JAVA

1. Pachetul com.tourer

- App

```
package com.tourer;


import com.tourer.gui.map.Location;
import java.io.IOException;
import java.lang.reflect.InvocationTargetException;
import java.io.File;
import java.util.Vector;
import java.util.concurrent.FutureTask;
import java.util.concurrent.Semaphore;
import java.awt.Toolkit;
import java.awt.Image;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.awt.image.BufferedImage;

import java.security.GeneralSecurityException;
import java.sql.SQLException;

import javax.imageio.ImageIO;

import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPasswordField;
import javax.swing.JTextField;

import com.tourer.gui.*;
```

```java
// import com.tourer.gui.map.GoogleMaps;
import com.tourer.jdbc.*;
import com.tourer.jdbc.JdbcRunner;

import java.awt.CardLayout;


import javafx.application.Platform;
import javax.swing.SpringLayout;
import javax.swing.SwingUtilities;
import javax.swing.UIManager;

import javafx.embed.swing.JFXPanel;
import javafx.scene.Scene;
import javafx.scene.web.WebEngine;
import javafx.scene.web.WebView;
import javafx.stage.Stage;
import javafx.application.Application;
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLContext;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;

import netscape.javascript.JSObject;

//"vmArgs": "--module-path C:\\Java\\JavaFX\\javafx-sdk-17.0.1\\lib --add-modules
javafx.controls,javafx.fxml",

public class App extends Application
{
    public static AccountCreationFrame accountCreationFrame;
    public static GradientColor gradientColor = new GradientColor(new Color(38, 0,
110), AppSettingsMenu.PURPLE_COLOR);
    static String logoPath = "C:\\Java\\PI\\demo\\Icons\\Logo.jpg";
    static String errorPath = "C:\\Java\\PI\\demo\\Icons\\Error.png";
    static String questionPath = "C:\\Java\\PI\\demo\\Icons\\Question.png";
    static final ImageIcon errorIcon = new ImageIcon(new
ImageIcon(errorPath).getImage().getScaledInstance(50, 50, Image.SCALE_SMOOTH));
    static final ImageIcon questionIcon = new ImageIcon(new
ImageIcon(questionPath).getImage().getScaledInstance(50, 50, Image.SCALE_SMOOTH));
    public static WebEngine engine;
    public static WebView view;
    public static MainFrame mainFrame = new MainFrame(gradientColor);
    public static Double currentLatitude;
    public static Double currentLongitude;
```

```
    public static JFXPanel fxpanel;
    static{
        mainFrame.setVisible(false);
        UIManager.put("OptionPane.errorIcon", errorIcon);
        UIManager.put("OptionPane.questionIcon", questionIcon);
        UIManager.put("OptionPane.background",
gradientColor.getSecondaryColor());
        UIManager.put("Panel.background", gradientColor.getSecondaryColor());
        UIManager.put("Button.background", Color.orange);
        UIManager.put("Button.foreground", Color.white);
        UIManager.put("OptionPane.messageForeground", Color.orange);
    }

    public static boolean checkUserExistance(String username, String password){


        if(username.equals(""))
            return false;

        if(password.equals(""))
            return false;

        try {
            return Connector.checkUserExistance(username, password);
        } catch (SQLException e) {
            return false;

        }

    }

    public static ButtonBox buttonBox;
    public static void initAndShowGUI(){

        Semaphore mutex1 = new Semaphore(0);
        Semaphore mutex = new Semaphore(1);
        runAndWait(new Runnable() {
            @Override
            public void run()
                {
                    try {
                        mutex.acquire();
                        fxpanel = new JFXPanel();
                        view = new WebView();
```

```
                        engine = view.getEngine();
                        engine.getCreatePopupHandler();
                        engine.setJavaScriptEnabled(true);
                        engine.setUserAgent("Mozilla/5.0 (Windows NT 6.2; WOW64)
AppleWebKit/537.36 Chrome/44.0.2403.155 Safari/537.36");
                        Scene scene = new Scene(view);
                        fxpanel.setScene(scene);
                        engine.load("file:///C:\\Java\\PI\\demo\\JavaScript\\test
.html");

                        fxpanel.revalidate();
                        fxpanel.revalidate();
                        mutex.release();
                        mutex1.release();
                    } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }




                }


        });
        try {
            SwingUtilities.invokeAndWait(new Runnable() {
                @Override
                public void run() {


                    try {
                        mutex1.acquire();
                        mutex.acquire();
                        mainFrame.getContentPane().add(fxpanel,
BorderLayout.CENTER);
                        Dimension buttonMenuDim = new Dimension(150,
MainFrame.screenSize.height - 50);


                        ColorPanel menu = new ColorPanel();
                        menu.setSize(buttonMenuDim);
                        menu.setPreferredSize(buttonMenuDim);
                        menu.setMaximumSize(buttonMenuDim);
```

```
                        menu.setMinimumSize(buttonMenuDim);
                        ColorPanel searchPanel = new ColorPanel();
                        searchPanel.setLayout(new CardLayout());
                        LocationSearchField locationSearchField = new
LocationSearchField();

                        UserSearchField userSearchField = new UserSearchField();

                        searchPanel.setSize(new
Dimension(MainFrame.screenSize.width, 50));

                        searchPanel.add(userSearchField);
                        searchPanel.add(locationSearchField);
                        buttonBox = new ButtonBox(locationSearchField,
userSearchField);



                        menu.add(buttonBox);
                        mainFrame.add(searchPanel, BorderLayout.NORTH);
                        mainFrame.add(menu, BorderLayout.WEST);

                        mainFrame.pack();
                        mainFrame.revalidate();
                        mainFrame.repaint();
                        mutex.release();
                    } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }


                }
            });
        } catch (InvocationTargetException | InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }
    public static void reloadMap(){

        runAndWait(new Runnable() {

            @Override
            public void run() {
```

```
                App.engine.reload();

                try {
                    Vector <Location> vLocation = Connector.getVisitedLocations();

                    for(int i = 0; i < min(10, vLocation.size()); i++){
                        Double lat = vLocation.get(i).getLatitude();
                        Double lng = vLocation.get(i).getLongitude();

                        engine.executeScript("addMarkerToList(" + lat + ", " + lng
+ ");");
                    }
                } catch (SQLException e1) {
                    // TODO Auto-generated catch block
                    e1.printStackTrace();
                }


                App.engine.executeScript("initMap();");
                // updateCurrentPosition();
            }
        });


    }
    private static int min(int i, int j) {
        if(i < j)
            return i;
        else
            return j;
    }
    public static void main( String[] args ) throws IOException{

        try{
            JdbcRunner.buildJdbc();
            System.out.println("Successfully conected to server");
        }catch(SQLException e1){
            e1.printStackTrace();
            System.exit(1);
        }
        String backgroundPath = "Icons\\LoginBackground.jpg";
        Image background = Toolkit.getDefaultToolkit().getImage(backgroundPath);
        try {
            BufferedImage bufferedImage = ImageIO.read(new File(backgroundPath));
```

```
            background = bufferedImage.getScaledInstance((int)
MainFrame.screenSize.width * 3 / 4 ,(int) MainFrame.screenSize.height * 3 / 4,
Image.SCALE_SMOOTH);
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        JFrame loginFrame = new JFrame();
        loginFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        SpringLayout springLayout = new SpringLayout();
        BackGroundPanel contentPane = new BackGroundPanel(background);

        contentPane.setLayout(springLayout);
        loginFrame.setSize(new Dimension((MainFrame.screenSize.width * 3) / 4 ,
(MainFrame.screenSize.height * 3) / 4));
        loginFrame.setContentPane(contentPane);


        JLabel label = new JLabel("UserName");
        label.setForeground(Color.ORANGE);
        label.setFont(new Font(Font.DIALOG, Font.BOLD, 50));
        JTextField userNameTextField = new JTextField();
        userNameTextField.setBorder(AccountCreationFrame.BLACK_BORDER);
        label.setFont(new Font(AppSettingsMenu.fontStyle,
AppSettingsMenu.fontType, AppSettingsMenu.textSize));
        contentPane.add(label);
        contentPane.add(userNameTextField);
        //ImageIcon logo = new ImageIcon(new
ImageIcon(logoPath).getImage().getScaledInstance(150, 200, Image.SCALE_SMOOTH));
        // JLabel logoLabel = new JLabel();
        // logoLabel.setIcon(logo);
        JLabel titleLabel = new JLabel("Timisoara Tourer");
        titleLabel.setFont(new Font(Font.DIALOG, Font.BOLD, 100));
        titleLabel.setForeground(Color.ORANGE);
        contentPane.add(titleLabel);
        //contentPane.add(logoLabel);
        springLayout.putConstraint(SpringLayout.NORTH, titleLabel, 40,
SpringLayout.NORTH, contentPane);
        springLayout.putConstraint(SpringLayout.WEST, titleLabel, 50,
SpringLayout.WEST, contentPane);
        //springLayout.putConstraint(SpringLayout.NORTH, logoLabel, 10,
SpringLayout.NORTH, contentPane);
        //springLayout.putConstraint(SpringLayout.EAST, logoLabel, -10,
SpringLayout.EAST, contentPane);
```

```
        springLayout.putConstraint(SpringLayout.NORTH, label, 250,
SpringLayout.NORTH, contentPane);
        springLayout.putConstraint(SpringLayout.WEST, label, 200,
SpringLayout.WEST, contentPane);

        springLayout.putConstraint(SpringLayout.WEST, userNameTextField, 25,
SpringLayout.EAST, label);
        springLayout.putConstraint(SpringLayout.NORTH, userNameTextField, 260,
SpringLayout.NORTH, contentPane);
        springLayout.putConstraint(SpringLayout.EAST, userNameTextField, -200,
SpringLayout.EAST, contentPane);


        JLabel label2 = new JLabel("Password");
        label2.setForeground(Color.ORANGE);
        label2.setFont(new Font(Font.DIALOG, Font.BOLD, 200));
        JPasswordField passwordTextField = new JPasswordField();
        passwordTextField.setBorder(AccountCreationFrame.BLACK_BORDER);
        contentPane.add(label2);
        contentPane.add(passwordTextField);
        label2.setFont(new Font(AppSettingsMenu.fontStyle,
AppSettingsMenu.fontType, AppSettingsMenu.textSize));
        springLayout.putConstraint(SpringLayout.WEST, label2, 200,
SpringLayout.WEST, contentPane);
        springLayout.putConstraint(SpringLayout.NORTH, label2, 100,
SpringLayout.NORTH, label);
        springLayout.putConstraint(SpringLayout.WEST, passwordTextField, 25,
SpringLayout.EAST, label2);
        springLayout.putConstraint(SpringLayout.EAST,  passwordTextField, -200,
SpringLayout.EAST, contentPane);
        springLayout.putConstraint(SpringLayout.NORTH,  passwordTextField, 80,
SpringLayout.SOUTH, label);
        loginFrame.setLocationRelativeTo(null);

        JButton loginButton = new JButton("Login");
        loginButton.setBackground(Color.orange);
        loginButton.setForeground(Color.white);
        loginButton.addActionListener(new ActionListener(){

            @Override
            public void actionPerformed(ActionEvent e) {
                String username = userNameTextField.getText();
                String password = new String(passwordTextField.getPassword());

                boolean ok = App.checkUserExistance(username, password);
```

```
                if(ok == true){
                    loginFrame.setVisible(false);
                    mainFrame.setVisible(true);
                    mainFrame.update();
                }
                else
                {
                    userNameTextField.setBorder(AccountCreationFrame.RED_BORDER);
                    passwordTextField.setBorder(AccountCreationFrame.RED_BORDER);
                    JOptionPane.showMessageDialog(loginFrame, "Wrong username or
password", "ERROR", JOptionPane.ERROR_MESSAGE);
                }
            }

        });


        userNameTextField.addKeyListener(new KeyAdapter() {
            @Override
            public void keyPressed(KeyEvent e) {
                if(e.getKeyCode() == KeyEvent.VK_ENTER){
                    passwordTextField.requestFocus();
                }
            }

        });

        passwordTextField.addKeyListener(new KeyAdapter() {
            @Override
            public void keyPressed(KeyEvent e) {
                if(e.getKeyCode() == KeyEvent.VK_ENTER){
                    loginButton.doClick();
                }
            }

        });

        accountCreationFrame = new AccountCreationFrame();
        ButtonSettings.cardDataDialog = new CardDataDialog(accountCreationFrame);

        JButton createAccountButton = new JButton("Create account");
        createAccountButton.setBackground(Color.orange);
        createAccountButton.setForeground(Color.white);
        createAccountButton.addActionListener(new ActionListener(){
```

```
            @Override
            public void actionPerformed(ActionEvent e) {

                    accountCreationFrame.setVisible(true);

            }

        });

        contentPane.add(createAccountButton);
        contentPane.add(loginButton);

        springLayout.putConstraint(SpringLayout.NORTH, loginButton, 75,
SpringLayout.SOUTH, label2);
        springLayout.putConstraint(SpringLayout.NORTH, createAccountButton, 75,
SpringLayout.SOUTH, label2);
        springLayout.putConstraint(SpringLayout.EAST, loginButton, -600,
SpringLayout.EAST, contentPane);
        springLayout.putConstraint(SpringLayout.WEST, createAccountButton, 10,
SpringLayout.EAST, loginButton);

        JLabel passwordReset = new JLabel("Forgot Password");
        contentPane.add(passwordReset);
        springLayout.putConstraint(SpringLayout.NORTH, passwordReset, 25,
SpringLayout.SOUTH, loginButton);
        springLayout.putConstraint(SpringLayout.EAST, passwordReset, -450,
SpringLayout.EAST, contentPane);
        springLayout.putConstraint(SpringLayout.WEST, passwordReset,  520,
SpringLayout.WEST, contentPane);

        loginFrame.setVisible(true);
        launch(App.class, args);


    }

    @Override
    public void start(Stage primaryStage) throws Exception {
        initAndShowGUI();


    }
    public static void updateCurrentPosition(){
```

```
        JSObject location = (JSObject)
App.engine.executeScript("getCurrentLocation();");
        currentLatitude =((Double) location.getMember("lat"));
        currentLongitude =((Double) location.getMember("lng"));
        System.out.println(currentLatitude + " " + currentLongitude);
    }
    public static void runAndWait(Runnable runnable) {
        try {/* ww  w . j a v a  2s .  c  o m*/
            if (Platform.isFxApplicationThread()) {
                runnable.run();
            } else {
                FutureTask<Object> futureTask = new FutureTask<>(runnable,
                        null);
                Platform.runLater(futureTask);
                futureTask.get();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }


}
```

2. Pachetul com.tourer.gui

- AccountCreationFrame

```
-   package com.tourer.gui;
-
-   import java.io.File;
-   import java.io.IOException;
-   import java.awt.Image;
-   import java.awt.Color;
-   import java.awt.Font;
-   import java.awt.event.KeyAdapter;
-   import java.awt.event.KeyEvent;
-   import java.awt.Dimension;
-   import java.awt.event.ActionListener;
-   import java.awt.image.BufferedImage;
-   import java.util.regex.Matcher;
```

```
- import java.util.regex.Pattern;
- import java.awt.event.ActionEvent;
- import java.awt.Toolkit;
-
- import javax.imageio.ImageIO;
- import javax.swing.ImageIcon;
- import javax.swing.JButton;
- import javax.swing.JFrame;
- import javax.swing.JLabel;
- import javax.swing.JOptionPane;
- import javax.swing.JPasswordField;
- import javax.swing.JTextField;
- import javax.swing.SpringLayout;
- import javax.swing.border.LineBorder;
-

- import com.tourer.jdbc.Connector;
-
- public class AccountCreationFrame extends JFrame{
-
-     public final static String createUserIcon =
  "Icons\\CreateUserIcon.png";
-     public final static String backgroundPath =
  "Icons\\AccountCreationBackground.png";
-     public final static LineBorder BLACK_BORDER = new
  LineBorder(Color.BLACK, 4);
-     public final static LineBorder RED_BORDER = new LineBorder(Color.RED,
  4);
-     public final static Pattern USERNAME_PATTERN =
  Pattern.compile("^.{6,}$");
-     //Minimum eight characters, at least one letter, one number and one
  special character
-     public final static Pattern PASSWORD_PATTERN =
  Pattern.compile("^(?=.*[A-Za-z])(?=.*\\d)(?=.*[@$!%*#?&])[A-Za-z\\d@$!%
  *#?&]{8,}$");
-     public final static Pattern MAIL_PATTERN =
  Pattern.compile("^[\\w-\\.]+@([\\w-]+\\.)+[\\w-]{2,4}$");
-     public static Image background =
  Toolkit.getDefaultToolkit().getImage(backgroundPath);
-     {
-         try {
-             BufferedImage bufferedImage = ImageIO.read(new
  File(backgroundPath));
```

```
-                background = bufferedImage.getScaledInstance((int)
    MainFrame.screenSize.getWidth() * 2 / 3, (int)
    MainFrame.screenSize.getHeight() * 2 / 3, Image.SCALE_SMOOTH);
-            } catch (IOException e) {
-                // TODO Auto-generated catch block
-                e.printStackTrace();
-            }
-        }
-
-        public AccountCreationFrame(){
-            this.setSize(new Dimension((int) MainFrame.screenSize.getWidth()
    * 2 / 3, (int) MainFrame.screenSize.getHeight() * 2 / 3));
-            this.setResizable(false);
-            this.setVisible(false);
-            this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
-
-
-            BackGroundPanel contentPane = new BackGroundPanel(background);
-
-            SpringLayout springLayout = new SpringLayout();
-            contentPane.setLayout(springLayout);
-
-
-            this.setContentPane(contentPane);
-
-
-
-            JLabel userIcon = new JLabel();
-            ImageIcon icon = new ImageIcon(new
    ImageIcon(createUserIcon).getImage().getScaledInstance(150, 150,
    Image.SCALE_SMOOTH));
-            userIcon.setIcon(icon);
-            contentPane.add(userIcon);
-            springLayout.putConstraint(SpringLayout.NORTH, userIcon, 50,
    SpringLayout.NORTH, contentPane);
-            springLayout.putConstraint(SpringLayout.WEST, userIcon, 420,
    SpringLayout.WEST, contentPane);
-
-            JLabel usernameLabel = new JLabel("UsserName");
-            usernameLabel.setForeground(Color.ORANGE);
-            usernameLabel.setFont(new Font(Font.DIALOG, Font.BOLD, 30));
-            contentPane.add(usernameLabel);
-            springLayout.putConstraint(SpringLayout.NORTH, usernameLabel,
    20, SpringLayout.SOUTH, userIcon);
-
```

```
-
-           JTextField usernameTextField = new JTextField();
-           JPasswordField passwordTextField = new JPasswordField();
-           JPasswordField reenterpasswordTextField = new JPasswordField();
-           JTextField mailTextField = new JTextField();
-           JButton signUpButton = new JButton("Sign Up");
-           usernameTextField.addKeyListener(new KeyAdapter() {
-               @Override
-               public void keyPressed(KeyEvent e) {
-                   if(e.getKeyCode() == KeyEvent.VK_ENTER){
-                       passwordTextField.requestFocus();
-                   }
-               }
-
-           });
-           usernameTextField.setBorder(BLACK_BORDER);
-           contentPane.add(usernameTextField);
-           springLayout.putConstraint(SpringLayout.WEST, usernameTextField,
    400, SpringLayout.WEST, contentPane);
-           springLayout.putConstraint(SpringLayout.EAST, usernameTextField,
    -100, SpringLayout.EAST, contentPane);
-           springLayout.putConstraint(SpringLayout.NORTH,
    usernameTextField, 35, SpringLayout.SOUTH, userIcon);
-           springLayout.putConstraint(SpringLayout.EAST, usernameLabel,
    -20, SpringLayout.WEST, usernameTextField);
-
-           JLabel passwordLabel = new JLabel("Password");
-           passwordLabel.setForeground(Color.ORANGE);
-           passwordLabel.setFont(new Font(Font.DIALOG, Font.BOLD, 30));
-           contentPane.add(passwordLabel);
-           springLayout.putConstraint(SpringLayout.NORTH, passwordLabel,
    20, SpringLayout.SOUTH, usernameLabel);
-
-           passwordTextField.addKeyListener(new KeyAdapter() {
-               @Override
-               public void keyPressed(KeyEvent e) {
-                   if(e.getKeyCode() == KeyEvent.VK_ENTER){
-                       reenterpasswordTextField.requestFocus();
-                   }
-               }
-
-           });
-           passwordTextField.setBorder(BLACK_BORDER);
-           contentPane.add(passwordTextField);
```

```
-           springLayout.putConstraint(SpringLayout.WEST, passwordTextField,
    400, SpringLayout.WEST, contentPane);
-           springLayout.putConstraint(SpringLayout.EAST, passwordTextField,
    -100, SpringLayout.EAST, contentPane);
-           springLayout.putConstraint(SpringLayout.NORTH,
    passwordTextField, 35, SpringLayout.SOUTH, usernameLabel);
-           springLayout.putConstraint(SpringLayout.EAST, passwordLabel,
    -20, SpringLayout.WEST, passwordTextField);
-
-           JLabel reenterpasswordLabel = new JLabel("Reenter Password");
-           reenterpasswordLabel.setForeground(Color.ORANGE);
-           reenterpasswordLabel.setFont(new Font(Font.DIALOG, Font.BOLD,
    30));
-           contentPane.add(reenterpasswordLabel);
-           springLayout.putConstraint(SpringLayout.NORTH,
    reenterpasswordLabel, 20, SpringLayout.SOUTH, passwordLabel);
-
-           reenterpasswordTextField.addKeyListener(new KeyAdapter() {
-               @Override
-               public void keyPressed(KeyEvent e) {
-                   if(e.getKeyCode() == KeyEvent.VK_ENTER){
-                       mailTextField.requestFocus();
-                   }
-               }
-
-           });
-           reenterpasswordTextField.setBorder(BLACK_BORDER);
-           contentPane.add(reenterpasswordTextField);
-           springLayout.putConstraint(SpringLayout.WEST,
    reenterpasswordTextField, 400, SpringLayout.WEST, contentPane);
-           springLayout.putConstraint(SpringLayout.EAST,
    reenterpasswordTextField, -100, SpringLayout.EAST, contentPane);
-           springLayout.putConstraint(SpringLayout.NORTH,
    reenterpasswordTextField, 35, SpringLayout.SOUTH, passwordLabel);
-           springLayout.putConstraint(SpringLayout.EAST,
    reenterpasswordLabel, -20, SpringLayout.WEST, reenterpasswordTextField);
-
-           JLabel mailLabel = new JLabel("Mail");
-           mailLabel.setForeground(Color.ORANGE);
-           mailLabel.setFont(new Font(Font.DIALOG, Font.BOLD, 30));
-           contentPane.add(mailLabel);
-           springLayout.putConstraint(SpringLayout.NORTH, mailLabel, 20,
    SpringLayout.SOUTH, reenterpasswordLabel);
-
-
```

```
-           mailTextField.addKeyListener(new KeyAdapter() {
-               @Override
-               public void keyPressed(KeyEvent e) {
-                   if(e.getKeyCode() == KeyEvent.VK_ENTER){
-                       signUpButton.doClick();
-                   }
-               }
-
-           });
-           mailTextField.setBorder(BLACK_BORDER);
-
-           contentPane.add(mailTextField);
-           springLayout.putConstraint(SpringLayout.WEST, mailTextField,
    400, SpringLayout.WEST, contentPane);
-           springLayout.putConstraint(SpringLayout.EAST, mailTextField,
    -100, SpringLayout.EAST, contentPane);
-           springLayout.putConstraint(SpringLayout.NORTH, mailTextField,
    35, SpringLayout.SOUTH, reenterpasswordLabel);
-           springLayout.putConstraint(SpringLayout.EAST,  mailLabel, -20,
    SpringLayout.WEST, mailTextField);
-
-
-           signUpButton.setBackground(Color.orange);
-           signUpButton.setForeground(Color.white);
-           signUpButton.addActionListener(new ActionListener(){
-
-               @Override
-               public void actionPerformed(ActionEvent e) {
-                   Boolean ok = true;
-                   String username = usernameTextField.getText();
-                   Matcher userMatcher = USERNAME_PATTERN.matcher(username);
-                   if(!userMatcher.find())
-                   {
-                       usernameTextField.setBorder(RED_BORDER);
-                       ok = false;
-                       JOptionPane.showMessageDialog(AccountCreationFrame.
    this, "Username must contain at least 6 characters", "ERROR",
    JOptionPane.ERROR_MESSAGE);
-                   }
-                   else
-                   {
-                       usernameTextField.setBorder(BLACK_BORDER);
-                   }
-
```

```
-                   String password = new
    String(passwordTextField.getPassword());
-
-                   Matcher passwordMatcher =
    PASSWORD_PATTERN.matcher(password);
-
-                   if(!passwordMatcher.find()){
-                       passwordTextField.setBorder(RED_BORDER);
-                       reenterpasswordTextField.setBorder(RED_BORDER);
-                       JOptionPane.showMessageDialog(AccountCreationFrame.
    this, "Password must contain minimum eight characters, at least one letter,
    one number and one special character", "ERROR", JOptionPane.ERROR_MESSAGE);
-                       ok = false;
-                   }
-                   else
-                   {
-                       passwordTextField.setBorder(BLACK_BORDER);
-                       reenterpasswordTextField.setBorder(BLACK_BORDER);
-                   }
-
-                   String reenterpassword =  new
    String(reenterpasswordTextField.getPassword());
-
-                   if(reenterpassword.equals(password) == false){
-                       passwordTextField.setBorder(RED_BORDER);
-                       reenterpasswordTextField.setBorder(RED_BORDER);
-                       JOptionPane.showMessageDialog(AccountCreationFrame.
    this, "Password doesn't match", "ERROR", JOptionPane.ERROR_MESSAGE);
-                       ok = false;
-                   }
-
-                   String mail = mailTextField.getText();
-                   Matcher mailMatcher = MAIL_PATTERN.matcher(mail);
-
-                   if(!mailMatcher.find()){
-                       mailTextField.setBorder(RED_BORDER);
-                       ok = false;
-                       JOptionPane.showMessageDialog(AccountCreationFrame.
    this, "Not a valid email address", "ERROR", JOptionPane.ERROR_MESSAGE);
-                   }
-                   else
-                   {
-                       mailTextField.setBorder(BLACK_BORDER);
-                   }
-
```

```
-                   if(ok == true){
-                       boolean ok2 = Connector.createUser(username,
    password, mail);
-
-                       if(ok2 == true)
-                           AccountCreationFrame.this.setVisible(false);
-
-                   }
-
-               }
-
-           });
-           contentPane.add(signUpButton);
-           springLayout.putConstraint(SpringLayout.NORTH, signUpButton, 35,
    SpringLayout.SOUTH, mailLabel);
-           springLayout.putConstraint(SpringLayout.WEST, signUpButton, 430,
    SpringLayout.WEST, contentPane);
-           springLayout.putConstraint(SpringLayout.EAST, signUpButton,
    -450, SpringLayout.EAST, contentPane);
-
-           this.setLocationRelativeTo(null);
-       }
-   }
-
```

- AddButton

```
-   package com.tourer.gui;
-
-   import com.tourer.App;
-   import com.tourer.jdbc.*;
-
-   import javafx.application.Platform;
-
-   import java.awt.event.ActionListener;
-   import java.sql.SQLException;
-

-   import javax.swing.JOptionPane;
-

-   import java.awt.event.ActionEvent;
-
```

```
-   import netscape.javascript.JSObject;
-
-   public class AddButton extends CostumButton{
-
-       public final static String iconPath = "Icons\\Add.png";
-       public final static String darkIconPath = "Icons\\AddDark.png";
-       public static AddLocationDialog addLocationDialog = new
    AddLocationDialog(App.mainFrame);
-       public static Double latitude = 0d;
-       public static Double longitude = 0d;
-       public AddButton(int w, int h) {
-           super(w, h, AddButton.iconPath);
-
-           this.addActionListener(new ActionListener(){
-
-               @Override
-               public void actionPerformed(ActionEvent e) {
-                   //this 2 need to be taken from the curent location on the
    map
-                   Platform.runLater(new Runnable() {
-
-                       @Override
-                       public void run() {
-                           // TODO Auto-generated method stub
-                           JSObject location = (JSObject)
    App.engine.executeScript("getLocationOfCommit();");
-                           latitude =((Double) location.getMember("lat"));
-                           longitude =((Double) location.getMember("lng"));
-                           AddLocationDialog.latitude.setText("Latitude: "
    + latitude.toString());
-                           AddLocationDialog.longitude.setText("Longitude:
    " + longitude.toString()  + "  ");
-                           addLocationDialog.revalidate();
-                           addLocationDialog.repaint();
-                           addLocationDialog.setVisible(true);
-
-                           while(addLocationDialog.isVisible() == true){
-
-                           }
-
-                           String locationDescription =
    AddLocationDialog.locationDescritpionTextField.getText();
-                           String locationName =
    AddLocationDialog.locationNameTextField.getText();
```

```
-                           if(locationDescription.equals("") == false &&
    locationName.equals("") == false){
-                               //insert into database
-
-                               try {
-                                   Connector.createLocation(latitude,
    longitude, locationName, locationDescription);
-                                   Platform.runLater(new Runnable() {
-
-                                       @Override
-                                       public void run() {
-                                           App.engine.executeScript("initM
    ap();");
-                                       }
-                                   });
-
-
-                               } catch (SQLException e1) {
-
-                                   JOptionPane.showMessageDialog(App.mainF
    rame, "Failde to add location to database", "ERROR",
    JOptionPane.ERROR_MESSAGE);
-                               }
-                           }
-
-                           AddLocationDialog.locationDescritpionTextField.
    setText("");
-                           AddLocationDialog.locationNameTextField.setText
    ("");
-                       }
-                   });
-
-
-
-
-               }
-
-           });
-
-       }
-
-       public static void updateLocation(Double newLatitude, Double
    newLongitude){
-           latitude = newLatitude;
-           longitude = newLongitude;
```

```
-       }
-
-   }
-
```

- AddLocationDialog

```
-   package com.tourer.gui;
-
-   import java.awt.Color;
-   import java.awt.Dimension;
-   import java.awt.Font;
-   import java.awt.Frame;
-   import java.awt.GridBagConstraints;
-   import java.awt.event.ActionListener;
-   import java.awt.event.ActionEvent;
-
-   import javax.swing.Box;
-   import javax.swing.JButton;
-   import javax.swing.JDialog;
-   import javax.swing.JLabel;
-   import javax.swing.JScrollPane;
-   import javax.swing.JTextArea;
-   import javax.swing.JTextField;
-
-
-   public class AddLocationDialog extends JDialog{
-       public static JLabel longitude = new JLabel("Longitude: ");
-       public static JLabel latitude = new JLabel("Latitude: ");
-       public static JLabel locationNameLabel = new JLabel("Location name: ");
-       public static JTextField locationNameTextField = new JTextField("");
-       public static JLabel locationDescriptionLabel = new
    JLabel("Description: ");
-       public static JTextArea locationDescritpionTextField = new
    JTextArea("");
-       static{
-           longitude.setForeground(Color.orange);
-           longitude.setPreferredSize(new Dimension((int)
    MainFrame.screenSize.getWidth() / 3 - 80, 20));
-           latitude.setForeground(Color.orange);
-           latitude.setPreferredSize(new Dimension((int)
    MainFrame.screenSize.getWidth() / 3 - 80, 20));
-           locationNameLabel.setForeground(Color.orange);
```

```
-
-           locationDescriptionLabel.setForeground(Color.orange);
-           locationDescriptionLabel.setPreferredSize(new Dimension((int)
    MainFrame.screenSize.getWidth() / 3 - 80, 20));
-           locationNameTextField.setPreferredSize(new Dimension((int)
    MainFrame.screenSize.getWidth() / 6, 20));
-           locationNameTextField.setBorder(AccountCreationFrame.BLACK_BORD
    ER);
-           locationDescritpionTextField.setBorder(AccountCreationFrame.BLA
    CK_BORDER);
-           locationDescritpionTextField.setPreferredSize(new
    Dimension((int) MainFrame.screenSize.getWidth() / 3 - 80, 70));
-
-           longitude.setFont(new Font(AppSettingsMenu.fontStyle,
    AppSettingsMenu.fontType, AppSettingsMenu.textSize));
-           latitude.setFont(new Font(AppSettingsMenu.fontStyle,
    AppSettingsMenu.fontType, AppSettingsMenu.textSize));
-           latitude.setFont(new Font(AppSettingsMenu.fontStyle,
    AppSettingsMenu.fontType, AppSettingsMenu.textSize));
-           locationNameLabel.setFont(new Font(AppSettingsMenu.fontStyle,
    AppSettingsMenu.fontType, AppSettingsMenu.textSize));
-           locationDescriptionLabel.setFont(new
    Font(AppSettingsMenu.fontStyle, AppSettingsMenu.fontType,
    AppSettingsMenu.textSize));
-       }
-       public AddLocationDialog(Frame frame){
-           super(frame,"Add new location", ModalityType.APPLICATION_MODAL);
-
-           this.setSize(new Dimension((int) MainFrame.screenSize.getWidth()
    / 3, (int) MainFrame.screenSize.getHeight() / 3));
-
-           ColorPanel contentPane = new ColorPanel();
-           contentPane.setSize(new Dimension((int)
    MainFrame.screenSize.getWidth() / 3, (int)
    MainFrame.screenSize.getHeight() / 3));
-           contentPane.add(longitude);
-
-           contentPane.add(latitude);
-
-           contentPane.add(locationNameLabel);
-           contentPane.add(locationNameTextField);
-
-           contentPane.add(locationDescriptionLabel);
-           JScrollPane scrollPane = new
    JScrollPane(locationDescritpionTextField);
```

```
-           scrollPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_
    SCROLLBAR_ALWAYS);
-           scrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCRO
    LLBAR_ALWAYS);
-           //scrollPane.setPreferredSize(new Dimension((int)
    MainFrame.screenSize.getWidth() / 6, (int)
    MainFrame.screenSize.getHeight() / 7));
-           GridBagConstraints c = new GridBagConstraints();
-           c.weightx = 1;
-           c.weighty = 1;
-           c.fill = GridBagConstraints.BOTH;
-           contentPane.add(scrollPane, c);
-           contentPane.add(Box.createHorizontalStrut(1));
-           JButton addButton = new JButton("Add");
-           addButton.addActionListener(new ActionListener(){
-
-               @Override
-               public void actionPerformed(ActionEvent e) {
-
-                   boolean ok = true;
-                   if(AddLocationDialog.locationNameTextField.getText().eq
    uals("")){
-                       AddLocationDialog.locationNameTextField.setBorder(A
    ccountCreationFrame.RED_BORDER);
-                       ok = false;
-                   }
-                   else
-                   {
-                       AddLocationDialog.locationNameTextField.setBorder(A
    ccountCreationFrame.BLACK_BORDER);
-                   }
-                   if(AddLocationDialog.locationDescritpionTextField.getTe
    xt().equals("")){
-                       AddLocationDialog.locationDescritpionTextField.setB
    order(AccountCreationFrame.RED_BORDER);
-                       ok = false;
-                   }
-                   else
-                   {
-                       AddLocationDialog.locationDescritpionTextField.setB
    order(AccountCreationFrame.BLACK_BORDER);
-                   }
-                   if(ok == true){
-                       AddLocationDialog.this.setVisible(false);
-                   }
```

```
-               }
-
-           });
-           contentPane.add(addButton);
-           this.setContentPane(contentPane);
-
-
-           this.setVisible(false);
-           this.setDefaultCloseOperation(JDialog.HIDE_ON_CLOSE);
-           this.setLocationRelativeTo(null);
-       }
-   }
-
```

- AppSettingsMenu

```
-   package com.tourer.gui;
-
-   import java.io.File;
-   import java.io.IOException;
-   import java.awt.Toolkit;
-   import java.awt.Image;
-   import java.awt.image.BufferedImage;
-   import java.awt.Font;
-   import java.awt.Component;
-   import java.awt.Dimension;
-   import java.awt.BorderLayout;
-   import java.awt.Color;
-
-   import javax.swing.BoxLayout;
-
-   import javax.imageio.ImageIO;
-   import javax.swing.BorderFactory;
-   import javax.swing.Box;
-   import javax.swing.JLabel;
-
-   import javax.swing.border.LineBorder;
-   import javax.swing.border.TitledBorder;
-
-
-   public class AppSettingsMenu extends SettingsMenu{
-
-       public final static int buttonHeight = 40;
```

```
-       public final static int buttonWidth  = 40;
-       final static int w = 40;
-       final static int h = 40;
-       public final static int textSize = 25;
-       public final static String fontStyle = Font.DIALOG;
-       public final static int fontType =  Font.PLAIN;
-       final static LineBorder border = new LineBorder(Color.orange,1, true);
-       public final static Color PURPLE_COLOR = new Color(101, 24, 115);
-       final static String[] languages = new String[]{"English", "Romana"};
-       final static GradientColor TRANSP_GRADIENT_COLOR = new
    GradientColor(new Color(213, 134, 145, 123), new Color(213, 134, 145, 123));
-       public AppSettingsMenu() {
-
-           Dimension windowPanelSize = new
    Dimension(MainFrame.screenSize.width * 4 / 5,
    MainFrame.screenSize.height);
-           String windowPanelbackgroundPath =
    "Icons\\AccountCreationBackground.png";
-           Image windowPanelbackground =
    Toolkit.getDefaultToolkit().getImage(windowPanelbackgroundPath);
-           try {
-               BufferedImage bufferedImage = ImageIO.read(new
    File(windowPanelbackgroundPath));
-               windowPanelbackground  =
    bufferedImage.getScaledInstance(MainFrame.screenSize.width * 4 / 5,
    MainFrame.screenSize.height + 10, Image.SCALE_SMOOTH);
-           } catch (IOException e) {
-               // TODO Auto-generated catch block
-               e.printStackTrace();
-           }
-           BackGroundPanel windowPanel = new
    BackGroundPanel(windowPanelbackground);
-           //ColorPanel windowPanel = new ColorPanel();
-           windowPanel.setPreferredSize(windowPanelSize);
-           windowPanel.setLayout(new BoxLayout(windowPanel,
    BoxLayout.PAGE_AXIS));
-
-           windowPanel.setAlignmentX(Component.LEFT_ALIGNMENT);
-           Dimension fieldsGridPanelSize = new Dimension(300,
    MainFrame.screenSize.height);
-           String fieldsGridPanelBackgroundPath =
    "Icons\\SettingsSideMenuBackground.jpg";
-           Image fieldsGridPanelbackground =
    Toolkit.getDefaultToolkit().getImage(fieldsGridPanelBackgroundPath);
-           try {
```

```
-               BufferedImage bufferedImage = ImageIO.read(new
    File(fieldsGridPanelBackgroundPath));
-               fieldsGridPanelbackground  =
    bufferedImage.getScaledInstance(300, MainFrame.screenSize.height,
    Image.SCALE_SMOOTH);
-           } catch (IOException e) {
-               // TODO Auto-generated catch block
-               e.printStackTrace();
-           }
-           BackGroundPanel fieldsGridPanel = new
    BackGroundPanel(fieldsGridPanelbackground);
-           fieldsGridPanel.setSize(fieldsGridPanelSize);
-           fieldsGridPanel.setPreferredSize(new
    Dimension(MainFrame.screenSize.width / 5 - 100,
    MainFrame.screenSize.height));
-
-           ButtonSettings gridPanelWindow = new ButtonSettings(Color.red,
    Color.yellow, "Window");
-           gridPanelWindow.setAlignmentX(Component.RIGHT_ALIGNMENT);
-           gridPanelWindow.setPreferredSize(new Dimension((int)
    fieldsGridPanel.getPreferredSize().getWidth(), buttonHeight));
-           gridPanelWindow.setBorder(border);
-           fieldsGridPanel.add(gridPanelWindow);
-
-
-
-
-
-           ButtonSettings gridPanelSecurity = new ButtonSettings(Color.red,
    Color.yellow, "Security");
-           gridPanelSecurity.setAlignmentX(Component.RIGHT_ALIGNMENT);
-           gridPanelSecurity.setPreferredSize(new Dimension((int)
    fieldsGridPanel.getPreferredSize().getWidth(), buttonHeight));
-           gridPanelSecurity.setBorder(border);
-           fieldsGridPanel.add(gridPanelSecurity);
-
-
-           this.add(fieldsGridPanel, BorderLayout.WEST);
-
-           GridPanel checkPanel = new GridPanel(TRANSP_GRADIENT_COLOR);
-           checkPanel.setBorder(BorderFactory.createTitledBorder(border,
    "Settings", TitledBorder.LEADING, TitledBorder.BELOW_TOP, new
    Font(fontStyle, fontType, textSize), PURPLE_COLOR));
-           checkPanel.setAlignmentX(Component.LEFT_ALIGNMENT);
-           CheckBox darkMode = new CheckBox("Dark Mode");
-           darkMode.setForeground(PURPLE_COLOR);
```

```
-           checkPanel.addLeft(darkMode);
-           checkPanel.addRight(Box.createHorizontalStrut(20));
-           CheckBox notifications = new CheckBox("Notifications");
-           notifications.setForeground(PURPLE_COLOR);
-           notifications.setSelected(true);
-           checkPanel.addLeft(notifications);
-           checkPanel.addRight(Box.createHorizontalStrut(20));
-           checkPanel.setOpaque(false);
-           windowPanel.add(checkPanel);
-
-           GridPanel gridPanel = new GridPanel(TRANSP_GRADIENT_COLOR);
-
-           gridPanel.setBorder(BorderFactory.createTitledBorder(border,
    "Resizing", TitledBorder.LEADING, TitledBorder.BELOW_TOP, new
    Font(fontStyle, fontType, textSize), PURPLE_COLOR));
-           gridPanel.setAlignmentX(Component.LEFT_ALIGNMENT);
-
-           JLabel textResizerLabel = new JLabel("Text Size");
-           textResizerLabel.setForeground(PURPLE_COLOR);
-           textResizerLabel.setFont(new Font(fontStyle, fontType,
    textSize));
-           gridPanel.addLeft(textResizerLabel);
-           gridPanel.addRight(Box.createVerticalStrut(20));
-           TextResizer textResizer = new TextResizer();
-           gridPanel.addLeft(Box.createVerticalStrut(20));
-           gridPanel.addRight(textResizer);
-
-           gridPanel.addSpacer(Box.createVerticalStrut(20));
-
-           JLabel iconResizerLabel = new JLabel("Icon Size");
-           iconResizerLabel.setForeground(PURPLE_COLOR);
-           iconResizerLabel.setFont(new Font(fontStyle, fontType,
    textSize));
-           gridPanel.addLeft(iconResizerLabel);
-           gridPanel.addRight(Box.createVerticalStrut(20));
-           IconResizer iconResizer = new IconResizer();
-           gridPanel.addLeft(Box.createVerticalStrut(20));
-           gridPanel.addRight(iconResizer);
-           gridPanel.setOpaque(false);
-           windowPanel.add(gridPanel);
-           GridPanel languagePanel = new GridPanel(TRANSP_GRADIENT_COLOR);
-           languagePanel.setBorder(BorderFactory.createTitledBorder(border
    , "Language", TitledBorder.LEADING, TitledBorder.BELOW_TOP, new
    Font(fontStyle, fontType, textSize), PURPLE_COLOR));
-           languagePanel.setAlignmentX(Component.LEFT_ALIGNMENT);
```

```
-
-           LanguagePicker languageField = new LanguagePicker();
-
-           languageField.setVisible(true);
-           languageField.search.setBackground(PURPLE_COLOR);
-           languageField.search.setForeground(Color.orange);
-
-
-           for(int i = 0; i < languages.length; i++)
-               languageField.addItem(languages[i]);
-           languageField.setSelectedItem("English");
-           languagePanel.addLeft(languageField);
-           languagePanel.addRight(Box.createVerticalStrut(20));
-           languagePanel.setOpaque(false);
-           windowPanel.add(languagePanel);
-
-
-           BackGroundPanel securityPanel = new
    BackGroundPanel(windowPanelbackground);
-           securityPanel.setPreferredSize(new
    Dimension(MainFrame.screenSize.width * 4 / 5,
    MainFrame.screenSize.height));
-           securityPanel.setLayout(new BoxLayout(securityPanel,
    BoxLayout.PAGE_AXIS));
-           securityPanel.setAlignmentX(Component.LEFT_ALIGNMENT);
-
-           GridPanel cardDataPanel = new GridPanel(TRANSP_GRADIENT_COLOR);
-           cardDataPanel.setOpaque(false);
-           cardDataPanel.setBorder(BorderFactory.createTitledBorder(border
    , "CardData", TitledBorder.LEADING, TitledBorder.BELOW_TOP, new
    Font(fontStyle, fontType, textSize), PURPLE_COLOR));
-           cardDataPanel.setAlignmentX(Component.LEFT_ALIGNMENT);
-
-           ButtonSettings cardUpdateButton = new ButtonSettings(Color.red,
    Color.yellow, "Update card data", 1);
-           cardDataPanel.addLeft(cardUpdateButton);
-           securityPanel.add(cardDataPanel);
-
-           ButtonSettings.cardPanel.add(windowPanel, "Window");
-           ButtonSettings.cardPanel.add(securityPanel, "Security");
-           ButtonSettings.cardLayout.show(ButtonSettings.cardPanel,
    "Window");
-
-
-
```

```
            this.add(ButtonSettings.cardPanel, BorderLayout.CENTER);
            this.update();
        }


    }

```

- BackGroundPanel

```
    package com.tourer.gui;

    import java.awt.Graphics;
    import java.awt.Image;
    import javax.swing.JPanel;

    public class BackGroundPanel extends JPanel{

        Image img;
        public BackGroundPanel(Image img) {
            super();
            this.img = img;
        }

        public Image getImg() {
            return this.img;
        }

        public void setImg(Image img) {
            this.img = img;
        }


        @Override
        protected void paintComponent(Graphics g){
            super.paintComponent(g);
            g.drawImage(img, 0, 0, null);

        }
    }

```

- ButtonBox

```
- package com.tourer.gui;
-
- import java.awt.Component;
-
- import javax.swing.Box;
- import javax.swing.BoxLayout;
-
-
- public class ButtonBox extends Box{
-
-     public final static int buttonHeight = 80;
-     public final static int buttonWidth  = 80;
-     static UserSettingsMenu userSettingsMenu = new UserSettingsMenu();
-     static AppSettingsMenu appMenuSettings = new AppSettingsMenu();
-
-     public ButtonBox(LocationSearchField locationSearchField,
  UserSearchField userSearchField) {
-         super(BoxLayout.PAGE_AXIS);
-
-         SearchButton searchButton = new
  SearchButton(ButtonBox.buttonWidth, ButtonBox.buttonHeight,
  userSearchField);
-         this.addWithSpacer(searchButton, ButtonBox.buttonWidth / 2);
-
-         LocationButton locationButton = new
  LocationButton(ButtonBox.buttonWidth, ButtonBox.buttonHeight,
  locationSearchField);
-         this.addWithSpacer(locationButton, ButtonBox.buttonWidth / 2);
-
-         AddButton addButton = new
  AddButton(ButtonBox.buttonWidth,  ButtonBox.buttonHeight);
-         this.addWithSpacer(addButton, ButtonBox.buttonWidth / 2);
-
-
-         UsserButton usserButton = new
  UsserButton(ButtonBox.buttonWidth,  ButtonBox.buttonHeight,
  userSettingsMenu);
-         this.addWithSpacer(usserButton, ButtonBox.buttonWidth / 2);
-
-         SettingsButton settingsButton = new
  SettingsButton(ButtonBox.buttonWidth,  ButtonBox.buttonHeight,
  appMenuSettings);
```

```
-           this.addWithSpacer(settingsButton, ButtonBox.buttonWidth / 2);
-
-           ReloadButton reloadButton = new
    ReloadButton(ButtonBox.buttonWidth,  ButtonBox.buttonHeight);
-
-           this.addWithSpacer(reloadButton, ButtonBox.buttonWidth / 2);
-       }
-       public void addWithSpacer(Component comp, int size){
-           this.add(comp);
-           this.add(Box.createVerticalStrut(size));
-       }
-   }
-
-
-
```

- ButtonSettings

```
-   package com.tourer.gui;
-
-   import java.awt.Font;
-   import java.awt.CardLayout;
-   import java.awt.Color;
-   import java.awt.event.MouseAdapter;
-   import java.awt.event.MouseEvent;
-



-   import javax.swing.JLabel;
-   import javax.swing.JPanel;
-
-   public class ButtonSettings extends GridPanel{
-
-       public static CardDataDialog cardDataDialog;
-       Color c1Select;
-       Color c2Select;
-       static Color c3Select = Color.black;
-       static Color c3 = Color.orange;
-       String text;
-       static CardLayout cardLayout;
-       static JPanel cardPanel;
-       public JLabel label;
-       String name;
```

```
-       static{
-
-           cardPanel = new JPanel();
-           cardLayout = new CardLayout();
-           cardPanel.setLayout(cardLayout);
-
-       }
-       public ButtonSettings(String name){
-           this.c1Select = Color.RED;
-           this.c2Select = Color.YELLOW;
-           this.name = name;
-           label = new JLabel(name);
-           label.setFont(new Font(AppSettingsMenu.fontStyle,
    AppSettingsMenu.fontType, AppSettingsMenu.textSize));
-           label.setForeground(c3);
-           this.addLeft(label);
-
-           this.addMouseListener(new MouseAdapter(){
-               @Override
-               public void mouseEntered(MouseEvent event) {
-                   ButtonSettings.this.swapColors();
-
-               }
-               @Override
-               public void mouseExited(MouseEvent event){
-                   ButtonSettings.this.swapColors();
-               }
-               @Override
-               public void mouseClicked(MouseEvent me){
-
-                   cardLayout.show(cardPanel, name);
-               }
-           });
-       }
-
-       public ButtonSettings(Color c1Select, Color c2Select, String name){
-           this.c1Select = c1Select;
-           this.c2Select = c2Select;
-           this.name = name;
-           label = new JLabel(name);
-           label.setForeground(c3);
-           label.setFont(new Font(AppSettingsMenu.fontStyle,
    AppSettingsMenu.fontType, AppSettingsMenu.textSize));
-           this.addLeft(label);
-
```

```
-           this.addMouseListener(new MouseAdapter(){
-               @Override
-               public void mouseEntered(MouseEvent event) {
-                   ButtonSettings.this.swapColors();
-
-               }
-               @Override
-               public void mouseExited(MouseEvent event){
-                   ButtonSettings.this.swapColors();
-               }
-               @Override
-               public void mouseClicked(MouseEvent me){
-
-                   cardLayout.show(cardPanel, name);
-               }
-           });
-       }
-
-       public ButtonSettings(Color c1Select, Color c2Select, String name,
    Integer nr){
-           this.c1Select = c1Select;
-           this.c2Select = c2Select;
-           this.name = name;
-           label = new JLabel(name);
-           label.setForeground(c3);
-           label.setFont(new Font(AppSettingsMenu.fontStyle,
    AppSettingsMenu.fontType, AppSettingsMenu.textSize));
-           this.addLeft(label);
-
-           this.addMouseListener(new MouseAdapter(){
-               @Override
-               public void mouseEntered(MouseEvent event) {
-                   ButtonSettings.this.swapColors();
-
-               }
-               @Override
-               public void mouseExited(MouseEvent event){
-                   ButtonSettings.this.swapColors();
-               }
-               @Override
-               public void mouseClicked(MouseEvent me){
-
-                   switch(nr){
-                       case 1:
-                           cardDataDialog.setVisible(true);
```

```
                            break;
                    }

                }
            });
        }

        public void swapColors(){
            Color auxc1 = c1;
            Color auxc2 = c2;
            Color auxc3 = c3;
            c1 = c1Select;
            c2 = c2Select;
            c3 = c3Select;
            c1Select = auxc1;
            c2Select = auxc2;
            c3Select = auxc3;
            label.setForeground(c3);
            repaint();
            revalidate();
        }
    }

```

- CardDataDialog

```
    package com.tourer.gui;

    import java.awt.Color;
    import java.awt.Dimension;
    import java.awt.Font;
    import java.awt.Frame;
    import java.awt.event.ActionListener;
    import java.awt.event.ActionEvent;
    import javax.swing.Box;
    import javax.swing.JButton;
    import javax.swing.JDialog;

    import javax.swing.JLabel;
    import javax.swing.JTextField;

    import com.tourer.jdbc.Connector;
```

```
-
-   public class CardDataDialog extends JDialog{
-
-       public static String[] cardTypes= new String[]{"Visa", "MasterCard"};
-       public static String[] mounth = new String[]{"01", "02", "03", "04",
    "05", "06", "07", "08", "09", "10", "11", "12"};
-       public static String[] year = new String[]{"2021", "2022", "2023",
    "2024", "2025", "2026", "2027", "2028", "2029", "2030", "2031", "2032"};
-       public CardDataDialog(Frame frame){
-           super(frame, "Add card data", ModalityType.APPLICATION_MODAL);
-           this.setVisible(false);
-           this.setSize(new Dimension((MainFrame.screenSize.width * 3) / 4 ,
    (MainFrame.screenSize.height * 3) / 4));
-
-           GridPanel contentPane = new GridPanel();
-           JLabel label = new JLabel("Credit card");
-           label.setForeground(Color.orange);
-           label.setFont(new Font(AppSettingsMenu.fontStyle,
    AppSettingsMenu.fontType, AppSettingsMenu.textSize));
-           contentPane.addLeft(label);
-           SearchField cardPick = new SearchField();
-           cardPick.setVisible(true);
-           for(String it : cardTypes){
-               cardPick.addItem(it);
-           }
-           contentPane.addRight(cardPick);
-           contentPane.addSpacer(Box.createVerticalStrut(20));
-           JLabel label2 = new JLabel("Card number");
-           label2.setForeground(Color.orange);
-           label2.setFont(new Font(AppSettingsMenu.fontStyle,
    AppSettingsMenu.fontType, AppSettingsMenu.textSize));
-           contentPane.addLeft(label2);
-           JTextField cardNumberTextField = new JTextField();
-           cardNumberTextField.setFont(new Font("TimesRoman", Font.PLAIN,
    20));
-           contentPane.addRight(cardNumberTextField);
-           contentPane.addSpacer(Box.createVerticalStrut(20));
-           JLabel label3 = new JLabel("Expiration date");
-           label3.setFont(new Font(AppSettingsMenu.fontStyle,
    AppSettingsMenu.fontType, AppSettingsMenu.textSize));
-           label3.setForeground(Color.orange);
-           contentPane.addLeft(label3);
-           SearchField datePick = new SearchField();
-           datePick.setVisible(true);
```

```
-           for(String it : mounth){
-               for(String it2 : year){
-                   datePick.addItem(it + "/" + it2);
-               }
-           }
-           contentPane.addRight(datePick);
-           contentPane.addSpacer(Box.createVerticalStrut(20));
-           JLabel label4 = new JLabel("Security Code");
-           label4.setFont(new Font(AppSettingsMenu.fontStyle,
    AppSettingsMenu.fontType, AppSettingsMenu.textSize));
-           label4.setForeground(Color.orange);
-           contentPane.addLeft(label4);
-           JTextField securityCodeTextField = new JTextField();
-           securityCodeTextField.setFont(new Font("TimesRoman", Font.PLAIN,
    20));
-           contentPane.addRight(securityCodeTextField);
-           contentPane.addSpacer(Box.createVerticalStrut(20));
-           JButton updateCardDataButton = new JButton("Updata card data");
-
-           updateCardDataButton.addActionListener(new ActionListener(){
-
-               @Override
-               public void actionPerformed(ActionEvent e) {
-                   String cardType = (String) cardPick.getSelectedItem();
-                   String cardNumber = cardNumberTextField.getText();
-                   String expriationDate = (String)
    datePick.getSelectedItem();
-                   String securityCode = securityCodeTextField.getText();
-
-                   Connector.insertCardData(cardType, cardNumber,
    expriationDate, securityCode);
-               }
-
-           });
-           contentPane.addRight(updateCardDataButton);
-           this.setContentPane(contentPane);
-           this.setDefaultCloseOperation(JDialog.HIDE_ON_CLOSE);
-           this.setLocationRelativeTo(null);
-       }
-   }
-
```

- CheckBox

```
-   package com.tourer.gui;
-
-   import javax.swing.ImageIcon;
-   import javax.swing.JCheckBox;
-
-   import java.awt.Font;
-   import java.awt.Image;
-   import java.awt.event.ActionListener;
-   import java.awt.event.ActionEvent;
-
-   public class CheckBox extends JCheckBox{
-       final static ImageIcon unCheckedIcon = new ImageIcon(new
    ImageIcon("Icons\\UnChecked.png").getImage().getScaledInstance(AppSetti
    ngsMenu.w, AppSettingsMenu.h, Image.SCALE_SMOOTH));
-       final static ImageIcon checkedIcon = new ImageIcon(new
    ImageIcon("Icons\\Checked.png").getImage().getScaledInstance(AppSetting
    sMenu.w, AppSettingsMenu.h, Image.SCALE_SMOOTH));
-       final static ImageIcon unCheckedIconRoll = new ImageIcon(new
    ImageIcon("Icons\\UnCheckedRoll.png").getImage().getScaledInstance(AppS
    ettingsMenu.w, AppSettingsMenu.h, Image.SCALE_SMOOTH));
-       final static ImageIcon checkedIconRoll = new ImageIcon(new
    ImageIcon("Icons\\CheckedRoll.png").getImage().getScaledInstance(AppSet
    tingsMenu.w, AppSettingsMenu.h, Image.SCALE_SMOOTH));
-
-       public CheckBox(String name){
-
-           super(name);
-            // Set default icon for checkbox
-           this.setIcon(unCheckedIcon);
-            // Set selected icon when checkbox state is selected
-           this.setSelectedIcon(checkedIcon);
-            // Set disabled icon for checkbox
-           this.setDisabledIcon(unCheckedIcon);
-            // Set disabled-selected icon for checkbox
-           this.setDisabledSelectedIcon(unCheckedIconRoll);
-            // Set checkbox icon when checkbox is pressed
-           this.setPressedIcon(checkedIcon);
-            // Set icon when a mouse is over the checkbox
-           this.setRolloverIcon(checkedIconRoll);
-            // Set icon when a mouse is over a selected checkbox
-           this.setRolloverSelectedIcon(checkedIconRoll);
```

```
-           this.setFont(new Font(AppSettingsMenu.fontStyle,
    AppSettingsMenu.fontType, AppSettingsMenu.textSize));
-
-           this.setOpaque(false);
-           this.setBorderPainted(false);
-           this.addActionListener(new ActionListener(){
-
-               @Override
-               public void actionPerformed(ActionEvent e) {
-
-                   if(CheckBox.this.isSelected()){
-                       //modify color scheme
-                   }
-
-               }
-
-           });
-       }
-   }
-
```

- ColorPanel

```
-   package com.tourer.gui;
-
-   import java.awt.Color;
-   import java.awt.Graphics;
-   import java.awt.GradientPaint;
-   import java.awt.Graphics2D;
-   import java.awt.RenderingHints;
-
-   import javax.swing.BorderFactory;
-   import javax.swing.JPanel;
-
-   public class ColorPanel extends JPanel{
-
-       Color c1;
-       Color c2;
-
-       public ColorPanel() {
-           super();
-           c1 =  MainFrame.gradientColor.getMainColor();
```

```
-           c2 = MainFrame.gradientColor.getSecondaryColor();
-           this.setBorder(BorderFactory.createLineBorder(Color.black));
-       }
-
-       public ColorPanel(GradientColor gc) {
-           super();
-           c1 =  gc.getMainColor();
-           c2 = gc.getSecondaryColor();
-           this.setBorder(BorderFactory.createLineBorder(Color.black));
-       }
-
-       @Override
-       protected void paintComponent(Graphics g){
-           super.paintComponent(g);
-
-           Graphics2D g2d = (Graphics2D) g;
-           g2d.setRenderingHint(RenderingHints.KEY_RENDERING,
    RenderingHints.VALUE_RENDER_QUALITY);
-           int w = getWidth();
-           int h = getHeight();
-
-           GradientPaint gp = new GradientPaint(0, 0, c1, 0, h, c2);
-           g2d.setPaint(gp);
-           g2d.fillRect(0, 0, w, h);
-       }
-
-   }
-
```

- CostumButton

```
-   package com.tourer.gui;
-   import javax.swing.JButton;
-   import javax.swing.ImageIcon;
-   import java.awt.event.MouseAdapter;
-   import java.awt.event.MouseEvent;
-
-   import java.awt.Image;
-
-   public class CostumButton extends JButton{
-
-       int w;
```

```
-       int h;
-       ImageIcon icon;
-       ImageIcon selectIcon;
-       String iconPath;
-       public  CostumButton(int w, int h, String iconPath) {
-           this.w = w;
-           this.h = h;
-           this.iconPath = iconPath;
-           icon = new ImageIcon(new
    ImageIcon(iconPath).getImage().getScaledInstance(w, h,
    Image.SCALE_SMOOTH));
-           selectIcon = new ImageIcon(new
    ImageIcon(iconPath).getImage().getScaledInstance(w + 20, h + 20,
    Image.SCALE_SMOOTH));
-           this.setIcon(icon);
-           this.setOpaque(false);
-           this.setContentAreaFilled(false);
-           this.setBorderPainted(false);
-           this.addMouseListener(new MouseAdapter(){
-               @Override
-               public void mouseEntered(MouseEvent event) {
-                   CostumButton.this.setIcon(CostumButton.this.selectIcon)
    ;
-
-               }
-               @Override
-               public void mouseExited(MouseEvent event){
-                   CostumButton.this.setIcon(CostumButton.this.icon);
-               }
-           });
-       }
-
-       public void updateIconSize(int w, int h){
-           ImageIcon icon = new ImageIcon(new
    ImageIcon(iconPath).getImage().getScaledInstance(w, h,
    Image.SCALE_SMOOTH));
-           this.setIcon(icon);
-       }
-       public void updateIcon(String path){
-           icon = new ImageIcon(new
    ImageIcon(path).getImage().getScaledInstance(w, h, Image.SCALE_SMOOTH));
-           selectIcon = new ImageIcon(new
    ImageIcon(path).getImage().getScaledInstance(w + 20, h + 20,
    Image.SCALE_SMOOTH));
-           this.setIcon(icon);
```

```
-         }
-     }
-
```

- GradientColor

```
-   package com.tourer.gui;
-
-   import java.awt.Color;
-   import java.util.Objects;
-
-   public class GradientColor {
-       Color mainColor;
-       Color secondaryColor;
-
-
-       public GradientColor() {
-       }
-
-       public GradientColor(Color mainColor, Color secondaryColor) {
-           this.mainColor = mainColor;
-           this.secondaryColor = secondaryColor;
-       }
-
-       public Color getMainColor() {
-           return this.mainColor;
-       }
-
-       public void setMainColor(Color mainColor) {
-           this.mainColor = mainColor;
-       }
-
-       public Color getSecondaryColor() {
-           return this.secondaryColor;
-       }
-
-       public void setSecondaryColor(Color secondaryColor) {
-           this.secondaryColor = secondaryColor;
-       }
-
-       public GradientColor mainColor(Color mainColor) {
-           setMainColor(mainColor);
-           return this;
```

```
-       }
-
-       public GradientColor secondaryColor(Color secondaryColor) {
-           setSecondaryColor(secondaryColor);
-           return this;
-       }
-
-       @Override
-       public boolean equals(Object o) {
-           if (o == this)
-               return true;
-           if (!(o instanceof GradientColor)) {
-               return false;
-           }
-           GradientColor gradientColor = (GradientColor) o;
-           return Objects.equals(mainColor, gradientColor.mainColor) &&
    Objects.equals(secondaryColor, gradientColor.secondaryColor);
-       }
-
-       @Override
-       public int hashCode() {
-           return Objects.hash(mainColor, secondaryColor);
-       }
-
-       @Override
-       public String toString() {
-           return "{" +
-               " mainColor='" + getMainColor() + "'" +
-               ", secondaryColor='" + getSecondaryColor() + "'" +
-               "}";
-       }
-
-   }
-
```

- GridPanel

```
-   package com.tourer.gui;
-
-
-
-   import java.awt.Component;
-   import java.awt.GridBagLayout;
```

```
-   import java.awt.GridBagConstraints;
-
-   public class GridPanel extends ColorPanel{
-       public static GridBagConstraints left;
-       public static GridBagConstraints right;
-       static{
-           left = new GridBagConstraints();
-           left.anchor = GridBagConstraints.EAST;
-           right = new GridBagConstraints();
-           right.weightx = 2.0;
-           right.fill = GridBagConstraints.HORIZONTAL;
-           right.gridwidth = GridBagConstraints.REMAINDER;
-       }
-       public GridPanel(){
-           this.setLayout(new GridBagLayout());
-       }
-
-       public GridPanel(GradientColor gColor){
-           super(gColor);
-           this.setLayout(new GridBagLayout());
-       }
-
-       public void addLeft(Component c){
-           this.add(c, left);
-       }
-
-       public void addRight(Component c){
-           this.add(c, right);
-       }
-       public void addSpacer(Component c){
-           this.addLeft(c);
-           this.addRight(c);
-       }
-   }
-
```

- IconResizer

```
-   package com.tourer.gui;
-
-   import java.awt.Component;
-   import java.awt.Container;
```

```
- import javax.swing.event.ChangeEvent;
- import javax.swing.event.ChangeListener;
-
- import com.tourer.App;
-
- public class IconResizer extends Resizer{
-
-     public static void changeIconSize ( Container container, int
  newIconHight, int newIconWidth)
-     {
-
-
-         for ( Component child :  container.getComponents () )
-         {
-             if ( child instanceof CostumButton ){
-                ((CostumButton) child).updateIconSize(newIconWidth,
  newIconHight);
-             }
-         }
-
-     }
-     public IconResizer(){
-         this.addChangeListener(new ChangeListener() {
-
-             @Override
-             public void stateChanged(ChangeEvent e) {
-                 int newIconHight = (int) (ButtonBox.buttonHeight
  *  (double)IconResizer.this.getValue() / 100);
-                 int newIconWidth = (int) (ButtonBox.buttonWidth
  *  (double)IconResizer.this.getValue() / 100);
-                 changeIconSize(App.buttonBox, newIconHight,
  newIconWidth);
-
-             }
-
-
-
-         });
-     }
- }
-
```

- LanguagePicker

```
- package com.tourer.gui;
-
- import javax.swing.border.TitledBorder;
- import javax.swing.border.Border;
- import java.awt.Component;
- import java.awt.Container;
- import java.awt.event.ActionListener;
- import java.util.HashMap;
- import java.util.Map;
-
- import javax.swing.JButton;
- import javax.swing.JComponent;
- import javax.swing.JLabel;
- import javax.swing.JTextField;
-
- import java.awt.event.ActionEvent;
- public class LanguagePicker extends SearchField{
-
-     static Map <String, String> englishToRomanian;
-     static Map <String, String> romanianToEnglish;
-
-     static{
-         englishToRomanian = new HashMap<String, String>();
-         englishToRomanian.put("Window", "Fereastra");
-         englishToRomanian.put("Security", "Securitate");
-         englishToRomanian.put("Text", "Text");
-         englishToRomanian.put("Settings", "Setari");
-         englishToRomanian.put("Text Size", "Dimensiunea Textului");
-         englishToRomanian.put("Icon Size", "Dimensiunea Iconitelor");
-         englishToRomanian.put("Language", "Limba");
-         englishToRomanian.put("Dark Mode", "Mod intunecat");
-         englishToRomanian.put("Notifications", "Notificari");
-         englishToRomanian.put("Resizing", "Redimensionare");
-         englishToRomanian.put("Longitude:", "Longitudine:");
-         englishToRomanian.put("Latitude:", "Latitudine:");
-         englishToRomanian.put("Location name:", "Numele locatiei");
-         englishToRomanian.put("Description:", "Descriere:");
-         romanianToEnglish = new HashMap<String, String>();
-         for(Map.Entry<String, String> entry :
  englishToRomanian.entrySet()){
-             romanianToEnglish.put(entry.getValue(), entry.getKey());
-         }
```

```
-
-       }
-       public static String getTranslatedVersion(String text, String
    language){
-
-           if(language == null)
-               return "-";
-           switch(language){
-               case "English":
-                   try{
-                       return romanianToEnglish.get(text);
-                   }catch(Exception e){
-                       return "-";
-                   }
-               case "Romana":
-                   try{
-                       return englishToRomanian.get(text);
-                   }catch (Exception e){
-                       return "-";
-                   }
-               default:
-                   return "-";
-           }
-       }
-
-       public static void updateLanguage ( Component component, String
    language)
-       {
-           if(component instanceof JComponent){
-
-
-               Border border = ((JComponent) component).getBorder();
-
-               if(border instanceof TitledBorder){
-                       String currentText = ((TitledBorder)
    border).getTitle();
-                       String translatedText =
    getTranslatedVersion(currentText, language);
-                       ((TitledBorder) border).setTitle(translatedText);
-               }
-           }
-
-           if(component instanceof JTextField){
-               String currentText = ((JTextField) component).getText();
```

```
-               String translatedText = getTranslatedVersion(currentText,
    language);
-               ((JTextField) component).setText(translatedText);
-           }
-           else
-           if(component instanceof JButton)
-           {
-               String currentText = ((JButton) component).getText();
-               String translatedText = getTranslatedVersion(currentText,
    language);
-               ((JButton) component).setText(translatedText);
-           }
-           else
-           if(component instanceof JLabel)
-           {
-               String currentText = ((JLabel) component).getText();
-               String translatedText = getTranslatedVersion(currentText,
    language);
-               ((JLabel) component).setText(translatedText);
-           }
-           else
-           if(component instanceof CheckBox){
-               String currentText = ((CheckBox) component).getText();
-               String translatedText = getTranslatedVersion(currentText,
    language);
-               ((CheckBox) component).setText(translatedText);
-           }
-           else
-           if ( component instanceof Container )
-           {
-               for ( Component child : ( ( Container ) component
    ).getComponents () )
-               {
-                   updateLanguage ( child, language );
-               }
-           }
-       }
-       public LanguagePicker(){
-
-           this.addActionListener(new ActionListener(){
-
-               @Override
-               public void actionPerformed(ActionEvent e) {
-                   String language = (String)
    LanguagePicker.this.getSelectedItem();
```

```
-                   updateLanguage(ButtonBox.userSettingsMenu, language);
-                   updateLanguage(ButtonBox.appMenuSettings, language);
-                   //bug when initializing windows
-                   //updateLanguage(AddButton.addLocationDialog, language);
-               }
-
-           });
-       }
-   }
-
```

- LocationButton

```
-   package com.tourer.gui;
-   import java.awt.event.ActionEvent;
-   import java.awt.event.ActionListener;
-
-
-   public class LocationButton extends CostumButton{
-
-       static LocationSearchField locationSearchField;
-       public final static String iconPath = "Icons\\Location.png";
-       public final static String darkIconPath = "Icons\\LocationDark.png";
-       public LocationButton(int w, int h, LocationSearchField
    locationSearchField) {
-           super(w, h, LocationButton.iconPath);
-           LocationButton.locationSearchField = locationSearchField;
-
-           this.addActionListener(new ActionListener(){
-
-               @Override
-               public void actionPerformed(ActionEvent e) {
-                   for(int i = 0; i < SearchField.allFields.size(); i++)
-                       SearchField.allFields.get(i).setVisible(false);
-                   LocationButton.locationSearchField.setVisible(true);
-                   //JTextField textLocal = (JTextField)
    LocationButton.locationSearchField.getEditor().getEditorComponent();
-                   // locationSearchField.requestFocus();
-                   // if(textLocal.getText().equals(""))
-                   //     textLocal.setText("Search a location");
-                   LocationButton.locationSearchField.requestFocus();
-               }
```

```
-
-           });
-       }
-
-
-   }
-
```

- LocationDescriptionDialog

```
-   package com.tourer.gui;
-
-   import java.awt.Image;
-
-   import javax.swing.Box;
-   import javax.swing.BoxLayout;
-   import javax.swing.ImageIcon;
-   import javax.swing.JButton;
-   import javax.swing.JDialog;
-   import javax.swing.JFileChooser;
-   import javax.swing.JLabel;
-   import javax.swing.JOptionPane;
-   import javax.swing.JScrollPane;
-   import javax.swing.JTextArea;
-   import javax.swing.JTextField;
-   import javax.swing.ScrollPaneConstants;
-   import javax.swing.SpringLayout;
-   import javax.swing.filechooser.FileNameExtensionFilter;
-
-   import com.tourer.App;
-   import com.tourer.gui.map.Location;
-   import com.tourer.jdbc.Connector;
-
-   import org.apache.commons.io.FileUtils;
-   import org.apache.commons.io.FilenameUtils;
-
-   import javafx.application.Platform;
-

-   import java.awt.Window;
-   import java.awt.Color;
-   import java.awt.Dimension;
-   import java.awt.Font;
```

```
- import java.awt.event.ActionListener;
- import java.io.File;
- import java.io.IOException;
- import java.sql.SQLException;
- import java.awt.event.ActionEvent;
-
- public class LocationDescriptionDialog extends JDialog{
-
-     public final static Dimension DIALOG_SIZE = new Dimension((int)
  MainFrame.screenSize.getWidth() * 2 / 3, (int)
  MainFrame.screenSize.getHeight() * 2 / 3);
-     public final static Dimension TEXTAREA_SIZE = new Dimension((int)
  MainFrame.screenSize.getWidth() * 2 / 3, ((int)
  MainFrame.screenSize.getHeight() * 2 / 3) * 2 / 3);
-     public ColorPanel contentPane;
-     public ImageIcon locationImage;
-     public Location location;
-     public JTextArea descriptionTextArea;
-     public JScrollPane textScrollPane;
-     public JButton updateLocation;
-     public JButton deleteLocation;
-     public CostumButton likeButton;
-     public static String username;
-     public CostumButton dislikeButton;
-     public static final String likePath = "Icons\\Like.png";
-     public static final String dislikePath = "Icons\\DisLike.png";
-     public static final String likeSelectedPath =
  "Icons\\LikeSelected.png";
-     public static final String dislikeSelectedPath =
  "Icons\\DisLikeSelected.png";
-     public static final Integer SPACER_SIZE = 10;
-     public JLabel likeCount;
-     public JLabel dislikeCount;
-     public JLabel locationPhoto;
-     public JLabel photo;
-     public JButton addPhotoButton;
-
-     public static final String defaultPhotoPath =
  "Icons\\DefaultPhoto.jpg";
-     public LocationDescriptionDialog(Window window){
-         super(window);
-
-         this.setSize(DIALOG_SIZE);
-         this.setPreferredSize(DIALOG_SIZE);
-         this.setResizable(false);
```

```
-           this.setVisible(false);
-
-           contentPane = new ColorPanel();
-           SpringLayout springLayout = new SpringLayout();
-           contentPane.setLayout(springLayout);
-           contentPane.setPreferredSize(new Dimension(this.getWidth() -
    100,this.getHeight() * 2 + 100));
-           this.setContentPane(contentPane);
-           this.setLocationRelativeTo(null);
-
-           photo = new JLabel("_");
-           ImageIcon photoIcon = new ImageIcon(new
    ImageIcon(defaultPhotoPath).getImage().getScaledInstance(this.getWidth(
    ), 400, Image.SCALE_SMOOTH));
-           photo.setIcon(photoIcon);
-           this.add(photo);
-           springLayout.putConstraint(SpringLayout.NORTH, photo,
    SPACER_SIZE, SpringLayout.NORTH, contentPane);
-           springLayout.putConstraint(SpringLayout.WEST, photo,
    SPACER_SIZE, SpringLayout.WEST, contentPane);
-           springLayout.putConstraint(SpringLayout.EAST, photo,
    -SPACER_SIZE, SpringLayout.EAST, contentPane);
-           JLabel description = new JLabel("Description");
-           description.setForeground(Color.orange);
-           description.setFont(new Font(AppSettingsMenu.fontStyle,
    AppSettingsMenu.fontType, AppSettingsMenu.textSize));
-
-           locationPhoto = new JLabel();
-           this.add(locationPhoto);
-           locationPhoto.setVisible(false);
-           this.add(description);
-           springLayout.putConstraint(SpringLayout.NORTH, description,
    SPACER_SIZE, SpringLayout.SOUTH, photo);
-           springLayout.putConstraint(SpringLayout.WEST, description,
    SPACER_SIZE, SpringLayout.WEST, contentPane);
-           descriptionTextArea = new JTextArea("");
-           descriptionTextArea.setFont(new Font(Font.DIALOG, Font.PLAIN,
    30));
-           descriptionTextArea.setEditable(false);
-           textScrollPane = new JScrollPane(descriptionTextArea);
-           textScrollPane.setHorizontalScrollBarPolicy(ScrollPaneConstants
    .HORIZONTAL_SCROLLBAR_AS_NEEDED);
-           textScrollPane.setVerticalScrollBarPolicy(ScrollPaneConstants.V
    ERTICAL_SCROLLBAR_ALWAYS);
```

```
-           textScrollPane.getVerticalScrollBar().setUI(new
    MyScrollbarUI(App.gradientColor, Color.white));
-           textScrollPane.getVerticalScrollBar().setUnitIncrement(20);
-           textScrollPane.getHorizontalScrollBar().setUI(new
    MyScrollbarUI(App.gradientColor, Color.white));
-           textScrollPane.getHorizontalScrollBar().setUnitIncrement(20);
-
-           textScrollPane.setPreferredSize(TEXTAREA_SIZE);
-
-           this.add(textScrollPane);
-           springLayout.putConstraint(SpringLayout.NORTH, textScrollPane,
    SPACER_SIZE, SpringLayout.SOUTH, description);
-           springLayout.putConstraint(SpringLayout.WEST, textScrollPane,
    SPACER_SIZE, SpringLayout.WEST, contentPane);
-           springLayout.putConstraint(SpringLayout.EAST, textScrollPane,
    -SPACER_SIZE, SpringLayout.EAST, contentPane);
-
-           Font buttonTextFont = new Font("Serif", Font.BOLD, 18);
-           addPhotoButton = new JButton("Change photo");
-           addPhotoButton.setFont(buttonTextFont);
-           addPhotoButton.setPreferredSize(new Dimension(this.getWidth() -
    50, 40));
-           addPhotoButton.addActionListener(new ActionListener(){
-
-               @Override
-               public void actionPerformed(ActionEvent e) {
-
-                   File photoDir = new File("UserPhotos\\" +
    Connector.USERNAME);
-                   if (!photoDir.exists()){
-                       photoDir.mkdirs();
-                   }
-
-                   JFileChooser chooser = new JFileChooser();
-                   FileNameExtensionFilter filter = new
    FileNameExtensionFilter(
-                       "JPEG file", "jpg", "jpeg", "png", "PNG file");
-                   chooser.setFileFilter(filter);
-                   int returnVal =
    chooser.showOpenDialog(UsserButton.userSettingsMenu);
-                   String photoname = location.getLongitude() + "---" +
    location.getLongitude();
-                   if(returnVal == JFileChooser.APPROVE_OPTION) {
-
```

```
-                       String filename =
    chooser.getSelectedFile().getAbsolutePath();
-                       File source = new File(filename);
-                       String extension = "." +
    FilenameUtils.getExtension(filename);
-                       File dest = new File("UserPhotos\\" +
    Connector.USERNAME + "\\" + photoname  + extension);
-
-                       File curpng = new File("UserPhotos\\" +
    Connector.USERNAME + "\\" + photoname +".png");
-                       if(curpng.exists()){
-                           curpng.delete();
-                       }
-                       File curjpg = new File("UserPhotos\\" +
    Connector.USERNAME + "\\" + photoname + ".jpg");
-                       if(curjpg.exists()){
-                           curjpg.delete();
-                       }
-                       try {
-                           FileUtils.copyFile(source, dest);
-                           try {
-                               Connector.updatePhoto(location.getName(),
    dest.getAbsolutePath());
-                               location.setPhoto(dest.getAbsolutePath());
-                               LocationDescriptionDialog.this.setVisible(f
    alse);
-                               LocationDescriptionDialog.this.updateLocati
    on(location);
-                               LocationDescriptionDialog.this.setVisible(t
    rue);
-                           } catch (SQLException e1) {
-
-                               e1.printStackTrace();
-                               JOptionPane.showMessageDialog(App.accountCr
    eationFrame, "Failed to update photo", "ERROR",
    JOptionPane.ERROR_MESSAGE);
-
-                           }
-
-
-                       } catch (IOException e1) {
-                           e1.printStackTrace();
-                       }
-                   }
-
```

```
-
-
-
-             }
-
-         });
-         this.add(addPhotoButton);
-         addPhotoButton.setVisible(false);
-         springLayout.putConstraint(SpringLayout.NORTH, addPhotoButton ,
    SPACER_SIZE, SpringLayout.SOUTH, textScrollPane);
-         springLayout.putConstraint(SpringLayout.WEST, addPhotoButton ,
    SPACER_SIZE, SpringLayout.WEST, contentPane);
-
-         JButton showOnMapButton = new JButton("Show on map and get
    directions");
-         showOnMapButton.setFont(buttonTextFont);
-         showOnMapButton.setPreferredSize(new Dimension(this.getWidth() -
    50, 40));
-         showOnMapButton.addActionListener(new ActionListener(){
-
-             @Override
-             public void actionPerformed(ActionEvent e) {
-                 Platform.runLater(new Runnable() {
-
-                     @Override
-                     public void run() {
-                         Double lat = location.getLatitude();
-                         Double lng = location.getLongitude();
-                         App.engine.executeScript("setTargetMarker(" +
    lat + ", " + lng + ");");
-                         App.mainFrame.requestFocus();
-                     }
-                 });
-
-
-             }
-         });
-         //setTargetMarker
-         this.add(showOnMapButton);
-         springLayout.putConstraint(SpringLayout.NORTH, showOnMapButton,
    SPACER_SIZE, SpringLayout.SOUTH, addPhotoButton);
-         springLayout.putConstraint(SpringLayout.WEST, showOnMapButton,
    SPACER_SIZE, SpringLayout.WEST, contentPane);
-         updateLocation = new JButton("Update location");
-         updateLocation.setFont(buttonTextFont);
```

```
-          updateLocation.setPreferredSize(new Dimension(this.getWidth() -
   50, 40));
-          updateLocation.addActionListener(new ActionListener(){
-
-              @Override
-              public void actionPerformed(ActionEvent e) {
-
-                  JTextField nameTextField = new JTextField();
-                  nameTextField.setText(location.getName());
-                  Object[] message = {
-                      "New name:", nameTextField
-                  };
-                  int result =
   JOptionPane.showConfirmDialog(LocationDescriptionDialog.this, message,
   "Save changes", JOptionPane.OK_CANCEL_OPTION);
-
-                  if(result == JOptionPane.OK_OPTION){
-                      if(Connector.modifyLocation( location.getLatitude(),
   location.getLongitude(), nameTextField.getText(),
   descriptionTextArea.getText()) == false){
-                          JOptionPane.showMessageDialog(UsserButton.userS
   ettingsMenu, "Failed to update location", "ERROR",
   JOptionPane.ERROR_MESSAGE);
-                      }
-                      else
-                      {
-                          UsserButton.userSettingsMenu.setVisible(false);
-                          try {
-                              UsserButton.userSettingsMenu.updateVisited(
   );
-                          } catch (SQLException e2) {
-                              // TODO Auto-generated catch block
-                              e2.printStackTrace();
-                          }
-                          UsserButton.userSettingsMenu.setVisible(true);
-                      }
-                  }
-
-              }
-
-
-          });
-
-
-          updateLocation.setVisible(false);
```

```
-
-
-           this.add(updateLocation);
-
-           springLayout.putConstraint(SpringLayout.NORTH, updateLocation,
    SPACER_SIZE, SpringLayout.SOUTH, showOnMapButton);
-           springLayout.putConstraint(SpringLayout.WEST, updateLocation,
    SPACER_SIZE, SpringLayout.WEST, contentPane);
-           deleteLocation = new JButton("Delete location");
-           deleteLocation.setFont(buttonTextFont);
-           deleteLocation.setPreferredSize(new Dimension(this.getWidth() -
    50, 40));
-           deleteLocation.addActionListener(new ActionListener(){
-
-               @Override
-               public void actionPerformed(ActionEvent e) {
-                   int rez =
    JOptionPane.showConfirmDialog(LocationDescriptionDialog.this, "Do you
    really want to delete this location?", "", JOptionPane.YES_NO_OPTION);
-
-                   if(rez == JOptionPane.OK_OPTION){
-                       try {
-                           Connector.deleteLocation(location.getName());
-                           UsserButton.userSettingsMenu.setVisible(false);
-                           try {
-                               UsserButton.userSettingsMenu.updateVisited(
    );
-                           } catch (SQLException e2) {
-                               // TODO Auto-generated catch block
-                               e2.printStackTrace();
-                           }
-                           UsserButton.userSettingsMenu.setVisible(true);
-                           LocationDescriptionDialog.this.setVisible(false
    );
-                       } catch (SQLException e1) {
-                           JOptionPane.showMessageDialog(UsserButton.userS
    ettingsMenu, "Failed to delete location", "ERROR",
    JOptionPane.ERROR_MESSAGE);
-                           e1.printStackTrace();
-                       }
-                   }
-
-               }
-
-           });
```

```
-
-           deleteLocation.setVisible(false);
-           this.add(deleteLocation);
-
-           springLayout.putConstraint(SpringLayout.NORTH, deleteLocation,
    SPACER_SIZE, SpringLayout.SOUTH, updateLocation);
-           springLayout.putConstraint(SpringLayout.WEST, deleteLocation,
    SPACER_SIZE, SpringLayout.WEST, contentPane);
-           likeButton = new CostumButton(80, 80, likePath);
-
-
-
-           likeButton.addActionListener(new ActionListener(){
-
-               @Override
-               public void actionPerformed(ActionEvent e) {
-                   String name = location.getName();
-                   int like = location.getLikes();
-                   try {
-                       Connector.modifylikeState(name, username, true);
-                       if(location.getUserlikes().contains(Connector.USERN
    AME) == false)
-                       {
-                           like++;
-                           likeButton.updateIcon(likeSelectedPath);
-                           location.getUserlikes().add(Connector.USERNAME)
    ;
-                       }
-                       else
-                       {
-                           like--;
-                           likeButton.updateIcon(likePath);
-                           location.getUserlikes().remove(Connector.USERNA
    ME);
-
-                       }
-
-                       Connector.like(name, username, like);
-
-                       location.setLikes(like);
-                       LocationDescriptionDialog.this.likeCount.setText(""
    + location.getLikes());
-                   } catch (SQLException e1) {
-                       JOptionPane.showMessageDialog(App.accountCreationFr
    ame, Connector.ERROR_LIKE_UPDATE, "ERROR", JOptionPane.ERROR_MESSAGE);
```

```
-                       e1.printStackTrace();
-                   }
-
-
-
-               }
-
-           });
-
-
-           dislikeButton = new CostumButton(80, 80, dislikePath);
-
-
-           dislikeButton.addActionListener(new ActionListener(){
-
-               @Override
-               public void actionPerformed(ActionEvent e) {
-                   // TODO Auto-generated method stub
-                   String name = location.getName();
-                   int dislikes = location.getDislikes();
-                   try {
-                       Connector.modifylikeState(name, username, false);
-                       if(location.getUserdislikes().contains(Connector.US
    ERNAME) == false)
-                       {
-                           dislikes++;
-                           dislikeButton.updateIcon(dislikeSelectedPath);
-                           location.getUserdislikes().add(Connector.USERNA
    ME);
-                       }
-                       else
-                       {
-                           dislikes--;
-                           dislikeButton.updateIcon(dislikePath);
-                           location.getUserdislikes().remove(Connector.USE
    RNAME);
-
-                       }
-
-                       Connector.dislike(name, username, dislikes);
-                       location.setDislikes(dislikes);
-
-                       LocationDescriptionDialog.this.dislikeCount.setText
    ("" + location.getDislikes());
-                   } catch (SQLException e1) {
```

```
-                     JOptionPane.showMessageDialog(App.accountCreationFr
   ame, Connector.ERROR_LIKE_UPDATE, "ERROR", JOptionPane.ERROR_MESSAGE);
-                     e1.printStackTrace();
-                 }
-
-
-
-             }
-
-         });
-
-         this.add(likeButton);
-         this.add(dislikeButton);
-         springLayout.putConstraint(SpringLayout.NORTH, likeButton,
   SPACER_SIZE, SpringLayout.SOUTH, deleteLocation);
-         springLayout.putConstraint(SpringLayout.NORTH, dislikeButton,
   SPACER_SIZE + 25, SpringLayout.SOUTH, deleteLocation);
-         springLayout.putConstraint(SpringLayout.WEST, likeButton,
   this.getWidth() / 3 + 40, SpringLayout.WEST, contentPane);
-         springLayout.putConstraint(SpringLayout.WEST, dislikeButton, 0,
   SpringLayout.EAST, likeButton);
-
-         likeCount = new JLabel("-");
-         likeCount.setForeground(Color.orange);
-         likeCount.setFont(new Font(AppSettingsMenu.fontStyle,
   AppSettingsMenu.fontType, AppSettingsMenu.textSize));
-         dislikeCount = new JLabel("-");
-         dislikeCount.setForeground(Color.orange);
-         dislikeCount.setFont(new Font(AppSettingsMenu.fontStyle,
   AppSettingsMenu.fontType, AppSettingsMenu.textSize));
-         this.add(likeCount);
-         this.add(dislikeCount);
-         springLayout.putConstraint(SpringLayout.NORTH, likeCount,
   SPACER_SIZE, SpringLayout.SOUTH, likeButton);
-         springLayout.putConstraint(SpringLayout.NORTH, dislikeCount,
   SPACER_SIZE, SpringLayout.SOUTH, likeButton);
-         springLayout.putConstraint(SpringLayout.WEST, likeCount,
   this.getWidth() / 3 + 40 + 70, SpringLayout.WEST, contentPane);
-         springLayout.putConstraint(SpringLayout.WEST, dislikeCount, 80,
   SpringLayout.EAST, likeCount);
-         JScrollPane scrolableContentPane = new
   JScrollPane(this.getContentPane());
-
-         this.setContentPane(scrolableContentPane);
-     }
```

```
-
-       public void updateLocation(Location location){
-           this.location = location;
-           String photoPath = location.getPhoto();
-           if(photoPath.equals("") == false){
-               ImageIcon imageIcon =  new ImageIcon(new
    ImageIcon(photoPath).getImage().getScaledInstance(this.getWidth(), 400,
    Image.SCALE_SMOOTH));
-               photo.setIcon(imageIcon);
-
-           }
-           else
-           {
-               ImageIcon imageIcon =  new ImageIcon(new
    ImageIcon(defaultPhotoPath).getImage().getScaledInstance(this.getWidth(
    ), 400, Image.SCALE_SMOOTH));
-               photo.setIcon(imageIcon);
-           }
-           this.descriptionTextArea.setText(location.getDescription());
-           this.setTitle(location.getName());
-           this.likeCount.setText("" + location.getLikes());
-           this.dislikeCount.setText("" + location.getDislikes());
-           ImageIcon photo = new ImageIcon();
-           this.locationPhoto.setIcon(photo);
-           this.locationPhoto.setVisible(true);
-       }
-
-       public void updatelikedislikeicons(){
-           if(location.getUserlikes().contains(Connector.USERNAME) ==
    false){
-               likeButton.updateIcon(likePath);
-           }
-           else
-           {
-               likeButton.updateIcon(likeSelectedPath);
-           }
-
-           if(location.getUserdislikes().contains(Connector.USERNAME) ==
    false){
-               dislikeButton.updateIcon(dislikePath);
-           }
-           else
-           {
-               dislikeButton.updateIcon(dislikeSelectedPath);
-           }
```

```
-       }
-   }
-
```

- LocationSearchField

```
-   package com.tourer.gui;
-
-   public class LocationSearchField extends SearchField{
-
-       public LocationSearchField() {
-
-           SearchField.allFields.add(this);
-       }
-   }
-
```

- MainFrame

```
-   package com.tourer.gui;
-
-   import java.awt.Toolkit;
-   import java.awt.Dimension;
-   import java.awt.BorderLayout;
-   import java.awt.Color;
-
-   import javax.swing.JFrame;
-   import javax.swing.border.LineBorder;
-
-
-   public class MainFrame extends JFrame {
-       public final static Dimension screenSize =
    Toolkit.getDefaultToolkit().getScreenSize();
-       static GradientColor gradientColor;
-       final static LineBorder nullBorder = new LineBorder(Color.black, 0,
    false);
-       public MainFrame(GradientColor gradientColor) {
-           MainFrame.gradientColor = gradientColor;
-           this.setLayout(new BorderLayout());
```

```
-           this.setSize(new Dimension(MainFrame.screenSize.width / 2,
    MainFrame.screenSize.height / 2));
-           this.setLocationRelativeTo(null);
-           this.setExtendedState(JFrame.MAXIMIZED_BOTH);
-           this.setLocationRelativeTo(null);
-           this.setVisible(true);
-
-
-       }
-
-       public void update(){
-           this.pack();
-           this.revalidate();
-           this.repaint();
-
-           this.setSize(new Dimension(MainFrame.screenSize.width / 2,
    MainFrame.screenSize.height / 2));
-           this.setLocationRelativeTo(null);
-           this.setExtendedState(JFrame.MAXIMIZED_BOTH);
-
-           this.setVisible(true);
-       }
-
-
-


-   }
-
```

- MyListCellRenderer

```
-   package com.tourer.gui;
-
-   import java.awt.Component;
-   import java.awt.Font;
-
-   import javax.swing.BorderFactory;
-   import javax.swing.JLabel;
-   import javax.swing.JList;
-   import javax.swing.ListCellRenderer;
-   import javax.swing.border.Border;
-
```

```
- import java.awt.Color;
-
- public class MyListCellRenderer implements ListCellRenderer{
-
-
-     Border lineBorder = BorderFactory.createLineBorder(Color.BLACK, 4,
  true);
-     Border emptyBorder = BorderFactory.createEmptyBorder(2, 2, 2, 2);
-
-     @Override
-     public Component getListCellRendererComponent(JList jList, Object
  value,
-             int index, boolean isSelected, boolean cellHasFocus) {
-         JLabel jlblCell = new JLabel(value.toString());
-         jlblCell.setFont(new Font(AppSettingsMenu.fontStyle,
  AppSettingsMenu.fontType, AppSettingsMenu.textSize));
-
-         if (isSelected) {
-             jlblCell.setOpaque(true);
-
-             jlblCell.setForeground(jList.getSelectionForeground());
-             jlblCell.setBackground(jList.getSelectionBackground());
-
-         } else {
-             jlblCell.setOpaque(false);
-
-             jlblCell.setForeground(jList.getForeground());
-             jlblCell.setBackground(jList.getBackground());
-         }
-
-         jlblCell.setBorder(cellHasFocus ? lineBorder : emptyBorder);
-
-         return jlblCell;
-     }
- }
```

- MyScrollbarUI

```
package com.tourer.gui;
```

```
import java.awt.Graphics2D;
import java.awt.Color;
import java.awt.Image;
import java.awt.GradientPaint;
import java.awt.RenderingHints;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Rectangle;
import java.awt.image.BufferedImage;

import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.plaf.metal.MetalScrollBarUI;

/**
    *
    * @since 2.0
    * Main class of the app that contains the main method
    */
class MyScrollbarUI extends MetalScrollBarUI {

    /**
     * Custom image
     */
    static class FauxImage {

        static public Image create(int w, int h, GradientColor gradient) {
            BufferedImage bi = new BufferedImage(
                w, h, BufferedImage.TYPE_INT_ARGB);
            Graphics2D g2d = bi.createGraphics();
            g2d.setRenderingHint(RenderingHints.KEY_RENDERING,
RenderingHints.VALUE_RENDER_QUALITY);

            GradientPaint gp = new GradientPaint(0, 0, gradient.getMainColor(), 0,
h, gradient.getSecondaryColor());
            g2d.setPaint(gp);
            g2d.fillRect(0, 0, w, h);
            g2d.dispose();
            return bi;
        }
    }
    private Image imageThumb, imageTrack;
    private JButton b = new JButton() {

        @Override
```

```
        public Dimension getPreferredSize() {
            return new Dimension(0, 0);
        }

    };



    /**
     * Constructor of the class
     * @param c1 main color
     * @param c2 secondary color
     * @param c3 third color
     */
    public MyScrollbarUI(GradientColor gradient, Color c3) {
        imageThumb = FauxImage.create(32, 32, gradient);
        imageTrack = FauxImage.create(32, 32, new GradientColor(c3, c3));
    }


    /**
     * @param g
     * @param c
     * @param r
     */
    @Override
    protected void paintThumb(Graphics g, JComponent c, Rectangle r) {

        ((Graphics2D) g).drawImage(imageThumb,
            r.x, r.y, r.width, r.height, null);
    }


    /**
     * @param g
     * @param c
     * @param r
     */
    @Override
    protected void paintTrack(Graphics g, JComponent c, Rectangle r) {
        ((Graphics2D) g).drawImage(imageTrack,
            r.x, r.y, r.width, r.height, null);
    }
```

```
    /**
     * @param orientation
     * @return JButton
     */
    @Override
    protected JButton createDecreaseButton(int orientation) {

        return b;
    }


    /**
     * @param orientation
     * @return JButton
     */
    @Override
    protected JButton createIncreaseButton(int orientation) {

        return b;
    }
}
```

- MySliderUI

```
-   package com.tourer.gui;
-
-   import javax.swing.JSlider;
-   import javax.swing.plaf.basic.BasicSliderUI;
-

-   import java.awt.Shape;
-   import java.awt.Graphics2D;
-
-   import java.awt.Graphics;
-   import java.awt.Color;
-

-   import java.awt.geom.RoundRectangle2D;
-   public class MySliderUI extends BasicSliderUI {
-
-       private static final int TRACK_HEIGHT = 8;
-       private static final int TRACK_WIDTH = 8;
-       private static final int TRACK_ARC = 5;
```

```
-
-       private final RoundRectangle2D.Float trackShape = new
    RoundRectangle2D.Float();
-
-
-       public MySliderUI(JSlider slider) {
-           super(slider);
-       }
-
-       private boolean isHorizontal() {
-           return slider.getOrientation() == JSlider.HORIZONTAL;
-       }
-       @Override
-       protected void calculateTrackRect() {
-           super.calculateTrackRect();
-           if (isHorizontal()) {
-               trackRect.y = trackRect.y + (trackRect.height - TRACK_HEIGHT)
    / 2;
-               trackRect.height = TRACK_HEIGHT;
-           } else {
-               trackRect.x = trackRect.x + (trackRect.width - TRACK_WIDTH) /
    2;
-               trackRect.width = TRACK_WIDTH;
-           }
-           trackShape.setRoundRect(trackRect.x, trackRect.y,
    trackRect.width, trackRect.height, TRACK_ARC, TRACK_ARC);
-       }
-
-       @Override
-       protected void calculateThumbLocation() {
-           super.calculateThumbLocation();
-           if (isHorizontal()) {
-               thumbRect.y = trackRect.y + (trackRect.height -
    thumbRect.height) / 2;
-           } else {
-               thumbRect.x = trackRect.x + (trackRect.width -
    thumbRect.width) / 2;
-           }
-       }
-
-       @Override
-           public void paintTrack(final Graphics g) {
-               Graphics2D g2 = (Graphics2D) g;
-               Shape clip = g2.getClip();
-
```

```
-               boolean horizontal = isHorizontal();
-               boolean inverted = slider.getInverted();
-
-               // Paint shadow.
-               g2.setColor(new Color(170, 170 ,170));
-               g2.fill(trackShape);
-
-               // Paint track background.
-               g2.setColor(new Color(200, 200 ,200));
-               g2.setClip(trackShape);
-               trackShape.y += 1;
-               g2.fill(trackShape);
-               trackShape.y = trackRect.y;
-
-               g2.setClip(clip);
-
-               // Paint selected track.
-               if (horizontal) {
-                   boolean ltr =
    slider.getComponentOrientation().isLeftToRight();
-                   if (ltr) inverted = !inverted;
-                   int thumbPos = thumbRect.x + thumbRect.width / 2;
-                   if (inverted) {
-                       g2.clipRect(0, 0, thumbPos, slider.getHeight());
-                   } else {
-                       g2.clipRect(thumbPos, 0, slider.getWidth() -
    thumbPos, slider.getHeight());
-                   }
-
-               } else {
-                   int thumbPos = thumbRect.y + thumbRect.height / 2;
-                   if (inverted) {
-                       g2.clipRect(0, 0, slider.getHeight(), thumbPos);
-                   } else {
-                       g2.clipRect(0, thumbPos, slider.getWidth(),
    slider.getHeight() - thumbPos);
-                   }
-               }
-               g2.setColor(Color.ORANGE);
-               g2.fill(trackShape);
-               g2.setClip(clip);
-           }
-
-
-       // @Override
```

```
-       // public void paintThumb(Graphics g) {
-
-       //     Rectangle knobBounds = thumbRect;
-       //     int w = knobBounds.width;
-       //     int h = knobBounds.height;
-
-       //     // Create graphics copy.
-       //     Graphics2D g2d = (Graphics2D) g.create();
-
-       //     // Create default thumb shape.
-       //     Shape thumbShape = new Ellipse2D.Double(0, 0, w-1, h-1);
-
-       //     // Draw thumb.
-       //     g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
-       //         RenderingHints.VALUE_ANTIALIAS_ON);
-       //     g2d.translate(knobBounds.x, knobBounds.y);
-
-       //     g2d.setColor(AppSettingsMenu.PURPLE_COLOR);
-       //     g2d.fill(thumbShape);
-
-       //     g2d.setColor(MainFrame.gradientColor.getSecondaryColor());
-       //     g2d.draw(thumbShape);
-
-       //     // Dispose graphics.
-       //     g2d.dispose();
-       // }
-   }
```

- ReloadButton

```
package com.tourer.gui;

import java.awt.event.ActionListener;

import com.tourer.App;

import javafx.application.Platform;

import java.awt.event.ActionEvent;
```

```java
public class ReloadButton extends CostumButton{

    public final static String iconPath = "Icons\\Reload.png";
    public final static String darkIconPath = "Icons\\ReloadDark.png";



    public ReloadButton(int w, int h) {
        super(w, h, ReloadButton.iconPath);

        this.addActionListener(new ActionListener(){

            @Override
            public void actionPerformed(ActionEvent e) {
                Platform.runLater(new Runnable() {

                    @Override
                    public void run() {
                        // TODO Auto-generated method stub

                        App.reloadMap();
                    }

                });


            }

        });
    }


}
```

- Resizer

```
-   package com.tourer.gui;
-
-   import javax.swing.JSlider;
-
```

```
-   import java.awt.Font;
-
-   public class Resizer extends JSlider{
-
-       public Resizer(){
-           super(50, 150, 100);
-           this.setPaintTicks(true);
-           this.setMinorTickSpacing(5);
-           this.setMajorTickSpacing(25);
-           this.setPaintLabels(true);
-           this.setOpaque(false);
-           this.setBorder(MainFrame.nullBorder);
-           this.setFont(new Font(AppSettingsMenu.fontStyle,
    AppSettingsMenu.fontType, 20));
-           this.setUI(new MySliderUI(this));
-       }
-   }
-
```

- SearchButton

```
-   package com.tourer.gui;
-
-   import java.awt.event.ActionListener;
-   import java.awt.event.ActionEvent;
-
-   public class SearchButton extends CostumButton{
-
-       public final static String iconPath = "Icons\\Search.png";
-       public final static String darkIconPath = "Icons\\SearchDark.png";
-
-       static UserSearchField userSearchField;
-
-       public SearchButton(int w, int h, UserSearchField userSearchField){
-           super(w, h, SearchButton.iconPath);
-           SearchButton.userSearchField = userSearchField;
-
-           this.addActionListener(new ActionListener(){
-
-               @Override
-               public void actionPerformed(ActionEvent e) {
-                   for(int i = 0; i < SearchField.allFields.size(); i++)
```

```
-                       SearchField.allFields.get(i).setVisible(false);
-                   SearchButton.userSearchField.setVisible(true);
-                   // JTextField textLocal = (JTextField)
   SearchButton.userSearchField.getEditor().getEditorComponent();
-                   // if(textLocal.getText().equals(""))
-                   //     textLocal.setText("Search a user");
-                   SearchButton.userSearchField.requestFocus();
-               }
-
-           });
-
-
-       }
-
-
-   }
-
```

- SearchField

```
-   package com.tourer.gui;
-
-   import java.awt.Font;
-   import java.awt.Dimension;
-   import java.awt.EventQueue;
-   import java.awt.event.KeyAdapter;
-   import java.awt.event.KeyEvent;
-   import java.awt.event.ItemListener;
-   import java.awt.event.ItemEvent;
-   import java.sql.SQLException;
-   import java.util.Collections;
-   import java.util.Vector;
-
-   import javax.swing.DefaultComboBoxModel;
-   import javax.swing.JComboBox;
-   import javax.swing.JOptionPane;
-   import javax.swing.JTextField;
-
-   import com.tourer.App;
-   import com.tourer.gui.map.Location;
-   import com.tourer.jdbc.*;
-
```

```
-   import javafx.application.Platform;
-
-   public class SearchField extends JComboBox<String>{
-
-       Vector <String> latestSearches;
-       static Vector <SearchField> allFields = new Vector <SearchField>();
-       public boolean hide_flag = false;
-       JTextField search;
-       public SearchField(){
-           this.setVisible(false);
-           this.setEditable(true);
-           this.setModel(new DefaultComboBoxModel<String>());
-           latestSearches = new Vector<String>();
-           this.setFont(new Font("TimesRoman", Font.PLAIN, 20));
-           search = (JTextField) super.getEditor().getEditorComponent();
-           search.addKeyListener(new KeyAdapter(){
-               public void keyTyped(KeyEvent e){
-                       EventQueue.invokeLater(new Runnable() {
-
-                           @Override
-                           public void run() {
-                               String text =
    SearchField.this.search.getText();
-                               if(text.length() == 0){
-                                   SearchField.super.hidePopup();
-                                   SearchField.this.setModel(new
    DefaultComboBoxModel<String>(SearchField.this.latestSearches), "");
-                               }
-                               else
-                               {
-                                   if(SearchField.this instanceof
    UserSearchField){
-                                       try {
-                                           latestSearches =
    Connector.getUserList(text);
-                                       } catch (SQLException e) {
-                                           // TODO Auto-generated catch
    block
-                                           e.printStackTrace();
-                                       }
-                                   }
-                                   else
-                                   if(SearchField.this instanceof
    LocationSearchField){
-                                       try {
```

```
-                                           latestSearches =
    Connector.getLocationList(text);
-                                       } catch (SQLException e) {
-                                           // TODO Auto-generated catch
    block
-                                           e.printStackTrace();
-                                       }
-                                   }
-                                   DefaultComboBoxModel<String> m =
    SearchField.this.getSuggestedModel(SearchField.this.latestSearches,
    text);
-                                   if(m.getSize() == 0 ||
    SearchField.this.hide_flag){
-                                       SearchField.super.hidePopup();
-                                       SearchField.this.hide_flag = false;
-                                   }
-                                   else
-                                   {
-                                       SearchField.this.setModel(m, text);
-                                       SearchField.this.showPopup();
-                                   }
-                               }
-
-                           }
-
-                       });
-               }
-               public void keyPressed(KeyEvent e){
-                   String text = SearchField.this.search.getText();
-                   int code = e.getKeyCode();
-                   if(code == KeyEvent.VK_ENTER){
-                       if(!SearchField.this.latestSearches.contains(text))
    {
-                           SearchField.this.latestSearches.addElement(text
    );
-                           Collections.sort(SearchField.this.latestSearche
    s);
-                           setModel(SearchField.this.getSuggestedModel(Sea
    rchField.this.latestSearches, text), text);
-                       }
-                       hide_flag = true;
-                   }
-                   else
-                   if(code==KeyEvent.VK_ESCAPE){
-                       hide_flag = true;
```

```
-                 }
-                 else
-                 if(code==KeyEvent.VK_RIGHT){
-                     for(int i = 0; i <
SearchField.this.latestSearches.size(); i++){
-                         String str =
SearchField.this.latestSearches.elementAt(i);
-                         if(str.startsWith(text)){
-                             SearchField.super.setSelectedIndex(-1);
-                             SearchField.this.search.setText(str);
-                             return;
-                         }
-                     }
-                 }
-             }
-         });
-         this.addItemListener(new ItemListener(){
-             public void itemStateChanged(ItemEvent event) {
-                 if (event.getStateChange() == ItemEvent.SELECTED) {
-
-                     if(SearchField.super.getSelectedIndex() != -1)
-                         if(SearchField.this instanceof UserSearchField){
-                             //System.out.println("selected");
-                             try {
-                                 UsserButton.userSettingsMenu.showOtherU
ser((String) SearchField.super.getSelectedItem());
-
-                             } catch (SQLException e) {
-                                 JOptionPane.showMessageDialog(UsserButt
on.userSettingsMenu, "Couldn't load user: "
+  SearchField.super.getSelectedItem() , "ERROR",
JOptionPane.ERROR_MESSAGE);
-                                 e.printStackTrace();
-                             }
-                         }
-                         else
-                         if(SearchField.this instanceof
LocationSearchField){
-                             //System.out.print("selected2");
-                             String locationName = (String)
SearchField.super.getSelectedItem();
-                             try {
-                                 Location searchedLocation =
Connector.getFirstLocationByName(locationName);
-                                 Platform.runLater(new Runnable() {
```

```
-
-                                       @Override
-                                       public void run() {
-                                           Double lat =
    searchedLocation.getLatitude();
-                                           Double lng =
    searchedLocation.getLongitude();
-                                           App.engine.executeScript("setTa
    rgetMarker(" + lat + ", " + lng + ");");
-                                           App.mainFrame.requestFocus();
-                                       }
-                                   });
-                               } catch (SQLException e) {
-                                   JOptionPane.showMessageDialog(UsserButt
    on.userSettingsMenu, "Failed to mark location" , "ERROR",
    JOptionPane.ERROR_MESSAGE);
-                                   e.printStackTrace();
-                               }
-
-                           }
-                   }
-               }
-           });
-           this.setModel(new DefaultComboBoxModel<String>(latestSearches),
    "");
-       }
-       private void setModel(DefaultComboBoxModel<String> mdl, String str){
-           SearchField.super.setModel(mdl);
-           SearchField.super.setSelectedIndex(-1);
-           SearchField.this.search.setText(str);
-       }
-
-       private  DefaultComboBoxModel<String> getSuggestedModel(Vector
    <String> list, String text){
-           DefaultComboBoxModel<String> m = new
    DefaultComboBoxModel<String>();
-           for(String s : list){
-               if(s.toLowerCase().contains(text.toLowerCase()))
-                   m.addElement(s);
-           }
-           return m;
-       }
-
-       @Override
-       public Dimension getMaximumSize() {
```

```
-           Dimension dim = super.getMaximumSize();
-           dim.height = getPreferredSize().height;
-           dim.width = getPreferredSize().width;
-           return dim;
-       }
-
-       @Override
-       public void addItem(String item){
-           latestSearches.add(item);
-           this.setModel(new DefaultComboBoxModel<String>(latestSearches),
    "");
-       }
-   }
-
```

- SettingsButton

```
-   package com.tourer.gui;
-
-   import java.awt.event.ActionListener;
-   import java.awt.event.ActionEvent;
-
-   public class SettingsButton extends CostumButton{
-
-       public final static String iconPath = "Icons\\Settings.png";
-       public final static String darkIconPath = "Icons\\SettingsDark.png";
-
-       static AppSettingsMenu appMenuSettings;
-
-       public SettingsButton(int w, int h, AppSettingsMenu appMenuSettings)
    {
-           super(w, h, SettingsButton.iconPath);
-           SettingsButton.appMenuSettings = appMenuSettings;
-           this.addActionListener(new ActionListener(){
-
-               @Override
-               public void actionPerformed(ActionEvent e) {
-                   UsserButton.userSettingsMenu.setVisible(false);
-                   SettingsButton.appMenuSettings.setVisible(true);
-
-               }
-
```

```
-           });
-       }
-
-
-   }
-
```

- SettingsMenu

```
-   package com.tourer.gui;
-
-   import java.awt.Dimension;
-
-   import javax.swing.JFrame;
-   import java.awt.BorderLayout;
-
-   public class SettingsMenu extends JFrame{
-       static BorderLayout layout = new BorderLayout();
-       public SettingsMenu() {
-
-           this.setSize(new Dimension((MainFrame.screenSize.width * 5) / 6 ,
    (MainFrame.screenSize.height * 5) / 6));
-           this.setLayout(layout);
-           this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
-           this.setLocationRelativeTo(null);
-           this.setResizable(false);
-           this.setVisible(false);
-
-       }
-
-       public void update(){
-           this.pack();
-           this.revalidate();
-           this.repaint();
-           this.setSize(new Dimension((MainFrame.screenSize.width * 5) / 6 ,
    (MainFrame.screenSize.height * 5) / 6));
-           this.setLocationRelativeTo(null);
-
-       }
-
-   }
-
```

- TextResizer

```
-   package com.tourer.gui;
-
-
-   import java.awt.Component;
-   import java.awt.Container;
-   import java.awt.Font;
-
-   import javax.swing.event.ChangeEvent;
-   import javax.swing.event.ChangeListener;
-
-   public class TextResizer extends Resizer{
-
-       public static void changeFont ( Component component, Font font )
-       {
-           component.setFont ( font );
-           if ( component instanceof Container )
-           {
-               for ( Component child : ( ( Container ) component
    ).getComponents () )
-               {
-                   changeFont ( child, font );
-               }
-           }
-       }
-
-       public TextResizer(){
-           this.addChangeListener(new ChangeListener() {
-
-               @Override
-               public void stateChanged(ChangeEvent e) {
-
-                   int newTextSize = (int) (AppSettingsMenu.textSize
    *  (double)TextResizer.this.getValue() / 100);
-                   Font newFont = new Font(AppSettingsMenu.fontStyle,
    AppSettingsMenu.fontType, newTextSize);
-                   changeFont(ButtonBox.userSettingsMenu, newFont);
-                   changeFont(ButtonBox.appMenuSettings, newFont);
-                   changeFont(ButtonSettings.cardDataDialog, newFont);
-                   changeFont(AddButton.addLocationDialog, newFont);
-               }
```

```
-
-
-
-           });
-       }
-   }
-
```

- UserSearchField

```
-   package com.tourer.gui;
-
-   public class UserSearchField extends SearchField{
-
-       public UserSearchField() {
-
-           SearchField.allFields.add(this);
-       }
-
-   }
-
```

- UserSettingsMenu

```
-   package com.tourer.gui;
-
-   import com.tourer.App;
-   import com.tourer.gui.map.Location;
-   import com.tourer.jdbc.Connector;
-
-   import java.awt.Color;
-   import java.awt.BorderLayout;
-   import java.awt.Dimension;
-   import java.awt.Font;
-   import java.awt.Toolkit;
-   import java.awt.event.MouseAdapter;
-   import java.awt.event.MouseEvent;
-
-   import java.awt.image.BufferedImage;
-   import java.io.File;
```

```
-   import java.io.IOException;
-   import java.sql.SQLException;
-   import java.util.Vector;
-
-   import javax.swing.JLabel;
-   import javax.swing.JList;
-   import javax.swing.JScrollPane;
-   import javax.swing.ScrollPaneConstants;
-   import javax.swing.event.ListSelectionEvent;
-   import javax.swing.event.ListSelectionListener;
-   import javax.swing.filechooser.FileNameExtensionFilter;
-   import javax.imageio.ImageIO;
-   import javax.swing.ImageIcon;
-   import javax.swing.JFileChooser;
-
-   import java.awt.Image;
-
-   import org.apache.commons.io.FileUtils;
-   import org.apache.commons.io.FilenameUtils;
-   public class UserSettingsMenu extends SettingsMenu{
-
-       public ColorPanel contentPanel;
-       public final static int USERICON_WIDTH = 200;
-       public final static int USERICON_HEIGHT = 200;
-       public final static String defaultProfileImagePath =
    "Icons\\DefaultUserProfileImage.jpg";
-       public static JLabel userIcon = new JLabel();
-       public static JScrollPane listScroller;
-       public JList list;
-       public final static String changeUserProfileImagePath =
    "Icons\\ChangeUserProfileImage.jpg";
-       public final static String changeUserImagePath =
    "Icons\\CreateUserIcon.png";
-       public BackGroundPanel backGroundPanel;
-       public final static MyListCellRenderer myListCellRenderer = new
    MyListCellRenderer();
-       public  LocationDescriptionDialog locationDescriptionDialog;
-       public static boolean owned = false;
-       public JLabel name;
-       static{
-           Dimension backgroundDimension = new
    Dimension((MainFrame.screenSize.width * 5) / 6 - 20,
    (((MainFrame.screenSize.height * 5) / 6) * 2) / 3 - 120 );
-           userIcon.setSize(USERICON_WIDTH,  USERICON_HEIGHT);
-
```

```
-           listScroller = new JScrollPane();
-           listScroller.setHorizontalScrollBarPolicy(ScrollPaneConstants.H
    ORIZONTAL_SCROLLBAR_AS_NEEDED);
-           listScroller.setVerticalScrollBarPolicy(ScrollPaneConstants.VER
    TICAL_SCROLLBAR_ALWAYS);
-           listScroller.getVerticalScrollBar().setUI(new
    MyScrollbarUI(App.gradientColor, Color.white));
-           listScroller.getVerticalScrollBar().setUnitIncrement(20);
-           listScroller.getHorizontalScrollBar().setUI(new
    MyScrollbarUI(App.gradientColor, Color.white));
-           listScroller.getHorizontalScrollBar().setUnitIncrement(20);
-
-           listScroller.setPreferredSize(backgroundDimension);
-           listScroller.setSize(backgroundDimension);
-       }
-
-       public UserSettingsMenu(){
-           locationDescriptionDialog = new LocationDescriptionDialog(this);
-           contentPanel = new ColorPanel();
-           contentPanel.setPreferredSize(this.getSize());
-           contentPanel.setSize(this.getSize());
-           userIcon.addMouseListener(new MouseAdapter(){
-               @Override
-               public void mouseEntered(MouseEvent event) {
-                   if(UserSettingsMenu.owned == true){
-
-                       ImageIcon icon = new ImageIcon(new
    ImageIcon(changeUserImagePath).getImage().getScaledInstance(USERICON_WI
    DTH, USERICON_HEIGHT, Image.SCALE_SMOOTH));
-                       userIcon.setIcon(icon);
-                   }
-               }
-               @Override
-               public void mouseExited(MouseEvent event){
-                   if(UserSettingsMenu.owned == true){
-                       UserSettingsMenu.this.updateUserProfileImage(Connec
    tor.USERNAME);
-                   }
-               }
-               @Override
-               public void mouseClicked(MouseEvent e) {
-                   if(UserSettingsMenu.owned == true){
-                       File photoDir = new File("UserPhotos\\" +
    Connector.USERNAME);
-                       if (!photoDir.exists()){
```

```
-                           photoDir.mkdirs();
-                       }
-
-                       JFileChooser chooser = new JFileChooser();
-                       FileNameExtensionFilter filter = new
    FileNameExtensionFilter(
-                           "JPEG file", "jpg", "jpeg", "png", "PNG file");
-                       chooser.setFileFilter(filter);
-                       int returnVal =
    chooser.showOpenDialog(UsserButton.userSettingsMenu);
-                       if(returnVal == JFileChooser.APPROVE_OPTION) {
-
-                               String filename =
    chooser.getSelectedFile().getAbsolutePath();
-                               File source = new File(filename);
-                               String extension = "." +
    FilenameUtils.getExtension(filename);
-                               File dest = new File("UserPhotos\\" +
    Connector.USERNAME + "\\ProfilePicture" + extension);
-
-                               File curpng = new File("UserPhotos\\" +
    Connector.USERNAME + "\\ProfilePicture.png");
-                               if(curpng.exists()){
-                                   curpng.delete();
-                               }
-                               File curjpg = new File("UserPhotos\\" +
    Connector.USERNAME + "\\ProfilePicture.jpg");
-                               if(curjpg.exists()){
-                                   curjpg.delete();
-                               }
-                               try {
-                                   FileUtils.copyFile(source, dest);
-                               } catch (IOException e1) {
-                                   e1.printStackTrace();
-                               }
-                       }
-
-
-                   }
-               }
-
-           });
-
-           addUserProfileImage();
-           backGroundPanel.addMouseListener(new MouseAdapter(){
```

```
-               @Override
-               public void mouseEntered(MouseEvent event) {
-                   if(UserSettingsMenu.owned == true){
-
-                       Image profileImage =
    Toolkit.getDefaultToolkit().getImage(changeUserProfileImagePath);
-                       Dimension backgroundDimension = new Dimension((int)
    UserSettingsMenu.this.getSize().getWidth(),(int)
    UserSettingsMenu.this.getSize().getHeight() / 3);
-                       try {
-                           BufferedImage bufferedImage = ImageIO.read(new
    File(changeUserProfileImagePath));
-                           profileImage  =
    bufferedImage.getScaledInstance((int)backgroundDimension.getWidth(),(in
    t) backgroundDimension.getHeight(), Image.SCALE_SMOOTH);
-                       } catch (IOException e) {
-                           // TODO Auto-generated catch block
-                           e.printStackTrace();
-                       }
-                       UserSettingsMenu.this.backGroundPanel.setImg(profil
    eImage);
-                       UserSettingsMenu.this.contentPanel.revalidate();
-                       UserSettingsMenu.this.contentPanel.repaint();
-
-                       UserSettingsMenu.this.update();
-                   }
-
-               }
-               @Override
-               public void mouseExited(MouseEvent event){
-                   if(UserSettingsMenu.owned == true){
-                       UserSettingsMenu.this.updateUserProfileImage(Connec
    tor.USERNAME);
-                   }
-               }
-
-               @Override
-               public void mouseClicked(MouseEvent e) {
-                   if(UserSettingsMenu.owned == true){
-                       File photoDir = new File("UserPhotos\\" +
    Connector.USERNAME);
-                       if (!photoDir.exists()){
-                           photoDir.mkdirs();
-                       }
-
```

```
-                       JFileChooser chooser = new JFileChooser();
-                       FileNameExtensionFilter filter = new
    FileNameExtensionFilter(
-                           "JPEG file", "jpg", "jpeg", "png", "PNG file");
-                       chooser.setFileFilter(filter);
-                       int returnVal =
    chooser.showOpenDialog(UsserButton.userSettingsMenu);
-                       if(returnVal == JFileChooser.APPROVE_OPTION) {
-
-                               String filename =
    chooser.getSelectedFile().getAbsolutePath();
-                               File source = new File(filename);
-                               String extension = "." +
    FilenameUtils.getExtension(filename);
-                               File dest = new File("UserPhotos\\" +
    Connector.USERNAME + "\\Background" + extension);
-
-                               File curpng = new File("UserPhotos\\" +
    Connector.USERNAME + "\\Background.png");
-                               if(curpng.exists()){
-                                   curpng.delete();
-                               }
-                               File curjpg = new File("UserPhotos\\" +
    Connector.USERNAME + "\\Background.jpg");
-                               if(curjpg.exists()){
-                                   curjpg.delete();
-                               }
-                               try {
-                                   FileUtils.copyFile(source, dest);
-                               } catch (IOException e1) {
-                                   e1.printStackTrace();
-                               }
-                       }
-
-
-                   }
-               }
-
-           });
-           name = new JLabel("");
-           name.setForeground(Color.yellow);
-           name.setFont(new Font(Font.DIALOG, Font.ITALIC, 40));
-           contentPanel.add(name);
-           contentPanel.add(listScroller);
-           contentPanel.revalidate();
```

```
-           contentPanel.repaint();
-           this.add(contentPanel, BorderLayout.CENTER);
-
-           this.update();
-       }
-
-       public void addUserProfileImage(){
-           Image profileImage =
    Toolkit.getDefaultToolkit().getImage(defaultProfileImagePath);
-           Dimension backgroundDimension = new Dimension((int)
    this.getSize().getWidth(),(int) this.getSize().getHeight() / 3);
-           try {
-               BufferedImage bufferedImage = ImageIO.read(new
    File(defaultProfileImagePath));
-               profileImage  =
    bufferedImage.getScaledInstance((int)backgroundDimension.getWidth(),(in
    t) backgroundDimension.getHeight(), Image.SCALE_SMOOTH);
-           } catch (IOException e) {
-               // TODO Auto-generated catch block
-               e.printStackTrace();
-           }
-           backGroundPanel = new BackGroundPanel(profileImage);
-
-           backGroundPanel.setPreferredSize(backgroundDimension);
-           backGroundPanel.setSize(backgroundDimension);
-
-           String iconPath = UsserButton.iconPath;
-           ImageIcon icon = new ImageIcon(new
    ImageIcon(iconPath).getImage().getScaledInstance(USERICON_WIDTH,
    USERICON_HEIGHT, Image.SCALE_SMOOTH));
-           userIcon.setIcon(icon);
-
-           backGroundPanel.add(userIcon);
-
-           contentPanel.add(backGroundPanel);
-       }
-

-       public void updateVisited() throws SQLException{
-           owned = true;
-           name.setText(Connector.USERNAME);
-           this.updateUserProfileImage(Connector.USERNAME);
-
-           Vector <Location> visitedLocations =
    Connector.getVisitedLocations();
```

```
-
-
-
-           JList <Location> locationList = new
    JList(visitedLocations.toArray());
-           locationList.setOpaque(false);
-           locationList.setCellRenderer(myListCellRenderer);
-           locationList.setForeground(Color.ORANGE);
-           locationList.setBackground(AppSettingsMenu.PURPLE_COLOR);
-           locationList.setSelectionForeground(AppSettingsMenu.PURPLE_COLO
    R);
-           locationList.setSelectionBackground(Color.ORANGE);
-           locationList.addListSelectionListener(new
    ListSelectionListener() {
-
-               @Override
-               public void valueChanged(ListSelectionEvent arg0) {
-                   if (!arg0.getValueIsAdjusting()) {
-                       UserSettingsMenu.this.locationDescriptionDialog.upd
    ateLocation(locationList.getSelectedValue());
-                       UserSettingsMenu.this.locationDescriptionDialog.des
    criptionTextArea.setEditable(true);
-                       UserSettingsMenu.this.locationDescriptionDialog.upd
    ateLocation.setVisible(true);
-                       UserSettingsMenu.this.locationDescriptionDialog.del
    eteLocation.setVisible(true);
-                       UserSettingsMenu.this.locationDescriptionDialog.lik
    eButton.setEnabled(false);
-                       UserSettingsMenu.this.locationDescriptionDialog.dis
    likeButton.setEnabled(false);
-                       UserSettingsMenu.this.locationDescriptionDialog.add
    PhotoButton.setVisible(true);
-                       UserSettingsMenu.this.locationDescriptionDialog.set
    Visible(true);
-
-
-
-                   }
-               }
-           });
-           list = locationList;
-
-           ViewPort viewPort = new ViewPort(App.gradientColor);
-           viewPort.setView(list);
-           listScroller.setViewportView(viewPort);
```

```
-           listScroller.revalidate();
-           listScroller.repaint();
-
-           contentPanel.revalidate();
-           contentPanel.repaint();
-           this.update();
-       }
-
-       public void showOtherUser(String username) throws SQLException{
-
-           name.setText(username);
-           LocationDescriptionDialog.username = username;
-           owned = false;
-           this.updateUserProfileImage(username);
-
-
-           //upload other user profile picture and so on
-           Vector <Location> visitedLocations =
    Connector.getOtherVisitedLocations(username);
-
-
-
-           JList <Location> locationList = new
    JList(visitedLocations.toArray());
-           locationList.setOpaque(false);
-           locationList.setCellRenderer(myListCellRenderer);
-           locationList.setForeground(Color.ORANGE);
-           locationList.setBackground(AppSettingsMenu.PURPLE_COLOR);
-           locationList.setSelectionForeground(AppSettingsMenu.PURPLE_COLO
    R);
-           locationList.setSelectionBackground(Color.ORANGE);
-           locationList.addListSelectionListener(new
    ListSelectionListener() {
-
-               @Override
-               public void valueChanged(ListSelectionEvent arg0) {
-                   if (!arg0.getValueIsAdjusting()) {
-                       UserSettingsMenu.this.locationDescriptionDialog.upd
    ateLocation(locationList.getSelectedValue());
-                       UserSettingsMenu.this.locationDescriptionDialog.upd
    atelikedislikeicons();
-                       UserSettingsMenu.this.locationDescriptionDialog.des
    criptionTextArea.setEditable(false);
-                       UserSettingsMenu.this.locationDescriptionDialog.upd
    ateLocation.setVisible(false);
```

```
-                     UserSettingsMenu.this.locationDescriptionDialog.del
   eteLocation.setVisible(false);
-                     UserSettingsMenu.this.locationDescriptionDialog.lik
   eButton.setEnabled(true);
-                     UserSettingsMenu.this.locationDescriptionDialog.dis
   likeButton.setEnabled(true);
-                     UserSettingsMenu.this.locationDescriptionDialog.add
   PhotoButton.setVisible(false);
-                     UserSettingsMenu.this.locationDescriptionDialog.set
   Visible(true);
-                 }
-             }
-         });
-         list = locationList;
-
-         ViewPort viewPort = new ViewPort(App.gradientColor);
-         viewPort.setView(list);
-         listScroller.setViewportView(viewPort);
-         listScroller.revalidate();
-         listScroller.repaint();
-         contentPanel.revalidate();
-         contentPanel.repaint();
-         this.update();
-         this.setVisible(true);
-     }
-
-     public void updateUserProfileImage(String username){
-
-             if((new File("UserPhotos\\" + username +
   "\\Background.jpg")).exists())
-             {
-                 String profileImagePath = "UserPhotos\\" + username +
   "\\Background.jpg";
-                 Image profileImage =
   Toolkit.getDefaultToolkit().getImage(profileImagePath);
-                 Dimension backgroundDimension = new Dimension((int)
   this.getSize().getWidth(),(int) this.getSize().getHeight() / 3);
-                 try {
-                     BufferedImage bufferedImage = ImageIO.read(new
   File(profileImagePath));
-                     profileImage  =
   bufferedImage.getScaledInstance((int)backgroundDimension.getWidth(),(in
  t) backgroundDimension.getHeight(), Image.SCALE_SMOOTH);
-                 } catch (IOException e) {
-                     // TODO Auto-generated catch block
```

```
-                     e.printStackTrace();
-                 }
-                 backGroundPanel.setImg(profileImage);
-             }
-             else
-             if((new File("UserPhotos\\" + username +
"\\Background.png")).exists())
-             {
-                 String profileImagePath = "UserPhotos\\" + username +
"\\Background.png";
-                 Image profileImage =
Toolkit.getDefaultToolkit().getImage(profileImagePath);
-                 Dimension backgroundDimension = new Dimension((int)
this.getSize().getWidth(),(int) this.getSize().getHeight() / 3);
-                 try {
-                     BufferedImage bufferedImage = ImageIO.read(new
File(profileImagePath));
-                     profileImage  =
bufferedImage.getScaledInstance((int)backgroundDimension.getWidth(),(in
t) backgroundDimension.getHeight(), Image.SCALE_SMOOTH);
-                 } catch (IOException e) {
-                     // TODO Auto-generated catch block
-                     e.printStackTrace();
-                 }
-                 backGroundPanel.setImg(profileImage);
-             }
-             else
-             if((new File("UserPhotos\\" + username +
"\\Background.jpeg")).exists())
-             {
-                 String profileImagePath = "UserPhotos\\" + username +
"\\Background.jpeg";
-                 Image profileImage =
Toolkit.getDefaultToolkit().getImage(profileImagePath);
-                 Dimension backgroundDimension = new Dimension((int)
this.getSize().getWidth(),(int) this.getSize().getHeight() / 3);
-                 try {
-                     BufferedImage bufferedImage = ImageIO.read(new
File(profileImagePath));
-                     profileImage  =
bufferedImage.getScaledInstance((int)backgroundDimension.getWidth(),(in
t) backgroundDimension.getHeight(), Image.SCALE_SMOOTH);
-                 } catch (IOException e) {
-                     // TODO Auto-generated catch block
-                     e.printStackTrace();
```

```
-                   }
-                   backGroundPanel.setImg(profileImage);
-               }
-               else
-               {
-                   Image profileImage =
    Toolkit.getDefaultToolkit().getImage(defaultProfileImagePath);
-                   Dimension backgroundDimension = new Dimension((int)
    this.getSize().getWidth(),(int) this.getSize().getHeight() / 3);
-                   try {
-                       BufferedImage bufferedImage = ImageIO.read(new
    File(defaultProfileImagePath));
-                       profileImage  =
    bufferedImage.getScaledInstance((int)backgroundDimension.getWidth(),(in
    t) backgroundDimension.getHeight(), Image.SCALE_SMOOTH);
-                   } catch (IOException e2) {
-                       // TODO Auto-generated catch block
-                       e2.printStackTrace();
-                   }
-                   backGroundPanel.setImg(profileImage);
-               }
-               if((new File("UserPhotos\\" + username +
    "\\ProfilePicture.jpg")).exists())
-               {
-                   String iconPath = "UserPhotos\\" + username +
    "\\ProfilePicture.jpg";
-                   ImageIcon icon = new ImageIcon(new
    ImageIcon(iconPath).getImage().getScaledInstance(USERICON_WIDTH,
    USERICON_HEIGHT, Image.SCALE_SMOOTH));
-                   userIcon.setIcon(icon);
-               }
-               else
-               if((new File("UserPhotos\\" + username +
    "\\ProfilePicture.png")).exists())
-               {
-                   String iconPath = "UserPhotos\\" + username +
    "\\ProfilePicture.png";
-                   ImageIcon icon = new ImageIcon(new
    ImageIcon(iconPath).getImage().getScaledInstance(USERICON_WIDTH,
    USERICON_HEIGHT, Image.SCALE_SMOOTH));
-                   userIcon.setIcon(icon);
-               }
-               else
-               if((new File("UserPhotos\\" + username +
    "\\ProfilePicture.jpeg")).exists())
```

```
-               {
-                   String iconPath = "UserPhotos\\" + username +
    "\\ProfilePicture.jpeg";
-                   ImageIcon icon = new ImageIcon(new
    ImageIcon(iconPath).getImage().getScaledInstance(USERICON_WIDTH,
    USERICON_HEIGHT, Image.SCALE_SMOOTH));
-                   userIcon.setIcon(icon);
-               }
-               else
-               {
-                   String iconPath = UsserButton.iconPath;
-                   ImageIcon icon = new ImageIcon(new
    ImageIcon(iconPath).getImage().getScaledInstance(USERICON_WIDTH,
    USERICON_HEIGHT, Image.SCALE_SMOOTH));
-                   userIcon.setIcon(icon);
-               }
-               userIcon.revalidate();
-               userIcon.repaint();
-
-               contentPanel.revalidate();
-               contentPanel.repaint();
-               this.update();
-
-
-       }
-   }
-
```

- UsserButton

```
-   package com.tourer.gui;
-
-   import java.awt.event.ActionListener;
-   import java.sql.SQLException;
-   import java.awt.event.ActionEvent;
-
-   public class UsserButton extends CostumButton{
-       public final static String iconPath = "Icons\\UserIcon.png";
-       public final static String darkIconPath = "Icons\\UserIconDark.png";
-
-       public static UserSettingsMenu userSettingsMenu;
-
```

```
-       public  UsserButton(int w, int h, UserSettingsMenu userSettingsMenu)
    {
-           super(w, h, UsserButton.iconPath);
-           UsserButton.userSettingsMenu = userSettingsMenu;
-           this.addActionListener(new ActionListener(){
-
-               @Override
-               public void actionPerformed(ActionEvent e) {
-                   SettingsButton.appMenuSettings.setVisible(false);
-                   try {
-                       UsserButton.userSettingsMenu.updateVisited();
-                   } catch (SQLException e1) {
-                       // TODO Auto-generated catch block
-                       e1.printStackTrace();
-                   }
-                   UsserButton.userSettingsMenu.setVisible(true);
-
-               }
-
-           });
-       }
-   }
-
```

- ViewPort

```
-   package com.tourer.gui;
-
-
-   import java.awt.Color;
-   import java.awt.GradientPaint;
-   import java.awt.Graphics;
-   import java.awt.Graphics2D;
-
-   import javax.swing.JViewport;
-
-   /**
-       *
-       * @since 2.0
-       * Class that is used to change the bacground of the logArea
-       */
-   public @SuppressWarnings("serial") class ViewPort extends JViewport {
-       private Color c1;
```

```
-       private Color c2;
-
-       /**
-        * Constructor of the class
-        * @param c1 main color
-        * @param c2 secondary color
-        */
-       public ViewPort(GradientColor g) {
-           this.c1 = g.mainColor;
-           this.c2 = g.secondaryColor;
-       }
-
-
-       /**
-        * Creates gradient background using the 2 colors that were given in
    the constructor
-        * @param g
-        */
-       @Override
-       protected void paintComponent(Graphics g) {
-           super.paintComponent(g);
-           GradientPaint gPaint = new GradientPaint(0, 0, c1, getWidth(),
    getHeight(), c2, false);
-           Graphics2D g2 = (Graphics2D) g;
-           g2.setPaint(gPaint);
-           g2.fillRect(0, 0, getWidth(), getHeight());
-       }
-
-   }
```

- WindowSettingButton

```
package com.tourer.gui;

public class WindowSettingButton extends CostumButton{

    public final static String iconPath = "Icons\\WindowSettings.png";
    public final static String darkIconPath = "Icons\\WindowSettings.png";
    public WindowSettingButton(int w, int h) {
        super(w, h, iconPath);
```

```
    }

}
```

3. Pachetul com.tourer.gui.map

- Location

```
-   package com.tourer.gui.map;
-
-   import java.util.Objects;
-   import java.util.Set;
-
-   public class Location {
-       String name;
-       String description;
-       Double longitude;
-       Double latitude;
-       int likes;
-       int dislikes;
-       String photo;
-       Set <String> userlikes;
-       Set <String> userdislikes;
-
-
-       public Location() {
-       }
-
-       public Location(String name, String description, Double longitude,
    Double latitude, int likes, int dislikes, String photo, Set<String>
    userlikes, Set<String> userdislikes) {
-           this.name = name;
-           this.description = description;
-           this.longitude = longitude;
-           this.latitude = latitude;
-           this.likes = likes;
-           this.dislikes = dislikes;
```

```
-           this.photo = photo;
-           this.userlikes = userlikes;
-           this.userdislikes = userdislikes;
-       }
-
-       public String getName() {
-           return this.name;
-       }
-
-       public void setName(String name) {
-           this.name = name;
-       }
-
-       public String getDescription() {
-           return this.description;
-       }
-
-       public void setDescription(String description) {
-           this.description = description;
-       }
-
-       public Double getLongitude() {
-           return this.longitude;
-       }
-
-       public void setLongitude(Double longitude) {
-           this.longitude = longitude;
-       }
-
-       public Double getLatitude() {
-           return this.latitude;
-       }
-
-       public void setLatitude(Double latitude) {
-           this.latitude = latitude;
-       }
-
-       public int getLikes() {
-           return this.likes;
-       }
-
-       public void setLikes(int likes) {
-           this.likes = likes;
-       }
-
```

```
-       public int getDislikes() {
-           return this.dislikes;
-       }
-
-       public void setDislikes(int dislikes) {
-           this.dislikes = dislikes;
-       }
-
-       public String getPhoto() {
-           return this.photo;
-       }
-
-       public void setPhoto(String photo) {
-           this.photo = photo;
-       }
-
-       public Set<String> getUserlikes() {
-           return this.userlikes;
-       }
-
-       public void setUserlikes(Set<String> userlikes) {
-           this.userlikes = userlikes;
-       }
-
-       public Set<String> getUserdislikes() {
-           return this.userdislikes;
-       }
-
-       public void setUserdislikes(Set<String> userdislikes) {
-           this.userdislikes = userdislikes;
-       }
-
-       public Location name(String name) {
-           setName(name);
-           return this;
-       }
-
-       public Location description(String description) {
-           setDescription(description);
-           return this;
-       }
-
-       public Location longitude(Double longitude) {
-           setLongitude(longitude);
-           return this;
```

```
-       }
-
-       public Location latitude(Double latitude) {
-           setLatitude(latitude);
-           return this;
-       }
-
-       public Location likes(int likes) {
-           setLikes(likes);
-           return this;
-       }
-
-       public Location dislikes(int dislikes) {
-           setDislikes(dislikes);
-           return this;
-       }
-
-       public Location photo(String photo) {
-           setPhoto(photo);
-           return this;
-       }
-
-       public Location userlikes(Set<String> userlikes) {
-           setUserlikes(userlikes);
-           return this;
-       }
-
-       public Location userdislikes(Set<String> userdislikes) {
-           setUserdislikes(userdislikes);
-           return this;
-       }
-
-       @Override
-       public boolean equals(Object o) {
-           if (o == this)
-               return true;
-           if (!(o instanceof Location)) {
-               return false;
-           }
-           Location location = (Location) o;
-           return Objects.equals(name, location.name) &&
    Objects.equals(description, location.description) &&
    Objects.equals(longitude, location.longitude) && Objects.equals(latitude,
    location.latitude) && likes == location.likes && dislikes ==
    location.dislikes && Objects.equals(photo, location.photo) &&
```

```
    Objects.equals(userlikes, location.userlikes) &&
    Objects.equals(userdislikes, location.userdislikes);
-       }
-
-       @Override
-       public int hashCode() {
-           return Objects.hash(name, description, longitude, latitude, likes,
    dislikes, photo, userlikes, userdislikes);
-       }
-
-       @Override
-       public String toString() {
-           return getName();
-
-       }
-
-
-
-   }
-
```

- GoogleMaps

```
-   package com.tourer.gui.map;
-
-
-   //key=AIzaSyCqAeOnRkRu6zylBXyMP0ilbvspn_pT5tE
-   public class GoogleMaps{
-
-
-
-   }
```

4. Pachetul com.tourer.jdbc

- Connector

```
-   package com.tourer.jdbc;
-
```

```
-   import com.tourer.gui.map.Location;
-
-   import java.sql.Statement;
-   import java.util.Set;
-   import java.util.TreeSet;
-   import java.util.Vector;
-   import java.util.regex.Matcher;
-   import java.util.regex.Pattern;
-
-   import javax.swing.JOptionPane;
-
-   import com.tourer.App;
-
-   import java.sql.DriverManager;
-   import java.sql.PreparedStatement;
-   import java.sql.ResultSet;
-   import java.sql.SQLException;
-   import java.sql.Connection;
-

-   public class Connector {
-
-       public static Pattern ALREADY_IN_USE_USERNAME =
    Pattern.compile("Duplicate entry '.+' for key 'userprofile.username'");
-       public final static String ERROR_DUBLICATE_USERNAME = "User already
    exist with given username";
-       public static Pattern ALREADY_IN_USE_EMAIL =
    Pattern.compile("Duplicate entry '.+' for key 'userprofile.email'");
-       public final static String ERROR_DUBLICATE_EMAIL = "User already exist
    with given email";
-       public final static String ERROR_CARD_UPDATE = "Couldn't load card
    data";
-       public final static String ERROR_LIKE_UPDATE = "Failed to update number
    of likes";
-       public static Connection connector;
-       public static Statement statement;
-       public static Integer USERID;
-       public static String USERNAME;
-       public static void Connect(String url, String user, String password)
    throws SQLException{
-           try {
-               Class.forName("com.mysql.cj.jdbc.Driver");
-           } catch (ClassNotFoundException e) {
-
-               e.printStackTrace();
```

```
-               System.exit(2);
-           }
-           Connector.connector =  DriverManager.getConnection(url, user,
    password);
-           Connector.statement = connector.createStatement();
-
-       }
-
-       public static void runUpdate(String query) throws SQLException{
-
-           Connector.statement.executeUpdate(query);
-       }
-
-       public static ResultSet runQuery(String query) throws SQLException{
-
-           ResultSet result = Connector.statement.executeQuery(query);
-           return result;
-       }
-
-       public static void closeStament() throws SQLException{
-           Connector.statement.close();
-       }
-
-       public static void closeConnection() throws SQLException{
-           Connector.connector.close();
-       }
-

-       public static boolean checkUserExistance(String username, String
    password) throws SQLException{
-
-
-           String query = "SELECT id from UserProfile WHERE username='" +
    username + "' AND password='" + password + "'";
-
-           ResultSet resultSet = runQuery(query);
-           if (!resultSet.next()){
-               resultSet.close();
-               return false;
-           }
-           USERNAME = username;
-           USERID = resultSet.getInt("id");
-           resultSet.close();
-           return true;
-       }
```

```
-
-       public static boolean createUser(String username, String password,
    String email){
-           String query = "INSERT INTO UserProfile (creditcardno, username,
    password, email) VALUES ('-','" + username + "','" + password + "','" +
    email  + "');";
-           try {
-               runUpdate(query);
-           } catch (SQLException e) {
-
-               String SQLConstraintError = e.getMessage();
-               System.out.println(SQLConstraintError);
-               Matcher userMatcher =
    ALREADY_IN_USE_USERNAME.matcher(SQLConstraintError);
-               if(userMatcher.find())
-               {
-                   JOptionPane.showMessageDialog(App.accountCreationFrame,
    ERROR_DUBLICATE_USERNAME, "ERROR", JOptionPane.ERROR_MESSAGE);
-                   return false;
-               }
-               Matcher mailMatcher =
    ALREADY_IN_USE_EMAIL.matcher(SQLConstraintError);
-               if(mailMatcher.find())
-               {
-                   JOptionPane.showMessageDialog(App.accountCreationFrame,
    ERROR_DUBLICATE_EMAIL, "ERROR", JOptionPane.ERROR_MESSAGE);
-                   return false;
-               }
-
-               JOptionPane.showMessageDialog(App.accountCreationFrame,
    "Failed to create user", "ERROR", JOptionPane.ERROR_MESSAGE);
-               return false;
-
-           }
-           try{
-               MailSender.sendEmail(email);
-           }catch(Exception e){
-               e.printStackTrace();
-               System.out.println("Failed to send email");
-           }
-           return true;
-       }
-
-       public static Vector <String> getUserList(String username) throws
    SQLException{
```

```
-           Vector <String> rez = new Vector <String>();
-
-           String query = "SELECT username FROM UserProfile WHERE username
    REGEXP '" + username + "' AND id!=" + USERID + " ORDER BY username LIMIT
    10";
-           ResultSet resultSet = runQuery(query);
-           while(resultSet.next()){
-               String name = resultSet.getString("username");
-               rez.add(name);
-           }
-           resultSet.close();
-           return rez;
-       }
-
-       public static void deleteLocation(String name) throws SQLException{
-           String query = "DELETE from location where id =" + Connector.USERID
    + " and name = '" + name + "';";
-           PreparedStatement statement =
    Connector.connector.prepareStatement(query);
-           statement.executeUpdate();
-       }
-
-       public static Vector <String> getLocationList(String name) throws
    SQLException{
-           Vector <String> rez = new Vector <String>();
-
-           String query = "SELECT name FROM  Location WHERE name REGEXP '" +
    name + "' ORDER BY name LIMIT 10";
-           ResultSet resultSet = runQuery(query);
-           while(resultSet.next()){
-               String lname = resultSet.getString("name");
-               rez.add(lname);
-           }
-           resultSet.close();
-           return rez;
-       }
-
-       public static boolean createLocation(double latitude, double
    longitude, String name, String description) throws SQLException{
-           String query = "INSERT INTO Location(id, latitude, longitude,
    description, name) VALUES (" + Connector.USERID + "," + latitude + "," +
    longitude + ",'" + description + "','" + name  + "');";
-
-           try {
-               runUpdate(query);
```

```
-          } catch (SQLException e) {
-
-              boolean updateok = modifyLocation(latitude, longitude, name,
    description);
-
-              if(updateok == false){
-                  JOptionPane.showMessageDialog(App.mainFrame, "Failde to
    add location to database", "ERROR", JOptionPane.ERROR_MESSAGE);
-                  return false;
-              }
-              else
-              {
-                  JOptionPane.showMessageDialog(App.mainFrame, "Current
    location already existant, updated it", "UPDATE",
    JOptionPane.INFORMATION_MESSAGE);
-
-              }
-
-
-          }
-
-          return true;
-      }
-
-      public static boolean modifyLocation(double latitude, double
    longitude, String name, String description){
-          String query = "UPDATE Location SET name='" + name + "',
    description='" + description  +"' WHERE id="+ Connector.USERID +" AND
    latitude="+ latitude +" AND longitude="+ longitude +";";
-
-          try {
-              runUpdate(query);
-          } catch (SQLException e) {
-              e.printStackTrace();
-              return false;
-
-          }
-
-          return true;
-      }
-
-      public static Vector <Location> getVisitedLocations() throws
    SQLException{
-          Vector <Location> locationList = new Vector<Location>();
-
```

```
-          String query = "SELECT latitude, longitude, description, likes,
   dislikes, photo, name FROM  Location WHERE id=" + USERID + ";";
-
-          ResultSet resultSet = runQuery(query);
-          while(resultSet.next()){
-
-              String name = resultSet.getString("name");
-              String description = resultSet.getString("description");
-              Double latitude = resultSet.getDouble("latitude");
-              Double longitude = resultSet.getDouble("longitude");
-              int likes = resultSet.getInt("likes");
-              int dislikes = resultSet.getInt("dislikes");
-              Set <String> userlikes = new TreeSet<>();
-              Set <String> userdislikes = new TreeSet<>();
-              String locationImage = resultSet.getString("photo");
-              locationList.add(new Location(name, description, longitude,
   latitude, likes, dislikes, locationImage, userlikes, userdislikes));
-          }
-          resultSet.close();
-          return locationList;
-
-      }
-
-      public static Vector <Location> getOtherVisitedLocations(String
   username) throws SQLException{
-          Vector <Location> locationList = new Vector<Location>();
-
-          String query = "SELECT latitude, longitude, description, likes,
   dislikes, photo, name FROM  Location WHERE id=( SELECT id FROM UserProfile
   WHERE username='" + username  +"');";
-          Statement statement2 = connector.createStatement();
-          ResultSet resultSet = runQuery(query);
-          while(resultSet.next()){
-
-              String name = resultSet.getString("name");
-              String description = resultSet.getString("description");
-              Double latitude = resultSet.getDouble("latitude");
-              Double longitude = resultSet.getDouble("longitude");
-              int likes = resultSet.getInt("likes");
-              int dislikes = resultSet.getInt("dislikes");
-              Set <String> userlikes = new TreeSet<>();
-              Set <String> userdislikes = new TreeSet<>();
-
-              String locationImage = resultSet.getString("photo");
```

```
-               String query2 = "SELECT nameother, type from likestate
    WHERE  id=( SELECT id FROM UserProfile WHERE username='" + username  +"')
    AND name = '" + name + "';";
-
-               ResultSet resultSet2 = statement2.executeQuery(query2);
-
-               while(resultSet2.next()){
-                   boolean type = resultSet2.getBoolean("type");
-                   String nameother = resultSet2.getString("nameother");
-
-                   if(type == true)
-                       userlikes.add(nameother);
-                   else
-                       userdislikes.add(nameother);
-
-               }
-               resultSet2.close();
-
-               locationList.add(new Location(name, description, longitude,
    latitude, likes, dislikes, locationImage, userlikes, userdislikes));
-           }
-           statement2.close();
-           resultSet.close();
-           return locationList;
-
-       }
-       public static void insertCardData(String cardType, String cardNumber,
    String expriationDate, String securityCode){
-           String query = "INSERT INTO CreditCard(id, type, number,
    securityCode, expirationDate) VALUES(" + USERID + ",'" + cardType + "',"
    + cardNumber + "," + securityCode + ",STR_TO_DATE('01/" + expriationDate
    + "', '%d/%m/%Y')" + ");";
-
-           try {
-               runUpdate(query);
-           } catch (SQLException e) {
-               updateCardData(cardType, cardNumber, expriationDate,
    securityCode);
-
-           }
-
-       }
-
-       public static void updateCardData(String cardType, String cardNumber,
    String expriationDate, String securityCode){
```

```
-           String query = "UPDATE CreditCard SET type='" + cardType +
    "',number=" + cardNumber + ",expirationDate=STR_TO_DATE('01/" +
    expriationDate + "', '%d/%m/%Y')" + ",securityCode=" + securityCode + "
    WHERE id=" + USERID + ";";
-
-           try {
-               runUpdate(query);
-           } catch (SQLException e) {
-
-               JOptionPane.showMessageDialog(App.accountCreationFrame,
    ERROR_CARD_UPDATE, "ERROR", JOptionPane.ERROR_MESSAGE);
-           }
-       }
-
-
-       public static void  like(String name, String username, int like){
-           String query = "UPDATE Location SET likes=" + like +  " WHERE id=(
    SELECT id FROM UserProfile WHERE username='" + username +  "') AND name =
    '" + name +  "';";
-           try {
-               runUpdate(query);
-           } catch (SQLException e) {
-               e.printStackTrace();
-               JOptionPane.showMessageDialog(App.accountCreationFrame,
    ERROR_LIKE_UPDATE, "ERROR", JOptionPane.ERROR_MESSAGE);
-           }
-       }
-
-       public static void dislike(String name, String username, int dislike){
-
-           String query = "UPDATE Location SET dislikes=" + dislike +  " WHERE
    id=( SELECT id FROM UserProfile WHERE username='" + username + "') AND name
    = '" + name +  "';";
-           try {
-               runUpdate(query);
-           } catch (SQLException e) {
-               e.printStackTrace();
-               JOptionPane.showMessageDialog(App.accountCreationFrame,
    ERROR_LIKE_UPDATE, "ERROR", JOptionPane.ERROR_MESSAGE);
-           }
-       }
-
-       public static Location getFirstLocationByName(String namelocation)
    throws SQLException{
-           Location location = new Location();
```

```
-          String query = "SELECT latitude, longitude, description, likes,
   dislikes, name FROM  Location WHERE name='" + namelocation  +"' LIMIT 1;";
-          ResultSet resultSet = runQuery(query);
-          if(resultSet.next()){
-
-              String name = resultSet.getString("name");
-              String description = resultSet.getString("description");
-              Double latitude = resultSet.getDouble("latitude");
-              Double longitude = resultSet.getDouble("longitude");
-              int likes = resultSet.getInt("likes");
-              int dislikes = resultSet.getInt("dislikes");
-              Set <String> userlikes = new TreeSet<>();
-              Set <String> userdislikes = new TreeSet<>();
-
-              location = new Location(name, description, longitude,
   latitude, likes, dislikes, "", userlikes, userdislikes);
-
-          }
-          return location;
-
-      }
-
-      public static void modifylikeState(String name, String username,
   boolean type) throws SQLException{
-
-          String query = "SELECT * FROM likestate WHERE nameother = '" +
   USERNAME + "' AND type = " + type + " AND name = '" + name + "' AND id =
   ( SELECT id FROM UserProfile WHERE username='" + username +  "');";
-
-          String query2 = "";
-          ResultSet resultSet = runQuery(query);
-
-          if(resultSet.next()){
-              query2 = "DELETE FROM likestate WHERE nameother = '" + USERNAME
   + "' AND type = " + type + " AND name = '" + name + "' AND id = ( SELECT
   id FROM UserProfile WHERE username='" + username +  "');";
-
-          }
-          else
-          {
-              query2 = "INSERT INTO likestate(id, name, nameother, type)
   VALUES (( SELECT id FROM UserProfile WHERE username='" + username +  "'),'"
   + name + "', '" + USERNAME + "', " + type + "  );";
-          }
-          resultSet.close();
```

```
-           runUpdate(query2);
-       }
-
-       public static void updatePhoto(String name, String newpath) throws
    SQLException{
-           newpath = newpath.replace("\\", "\\\\");
-           String query = "UPDATE Location SET photo='" + newpath + "' WHERE
    name='" + name + "' AND id=" + USERID + ";";
-
-           System.out.println(query);
-           runUpdate(query);
-
-
-       }
-   }
-
```

- JdbcRunner

```
-   package com.tourer.jdbc;
-
-
-   import java.sql.SQLException;
-
-
-   public class JdbcRunner {
-
-       public static void buildJdbc() throws SQLException{
-           String url = "jdbc:mysql://localhost:3306/tourer";
-           String user = "root";
-           String password = "Calebbb1234567890";
-
-           Connector.Connect(url, user, password);
-
-       }
-   }
-
```

- MailSender

```
-   package com.tourer.jdbc;
-
-   import javax.mail.PasswordAuthentication;
-
-   import java.io.File;
-   import java.io.FileNotFoundException;
-   import java.util.Properties;
-   import java.util.Scanner;
-
-   import javax.mail.Authenticator;
-   import javax.mail.Message;
-   import javax.mail.MessagingException;
-   import javax.mail.Session;
-   import javax.mail.Transport;
-   import javax.mail.internet.InternetAddress;
-   import javax.mail.internet.MimeMessage;
-
-   public class MailSender {
-       protected static String MAIL = "mihai.gherghinescu18@gmail.com";
-       protected static String PASSWORD = "Calebbb1234567890";
-       protected static Properties properties = new Properties();
-       protected static Session session;
-       static{
-
-           properties.put("mail.smtp.auth", "true");
-           properties.put("mail.smtp.starttls.enable", "true");
-           properties.put("mail.smtp.host", "smtp.gmail.com");
-           properties.put("mail.smtp.port", "587");
-
-           session = Session.getInstance(properties, new Authenticator() {
-               @Override
-               protected PasswordAuthentication
    getPasswordAuthentication(){
-                   return new PasswordAuthentication(MAIL, PASSWORD);
-               }
-        });
-       }
-       public static void sendEmail(String recipient) throws
    MessagingException{
-
-           Message message = createMessage(recipient);
-           Transport.send(message);
-       }
-       private static Message createMessage(String recipient) {
-
```

```
-           try {
-               String template = "Hello, thank you for creating a new account
    for Timisoara Tourer\nWe hope that the app will give you a better look of
    Timisoara and you'll enjoy using it\n";
-               try {
-                   StringBuilder stringBuilder = new StringBuilder("");
-                   File myObj = new File("Email\\Template.mail.html");
-                   Scanner myReader = new Scanner(myObj);
-                   while (myReader.hasNextLine()) {
-                       String data = myReader.nextLine();
-                       stringBuilder.append(data);
-                   }
-                   myReader.close();
-                   template = stringBuilder.toString();
-                   } catch (FileNotFoundException e) {
-
-                       e.printStackTrace();
-                   }
-               Message message = new MimeMessage(session);
-               message.setFrom(new InternetAddress(MAIL));
-               message.setRecipient(Message.RecipientType.TO, new
    InternetAddress(recipient));
-               message.setSubject("New account for Timisoara Tourer");
-               message.setContent(template, "text/html; charset=utf-8");
-
-               return message;
-           } catch (MessagingException e) {
-               // TODO Auto-generated catch block
-               e.printStackTrace();
-           }
-
-
-           return null;
-       }
-   }
-
```

5. Dependencies

```
6. <?xml version="1.0" encoding="UTF-8"?>
7.
8. <project xmlns="http://maven.apache.org/POM/4.0.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
9.    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
   http://maven.apache.org/xsd/maven-4.0.0.xsd">
10.   <modelVersion>4.0.0</modelVersion>
11.
12.   <groupId>com.example</groupId>
13.   <artifactId>demo</artifactId>
14.   <version>1.0-SNAPSHOT</version>
15.
16.   <name>demo</name>
17.   <!-- FIXME change it to the project's website -->
18.   <url>http://www.example.com</url>
19.
20.   <properties>
21.     <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
22.     <maven.compiler.source>1.8</maven.compiler.source>
23.     <maven.compiler.target>1.8</maven.compiler.target>
24.   </properties>
25.
26.   <dependencies>
27.
28.      <!-- https://mvnrepository.com/artifact/javax.mail/mail -->
29.     <dependency>
30.         <groupId>javax.mail</groupId>
31.         <artifactId>mail</artifactId>
32.         <version>1.4</version>
33.     </dependency>
34.
35.     <dependency>
36.       <groupId>org.openjfx</groupId>
37.       <artifactId>javafx-archetype-fxml</artifactId>
38.       <version>0.0.3</version>
39.     </dependency>
40.         <!-- https://mvnrepository.com/artifact/org.openjfx/javafx-web -->
41.     <dependency>
42.         <groupId>org.openjfx</groupId>
43.         <artifactId>javafx-web</artifactId>
44.         <version>18-ea+6</version>
45.     </dependency>
46.     <!-- https://mvnrepository.com/artifact/commons-io/commons-io -->
47.       <dependency>
48.           <groupId>commons-io</groupId>
49.           <artifactId>commons-io</artifactId>
50.           <version>2.6</version>
51.       </dependency>
52.
```

```
53.     <dependency>
54.       <groupId>org.openjfx</groupId>
55.       <artifactId>javafx-graphics</artifactId>
56.       <version>11</version>
57.     </dependency>
58. <!-- Thanks for using https://jar-download.com -->
59.         <!-- https://mvnrepository.com/artifact/org.openjfx/javafx-swing
    -->
60.     <dependency>
61.         <groupId>org.openjfx</groupId>
62.         <artifactId>javafx-swing</artifactId>
63.         <version>11-ea+24</version>
64.     </dependency>
65.     <dependency>
66.       <groupId>org.openjfx</groupId>
67.       <artifactId>javafx-fxml</artifactId>
68.       <version>13</version>
69.     </dependency>
70.     <dependency>
71.       <groupId>org.openjfx</groupId>
72.       <artifactId>javafx-controls</artifactId>
73.       <version>13</version>
74.     </dependency>
75.
76.     <dependency>
77.       <groupId>com.google.maps</groupId>
78.       <artifactId>google-maps-services</artifactId>
79.       <version>1.0.1</version>
80.     </dependency>
81.
82.     <dependency>
83.       <groupId>org.slf4j</groupId>
84.       <artifactId>slf4j-simple</artifactId>
85.       <version>1.7.25</version>
86.     </dependency>
87.     <dependency>
88.       <groupId>com.google.code.gson</groupId>
89.       <artifactId>gson</artifactId>
90.       <version>2.8.9</version>
91.     </dependency>
92.     <!-- https://mvnrepository.com/artifact/com.squareup.okhttp3/okhttp -->
93.     <dependency>
94.         <groupId>com.squareup.okhttp3</groupId>
95.         <artifactId>okhttp</artifactId>
96.
```

```
97.     </dependency>
98.
99.
100.          <dependency>
101.            <groupId>junit</groupId>
102.            <artifactId>junit</artifactId>
103.            <version>4.11</version>
104.            <scope>test</scope>
105.          </dependency>
106.          <!--
    https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
107.          <dependency>
108.              <groupId>mysql</groupId>
109.              <artifactId>mysql-connector-java</artifactId>
110.              <version>8.0.27</version>
111.          </dependency>
112.
113.        </dependencies>
114.
115.        <build>
116.          <pluginManagement><!-- lock down plugins versions to avoid using
    Maven defaults (may be moved to parent pom) -->
117.            <plugins>
118.              <!-- clean lifecycle, see
    https://maven.apache.org/ref/current/maven-core/lifecycles.html#clean_Life
    cycle -->
119.              <plugin>
120.                <artifactId>maven-clean-plugin</artifactId>
121.                <version>3.1.0</version>
122.              </plugin>
123.              <!-- default lifecycle, jar packaging: see
    https://maven.apache.org/ref/current/maven-core/default-bindings.html#Plug
    in_bindings_for_jar_packaging -->
124.              <plugin>
125.                <artifactId>maven-resources-plugin</artifactId>
126.                <version>3.0.2</version>
127.              </plugin>
128.              <plugin>
129.                <artifactId>maven-compiler-plugin</artifactId>
130.                <version>3.8.0</version>
131.              </plugin>
132.              <plugin>
133.                <artifactId>maven-surefire-plugin</artifactId>
134.                <version>2.22.1</version>
135.              </plugin>
```

```
            <plugin>
              <artifactId>maven-jar-plugin</artifactId>
              <version>3.0.2</version>
            </plugin>
            <plugin>
              <artifactId>maven-install-plugin</artifactId>
              <version>2.5.2</version>
            </plugin>
            <plugin>
              <artifactId>maven-deploy-plugin</artifactId>
              <version>2.8.2</version>
            </plugin>
            <!-- site lifecycle, see
   https://maven.apache.org/ref/current/maven-core/lifecycles.html#site_Lifec
   ycle -->
            <plugin>
              <artifactId>maven-site-plugin</artifactId>
              <version>3.7.1</version>
            </plugin>
            <plugin>
              <artifactId>maven-project-info-reports-plugin</artifactId>
              <version>3.0.0</version>
            </plugin>
            <plugin>
              <groupId>com.zenjava</groupId>
              <artifactId>javafx-maven-plugin</artifactId>
              <version>8.7.0</version>
              <configuration>
                  <options>
                    <option>--add-opens</option>
                    <option>javafx.graphics/javafx.css=ALL-UNNAMED</opti
   on>
                    <option>--add-opens</option>
                    <option>javafx.base/com.sun.javafx.runtime=ALL-UNNAM
   ED</option>
                    <option>--add-opens</option>
                    <option>javafx.controls/com.sun.javafx.scene.control
   .behavior=ALL-UNNAMED</option>
                    <option>--add-opens</option>
                    <option>javafx.controls/com.sun.javafx.scene.control
   =ALL-UNNAMED</option>
                    <option>--add-opens</option>
                    <option>javafx.base/com.sun.javafx.binding=ALL-UNNAM
   ED</option>
                    <option>--add-opens</option>
```

```
174.                     <option>javafx.base/com.sun.javafx.event=ALL-UNNAMED
   </option>
175.                     <option>--add-opens</option>
176.                     <option>javafx.graphics/com.sun.javafx.stage=ALL-UNN
   AMED</option>
177.                     <option>--add-exports</option>
178.                     <option>javafx.controls/com.sun.javafx.scene.control
   .behavior=com.jfoenix</option>
179.                     <option>--add-exports</option>
180.                     <option>javafx.controls/com.sun.javafx.scene.control
   =com.jfoenix</option>
181.                     <option>--add-exports</option>
182.                     <option>javafx.base/com.sun.javafx.binding=com.jfoen
   ix</option>
183.                     <option>--add-exports</option>
184.                     <option>javafx.graphics/com.sun.javafx.stage=com.jfo
   enix</option>
185.                     <option>--add-exports</option>
186.                     <option>javafx.base/com.sun.javafx.event=com.jfoenix
   </option>
187.                   </options>
188.                   <mainClass>demo.com.tourer.App</mainClass>
189.                </configuration>
190.              </plugin>
191.            </plugins>
192.          </pluginManagement>
193.        </build>
194.
195.        <dependencyManagement>
196.          <dependencies>
197.
198.            <dependency>
199.              <groupId>org.jetbrains.kotlin</groupId>
200.              <artifactId>kotlin-stdlib</artifactId>
201.              <version>1.4.0</version>
202.            </dependency>
203.
204.            <dependency>
205.              <groupId>org.jetbrains.kotlin</groupId>
206.              <artifactId>kotlin-stdlib-common</artifactId>
207.              <version>1.4.10</version>
208.            </dependency>
209.
```

```
            <dependency>
              <groupId>org.slf4j</groupId>
              <artifactId>slf4j-api</artifactId>
              <version>1.7.25</version>
            </dependency>

            <dependency>
              <groupId>com.squareup.okhttp3</groupId>
              <artifactId>okhttp</artifactId>
              <version>4.9.2</version>
            </dependency>
          </dependencies>
        </dependencyManagement>
      </project>

```

MYSQL

“CREATE DATABASE tourer;

USE tourer;

CREATE TABLE UserProfile(id INT UNSIGNED NOT NULL AUTO_INCREMENT,

username VARCHAR(40) UNIQUE NOT NULL,

password VARCHAR(40) NOT NULL,

creditcardno VARCHAR(40) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

```
                                    CONSTRAINT PK_ID PRIMARY
KEY(id)
                                );


CREATE TABLE Location(id INT UNSIGNED NOT NULL ,
                                    latitude double NOT NULL,
                                    longitude double NOT NULL,
                                    description TEXT(500) NOT NULL,
                                     name VARCHAR(40) NOT NULL,
                                    CONSTRAINT FK_IDLocation
FOREIGN KEY (id) REFERENCES UserProfile(id),
                                     CONSTRAINT AK_COORDINATE
UNIQUE(id, latitude, longitude)

                                     );


CREATE TABLE likestate(id int UNSIGNED NOT NULL,
                                   name VARCHAR(40) NOT NULL,
                                    nameother VARCHAR(40) NOT
NULL,
                                    type boolean NOT NULL,
```

```
                                CONSTRAINT UK_SET UNIQUE(id,
nameother, name, type)
                                );

SELECT * FROM likestate;

CREATE TABLE CreditCard(id INT UNSIGNED NOT NULL,
                                type VARCHAR(40) NOT NULL,
                                number BIGINT UNSIGNED NOT
NULL,
                                securityCode SMALLINT
unsigned NOT NULL,
                                expirationDate DATE NOT NULL,
                                CONSTRAINT FK_IDUserCard
FOREIGN KEY (id) REFERENCES UserProfile(id),
                                CONSTRAINT PK_IDUserCard
PRIMARY KEY (id)
                                );

ALTER TABLE Location
ADD CONSTRAINT CK_COORDINATE_VALUES CHECK(latitude >=
-90
                                                AND
latitude <= 90
```

```
                                                                AND
longitude >= -180
                                                                AND
longitude <= 180);

ALTER TABLE location
ADD    CONSTRAINT AK_NAME UNIQUE(name, id);

ALTER TABLE Location
ADD likes BIGINT DEFAULT 0,
ADD dislikes BIGINT DEFAULT 0;

ALTER TABLE Location
ADD photo VARCHAR(250) DEFAULT '';

select dislikes from location;

select * from location;



DELETE from location where id = 1 and name = 'asda';
```

```
ALTER TABLE UserProfile

ADD CONSTRAINT CK_USSERNAME
CHECK(CHAR_LENGTH(username) >= 6),

ADD CONSTRAINT CK_PASSWORD CHECK(password REGEXP
'^(?=.*[A-Za-z])(?=.*\\d)(?=.*[@$!%*#?&])[A-Za-z\\d@$!%*#?
&]{8,}$'),

ADD CONSTRAINT CK_EMAIL CHECK(email REGEXP
'^[\\w-\\.]+@([\\w-]+\\.)+[\\w-]{2,4}$');


INSERT INTO UserProfile (creditcardno, username, password,
email) VALUES
('-','admin123','Calebbb123456789*','mihai.gherghinescu00@e
-uvt.ro');

select * from userprofile;


DELETE FROM location WHERE id=17;

SELECT username FROM UserProfile WHERE username REGEXP
'admin' ORDER BY username LIMIT 10;


insert INTO Location (id,latitude, longitude, description, name)
values (1,0,0,'-',"centru_Test");

insert INTO Location (id,latitude, longitude, description, name)
values (1,1,1,'-',"primarie_Test");
```

```
DELETE from location where id=1 AND
latitude=45.74731163282716 AND
longitude=21.236353005371086;

UPDATE Location SET name='centru_Test' where id=1 and
latitude=0 and longitude=0;

INSERT INTO Location(id, latitude, longitude, description,
name) VALUES        (1, 10, 10, '-', 'Location1_Test');

INSERT INTO Location(id, latitude, longitude, description,
name) VALUES        (15, 10, 10, '-', 'Location2_Test');

UPDATE Location

SET name='name', description='description'

WHERE id=1 AND latitude=0 AND longitude=0;

select * from creditcard;

INSERT INTO CreditCard(id, type, number, securityCode,
expirationDate)
VALUES(1,'MasterCard',1234123412341,1234,01/01/2023);

SELECT latitude, longitude, description, name FROM     Location
WHERE id=1;

SELECT latitude, longitude, description, name FROM     Location
WHERE id=( SELECT id FROM UserProfile WHERE
username='admin123');
```

“

JAVASCRIPT/HTML/CSS

```
<!DOCTYPE html>
<html>
  <head>
    <title>Add Map</title>

    <style type="text/css">
      /* Set the size of the div element that contains the map */
      #map {
        height: 800px;
        /* The height is heihgt of the web page */
        width: 100%;
        /* The width is the width of the web page */
      }
    </style>
    <script language="javascript">
      // Initialize and add the map
        var markertoremove = null;
        var targetmarker =  null ;
        var markers = [];
        var locationlat = 0.0;
        var locationlng = 0.0;
        var map;
        var curLocation = "Giroc, Strada Frasinusului";
        const timisoara = { lat: 45.75372, lng: 21.22571 };

        function initMap() {
          var directionsService = new google.maps.DirectionsService();
          var directionsRenderer = new google.maps.DirectionsRenderer();
            // The location of Timisoara

            // The map, centered at Timisoara


            if(markertoremove != null){
```

```
  markers.push(markertoremove.position);
}
    map = new google.maps.Map(document.getElementById("map"), {
    zoom: 14,
    center: timisoara,
    mapTypeControl: false,
    streetViewControl: false,
    mapTypeId: 'hybrid'
});
directionsRenderer.setMap(map);


if(targetmarker != null){
    //addTargetMarker(targetmarker);
    calcRoute();
    targetmarker = null;
}
for(var i = 0; i < markers.length; i++){
  addMarker(markers[i]);

}
markertoremove = null;
//Add click listener to map

google.maps.event.addListener(map, 'rightclick',
function(event){


    //add marker to map
    if(markertoremove != null)
        markertoremove.setMap(null);

    locationlat = event.latLng.lat();
    locationlng = event.latLng.lng();
    addMarker(event.latLng);
});
function calcRoute() {


  if (navigator.geolocation) {

    var start = curLocation;
    var end = targetmarker;

          directionsService.route({
```

```
                    origin: start,
                    destination: end,
                    travelMode: 'DRIVING'
                  })
                  .then((response => {

                    directionsRenderer.setDirections(response);
                  }))
                  .catch((e) => window.alert("Request failed to " + status));
            }




          }

          function addMarker(location){

              var marker = new google.maps.Marker({
                  position: location,
                  map: map

              });
              markertoremove = marker;
          }
          function addTargetMarker(target){

            var marker = new google.maps.Marker({
              position: target,
              map: map,
              icon:'https://developers.google.com/maps/documentation/javascript
/examples/full/images/beachflag.png',
              scaledSize: new google.maps.Size(200, 200)
            });

          }




      }
      function getLocationOfCommit(){
            return {lat: locationlat, lng: locationlng};
```

```
        }


        function resetTargetMarker(){
          targetmarker = null;
          initMap();
        }

        function setTargetMarker(lat, lng){
          markertoremove = null;
          targetmarker = {lat : lat, lng : lng};
          initMap();

        }

        function addMarkerToList(lat, lng){
          markers.push({lat: lat, lng : lng});

        }

        function getCurrentLocation(){
              if (navigator.geolocation) {
                navigator.geolocation.getCurrentPosition(
                  (position) => {
                    const pos = {
                      lat: position.coords.latitude,
                      lng: position.coords.longitude,
                    };

                    return pos;
                  }
                );
                return pos;
              } else {

                return {lat: -999, lng: -999};
              }
        }



  </script>
</head>
<body>
```

```
    <!--The div element for the map -->
    <div id="map"></div>

    <!-- Async script executes immediately and must be after any DOM elements used
in callback. -->
    <script
      src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCqAeOnRkRu6zylBXyMP0
ilbvspn_pT5tE&callback=initMap&libraries=&v=weekly"
      async
    ></script>
  </body>
</html>
```