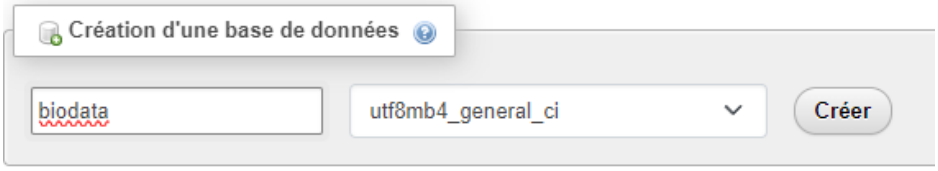
Compte rendue GSB

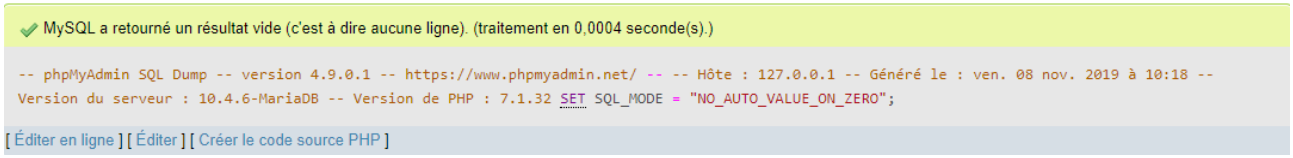
Création de la base de donnée :



Pour le scripte sql j'ai récupérer le scripte dans le répertoire TP 13 du drive :

Cependant j'ai fait un petit changement dans la modification table admin. Il y a un problème avec la syntaxe il manque le champs column

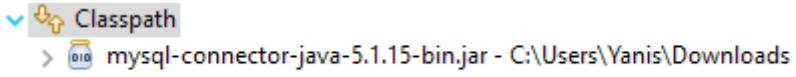
```
ALTER TABLE `admins`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;


 ALTER TABLE `admins` MODIFY COLUMN
 `id` INT AUTO_INCREMENT PRIMARY KEY, AUTO_INCREMENT=2;
```



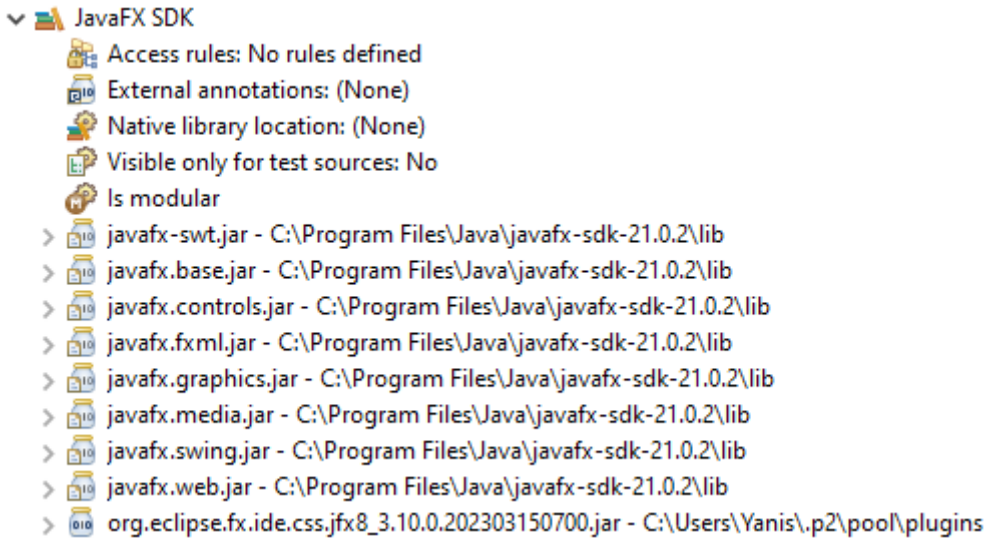
Création du projet :

1.Connexion

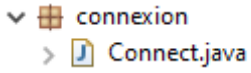
On commence par crée un projet javafx puis on ajoute le module de connexion dans Classpath :



Je pense qu'il y a pas besoin de rajouter les module javafx car la libraire JavaFX SDK possède déjà les modules :



Je crée ensuite la classe Connect dans le package connexion :



Le scripte :

```java
package connexion;

import com.mysql.jdbc.jdbc2.optional.MysqlDataSource;

public class Connect {
    static Connection con;

    public static Connection Connexion() {
        if (con == null) {
            MysqlDataSource data = new MysqlDataSource();
            data.setDatabaseName("biodata");
            data.setUser("root");
            data.setPassword("");
            try {
                con = data.getConnection();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
        return con;
    }
}
```

Ce scripte permet de connecter a la base de donnée à partir d'une méthode static qui permet de faire appelle à cette méthode sans instancier de classe avant.

Cette méthode permet donc de définir la base de donnée, l'utilisateur et le mot de passe se qui permet de ce connecter et renvoie la connexion.

2. Le model

Le model user permet de communiquer avec la base de donnée en définissant et en réprésentant les différents composant du en récupérant et en envoyant avec des getter et des setter les valeurs des composant du fichier fxml.

```
package model;

public class User {

    private String id;

    private String email;

    private String dob;

    private String gender;

    private String lastname;

    private String firstname;

    public String getId ()
    {
        return id;
    }

    public void setId (String id)
    {
        this.id =  id;
    }

    public String getEmail ()
    {
        return email;
    }

    public void setEmail (String email)
    {
        this.email =  email;
    }

    public String getDob ()
    {
        return dob;
    }

    public void setDob (String dob)
    {
        this.dob =  dob;
    }
```

```
    public String getGender ()
    {
        return gender;
    }

    public void setGender (String gender)
    {
        this.gender =  gender;
    }

    public String getLastname ()
    {
        return lastname;
    }

    public void setLastname (String lastname)
    {
        this.lastname =  lastname;
    }

    public String getFirstname ()
    {
        return firstname;
    }

    public void setFirstname (String firstname)
    {
        this.firstname =  firstname;
    }

    public String toString()
    {
        return "ClassPojo [id = "+id+",email="+email+",Date de naissance="+dob+
                ",Statut="+gender+",Nom ="+lastname+",Prénom="+firstname+"]";
    }


}
```

3. FXML

A partir de scenebuilder et du script sur le drive j’ai pue crée se visuel :

Voici le script :

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

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.PasswordField?>
<?import javafx.scene.control.Separator?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.Pane?>
<?import javafx.scene.text.Font?>

<AnchorPane id="AnchorPane" prefHeight="503.0" prefWidth="854.0" style="-fx-background-color: #fff;" xmlns="http://javafx.com/javafx/21" xmlns:fx="http://javafx.com/fxml/1" fx:controller="controller.LoginController">
    <children>
        <AnchorPane prefHeight="503.0" prefWidth="382.0" style="-fx-background-color: #223494;" stylesheets="css.fullpackstyling.css" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.topAnchor="0.0">
            <children>
                <ImageView fitHeight="106.0" fitWidth="194.0" layoutX="100.0" layoutY="146.0" pickOnBounds="true" preserveRatio="true">
                    <image>
                        <Image url="@../../images/logo_GSB.jpg" />
                    </image>
                </ImageView>
                <Label alignment="CENTER" layoutX="71.0" layoutY="262.0" prefHeight="23.0" prefWidth="223.0" text="GSB" textFill="WHITE">
                    <font>
                        <Font name="Segoe UI Bold" size="30.0" />
                    </font>
                </Label>
                <Label alignment="CENTER" layoutX="89.0" layoutY="295.0" prefHeight="23.0" prefWidth="205.0" text="BTS SIO 2ème année" textFill="WHITE">
                    <font>
                        <Font name="Segoe UI Bold Italic" size="12.0" />
                    </font>
                </Label>
            </children>
        </AnchorPane>
        <Pane layoutX="488.0" layoutY="100.0" AnchorPane.bottomAnchor="73.0" AnchorPane.topAnchor="100.0">
            <children>
                <TextField fx:id="txtUsername" layoutX="7.0" prefHeight="34.0" prefWidth="275.0" promptText="identifiant ou mail">
                    <font>
                        <Font name="System Bold" size="18.0" />
                    </font>
                </TextField>
                <PasswordField fx:id="txtPassword" layoutX="7.0" layoutY="61.0" prefHeight="34.0" prefWidth="275.0" promptText="mot de passe" stylesheets="css.fullpackstyling.css">
                    <font>
                        <Font name="System Bold Italic" size="18.0" />
                    </font>
                </PasswordField>
                <Button fx:id="btnSignin" layoutX="145.0" layoutY="125.0" mnemonicParsing="false" onMouseClicked="#handleButtonAction" prefHeight="34.0" prefWidth="137.0" stylesheets="css.fullpackstyling.css" text="se connecter">
                    <font>
                        <Font name="System Bold Italic" size="15.0" />
                    </font>
                </Button>
                <Label fx:id="btnForgot" alignment="CENTER" layoutX="8.0" layoutY="131.0" prefHeight="23.0" prefWidth="120.0" text="Mot de passe oublié" textFill="#223494">
                    <font>
                        <Font name="Segoe UI Bold" size="12.0" />
                    </font>
                </Label>
                <Button fx:id="btnFB" layoutX="4.0" layoutY="215.0" mnemonicParsing="false" prefHeight="34.0" prefWidth="275.0" style="-fx-backg>round-color: #223494;" text="Se connecter avec Facebook" textFill="WHITE">
                    <graphic>
                        <ImageView fitHeight="16.0" fitWidth="25.0" pickOnBounds="true" preserveRatio="true">
                            <image>
                                <Image url="@../../images/icons8_Facebook_F_48px.png" />
                            </image>
                        </ImageView>
                    </graphic>
                    <font>
                        <Font name="System Bold Italic" size="12.0" />
                    </font>
                </Button>
                <Button fx:id="btnSignup" layoutX="4.0" layoutY="296.0" mnemonicParsing="false" prefHeight="34.0" prefWidth="275.0" stylesheets="css.fullpackstyling.css" text="S'inscrire">
                    <font>
                        <Font name="System Bold Italic" size="18.0" />
                    </font>
                </Button>
                <Separator layoutX="8.0" layoutY="190.0" prefHeight="7.0" prefWidth="275.0" />
                <Label alignment="CENTER" layoutX="126.0" layoutY="179.0" prefHeight="23.0" prefWidth="31.0" style="-fx-background-color: #fff;" text="Ou" textFill="#5b5a5a">
                    <font>
                        <Font name="Segoe UI" size="12.0" />
                    </font>
                </Label>
                <Separator layoutY="270.0" prefHeight="7.0" prefWidth="275.0" />
                <Label alignment="CENTER" layoutX="122.0" layoutY="262.0" prefHeight="23.0" prefWidth="31.0" style="-fx-background-color: #fff;" text="Ou" textFill="#5b5a5a">
                    <font>
                        <Font name="Segoe UI" size="12.0" />
                    </font>
                </Label>
                <Label fx:id="lblErrors" alignment="CENTER" layoutX="8.0" layoutY="95.0" prefHeight="23.0" prefWidth="275.0" textFill="#ff6354">
                    <font>
                        <Font name="Segoe UI Bold" size="10.0" />
                    </font>
                </Label>
            </children>
        </Pane>
        <Label alignment="CENTER" layoutX="394.0" layoutY="466.0" prefHeight="23.0" prefWidth="120.0" text="Design by Balbali" textFill="#5b5a5a" AnchorPane.bottomAnchor="14.0">
            <font>
                <Font name="Segoe UI Bold" size="12.0" />
            </font>
        </Label>
        <Label alignment="CENTER" layoutX="747.0" layoutY="466.0" prefHeight="23.0" prefWidth="93.0" text="Télécharger App" textFill="#a6b1cc" AnchorPane.bottomAnchor="14.0">
            <font>
                <Font name="Segoe UI Bold" size="12.0" />
            </font>
        </Label>
    </children>
```

4. Le controller :

On commence par importer les modules, les attribut lblerrors, txtUsername,txtPassword,btnSignun de la classe fxml. On crée un attribut con de type Connection, une requête préparer et un ResultSet tout les trois null ;

```java
package controller;

import connexion.Connect;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Node;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.input.MouseEvent;
import javafx.scene.paint.Color;
import javafx.stage.Stage;
import java.io.IOException;
import java.net.URL;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ResourceBundle;

public class LoginController implements Initializable {

    @FXML
    private Label lblErrors;

    @FXML
    private TextField txtUsername;

    @FXML
    private TextField txtPassword;

    @FXML
    private Button btnSignin;

    Connection con= null;
    PreparedStatement preparedStatement = null;
    ResultSet resultSet = null;
```

Les méthodes :

La méthodes handleButtonAction :

Cette méthode prend en paramètre une action event. Si l'action est égal au bouton signup alors elle redirige vers la scene biodata.fxml qui permet l'inscription.

Pour ce faire elle analyse l'action avec la classe Node qui permet de manipuler les fichier fxml et d'obtenir des informations.

Puis la classe stage permet de de crée une fenêtre et récupère l'attribut node.

Puis on ferme la fenêtre actuelle avec la méthode close de la classe stage puis on en ouvre une nouvelle avec la classe scene qui charge le fichier biodata.fxml. Puis on finis par définir la scene et l'afficher avec méthode setScene et show de l'attribut stage.

```java
@FXML
public void handleButtonAction(MouseEvent event) {
    if (event.getSource() == btnSignin) {

        if (login().equals("Succès")) {
            try {
                Node node = (Node) event.getSource();
                Stage stage = (Stage) node.getScene().getWindow();
                //stage.setMaximized(true);
                stage.close();
                Scene scene = new Scene(FXMLLoader.load(getClass().getResource("view.biodata.fxml")));
                stage.setScene(scene);
                stage.show();
            } catch (IOException ex) {
                System.err.println(ex.getMessage());
            }
        }
    }
}
```

La méthode initializable :

Cette méthode permet de vérifier la connexion de la base de donnée.

Si la connexion est nulll alors elle fait appel au composant txtErrors et modifie sont contenu en changeant la couleur avec la méthode setTextFill et change la couleur en rouge puis on change le message du text avec la méthode set text.

Sinon la couleur est verte et le text indique que la base de donné est en place.

```java
@Override
public void initialize(URL url, ResourceBundle rb) {
    // TODO
    if (con == null) {
        lblErrors.setTextFill(Color.TOMATO);
        lblErrors.setText("Erreur serveur: vérification");
    } else {
        lblErrors.setTextFill(Color.GREEN);
        lblErrors.setText("Le serveur est en place: prêt");
    }
}
```

La méthode logincontroller :

Cette méthode permet la connection avec la base de donnée en utilisant la méthode connexion de la classe connect.

```java
public LoginController() {
    con = Connect.Connexion();
}
```

La méthode login :

Cette méthode permet de connecter l'utilisateur en vérifiant son identifiant et retourne status de la vérification.

Elle commence par définir le status, mail et le mot de passe.

Puis elle vérifie si le mail et le mot de passe ne soit pas vide. Si ils sont vide alors alors on affiche un message d'erreur avec le setter du composant lblerror que nous verrons plus tard et change le status en erreur.

Sinon et on crée un attribut String qui permet de sélectionner le mail et le mot de passe de la table admin.

Et on essaie de récupérer le contenu du mail et mot de passe avec une requête préparer puis on stocke le résultat dans une variable qui contient la méthode preparedStatement.

Ensuite si le résultat ne peut pas récupérer les lignes avec la méthode next alors on affiche un message d'erreur toujours avec le setter du composant lblerror . Sinon on affiche un message de réussite.

Si le try ne marche pas alors le catch affiche une erreur SQLException et change le statut en ''Exception''.

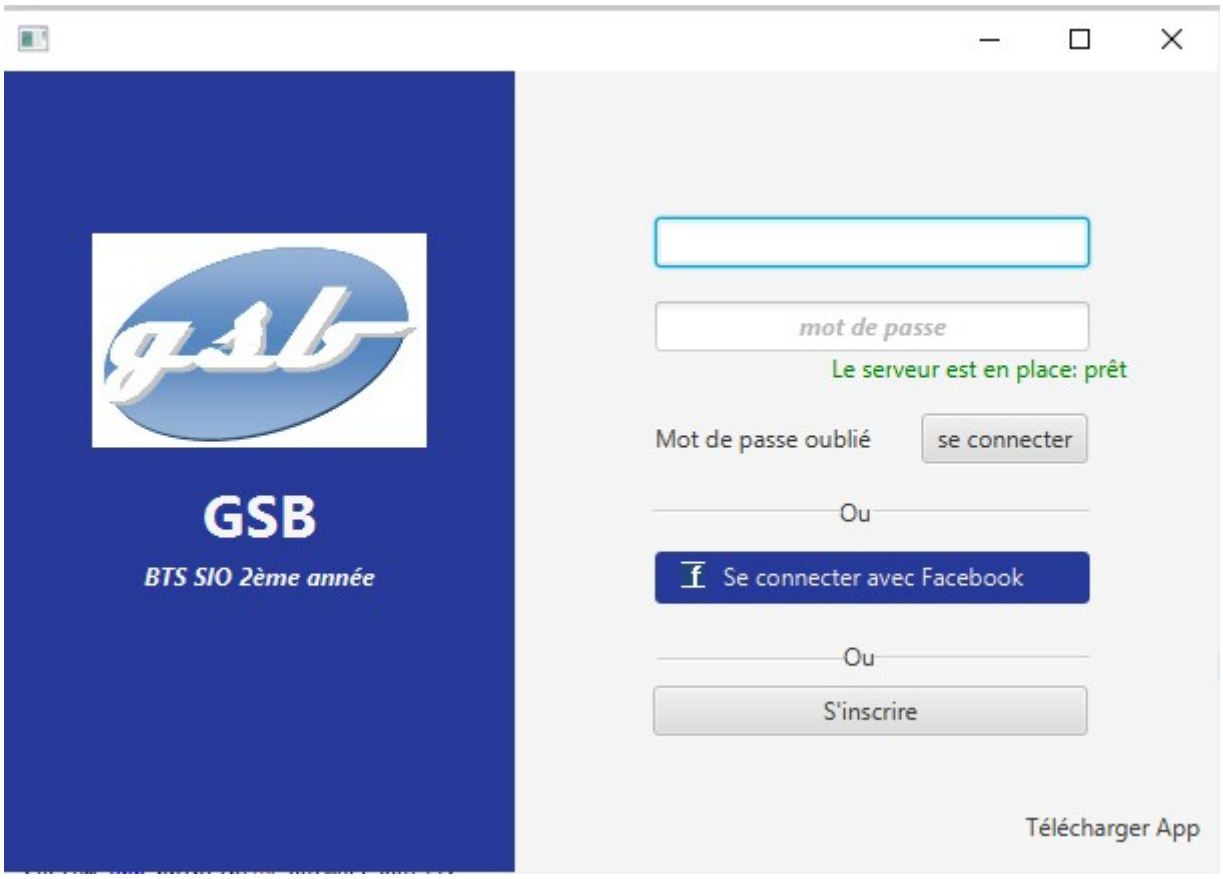
```
private String login() {
    String status="Succès";
    String email = txtUsername.getText();
    String password= txtPassword.getText();
    if(email.isEmpty() || password.isEmpty()) {
        setLblError(Color.TOMATO, "\n" +"Identifiants vides");
        status = "Erreur";
    } else {
        String sql = "SELECT * FROM admins Where email = ? and password = ?";
        try {
            preparedStatement = con.prepareStatement(sql);
            preparedStatement.setString(1, email);
            preparedStatement.setString(2, password);
            resultSet = preparedStatement.executeQuery();
            if (!resultSet.next()) {
                setLblError(Color.TOMATO, "Entrer Email/Password Correct");
                status = "Error";
            } else {
                setLblError(Color.GREEN, "\n" +"Connexion réussie..Redirection..");
            }
        } catch (SQLException ex) {
            System.err.println(ex.getMessage());
            status = "Exception";
        }
    }
    return status;
}
```

La méthode setLblError :

Elle prend en paramètre une coleur et du text et ne retourne rien.

Cette méthode permet de modifier le setter de de l'attribut lblError avec les méthode setTextFill qui permet de définir une couleur et la méthode setText permet de définir le text.

Cette méthode finis par afficher le text.
Résultat :



Création du tableau:

Partie 1 : La vue

A partir de se script de ce scripte fxml suivant :

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.DatePicker?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.Separator?>
<?import javafx.scene.control.Tab?>
<?import javafx.scene.control.TabPane?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.control.TextArea?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>
<?import javafx.scene.text.Text?>

<AnchorPane id="AnchorPane" prefHeight="547.0" prefWidth="953.0" stylesheets="@..\css\biodataCss.css" xmlns="http://javafx.com/javafx/21" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="controller.biodataController">
    <children>
        <TabPane layoutY="85.0" prefHeight="462.0" prefWidth="953.0" tabClosingPolicy="UNAVAILABLE" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0"
        AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="85.0">
            <tabs>
                <Tab fx:id="tabBiodata" text="Informations personnelles">
                    <content>
                        <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="433.0" prefWidth="975.0">
                            <children>
                                <Separator layoutX="723.0" layoutY="-1.0" orientation="VERTICAL" prefHeight="436.0" prefWidth="6.0" />
                                <Label layoutX="744.0" layoutY="25.0" text="Saisie informations" textFill="#000000c8" underline="true">
                                    <font>
                                        <Font name="System Bold Italic" size="26.0" />
                                    </font>
                                </Label>
                                <VBox layoutX="744.0" layoutY="75.0" spacing="10.0">
                                    <children>
                                        <VBox layoutX="714.0" layoutY="165.0" spacing="3.0">
                                            <children>
                                                <Label layoutX="714.0" layoutY="165.0" text="ID">
                                                    <font>
                                                        <Font name="System Bold" size="15.0" />
                                                    </font>
                                                </Label>
                                                <TextField fx:id="txtId" layoutX="714.0" layoutY="182.0" prefHeight="25.0" prefWidth="228.0" promptText="Id Auto">
                                                    <font>
                                                        <Font name="System Bold" size="15.0" />
                                                    </font>
                                                </TextField>
                                            </children>
                                        </VBox>
                                        <VBox layoutX="712.0" layoutY="210.0" spacing="3.0">
                                            <children>
                                                <Label layoutX="714.0" layoutY="165.0" text="Nom">
                                                    <font>
                                                        <Font name="System Bold" size="15.0" />
                                                    </font>
                                                </Label>
                                                <TextField fx:id="txtNom" layoutX="714.0" layoutY="182.0" prefHeight="25.0" prefWidth="228.0" promptText="nom et prénom">
                                                    <font>
                                                        <Font name="System Bold" size="15.0" />
                                                    </font>
                                                </TextField>
                                            </children>
                                        </VBox>
                                        <VBox spacing="3.0">
                                            <children>
                                                <Label layoutX="714.0" layoutY="165.0" text="Adresse">
                                                    <font>
                                                        <Font name="System Bold" size="15.0" />
                                                    </font>
                                                </Label>
                                                <TextArea fx:id="txtAdresse" prefHeight="62.0" prefWidth="228.0" promptText="adresse complète">
                                                    <font>
                                                        <Font name="System Bold" size="15.0" />
                                                    </font>
                                                </TextArea>
                                            </children>
                                        </VBox>
                                        <VBox spacing="3.0">
                                            <children>
                                                <Label layoutX="714.0" layoutY="165.0" text="Date de naissance">
                                                    <font>
                                                        <Font name="System Bold" size="15.0" />
                                                    </font>
                                                </Label>
                                                <DatePicker fx:id="dateNaissance" prefHeight="25.0" prefWidth="228.0" promptText="Date de naissance" />
                                            </children>
                                        </VBox>
                                    </children>
                                </VBox>
                                <Button fx:id="btnEnregistrer" layoutX="862.0" layoutY="372.0" mnemonicParsing="false" onAction="#enregistrer" prefHeight="31.0" prefWidth="117.0" text="Enregistrer">
```

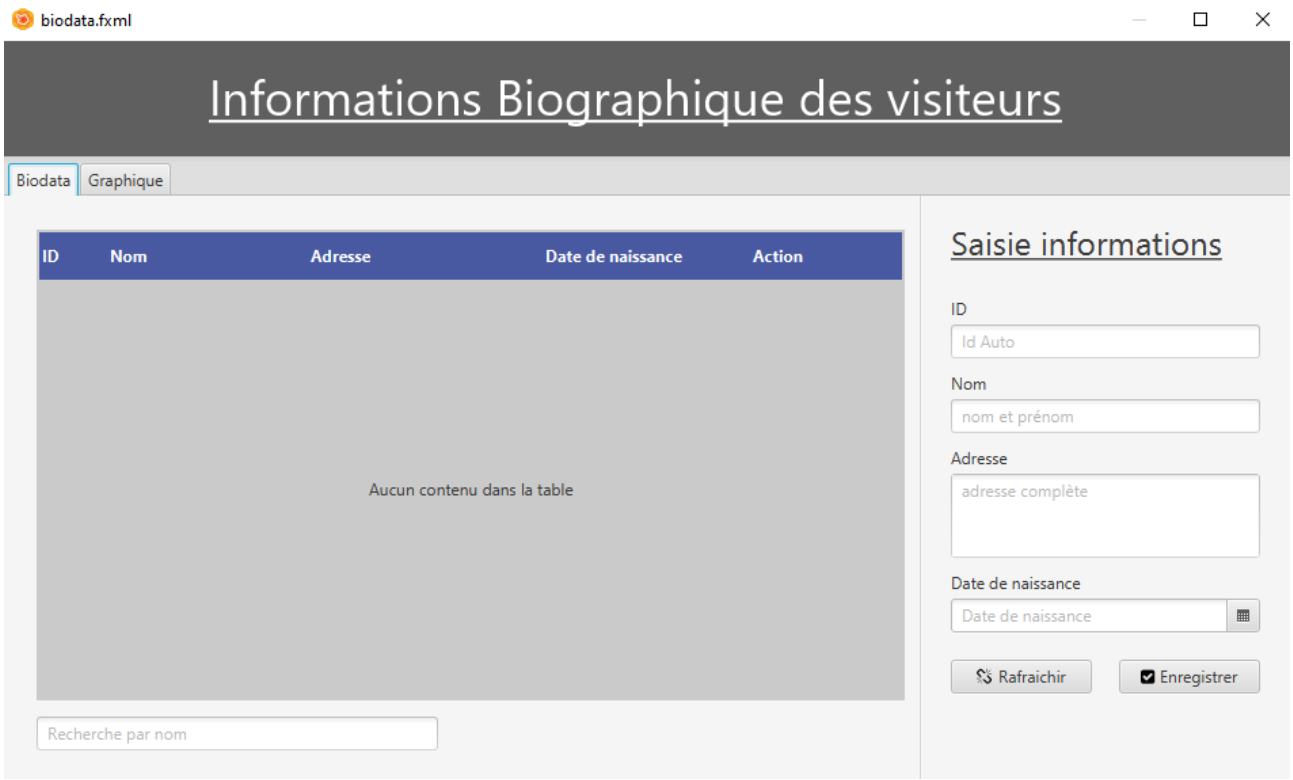
```
                            <font>
                                <Font name="System Bold" size="15.0" />
                            </font>
                        </Button>
                        <TableView fx:id="tableData" layoutX="26.0" layoutY="25.0" onMouseClicked="#klikTableData" prefHeight="347.0" prefWidth="686.0">
                            <columns>
                                <TableColumn minWidth="8.0" prefWidth="9.0" />
                                <TableColumn fx:id="colId" minWidth="8.0" prefWidth="50" text="ID" />
                                <TableColumn fx:id="colNom" prefWidth="148.0" text="Nom" />
                                <TableColumn fx:id="colAdresse" prefWidth="146.0" text="Adresse" />
                                <TableColumn fx:id="colAnnee" prefWidth="152.0" text="Date de naissance" />
                                <TableColumn fx:id="colAction" prefWidth="179.0" text="Action" />
                            </columns>
                        </TableView>
                        <TextField fx:id="txtChercher" layoutX="26.0" layoutY="384.0" onKeyReleased="#rechercher" prefHeight="25.0" prefWidth="296.0" promptText="Recherche par nom" />
                        <Button fx:id="btnRafraichir" layoutX="737.0" layoutY="372.0" mnemonicParsing="false" onAction="#actualiser" prefHeight="31.0" prefWidth="114.0" text="Rafraichir">
                            <font>
                                <Font name="System Bold" size="15.0" />
                            </font>
                        </Button>
                    </children>
                </AnchorPane>
            </content>
        </Tab>
        <Tab fx:id="tabGrafiknaissance" text="Graphique">
            <content>
                <AnchorPane fx:id="paneLoadGrafik" minHeight="0.0" minWidth="0.0" prefHeight="180.0" prefWidth="200.0" />
            </content>
        </Tab>
    </tabs>
</TabPane>
<AnchorPane prefHeight="85.0" prefWidth="1019.0" styleClass="headerPane" >
    <children>
        <Text fill="WHITE" layoutX="175.0" layoutY="63.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Informations Biographiques des visiteurs" textAlignment="CENTER" underline="true">
            <font>
                <Font name="System Bold Italic" size="36.0" />
            </font>
        </Text>
    </children>
</AnchorPane>
</children>
</AnchorPane>
```

Mais également grâce au scripte css qui permet de personnaliser la page fxml :

```css
.hyperlink{
    -fx-text-fill: #cb0006;
}

.headerPane{
    -fx-background-color: derive(grey, -30%);
}

.table-view {
    -fx-base: white;
    -fx-control-inner-background: white;
    -fx-background-color: #c7c7c7;
    -fx-table-cell-border-color: transparent;
    -fx-table-header-border-color: transparent;
    -fx-padding: 2;
}

.table-view .column-header-background {
    -fx-background-color: #42539d;
}

.table-view .column-header, .table-view .filler {
    -fx-size: 35;
    -fx-border-width: 0 0 0 0;
    -fx-background-color: transparent;
    -fx-border-color:
        transparent
        transparent
        derive(-fx-base, 80%)
        transparent;
    -fx-border-insets: 0 10 1 0;

}

.table-view .column-header .label {
    -fx-font-size: 12px;
    -fx-font-family: "Segoe UI bold";
    -fx-text-fill: white;
    -fx-alignment: center-left;

}

.table-view:focused .table-row-cell:filled:focused:selected {
    -fx-background-color: -fx-focus-color;
}
```

```css
.table-cell {
    -fx-cell-size: 4.0em;
    -fx-padding: 1em 0em 0.1em 0.1em;
    -fx-font: 12px "Segoe UI";
    -fx-alignment: bottom-left;
}

.table-row-cell:empty {
    -fx-background-color: #f8f8f8;
    -fx-base: transparent;
    -fx-control-inner-background: transparent;
    -fx-table-cell-border-color: transparent;
    -fx-table-header-border-color: transparent;
    -fx-padding: 0;
}

.table-row-cell:empty table-cell {
    -fx-border-width: 0px;
}
```

Nous avons pu obtenir ce résultat :



Partie 2 : Le modèle

Le modèle va nous permettre de définir les attributs de la classe fxml ce qui va nous permettre de représenter le contenu de la table biodata et ainsi communiquer avec la base de donnée.

Pour cela on commence par importer les modules javafx et et java qui vont permettre l'utilisation de méthode spécifique.

```
import javafx.beans.property.ObjectProperty;
import javafx.beans.property.SimpleObjectProperty;
import javafx.beans.property.SimpleStringProperty;
import javafx.beans.property.StringProperty;
import java.text.SimpleDateFormat;
import java.util.Date;
```

Puis on défini en privée l'id,le nom l'adresse la date de naissance et le format de la date avec la classe StringProperty et ObjectProperty en ajoutant également le type final qui permet de ne pas changer les attibut:

```
public class modelBiodata {
    private final StringProperty id = new SimpleStringProperty();
    private final StringProperty nom = new SimpleStringProperty();
    private final StringProperty adresse = new SimpleStringProperty();
    private final ObjectProperty<Date> DateDeNaissance = new SimpleObjectProperty<>();
    private String formatdate;
```

On ajoute un constructeur par défaut qui va permettre de crée des objets :

```
public modelBiodata() {
}
```

Puis comme pour la classe modèle user on ajoute les getter,les setters :

```
public String getId() {
    return id.get();
}

public void setId(String value) {
    id.set(value);
}

public StringProperty idProperty() {
    return id;
}

public String getNom() {
    return nom.get();
}

public void setNom(String value) {
    nom.set(value);
}

public StringProperty nomProperty() {
    return nom;
}

public String getAdresse() {
    return adresse.get();
}

public void setAdresse(String value) {
    adresse.set(value);
}

public StringProperty adresseProperty() {
    return adresse;
}

public Date getDateDeNaissance() {
    return DateDeNaissance.get();
}

public void setDateDeNaissance(Date value) {
    DateDeNaissance.set(value);
}

public ObjectProperty<Date> DateDeNaissanceProperty() {
    return DateDeNaissance;
}

public String getFormatDate() {
    Date date = getDateDeNaissance();
    SimpleDateFormat df = new SimpleDateFormat("dd-MM-yyyy");
    String format = df.format(date);
    return format;
}

public void setFormatdate(String formatdate) {
    this.formatdate = formatdate;
}
```

Concernant la méthode getFormatDate on crée un objet Date qui récupère la getter de la date de naissance puis on crée un format qu'on attribut ensuite a date.

Partie 3 : l'interface

Cette interface permettra d'importer d'utiliser dans le contrôleur les méthode de type void insert,delete,update,autoid et prennent en paramètre un objet de type modelBiodate mais également getAll et recherche par nom qui sont eux de type ObservableList qui permet d'afficher une liste de type modelBiodata qui affiche le changement.

```java
package interfaces;
import model.modelBiodata;
import javafx.collections. ObservableList;

public interface interBiodata {
    void insert (modelBiodata m);
    void delete (modelBiodata m);
    void update (modelBiodata m);
    ObservableList<modelBiodata> getAll();
    ObservableList<modelBiodata> Rechercheparnom (String a);
    void autoId (modelBiodata m);
}
```

Partie 4 : L'implémentation

On commence par importer les classes :

```java
import connexion.Connect;

import interfaces.interBiodata;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import model.modelBiodata;
import java.sql.Date;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
```

Les méthodes :

La méthode insert :

```
@Override
public void insert(modelBiodata m) {
    k = new Connect();
    PreparedStatement ps;
    try {
        ps = k.Connexion().prepareStatement("insert into tablebiodata values (?,?,?,?)");
        ps.setString(1, m.getId());
        ps.setString(2, m.getNom());
        ps.setString(3, m.getAdresse());
        ps.setDate(4, (Date) m.getDateDeNaissance());
        ps.executeUpdate(); // Use executeUpdate for INSERT operation
    } catch (Exception e) {
        Logger.getLogger(implBiodata.class.getName()).log(Level.SEVERE, null, e);
    }
}
```

Elle prend en paramètre un objet m puis dans un try catch se connecte à la base de donnée prépare une requête qui insère dans la table tablebiodata l'id, le nom, l'adresse et la date converti en type date. En cas d'erreur un message s'affiche qui indique l'erreur précise et un niveau sévère.

La méthode delete :

```
@Override
public void delete(modelBiodata m) {
     k = new Connect();
     PreparedStatement ps;
     try {
         ps = k.Connexion().prepareStatement("delete from tablebiodata WWwhere id= ?");
         ps.setString(1, m.getId());
         ps.executeUpdate();
     } catch (Exception e) {
         Logger.getLogger(implBiodata.class.getName()).log(Level.SEVERE, null, e);
     }
}
```

Elle prend en paramètre un objet m puis dans un try catch se connecte à la base de donnée prépare une requête qui supprime dans la table tablebiodata quand l'id est égale au paramètre getId. En cas d'erreur un message s'affiche qui indique l'erreur précise et un niveau sévère.

La méthode update :

```
@Override
public void update(modelBiodata m) {
    k = new Connect();
    PreparedStatement ps;
    try {
        ps = k.Connexion().prepareStatement("update tablebiodata set nom=?, adresse=?, datedenaissance=? where id= ?");
        ps.setString(4, m.getId());
        ps.setString(1, m.getNom());
        ps.setString(2, m.getAdresse());
        ps.setDate(3, (Date) m.getDateDeNaissance());
        ps.executeUpdate();
    } catch (Exception e) {
        Logger.getLogger(implBiodata.class.getName()).log(Level.SEVERE, null, e);
    }
}
```

Elle prend en paramètre un objet m puis dans un try catch se connecte à la base de donnée prépare une requête qui modifie dans la table tablebiodata le nom, l'adresse, la date converti en type date quand l'id est égal à l'id en paramètre. En cas d'erreur un message s'affiche qui indique l'erreur précise et un niveau sévère.

La méthode getAll :

```
@Override
public ObservableList<modelBiodata> getAll() {
    k = new Connect();
    ObservableList<modelBiodata> listData = FXCollections.observableArrayList();
    try {
        String sql = "select * from tablebiodata";
        ResultSet rs = k.Connexion().createStatement().executeQuery(sql);
        while (rs.next()) {
            modelBiodata m = new modelBiodata();
            m.setId(rs.getString(1));
            m.setNom(rs.getString(2));
            m.setAdresse(rs.getString(3));
            m.setDateDeNaissance(rs.getDate(4));
            listData.add(m);
        }
    } catch (Exception e) {
        Logger.getLogger(implBiodata.class.getName()).log(Level.SEVERE, null, e);
    }
    return listData;
}
```

Elle est de type ObservableList avec en paramètre le modèle modelBiodata elle se connecte à la base de donnée puis initialise une liste de type ObservableList. Dans un try catch on défini la requête sql et on l'exécute. Tant que le résultat retourne à la ligne alors on attribut aux setters de l'id, du nom, de l'adresse et de la date de naissance le contenu que la base de donnée récupérer. On finis par insérer le contenu dans la liste et on la retourne après le catch qui affiche le message d'erreur.

La méthode Rechercherparnom :

```
@Override
public ObservableList<modelBiodata> Rechercheparnom(String a) {
    k = new Connect();
    ObservableList<modelBiodata> listData = FXCollections.observableArrayList();
    try {
        String sql = "select * from tablebiodata where nom like '%" + a + "%'";
        ResultSet rs = k.Connexion().createStatement().executeQuery(sql);
        while (rs.next()) {
            modelBiodata m = new modelBiodata();
            m.setId(rs.getString(1));
            m.setNom(rs.getString(2));
            m.setAdresse(rs.getString(3));
            m.setDateDeNaissance(rs.getDate(4));
            listData.add(m);
        }
    } catch (Exception ex) {
        Logger.getLogger(implBiodata.class.getName()).log(Level.SEVERE, null, ex);
    }
    return listData;
}
```

C'est le même fonctionnement que la méthode getAll mais avec une requête sql différente qui sélectionne tout de la table tablebiodata quand le nom ressemble au nom entrer en paramètre de la méthode.

La méthode autoid :

```
@Override
public void autoId(modelBiodata m) {
    k = new Connect();
    try {
        ResultSet rs = k.Connexion().createStatement().executeQuery("select * from tablebiodata");
        while (rs.next()) {
            String code = rs.getString(1).substring(2);
            String auto = "" + (Integer.parseInt(code) + 1);
            String nol = "";
            if (auto.length() == 1) {
                nol = "00";
            } else if (auto.length() == 2) {
                nol = "0";
            } else if (auto.length() == 3) {
                nol = "";
            }
            m.setId("B." + nol + auto);
        }
        if (m.getId() == null) {
            m.setId("B.001");
        }
    } catch (Exception ex) {
        Logger.getLogger(implBiodata.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Cette méthode permet d'afficher les id de la table tablebiodata automatiquement. Pour ce faire elle prend en paramètre un objet de type modelBiodata puis se connecte à la base de donnée et selectionne tout de la table. Puis recupère les lignes, et tant qu'il y a des ligne elle defini plusieurs attribut qui sont : code qui récupère le résultat en deux parti, auto qui ajoute un espace et la deuxième parti du resultat et ajoute et nol qui est provisoirement +1 null. Si la longueur est égale à 1 alors nol prend deux 0 ,si la longueur de l'auto est de de 2 l'alors nol est égale à 1 zero et 3 on enleve les zero. On fini par afficher l'id automatique sinon par defaut on affiche b.001. Le catch affiche toujours la même erreur.

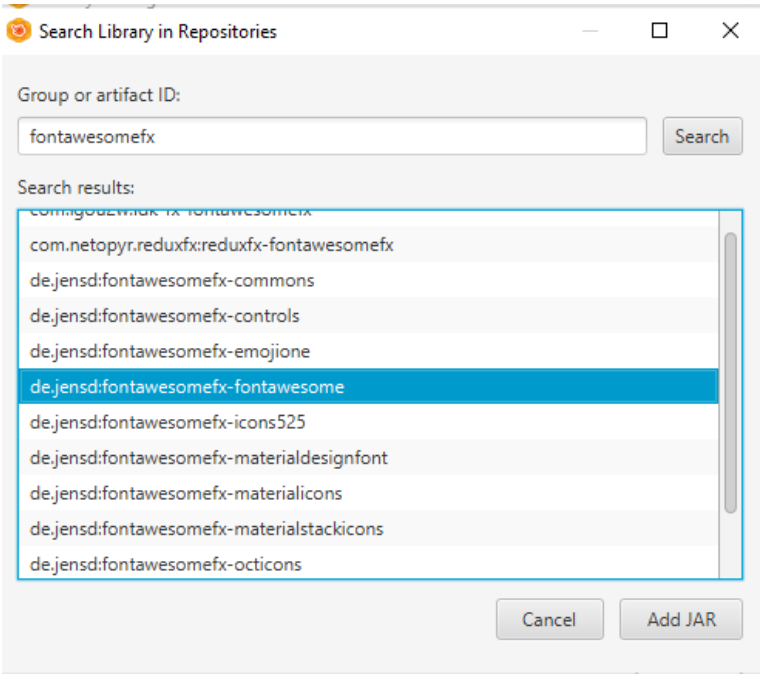
Partie 5 : Le contrôleur

Information importante : A cause de problème de compatibilité entre le logiciel du prof qui est maven et eclipse qui est le mien je rencontre des problèmes lors de la création des icones avec ma classe Awesome je vais donc importer les icones depuis scenebuilder.

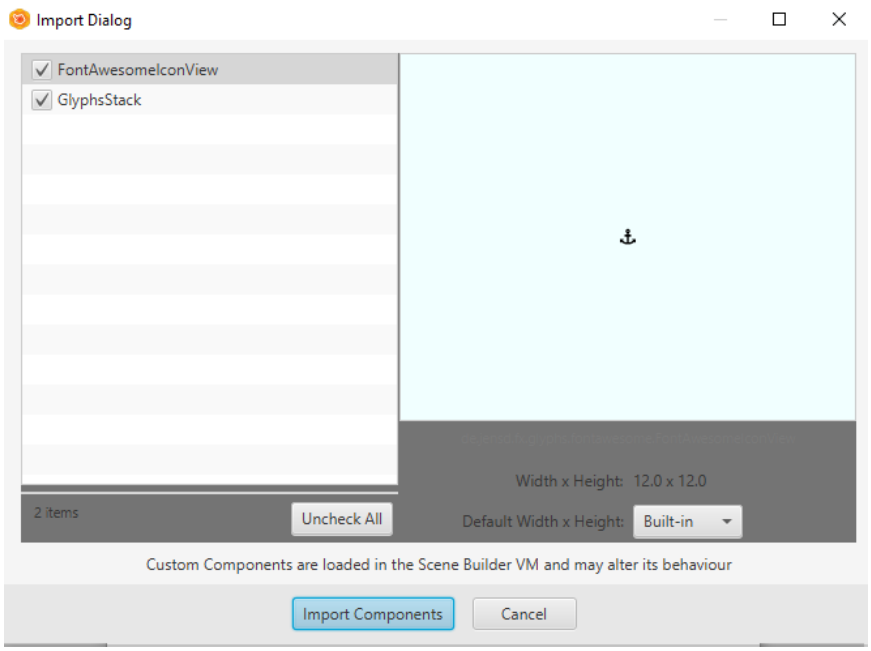
```
import de.jensd.fx.fontawesomefx.Awesomedude;
import de.jensd.fx.fontawesome.AwesomeIcon;
```

Pour ce faire :

On commence par chercher la libraire fontawesomfx dans scenebuilder :

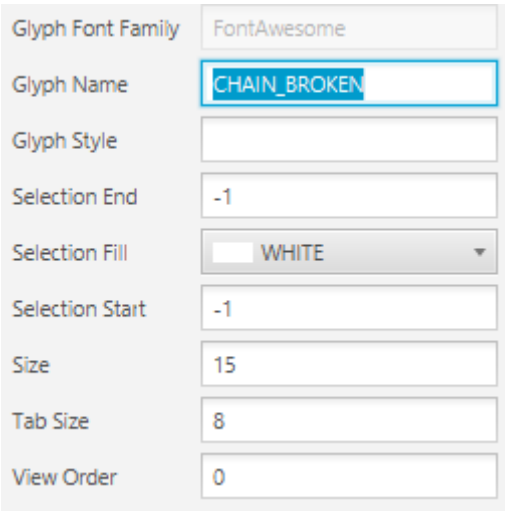


Puis on importe :

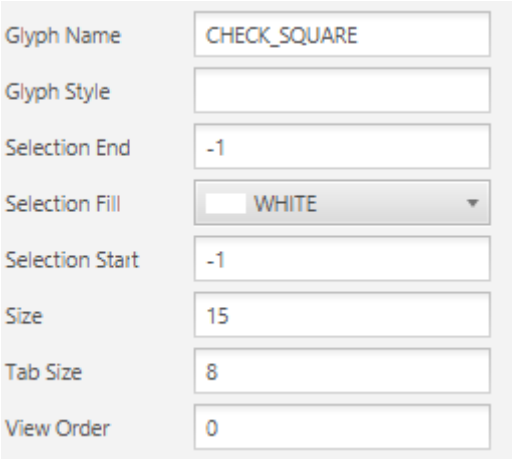


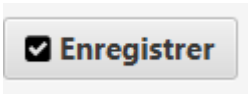
J'ajoute ensuite le composant fontawesomeicon au bouton rafraîchir :

Puis je choisi le nom du glyph qui va changer l'icon et change la taille :



Pareil pour le bouton enregistrer :





Maintenant les explications du controller :

On commence par importer tout les classe (sauf fontawesome) :

```
import implement.implBiodata;
import interfaces.interBiodata;
import model.modelBiodata;
import java.io.IOException;
import java.net.URL;
import java.sql.Date;
import java.time.LocalDate;
import java.util.ResourceBundle;
import javafx.beans.property.SimpleBooleanProperty;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.DatePicker;
import javafx.scene.control.Hyperlink;
import javafx.scene.control.Tab;
import javafx.scene.control.TableCell;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.KeyEvent;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.HBox;
import javafx.stage.StageStyle;
import javafx.util.Callback;
```

Puis on importe toute les id de la classe fxml :

```
@FXML
private Tab tabGrafiknaissance, tabBiodata;

@FXML
private TextField txtId;

@FXML
private TextField txtNom;

@FXML
private TextArea txtAdresse;

@FXML
private DatePicker dateNaissance;

@FXML
private Button btnEnregistrer;

@FXML
private TableView<modelBiodata> tableData;

@FXML
private TableColumn<modelBiodata, String> colId;

@FXML
private TableColumn<modelBiodata, String> colNom;

@FXML
private TableColumn<modelBiodata, String> colAdresse;

@FXML
private TableColumn<modelBiodata, String> colAnnee;

@FXML
private TableColumn colAction;

@FXML
private TextField txtChercher;

@FXML
private Button btnRafraichir;

@FXML
private AnchorPane paneLoadGrafik;
```

On crée également des objet de type observablelist qui permettent de récupérer les listes delete et data. On crée également des attribut statucode et statu clic qui nous permettrons plus tard la communication entre le client et le scripte.

```
interBiodata crudData = new implBiodata();
ObservableList<modelBiodata> listData;

private String StatusCode, statusclic;
ObservableList<modelBiodata> listDelete;
```

Les méthodes :

La méthode initialize :

Elle prend en paramètre l'url du logiciel avec la classe URL et la ressource c'est à dire le contenu avec la classe ResourceBundle.

Cette méthode permet d'initialiser les valeur des case à partir des attribut fxml qui sont l'id, le nom etc. Grâce à la méthode setCellValueFactory qui récupère la case et insère les propriétés des attribut fxml.

On défini également la colonne action qui crée un nouveau tableau et qui permet de communiquer avec les bouton enregistrer et rafraichir. La communication se fait via les méthode observable et tablecell qui rend la colonne action null pour le rafraichir et retourne le tableau de donnée si il est enregistrer.

On finis par attribuer aux attribut initialiser avant des valeur par défaut comme 0 pour statusCode.

```
@Override
public void initialize(URL url, ResourceBundle rb) {
    colId.setCellValueFactory(
            (TableColumn.CellDataFeatures<modelBiodata, String> cellData) ->
                    cellData.getValue().idProperty());
    colNom.setCellValueFactory(
            (TableColumn.CellDataFeatures<modelBiodata, String> cellData) ->
                    cellData.getValue().nomProperty());
    colAdresse.setCellValueFactory(
            (TableColumn.CellDataFeatures<modelBiodata, String> cellData) ->
                    cellData.getValue().adresseProperty());
    colAnnee.setCellValueFactory(new PropertyValueFactory("formatdate"));
    colAction.setCellValueFactory(new Callback<TableColumn.CellDataFeatures<Object, Boolean>,
            ObservableValue<Boolean>>() {
        @Override
        public ObservableValue<Boolean> call(TableColumn.CellDataFeatures<Object, Boolean> p) {
            return new SimpleBooleanProperty(p.getValue() != null);
        }
    });


    colAction.setCellFactory(new Callback<TableColumn<Object, Boolean>, TableCell<Object, Boolean>>() {
        @Override
        public TableCell<Object, Boolean> call(TableColumn<Object, Boolean> p) {
            return new ButtonCell(tableData);
        }
    });
    listData = FXCollections.observableArrayList();
    dateNaissance.setValue(LocalDate.of(1990, 01, 01));
    StatusCode = "0";
    statusclic = "0";
    affichageDonnees();
    autoId();
    tableData.getSelectionModel().clearSelection();
    loadGrafik();
}
```

La méthode loadgrafik :

Elle permet de charger le graphique de la classe fxml grafik.fxml. Elle crée un objet FXMLLoader qui permet de de charger la classe, puis l'affiche si le try marche avec l'attribut fxml et l'affiche en sous page.

```
private void loadGrafik() {
    try {
        FXMLLoader fxmlLoader = new FXMLLoader();
        Parent p = fxmlLoader.load(getClass().getResourceAsStream("/view/grafik.fxml"));
        paneLoadGrafik.getChildren().add(p);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

La méthode dialog:

Elle prend en paramètre le type d'alerte et le message. Elle permet d'afficher un message en cas d'erreur et arrête se processus .

```
private void dialog(Alert.AlertType alertType, String s) {
    Alert alert = new Alert(alertType, s);
    alert.initStyle(StageStyle.UTILITY);
    alert.setTitle("Info");
    alert.showAndWait();
}
```

La méthode clear :

Elle permet de nettoyer les attribut de la classe fxml et tout remettre à zero.

```
private void clear() {
    txtId.clear();
    txtNom.clear();
    txtAdresse.clear();
    txtChercher.clear();
    dateNaissance.setValue(LocalDate.of(1990, 01, 01));
    StatusCode = "0";
}
```

La méthode affichageDonnees :

Elle permet d'afficher la liste avec la méthode getAll de implBiodata à partir de l'objet cruddata initaliser avant. Puis défini le tableau avec la liste.

```
private void affichageDonnees() {
    listData = crudData.getAll();
    tableData.setItems(listData);
}
```

La méthode autoid :

Permet d'afficher dans la colonne id tout les id de la table et d'afficher la suite automatiquement grâce à la méthode autoid de la classe implement.

```
private void autoId() {
    modelBiodata m = new modelBiodata();
    crudData.autoId(m);
    txtId.setText(m.getId());
}
```

La méthode enregistrer :

Cette méthode prend en paramètre une action de l'utilisateur et permet d'insérer ou de mettre à jours les attribut de la classe fxml si le statut est égale à 0. Elle affiche une erreur avec la méthode dialog puis change les case du tableau.

```
@FXML
private void enregistrer(ActionEvent event) {
    modelBiodata m = new modelBiodata();
    m.setId(txtId.getText());
    m.setNom(txtNom.getText());
    m.setAdresse(txtAdresse.getText());
    m.setDateDeNaissance(Date.valueOf(dateNaissance.getValue()));
    if (StatusCode.equals("0")) {
        crudData.insert(m);
    } else {
        crudData.update(m);
    }
    dialog(Alert.AlertType.INFORMATION, "Données sauvegardées");
    affichageDonnees();
    clear();
    autoId();
}
```

La méthode :

Cette méthode prend également en paramètre une action. Elle permet de sélectionner une tableau et de selectionner l'id, le nom, l'adresse et la date de naissance.

```
@FXML
private void klikTableData(MouseEvent event) {
    if (statusclic.equals("1")) {
        StatusCode = "1";
        try {
            modelBiodata clic = tableData.getSelectionModel().getSelectedItem();
            txtId.setText(clic.getId());
            txtNom.setText(clic.getNom());
            txtAdresse.setText(clic.getAdresse());
            dateNaissance.setValue(LocalDate.parse(clic.getDateDeNaissance().toString()));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

La méthode rechercher:

Elle prend en paramètre un événement et recherche la nom avec la méthode rechercherparnom de la classe implement puis change le tableau avec la liste des nom.

```
@FXML
private void rechercher(KeyEvent event) {
    listData = crudData.Rechercheparnom(txtChercher.getText());
    tableData.setItems(listData);
}
```

La méthode actualiser :

Elle permet d'actualiser et affiche le nouveau tableau et les id avec les méthodes clear,affichagedonnes et autoid.

```
@FXML
private void actualiser(ActionEvent event) {
    clear();
    affichageDonnees();
    autoId();
}
```

La classe buttonCell :

Cette classe privée hérite de la classe tablecell et permet la suppression et la modification client à partir des lien supprimer et éditer dans la colonne action.

Pur ce faire on crée trois objet en final qui sont :

-Le lien et bouton delete

-Le lien et bouton editer

-Et des boite qui contient les deux boutons.

Les boutons delete et editer sont de type Hyeperlink et la boite de type Hbox.

Dans le constructeur on attribut une action au lien delete et edit. Le bouton delete sélectionne la ligne puis la supprime avec la méthode delete de la classe implement elle met ensuite à jours le tableau avec le tableau avec les méthode clear,affichedonne etc. Elle finis pas afficher un message qui indique que les donnée on bien été supprimer.

Pour le lien edit on change le statuclic à 1 et grâce à la méthode klikTableData on peut changer les valeur du nom etc..

```
private class ButtonCell extends TableCell<Object, Boolean> {

    final Hyperlink cellButtonDelete = new Hyperlink("Supprimer");
    final Hyperlink cellButtonEdit = new Hyperlink("Editer");
    final HBox hb = new HBox(cellButtonDelete, cellButtonEdit);

    ButtonCell(final TableView tblview) {
        hb.setSpacing(4);
        cellButtonDelete.setOnAction((ActionEvent t) -> {
            statusclic = "1";
            int row = getTableRow().getIndex();
            tableData.getSelectionModel().select(row);
            klikTableData(null);
            modelBiodata m = tableData.getSelectionModel().getSelectedItem();
            crudData.delete(m);
            affichageDonnees();
            clear();
            autoId();
            dialog(Alert.AlertType.INFORMATION, "Les données ont été bien supprimées");
            statusclic = "0";
            StatusCode = "0";
        });

        cellButtonEdit.setOnAction((ActionEvent event) -> {
            statusclic = "1";
            int row = getTableRow().getIndex();
            tableData.getSelectionModel().select(row);
            klikTableData(null);
            statusclic = "0";
        });
    }
```

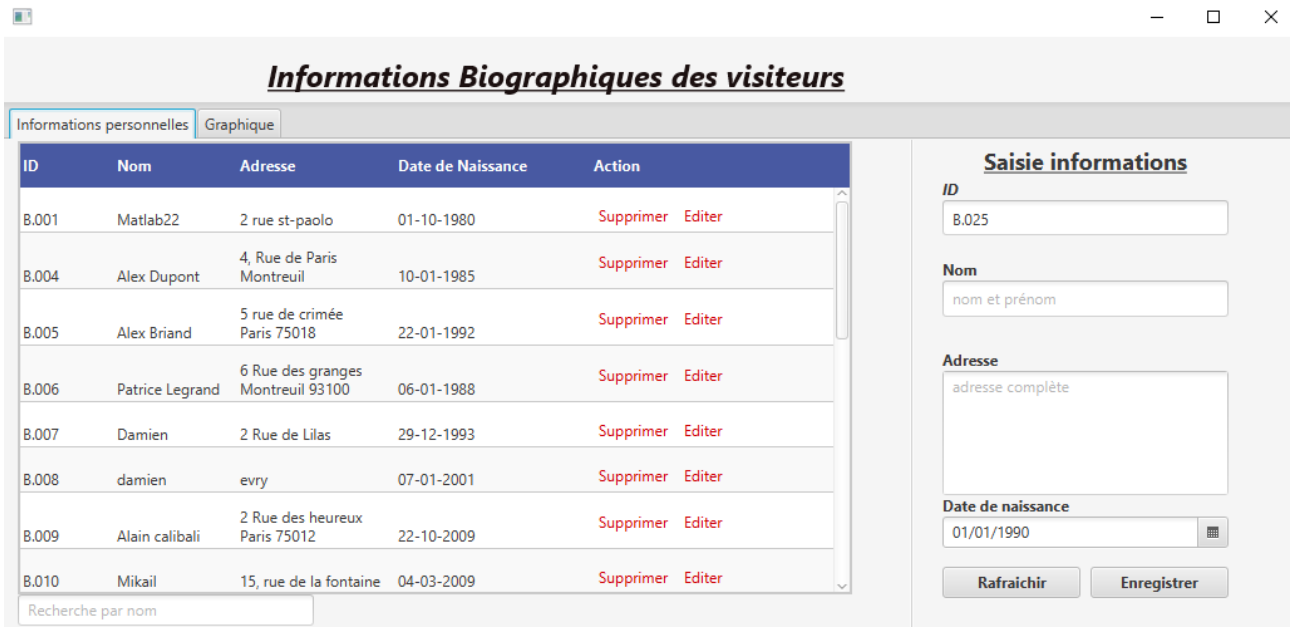
La méthode updateItem :

Elle prend en paramètre deux attribut de type boolean permet de mettre à jour les lien delete et edit.

Grâce à la méthode super de la classe TableCell si elle est vide alors on affiche le lien sinon on affiche pas.

```
    @Override
    protected void updateItem(Boolean t, boolean empty) {
        super.updateItem(t, empty);
        if (!empty) {
            setGraphic(hb);
        } else {
            setGraphic(null);
        }
    }
}
```

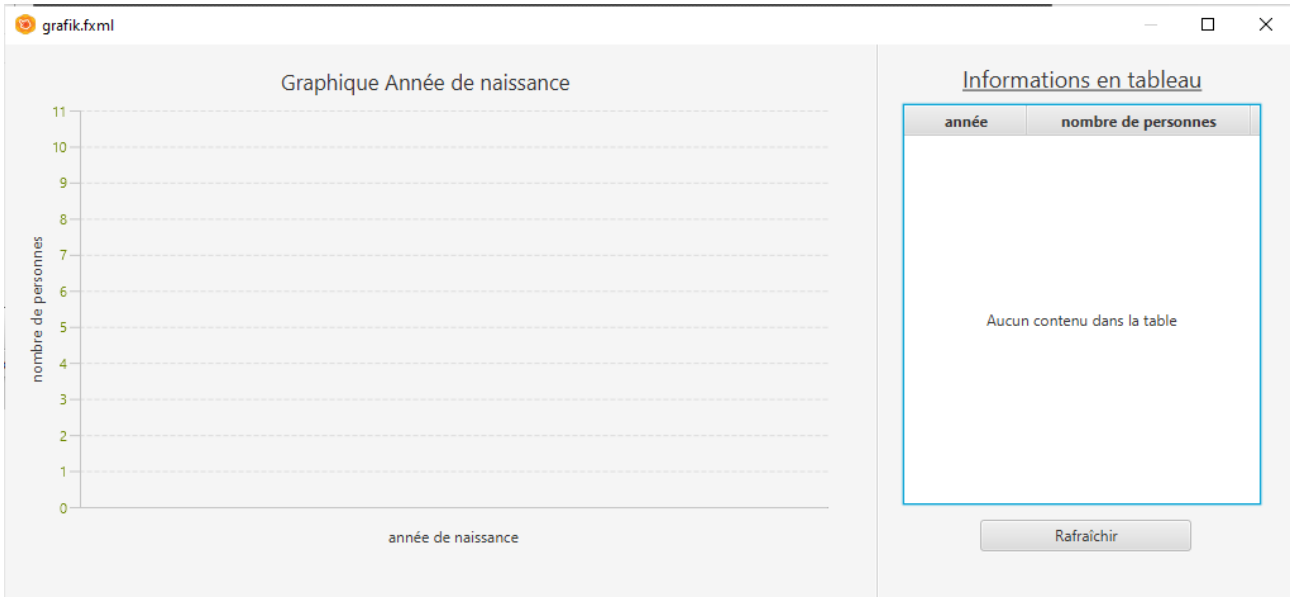
Resultat :



Création du graphique :

Partie 1 : La vue

A partir de la classe fxml login.fxml j'ai pu obtenir grâce au scripte du prof ce résultat :



Voici le scripte :

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.chart.*?>
<?import javafx.scene.text.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<AnchorPane prefHeight="437.0" prefWidth="1020.0"
xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1"
    fx:controller="controller.grafikController">
    <children>
        <Label layoutX="757.0" layoutY="14.0" text="Informations en tableau"
        textAlignment="CENTER" underline="true">
            <font>
                <Font size="18.0" />
            </font>
        </Label>
        <TableView fx:id="tableDetail" layoutX="710.0" layoutY="47.0"
        prefHeight="316.0" prefWidth="283.0">
            <columns>
                <TableColumn fx:id="colDetailAnnee" prefWidth="97.0" text="année" />
                <TableColumn fx:id="colDetailNombre" prefWidth="177.0" text="nombre de personnes" />
            </columns>
        </TableView>
        <StackedBarChart fx:id="bar" categoryGap="20.0" layoutY="13.0"
        prefHeight="399.0" prefWidth="664.0" title="Graphique Année de naissance">
            <xAxis>
                <CategoryAxis fx:id="barX" animated="false" label="année de naissance" side="BOTTOM"
                tickLabelFill="#3208da" tickLabelRotation="-45.0">
                    <tickLabelFont>
                        <Font size="11.0"/>
                    </tickLabelFont>
                </CategoryAxis>
            </xAxis>
            <yAxis>
                <NumberAxis fx:id="barY" label="nombre de personnes" minorTickCount="1" side="LEFT"
                tickLabelFill="#698300" upperBound="10.0">
                    <tickLabelFont>
                        <Font size="11.0" />
                    </tickLabelFont>
                </NumberAxis>
            </yAxis>
        </StackedBarChart>
        <Separator layoutX="688.0" orientation="VERTICAL" prefHeight="437.0" prefWidth="6.0" />
        <Button fx:id="btnRafraichir" layoutX="771.0" layoutY="375.0" mnemonicParsing="false"
        onAction="#rafraichir" prefHeight="25.0" prefWidth="167.0" text="Rafraîchir" />
    </children>
</AnchorPane>
```

Partie 2 : Le model

Pour le model on commence par importer toute les classe javafx

```
import javafx.beans.property.IntegerProperty;
import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property. SimpleStringProperty;
import javafx.beans.property.StringProperty;
```

On crée des attributs privée final ce qui permet de ne pas changer les valeurs plus tard puis on donne le type. Ses attributs représentes les colonnes annee et nombre de personne javafx.

```
public class modelGrafik {
    private final StringProperty annee;
    private final IntegerProperty nombredenoms;
```

```
public modelGrafik() {
    this.annee = new SimpleStringProperty();
    this.nombredenoms = new SimpleIntegerProperty();
}
```

On crée un constructeur avec des paramètres qui crée un nouvel objet StringProperty et un constructeur par défaut qui permet d'importer la classe sans ajouter des paramètres.

On crée ensuite les setters et le getters :

```
public String getAnnee() {
    return annee.get();
}

public void setAnnee (String value) {
    annee.set (value);
}

public StringProperty anneeProperty() {
    return annee;
}

public Integer getNombredenoms () {
    return nombredenoms.get();
}

public void setNombredenoms (Integer value) {
    nombredenoms.set(value);
}

public IntegerProperty nombredenomsProperty () {
    return nombredenoms;
}
```

Partie 3 : L'interface et l'implémentation

On crée deux méthode de type ObservableList qui permet d'afficher une liste qui change et on ajoute en paramètre la classe modelgrafik ce qui permet de récupérer toute les valeur de cette classe, et la classe Objet qui permet d'utiliser toutes les valeurs de la classe objet.

```
package interfaces;

import model.modelGrafik;
import javafx.collections.ObservableList;

public interface interGrafik {
    ObservableList<modelGrafik> getAnneDeNaissance();
    ObservableList<Object> anneedenaissanceToGrafik();
}
```

L'implémentation :

On commence par importer toutes les classes qui seront utile.

```
package implement;

import connexion.Connect;
import interfaces.interGrafik;
import model.modelGrafik;
import java.sql.ResultSet;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.scene.chart.XYChart;
```

On implémente ensuite la l'interface et on crée attribut con de Connect qui va nous permettre de crée une connection.

```
public class implGrafik implements interGrafik {
    Connect conn;
```

Les méthodes :

On implémente seulement les méthodes de l'interface.

La méthode getAnneDeNaissance :

Elle permet de définir dans le graphique la colonne année et nombre de personnes.

Pour ce faire on commence par crée une connection avec la classe connecte puis o crée une liste de type observablelist.

Dans un try catch on crée un requête qui permet de sélectionner les années de naissance distinctes et comptet le nombre de personnes nées chaque année et organise les résultats par année de naissance.

Puis tant qu'il y a une suite on insère le résultat de la requête dans le setter de année et setter de nombredenoms et on insère dans la liste qui est ensuite retourner.

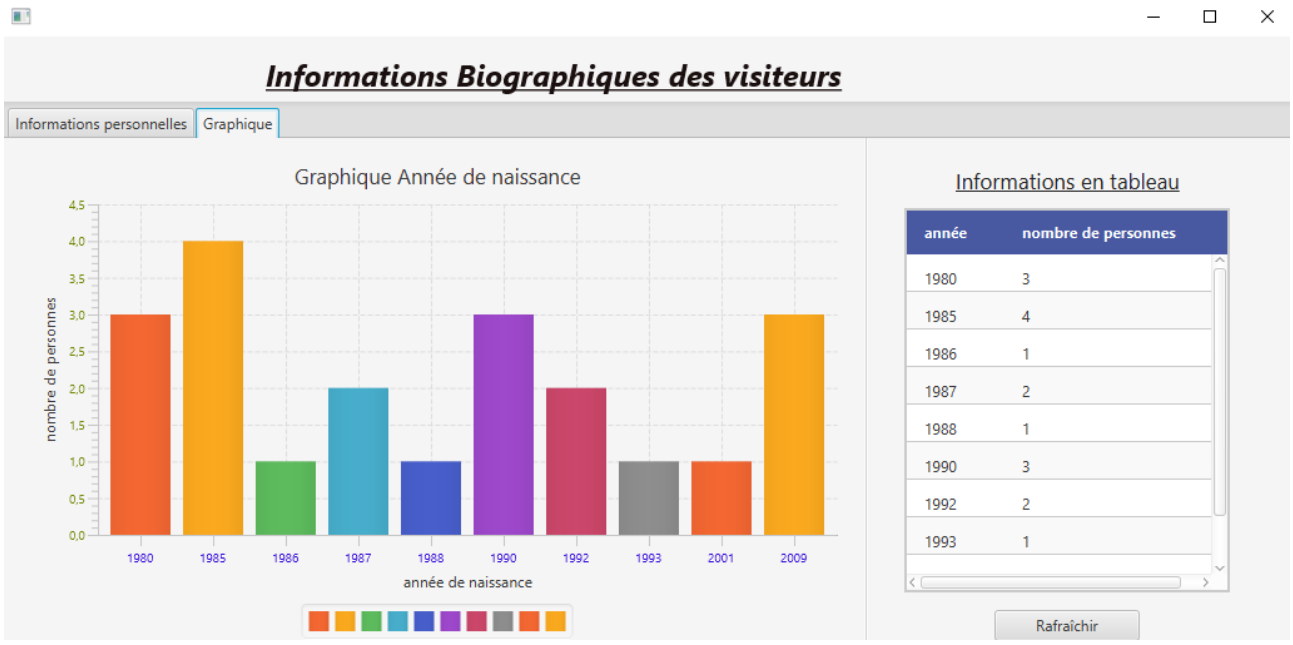
```
@Override
public ObservableList<modelGrafik> getAnneDeNaissance () {
    conn = new Connect();
    ObservableList<modelGrafik> listData = FXCollections.observableArrayList();
    try {
        String sql = "select distinct (extract(year from DateDeNaissance)) as annee,"
        +"count(nom) as nombredepersonnes"
        + "from tablebiodata "
        + "group by annee "
        + "order by annee";
    ResultSet rs = Connect.Connexion().createStatement().executeQuery(sql);
    while (rs.next()) {
        modelGrafik m = new modelGrafik();
        m.setAnnee (rs.getString("annee"));
        m.setNombredenoms (rs.getInt("nombredepersonnes"));
        listData.add(m);
    }
    } catch (Exception e) {
    }
    return listData;
}
```

La méthode anneedenaissanceToGrafik :

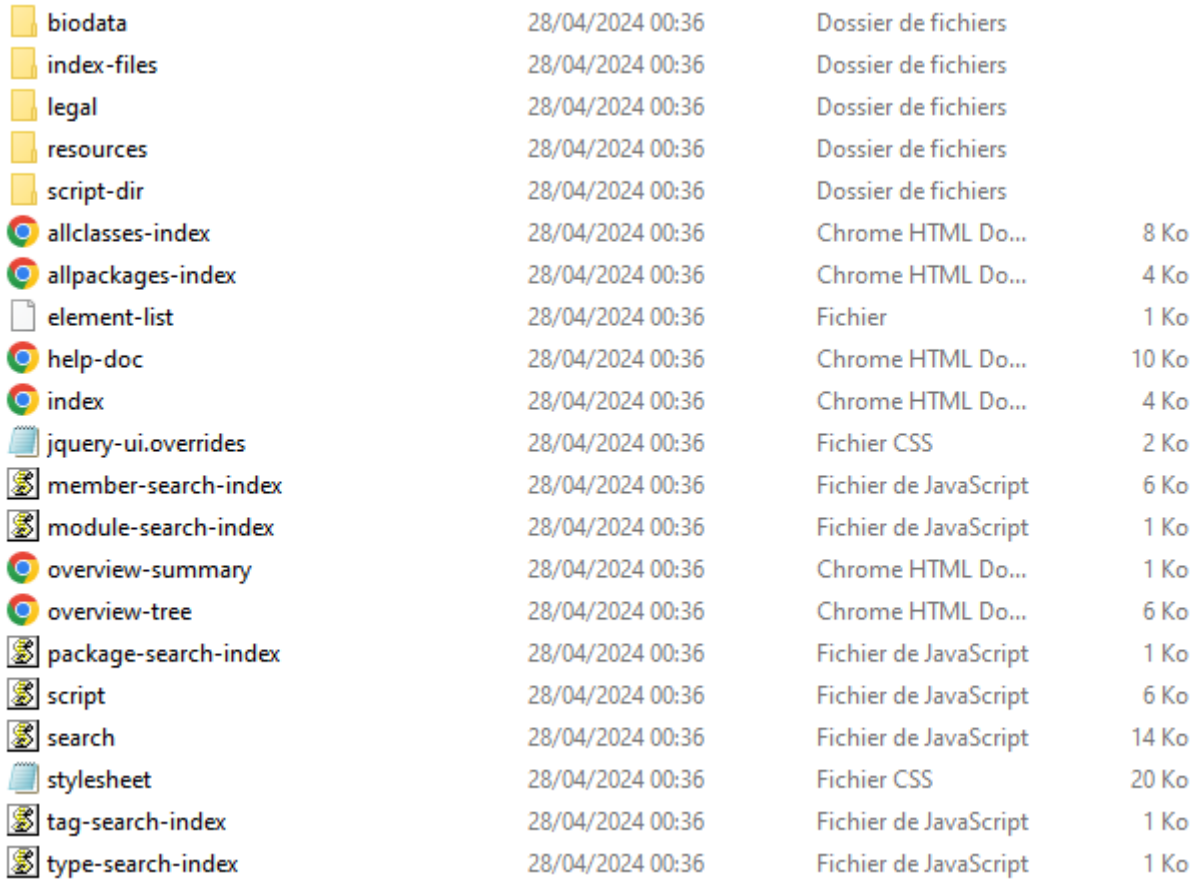
Cette récupère les années de naissance et le nombre de personnes correspondant à chaque année depuis la table "tablebiodata", puis crée une série de données pour un graphique à barres, représentant les années sur l'axe horizontal et le nombre de personnes sur l'axe vertical, avant de les ajouter à une liste observable.

```java
@Override
public ObservableList<Object> anneedenaissanceToGrafik() {
    ObservableList<Object> barCar = FXCollections.observableArrayList();
    try {
        conn = new Connect();
        String sql = "select distinct (extract (year from DateDeNaissance)) as annee,"
        + "count (nom) as nombredepersonnes "
        + "from tablebiodata "
        + "group by annee "
        + "order by annee";
        ResultSet rs = Connect.Connexion().createStatement ().executeQuery(sql);
        while (rs.next()) {
        XYChart.Series<String, Integer> aSeries = new XYChart.Series<>();
        aSeries.getData().add(new XYChart.Data(rs.getString("annee"), rs.getInt("nombredepersonnes")));

        barCar.add(aSeries);
        }
    } catch (Exception e) {
        }
    return barCar;
}
```

résultat :



javadoc :



biodata	28/04/2024 00:36	Dossier de fichiers	
index-files	28/04/2024 00:36	Dossier de fichiers	
legal	28/04/2024 00:36	Dossier de fichiers	
resources	28/04/2024 00:36	Dossier de fichiers	
script-dir	28/04/2024 00:36	Dossier de fichiers	
allclasses-index	28/04/2024 00:36	Chrome HTML Do...	8 Ko
allpackages-index	28/04/2024 00:36	Chrome HTML Do...	4 Ko
element-list	28/04/2024 00:36	Fichier	1 Ko
help-doc	28/04/2024 00:36	Chrome HTML Do...	10 Ko
index	28/04/2024 00:36	Chrome HTML Do...	4 Ko
jquery-ui.overrides	28/04/2024 00:36	Fichier CSS	2 Ko
member-search-index	28/04/2024 00:36	Fichier de JavaScript	6 Ko
module-search-index	28/04/2024 00:36	Fichier de JavaScript	1 Ko
overview-summary	28/04/2024 00:36	Chrome HTML Do...	1 Ko
overview-tree	28/04/2024 00:36	Chrome HTML Do...	6 Ko
package-search-index	28/04/2024 00:36	Fichier de JavaScript	1 Ko
script	28/04/2024 00:36	Fichier de JavaScript	6 Ko
search	28/04/2024 00:36	Fichier de JavaScript	14 Ko
stylesheet	28/04/2024 00:36	Fichier CSS	20 Ko
tag-search-index	28/04/2024 00:36	Fichier de JavaScript	1 Ko
type-search-index	28/04/2024 00:36	Fichier de JavaScript	1 Ko