UNIVERSITÀ DEGLI STUDI FIRENZE

PhD Angular

An Angular 2+ application for the management of PhD programs

Software Architectures and Methodologies

Alessandro Sestini

Matricola: 6226094

alessandro.sestini@stud.unifi.it

Idea

UNIVERSITÀ DEGLI STUDI FIRENZE

- The Idea was to make an **User Interface Angular 2+** for the management of PhD Programs

- This interface is based on a back-end already created - but not complete - that will offer **REST services**

- The development of the application was made following 2 basic steps: **Requirements Analysis** and **Implementation**

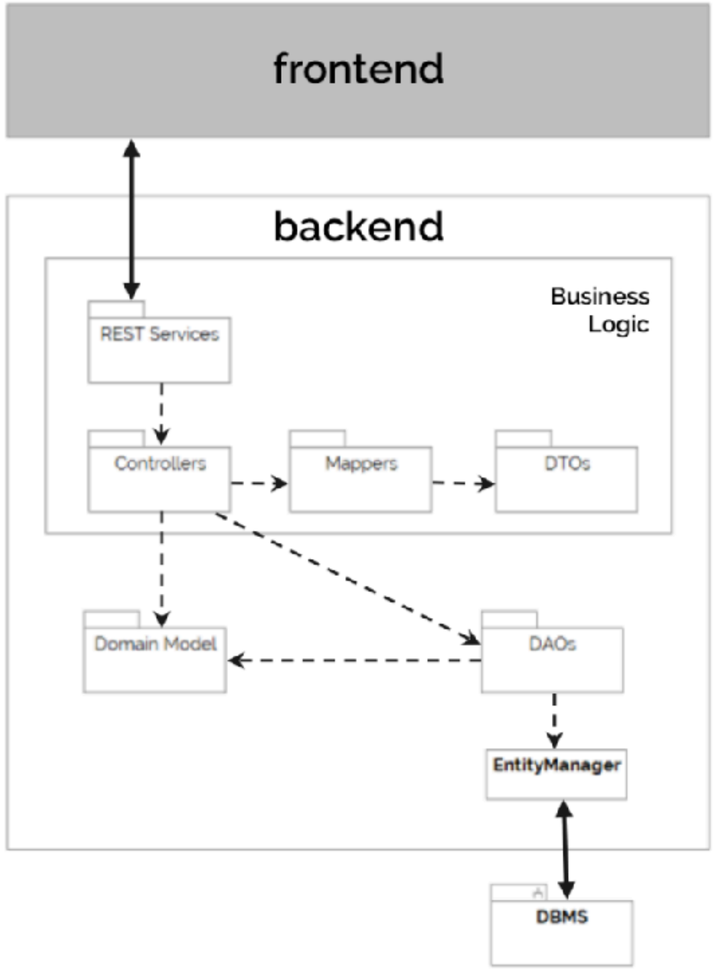
RESTful architectures

UNIVERSITÀ DEGLI STUDI FIRENZE

- **REST** is an architectural style to be generally used for creating web services
- In this type of software, the back-end exposes some **services** through REST APIs
- All of these services are managed by **endpoints**, and each service is associated with a **hierarchic URI**, and it is based on client-server paradigm
- To access these services, the client must use **HTTP requests**

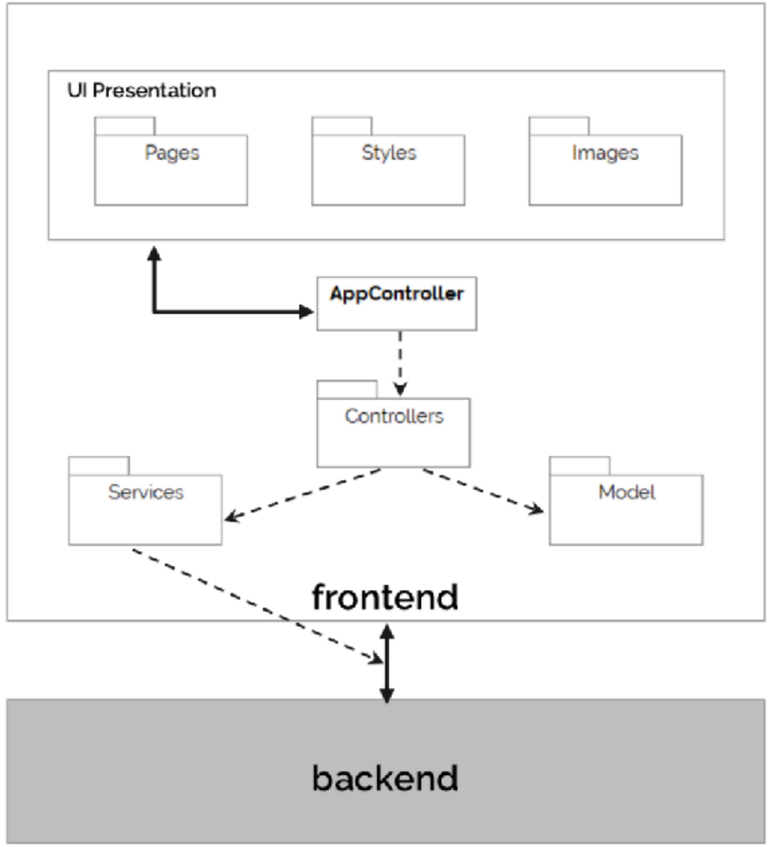
Back-end

- **DAO**: provide an abstract interface for the persistence level
- **DTO**: represent a simplified version of objects referring to the domain model in order to communicate to front-end
- **Mapper**: ‘convert’ the object model to structure of DTOs



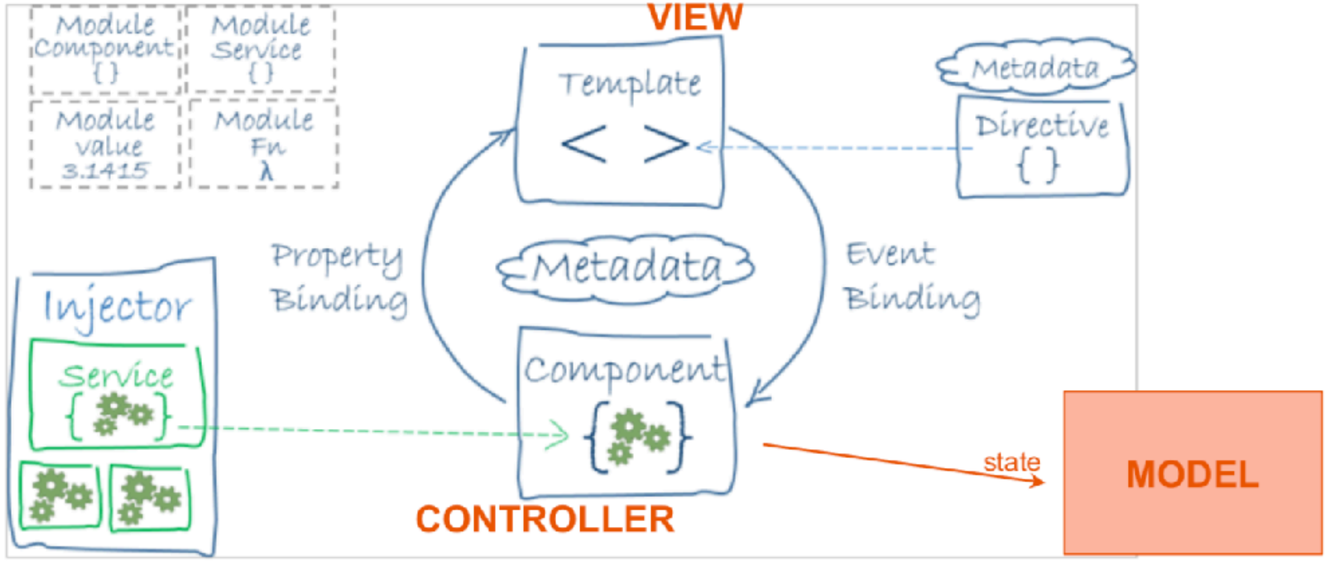
Front-end

- The front-end is the software part that **uses the services** offered by REST API
- There is a strong **decoupling** between front-end and back-end
- Therefore it is possible to use **different front-ends** for the same REST services
- The front-end also defines the **presentation layer** of the application



Angular 2+

- **Angular 2+** is a framework for the development of web application based on RESTFul architectures



Requirements Analysis

UNIVERSITÀ DEGLI STUDI FIRENZE

- The analysis started with the definition of some **scenarios**:

- **Scenario 1**: “a staff member publishes the data of new PhDs, students, faculties and can associate one or more advisor to a student”

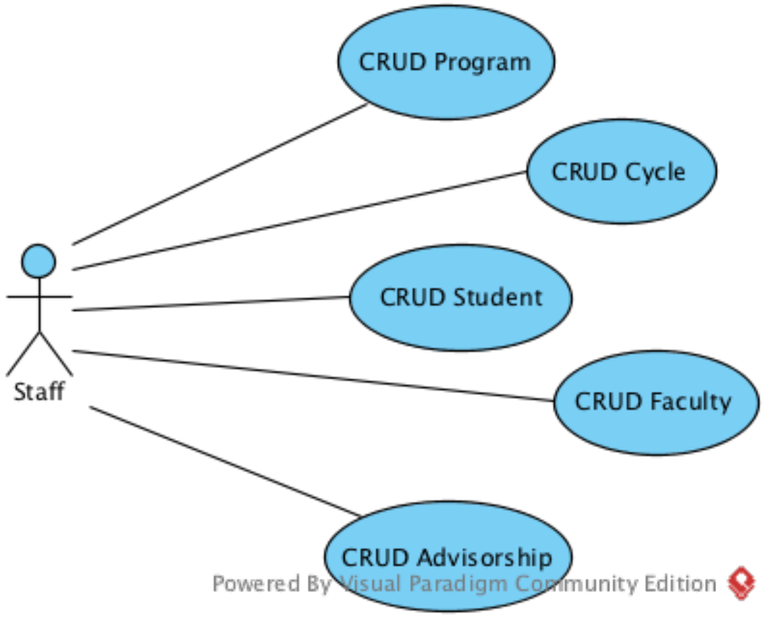
- **Scenario 2**: “a coordinator accesses the system and adds a course to a PhD; a student can register to the course; a qualified teacher can interact with students and with the course; at the end of the course, the teacher evaluates the students enrolled”

Requirements Analysis - Scenarios

- **Scenario 3**: “a student can record some events on his page and can propose some CFUs for these events; the student advisor can support these request; the coordinator approves the requests. A student can also upload a year-end report; the student advisor can modify or approve the report; the coordinator approves the CFUs proposed with the report”

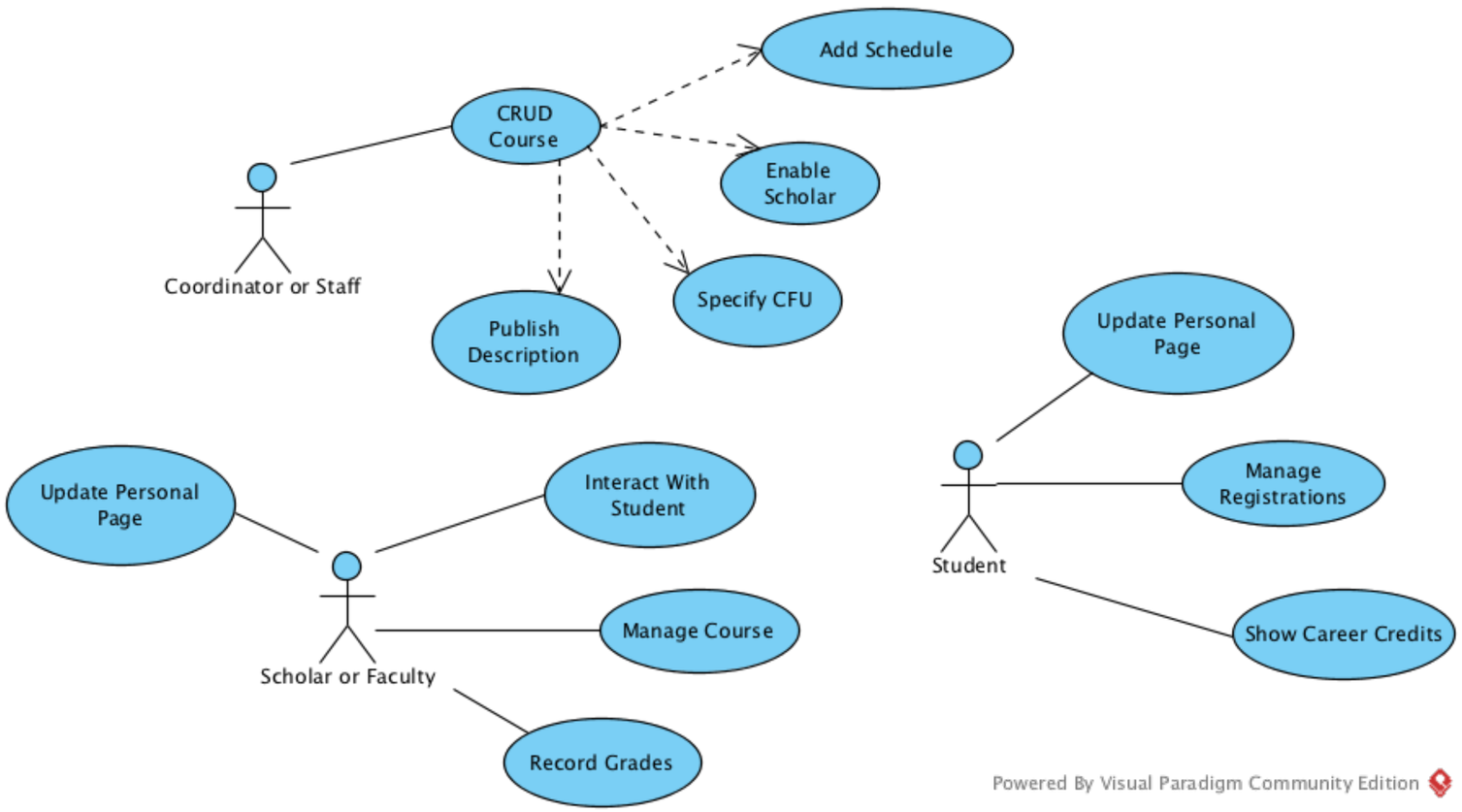
Requirements Analysis - Use Cases

- From these scenarios, we can define the **requirements and use cases** of the application



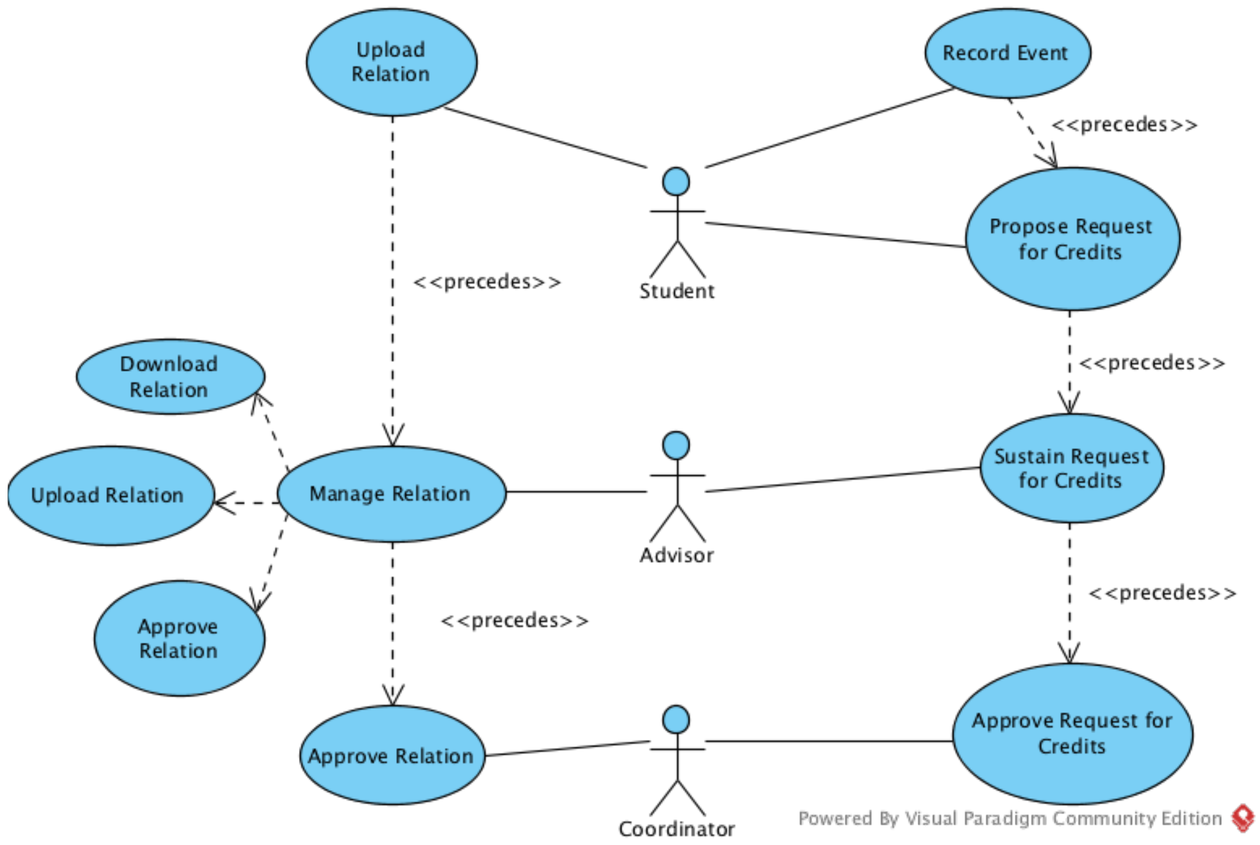
Use case of scenario 1

Requirements Analysis - Use Cases



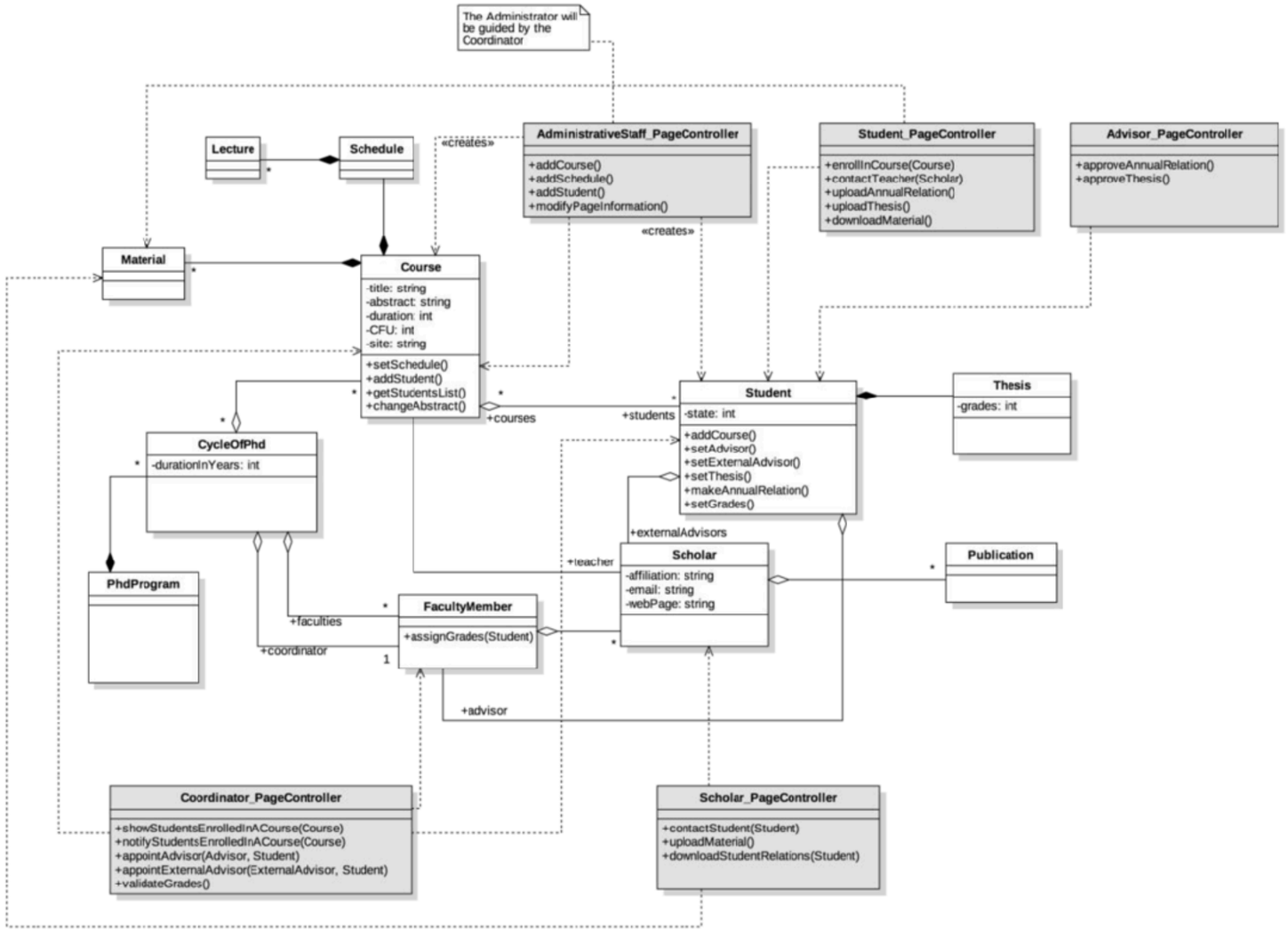
Use cases of scenario 2

Requirements Analysis - Use Cases



Use cases of scenario 3

Requirements Analysis - Domain Model

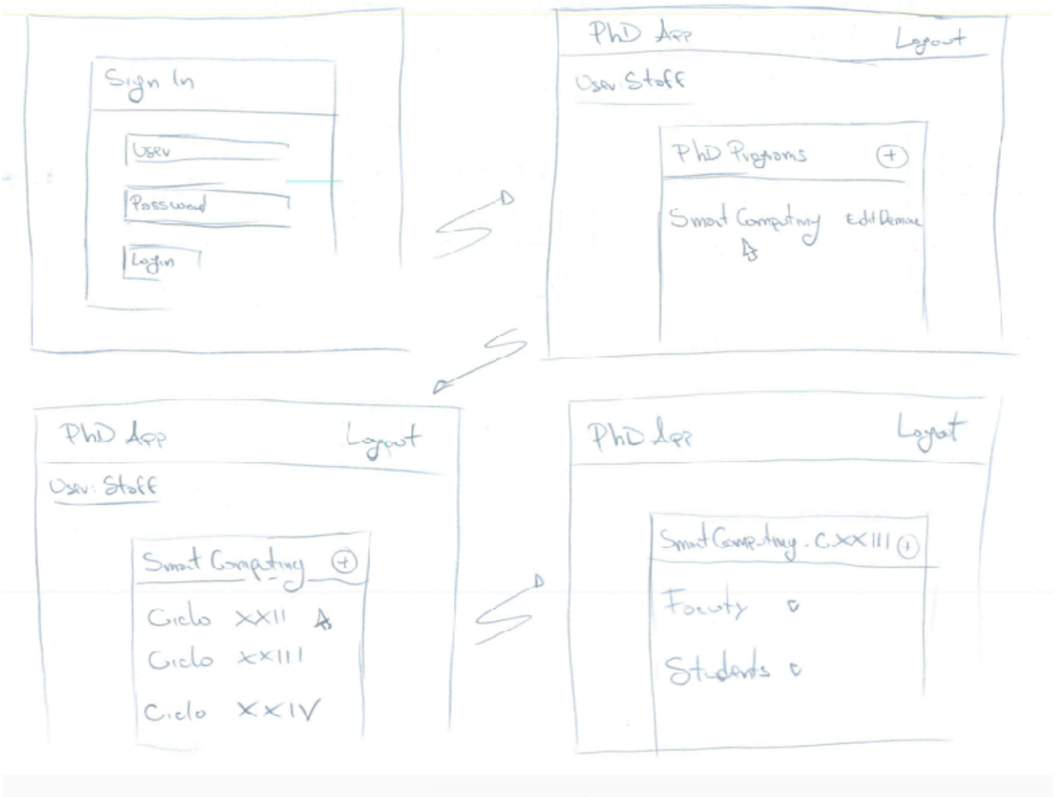


Implementation

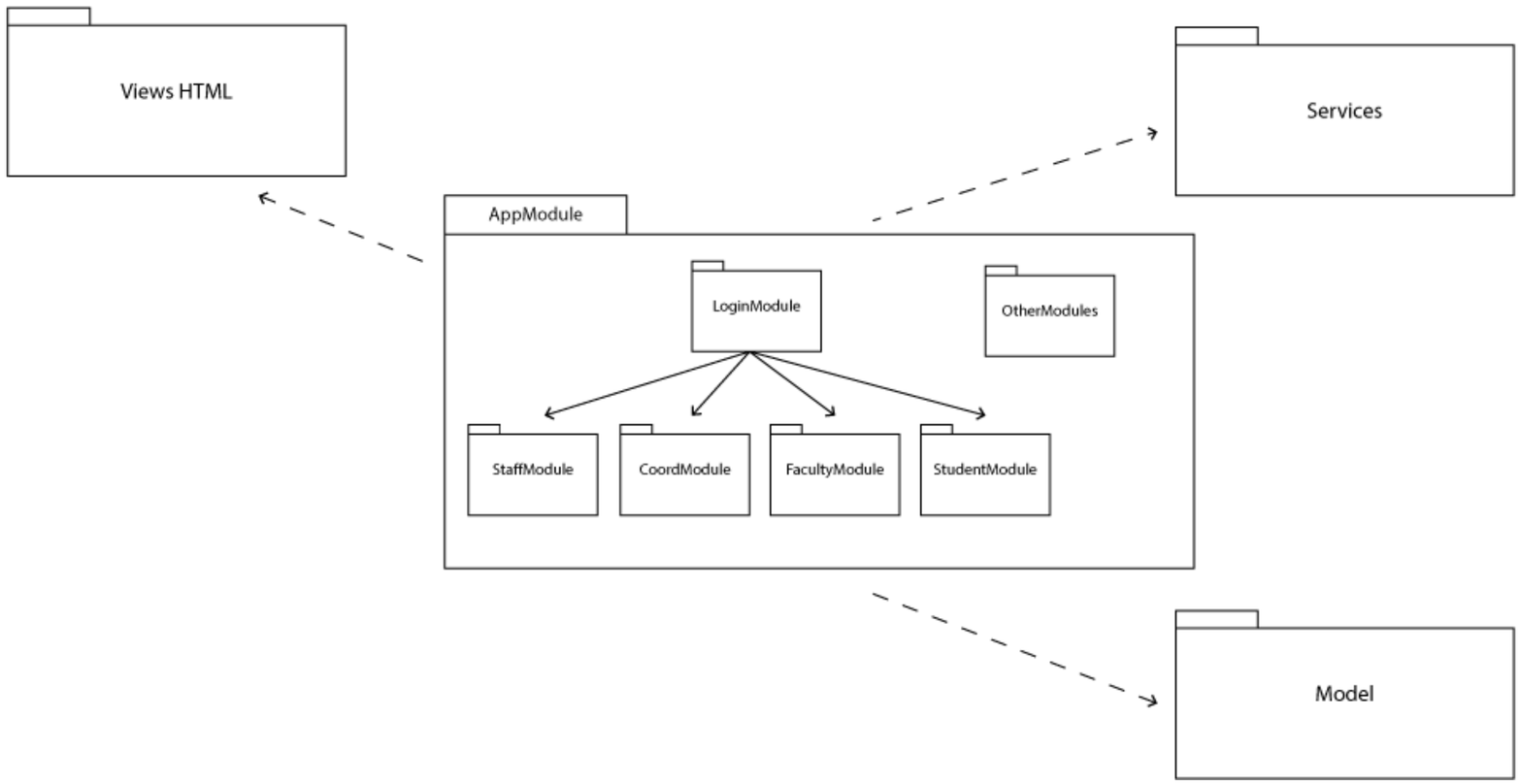
- The implementation of the interface started with some **mock-ups** that define the general layout

- The **material design** by Google was used for the graphic elements of the application

- The framework **JsonServer** was used for faking a REST back-end



Implementation - Application Structure



Implementation - Model and DTO

UNIVERSITÀ DEGLI STUDI FIRENZE

- The Angular classes reflect the domain logic, but we need to clarify some aspects about **relationships**

- In this work it has been used a ***'component driven'*** approach: the entities have only the attributes needed for the component that use them

- In this way, the informations exchanged between back-end and front-end are **minimized** and the whole presentation is more **coherent** with the state of the database

- The definition of the model is fundamental for the creation of **DTOs** that needs to reflect the structure of these classes

Implementation - Model and DTO

UNIVERSITÀ DEGLI STUDI FIRENZE

```
export class PhdProgram {
  id: number;
  name: string;
}
```

```
export class CycleOfPhd {
  id: number;
  num: string;
  numNumber: number;
  durationInYears: number;
  phdProgramId: number;
}
```

```
export class Student {
  id: number;
  facultyId = [];
  firstname: string;
  lastname: string;
  webpage: string;
  email: string;
  faculties: Faculty[];
  externalAdvisor: Scholar = new Scholar();
  role = 'student';
  username: string;
  password: string;
  cycleOfPhdId: number;
  phdProgramId: number;
  events: Event[];
  reports: Report[];
}
```

Implementation - Services

- An Angular service is a class that implements functionalities **shared** between different entities of the application

- The services are usually the only objects that **communicate with REST** services exposed by the back-end (they can be seen as DAOs)

- The **addresses** used by services reflect the domain logic relationships (with plural names)

Implementation - Services

UNIVERSITÀ DEGLI STUDI FIRENZE

```
export class PhdProgramService {

  constructor(
    private http: HttpClient,
    private general: GeneralService
  ) { }

  // Get all PhDs
  getPhdPrograms() {
    return this.http.get<PhdProgram[]>(`${ this.general.uri }/phdPrograms`)
      .pipe(catchError(error => of(this.general.setError(true, error))));
  }

  // Add a PhD
  addPhd(phd) {
    return this.http.post(`${ this.general.uri }/phdPrograms`, phd)
      .pipe(catchError(error => of(this.general.setError(true, error))));
  }

  // Get a PhD
  getPhd(id) {
    return this.http.get<PhdProgram>(`${ this.general.uri }/phdPrograms/${ id }`)
      .pipe(catchError(error => of(this.general.setError(true, error))));
  }

  // Delete a PhD
  deletePhd(id) {
    return this.http.delete(`${ this.general.uri }/phdPrograms/${ id }`)
      .pipe(catchError(error => of(this.general.setError(true, error))));
  }

  // Put a PhD
  putPhd(id, changes) {
    return this.http.put(`${ this.general.uri }/phdPrograms/${ id }`, changes)
      .pipe(catchError(error => of(this.general.setError(true, error))));
  }
}
```

Implementation - JsonServer limitations

UNIVERSITÀ DEGLI STUDI FIRENZE

- JsonServer was used to fake REST API, but has some limitations:
- it doesn’t provide nested routes deeper than 1 level
- it doesn’t provide many-to-many relationships
- it has a non-editable cascade delete mechanism
- All of these problems however should not be present in the back-end implementation

Implementation - Login Module

UNIVERSITÀ DEGLI STUDI FIRENZE

- The first module of the interface is the Login Module; the login mechanism is based on the **JsonWebToken** framework: when the user enters his credentials, the system will make a post request to check if the user is saved in the db

- Then the server-side responds to the front-end with the user-data, including the **token and the user role**, that will be recorded on the **local storage** of web browser

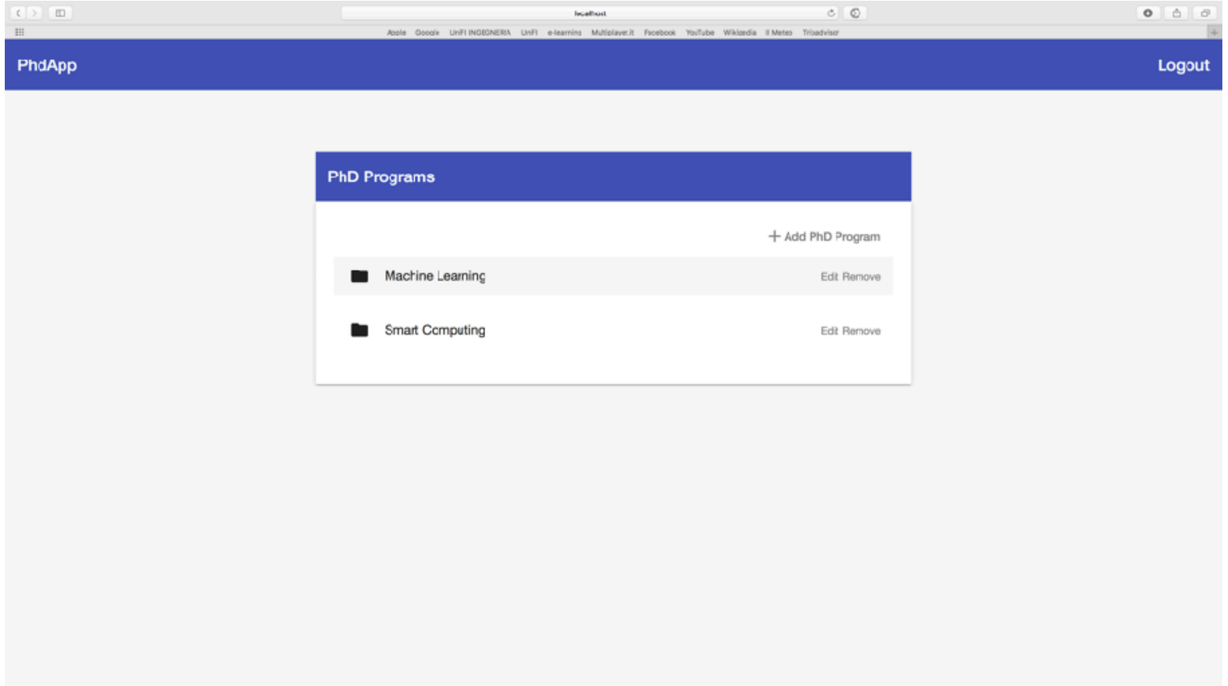
- The front-end will check if the user has the **right role** to use all the components and all of the http requests will be made **adding the user token**

Implementation - Scenarios

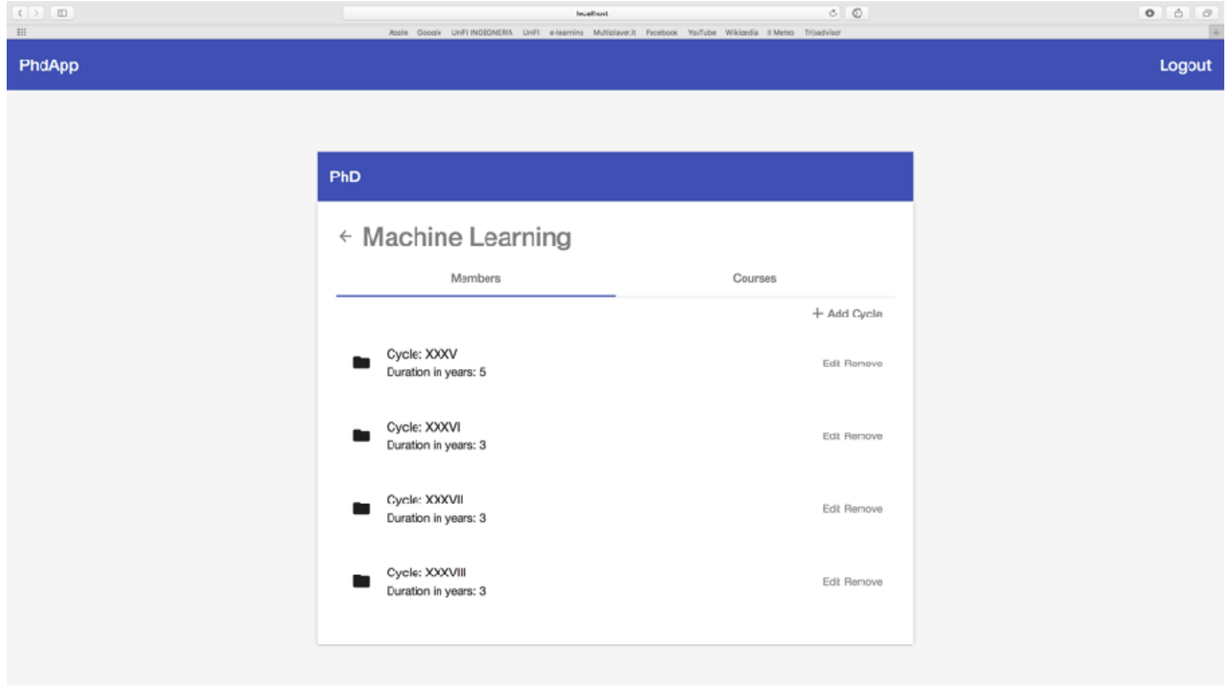
UNIVERSITÀ DEGLI STUDI FIRENZE

LIVE DEMO

Implementation - Scenario 1

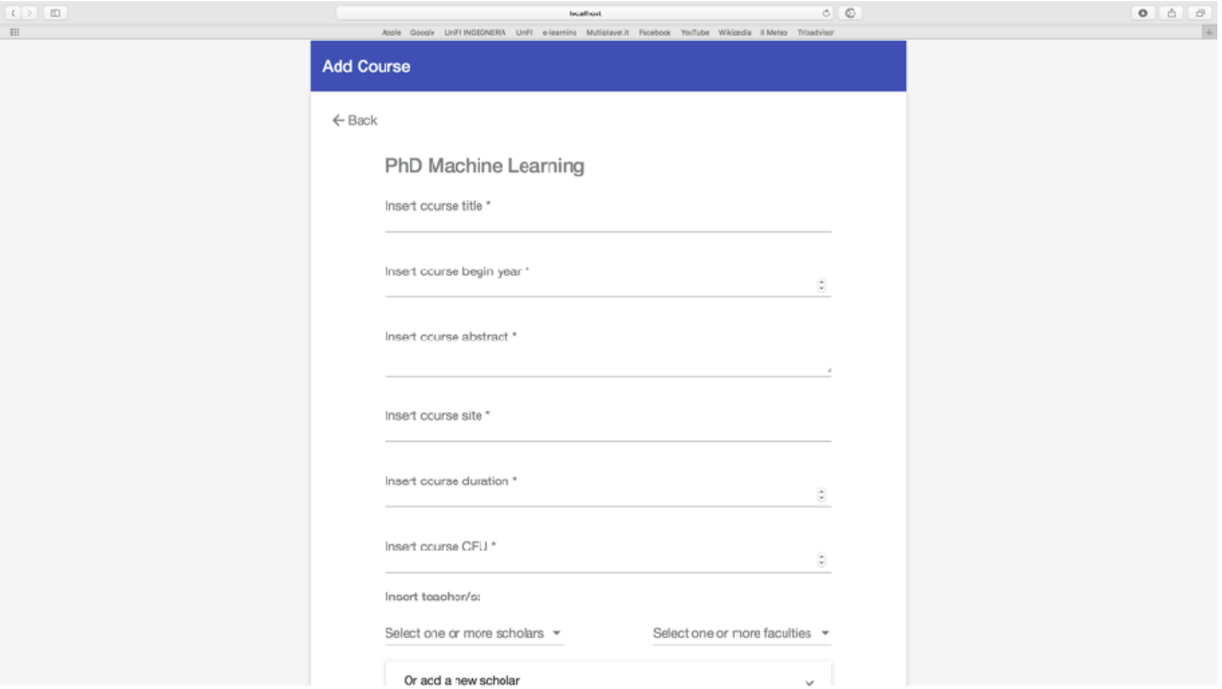


The staff member is able to see all of the PhDs

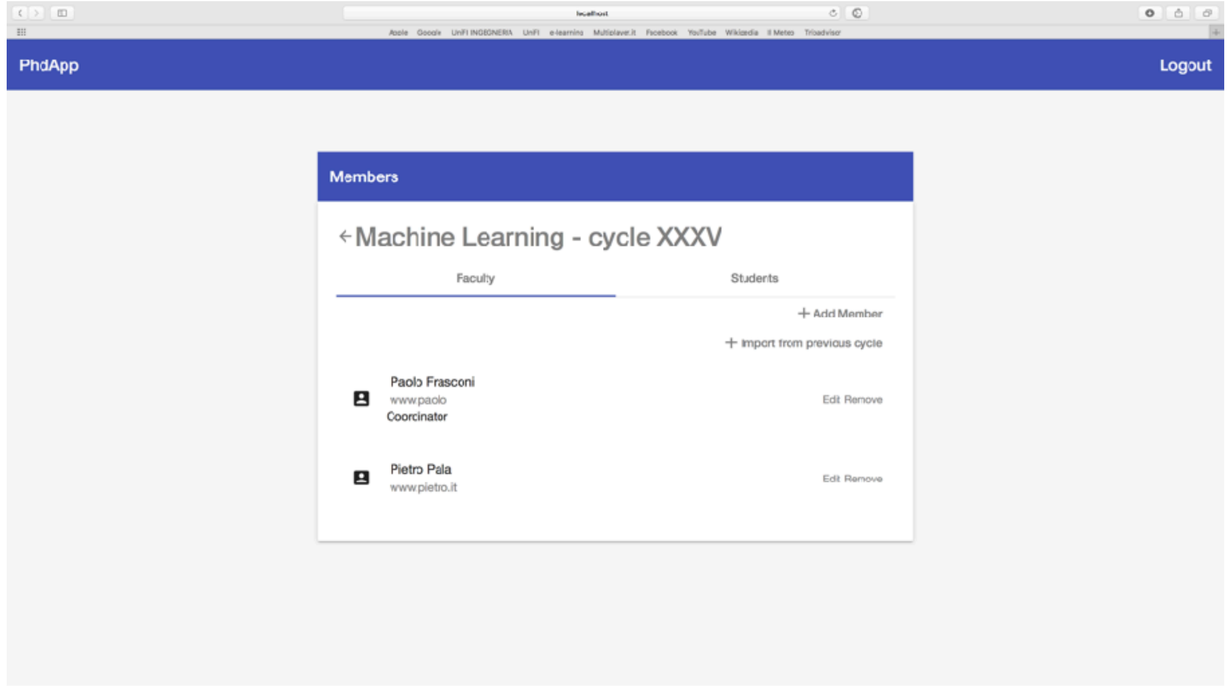


Entering one PhD, the staff member can see all courses and all of its cycles

Implementation - Scenario 1

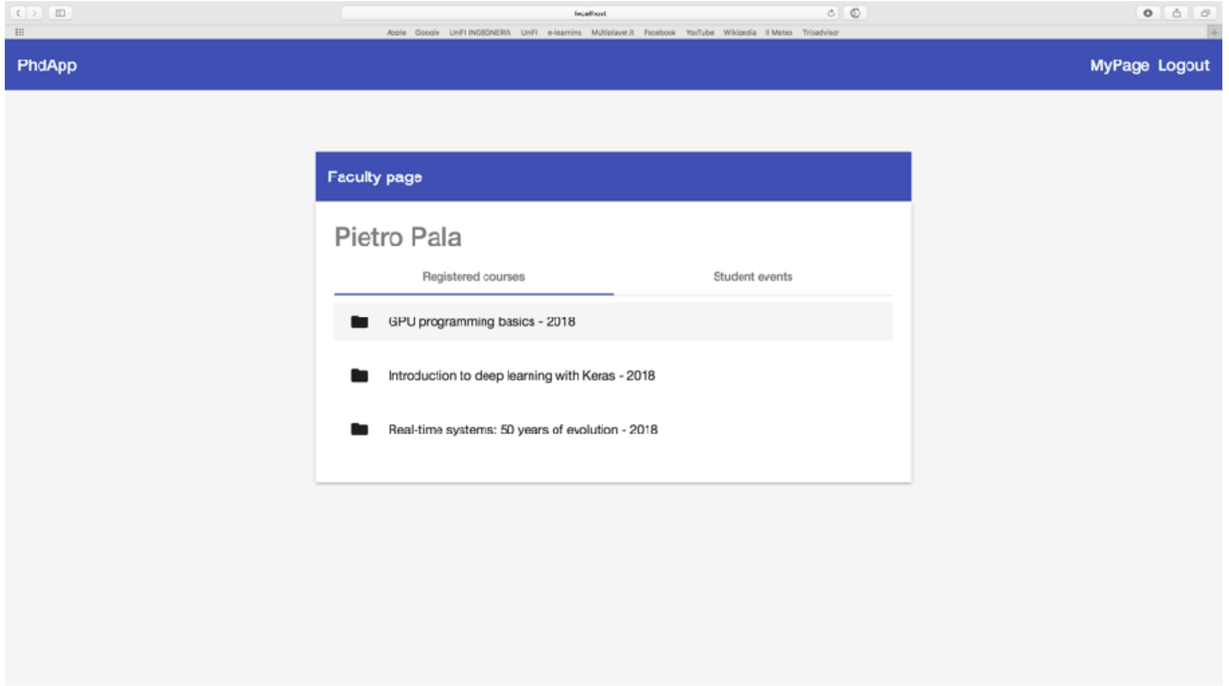


A staff member (or a coordinator) can add a course

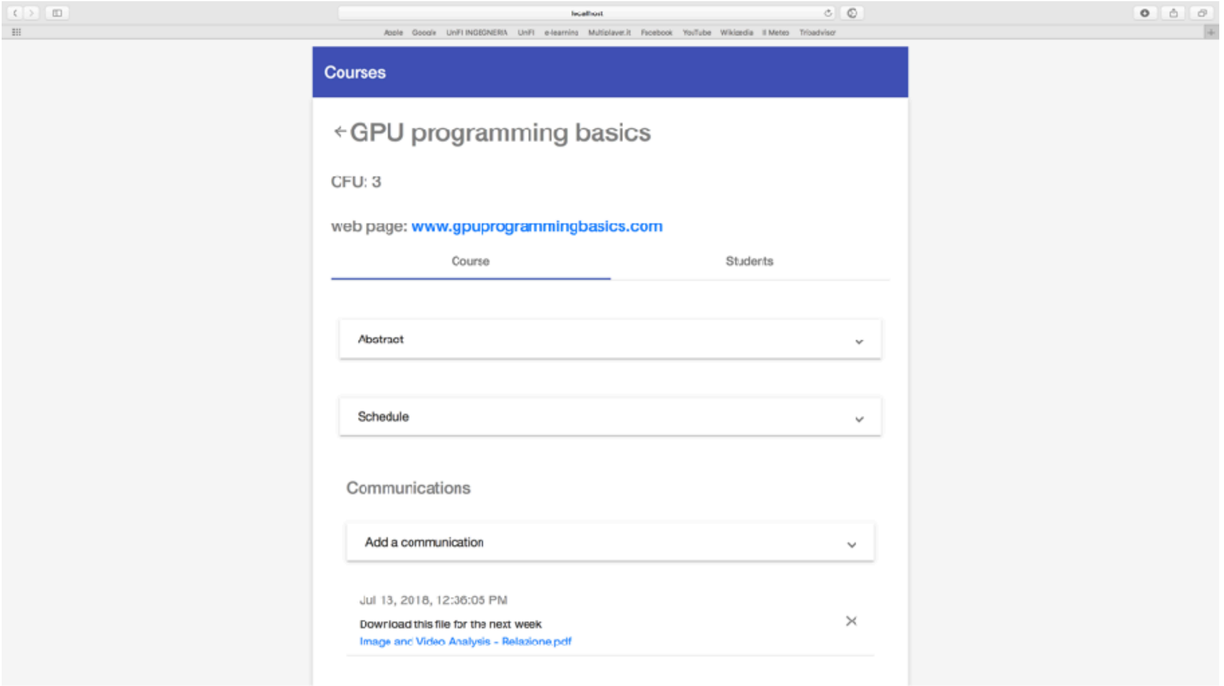


Or he can see or add members of a cycle

Implementation - Scenario 2

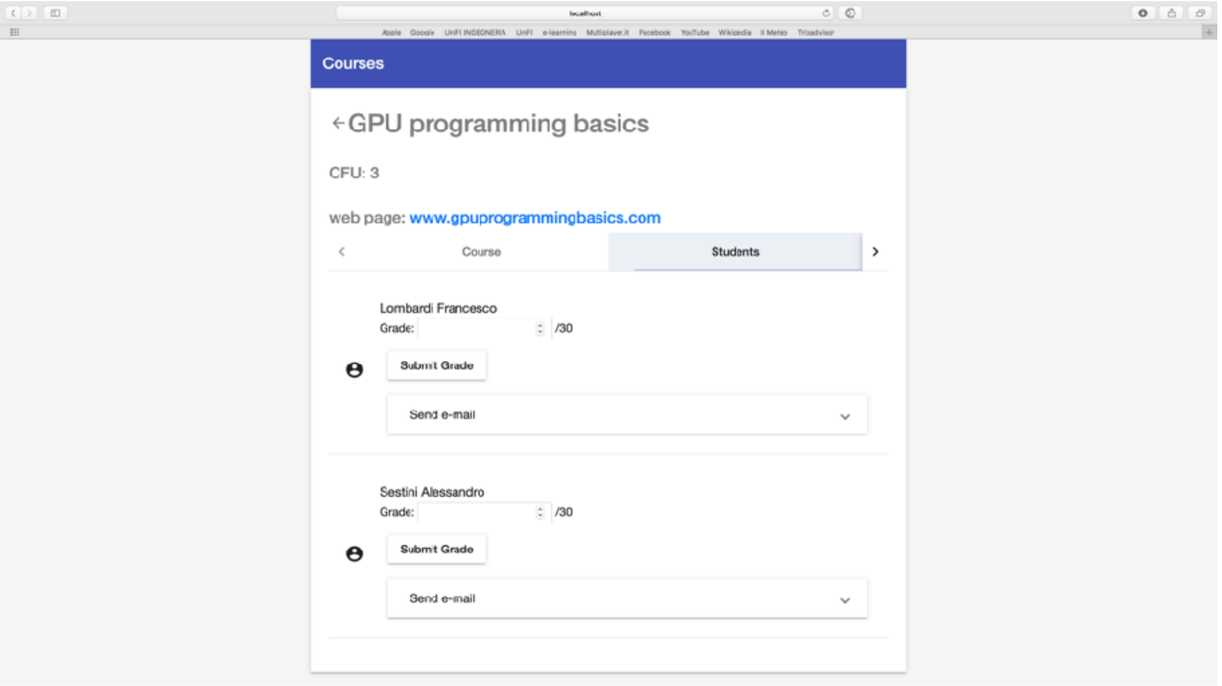


A faculty (or scholar) can see all the courses for which it is enabled

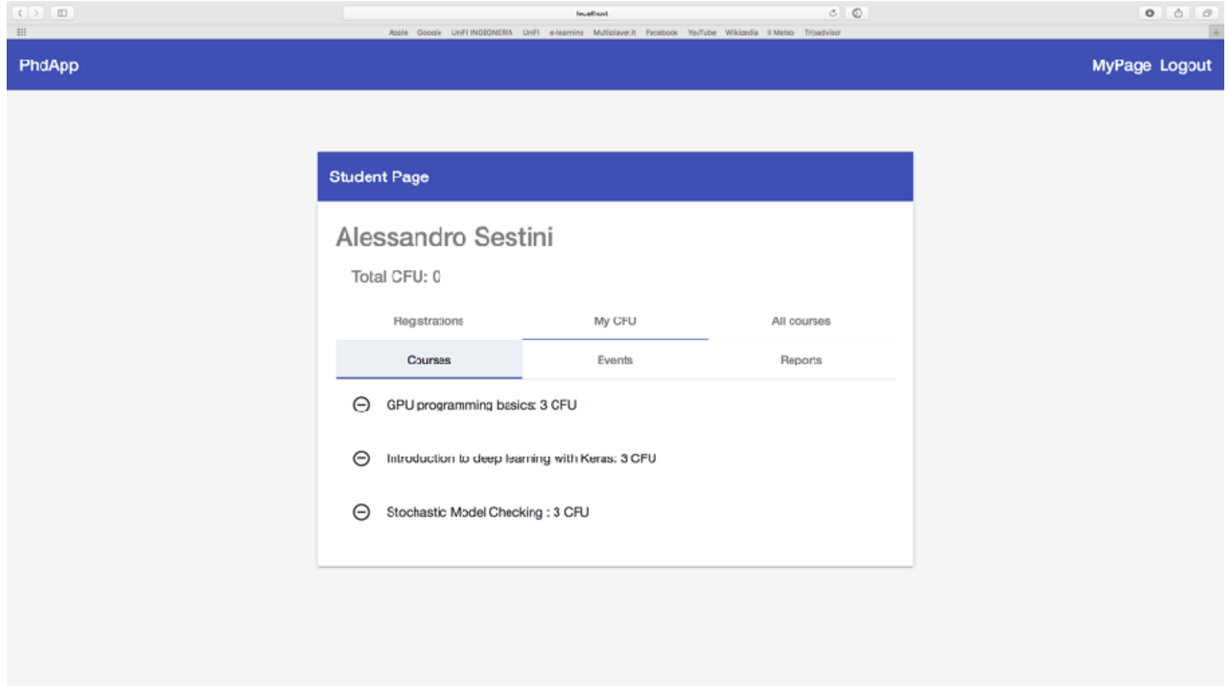


He can interact with the course materials or with the student enrolled in that course

Implementation - Scenario 2

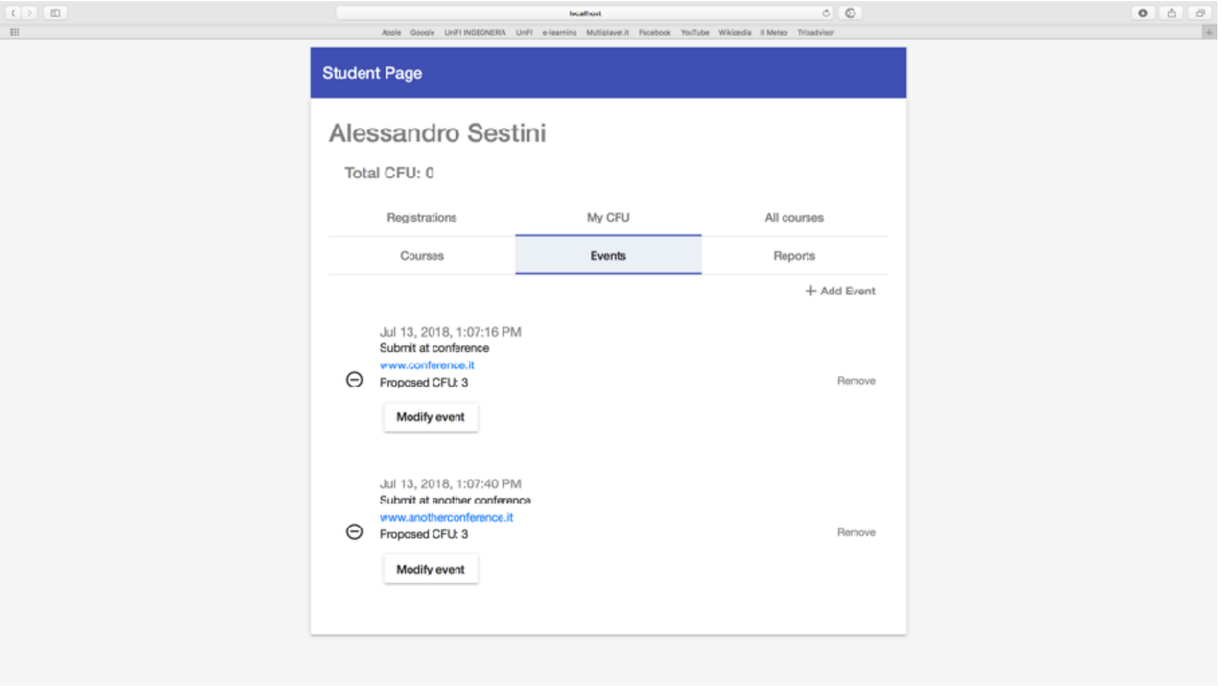


The faculty (or scholar) can register a grade for each student enrolled in that course

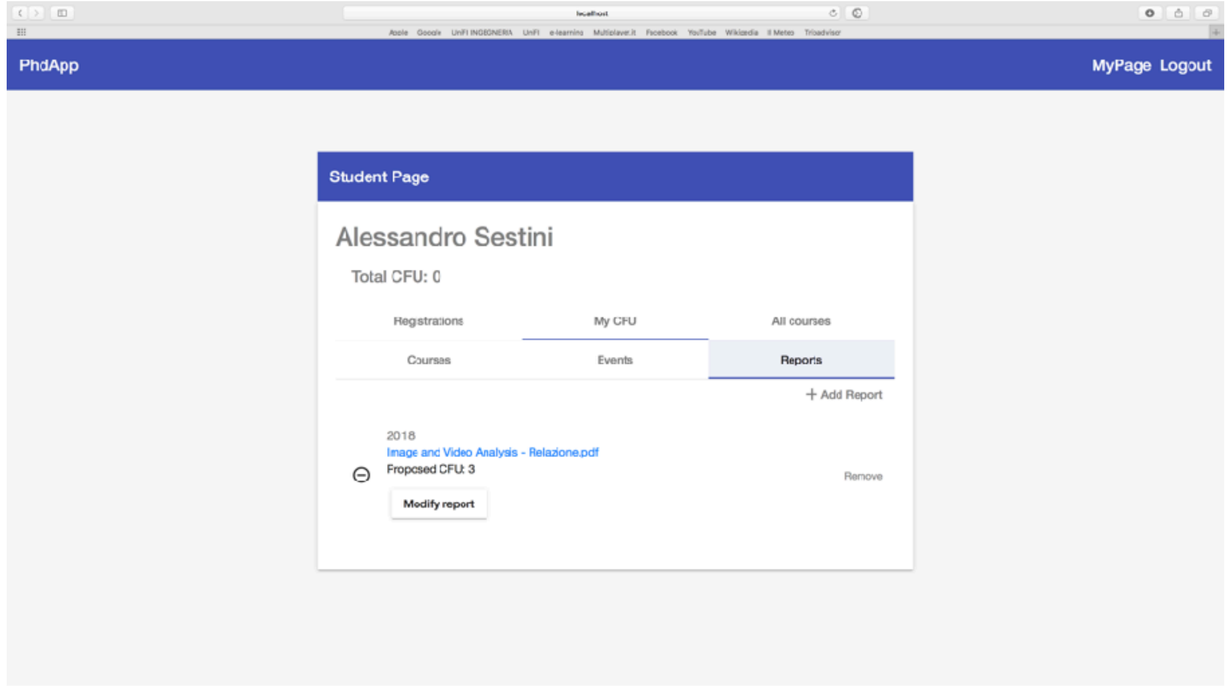


In his page, the student can see all his registrations and all the completed courses

Implementation - Scenario 3

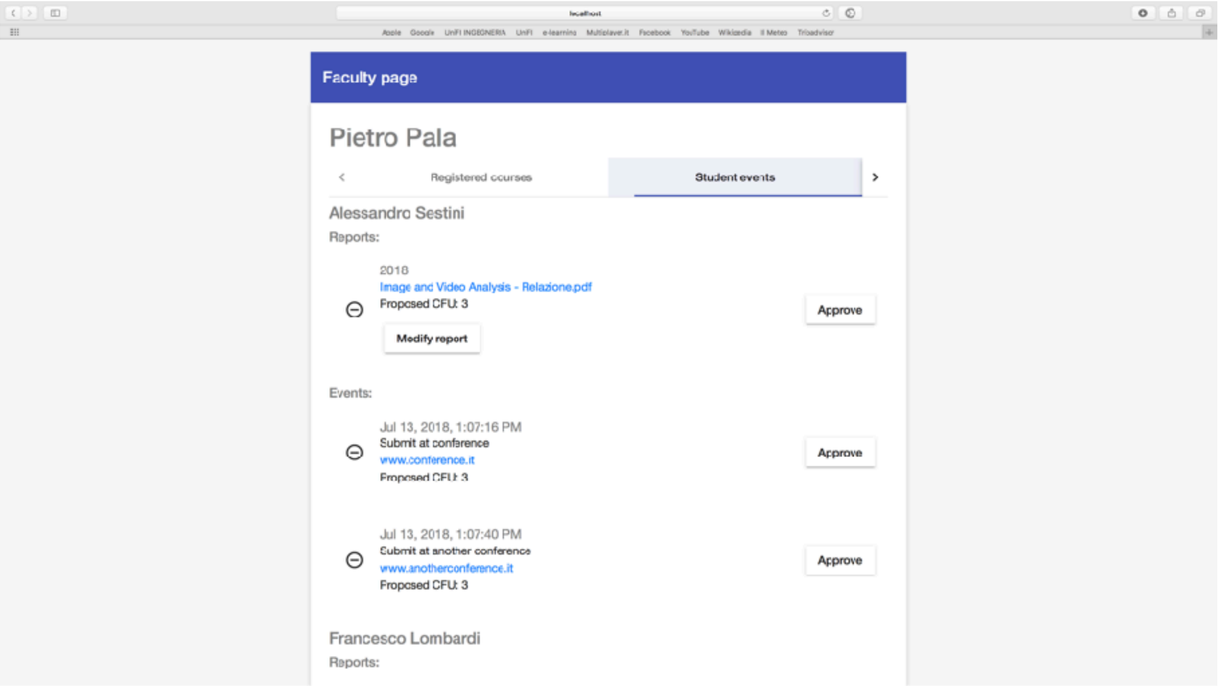


A student can see in his page all the events added previously

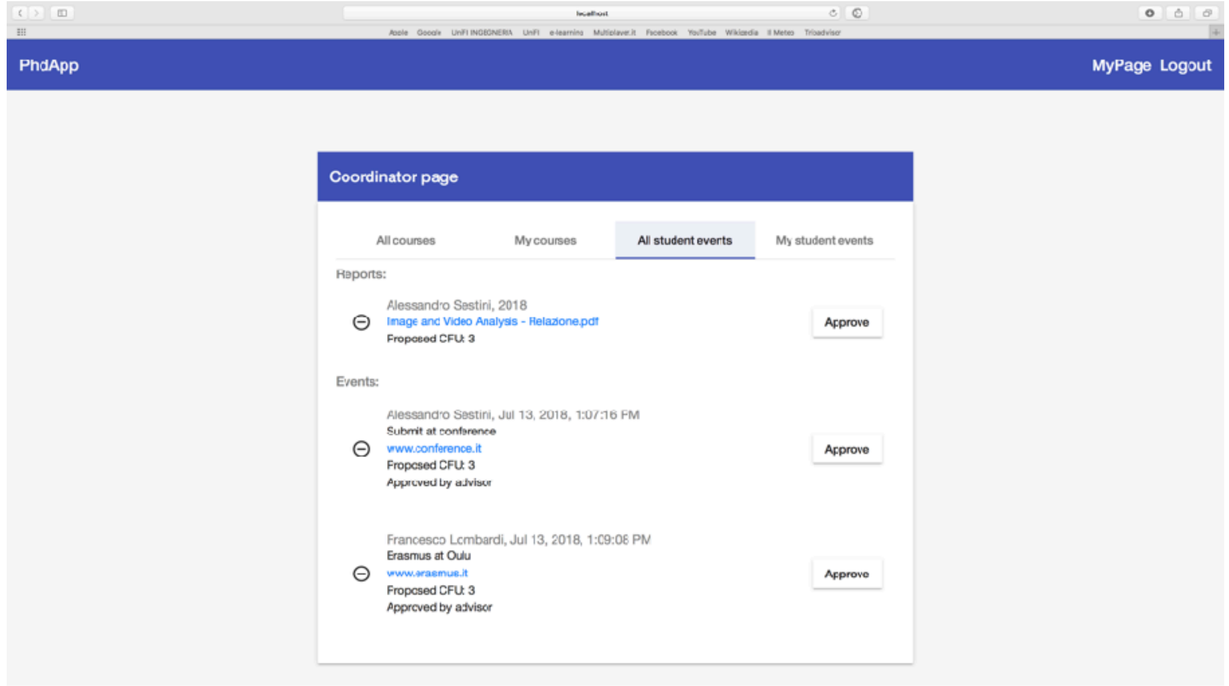


He can do the same for the end-year reports

Implementation - Scenario 3



The student advisor can sustain the events added by a student; the same happens for the reports



Then the coordinator can approve the events or the reports; the student page will be updated with new credits

Conclusion

- In this project we saw the study and the development of a User Interface Angular 2+ of an application for the Management of PhD Programs
- The application was based on a back-end already created but not complete
- The development started with the Requirements Analysis and it continued with the implementations of some scenarios
- The next steps could be: waiting for the completion of the back-end, increasing the usability of the interface and adding more functionalities to the application

UNIVERSITÀ DEGLI STUDI FIRENZE

PhD Angular

An Angular 2+ application for the management of PhD programs

Software Architectures and Methodologies

Alessandro Sestini

Matricola: 6226094

alessandro.sestini@stud.unifi.it