Réservation de véhicule : partie 1 - les véhicules

Dans ce projet, nous nous proposons de développer une application pour gérer les réservations de véhicules. Il y a différents véhicules associés à leur immatriculation respective. Qu'un client peut réserver. L'application est réservée aux personnels de l'entreprise CONFORVOITE qui seraient au comptoir et qui accueillerait les clients pour récolter les informations personnelles afin d'aller jusqu'au bout de la réservation. Il s'agit d'une application d'entreprise et non d'une application grand public.

Dans cette première partie de ce projet, nous attaquons la gestion des véhicules.

Le rendu de ce projet pour cette première partie est décrit en vidéo au lien ci-dessous :

- Navigation et architecture : https://xonatis.academy/videos/javafx/1.mp4
- Gestion des véhicules : https://xonatis.academy/videos/javafx/2.mp4

Cette première partie de ce projet se décompose en 3 temps. Le premier temps est la mise en place de la navigation et l'architecture de l'application, le 2nd et la mise en place du `domain`, c'est-à-dire le modèle de données pour pouvoir stocker les réservations et les véhicules, et enfin, dans la 3e partie, nous mettrons en place les écrans de gestion des véhicules.

Au fur et à mesure de la réalisation de ce projet, merci de m'envoyer des captures d'écran en message privé, par discorde, avec bien sûr le titre de l'application qui sera « Airfrense de … » avec votre prénom :)

1. La navigation de l'application

Dans cette partie, nous allons mettre en place la navigation et l'architecture de l'application. Le rendu vidéo est disponible à l'adresse suivante :

Rendu vidéo : https://xonatis.academy/videos/javafx/1.mp4

Comme il y a beaucoup d'écran où ne souhaitons pas copier coller le menu sur tous les écrans. Nous souhaitons factoriser le code du menu et pouvoir l'inclure dans les différents écrans afin de coder le menu qu'une seule fois.

Pour ce faire, nous allons, nous allons développer 2 fichiers spécifiques au menu :

1. un contrôleur juste pour le menu pour gérer les clics
2. Une vue FXML juste pour le menu qu'on viendra inclure dans les autres écrans

```
public class MenuController {

    @FXML
    private void clickClose() {
        AirfrenseApplication.close();
    }

    @FXML
    private void clickDashboard() {
        AirfrenseApplication.navigateTo("dashboard");
    }
```

```
    // ...
}
```

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.*?>
<?import javafx.scene.layout.VBox?>
<VBox xmlns="http://javafx.com/javafx"
      xmlns:fx="http://javafx.com/fxml"
      fx:controller="com.example.demo.MenuController">

    <MenuBar>
        <Menu text="Fichier">
            <MenuItem text="Dashboard" onAction="#clickDashboard"/>
            <MenuItem text="Nouvelle réservation ..."
onAction="#clickBookingCreate"/>
            <SeparatorMenuItem/>
            <MenuItem text="S'identifier" onAction="#clickLogin"/>
            <SeparatorMenuItem/>
            <MenuItem text="Fermer" onAction="#clickClose" />
        </Menu>

        <!-- Autre menu -->
    </MenuBar>
</VBox>
```

Pour inclure le fichier FXML du menu dans les autres écrans, rien de plus simple ! Il suffit d'utiliser la directive `fx:include` comme dans l'exemple ci-dessous :

```
<!-- En-têtes habituelles (imports etc.) -->

<VBox spacing="20.0" xmlns:fx="http://javafx.com/fxml"
      xmlns="http://javafx.com/javafx"
      fx:controller="com.example.demo.HelloController">

    <fx:include source="menu.fxml" />

</VBox>
```

Par ailleurs, je vous demanderai de bien organiser vos écrans également, selon l'arborescence présentée dans l'image de l'annexe 1. Si vous vous posiez la question sur le nom à choisir dans le HashMap qui référence tous les écrans, vous pouvez utiliser la convention de nommage suivante :

```
screenByName.put("login",
FXMLLoader.load(AirfrenseApplication.class.getResource("login.fxml")));

screenByName.put("booking/create",
FXMLLoader.load(AirfrenseApplication.class.getResource("booking/create.fxml")));
```

```
screenByName.put("booking/delete",
FXMLLoader.load(AirfrenseApplication.class.getResource("booking/delete.fxml")));
```

2. Mise en place du domain

Dans cette étape, nous allons mettre en place le `domain`, tel que décrit par le diagramme de classe de l'annexe 2.

Ces 2 classes vont permettre de stocker les réservations ainsi que les différents véhicules qui sont répertoriés dans l'application.

Veuillez créer ces entités dans un nouveau package nommé `domain` dans le code source de l'application, comme dans l'image de l'annexe 3.

Avez-vous remarqué le polymorphisme de méthode ? :)

3. Mettre en place de la gestion des vehicules

Il s'agit donc cette dernière section de cette première partie du projet, de mettre en place la gestion des véhicules. Selon le rendu vidéo ci-dessous :

Rendu vidéo : https://xonatis.academy/videos/javafx/1.mp4

Je vous demanderai de mettre en place une `ArrayList` de véhicules et non pas un `ArrayList` de `String`. Cela permettra au contrôleur de travailler avec le modèle avant l'affichage, géré exlusivement par le `ObservableList`, comme dans le code ci-dessous.

Vous devez ajouter des véhicules par défaut pour que lorsque l'écran s'affiche, il y a déjà des véhicules disposés dans la liste.

```
public class VehicleController {
    @FXML
    private ListView<String> listView;

    private ArrayList<Vehicle> dataList = new ArrayList<>();

    private ObservableList<String> observableList =
FXCollections.observableArrayList();

    public void initialize() {
        // Ajoutez les véhicules
    }
```

Vous pouvez créer un constructeur pour les véhicules, pour les créer plus facilement, mais ceci n'est pas obligatoire.

```
Vehicle vehicle = new Vehicle(name, license);
```

Quelques tips de développement Java

Vous aurez peut-être besoin des tips suivants pour développer du en Java les écrans qui vous sont demandées par le client.

Pour convertir un int en String :

```
int index = 5;
String.valueOf(index);
```

et pour convertir un String vers un int :

```
String value = "4";
Integer.parseInt(value);
```

Pour remplacer un élément dans un ObservableList :

```
observableList.set(index, value);
```

Au fur et à mesure de la réalisation de ce projet, merci de m'envoyer des captures d'écran en message privé, par discorde, avec bien sûr le titre de l'application qui sera « Airfrense de … » avec votre prénom :)