***BarberShop***

***Table of contents***

**I. Introduction………………………………………………………………………………..2**

**II. The beginning……………………………………………………………………………..2**

**1. How the project began……………………………………………………………….2**

1. **The initial java classes………………………………………………………………..2**

**III. The project………………………………………………………………………………..2**

**1. Login controller……………………………………………………………………….2**

**2. Register controller…………………………………………………………………….3**

**3. Appointments controller……………………………………………………………...3**

**4. MyAccount controller………………………………………………………………...3**

**5. Change Password Controller…………………………………………………………4**

**6. My Appointments Controller for employee or user…………………………………4**

**7. Select appointment Controller……………………………………………………….4**

**8. Barber date and time Controller…………………………………………………….4**

**IV. Libraries used………………………………………………………………………….…5**

1. **JavaFX………………………………………………………………………………..5**
2. **Java Database Connectivity (JDBC)………………………………………………..5**
3. **Java Standard Libraries……………………………………………………………..5**
4. **JavaFX Controls……………………………………………………………………...5**
5. **JavaFX Layout Components………………………………………………………...5**
6. **JavaFX Properties: JavaFX Properties……………………………………………..5**
7. **JavaFX Collections…………………………………………………………………...5**
8. **JavaFX Web Components…………………………………………………………...6**
9. **Java Desktop Integration…………………………………………………………….6**
10. **JavaFX Concurrency………………………………………………………………...6**
11. **Java IO………………………………………………………………………………..6**
12. **JavaFX Dialogs……………………………………………………………………….6**
13. **Date and Time Formatting…………………………………………………………..6**
14. **JavaFX Initialization………………………………………………………………...6**
15. **BCrypt**………………………………………………………………………………...6
16. **Custom Classes……………………………………………………………………….6**
17. **Introduction**

The idea behind creating a Barbershop Java project came actually from my barber, when he said he wants to open a new barbershop, having a partner that works with him. Initially I wanted to create a fully working chess game, but then I saw that, already a classmate has done that so I ended with the barbershop.

1. **The beginning**
2. **How the project began?**

Firstly, I need to do some documentation about what I will use. Our lab teacher told us about Scene Builder, so I downloaded it. It is great, and I highly recommend it.

Secondly, I needed to understand how exactly JavaFX works, so I went on google to search about it. Little by little everything came together, and then I started working on the project.

1. **The initial java classes**

Creating a new JavaFX project, and running it opens a new file, that contains a button, and when pressed it shows the known message: “Hello World!”. The design from the opened file comes from the. fxml extension, and when the file is opened in Scene Builder it is easy to manipulate it, for example adding an image, or a new button to which it is assigned an FXacid, that can have different commands when pressed, like the one from the initial project, that displayed the text.

1. **The project**
2. **Login Controller**

The login controller contains the basic login-from found in every application/website. It has the username/password fields that receive data from the keyboard. Below them there is the login button that verifies the data from the above fields, searching them in a PostgreSQL database. If the login button is pressed, without introducing data, a message will appear, and same if the username is not found, or the username is found but the password is incorrect. Also, the cancel button works like an Alt+F4 command. On the left side there is an image and data about the barber shop. In case the user wants to create a new account, there is a button that takes you to the user registration.

1. **Register Controller**

The register controller also has fields that are entered from the keyboard. First name, last name, username, password and confirm password. All of the data introduced are then taken and put in the database. The password is hashed in the database, so in case somebody breaks the database, the passwords are safe, but there is something wrong about this, because if someone forgets the password, the account cannot be recovered. The password and confirm password have to match, and there can’t be no password/ no username/ no first name/ no last name. When the button register is clicked, it checks if there are data in the fields, the passwords match, and if the username is taken. If everything goes right, a successful message will appear. Go back, and close buttons are available also.

1. **Appointments Controller**

At the top of the file a message appears: “Hello!” + username, displaying below the fully working barbers: Adam, Ionel and Costi. All three of them have the same services, but we will talk about that later. On the left side, there is data about the barbershop, and also available to follow the barber shop on Instagram, Facebook, and TikTok. When the three icons are pressed, it will take user to the pages of the social media. Top right corner there is a menu, and when pressed it has two options: my account or logout. The logout button takes the user to the login controller and my account button to the user account.

1. **MyAccount Controller**

My account has some features, for example at the top it displays the full name of the user and the username, below showing the number of appointments so far. If the number reaches a multiple of 15, then the user has a 50% discount on one cut. At the top left corner there is a back-button that takes the user to the appointment’s controller. Also, the logout button is available from here, but also the change password button, which takes you to the change password controller. The most important one is the “See your appointments”, which takes the user to my appointments controller, displaying the past and future appointments. Depending on the user, which can be the either the employee or the client, the button can take it to my appointments controller or my appointments for employee controller.

1. **Change Password Controller**

The change password controller displays a message, and has 3 data fields which takes data from the keyboard. First takes the current password, then the new password along with the confirm password. If the button change password is pressed it checks the database of the user, searching the current password. If there is a match between them, checks if the new password and confirm password are the same or not null. If everything goes correctly, the password is updated in the database. In the left corner there is a back button which takes the user to my account controller.

1. **My Appointments Controller for employee or user**

My appointments controller has a table which displays all the past and future appointments, having also the delete selected, which cancels an appointment which is in the future, not in the past. If there is not a selected appointment then an error message will appear, same if there is a past one. For the employee it shows the appointments, and he can delete the appointments.

1. **Select appointment Controller**

The select appointment controller has at the top left corner a back-button which, when clicked, takes the user back to the main program. Also, it has the menu item Choose the location, in which the user selects the location, but for the moment only one location is opened 😊. As for the last 3 menus, the user chooses for which one he wants an appointment.: beard, haircut or both. The continue button works only if the location and one options from the appointment are chosen. Then, it takes the user to the Barber date and time controller. On the right side is shown the location on google maps but unfortunately it does not work ☹.

1. **Barber date and time Controller**

The barber date and time controller take the type of appointment chosen in the previous tab, and the user has to choose a specialist out of the three then a date that is a working day, not Saturday or Sunday. After, he has to choose an available hour, between 08:00 and 17:30. Example: if I choose to make an appointment for Ionel on 08/01/2025 at 08:00 which takes 30 minutes, and another person makes an appointment for Ionel, on the same day, the next available hour is 08:45. Then, the confirm button appears, and if everything works correctly a success tab pops up.

1. **Libraries used**

The barbershop project utilizes several libraries and technologies to create a functional Java application. Here's a comprehensive list of the main components used:

1. JavaFX:
   * The primary framework for creating the graphical user interface (GUI)
   * Used for building windows, scenes, and controls
   * Includes FXML for defining the UI layout
2. Java Database Connectivity (JDBC):
   * Used for connecting to and interacting with the PostgreSQL database
   * Handles database operations such as querying, inserting, updating, and deleting records
3. Java Standard Libraries:
   * java.sql: For database operations (Connection, PreparedStatement, ResultSet)
   * java.time: For handling dates and times (LocalDate, LocalDateTime, LocalTime)
   * java.util: For various utility classes (ResourceBundle)
4. JavaFX Controls:
   * Button, Label, TextField, PasswordField
   * TableView and TableColumn for displaying data in tabular format
   * MenuButton and MenuItem for dropdown menus
   * DatePicker for date selection
5. JavaFX Layout Components:
   * Parent, Scene, and Stage for structuring the application windows
6. JavaFX Properties:
   * SimpleStringProperty, SimpleIntegerProperty, SimpleBooleanProperty for data binding
7. JavaFX Collections:
   * ObservableList and FXCollections for managing dynamic lists of data
8. JavaFX Web Components:
   * WebView for displaying web content (used for showing Google Maps)
9. Java Desktop Integration:
   * java.awt.Desktop for opening web links in the default browser
10. JavaFX Concurrency:
    * Platform.runLater() for updating UI elements from non-JavaFX threads
11. Java IO:
    * For file operations (loading FXML files)
12. JavaFX Dialogs:
    * Alert for showing information and error messages to the user
13. Date and Time Formatting:
    * DateTimeFormatter for parsing and formatting date-time strings
14. JavaFX Initialization:
    * Initializable interface for controller initialization
15. **BCrypt**:

* Used for securely hashing and verifying passwords.
* Generates a salt automatically for each password hash.
* Uses key stretching to make hashing more computationally expensive.
* Functions:
  + BCrypt.hashpw() for hashing a password.
  + BCrypt.checkpw() for verifying if a password matches a hash

1. Custom Classes:
   * DataBaseConnection: A custom class for managing database connections
   * Appointments: A custom class for representing appointment data

**Github:** [**https://github.com/Iliutz12/barberShop.git**](https://github.com/Iliutz12/barberShop.git)