## REPUBLIQUE DU SENEGAL



UN PEUPLE - UN BUT - UNE FOI

#### MINISTERE DE L'ENSEIGNEMENT SUPERIEUR, DE LA RECHERCHE

ET DE L'INNOVATION



7, Avenue FAIDHERBE BP 21354, DAKAR SENEGAL

TEL: 33 849 69 19 / FAX: 33 821 50 74

Département ESITEC (École Supérieure d'Informatique et des Technologies)



 ${f RAPPORT\ DE\ TRAVAIL}\ du\ projet\ bloggy$ 

<u>Sujet</u> : Création d'un mini-blog suivant l'architecture MVC en Java

## **PRESENTE PAR:**

## **SOUS LA DIRECTION DE :**

Cheikh Ahmadou Bamba Mbacké FALL

M. Alioune Kanouté

Année académique :2021-2022

## A. Description générale

*Bloggy* est une application monolithique mettant en avant les fonctionnalités d'un blog. Sur *bloggy*, les utilisateurs peuvent donc s'inscrire et poster des articles de tout type.

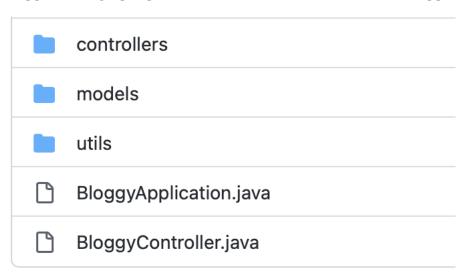
## B. Architecture du projet

*Bloggy* est conçu suivant l'architecture MVC, le projet comprend donc 3 parties essentielles : les Models, les Views et le Controllers.

- Model: contient les données à afficher/exploiter provenant de la base de données.
  Il est composé d'une multitude de classe ayant des méthodes publiques qui seront exploitées de l'extérieur par les Controllers
- **Controller**: contient la logique, les traitements à effectuer suivant l'action de l'utilisateur. Il est aussi composé de classes dont les méthodes font appel à celles des classes dans le Model
- **View** : contient la présentation de l'interface graphique. S'appuie sur les Controllers pour avoir des données à afficher

A cela s'ajoute la partie **DAO** qui contient toutes les opérations d'interaction directe avec la base de données (principalement la connexion). Il joue le rôle d'intermédiaire entre le Model et la base de données.

Bloggy a été implémenté dans le langage Java, utilisant la plateforme de création d'application graphiques JavaFX. Ci-dessous la structure de l'application :

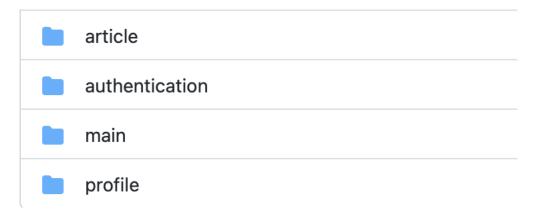


• **BloggyApplication.java** représente le point d'entrée de notre application, c'est le premier fichier qui est lu puis chargé. Il contient donc du code permettant d'ouvrir la toute première fenêtre de notre application

- Les packages (dossier) controllers et models représentent respectivement les Controllers et les Models de notre application comme défini plus haut
- Le package *utils* centralise toute opération répétitives et partagées entre les Controllers

A présent, faisons un focus sur chaque package séparément.

#### Controllers



Dans le package *article* nous avons la logique de toutes les opérations concernant un article à savoir *ajouter un article*, *lister tous les articles* 

Dans le package *authentication* nous avons la logique de toutes les opérations concernant l'authentification des utilisateurs à savoir *s'inscrire*, *se connecter* 

Dans le package *main* nous avons la logique de toutes les opérations concernant la page d'accueil de notre application à savoir : *naviguer entre les différentes pages* 

Dans le package *profile* nous avons la logique de toutes les opérations concernant le profil de l'utilisateur connecté à savoir *modifier son profil* 

#### **Models**



Le package *article* représente un article du blog matérialisé par une classe contenant les différents attributs d'un article à savoir *id, titre, contenu, posteur, date de publication* 

Le package *user* représente un utilisateur du blog matérialisé par une classe contenant les différents attributs d'un utilisateur à savoir *id, nom complet, nom d'utilisateur* et *mot de passe* 

### Views (ressources)



Sur JavaFX, les View sont tous mis dans le dossier nommé *ressources*. Il contient des fichiers **.fxml** représentants les différentes pages dont on a besoin.

Dans le package *article* nous avons regroupé toutes les pages concernant les articles à savoir *ajouter un article*, *lister tous les articles* 

Dans le package *authentication* nous avons regroupé toutes les pages concernant les utilisateurs à savoir *s'inscrire*, *se connecter* 

Dans le package *main* nous avons la page d'accueil de l'application, une fois que l'utilisateur est connecté : depuis cette page il peut naviguer entre les différentes autres pages

Dans le package *profile* nous avons regroupé toutes les pages concernant le profil de l'utilisateur connecté à savoir : *modifier son profil* 

<u>Utils</u>	
	Alerter.java
	Database.java

Ici nous retrouvons deux fichiers:

- *Alerter.java*: qui contient une classe qui nous permet d'afficher des notifications « popup » sur l'écran
- *Database.java* : qui est en fait notre DAO.

Bloggy exploite une base de données PostgreSQL *bloggy\_db*, contenant deux tables définies comme suit :

- Table *users*: contient les informations des utilisateurs (*id, nom complet, nom d'utilisateur, mot de passe, date d'inscription*)
- Table *articles* : contient les informations des articles créés sur bloggy (id, titre, contenu, posteur, date de publication)

# C. <u>Description fonctionnelle</u>

Pour expliquer le fonctionnement de bloggy, prenons l'exemple de deux scénarios :

#### « L'utilisateur veut se connecter »

- 1. Il renseigne ses informations de connexion sur la page *login-view.fxml*
- 2. Le Controller de connexion *LoginController* fait appel à sa méthode *handleLogin* avec comme paramètres les informations que l'utilisateur a renseignées
- 3. *handleLogin* s'appuie ensuite sur la méthode statique *authenticate* du model *BloggyUser* en lui passant les paramètres qu'il a reçu
- 4. *authenticate* passe par notre DAO (*Database.java*) pour communiquer avec la base de données et vérifier l'authenticité des informations que l'utilisateur a renseigné à travers une requête SQL
- 5. *authenticate* retourne le résultat de la requête SQL qu'il passe à *LoginController*
- 6. *LoginController* analyse le résultat de la requête et effectue l'action correspondante

### « L'utilisateur veut poster un article »

- 1. Il clique sur le bouton « *Post an Article* »
- 2. Le Controller de la page d'accueil *HomeController* charge la page correspondante (navigation) *post\_article-view.fxml*
- 3. L'utilisateur renseigne les informations du nouvel article puis clique sur le bouton « *Post it* »
- 4. Le Controller des articles *ArticleController* fait appel à sa méthode *saveArticle* qui crée une nouvelle instance du Model *BloggyArticle* et l'initialise avec les informations de l'article que l'utilisateur a renseignées
- 5. *saveArticle* fait ensuite appel à la méthode *createNew* du Model *BloggyArticle* en lui passant comme paramètre la nouvelle instance d'article créée
- 6. *createNew* exécute la requête de création d'article en s'appuyant sur notre DAO puis renvoie le résultat de la requête à *ArticleController*
- 7. *ArticleController* analyse le résultat de la requête pour afficher le message correspondant à l'utilisateur

Ceci est le fonctionnement général de bloggy qui illustre la parfaite interaction

View <-> Controller <-> Model <-> DAO <-> Database