevat kokeet. Tässä valikossa voit klikkaamalla koetta lisätä sen omalle listallesi (Add to my list) tai tulostaa kokeen pdf-tiedostoksi (print experiment) tai peruuttaa valinnan (cancel). Nämä valinnat tulevat mahdollisiksi avautuvassa ikkunassa. Ikkuna avautuu, kun klikkaat listalla olevaa koetta. Mikäli valitset tulostaa kokeen, pdf-tiedosto ei automaattisesti siirry tulostimelle, vaan tallentuu samaan kansioon ohjelman kanssa, josta sen voi tulostaa.

Kuva 7: Browse all – näkymä

Kuva 8: Lisää listalle tai tulosta

Browse own list-napia painamalla näet oman listasi. Siellä voit poistaa kokeen omalta listaltasi klikkaamalla koetta, jonka haluat poistaa. Tämä avaa varmistusikkunan, josta näet, mitä koetta olet listaltasi poistamassa. Jos haluat poistaa kokeen listaltasi, valitse "Delete from my list". Jos et halua, paina "Cancel".

## ADD NEW

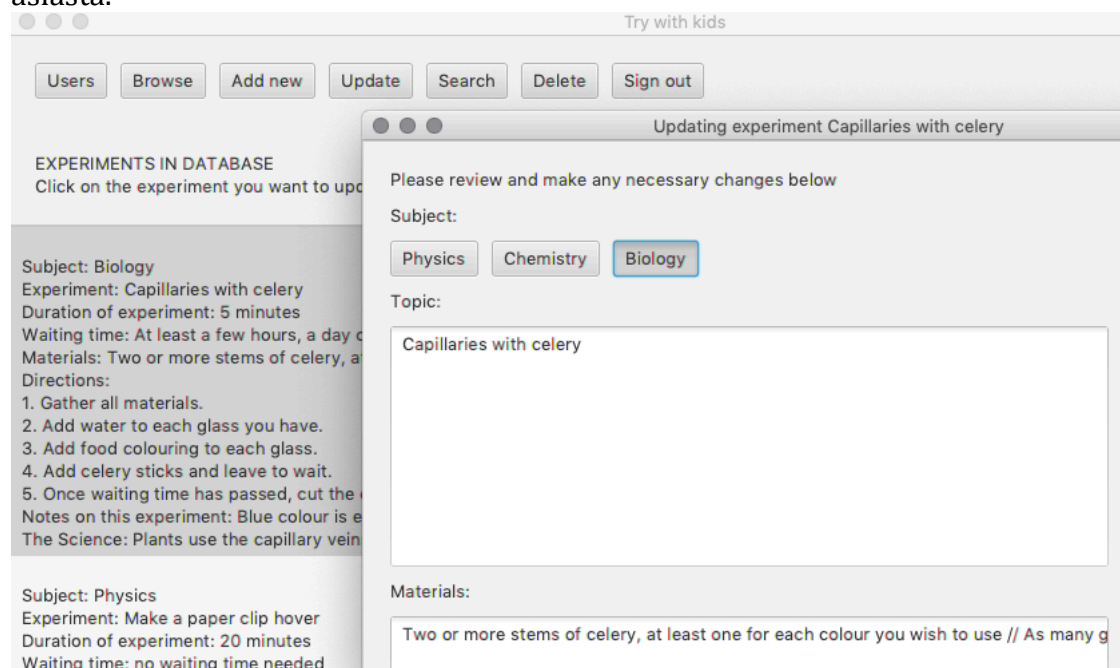
Lisätäksesi uusia kokeita ohjelmaan, valitse ylävalikosta Add new. Se avaa alla olevan näkymän, josta kuvassa on tosin vain osa. Suosittelen, että kirjoitat ohjeet ensin vaikka word-tiedostoon ja kopioit sieltä tähän ohjelmaan. Avoimissa tekstilaatikoissa on sanarajat, joita ei voi ylittää. Topic (otsikko) sanaraja on 50 sanaa, materials sanaraja on 1000 sanaa, waiting period on 200 sanaa. Muiden sanarajoitus on 5 000 sanaa. Näiden ei pitäisi kovin helpolla täyttyä.

Ohjelmassa on tarkat ohjeet siitä, mitkä tiedot ovat pakollisia (compulsory) ja mitä ovat vapaaehtoisia (not compulsory). Koe ei tallennu tietokantaan, jos kaikkia pakollisia tietoja ei ole täytetty. Pakollisia on valita subject (physics, chemistry tai biology), antaa otsikko kokeelle (topic), vaadittavat materiaalit (materials), kokeen kesto (duration) ja kokeen ohjeet (directions). Näiden lisäksi osassa kokeita voi olla odotusaika kokeen teon ja tulosten näkymisen välillä (waiting period). Lisäksi voit halutessasi lisätä huomioita kokeesta (additional notes) sekä selittää tarkemmin, miksi kokeessa tulee tiettyjä tuloksia (explain the science behind the experiment). Kun olet täyttänyt kaikki tiedot huolellisesti, paina "Save to database". Jos kaikki on täytetty oikein, koe tallentuu ja napin alapuolelle tulee teksti "saved to database". Muussa tapauksessa tarkista, onko jotain jäänyt täyttämättä tai esimerkiksi sanarajat ylittyneet. Ohjelma antaa virheilmoituksia tässä tapauksessa. Ohjelma ei kuitenkaan varoita, jos subject tai duration on jäänyt valitsematta. Koe vain ei tällöin tallennu, joten tässä kannattaa olla tarkkana. Jos ei tule "saved to database" -ilmoitusta eikä virheilmoitusta, jompikumpi näistä on jäänyt valitsematta.

Kuva 9: Add Experiment - ikkuna

## UPDATE

Update-nappia ylävalikosta painamalla ensin ikkunassa listataan kaikki tietokannassa olevat kokeet. Valitse näistä se koe, jota haluat päivittää. Tällöin avautuu ikkuna, jossa on kokeen tietokannassa olevat tiedot. Päivitä haluamasi tiedot. Voit päivittää vain haluamasi tiedot. Muut säilyvät muuttumattomina. Subject-osiota eikä duration-osaa voi jättää tyhjäksi. Jos ne jäävät tyhjäksi, kokeeseen yhdistetään edelleen ne subject ja duration-tiedot, jotka olivat alkuperäisessä kokeessa. Myös täällä tarkistetaan tekstien pituus. Topic saa olla maksimissaan 50 sanaa, waiting time maksimissaan 200 sanaa, materials maksimissaan 1000 sanaa ja muut maksimissaan 5000 sanaa. Jos tekstit ovat pidempiä, koe ei tallennu tietokantaan ja liian pitkään kohtaan tulee ilmoitus asiasta.



Try with kids

Users Browse Add new Update Search Delete Sign out

EXPERIMENTS IN DATABASE  
Click on the experiment you want to update

Subject: Biology  
Experiment: Capillaries with celery  
Duration of experiment: 5 minutes  
Waiting time: At least a few hours, a day or more  
Materials: Two or more stems of celery, at least one for each colour you wish to use // As many g  
Directions:  
1. Gather all materials.  
2. Add water to each glass you have.  
3. Add food colouring to each glass.  
4. Add celery sticks and leave to wait.  
5. Once waiting time has passed, cut the sticks and observe the colour change.  
Notes on this experiment: Blue colour is seen in the celery sticks.  
The Science: Plants use the capillary vein to transport water and nutrients.

Subject: Physics  
Experiment: Make a paper clip hover  
Duration of experiment: 20 minutes  
Waiting time: no waiting time needed

Updating experiment Capillaries with celery

Please review and make any necessary changes below

Subject:  
Physics Chemistry Biology

Topic:  
Capillaries with celery

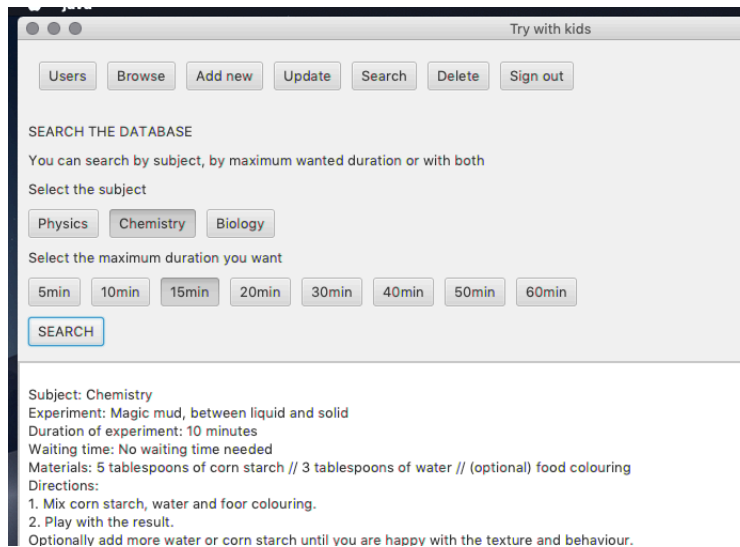
Materials:  
Two or more stems of celery, at least one for each colour you wish to use // As many g

Kuva 10: Update experiment – näkymä

Kun olet tyytyväinen sisältöön, voit painaa "Update info". Jos kaikki on kunnossa, ikkuna sulkeutuu ja voit katsoa uudet tiedot listaikkunasta. Jos et haluakaan päivittää koetta, voit sulkea ikkunan painamalla "Cancel" tai painaalla ruksia vasemmassa yläkulmassa.

## SEARCH

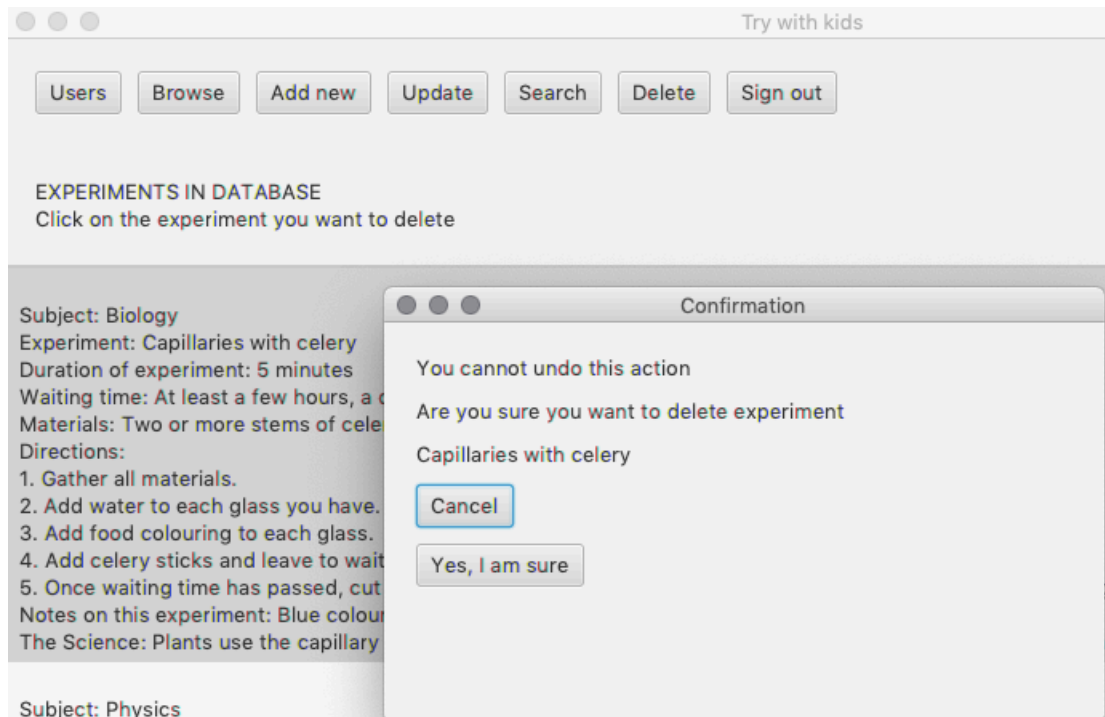
Search-nappia ylävalikosta painamalla voit tehdä hakuja tietokantaan kokeen keston tai aihealueen (physics, chemistry tai biology) suhteen. Voit myös hakea vain toisen hakukriteerin perusteella. Jos haluat valita vain toisen kriteerin, jätä toinen kokonaan valitsematta. Kestossa valitaan kuinka kauan koe saa maksimissaan kestää. Kun painat "SEARCH", tulee napin alle näkymä, jossa on listattuna kaikki ohjelmassa olevat kokeet, jotka täyttävät hakuehdot.



Kuva 11: Search - näkymä

## DELETE

DELETE-nappia painamalla näet jälleen listauksen tietokannassa olevista kokeista. Klikkaamalla koetta listalla avautuu ikkuna, jossa näkyy kokeen tiedot. Jos haluat poistaa kokeen tietokannasta, paina "Yes, I am sure", jolloin koe poistetaan ohjelmasta. Muussa tapauksessa paina "Cancel", jolloin koetta ei poisteta. Ohjelmistosta voi poistaa kaikki kokeet. Jos kuitenkin ohjelma käynnistetään niin, ettei siinä ole yhtään koetta, latautuu tietokantaan automaattisesti neljä ohjelmistossa alun perin ollutta koetta uudelleen.



Kuva 12: Delete experiment – näkymä sekä varmistusikkuna

## SIGN OUT

Sign out-nappia painamalla käyttäjä kirjautuu ulos ohjelmasta ja näkymäksi tulee jälleen alun kirjautuminen.

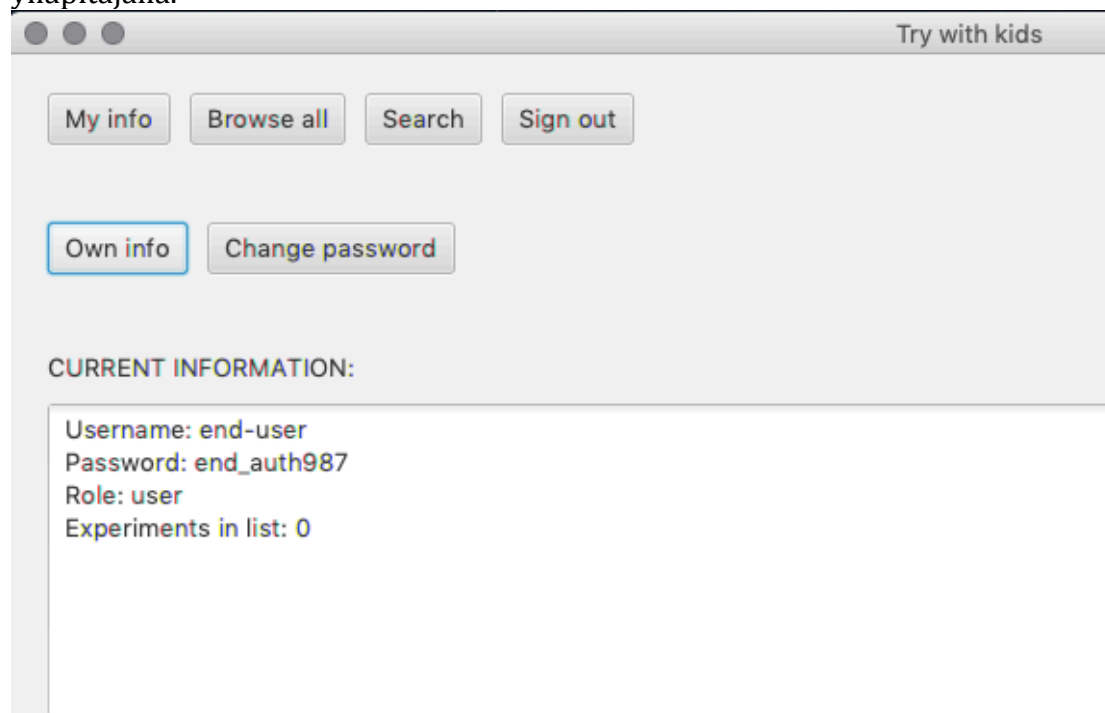
## USER - own info

Kirjaututaan seuraavaksi peruskäyttäjän (lapsen, oppilaan) oikeuksilla. Tietokannassa automaattisesti on käyttäjä, jonka tiedot ovat:

Username: end-user

Password: end\_auth987

Kirjautumalla näillä tiedoilla avautuu alla oleva näkymä ilman "Current information"-osiota. Sen saa näkyviin painamalla "Own info". Eli oppilas/lapsi voi tarkistaa omat tietonsa. Lisäksi lapsi pystyy vaihtamaan salasanaansa. Emme käy sitä uudelleen läpi, sillä käyttöliittymä on niiltä osin samanlainen kuin ylläpitäjällä.



Kuva 13: Own info - näkymä

Myös Browse all, Search ja Sign out -nappien toiminnot ovat identtisiä ylläpitäjän toimintojen suhteen, joten emme käy niitä uudelleen läpi. Nyt osaat käyttää ohjelmistoa. Suosittelemme kokeilemaan ja testailemaan ohjelmaa kaikin mahdollisin keinoin, jotta se tulee tutuksi ja voit opettaa sen keskeiset peruskäyttäjän toiminnot myös lapsellesi / oppilaillesi.

Ohjelman pyörittäminen loppuu, kun suljet kaikki ohjelmaan liittyvät ikkunat vasemmassa yläkulmassa olevasta ruksista.

### 3.4. Sananen turvallisuudesta käyttäjille

Ohjelmiston mukana tulee tosiaan kaksi oletuskäyttäjää. Yksi näistä omaa ylläpitäjän oikeudet ja toinen loppukäyttäjän oikeudet. Näiden avulla on mahdollista esimerkiksi testata ohjelman toimintaa. Tietoturvan kannalta kuitenkin suosittelen poistamaan nämä käyttäjät ja luomaan uudet, jolloin

näiden salasanat eivät ole muiden kuin kyseisten käyttäjien ja tuota nimenomaista ohjelmaversiota käyttävien ylläpitäjien tiedossa.

Suosittelena erittäin vahvasti, että tälle ohjelmistolle luodaan oma salasana, jota ei käytetä missään muualla. Suositus johtuu erityisesti siitä, että salasanat tallennetaan ohjelmistossa tekstinä eikä niitä salata millään tavalla. Lisäksi ylläpitäjä näkee kaikkien ohjelmiston käyttäjien salasanat. Tämä on mielekästä siinä mielessä, että vanhempi tai opettaja pystyy auttamaan lapsia kirjautumaan ohjelmaan, mutta tietoturvallisuuden kannalta salasana ei siis tällöin pysy salassa. Siksi on olennaista, että ohjelmassa käytetään salasanaa, joka on vain tässä ohjelmassa käytössä.

### 3.5. Ohjelmiston rajoitteet, virhetilanteet ja bugit ohjelmassa

Ohjelmiston rajoitteena on yksi käyttäjä kerrallaan. Ohjelmiston tietokantaratkaisua ei ole testattu siten, että ohjelmistolla olisi monta samanaikaista käyttäjää.

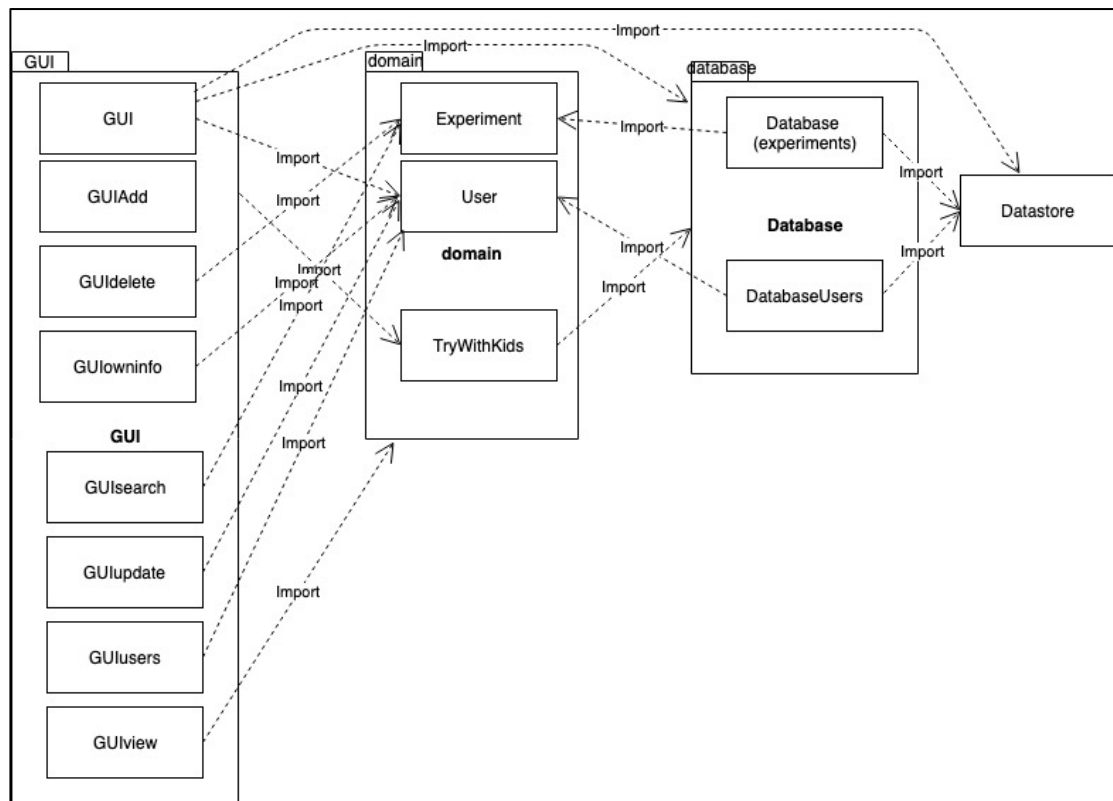
Ohjelmistoa testatessa ei löytynyt virheitä, jotka johtaisivat ohjelman kaatumiseen. Täysin virheetön ohjelma ei kuitenkaan ole. Ensinnäkin lisättäessä uutta koetta ohjelmaan, jos subject tai duration puuttuu, ei ohjelma anna kohdennettua virheilmoitusta. Tätä on pyritty korjaamaan käyttäjäohjeistuksella, mutta tilanne ei ole täysin korjaantunut. Toisena haastena kävi ilmi ohjelmaa testatessa, että virheilmoitukset jäävät ikkunaan, kun esimerkiksi testasin, mistä kaikesta saa virheilmoituksen. Eli mikäli käyttäjä tekee monta virhettä esimerkiksi koetta lisätessään, on ikkunassa nähtävissä myös edellisten virheiden ilmoitukset. Tilanne korjaantuu, jos käyttäjä käy välillä toisessa näkymässä. Kolmas virhe on enemmän kosmeettinen ongelma eli listView-ratkaisuissa, josta näkyy kokeiden sisältö, pitää scrollata myös horisontaalisesti eikä teksti rajaudu kauniisti näkymään. Tähän ei kuitenkaan käytettävissä olleessa ajassa löytynyt toimivaa ratkaisua. Neljäs bugi on ilmoitus, joka tulostuu outputtiin osassa tietokantatoimintoja. Tällöin tulostuu:

```
2019-04-21 09:45:09.629 java[6696:29499020] unrecognized type is
4294967295
```

```
2019-04-21 09:45:09.630 java[6696:29499020] *** Assertion failure in -
[NSEvent_initWithCGEvent:eventRef:],
/BuildRoot/Library/Caches/com.apple.xbs/Sources/AppKit/AppKit-
1671.20.108/AppKit.subproj/NSEvent.m:1969
```

Tämä ei kuitenkaan ensinnäkään vaikuta tietokannan toimintaan eli kaikki toiminnot toimivat halutusti. Tämä ei myöskään näy käyttäjälle.

## 4. Ohjelmiston rakenteen ja toiminnan kuvaus yleisellä tasolla



Kuva 14: Pakkauskaavio

Ohjelmassa on kerrosarkkitehtuuri. Alimmaisella tasolla on MongoDB-dokumenttitietokanta, jossa on kaksi kokoelmaa, jotka ovat käyttäjät (user) ja kokeet (experiment). Näitä kokoelmia käyttää pakkaus database, jossa on luokat database, joka käsittelee kokeita, sekä databaseUsers, joka käsittelee käyttäjiä. Tiedot tallennetaan ohjelmassa tietokantaan, joka on myös ainoa tiedon pitkäkestoisempi säilytyspaikka. Tietokannasta tietoa haetaan tarpeen mukaan ohjelmaan katsottavaksi, muokattavaksi tai poistettavaksi.

Tämän tietokantakerroksen yläpuolella on sovelluskerros (domain), jossa on luokat user ja experiment, jotka kuvaavat sovelluksen keskeisiä käsitteitä. Sovelluskerroksella on myös luokka TryWithKids, jossa on sovelluksen logiikka eli kaikki ne metodit, jotka saavat tietoa käyttöliittymältä ja ohjaavat tietoa sopivalle tietokantakokoelmalle käyttävälle luokalle.

Käyttöliittymäkerroksella poikkeuksellinen luokka on GUI, joka on vastuussa ohjelman käynnistämisestä, kirjautumisesta ohjelmaan sekä myös käyttäjän tyyppiin liittyvien näkymien hakemisesta. Tästä syystä tämä luokka saa käyttöönsä alemmat luokat. Kaikki käyttöliittymäluokat saavat käyttöönsä sovelluslogiikan (TryWithKids) sekä sisältönsä puolesta osa saa käyttöönsä luokan Experiment ja ne luokat, jotka tarvitsevat tietoa kirjautuneesta käyttäjästä, saavat käyttöönsä luokan User. Käyttöliittymäluokka GUI hakee käyttäjätyyppiin liittyvät näkymät ja näyttää ne käyttäjälle. Loppukäyttäjä (end-user) saa käyttöönsä näkymät GULowninfo, GUIsearch ja GUIview. Ylläpitäjä

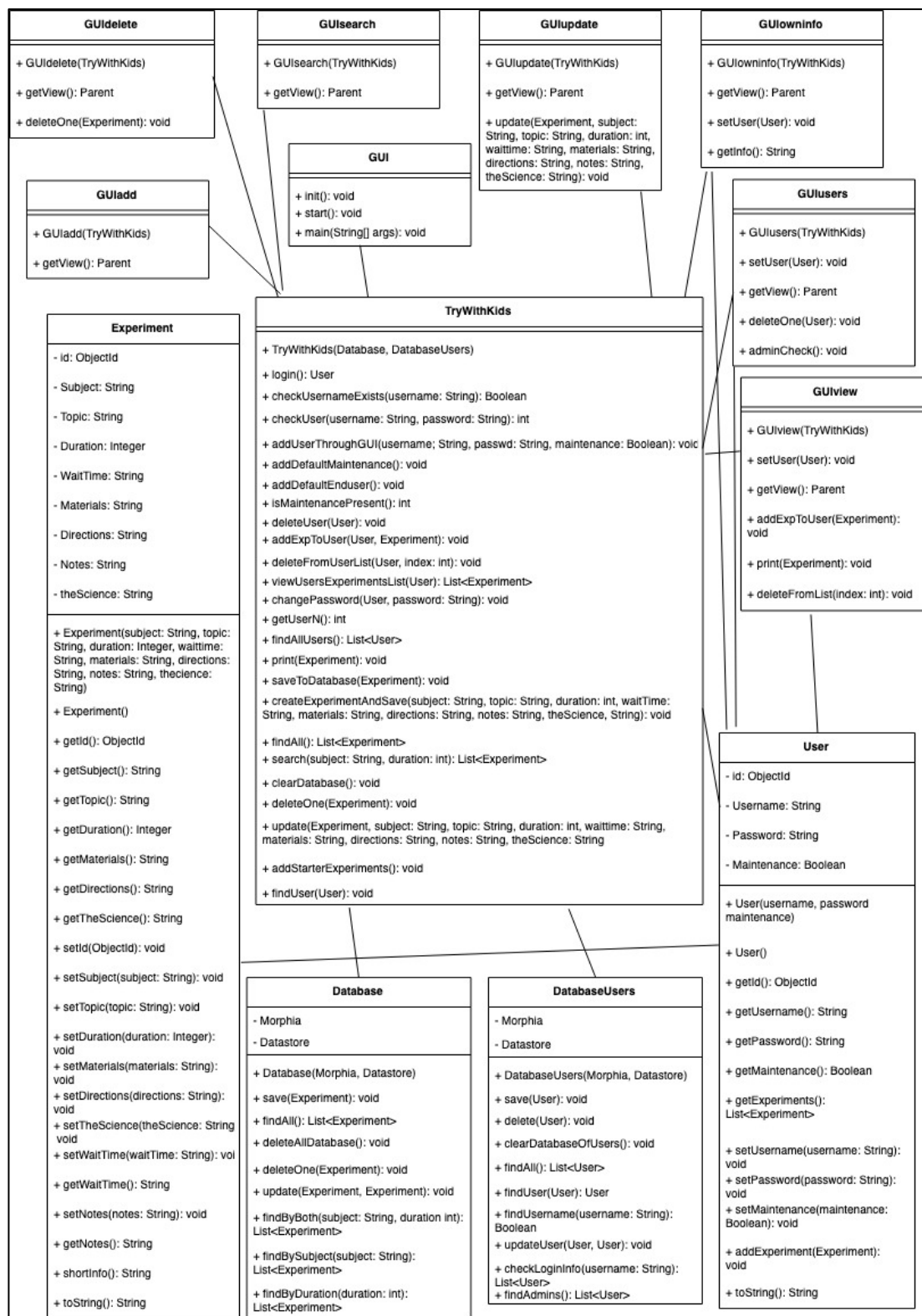
(maintenance/admin) saa käyttöönsä kaikki muut paitsi GUIowninfo. Ylläpitäjä saa käyttöönsä vastaavat tiedot ja toiminnallisuudet luokasta GUIusers, jossa ylläpitäjä näkee omien tietojensa lisäksi myös muiden tiedot.

Tietoa syötetään ohjelmaan graafisen käyttöliittymän avulla. Suurin osa syötteistä tarkastetaan suoraan käyttöliittymässä. Esimerkiksi kokeita lisätessä tarkistetaan käyttöliittymässä, ettei tekstimäärä ole liian suuri ja että pakolliset kentät on täytetty. Salasanaa vaihtaessa esimerkiksi tarkastetaan, että uusi salasana on riittävän pitkä ja että uudelleen kirjoitettuna salasana täsmää.

Kun tiedot on tarkistettu käyttöliittymässä siltä osin, kun tiedot siellä tarkastetaan, ne välitetään sovelluslogiikalle, jossa on myös jotain tietoon liittyviä tarkistuksia. Sovelluslogiikassa tarkastetaan esimerkiksi kokeen tietoa päivitettäessä se, että hakutoiminnalle olennaiset tiedot on täytetty, tai käyttäjää lisättäessä, että samannimistä käyttäjää ei ole jo tietokannassa. Myös kirjautuminen hoidetaan sovelluslogiikan puolella. Sovelluslogiikasta tieto välitetään tietokantaa käsitteleville luokille aina sen mukaan, koskeeko tieto käyttäjiä vai kokeita. Tietokantaa käsittelevät luokat välittävät tiedot kumpikin omaan tietokannan kokoelmaan.



## 5. Tarkempi kuvaus sovelluslogiikasta



Kuva 15: Luokkakaavio sisältäen muuttujat ja metodit

Ohjelma käynnistetään graafisen käyttöliittymän puolella GUI-luokasta, jossa init-metodissa käynnistetään Morphia, joka kartoittaa pakkaukset domain ja gui. Sen jälkeen luodaan Datastore, joka luo uuden tietokannan ("experiments").

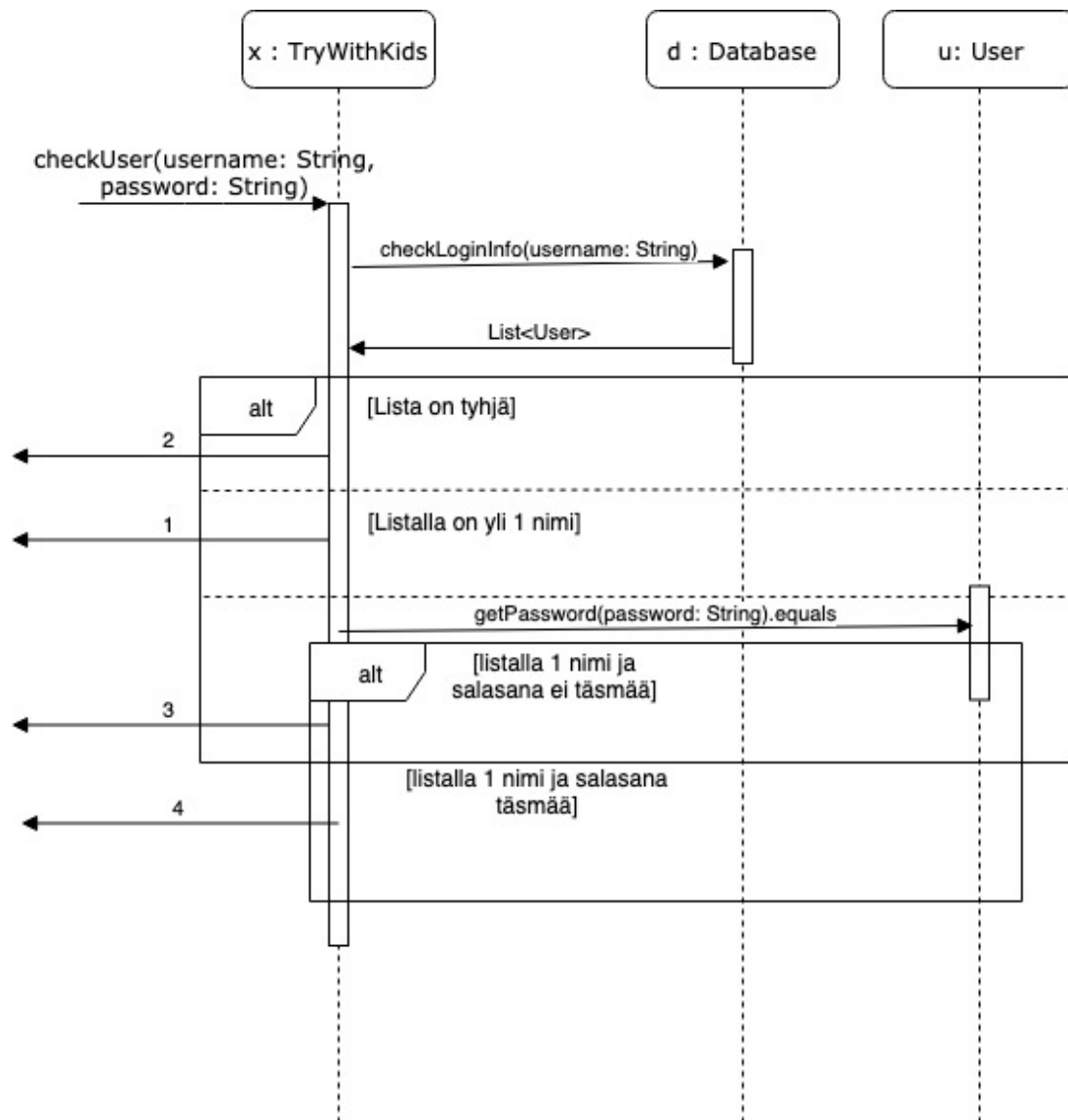
Tämän jälkeen luodaan tietokannan kanssa keskustelevat luokat Database ja DatabaseUsers sekä sovelluslogiikka TryWithKids, jolle annetaan tietokannan kanssa keskustelevat luokat konstruktorin parametreinä. Tämän jälkeen käynnistuksen yhteydessä tarkastetaan käyttäjien määrä yleensä (trywithkids.getUserN()). Mikäli getUserN()-metodi palauttaa arvon 0, hyödynnetään trywithkids.addDefaultMaintenance() ja trywithkids.addDefaultEnduser()-metodeja ja lisätään yksi maintenance-tason käyttäjä ja yksi peruskäyttäjä ohjelmistoon. Molemmissa näissä metodeissa luodaan ensin käyttäjä ja sen jälkeen käyttäjä tallennetaan tietokantaan.

Lisäksi tarkistetaan, onko tietokannassa yhtään maintenance-tason käyttäjää. Tässä hyödynnetään metodia trywithkids.isMaintenancePresent(). Tässä metodissa trywithkids hakee DatabaseUsers-luokan intanssilta tietoa, kuinka monta maintenance-tason käyttäjää tietokannassa on userDatabase.findAdmins()-metodin avulla, jolloin tietokantaan tehdään kysely, jossa haetaan kaikki ne käyttäjät, joiden maintenance-luokamuuttujassa on totuusarvo true ja palautetaan lista sovelluslogiikalle. Sovelluslogiikassa tarkistetaan, onko listan koko alle yksi. Jos näin on, ajetaan aiemmin jo käsitelty metodi addDefaultMaintenance(), jolla lisätään yksi maintenance-tason käyttäjä tietokantaan. Metodi palauttaa maintenance-tyypin käyttäjien määrän. Tätä tietoa ei käytetä muuten ohjelmistossa, mutta testauksessa tietoa tarvitaan.

Tällä varmistetaan, että ohjelmaa pystyy käyttämään, testaamaan ja sisältöä aina muokkaamaan. Lisäksi käynnistyksessä arvioidaan, onko tietokanta kokeiden suhteen tyhjä metodilla trywithkids.databaseSize(), joka kutsuu findAll()-metodia ja sen jälkeen palauttaa findAll()-metodin palauttaman listan koon. Jos koko on 0 eli tietokannassa ei ole lainkaan kokeita, lisätään sovelluslogiikasta neljä koetta tietokantaan trywithkids.addStarterExperiments()-metodilla, jossa luodaan neljä koetta kaikkine yksityiskohtineen ja sen jälkeen tallennetaan yksitellen tietokantaan database.saveToDatabase(experiment) -metodia hyödyntäen. Tämän init-metodin jälkeen käynnistetään käyttöliittymä start-metodilla.

Start-metodissa käyttöliittymässä ensin aukeaa kirjautumis-sivu. Tässä kohtaa hyödynnetään sovelluslogiikan checkUser(username: String, password: String)-metodia, jolle annetaan parametreiksi käyttöliittymään kirjoitettu käyttäjätunnus sekä salasana. Metodissa checkUser(username: String, password: String) ensin haetaan tietokannasta DatabaseUsers-luokan avulla kaikki ne käyttäjät, joilla on käyttöliittymään kirjattu käyttäjätunnus metodilla userDatabase.checkLoginInfo(username: String). Tämän jälkeen tarkistetaan listan koko. Jos lista on tyhjä, ei tietokannassa ole sen nimistä käyttäjää ja käyttöliittymälle palautetaan arvo 2, joka käyttöliittymässä muunnetaan virheilmoituksesi. Mikäli saman nimisiä käyttäjiä on useita, palautetaan arvo 1. Tämä arvo antaa virheilmoituksen, mutta ei käyttöliittymään vaan outputin puolelle, sillä tällöin on virhe uuden käyttäjän lisäyksessä eikä metodi käyttäjänimien ainutkertaisuuteen ole toiminut. Mikäli käyttäjätunnus löytyy tietokannasta, mutta salasana ei täsmää, palautetaan käyttöliittymään arvo 3, jonka käyttöliittymä muuntaa asiaan sopivaksi virheilmoitukseksi. Muussa tapauksessa palautetaan arvo 4 ja käyttäjä pystyy kirjautumaan sisään. Tällöin

annetaan tämän käyttäjän tiedot luokkamuuttujalle `this.user`. Kun käyttöliittymä saa palautusarvona 4, haetaan käyttöliittymän luokkamuuttujaan `this.user` sen käyttäjän tiedot, jotka juuri tallennettiin myös sovelluslogiikan puolelle. Haku toteutetaan sovelluslogiikan `trywithkids.login()` -metodilla. Tämän jälkeen käyttäjän tiedot välitetään myös niille käyttöliittymän osille, jotka näitä tietoja tarvitsevat. Esimerkiksi tätä tietoa hyödynnetään, kun vaihdetaan salasanaa, kun tallennetaan käyttäjän listalle kokeita ja kun selataan omia tietoja.



Kuva 16. Sekvenssikaavio metodista `checkUser()`

Jos kirjautunut käyttäjä on peruskäyttäjä (end-user), näkee hän omat tietonsa `GUIowninfo`-luokan avulla. Täällä käyttäjän tiedot haetaan tietokannasta `trywithkids.findUser(this.user)`-metodin avulla. Tässä välitetään kirjautumisen yhteydessä saadut tiedot käyttäjästä sovelluslogiikan kautta `userDatabase.findUser()`-metodille, jossa tietokannasta haetaan käyttäjä, jolla on sama id ja palautetaan se sovelluslogiikalle, joka palauttaa tiedot käyttöliittymälle. Tällä tavalla saadaan ajantasaiset tiedot esimerkiksi salasanoista ja käyttäjän omalla listalla olevista kokeista.

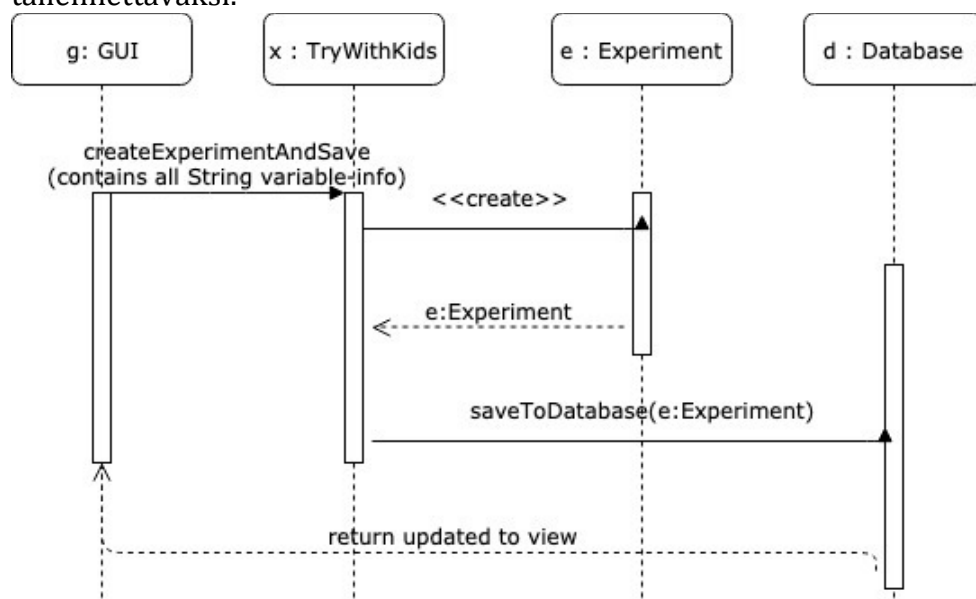
Omien tietojen yhteydessä käyttäjä kykenee myös vaihtamaan salasansa. Tällöin käyttöliittymässä tarkistetaan, että uusi salasana on yli 8 merkkiä pitkä ja että kahdesti kirjoitettu uusi salasana on kirjoitettu molemmilla kerroilla samalla tavalla ja että salasana on sama kuin ohjelman sen hetkellä käyttäjällä, jonka tiedot on luokkamuuttujalla `this.user`. Mikäli nämä tarkistukset läpäistään, välitetään sovelluslogiikalle tieto muuttuvasta salasanasta `trywithkids.changePassword(user: User, newPW: String)`-metodin avulla, jossa sovelluslogiikalle välitetään tieto käyttäjästä sekä tämän uusi salasana. Sovelluslogiikan puolella ensin haetaan tietokannasta kyseinen käyttäjä `this.userDatabase.findUser(userfromGUI: User)`-metodilla. Sen jälkeen tietokannasta haetulle käyttäjälle päivitetään uusi salasana `.setPassword(newPassWd: String)`-metodilla. Tämän jälkeen päivitetty käyttäjä annetaan tietokannalle `userDatabase.updateUser(vanhatTiedot: User, uudetTiedot: User)`-metodin avulla. `DatabaseUsers`-luokassa kyseinen metodi ensin deletoi tietokannasta käyttäjän vanhat tiedot `datastore.delete(User)` -metodin avulla ja sen jälkeen tallentaa uudet tiedot tietokantaan `datastore.save(User)` -metodin avulla.

Myös maintenance-tyypin käyttäjät hyödyntävät näitä samoja metodeja päivittääkseen salasansa. Tämän tyypin käyttäjät kykenevät myös näkemään muiden käyttäjien tietoja `GUIusers`-luokan avulla. Omien tietojensa lisäksi he kykenevät näkemään kaikkien muidenkin tiedot `trywithkids.findAllUsers()`-metodin avulla. Tällöin sovelluslogiikka pyytää `userDatabase.findAll()`-metodilla kaikkia käyttäjätietoja tietokannasta. Nämä haetaan tietokannasta ja palautetaan listamuodossa sovelluslogiikalle, joka välittää tiedot käyttöliittymään. Tältä listalta maintenance-tyypin käyttäjä pystyy poistamaan ohjelmasta käyttäjän. Tällöin ensin varmistetaan käyttöliittymän puolella, että käyttäjä on varma asiasta ja että poistettava käyttäjä on juuri oikea. Tämän jälkeen ensinnäkin välitetään poistettavan käyttäjän tiedot sovelluslogiikalle. Se tehdään hyödyntämällä `trywithkids.deleteUser(user)`-metodia. Tämä metodi kutsuu `userDatabase.delete(user)`-metodia, joka poistaa käyttäjän tietokannasta. Sen jälkeen poistetun käyttäjän tiedot poistetaan käyttöliittymästä. Sitten tarkistetaan käyttöliittymän puolella, onko poistettava käyttäjä maintenance-tyypin käyttäjä. Jos on, varmistetaan yhden maintenance-tason käyttäjän olemassaolo ohjelmistossa `trywithkids.isMaintenancePresent()`-metodin avulla, joka on jo aiemmin käsitelty. Maintenance-tason käyttäjät kykenevät myös lisäämään käyttäjiä ohjelmistoon. Tällöin ensin tarkistetaan käyttöliittymän puolella, että kaikki oleelliset kentät on täydennetty. Sen jälkeen `trywithkids.checkUsernameExists(username: String)`-metodilla tarkistetaan, onko käyttäjätunnus vapaa. Tässä metodissa `userDatabase.findUsername(username: String)`-metodilla haetaan lista halutun käyttäjätunnuksen omaavista käyttäjistä tietokannassa. Jos sellaisia löytyy, palautetaan totuusarvo `true`. Muuten palautetaan `false`. Tämä tieto välitetään käyttöliittymään, jossa totuusarvon ollessa `true` annetaan käyttäjälle virheviesti varatusta käyttäjätunnuksesta. Mikäli salasana on oikean mittainen, uudet salasanat täsmäävät, vanha salasana täsmää tietokannassa olevan kanssa, kaikki kentät on täytetty ja käyttäjätunnus on vapaa, luodaan uusi käyttäjä `trywithkids.addUserThroughGUI(username: String, username: String, maintenance: Boolean)` -metodilla, jossa välittää haluttu käyttäjätunnus,

salasana sekä admin-status trywithkids-luokalle. Täällä näiden tietojen avulla luodaan uusi käyttäjä, joka sitten tallennetaan tietokantaan kutsumalla `userDatabase.save(user)`-metodia.

Näiden käyttäjään liittyvien metodien lisäksi `DatabaseUser`-luokan instanssissa on menetelmä tyhjentää tietokanta käyttäjistä. Metodi `clearDatabaseOfUsers()` hakee kaikki käyttäjät tietokannasta ja poistaa jokaisen kerralla. Metodia hyödynnetään testauksen puolella sekä ohjelman tekoprosessissa myös tilanteissa, jossa piti varmistaa, ettei tietokannassa ollut käyttäjiä.

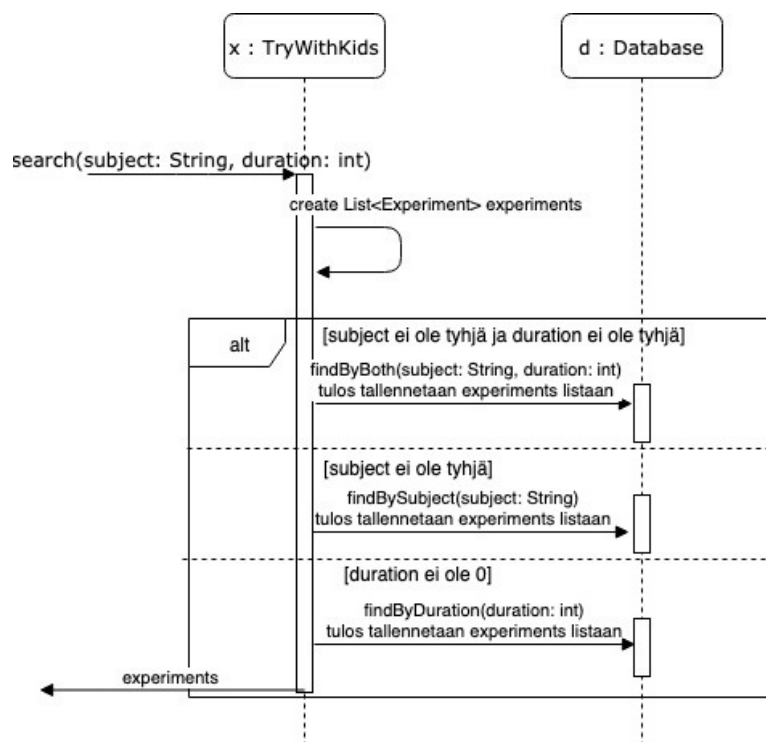
Maintenance-tyypin käyttäjä kykenee lisäämään käyttäjien lisäksi myös kokeita käyttöliittymään `GUIadd`-luokan avulla, jossa ensin varmistetaan syötteiden oikeellisuus. Ensinnäkin tarkistetaan, että kaikki pakolliset tiedot on annettu. Toiseksi tarkistetaan, ettei avoimiin tekstikenttiin ole lisätty liian pitkiä tekstejä, joiden tallentamisessa tietokantaan voisi tulla ongelmia. Kun nämä virhetilanteet on suljettu pois, hyödynnetään `trywithkids.createExperimentAndSave(subject: String, topic: String, materials: String, duration: integer, waitTime: String, directions: String, notes: String, theScience: String)`-metodia, jolle annetaan käyttöliittymästä saadut tiedot muuttujina. Metodissa annetut muuttujat annetaan `Experiment`-luokan konstruktorille ja luodaan niiden avulla uusi koe, joka annetaan sitten `database.save(experiment)`-metodilla tietokannalle tallennettavaksi.



Kuva 17: Sekvenssikaavio metodista `createExperimentAndSave()`

Kokeita voi poistaa `GUIdelete`-luokan avulla. Tällöin ensin haetaan `trywithkids.findAll()`-metodilla kaikki tietokannassa olevat kokeet, jotka näytetään käyttäjälle `ListView`n avulla. Kun käyttäjä klikkaa jotain koetta listalta, saa käyttäjä varmistusikkunan, jossa on poistettavan kokeen tiedot. Tästä pystyy peruuttamaan pois `cancel`-nappia painamalla tai sitten voi jatkaa kokeen poistamista. Mikäli koe halutaan poistaa, tehdään tämä `trywithkids.deleteOne(experiment)`-metodin avulla, joka kutsuu `database.deleteOne(experiment)`-metodia, joka poistaa kokeen tietokannasta.

Molempien tyyppiset käyttäjät kykenevät etsimään kokeita tietokannasta. Tällöin käyttöliittymässä valitaan kahdesta etsintäkriteeristä eli aihepiiristä ja kokeen kesto. Käyttäjä voi valita toisen tai molemmat. Haku tehdään trywithkids.search(subjectToggle, durationToggle)-metodin avulla. Tässä metodissa ensin luodaan tyhjä ArrayList. Tämän jälkeen metodissa tarkistetaan, mitä ollaan hakemassa. Mikäli molempia kriteereitä on käytetty, hyödynnetään database.findByBoth(subject: String, duration: int)-metodia, jolloin tietokannasta haetaan kaikki ne kokeet, jotka ovat tietystä aihepiiristä sekä joiden kesto on korkeintaan parametrina olevan luvun pituisia. Nämä tallennetaan aiemmin luodulle tyhjälle listalle ja palautetaan lista käyttöliittymälle. Mikäli haetaan vain aihepiirin perusteella, hyödynnetään database.findBySubject(subject: String)-metodia, jolla etsitään kaikki tietyn aihepiirin kokeet tietokannasta ja palautetaan lista sovelluslogikalle ja tallennetaan se tyhjälle ArrayListalle, joka palautetaan käyttöliittymälle. Mikäli haetaan vain keston perusteella, hyödynnetään database.findByDuration(duration: int)-metodia, joka etsii kaikki ne kokeet, joiden duration-muuttujassa on korkeintaan parametrina ollut arvo. Nämä haetaan tietokannasta ja tallennetaan listalle, joka välitetään käyttöliittymälle. Valitettavasti tästä näkymästä ei pysty kokeita lisäämään omalle listalle. Tämä mahdollisuus tuli vasta tätä kirjoittaessa tekijän mieleen.



Kuva 18. Sekvenssikaavio search(subject: String, duration: int)-metodista

Maintenance-tason käyttäjät voivat myös päivittää kokeita GUIupdate-luokan avulla. Tässä luokassa haetaan ensin kaikki kokeet tietokannasta trywithkids.findAll()-metodin avulla. Nämä näytetään käyttöliittymässä listana, josta käyttäjä voi valita kokeen, jonka tietoja haluaa päivittää. Tällöin päivitettävän kokeen tiedot tarjotaan käyttäjälle päivitettäväksi. Päivitetyt tiedot varmennetaan käyttöliittymässä. Ensinnäkin tarkistetaan, onko pakolliset tiedot

edelleen täytettynä. Lisäksi tarkistetaan, ettei tekstit ole liian pitkiä aiheuttaakseen ongelmaa tietokannalle. Mikäli tiedot ovat ohjeiden mukaisesti täytetty, päivitetään kokeen tietoja `trywithkids.update(experiment, subject: String, topic: String, duration: int, waittime: String, materials: String, directions: String, notes: String, theScience: String)`-metodilla, jolle annetaan parametriksi päivitettävä koe sekä kaikki uudet tiedot käyttöliittymästä. `TryWithKids`-luokassa ensin tarkistetaan, että aihepiiri ja kokeen kesto on täytetty. Mikäli näitä ei olisi täytetty, säilytetään vanhat tiedot. Muussa tapauksessa nämäkin tiedot päivitetään. Tämän jälkeen `database.update(vanhatTiedot: Experiment, uudetTiedot: Experiment)`-metodin avulla ensin tietokannasta poistetaan vanhentuneet tiedot `datastore.delete(Experiment)` -metodin avulla ja sen jälkeen tallennetaan päivitetyt tiedot uutena kokeena `datastore.save(Experiment)` -metodin avulla.

Molempien tyyppien käyttäjät kykenevät selaamaan kokeita `GUIview`-luokan avulla. Tämän luokan avulla käyttäjä kykenee selaamaan kokeita, lisäämään niitä listalleen `trywithkids.addExpToUser(user, experiment)`-metodin avulla, printtaamaan kokeen pdf-tiedostoksi `trywithkids.print(experiment)` -metodin avulla, selaamaan omia kokeitaan `trywithkids.viewUsersExperimentsList(user)` -metodin avulla sekä poistamaan niitä listaltaan `trywithkid.deleteFromUserList(user, index: Integer)` -metodin avulla. Tarkastellaan ensin kokeen lisäämistä käyttäjälle. Metodissa `trywithkids.addExpToUser(user, experiment)` ensin haetaan `userDatabase.finduser(user)`-metodin avulla käyttäjän tiedot tietokannasta. Käyttöliittymästä tulevia tietoja käyttäjästä käytetään puhtaasti siis keinona löytää oikea käyttäjä tietokannasta. Sen jälkeen tälle käyttäjälle lisätään koe `.addExperiment(Experiment)`-metodin avulla. Tämän jälkeen käyttäjätiedot päivitetään jo aiemmin käsitellyn `userDatabase.updateUser(vanhatTiedot: User, uudetTiedot: User)`-metodin avulla. Koe poistetaan käyttäjän listalta `trywithkid.deleteFromUserList(user, index: int)` -metodilla, joka noudattaa samaa logiikkaa kuin lisääminenkin. Ensin haetaan käyttäjän tiedot tietokannasta `userDatabase.findUser()`-metodilla. Sen jälkeen haetaan käyttäjän kokeet ja poistetaan tietty indeksi. Tämän jälkeen päivitetään käyttäjän tiedot tietokannasta `userDatabase.updateUser()`-metodilla. Omalla listalla olevat kokeet pystyy näkemään `trywithkids.viewUsersExperimentsList(user)` -metodin avulla, joka hakee tietokannasta sen käyttäjän tiedot, joka annetaan metodille parametrinä ja palauttaa listan avulla tämän käyttäjän kokeet käyttöliittymälle.

Lopulta tietyn kokeen voi tulostaa ohjelmasta pdf-tiedostoksi `trywithkids.print(experiment)`-metodin avulla. Tämä metodi hyödyntää `iText`-kirjastoa, jolla on AGPL-lisenssi eli kyse on avoimesta lähdekoodista. Tämä rajaa kyseisen ohjelman toimintaa siten, että jokaisessa tulostetussa pdf-tiedossa tulee olla tieto `iText`-kirjaston käytöstä ja sen lisenssistä. Koska kuitenkin en muuta `iText`-lähdekirjastoa millään tavalla, eivät muut lisenssin rajoitukset koske tätä projektia. Metodissa luodaan tekstinpätkiä nimeltä `chunk`, joihin sitten voidaan asettaa määritteitä fontista, fonttikoosta sekä väristä. Nämä kerätään yhteen kappaleiksi (`paragraph`), jotka lisätään dokumenttiin. Ohjelma tallentaa nämä pdf-dokumentit samaan pääkansioon, missä on muut ohjelman tiedostot.





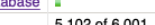

Näiden metodien lisäksi database-luokassa on metodi `deleteAllDatabase()`, joka tyhjentää tietokannan kokeista. Metodia hyödynnetään testauksessa sekä hyödynnettiin ohjelmaa rakennettaessa. Lisätietoa metodeista, niiden parametreista ja palautusarvoista löytyy javadoceista, jotka löytyvät ohjelman mukana `target-kansion` alakansiosta nimeltä `apidocs`. Liikkeenä näiden tutkimisessa pääsee hyvin tiedosta nimeltä `index.html`.

## 6. Testausselostus

### 6.1. Yksikkö- ja integraatiotestaus

Kokonaisuudessaan koodirivejä syntyi Jacocon raportin mukaan 1329 (kuva 19). Suurin osa näistä on käyttöliittymän puolella johtuen ensinnäkin siitä, että se on uusi osa-alue, joten sen koodin optimointi on vielä varsin haastavaa. Toiseksi validoin osan syötteistä käyttöliittymän puolella, koska se selvästi helpotti virheilmoitusten antamista käyttäjälle. Kuitenkin pakkauksessa `domain` on 254 riviä koodia sekä 60 metodia ja pakkauksessa `database` on 56 riviä koodia ja 19 metodia. Näiden kahden luokan testikattavuus on varsin hyvä. Käyttöliittymää ei ohjeiden mukaisesti lähdetty JUnitilla testaamaan vaan sen testaus tapahtui järjestelmätestauksessa. `Domain`-pakkauksen testikattavuus on 71% ja haaraumista on katettu 100%. `Database`-pakkauksesta on testattu 100%.

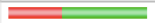
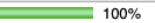




TryWithKids

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed Cxty	Missed Lines	Missed Methods	Missed Classes
trywithkids.gui		0%		0%	190 190	1,019 1,019	76 76	13 13
trywithkids.domain		71%		100%	3 71	58 254	3 60	0 3
trywithkids.database		100%		100%	0 20	0 56	0 19	0 2
Total	5,102 of 6,001	14%	228 of 252	9%	193 281	1,077 1,329	79 155	13 18

Kuva 19: Testikattavuus ohjelmassa

Tarkastellaan ensin `domain`-pakkausta (kuva 20). Pääasiallisesta sovelluslogiikka-luokasta (`TryWithKids`) on testattu 26/27 metodeista ja 100% haaraumista.

TryWithKids > trywithkids.domain

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed Cxty	Missed Lines	Missed Methods	Missed Classes
TryWithKids		61%		100%	1 36	55 183	1 26	0 1
Experiment		95%		n/a	2 22	3 43	2 22	0 1
User		100%		100%	0 13	0 28	0 12	0 1
Total	282 of 975	71%	0 of 22	100%	3 71	58 254	3 60	0 3

Kuva 20: Testikattavuus domain-pakkauksessa

Tässä luokassa erittäin haasteelliseksi testata muodostui `print()`-metodin testaaminen. Siihen ei löydetty tässä ajassa keinoa JUnitin avulla. Koska koodirivejä ”mielekkäästi muotoiltuun tulostukseen” syntyi kuitenkin jopa 55, laski luokan testien kattavuus alle 70%, vaikka muut metodit olivatkin testattuja. Tässä siis tehtiin päätös, jossa toisessa vaakakupissa on vaatimusmäärittely ja toisessa testikattavuus. Valittiin vaatimusmäärittely, sillä vaikka `print()`-metodin



olisi saanut toteutettua selvästi vähemmällä määrällä rivejä, mikä olisi nostanut testikattavuuden 70%, ei se tällöin olisi enää ollut yhtä mielekkäästi tulostettu.

## TryWithKids

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
print(Experiment)	0%	n/a	1	1	55	55	1	1		
addStarterExperiments()	100%	n/a	0	1	0	41	0	1		
update(Experiment, String, String, int, String, String, String, String, String)	100%	100%	0	3	0	15	0	1		
checkUser(String, String)	100%	100%	0	4	0	10	0	1		
search(String, int)	100%	100%	0	5	0	8	0	1		
createExperimentAndSave(String, String, int, String, String, String, String, String)	100%	n/a	0	1	0	3	0	1		
deleteFromUserList(User, int)	100%	n/a	0	1	0	4	0	1		
addExpToUser(User, Experiment)	100%	n/a	0	1	0	4	0	1		
changePassword(User, String)	100%	n/a	0	1	0	4	0	1		
addDefaultMaintenance()	100%	n/a	0	1	0	3	0	1		
addDefaultEnduser()	100%	n/a	0	1	0	3	0	1		
addUserThroughGUI(String, String, Boolean)	100%	n/a	0	1	0	3	0	1		
isMaintenancePresent()	100%	100%	0	2	0	4	0	1		
viewUsersExperimentsList(User)	100%	n/a	0	1	0	3	0	1		
TryWithKids(Database, DatabaseUsers)	100%	n/a	0	1	0	4	0	1		
checkUsernameExists(String)	100%	n/a	0	1	0	2	0	1		
getUserNi()	100%	n/a	0	1	0	2	0	1		
findAllUsers()	100%	n/a	0	1	0	2	0	1		
findAll()	100%	n/a	0	1	0	2	0	1		
deleteUser(User)	100%	n/a	0	1	0	2	0	1		
saveToDatabase(Experiment)	100%	n/a	0	1	0	2	0	1		
deleteOne(Experiment)	100%	n/a	0	1	0	2	0	1		
findUser(User)	100%	n/a	0	1	0	1	0	1		
databaseSize()	100%	n/a	0	1	0	1	0	1		
clearDatabase()	100%	n/a	0	1	0	2	0	1		
login()	100%	n/a	0	1	0	1	0	1		
Total	275 of 714	61%	0 of 20	100%	1	36	55	183	1	26

Kuva 21. Testikattavuus trywithkids-luokassa ja sen metodeissa

Experiment-luokassa getterit ja setterit testattiin poislukien setld ja getld. Näitä ei testattu siitä syystä, että ObjectId on muuttujana sellainen, etten tässä ajassa ennättänyt selvittää, miten tuota testaisin. Mikäli id olisi ollut integer, olisi sen testaus ollut varsin yksinkertaista. Nyt se kuitenkin jäi testaamatta. Muut testattiin pääosin integraatiotestien avulla eli yksittäisiä gettereitä tai setttereitä ei niinkään testattu vaan testattiin laajempia toiminnallisuuksia, joiden toimiminen edellyttää näiden toimimista ja näin ne tulivat samalla testattua.

TryWithKids > trywithkids.domain > Experiment














## Experiment

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
setld(ObjectId)	0%	n/a	1	1	2	2	1	1		
getld()	0%	n/a	1	1	1	1	1	1		
toString()	100%	n/a	0	1	0	2	0	1		
Experiment(String, String, Integer, String, String, String, String, String)	100%	n/a	0	1	0	10	0	1		
shortInfo()	100%	n/a	0	1	0	2	0	1		
setSubject(String)	100%	n/a	0	1	0	2	0	1		
setTopic(String)	100%	n/a	0	1	0	2	0	1		
setDuration(Integer)	100%	n/a	0	1	0	2	0	1		
setMaterials(String)	100%	n/a	0	1	0	2	0	1		
setDirections(String)	100%	n/a	0	1	0	2	0	1		
setTheScience(String)	100%	n/a	0	1	0	2	0	1		
setWaitTime(String)	100%	n/a	0	1	0	2	0	1		
setNotes(String)	100%	n/a	0	1	0	2	0	1		
Experiment()	100%	n/a	0	1	0	2	0	1		
getSubject()	100%	n/a	0	1	0	1	0	1		
getTopic()	100%	n/a	0	1	0	1	0	1		
getDuration()	100%	n/a	0	1	0	1	0	1		
getMaterials()	100%	n/a	0	1	0	1	0	1		
getDirections()	100%	n/a	0	1	0	1	0	1		
getTheScience()	100%	n/a	0	1	0	1	0	1		
getWaitTime()	100%	n/a	0	1	0	1	0	1		
getNotes()	100%	n/a	0	1	0	1	0	1		
Total	7 of 169	95%	0 of 0	n/a	2	22	3	43	2	22

Kuva 22: Testikattavuus Experiment-luokassa ja sen metodeissa

User-luokan JUnit-testeissä ei ollut erityisiä haasteita. Kokonaisuutenaan domain-pakkauksen testikattavuus nousee yli 70%, vaikka TryWithKids-luokan testikattavuus erityisesti nimenomaan print-metodista johtuen jääkin tuon rajan alle.



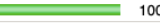
## User

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed Cxty	Missed Lines	Missed Methods
toString()		100%		100%	0 2	0 7	0 1
User(String, String, Boolean)		100%	n/a	n/a	0 1	0 6	0 1
addExperiment(Experiment)		100%	n/a	n/a	0 1	0 2	0 1
setUsername(String)		100%	n/a	n/a	0 1	0 2	0 1
setPassword(String)		100%	n/a	n/a	0 1	0 2	0 1
setMaintenance(Boolean)		100%	n/a	n/a	0 1	0 2	0 1
getId()		100%	n/a	n/a	0 1	0 1	0 1
User()		100%	n/a	n/a	0 1	0 2	0 1
getUsername()		100%	n/a	n/a	0 1	0 1	0 1
getPassword()		100%	n/a	n/a	0 1	0 1	0 1
getMaintenance()		100%	n/a	n/a	0 1	0 1	0 1
getExperiments()		100%	n/a	n/a	0 1	0 1	0 1
Total	0 of 92	100%	0 of 2	100%	0 13	0 28	0 12

Kuva 23. Testikattavuus User-luokassa ja sen metodeissa

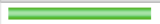







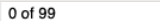
Database-pakkausessa testit kattoivat 100% luokista, kuten alla olevista kuvista voi nähdä. Näiden metodien testaaminen oli hyvin yksinkertaista JUnitin avulla eikä ongelmiin törmätty.

## trywithkids.database

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed Cxty	Missed Lines	Missed Methods	Missed Classes
DatabaseUsers		100%		100%	0 11	0 29	0 10	0 1
Database		100%	n/a	n/a	0 9	0 27	0 9	0 1
Total	0 of 206	100%	0 of 2	100%	0 20	0 56	0 19	0 2

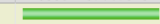
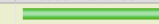









Kuva 24. Testikattavuus Database-pakkauksessa

## Database

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed Cxty	Missed Lines	Missed Methods
findByBoth(String, int)		100%	n/a	n/a	0 1	0 4	0 1
findByDuration(int)		100%	n/a	n/a	0 1	0 3	0 1
findBySubject(String)		100%	n/a	n/a	0 1	0 3	0 1
deleteAllDatabase()		100%	n/a	n/a	0 1	0 3	0 1
update(Experiment, Experiment)		100%	n/a	n/a	0 1	0 3	0 1
findAll()		100%	n/a	n/a	0 1	0 3	0 1
Database(Morphia, Datastore)		100%	n/a	n/a	0 1	0 4	0 1
save(Experiment)		100%	n/a	n/a	0 1	0 2	0 1
deleteOne(Experiment)		100%	n/a	n/a	0 1	0 2	0 1
Total	0 of 99	100%	0 of 0	n/a	0 9	0 27	0 9

Kuva 25: Testikattavuus Database-luokassa

## DatabaseUsers

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed Cxty	Missed Lines	Missed Methods
findUsername(String)		100%		100%	0 2	0 6	0 1
checkLoginInfo(String)		100%	n/a	n/a	0 1	0 3	0 1
findAdmins()		100%	n/a	n/a	0 1	0 3	0 1
clearDatabaseOfUsers()		100%	n/a	n/a	0 1	0 3	0 1
updateUser(User, User)		100%	n/a	n/a	0 1	0 3	0 1
DatabaseUsers(Morphia, Datastore)		100%	n/a	n/a	0 1	0 4	0 1
findAll()		100%	n/a	n/a	0 1	0 2	0 1
findUser(User)		100%	n/a	n/a	0 1	0 1	0 1
save(User)		100%	n/a	n/a	0 1	0 2	0 1
delete(User)		100%	n/a	n/a	0 1	0 2	0 1
Total	0 of 107	100%	0 of 2	100%	0 11	0 29	0 10

Kuva 26: Testikattavuus DatabaseUsers-luokassa

## 6.2. Järjestelmätestaus

Järjestelmätestauksessa käytiin käyttöliittymän kautta ohjelman toiminto läpi kattavasti. Jokainen toiminnallisuus pyrittiin tarkistamaan kaikkien käyttäjäluokkien näkökulmasta. Syötteet ohjelmaan testattiin sekä siinä tapauksessa, että niiden tulee toimia että mahdollisilla virhesyötteillä, jolloin odotettiin, että saadaan ohjelmoitu virheilmoitus. Järjestelmätestauksen aikana löydettiin muutama bugi ohjelmasta, jota pääasiallisesti korjattiin. Jäljelle jääneet bugit on raportoitu aiemmin kohdassa 3.5. Alla on tarkempi kuvaus testauksesta, eli mitä testattiin ja mitä saatiin tulokseksi ja mitä bugeja löydettiin. Alla on oma järjestelmätestaukseni. Lisäksi järjestelmän testasi vertaisarvioija vielä hieman keskeneräisenä että toinen ohjelman kehittämiseen liittymätön arvioija, jonka tehtävä oli kaataa ohjelmisto ja löytää virheitä. Hän ei näitä löytänyt eikä saanut ohjelmaa kaatumaan.

### LOGIN JA LOGOUT (maintenance, end-user) - TOIMII

- Väärä käyttäjätunnus – oikea salasana: ei kirjaa sisään. Antaa virheilmoituksen: no such user in database
- Oikea käyttäjätunnus – väärä salasana: ei kirjaa sisään. Antaa virheilmoituksen: password does not match.
- Molemmat väärin: virheilmoitus käyttäjästä. Ei virheilmoitusta salasanasta (tietenkään, kun ei ole mitään, mihin verrata)
- Molemmat oikein end-user default: toimii ja avaa oikeat toiminnallisuudet.
- Molemmat oikein maintenance-default: toimii ja avaa oikeat toiminnallisuudet.
- Logout palauttaa login-ikkunaan

### CHANGE PASSWORD (maintenance, end-user) - TOIMII

- Väärä vanha salasana: virheilmoitus password incorrect. Kaavake ei tyhjene.
- Väärä vanha salasana, ei täsmäävät uudet: virheilmoitus password does not match ja password incorrect.
- Oikea vanha salasana ja täsmäävät uudet salasanat: saa varmistustiedon "new password saved"
- password change form ei tyhjene, kun vaihtaa sivua. BUGI KORJATAAN. Kun salasanan tallentaa, kentät tyhjäntyvät.

### MY INFO (end-user) - TOIMII

- Own info: palauttaa oikein käyttäjän tiedot
- Päivittykö tiedot, jos salasanan vaihtaa: päivittyy. Näyttää uuden salasanan
- Päivittykö tiedot, jos listalle lisää kokeen: päivittyy. Yhden kokeen lisäämisen jälkeen listan koko on 1.
- Kun lisättiin kokeita listalle ja poistettiin sieltä yksi, tämä päivittyi vasta, kun kirjauduttiin ulos.

#### BROWSE (end-user, maintenance) - TOIMII

- browse all näyttää automaattisesti oikean sivun eli listauksen kaikista kokeista
- Koetta klikkaamalla näkee oikean pop-up valikon. Sieltä voi lisätä omalle listalleen ja koe lisäytyy omalle listalle.
- Tulostus tulostaa pdf-tiedoston kokeesta pääkansioon.
- Pdf-tiedostosta löytyy oikeat ohjeet.
- Pdf-tiedostossa on myös lisenssin vaatima merkintä ohjelmasta ja sen lisenssistä.
- Pdf-tiedoston muotoilusta saa selvää.
- Browse own list näyttää oman listan.
- Delete from my list poistaa kokeen omalta listalta. Tieto päivittyy myös omiin tietoihin, mutta vasta uloskirjautumisen kautta. KORJATTIIN BUGI. Nyt joka kerta, kun painetaan own info-nappia, käyttäjän tiedot haetaan tietokannasta.
- Cancel toimii kummassakin popupissa.

#### SEARCH (end-user, maintenance) - TOIMII

- Subject löytää oikeat kokeet tietokannasta ja palauttaa vain oikeat kokeet. Esim. kun haettiin biologian kokeita, vain biologian kokeet palautuivat.
- Duration löytää oikean pituiset kokeet ja palauttaa vain ne.
- Yhdistelmähaku löytää oikeat kokeet ja palauttaa vain ne

#### USERS (maintenance) - TOIMII

- Add users: virhetilanteessa ei tallenna käyttäjää
  - o Too short password: error message + oikea text prompt
  - o Role missing: oikeat virheilmoitukset
  - o Salasanat eivät täsmää: oikeat virheilmoitukset.
  - o Kun kaikki on kunnossa, käyttäjä tallentuu
- Muutokset käyttäjissä näkyvät listauksessa (browse and delete users)
- Salasanan vaihto onnistuu kuten käyttäjälläkin (samat testit ja samat tulokset)
- Käyttäjän voi deletoida
- Viimeisen maintenance-tyypin käyttäjän poisto lisää default-käyttäjän tietokantaan. Ei päivity näkymään välittömästi, vaan pitää käydä muualla ensin.

#### ADD NEW experiment (maintenance) - TOIMII

- kun kaikki on oikein täytetty, koe tallentuu tietokantaan ja näkyy browse-näkymässä
- subject puuttuu: ei tallennu, ei virheviestiä
- topic puuttuu: ei tallennu, virheviestit oikeat
- duration puuttuu: ei tallennu, ei virheviestiä
- directions puuttuu: ei tallennu, virheviestit oikeat
- materials puuttuu: ei tallennu, virheviestit oikeat
- topic liian pitkä teksti: topic -kenttä tyhjenee ja antaa oikean virheviestin
- muutkin paitsi directions liian pitkä teksti: kenttä tyhjenee ja antaa oikean virheviestin

- directions-kohdassa virhe. Korjataan bugi. Nyt toimii oikein. Nyt kenttä tyhjenee ja antaa virheviestin.

#### DELETE experiment (maintenance) – TOIMII

- kokeen pystyy deletoimaan. Se näkyy listauksissa
- cancel toimii.

#### UPDATE experiment (maintenance) - TOIMII

- kaikki tiedot ok : päivitys tallentuu oikein ja kokeen tiedot päivittyvät
- cancel toimii
- duration ja subject ei pysty jättämään tyhjäksi
- ei tallennu liian pitkällä teksteillä. BUGI: ei anna myöskään mitään virheilmoitusta. Korjattu. Nyt tulee virheilmoitus.

#### BUGIT, jota ei ole korjattu:

- omituinen virhe – ei vaikuta toimintaan: 2019-04-21 09:45:09.629  
java[6696:29499020] unrecognized type is 4294967295  
2019-04-21 09:45:09.630 java[6696:29499020] \*\*\* Assertion failure in -  
[NSEvent\_initWithCGEvent:eventRef:],  
/BuildRoot/Library/Caches/com.apple.xbs/Sources/AppKit/AppKit-  
1671.20.108/AppKit.subproj/NSEvent.m:1969
- jos add new experiment subject tai duration puuttuu, siitä ei saa virheviestiä -> korjataan käyttäjäohjeistusta.
- virheviestit jäävät ikkunaan, kun testailee, mikä kaikki menee läpi uuden kokeen lisäämisessä

### 6.3. Ongelmia testauksessa

Testauksessa oli kolme selkeää haastetta, jotka on aiemmin tässä luvussa kuvattu, joten tässä kohtaa haasteet käydään hyvin suppeasti läpi. Ensinnäkin sovelluslogiikan luokassa osoittautui tässä ajassa mahdottomasti nostaa testikattavuus 70%, koska yksi ainoa pidempi metodi riitti tiputtamaan muuten hyvin korkean testikattavuuden 62 prosenttiin. Kyseessä on pdf:n tulostaminen, jonka testauksessa ei tässä kohtaa edetty niin, että toimintoa olisi kyetty testaamaan. Vaikka tämä raja ei nyt täyttynytkään, sovelluslogiikan 27 metodista testattiin 26, joista suurin osa 100%.

Toinen haasteellinen testattava oli if-rakenteissa jokaisen polun testaus. Välillä vaati jonkin verrankin pätkäilemistä, että tajusi, mitä kaikkia mahdollisuuksia tietokone omista koodirakenteista löytäisi.

Experiment-luokan ObjectIdn getterit ja setterit myös ovat tarkistamatta. Hyödynnän näitä kyllä testauksessakin, mutta varsinaisia testejä ei näille kirjoitettu.

Järjestelmätestauksessa lähinnä haasteeksi muodostui riittävä kärsivällisyys testata jokainen eri iteraatio virheilmoituksista rauhallisesti. Tämän johdosta pyydettiin vielä kurssin ulkopuolista ystävää testaamaan ohjelma, kokeilemaan sen kaatamista sekä etsimään mahdollisia virheitä.

## 7. Pohdinta ja itsereflektio

### Miten työskentely sujui?

Työskentely sujui mielestäni varsin hyvin. Sovelluslogiikka itsessään oli tuttua. Käyttöliittymässä näytettiin tietoa ja siellä tuotettiin tietoa. Nämä tiedot tulivat sovelluslogiikan luokista ja tietoa tallennettiin ja haettiin tietokannasta. Pohjimmiltaan siis hyvin perustavanlaatuisia ominaisuuksia tietokantapohjaisessa ohjelmistossa.

Kuitenkin paljon uutta piti selvittää ja monta ongelmaa itsenäisesti ratkaista, minkä takia lisäominaisuuksia ei tällä kertaa ohjelmistoon lisätty vaan pitäydettiin alun vaatimusmäärittelyn lupauksissa. Esimerkiksi haastavaa oli eriyttää käyttöliittymä käyttäjätyypin mukaan. Toinen haaste oli selvittää, miten ohjelmistosta tulostetaan kokeita mielekkäästi muotoiltuna. Tämänlaiset epämääräiset kommentit, kuten ”mielekkäästi muotoiltuna”, alun vaatimusmäärittelyssä aiheuttivat välillä paljonkin päänvaivaa. Kolmas haaste oli käyttäjien kirjautuminen, sillä sitä ei oltu aiemmilla kursseilla käsitelty.

Muutama isompi oivallus projektia tehdessä tuli. Ensinnäkin määritellyt vaatimukset voidaan toteuttaa hyvin monella eri tavalla erityisesti, jos vaatimus on määritelty hyvin ylimalkaisesti, kuten ”ohjelmistolla on käyttäjärooleja”. Tässä projektissa esimerkiksi päätin toteuttaa käyttäjien lisäämisen niin, että ylläpitäjä lisää käyttäjiä. Tämä on mielekästä ajatellen sitä käyttöä luokkahuonetilanteessa tai vanhempien touhutessa lastensa kanssa, mutta vertaisarvioijan toive siitä, että käyttäjän voisi luoda alun kirjautumisessa on sekin varsin perusteltu ja yksi hyvä jatkokehityskohde. Tässä projektissa asia ei ole niin merkityksellinen, koska asiakkaita ei ole, mutta projekti toi selkeästi esiin, miten vaikeaa vaatimuksia on määritellä siten, että kaikilla on suhteellisen samanlainen käsitys vaatimuksista.

Toinen iso huomio oli, että ”pienet” lisäominaisuudet voivat aiheuttaa hyvinkin paljon lisätöitä. Tulostaminen mielekkäästi muotoiltuna toi 49 riviä koodia tosin sisältäen muutaman tyhjän rivin luettavuutta lisäämässä. Käyttäjäroolien lisääminen lähes tuplasi siihen asti koodattujen rivien määrän.

Kolmas huomio on, että vaikka pääosa logiikasta piti olla sovelluslogiikan puolella, esimerkiksi erilaisten lomakkeiden sisällön varmistaminen tehtiin tässä projektissa pitkälti käyttöliittymän puolella, koska tällöin oli helpompi muun muassa antaa virheilmoituksia käyttäjälle. Täysin ongelmaton tämä ratkaisu ei ollut, kuten avaan alla kohdassa ”Tekisitkö jotain toisin?”.

Neljäntenä huomiona oli iteraatioittain etenevän työn mielekkyys. Koin mielekkäänä sen, että alussa määriteltiin suunta, jota sitten tarkennettiin iteraatio iteraatiolta. Toisaalta tämän projektin malli oli enemmän vesiputousmallin tyyppinen, sillä vaatimukset määriteltiin jokaiselle iteraatiolle heti alussa sen sijaan, että vaatimukset olisi määritelty aina iteraatiokohtaisesti. Kuitenkin mahdollisuus oli muokata vaatimuksia. Esimerkiksi alun perin kyky lisätä kokeita listalle oli määritelty iteraatioon 2, mutta projektin edetessä huomasin, että se kyky liittyy käyttäjän tietoihin ja käyttäjät lisätään projektiin vasta iteraatiossa 3. Siirsin sitten hakutoiminnot iteraatioon 2 ja listalle lisäämisen iteraatioon 3. Tällöin työmäärä pysyi suunnilleen samassa, mutta etenemisjärjestys oli mielekkäämpi.

Viidentenä huomiona on se, että on varsin hankalaa testata kaikkia mahdollisia ongelmia. Itse sitä ei saa ikinä tehtyä. Siksi tarvitaan testikäyttäjiä, jotka käyttävän ohjelmistoa myös toisin kuin tekijä on sen ajatellut.

Kuudentena huomiona oli dokumentaation kirjoittamisen hyödyllisyys. Dokumentaatio pakotti tarkastelemaan ohjelmakoodia hyvin monesta eri näkökulmasta, miettimään sen logiikkaa, testaamaan kaikki toiminnallisuudet ja selittämään ne auki. Tämä avulla löytyi useampi bugi. Lisäksi yleensäkin ottaen dokumentaation teon myötä ohjelman toimintaa tuli tarkasteltua paljon syvemmällä tasolla kuin olisi tapahtunut ilman dokumentaation vaatimusta.

### **Tyydyttääkö lopputulos?**

Olen hyvin tyytyväinen lopputulokseen. Se ei ole täydellinen. Keksinkin siitä montakin kohtaa, jota voisi vielä kehittää ja viedä eteenpäin ja niistä lisää alla. Kuitenkin ohjelma toimii. Vertaisarvioija ja ulkopuolinen testaaja eivät saaneet yrityksistä huolimatta kaadettua ohjelmistoa. Se toimii. Se tekee kaiken sen, minkä sen pitäisikin tehdä. Sain ongelmat ratkaisua Googlen avulla itsenäisesti. Kaunis se ei ehkä ole, mutta siellä on selkeät ohjeet käyttäjälle. Opin käyttämään NoSQL-tietokantaa. Ennen tätä kurssia en ollut kertaakaan sellaista käyttänyt. Opin tulostamaan java-ohjelmasta. Opin tekemään monipuolisia graafisia käyttöliittymiä. Sain korjattua monta bugia. Olen ihan suoraan sanoen aivan naurettavan ylpeä ensimmäisestä omatekoisesta ohjelmastani.

### **Tekisitkö jotain toisin?**

Edelleen pohdin päätöstä pitää iso osa sovelluslogiikkaa yhdessä luokassa (trywithkids). Luokassa on varsin paljon metodeja, joista osa liittyy käyttäjiin ja osa kokeisiin. Tästä syystä luokasta voi olla hetimitäin vaikea löytää haluamaansa metodia. Pohdin pitkään sen jakamista kahdeksi luokaksi, jossa toisessa keskityttäisiin käyttäjiin ja toisessa kokeisiin, mutta päädyin pitämään kaiken samassa sovelluslogiikka-luokassa, koska osa metodeista sisältää sekä käyttäjiä että kokeita ja tällä tavalla tietää varmasti, mistä luokasta jokin metodi löytyy. Tätä kuitenkin edelleen hieman pohdin.

Toinen, mitä tekisin toisin, on, että yksinkertaistaisin käyttöliittymää. Nyt kokeet listataan kolmessa eri näkymässä (browse, update ja delete). Jokaisessa listauksesta pystyy tekemään eri asioita, mutta jos aloittaisin ohjelman tekoa uudelleen, yksinkertaistaisin käyttöliittymää niin, että kokeet listataan vain kerran ja koetta valitsemalla pystyisi sen jälkeen saamaan laajasti eri vaihtoehtot käyttöönsä sillä, mitä haluaa kokeelle tehdä.

Kolmantena muutoksena on sisällön tarkistus käyttöliittymässä. Käyttöohjeita tehdessäni huomasin, etten testaa käyttöliittymästä syötettävien kenttien sisältöä, kuten testaan ne uutta koetta luodessa. Nyt testaan sitten aikalailla samat asiat kahdessa paikassa. Koodi olisi kauniimpaa, jos testaus tehtäisiin vain yhdessä paikassa metodina ja siihen viitattaisiin useammasta paikasta. Tässä vaiheessa havaittuna kuitenkin refaktorointi virheilmoituksineen ja käyttöliittymään ja testauksineen ei ollut enää mahdollista, mutta seuraavalla kerralla pitää tämä asia miettiä toisella tavalla.

### **Miten ohjelmistoa voisi kehittää?**

Ohjelmistoa voisi kehittää ensinnäkin suolaamalla ja salaamalla salasanat. Opin tekemään tämän java Spring-ohjelmistossa pääsiäisen tienoilla, joten en sitä

tehnyt sitten tälle ohjelmistolle. Tämä on kuitenkin yksi keskeinen kehityskohde. Jatkossa en halua tehdä yhtään ohjelmistoa, jossa on käyttäjien salasanoja, joita ei olisi suolattu ja salattu.

Ohjelmistoon voisi lisäksi rakentaa kyvyn rekisteröityä sisään kirjautumisen yhteydessä. Tarkoituksena on jossain kohtaa rakentaa lisäksi tältä pohjalta web-sovellus ja tällöin voisi nimenomaan lisätä tämänlaisen rekisteröitymisen käyttäjäksi.

Lisäksi kokeisiin voisi liittää kuvia ja ehkä jopa videoita. Kokeita voisi kukin käyttäjä arvioida. Nämä arviot voisi liittää kokeisiin hashmapin avulla, jossa käyttäjätunnus on avain ja arvio on avaimen arvo. Täten jokaisella käyttäjällä olisi yksi arvio per koe. Näistä sitten voisi kerätä keskiarvon, joka voitaisiin näyttää kokeen yleistietojen kohdalla. Mietin kovasti tämän ominaisuuden lisäämistä, mutta aika loppui valitettavasti kesken ja halusin keskittyä myös dokumentaatioon.

Lisäkehityskohteena on myös selvittää, mitä pitää ottaa huomioon, jos ohjelmistolla on useampia käyttäjiä ja miten tällöin varmistetaan esimerkiksi tietokannan eheys.

### **Miten työ eteni iteraatioittain? Mitä milloinkin oli valmiina ja etenisitkö toisella tavalla?**

Etenin hyvin pitkälle alun vaatimusmäärittelyn mukaisesti. Tarkemmat päivämäärät löytyvät työskentelypäiväkirjasta. Ensimmäisessä iteraatiossa valitsin tietokantaratkaisun, tein luokat sovelluslogiikalle, kokeille, tietokantatoiminnoille ja käyttöliittymän alun. Päätin, etten tee tekstikäyttöliittymää, vaan rakennan heti graafisen liittymän. Olen edelleen varsin tyytyväinen tähän päätökseen, sillä vaikka se vei paljon aikaa, ei tarvinnut käyttää aikaa kahteen täysin erilliseen käyttöliittymään. Ensimmäisessä iteraatiossa lisäksi luotiin mahdollisuus lisätä kokeita, selata kokeita, lukea kokeiden ohjeet, aloitettiin testaus ja luotiin 4 koetta tietokantaan, jotta ohjelmaa on helppo testata ja jotta näkyy, minkälaista sisältöä varten ohjelma on luotu.

Toisessa iteraatiossa loin kyvyn poistaa kokeita, muokata kokeita ja hakea kokeita. Laajensin JUnit-testausta. Työskentelyssä koko ajan myös testasin käyttöliittymän toimintaa. Uudet toiminnot lisättiin jokaiselle ohjelman kerrosarkkitehtuurin tasolle tietokannasta käyttöliittymään.

Kolmannessa iteraatiossa keskityin käyttäjiin. Loin käyttäjätyypit, niitä käyttävän tietokantaluokan, kyvyn selata käyttäjätietoja, lisätä käyttäjiä, poistaa käyttäjiä, nähdä omat tietonsa sekä vaihtaa salasanaa. Löysin ratkaisun, miten näyttää eri sisältö ohjelmassa käyttäjäprofiilin mukaisesti. Ratkaisin, miten jokaisella käyttäjällä on oma lista, johon voi lisätä kokeita ja poistaa niitä sekä selata listaa. Tämän iteraation tavoitteet olivat ehkä suuritöisimpiä projektissa. Jatkossa jakaisin näin isojen ohjelmamuutosten teon useammalle iteraatiolle.

Neljännessä iteraatiossa hoidin javadocit kuntoon, korjasin löytyneitä bugeja ja keskityin tekemään hyvän dokumentaation ja loppuraportin.

Viidennessä iteraatiossa kirjoitin dokumentaation valmiiksi. Tähän sisältyi muun muassa järjestelmätestausta ja ohjelmiston toiminnan kuvausta erilaisilla kaavioilla.

Näin jälkikäteen ajatellen etenisin edelleen lähes samalla tavalla. Tosiaan muutin kahden ominaisuuden luomisen ajankohtaa, kuten ylempänä on kerrottu. Lisäksi



jakaisin isot muutokset useampaan iteraatioon. Mutta ominaisuuksien luomisen järjestyksen pitäisin edelleen tässä, kun se nyt on. Tästä eteenpäin ohjelmaa on helppo muokata edelleen.

## Liitteet:

### Readme.md

# Welcome to TryWithKids

## What is this program about

This is a fairly simple program to browse, add, edit, search and delete scientific experiments with kids. It will have separate GUIs for maintenance that can do all the above-mentioned things (f.ex. for parents or teachers). Maintenance-type user can also add users and delete users. One maintenance-type user will come with the software, including password, which can be changed.

User-type can browse and search for experiments (f.ex. for children or parents if someone else adds experiments for them). The idea is that a teacher can add his/her students that can then perform the experiments as groups or individuals. Or parent can add their children so they can browse the experiments on their own and add interesting experiments to their personal lists to conduct when they so choose. Users can also change their password.

Some experiments come with the software.

This program will save passwords in plain text into the database at this time. Although developer is aware salting and hashing of passwords is advised, there is insufficient time to learn to do this properly during this course.

The database-solution is MongoDB.

## Environment

This program has been written with Netbeans version 8.2. and java version 1.8.0\_181. Mongo java-driver version is 3.10.1 and Morphia version is 1.4.0.

MongoDB is 4.0 Community edition.

The computer operating system used in development is MacOS Mojave.

## Licenses

Morphia is licensed through Apache License 2.0, which allows its use in private and public works with the same license or with another license.

iText used in creating pdf-documents is licensed under AGPL-license, that requires a mention of the source code and the licence in each pdf-document created. Also all modifications should be made public. No such modifications are created in this project.

## Development

### Iteration 1:

Currently the GUI (and program) works for add and browse all.

It is possible to add to database and to find all from database and clear database.  
No separation of user profiles yet.  
Four experiments come with the software at this time.

### ### Iteration 2:

GUI (and program) works for delete, update and search.  
It is possible to delete an experiment from the database (from gui), update an experiment and search by subject, by maximum desired duration or both.  
These features are tested with JUnit.

### ### Iteration 3:

Ability for maintenance-type users to add and delete users.  
Ability for maintenance-type users to see all uses.  
Ability to change password.  
Ability to log in.  
Different views for different user-types.  
Ability to add experiments to users own list. Ability to see own list and delete from it.  
Ability to print experiments from application : the pdf is saved to TryWithKids main folder.  
In Netbeans it can be found in the files-section. The name of the document is named after the experiment topic (+ .pdf).

### ## Iteration 4:

JUnit tests (excluding GUI) at least 70%  
Javadoc on all classes and methods - can be found in target-folder -> apidocs-folder  
Documentation : testing report, user guide f.ex. started  
Fixed bugs:  
- maintenance-type user could delete all maintenance-type users from database. Now there will always be at least one. If the last one is deleted, the default-maintenance user will be automatically added.  
- user can no longer try to add text that is too long for the database when adding experiments. They will also receive a notification if their text length exceeds the limits.

### ## Iteration 5: by the 3rd of May

Documentation finished

### ## How to install this software

Install Netbeans by following instructions here:  
<http://moocfi.github.io/courses/general/programming/how-to-get-started.html>  
Install Mongo by following instructions here:  
<https://docs.mongodb.com/manual/administration/install-community/>  
Find the Java driver mongo-java-driver-3.10.1.jar for mongo at  
<https://mongodb.github.io/mongo-java-driver/>  
and download the driver, make sure it is in the same folder as the project folders src and target. If it is not, please move it there.  
Download the .zip-file containing this program.

From Netbeans, choose "file" -> "import project" -> from ZIP... and choose the zip-file containing this file. Choose open. Once opened, press the green play-button to run the software.

### ## Default users

This program comes with two default users. This info is also found in the code, so it is HIGHLY advisable to change the passwords after installation.

User-information for maintenance is:

username : maintenance

default password : main\_auth123

User-information for user is:

username: end-user

default password: end\_auth987

### ## Note on security --- IMPORTANT ---

The security of this application is very poor. Default-maintenance passwords are found in this readme as well as in the code. They are clearly visible. Also all passwords are saved in plain text (String) at this time. All maintenance-type users can see all user passwords that are stored in the same device. From the security -point of view this is not wise, but from the teacher's point of view, this is almost a must as they need to help students login very frequently. Passwords should be salted and hashed. However, they will not be at this time.

### ## Known bugs at this time:

- listview scrolls also horizontally. Content should be wrapped.
- error messages remain in window until user visits another part of application

### ## Creator information

The creator and developer for this project is Satu Korhonen.

This software is being built for a course "Ohjelmistotekniikka" for the University of Helsinki in the Spring 2019. However, the idea for this has been brewing in Satu's mind for a longer period and the experiments come from her blog [isbel.org](http://isbel.org)

## Työpäiväkirja

### Ohjelmistotekniikka – kevät 2019

Satu Korhonen

#### Työskentelypäiväkirja

13.2.2019 Keskiviikko 1,5t

Tutustumista materiaaliin ja ensimmäisen viikon versionhallinnantehtäviin

15.2.2019 Perjantai 3t

Git ja versionhallinta tehtävät ja raportointi 2t

Vaatusmäärittely 1t

19.2.2019 tiistaina 1t

Vaatusmäärittely 45min

Testausta 15min

20.2. keskiviikkona 1,5t

Testausta klo 13-14.30 -> 1,5t

+ vaatusmäärittelyn palautus

24.2. sunnuntaina 2,5t

Lukemista: 1,5

Sopivan tietokannan etsintää 1t

25.2. maanantaina 2t

MongoDB:hen tutustumista 15min

Ohjelmistoprojektin luominen TryWithKids 1,5t : MongoDB ja Morphia

UML-kaaviot klo 15min

Muistiinpanoja:

- Käyttöliittymä roolin valintaan => Roolin valitsija
- Oma käyttö ylläpitäjälle ja käyttäjälle
- Omat morphiat tietokantakeskusteluun ylläpitäjälle ja käyttäjälle
- Ekassa iteraatiossa keskity ylläpitäjän rooliin.
- Kerrokset: Käyttöliittymä, Sovelluslogiikka, Se, mikä keskustelee tietokannan kanssa

26.2. tiistaina 3,5t

3,5t – koodausta: luokka Experiments ja GUI alku, gui-pakkaus

2.3. perjantaina 2t

2t koodausta : gui-luokat eri ylläpitäjän ikkunalle. Alustusta tietokannan kanssa keskusteluun

3.3. lauantaina 8t

8t koodausta: **Tietokanta ja ohjelma keskustelee** keskenään. Tietoa voi lisätä tietokantaan, hakea sieltä (kaikki) ja deletoida (kaikki). Lisätty starter kokeita, joilla voi testata hakua ja ohjelmaa. Testikattavuus trywithkids.class on 100%, mutta Experiments vasta osin testattu.

### 5.3. Tiistaina 6t

6t koodausta: GUIladd ikkuna toimii viimein tämän työn jälkeen ja kokeen pystyy lisäämään tietokantaan. Pitää vielä selvittää, miten reagoi tahallisiin virheisiin. Trywithkids on yksinkertaistettu, testikattavuutta ei uudelleen tarkistettu.

### 6.3. Keskiviikkona 3,5t

3,5t koodausta: **GUIview** toimii ja **GUIladd** on testattu (manuaalisesti). Ei lisää puutteellista tietoa tietokantaan, mutta ei anna oikein virheilmoitusta, jos toggle ei ole valittu. Nyt sitten valittiin vaihtoehtoinen reitti. GUIview pitää vielä jossain kohtaa katsoa, että teksti pysyy ikkunassa.

### 7.3. Torstaina 1t

1t Readme-kirjoitusta ohjelmaan ja .gitignoren selvittelyä

### 8.3. perjantaina 0,75t

45min gitignore ja palautus.

### 10.3. sunnuntaina 2t

2t, että jacoco löytää taas oikein luokkani ja saisin koko projektin zip-tiedostoon.

### 15.3. perjantaina 6,5t

6,5t koodausta: **delete** one experiment database – trywithkids – gui. Ei vielä testausta JUnitilla. Yksi bugi käynnistysvaiheessa (lisäili jatkuvasti peruskokeet) korjattu.

### 16.3. lauantaina 2,5t

2,5t koodausta: **update** experiment database – trywithkids – gui. Ei vielä testausta JUnitilla. Refresh listview sekä updatella että deletellä, niin ui pysyy kartalla muutoksista. Updatella yksi toggle pysyy aina valittuna sekä subject että durationissa.

Muutos vaatimusmäärittelyn järjestykseen: Listat jää kolmannelle iteraatiolle, kun siihen tarvitaan käyttäjätietoja. Nyt sitten tehdään toisessa iteraatiossa hakutoimintoja.

### 18.3. Maanantaina 1,5t

1,5t koodausta: Javadoc toistaiseksi olemassa oleville luokille ja metodeille poislukien perus getterit ja setterit. Lisäksi luokka User –hahmottelua.

### 20.3. keskiviikkona 4,75t

45min dokumentaatiota: alustava pakkauskaavio, luokkakaavio ja ensimmäinen sekvenssikaavio (create new and save to database)

2t koodausta: **search** gui - trywithkids – database toimii. Ei JUnit testejä vielä.

2t JUnit tests: nyt on 100% trywithkids ja lähes sama Experiments (ei vielä setId ja getId – pitää tarkistaa, tarvitaanko näitä ollenkaan).

### 29.3. perjantaina 3,5t

1t vertaisarviota toisen työstä:

Vertaisarvioon tutustumista ja sen ehdottamien korjausten tekoa:

Ohjelma toimii hyvin, sen ominaisuudet toimivat, eikä se kaadu käytettäessä.

Ainakin yhden bugin löysin: lisättäessä kurssia on kenttiin mahdollista syöttää tekstiä enemmän, kuin tietokanta pystyy käsittelemään. Tällöin ohjelma heittää konsoliin virheen:

“Exception in thread "JavaFX Application Thread" com.mongodb.WriteConcernException: Write failed with error code 17280 and error message 'WiredTigerIndex::insert: key too large to index'”

Tällöinkään ohjelma ei kaadu, mikä on hyvä asia, mutta käyttäjälle esitetä mitään virheilmoitusta, vaikka kurssia ei lisätä kantaan.

Koodi näyttää erittäin hyvältä ja siistiltä. Ainoa vika jonka löysin, on että luokkiin on tuotu runsaasti tarpeettomia kirjastoja (unused import). Koodissa käytetään myös joitain suomenkielisiä termejä.

Testit näyttävät hyviltä, ja ne läpäistään. Myös Jacocon kattavuusraportti löytyy, mutta testejä on tehty vain yhdelle luokalle, joten kattavuus jää pieneksi.

Kaikkia asetettuja tavoitteita ei ole saavutettu tässä vaiheessa - “kyky yhdistää kokeita listaksi” puuttuu ja kaikkea toiminnallisuutta ei ole testattu.

Parannusehdotukseksi voisin ehdottaa sitä, että ohjeisiin voisi lisätä myös kuvia.

- ➔ rajoita kenttiin syötettävän tekstin määrää
- ➔ virheilmoitus, jos ei mitään tallennu tietokantaan
- ➔ suomenkieliset termit pois \*
- ➔ unused imports pois \*
- ➔ kattavampaa testausta \*
- ➔ kuvia

2,5t: koodia ja ihmettelyä: unused imports pois – oman vertaisarvioinnin kommentteja tutkimaan ja korjaamaan. Lisäksi poistin suomenkieliset termit pois. Testaus on kattavaa, mutta käyttöliittymää ei ole JUnitilla testattu, kuten ei vaaditakaan. Kuvia lisätään, jos ennätetään. Se on sitten lopuksi. Asetetaan pituusrajat kokeiden lisäämiseen. Autentikaatio todennäköisesti jätetään tässä vaiheessa pois ohjelmasta.

Seuraavaksi:

- Kyky luoda käyttäjiä: luokkaan, tietokantaan, käyttöliittymässä Tässä siis maintenance luo käyttäjät,
- kyky luoda oma lista: view-näkymään : click on experiment, popup-> add to my list
  - o miten kuljetetaan tietoa käyttäjistä – saisiko parametrinä view-näkymään

30.3.2019 lauantaina 5,5t

5,5t koodausta ja selvittelyä, miten kannattaa tehdä käyttäjien lisääminen.

Lopputulos näistä tunteista: Kykenen luomaan **käyttäjän** ja viemään sen tietokantaan. Kykenen palauttamaan käyttäjän tiedot tietokannasta ja tuomaan ne käyttöliittymään (toistaiseksi vain default-maintenance).

Käyttöliittymässä kyky luoda lisää käyttäjiä, selata käyttäjien tietoja ja vaihtaa salasana. Näitä toiminnallisuuksia ei ole vielä muualla kuin käyttöliittymässä.

Lisäksi: löysin keinon rajoittaa textAreaa niin, ettei tule vaakatasossa scrollia eli teksti pysyy ikkunassa (ellei ole liian pitkä). ListViewlle ominaisuus vielä puuttuu.

## Vaatimusmäärittelyn päivittäminen

31.3.2019 sunnuntaina 11t

- 5,5t koodausta kirjautumistoiminto (**login**), tarkistetaan, onko käyttäjää tietokannassa, **näkymien erottaminen** käyttäjäprofiilin mukaan. JIHUU! Viimein toimii.
- 30min vertaisarviointia toisen ohjelmasta
- 3,5t **salasanan vaihto** toimii ja virheilmoitukset toimii. **Käyttäjiä voi lisätä ja poistaa** käyttöliittymässä, tallentuvat tietokantaan ja poistuvat sieltä. User-tyypin käyttäjä kykenee näkemään **omat tietonsa ja vaihtamaan salasanansa**.
- raportointia 1t
- tulostuksen selvitystä 30min

1.4.2019 maanantaina 4,75t

klo 12-> 14.30 : view own on rikki.

Klo 21.15->22.30: käyttäjä kykenee näkemään omat kokeensa, lisäämään kokeita omalle listalleen ja poistamaan kokeita omalta listaltaan. **Lista-funktionaalisuus** toimii

2.4.2019 tiistaina 5t

30min refaktorointia ja testauskattavuuden tutkimusta

klo11.30->14.30 **print** toimii!! Testausta.

21.15->22.45 testausta

4.4.2019 torstai 0,5t

Jar-tiedoston generointi ja ohjelmiston palautus vertaisarvioitavaksi

7.4.2019 sunnuntaina 1,5t

javadocien ihmettelyä

11.4.2019 torstaina 2t

klo 16->18 javadoc **toimii**

14.4.2019 sunnuntaina 1,5t

vertaisarvioinnin tekoa noin 1t.

Oman työn vertaisarviontiin tutustuminen ja ratkaisujen pohdinta: 0,5t

[Antti H T Kosonen](#) - su, 14 huhti 2019, 21:05Arvioin tämän ohjelmiston nyt toista kertaa, ja se on selkeästi edistynyt 2. iteraation palautuksen jälkeen. Kolmannelle iteraatiolle asetetut tavoitteet on saavutettu.

Ohjelmiston toiminta on nopeaa ja selkeää, joskin sisäänkirjautumista varten täytyi käydä etsimässä Readme:sta tunnukset. Mahdollisesti sisäänkirjautumisruutuun voisi lisätä mahdollisuuden luoda tunnukset itse? Ohjelmiston nykyisessä versiossa vain admin-oikeuksilla varustettu käyttäjä voi lisätä user-käyttäjiä. Ohjelmisto ei kaatunut missään vaiheessa.

Onnistuin maintenance-käyttäjänä toimiessani deletoimaan tuon käyttäjän tietokannasta, jolloin ohjelmistolla ei ollut enää yhtään admin-tason käyttäjää, jolloin uusia käyttäjiäkään ei voi enää lisätä. Tämä on aika selkeä vika -

admin-tason käyttäjän deletoinnissa voisi aina katsoa, ettei viimeistä adminia pysty poistamaan tietokannasta. Jostain syystä en pystynyt enää ollenkaan kirjautumaan sisään maintenance-tunnuksilla, vaikka latasin alkuperäisen tietokannan uudestaan zipistä.

Ohjelmistossa myös on edelleen sama (pieni) bugi jonka löysin edellisessä iteraatiossa: jos kentiin laittaa enemmän tekstiä kuin tietokanta pystyy käsittelemään, käyttäjä ei saa tästä mitään virheilmoitusta. Ohjelmisto ei kuitenkaan kaadu edelleenkään.

Koodi on taas kerran varsin selkeää ja helposti seurattavaa. Kommentointi on hyvää. Rivit ovat ajoittain varsin pitkiä - niiden selkeyttämiseksi esimerkiksi funktiokutsuja tai argumentteja voi laittaa päällekkäin, sensijaan että ne laittaa riviin. Eli esimerkiksi tämän rivin:

```
this.tryWithKids.createExperimentAndSave(subjectToggle, topic.getText(),
durationToggle, waitTime.getText(), materials.getText(), directions.getText(),
notes.getText(), theScience.getText());
```

voi esittää myös paljon helpommin luettavassa muodossa:

```
this.tryWithKids.createExperimentAndSave(
subjectToggle,
topic.getText(),
durationToggle,
materials.getText(),
waitTime.getText(),
directions.getText(),
notes.getText(),
theScience.getText()
);
```

Testauskattavuus on riittävällä tasolla ja JaCoCo-dokumentaatio on ajettu. Tein tässä suhteessa virheen edellisen iteraation arvioinnissa, sillä en tuolloin ottanut huomioon, että GUI-luokkia ei tarvitse tehtävänannon mukaan testata.

Jar-tiedostoja löytyy target-kansiosta kaksin kappalein, mutta en onnistunut kummarkaan käynnistämisessä. Sain virheilmoituksen "no main manifest attribute, in TryWithKids-1.0-SNAPSHOT.jar" ensimmäisestä ja "Error: Could not find or load main class trywithkids.gui.GUI.java" toisesta. Tämä saattaa johtua käyttöjärjestelmäeroista.

Javadoc-dokumentaatiota ei ohjelmistosta ollut ajettu, mutta kommentit oli kirjoitettu oikein, ja sain itse ajettua tuon dokumentaation projektikansioon. Javadoc-dokumentaatio on kattavaa.



Satu M Korhonen - su, 14 huhti 2019, 21:28 Kiitos arviosta! Tuon viimeisen adminin deletoiminen on bugi, joka on tarkoitus tässä seuraavassa iteraatiossa poistaa. Tuota .jar toimimattomuutta en ollut huomannut. Pitää testata. Hyviä huomioita!

16.4.2019 Tiistaina 8,5t



klo 11-15 koodin siistimistä, parametrien sijoittamista usealle riville luettavuuden lisäämiseksi.

Korjataan bugia, jossa kaikki maintenance (admin) tyyppin käyttäjät saa deletoitua. Ainakin yksi pitää olla. Jos ei ole, default-palautetaan käyttöön.

**Toimii ja on testattu (aina väh yksi admin)**

Korjataan bugia, jossa liian pitkää tekstiä yritetään lisätä tietokantaan eikä saada virhettä siitä, ettei onnistu. **Fixed**

Selkiytetty käyttäjien lisäämisen kohtaa käyttöliittymässä

Täydennetty readme

Klo 20 -20.30 vaatimusmäärittelyn päivittäminen

Jar-tiedoston toiminnan korjaaminen (shaded jar toimii).

Loppuraportin kirjoittamista: Käyttöopas

-> 00.30.

17.4.2019 keskiviikkona 4,25t

klo 10.45->15 loppuraporttia + bugien korjausta.

18.4.2019 torstaina 4t

klo 02.30-06.30 UML-kaavioita ja loppuraporttia (käyttöopasta)

19.4.2019 perjantaina 3t

neljännen iteraation palautus

klo 21->24 Loppuraporttia (JUnit-testausta)

21.4.2019 sunnuntaina 8,5t

Järjestelmätestausta klo 9.30- 12.30.

Testauksen raportointia -> klo 14

Klo 20.30->00.30 Sovelluslogiikan sanallinen selittäminen, lisenssien tarkistus ja kirjoitus, muiden osa-alueiden viilausta

22.4.2019 maanantaina 4t

kuviot sovelluslogiikkaan + loppuraportin hiomista (lähinnä luokkakaaviota ohjelmasta) klo 20.00-24.00

23.4.2019 tiistaina 4t

Sekvenssikaaviot sovelluslogiikkaan ja lopputyön kirjoitusta klo 8-12

24.5.2019 keskiviikkona 1t

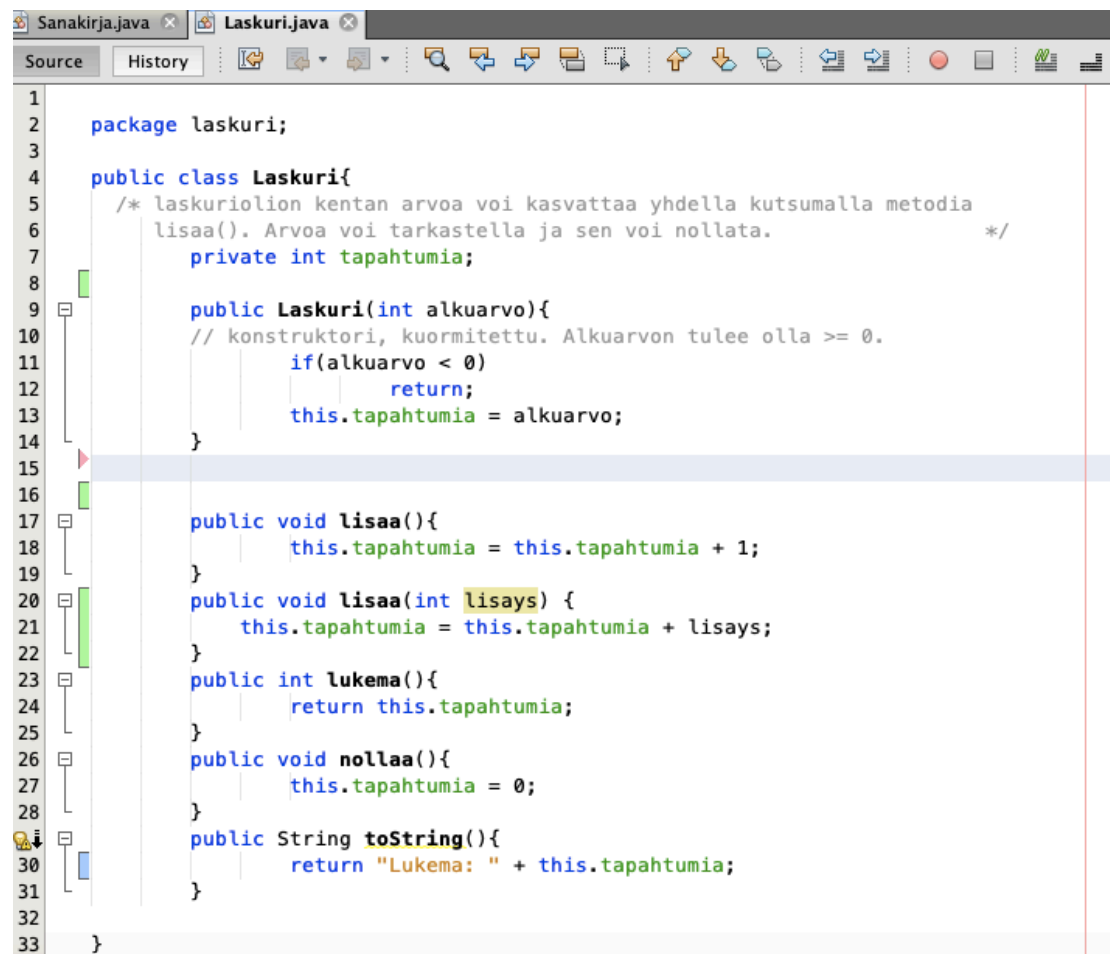
viimeistelyä

Tunteja toistaiseksi yhteensä: 125t

HUOM: mukaan ei ole laskettu niitä tunteja, kun ratkaisuja on pohdittu muuta samalla tehdessä. Esim. ruokaa laittaessa on helppo samalla miettiä, miten rajaa ohjelman toiminnallisuuksia (vaatimusmäärittely) tai deletei käyttäjiä ohjelmasta. Todellinen tuntimäärä on siis vielä suurempi. Tässä näkyy vain ne tunnit, jotka olen kirjoittanut, koodannut, googlannut, testannut ja raportoinut.

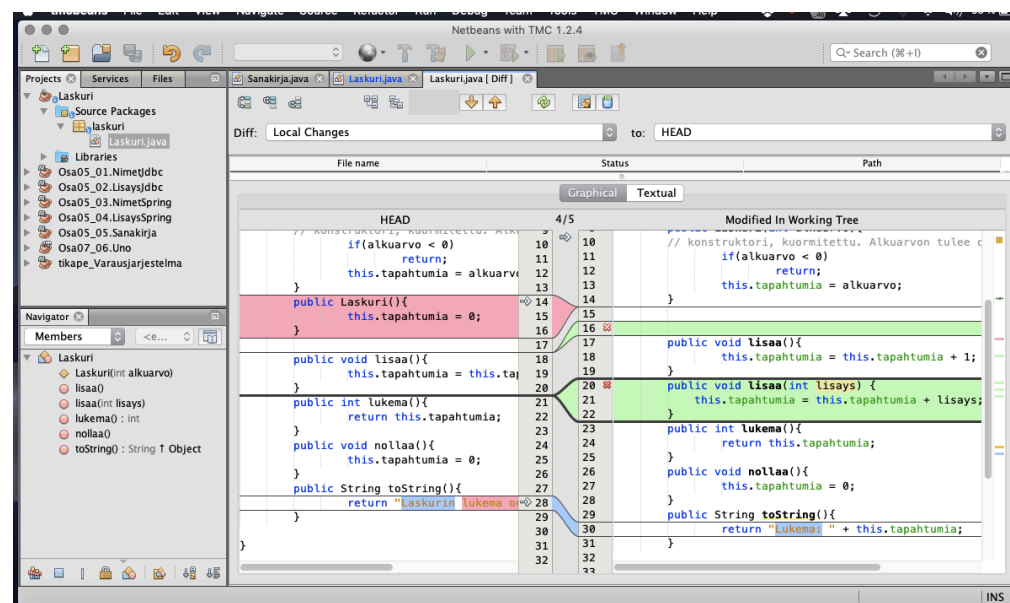
## Git ja Github

### 1. Muutokset väritettynä



```
1 package laskuri;
2
3
4 public class Laskuri{
5     /* laskuriolion kentan arvoa voi kasvattaa yhdellä kutsumalla metodia
6        lisaa(). Arvoa voi tarkastella ja sen voi nollata. */
7     private int tapahtumia;
8
9
10    public Laskuri(int alkuarvo){
11        // konstruktori, kuormitettu. Alkuarvon tulee olla >= 0.
12        if(alkuarvo < 0)
13            return;
14        this.tapahtumia = alkuarvo;
15    }
16
17
18    public void lisaa(){
19        this.tapahtumia = this.tapahtumia + 1;
20    }
21    public void lisaa(int lisays) {
22        this.tapahtumia = this.tapahtumia + lisays;
23    }
24    public int lukema(){
25        return this.tapahtumia;
26    }
27    public void nollaa(){
28        this.tapahtumia = 0;
29    }
30    public String toString(){
31        return "Lukema: " + this.tapahtumia;
32    }
33 }
```

### 2. Diff to HEAD -ikkunasta



### 3. Switch-ikkuna

Switch To Selected Branch

Select branch you want to switch to

Branch:

Commit ID:

Author:

Date:

Message:

☐ Checkout as New Branch

Branch Name:

### 4. Github-projekti

Stats < In Search for Better... Avo\_Korpiemies\_AYTKT200... Tietokantojen perusteet -... TJSP: Tehtävä 7 Using Git Support in NetB... x satumk/laskuri

Search or jump to... Pull requests Issues Marketplace Explore

satumk / laskuri Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

No description, website, or topics provided. Edit

Manage topics

2 commits 2 branches 0 releases 1 contributor

Your recently pushed branches:

commit1Kehityshaara (3 minutes ago) Compare & pull request

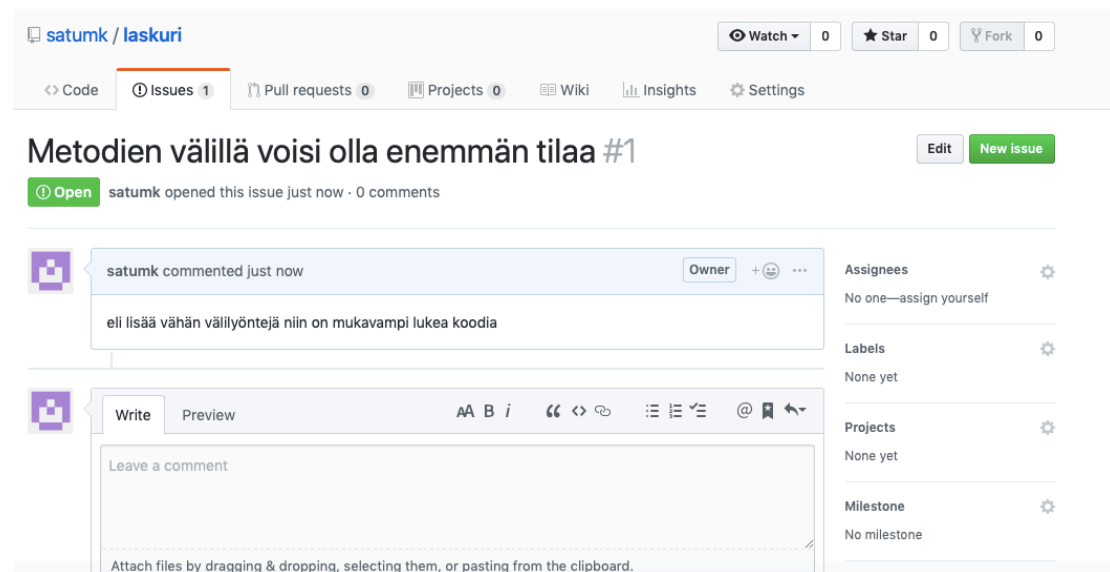
Branch: master New pull request Create new file Upload files Find file Clone or download

satumk commit 2 - parametritön konstruktori Latest commit 1468c98 19 minutes ago

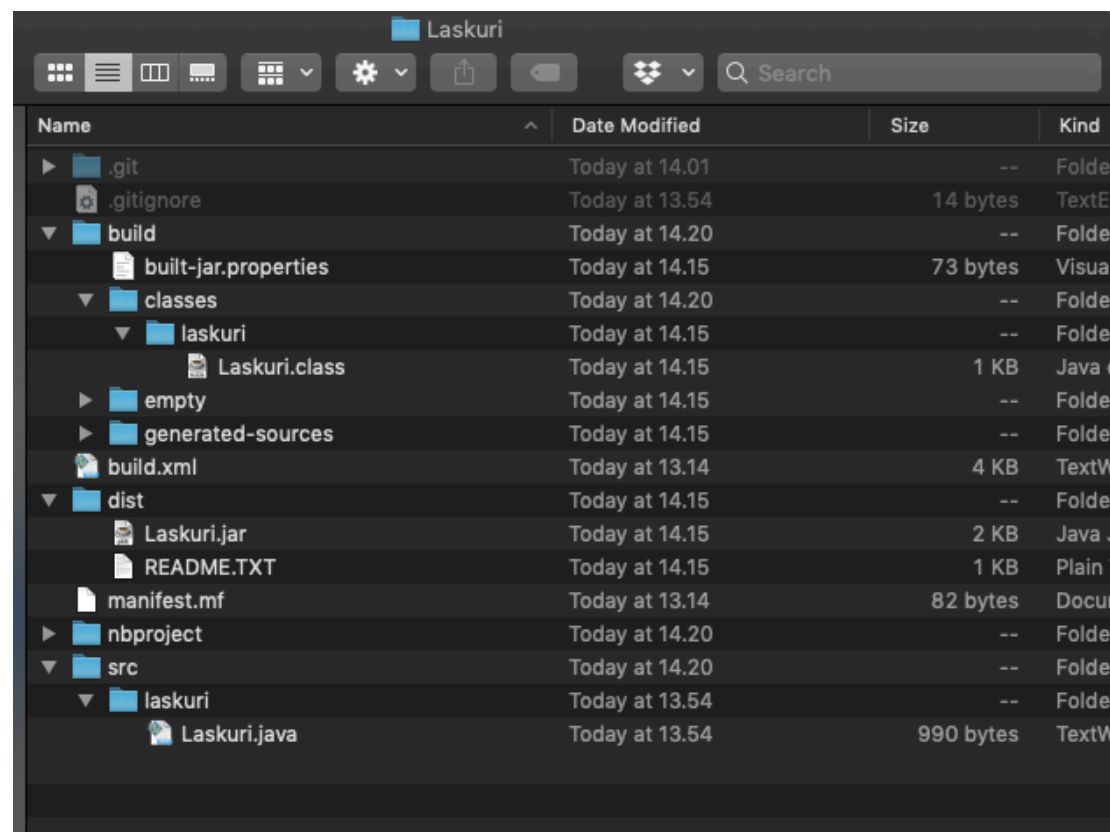
src/laskuri commit 2 - parametritön konstruktori 19 minutes ago

Help people interested in this repository understand your project by adding a README. Add a README

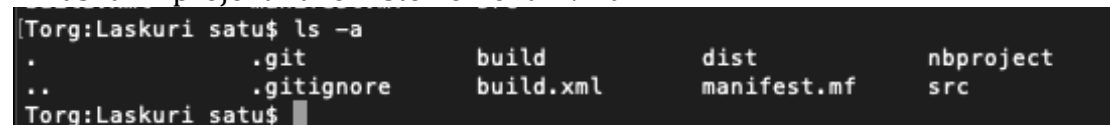
## 5. Issue-viesti Githubissa



## 6. Laskurin projektihakemisto



## 7. Laskurin projektihakemisto komentirivillä



#### 8. Suoritetun jar-tiedoston tulostus



















```
Laskuri.jar    README.TXT  
[Torg:dist satu$ java -jar Laskuri.jar  
Lukema: 5  
Torg:dist satu$ █
```

## Testausta – viikko 2

### 100% Kattavuus

 MuistavaLaskuri >  muistavalaskuri.domain >  MuistavaLaskuri

#### MuistavaLaskuri

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
 toString()		100%		n/a	0	1	0	1	0	1
 lisaa()		100%		100%	0	2	0	4	0	1
 MuistavaLaskuri(int)		100%		100%	0	2	0	6	0	1
 MuistavaLaskuri()		100%		n/a	0	1	0	3	0	1
 nollaa()		100%		n/a	0	1	0	2	0	1
 lukema()		100%		n/a	0	1	0	1	0	1
Total	0 of 58	100%	0 of 4	100%	0	8	0	17	0	6

#### Testit-tiedoston koodi:

```
package muistavalaskuri.domain;
```

```
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.*;
```

```
public class MuistavaLaskuriTest {

    MuistavaLaskuri m;

    @Before
    public void setUp() {
        m = new MuistavaLaskuri();
    }

    @Test
    public void parametrinKonstruktoriAlkuarvoNolla() {
        assertEquals(0, m.lukema());
    }

    @Test
    public void alkuarvoEiNegatiivinen() {
        MuistavaLaskuri m2 = new MuistavaLaskuri(0);
        assertEquals(0, m2.lukema());
    }

    @Test
    public void alkuarvoNegatiivinen() {
        MuistavaLaskuri m2 = new MuistavaLaskuri(-1);
        assertEquals(0, m2.lukema());
    }

    @Test
    public void lisaaToimii() {
        m.lisaa();
    }
}
```

```

    assertEquals(1, m.lukema());
}

@Test
public void nollaaToimii() {
    MuistavaLaskuri m3 = new MuistavaLaskuri(3);
    m3.nollaa();
    assertEquals(0, m3.lukema());
}

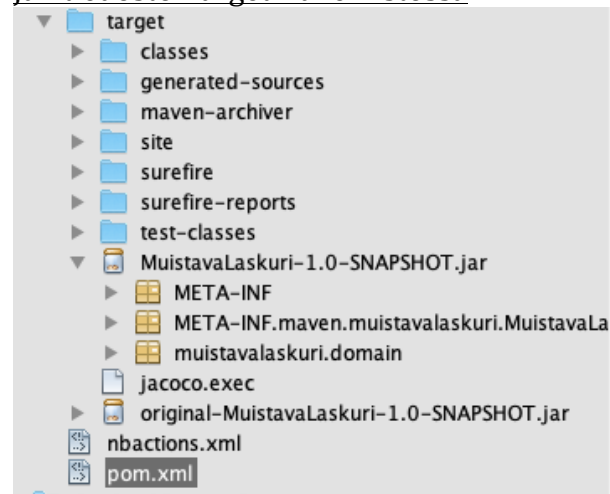
@Test
public void tekstiksiToimii() {
    MuistavaLaskuri m4 = new MuistavaLaskuri(3);
    assertEquals("Laskurin lukema on 3 (max. 3)", m4.toString());
}

@Test
public void lisaaMaksiminYliTapahtumia() {
    m.lisaa();
    m.lisaa();
    m.lisaa();
    assertEquals(3, m.lukema());
    assertEquals("Laskurin lukema on 3 (max. 3)", m.toString());
}

@Test
public void tapahtumiaEriKuinMaksimi() {
    MuistavaLaskuri m5 = new MuistavaLaskuri(5);
    m5.nollaa();
    m5.lisaa();
    assertEquals(1, m5.lukema());
    assertEquals("Laskurin lukema on 1 (max. 5)", m5.toString());
}
}

```

#### Jar-tiedosto Target-hakemistossa



## Vertaisarvio 1

Vertaisarvio ohjelmistotekniikan kurssille

Arvioija Satu Korhonen

31.3.2019

Arvioitava ohjelmisto: Henkilöstöhallinnan ohjelmisto

Tekijä: Annika Heino

Ohjelmiston .zip-tiedostosta puuttui driver (sqlite-jdbc-3.27.2.jar), joka piti hakea netistä, jotta sai ohjelmiston toimimaan. Kun sen oli löytänyt, ohjelmisto käynnistyi hyvin ja kehitys näyttää lähteneen tehokkaasti ja vauhdilla liikenteeseen. Ohjelmistossa oli toistaiseksi tekstikäyttöliittymä, jossa kaikki ominaisuudet ainakin päätasolla toimivat. Ohjelmassa oli

- person-luokka, jossa on henkilön tiedot (nimet, syntymäaika, sukupuoli) sekä getterit ja setterit ja toString,
- Employee-luokka, jossa on työsuhteen tiedot + person-luokan ilmentymä
- Persons-luokka, jossa on lista person-luokan ilmentymiä. Tähän listaan tehdään mm. getPerson-pyyntö. AddPerson ja deletePerson -metodit keskustelevat tietokannan kanssa ja tallentavat tai deletoivat henkilön tietokannasta sekä päivittävät listan.
- personnel-luokka, jossa on map työntekijöistä. addEmployee-lisää työntekijän tietokantaan ja uudistaa mapin. Update tekee samoin. Delete myös. Haku tehdään mappiin (kun haetaan yksittäistä henkilöä)
- utils-luokka, jossa on keino capitaliseString
- tekstikäyttöliittymä, jossa voidaan lisätä uusi henkilö (person), deletoida henkilö, deletoida työntekijä, hakea työntekijä id:n perusteella, lisätä työntekijä, ja päivittää työntekijän tiedot.

.zip-tiedostosta puuttuivat target-hakemisto, jossa olisi jacoco-raportti, documents-kansio, jossa olisi vaatimusmäärittely, sekä luokkakaaviot. Jäin miettimään, onkohan tekijä käyttänyt Netbeansin omaa export->zip-toimintoa, joka ei ota näitä mukaan. Myös javadocit pääosin puuttuivat muualta kuin testiluokasta.

Testaus on aloitettu. Toistaiseksi testataan person-luokan settereitä ja gettereitä.

Ohjelmiston toiminnot pääosin toimivat, mutta muutamia bugeja vielä löytyi.

Alla lista löytämistäni bugeista jatkokehitystä avittamaan:

- ohjelma kaatui, kun antoi kuukauden arvona 13
- kun lisäsi n sukupuoli-kohtaan, tietokanta merkkasi henkilön mieheksi
- kun lisäsi vuosilukuna 1000, ei tullut virhettä tai virheilmoitusta, vaan henkilö tallentui tietokantaan varsin iäkkäänä
- kun vuosilukuna oli 2019, myöskään tästä ei tullut virhettä tai ilmoitusta, vaan varsin nuori henkilö tallentui tietokantaan.
- vaativuusluokka 1-3 henkilön kiinnittämisessä hyväksyi arvon 4
- jos haki kohdassa "Syötä tehtävään kiinnitetyn työntekijän tunnus (Id) hakua varten: " henkilön id:n, ohjelma kaatui. Listausta vaihtoehtoja ei tullut, joista olisi voinut valita.



- suoritusarviointi kohdassa "8 - muokkaa tehtävään kiinnitetyn työntekijän tietoja tunnuksella (Id)" hyväksyi suoritusarvon 55, vaikka skaala loppui 50.

Kehitysehdotus: Tuntuu raskaalta ratkaisulta, että tiedot säilytetään sekä sovelluksessa listoina ja hashmappeina sekä tietokannassa. Ehdotan, että tiedot säilytettäisiin vain tietokannassa ja ohjelmiston sovelluslogiikka keskittyisi käsittelemään tietoja niiden säilyttämisen sijaan.

Yhteenveto: Ohjelmisto oli keskeneräinen, kuten sen pitikin tässä vaiheessa vielä olla. Siksi ohjelmistosta löytyi bugeja, kun tietoisesti testasin, miten ohjelma käsittelee virheellisiä syötteitä. Toivon näistä huomioista olevan apua tekijälle. Nähdäkseni työ on lähtenyt hyvin liikkeelle ja paljon töitä on tehty. Eli loppusanoina: hyvässä vaiheessa. Kiitos, kun sain sitä tutkia.

## Vertaisarvio 2

Vertaisarvio ohjelmistotekniikan kurssille  
Arvioija Satu Korhonen  
14.4.2019

Arvioitava ohjelmisto: Henkilöstöhallinnan ohjelmisto  
Tekijä: Annika Heino

Tämänkertainen arvio oli hankalampi kirjoittaa, koska työ ei ollut edennyt työn tekijän omankaan ilmoituksen (zipin nimi) mukaan ja zip-tiedostosta edelleen puuttui driver sekä target-kansio, jossa olisi ollut jacoco-raportti. Vaikka työ ei ollutkaan edennyt, olisin mieluusti nähnyt nämä osiot kansiossa.

Koska elämässä kuitenkin sattuu ja tapahtuu, ja välillä muut asiat vaan nousevat tärkeysjärjestyksessä opiskeluhommia korkeammalle, ei aina ennätä tehdä asioita, joita oli ajatellut. **Mielestäni ohjelman eteen on tehty paljon töitä ja nähty paljon vaivaa. Siinä suurin osa asioista toimii todella hienosti ja kuten pitääkin. Toivon, että tekijä jaksaa ja ennättää edistää ohjelmaa niin, että saa sen valmiiksi.** Sitä tarkoitusta silmällä pitäen nostan muutaman keskeisen asian viime arviosta.

- Syötteiden tarkistaminen: Voi tehdä suoraan käyttöliittymässä, jolloin virheellisiä syötteitä ei lähetetä edes sovelluslogiikkaan, tai sovelluslogiikassa. Suosittelen tämän tekoa vahvasti, sillä virheelliset syötteet voivat kaataa ohjelman.
- Tiedon tuplasäilytys sovelluslogiikan luokissa sekä tietokannassa. Kannattaa pyrkiä poistamaan tiedonsäilytys sovelluslogiikan luokissa ja pitää tieto vain yhdessä paikassa. Jos tietoa säilytetään useassa paikassa, on se ensinnäkin turhan raskasta eli ohjelma toimisi tehokkaammin, jos tieto on vain yhdessä paikassa, mutta keskeisempää on riski ristiriitaisesta tiedosta, jos ei muista jokaista paikkaa päivittää samaan aikaan ja samalla tavalla.

Jos kurssin loppuaikana kiire jatkuu eikä ohjelmaa ennätä tehdä niin pitkälle kuin vaatimusmäärittelyssä aikoi, ehdotan myös tekijälle, että hän jättää graafisen käyttöliittymän tekemättä ja keskittyy sovelluslogiikkaan. Tämä vaikuttaa pisteisiin, mutta pisteissä on selvästi esitetty sen vaikutus eli tekstikäyttöliittymä on hyväksytty lopputulos kurssin läpäisyn näkökulmasta. Minulla eniten aikaa on mennyt nimenomaan käyttöliittymän kanssa painimiseen, koska se on minulle vierainta kurssilla.

Testauksessakin voi testata ensin ylemmän tason toiminnallisuuksia ja sen jälkeen tarkistaa, mitä alemman tason (getterit ja setterit esim.) ovat vielä testaamatta. Ylemmän tason toiminnot vaativat alemman tason toimintoja, joten ne tulee testattua samassa. Tällä tavalla vähemmällä määrällä syvemmälle menevillä testeillä saa testattua ohjelmistoa laajemmin.

Loppusanoina totean, että ohjelmisto on monipuolinen siinä, minkälaista tietoa siellä on ja miten tietoa voi muokata. Tiedon saa talletettua tietokantaan ja haettua sieltä. Ohjelmassa on toimiva tekstikäyttöliittymä. Syötteitä kontrolloimalla ja testaukseen keskittymällä ohjelmiston saa pisteeseen, jossa se täyttää kurssin läpäisyn kriteerit. Toivon tekijälle onnea ja menestystä loppukurssin kanssa ja toivon, että hän ennättää hyvin alkaneeseen ohjelmistoon paneutua.