Pluto - simple course management application

Author: Trisztán Piller, trisz t4n, piller.trisztan@simonyi.bme.hu

Published: December 5th 2020

1. User documentation

1.1. Story

This is my homework solution for the subject Introduction to Programming 3 at Budapest University of Technology and Economics.

1.2. Use-cases

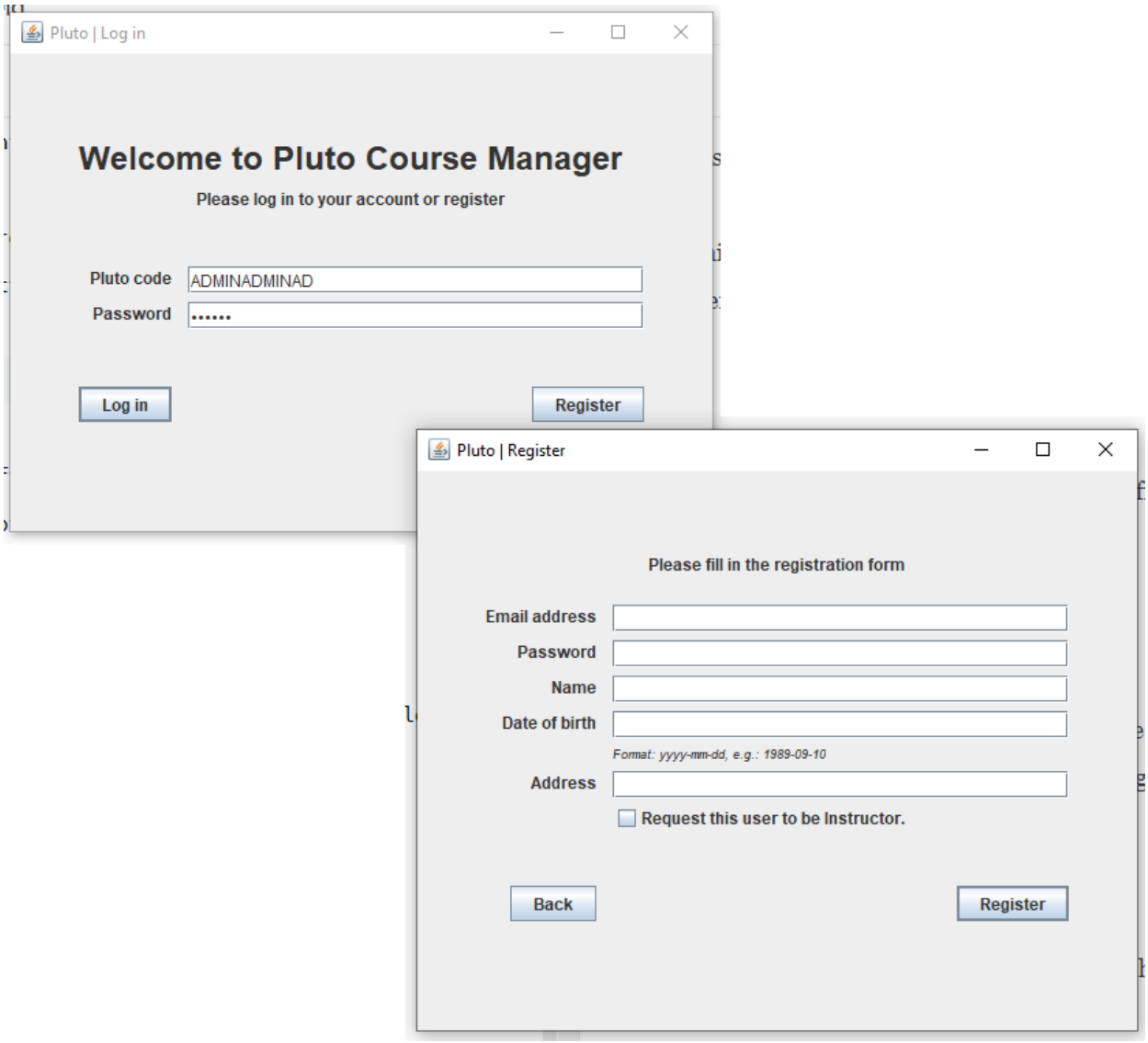
- Use-case: Administrate the database - **Administrator** may be able to do these:
    - Accept instructors' registration into the system.
    - Delete, modify users
    - Clear or seed database
    - Create subjects as coordinator, create courses under them
    - Modify subjects
    - List all subjects
    - List all courses
    - List all users
    - They cannot function as instructor of a course when attaching instructor to the course
- Use-case: Manage subjects and courses as an Instructor - **Instructor** may be able to do these:
    - List all the subjects created by me
    - List courses where I instruct
    - Modify subjects created by them
    - Create subjects as coordinator, create courses under them
    - Modify courses, kick students from course
    - Close subject for further applications from students
    - Modify profile data, password
- Use-case: Manage subjects and courses taken as a Student - **Student** may be able to do these:
    - List all the subjects, and take courses of those
    - List subjects of taken courses
    - List taken courses

- Take or drop a course
- Drop all the courses taken under a subject
- Modify profile data, password

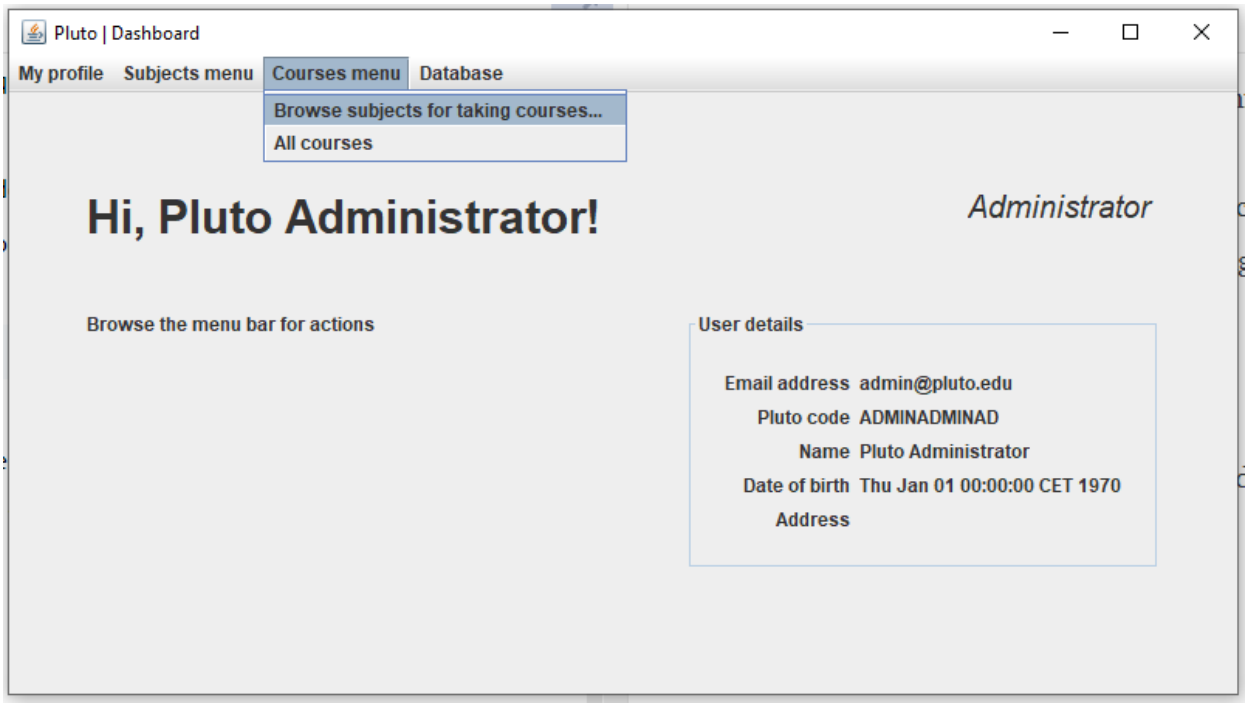
1.3. Guide to using the application

The Pluto Course Manager application works like a dynamic content management system's website.
In Pluto's system, everything is identified by a unique identifier: their Pluto code.

After opening the app, you can log in using your credentials (your designated Pluto code and set password), or you can register on the registration form page clicking 'Register' button.

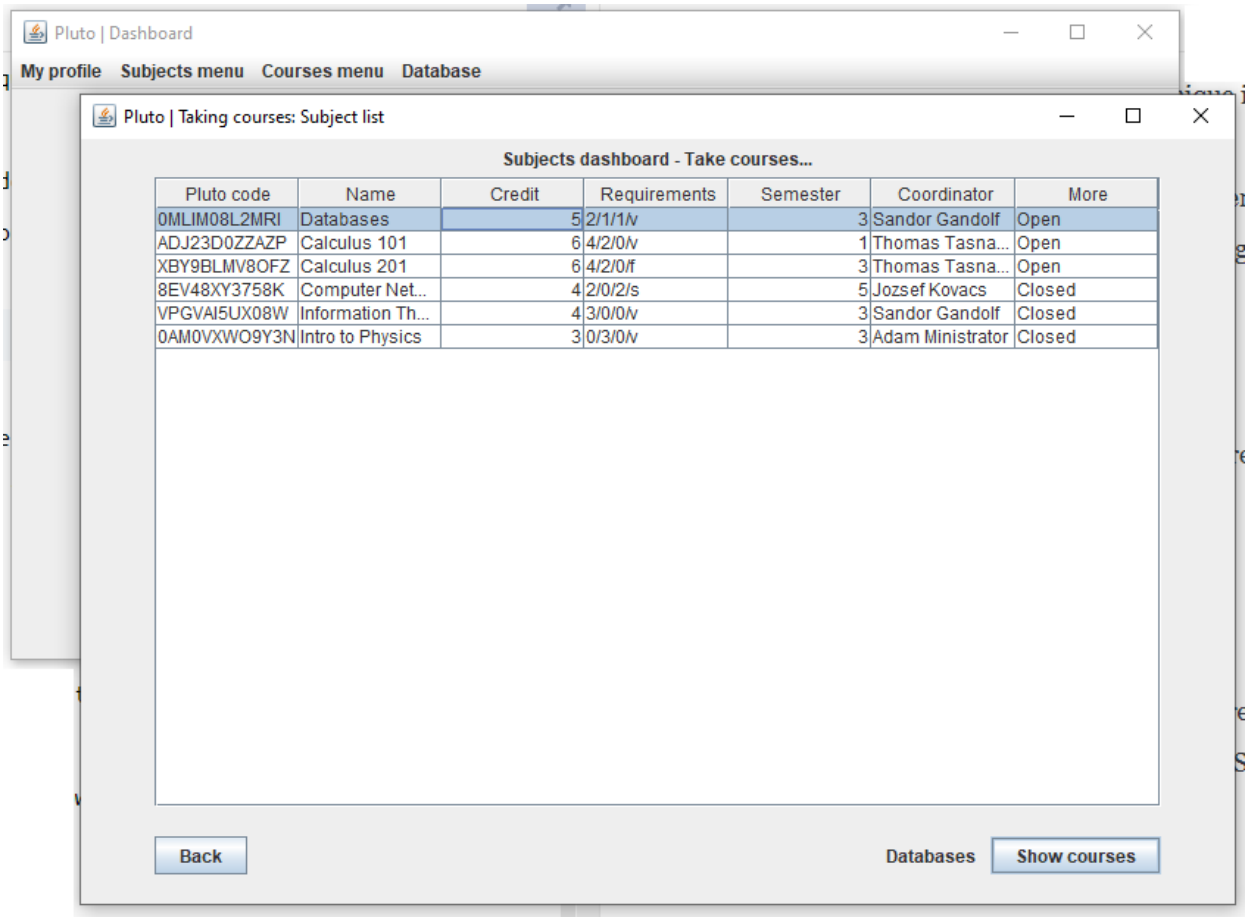


After that, on the dashboard page you can reach different functionalities, all of them are easily understandable and reachable with the help of the menu bar.

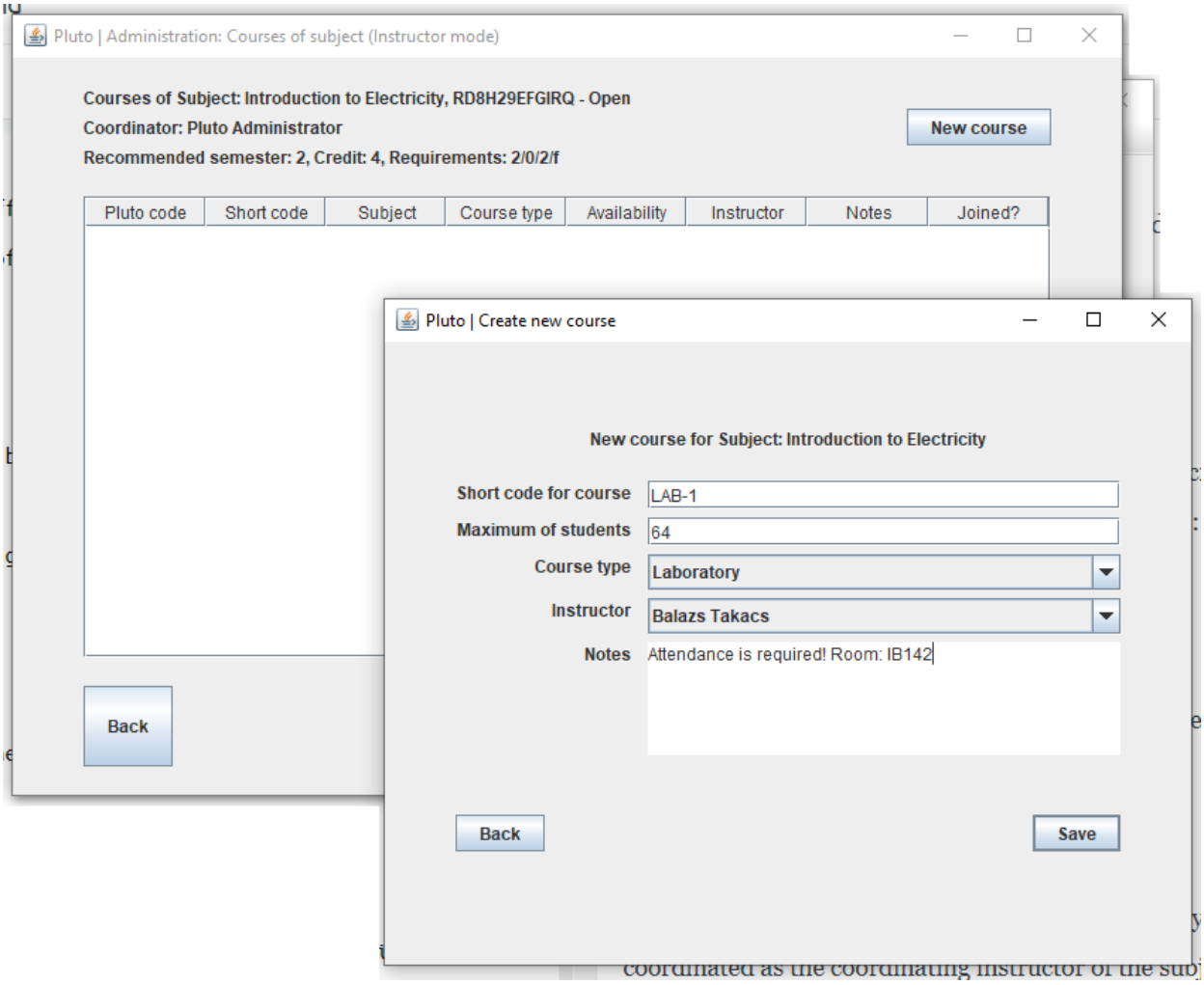


All the resources (users, subjects, courses) could be created, read, modified and deleted (CRUD functionalities).

Resources can be read/deleted via button clicks, e.g.: Show courses.



Resources can be created and then edited with the help of filling forms.



You can manage your courses taken as a student, or your courses instructed as an instructor, your subjects coordinated as the coordinating instructor of the subject, or the database all around as the administrator — this is all depending on your user profile.

2. Developer documentation

Project can be found on github.

2. 1. Environment

- Developed in JetBrains IntelliJ IDEA 2019.3.3
- Developed with Java SDK 15
- Packages downloaded from maven
    - Uses javax.json 1.4
    - Uses JUnit 5.4
    - Uses dotenv-java 2.2 ***(requires Java 8 or greater)***

2.2. Running

- Import the project into IntelliJ. ***(Recommended)***
- Import the project into Eclipse.
- Run `javac` and `java` from the command line:
    - Windows:

```
javac @.sources -d out\production\pluto -cp lib\junit-jupiter-5.4.2.jar;
lib\junit-jupiter-api-5.4.2.jar;lib\apiguardian-api-1.0.0.jar;lib\opente
st4j-1.1.1.jar;lib\junit-platform-commons-1.4.2.jar;lib\junit-jupiter-pa
rams-5.4.2.jar;lib\junit-jupiter-engine-5.4.2.jar;lib\junit-platform-eng
ine-1.4.2.jar;lib\javax.json-1.1.4.jar;lib\dotenv-java-2.2.0.jar

java -Dfile.encoding=UTF-8 -classpath out\production\pluto;lib\junit-jup
iter-5.4.2.jar;lib\junit-jupiter-api-5.4.2.jar;lib\apiguardian-api-1.0.0
.jar;lib\opentest4j-1.1.1.jar;lib\junit-platform-commons-1.4.2.jar;lib\j
unit-jupiter-params-5.4.2.jar;lib\junit-jupiter-engine-5.4.2.jar;lib\jun
it-platform-engine-1.4.2.jar;lib\javax.json-1.1.4.jar;lib\dotenv-java-2.
2.0.jar pluto.app.Application
```

    - Unix

```
javac @.sources -d out/production/pluto -cp lib/junit-jupiter-5.4.2.jar;
lib/junit-jupiter-api-5.4.2.jar;lib/apiguardian-api-1.0.0.jar;lib/opente
st4j-1.1.1.jar;lib/junit-platform-commons-1.4.2.jar;lib/junit-jupiter-pa
rams-5.4.2.jar;lib/junit-jupiter-engine-5.4.2.jar;lib/junit-platform-eng
ine-1.4.2.jar;lib/javax.json-1.1.4.jar;lib/dotenv-java-2.2.0.jar  pluto.a
pp.Application

java -Dfile.encoding=UTF-8 -classpath out/production/pluto;lib/junit-jup
iter-5.4.2.jar;lib/junit-jupiter-api-5.4.2.jar;lib/apiguardian-api-1.0.0
.jar;lib/opentest4j-1.1.1.jar;lib/junit-platform-commons-1.4.2.jar;lib/j
unit-jupiter-params-5.4.2.jar;lib/junit-jupiter-engine-5.4.2.jar;lib/jun
it-platform-engine-1.4.2.jar;lib/javax.json-1.1.4.jar;lib/dotenv-java-2.
2.0.jar pluto.app.Application
```

2.3. Development principles

- The project uses the MVC Design Pattern.
    - **M**odels communicate with **C**ontrollers via exceptions.
    - **V**iews provide interface between the user and the **C**ontroller.
    - **C**ontrollers manage data through a **M**odel,
    - **M**odels validate and represent the used entities in the Database.
- The structure is based on my experiences with the framework Ruby on Rails.
- Controllers are realized as *de facto singleton classes*: instantiated once, only passed if needed.
- Database is a *de jure singleton class*, containing only static fields and methods.
- Password encryption is secured — but might be prone to memory extractions.

Environments variables

- With the help of the `dotenv-java` package environment variables can be set in the `.env` file in the home directory.
- This way, you can set the administrator settings and the debugging settings:

```
PLUTO_ENVIRONMENT=DEV (nothing else used yet)
PLUTO_CONSOLE_COLORFUL=FALSE or TRUE
PLUTO_ADMIN_CODE=ADMINADMINAD
PLUTO_ADMIN_NAME="Pluto Administrator"
PLUTO_ADMIN_EMAIL=admin@pluto.edu
PLUTO_ADMIN_PASSWORD=qwertz
```

File system and serialization of database entities

The entities are saved as JSON objects in a JSON array, serialized into JSON files:

- `users.json`: **IMPORTANT**: courses attribute is present only by StudentModels meaning the courses taken by the user!

```
{
    "type": "Student" or "Instructor" or "Administrator",
    "courses": [String array (pluto codes)],
    "details": {
        "pluto": String,
        "email": String,
        "name": String,
```

```
        "dob": "yyyy-MM-dd",
        "address": String,
        "credentials": {
            "password": [byte array],
            "salt": [byte array]
        }
    }
}
```

- `subjects.json`:

```
{
    "pluto": String,
    "name": String,
    "credit": Number,
    "requirements": String,
    "semester": Number,
    "coordinator": String (pluto code),
    "isOpened": Boolean
}
```

- `courses.json`:

```
{
    "pluto": String,
    "shortCode": String,
    "type": Number,
    "maxStudents": Number,
    "notes": String,
    "subject": String (pluto code),
    "instructor": String (pluto code)
}
```

With saving only the...

- taken courses of students
- coordinator of subjects
- instructor and parent subject of the courses

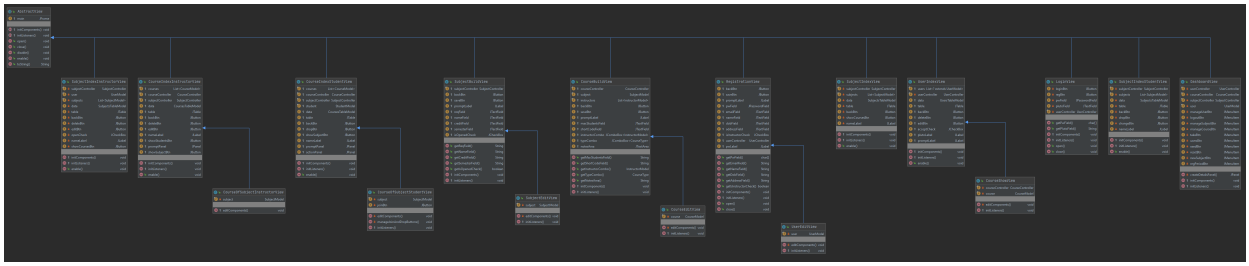
...the database is consistent, has an integrity, the loading is clear and unambiguous, every entities' associations can be fetched easily.

2.4. Documentation

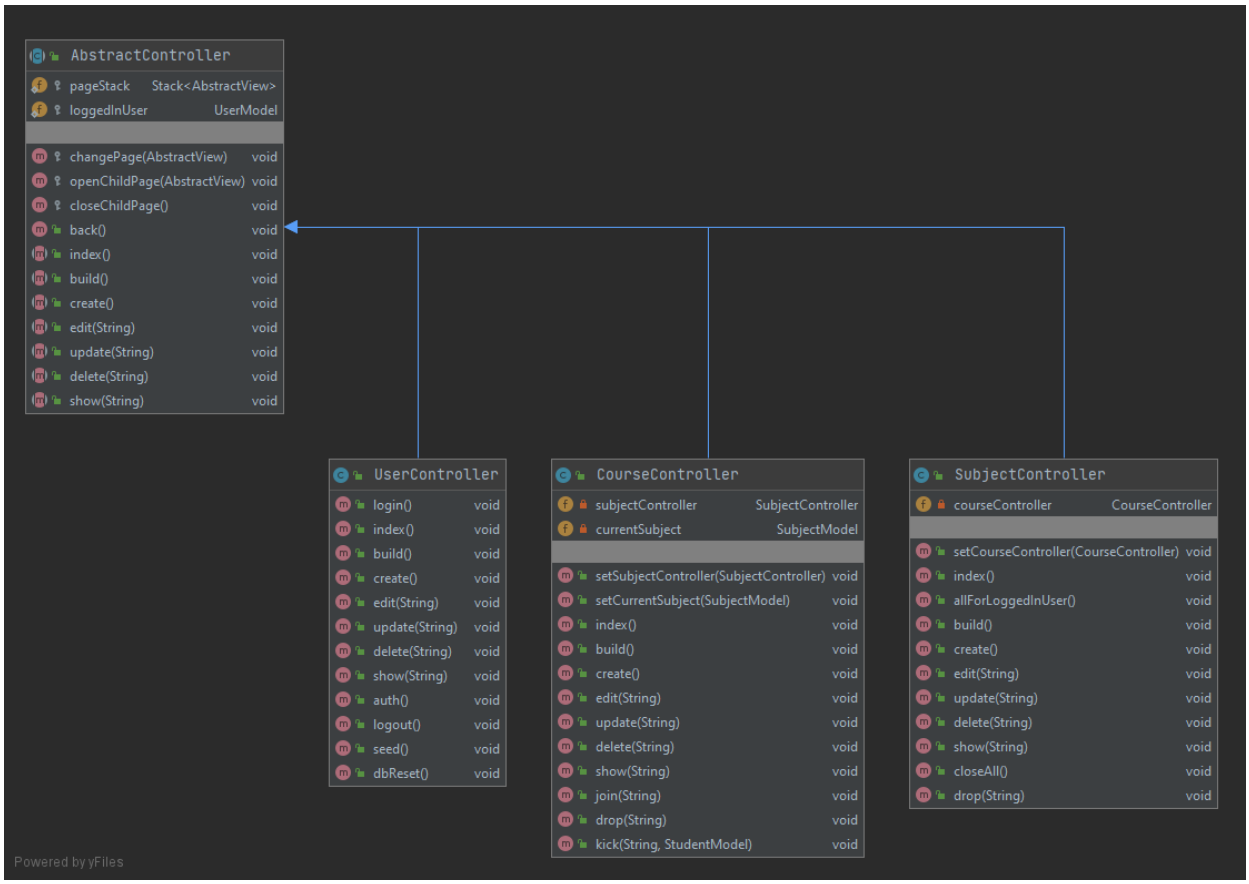
- Developer documentation is a generated html document residing in the `.\doc` directory
- Class and functional documentation can be found in the source code as Javadoc.

Class diagrams

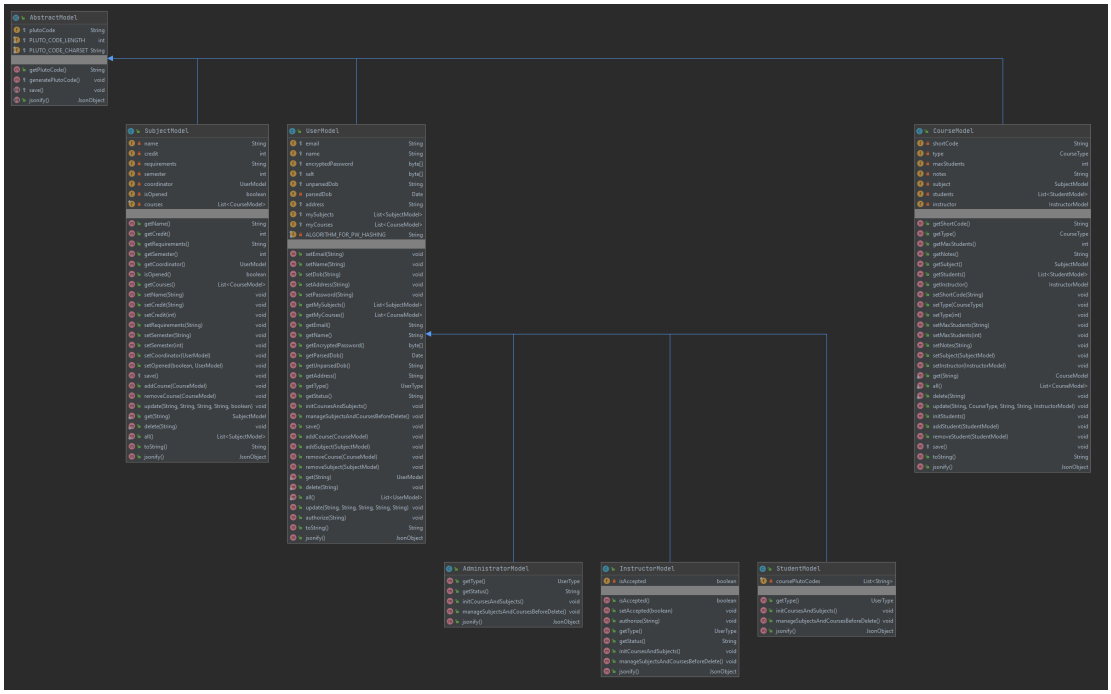
Views:



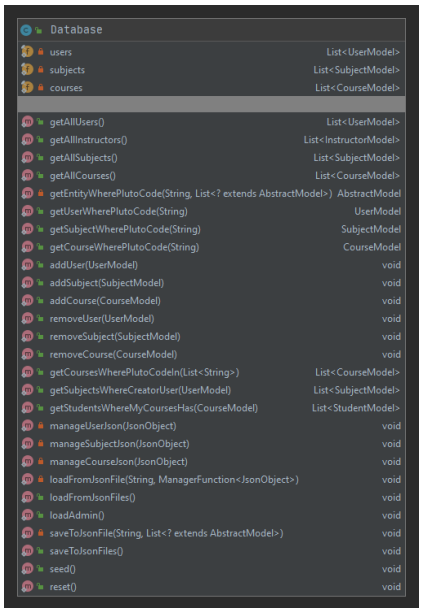
Controllers:



Models:



Database:



2.5. Testing

- The project supports JUnit 5 tests
    - Currently the models are tested with unit tests.
- After importing the project, you can run the tests with the help of the IDE.