Bloc 3 – Développement d'une solution digitale avec Python

1.1 Énumérez toutes les fonctionnalités attendues par le client sous forme d'User Story (agilité)

- L'administrateur veut un espace administrateur dédié avec un système d'authentification pour y accéder
- L'administrateur doit créer des produits (libellé, description, prix, image et catégorie)
- L'administrateur, doit également pouvoir ajouter des promotions à des produits en précisant la date de début, la date de fin et le pourcentage de remise.
- L'administrateur peut voir la liste des produits existants et les filtrer par catégorie

- L'utilisateur peut consulter la liste des produits existants et les filtrer par catégorie sur la page « catalogue » sans avoir à s'authentifier
- L'utilisateur doit pouvoir voir clairement les produits en promotion (prix en gras et rouge).
- Depuis la page catégorie l'utilisateur doit pouvoir voir l'ensemble des informations relatives au produit (libellé, description, prix, image et catégorie)

1.2 Quels sont les éléments que vous allez sécuriser ? Comment allez-vous procéder ?

Tout d'abord le système d'authentification pour les administrateurs, avec un système de token et de hachage de mot de passe.

Nous allons également sécuriser l'application contre de possible injections sql pour l'ensemble des champs input utilisateurs. Nous ferons cela en limitant les caractères que l'utilisateurs peut rentrer dans chaque champ.

Nous veillerons aussi à ne pas communiquer d'informations sensibles lors des messages d'erreurs dans l'environnement de production.

Nous vous recommandons également de garder systématiquement les bibliothèques et frameworks utilisés dans l'application, à jour avec les derniers correctifs de sécurité.

1.3 Évoquez les choix techniques que vous avez choisis concernant votre application et justifiez-les (tout en faisant référence au besoin client)

Le projet s'appuie sur le framework Flask. Plusieurs raisons pour ce choix : c'est un framework simple et rapide qui permet le développement en python de l'intégralité du back-end et une partie du front-end, combiné à Javascript, html et css. C'est l'un des frameworks les plus populaires avec Django et est open source, ce qui permet de s'appuyer sur une grande communauté et de nombreuses ressources. Il est distribué sous licence BSD (Berkeley Software Distribution), qui est une licence libre et non restrictive y compris pour les applications à but commerciales. Il permet de lancer rapidement le développement de son application et de travailler block par block.

PostgreSQL est un système de gestion de data base open-source réputé fiable, très populaire, qui évolue régulièrement face à sa grande communauté. Grace à ces nombreuses fonctionnalités il est plus puissant que MySQL et est une bonne solution pour gérer des problèmes complexes. C'est

également un système de gestion de base de données que nous connaissons déjà et utilisons sur d'autres projets. Il est gratuit et peut être utilisé pour des projets commerciaux.

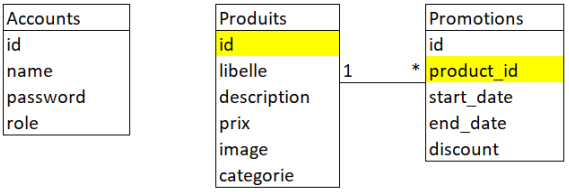
PythonAnywhere est une plateforme d'hébergement spécialisée dans python et ses frameworks comme Flask. Cela correspond donc parfaitement au projet. On pourra y héberger l'ensemble de l'application, de la base de données jusqu'au front. Nous travaillons déjà avec Pythonanywhere.

Pour l'aspect sécurité comme dis précédemment nous hacherons les mots de passe (Werkzeug) et créerons des tokens (jwt) avec des technologies que nous maitrisons déjà et qui ont fait leur preuves (populaires). C'est le mot de passe haché qui est sauvegardé dans la base de données et utilisé par comparaison lors de la demande de connexion d'un compte. Le token quant à lui est créée lors de la connexion au compte, il est personnalisé et à durée limitée. Il sera vérifié pour accéder à certain contenus/fonctionnalités.

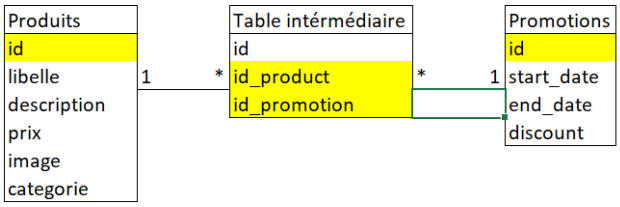
Par ailleurs pour communiquer avec la base de données, nous avons opté pour psycopg2. Psycopg2 est l'une des librairies python les plus populaires pour communiquer avec une base de données postgresql. En suivant sa documentation, l'un de ses avantages et que psycopg2 effectue lui-même le « sanitazing » des données inputs pour éviter les potentiels injections sql.

2.1 Développez la solution demandée par le client, pour ce faire vous devrez également :

- Produire le MCD ou MPD ou Diagramme de classe

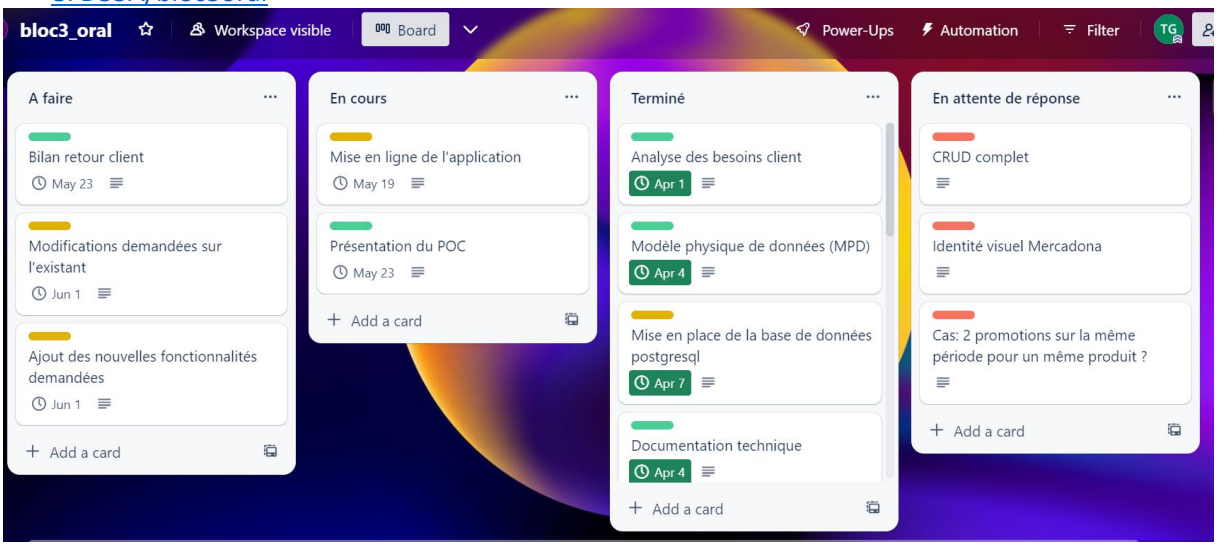


L'application, dans l'état actuel de la demande, nécessite une architecture des données assez simple.
Une table 'accounts' pour gérer les identifiants. Nous avons prévu un champ rôle au cas où il y aura besoin d'attribuer différentes permissions à l'avenir puisque nous sommes sur une application avec une partie back administrateur.
Une table 'produits' et une table 'promotions' pour la gestion respective de ces deux éléments. Il y a une clé étrangère dans la table promotions faisant référence à l'id du produit en question. Un produit peut avoir plusieurs promotions mais sur des intervalles de dates toujours différentes (logique mise en place dans les requêtes sql du backend). Une promotion, d'un point de vue de la base de données, ne correspond qu'à un seul produit. Dans l'application on peut rentrer simultanément 'la même promotion' pour plusieurs produits, mais dans la base elles seront toutes indépendantes.
On pourrait également imaginer une configuration avec une troisième table intermédiaire :



Avec cette configuration, cela matérialiserait dans la base de données, que plusieurs produits peuvent dépendre de la même promotion.

- Effectuez une gestion de projet
    - Adresse du Trello : https://trello.com/invite/b/nUnZ2ooX/ATTI9c5f509f12bdd954a45a25142d2149d8F58FD35A/bloc3oral



- Produire une documentation technique
    - Voir document ci-joint (intitulé ‘documentation_technique’)

- Produire un manuel d’utilisation
    - Voir document ci-joint (intitulé ‘manuel_utilisation’)

- Produire l’application
    - Adresse de l’application : http://tristanuser.pythonanywhere.com
    - Adresse du github : https://github.com/TWR-RWT/bloc3_oral