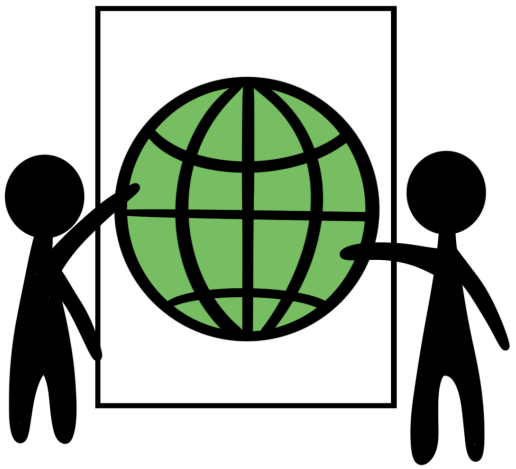
POLYTECH MONTPELLIER
IG4

SE Project

Togæther



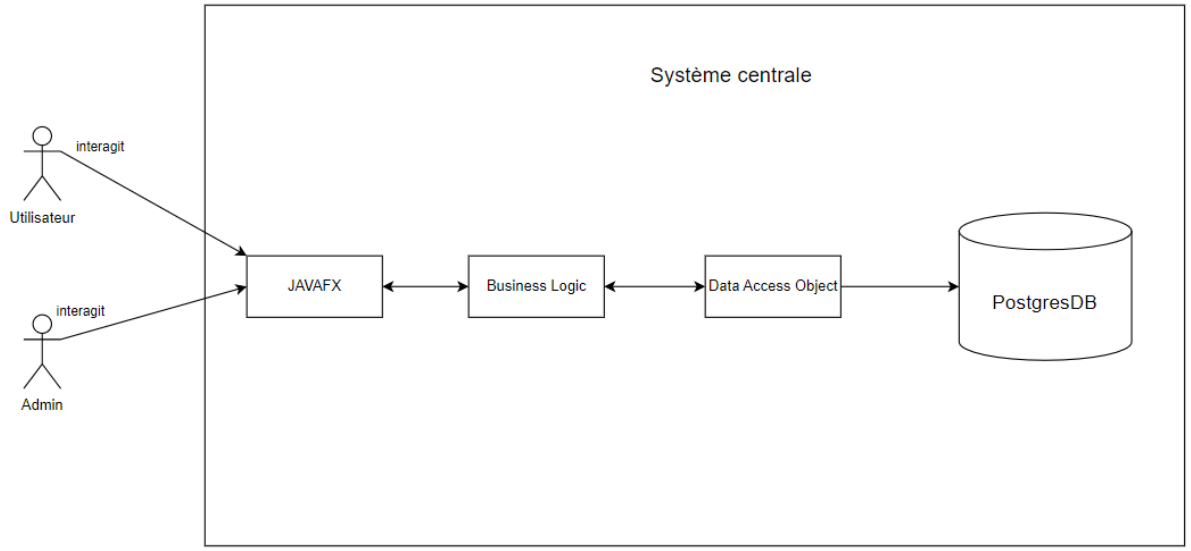
Prototype et Conception
Rapport de groupe

BENAITON Laura : Project manager & Documentation Responsable
CORREIA-MATEUS Dorian : BD Responsable
BOUCHEZ Loris : BD Connection
STEFANI Maxime : Conception Responsable

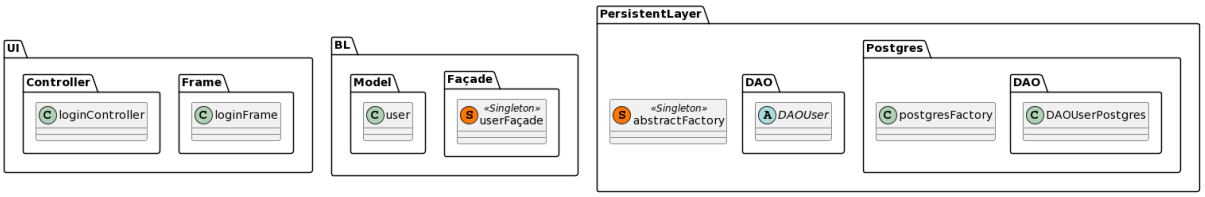
Table des matières :

1. Diagramme d’architecture global :

Un diagramme architectural est une **représentation visuelle qui cartographie l'implémentation physique des composants d'un système logiciel**. Il montre la structure générale du système logiciel et les associations, limites et frontières entre chaque élément.



2. Diagramme de package :



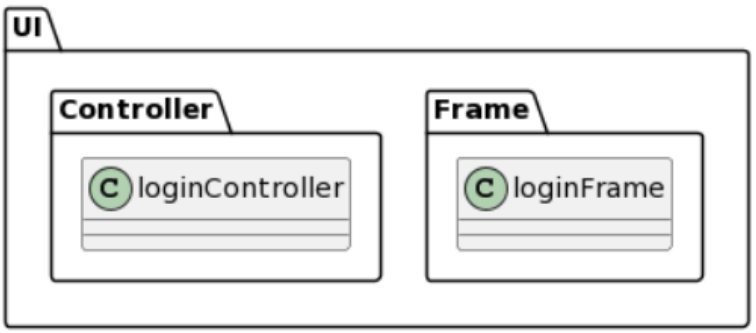
L'organisation des classes de l'application Togæther peut se décomposer en trois packages majeurs qui sont :

- Le package Interface Utilisateur
- Le package Business Logic
- Le package Persistance des données

2.1. Le package Interface Utilisateur (UI) :

Le package interface utilisateur contient toutes les classes qui serviront à manipuler la présentation et l'affichage des composants interactifs avec l'utilisateur.

Ce package est divisé en deux sous-packages qui sont Frame et Controller.
Frame contient les différentes classes d'affichage de fenêtres/scènes qui seront montrées à l'utilisateur et leurs différents composants.
Controller contient les différentes classes liées à ces mêmes fenêtres et manipule les composants des classes de Frame afin de modifier la fenêtre correspondante.



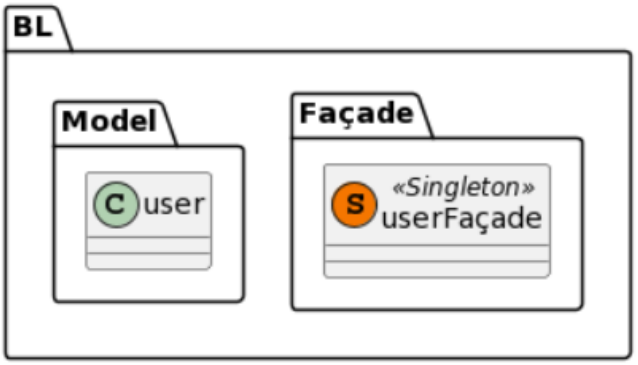
Dans cet exemple :

- loginFrame affiche les éléments de la page login (bouton valider, champs textuels pour email et mot de passe)
- loginController manipule les champs textuels et invoque une méthode spécifique quand le bouton est cliqué)

2.2. Le package Business Logic :

La logique métier est la partie du programme qui encode les règles métier du monde réel qui déterminent comment les données peuvent être créées, stockées et modifiées.

Le package Business Logic est composé de deux sous-packages qui sont Model et Façade.
Model contient toutes les classes de modèles qui seront utilisés pour personnaliser les pages selon chaque utilisateur.
Façade contient toutes les classes qui réunissent plusieurs fonctionnalités.



Dans l'exemple ci-dessus :

- Model contient la classe User qui contient toutes les informations d'un User (email, mot de passe, identifiant)
- Façade contient la classe UserFaçade qui fournit un certain nombre de méthodes comme login ou logout.

2.3 Le package Persistance des Données :

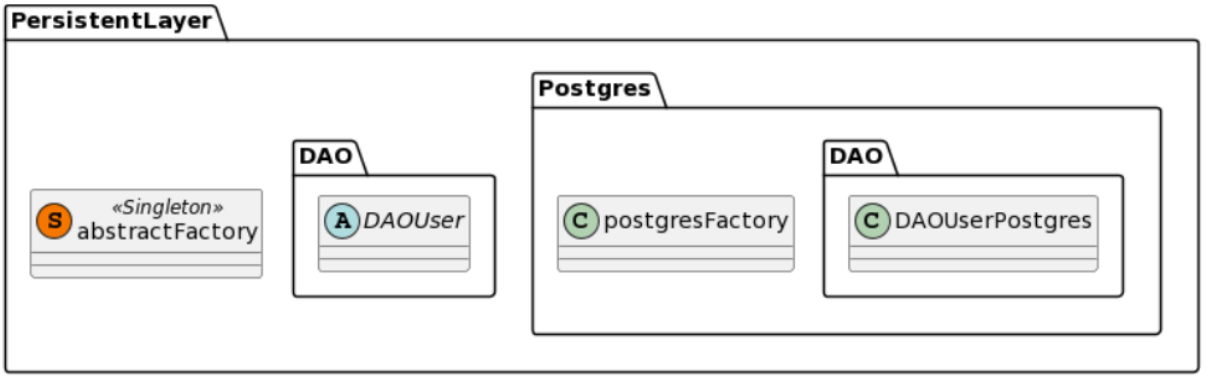
Le package Persistance des données contient toutes les classes liées à la lecture, modification, suppression, insertion de données dans une ou plusieurs base de données.

Il est composé d'une classe abstractFactory et de deux autres packages DAO et Postgres.

- La classe abstractFactory crée des factories propres à chaque type de base de données.
- DAO contient toutes les classes abstraites qui permettent l'accès à une table de la base de données, ceux sont des Data Access Object.
- Postgres est un package contenant une postgresFactory permettant la création de DAO spécifique à la base de données Postgres (héritant donc des DAO correspondant au package DAO supérieur dans la hiérarchie).

L'organisation du package Postgres est reproductible pour tout autre type de base de données. Si nous avions utilisé une autre base de données, par exemple ORACLE, nous

aurions reproduit un autre package ORACLE contenant une oracleFactory et un package DAO qui lui est propre.

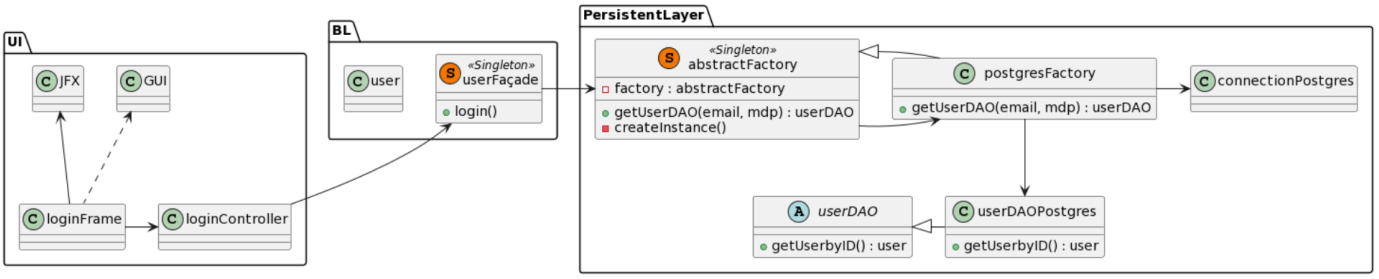


3. Diagramme de classes :

3.1. Use-case : login :

Le diagramme de classe correspondant au use-case login se trouve composé de trois types de classes :

- Des classes de présentations faisant partie du package UI
- Des classes de logique métier faisant partie du package BL
- Des classes de persistance de données/communication avec la base de donnée faisant partie du package PersistentLayer



Nous avons choisi de séparer les classes dans ces trois différents packages afin de dissocier les couches de Présentation, Manipulation/Calcul, et Sauvegarde/Lecture/Suppression/Modification des données.

Plusieurs Design Patterns sont utilisés :

- Singleton puisqu'il n'est pas nécessaire d'avoir des façades ou abstractFactory différentes.
- Façade puisque l'on souhaite réunir un panel de méthodes différentes pour un type d'objet défini (ici par exemple les Users)
- Factory afin de créer différents types de factory et ainsi différents types de DAO spécifiques à cette factory.

4. Diagramme de séquence :

4.1 Use-case : login :