Università degli studi di Firenze

Tutoring Web App: GoTutor

DPWIM Project

06 February 2024

Student: Giamberini Giulia

Email: giulia.giamberini@edu.unifi.it

Identification number: 7149574

A.A.: 2023/2024

SOMMARIO

PROJECT OVERVIEW

GoTutor is a web application that facilitates seamless interaction between students and tutors. The platform allows students to prenote lessons, interact with graphical content, and browse available tutors. Tutors can manage their schedules and availability through a Command Line Interface (CLI), ensuring efficient coordination of lessons.

DESCRIPTION OF DESIGN

Architecture Overview:

GoTutor adopts a client-server architecture, with the server tasked with managing HTTP requests from clients (web browsers) and interfacing with the SQLite database for data retrieval and storage. This server is constructed using the GoLang programming language, leveraging the Gin framework for efficient routing and middleware management.

Serving as the core component of the application, it orchestrates communication between clients and the database.

On the client side, the interface is crafted using a combination of HTML, CSS, and Bootstrap, providing users with a dynamic and intuitive experience.
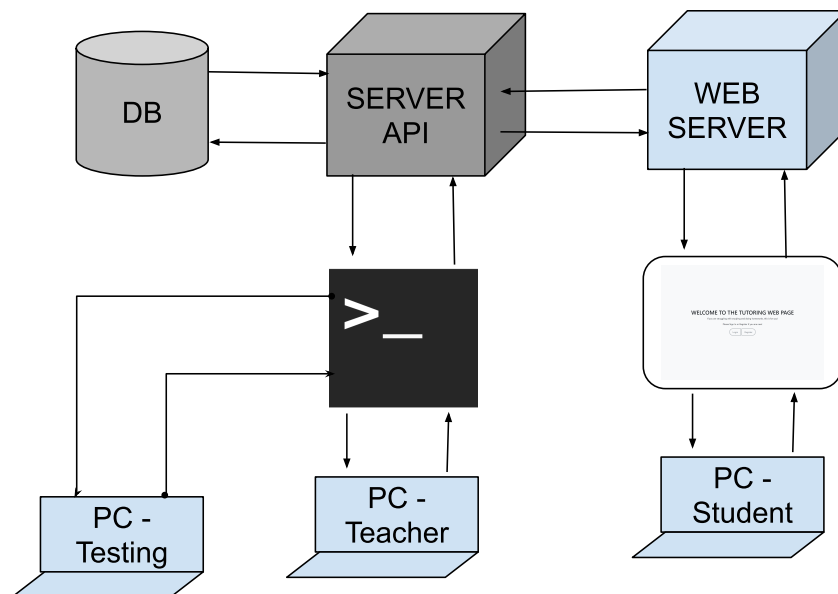
Backend Design

The backend of GoTutor comprises three key components: a server responsible for managing HTTP requests, a database for storing data, and a server API facilitating communication between the server and the database.

1. **Server (over port 5050):**

   The server handles incoming HTTP requests from clients and coordinates interactions between the client-side interface and the database. It was developed using the GoLang programming language and the Gin framework for robust routing and middleware management. It serves as the central hub of the application, orchestrating the flow of data and responses between clients and the database.

```go
func server() {
    fmt.Println("Web server is running on port 5050")
    http.HandleFunc("/", rootHandler)
    http.HandleFunc("/login", loginHandler)
    http.HandleFunc("/logout", logoutHandler)
    http.HandleFunc("/registration", registrationHandler)
    http.HandleFunc("/userregistration", userRegistrationHandler)
    http.HandleFunc("/welcome", welcomeHandler)
    http.HandleFunc("/profile", profileHandler)
    http.HandleFunc("/bookings", bookingsHandler)
    http.HandleFunc("/deleteBooking", deleteBookingHandler)
    http.HandleFunc("/booklesson", bookLessonHandler)
    http.HandleFunc("/availability", availabilityHandler)
    http.HandleFunc("/bookedLesson", bookedLessonHandler)

    http.ListenAndServe("localhost:5050", nil)
}
```

2. **Database:**

   GoTutor utilizes an SQLite database for efficient and lightweight data storage. The database stores various data related to students, tutors, lessons, availability slots, and reservations, ensuring data integrity and consistency. Through structured queries and transactions, the database efficiently retrieves and updates information based on requests from the server.
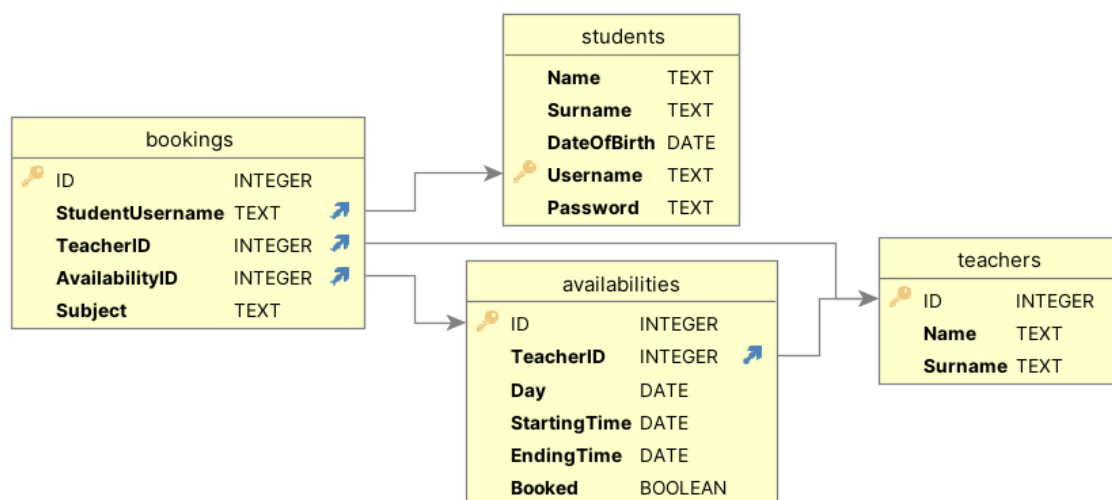
3. **Server API (over port 8080):**

The server API acts as an intermediary layer between the server and the database, facilitating communication and data exchange. It was developed to provide a set of endpoints for performing CRUD (Create, Read, Update, Delete) operations on the database.

```go
func routingAPI() {
    fmt.Println("API server is running on port 8080")

    router := gin.Default() // Using gin.Default() to set up the default middleware

    api := router.Group("/api")

    teachersGroup := api.Group("/teachers")
    teachersGroup.GET("", getTeachers)
    teachersGroup.GET("/:name/:surname", getTeacherIDByNameAndSurname)
    teachersGroup.POST("/addteacher", createNewTeacher)

    teacherGroup := api.Group("/teacher")
    teacherGroup.GET("/:id/availability", getTeacherAvailability)
    teacherGroup.GET("/:id/bookings", getTeacherBookings)
    teacherGroup.POST("/:id/availability", createTeacherAvailability)

    studentGroup := api.Group("/student")
    studentGroup.POST("/addstudent", createNewStudent)
    studentGroup.GET("/allstudents", getStudents)
    studentGroup.GET("/:username/profile", getProfileStudent)
    studentGroup.GET("/:username/bookings", getStudentBookings)
    studentGroup.POST("/:username/bookings", createStudentBooking)
    studentGroup.POST("/bookings/:id", deleteStudentBooking)
    // Run the server
    router.Run("localhost:8080")
}
```

Frontend Design

1. **Web Interface (student):**

The student can access the web app using the browser. It can interact with some HTML pages that are rendered by the server. Using the Bootstrap framework and some CSS, the user experience is optimized. The content of each page is dynamically added using the Templates. The navigation on each page is ensured due to the submit button. Before loading the page, the session is checked so, that if it is expired, the login page is rendered.

2. **CLI interface (teacher):**

The teacher can be added only via CLI. A user-friendly interface was developed to help the admin or the teacher himself to add his information, such as his name and his availabilities. The CLI can be used also for testing all the teacher and student-related operations It is possible to use a specific command for running the project in this mode.

TECHNOLOGIES AND TOOLS

- GoLang (Go): Used for backend server development due to its efficiency and robust standard library.
- Gin: Utilized for building the HTTP server in GoLang. Gin provides routing and other essential functionalities for web development.
- SQLite: Chosen for its lightweight nature, simplicity, and ease of integration with GoLang. SQLite is used as the backend database for storing data related to students, tutors, lessons, and availability slots.
- HTML: Used for structuring the content and layout of web pages.
- CSS: Employed for styling the presentation of the frontend interface.
- Bootstrap: Integrated for front-end development to ensure responsive design.
- Templates: used for adding dynamic content to the HTML pages, according to the result of the request made by the user
- Sessions: It was used to manage user authentication and authorization. Sessions enable secure user access to protected resources within the application

INSTRUCTION TO COMPILE

To compile the project it's necessary to move into the folder from the CMD (or in VSCode) and run:

```
go build -o server.exe
```

It will generate an executable file using all the files in the folder.

For running the project is necessary to use:

1. For launching the API server and the database
   ```
   server.exe -m server
   ```

2. For launching the HTTP server:
   ```
   server.exe -m web
   ```

3. For launching the CLI interface for the tutors:
   ```
   server.exe -m cli
   ```

4. For launching the CLI as the only interface of the app, to test all the functionalities from the command prompt:

```
server -m cli -test
```

TEST CASES

When the command for launching the server is executed, this is what is shown in the CMD:

```
>go build -o server.exe

>server.exe -m server
API server is running on port 8080
Database connection...
```

With only the API server and the database up, it's possible to launch the web server or the CLI interface for managing Student's reservations and Tutors' availabilities, respectively.

If the CLI interface was launched, it is what is shown:

```
>server.exe -m cli
Welcome to the Menu!

Menu Options:
1. Add a teacher
2. Add an availability for a specific teacher
3. List all availabilities for a specific teacher
4. List all teachers
0. Exit
Select an option (0-4):
```

From the Menu is possible to select the best option for the goal of the interaction. For example, in this case, was added a teacher, an availability for this specific tutor, and listed all the availabilities.

In the following screenshots, it was shown the responses of the server for each operation made in the CLI interface.

```
>go build -o server.exe

>server.exe -m server
API server is running on port 8080
Database connection...
[GIN] 2024/02/07 - 16:01:43 | 201 |     11.1718ms |
 127.0.0.1 | POST    "/api/teachers/addteacher"
```

```
>server.exe -m cli
Welcome to the Menu!

Menu Options:
1. Add a teacher
2. Add an availability for a specific teacher
3. List all availabilities for a specific teacher
4. List all teachers
0. Exit
Select an option (0-4): 1
Adding a teacher...
Enter the teacher's name: Jack
Enter the teacher's surname: Frost

 ┌─────────────────────────────┐
 │ Teacher added successfully! │
 └─────────────────────────────┘

Menu Options:
1. Add a teacher
2. Add an availability for a specific teacher
3. List all availabilities for a specific teacher
4. List all teachers
0. Exit
Select an option (0-4):
```

Adding a new teacher by Name and Surname. No control over homonym was made. If this is necessary, a new field could be added to identify the teacher, such as the date of birth.

After the creation of the teacher, is possible to add an availability. The availability needs to be of 1 hour max and it can't overlap with other availabilities of the same teacher. If this happens, an error message occurs.

```
>go build -o server.exe

>server.exe -m server
API server is running on port 8080
Database connection...
[GIN] 2024/02/07 - 16:01:43 | 201 |     11.1718ms |
 127.0.0.1 | POST    "/api/teachers/addteacher"
[GIN] 2024/02/07 - 16:03:10 | 200 |       518.6µs |
 127.0.0.1 | GET     "/api/teachers/Jack/Frost"
[GIN] 2024/02/07 - 16:03:26 | 201 |     12.2362ms |
 127.0.0.1 | POST    "/api/teacher/3/availability"
```

```
>server.exe -m cli
Welcome to the Menu!

Menu Options:
1. Add a teacher
2. Add an availability for a specific teacher
3. List all availabilities for a specific teacher
4. List all teachers
0. Exit
Select an option (0-4): 1
Adding a teacher...
Enter the teacher's name: Jack
Enter the teacher's surname: Frost

 ┌─────────────────────────────┐
 │ Teacher added successfully! │
 └─────────────────────────────┘

Menu Options:
1. Add a teacher
2. Add an availability for a specific teacher
3. List all availabilities for a specific teacher
4. List all teachers
0. Exit
Select an option (0-4): 2
Adding an availability for a specific teacher...
Enter the teacher's name: Jack
Enter the teacher's surname: Frost
Enter the day: 04/02/2024
Enter the starting time: 15:00
Enter the starting time: 16:00

 ┌────────────────────────────────────┐
 │ Availability added successfully!   │
 └────────────────────────────────────┘
```

```
>go build -o server.exe

>server.exe -m server
API server is running on port 8080
Database connection...
[GIN] 2024/02/07 - 16:01:43 | 201 |     11.1718ms |
 127.0.0.1 | POST    "/api/teachers/addteacher"
[GIN] 2024/02/07 - 16:03:10 | 200 |       518.6µs |
 127.0.0.1 | GET     "/api/teachers/Jack/Frost"
[GIN] 2024/02/07 - 16:03:26 | 201 |     12.2362ms |
 127.0.0.1 | POST    "/api/teacher/3/availability"
[GIN] 2024/02/07 - 16:04:15 | 200 |       503.4µs |
 127.0.0.1 | GET     "/api/teachers/Jack/Frost"
[GIN] 2024/02/07 - 16:04:15 | 200 |       523.2µs |
 127.0.0.1 | GET     "/api/teacher/3/availability"
```

```
Menu Options:
1. Add a teacher
2. Add an availability for a specific teacher
3. List all availabilities for a specific teacher
4. List all teachers
0. Exit
Select an option (0-4): 2
Adding an availability for a specific teacher...
Enter the teacher's name: Jack
Enter the teacher's surname: Frost
Enter the day: 04/02/2024
Enter the starting time: 15:00
Enter the starting time: 16:00

 ┌────────────────────────────────────┐
 │ Availability added successfully!   │
 └────────────────────────────────────┘

Menu Options:
1. Add a teacher
2. Add an availability for a specific teacher
3. List all availabilities for a specific teacher
4. List all teachers
0. Exit
Select an option (0-4): 3
Listing all availabilities for a specific teacher...
Enter the teacher's name: Jack
Enter the teacher's surname: Frost
Availabilities:
ID:  7
Day:  Sunday, 4 February 2024
Starting time:  15:00
Ending time:  16:00
Booked:  false
-----------------------------------------------------------------
```

It is also possible to list all the availabilities of a specific teacher and see if were booked or not.

In this case, isn't possible to see which student has booked the lesson and which subject was chosen.

8

```
>go build -o server.exe

>server.exe -m server
API server is running on port 8080
Database connection...
[GIN] 2024/02/07 - 16:01:43 | 201 |      11.1718ms |
  127.0.0.1 | POST    "/api/teachers/addteacher"
[GIN] 2024/02/07 - 16:03:10 | 200 |       518.6µs |
  127.0.0.1 | GET     "/api/teachers/Jack/Frost"
[GIN] 2024/02/07 - 16:03:26 | 201 |      12.2362ms |
  127.0.0.1 | POST    "/api/teacher/3/availability"
[GIN] 2024/02/07 - 16:04:15 | 200 |       503.4µs |
  127.0.0.1 | GET     "/api/teachers/Jack/Frost"
[GIN] 2024/02/07 - 16:04:15 | 200 |       523.2µs |
  127.0.0.1 | GET     "/api/teacher/3/availability"
[GIN] 2024/02/07 - 16:05:01 | 200 |       506.8µs |
  127.0.0.1 | GET     "/api/teachers"
[GIN] 2024/02/07 - 16:05:08 | 200 |       507.7µs |
  127.0.0.1 | GET     "/api/teachers"
```

```
Menu Options:
1. Add a teacher
2. Add an availability for a specific teacher
3. List all availabilities for a specific teacher
4. List all teachers
0. Exit
Select an option (0-4): 4
Listing all teachers...
Teachers:
Teacher ID:  1
Name:  Jhon
Surname:  Doe
-----------------------------------------------------------------
Teacher ID:  2
Name:  Erik
Surname:  Smith
-----------------------------------------------------------------
Teacher ID:  3
Name:  Jack
Surname:  Frost
-----------------------------------------------------------------
```

In the end, is possible to list all the teachers who are memorized in the database

For testing all the operations from the CLI (not only teachers but also students) is possible to use the following command:

```
>server.exe -m cli -test
Welcome to the Menu!

Menu Options:
1. Add a teacher
2. Add an availability for a specific teacher
3. List all availabilities for a specific teacher
4. List all teachers
5. Add a student
6. List all students
7. Get profile of a specific student
8. Book an availability for a specific teacher
9. List all bookings for a specific student
0. Exit
Select an option (0-9):
```

From this interface is possible to add a new student by Name, Surname, Date of birth, username and password.

Before the creation of the student, is checked if the username is not already in use. If yes, an error message is displayed in the prompt.

```
>server.exe -m server
API server is running on port 8080
Database connection...
[GIN] 2024/02/07 - 16:08:43 | 201 |      66.4882ms |
  127.0.0.1 | POST    "/api/student/addstudent"
[GIN] 2024/02/07 - 16:08:55 | 200 |       1.1046ms |
  127.0.0.1 | GET     "/api/student/StudentUsername/profi
le"
```

```
Menu Options:
1. Add a teacher
2. Add an availability for a specific teacher
3. List all availabilities for a specific teacher
4. List all teachers
5. Add a student
6. List all students
7. Get profile of a specific student
8. Book an availability for a specific teacher
9. List all bookings for a specific student
0. Exit
Select an option (0-9): 5
Adding a student...
Enter the student's name: StudentName
Enter the student's surname: StudentSurname
Enter the student Date of Birth: 01/01/2000
Enter the student's username: StudentUsername
Enter the student's password: Password

Student added successfully!
```

If the data provided conforms to the specific of the Student struct, it is saved in the database, ensuring that the password is encrypted to avoid password thefts

From this interface is also possible to see the profile of a student (without the password), book an appointment for a specific teacher and list all the bookings made by the student.

```
>server.exe -m server                          Menu Options:
API server is running on port 8080             1. Add a teacher
Database connection...                         2. Add an availability for a specific teacher
[GIN] 2024/02/07 - 16:08:43 | 201 |    66.4882ms |   3. List all availabilities for a specific teacher
 127.0.0.1 | POST    "/api/student/addstudent" 4. List all teachers
[GIN] 2024/02/07 - 16:08:55 | 200 |     1.1046ms |   5. Add a student
 127.0.0.1 | GET    "/api/student/StudentUsername/profi  6. List all students
le"                                            7. Get profile of a specific student
|                                              8. Book an availability for a specific teacher
                                               9. List all bookings for a specific student
                                               0. Exit
                                               Select an option (0-9): 7
                                               Showing profile of a specific student...
                                               Enter the student's username: StudentUsername
                                               Enter the student's password: Password

                                               ┌─────────────────────────────────────────┐
                                               │ Student Profile                           │
                                               ├─────────────────────────────────────────┤
                                               │ Name:           StudentName               │
                                               │ Surname:        StudentSurname            │
                                               │ Date of Birth: Saturday, 1 January 2000  │
                                               │ Username:       StudentUsername           │
                                               └─────────────────────────────────────────┘
```

```
>server.exe -m server                          Menu Options:
API server is running on port 8080             1. Add a teacher
Database connection...                         2. Add an availability for a specific teacher
[GIN] 2024/02/07 - 16:08:43 | 201 |    66.4882ms |   3. List all availabilities for a specific teacher
 127.0.0.1 | POST    "/api/student/addstudent" 4. List all teachers
[GIN] 2024/02/07 - 16:08:55 | 200 |     1.1046ms |   5. Add a student
 127.0.0.1 | GET    "/api/student/StudentUsername/profi  6. List all students
le"                                            7. Get profile of a specific student
[GIN] 2024/02/07 - 16:10:04 | 200 |      577.5µs |   8. Book an availability for a specific teacher
 127.0.0.1 | GET    "/api/student/StudentUsername/profi  9. List all bookings for a specific student
le"                                            0. Exit
[GIN] 2024/02/07 - 16:10:09 | 200 |      608.5µs |   Select an option (0-9): 8
 127.0.0.1 | GET    "/api/teachers/Jack/Frost" Adding a booking for a specific teacher by the student X...
[GIN] 2024/02/07 - 16:10:09 | 200 |         0s |   Enter the student's username: StudentUsername
 127.0.0.1 | GET    "/api/teacher/3/availability"  Enter the teacher's name: Jack
[GIN] 2024/02/07 - 16:10:32 | 201 |    32.7692ms |   Enter the teacher's surname: Frost
 127.0.0.1 | POST    "/api/student/StudentUsername/booki  Availabilties of Jack Frost
ngs"                                           7. 04/02/2024 15:00 - 16:00
|                                              Enter the ID of the availability you want to book: 7
                                               Enter the subject you want to book: Algebra

                                               ┌──────────────────────────────┐
                                               │ Lesson booked successfully   │
                                               └──────────────────────────────┘
```

```
>server.exe -m server                          Menu Options:
API server is running on port 8080             1. Add a teacher
Database connection...                         2. Add an availability for a specific teacher
[GIN] 2024/02/07 - 16:08:43 | 201 |    66.4882ms |   3. List all availabilities for a specific teacher
 127.0.0.1 | POST    "/api/student/addstudent" 4. List all teachers
[GIN] 2024/02/07 - 16:08:55 | 200 |     1.1046ms |   5. Add a student
 127.0.0.1 | GET    "/api/student/StudentUsername/profi  6. List all students
le"                                            7. Get profile of a specific student
[GIN] 2024/02/07 - 16:10:04 | 200 |      577.5µs |   8. Book an availability for a specific teacher
 127.0.0.1 | GET    "/api/student/StudentUsername/profi  9. List all bookings for a specific student
le"                                            0. Exit
[GIN] 2024/02/07 - 16:10:09 | 200 |      608.5µs |   Select an option (0-9): 9
 127.0.0.1 | GET    "/api/teachers/Jack/Frost" Listing all the booking made by a specific student...
[GIN] 2024/02/07 - 16:10:09 | 200 |         0s |   Enter the student's username: StudentUsername
 127.0.0.1 | GET    "/api/teacher/3/availability"  8. 2024-02-04T00:00:00Z 15:00 - 16:00 - Algebra
[GIN] 2024/02/07 - 16:10:32 | 201 |    32.7692ms |
 127.0.0.1 | POST    "/api/student/StudentUsername/booki
ngs"
[GIN] 2024/02/07 - 16:11:04 | 200 |     2.0179ms |
 127.0.0.1 | GET    "/api/student/StudentUsername/booki
ngs"
```
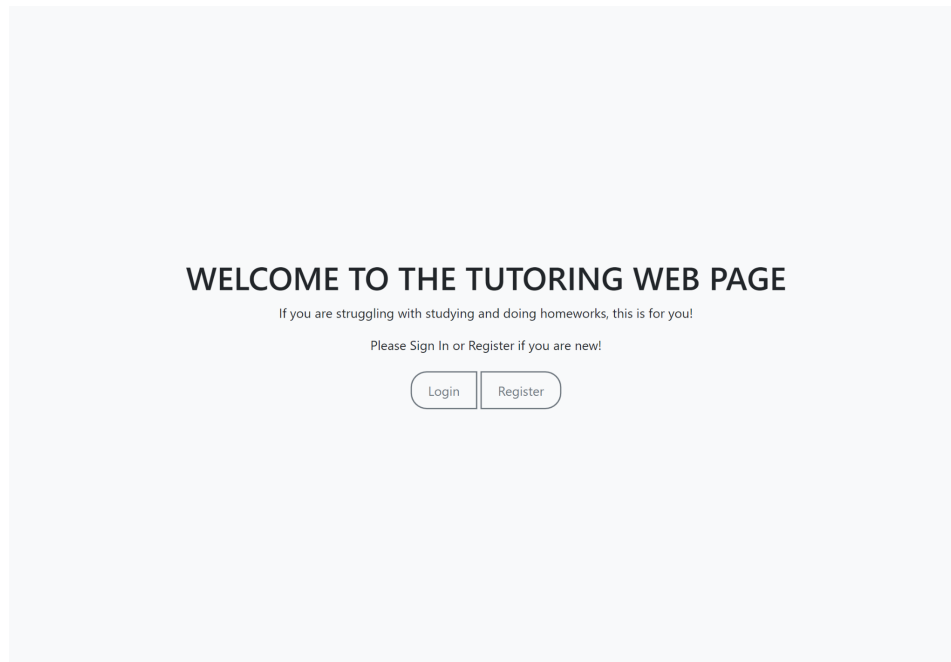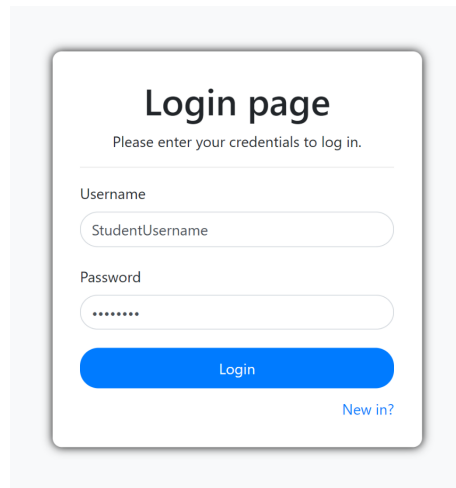
Otherwise, if the web server is launched, the student behavior can be tested from the web browser, sending HTTP requests to the server.

```
>server.exe —m web
Web server is running on port 5050
```

Loading the page localhost:5050 makes it possible to retrieve the welcoming page of the web app, which allows the client to log in or register.

During the login phase, the student provides the Username and Password. An HTTP request to access the Profile page is sent to the server who determine if the password associated with the username is correct, using the bycrypt library. If the username and password are correct, the profile page is rendered, otherwise, if the username is not present in the database, the request is redirected to the registration page. If the password provided is not the correct one, the same login page is reloaded with "Password doesn't match".
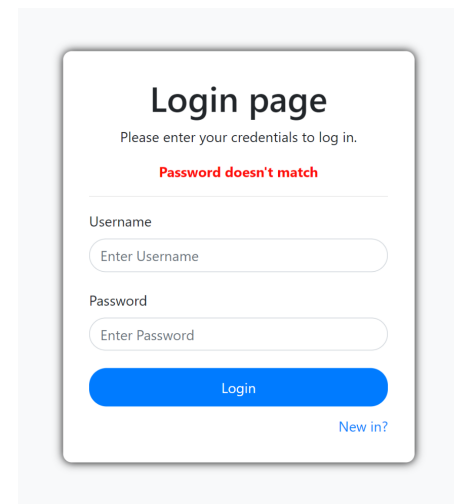
The login is correct so the submission of the form will provide the profile page.
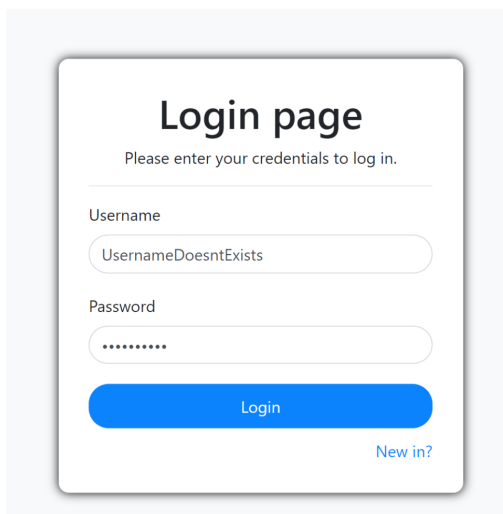
The password submitted doesn't match the password stored in the database
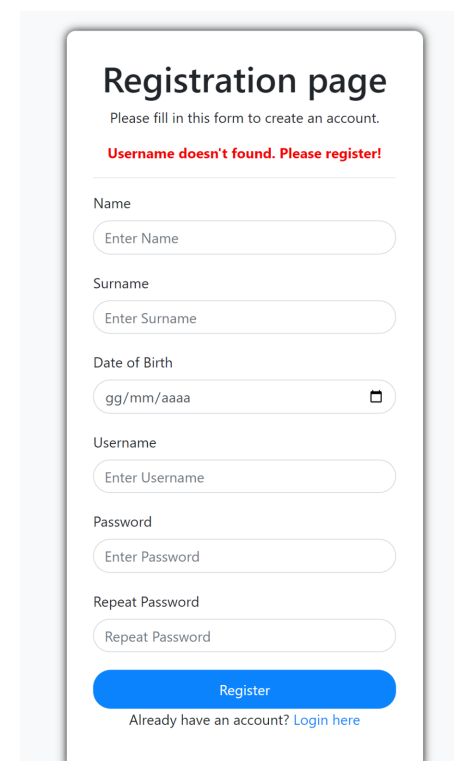
The username is not in the database so the registration form is loaded.

If the student doesn't have an account, he can register himself by providing his Name, Surname, Date of birth, Username and password. In this case, the password needs to be provided twice to simulate the correct way to register a new user. If the password are not the same, the page is reloaded with the warning message. If the Passwords are the same but the username is already in use, a message will be displayed. If no problems are raised, the user is saved in the database and the login page is rendered.

Blank registration page	If the username used for login isn't saved in the database	If the passwords are not the same

Later, passing through the login page, is possible to access the profile page with all the user information.

In the top right of the page is possible to navigate in the website, retrieving all the reservations booked or booking others. If no reservations were booked an appropriate message will be displayed.
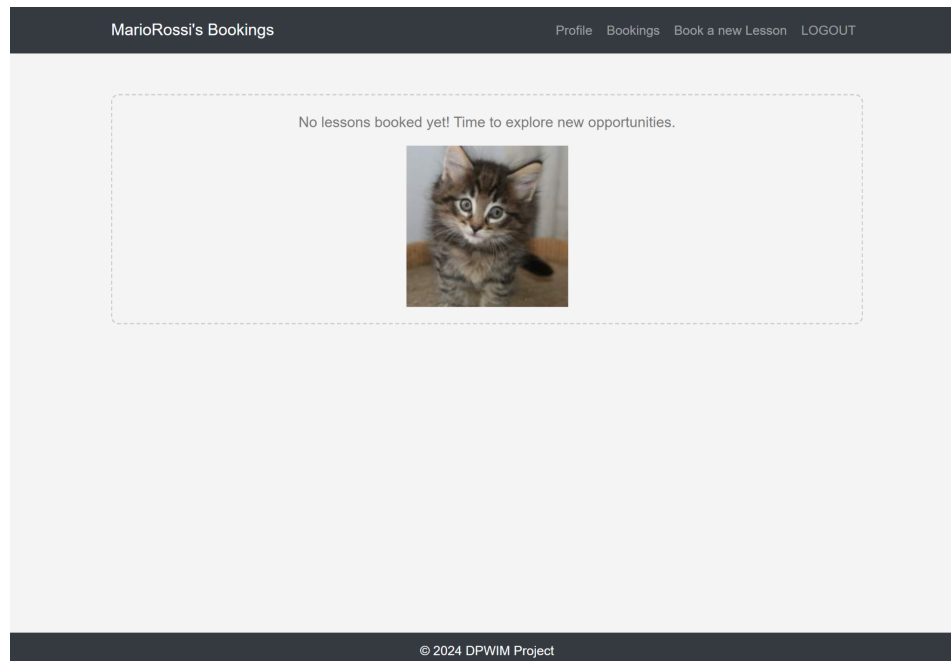
For booking a new lesson, it will be shown a list of teachers. Selecting one of them, the corresponding page of all the availabilities is loaded (only those not booked).
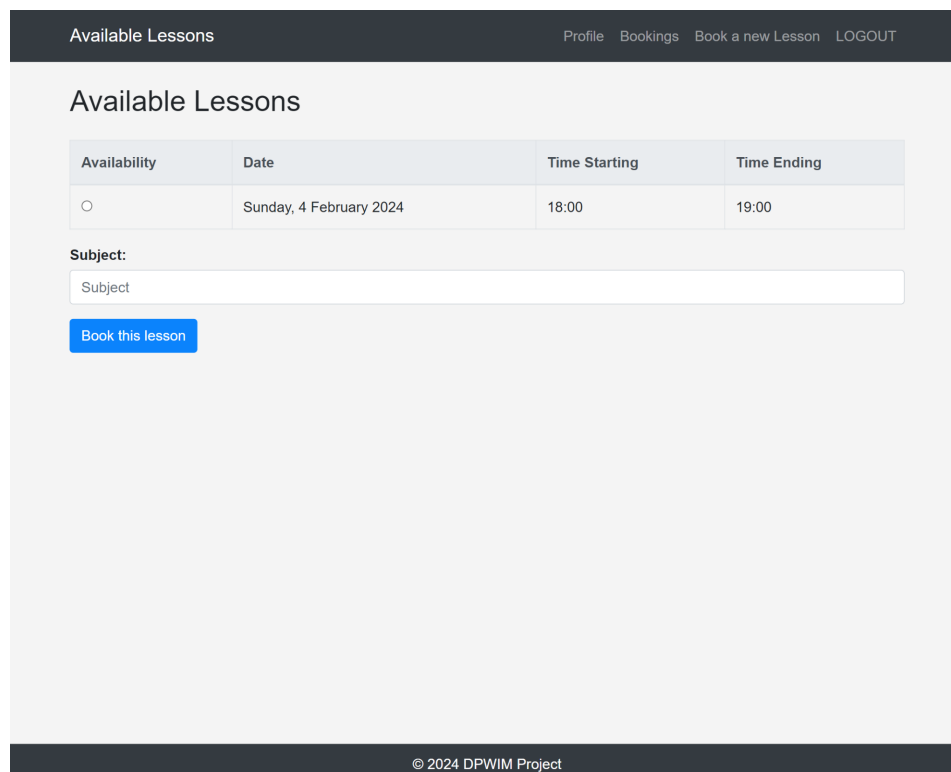
If the table is empty, no availabilities are free to book.

If the student could select one of the availabilities, he needs to specify the subject of the tutoring lesson.

If the selected availability is overlapped with another reservation made by the same student, the lesson will not be booked.

After a booking reservation is defined as suitable for the student, it will be saved and the corresponding availability of the teacher is flagged as booked. After the memorization of that information, the reservation page is loaded to list all the reservations made by the student. The student can also delete a reservation, by clicking on the trash can button.

On the following screws is possible to see the API server managing all the different requests.

```
[GIN] 2024/02/07 - 16:22:49 | 200 |      1.1015ms |
 127.0.0.1 | GET      "/api/student/StudentUsername/profi
le"
[GIN] 2024/02/07 - 16:23:03 | 200 |      1.0431ms |
 127.0.0.1 | GET      "/api/student/StudentUsername/booki
ngs"
[GIN] 2024/02/07 - 16:23:13 | 200 |      1.035ms |
 127.0.0.1 | GET      "/api/teachers"
[GIN] 2024/02/07 - 16:23:37 | 200 |      1.0478ms |
 127.0.0.1 | GET      "/api/teacher/3/availability"
[GIN] 2024/02/07 - 16:23:45 | 200 |      1.6274ms |
 127.0.0.1 | GET      "/api/teachers"
[GIN] 2024/02/07 - 16:23:47 | 200 |      1.0611ms |
 127.0.0.1 | GET      "/api/teacher/1/availability"
[GIN] 2024/02/07 - 16:24:09 | 201 |      12.7261ms |
 127.0.0.1 | POST     "/api/student/StudentUsername/booki
ngs"
[GIN] 2024/02/07 - 16:24:09 | 200 |      1.0813ms |
 127.0.0.1 | GET      "/api/student/StudentUsername/booki
ngs"
[GIN] 2024/02/07 - 16:24:56 | 200 |      1.0929ms |
 127.0.0.1 | GET      "/api/student/StudentUsername/profi
le"
[GIN] 2024/02/07 - 16:27:49 | 200 |      1.0996ms |
 127.0.0.1 | GET      "/api/student/StudentUsername/profi
le"
[GIN] 2024/02/07 - 16:28:35 | 404 |      1.3016ms |
 127.0.0.1 | GET      "/api/student/MarioRossi/profile"
[GIN] 2024/02/07 - 16:28:35 | 201 |      78.0216ms |
 127.0.0.1 | POST     "/api/student/addstudent"
[GIN] 2024/02/07 - 16:28:47 | 200 |      567.2µs |
 127.0.0.1 | GET      "/api/student/MarioRossi/profile"
[GIN] 2024/02/07 - 16:28:57 | 200 |      1.1224ms |
 127.0.0.1 | GET      "/api/student/MarioRossi/bookings"
[GIN] 2024/02/07 - 16:29:10 | 200 |      505.3µs |
 127.0.0.1 | GET      "/api/teachers"
```

```
>server.exe -m web
Web server is running on port 5050
#### No student found as MarioRossi ####
```