Scheduling App

Julia Ortheden, Johan Wennerbeck, Rasmus Lindgren

March 2019

1 Introduction

The web-application developed in this course is a booking application that students can use to book study rooms, music room or the physical education hall at Chalmers Union building. We chose to do this because the current system does not work very well and it seemed like a fun challenge to try to improve it. Here is a link to the project on Github: https://github.com/JuliaOrtheden/schema-webb-app.

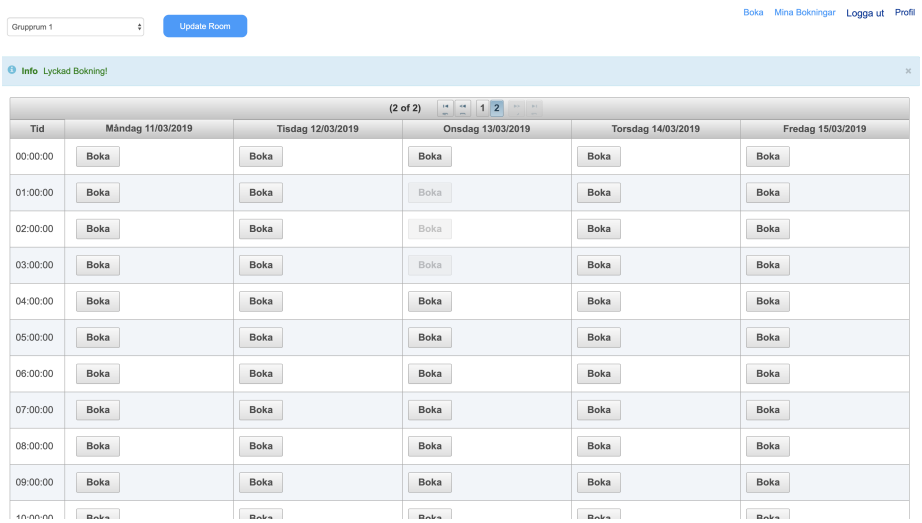

Figure 1: A screen shot of our main booking view

2 Use cases

- **Login**
  -Actor: User
  -Goal: Login to the system

-Description: By entering the correct information the user will be able to login to the system

- **Register**
  -Actor: User
  -Goal: Register a new User
  -Description: Fill out the information required and register an account

- **Log out**
  -Actor: User
  -Goal: Log out of the system
  -Description: By clicking the "Log out" button in the header a user logs out

- **Show already booked times in the schedule**
  -Actor: User
  -Goal: Be able to see which times are free to book
  -Description: The user can see booked times as disabled in the schedule

- **Filter timeslots by room** -Actor: User
  -Goal: Only show timeslots for a specific room
  -Description: By choosing a room from the dropdown a user can filter by that room to enable a booking

- **Book a time**
  -Actor: User
  -Goal: Be able to book a time in a room
  -Description: The user selects a room and a time and clicks the "book" button to book it

- **Show Bookings**
  -Actor: User
  -Goal: Be able to show the user's bookings
  -Description: The user clicks "My Bookings" and can see all the booked times with respective details in a table

- **Edit booking description**
  -Actor: User
  -Goal: Edit the description of a booking
  -Description: From "My bookings" a user can edit the description of a booking

- **Delete a booking**
  -Actor: User
  -Goal: Remove a booking
  -Description: Remove a booking from the users booking list

- **Edit user profile**
  -Actor: User
  -Goal: A user should be able to edit their profile
  -Description: By visiting the "Profile" a user can change the personal information

- **Edit a user** -Actor: Admin user -Goal: To edit a user profile. -Description: In admin view the admin can access a list of all users. In the list view the admin can edit any user.

- **Delete user** -Actor: Admin user -Goal: To delete a user. -Description: In admin view the admin can access a list of all users. In the list view the admin can delete any user.

- **Edit room** -Actor: Admin user -Goal: To edit a room. -Description: An admin user can click "Show all room" and from that table edit certain rooms.

- **Delete room** -Actor: Admin user -Goal: To delete a room -Description: An admin user can click "Show all rooms" and from that table delete certain rooms

- **Create room** -Actor: Admin user -Goal: To create a room -Description: An admin user can click "Show all rooms" and from that table click "create a new room" and from there fill in the necessary details

- **Admin can delete other users bookings** -Actor: Admin user -Goal: To delete bookings of other users -Description: An admin user can click "Show all timeslots" and from that table click "Unbook" on a chosen item to delete the booking

3 User manual

The first page is a login page, where you can choose to register as a new user or use your cid and password to log into an existing account. After you are logged in you are met by two altering views depending on if you have a student account or an admin account. From the student's point of view you are met by a welcome page with two options, either to book something new or to view your bookings. If you choose to book you are met by a schedule view, where you in the upper left corner filter by the room you want to book. You can see the already occupied times as disabled buttons and all the others are possible

to book by clicking the book button. Thereafter you are met by a modal with all the information and have to confirm that you want to book the time. If you confirm the modal closes and a success message appears. Otherwise the modal closes and nothing happens. Now it is possible to go to "My bookings" and view your booking, it is also there you can change the description or delete your booking.

In the application there is a header that is consistent on each page as soon as you are logged in. From the header you can log out or see you profile, it is also from the profile you are able to edit your personal information.

If you instead have admin access you will be met by more options on the welcome page. To see the booking schedule will still be a possibility, but to see all users, all rooms and all booked times is different from a regular user. Both the users, rooms and booked times will be viewed in a table. Where an admin user can add a new item, change an item or delete an item.

4 Design

The application is based on Java Server Faces(JSF) and managed beans, similiar to what has been taught in the course. When building the GUI two frameworks where choosen, PrimeFaces and BootsFaces. These frameworks were chosen since they are widely used and is easy to get help with but also since they were used in previous labs. The backend is created in Java EE, also this similar to what we did during the labs. For handling the login and logout of users we use HttpSessions, where we create a new session and sets a user parameter when logging in and invalidates a session when logging out.

The main architecture of the application is MVC, where the controller package handle the communication between the model and the view. The controllers also communicate with the facade classes that is our link to the database, which is a Java db created through NetBeans. The testing is done with the framework junit.

5 Application Flow

In the following section the application flow for three use cases will be described in more detail.

5.1 Register and log in

When the application starts the user sees the Login and Register page "Login.xhtml". The user clicks on "Registera" and is redirected to "Create.xhtml. Here the user fills in all necessary fields and then click on "Spara". Each value from the inputtext fields are saved temporarily in userController as current user. The method create in userController is called. The method starts to retrieve the UserFacade and calls the method create i UserFacade. Since the UserFacade does not have the method create it sends the request to its superclass

AbstractFacade. AbstractFacade creates the new user in the database. The userController then continues to automatically log in the user via userFacade where the login function controls if the user exists in the database. When the user is confirmed the userController sets the sessionsattribute user to the new user created and sends the user to the WelcomePage.xhtml.

If the user wants to log in directly it fills out the cid and password on the first login page. The call goes directly to UserFacade login in this case.

5.2 Book a timeslot

When the user is on the **Schema3.xhtml** page which shows all the possible timeslots available for booking, the user can choose to press the "Boka" button which will open a Dialog in Schema3.xhtml. In the Dialog the user will see the information that belongs to the selected slot and also a button that says "Boka", the information is retrieved using the **TimeslotController**. When the user clicks the button the method "bookTimeslot()" is called in the **TimeslotController**. The method then creates a new Timeslot using the method **convertSlotToTimeslot(selectedSlot)** with the selected slot as its parameter. It then sets the description using the selected slot as well, and it asigns a user to the timeslot by getting the current HTTPSession and getting the attribute "user". The TimeslotController then calls TimeslotFacade to add the timeslot to the database and it then recreates the week using recreateWeek() to update the the week to show that the timeslot has been booked.

5.3 Edit a user profile

6 Responsibilities

No member was exclusively responsible for a specific part of the project. But instead we spent nearly all of the time coding together, often pair coding, and helping each other. Everyone contributed equally even though some members got more code than others, a side effect of pair coding.

Figure 2: Rasmus Lindgren

Figure 3: Johan Wennerbeck

Figure 4: Julia Ortheden