WEB ARCHITECTURES
Assignment 4

Riccardo Gennaro

December 20, 2022

1 Introduction

Problem statement

This project aims at developing a simple web application capable of displaying to a user information about a given student. In particular, there are two use case: the request for student info, that displays anagraphics and choosen courses, and the advisor choice, that displays the anagraphics of a given student and the teachers of the courses he/she is enrolled in.

As requested by the assignment, the business logic must be implemented through EJB and deployed on a WildFly server that lays outside of the Tomcat were the client side must reside.

2 Proposed solution

This assignment is divided in two projects, one deployed on WildFly, one on Tomcat.

Tomcat - Servlets

HomeServlet

It is the welcome file, and it simply presents the `index.jsp` from which it is possible to input (through a form) the student ID for requesting the student info or to access the advisor choice. Using a non-registered student ID will display an error message.

StudentServlet

This servlet makes use of the `StudentManagerBD` to access the data of a given student and displays it through `studentPage.jsp`. The requested data are:

- Student name;

- Student surname;
- Student ID;
- Courses in which the student is enrolled;
- Grades of the above courses, if any.

StudentServlet

This servlet makes use of the `AdvisorChoiceBD` to access the data of a given student and displays it through `advisorChoice.jsp`. The requested data are:

- Student name;
- Student surname;
- Student ID;
- Names of the teachers that theac the courses in which the student is enrolled;

Tomcat - Business Delegates

Both classes `AdvisorChoiceBD` and `StudentManagerBD`, are used to access facade `AdvisorChoiceManagerFacade` and facade `StudentManagerFacade` functions. To do so, the business delegates instantiate these interfaces through the Service Locator.

Tomcat - Service Locator

RemoteServiceInitializer

This class is a singleton used to set the JNDI properties and to operate the lookup in order to make accessible the services that resides in the WildFly server. It also implements a caching of the already looked up services. The caching is implemented through the `cache` hashmap;

WildFly - Facade Beans

As anticipated above, this solution impents two facade. This facades implementations are two beans that expose their methods to execute high-level operations. To do so, To answer the client, this beans converts the entity to a detatched version of the object, a DTO. To build the DTO, class `DTOAssembler` is used.

WildFly - Entity Beans

This beans are not exposed to the client, but are used in order to interact with the entities through the `EntityManager` class. This beans implement the communication with the database. This beans are accessed by the facade implementation through injection operated by the @EJB annotation used in the facade bean.

Database and entities

Following the database ER-schema.

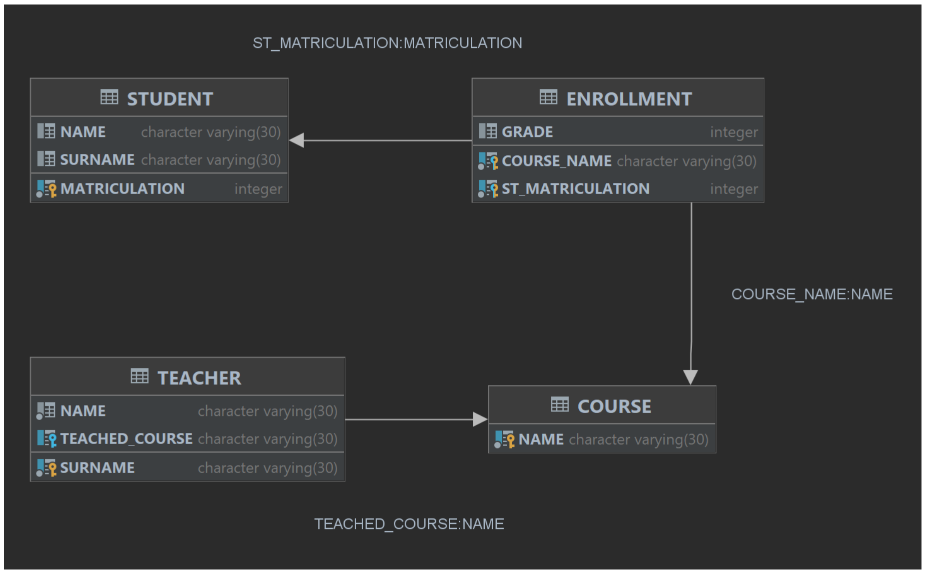

ST_MATRICULATION:MATRICULATION
STUDENT
NAME character varying(30)
SURNAME character varying(30)
MATRICULATION integer
ENROLLMENT
GRADE integer
COURSE_NAME character varying(30)
ST_MATRICULATION integer
COURSE_NAME:NAME
TEACHER
NAME character varying(30)
TEACHED_COURSE character varying(30)
SURNAME character varying(30)
COURSE
NAME character varying(30)
TEACHED_COURSE:NAME

Figure 1: Database ER

There are four tables

- STUDENT, with three columns:
  - NAME
  - SURNAME
  - MATRICULATION (Primary Key)
- TEACHER, with three columns
  - NAME
  - SURNAME (Primary Key)
  - TEACHED_COURSE (Foreign Key, for 1:1 relation with table COURSE)

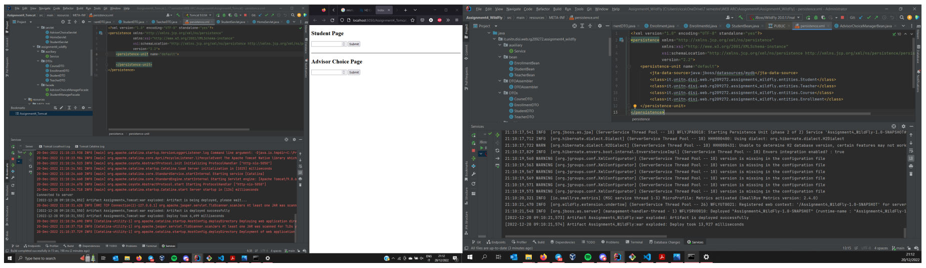
- COURSE, with a single column
  - NAME
- ENROLLMENT, table used to implement the N:M relation between tables STUDENT and COURSE.
  - GRADE
  - COURSE_NAME
  - ST

  ENROLLMENT primary key is given by pair (COURSE_NAME, ST_MATRICULATION).

To connect the database the file `resources/META-INF/persistence.xml` has been modified to include tag `<jta-data-source> java:jboss/datasources/mydb </jta-data-source>`. The datasource has been configured via WildFly's `standalone.xml`. The edited `datasource` tag of the `standalone.xml` can be found at the base path of this deliverable in file `standaloneSnippet.xml`. The database data have been exported through files `mydb.mv.db` and `mydb.trace.db` and can be found at the base path of this deliverable.

Running application

Welcome Page



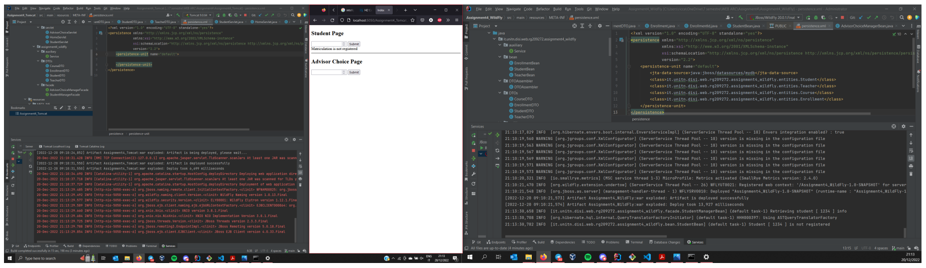
Student Page

Submit

Advisor Choice Page

Submit

Figure 2: Initial page. There are two forms: one redirects to Student Info, the other to the Advisor Choice



Student Page

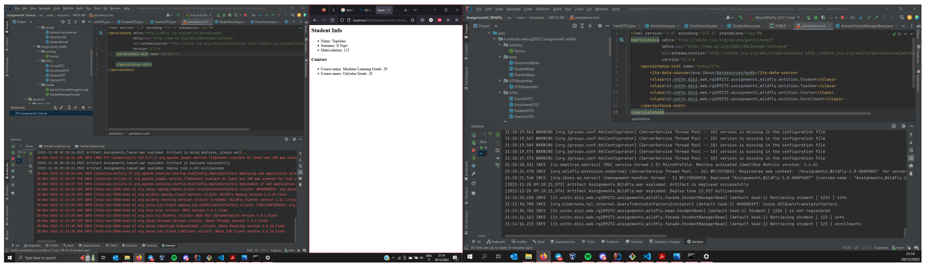
Submit

Matriculation is not registered

Advisor Choice Page

Submit

Figure 3: Initial page. A non-registered student ID was requested



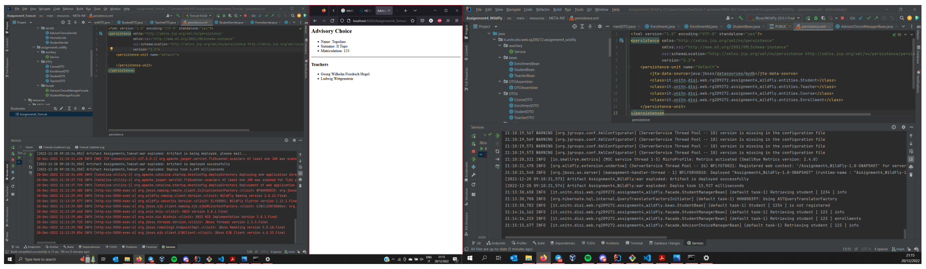
Student Info

- Name: Topolino
- Surname: Il Topo
- Matriculation: 123

Courses

- Course name: Machine Learning Grade: 29
- Course name: Calculus Grade: 28

Figure 4: Student page.



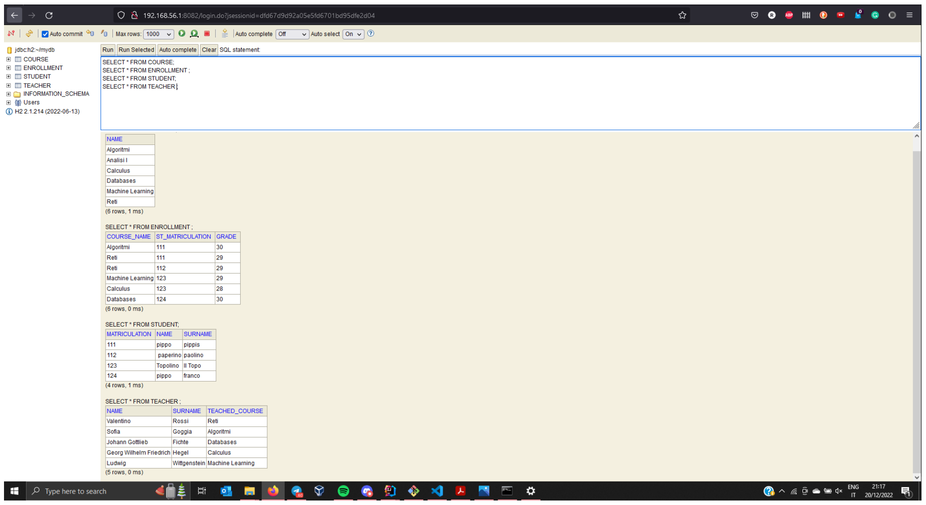
Advisory Choice

- Name: Topolino
- Surname: Il Topo
- Matriculation: 123

Teachers

- Georg Wilhelm Friedrich Hegel
- Ludwig Wittgenstein

Figure 5: Advisor Choice page



```
SELECT * FROM COURSE;
SELECT * FROM ENROLLMENT ;
SELECT * FROM STUDENT;
SELECT * FROM TEACHER ;
```

NAME
Algoritmi
Analisi I
Calculus
Databases
Machine Learning
Reti

(6 rows, 1 ms)

SELECT * FROM ENROLLMENT ;

COURSE_NAME	ST_MATRICULATION	GRADE
Algoritmi	111	30
Reti	111	29
Reti	112	29
Machine Learning	123	29
Calculus	123	28
Databases	124	30

(6 rows, 0 ms)

SELECT * FROM STUDENT;

MATRICULATION	NAME	SURNAME
111	pippo	pippis
112	paperino	paolino
123	Topolino	Il Topo
124	pippo	franco

(4 rows, 1 ms)

SELECT * FROM TEACHER ;

NAME	SURNAME	TEACHED_COURSE
Valentino	Rossi	Reti
Sofia	Goggia	Algoritmi
Johann Gottlieb	Fichte	Databases
Georg Wilhelm Friedrich	Hegel	Calculus
Ludwig	Wittgenstein	Machine Learning

(5 rows, 0 ms)

Figure 6: Database Tables

Comments and notes

The project presents duplicated code. In particular, packages `it.unitn.disi.web.rg209272.assignment4_wildfly.auxiliary | DTOs | facade` are duplicated both in Tomcat and WildFly sides. A solution could be including this packages through maven dependencies. I tried doing so, but WildFly threw a bunch of errors about the name of the service and I didn't have time to investigate.

Also, I must say that one of the biggest challenges was the configuration of WildFly and the understanding of the requested patterns.