VAD socializing app

Mihali Vlad

January 2021

1 Team

- Mihali Vlad
- Sauchea Alexandra Teodora
- Stioiu Denis Adrian

2 Abstract

The project is a further development of the miniproject. We developed a socializing application named VAD, in which every user is the main actor from his/her point of view.
We used the following technologies:

- Java
- Spring
- Vaadin
- Hibernate
- Jpa
- Lombok

At the beginning of the implementation we set a couple of actions that person can perform in the application, that are shown in the figure 1, use case diagram.

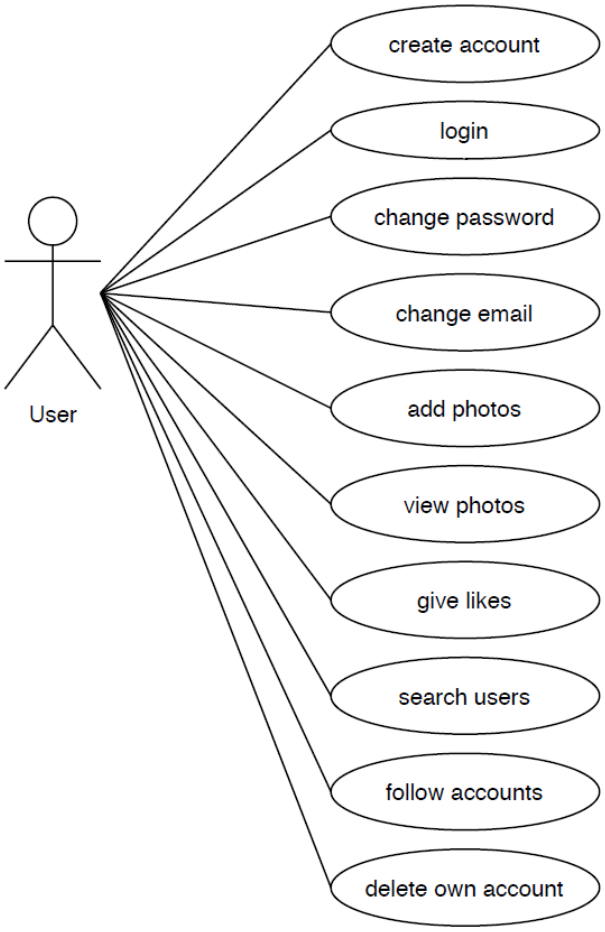

Figure 1: Use Case diagram

All the proposed use cases were developed successfully.
Bellow is presented the deployed diagram.
The class diagrams, both for backend and frontend are presented after the code section for page layout reasons.

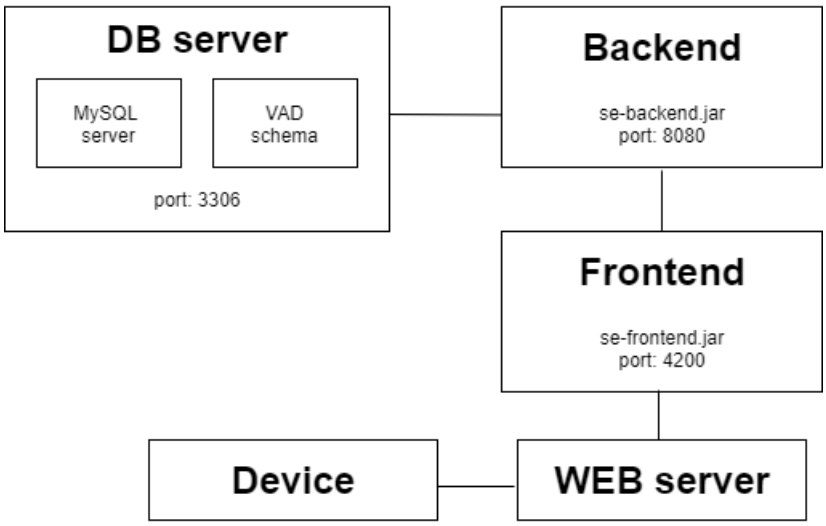

Figure 2: Deployed diagram

3 Code

3.1 Frontend

PhotoApi

```
1  package com.example.application.callApi;
2
3  import com.example.application.data.PhotoModel;
4  import com.example.application.data.entity.RequestPhoto;
5  import com.example.application.data.entity.User;
6  import com.example.application.views.main.MainView;
7  import org.springframework.beans.factory.annotation.Autowired;
8  import org.springframework.http.*;
9  import org.springframework.http.converter.
       ByteArrayHttpMessageConverter;
10 import org.springframework.http.converter.json.
       MappingJackson2HttpMessageConverter;
11 import org.springframework.web.bind.annotation.RequestBody;
12 import org.springframework.web.client.RestTemplate;
13
14 import java.io.File;
15 import java.io.IOException;
16 import java.nio.file.Files;
17 import java.util.Arrays;
18 import java.util.List;
19
20 public class PhotoApi {
21     private RestTemplate restTemplate = new RestTemplate();
22
23     public List<PhotoModel> getAll(){
```

```
24          PhotoModel[] photoModelsArray = restTemplate.getForObject("
    http://localhost:8080/photo/", PhotoModel[].class);
25          return Arrays.asList(photoModelsArray);
26      }
27
28      public List<PhotoModel> getAllByUsername(){
29          PhotoModel[] photoModelsArray = restTemplate.getForObject("
    http://localhost:8080/photo/user/?username="+MainView.
    authResponse.getUserName(), PhotoModel[].class);
30          return Arrays.asList(photoModelsArray);
31      }
32
33      public List<PhotoModel> getAllByUsername(String username){
34          PhotoModel[] photoModelsArray = restTemplate.getForObject("
    http://localhost:8080/photo/user/?username="+username,
    PhotoModel[].class);
35          return Arrays.asList(photoModelsArray);
36      }
37
38
39
40      public PhotoModel addPhoto(byte[] imageData, String description
    ) throws IOException {
41          RequestPhoto requestPhoto = new RequestPhoto();
42          requestPhoto.setImage(imageData);
43          requestPhoto.setUsername(MainView.authResponse.getUserName
    ());
44          requestPhoto.setDescription(description);
45          return restTemplate.postForObject("http://localhost:8080/
    photo/post",requestPhoto, PhotoModel.class);
46      }
47
48      public byte[] getImage(Long id){
49          restTemplate.getMessageConverters().add(new
    ByteArrayHttpMessageConverter());
50          HttpHeaders headers = new HttpHeaders();
51          headers.setAccept(Arrays.asList(MediaType.
    APPLICATION_OCTET_STREAM));
52          HttpEntity<String> entity = new HttpEntity<String>(headers)
    ;
53          return restTemplate.getForObject("http://localhost:8080/
    photo/get/?id="+id,byte[].class);
54      }
55
56      public <T> T getLast(List<T> list) {
57          return list != null && !list.isEmpty() ? list.get(list.size
    () - 1) : null;
58      }
59
60      public PhotoModel updateLike(PhotoModel photoModel){
61          HttpEntity<Long> httpEntity = new HttpEntity(photoModel.
    getId());
62          return restTemplate.exchange("http://localhost:8080/photo/
    like?username="+MainView.authResponse.getUserName(),HttpMethod.
    PUT, httpEntity, PhotoModel.class).getBody();
63      }
64
```

```
65      public PhotoModel updateDislike(PhotoModel photoModel){
66          HttpEntity<Long> httpEntity = new HttpEntity(photoModel.
        getId());
67          return restTemplate.exchange("http://localhost:8080/photo/
        dislike?username="+MainView.authResponse.getUserName(),
        HttpMethod.PUT, httpEntity, PhotoModel.class).getBody();
68      }
69
70      public void updateLikeDislike(PhotoModel photoModel){
71          HttpEntity<Long> httpEntity = new HttpEntity(photoModel.
        getId());
72          restTemplate.exchange("http://localhost:8080/photo/
        likedislike?username="+MainView.authResponse.getUserName(),
        HttpMethod.PUT, httpEntity, Void.class);
73      }
74
75      public boolean isLike(PhotoModel photoModel){
76          return restTemplate.postForObject("http://localhost:8080/
        photo/like?username="+MainView.authResponse.getUserName(),
        photoModel.getId(), Boolean.class);
77      }
78
79      public boolean isDislike(PhotoModel photoModel){
80          HttpEntity<PhotoModel> httpEntity = new HttpEntity(
        photoModel);
81          return restTemplate.postForObject("http://localhost:8080/
        photo/dislike?username="+MainView.authResponse.getUserName(),
        photoModel.getId(), Boolean.class);
82      }
83
84      public void getBy(PhotoModel photoModel){
85          PhotoModel aux = restTemplate.getForObject("http://
        localhost:8080/photo/getBy?id="+photoModel.getId(), PhotoModel.
        class);
86          photoModel.setPicture(aux.getPicture());
87          photoModel.setDescription(aux.getDescription());
88      }
89
90      public void setLikes(PhotoModel photoModel){
91          photoModel.setLikes(restTemplate.getForObject("http://
        localhost:8080/photo/like?id="+photoModel.getId(),Integer.class
        ));
92      }
93
94      public void setDislikes(PhotoModel photoModel){
95          photoModel.setDislikes(restTemplate.getForObject("http://
        localhost:8080/photo/dislike/?id="+photoModel.getId(),Integer.
        class));
96      }
97
98      public void getUsername(PhotoModel photoModel){
99          String username = restTemplate.getForObject("http://
        localhost:8080/photo/username?id="+photoModel.getId(),String.
        class);
100         User user = new User();
101         user.setUserName(username);
102         photoModel.setUser(user);
```

```
103     }
104
105 }
```

RecommendationApi

```
1  package com.example.application.callApi;
2
3  import com.example.application.data.PhotoModel;
4  import com.example.application.views.main.MainView;
5  import com.vaadin.flow.component.UI;
6  import org.springframework.web.client.HttpServerErrorException;
7  import org.springframework.web.client.RestTemplate;
8
9  import java.util.Arrays;
10 import java.util.List;
11
12 public class RecommendationApi {
13
14     private RestTemplate restTemplate = new RestTemplate();
15
16     public List<PhotoModel> getRecom(){
17         PhotoModel[] photoModelsArray = restTemplate.getForObject("
       http://localhost:8080/recom/?username="+ MainView.authResponse.
       getUserName(), PhotoModel[].class);
18         return Arrays.asList(photoModelsArray);
19     }
20
21     public Integer getIndex(){
22         return restTemplate.getForObject("http://localhost:8080/
       recom/index?username="+ MainView.authResponse.getUserName(),
       Integer.class);
23     }
24
25     public Integer setIndex(Integer value){
26         return restTemplate.postForObject("http://localhost:8080/
       recom/index?username="+ MainView.authResponse.getUserName(),
       value, Integer.class);
27     }
28
29     public List<String> getFollow(){
30         return Arrays.asList(restTemplate.getForObject("http://
       localhost:8080/recom/follow?username="+MainView.authResponse.
       getUserName(),String[].class));
31     }
32
33     public List<String> setFollow(String username){
34         return Arrays.asList(restTemplate.postForObject("http://
       localhost:8080/recom/follow/?username="+MainView.authResponse.
       getUserName(),username,String[].class));
35     }
36
37     public Boolean isFollow(String username) {
38         try {
39             return restTemplate.getForObject("http://localhost
       :8080/recom/follow/" + MainView.authResponse.getUserName() +
       "/" + username, Boolean.class);
40
```

```
        } catch (HttpServerErrorException ex) {
            UI.getCurrent().navigate("account");
            return null;
        }
    }

}
```

RefreshTokenData

```
package com.example.application.callApi;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.validation.constraints.NotBlank;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class RefreshTokenData {
    @NotBlank
    private String refreshToken;
    private String userName;
}
```

UserApi

```
package com.example.application.callApi;

import com.example.application.data.AuthResponse;
import com.example.application.data.LoginModel;
import com.example.application.data.entity.User;
import com.example.application.views.account.UserUpdateDetails;
import com.example.application.views.main.MainView;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.web.client.HttpClientErrorException;
import org.springframework.web.client.HttpServerErrorException;
import org.springframework.web.client.RestTemplate;

import java.util.Arrays;
import java.util.List;

public class UserApi {
    private RestTemplate restTemplate;

    @Autowired
    public UserApi(RestTemplate restTemplate) {
        this.restTemplate = restTemplate;
    }

    public static void callServiceSignup(User user) {
        RestTemplate restTemplate = new RestTemplate();
        System.out.println(restTemplate.postForObject("http://
    localhost:8080/api/auth/signup", user, String.class));
```

```
30      }
31
32      public static AuthResponse callServiceLogin(LoginModel
        loginModel) throws HttpClientErrorException {
33          RestTemplate restTemplate = new RestTemplate();
34          AuthResponse authResponse = restTemplate.postForObject("
        http://localhost:8080/api/auth/login", loginModel, AuthResponse
        .class);
35          return authResponse;
36      }
37
38      public static String callServiceUpdate(UserUpdateDetails
        userUpdateDetails) {
39          RestTemplate restTemplate = new RestTemplate();
40          HttpHeaders headers = new HttpHeaders();
41          headers.setBearerAuth(MainView.authResponse.
        getAuthenticationToken());
42          try {
43              return restTemplate.postForObject("http://localhost
        :8080/api/account/update", new HttpEntity<>(userUpdateDetails,
        headers), String.class);
44          } catch (HttpClientErrorException ex) {
45              if (ex.getStatusCode() == HttpStatus.BAD_REQUEST) {
46                  return "Invalid password";
47              }
48          } catch (HttpServerErrorException ex) {
49              return "Internal Server Error";
50          }
51          return "Unknown error!";
52      }
53
54      public static String callServiceDelete(String userName) {
55          RestTemplate restTemplate = new RestTemplate();
56          HttpHeaders headers = new HttpHeaders();
57          headers.setBearerAuth(MainView.authResponse.
        getAuthenticationToken());
58          try {
59              restTemplate.delete("http://localhost:8080/api/account/
        delete/" + userName, new HttpEntity<>("", headers), String.
        class);
60              return "Account deleted!";
61          } catch (HttpClientErrorException ex) {
62              if (ex.getStatusCode() == HttpStatus.BAD_REQUEST) {
63                  return "Something went wrong!";
64              }
65          } catch (HttpServerErrorException ex) {
66              return "Internal Server Error";
67          }
68          return "Unknown error!";
69      }
70
71      public static void refreshToken() throws
        HttpClientErrorException {
72          RefreshTokenData refreshTokenData = new RefreshTokenData();
73          refreshTokenData.setRefreshToken(MainView.authResponse.
        getRefreshToken());
74          refreshTokenData.setUserName(MainView.authResponse.
```

```
    getUserName());
75          RestTemplate restTemplate = new RestTemplate();
76          AuthResponse authResponse = restTemplate.postForObject("
    http://localhost:8080/api/auth/refresh/token", refreshTokenData
    , AuthResponse.class);
77          MainView.authResponse.setAuthenticationToken(authResponse.
    getAuthenticationToken());
78      }
79
80      public static List<String> getAllUsers(){
81          RestTemplate restTemplate = new RestTemplate();
82          return Arrays.asList(restTemplate.getForObject("http://
    localhost:8080/api/account/usernames", String[].class));
83
84      }
85  }
```

AuthResponse

```
1  package com.example.application.data;
2
3  import lombok.AllArgsConstructor;
4  import lombok.Builder;
5  import lombok.Data;
6  import lombok.NoArgsConstructor;
7
8  import java.time.Instant;
9
10 @Data
11 @AllArgsConstructor
12 @NoArgsConstructor
13 @Builder
14 public class AuthResponse {
15     private String authenticationToken;
16     private String refreshToken;
17     private Instant expiresAt;
18     private String userName;
19     private String email;
20 }
```

LoginModel

```
1  package com.example.application.data;
2
3  import lombok.AllArgsConstructor;
4  import lombok.Data;
5  import lombok.NoArgsConstructor;
6
7  @Data
8  @AllArgsConstructor
9  @NoArgsConstructor
10 public class LoginModel {
11
12     private String userName;
13     private String password;
14 }
```

PhotoModel

```java
package com.example.application.data;

import com.example.application.data.entity.User;
import lombok.*;

import javax.persistence.*;

import java.sql.Blob;

import static javax.persistence.GenerationType.IDENTITY;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Setter
@Getter
public class PhotoModel {
    private Long id;
    private User user;
    private byte[] picture;
    private Integer likes;
    private Integer dislikes;
    private String description;

}
```

RequestPhoto

```java
package com.example.application.data.entity;

import lombok.*;

import java.sql.Blob;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Setter
@Getter
public class RequestPhoto {
    private String username;
    private byte[] image;
    private String description;
}
```

User

```java
package com.example.application.data.entity;

import lombok.*;
import org.atmosphere.config.service.Get;

import java.io.Serializable;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Getter
@Setter
```

```
public class User implements Serializable {
  private String email;
  private String password;
  private String userName;
}
```

AboutView

```
package com.example.application.views.about;

import com.vaadin.flow.component.dependency.CssImport;
import com.vaadin.flow.component.html.Div;
import com.vaadin.flow.component.html.Label;
import com.vaadin.flow.router.PageTitle;
import com.vaadin.flow.router.Route;
import com.example.application.views.main.MainView;

@Route(value = "", layout = MainView.class)
@PageTitle("About")
@CssImport("./styles/views/about/about-view.css")
public class AboutView extends Div {

    public AboutView() {
        setId("about-view");
        add(new Label("Content placeholder"));
    }

}
```

AccountView

```
package com.example.application.views.account;

import com.example.application.callApi.UserApi;
import com.example.application.data.AuthResponse;
import com.example.application.data.entity.User;
import com.example.application.views.main.MainView;
import com.vaadin.flow.component.UI;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.combobox.ComboBox;
import com.vaadin.flow.component.dependency.CssImport;
import com.vaadin.flow.component.dependency.JsModule;
import com.vaadin.flow.component.html.Div;
import com.vaadin.flow.component.html.Label;
import com.vaadin.flow.component.notification.Notification;
import com.vaadin.flow.component.orderedlayout.FlexComponent;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.component.textfield.Autocomplete;
import com.vaadin.flow.component.textfield.EmailField;
import com.vaadin.flow.component.textfield.PasswordField;
import com.vaadin.flow.component.textfield.TextField;
import com.vaadin.flow.data.binder.Binder;
import com.vaadin.flow.router.PageTitle;
import com.vaadin.flow.router.Route;
import org.springframework.web.client.HttpServerErrorException;

import java.util.List;
import java.util.Objects;
import java.util.stream.Collectors;
```

```
29
30 import java.time.Instant;
31
32 @Route(value = "account", layout = MainView.class)
33 @PageTitle("Account")
34 @CssImport(value = "./styles/views/myprofile/myprofile-view.css",
       include = "lumo-badge")
35 @JsModule("@vaadin/vaadin-lumo-styles/badge.js")
36 public class AccountView extends Div {
37
38     private EmailField email = new EmailField("Email address");
39     private PasswordField oldPassword = new PasswordField("Old
       Password");
40     private PasswordField newPassword = new PasswordField("New
       Password");
41     private PasswordField verifyPassword = new PasswordField("
       Verify New Password");
42
43     private Button delete = new Button("Delete account");
44
45     private ComboBox<String> search = new ComboBox<>("User");
46     private Button searchButton = new Button("search");
47
48
49     private Button save = new Button("Save changes");
50     private Binder<User> binder = new Binder(User.class);
51
52     public AccountView() {
53         if (MainView.authResponse.getUserName().equals("")) {
54             add(new Label("You are not login!"));
55         } else {
56
57
58             List<String> usernames = UserApi.getAllUsers().stream()
       .filter(Objects::nonNull).collect(Collectors.toList());
59             usernames.remove("");
60             search.setItems(usernames);
61             add(search);
62             add(searchButton);
63             add(new Label("\n\n\n"));
64             searchButton.addClickListener(e->{
65                 try {
66                     UI.getCurrent().navigate("username/" + search.
       getValue());
67                 }catch (HttpServerErrorException ex){
68                     UI.getCurrent().navigate("account");
69                     Notification.show("username invalid", 2000,
       Notification.Position.MIDDLE);
70                 }
71             });
72
73
74             VerticalLayout fl = new VerticalLayout();
75             VerticalLayout flD = new VerticalLayout();
76             VerticalLayout flS = new VerticalLayout();
77             flD.add(delete);
78             fl.add(email);
```

```
79              fl.add(oldPassword);
80              fl.add(newPassword);
81              fl.add(verifyPassword);
82              fl.add(save);
83              flD.setAlignItems(FlexComponent.Alignment.END);
84              fl.setAlignItems(FlexComponent.Alignment.CENTER);
85              add(flD);
86              add(fl);
87
88              email.setValue(MainView.authResponse.getEmail());
89
90              setId("account-view");
91              addClassName("account-view");
92
93              delete.addClickListener(e -> {
94
95                  String response = UserApi.callServiceDelete(
    MainView.authResponse.getUserName());
96                  if (!response.isEmpty()) {
97                      Notification.show(response, 1000, Notification.
    Position.MIDDLE);
98                  }
99                  MainView.authResponse = new AuthResponse("","",
    Instant.MIN, "", "");
100                 UI.getCurrent().getPage().setLocation("http://
    localhost:4200/sign-up");
101             });
102
103             save.addClickListener(e -> {
104                 if (oldPassword.isInvalid() || email.isInvalid() ||
105                         oldPassword.getValue().equals("") || email.
    getValue().equals("")) {
106                     Notification.show("Password or email field
    empty!", 1000, Notification.Position.MIDDLE);
107                 } else {
108                     if (newPassword.getValue().equals("")) {
109                         UserUpdateDetails userUpdate = new
    UserUpdateDetails();
110                         userUpdate.setEmail(email.getValue());
111                         userUpdate.setUserName(MainView.
    authResponse.getUserName());
112                         String response = UserApi.callServiceUpdate
    (userUpdate);
113                         if (!response.isEmpty()) {
114                             Notification.show(response, 1000,
    Notification.Position.MIDDLE);
115                         }
116                         Notification.show("Changes saved!", 1000,
    Notification.Position.MIDDLE);
117                     } else {
118                         if (newPassword.isInvalid() ||
    verifyPassword.isInvalid() || !newPassword.getValue().equals(
    verifyPassword.getValue())) {
119                             Notification.show("Passwords do not
    match or are invalid!", 1000, Notification.Position.MIDDLE);
120                         } else {
121                             UserUpdateDetails userUpdate = new
```

```
    UserUpdateDetails();
                                userUpdate.setEmail(email.getValue());
                                userUpdate.setNewPassword(newPassword.
    getValue());
                                userUpdate.setOldPassword(oldPassword.
    getValue());
                                userUpdate.setUserName(MainView.
    authResponse.getUserName());
                                String response = UserApi.
    callServiceUpdate(userUpdate);
                                if (!response.isEmpty()) {
                                    Notification.show(response, 1000,
    Notification.Position.MIDDLE);
                                }
                                Notification.show("Changes saved!",
    1000, Notification.Position.MIDDLE);
                            }
                        }
                    }
                });
        }
    }
}
```

UserUpdateDetails

```
package com.example.application.views.account;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor

public class UserUpdateDetails {
    private String userName;
    private String email;
    private String oldPassword;
    private String newPassword;
}
```

FriendFeed

```
package com.example.application.views.friendFeed;

import java.io.ByteArrayInputStream;
import java.util.*;
import java.util.List;

import com.example.application.callApi.PhotoApi;
import com.example.application.callApi.RecommendationApi;
import com.example.application.data.PhotoModel;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.dependency.CssImport;
import com.vaadin.flow.component.dependency.JsModule;
import com.vaadin.flow.component.grid.Grid;
import com.vaadin.flow.component.grid.GridVariant;
```

```
15 import com.vaadin.flow.component.html.*;
16 import com.vaadin.flow.component.html.Image;
17 import com.vaadin.flow.component.html.Label;
18 import com.vaadin.flow.component.icon.Icon;
19 import com.vaadin.flow.component.icon.VaadinIcon;
20 import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
21 import com.vaadin.flow.component.orderedlayout.VerticalLayout;
22 import com.vaadin.flow.router.PageTitle;
23 import com.vaadin.flow.router.Route;
24 import com.example.application.views.main.MainView;
25 import com.vaadin.flow.server.StreamResource;
26
27 @Route(value = "friend-feed", layout = MainView.class)
28 @PageTitle("Friend Feed")
29 @CssImport(value = "./styles/views/photofeed/photo-feed-view.css",
       include = "lumo-badge")
30 @JsModule("@vaadin/vaadin-lumo-styles/badge.js")
31 public class FriendFeed extends Div {
32
33     Grid<PhotoModel> grid = new Grid<>();
34     private int index;
35     private List<PhotoModel> photoModelList = new ArrayList<>();
36     private Button next = new Button(">");
37     private Button prev = new Button("<");
38     private PhotoApi photoApi = new PhotoApi();
39     private RecommendationApi recommendationApi = new
       RecommendationApi();
40     private HorizontalLayout card =new HorizontalLayout();
41     private PhotoModel photoModel;
42
43     public FriendFeed() {
44         if(MainView.authResponse.getUserName().equals("")){
45             add(new Label("You are not login"));
46         }else{
47             index = 0;
48             List<String> friends = recommendationApi.getFollow();
49             for (String username:friends) {
50                 photoModelList.addAll(photoApi.getAllByUsername(
       username));
51             }
52             try {
53                 photoModelList.sort(Comparator.comparingLong(
       PhotoModel::getId).reversed());
54                 for (PhotoModel photoModel : photoModelList) {
55                     //photoModel.setUser(new User(photoApi.
       getUsername(photoModel)));
56                     photoApi.getUsername(photoModel);
57                 }
58                 setId("photo-feed-view");
59                 addClassName("photo-feed-view");
60                 setSizeFull();
61                 grid.setHeight("100%");
62                 grid.addThemeVariants(GridVariant.LUMO_NO_BORDER,
       GridVariant.LUMO_NO_ROW_BORDERS);
63                 photoModel = photoModelList.get(index);
64                 photoModel.setPicture(photoApi.getImage(photoModel.
       getId()));
```

```
65                  createCard(photoModel);
66                  add(prev);
67                  add(card);
68                  add(next);
69                  prev.addClickListener(e->{
70                      next.setVisible(true);
71                      try {
72                          photoModel = photoModelList.get(index-1);
73                          photoModel.setPicture(photoApi.getImage(
    photoModel.getId()));
74                          createCard(photoModel);
75                          index--;
76                      }catch (IndexOutOfBoundsException ex){
77                          prev.setVisible(false);
78                      }
79                  });
80                  next.addClickListener(e->{
81                      prev.setVisible(true);
82                      try {
83                          photoModel = photoModelList.get(index+1);
84                          photoModel.setPicture(photoApi.getImage(
    photoModel.getId()));
85                          createCard(photoModel);
86                          index++;
87                      }catch (IndexOutOfBoundsException ex){
88                          next.setVisible(false);
89                          //Notification.show("!", 2000, Notification
    .Position.MIDDLE);
90                      }
91                  });
92              }catch (IndexOutOfBoundsException ex){
93                  add(new Label("You have no friends"));
94              }
95
96          }
97
98      }
99
100     private void createCard(PhotoModel photoModel) {
101         photoApi.setDislikes(photoModel);
102         photoApi.setLikes(photoModel);
103         card.addClassName("card");
104         card.setSpacing(false);
105         card.getThemeList().add("spacing-s");
106
107         StreamResource resource = new StreamResource("
    dummyImageName.jpg", () -> new ByteArrayInputStream(photoModel.
    getPicture()));
108         Image image = new Image(resource, "dummy image");
109         VerticalLayout description = new VerticalLayout();
110         description.addClassName("description");
111         description.setSpacing(false);
112         description.setPadding(false);
113
114         HorizontalLayout header = new HorizontalLayout();
115         header.addClassName("header");
116         header.setSpacing(false);
```

```
117         header.getThemeList().add("spacing-s");
118
119         Anchor name = new Anchor("http://localhost:4200/username/"+
    photoModel.getUser().getUserName()
120                 ,photoModel.getUser().getUserName());
121         name.addClassName("name");
122         header.add(name);
123
124         Span post = new Span(photoModel.getDescription());
125         post.addClassName("post");
126
127         HorizontalLayout actions = new HorizontalLayout();
128         actions.addClassName("actions");
129         actions.setSpacing(false);
130         actions.getThemeList().add("spacing-s");
131         Icon likeIcon = new Icon(VaadinIcon.THUMBS_UP);
132         if(photoApi.isLike(photoModel)){
133             likeIcon.setColor("green");
134         }else {
135             likeIcon.setColor("black");
136         }
137         Span likes = new Span(String.valueOf(photoModel.getLikes())
    );
138         likes.addClassName("likes");
139         Icon dislikeIcon = new Icon(VaadinIcon.THUMBS_DOWN);
140         if(photoApi.isDislike(photoModel)){
141             dislikeIcon.setColor("red");
142         }else {
143             dislikeIcon.setColor("black");
144         }
145         Span dislikes = new Span(String.valueOf(photoModel.
    getDislikes()));
146         dislikes.addClassName("dislikes");
147         actions.add(likeIcon, likes, dislikeIcon, dislikes);
148         likeIcon.addClickListener(e->{
149             if(likeIcon.getColor().equals("green")){
150                 likeIcon.setColor("black");
151                 photoModel.setLikes(photoModel.getLikes()-1);
152                 likes.setText(String.valueOf(photoModel.getLikes())
    );
153                 photoApi.updateLike(photoModel);
154             }else {
155                 likeIcon.setColor("green");
156                 photoModel.setLikes(photoModel.getLikes()+1);
157                 likes.setText(String.valueOf(photoModel.getLikes())
    );
158                 if(dislikeIcon.getColor().equals("red")) {
159                     photoModel.setDislikes(photoModel.getDislikes()
    -1);
160                     dislikes.setText(String.valueOf(photoModel.
    getDislikes()));
161                     dislikeIcon.setColor("black");
162                     photoApi.updateLikeDislike(photoModel);
163                 }else{
164                     photoApi.updateLike(photoModel);
165                 }
166             }
```

```
167         });
168
169         dislikeIcon.addClickListener(e->{
170             if(dislikeIcon.getColor().equals("red")){
171                 dislikeIcon.setColor("black");
172                 photoModel.setDislikes(photoModel.getDislikes()-1);
173                 dislikes.setText(String.valueOf(photoModel.
    getDislikes()));
174                 photoApi.updateDislike(photoModel);
175             }else {
176                 dislikeIcon.setColor("red");
177                 photoModel.setDislikes(photoModel.getDislikes()+1);
178                 dislikes.setText(String.valueOf(photoModel.
    getDislikes()));
179                 if(likeIcon.getColor().equals("green")) {
180                     photoModel.setLikes(photoModel.getLikes()-1);
181                     likes.setText(String.valueOf(photoModel.
    getLikes()));
182                     likeIcon.setColor("black");
183                     photoApi.updateLikeDislike(photoModel);
184                 }else{
185                     photoApi.updateDislike(photoModel);
186                 }
187             }
188         });
189
190         description.add(header, post, actions);
191         card.removeAll();
192         card.add(image, description);
193     }
194
195
196 }
```

Login

```
1  package com.example.application.views.login;
2
3  import com.example.application.callApi.RecommendationApi;
4  import com.example.application.callApi.UserApi;
5  import com.example.application.data.LoginModel;
6  import com.example.application.views.main.MainView;
7  import com.example.application.views.photofeed.PhotoFeedView;
8  import com.vaadin.flow.component.UI;
9  import com.vaadin.flow.component.dependency.CssImport;
10 import com.vaadin.flow.component.html.Anchor;
11 import com.vaadin.flow.component.html.Div;
12 import com.vaadin.flow.component.login.LoginForm;
13 import com.vaadin.flow.component.notification.Notification;
14 import com.vaadin.flow.component.notification.Notification.Position
       ;
15 import com.vaadin.flow.component.orderedlayout.FlexComponent;
16 import com.vaadin.flow.component.orderedlayout.VerticalLayout;
17 import com.vaadin.flow.router.PageTitle;
18 import com.vaadin.flow.router.Route;
19 import lombok.extern.slf4j.Slf4j;
20 import org.springframework.web.client.HttpClientErrorException;
21
```

```
@Slf4j
@Route(value = "login", layout = MainView.class)
@PageTitle("Login")
@CssImport("./styles/views/signupform/sign-up-view.css")
public class Login extends Div {

    public static final String ROUTE = "login";

    private LoginForm login = new LoginForm(); //
    private Anchor anchor = new Anchor("http://localhost:4200/sign-up", "Sign Up");

    public Login() {
        login.setForgotPasswordButtonVisible(false);
        VerticalLayout fl = new VerticalLayout();
        fl.add(login);
        fl.add(anchor);
        fl.setAlignItems(FlexComponent.Alignment.CENTER);
        add(fl); //

        login.addLoginListener(e -> {
            try {
                MainView.authResponse = UserApi.callServiceLogin(new LoginModel(e.getUsername(), e.getPassword()));
                UI.getCurrent().getPage().setLocation("http://localhost:4200/photo-feed");
                RecommendationApi recommendationApi = new RecommendationApi();
                PhotoFeedView.setIndex(recommendationApi.getIndex());
                PhotoFeedView.setMaxIndex(PhotoFeedView.getIndex());
            } catch (HttpClientErrorException ex) {
                if (ex.getRawStatusCode() == 403) {
                    Notification.show("username or password invalid", 2000, Position.MIDDLE);
                    login.setError(true);
                }
            }

        });
    }

}
```

MainView

```
package com.example.application.views.main;

import java.time.Instant;
import java.util.Optional;

import com.example.application.callApi.RecommendationApi;
import com.example.application.callApi.UserApi;
import com.example.application.data.AuthResponse;
import com.example.application.views.account.AccountView;
import com.example.application.views.friendFeed.FriendFeed;
import com.example.application.views.login.Login;
```

```
12 import com.example.application.views.myprofile.MyprofileView;
13 import com.example.application.views.photofeed.PhotoFeedView;
14 import com.vaadin.flow.component.Component;
15 import com.vaadin.flow.component.ComponentUtil;
16 import com.vaadin.flow.component.UI;
17 import com.vaadin.flow.component.applayout.AppLayout;
18 import com.vaadin.flow.component.button.Button;
19 import com.vaadin.flow.component.dependency.CssImport;
20 import com.vaadin.flow.component.dependency.JsModule;
21 import com.vaadin.flow.component.html.Anchor;
22 import com.vaadin.flow.component.html.H1;
23 import com.vaadin.flow.component.html.Image;
24 import com.vaadin.flow.component.html.Label;
25 import com.vaadin.flow.component.orderedlayout.FlexComponent;
26 import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
27 import com.vaadin.flow.component.orderedlayout.VerticalLayout;
28 import com.vaadin.flow.component.tabs.Tab;
29 import com.vaadin.flow.component.tabs.TabVariant;
30 import com.vaadin.flow.component.tabs.Tabs;
31 import com.vaadin.flow.router.RouterLink;
32 import com.vaadin.flow.server.PWA;
33 import com.example.application.views.about.AboutView;
34 import com.example.application.views.personform.PersonFormView;
35 import lombok.Getter;
36 import lombok.Setter;
37
38 /**
39  * The main view is a top-level placeholder for other views.
40  */
41 @JsModule("./styles/shared-styles.js")
42 @CssImport(value = "./styles/views/main/main-view.css", themeFor =
      "vaadin-app-layout")
43 @CssImport("./styles/views/main/main-view.css")
44 @PWA(name = "VAD Social", shortName = "VAD Social",
      enableInstallPrompt = false)
45
46 public class MainView extends AppLayout {
47
48     public static AuthResponse authResponse = new AuthResponse
      ("","", Instant.MIN, "", "");
49     private final Tabs menu;
50     Button logout = new Button("Logout");
51     Anchor userNameLabel = new Anchor();
52
53     public MainView() {
54         HorizontalLayout header = createHeader();
55         menu = createMenuTabs();
56         if(!authResponse.getUserName().equals("")) {
57             userNameLabel.setHref("http://localhost:4200/my-profile
      ");
58             userNameLabel.setText(authResponse.getUserName());
59             header.add(userNameLabel);
60             header.add(logout);
61             logout.addClickListener(e -> {
62                 RecommendationApi recommendationApi = new
      RecommendationApi();
63                 recommendationApi.setIndex(PhotoFeedView.
```

```
    getMaxIndex());
64                  authResponse = new AuthResponse("","", Instant.MIN,
     "", "");
65                  UI.getCurrent().getPage().setLocation("http://
    localhost:4200/sign-up");
66              });
67          }
68          addToNavbar(createTopBar(header, menu));
69      }
70
71      private VerticalLayout createTopBar(HorizontalLayout header,
        Tabs menu) {
72          VerticalLayout layout = new VerticalLayout();
73          layout.getThemeList().add("dark");
74          layout.setWidthFull();
75          layout.setSpacing(false);
76          layout.setPadding(false);
77          layout.setAlignItems(FlexComponent.Alignment.CENTER);
78          layout.add(header, menu);
79          return layout;
80      }
81
82      private HorizontalLayout createHeader() {
83          HorizontalLayout header = new HorizontalLayout();
84          header.setPadding(false);
85          header.setSpacing(false);
86          header.setWidthFull();
87          header.setAlignItems(FlexComponent.Alignment.CENTER);
88          header.setId("header");
89          Image logo = new Image("images/logo.png", "My Project logo
        ");
90          logo.setId("logo");
91          header.add(logo);
92          Image avatar = new Image("images/user.svg", "Avatar");
93          avatar.setId("avatar");
94          header.add(new H1("VAD Social"));
95          header.add(avatar);
96          return header;
97      }
98
99      private static Tabs createMenuTabs() {
100         final Tabs tabs = new Tabs();
101         tabs.getStyle().set("max-width", "100%");
102         tabs.add(getAvailableTabs());
103         return tabs;
104     }
105
106     private static Tab[] getAvailableTabs() {
107                 if(authResponse.getUserName().equals("")) {
108                     return new Tab[] {
109                     createTab("About", AboutView.class),
110                             createTab("Sign up", PersonFormView.
        class),
111                             createTab("Login", Login.class)};
112                 }
113                     return new Tab[] {
114                             createTab("About", AboutView.class),
```

```
                createTab("Photo feed", PhotoFeedView.
    class),
                createTab("Friend feed", FriendFeed.
    class),
                createTab("My profile", MyprofileView.
    class),
                createTab("Account", AccountView.class)
    };

    }

    private static Tab createTab(String text, Class<? extends
    Component> navigationTarget) {
        final Tab tab = new Tab();
        tab.addThemeVariants(TabVariant.LUMO_ICON_ON_TOP);
        tab.add(new RouterLink(text, navigationTarget));
        ComponentUtil.setData(tab, Class.class, navigationTarget);
        return tab;
    }

    @Override
    protected void afterNavigation() {
        super.afterNavigation();
        getTabForComponent(getContent()).ifPresent(menu::
    setSelectedTab);
    }

    private Optional<Tab> getTabForComponent(Component component) {
        return menu.getChildren()
                .filter(tab -> ComponentUtil.getData(tab, Class.
    class)
                        .equals(component.getClass()))
                .findFirst().map(Tab.class::cast);
    }
}
```

MyProfileView

```
package com.example.application.views.myprofile;

import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.nio.charset.StandardCharsets;
import java.util.*;

import com.example.application.callApi.PhotoApi;
import com.example.application.data.PhotoModel;
import com.vaadin.flow.component.*;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.dependency.CssImport;
import com.vaadin.flow.component.dependency.JsModule;
import com.vaadin.flow.component.grid.Grid;
import com.vaadin.flow.component.grid.GridVariant;
import com.vaadin.flow.component.html.Div;
import com.vaadin.flow.component.html.Image;
import com.vaadin.flow.component.html.Label;
import com.vaadin.flow.component.html.Span;
```

```
21 import com.vaadin.flow.component.icon.Icon;
22 import com.vaadin.flow.component.icon.VaadinIcon;
23 import com.vaadin.flow.component.notification.Notification;
24 import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
25 import com.vaadin.flow.component.orderedlayout.VerticalLayout;
26 import com.vaadin.flow.component.textfield.TextArea;
27 import com.vaadin.flow.component.upload.Upload;
28 import com.vaadin.flow.component.upload.receivers.
       MultiFileMemoryBuffer;
29 import com.vaadin.flow.internal.MessageDigestUtil;
30 import com.vaadin.flow.router.AfterNavigationEvent;
31 import com.vaadin.flow.router.AfterNavigationObserver;
32 import com.vaadin.flow.router.PageTitle;
33 import com.vaadin.flow.router.Route;
34 import com.example.application.views.main.MainView;
35 import com.vaadin.flow.server.StreamResource;
36 import org.apache.commons.io.IOUtils;
37
38 import javax.imageio.ImageIO;
39 import javax.imageio.ImageReader;
40 import javax.imageio.stream.ImageInputStream;
41
42 @Route(value = "my-profile", layout = MainView.class)
43 @PageTitle("My profile")
44 @CssImport(value = "./styles/views/myprofile/myprofile-view.css",
       include = "lumo-badge")
45 @JsModule("@vaadin/vaadin-lumo-styles/badge.js")
46 public class MyprofileView extends Div implements
       AfterNavigationObserver {
47
48     Grid<PhotoModel> grid = new Grid<>();
49     Button button = new Button("Post");
50     Div output = new Div();
51     MultiFileMemoryBuffer buffer = new MultiFileMemoryBuffer();
52     Upload upload = new Upload(buffer);
53     private boolean enableUpload = true;
54     private String filename;
55     private PhotoApi photoApi = new PhotoApi();
56     private List<PhotoModel> photoModels;
57     private TextArea description = new TextArea("Description");
58
59     public MyprofileView() {
60         if(MainView.authResponse.getUserName().equals("")){
61             add(new Label("You are not login"));
62         }else {
63             add(description);
64             upload.setAcceptedFileTypes("image/jpeg", "image/png",
       "image/gif");
65
66             upload.addSucceededListener(event -> {
67                 if(enableUpload) {
68                     filename = event.getFileName();
69                     Component component = createComponent(event.
       getMIMEType(),
70                             filename,
71                             buffer.getInputStream(filename));
72                     showOutput(filename, component, output);
```

```
73                      enableUpload = false;
74                  }
75              });
76
77              add(upload, output);
78              add(button);
79              setId("myprofile-view");
80              addClassName("myprofile-view");
81              setSizeFull();
82              grid.setHeight("100%");
83              grid.addThemeVariants(GridVariant.LUMO_NO_BORDER,
    GridVariant.LUMO_NO_ROW_BORDERS);
84              grid.addComponentColumn(photoModel -> createCard(
    photoModel));
85              add(grid);
86              button.addClickListener(e->{
87                  try {
88                      byte[] bytes = IOUtils.toByteArray(buffer.
    getInputStream(filename));
89                      photoApi.addPhoto(bytes, description.getValue()
    );
90                      //photoModels.add(photoApi.getLast(photoApi.
    getAllByUsername()));//
91                      //grid.setItems(photoModels);
92                  } catch (IOException ioException) {
93                      ioException.printStackTrace();
94                  }
95                  Notification.show("Post was uploaded!", 2000,
    Notification.Position.MIDDLE);
96                  UI.getCurrent().getPage().reload();
97              });
98          }
99      }
100
101
102     private HorizontalLayout createCard(PhotoModel photoModel) {
103         photoApi.setDislikes(photoModel);
104         photoApi.setLikes(photoModel);
105         HorizontalLayout card = new HorizontalLayout();
106         card.addClassName("card");
107         card.setSpacing(false);
108         card.getThemeList().add("spacing-s");
109
110         StreamResource resource = new StreamResource("
    dummyImageName.jpg", () -> new ByteArrayInputStream(photoModel.
    getPicture()));
111         Image image = new Image(resource, "dummy image");
112         VerticalLayout description = new VerticalLayout();
113         description.addClassName("description");
114         description.setSpacing(false);
115         description.setPadding(false);
116
117         HorizontalLayout header = new HorizontalLayout();
118         header.addClassName("header");
119         header.setSpacing(false);
120         header.getThemeList().add("spacing-s");
121
```

```
122         Span name = new Span(MainView.authResponse.getUserName());
123         name.addClassName("name");
124         header.add(name);
125 
126         Span post = new Span(photoModel.getDescription());
127         post.addClassName("post");
128 
129         HorizontalLayout actions = new HorizontalLayout();
130         actions.addClassName("actions");
131         actions.setSpacing(false);
132         actions.getThemeList().add("spacing-s");
133 
134         Icon likeIcon = new Icon(VaadinIcon.THUMBS_UP);
135         if(photoApi.isLike(photoModel)){
136             likeIcon.setColor("green");
137         }else {
138             likeIcon.setColor("black");
139         }
140         Span likes = new Span(String.valueOf(photoModel.getLikes())
    );
141         likes.addClassName("likes");
142         Icon dislikeIcon = new Icon(VaadinIcon.THUMBS_DOWN);
143         if(photoApi.isDislike(photoModel)){
144             dislikeIcon.setColor("red");
145         }else {
146             dislikeIcon.setColor("black");
147         }
148         Span dislikes = new Span(String.valueOf(photoModel.
    getDislikes()));
149         dislikes.addClassName("dislikes");
150         actions.add(likeIcon, likes, dislikeIcon, dislikes);
151         likeIcon.addClickListener(e->{
152             if(likeIcon.getColor().equals("green")){
153                 likeIcon.setColor("black");
154                 photoModel.setLikes(photoModel.getLikes()-1);
155                 likes.setText(String.valueOf(photoModel.getLikes())
    );
156                 photoApi.updateLike(photoModel);
157             }else {
158                 likeIcon.setColor("green");
159                 photoModel.setLikes(photoModel.getLikes()+1);
160                 likes.setText(String.valueOf(photoModel.getLikes())
    );
161                 if(dislikeIcon.getColor().equals("red")) {
162                     photoModel.setDislikes(photoModel.getDislikes()
    -1);
163                     dislikes.setText(String.valueOf(photoModel.
    getDislikes()));
164                     dislikeIcon.setColor("black");
165                     photoApi.updateLikeDislike(photoModel);
166                 }else{
167                     photoApi.updateLike(photoModel);
168                 }
169             }
170         });
171 
172         dislikeIcon.addClickListener(e->{
```

```
173             if(dislikeIcon.getColor().equals("red")){
174                 dislikeIcon.setColor("black");
175                 photoModel.setDislikes(photoModel.getDislikes()-1);
176                 dislikes.setText(String.valueOf(photoModel.
    getDislikes()));
177                 photoApi.updateDislike(photoModel);
178             }else {
179                 dislikeIcon.setColor("red");
180                 photoModel.setDislikes(photoModel.getDislikes()+1);
181                 dislikes.setText(String.valueOf(photoModel.
    getDislikes()));
182                 if(likeIcon.getColor().equals("green")) {
183                     photoModel.setLikes(photoModel.getLikes()-1);
184                     likes.setText(String.valueOf(photoModel.
    getLikes()));
185                     likeIcon.setColor("black");
186                     photoApi.updateLikeDislike(photoModel);
187                 }else{
188                     photoApi.updateDislike(photoModel);
189                 }
190             }
191         });
192 
193         description.add(header, post, actions);
194         card.add(image, description);
195         return card;
196     }
197     @Override
198     public void afterNavigation(AfterNavigationEvent event) {
199         photoModels = photoApi.getAllByUsername();
200         Collections.reverse(photoModels);
201         for (PhotoModel photoModel: photoModels) {
202             photoModel.setPicture(photoApi.getImage(photoModel.
    getId()));
203         }
204 
205         grid.setItems(photoModels);
206     }
207 
208 
209 
210 
211 
212     private Component createComponent(String mimeType, String
    fileName,
213                                       InputStream stream) {
214         if (mimeType.startsWith("text")) {
215             return createTextComponent(stream);
216         } else if (mimeType.startsWith("image")) {
217             Image image = new Image();
218             try {
219 
220                 byte[] bytes = IOUtils.toByteArray(stream);
221                 image.getElement().setAttribute("src", new
    StreamResource(
222                         fileName, () -> new ByteArrayInputStream(
    bytes)));
```

```
223                 try (ImageInputStream in = ImageIO.
    createImageInputStream(
224                         new ByteArrayInputStream(bytes))) {
225                     final Iterator<ImageReader> readers = ImageIO
226                             .getImageReaders(in);
227                     if (readers.hasNext()) {
228                         ImageReader reader = readers.next();
229                         try {
230                             reader.setInput(in);
231                             image.setWidth(reader.getWidth(0) + "px
    ");
232                             image.setHeight(reader.getHeight(0) + "
    px");
233                         } finally {
234                             reader.dispose();
235                         }
236                     }
237                 }
238             } catch (IOException e) {
239                 e.printStackTrace();
240             }
241 
242             return image;
243         }
244         Div content = new Div();
245         String text = String.format("Mime type: '%s'\nSHA-256 hash:
     '%s'",
246                 mimeType, MessageDigestUtil.sha256(stream.toString
    ()));
247         content.setText(text);
248         return content;
249 
250     }
251 
252     private Component createTextComponent(InputStream stream) {
253         String text;
254         try {
255             text = IOUtils.toString(stream, StandardCharsets.UTF_8)
    ;
256         } catch (IOException e) {
257             text = "exception reading stream";
258         }
259         return new Text(text);
260     }
261 
262     private void showOutput(String text, Component content,
263                             HasComponents outputContainer) {
264         HtmlComponent p = new HtmlComponent(Tag.P);
265         p.getElement().setText(text);
266         outputContainer.add(p);
267         outputContainer.add(content);
268     }
269 
270 }
```

PersonFormView

```
1 package com.example.application.views.personform;
```

```
2
3  import com.example.application.callApi.RecommendationApi;
4  import com.example.application.callApi.UserApi;
5  import com.example.application.data.entity.User;
6  import com.vaadin.flow.component.Component;
7  import com.vaadin.flow.component.button.Button;
8  import com.vaadin.flow.component.button.ButtonVariant;
9  import com.vaadin.flow.component.dependency.CssImport;
10 import com.vaadin.flow.component.formlayout.FormLayout;
11 import com.vaadin.flow.component.html.Anchor;
12 import com.vaadin.flow.component.html.Div;
13 import com.vaadin.flow.component.html.H3;
14 import com.vaadin.flow.component.notification.Notification;
15 import com.vaadin.flow.component.notification.Notification.*;
16 import com.vaadin.flow.component.orderedlayout.FlexComponent;
17 import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
18 import com.vaadin.flow.component.textfield.EmailField;
19 import com.vaadin.flow.component.textfield.PasswordField;
20 import com.vaadin.flow.component.textfield.TextField;
21 import com.vaadin.flow.data.binder.Binder;
22 import com.vaadin.flow.data.validator.EmailValidator;
23 import com.vaadin.flow.router.PageTitle;
24 import com.vaadin.flow.router.Route;
25 import com.example.application.views.main.MainView;
26
27 @Route(value = "sign-up", layout = MainView.class)
28 @PageTitle("Sign up")
29 @CssImport("./styles/views/signupform/sign-up-view.css")
30 public class PersonFormView extends Div {
31
32     private TextField username = new TextField("Username");
33     private EmailField email = new EmailField("Email address");
34     private PasswordField password = new PasswordField("Password");
35     private PasswordField verifyPassword = new PasswordField("
       Verify Password");
36     private Anchor anchor = new Anchor("http://localhost:4200/login
       ", "Log In");
37
38     private Button signup = new Button("Sign up");
39
40     private Binder<User> binder = new Binder(User.class);
41     private RecommendationApi recommendationApi = new
       RecommendationApi();
42
43     public PersonFormView() {
44         setId("sign-up-view");
45
46         add(createTitle());
47         add(createFormLayout());
48         add(createButtonLayout());
49
50         clearForm();
51         signup.addClickListener(e -> {
52             if(username.isInvalid() || email.isInvalid() ||
       password.isInvalid() || verifyPassword.isInvalid() ||
53                 username.getValue().equals("") || email.getValue().
       equals("") || password.getValue().equals("")) {
```

```
54                  Notification.show("Error",1000,Position.MIDDLE);
55              }else{
56                      UserApi.callServiceSignup(new User(email.
    getValue(), password.getValue(), username.getValue()));
57                      Notification.show("User registered!",1000,
    Position.MIDDLE);
58                      clearForm();
59              }
60          });
61      }
62
63      private void clearForm() {
64          binder.setBean(new User());
65      }
66
67      private Component createTitle() {
68          return new H3("User registration");
69      }
70
71      private Component createFormLayout() {
72          FormLayout formLayout = new FormLayout();
73          email.setErrorMessage("Please enter a valid email address")
    ;
74          password.setErrorMessage("Please verify your password");
75          username.setErrorMessage("Name must contain at least three
    characters");
76          allBinder();
77          formLayout.add(username, email, password, verifyPassword);
78          return formLayout;
79      }
80
81      private void allBinder() {
82          binder.forField(email)
83                  .withValidator(new EmailValidator(""))
84                  .withValidationStatusHandler(status -> {
85                      email.setInvalid(status.isError());
86                  })
87                  .bind(User::getEmail, User::setEmail);
88
89
90          binder.forField(username)
91                  .withValidator(
92                          name -> name.length() >= 3, username.
    getErrorMessage())
93                  .withValidationStatusHandler(status -> {
94                      username.setInvalid(status.isError());
95                  })
96                  .bind(User::getUserName, User::setUserName);
97          binder.forField(password)
98                  .withValidator(
99                          password -> password.equals(verifyPassword.
    getValue()), password.getErrorMessage())
100                 .withValidationStatusHandler(status -> {
101                     password.setInvalid(status.isError());
102                 })
103                 .bind(User::getPassword, User::setPassword);
104         binder.forField(verifyPassword)
```

```
                .withValidator(
                        verifyPassword -> verifyPassword.equals(
    password.getValue()), password.getErrorMessage())
                .withValidationStatusHandler(status -> {
                    password.setInvalid(status.isError());
                })
                .bind(User::getPassword, User::setPassword);
    }

    private Component createButtonLayout() {
        HorizontalLayout buttonLayout = new HorizontalLayout();
        buttonLayout.addClassName("button-layout");
        signup.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
        buttonLayout.add(signup);
        buttonLayout.add(anchor);
        buttonLayout.setDefaultVerticalComponentAlignment(
    FlexComponent.Alignment.CENTER);
        return buttonLayout;
    }

}
```

PhotoFeedView

```
package com.example.application.views.photofeed;

import com.example.application.callApi.PhotoApi;
import com.example.application.callApi.RecommendationApi;
import com.example.application.data.PhotoModel;
import com.example.application.views.main.MainView;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.dependency.CssImport;
import com.vaadin.flow.component.dependency.JsModule;
import com.vaadin.flow.component.grid.Grid;
import com.vaadin.flow.component.grid.GridVariant;
import com.vaadin.flow.component.html.*;
import com.vaadin.flow.component.icon.Icon;
import com.vaadin.flow.component.icon.VaadinIcon;
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.router.PageTitle;
import com.vaadin.flow.router.Route;
import com.vaadin.flow.server.StreamResource;

import java.io.ByteArrayInputStream;
import java.util.ArrayList;
import java.util.List;

@Route(value = "photo-feed", layout = MainView.class)
@PageTitle("Photo Feed")
@CssImport(value = "./styles/views/photofeed/photo-feed-view.css",
    include = "lumo-badge")
@JsModule("@vaadin/vaadin-lumo-styles/badge.js")
public class PhotoFeedView extends Div {

    Grid<PhotoModel> grid = new Grid<>();
    private static int index;
    private static int maxIndex;
```

```
34      private List<PhotoModel> photoModelList = new ArrayList<>();
35      private Button next = new Button(">");
36      private Button prev = new Button("<");
37      private PhotoApi photoApi = new PhotoApi();
38      private RecommendationApi recommendationApi = new
        RecommendationApi();
39      private HorizontalLayout card = new HorizontalLayout();
40      private PhotoModel photoModel;
41
42      public PhotoFeedView() {
43          if (MainView.authResponse.getUserName().equals("")) {
44              add(new Label("You are not login"));
45          } else {
46
47              photoModelList = recommendationApi.getRecom();
48              for (PhotoModel photoModel : photoModelList) {
49                  photoApi.getUsername(photoModel);
50              }
51              setId("photo-feed-view");
52              addClassName("photo-feed-view");
53              setSizeFull();
54              grid.setHeight("100%");
55              grid.addThemeVariants(GridVariant.LUMO_NO_BORDER,
        GridVariant.LUMO_NO_ROW_BORDERS);
56              photoModel = photoModelList.get(index);
57              photoModel.setPicture(photoApi.getImage(photoModel.
        getId()));
58              createCard(photoModel);
59              add(prev);
60              add(card);
61              add(next);
62              prev.addClickListener(e -> {
63                  next.setVisible(true);
64                  try {
65                      photoModel = photoModelList.get(index - 1);
66                      photoModel.setPicture(photoApi.getImage(
        photoModel.getId()));
67                      createCard(photoModel);
68                      index--;
69                  } catch (IndexOutOfBoundsException ex) {
70                      prev.setVisible(false);
71                  }
72              });
73              next.addClickListener(e -> {
74                  prev.setVisible(true);
75                  try {
76                      photoModel = photoModelList.get(index + 1);
77                      photoModel.setPicture(photoApi.getImage(
        photoModel.getId()));
78                      createCard(photoModel);
79                      index++;
80                      if (index > maxIndex) {
81                          maxIndex = index;
82                      }
83                  } catch (IndexOutOfBoundsException ex) {
84                      next.setVisible(false);
85                  }
```

```
            });
        }
    }

    public static int getMaxIndex() {
        return maxIndex;
    }

    public static void setMaxIndex(int maxIndex) {
        PhotoFeedView.maxIndex = maxIndex;
    }

    public static int getIndex() {
        return index;
    }

    public static void setIndex(int index) {
        PhotoFeedView.index = index;
    }

    private void createCard(PhotoModel photoModel) {
        photoApi.setDislikes(photoModel);
        photoApi.setLikes(photoModel);
        card.addClassName("card");
        card.setSpacing(false);
        card.getThemeList().add("spacing-s");

        StreamResource resource = new StreamResource("
    dummyImageName.jpg", () -> new ByteArrayInputStream(photoModel.
    getPicture()));
        Image image = new Image(resource, "dummy image");
        VerticalLayout description = new VerticalLayout();
        description.addClassName("description");
        description.setSpacing(false);
        description.setPadding(false);

        HorizontalLayout header = new HorizontalLayout();
        header.addClassName("header");
        header.setSpacing(false);
        header.getThemeList().add("spacing-s");

        Anchor name = new Anchor("http://localhost:4200/username/"
    + photoModel.getUser().getUserName()
                , photoModel.getUser().getUserName());
        name.addClassName("name");
        header.add(name);

        Span post = new Span(photoModel.getDescription());
        post.addClassName("post");

        HorizontalLayout actions = new HorizontalLayout();
        actions.addClassName("actions");
        actions.setSpacing(false);
        actions.getThemeList().add("spacing-s");
        Icon likeIcon = new Icon(VaadinIcon.THUMBS_UP);
        if (photoApi.isLike(photoModel)) {
            likeIcon.setColor("green");
```

```
140         } else {
141             likeIcon.setColor("black");
142         }
143         Span likes = new Span(String.valueOf(photoModel.getLikes())
    );
144         likes.addClassName("likes");
145         Icon dislikeIcon = new Icon(VaadinIcon.THUMBS_DOWN);
146         if (photoApi.isDislike(photoModel)) {
147             dislikeIcon.setColor("red");
148         } else {
149             dislikeIcon.setColor("black");
150         }
151         Span dislikes = new Span(String.valueOf(photoModel.
    getDislikes()));
152         dislikes.addClassName("dislikes");
153         actions.add(likeIcon, likes, dislikeIcon, dislikes);
154         likeIcon.addClickListener(e -> {
155             if (likeIcon.getColor().equals("green")) {
156                 likeIcon.setColor("black");
157                 photoModel.setLikes(photoModel.getLikes() - 1);
158                 likes.setText(String.valueOf(photoModel.getLikes())
    );
159                 photoApi.updateLike(photoModel);
160             } else {
161                 likeIcon.setColor("green");
162                 photoModel.setLikes(photoModel.getLikes() + 1);
163                 likes.setText(String.valueOf(photoModel.getLikes())
    );
164                 if (dislikeIcon.getColor().equals("red")) {
165                     photoModel.setDislikes(photoModel.getDislikes()
     - 1);
166                     dislikes.setText(String.valueOf(photoModel.
    getDislikes()));
167                     dislikeIcon.setColor("black");
168                     photoApi.updateLikeDislike(photoModel);
169                 } else {
170                     photoApi.updateLike(photoModel);
171                 }
172             }
173         });
174 
175         dislikeIcon.addClickListener(e -> {
176             if (dislikeIcon.getColor().equals("red")) {
177                 dislikeIcon.setColor("black");
178                 photoModel.setDislikes(photoModel.getDislikes() -
    1);
179                 dislikes.setText(String.valueOf(photoModel.
    getDislikes()));
180                 photoApi.updateDislike(photoModel);
181             } else {
182                 dislikeIcon.setColor("red");
183                 photoModel.setDislikes(photoModel.getDislikes() +
    1);
184                 dislikes.setText(String.valueOf(photoModel.
    getDislikes()));
185                 if (likeIcon.getColor().equals("green")) {
186                     photoModel.setLikes(photoModel.getLikes() - 1);
```

```
                        likes.setText(String.valueOf(photoModel.
    getLikes()));
                        likeIcon.setColor("black");
                        photoApi.updateLikeDislike(photoModel);
                    } else {
                        photoApi.updateDislike(photoModel);
                    }
                }
            });

            description.add(header, post, actions);
            card.removeAll();
            card.add(image, description);
        }


}
```

UserView

```
package com.example.application.views.userView;

import com.example.application.callApi.PhotoApi;
import com.example.application.callApi.RecommendationApi;
import com.example.application.data.PhotoModel;
import com.example.application.views.main.MainView;
import com.vaadin.flow.component.*;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.dependency.CssImport;
import com.vaadin.flow.component.dependency.JsModule;
import com.vaadin.flow.component.grid.Grid;
import com.vaadin.flow.component.grid.GridVariant;
import com.vaadin.flow.component.html.Div;
import com.vaadin.flow.component.html.Image;
import com.vaadin.flow.component.html.Label;
import com.vaadin.flow.component.html.Span;
import com.vaadin.flow.component.icon.Icon;
import com.vaadin.flow.component.icon.VaadinIcon;
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.component.upload.Upload;
import com.vaadin.flow.component.upload.receivers.
    MultiFileMemoryBuffer;
import com.vaadin.flow.router.*;
import com.vaadin.flow.server.StreamResource;
import org.springframework.web.client.HttpServerErrorException;

import java.io.ByteArrayInputStream;
import java.util.Collections;
import java.util.List;

@Route(value = "username", layout = MainView.class)
@PageTitle("user profile")
@CssImport(value = "./styles/views/myprofile/myprofile-view.css",
    include = "lumo-badge")
@JsModule("@vaadin/vaadin-lumo-styles/badge.js")
public class UserView extends Div
        implements HasUrlParameter<String>, AfterNavigationObserver
```

```
    {

    Grid<PhotoModel> grid = new Grid<>();
    Button button = new Button("post");
    Div output = new Div();
    MultiFileMemoryBuffer buffer = new MultiFileMemoryBuffer();
    Upload upload = new Upload(buffer);
    Button followButton = new Button();
    private boolean enableUpload = true;
    private String filename;
    private PhotoApi photoApi = new PhotoApi();
    private RecommendationApi recommendationApi = new
    RecommendationApi();
    private List<PhotoModel> photoModels;
    private String username;

    @Override
    public void setParameter(BeforeEvent event,
                             String parameter) {
        username = parameter;
        if(MainView.authResponse.getUserName().equals("")){
            add(new Label("You are not login"));
        }else {
            try {
                if (recommendationApi.isFollow(username)) {
                    followButton.setText("Unfollow");
                } else {
                    followButton.setText("Follow");
                }
            }catch (NullPointerException ex){
                UI.getCurrent().navigate("account");
                return;
            }

            followButton.addClickListener(e->{
                recommendationApi.setFollow(username);
                if(followButton.getText().equals("Follow")){
                    followButton.setText("Unfollow");
                }else{
                    followButton.setText("Follow");
                }
            });
            setId("myprofile-view");
            addClassName("myprofile-view");
            setSizeFull();
            grid.setHeight("100%");
            grid.addThemeVariants(GridVariant.LUMO_NO_BORDER,
    GridVariant.LUMO_NO_ROW_BORDERS);
            grid.addComponentColumn(photoModel -> createCard(
    photoModel));
            add(followButton);
            add(grid);
        }
    }
    private HorizontalLayout createCard(PhotoModel photoModel) {
        photoApi.setDislikes(photoModel);
        photoApi.setLikes(photoModel);
```

```
90          HorizontalLayout card = new HorizontalLayout();
91          card.addClassName("card");
92          card.setSpacing(false);
93          card.getThemeList().add("spacing-s");
94
95          StreamResource resource = new StreamResource("
        dummyImageName.jpg", () -> new ByteArrayInputStream(photoModel.
        getPicture()));
96          Image image = new Image(resource, "dummy image");
97          VerticalLayout description = new VerticalLayout();
98          description.addClassName("description");
99          description.setSpacing(false);
100         description.setPadding(false);
101
102         HorizontalLayout header = new HorizontalLayout();
103         header.addClassName("header");
104         header.setSpacing(false);
105         header.getThemeList().add("spacing-s");
106
107         Span name = new Span(username);
108         name.addClassName("name");
109         header.add(name);
110
111         Span post = new Span(photoModel.getDescription());
112         post.addClassName("post");
113
114         HorizontalLayout actions = new HorizontalLayout();
115         actions.addClassName("actions");
116         actions.setSpacing(false);
117         actions.getThemeList().add("spacing-s");
118
119         Icon likeIcon = new Icon(VaadinIcon.THUMBS_UP);
120         if(photoApi.isLike(photoModel)){
121             likeIcon.setColor("green");
122         }else {
123             likeIcon.setColor("black");
124         }
125         Span likes = new Span(String.valueOf(photoModel.getLikes())
        );
126         likes.addClassName("likes");
127         Icon dislikeIcon = new Icon(VaadinIcon.THUMBS_DOWN);
128         if(photoApi.isDislike(photoModel)){
129             dislikeIcon.setColor("red");
130         }else {
131             dislikeIcon.setColor("black");
132         }
133         Span dislikes = new Span(String.valueOf(photoModel.
        getDislikes()));
134         dislikes.addClassName("dislikes");
135         actions.add(likeIcon, likes, dislikeIcon, dislikes);
136         likeIcon.addClickListener(e->{
137             if(likeIcon.getColor().equals("green")){
138                 likeIcon.setColor("black");
139                 photoModel.setLikes(photoModel.getLikes()-1);
140                 likes.setText(String.valueOf(photoModel.getLikes())
        );
141                 photoApi.updateLike(photoModel);
```

```
142             }else {
143                 likeIcon.setColor("green");
144                 photoModel.setLikes(photoModel.getLikes()+1);
145                 likes.setText(String.valueOf(photoModel.getLikes())
    );
146                 if(dislikeIcon.getColor().equals("red")) {
147                     photoModel.setDislikes(photoModel.getDislikes()
    -1);
148                     dislikes.setText(String.valueOf(photoModel.
    getDislikes()));
149                     dislikeIcon.setColor("black");
150                     photoApi.updateLikeDislike(photoModel);
151                 }else{
152                     photoApi.updateLike(photoModel);
153                 }
154             }
155         });
156 
157         dislikeIcon.addClickListener(e->{
158             if(dislikeIcon.getColor().equals("red")){
159                 dislikeIcon.setColor("black");
160                 photoModel.setDislikes(photoModel.getDislikes()-1);
161                 dislikes.setText(String.valueOf(photoModel.
    getDislikes()));
162                 photoApi.updateDislike(photoModel);
163             }else {
164                 dislikeIcon.setColor("red");
165                 photoModel.setDislikes(photoModel.getDislikes()+1);
166                 dislikes.setText(String.valueOf(photoModel.
    getDislikes()));
167                 if(likeIcon.getColor().equals("green")) {
168                     photoModel.setLikes(photoModel.getLikes()-1);
169                     likes.setText(String.valueOf(photoModel.
    getLikes()));
170                     likeIcon.setColor("black");
171                     photoApi.updateLikeDislike(photoModel);
172                 }else{
173                     photoApi.updateDislike(photoModel);
174                 }
175             }
176         });
177 
178         description.add(header, post, actions);
179         card.add(image, description);
180         return card;
181     }
182     @Override
183     public void afterNavigation(AfterNavigationEvent event) {
184 
185         // Set some data when this view is displayed.
186         try {
187             photoModels = photoApi.getAllByUsername(username);
188             Collections.reverse(photoModels);
189         }catch (HttpServerErrorException ex){
190             UI.getCurrent().navigate("account");
191             return;
192         }
```

```
193         for (PhotoModel photoModel: photoModels) {
194             photoModel.setPicture(photoApi.getImage(photoModel.
    getId()));
195         }
196
197         grid.setItems(photoModels);
198     }
199
200
201
202
203 }
```

Application

```
1 package com.example.application;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication
    ;
5 import org.springframework.boot.web.servlet.support.
    SpringBootServletInitializer;
6 import org.springframework.context.annotation.Bean;
7 import org.springframework.web.client.RestTemplate;
8 import org.vaadin.artur.helpers.LaunchUtil;
9
10 /**
11  * The entry point of the Spring Boot application.
12  */
13 @SpringBootApplication
14 public class Application extends SpringBootServletInitializer {
15
16     @Bean
17     public RestTemplate create(){
18         return new RestTemplate();
19     }
20
21     public static void main(String[] args) {
22         LaunchUtil.launchBrowserInDevelopmentMode(SpringApplication
    .run(Application.class, args));
23     }
24 }
```

3.2 Backend

GUIConfig

```
1 package com.project.socializingApp.configuration;
2
3 import org.springframework.context.annotation.Configuration;
4 import org.springframework.web.servlet.config.annotation.
    CorsRegistry;
5 import org.springframework.web.servlet.config.annotation.
    EnableWebMvc;
6 import org.springframework.web.servlet.config.annotation.
    ResourceHandlerRegistry;
```

```
import org.springframework.web.servlet.config.annotation.
    WebMvcConfigurer;

@Configuration
@EnableWebMvc
public class GUIConfig implements WebMvcConfigurer {

    @Override
    public void addCorsMappings(CorsRegistry corsRegistry) {
        corsRegistry.addMapping("/**")
                .allowedOrigins("*")
                .allowedMethods("*")
                .maxAge(3600L)
                .allowedHeaders("*")
                .exposedHeaders("Authorization")
                .allowCredentials(true);
    }

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry
    registry) {
        registry.addResourceHandler("swagger-ui.html")
                .addResourceLocations("classpath:/META-INF/
    resources/");

        registry.addResourceHandler("/webjars/**")
                .addResourceLocations("classpath:/META-INF/
    resources/webjars/");
    }
}
```

SecurityConfig

```
package com.project.socializingApp.configuration;

import com.project.socializingApp.service.UserDetService;
import lombok.AllArgsConstructor;
import lombok.NoArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.http.HttpMethod;
import org.springframework.security.authentication.
    AuthenticationManager;
import org.springframework.security.config.BeanIds;
import org.springframework.security.config.annotation.
    authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.
    HttpSecurity;
import org.springframework.security.config.annotation.web.
    configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.config.annotation.web.
    configuration.EnableWebSecurity;
import org.springframework.security.core.userdetails.
    UserDetailsService;
import org.springframework.security.crypto.bcrypt.
    BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder
    ;
```

```
18
19 @EnableWebSecurity
20 @AllArgsConstructor
21 public class SecurityConfig extends WebSecurityConfigurerAdapter {
22     private UserDetailsService userDetailsService;
23
24     @Bean(BeanIds.AUTHENTICATION_MANAGER)
25     @Override
26     public AuthenticationManager authenticationManagerBean() throws
        Exception {
27         return super.authenticationManagerBean();
28     }
29
30     @Autowired
31     public void configureGlobal(AuthenticationManagerBuilder
       authenticationManagerBuilder) throws Exception {
32         authenticationManagerBuilder.userDetailsService(
       userDetailsService)
33                 .passwordEncoder(passwordEncoder());
34     }
35     @Override
36     public void configure(HttpSecurity httpSecurity) throws
       Exception {
37         httpSecurity.cors().and()
38                 .csrf().disable()
39                 .authorizeRequests()
40                 .antMatchers("/api/auth/**","/photo/**", "/api/
       account/**","/recom/**")
41                 .permitAll()
42                 .antMatchers("/v2/api-docs",
43                         "/configuration/ui",
44                         "/swagger-resources/**",
45                         "/configuration/security",
46                         "/swagger-ui.html",
47                         "/webjars/**")
48                 .permitAll()
49                 .anyRequest()
50                 .authenticated();
51     }
52
53     @Bean
54     public BCryptPasswordEncoder passwordEncoder(){
55         return new BCryptPasswordEncoder();
56     }
57
58 }
```

SwagConfig

```
1 package com.project.socializingApp.configuration;
2
3 import org.springframework.context.annotation.Bean;
4 import org.springframework.context.annotation.Configuration;
5 import springfox.documentation.builders.ApiInfoBuilder;
6 import springfox.documentation.builders.PathSelectors;
7 import springfox.documentation.builders.RequestHandlerSelectors;
8 import springfox.documentation.service.ApiInfo;
9 import springfox.documentation.service.Contact;
```

```
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

@Configuration
@EnableSwagger2
public class SwagConfig {
    @Bean
    public Docket redditCloneApi() {
        return new Docket(DocumentationType.SWAGGER_2)
                .select()
                .apis(RequestHandlerSelectors.any())
                .paths(PathSelectors.any())
                .build()
                .apiInfo(getApiInfo());
    }

    private ApiInfo getApiInfo() {
        return new ApiInfoBuilder()
                .title("API")
                .version("1.0")
                .description("API for VAD Social")
                .contact(new Contact("Mihali Vlad, Stioiu Denis,
    Sauchea Alexandra", "http://localhost:4200", "xyz@email.com"))
                .license("Apache License Version 2.0")
                .build();
    }
}
```

AccountController

```
package com.project.socializingApp.controller;

import com.project.socializingApp.dataLayer.UserUpdateDetails;
import com.project.socializingApp.model.User;
import com.project.socializingApp.service.UserDetService;
import lombok.AllArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.crypto.bcrypt.
    BCryptPasswordEncoder;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/account")
@AllArgsConstructor
public class AccountController {

    private final UserDetService userService;

    public BCryptPasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @PostMapping("/update")
    public ResponseEntity<String> update(@RequestBody
```

```
    UserUpdateDetails userUpdateDetails) {
27          try {
28              User user = userService.getUserDetails(
    userUpdateDetails.getUserName());
29
30              if (!userUpdateDetails.getEmail().equals(user.getEmail
    ())) {
31                  user.setEmail(userUpdateDetails.getEmail());
32              }
33
34              if (userUpdateDetails.getNewPassword() != null && !
    userUpdateDetails.getNewPassword().isEmpty()) {
35                  BCryptPasswordEncoder encoder = passwordEncoder();
36
37                  if (encoder.matches(userUpdateDetails.
    getOldPassword(), user.getPassword())) {
38                      user.setPassword(encoder.encode(
    userUpdateDetails.getNewPassword()));
39                  } else {
40                      return new ResponseEntity<>("Wrong password!",
    HttpStatus.BAD_REQUEST);
41                  }
42              }
43
44              userService.updateUserDetails(user);
45
46              return ResponseEntity.ok().build();
47
48          } catch (Exception ex) {
49              System.out.println(ex);
50              return ResponseEntity.status(HttpStatus.
    INTERNAL_SERVER_ERROR).build();
51          }
52      }
53
54      @DeleteMapping("/delete/{userName}")
55      public ResponseEntity<String> delete(@PathVariable String
    userName) {
56          try {
57              User user = userService.getUserDetails(userName);
58              userService.deleteUserAccount(user);
59              return ResponseEntity.ok().build();
60          } catch (Exception ex) {
61              System.out.println(ex);
62              return ResponseEntity.status(HttpStatus.
    INTERNAL_SERVER_ERROR).build();
63          }
64      }
65
66      @GetMapping("/usernames")
67      @ResponseStatus(HttpStatus.OK)
68      public List<String> getAllUsers(){
69          return userService.listAllUsers();
70
71      }
72  }
```

Autentification

```java
package com.project.socializingApp.controller;


import com.project.socializingApp.dataLayer.AuthResponse;
import com.project.socializingApp.dataLayer.Login;
import com.project.socializingApp.dataLayer.RefreshTokenData;
import com.project.socializingApp.dataLayer.RegisterData;
import com.project.socializingApp.model.RefreshToken;
import com.project.socializingApp.service.AutentificationService;
import lombok.AllArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.client.HttpClientErrorException;

import javax.validation.Valid;

import java.time.LocalDate;
import java.util.HashSet;

import static org.springframework.http.HttpStatus.OK;

@RestController
@RequestMapping("/api/auth")
@AllArgsConstructor
public class Autentification {

    private final AutentificationService autentificationService;

    @PostMapping("/signup")
    public ResponseEntity<String> signup(@RequestBody RegisterData
    registerData) {
        try {
            autentificationService.signUp(registerData);
            return new ResponseEntity<>("User registered!",
    HttpStatus.OK);
        }catch (Exception ex){
            return new ResponseEntity<>("User taken! "+ex.
    getLocalizedMessage(), HttpStatus.FORBIDDEN);
        }
    }

    @PostMapping("/login")
    public AuthResponse login(@RequestBody Login login) {
        return autentificationService.login(login);
    }

    @PostMapping("/refresh/token")
    public AuthResponse refreshTokens(@Valid @RequestBody
    RefreshTokenData refreshTokenData) {
        return autentificationService.refreshToken(refreshTokenData
    );
    }
}
```

PhotoController

```java
package com.project.socializingApp.controller;
```

```
2
3  import com.project.socializingApp.dataLayer.RequestPhoto;
4  import com.project.socializingApp.model.PhotoModel;
5  import com.project.socializingApp.model.User;
6  import com.project.socializingApp.service.PhotoService;
7  import static org.springframework.http.HttpStatus.*;
8
9  import com.sun.mail.iap.Response;
10 import org.springframework.http.MediaType;
11 import org.springframework.http.ResponseEntity;
12 import org.springframework.stereotype.Controller;
13 import org.springframework.web.bind.annotation.*;
14 import org.springframework.web.context.annotation.RequestScope;
15
16 import java.util.List;
17 import java.util.Set;
18
19 @RestController
20 @RequestMapping("/photo")
21 public class PhotoController {
22     private PhotoService photoService;
23
24     public PhotoController(PhotoService photoService) {
25         this.photoService = photoService;
26     }
27
28     @GetMapping("/")
29     @ResponseStatus(OK)
30     public List<PhotoModel> getAll(){
31         return photoService.set();
32     }
33
34     @GetMapping("/user/")
35     @ResponseStatus(OK)
36     public List<PhotoModel> getAll(@RequestParam String username){
37         return photoService.setByUsername(username);
38     }
39
40     @PostMapping("/post")
41     @ResponseStatus(OK)
42     public PhotoModel add(@RequestBody RequestPhoto requestPhoto){
43         System.out.println(requestPhoto.getDescription());
44         return photoService.addPhoto(requestPhoto);
45     }
46
47     @DeleteMapping("/")
48     @ResponseStatus(OK)
49     public void delete(@RequestParam Long id){
50         photoService.delete(id);
51     }
52
53     @GetMapping("/get")
54     @ResponseStatus(OK)
55     public byte[] getPhoto(@RequestParam Long id){
56         return photoService.getPhoto(id);
57     }
58
```

```
59      @PutMapping("/like")
60      @ResponseStatus(OK)
61      public PhotoModel updateLike(@RequestBody Long id,
        @RequestParam String username){
62          return photoService.updateLike(id, username);
63      }
64
65      @PutMapping("/dislike")
66      @ResponseStatus(OK)
67      public PhotoModel updateDislike(@RequestBody Long id,
        @RequestParam String username){
68          return photoService.updateDislike(id, username);
69      }
70
71
72      @PostMapping("/like")
73      @ResponseStatus(OK)
74      public boolean isLike(@RequestBody Long id, @RequestParam
        String username){
75          return photoService.getLike(id, username);
76      }
77
78      @PostMapping("/dislike")
79      @ResponseStatus(OK)
80      public boolean isDislike(@RequestBody Long id, @RequestParam
        String username){
81          return photoService.getDislike(id, username);
82      }
83
84      @GetMapping("/getBy")
85      @ResponseStatus(OK)
86      public PhotoModel getById(@RequestParam Long id){
87          return photoService.getById(id);
88      }
89
90      @GetMapping("/like")
91      @ResponseStatus(OK)
92      public Integer getNrLikes(@RequestParam Long id) {
93          return photoService.getLikes(id);
94      }
95
96      @GetMapping("/dislike")
97      @ResponseStatus(OK)
98      public Integer getNrDislikes(@RequestParam Long id) {
99          return photoService.getDislikes(id);
100     }
101
102     @GetMapping("/username")
103     @ResponseStatus(OK)
104     public String getUsername(@RequestParam Long id){
105         return photoService.getUsername(id);
106     }
107
108     @PutMapping("/likedislike")
109     @ResponseStatus(OK)
110     public void updateLikeDislike(@RequestBody Long id,
        @RequestParam String username){
```

```
111             photoService.updateDislikeAndLike(id, username);
112         }
113
114 }
```

RecommendationController

```
1  package com.project.socializingApp.controller;
2
3  import com.project.socializingApp.model.PhotoModel;
4  import com.project.socializingApp.model.User;
5  import com.project.socializingApp.service.UserDetService;
6  import org.springframework.beans.factory.annotation.Autowired;
7  import org.springframework.http.HttpStatus;
8  import org.springframework.security.core.userdetails.
       UserDetailsService;
9  import org.springframework.transaction.annotation.Transactional;
10 import org.springframework.web.bind.annotation.*;
11
12 import java.util.List;
13 import java.util.Set;
14 import java.util.stream.Collectors;
15
16 @RestController
17 @RequestMapping("/recom")
18 public class RecommendationController {
19     private UserDetService userDetService;
20     @Autowired
21     public RecommendationController(UserDetService userDetService)
       {
22         this.userDetService = userDetService;
23     }
24     @GetMapping("/")
25     @ResponseStatus(HttpStatus.OK)
26     public List<PhotoModel> recomSet(@RequestParam String username)
       {
27         return userDetService.recommend(username);
28     }
29
30     @PostMapping("/")
31     @ResponseStatus(HttpStatus.OK)
32     public List<PhotoModel> initRecomSet(@RequestBody String
       username){
33         return userDetService.initRecommend(username);
34     }
35
36     @GetMapping("/index")
37     @ResponseStatus(HttpStatus.OK)
38     public Integer getRecomIndex(@RequestParam String username){
39         return userDetService.getRecomIndex(username);
40     }
41
42     @PostMapping("/index")
43     @ResponseStatus(HttpStatus.OK)
44     public Integer setRecomIndex(@RequestParam String username,
       @RequestBody Integer value){
45         return userDetService.setRecomIndex(username, value);
46     }
```

```
47
48      @GetMapping("/follow")
49      @ResponseStatus(HttpStatus.OK)
50      public List<String> getFollow(@RequestParam String username){
51          return userDetService.getFollow(username).stream().map(User
        ::getUserName).distinct().collect(Collectors.toList());
52      }
53
54      @PostMapping("/follow/")
55      @ResponseStatus(HttpStatus.OK)
56      public List<String> setFollow(@RequestParam String username,
        @RequestBody String follow){
57          return userDetService.setFollow(username, follow).stream().
        map(User::getUserName).distinct().collect(Collectors.toList());
58      }
59
60      @GetMapping("/follow/{username}/{follow}")
61      @ResponseStatus(HttpStatus.OK)
62      public boolean isFollow(@PathVariable("username") String
        username,@PathVariable("follow") String follow){
63          return userDetService.isFollow(username, follow);
64      }
65  }
```

AuthResponse

```
1  package com.project.socializingApp.dataLayer;
2
3  import lombok.AllArgsConstructor;
4  import lombok.Builder;
5  import lombok.Data;
6  import lombok.NoArgsConstructor;
7
8  import java.time.Instant;
9
10 @Data
11 @AllArgsConstructor
12 @NoArgsConstructor
13 @Builder
14 public class AuthResponse {
15      private String authenticationToken;
16      private String refreshToken;
17      private Instant expiresAt;
18      private String userName;
19      private String email;
20 }
```

Login

```
1  package com.project.socializingApp.dataLayer;
2
3  import lombok.AllArgsConstructor;
4  import lombok.Data;
5  import lombok.NoArgsConstructor;
6
7  @Data
8  @AllArgsConstructor
9  @NoArgsConstructor
10 public class Login {
```

```
11
12      private String userName;
13      private String password;
14  }
```

PhotoResponse

```
1  package com.project.socializingApp.dataLayer;
2
3  import com.fasterxml.jackson.annotation.JsonIgnore;
4  import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
5  import com.project.socializingApp.dataLayer.AuthResponse;
6  import com.project.socializingApp.model.User;
7  import lombok.*;
8  import net.bytebuddy.agent.builder.AgentBuilder;
9  import org.hibernate.annotations.Fetch;
10 import org.springframework.security.core.Transient;
11
12 import javax.persistence.*;
13
14 import java.io.Serializable;
15
16 import static javax.persistence.GenerationType.IDENTITY;
17
18 @Data
19 @AllArgsConstructor
20 @NoArgsConstructor
21 @Setter
22 @Getter
23 public class PhotoResponse {
24     private Long id;
25
26     private User user;
27
28     private Integer likes;
29     private Integer dislikes;
30     private String description;
31
32 }
```

RefreshTokenData

```
1  package com.project.socializingApp.dataLayer;
2
3  import lombok.AllArgsConstructor;
4  import lombok.Data;
5  import lombok.NoArgsConstructor;
6
7  import javax.validation.constraints.NotBlank;
8
9  @Data
10 @AllArgsConstructor
11 @NoArgsConstructor
12 public class RefreshTokenData {
13     @NotBlank
14     private String refreshToken;
15     private String userName;
16 }
```

RegisterData

```
package com.project.socializingApp.dataLayer;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class RegisterData {
    private String userName;
    private String email;
    private String password;
}
```

RequestPhoto

```
package com.project.socializingApp.dataLayer;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import com.project.socializingApp.model.User;
import lombok.*;

import java.io.Serializable;
import java.sql.Blob;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Setter
@Getter
@Builder
public class RequestPhoto implements Serializable {
    private String username;
    @JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})
    private byte[] image;
    private String description;
}
```

UserUpdateDetails

```
package com.project.socializingApp.dataLayer;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor

public class UserUpdateDetails {
    private String userName;
    private String email;
    private String oldPassword;
    private String newPassword;
}
```

Friendship

```
package com.project.socializingApp.model;

import lombok.Getter;
import lombok.Setter;

import javax.persistence.*;

@Entity
@Getter
@Setter
public class Friendship {

    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private Long friendshipId;

    @OneToOne
    private User owner;

    @OneToOne
    private User target;

    // other info
    public boolean compareTo(Friendship friendship){
        return this.getOwner().equals(friendship.getOwner()) &&
                this.getTarget().equals(friendship.getOwner());
    }
}
```

MyException

```
package com.project.socializingApp.model;

public class MyException extends RuntimeException{
    public MyException(String str) {
        super(str);
    }
    public MyException(String str, Exception exception) {
        super(str, exception);
    }
}
```

PhotoModel

```
package com.project.socializingApp.model;

import com.fasterxml.jackson.annotation.JsonIgnore;
import lombok.*;
import javax.persistence.*;
import java.util.List;
import static javax.persistence.GenerationType.IDENTITY;

//@Data
@AllArgsConstructor
@NoArgsConstructor
@Setter
@Getter
```

```
@Entity
public class PhotoModel {
    @Id
    @GeneratedValue(strategy = IDENTITY)
    private Long id;
    @ManyToOne(fetch = FetchType.LAZY, cascade = {CascadeType.ALL})
    @JsonIgnore
    private User user;

    //@Basic(fetch = FetchType.LAZY)
    @Lob
    @Column(columnDefinition="BLOB")
    @JsonIgnore
    private byte[] picture;

    @ManyToMany(fetch = FetchType.LAZY, cascade = {CascadeType.ALL
    })
    @JsonIgnore
    private List<User> likes;
    @ManyToMany(fetch = FetchType.LAZY,cascade = {CascadeType.ALL})
    @JsonIgnore
    private List<User> dislikes;
    private String description;

}
```

RefreshToken

```
package com.project.socializingApp.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import java.time.Instant;

@Data
@Entity
@AllArgsConstructor
@NoArgsConstructor
public class RefreshToken {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String token;
    private Instant createdDate;
}
```

User

```
package com.project.socializingApp.model;
import lombok.*;

import javax.persistence.*;
import javax.validation.constraints.Email;
```

```
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotEmpty;
import java.time.Instant;
import java.util.List;

import static javax.persistence.GenerationType.IDENTITY;

    //@Data
    @AllArgsConstructor
    @NoArgsConstructor
    @Setter
    @Getter
    @Entity
    public class User {
        @Id
        @GeneratedValue(strategy = IDENTITY)
        private Long userId;
        @NotBlank(message = "Username is required")
        private String userName;
        @NotBlank(message = "Password is required")
        private String password;
        @Email
        @NotEmpty(message = "Email is required")

        private String email;
        private Instant created;
        private boolean enabled;

        @ManyToMany(fetch = FetchType.LAZY,cascade = {CascadeType.
    ALL})
        private List<Friendship> friends;

        @OneToMany(fetch = FetchType.LAZY,cascade = {CascadeType.
    ALL})
        private List<PhotoModel> photos;

        @ManyToMany(fetch = FetchType.LAZY,cascade = {CascadeType.
    ALL})
        private List<PhotoModel> likes;

        @ManyToMany(fetch = FetchType.LAZY, cascade = {CascadeType.
    ALL})
        private List<PhotoModel> dislikes;

        private String likeString;
        private String dislikeString;

        @ManyToMany(fetch = FetchType.LAZY, cascade = {CascadeType.
    ALL})
        private List<PhotoModel> recommendation;

        private Integer recomIndex;

    }
```

VerificationToken

```
package com.project.socializingApp.model;
```

```
2
3  import jdk.jfr.Label;
4  import lombok.AllArgsConstructor;
5  import lombok.Data;
6  import lombok.NoArgsConstructor;
7
8  import javax.persistence.*;
9  import java.time.Instant;
10
11
12 @AllArgsConstructor
13 @NoArgsConstructor
14 @Data
15 @Entity
16 @Table(name = "token")
17 public class VerificationToken {
18     @Id
19     @GeneratedValue(strategy = GenerationType.IDENTITY)
20     private Long id;
21     private String token;
22
23     @OneToOne(fetch = FetchType.LAZY)
24     private User user;
25     private Instant expireDate;
26 }
```

FriendRepo

```
1  package com.project.socializingApp.repository;
2
3  import com.project.socializingApp.model.Friendship;
4  import com.project.socializingApp.model.User;
5  import org.springframework.data.jpa.repository.JpaRepository;
6  import org.springframework.data.repository.query.Param;
7
8  public interface FriendRepo  extends JpaRepository<Friendship, Long
       > {
9      void deleteByOwnerAndTarget(@Param("owner") User owner, @Param
       ("target") User target);
10 }
```

PhotoRepo

```
1  package com.project.socializingApp.repository;
2
3  import com.project.socializingApp.model.PhotoModel;
4  import org.springframework.data.jpa.repository.JpaRepository;
5  import org.springframework.stereotype.Repository;
6
7  import java.util.List;
8  import java.util.Set;
9
10 @Repository
11 public interface PhotoRepo extends JpaRepository<PhotoModel, Long>
       {
12     //Set<PhotoModel> findAllByUserUserIdContains(Long id);
13     List<PhotoModel> findAllByOrderByIdDesc();
14     List<PhotoModel> findAllByUser_UserId(Long id);
15 }
```

RefreshTokenRepo

```
package com.project.socializingApp.repository;

import com.project.socializingApp.model.RefreshToken;
import com.project.socializingApp.model.User;
import com.project.socializingApp.model.VerificationToken;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.Optional;
@Repository
public interface RefreshTokenRepo  extends JpaRepository<
    RefreshToken, Long> {
    Optional<RefreshToken> findByToken(String token);
    void deleteByToken(String token);
}
```

UserRepo

```
package com.project.socializingApp.repository;

import com.project.socializingApp.model.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.Optional;

@Repository
public interface UserRepo extends JpaRepository<User, Long> {
    Optional<User> findByUserName(String userName);
    User findFirstByUserName(String userName);
    void delete(User user);
}
```

VerificationTokenRepo

```
package com.project.socializingApp.repository;

import com.project.socializingApp.model.VerificationToken;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.Optional;

@Repository
public interface VerificationTokenRepo extends JpaRepository<
    VerificationToken, Long> {
    Optional<VerificationToken> findByToken(String
    verificationToken);
}
```

JwtFilter

```
package com.project.socializingApp.security;

import lombok.AllArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.authentication.
    UsernamePasswordAuthenticationToken;
```

```
import org.springframework.security.core.context.
    SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.
    UserDetailsService;
import org.springframework.security.web.authentication.
    WebAuthenticationDetailsSource;
import org.springframework.stereotype.Component;
import org.springframework.util.StringUtils;
import org.springframework.web.filter.OncePerRequestFilter;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@Component
@AllArgsConstructor
public class JwtFilter extends OncePerRequestFilter {

    private JwtProvider jwtProvider;
    private UserDetailsService userDetailsService;

    @Override
    protected void doFilterInternal(HttpServletRequest request,
                                    HttpServletResponse response,
                                    FilterChain filterChain) throws
     ServletException, IOException {
        String jwt = getJwtFromRequest(request);

        if (StringUtils.hasText(jwt) && jwtProvider.validateToken(
    jwt)) {
            String username = jwtProvider.getUsernameFromJwt(jwt);

            UserDetails userDetails = userDetailsService.
    loadUserByUsername(username);
            UsernamePasswordAuthenticationToken authentication =
    new UsernamePasswordAuthenticationToken(userDetails,
                    null, userDetails.getAuthorities());
            authentication.setDetails(new
    WebAuthenticationDetailsSource().buildDetails(request));

            SecurityContextHolder.getContext().setAuthentication(
    authentication);
        }
        filterChain.doFilter(request, response);
    }

    private String getJwtFromRequest(HttpServletRequest request) {
        String bearerToken = request.getHeader("Authorization");

        if (StringUtils.hasText(bearerToken) && bearerToken.
    startsWith("Bearer ")) {
            return bearerToken.substring(7);
        }
        return bearerToken;
```

```
    }
}
```

JwtProvider

```
package com.project.socializingApp.security;

import com.project.socializingApp.model.MyException;
import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.userdetails.User;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
import java.io.IOException;
import java.io.InputStream;
import java.security.*;
import java.security.cert.CertificateException;
import java.sql.Date;
import java.time.Instant;

import static io.jsonwebtoken.Jwts.parser;
import static java.util.Date.from;

@Service
public class JwtProvider {

    private KeyStore keyStore;
    @Value("${jwt.expiration.time}")
    private Long jwtExpirationInMillis = 100000000000L;

    @PostConstruct
    public void init() {
        try {
            keyStore = KeyStore.getInstance("JKS");
            InputStream resourceAsStream = getClass().
    getResourceAsStream("/springblog.jks");
            keyStore.load(resourceAsStream, "secret".toCharArray())
    ;
        } catch (KeyStoreException | CertificateException |
    NoSuchAlgorithmException | IOException e) {
            throw new MyException("Exception occurred while loading
     keystore", e);
        }

    }

    public String generateToken(Authentication authentication) {
        User principal = (User) authentication.getPrincipal();
        return Jwts.builder()
                .setSubject(principal.getUsername())
                .setIssuedAt(from(Instant.now()))
                .signWith(getPrivateKey())
                .setExpiration(Date.from(Instant.now().plusMillis(
    jwtExpirationInMillis)))
                .compact();
```

```
49      }
50
51      public String generateTokenWithUserName(String username) {
52          return Jwts.builder()
53                  .setSubject(username)
54                  .setIssuedAt(from(Instant.now()))
55                  .signWith(getPrivateKey())
56                  .setExpiration(Date.from(Instant.now().plusMillis(
        jwtExpirationInMillis)))
57                  .compact();
58      }
59
60      private PrivateKey getPrivateKey() {
61          try {
62              return (PrivateKey) keyStore.getKey("springblog", "
        secret".toCharArray());
63          } catch (KeyStoreException | NoSuchAlgorithmException |
        UnrecoverableKeyException e) {
64              throw new MyException("Exception occured while
        retrieving public key from keystore", e);
65          }
66      }
67
68      public boolean validateToken(String jwt) {
69          parser().setSigningKey(getPublickey()).parseClaimsJws(jwt);
70          return true;
71      }
72
73      private PublicKey getPublickey() {
74          try {
75              return keyStore.getCertificate("springblog").
        getPublicKey();
76          } catch (KeyStoreException e) {
77              throw new MyException("Exception occured while " +
78                      "retrieving public key from keystore", e);
79          }
80      }
81
82      public String getUsernameFromJwt(String token) {
83          Claims claims = parser()
84                  .setSigningKey(getPublickey())
85                  .parseClaimsJws(token)
86                  .getBody();
87
88          return claims.getSubject();
89      }
90
91      public Long getJwtExpirationInMillis() {
92          return jwtExpirationInMillis;
93      }
94  }
```

AutentificationService

```
1  package com.project.socializingApp.service;
2
3  import com.project.socializingApp.dataLayer.AuthResponse;
4  import com.project.socializingApp.dataLayer.Login;
```

```
5  import com.project.socializingApp.dataLayer.RefreshTokenData;
6  import com.project.socializingApp.dataLayer.RegisterData;
7  import com.project.socializingApp.model.*;
8  import com.project.socializingApp.repository.PhotoRepo;
9  import com.project.socializingApp.repository.UserRepo;
10 import com.project.socializingApp.repository.VerificationTokenRepo;
11 import com.project.socializingApp.security.JwtProvider;
12 import lombok.AllArgsConstructor;
13 import org.springframework.http.HttpStatus;
14 import org.springframework.security.authentication.
       AnonymousAuthenticationToken;
15 import org.springframework.security.authentication.
       AuthenticationManager;
16 import org.springframework.security.authentication.
       UsernamePasswordAuthenticationToken;
17 import org.springframework.security.core.Authentication;
18 import org.springframework.security.core.context.
       SecurityContextHolder;
19 import org.springframework.security.crypto.password.PasswordEncoder
       ;
20 import org.springframework.stereotype.Service;
21 import org.springframework.transaction.annotation.Transactional;
22 import org.springframework.security.crypto.password.PasswordEncoder
       ;
23 import org.springframework.security.crypto.bcrypt.
       BCryptPasswordEncoder;
24 import org.springframework.web.client.HttpClientErrorException;
25
26
27 import java.time.Instant;
28 import java.util.*;
29
30 @AllArgsConstructor
31 @Service
32 public class AutentificationService {
33
34     private final BCryptPasswordEncoder encoder;
35     private final UserRepo userRepo;
36     private final VerificationTokenRepo verificationTokenRepo;
37     private final AuthenticationManager authenticationManager;
38     private final JwtProvider jwtProvider;
39     private final RefreshTokenService refreshTokenService;
40     private final PhotoRepo photoRepo;
41
42     @Transactional
43     public void signUp (RegisterData registerData ) throws
       Exception {
44         try{
45             userRepo.findByUserName(registerData.getUserName());
46         }catch ( Exception ex){
47             throw new Exception("Not ok");
48         }
49         User user = new User();
50         user.setUserName(registerData.getUserName());
51         user.setEmail(registerData.getEmail());
52         user.setPassword(encoder.encode(registerData.getPassword())
       );
```

```
53          user.setCreated(Instant.now());
54          user.setEnabled(true); // for email verification set it to
        false
55          user.setRecommendation(photoRepo.findAllByOrderByIdDesc());
56          user.setRecomIndex(0);
57          user.setFriends(new ArrayList<>());
58          userRepo.save(user);
59          generateVerification(user);
60      }
61
62      private String generateVerification(User user) {
63          String token = UUID.randomUUID().toString();
64          VerificationToken verificationToken = new VerificationToken
        ();
65          verificationToken.setToken(token);
66          verificationToken.setUser(user);
67
68          verificationTokenRepo.save(verificationToken);
69          return token;
70      }
71
72      private void fetchUserAndEnable(VerificationToken
        verificationToken) {
73          String username = verificationToken.getUser().getUserName()
        ;
74          User user = userRepo.findByUserName(username).orElseThrow
        (() -> new MyException("User not found with name - " + username
        ));
75          user.setEnabled(true);
76          userRepo.save(user);
77      }
78
79      public void verifyAccount(String token) {
80          Optional<VerificationToken> verificationToken =
        verificationTokenRepo.findByToken(token);
81          fetchUserAndEnable(verificationToken.orElseThrow(() -> new
        MyException("Invalid Token")));
82      }
83
84      public AuthResponse login(Login loginRequest) {
85          Optional<User> user = userRepo.findByUserName(loginRequest.
        getUserName());
86          Authentication authenticate = authenticationManager.
        authenticate(new UsernamePasswordAuthenticationToken(
        loginRequest.getUserName(),
87                  loginRequest.getPassword()));
88          SecurityContextHolder.getContext().setAuthentication(
        authenticate);
89          String token = jwtProvider.generateToken(authenticate);
90          return AuthResponse.builder()
91                  .authenticationToken(token)
92                  .refreshToken(refreshTokenService.
        generateRefreshToken().getToken())
93                  .expiresAt(Instant.now().plusMillis(jwtProvider.
        getJwtExpirationInMillis()))
94                  .userName(loginRequest.getUserName())
95                  .email(user.get().getEmail())
```

```
                .build();
    }

    public AuthResponse refreshToken(RefreshTokenData
    refreshTokenData) {
        refreshTokenService.validateRefreshToken(refreshTokenData.
    getRefreshToken());
        String token = jwtProvider.generateTokenWithUserName(
    refreshTokenData.getUserName());
        return AuthResponse.builder()
                .authenticationToken(token)
                .refreshToken(refreshTokenData.getRefreshToken())
                .expiresAt(Instant.now().plusMillis(jwtProvider.
    getJwtExpirationInMillis()))
                .userName(refreshTokenData.getUserName())
                .build();
    }

    public boolean isLoggedIn() {
        Authentication authentication = SecurityContextHolder.
    getContext().getAuthentication();
        return !(authentication instanceof
    AnonymousAuthenticationToken) && authentication.isAuthenticated
    ();
    }
}
```

PhotoService

```
package com.project.socializingApp.service;

import com.project.socializingApp.dataLayer.RequestPhoto;
import com.project.socializingApp.model.PhotoModel;
import com.project.socializingApp.model.User;
import com.project.socializingApp.repository.PhotoRepo;
import com.project.socializingApp.repository.UserRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import recom.RecommendationSystem;

import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

@Service
public class PhotoService {

    private PhotoRepo photoRepo;
    private UserRepo userRepo;

    @Autowired
    public PhotoService(PhotoRepo photoRepo, UserRepo userRepo) {
        this.photoRepo = photoRepo;
        this.userRepo = userRepo;
    }
```

```
30      @Transactional
31      public PhotoModel addPhoto(RequestPhoto requestPhoto){
32          PhotoModel photoModel = new PhotoModel();
33          User user = userRepo.findByUserName(requestPhoto.
        getUsername()).orElseThrow();
34          photoModel.setUser(user);
35          photoModel.setPicture(requestPhoto.getImage());
36          photoModel.setDescription(requestPhoto.getDescription());
37          photoModel.setLikes(new ArrayList<>());
38          photoModel.setDislikes(new ArrayList<>());
39          for (User user1: userRepo.findAll()) {
40              if(!user1.equals(user)) {
41                  user1.getRecommendation().add(photoModel);
42                  refreshRecom(user1);
43              }
44              //userRepo.save(user1);
45          }
46          //photoRepo.save(photoModel);
47          return photoModel;
48      }
49      public List<PhotoModel> set(){
50          List<PhotoModel> photoModels = new ArrayList<>(photoRepo.
        findAll());
51          for (PhotoModel p:photoModels) {
52              p.setPicture(new byte[]{1});
53          }
54          return photoModels;
55      }
56
57      public List<PhotoModel> setByUsername(String username){
58          return photoRepo.findAllByUser_UserId(userRepo.
        findByUserName(username).orElseThrow().getUserId());
59      }
60      @Transactional
61      public PhotoModel updateLike(Long id, String username){
62          PhotoModel photoModel = photoRepo.getOne(id);
63          User user = userRepo.findByUserName(username).orElseThrow()
        ;
64          List<User> users = photoModel.getLikes();
65          if(users.contains(user)){
66              users.remove(user);
67              user.getLikes().remove(photoModel);
68          }else {
69              users.add(user);
70              user.getLikes().add(photoModel);
71          }
72          refreshRecom(user);
73          //photoRepo.save(photoModel);
74          return photoModel;
75      }
76      @Transactional
77      public PhotoModel updateDislikeAndLike(Long id, String username
        ){
78          PhotoModel photoModel = photoRepo.getOne(id);
79          User user = userRepo.findByUserName(username).orElseThrow()
        ;
80          List<User> users = photoModel.getLikes();
```

```
81          List<User> users1 = photoModel.getDislikes();
82          if(users.contains(user)){
83              users.remove(user);
84              user.getLikes().remove(photoModel);
85          }else {
86              users.add(user);
87              user.getLikes().add(photoModel);
88          }
89          if(users1.contains(user)){
90              users1.remove(user);
91              user.getDislikes().remove(photoModel);
92          }else {
93              users1.add(user);
94              user.getDislikes().add(photoModel);
95          }
96          refreshRecom(user);
97          //photoRepo.save(photoModel);
98          return photoModel;
99      }
100
101
102
103     @Transactional
104     public PhotoModel updateDislike(Long id, String username){
105         PhotoModel photoModel = photoRepo.getOne(id);
106         User user = userRepo.findByUserName(username).orElseThrow()
    ;
107         List<User> users = photoModel.getDislikes();
108         if(users.contains(user)){
109             users.remove(user);
110             user.getDislikes().remove(photoModel);
111         }else {
112             users.add(user);
113             user.getDislikes().add(photoModel);
114         }
115         refreshRecom(user);
116         //photoRepo.save(photoModel);
117         return photoModel;
118     }
119
120     public boolean getLike(Long id, String username){
121         PhotoModel photoModel = photoRepo.getOne(id);
122         User user = userRepo.findByUserName(username).orElseThrow()
    ;
123         List<User> users = photoModel.getLikes();
124         if(users.contains(user)){
125             return true;
126         }else {
127             return false;
128         }
129     }
130
131     public boolean getDislike(Long id, String username){
132         PhotoModel photoModel = photoRepo.getOne(id);
133         User user = userRepo.findByUserName(username).orElseThrow()
    ;
134         List<User> users = photoModel.getDislikes();
```

```
135         if(users.contains(user)){
136             return true;
137         }else {
138             return false;
139         }
140     }
141 
142     public PhotoModel getById(Long id){
143         return photoRepo.getOne(id);
144     }
145 
146     public PhotoModel delete(Long id){
147         PhotoModel photoModel = photoRepo.getOne(id);
148         if(photoModel != null){
149             photoRepo.delete(photoModel);
150         }
151         return photoModel;
152     }
153 
154     public byte[] getPhoto(Long id){
155         return photoRepo.getOne(id).getPicture();
156     }
157 
158     public Integer getLikes(Long id){
159         return photoRepo.getOne(id).getLikes().size();
160     }
161 
162     public Integer getDislikes(Long id){
163         return photoRepo.getOne(id).getDislikes().size();
164     }
165 
166     public String getUsername(Long id){
167         return photoRepo.getOne(id).getUser().getUserName();
168     }
169 
170     private void refreshRecom(User user) {
171         RecommendationSystem recommendationSystem = new
    RecommendationSystem();
172         try {
173             user.setRecommendation(recommendationSystem.
    recommendations(user.getRecomIndex()
174                     ,user.getRecommendation()
175                     ,user.getLikes()
176                     ,user.getDislikes()));
177         } catch (FileNotFoundException e) {
178             e.printStackTrace();
179         }
180     }
181 
182 }
```

RefreshTokenService

```
1 package com.project.socializingApp.service;
2 
3 import com.project.socializingApp.model.MyException;
4 import com.project.socializingApp.model.RefreshToken;
5 import com.project.socializingApp.repository.RefreshTokenRepo;
```

```
import lombok.AllArgsConstructor;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.time.Instant;
import java.util.UUID;

@Service
@AllArgsConstructor
@Transactional
public class RefreshTokenService {

    private final RefreshTokenRepo refreshTokenRepo;

    public RefreshToken generateRefreshToken() {
        RefreshToken refreshToken = new RefreshToken();
        refreshToken.setToken(UUID.randomUUID().toString());
        refreshToken.setCreatedDate(Instant.now());

        return refreshTokenRepo.save(refreshToken);
    }

    void validateRefreshToken(String token) {
        refreshTokenRepo.findByToken(token)
                .orElseThrow(() -> new MyException("Invalid refresh
 Token"));
    }

    public void deleteRefreshToken(String token) {
        refreshTokenRepo.deleteByToken(token);
    }
}
```

UserDetService

```
package com.project.socializingApp.service;

import com.project.socializingApp.model.Friendship;
import com.project.socializingApp.model.PhotoModel;
import com.project.socializingApp.model.User;
import com.project.socializingApp.repository.FriendRepo;
import com.project.socializingApp.repository.PhotoRepo;
import com.project.socializingApp.repository.UserRepo;
import lombok.AllArgsConstructor;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.
    SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.
    UserDetailsService;
import org.springframework.security.core.userdetails.
    UsernameNotFoundException;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.Collection;
import java.util.List;
import java.util.Optional;
```

```
21 import java.util.stream.Collectors;
22
23 import static java.util.Collections.singletonList;
24 @AllArgsConstructor
25 @Service
26 public class UserDetService implements UserDetailsService {
27     private final UserRepo userRepo;
28     private final PhotoRepo photoRepo;
29     private final FriendRepo friendRepo;
30
31     @Override
32     @Transactional(readOnly = true)
33     public UserDetails loadUserByUsername(String username) {
34         Optional<User> userOptional = userRepo.findByUserName(
       username);
35         User user = userOptional
36                 .orElseThrow(() -> new UsernameNotFoundException("
       No user " +
37                         "Found with username : " + username));
38
39         return new org.springframework.security
40                 .core.userdetails.User(user.getUserName(), user.
       getPassword(),
41                 user.isEnabled(), true, true,
42                 true, getAuthorities("USER"));
43     }
44
45     private Collection<? extends GrantedAuthority> getAuthorities(
       String role) {
46         return singletonList(new SimpleGrantedAuthority(role));
47     }
48
49     public void updateUserDetails(User user)
50     {
51         userRepo.save(user);
52     }
53
54     public void deleteUserAccount(User user)
55     {
56         user.setEnabled(false);
57         userRepo.save(user);
58     }
59
60     public User getUserDetails(String userName)
61     {
62        return userRepo.findByUserName(userName).get();
63     }
64
65     public List<PhotoModel> recommend(String name){
66         return userRepo.findByUserName(name).orElseThrow().
       getRecommendation();
67     }
68     @Transactional
69     public List<PhotoModel> initRecommend(String name){
70         User user = userRepo.findByUserName(name).orElseThrow();
71         List<PhotoModel> set = photoRepo.findAll();
72         for (PhotoModel photoModel:set) {
```

```
73              if(!user.getRecommendation().contains(photoModel)){
74                  user.getRecommendation().add(photoModel);
75              }
76          }
77
78          return user.getRecommendation();
79      }
80
81      public Integer getRecomIndex(String username){
82          return userRepo.findByUserName(username).orElseThrow().
    getRecomIndex();
83      }
84
85      @Transactional
86      public Integer setRecomIndex(String username, Integer value){
87          User user = userRepo.findByUserName(username).orElseThrow()
    ;
88          user.setRecomIndex(value);
89          return user.getRecomIndex();
90      }
91
92      public List<User> getFollow(String username){
93          return friendRepo.findAll().stream()
94                  .filter(e -> e.getOwner().getUserName().equals(
    username))
95                  .map(Friendship::getTarget)
96                  .collect(Collectors.toList());
97      }
98      @Transactional
99      public List<User> setFollow(String username, String follow){
100         User user = userRepo.findByUserName(username).orElseThrow()
    ;
101         User followed = userRepo.findByUserName(follow).orElseThrow
    ();
102         List<User> friends = friendRepo.findAll().stream()
103                 .filter(e -> e.getOwner().getUserName().equals(
    username))
104                 .map(Friendship::getTarget)
105                 .collect(Collectors.toList());
106         Friendship friendship = new Friendship();
107         friendship.setOwner(user);
108         friendship.setTarget(followed);
109         if(friends.contains(followed)){
110             friends.remove(followed);
111             friendRepo.deleteByOwnerAndTarget(user,followed);
112         }else{
113             friends.add(followed);
114            friendRepo.save(friendship);
115         }
116
117         return friends;
118     }
119
120     public boolean isFollow(String username, String follow){
121         User followed = userRepo.findByUserName(follow).orElseThrow
    ();
122         return friendRepo.findAll().stream()
```

```
123                 .filter(e -> e.getOwner().getUserName().equals(
    username))
124                 .anyMatch(e -> e.getTarget().getUserName().equals(
    follow));
125     }
126
127     public List<String> listAllUsers(){
128         return userRepo.findAll().stream()
129                 .filter(User::isEnabled)
130                 .map(User::getUserName)
131                 .collect(Collectors.toList());
132     }
133 }
```

SocializingAppApplication

```
1 package com.project.socializingApp;
2
3 import com.project.socializingApp.configuration.SwagConfig;
4 import org.springframework.boot.SpringApplication;
5 import org.springframework.boot.autoconfigure.SpringBootApplication
    ;
6 import org.springframework.context.annotation.Import;
7 import org.springframework.scheduling.annotation.EnableAsync;
8
9 @SpringBootApplication
10 @EnableAsync
11 public class SocializingAppApplication {
12
13   public static void main(String[] args) {
14     SpringApplication.run(SocializingAppApplication.class, args);
15   }
16
17 }
```

4 Class Diagrams

4.1 Frontend

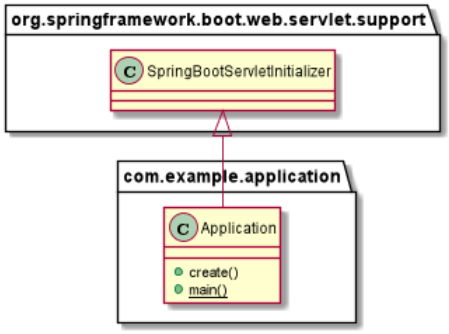

Figure 3: Application UML

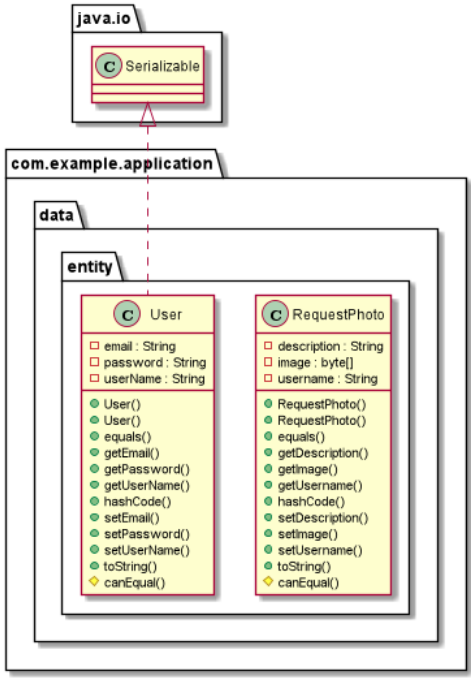

Figure 4: entity UML

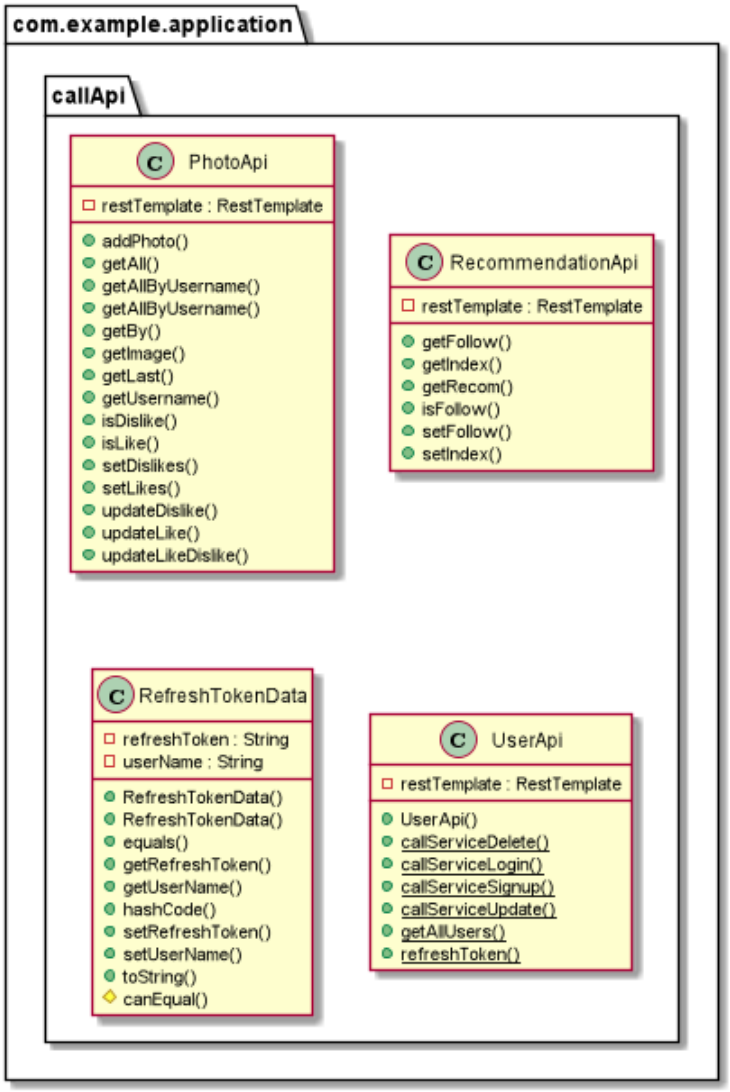

Figure 5: callApi UML

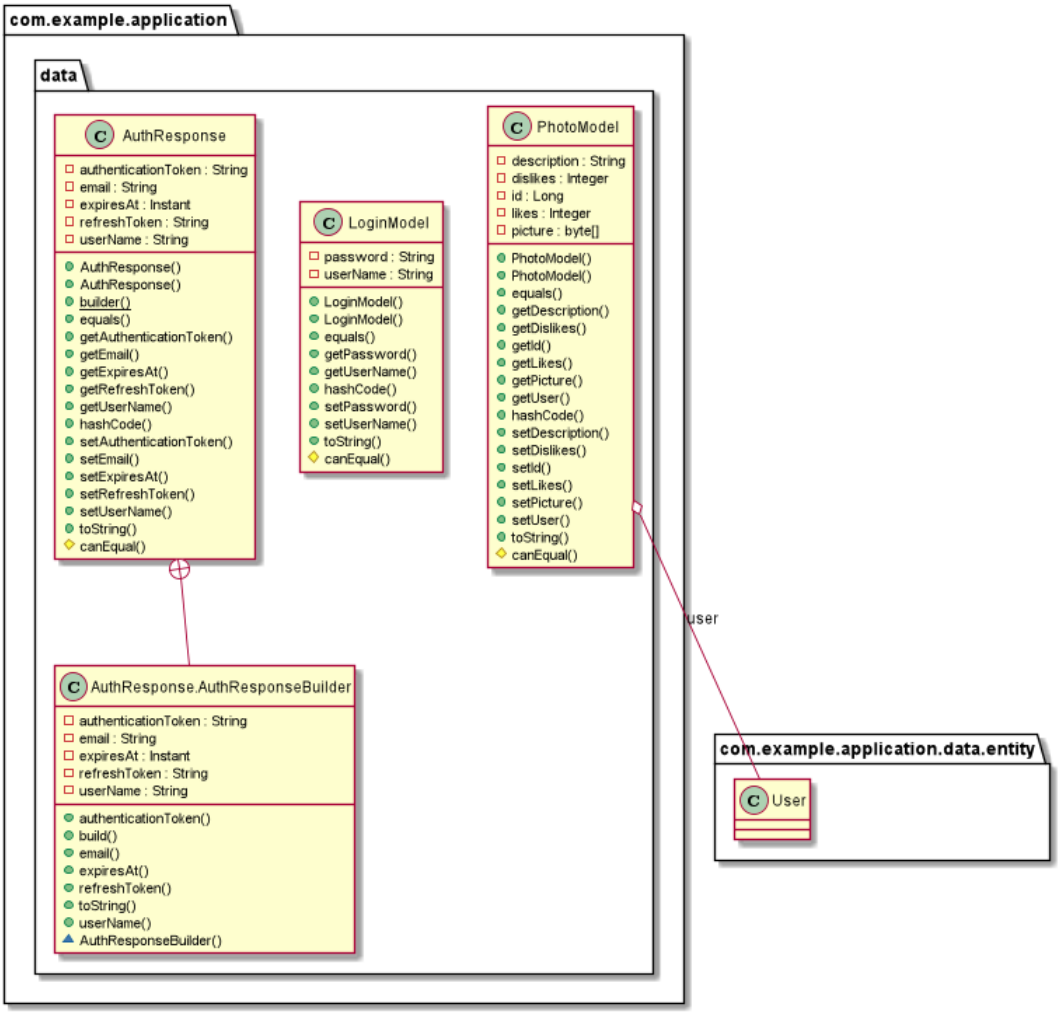

Figure 6: Data UML

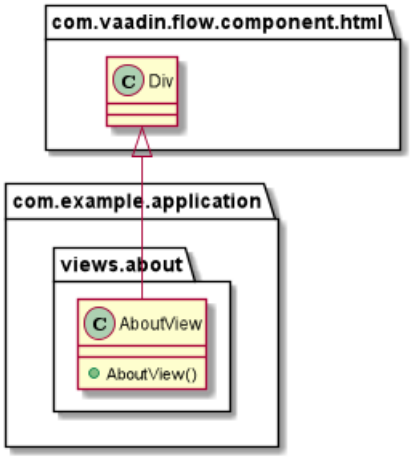

Figure 7: views.about UML

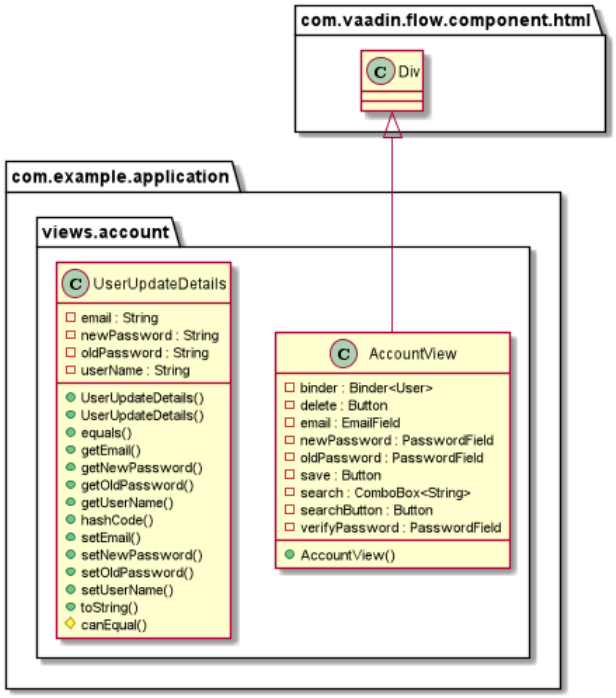

Figure 8: views.account UML

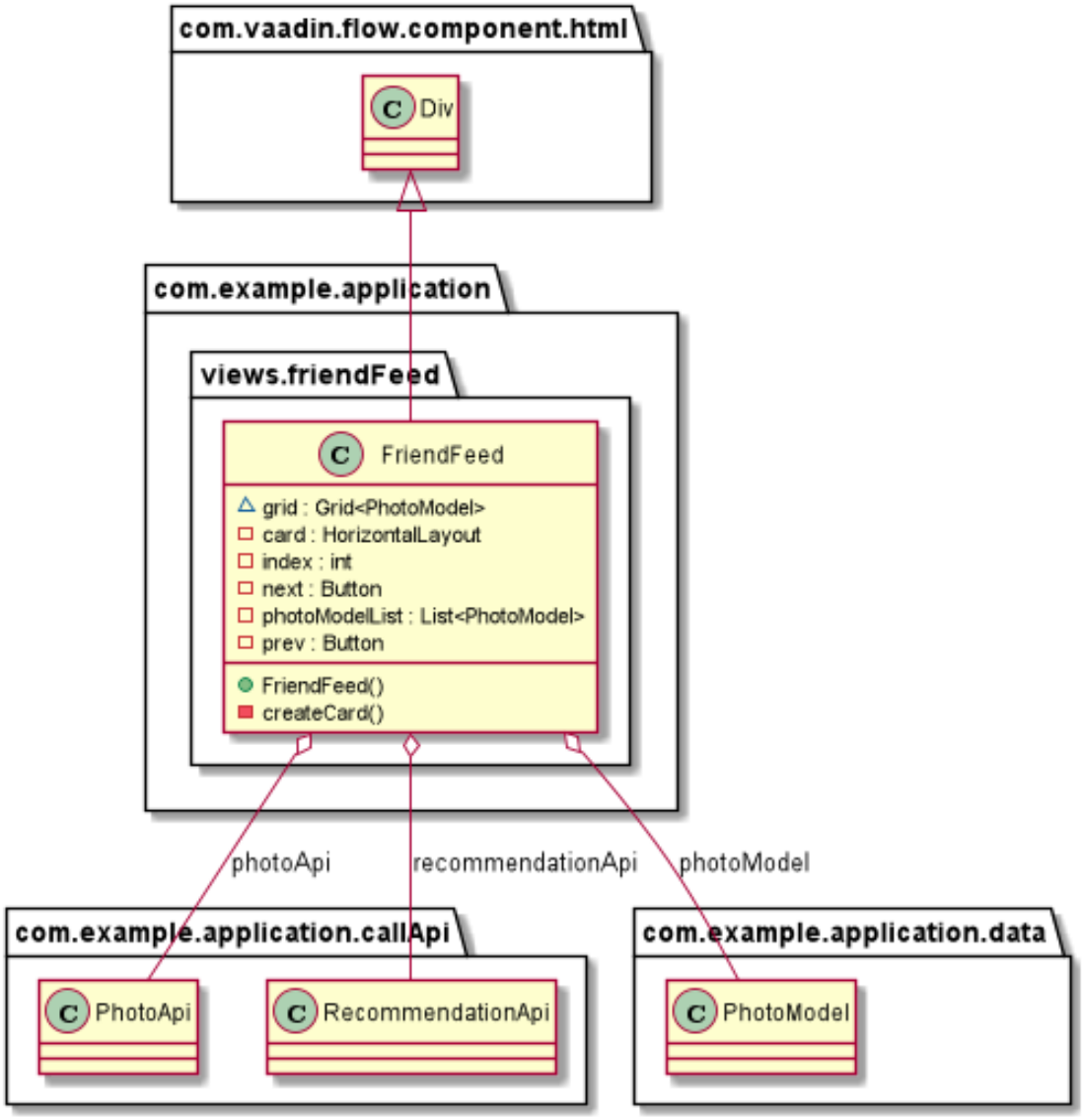

Figure 9: views.friendFeed UML

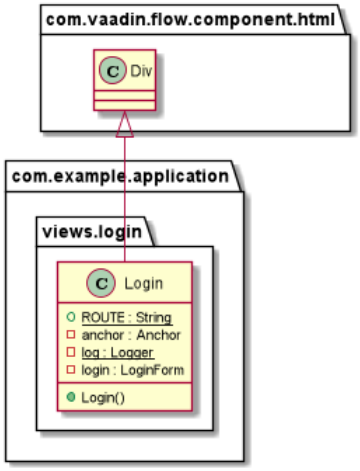

Figure 10: views.login UML

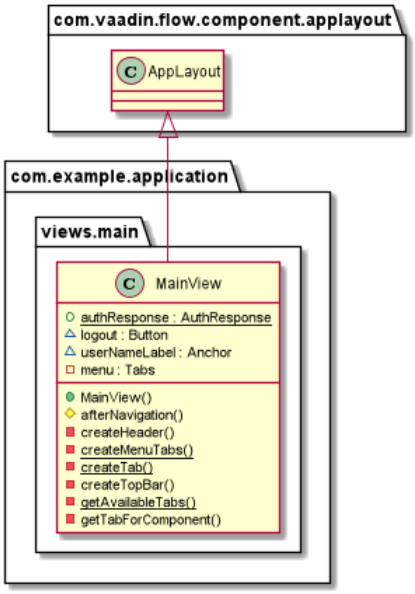

Figure 11: views.main UML

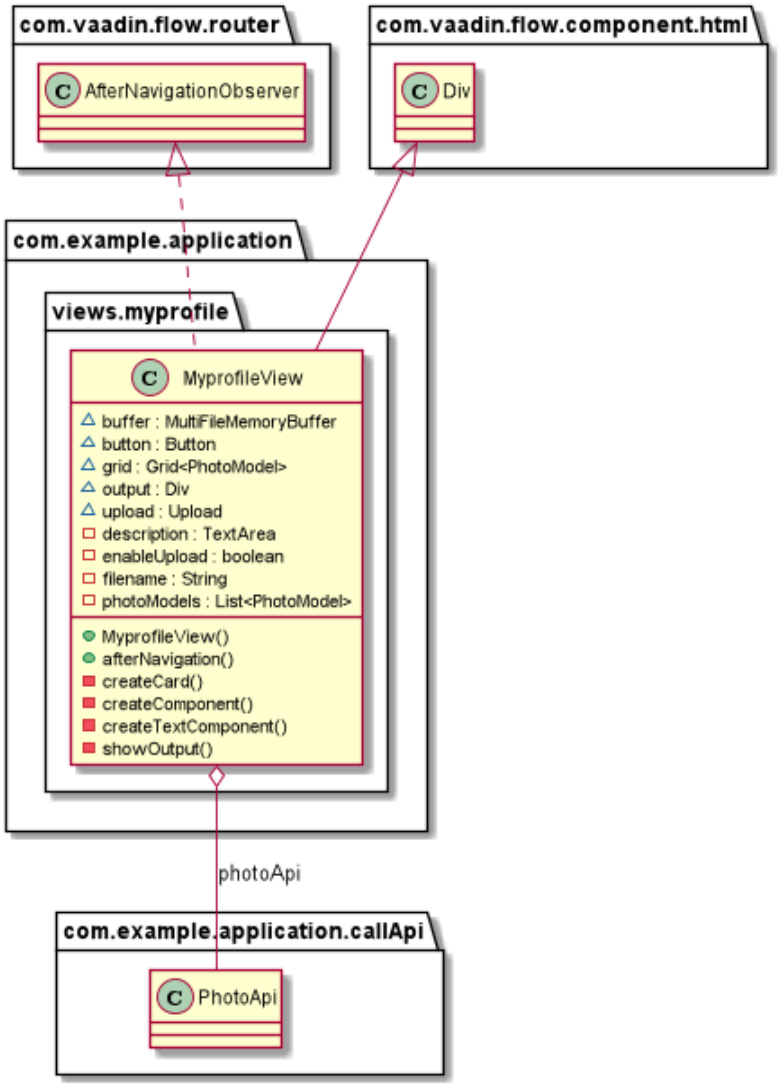

Figure 12: views.myprofile UML

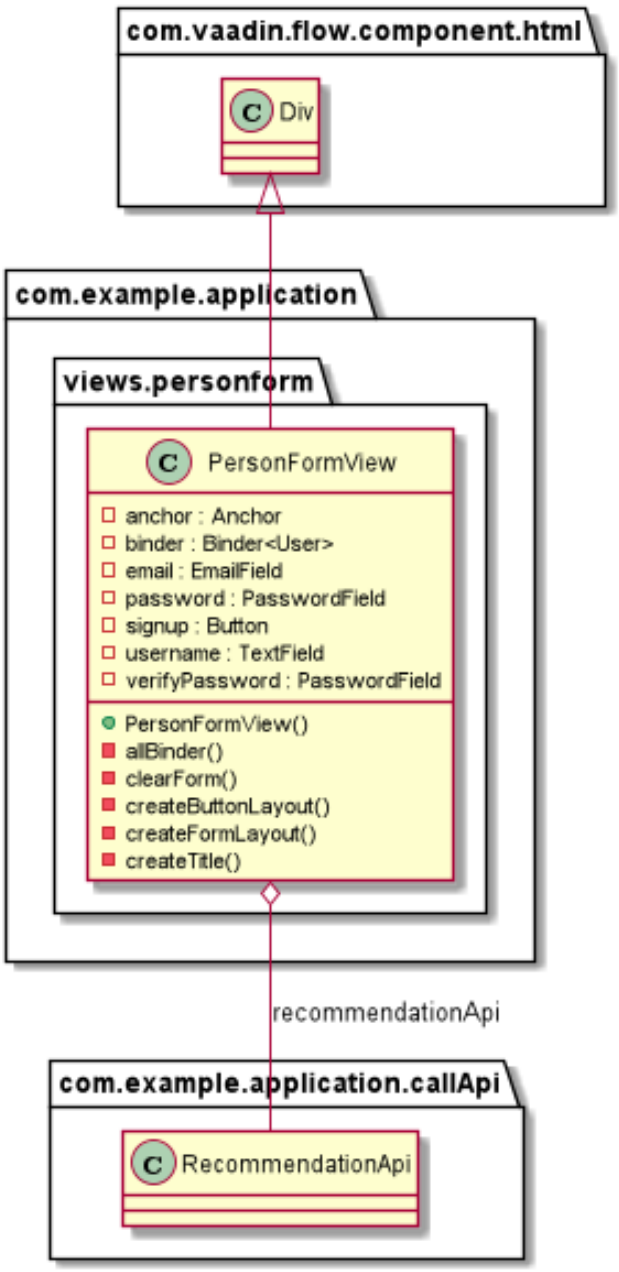

Figure 13: views.personform UML

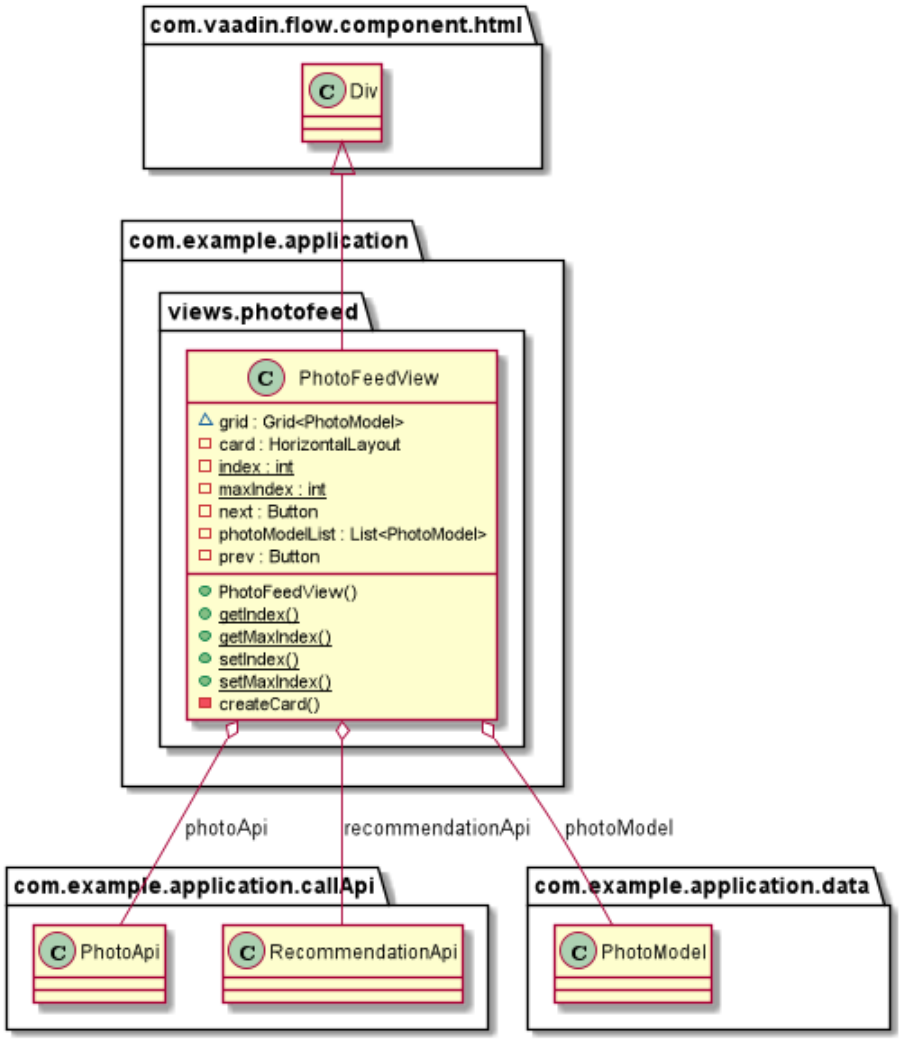

Figure 14: views.photofeed UML

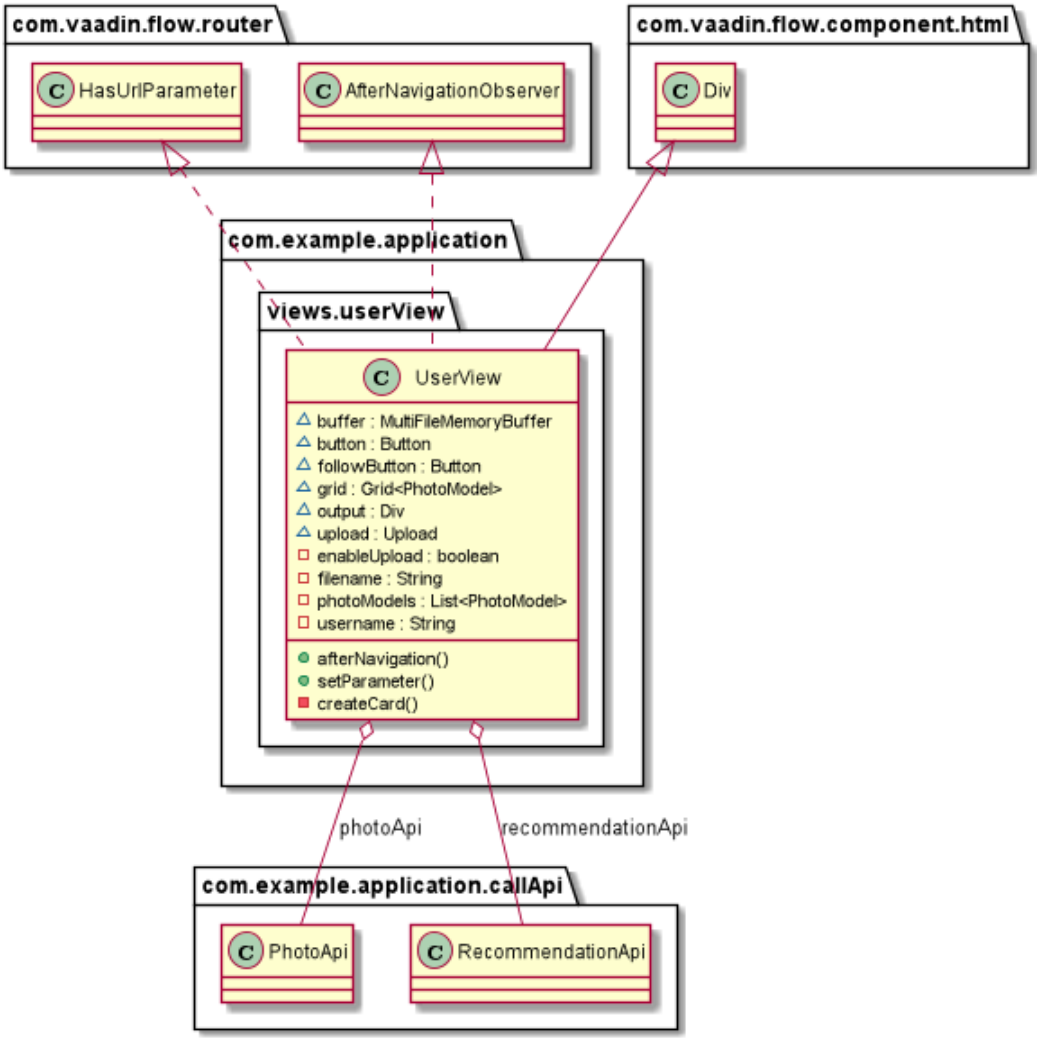

Figure 15: views.userView UML

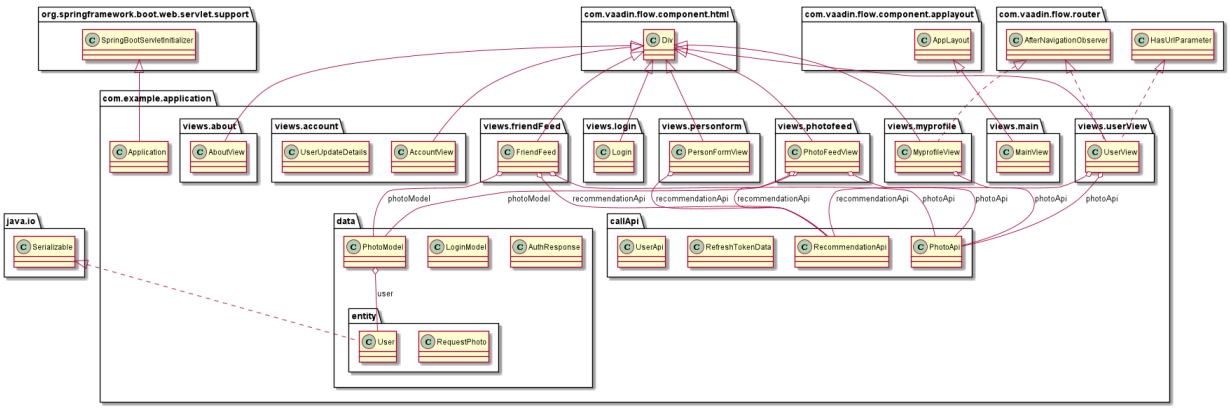

Figure 16: Frontend UML

4.2 Backend

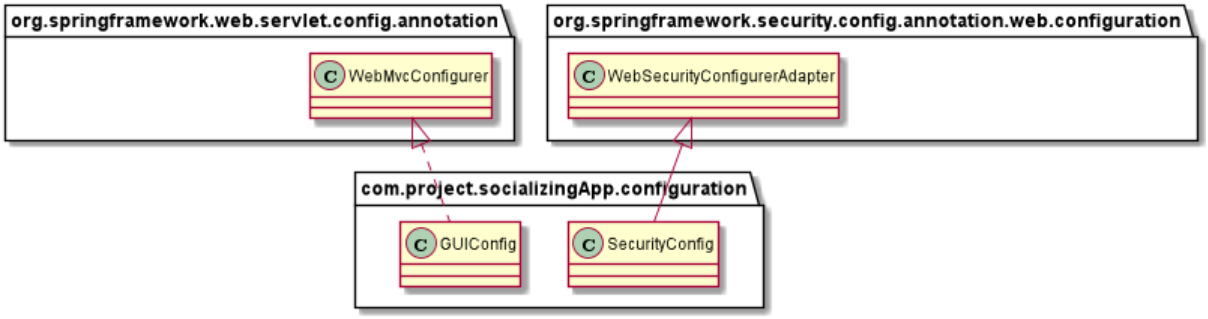

Figure 17: Configuration UML

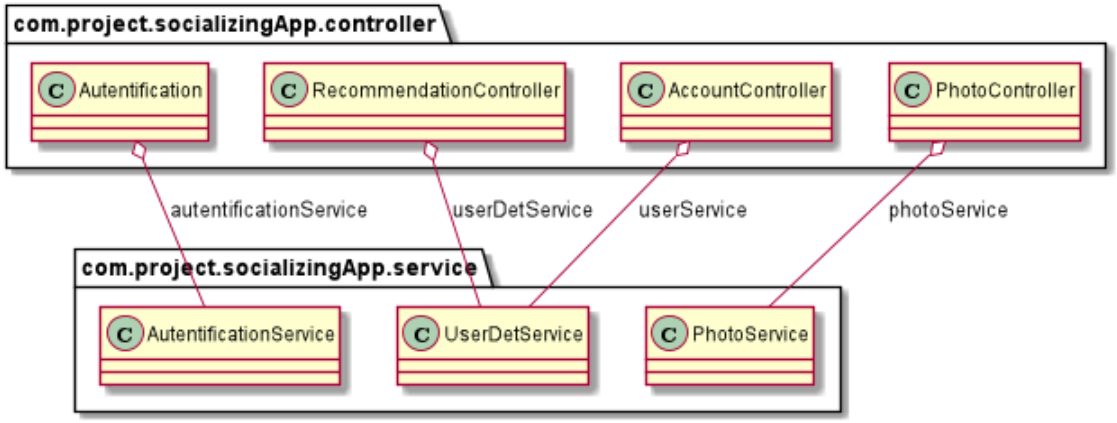

Figure 18: Controller UML

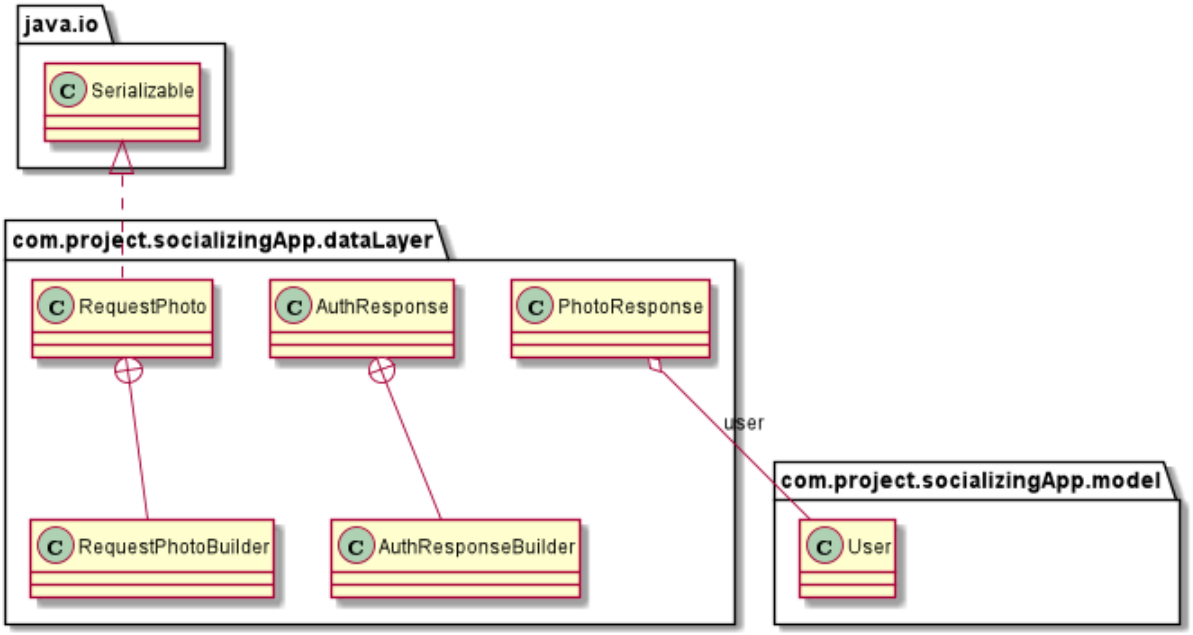

Figure 19: dataLayer UML

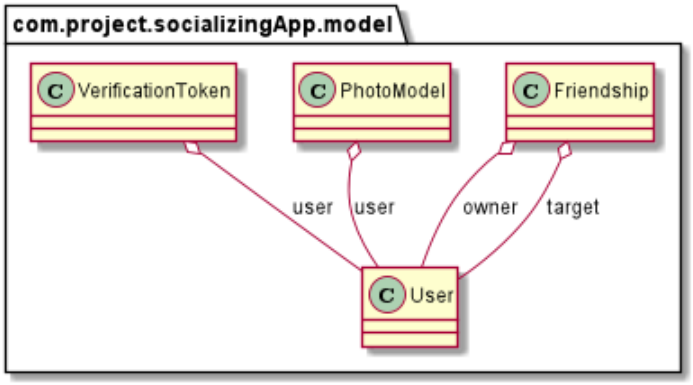

Figure 20: Model UML

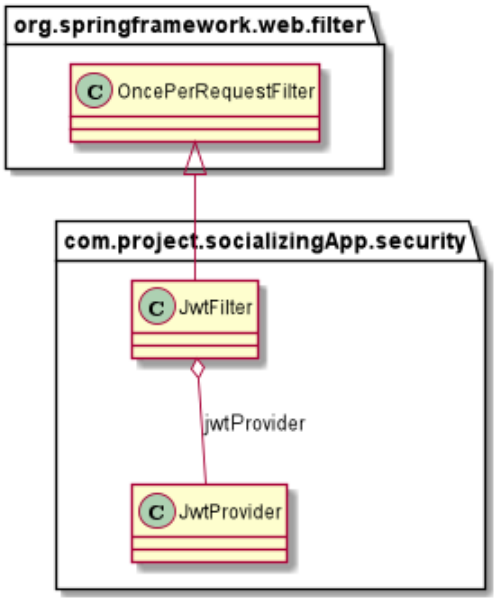

Figure 21: Security UML

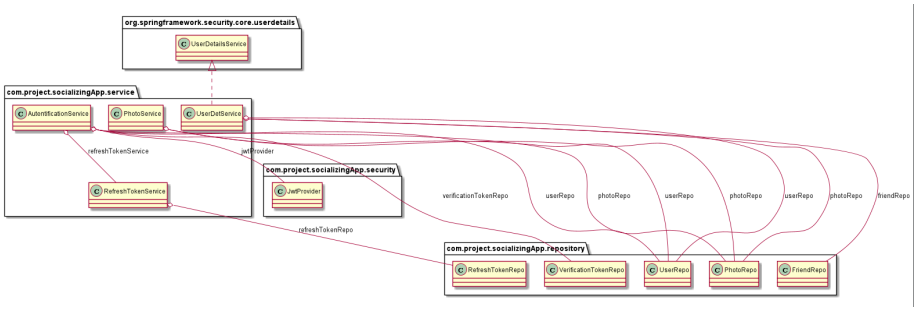

Figure 22: Service UML

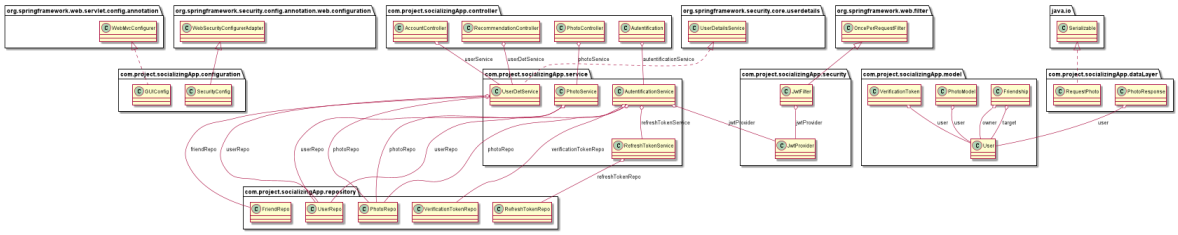

Figure 23: Backend UML

5 References

1. Tf-Idf and Cosine similarity [Online]. Available: https://janav.wordpress.com/2013/10/27/tf-idf-and-cosine-similarity/

2. Content-Based Filtering model [Online]. Available: https://github.com/youonf/recommendation_system/blob/master/content_based_filtering/content_based_recommender_approach1.ipynb

3. Spring Web Annotations [Online]. Available: https://www.baeldung.com/spring-mvc-annotations

4. Vaadin Framework [Online]. Available: https://vaadin.com/docs/v14/flow/Overview.html

5. Consuming Rest Web Service using Spring Boot [Online]. Available: https://www.youtube.com/watch?v=UP-v1dTOT9gab_channel=ChargeAhead