1222·2022 800 ANNI

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Web Applications A.Y. 2022-2023
Homework 1 – Server-side Design and Development

Master Degree in Computer Engineering
Master Degree in Cybersecurity
Master Degree in ICT for Internet and Multimedia

Deadline: 28 April, 2023

Group Acronym	ACME	
Last Name	**First Name**	**Badge Number**
Ornella	Irrera	123456

1 Objectives

The overall goal of this project is to develop and describe CRANE a web application for the Department of Information Engineering of the University of Padova. CRANE is a web application whose main goal is to ease and speed up the management of groups students who collaboratively work on the homeworks delivered for a course during the semester. This project aims at providing professors with a platform where they can keep under control students' work, and publish homeworks grades; on the other hand, students are provided with a useful interface they can rely on to join groups, and evaluate teammates work.

2 Main Functionalities

CRANE has two main users: students and professors.

In order to use CRANE functionalities, both students and professors must register in the web application by providing name, surname, email, badge number, and password. Only the students are also required to provide the attended course and the master they belong to. Students are allowed to create new groups of students for a course; the creator of the group will be able to define the name and the description of the group, and the group members. CRANE main functionality is to allow students to evaluate the work done by their teammates: each student is required to score and comment all his team mates. The application will use these information to re-calibrate the final score of each student according not only to the scores assigned for homeworks and exam, but also to the scores assigned by all his/her teammates. Finally, students can visualize the score and the comment assigned to the group for each submitted homework. At the end of the semester also the final score will be provided.
Professors are able to create new courses, set the and update the deadlines defined to signup in CRANE, create a group and evaluate team members. Once that the homeworks are submitted professors can publish the homeworks evaluations which will be visible exclusively to the members of each group.
The CRANE admin is allowed to create new accounts for the professors who want to rely on CRANE to manage the groups of students of their courses.

3 Presentation Logic Layer

CRANE is subdivided into two main areas: the professor area and the student area. They are accessible through the starting page where the user, according to her role, can decide whether to log in as a student or a professor. Below we define the pages we are going to develop in CRANE. All the pages have been developed via jsp.

- Starting page: the very first page the user sees. On this page the user can access professor and student areas;
- Registration page: this page allows the user to register in CRANE;
- Homepage: the page seen by the student once she logged in;
- Group creation page: the page where students can create a new group;
- Team members evaluation page: the page where the students can evaluate all the team members of the group;
- Homework evaluation page: the page where the homeworks evaluations and additional information are provided;
- Professor homeworks publishing page: the page where a professor can add a new evaluation for the homeworks submitted by each group.

Each page, except for the login and the starting page include and header with the logged in student information which allows her to change course and to logout. All the pages contain a footer written in HTML containing the logos of the University of Paodva and the Department of Information Engineering.

3.1 Starting Page

The starting page is the first page a user has access once that he opens CRANE on the browser. The first element the user sees is the logo of the web application, and two buttons are placed below the logo. Each button can redirect the user to the student area and to the professor area.

3.2 Registration pages (Interface Mockup)

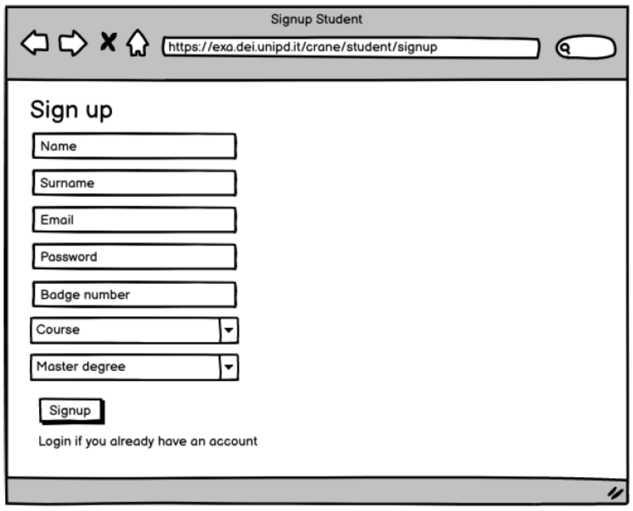
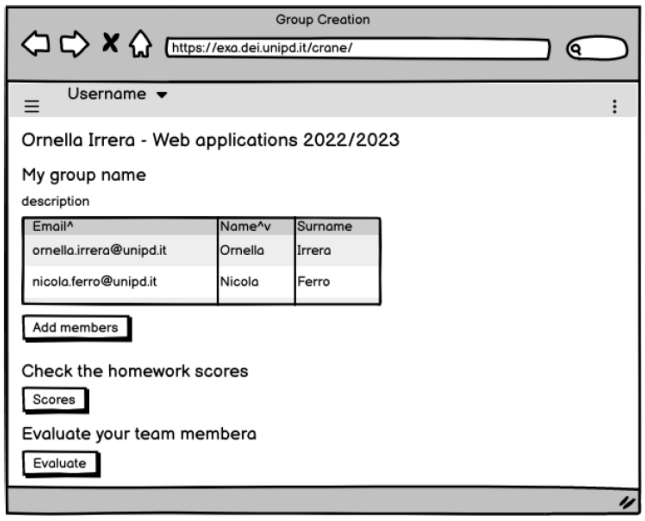


Figure 1: Registration page

On the Registration page, the student can provide her credentials to use to access more personal information about her and her group. The information to provide is the email, the password, the name, the surname, the badge number (which is not mandatory since a student may have not a badge number yet), the course, and a master degree. The lists of courses and master degrees a student can choose from are already provided so to guide the user during the selection. The password to be provided must contain at least 8 characters, at least one number, and at least one upper case letter. Clicking on *Signup* the student adds her credentials to the database and is redirected to the homepage. If the credentials do not comply with those required, or the deadline for the registration is expired, an error message is returned.
The login form contains also a button that allows the user to regenerate a new password for the email he provided during the registration: in this case, a new password will be automatically generated by the system and returned to the user via email. The login form is similar to the registration form (hence we did not treat it as a separate page). In Figure 9 we illustrate the registration page.

3.3 Students homepage (Interface Mockup)

On the homepage, the student is provided with an overview of her group including the group name, the group description, and the list of members belonging to the group. If the logged-in student does not belong to any group, a button allows her to redirect to the group creation page. If the deadline for group creation is not expired yet, the students belonging to a group are able to delete one or more members from the group (if needed). This may happen if the professor increases the maximum number of members of a group, or if a specific student decides to drop the course. Information such as the name and the description of the group is provided but cannot be updated. From the homepage, by clicking on the *Evaluate* button it is possible to be redirected to the students evaluation page where a student can evaluate her team mates. By clicking on the *Score* button instead, it is possible to be



Ornella Irrera - Web applications 2022/2023

My group name

description

Email^	Name^v	Surname
ornella.irrera@unipd.it	Ornella	Irrera
nicola.ferro@unipd.it	Nicola	Ferro

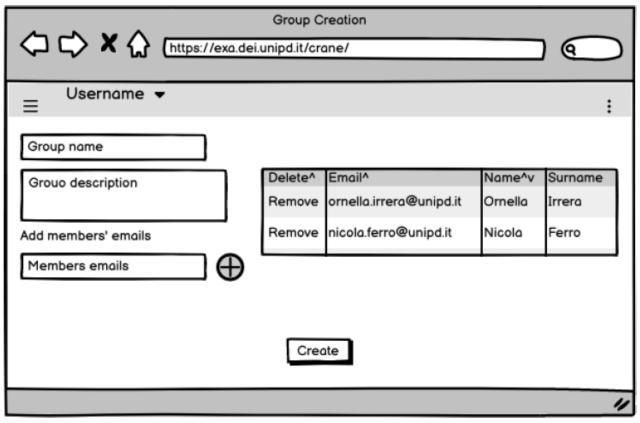
Check the homework scores

Evaluate your team membera

Figure 2: Home page

redirected to the page related to the score assigned by the professor to each submitted homework. Figure 2 illustrates the registration page.

3.4 Group creation page (Interface Mockup)

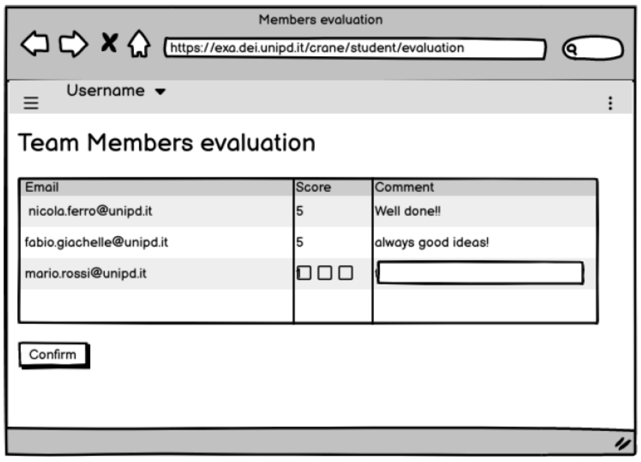


Delete^	Email^	Name^v	Surname
Remove	ornella.irrera@unipd.it	Ornella	Irrera
Remove	nicola.ferro@unipd.it	Nicola	Ferro

Figure 3: Group Creation page

The group creation page can be reached through the homepage of a student. It contains a form where the creator of a group must provide: the name (an acronym is preferred), a description, and a list of members. Each member must be added providing his/her email used in CRANE. If the searched student does not exist or she already belongs to another group, an error message is returned and another member should be selected. If the student is available, she is added to a table which contains all the added members. If a member has been wrongly added, it is possible to removed her by clicking on the *Delete* button next to her email: the student will not be added to the group. Once the group is correctly created, the student is redirected to her homepage where she is provided with the information about the created group, In addition to this, an email is sent to all the members of the selected group in order to warn them that they have been added to a group. Figure 3 illustrates the group creation page.

3.5 Team members evaluation page (Interface Mockup)



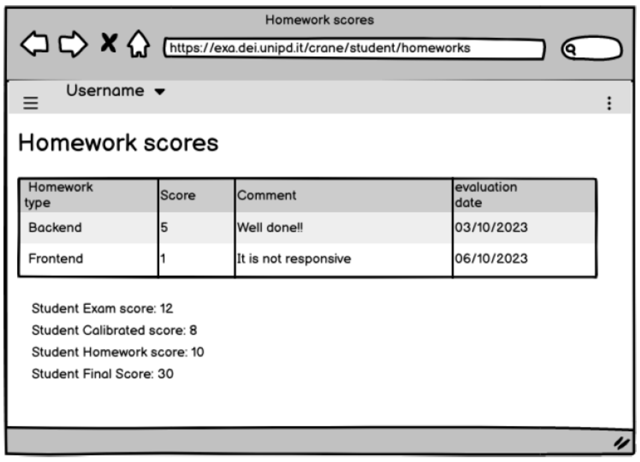
Team Members evaluation

Email	Score	Comment
nicola.ferro@unipd.it	5	Well done!!
fabio.giachelle@unipd.it	5	always good ideas!
mario.rossi@unipd.it	☐ ☐ ☐	

Figure 4: Members evaluation page

The students evaluation page contains a table with as many rows as the members of the group. For each member, the student is required to add a score from 1 to 5 via a radio button, and a comment via a text area that supports long text. The evaluation can be updated until the student evaluation deadline set by the professor. In this case, every time the student wants to update the evaluation, the last submitted evaluation is displayed as soon as she opens the evaluation page. By clicking on *Confirm* the evaluation is submitted and saved in the database. Figure 4 illustrates the students' evaluation page.

3.6 Homeworks evaluation page (Interface Mockup)



Homework scores

Homework type	Score	Comment	evaluation date
Backend	5	Well done!!	03/10/2023
Frontend	1	It is not responsive	06/10/2023

Student Exam score: 12
Student Calibrated score: 8
Student Homework score: 10
Student Final Score: 30

Figure 5: Homeworks evaluation page

On this page, the student is provided with a table that contains the evaluations assigned by the professor for each submitted homework. Each row of the table contains the type of the submitted homework – e.g., homework1

- back-end, the date of the evaluation, the score obtained, and the comment, which can contain very long text. At the end of the semester, below the table, it will be displayed a list of scores which comprise: the score of the homeworks, the score of the exam, extra points, and the total score calibrated based on the evaluation provided by the teammates. Figure 5 illustrates the homeworks' evaluation page.

3.7 Professor homeworks publishing page (Interface Mockup)

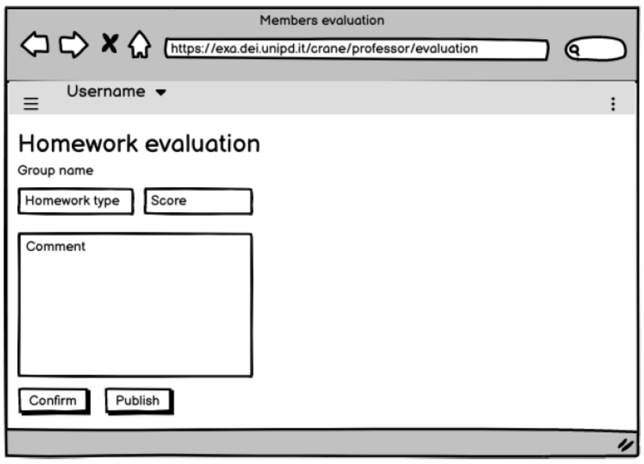


Figure 6: Professor homeworks publishing page

On this page, the professor inserts the evaluations for the homeworks submitted by each group. For each group, the professor must define the homework type she is adding the evaluation of, a number defining the score, and a comment. There are two buttons: *Confirm* button allows the professor to save the evaluation in the database, and *Publish* in order to make the evaluation visible to the group. Evaluation publishing can be performed once the evaluation is confirmed. This distinction allows the professor to insert the evaluation in different days and make them available to all the groups at the same time, without providing the evaluation to the students on different days. Figure 6 illustrates the homeworks' evaluations publishing page.

4 Data Logic Layer

4.1 Entity-Relationship Schema

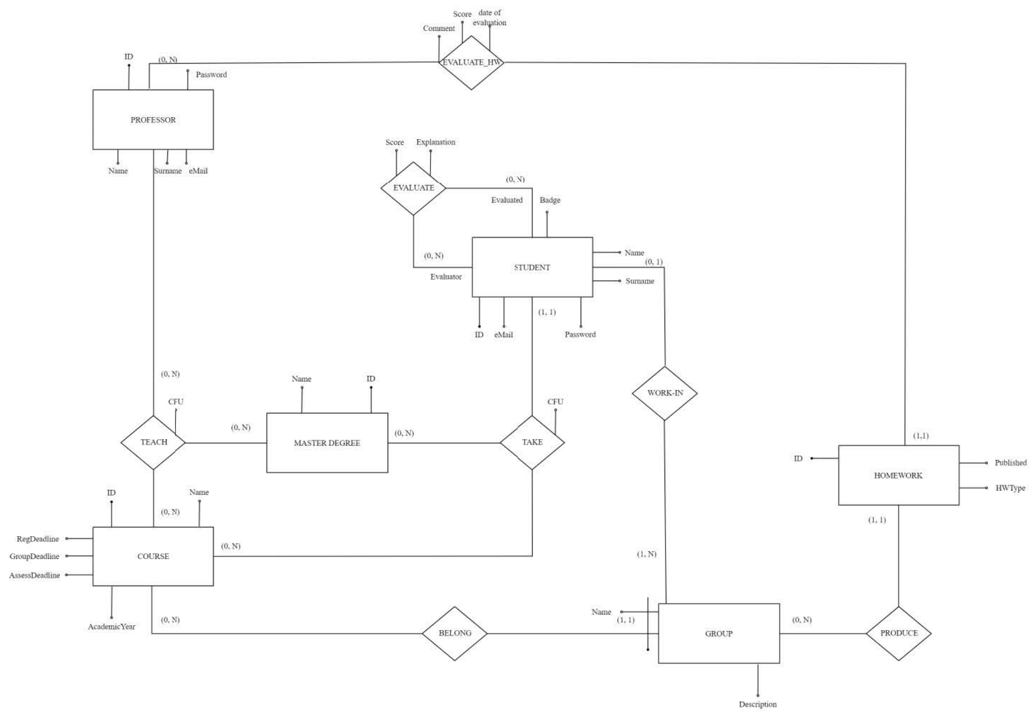


Figure 7: ER Schema of CRANE

The ER schema contains the following entities:

- **Student**: the student entity represents the students who attend a course and use CRANE to join a group. The primary key of the student is a serial number which auto-increments as soon as a new student instance is added. The other attributes are: the name (type: TEXT), the surname (type: TEXT), the email (type: CHARACTER VARYING), the password (type: CHARACTER VARYING), the badge number (type: CHARACTER VARYING). The password is hashed through md5. The badge number can be null; while all the other attributes cannot. Each student is in a 1-1 relation with the Course and Master Degree entities (relationship: *Take*). As consequence, student contains as foreign keys both the course ID and the master degree ID. Each student instance can work in at most one group, hence, there is 0-1 relationship (*Work-in*) with the entity Group. Finally, each Student instance can evaluate her teammates: this is enclosed in the relationship *Evaluate* whose tuples store the information about the evaluator and the evaluated students' IDs and the assigned scores and comments;

- **Master degree**: this entity represents the master degree the students are enrolled into. There are two attributes: the ID which is also the primary key, and the name (type: TEXT) representing the name of the master degree (e.g., "ICT - Cybersystems");

- **Course**: the course a student attends. This entity has as primary key and ID (type: serial); the other attributes are the name (type: TEXT) which represents the name of the course, and three deadlines (type: DATERANGE) which defines the intervals of time where registration, group creation and student evaluation respectively are allowed;

- **Group**: the group entity represents the group created by a student. The primary key are a name (type: CHARACTER VARYING), and the course ID which is an external identifier referencing the Course entity. The other attribute is the description (type: TEXT). Each group must belong to exactly one Course instance (1-1 cardinality in the relationship *Belong*). Each Group instance contains 1-N students (this information is included in the relationship: *Work-in*), and produces 0-N homeworks (*Produce* relationship).

- **Homework**: The homework entity represents the tasks assigned by the professor during the semester to be solved by the groups of students. It has an ID as the primary key which is of type SERIAL. Other attributes are: the assigned score (REAL), the comment (TEXT), the type of homework (TEXT), and a boolean attribute which tells if the homework evaluation has been published. Each Homework instance is produced by exactly one group (1-1 cardinality in the *Produce* relationship), and is evaluated by exactly one Professor instance at a specific timestamp (1-1 cardinality in the *Evaluate_HW* relationship). Given that each homework is produced by exactly one group, the group ID (and the course ID) are foreign keys of the homework.

- **Professor**: The Professor instance represents the professor of a course. The professor has an auto-incremental ID as a primary key. Other attributes are: the name, surname, badge number. A professor can teach 0-N courses in 0-N master degrees (*Teach* relationship): given the cardinality, the tuples of *Teach* relation store the professor ID, the course ID, and the master ID.

4.2 Other Information

It is possible to interact with all the entities, in particular, perform INSERT, UPDATE, and DELETE operations directly via the interface for what concerns the Group entity and the Work-in relations. All the other entities cannot be deleted instead, and only insertions and updates are possible.
The external identification of the Group entity allows students to create for different courses groups having the same name: in this way, if students want to keep the same group for several courses, they can create a group for each course having the same name.
Given that each student instance must attend one course, for each course a student attends, must provide a new registration, meaning that, there exist multiple instances with the same email, name, and surname, pointing to the same person.

5 Business Logic Layer

5.1 Class Diagram

The class diagram contains (some of) the classes used to handle students and groups. We describe three servlets: *HomeServlet* that retrieves the information of a student, the *StudentServlet* that handles the Student resource and that allows a student to login, register, and handles all the information of a student (or a list of students), and the *GroupServlet* that handles the group resource allowing students to create new groups, get group information, and delete members from one or more groups.
Each servlet extends the *AbstractDatabaseServlet* which is needed to acquire the connection to the database.
The student resource is entirely handled by the StudentServlet: this servlet implements the doGet and the doPost methods. Not all the URIs are freely accessible: while login and registration are, the URIs to get the information about one or more users can be accessed by professors or students (filters are not provided in this class diagram due to space reasons). Since several URIs maps to the StudentServlet, the URIs are parsed to decide the method requested by the user who is interacting with CRANE. We report two of the operations handled by the StudentServlet are: the student login, and the retrieval of student information. The StudentLoginDAO class allows to check the presence of a student in the database, and creates a new student resource, while the GetStudentsByEmailDAO class allows to retrieve the information about a student given the email.
The *HomeSevlets* implements only the doGet method (whose details such as the associated DAO classes are not shown). The *GroupServlet* implements the doGet, doPost, and doDelete methods. This servlet handles the group resource, in particular it handles the retrieval of the groups given a course, the insertion of a new group, and the deletion of a member from a group. The CreateGroupDAO allows the insertion of a new group in the database, and the GetGroupsByCourseDAO retrieves all the groups given a course, hence these two DAO classes allow to retrieve and insert new data in the database. Each DAO class extends the AbstractDAO class (which is delegated to the interaction with the database), and defines the *doAccess()* method. The AbstractDAO abstract class implements the DataAccessObject interface.
The Message resource is crucial to communicate messages to the final user – e.g, errors that occurred handling the resources. All the resources (message, student, group) are java classes: each one with its constructors and methods to access the resource.

5.2 Sequence Diagram

Here we report the sequence diagram for the group creation operation. The student, in order to create a new group, the student clicks on the *Create* button, and a POST request is issued to the web server at the following URI: /group/create/. Tue data passed to the web server are the name and the description. A similar description and sequence diagram holds for the members added to the instantiated group. The web server instantiates the GroupServelt which calls its doPost method, passing the HttpServletRequest and HttpServletResponse. Given the POST data, it is created a new Group object with those data. The control is passed to the GroupCreationDAO which receives as arguments the connection (defined in the AbstractDatabaseServlet extended by the GroupServlet) and the instantiated group. The GroupCreationDAO extends the AbstractDAO and contacts the Database Server which executes the SQL statement for group creation. If some errors occur, the control is returned to the SessionServlet and a new Message object is created. Finally, a new page is returned to the user with the information concerning the created group.

5.3 REST API Summary

The largest part of the endpoints require an additional parameter on the request to specify the request to perform, for example the endpoint associated to the homework publishing and insertion.
Part of the URIs are filtered through different filters. S indicates that only students can access to the endpoint, P

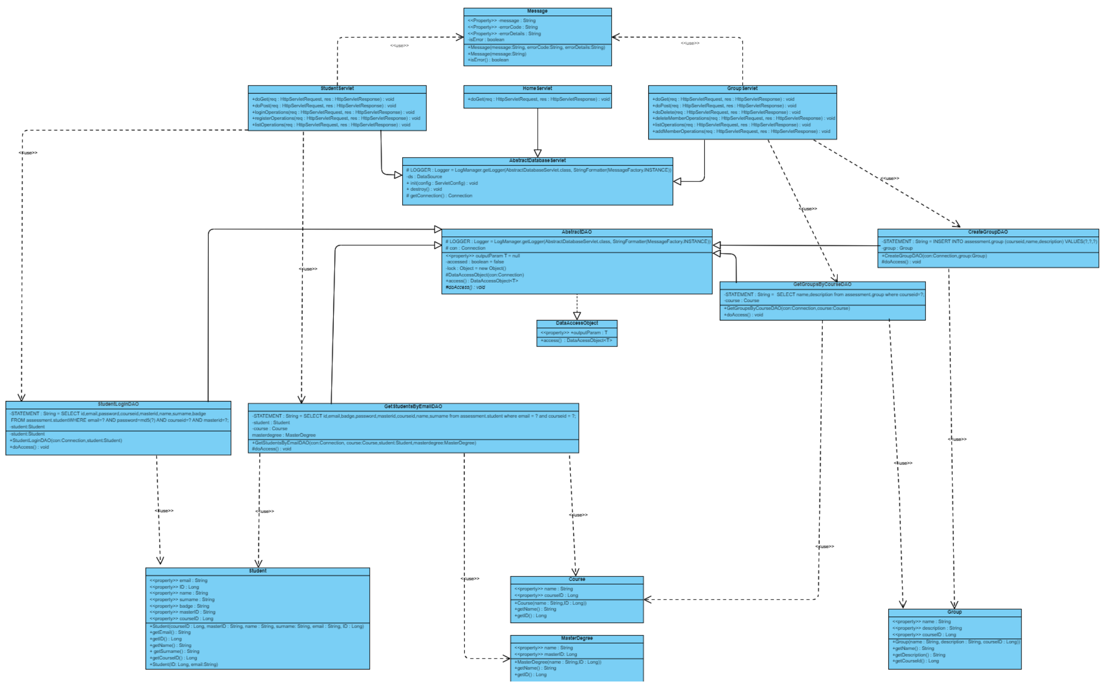


Figure 8: class diagram for part of student and group servlets

indicates that only professors can access the endpoint, A that only administrators can request the specified URI to the server.

URI	Method	Description	Filter
/student/register/	POST	Allows to register a new student in CRANE	
/student/login/	POST	Allows to login a student in CRANE with her credentials	
/student/update/	POST	Allows to update the scores a student obtained in homeworks, exam, and teammates	S
/student/logout/	POST	Allows to logout a session from CRANE, and the session is cleared	S
/student/info/	GET	returns the information about a student	S
/group/create/	POST	Allows to create a new group	S
/group/addmember/	POST	Allows to add a new member to an already existing group	S
/group/deletemember	DELETE	Delete a member from a group	S
/group/	DELETE	Delete a group	S
/home/	GET	Returns the information to be added in the homepage: group information, memebrs..	S
/evaluation/	GET	Returns a json with all the evaluation given by the logged in user to her teammates.	S

/evaluation/	POST	Add the new teammates evaluations performed by a student	S
/evaluation/update/	POST	Update the teammates evaluations performed by a student	S
/password/reset/	POST	Updates the password of a student	
/courses/list/	GET	Returns the list of courses	
/masterdegree/list/	GET	Returns the list of master degrees	
/homeworkscore/group/	GET	Returns a json with the homeworks evaluations provided by a professor for a course	P
/homeworkscore/insert/	POST	Insert in the database the evaluation for the homeworks of one or more groups	P
/homeworkscore/publish/	POST	Publish the evaluations and makes them visible to the students	P
/courses/update-deadlines/	POST	Update the deadlines for a course	P
/courses/insert/	POST	Insert a new course	P
/professor/register/	POST	Register a new professor	A

Table 2: Describe in this table your REST API

5.4 REST Error Codes

Here the list of errors defined in the application. Application specific errors have the application error which follows a progressive numeration starting from -100. METHOD NOT ALLOWED errors are identified with the error code -500. Internal errors, which correspond to crashes, servlet exceptions, or problems with the input/output streams are identified with the Error Code -999.

Error Code	**HTTP Status Code**	**Description**
-100	BAD_REQUEST	Wrong format
-101	NOT_FOUND	No homework to publish
-102	BAD_REQUEST	One or more fields of the registration form are empty
-103	BAD_REQUEST	Email of the registration/login form is missing
-104	BAD_REQUEST	Password of the registration/login form is missing
-105	BAD_REQUEST	Course of the registration/login form is missing
-106	BAD_REQUEST	Master degree of the registration/login form is missing
-108	CONFLICT	Email already used
-109	BAD_REQUEST	Wrong group information
-110	BAD_REQUEST	Problems in students update
-111	BAD_REQUEST	Registration deadline expired
-112	CONFLICT	The selected course is not provided by the selected master degree
-113	BAD_REQUEST	Wrong json format
-114	BAD_REQUEST	Deadline expired
-115	NOT_FOUND	Group member not found
-200	BAD_REQUEST	Operation unknown
-500	METHOD_NOT_ALLOWED	The method is not allowed
-999	INTERNAL_SERVER_ERROR	Error server is not able to manage

Table 3: Describe in this table your REST API

5.5 REST API Details

We report three different resource types handled by CRANE.

Student login

The following endpoint allows to login a student in CRANE

- URL: `/student/login/`
- Method:
  POST
- URL Parameters:
  None
- Data Parameters:
  Required:
  `email = {string}`
  `password = {string}`
  need to comply with the constraints imposed for the password
  `course = {string}`
  `master degree = {string}`

  Optional:
  `badge number = {string}`

- Success Response
  Code: 200
  Content: the user is redirected to the homepage once that a new session for the user is created.
- Error Response:
  Code: 500 INTERNAL_SERVER_ERROR
  Content: `{"error":{"code":-999, "message":"Internal Error."}}`
  When: if there is an SQLException or a NamingException, this error is returned

  Code: -400 BAD_REQUEST
  Content: `{"error":{"code":-102, "message":"Some fields are empty."}}`
  When: the student is trying to register without inserting some mandatory fields.

  Code: 409 CONFLICT
  Content: `{"error":{"code":-108, "message":"Mail already used."}}`
  When: the student is trying to register but she has already registered before.

  Code: 401 UNAUTHORIZED
  When: The uri is accessible only to logged users with at least the role "builder".

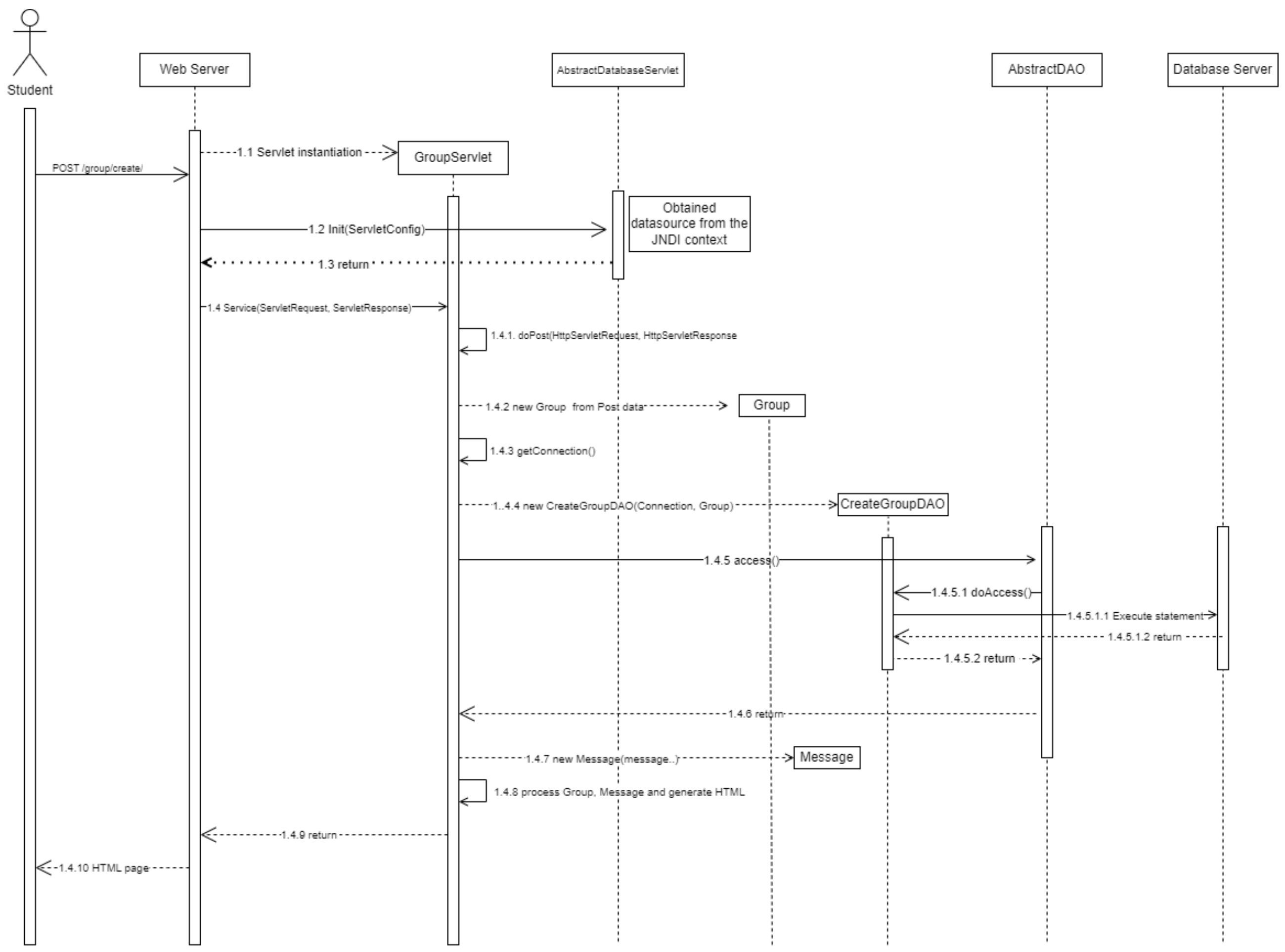

Student
Web Server
AbstractDatabaseServlet
AbstractDAO
Database Server
POST /group/create/
1.1 Servlet instantiation
GroupServlet
1.2 Init(ServletConfig)
Obtained datasource from the JNDI context
1.3 return
1.4 Service(ServletRequest, ServletResponse)
1.4.1. doPost(HttpServletRequest, HttpServletResponse
1.4.2 new Group from Post data
Group
1.4.3 getConnection()
1..4.4 new CreateGroupDAO(Connection, Group)
CreateGroupDAO
1.4.5 access()
1.4.5.1 doAccess()
1.4.5.1.1 Execute statement
1.4.5.1.2 return
1.4.5.2 return
1.4.6 return
1.4.7 new Message(message..)
Message
1.4.8 process Group, Message and generate HTML
1.4.9 return
1.4.10 HTML page

Figure 9: Sequence diagram for group creation operation

Courses list

The following endpoint allows to get the list of courses in CRANE

- URL: `/courses/list/`
- Method:
  POST
- URL Parameters:
  None
- Data Parameters: None

- Success Response

  Code: 200
  Content:
  `{"data":{[{"id":1,"name":"Search Engines"},`

`{"id":2,"name":"Web applications"},`
`{"id":3,"name":"Foundations of Databases"}]}}`

- Error Response:
  Code: 500 INTERNAL_SERVER_ERROR
  Content: `{"error":{"code":-999, "message":"Internal Error."}}`
  When: if there is an SQLException or a NamingException, this error is returned

Delete group member

The following endpoint allows to delete a member from a group

- URL: `/group/deletemember/`
- Method:
  DELETE
- URL Parameters:
  None
- Data Parameters:
  `groupname = {string}`
  `courseID = {string}`
  the email of the member to be removed
  `email = {string}`
- Success Response **Code**: 200
  Content:
  The webpage is updated with the new information
- Error Response:
  Code: -400 BAD_REQUEST
  Content: `{"error":{"code":-103, "message":"Email missing."}}`
  When: The email of the user to delete is missing.

  Code: -400 BAD_REQUEST
  Content: `{"error":{"code":-105, "message":"wrong credentials."}}`
  When: email of the user to delete not found.

  Code: 500 INTERNAL_SERVER_ERROR
  Content:
  When: if there is an SQLException or a NamingException, this error is returned

6 Group Members Contribution

Ornella Irrera contributed to the entire homework.