Hogeschool van Amsterdam

Fasten Your Seatbelts

Source Code

Tijmen Stor, 500752989, IS205

Hogeschool van Amsterdam

Inhoudsopgave

Customer Overview

Voor het baggagesysteem van het FYS project was ik verantwoordelijk voor de functies en de lay-out van de Customer Overview. Hieronder de code die ik had gemaakt voor de Class.

```
public class CustomerOverview extends BorderPane {

    private DbManager dbManager;
    private Stage primaryStage;

    private ObservableList<CustomerRecord> data
            = FXCollections.observableArrayList();
    private ObservableList<CustomerRecord> tableData
            = FXCollections.observableArrayList();
    private ObservableList<CustomerRecord> searchResults
            = FXCollections.observableArrayList();

    private TableView<CustomerRecord> tableView4;

    private VBox controlBox = new VBox();
    private HBox topBar1 = new HBox();
    private HBox topBar2 = new HBox();
    private BorderPane border1 = new BorderPane();
          private  Image corLogo = new Image("Corendon.png");
      private  ImageView logo = new ImageView();

    private Button refresh = new Button("Refresh table");
    private Button back = new Button("Back");
    private Button searchButton = new Button("Search");
    private TextField searchBar = new TextField();
    private Label tableStatus = new Label("Search customers:");
    private boolean isShowingSearch = false;

    public void initScreen(Stage primaryStage) {
        this.primaryStage = primaryStage;
        dbManager = new DbManager();

        //alle data uit database in tabledata
        this.data = dbManager.getCustomerListFromDB();
        for (int i = 0; i < data.size(); i++) {
            tableData.add(data.get(i));
        }
        //tableview maken van de dbmanager table
        tableView4 = dbManager.createCustomerTable();

        this.setTop(topBar1);
        this.setRight(controlBox);
        this.setCenter(border1);
        border1.setCenter(tableView4);

        //corendon logo
        logo.setImage(corLogo);
        logo.setFitWidth(300);
        logo.setPreserveRatio(true);
        logo.setSmooth(true);

        //tablekolommen vullen de hele breedte van de tabel

tableView4.setColumnResizePolicy(TableView.CONSTRAINED_RESIZE_POLICY);
//topbar vullen met logo, zoekfunctie en refresh functie
        topBar1.getChildren().addAll(topBar2, tableStatus, searchBar,
searchButton, refresh);
        searchButton.setMinSize(20, 25);
        topBar1.setSpacing(30);
        topBar1.setMinHeight(50);
        topBar1.setAlignment(Pos.CENTER);
        topBar2.getChildren().addAll(logo);
        topBar2.setAlignment(Pos.CENTER_RIGHT);

        //grootte van table en table vullen met de database data
        tableView4.setMinSize(1300, (25 * 24) + 26);
        tableView4.setMaxSize(1300, (25 * 24) + 26);
        tableView4.setItems(this.tableData);

        //refresh knop voert refresh methode uit
        refresh.setOnAction((ActionEvent e) -> {
            for (int i = 0; i < tableView4.getItems().size(); i++) {
                tableView4.getItems().clear();
            }
            updateData();

        });
        //de searchknop voert ook de seach methode uit
        searchButton.setOnAction((ActionEvent e) -> {
            isShowingSearch = true;
            searchItems();
            tableStatus.setText("Search Results:");
            topBar1.getChildren().add(back);
        });
        //zet alles terug op zijn oude plek
        back.setOnAction((ActionEvent e) -> {
            isShowingSearch = false;
            tableView4.setItems(tableData);
            tableStatus.setText("Search customers:");
            topBar1.getChildren().removeAll(back);
        });
    }
    //methode om te zoeken naar klanten in de database
    public void searchItems() {
        searchResults.clear();
        String keyword = searchBar.getText();
        for (CustomerRecord record : tableData) {
            SimpleStringProperty[] properties = record.toArray();
            boolean relevance = false;
            for (int i = 0; i < properties.length; i++) {
                if (keyword.equals(properties[i].getValueSafe())) {
                    relevance = true;
                }
                System.out.println(properties[i].toString());
                System.out.println(relevance);
            }
            if (relevance == true) {
                searchResults.add(record);
            }
        }
        tableView4.setItems(searchResults);
        System.out.println(searchResults.toString());
    }
```

```
//refresht de data in table
    public void updateData() {
        data = dbManager.getCustomerListFromDB();
        for (int i = 0; i < this.data.size(); i++) {
            this.tableData.add(this.data.get(i));
        }
        tableView4.setItems(this.tableData);
    }
}
```

User Overview

Ik was ook mede verantwoordelijk voor het maken van het overzicht van de gebruikers. Dit deed ik voor een deel samen met Jeroen. Hieronder de code.

```
public class UsersOverview extends BorderPane {

    private DbManager dbManager;

    private Button addBut = new Button("Submit");
    private Button cancel = new Button("Cancel");
    private Button refresh = new Button("Refresh table");
    private Button delete = new Button("Delete from table");
    private Button back = new Button("Back");
    private Button b = new Button("Add user");
    private Button searchButton = new Button("Search");

    private Image corLogo = new Image("Corendon.png");
    private ImageView logo = new ImageView();

    private HBox topBar = new HBox();
    private HBox topBar2 = new HBox();
    private VBox sideBar = new VBox();
    private BorderPane border1 = new BorderPane();

    private TextField searchBar = new TextField();
    private Label tableStatus = new Label("Search:");
    private Stage primaryStage;

    private ObservableList<UserRecord> data
            = FXCollections.observableArrayList();
    private ObservableList<UserRecord> tableData
            = FXCollections.observableArrayList();
    private ObservableList<UserRecord> searchResults
            = FXCollections.observableArrayList();

    private Connection conn;
    private PreparedStatement prepS = null;
    private TableView<UserRecord> tableView4 = new TableView();
    private boolean isShowingSearch = false;

    public void initScreen(Stage primaryStage) {
        this.primaryStage = primaryStage;

        dbManager = new DbManager();

        // informatie uit de database halen en in tabledata stoppen
        data = dbManager.getUserListFromDB();
        for (int i = 0; i < this.data.size(); i++) {
            this.tableData.add(this.data.get(i));
        }

        //Corendon logo top
        logo.setImage(corLogo);
        logo.setFitWidth(300);
        logo.setPreserveRatio(true);
        logo.setSmooth(true);

        //usertabel tableview halen uit dbManager
        tableView4 = dbManager.createUserTable();
        this.setTop(topBar);
        this.setRight(sideBar);
        this.setCenter(border1);
        border1.setCenter(tableView4);

        //logo toevoegen aan de topbar
        topBar2.getChildren().addAll(logo);

        // topbar voor corendon logo + zoeken in de table
        topBar.getChildren().addAll(topBar2, tableStatus, searchBar,
searchButton);
        topBar.setSpacing(30);
        topBar.setAlignment(Pos.CENTER);

        //ruimte tussen sidebar en table + knoppen toevoegen
        sideBar.setPadding(new Insets(10, 10, 10, 10));
        sideBar.getChildren().addAll(b, delete, refresh);

        //grootte van de buttons
        refresh.setMinSize(200, 75);
        delete.setMinSize(200, 75);
        b.setMinSize(200, 75);

        //refresh knop voert methode uit
        refresh.setOnAction((ActionEvent e) -> {
            for (int i = 0; i < tableView4.getItems().size(); i++) {
                tableView4.getItems().clear();

            }
            data.clear();
            tableData.clear();
            updateData();
        });

        //delete knop voert methode uit
        delete.setOnAction((ActionEvent e) -> {
            deletePerson();
        });

        //search knop voert methode uit
        searchButton.setOnAction((ActionEvent e) -> {
            isShowingSearch = true;
            searchItems();
            tableStatus.setText("Search Results:");
            topBar.getChildren().add(back);
        });

        //back knop
        back.setOnAction((ActionEvent e) -> {
            isShowingSearch = false;
            tableView4.setItems(tableData);
            tableStatus.setText("Search customers:");
            topBar.getChildren().removeAll(back);
        });
        // opent venster om usert toe te voegen
        b.setOnAction((ActionEvent e) -> {
            addUser(primaryStage);
        });

        //table niet resizable
```

```
tableView4.setColumnResizePolicy(TableView.CONSTRAINED_RESIZE_POLICY);

        //table grootte veranderen en vullen
        tableView4.setMinSize(1000, 700);
        tableView4.setMaxSize(1200, 800);
        tableView4.setItems(this.tableData);

    }

    //de stage voor toevoegen user
    public void addUser(Stage primaryStage) {

        final GridPane grid1 = new GridPane();

        //spreekt voor zich
        Label usernameLB = new Label("Username:   ");
        Label passwordLB = new Label("Password    ");
        Label firstnameLB = new Label("First name:    ");
        Label tussenLB = new Label("Prefix:    ");
        Label surnameLB = new Label("Surname:   ");
        Label emailLB = new Label("Email:    ");
        Label functionLB = new Label("Function:    ");

        //spreekt voor zich
        TextField usernameTX = new TextField();
        usernameTX.setPromptText("Username");
        TextField passwordTX = new TextField();
        passwordTX.setPromptText("Password");
        TextField firstnameTX = new TextField();
        firstnameTX.setPromptText("First name");
        TextField tussenTX = new TextField();
        tussenTX.setPromptText("Prefix");
        TextField surnameTX = new TextField();
        surnameTX.setPromptText("Surname");
        TextField emailTX = new TextField();
        emailTX.setPromptText("Email");
        TextField functionTX = new TextField();
        functionTX.setPromptText("Function");

        //stage voor het scherm
        final Stage adduserStage = new Stage();

        // zorgt ervoor dat er nergens anders behalve het main venster
gedrukt kan worden
        adduserStage.initModality(Modality.APPLICATION_MODAL);
        adduserStage.initOwner(primaryStage);

        //logisch deze
        cancel.setMinSize(70, 20);
        addBut.setMinSize(70, 20);
        // boxen (getal is voor spacing)
        VBox useraddVragen = new VBox(10);
        VBox useraddAntwoorden = new VBox(10);
        HBox buttonBox = new HBox(10);

        useraddVragen.setPadding(new Insets(10, 10, 10, 10));

        // buttons toevoegen aan een hbox, toevoegen aan grid en de padding
aanpassen
        buttonBox.getChildren().addAll(addBut, cancel);
        grid1.getChildren().addAll(useraddVragen, useraddAntwoorden,
buttonBox);
        addBut.setPadding(new Insets(1, 1, 1, 1));
        cancel.setPadding(new Insets(1, 1, 1, 1));
        cancel.setAlignment(Pos.CENTER);

        // plaatsen van alle labels, buttons en textfields
        grid1.setVgap(15);
        grid1.setHgap(15);
        grid1.add(usernameLB, 1, 1);
        grid1.add(usernameTX, 2, 1);
        grid1.add(passwordLB, 1, 2);
        grid1.add(passwordTX, 2, 2);
        grid1.add(firstnameLB, 1, 3);
        grid1.add(firstnameTX, 2, 3);
        grid1.add(tussenLB, 1, 4);
        grid1.add(tussenTX, 2, 4);
        grid1.add(surnameLB, 1, 5);
        grid1.add(surnameTX, 2, 5);
        grid1.add(emailLB, 1, 6);
        grid1.add(emailTX, 2, 6);
        grid1.add(functionLB, 1, 7);
        grid1.add(functionTX, 2, 7);
        grid1.add(addBut, 2, 8);
        grid1.add(cancel, 3, 8);

//addBut leest de gegevens in en zet ze in database
        addBut.setOnAction((ActionEvent e) -> {
            PreparedStatement prepS = null;
            try (Connection conn = Sql.DbConnector();) {
                primaryStage.setTitle("Adding user");

                String query = "insert into users"
                        + "(username, password, firstname, tussenvoegsel,
surname, email,  function) VALUES"
                        + "(?, ?, ?, ?, ?, ?, ?)";
                prepS = conn.prepareStatement(query);
                prepS.setString(1, usernameTX.getText());
                prepS.setString(2, passwordTX.getText());
                prepS.setString(3, firstnameTX.getText());
                prepS.setString(4, tussenTX.getText());
                prepS.setString(5, surnameTX.getText());
                prepS.setString(6, emailTX.getText());
                prepS.setString(7, functionTX.getText());

                if (usernameTX.getText().isEmpty() ||
passwordTX.getText().isEmpty() || firstnameTX.getText().isEmpty()
                        || surnameTX.getText().isEmpty() ||
functionTX.getText().isEmpty()) {

                    Alert alert1 = new Alert(Alert.AlertType.WARNING);
                    alert1.setTitle("Adding user");
                    alert1.setHeaderText(null);
                    alert1.setContentText("Some information is not filled
in, please try again.");
                    alert1.showAndWait();

                } else {

                    Alert alert = new Alert(Alert.AlertType.INFORMATION);
                    alert.setTitle("Adding user");
                    alert.setHeaderText(null);
                    alert.setContentText("User added succesfully");
                    alert.showAndWait();
                    adduserStage.close();
                    prepS.executeUpdate();

                }
            } catch (Exception e1) {
                Alert alert = new Alert(Alert.AlertType.WARNING);
                alert.setTitle("Corendon - Luggage");
                alert.setHeaderText(null);
                alert.setContentText("There is an error in the database, 
please try again later.");
                alert.showAndWait();
                System.out.println("SQL ERROR");
                System.err.println(e1);

            }

        });
        cancel.setOnAction((ActionEvent e) -> {
            adduserStage.close();
        });

        //scene bouwen enzo
        Scene dialogScene = new Scene(grid1, 500, 500);
        adduserStage.setScene(dialogScene);
        adduserStage.show();

    }

    //refresh methode voor de table
    public void updateData() {

        data = dbManager.getUserListFromDB();
        for (int i = 0; i < this.data.size(); i++) {
            this.tableData.add(this.data.get(i));
        }
        tableView4.setItems(this.tableData);

    }




//functie voor zoeken in database
    public void searchItems() {
        searchResults.clear();
        String keyword = searchBar.getText();
        for (UserRecord record : tableData) {
            SimpleStringProperty[] properties = record.toArray();
            boolean relevance = false;
            for (int i = 0; i < properties.length; i++) {
                if (keyword.equals(properties[i].getValueSafe())) {
                    relevance = true;
                }
                System.out.println(properties[i].toString());
                System.out.println(relevance);
            }
            if (relevance == true) {
                searchResults.add(record);
            }
        }
        tableView4.setItems(searchResults);
        System.out.println(searchResults.toString());
    }

    //methode voor deleten uit database
    public void deletePerson() {

        int selectedIndex =
tableView4.getSelectionModel().getSelectedIndex();
        if (selectedIndex >= 0) {
            Alert alert = new Alert(AlertType.CONFIRMATION, "Delete
person?", ButtonType.YES, ButtonType.NO, ButtonType.CANCEL);
            alert.setHeaderText(null);
            alert.showAndWait();

            if (alert.getResult() == ButtonType.YES) {
                PreparedStatement prepS = null;

                try (Connection conn = Sql.DbConnector();) {

                    String query = "delete from users where user_id = ?";
                    prepS = conn.prepareStatement(query);
                    prepS.setString(1,
tableView4.getSelectionModel().getSelectedItem().getUser_id());
                    prepS.executeUpdate();

System.out.println(tableView4.getSelectionModel().getSelectedItem().getUser
_id());
                    tableView4.getItems().remove(selectedIndex);
                } catch (Exception e1) {
                    System.out.println("SQL ERROR");
                    System.err.println(e1);

                }

            }

        } else {

            Alert alert = new Alert(AlertType.WARNING);
            alert.setTitle("No Selection");
            alert.setHeaderText(null);
            alert.setContentText("Please select a person in the table.");

            alert.showAndWait();
        }
    }

}
```