Applivisiteurs

Application lourde de saisie de comptes-rendus par les visiteurs médicaux

Contexte:

GSB, Galaxy Swiss Bourdin, est une fusion entre deux grands laboratoires pharmaceutiques, Galaxy qui est un laboratoire américain visant les maladies virales et Swiss Bourdin qui est un conglomérat européen concepteur de médicament plus conventionnel.

Suite à leur fusion, une grande restructuration a été mise en place pendant deux ans tant au niveau du personnel qu'au niveau administratif. Maintenant, la priorité est de moderniser l'activité de visiteur médical.

Objectif:

Développer une application lourde permettant de saisir et de consulter des rapports de visiteurs médicaux.
L’application doit être conforme au cahier des charges.

Documentation réalisée:

ARBORESANCE DE L'APPLICATION APPLIVISITEUR

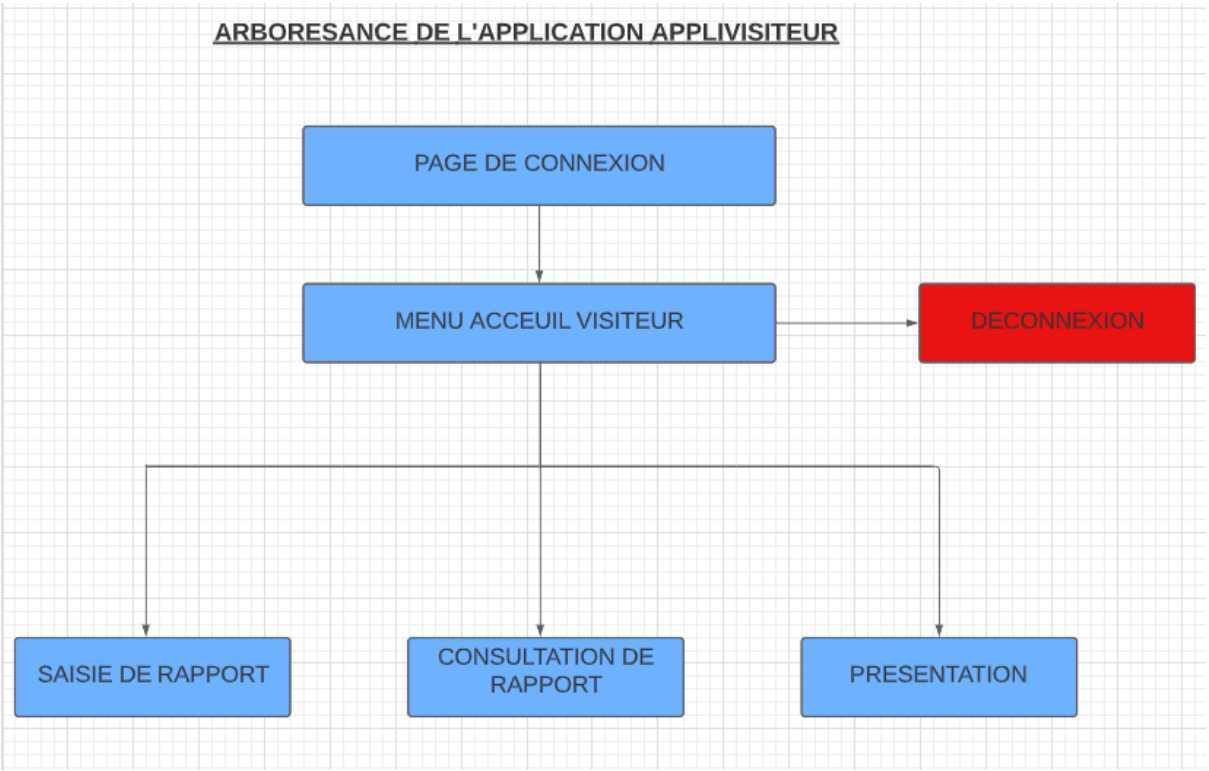
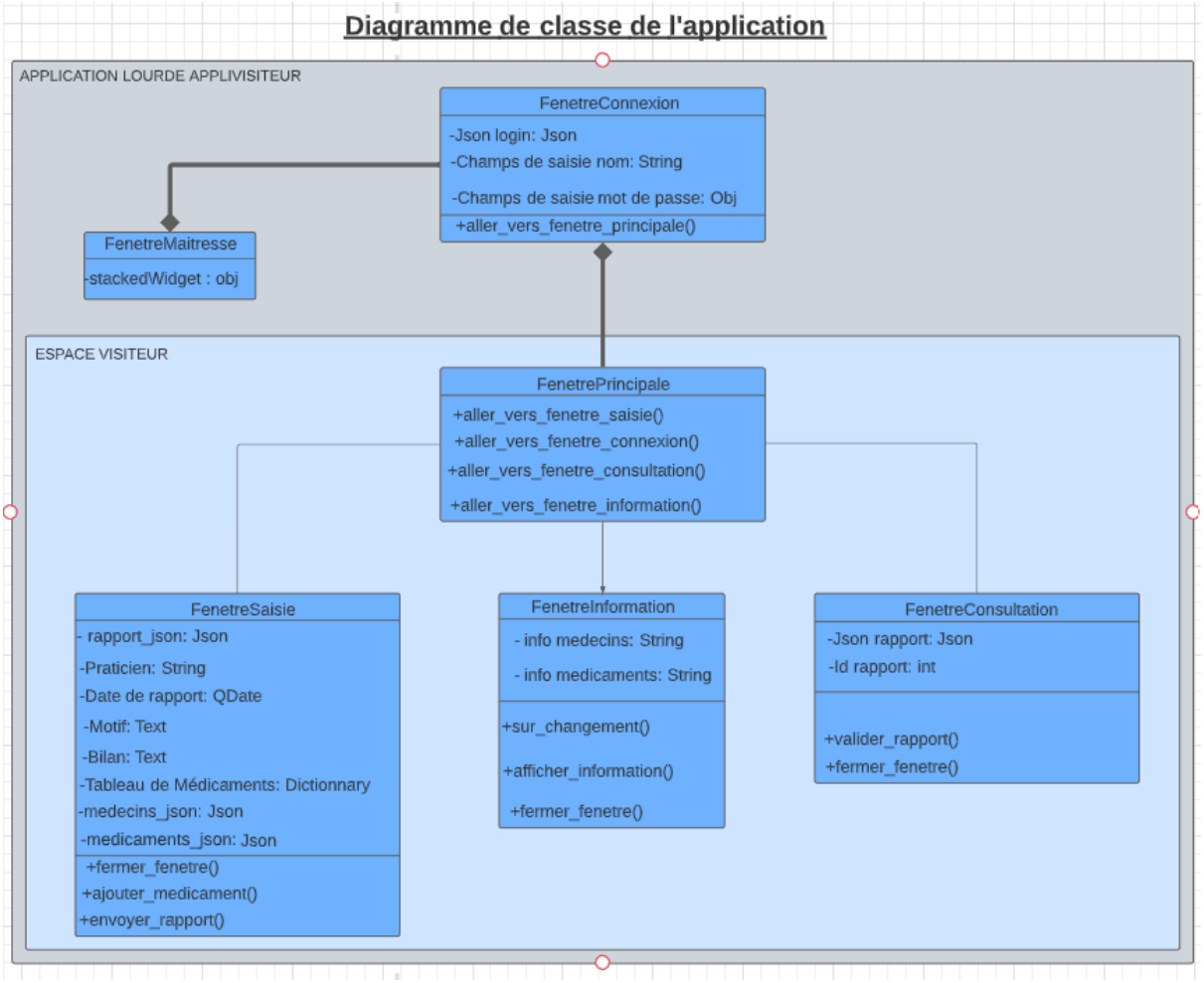


Diagramme de classe de l'application



UML de la base de données

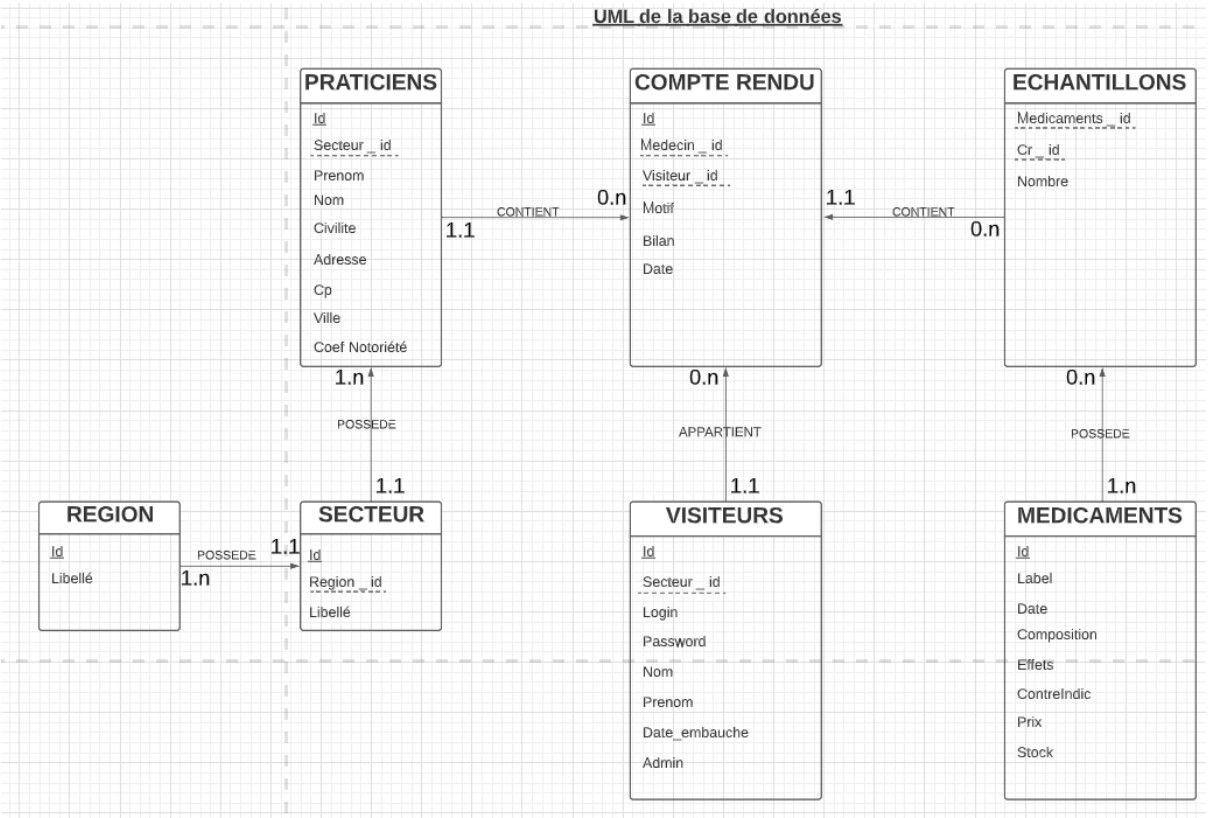
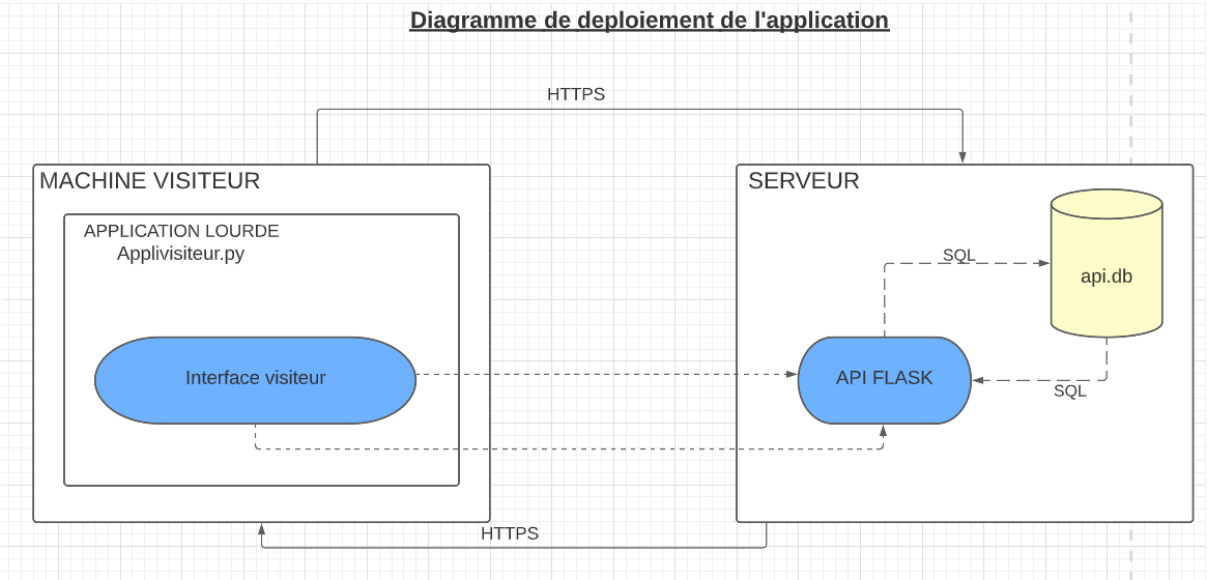
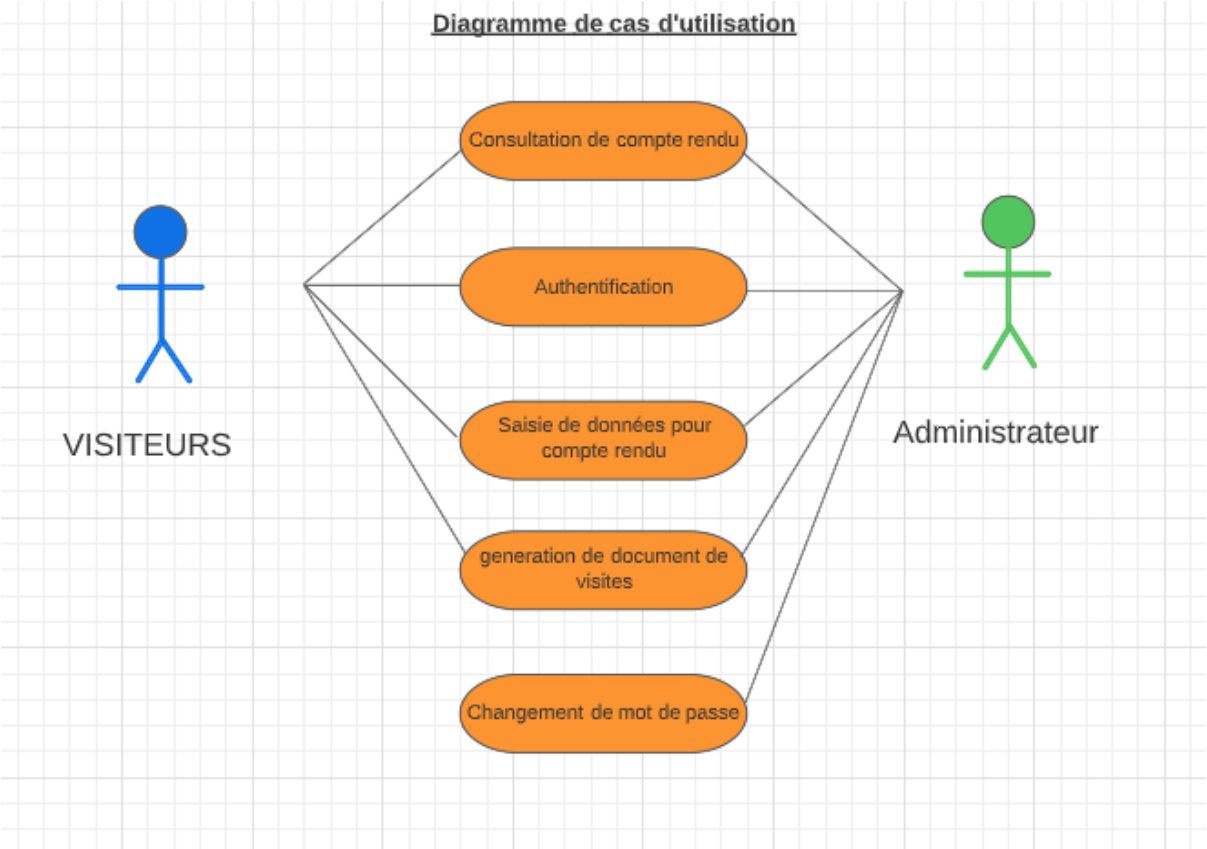


Diagramme de deploiement de l'application





Quelques frameworks et bibliothèques externes utilisés pour le projet

Frameworks :

Flask

- Serveur de développement (API)
- Documentation complète
- Multi-plateforme

- Interface utilisateur pour Python
- Multi-plateforme, stable, natif
- Documentation complète
- Facile d'utilisation

Bibliothèques :

Jsonify

Crée une réponse sous la représentation JSON

datetime

Permet de manipuler les dates et les heures

sys

Permet d'intéragir avec des commandes systèmes

requests

Permet d'utiliser le protocole HTTP de façon simple

Mise en place de l'organisation:

Pour s'organiser, nous avons :

- construit un schéma PERT (voir cloud).

- utilisé Trello pour faire des “To-Do list” (à faire, en cours, fait)
- créé un espace github pour versionner notre travail

Réalisation:

L’application respecte le paradigme MVC (modèle, vue, contrôleur) et est développée en Python.

Pour développer l’application lourde, nous avons utilisé l’outil de création d’interface QT Designer qui permet de construire une vue composée d’objets comme un champ de saisie, ou un libellé. Une fois construite, on peut extraire cette interface sous le format .ui qui est en réalité un fichier XML.

Une fois l’interface construite, il manque la partie logique qui permet de remplir l’interface avec les différentes données que me renvoie l’API sous le format JSON, c’est donc dans le fichier applivisteur.py que j’ai créé une class par fenêtre où je charge l’interface pour ensuite les modifier selon les méthodes de la classe.

Nous avons choisi le framework Flask pour l’API car il est propice au développement et bien documenté. Nous avions prévu de porter l’API sous Django qui est un framework plus professionnel et possédant plus d’outils pour sécuriser l’application, mais nous ne l’avons finalement pas fait par manque de temps.

Nous avons utilisé une base de données SQLITE pour notre démonstrateur. C’est une base locale, cela est une meilleur solution comparée à une base de donnée MYSQL

car la donnée est stockée sous un fichier facilement modifiable, partageable et ne nécessitant pas de serveur.

Pour organiser notre travail, nous avons utilisé Github qui permet de versionner notre projet et de développer des parties du projet sans empiéter sur le travail de quelqu'un d'autre.

Pour la répartition des tâches, nous avons utilisé l'application web Trello ainsi qu'un PERT.

Page de connexion de l'application

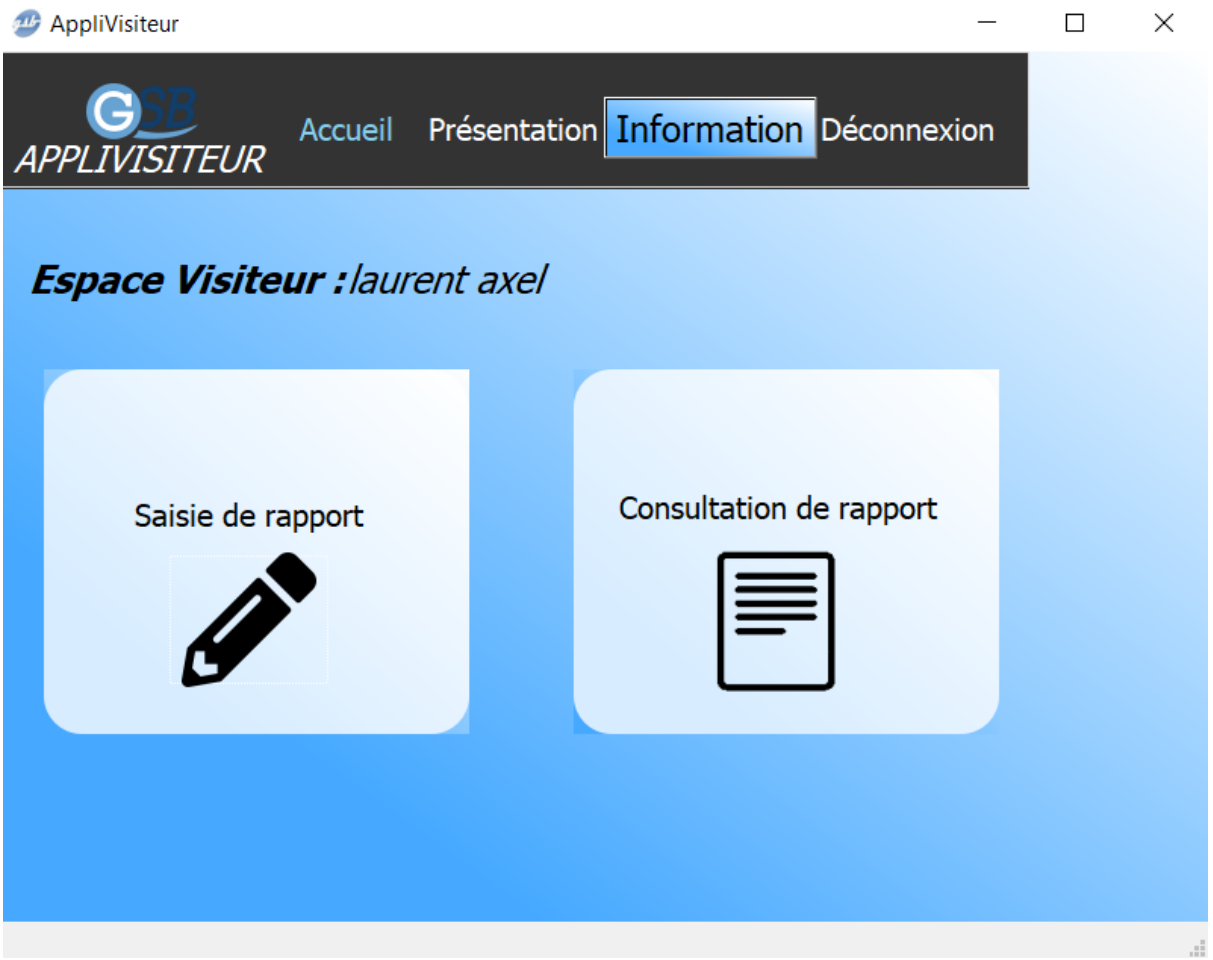
L'utilisateur va pouvoir se connecter en renseignant son nom d'utilisateur et mot de passe. Le logiciel va ensuite réaliser l'authentification via l'API. L'API communique

ensuite avec la base de données, tous les mots de passe sont hashés.

La méthode de hash est le MD5, on compare l'empreinte de la base de données avec le mot de passe hashé via ce code:

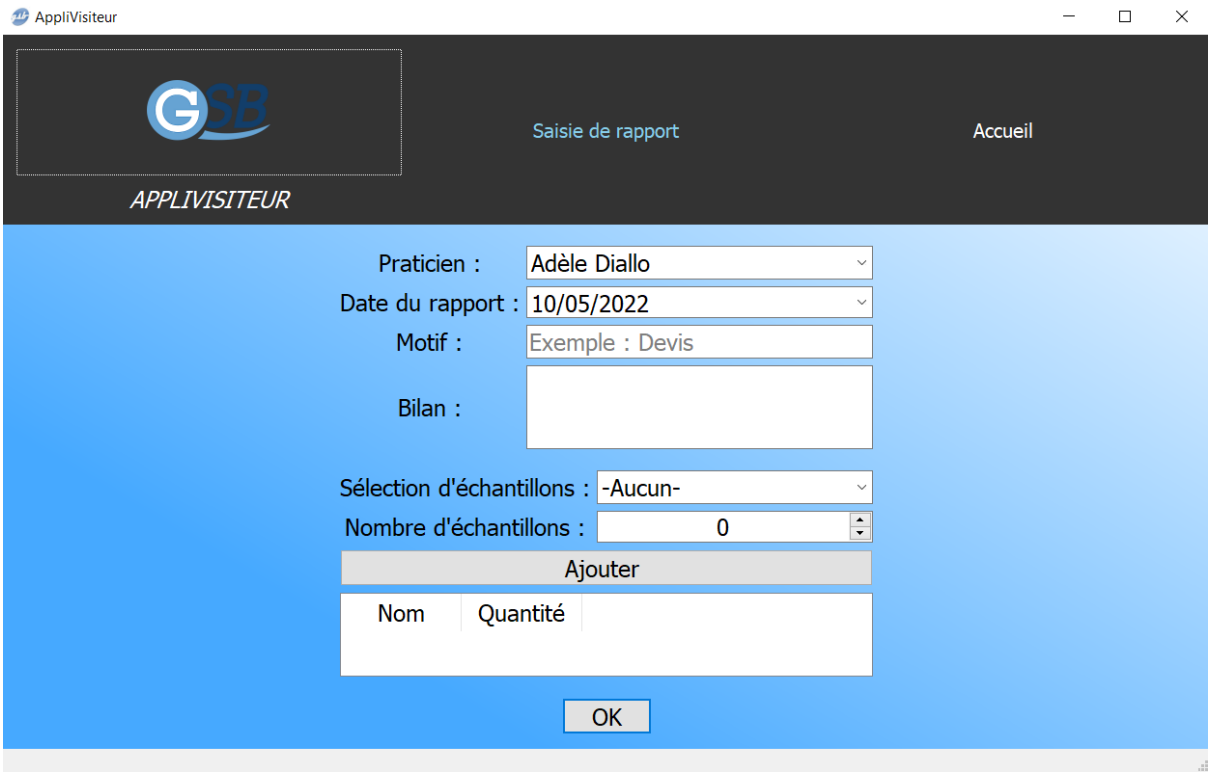
```
"""
Route utilisée pour s'authentifier auprès de l'API

Returns:
    HTTP CODE: 200, 401
"""
ressources = request.get_json()
res_____= query("SELECT password, id, Secteur_id, nom, prenom, admin FROM Visiteurs WHERE login = ? ", (ressources['login'],) )

password_____= ressources["password"]
hash_password = hashlib.md5(password.encode("utf-8")).hexdigest()
```

on récupère le mot de passe dans le json de connexion ressources["password"] pour le hashé grâce à hashlib.md5(password). On encode en ut8 pour éviter des problèmes de caractères mal lus. Comme le résultat du hash est un objet "HASH", on le transforme en chaîne de caractères avec hexdigest().



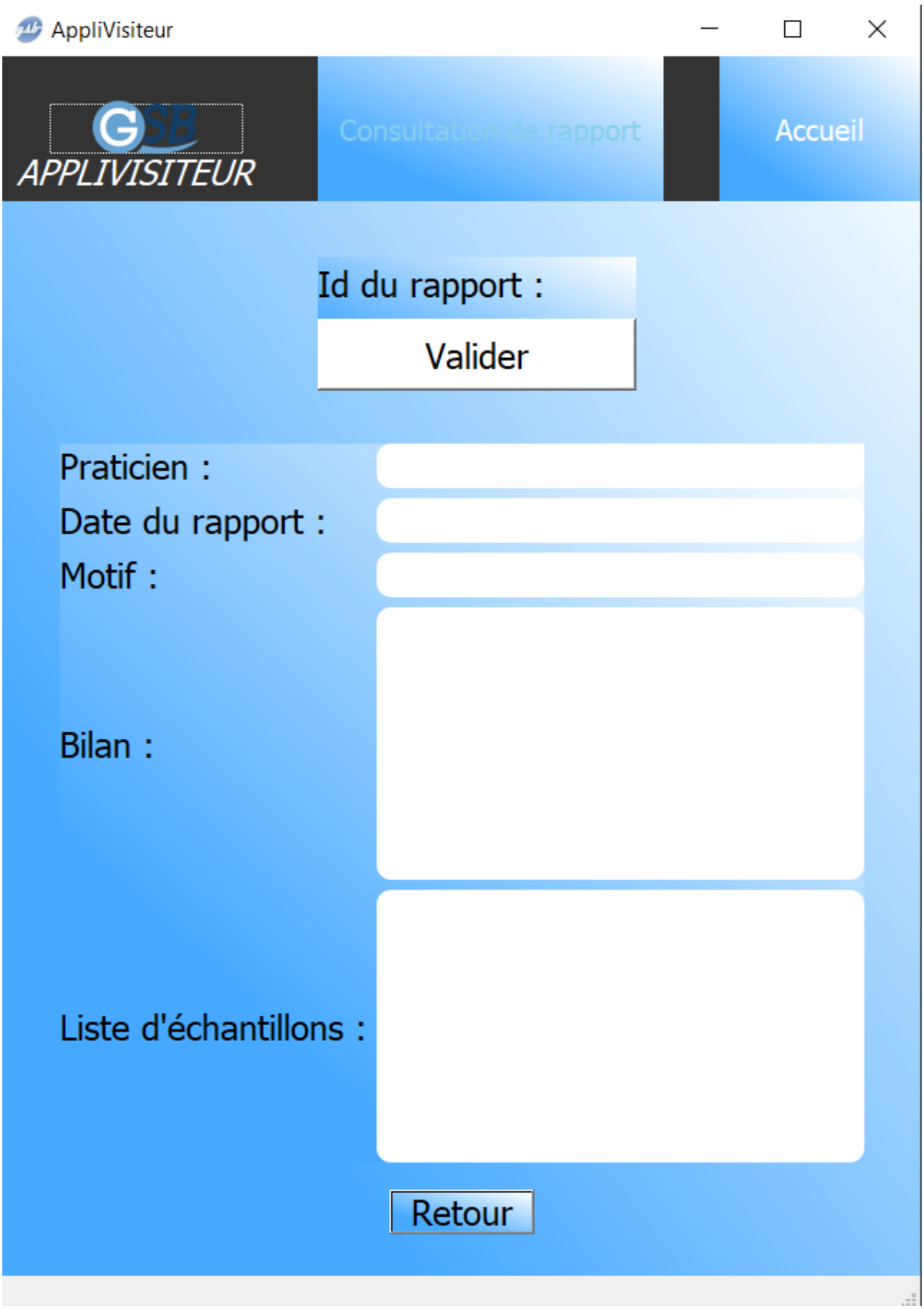
Menu principale de l'application

Une fois l'utilisateur authentifié et identifié, on l'amène au menu principal, ici il peut choisir entre saisir ou consulter un rapport, créer une fiche de présentation pdf pour sa visite ou alors consulter les informations des médicaments ou praticiens.



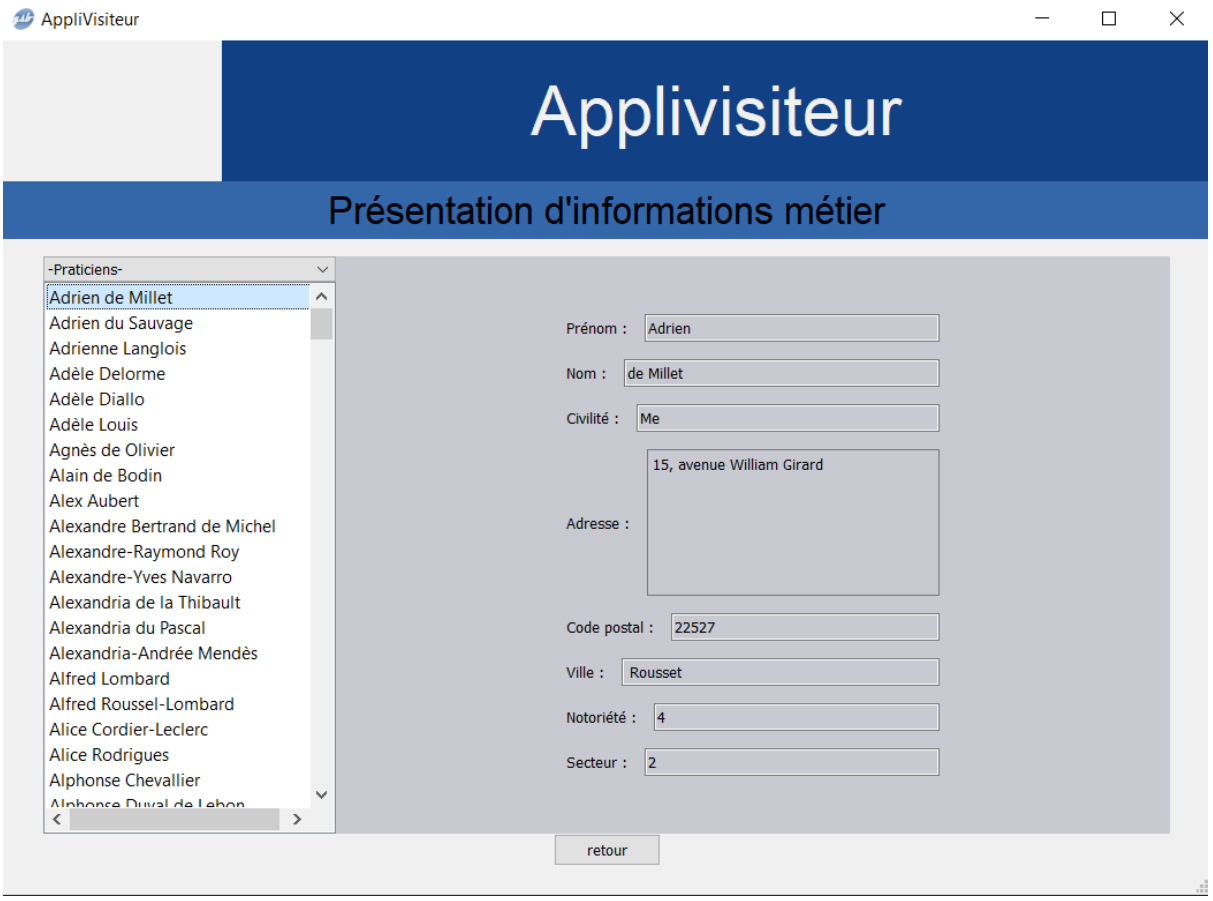
Page de saisie de rapport

On va pouvoir y saisir le rapport d'une visite, en y renseignant les informations du praticien, la date du rapport, le motif (conformément à la liste des motifs présents dans le cahier des charges), le bilan et les échantillons qui ont été présentés. Les praticiens disponibles sont limités au secteur attribué l'utilisateur, conformément au cahier des charges.



Page de consultation de rapport

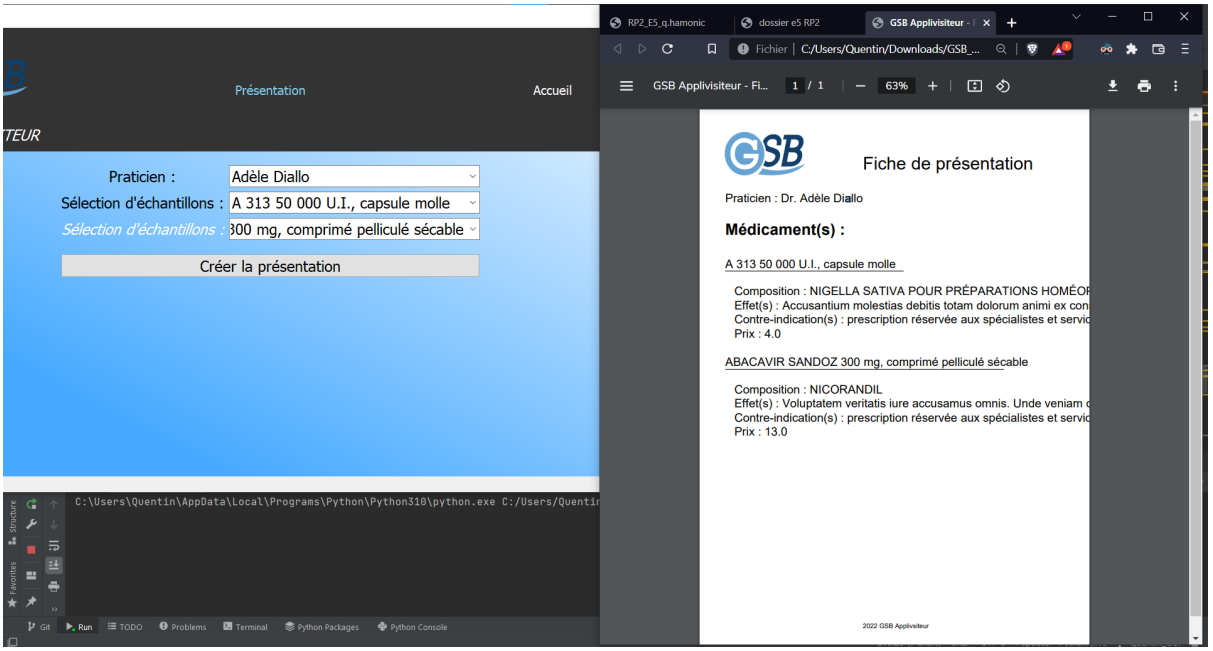
On va pouvoir consulter la liste des rapports nous concernant en le sélectionnant dans une liste déroulante. La consultation est restreinte aux rapports que l'utilisateur a saisis.



Page de consultation des informations métiers

Un écran pour consulter la liste des informations concernant les praticiens et médicaments.

L'API possède un système de log : à chaque fois qu'une route est consultée, une opération sur la base de données est faite qu'une erreur est levée, l'API l'historise.



Page de présentation ainsi que le PDF généré

En cas de visite, un onglet présentation permet de générer un PDF pour le visiteur contenant le nom du praticien visité ainsi que les produits présentés. On utilise la bibliothèque ReportLab pour la génération de PDF.

Evolution possibles:

- Sélectionner un rapport à l'aide d'une liste réduite au secteur du visiteur.
- Mot de passe oublié et impossibilité de le changer avec un des trois derniers mots de passe
- système d'envoi de mail pour des mot de passe oublié