

**Universiteti i Prishtinës “Hasan Prishtina”**  
**Fakulteti i Inxhinierisë Elektrike dhe Kompjuterike**



**PUNIM SEMINARIK**

**Lënda: Komunikimi Njeri-Kompjuter**

**Profesor:**  
**Prof. Dr. Isak Shabani**

**Punuar nga:**  
**Rina Jasharaj**  
**Riga Dibrani**  
**Valëza Grainca**  
**Donat Sinani**  
**Erlis Lushtaku**  
**Enes Hasani**  
**Vesa Galica**  
**Petrit Luta**

# Përmbajtja

<b>Hyrje</b>	<b>3</b>
<b>Kartela e veturave(Car Card)</b>	<b>4</b>
CarCardController	5
CarCardComponent	5
<b>Lista e veturave(Car List)</b>	<b>7</b>
PageBtnComponent	8
CarListController	11
<b>1. Insertimi dhe editimi i makinave</b>	<b>13</b>
1.1. Pamja për insert/edit	13
1.2. Kodi për insert/edit	14
<b>2.Login/Register</b>	<b>18</b>
2.1.Login /Register views	18
2.2.Login/Register controllers	19
<b>3. GJUHËSIA</b>	<b>25</b>

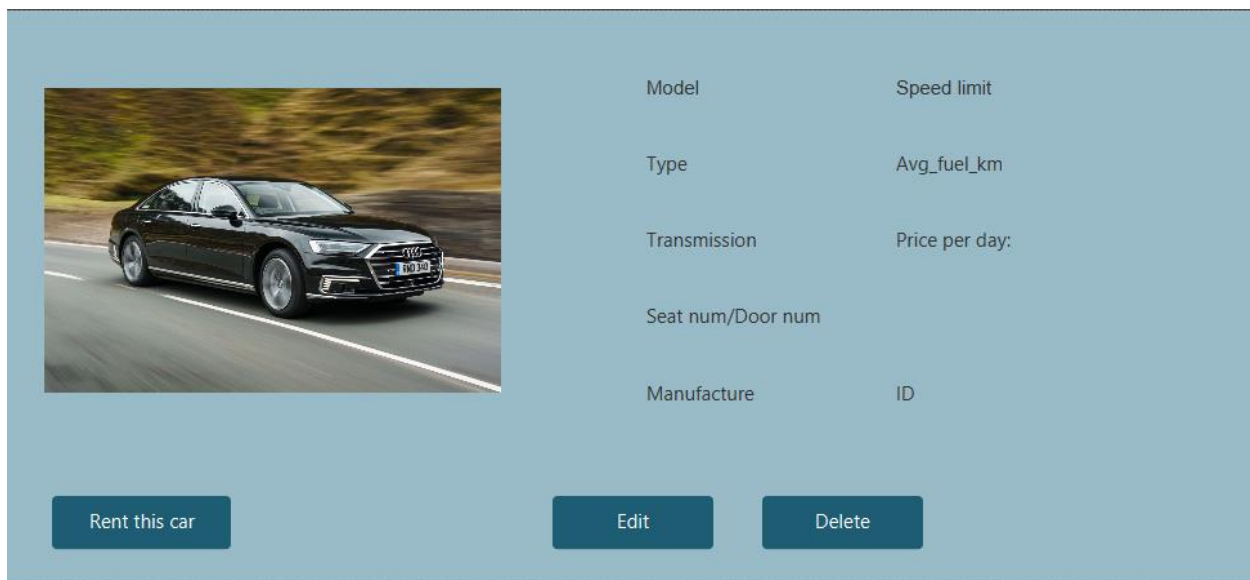
# Hyrje

Në këtë seminar do të flasim për punën ekipore që kemi bërë në projektin në lëndën Komunikimi Njeri-Kompjuter.

Gjatë këtij seminari, në pjesë të ndryshme të tij kemi sqaruar deri në detaje pjesët respektive të projektit. Që nga pjesa e kodit e deri tek pamja në Scene Builder, të gjitha janë të përmbledhura në këtë seminar.

Aplikacioni që kemi zhvilluar ka të bëjë me marrjen e veturave me qera. Meqenëse kjo temë është mjaft aktuale, jo vetëm në kohët tona por qysh moti, kemi përfshirë pjesë të shumta në projekt, që e bëjnë aplikacionin tonë shumë të përdorshëm dhe praktik. Marrja e veturave me qera, nëse bëhet online, e lehtëson shumë punën e njerëzve të cilët sa herë dëshirojnë të marrin veturë me qera, shkojnë tek vendi i caktuar dhe bëjnë kilometra të tëra, për një punë që mund të kryhet nga rehatia e shtëpisë, siç e mundëson aplikacioni që ne kemi zhvilluar me njohuritë nga lënda Komunikimi Njeri-Kompjuter.

# Kartela e veturave(Car Card)



-Kjo kartelë të dhënat(fushat si: model, type, id, manufacture) i merr në mënyrë dinamike nga databaza, poashtu edhe foton e veturës. Përpos fushave kjo kartelë ka gjithsej 3 funksionalitete tjera, pra butonat rent this car, edit dhe delete.

- Rent this car - përdoret kur dëshirojmë një veturë ta vendosim si rented.
- Edit - përdoret për editimin e vetive përkatësisht fotos së veturës.
- Delete - përdoret për fshirjen e një karteje nga lista e veturave.

-Shqyrtojmë FXML-in e kartelës së veturës, përmban një VBox dhe dy HBox-a të futur brenda, VBox-i sipër përmban brenda vetës dy VBox-a tjerë, njëri prej tyre përmban foton e tjetri përmban një GridPane për sistemimin e vetive të veturës. HBox-i i poshtëm shërben për vendosjen e butonave të përmendur më sipër.

-Për vetitë e veturës sigurisht që na nevojitet një model i veturës, pra një klasë Car e cila i përmban të gjitha vetitë e veturës të cilat janë:

```
1 package models;
2 import java.util.Date;
3
4 public class Car {
5     private int id;
6     private int publisher;
7     private int manufacture;
8     private String model;
9     private double price_per_day;
10    private double avg_fuel_km;
11    private Transmission transmission;
12    private double speed_limit;
13    private Type type;
14    private int seat_num;
15    private int door_num;
16    private Date inserted_at;
17    private Date updated_at;
18    private String car_img;
19 }
```

-Për kartelë sigurisht që na duhet një Controller për të.

## CarCardController

-Ky kontrolleri i kartelës së veturës i ka të gjitha fushat e qasshme me FXML, dhe ka një funksion setCar i cili shërben për vendosjen e vetive të një veture në Text fields të vetive.

```
public void setCar(Car car) {
    try {
        id.setText("Identifier: " + car.getId());
        model.setText("Model: " + car.getModel());
        transmission.setText("Transmission: "+car.getTransmission());
        type.setText("Type: " + car.getType());
        seatnum_doornum.setText("door num: "+car.getDoor_num()+"/seat num:"+car.getSeat_num());
        manufacture.setText("manufacture id:"+car.getManufacture());
        speedlimit.setText("speed limit: " + car.getSpeed_limit());
        avg_fuel_km.setText("Fuel consumption: "+(int)car.getAvg_fuel_km());
        price_per_day.setText("Price per day: "+(int)car.getPrice_per_day());
        Image image = new Image(getClass().getResource("name: ../../"+car.getCar_img()).toURI().toString());
        img.setImage(image);
    }
}
```

-Pjesa e fundit e kodit ka të bëjë me marrjen e imazheve në mënyrë dinamike (2 rreshtat e fundit). Si parametër të imazhit i kemi dërguar path-in relativ të fotos ndaj CarCardController, i cili konvertohet në URI format ngase java shpeshherë nuk i merr path-et si plain stringje.

-Përpos setCar, kartela ka edhe 3 funksione të cilat shërbejnë si Event Handlers për butonat e përmendur lartë, të cilët i shqyrtojmë më vonë, tani për tani mjafton të dihet se këto funksione marin si parametër Event Handler i cili do implementohet në CarCardListController.

```
public void setOnEditAction(EventHandler<ActionEvent> handler) { this.editButton.setOnAction(handler); }
public void setOnDeleteAction(EventHandler<ActionEvent> handler) { this.deleteButton.setOnAction(handler); }
public void setOnShowButton(EventHandler<ActionEvent> handler) { this.showButton.setOnAction(handler); }
```

## CarCardComponent

-Jashta CarCardController na duhet edhe një component për veturë, ideja mrapa saj është që ta trajtojë një kartelë të veturës sikur një node apo nyje, edhe që ne të mundemi me i vendos në mënyrë distinkte.

-Metoda getContent mer si parametra një veturë dhe 3 event handlera për butonat, poashtu në kuadër të node e vendos FXML-in përkatës pra car-card.fxml, e pastaj vendos për një node në mënyrë të ndarë nga node-s tjera kontrollerat përkatës(pra që editimi, fshirja e rent butonave të funksionojë në mënyrë unike pë secilën kartelë).

```

11 public class CarCardComponent {
12     public Node getContent(Car car, EventHandler<ActionEvent> editHandler, EventHandler<ActionEvent> deleteHandler,
13         EventHandler<ActionEvent> showHandler) throws Exception {
14
15         FXMLLoader loader = new FXMLLoader();
16         loader.setLocation(getClass().getResource("name: \"../views/partials/car-card.fxml\""));
17         Node node = loader.load();
18
19         CarCardController controller = loader.getController();
20         controller.setCar(car);
21
22         controller.setOnEditAction(editHandler);
23         controller.setOnDeleteAction(deleteHandler);
24         controller.setOnShowButton(showHandler);
25
26         return node;
27     }
28 }

```

-Pas implementimit të kësaj metode kemi implementuar në CarCardListController metodat përkatëse që janë event handlers për butonat e kartelës së veturave:

```

public void rentCar(Car car){
    try {
        FXMLLoader loader = new FXMLLoader();
        loader.setLocation(getClass().getResource("name: \"../views/rented-car.fxml\""));
        Pane pane = loader.load();
        RentedCarController controller = loader.getController();
        controller.setId(car);
        parentController.setView(MainScreenController.CARS_DETAILS_VIEW, pane, controller);
    } catch (Exception e) {
        ErrorPopupComponent.show(e);
    }
}

private void removeCar(Car car) {
    try {
        CarRepo.remove(car.getId());
        //System.out.println("aa");
        showCars(paginationComponent.getCursor());
    } catch (Exception e) {
        ErrorPopupComponent.show(e);
    }
}

private void showCar(Car car) {
    try {
        FXMLLoader loader = new FXMLLoader();
        loader.setLocation(getClass().getResource("name: \"../views/cars-details.fxml\""));
        Pane pane = loader.load();
        CarsDetailsController controller = loader.getController();
        controller.setModel(car);
        controller.setEditable(true);
        parentController.setView(MainScreenController.CARS_DETAILS_VIEW, pane, controller);
    } catch (Exception e) {
        ErrorPopupComponent.show(e);
    }
}

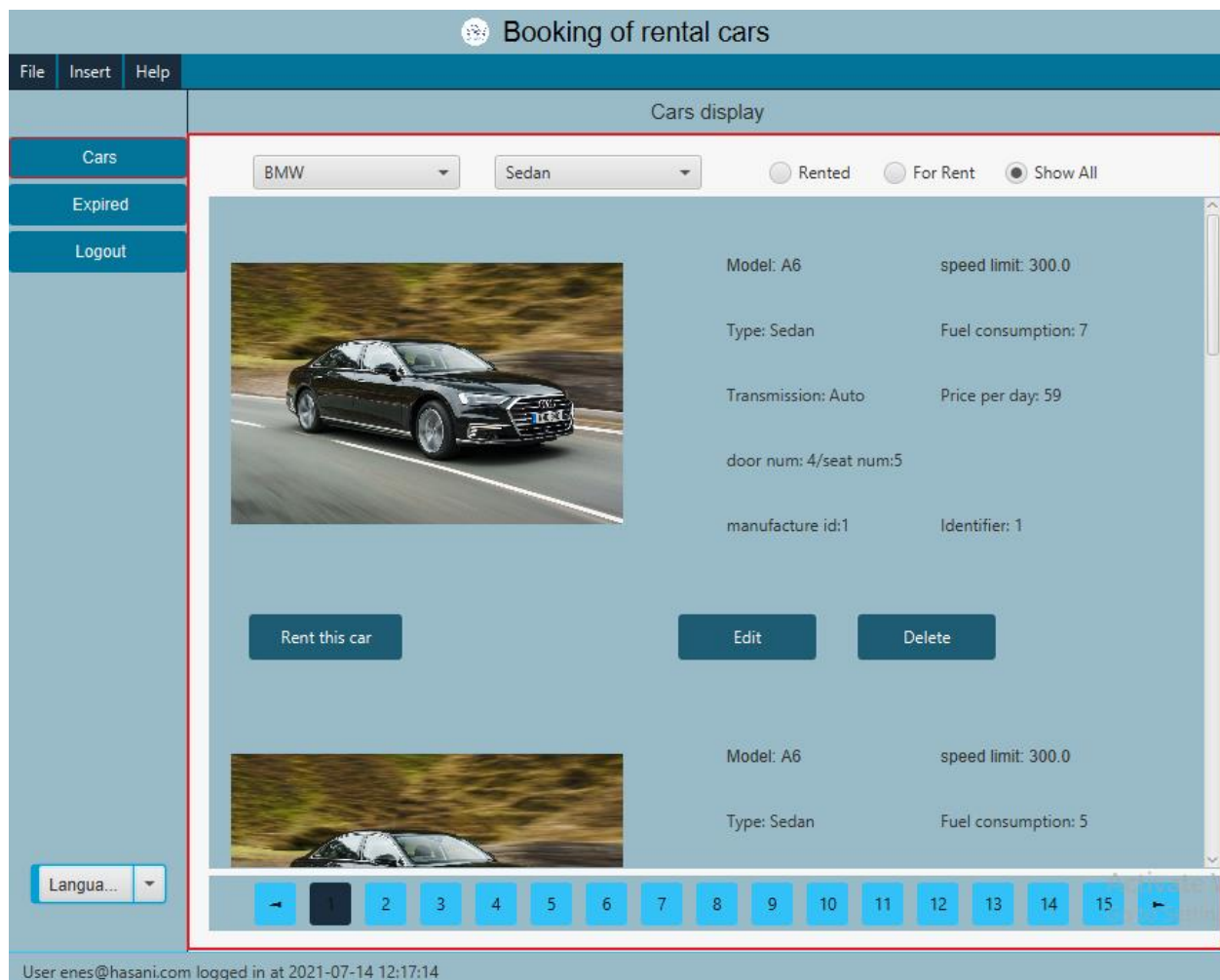
```

-Metoda remove car fshin një kartelë duke thirrur metodën e implementuar në CarRepo e cila është repository për menaxhimin e query-ve për databazë, pra thjesht remove përmban një query

e cila bën delete nga tabela ku gjendet vetura me id-në përkatëse, edit dhe rent na dërgojnë në fxml-ët përkatës (metodat brenda tyre komentohen tek insertimi).

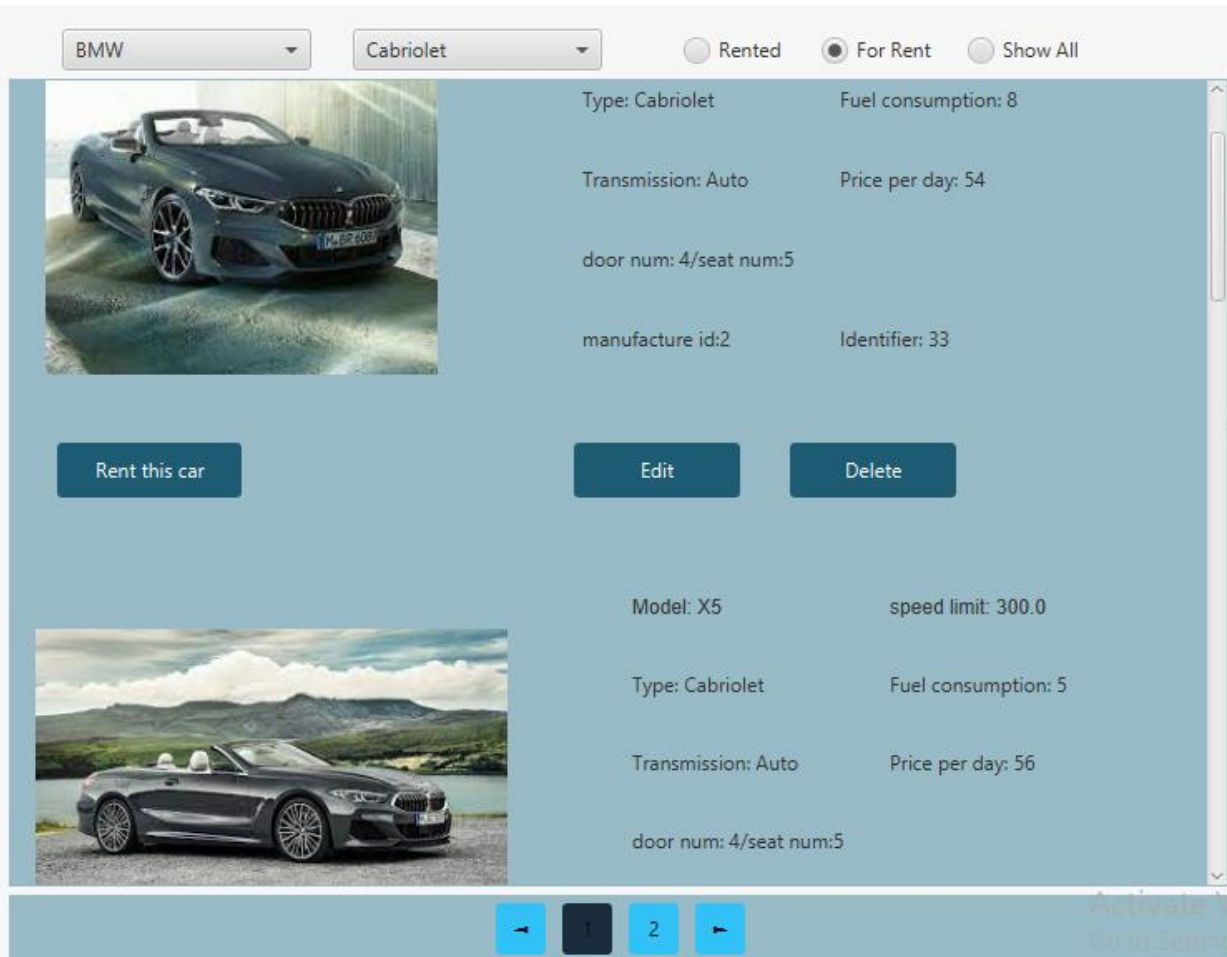
-Këto event handlers thirren prapë tek CarListController, pamjet e kodit të cilat gjenden më poshtë.

## Lista e veturave (Car List)



Pamja e nënvizuar në të kuqe (kufiri i kuq) është një pamje e të gjitha veturave shënimet e të cilave gjenden në databazë. Kjo pamje shfaqet automatikisht pasi që useri të kyçet në faqe. Kjo pamje aktualisht është një vertical box që përmban një horizontal box (pjesa e epërme me combo boxes dhe radiobutona), një scroll pane në të cilën është vendosur një vertical box që përmban disa view që përmbajnë fushat me të dhëna të cilat popullohen në mënyrë dinamike nga databaza. Në fund kemi një horizontal box i përmban butona, numri i të cilëve është i barabartë

me numrin e faqeve të nevojshme për t'i paraqitur në këtë rast të gjitha veturat. Kjo listë mund të popullohet me vetura (rekorde nga databaza) në disa raste, i pari vetëm se u cek, rasti kur përdoruesi kyçet. Në pjesën e epërme gjenden dy comboboxa dhe tre radio butona të cilët janë toggle në një grup. Nga combobox-i i parë mund të zgjedhim brandin e veturave (BMW, AUDI ose Mercedes), nga combobox-i i dytë mund të zgjedhim llojin e veturave për atë brand (Sedan, SUC, Cabriolet apo Sports Car), ndërsa përmes radiobutonave zgjedhim nëse dëshirojmë ti shohim ato që janë rented apo ato që janë për rent, apo të gjitha të atij llojit. Butonat “Rent this car”, “Edit” dhe “Delete” shërbejnë për ta lëshuar me qira një veturë, për ta edituar dhe për ta fshirë, respektivisht. Shqyrtohen poashtu me një seksion tjetër kur shpjegohet “Car Card Component”.



Një pamje tjetër ku janë selektuar veturat BMW të tipit Cabriolet që janë për t'u lëshuar me qira (For Rent). Vëreni se si numri i butonave është reduktuar në 2 (pasi që kemi më pak vetura dhe mjaftojnë dy faqe për t'i shfaqur). Le të shohim kodin për këtë.



## PageBtnComponent

Së pari të fillojmë me klasën PageBtnComponent. Esencialisht funksioni i kësaj klase është të vendosë butonat në pjesën e poshtme të pamjes aq sa është numri i faqeve i nevojshëm për t'i shfaqur të gjitha recordat e marra nga databaza, po ashtu dy butona për të kaluar nga “faqja” në “faqe”. Funksion tjetër i kësaj klase është vendosja e eventHandlers në secilin buton përkatës. Secili buton e ekzekuton një query të ri në databazë. Komponenti kyç i kësaj klase është subinterfejsi PaginationHandler i cili duket kështu:

```
public interface PaginationHandler {  
    public void run(Object page);  
}
```

Ky subinterfejs definon metodën run, pra cdo instance e një klase që implementon këtë interfejs duhet ta implmentojë këtë metodë(cdo instancë e klasës konkrete). Metoda run si parametër mund të merr cfaredo tipi të dhënave (Object page). Kjo ka qenë e nevojshme pasi që është dashur njëherë ta thërrasim metodën run me një String si parametër, njëherë me një Integer si parametër.

```
public class PageBtnComponent {  
    private int cursor;  
    private int totalItems;  
    private int itemsPerPage;  
    private int totalPages;  
  
    public PageBtnComponent(int totalItems, int itemsPerPage) {  
        this.totalItems = totalItems;  
        this.itemsPerPage = itemsPerPage;  
        this.cursor = 0;  
        this.totalPages = (int) (Math.ceil(((double) this.totalItems / (double) this.itemsPerPage)));  
    }  
  
    public int getCursor() { return this.cursor; }
```

Kjo klasë ka një konstruktor i cili merr dy int si parametra formal, përkatësisht merr numrin total të faqeve dhe numrin e veturave të cilat dëshirojmë ti shfaqim në një faqe. Konstruktori ia vendos objektit përkatës numrin total të faqeve dhe numrin e veturave për një faqe(itemsPerPage) e vendos cursorin (tregon cili buton është aktiv përkatësisht cila faqe) në 0 dhe e llogarit numrin e faqeve totale (totalPages) si funksioni i sipërm(Math.ceil) i raportit në mes të numrit total të veturave që do të shfaqen dhe veturave që shfaqen në një “faqe”(që i korrespondon një butoni). Funksioni getCursor() e kthen kursorin.

```

public void render(Pane paginationPane, PaginationHandler handler, String queryString) {
    if (totalPages == 0) return;
    if(queryString == null){
        buttonRender(paginationPane,handler);
    }
    else{
        buttonRender(paginationPane,handler,queryString);
    }
}
}

```

Funksioni render e merr një pane, një objekt të një klase që implmenton subinterfejsin Pagination handle, dhe një queryString. Nëse numri i faqeve është 0 nuk kryen ndonjë funksion sepse i bie që totalItems është 0 dhe nuk ka nevojë të vendosë ndonjë buton.

Pastaj varësisht nëse është null apo jo parametri queryString e thirr funksionin e mbingarkuar buttonRender me dy ose tre parametra.

```

private void buttonRender(Pane paginationPane, PaginationHandler handler,String queryString){

    paginationPane.getChildren().clear();
    ArrayList<Button> pageButtons = new ArrayList<>();
    for (int i = 0; i < totalPages; i++) {
        final int page = i;
        Button button = new Button(Integer.toString( i page + 1));
        button.setOnAction(event -> {
            String newQString = queryString.replaceAll( regex: "[\\d]+$",String.valueOf(page*itemsPerPage));
            handler.run(newQString);
            cursor = page;
            pageButtons.forEach(btn -> btn.getStyleClass().remove( o: "active-page"));
            button.getStyleClass().add("active-page");
        });
        pageButtons.add(button);
    }
    pageButtons.get(cursor).getStyleClass().add("active-page");

    Button prevButton = new Button( s: "←");
    prevButton.setOnAction(event -> {
        if (cursor - 1 < 0) return;
        cursor--;
        String newQString = queryString.replaceAll( regex: "[\\d]+$",String.valueOf(cursor*itemsPerPage));
        handler.run(newQString);
        pageButtons.forEach(btn -> btn.getStyleClass().remove( o: "active-page"));
        pageButtons.get(cursor).getStyleClass().add("active-page");
    });
}

```

Detyra e këtij funksioni është t'ia vendos numrin e saktë të butonave paginationPane që e merr si parametër si dhe atyre butonave t'ua vendosë nga një eventHandler në bazë të queryString. Numri i butonave në faqe është sa numri i faqeve totale(totalPages) andaj aq herë iterojmë dhe aq butona shtojmë. Në unazën for pasi që të krijojmë butonin menjëherë i'a shtojmë eventHandler in i cili e ka kuptimin: në klik të këtij butoni bëje replace numrin e fundit në queryString me page\_itemsPerPage pastaj thirre metodën run të objektit handler(e popullon faqen që e representon butoni me veturat e duhura). Vendose kursorin te kjo faqe. Secilit buton largoja klasën “active-page” ndërsa këtij butoni shtoja klasën active-page.

Në vazhdim e shton buttonin prevButton i cili përdoret për tu zhvendosur nga një faqe përkatësisht buton majtas, funksionaliteti i këtij butoni është komplet analog me butonet e

më sipërme andaj nuk do të shpjegohet. Në mënyrë të ngjashme shtohet edhe nextButton dhe në analogji totale iu shtohet funksionaliteti.

```
paginationPane.getChildren().add(prevButton);
paginationPane.getChildren().addAll(pageButtons);
paginationPane.getChildren().add(nextButton);
```

Së fundimi ia shtojmë butonat paginationPane që e morem si parametër

Në mënyrë tërësisht analoge shpjegohet edhe funksioni buttonRender me dy parametra.

## CarListController

CarListController është klasa e cila i jep funksionalitet pamjes së sipërme. Kjo klasë e përdorë klasën PageBtnComponent për t'i vendosur në pjesën e poshtme butonat e nevojshëm për shfaqjen e veturave dhe për t'iu vendosur atyre funksionalitetin që u shtjellua më sipër.

```
paginationComponent = new PageBtnComponent(carCount(), PAGE_SIZE);
paginationComponent.render(btnPane, (Object page) -> {
    try {
        showCars((Integer) page);
    } catch (Exception ex) {
        ErrorPopupComponent.show(ex);
    }
}, queryString: null);
```

Funksioni më i rëndësishëm është funksioni i mbingarkuar showCars(int page), showCars(String queryString).

```
private void showCars(int page) throws Exception {
    carsPane.getChildren().clear();
    List<Car> cars = CarRepo.getAll(PAGE_SIZE, page);
    for (Car car : cars) {
        Node carCard = new CarCardComponent().getContent(car, e->showCar(car), e->removeCar(car), e->rentCar(car));
        carsPane.getChildren().add(carCard);
    }
}

private void showCars(String queryString) throws Exception {
    carsPane.getChildren().clear();
    List<Car> cars = CarRepo.getSelectedCars(queryString);
    for (Car car : cars) {
        Node carCard = new CarCardComponent().getContent(car, e->showCar(car), e->removeCar(car), e->rentCar(car));
        carsPane.getChildren().add(carCard);
    }
}
```

Në rastin e parë showCars merr si parametër një int që reprezenton numrin e faqes. I fshin veturat aktuale nga pane-i, pastaj përmes funksionit getAll i cili kthen një List merr nga databaza veturat për faqen që representohet nga parametri page. Pastaj me një unazë for për secilën veturë që gjendet në listën cars vendos veturën në kartën e tij përkatëse (duke krijuar instancë të CarCardComponent, klasë e cila do të shtjellohet në një seksion tjetër). Së fundmi në carsPane-in e zbrazët e shton këtë kartë.

Funksioni tjetër që merr si parametër një String funksionon shumë ngjashëm vetëm se për të marr një listë me vetura nga databaza përdorë funksionin getSelectedCars(queryString) i cili kthen një listë të veturave që i përgjigjen queryt që gjendet në queryString.

```
@FXML
private void onSelectBrandSetAction(ActionEvent ev) throws Exception{
    String brand = brands.getValue();
    String type = types.getValue();
    String radBtnString = rented.isSelected() ? "rented" : (forRent.isSelected() ? "forRent" : "showAll" );
    String queryString = CarRepo.getQuery(brand,type,radBtnString,PAGE_SIZE, page: 0);
    String newQString = queryString.replaceAll( regex: "LIMIT \\d+ OFFSET \\d+", replacement: "");

    paginationComponent = new PageBtnComponent(CarRepo.getAffectedRows(newQString), PAGE_SIZE);
    paginationComponent.render(btnPane, qs -> {
        try {
            showCars((String)qs);
        }catch (Exception ex){
            ErrorPopupComponent.show(ex);
        }
    },queryString);
    System.out.println("erdhh");
    showCars(queryString);
}
```

Funksioni onSelectBrandSetAction ia shton funksionalitetin combobox it për zgjedhjen e brand it. Më konkretisht ky funksion thirret kur zgjedhet një vlerë e re në comboboxin për zgjedhjen e brand-eve. Ky funksion merr vlerat e te dy comboboxave dhe përmes operatorit ternar e shikon se cili prej radiobutonave është zgjedhur dhe nga kjo e krijon një String duke ia jep një vlerë që reprezenton butonin e zgjedhur. Përmes funksionit getQuery duke ia dërguar informatat e dhëna e merr queryn e përshtatshëm që do të përdoret për nxjerrje të të dhënave nga DB. Duke e larguar pjesën e cekur në regex formon një queryString të ri numri i rekordeve të së cilit është i përshtatshëm për krijimin e një instance të PageBtnComponent. Kështu e krijon një paginationComponent duke ia dërguar numrin e tërësishëm të veturave të kthyer nga ky query. Përmes metodës render ku ia dërgojmë pane in që i mban butonat, një instancë të një klase që implementon PagonationHandler (përmes lambda expression krijohet), dhe queryStringun. Në fund thirret funksioni showCars (queryString) për ti shfaqur veturat që reprezentohen nga ky query. Njejtë funksionojnë edhe metodat onSelectTypeSetAcion, onSelectForRentSetAcion, onSelectRentedSetAcion dhe onSelectShowAllSetAcion.

# 1. Insertimi dhe editimi i makinave

## 1.1. Pamja për insert/edit

The screenshot displays a web application titled "Interactive system for booking rental cars" with a sub-header "Booking of rental cars". The interface includes a navigation bar with "File", "Insert", and "Help" menus. A sidebar on the left contains links for "Cars", "Expired", and "Logout". The main content area, titled "Cars display", features a form for adding or editing a car. The form fields are as follows:

Field	Value
Handler	-1
Publisher	1
Manufacture	0
Model	
Image	
Price per day	0.0
Transmission	Auto
Date Inserted	2021-07-13 11:11:13
Date Updated	2021-07-13 11:11:13
Door number	0
Seat number	0
Type	SUV
Speed limit	0.0
Avg fuel	0.0

At the bottom right of the form are "Save" and "Cancel" buttons. A footer at the bottom left shows a "Language" dropdown menu, and a status bar at the bottom indicates "User Erlis logged in at 2021-07-13 11:11:13".

Kur klikojmë butonin Insert në navBar ose butonin Edit për ndonjë makinë atëherë hapet forma për insertimin përkatësisht editimin e një makine.

Fusha Handler paraqet Id qe identifikon një makinë në mënyrë unike. është disabled sepse kalkulohet në mënyrë automatike. Te fusha Image mund të zgjedhet fotografia e makinës nga kompjuteri. Fushat Date Inserted dhe Date Updated gjithashtu kalkulohen në mënyrë automatike. Në fushat tjera mund t'i shkruajmë vlerat që dëshirojmë.

Nëse klikojmë butonin Cancel të dhënat nuk ruhen dhe drejtohem te faqja cars-list.fxml. Nëse klikojmë butonin Save të dhënat do të krijojnë një makinë të re në databazë në rast se jemi duke bërë insert ose do të ndërrojnë fushat e makinës përkatëse nëse bëjmë update.

## 1.2. Kodi për insert/edit

Për të bërë më të lehtë marrjen e të dhënave nga forma për vendosjen e tyre në databazë përdorim Two way Binding, veti që mundëson që dy variabla të tipit Property të kenë vlera të sinkronizuara(kur ndërrohet njëra ndërrohet edhe tjetra).

Two way Binding funksionon vetëm me variabla të tipit Property prandaj duhet që variablat e modelit Car ti mbështjellim në variabla të tipit Property. Për kod më të pastër krijojmë klasë të re CarViewModel që përmban këto variabla dhe funksionet getter dhe setter.

```
public class CarViewModel {  
    private IntegerProperty id;  
    private IntegerProperty publisher;  
    private IntegerProperty manufacture;  
    private StringProperty model;  
    private DoubleProperty price_per_day;  
    private DoubleProperty avg_fuel_km;  
    private StringProperty transmission;  
    private DoubleProperty speed_limit;  
    private StringProperty type;  
    private IntegerProperty seat_num;  
    private IntegerProperty door_num;  
    private StringProperty inserted_at;  
    private StringProperty updated_at;  
    private StringProperty car_img;  
  
    public CarViewModel() {...}  
  
    public CarViewModel(Car model) {...}  
  
    public IntegerProperty idProperty() { return id; }  
    public int getId() { return id.getValue(); }  
    public void setId(int value) { id.setValue(value); }
```

Kjo klasë ka edhe një metodë që kthen objekt të tipit Car nga vlerat e variablave Property.

```
public Car getModel() {  
    return new Car(getId(), getPublisher(), getManufacture(), getModel(), getPrice_per_day(), getAvg_fuel_km(),  
        Transmission.valueOf(getTransmission()), getSpeed_limit(), Type.valueOf(getType()),  
        getSeat_num(), getDoor_num(), getInserted_at(), getUpdated_at(), getCar_img());  
}
```

Për të vendosur a duhet të bëjmë fushat e formës disabled (kur përdoret vetëm për shiqimin e të dhënave) apo enabled (kur përdoret për insert ose edit) e përdorim funksionin `setEditable` që duhet të thirret nga faqja paraprake me parametër `editable` të vendosur varësisht nga qëllimi i hapjes së formës.

```
public void setEditable(boolean editable) {
    this.isEditable = editable;

    publisherField.setDisable(!isEditable);
    manufactureField.setDisable(!isEditable);
    modelField.setDisable(!isEditable);
    priceField.setDisable(!isEditable);
    transmissionField.setDisable(!isEditable);
    seatnumField.setDisable(!isEditable);
    doornumField.setDisable(!isEditable);
    typeField.setDisable(!isEditable);
    speedlimitField.setDisable(!isEditable);
    avgfuelField.setDisable(!isEditable);

    saveButton.setDisable(!editable);

    if (isEditable) {
        imgField.setOnMouseClicked(e -> this.onImageClick());
    } else {
        imgField.setOnMouseClicked(null);
    }
}
```

Funksioni `setModel` përdoret për caktimin e vlerave të fushave të formës kur kjo formë hapet.

```
public void setModel(Car model) {
    viewModel = new CarViewModel(model);

    idField.setText(Integer.toString(viewModel.getId()));
    publisherField.setText(Integer.toString(viewModel.getPublisher()));
    manufactureField.setText(Integer.toString(viewModel.getManufacture()));
    modelField.setText(viewModel.getModel());
}
```

Gjithashtu në këtë funksion bëjmë Two way Binding mes fushave të formës dhe fushave të `viewModel`it.



```

modelField.textProperty().bindBidirectional(viewModel.modelProperty());
priceField.textProperty().addListener((ov, oldVal, newVal) -> {
    if (!Utils.isEmpty(newVal)) {
        try {
            viewModel.setPrice_per_day(Double.parseDouble(newVal));
        } catch (Exception e) {
        }
    }
});

```

Me klikimin e fushës së fotografisë thirret funksioni `onImageClick` që hap një dritare për zgjedhjen e fotografisë. Ky funksion përdor klasën `FileHelper`. Fotografinë e zgjedhur e kopjon në një folder të caktuar për fotografi dhe vendos këtë path si vlerë të fushës `carImg` të `viewModel`it dhe tek fusha e formës `imgField`.

```

private void onImageClick() {
    try {
        Stage primaryStage = (Stage) idField.getScene().getWindow();
        File srcFile = fileChooser.showOpenDialog(primaryStage);

        if (srcFile != null) {
            FileHelper fh = FileHelper.get();
            String filename = new Date().getTime() + (int) (Math.random() * 100) + "." + fh.fileExt(srcFile);
            File destFile = new File( pathname: fh.getImageDir() + "/" + filename);
            fh.copyFile(srcFile, destFile);

            Image image = new Image(destFile.toURI().toString());
            imgField.setImage(image);
            viewModel.setCar_img("resources/images/Cars/"+filename);
        }
    } catch (Exception e) {
        ErrorPopupComponent.show(e);
    }
}

```

Në klikim të butonit `Cancel` thirret funksioni `onCancelButtonClick` që na ridrejton te faqja `carsListView`.

Në klikim të butonit `Save`, nëse fusha e `Id` është `-1` e dijme që kemi insert dhe thirrjm funksionin `create` nga klasa `CarRepo` përndryshe kemi edit dhe thirrjm funksionin `update` poashtu nga klasa `CarRepo`. Të dy këto funksione marrin si parametër një objekt të tipit `Car` që kthehet nga funksioni `getModel` i klasës `CarViewModel`. Pas kësaj kalojmë te faqja `CarListView`.



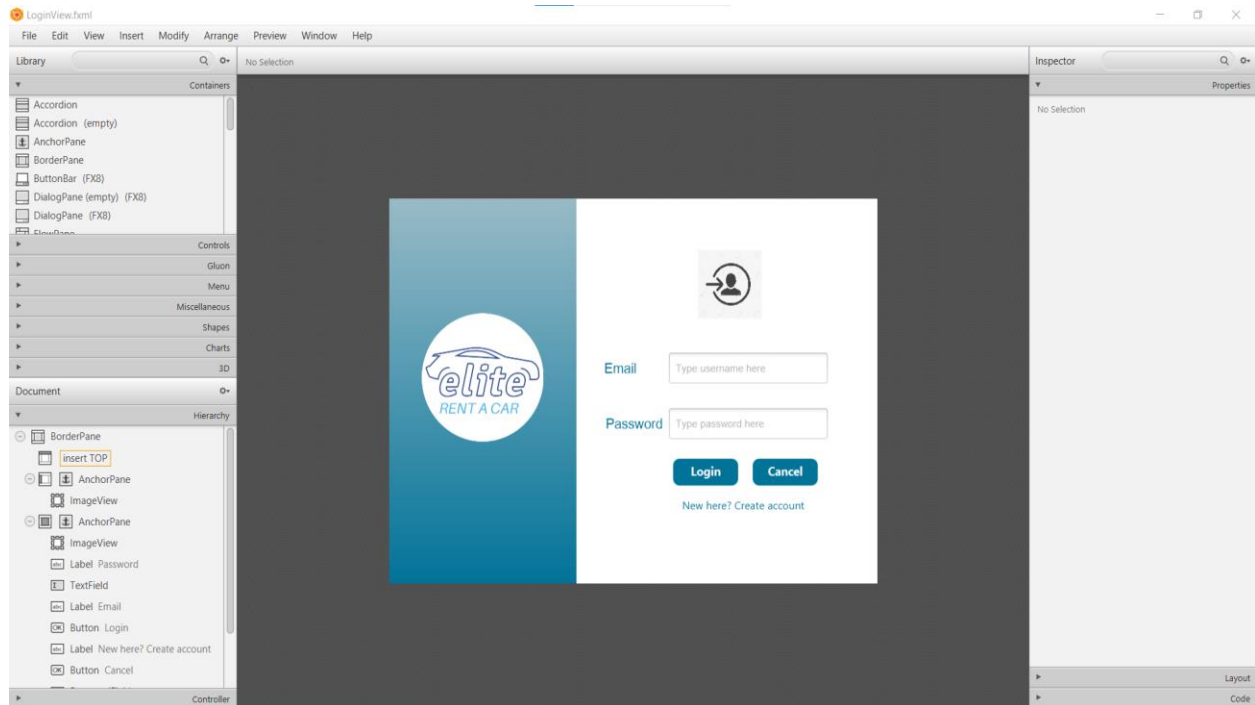
```
@FXML
private void onCancelButtonClick(ActionEvent event) {
    try {
        parentController.setView(MainScreenController.CARS_LIST_VIEW);
    } catch (Exception e) {
        ErrorPopupComponent.show(e);
    }
}
```

```
@FXML
private void onSaveButtonClick(ActionEvent event) {
    try {
        if (viewModel.getId() > 0) {
            CarRepo.update(viewModel.getModel());
        } else {
            CarRepo.create(viewModel.getModel());
        }
        parentController.setView(MainScreenController.CARS_LIST_VIEW);
    } catch (Exception e) {
        ErrorPopupComponent.show(e);
    }
}
```

## 2.Login/Register

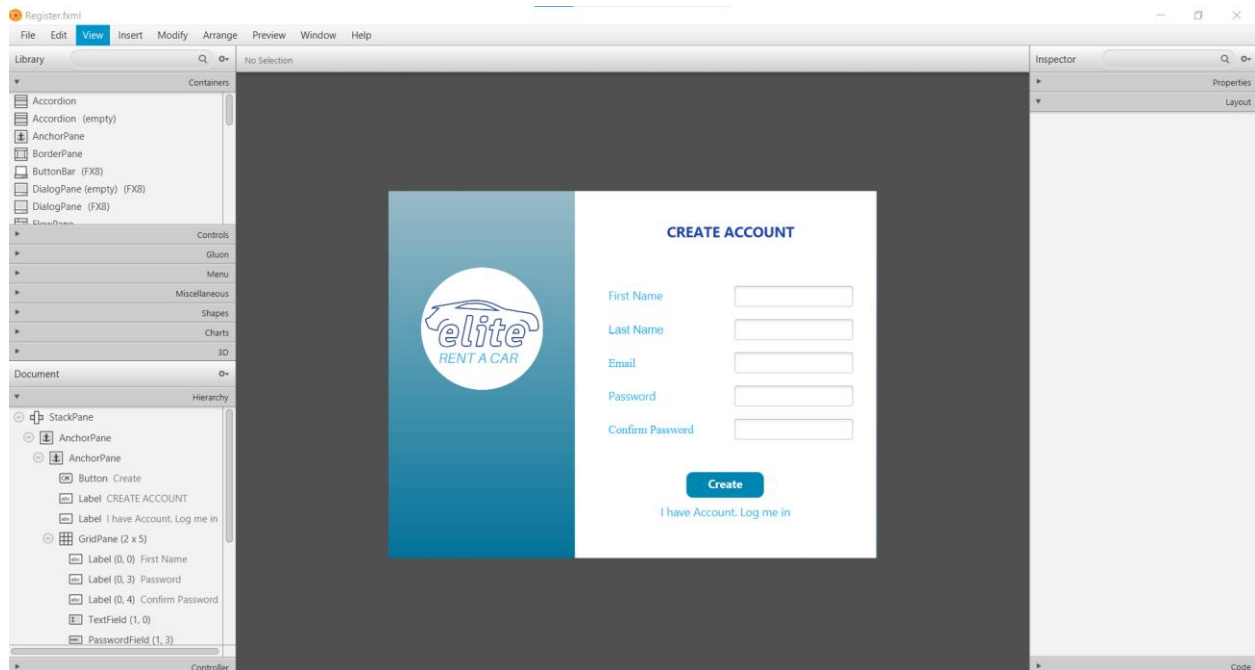
### 2.1.Login /Register views

Me anë të Scene Builder e kemi dizajnuar Login dhe Register view, duke përdorur panela të ndryshme dhe tools të tjera që janë nevojitur për ta paraqitur pamjen e dëshiruar.



Tek Login View kemi përdorur BorderPane për organizimin e faqes, ku në pjesën e majtë kemi vendosur një AnchorPane në të cilin si background është përdorur linear gradient për vendosjen e ngjyrës, dhe brenda saj kemi insertuar një foto. Ndërsa në qendër është vendosur një AnchorPane tjetër i cili përmban imazhe, labella, textfields, password dhe butona.

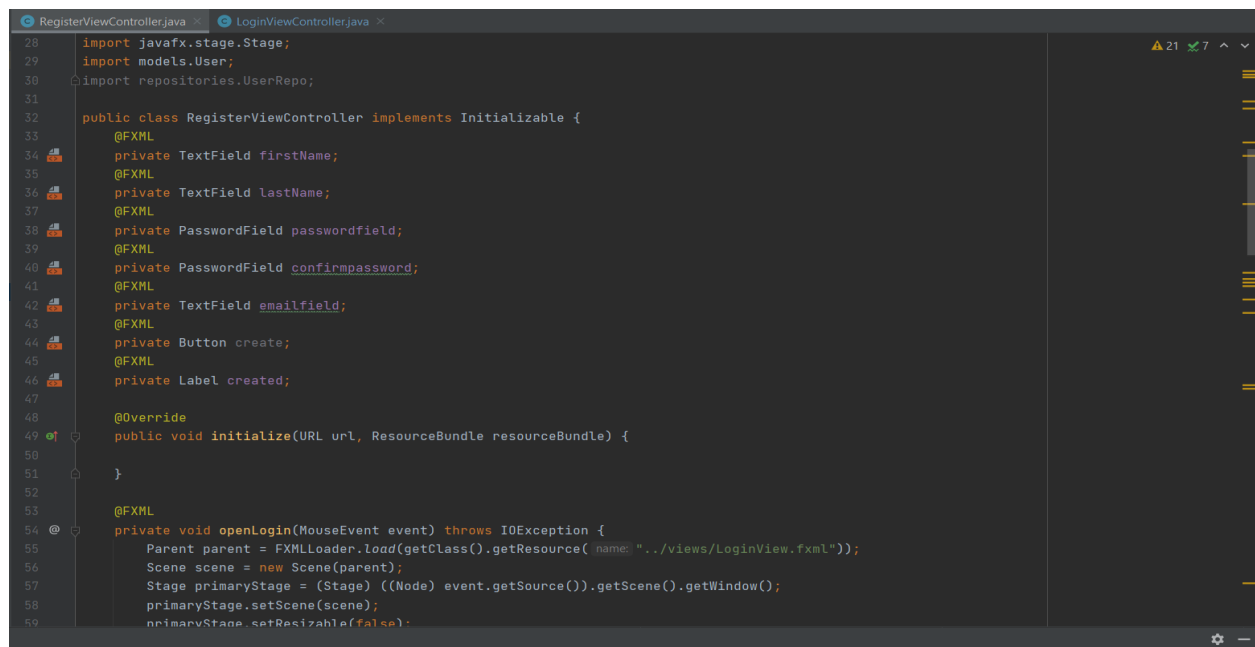
Pothuajse secilës prej tyre i kemi vendosur nga një id, në mënyrë që pastaj me anë të kontrollerave ta bëjmë faqen sa më interactive.



Dritarja e register view është e dizajnuar me StackPane ku brenda saj ndodhen dy AnchorPane, brenda AnchorPane të parë është insertuar një imazh, kurse brenda AnchorPane të dytë është vendosur një Grid Pane me anë të cilit kemi dizajnuar formën për ti marrë të dhënat prej user-it.

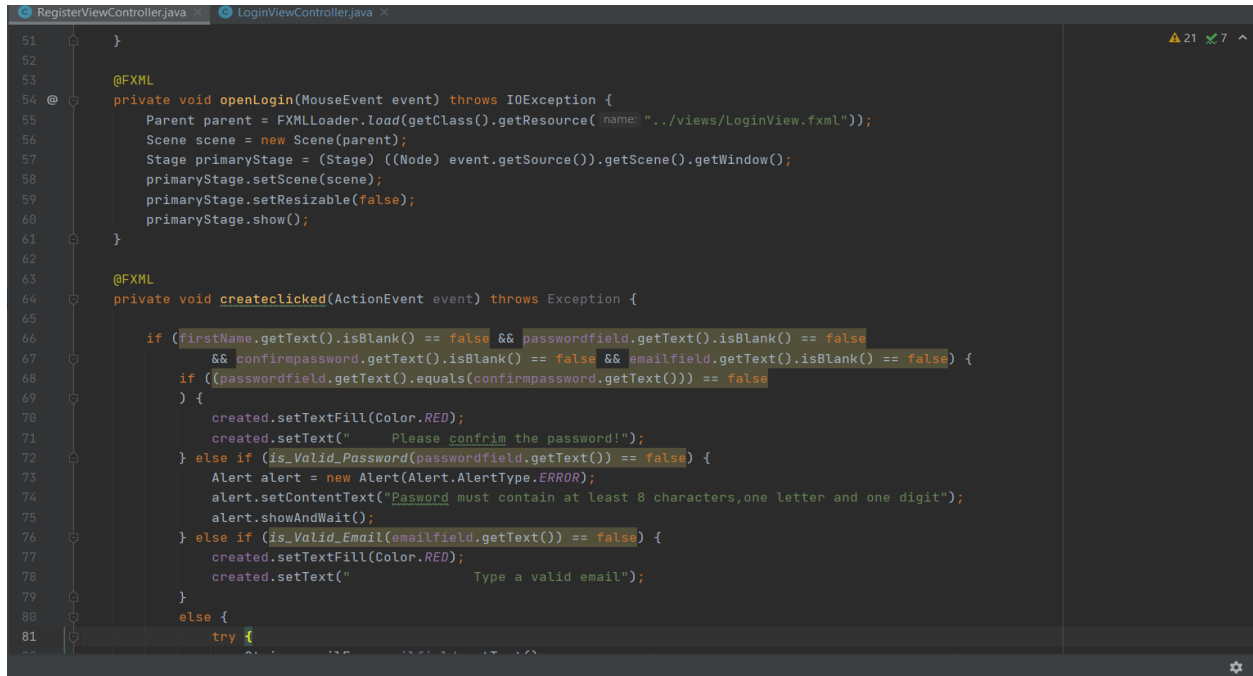
Tek pjesa e register, sikurse te pjesa e login ju kemi vendosur id elementeve.

## 2.2.Login/Register controllers



Lidhja e elemnteve të fxml për të manipuluar me to.

Në momentin e klikimit me anë të mausit mbi elementin i cili ka id-në open login hapet një dritare tjetër “LoginView.fxml”.



```
51 }
52
53 @FXML
54 private void openLogin(MouseEvent event) throws IOException {
55     Parent parent = FXMLLoader.load(getClass().getResource("../views/LoginView.fxml"));
56     Scene scene = new Scene(parent);
57     Stage primaryStage = (Stage) ((Node) event.getSource()).getScene().getWindow();
58     primaryStage.setScene(scene);
59     primaryStage.setResizable(false);
60     primaryStage.show();
61 }
62
63 @FXML
64 private void createClicked(ActionEvent event) throws Exception {
65
66     if (firstName.getText().isBlank() == false && passwordfield.getText().isBlank() == false
67         && confirmPassword.getText().isBlank() == false && emailfield.getText().isBlank() == false) {
68         if ((passwordfield.getText().equals(confirmpassword.getText())) == false) {
69             {
70                 created.setTextFill(Color.RED);
71                 created.setText("Please confirm the password!");
72             } else if (is_Valid_Password(passwordfield.getText()) == false) {
73                 Alert alert = new Alert(Alert.AlertType.ERROR);
74                 alert.setContentText("Pasword must contain at least 8 characters,one letter and one digit");
75                 alert.showAndWait();
76             } else if (is_Valid_Email(emailfield.getText()) == false) {
77                 created.setTextFill(Color.RED);
78                 created.setText("Type a valid email");
79             }
80         } else {
81             try {
```

Në klikimin e butonit që ka id-në createclicked, kemi kufizuar në atë mënyrë që të mos lejojë fusha të zbrazëta, validimin e confirmpassword në mënyrë që të jetë i njëjtë me password-in, gjithashtu kemi validuar passwordin dhe email-in me anë të dy funksioneve që e kufizojnë insertimin e të dhënave.

```

81 try {
82     String emailF = emailfield.getText();
83     String passwordF = passwordfield.getText();
84     String firstName = firstName.getText();
85     String lastName = lastName.getText();
86     User user = new User(firstName, lastName, emailF, passwordF);
87
88
89     if (Authenticate.register(user) != null) {
90
91         created.setTextFill(Color.GREEN);
92         created.setText("Your account has been created successfully");
93
94     } else {
95         String errStr = "";
96         ArrayList<Notification> errors = Authenticate.getNotifications();
97         for (int i = 0; i < errors.size(); i++) {
98             errStr += errors.get(i).getMsg();
99         }
100         Authenticate.clearNotifications();
101         created.setTextFill(Color.RED);
102         created.setText(errStr);
103     }
104
105 } catch (Exception ex) {
106     ex.printStackTrace();
107 }
108
109 } else {
110     Alert alert = new Alert(Alert.AlertType.ERROR);
111

```

```

}

@
public static boolean is_Valid_Password(String password) {

    if (password.length() < 8) return false;

    int charCount = 0;
    int numCount = 0;
    for (int i = 0; i < password.length(); i++) {

        char ch = password.charAt(i);

        if (is_Numeric(ch)) numCount++;
        else if (is_Letter(ch)) charCount++;
        else return false;

    }

    return (charCount >= 1 && numCount >= 1);

}

public static boolean is_Letter(char ch) {
    ch = Character.toUpperCase(ch);
    return (ch >= 'A' && ch <= 'Z');
}

public static boolean is_Numeric(char ch) {

    return (ch >= '0' && ch <= '9');

}

}

```

Funksioni i validimit të email merr një string si hyrje. Ky funksion nuk lejon që password të jetë me më pak se 8 karaktere, gjithashtu shikon nëse ka të paktën një shkronjë dhe një num, në qoftë se nuk plotësohen këto kushte kthen false.

```

137
138 public static boolean is_Letter(char ch) {
139     ch = Character.toUpperCase(ch);
140     return (ch >= 'A' && ch <= 'Z');
141 }
142
143
144 public static boolean is_Numeric(char ch) {
145
146     return (ch >= '0' && ch <= '9');
147 }
148
149 public static boolean is_Valid_Email(String email) {
150     boolean rez;
151     String regex = "[a-zA-Z0-9_!#$%&'*/=?~^.-]+@[a-zA-Z0-9.-]+$";
152     //Compile regular expression to get the pattern
153     Pattern pattern = Pattern.compile(regex);
154     Matcher matcher = pattern.matcher(email);
155     if (matcher.matches() == true) {
156         rez = true;
157     } else
158         rez = false;
159     return rez;
160 }
161
162 }

```

Funksioni is\_Letter dhe is\_Numeric thirren te is\_Valid\_Password, për ta shikuar nëse karakteri është numer ose shkronjë.

Me anë të is\_Valid\_Email me anë të një regex e kemi kufizuar se çfarë karaktere të pranojë në email field.

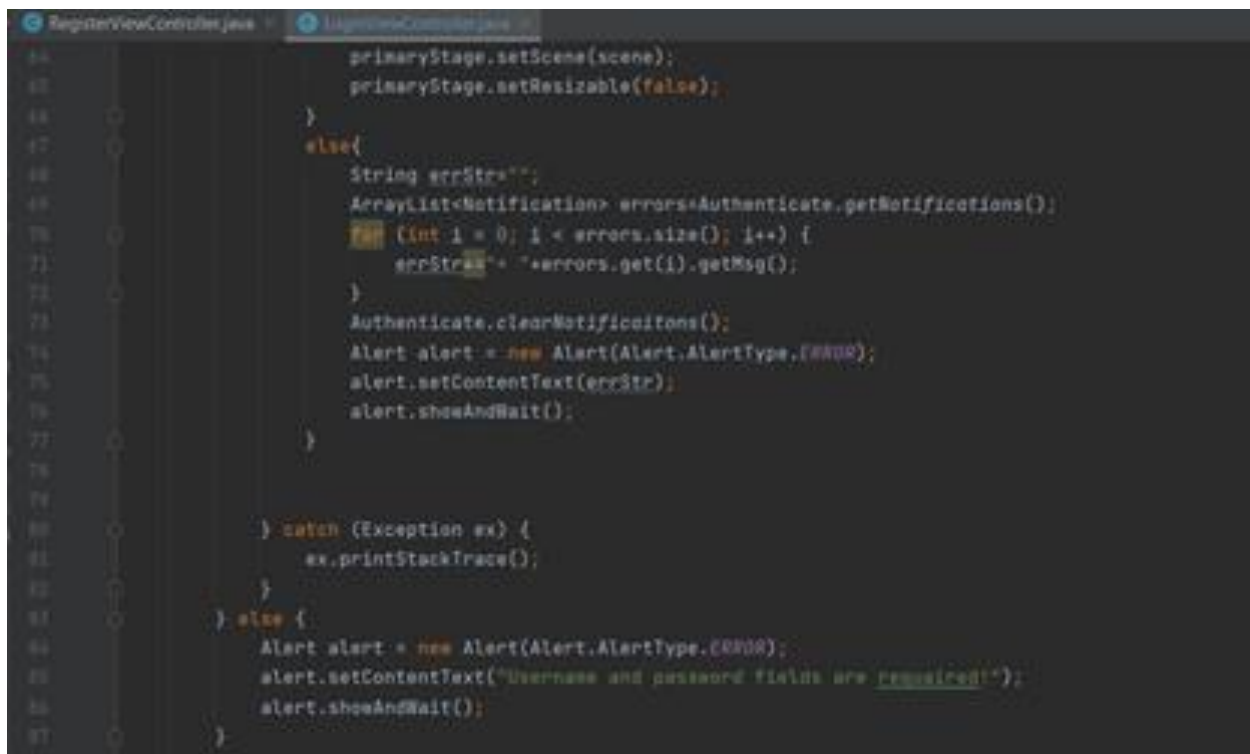
```

17 public void initialize(URL url, ResourceBundle resourceBundle) {
18
19 }
20
21 @FXML
22 private void loginClicked (ActionEvent event) throws Exception {
23     if (username.getText().isBlank()==false && password.getText().isBlank()==false) {
24         try {
25             User user = null;
26             String emailF = username.getText();
27             String passwordF = password.getText();
28             user=Authenticate.login(emailF,passwordF);
29             if(user!=null)
30             {
31                 SessionManager.employer = user;
32                 SessionManager.lastLogin = new Date();
33
34
35                 FXMLLoader loader = new FXMLLoader();
36                 loader.setLocation(getClass().getResource( "name ../../views/main-screen.fxml"));
37
38                 Parent parent = loader.load();
39                 MainScreenController controller = loader.getController();
40                 controller.setView(MainScreenController.CARS_LIST_VIEW);
41
42                 Stage primaryStage = (Stage) ((Node) event.getSource()).getScene().getWindow();
43                 Scene scene = new Scene(parent);
44                 primaryStage.setScene(scene);
45                 primaryStage.setResizable(false);
46             }
47         }
48         else{
49             String errStrg="";

```

Ashtu siç mund të shihet LoginViewController i cili përdoret për kontrollin e elementeve të file LoginView.fxml dhe që gjendet në package ku ndodhen Controllers të tjerë përmban

metodën loginClicked e cila ekzekutohet kur klikohet butoni Log in. Fillimisht shikohet se a janë blank fushat username dhe password ashtu që të mund të vazhdohet procesi i login vetëm nëse janë të mbushura me të dhënat e duhura. Në rast se nuk janë blank, siç mund të shihet nga kodi, vazhdohet më tutje ku variablat emailF dhe passwordF marrin të dhënat që user-i i shkruan në fushat përkatëse. Pastaj thirret metoda statike login e klasës Authenticate ku i dërgohen si parametra variablat e lartëpërmendura. Në këtë mënyrë shikojmë nëse në databazë ekziston një llogari me email-in dhe password-in që shfrytëzuesi ka dhënë. Në rast se ekziston atëherë përmes klasës SessionManager ruhet gjendja si ‘i kyçur’ në të gjitha faqet e tjera dhe pasi që user-i të kyçet kalon në screen që është specifikuar nga path në rreshtin 56.



```
14         primaryStage.setScene(scene);
15         primaryStage.setResizable(false);
16     }
17     else{
18         String errStr="";
19         ArrayList<Notification> errors=Authenticate.getNotifications();
20         for (int i = 0; i < errors.size(); i++) {
21             errStr+= " "+errors.get(i).getMsg();
22         }
23         Authenticate.clearNotifications();
24         Alert alert = new Alert(Alert.AlertType.ERROR);
25         alert.setContentText(errStr);
26         alert.showAndWait();
27     }
28 }
29
30 } catch (Exception ex) {
31     ex.printStackTrace();
32 }
33
34 } else {
35     Alert alert = new Alert(Alert.AlertType.ERROR);
36     alert.setContentText("Username and password fields are required!");
37     alert.showAndWait();
38 }
```

Kodi i cili ndodhet pas else-clause të parë do të ekzekutohej nëse në databazë nuk do të gjendej asnjë rekord që përmban të dhënat që user-i i ka dërguar në fushën e email-it dhe password-it. Sic mund të shihet do të thirrej metoda statike getNotification nga klasa Authenticate në mënyrë që të vendoset një lajmërim për shfrytëzuesin se cfarë ka gabuar. Ndërsa else-clause i dytë do të ekzekutohej nëse fushat e email dhe password do të lireshin të zbrazëta; do të shfaqej një alert me mesazhin “Username and password fields are required!”.

```

107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Cancelclicked metoda ekzekutohet nëse user-i klikon në butonin X dhe tenton ta mbyll aplikacionin. Fillimisht user-it do t'i shfaqet një Alert me mesazhin “Are you sure you want to leave?” dhe në rast se në atë Alert klikon OK atëherë aplikacioni do të mbyllet dhe ekzekutimi do të terminohet.

Createaccountclicked metoda lidhet me labelën që përmban tekstin “New here? Create account”. Sic mund ta shohim në rast se klikohet në këtë labelë atëherë do të shfaqet file i specifikuar në path në rreshtin 111. Ndërsa metoda loadLangTexts lidhet me dygjuhësinë e cila gjithashtu është implementuar në aplikacion.



## 3. GJUHËSIA

Internacionalizimi i një aplikacioni kërkon përdorimin e dy klasave dhe një enumeracioni;

1. Klasa **ResourceBundle**
2. Klasa **Locale**
3. Enumeracioni **LangEnum**

**Klasa ResourceBundle** lejon aplikacionin të përdor/shton të dhëna nga local files. Për këtë arsye është krijuar package bundles me dy .properties files të quajtura LangBundle\_al.properties dhe LangBundle\_en.properties për shkak se gjuhët e përdorura kanë qenë gjuha shqipe dhe ajo angleze. Këto dy files përmbajnë përkthimet e shkruajtura në mënyrë manuale të faqeve të aplikacionit, duke ndjekur formatin label=përkthim. P.sh.:

```
login_username=Email  
login_password=Fjalekalimi  
login_button_login=Kycu  
login_button_register=Regjistrohu
```

**Klasa Locale** me metodën e definuar në file-in ‘SessionManager’ tek package Utils, identifikon objektet që dallojnë për kah vendndodhja gjeografike. Në rastin tonë ‘al’ shërben për Albania dhe ‘en’ për United States.

```
public static Locale getLocale() {  
    LangEnum lang = AppConfig.get().getLanguage();  
    return lang == LangEnum.EN ? new Locale( language: "en", country: "US") :  
        new Locale( language: "al", country: "AL");  
}
```

**Enumeracioni LangEnum** përmban dy elemente:

```
public enum LangEnum {  
    EN, AL  
}
```

dhe përdoret në setter dhe getter të file-it ‘AppConfig’:

```

public LangEnum getLanguage() {
    LangEnum lang = props.getProperty( key: "lang", defaultValue: "en").equals("en") ? LangEnum.EN : LangEnum.AL;
    return lang;
}

public void setLanguage(LangEnum lang) throws Exception {
    URI confPath = getClass().getResource( name: "../resources/config.properties").toURI();
    String langStr = lang == LangEnum.EN ? "en" : "al";
    props.setProperty("lang", langStr);
    props.store(new FileOutputStream(new File(confPath)), comments: "");
}

```

ashtu që varësisht nga zgjedhja e user-it, kodi të jetë në gjendje të ofrojë (dy) gjuhësinë për aplikacion.

Në pjesën e mësipërme të kodit janë vendosur bazat e gjuhësisë. Implementimi është bërë në controllers përkatës të faqeve të aplikacionit.

Në LoginViewController fillimisht është përdorur @FXML shënimi për të etiketuar fushat dhe metodat jopublike të controllers:

```

@FXML
private TextField username;
@FXML
private PasswordField password;

```

Më pas është vendosur label që e përmban përkthimin e TextField dhe PasswordField të definuar në .properties, ekzekutimi i të cilit është bërë brenda një try-catch.

```

@Override
public void loadLangTexts(ResourceBundle langBundle) {
    username.setPromptText(langBundle.getString( key: "login_username"));
    password.setPromptText(langBundle.getString( key: "login_password"));
    try {
        String buttonText = langBundle.getString(hasUsers() ?
            "login_button_login" : "login_button_register");
    } catch (Exception ex) {
        ErrorPopupComponent.show(ex);
    }
}

```

Hapat e njëjtë janë ndjekur edhe tek faqet tjera, psh në MainScreenController:

```

@FXML
private Button navCarsButton;
@FXML
private Button navExpiredButton;
@FXML
private Button navLogoutButton;
@FXML
private VBox contentPage;
@FXML
private Label statusLabel;
@FXML
private Label sectionLabel;
@FXML
private CheckMenuItem enCheckMenuItem;
@FXML
private CheckMenuItem alCheckMenuItem;

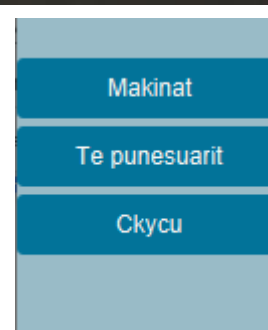
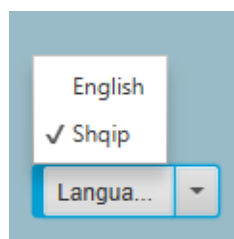
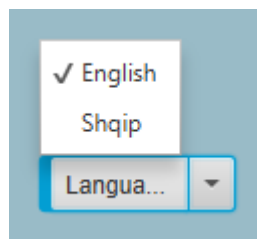
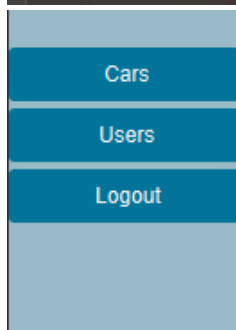
```

```

@Override
public void loadLangTexts(ResourceBundle langBundle) {
    String navCarsTxt = langBundle.getString( key: "main_nav_cars");
    String navExpiredTxt = langBundle.getString( key: "main_nav_employers");
    String navLogoutTxt = langBundle.getString( key: "main_nav_logout");
    String statusLabelTxt = langBundle.getString( key: "main_status_label");

    String employer = SessionManager.employer.getEmail();
    String loginTime = DateHelper.toSqlFormat(SessionManager.lastLogin);
    statusLabel.setText(String.format(statusLabelTxt, employer, loginTime));
    navCarsButton.setText(navCarsTxt);
    navExpiredButton.setText(navExpiredTxt);
    navLogoutButton.setText(navLogoutTxt);
}

```



Pjesa e ndryshimit të gjuhëve varësisht nga zgjedhja e user-it, është implementuar këtu:

```
private void updateLanguage() {  
    try {  
        LangEnum lang = enCheckMenuItem.isSelected() ? LangEnum.EN : LangEnum.AL;  
        AppConfig conf = AppConfig.get();  
        conf.setLanguage(lang);  
        ResourceBundle bundle = getLangBundle();  
        loadLangTexts(bundle);  
    } catch (Exception ex) {  
        ErrorPopupComponent.show(ex);  
    }  
}
```

```
@FXML  
public void onAlMenuItemClick(ActionEvent ev) {  
    enCheckMenuItem.setSelected(false);  
    alCheckMenuItem.setSelected(true);  
    updateLanguage();  
}  
  
@FXML  
public void onEnMenuItemClick(ActionEvent ev) {  
    enCheckMenuItem.setSelected(true);  
    alCheckMenuItem.setSelected(false);  
    updateLanguage();  
}
```

Në momentin që user-i zgjedh gjuhën angleze, selektohet enCheckMenuItem dhe thirret metoda updateLanguage().

# Përfundime

Kjo ishte ajo çka kemi punuar ne në projektin tonë. Edhe pse në kohë provimesh, ka qenë një nga projektet më interesante që kemi punuar deri tani. Në këtë seminar u munduam në pika mjaft të shkurta ta përmbledhim atë se çka kemi bërë gjatë javëve që kemi shpenzuar në këtë projekt. Përmes këtij projekti i kemi përdorur konceptet e mësuara në lëndën e KNK- së duke i përvëtsuar ato dukshëm më mirë, si dhe kemi parë si ndërtohen aplikacionet reale. Tani njohim me mire programimin e udhehequr nga ngjarjet dhe jemi në gjendje t'i përdorim në mënyrë efektive handlers.

Në përgjithësi jemi më të aftë në ndërtimin e aplikacioneve, dhe mendojmë se këto çka kemi mësuar gjatë projektit në KNK do të na shërbejnë shumë në profesionet tona të ardhshme.