**COLLECTIVE PROJECT**

**DOCUMENTATION**

Echipa 0:

Crețu Mihai

Darie George

Ene Andrei

Fărcaș Alexandru

Feier Mihai

Gaidur Ioan

Galan Radu

Gașpar Patricia

Gheorghiu Dragoș

Ghioci Stefan

Sikolya David

Table of Contents

[1) Introduction 3](#_Toc29537428)

[1.1. Project Scope 3](#_Toc29537429)

[1.2. References 3](#_Toc29537430)

[2) System Documentation 3](#_Toc29537431)

[2.1. Domain (Entities) 3](#_Toc29537432)

[2.2. UML Diagram 4](#_Toc29537433)

[2.3. DB Relationships 5](#_Toc29537434)

[3) User Documentation 6](#_Toc29537435)

[3.1. Functional requirements 6](#_Toc29537436)

[3.1.1. Use Cases Diagram 6](#_Toc29537437)

[3.1.2. Use Cases Scenarios 6](#_Toc29537438)

[4) Non-functional Requirements 9](#_Toc29537439)

[4.1. Performance Requirements 9](#_Toc29537440)

[4.2. Security Requirements 10](#_Toc29537441)

[5) Retrospect 10](#_Toc29537442)

[6) Bibliography 10](#_Toc29537443)

# Introduction

## Project Scope

The purpose of this project is to help the students with an easy-to-use application, to manage their activities related to the faculty (projects, attendances, exams) along with personal ones. The system is getting data from the official website of the faculty, so, it will be always up to date with the courses schedule. ( view more in the description file )

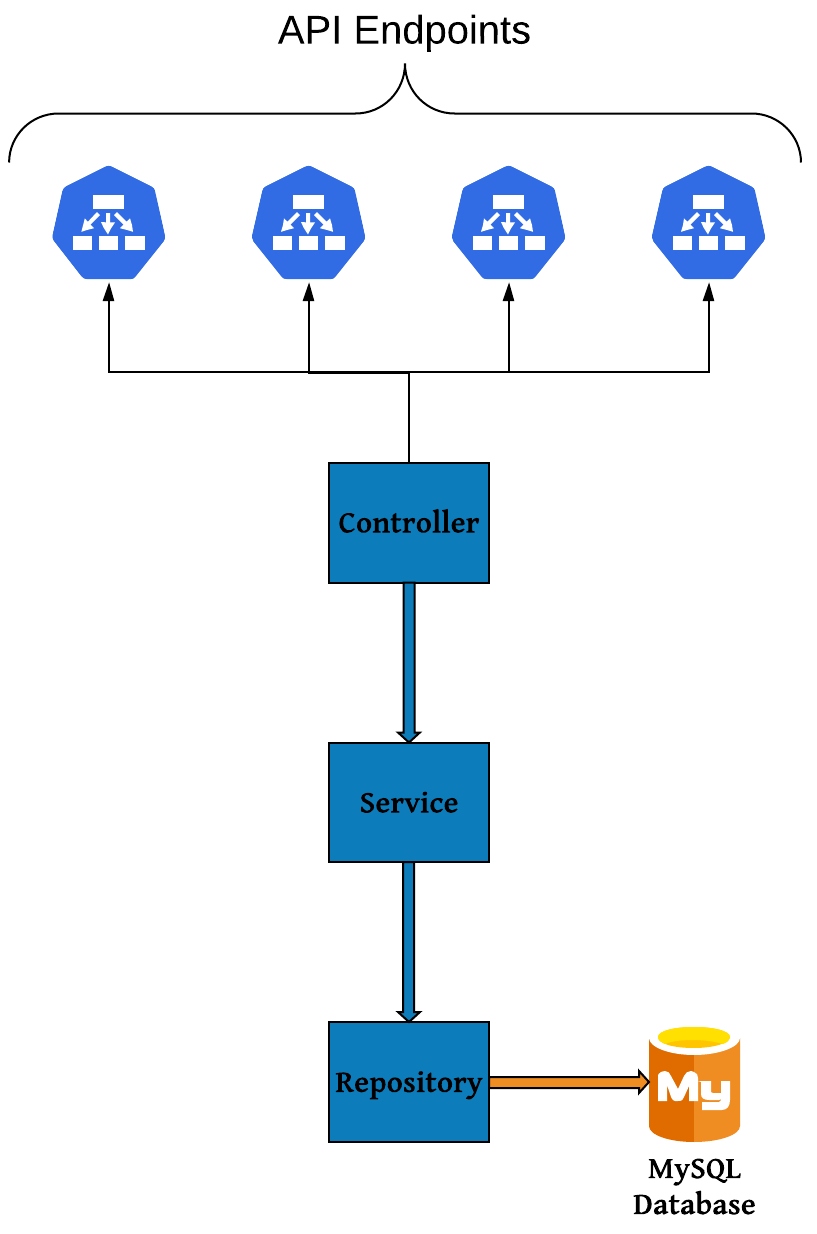
## References

-Trello : <https://trello.com/b/Ban66lz7/proiectcalendar>

-Bitbucket:

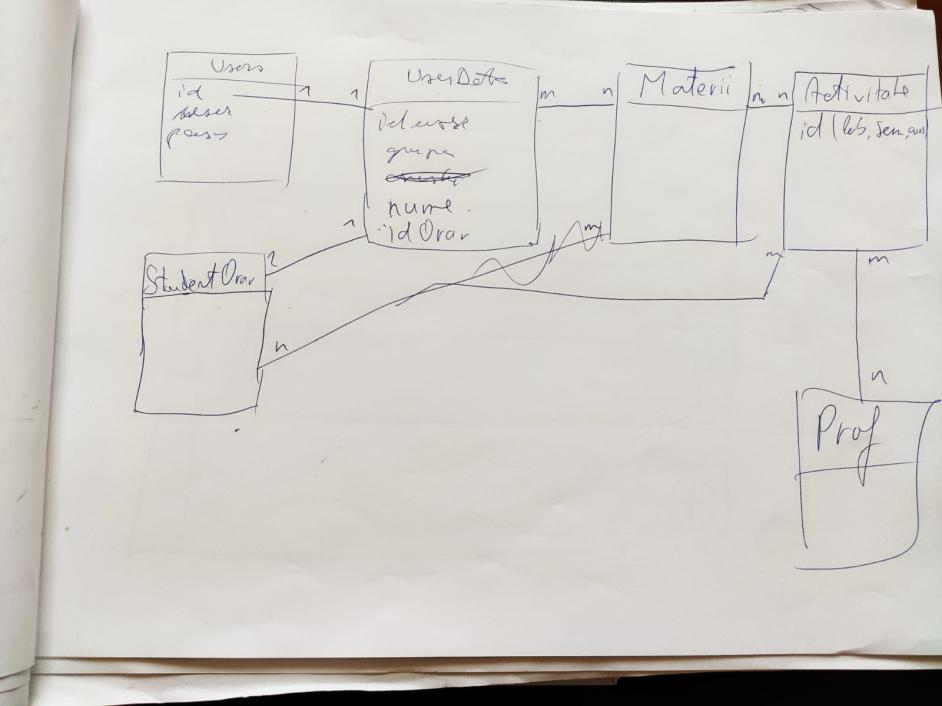
* BackEnd: <https://bitbucket.org/georgedarie/proiectcalendar>
* FrontEnd: <https://bitbucket.org/StefanGhioci/proiectcalendarfront/src/master/>

# System Documentation

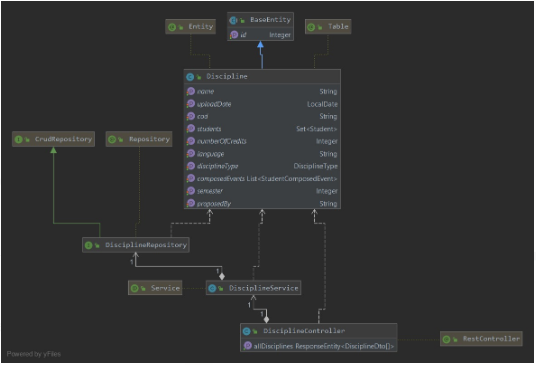


## UML Diagram

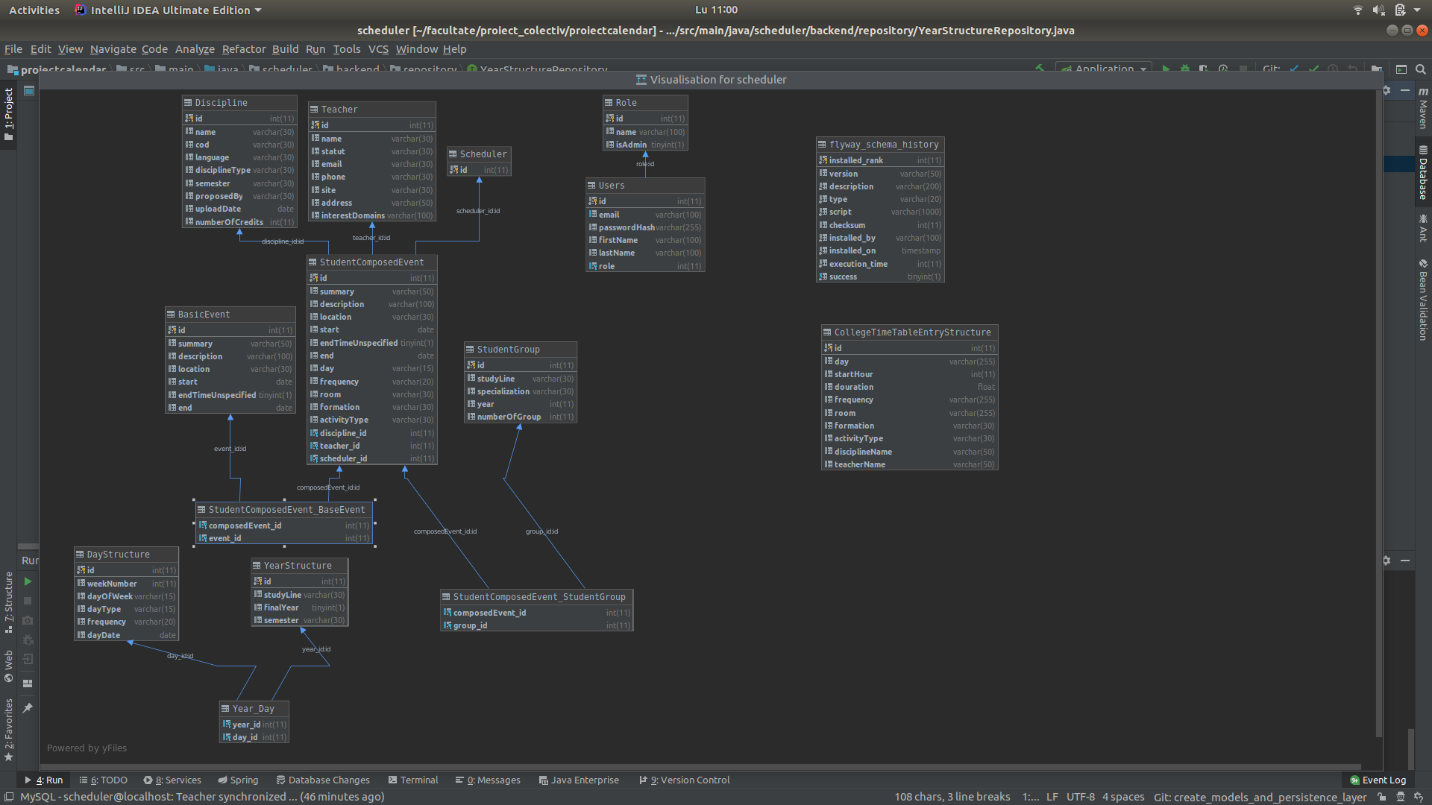
(v1)



v2 – full example of workflow for one entity (the rest of them are the same)



## DB Relationships

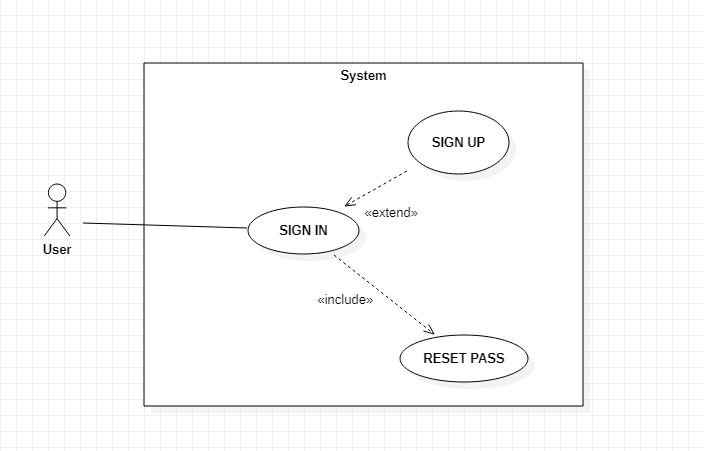


# User Documentation

## Functional requirements

### Use Cases Diagram

(Old one)



### Use Cases Scenarios

**Use Case:** SIGN UP

**Primary Actor:** User

**Brief:** The user creates a new account for the system

**Preconditions:** The account does not already exist

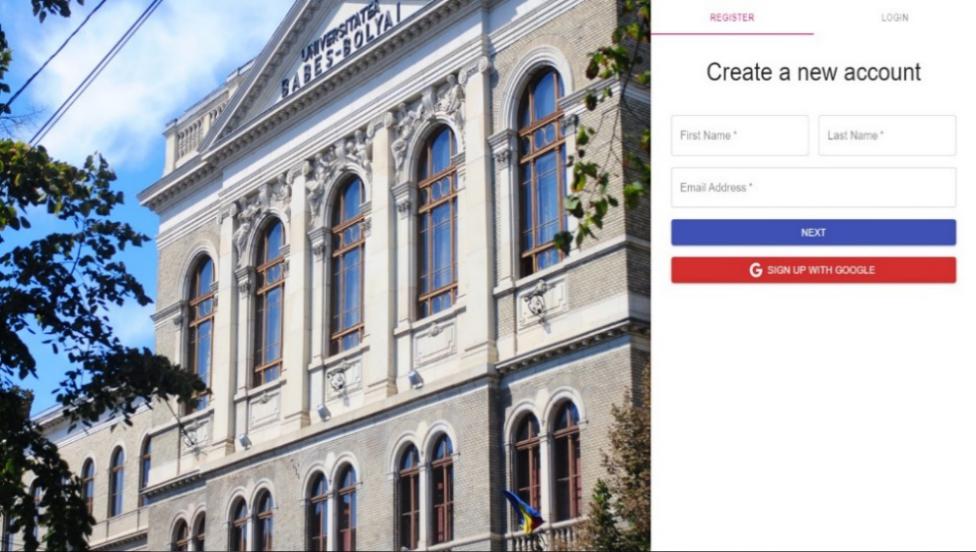
**Postcondition:**

* A new account is created
* The user can use the account to sign in to the system

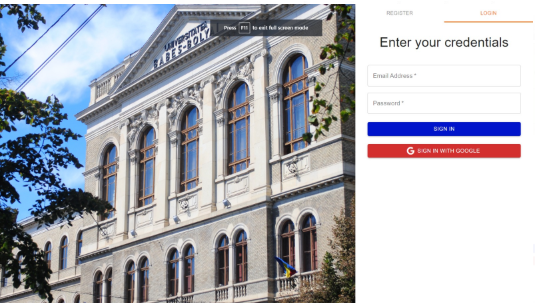
**Basic flow:**

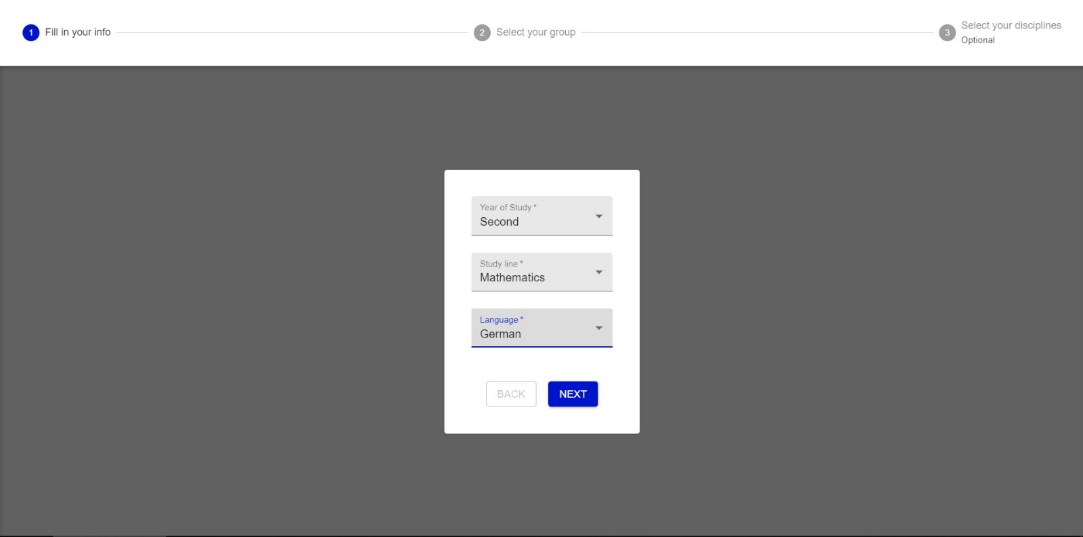
1. The user chooses “REGISTER” tab
2. The user fills the required information
3. The system creates a new account

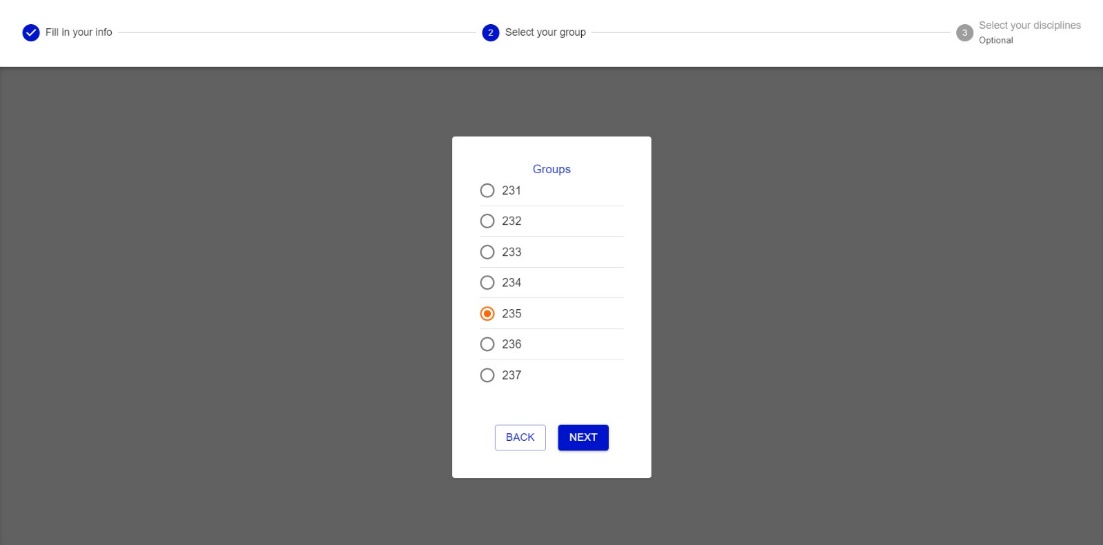
(v1)

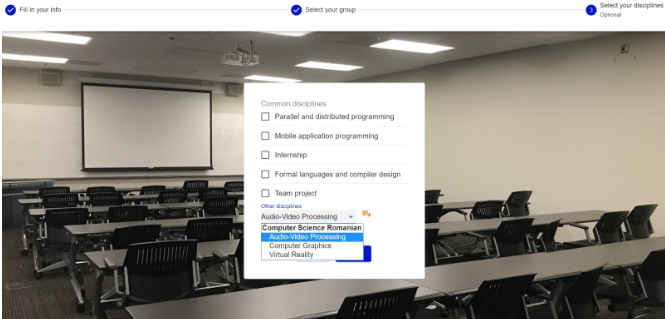


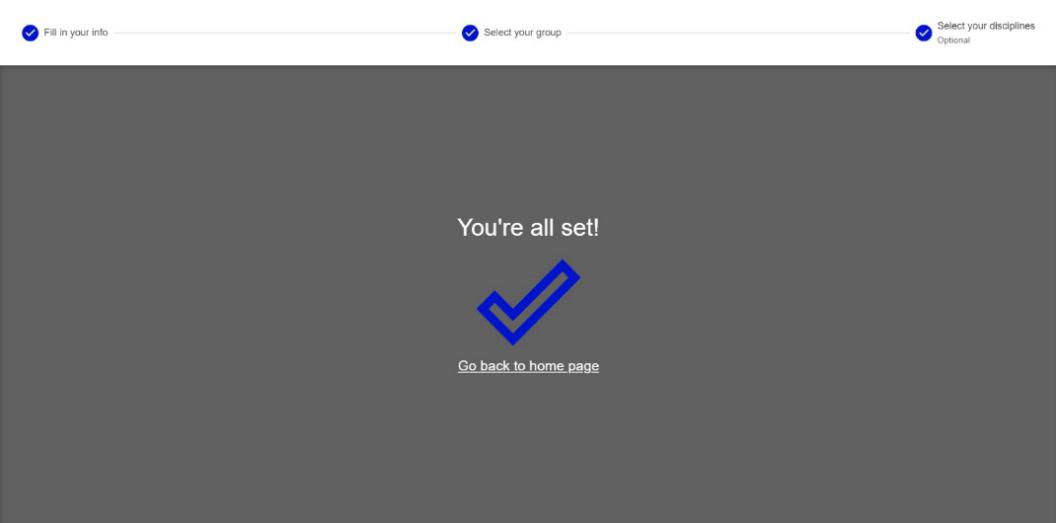
(v2)





****



****

**Use Case:** SIGN IN

**Primary Actor:** User

**Brief:** The user logs in to the system

**Preconditions:** The user has the credentials for an available account

**Postcondition:**

* The user is logged in

**Basic flow:**

1. The user types the username and the password
2. The system checks the Username and the Password
3. If all good, the user gets logged in, else, an error occurs

**Use Case:** RESET PASS

**Primary Actor:** User

**Brief:** The user changes account’s password

**Preconditions:** The user is logged in

**Postcondition:** The account has a new password

**Basic flow:**

1. The user chooses the “Reset Password” functionality
2. The user has to type the desired password and the current one
3. The account’s password is changed

**Use Case:** ADD\_EXAM

**Primary Actor:** User

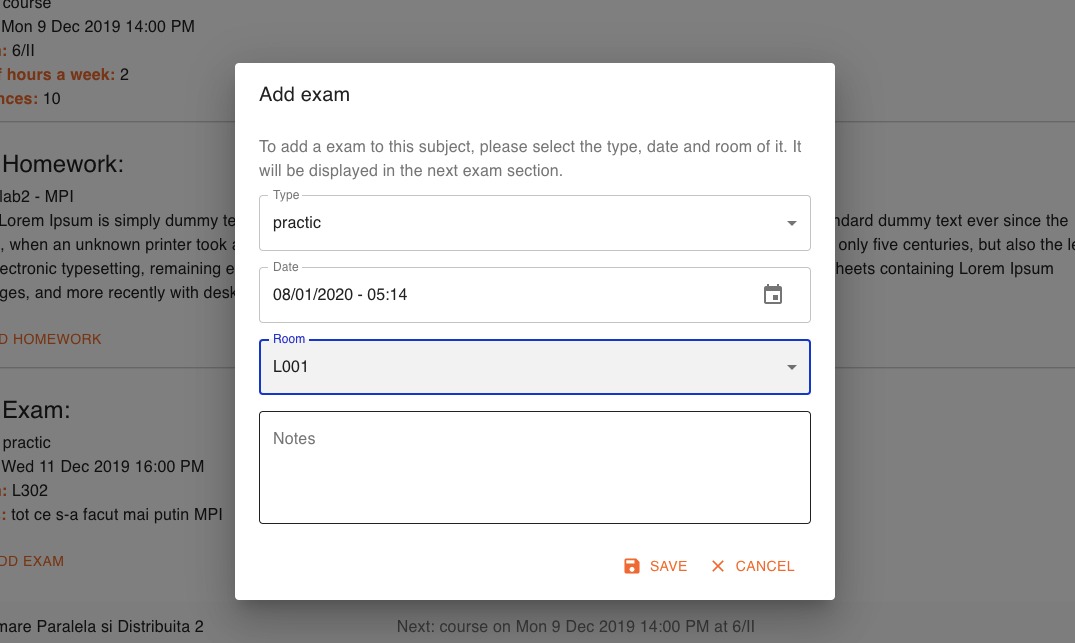
**Brief:** The user adds a new Exam (event)

**Preconditions:** The user is logged in

**Postcondition:** A new exam is added to calendar

**Basic flow:**

1. The user chooses the “Add Exam” functionality
2. The user must fill the required information
3. The event is added to calendar
4. The user is notified



**Use Case:** ADD\_HOMEWORK

**Primary Actor:** User

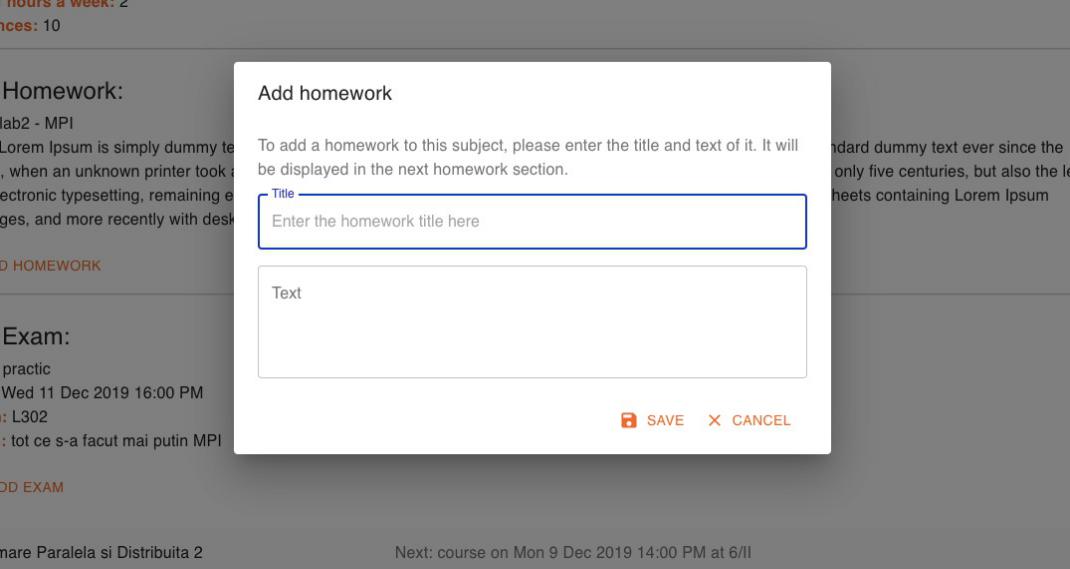
**Brief:** The user adds a new Homework (event)

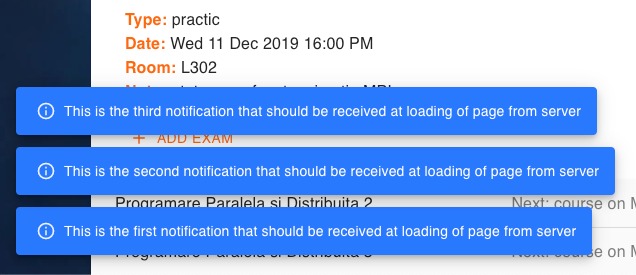
**Preconditions:** The user is logged in

**Postcondition:** A new Homework is added to calendar

**Basic flow:**

1. The user chooses the “Add Homework” functionality
2. The user must fill the required information
3. The event is added to calendar
4. The user is notified





# Non-functional Requirements

## Performance Requirements

The server take to start up a maximum of 10 seconds and for the website the time is a maximum of 5 seconds.

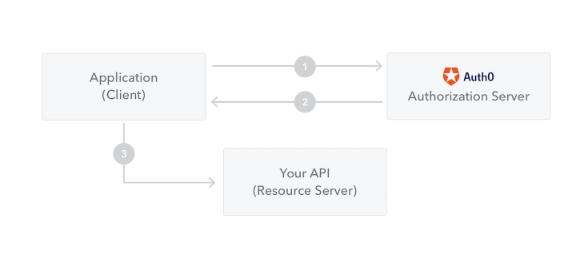
The api calls do not take more than a second.

The scrapping from all the sources takes less the 10 seconds with the populating of the database including. It can be set to update the data at any interval of time.

## Security Requirements

For security, the application is using JSON Web Token which is an open  standard that defines a compact ( it can be sent through an URL, POST parameter, or inside an HTTP header, additionally, due to its size its transmission is fast) and self-contained (the payload contains all the required information about the user) way for securely transmitting information between parties as a JSON object.

The following diagram shows how a JWT is obtained and used to access APIs or resources:



# Retrospect

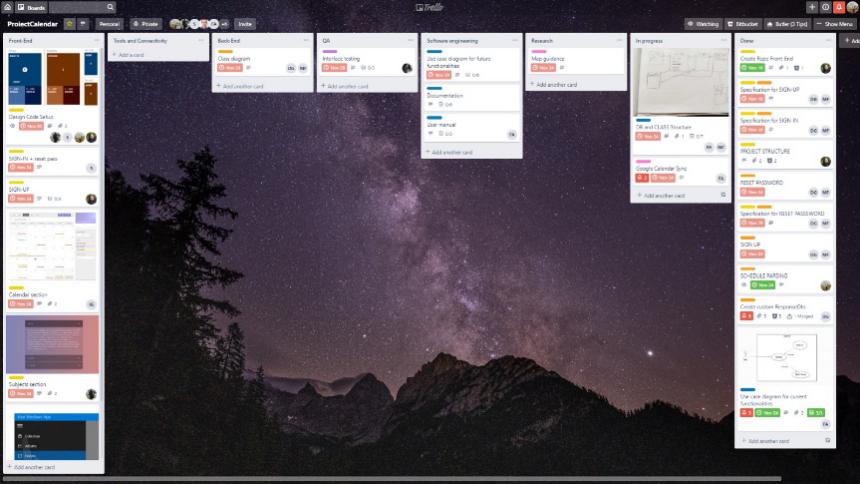
## The Development Process

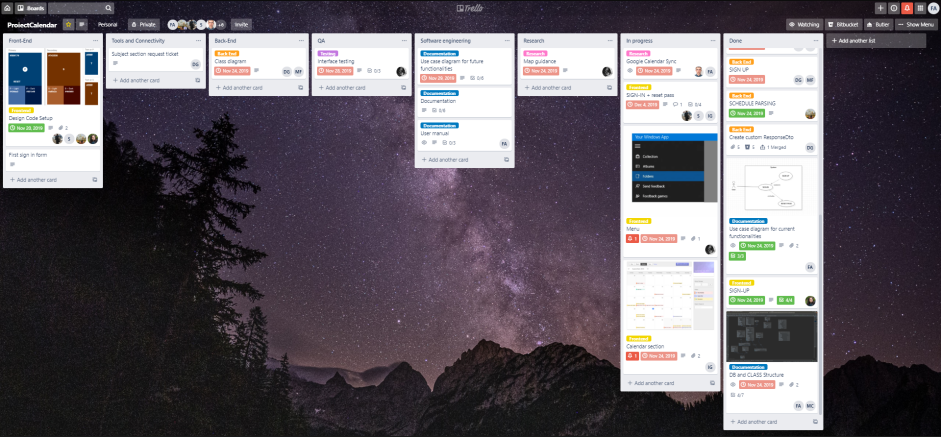
Trello: We used this web application to sync with one another and keep track of all the task that we have to do. All the tasks have been classified in several groups :

-Front End -Back End -Tools & Connectivity -Testing

-Documentation -Software engineering -Research

We also made classes for tasks in progress and done. The main reason for which we decided to use this is because we needed to organize better, one man couldn’t constantly keep track of everyone’s progress. We also logged on every task as description, references and todo list everything that has been decided during the biweekly stand-ups.





For the hosting of our software product with version control we used a git based platform : BitBucket. Two repositories were created , one for the front-end and one for the back-end.



….

## Planning

Although there were a few impulses to jump right into the code, the project was far more complicated than any of us could have expected. Even though we quickly decided upon the technologies , the first couple of weeks was mostly dedicated to the engineering of the layers and architecture , the design of the interface and most importantly the structure of the data base. Data is the foundation of our app and we had to take it as seriously as possible. We want through several stages of software engineering, but in the end we had our almost complete structure and plan.

The next big thing to be done was the application frame, which had to be done by one of us. Meanwhile we created and choose the tasks each of us would have to do. Slowly, but steady the app started working and we ended up with a decent final prototype.

## Top 3 Challenges

* + 1. It was way harder for everyone to keep in touch with everyone else than we expected

Solution:

-we tried biweekly stand-ups to meet face to face

-we resumed to group voice calls

* + 1. Finding the specifications needed for an api call in the front or in the back end

Solution:

-created a drive with api function documentation to keep track of them all without everyone having to clone both projects

-creating pairs of two students, one working on the front-end and one working on the back-end so that communication would resume to only one communication between only two people during the development of a functionality

* + 1. Being able to motivate everyone to work consciously and constantly

Solution:

-creating a task board in trello

-constantly remind the team of tasks and deadline

## Lessons learned

It has truly been an interesting experience, maybe it did not come at the perfect time, but it certainly did come with an impact on some of us to say at least. It’s hard to get along with everyone, it’s hard to keep everyone active and implicated, it’s also hard to develop a project of this size ( size we haven’t even come close to in any of our personal projects or home-works ),but from now on it will only get easier as we gain more experience. We learned that during a project the most crucial thing is neither the code, nor how you write it , but who you are writing it with.

# Bibliography

<https://jwt.io/introduction/>

<https://flywaydb.org/>

<https://stackoverflow.com/>