# Online Diary Web Application

# 1. Introduction

The developed system is an Online Diary system, which allows users to share their contact details and schedule appointments with other users around the world. This report describes the development process of this system.

# 2. Design

After analyzing the requirements of the system, I have decided to divide the system into the following views:

* Account creation and login
  + The pages which will allow the user to create an account and log in to the system
* Home page
  + The page which will display the list of all the users, with a search functionality
* Appointments
  + The page which will display a list of all the appointments for the logged user
* Booking appointments
  + The page which will allow the user to book appointments with other users
* Profile
  + The page which will allow the user to modify their details

The application is going to use material design, as the styling and colour schemes are well known to the users. This will increase the usability of the system and bring modern design and look. The users will navigate through the application by a navigation bar at the top of the page.

All the error messages displayed to the user will be shown under the input which has caused an error with a red colour. This is to make sure that the user understands the error message and where did the error happen.

The key components of the application:

* Entity classes

1. The Person entity class will hold all the information specific to the person such as names, username, email etc. Additionally, it will contain the related address for this user using Address class, and its list of appointments using the Appointment class.
2. The Address entity class will hold all the address information such as street, city, country etc.
3. The Appointment entity class will hold all the appointment details, such as time, date, and guests.

* Controllers

1. PersonController will be responsible for the management with everything related to the Person entity class (Creating a person, login, displaying a person list).
2. AppointmentController will be responsible for the management of the appointments, used in appointments screens to show all appointments for the logged user, and all the methods for creating appointments.
3. NavigationController will be used to generate the links in the navigation bar, additionally different colour will be displayed on the currently active link.

### 2.1. User interface design

Below are low prototype user interface designs which will be used as a blueprint when it comes to implementation of the system.

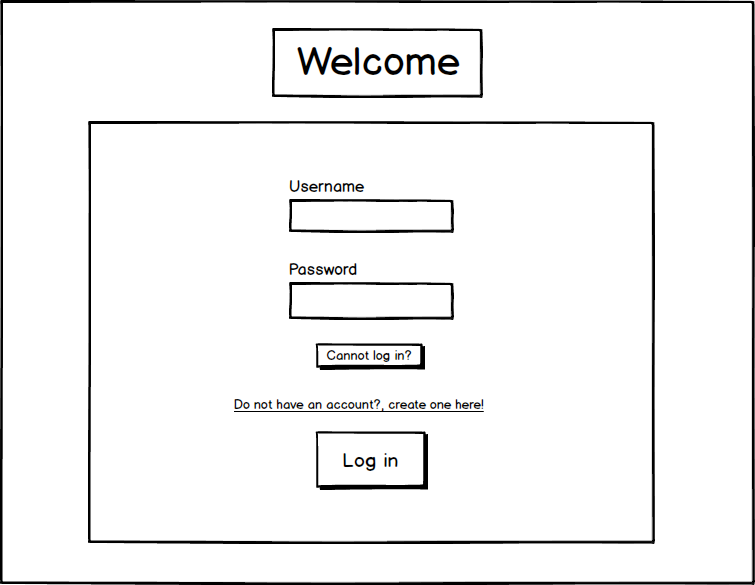


Figure 1. Login screen

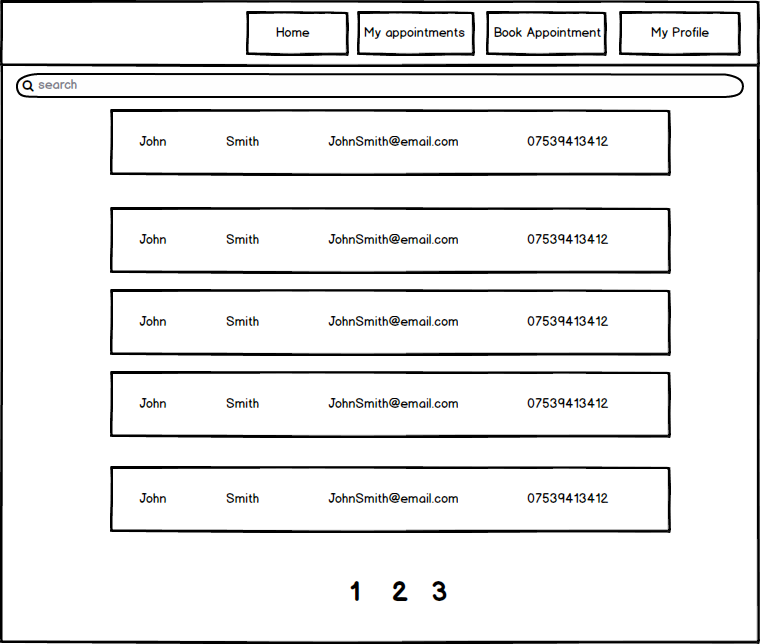


Figure 2. Home screen

### 2.2. The flow diagram of the system

For the user to use the system, he must create an account and log in. Then he will be redirected to the home page of the application. From there a list of all users will be paginated and displayed. After login, the user can navigate through the app using the navigation bar located at the top of the page. The available options will be Home, Appointments, Book Appointment, Profile and Sign out.

# 3.Implementation

## 3.1. Development tools

Java Enterprise Edition 7, Java Server Faces (JSF), Java DB, Glassfish 4.1 and NetBeans IDE were the development tools used to develop the system. I have decided to use NetBeans because it supports the development of applications with the frameworks specified above. It was not easy at the start to use the IDE, as I had no previous experience, however, the use of code generation mechanics, such as getters and setters, together with the wizards to create different classes for facades and beans made the development much easier. Additionally, Version control was used to manage the project, which allowed to easily keep the commit and merge history.

## 3.2. Development

Before starting the development of all the functional requirements, I have set up a correct project structure and creates all the necessary project files. Following the design, I have created all the entity classes, controllers, services and facades. At last, XHTML files were created for each of the screens mentioned in the design.

At first research was done to find out how can I import an external CSS or JavaScript files into the project, as simply linking the stylesheet in HTML did not work. Finally, a solution was found to include the files inside a resources folder in the web directory of the project. Even though the files were in the resources folder, the stylesheet still did not load properly. The problem was that using the standard HTML way of linking a stylesheet is wrong in JSF, and a separate component must be used to link styles and JavaScript files into the project. I soon have realized that a lot of code which was in the head or the navigation bar was copied over which caused duplicated code. Meaning that if I wanted to make a change in the navigation bar, I had to do it in several different places, which was not good. After performing small research, I have found out that JSF provides a way of creating templates for pieces of code which could be re-used.

The use of **ui: composition**, small templates were created which were used as a template for navigation and pagination.

After sorting out all the issues with importing external CSS and JavaScript MaterializeCSS library was used to style all the elements such as buttons, inputs, text and navigation bar. I have created a separate style sheet to position all elements in correct places.

Another problem which I have encountered was when I was developing the login system. Upon successful login to the system, the user had to be redirected to the home page of the application. The user in fact was getting redirected to the home screen but all the properties which were set in the PersonController, after the login were lost. Meaning that If I wanted to access any property of the currently logged user, I was getting a NullPointerException, as the currentPerson in the mentioned controller was null. The reason that this happened was because I have set the scope of the PersonController to RequestScoped, meaning that It only lasted if the HTTP request. Being redirected to the home page after login, recreated the request and discarded all the previous changes. To resolve this problem a SessionScoped annotation was used instead and it has worked because the information of the user was kept in the session storage on the server.

I wanted to use AJAX often in the pages as it allowed me to display new data on the page without the need of refreshing the whole page and making the application much for faster and easier to use. When using AJAX component in JSF, we need to pass the triggering event, the action and then what to re-render. When I first time attempted this, I have encountered problems where the page would completely re-render, therefore all the data kept in the fields would have been lost. This created issues as the user was required to pass all the information again. Initially I was passing the ID of the form into the re-render attribute of the AJAX component, and this was the reason why it has refreshed the whole form. To resolve this issue the ID of the form and the ID of the container could be used to just refresh this content on the page.

<f:ajax execute = "input-component-name" render = "MyForm:myContainer" />

Using the example above and passing the ID, separated by: and passing another ID, I was able to target the specific part of the page to be updated.

## 4. Testing

Each of the functional requirement was tested manually. After implementing a requirement, manual tests were performed to check if the requirement is met. Testing classes were not used; therefore, it is hard to prove that all the functionality works as expected. This is due to a time constraint.

## 5. Summary

I have successfully completed the application in a framework which I have not used before. I have followed the requirements and design specifications. I am happy with the outcome of the application. In terms of a design of the page it looks modern, if compared to any other web applications. With the use of AJAX in JSF, the application has no reloads and all the data is automatically displayed to the user. The application handles the validation well, the messages are shown under each input which has not passed the validation, helping the user to correct the errors. Throughout the development I was encountering many errors when developing the software, and I would like to credit the debug functionality of the NetBeans IDE, as at first I have wasted a lot of time trying to print the values out in the console to find the error, but placing a breakpoint allowed me to slowly move through the code line by line, therefore I was able to see the point of error in the code. On top of that all the variables were inspectable, allowing me to easily see if the values of the them were correct.

I think it was a bad decision for me to use external CSS library which had a high use of JavaScript. The JSF components make the development of the application much easier, as we just use them, and they automatically work very well with the server. However, using a separate JavaScript component such as date-picker which is not linked with any JSF component make the code little messy. It would have been much better for me to stick to a framework such as PrimeFaces and use their components. Another bad decision was that I held the time and date for the appointment in the same field. I should have held a separate field for a date and time, because it was very difficult to create queries where I only wanted to query the time or vice-versa.

I have really enjoyed the way of developing web software through this framework as everything links very well through this architecture. As I have used many other frameworks to develop web applications, they often get quite messy and complicated due to not having any architecture defined. The development of web applications in MVC architecture has been enjoyable, as every component does something different, it is very easy to create extra functionality and it is very easy to read and follow the code, meaning it is easier to spot any errors in the project.

# References

Briggs, J. (2019). Step by step JPA/JSF webapp. Retrieved from http://briggs.myweb.port.ac.uk/WEBP/notes/db/jpasteps.html

Getting Started - Materialize. (2019). Retrieved from <https://materializecss.com/getting-started.html>

JSF Managed Beans. (2019). Retrieved from <https://www.tutorialspoint.com/jsf/jsf_managed_beans.htm>

JSF UI Components - javatpoint. (2019). Retrieved from <https://www.javatpoint.com/jsf-ui-components>

Lars Vogel, 2. (2019). JSF (JavaServer Faces) - Tutorial. Retrieved from <https://www.vogella.com/tutorials/JavaServerFaces/article.html>