



Ecole Polytechnique de l'Université de Tours

Département Informatique

64 avenue Jean Portalis

37200 Tours, France

Tél. +33 (0)2 47 36 14 14

polytech.univ-tours.fr

**Projet Recherche & Développement
2019-2020**

**Une IA en Web service pour la
connaissances d'image**

Tuteur académique
Barthelemy SERRES

Étudiant
Miyuan SONG (DI5)

12 avril 2020



Liste des intervenants

Nom	Email	Qualité
Miyuan SONG	miyuan.song@etu.univ-tours.fr	Étudiant DI5
Barthelemy SERRES	barthelemy.serres@univ-tours.fr	Tuteur académique, Département Informatique



Avertissement

Ce document a été rédigé par Miyuan Song surnommé l'auteur.

L'Ecole Polytechnique de l'Université de Tours est représentée par Barthelemy Serres surnommé le tuteur académique.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnaît assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assume l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnaît qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable du tuteur académique et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.



Pour citer ce document

Miyuan Song, *Une IA en Web service pour la connaissances d'image*, Projet Recherche & Développement, Ecole Polytechnique de l'Université de Tours, Tours, France, 2019-2020.

```
@mastersthesis{
  author={Song, Miyuan},
  title={Une IA en Web service pour la connaissances d'image},
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université de Tours},
  address={Tours, France},
  year={2019-2020}
}
```

Table des matières

Liste des intervenants	a
Avertissement	b
Pour citer ce document	c
Table des matières	i
Table des figures	v
1 Introduction	1
1 Contexte de la réalisation et acteurs	1
2 Objectifs.....	1
3 Hypothèse	1
4 Bases méthodologiques.....	2
2 Description générale	3
1 Environnement du projet	3
2 Caractéristiques des utilisateurs.....	3
3 Fonctionnalités du système	3
4 Structure générale du système.....	5
3 Etat de l'art	8
1 Plateforme de déploiement	8
1.1 Cloud Foundry	8
1.2 Docker.....	10
2 Cadre d'Algorithmes d'IA	12
2.1 H2O	12

2.2	TensorFlow	13
2.3	Api d'IA	13
4	Analyse et conception	14
1	Compréhension de la plateforme serveur	14
1.1	Amazon EC2	14
1.2	Images Docker	14
2	Les composants du système	14
5	Mise en oeuvre	16
1	Test de faisabilité.....	16
1.1	Test du framework Django.....	16
1.2	Test de déploiement de Docker.....	17
2	Implémentation	20
2.1	Technologies	20
2.2	Construction d'architecture.....	20
2.3	Implémentation de la base de données.....	20
2.4	Implémentation front-end	21
2.5	Implémentation de l'API.....	22
2.6	Implémentation de la logique métier	22
2.7	Test local.....	22
2.8	Déploiement de mise en ligne	23
3	Amélioration.....	24
3.1	Amélioration de la remarque de reconnaissance d'image	24
3.2	Mécanisme de session.....	24
3.3	Positionnement du résultat de retour	25
3.4	Correction de bug.....	25
4	Contrôle de version	25
4.1	Contrôle de version Github	25
4.2	Contrôle de version Dockerhub.....	26
6	Conclusion	27
1	Bilan du semestre 9	27
2	Planning du semestre 10	28
3	Bilan du semestre 10.....	28
4	Bilan auto-critique.....	31
	Annexes	32
A	Cahier de test	33
1	Typologie de test	33

2	Tests unitaires	33
2.1	Tests unitaires de « Api-upload»	33
2.2	Tests unitaires de «Api-download»	34
2.3	Tests unitaires de « Api-database»	35
2.4	Tests unitaires de « Api-baidu-image»	35
3	Test fonctionnel	36
3.1	Test fonctionnel de «Authentification»	36
3.2	Test fonctionnel de «Index»	37
3.3	Test fonctionnel de «Reconnaissance d'image»	38
3.4	Test fonctionnel de «Gestion des utilisateurs»	39
3.5	Test fonctionnel de «Gestion de site Web»	40
B	Cahier de développeur	42
1	Introduction	42
2	Aperçu de la diagramme de classe	42
2.1	Diagramme de classe en branche de webservice-app-new	43
2.2	Diagramme de classe en branche de webservice-api	45
3	Structure de projet	52
4	Processus de développement	53
4.1	Développement de services Web	53
4.2	Écrire un dockerfile	54
4.3	Créer un docker image	55
4.4	Exécuter en serveur	55
C	Document d'utilisation	57
1	Installation et configuration	57
1.1	Prérequis	57
1.2	Démarrer le portainer	57
1.3	Démarrer le service	58
2	Guide d'utilisation général	59
2.1	Démarche	59
2.2	Log in et Log out	59
2.3	Reconnaissance d'image	60
2.4	Administration	60
2.5	Gestion de site Web	61
3	Guide du développeur	62
D	Gestion de projet	63
1	Aperçu de gestion de projet	63
1.1	Découpage des tâches	63

2	Gestion des aléas	65
2.1	Le projet ne peut pas être achevé à temps.....	65
2.2	Progrès en retard à un certain stade	65
2.3	Les livrables fournis ne répondent pas aux exigences	66
E	Description des interfaces externes du système	67
1	Interfaces matériel/logiciel	67
2	Interfaces homme/machine.....	68
2.1	Page de login.....	68
2.2	Page d'accueil.....	69
2.3	Page de service	69
2.4	Page d'administration.....	70
2.5	Page de gestion de web site.....	70
F	Spécifications fonctionnelles	71
1	Fonction de Authentification	71
2	Fonction de Upload.....	71
3	Fonction de Choix de service.....	72
4	Fonction de reconnaissance d'image	72
5	Fonction de Download	72
G	Spécifications non fonctionnelles	73
1	Contraintes de développement et conception	73
2	Contraintes de fonctionnement et d'exploitation.....	73
2.1	Performances	73
2.2	Contrôlabilité	73
2.3	Sécurité	73
H	Comptes rendus	74
I	Bibliographie	79

Table des figures

2 Description générale

1	– Diagramme de cas d'utilisation	4
2	– Diagramme d'activité	5
3	– Diagramme d'architecture	6
4	– Diagramme de déploiement	7

3 Etat de l'art

1	– Cloud Foundry.....	8
2	–Pivotal.....	9
3	–IBM CF.....	10
4	–dockerhub	11
5	–Portainer	12

5 Mise en oeuvre

1	–App_helloworld	16
2	–Page d'App_helloworld.....	17
3	–Dockerfile.....	18
4	–L'apprentissage Docker	19
5	–Test de Docker	19
6	–La base de données	21
7	–Front-end.....	21
8	–Fichier statique.....	22
9	–Docker Desktop.....	23
10	–AWS EC2	23

11	–Xshell	24
12	–Remark.....	24
13	–SSession.....	24
14	–Branche de Github.....	25
15	–Version sur Dockerhub	26
16	–Build sur Dockerhub	26
 6 Conclusion		
1	–Les tâche de S9 sur Trello 15.....	27
2	–Carte du cerveau 1.....	29
3	–Carte du cerveau 2.....	29
4	–Carte du cerveau 3.....	30
5	–Carte du cerveau 4.....	30
 A Cahier de test		
1	– Les Types de test.....	33
2	– Api-upload :Résultat	34
3	– Api-download :Résultat	35
4	– Api-database :Résultat	35
5	– Api-baidu-image :Résultat	36
6	– Authentification : Résultat	37
7	– Index : Résultat	38
8	– Reconnaissance d'image : Upload	39
9	– Reconnaissance d'image : Affichage des résultats	39
10	– Gestion des utilisateurs : Résultat	40
11	– Gestion de site Web : Résultat	40
12	– Gestion de site Web : Gestion de log	41
 B Cahier de développeur		
1	– Diagramme de classe en branche de webservice-app-new	43
2	– Diagramme de classe App1-login-home	44
3	– Diagramme de classe App2-image-rec	45
4	– Diagramme de classe Api-upload	46
5	– Diagramme de classe Api-download	47
6	– Diagramme de classe Api-database	49
7	– Diagramme de classe Api-baidu-image	51
8	– Diagramme de Structure	52
9	– Diagramme de pydoc	53

10	– Processus de développement	53
11	– webservice-template	54
12	– dockerfile	54
13	– dockerhub	55
14	– docker-compose	56
 C Document d'utilisation		
1	– docker-compose up.....	58
2	– Des règle d'aws	59
3	– Reconnaissance d'image	60
4	– Administration.....	61
5	– Gestion de site Web	61
 D Gestion de projet		
1	La diagramme Gantt du planning	63
 E Description des interfaces externes du système		
1	– Diagramme de déploiement	67
2	–Page de login	68
3	– Page d'accueil.....	69
4	– Page de service	69
5	– Page d'administration	70
6	–Page de gestion de web site	70
 I Bibliographie		
1	– Docker	80
2	– Diagramme de structure	80
3	– Image de docker	80
4	– Docker	81
5	– Diagramme de structure	81
6	– Image de docker	81

1

Introduction

Ce document est une spécification système pour un projet de recherche et développement pour le sujet de « Une IA en Web service pour la connaissances d'images » .

Ce projet a été choisi par l'élève Miyuan Song et est encadré par monsieur Barthelemy Serres.

1 Contexte de la réalisation et acteurs

Avec le développement de la technologie d'intelligence artificielle, de plus en plus d'algorithmes et de cadres de mise en oeuvre sont en open source. Cependant, l'utilisation de ces technologies nécessite une grande expertise en informatique. Nous avons constaté que l'utilisation de Web services est un excellent moyen pour mise en place ces ressources.

Par conséquent, afin de faciliter l'utilisation de ces ressources open source et d'effectuer des recherches connexes, ce projet portera sur l'exploration d'un moyen de déployer rapidement des Web services liés à l'IA. L'accès à ce serveur se devra d'être simple et permettra aux utilisateurs de pouvoir mettre en oeuvre les bonnes pratiques de domaine d'AI.

2 Objectifs

Ce projet contient deux objectifs distincts ils ciblent deux types d'utilisateurs : le developpeur et le utilisateur non informaticien.

Pour developpeur : Il peut deployer ses services web facilement, et il peut aussi monitorer leur etat au cours du temps (en ligne, en pause, arrete etc.)

Pour un utilisateur non informaticien : Il peut utiliser des services système tels que la reconnaissance d'image.

Pour un administrateur, il peut gérer les utilisateurs et les sites Web.

3 Hypothèse

Lors du déploiement de Web service, je vais explorer l'utilisation de PaaS (Platform as a Service) ou Docker. Je vais utiliser du code open source connu.

Si ce n'est pas faisable, j'utiliserai d'autres méthodes pour déployer, telles que le déploiement dans une machine virtuelle.

4 Bases méthodologiques

- **Les outils :**
 - Pour le contrôle de version, je utilise Github et Dockerhub ;
 - Pour la gestion de projet, je utilise la plateforme « Trello » pour gérer les tâches et diagramme de Gantt pour contrôler le déroulement du projet ;
 - Pour la modélisation logicielle, je utilise les diagrammes d'UML, diagramme de cas d'utilisation, diagramme d'architecture et diagramme de déploiement ;
 - Pour la modélisation mathématique et la rédaction de rapport, je utilise « LaTeX ».
- **La méthode de gestion de projet :**
 - Les détails des tâches spécifiques sont décrite dans l'annexe « Planification ».

2

Description générale

Dans ce chapitre, on décrit l'environnement du projet, les caractéristiques des utilisateur, les fonctionnalités du système, et la structure générale du système.

1 Environnement du projet

Je vais explorer la possibilité de déployer le serveur sur la plate-forme PaaS(cloud foundry) ou Docker. PaaS extrait les détails du matériel et du système d'exploitation, cela simplifiera la configuration de l'environnement.

Pour le langage de programmation, je vais utiliser Python 3.7. Et l'IDE pour le développement est PyCharm.

Pour l'implémentation de reconnaissance d'image, j'utilise l'API Baidu AI comme interface.

2 Caractéristiques des utilisateurs

On a trois types d'utilisateurs, ils ont des autorisations différentes :

- **Les utilisateur non informaticiens** : Ils peuvent facilement manipuler l'image à l'aide d'intelligence artificielle dépendants du système.
- **Les administrateurs** : Ils peuvent gérer les utilisateurs et les sites Web.
- **Les developpeurs** : ils peuvent deployer ses services web,et ils peuvent gérer ces services web.

3 Fonctionnalités du système

Le diagramme de cas d'utilisation est :

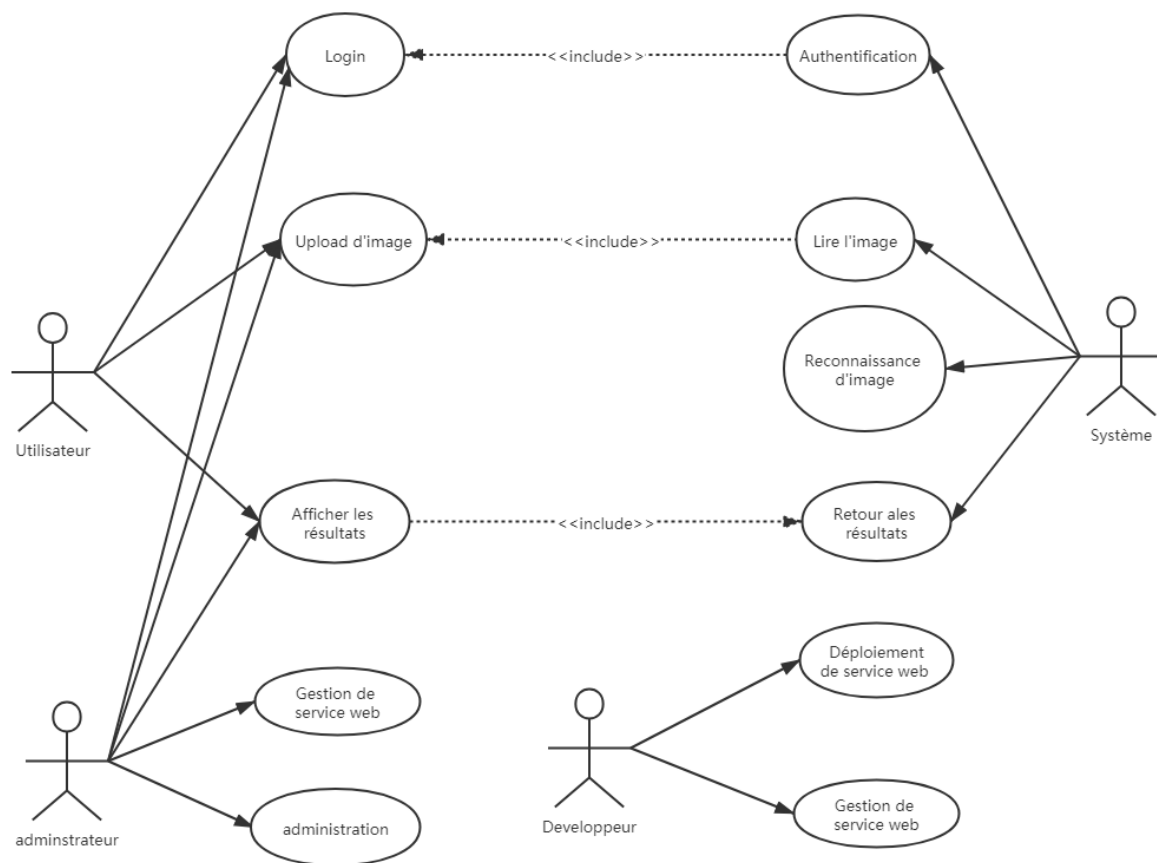


Figure 1 – – Diagramme de cas d'utilisation

Il y a quatre rôles dans ce diagramme : l'utilisateur, l'administrateur, le développeur et le système.

- **L'utilisateur peut se connecter au système. Il peut ensuite sélectionner différents services et télécharger son image.**
- **Le système traite son image et renvoie le résultat.**
- **L'administrateur gère les utilisateurs et les sites Web.**
- **Le développeur peut déployer son service web et gérer les services existant.**

Le diagramme d'activité est :

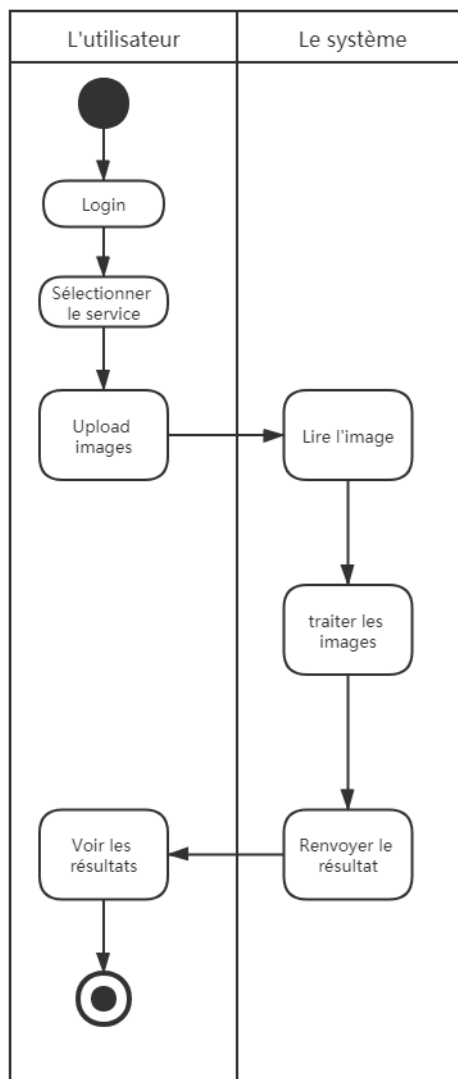


Figure 2 – – *Diagramme d'activité*

Les descriptions détaillées des fonctionnalités sont dans l'annexe « Spécifications fonctionnelles ».

4 Structure générale du système

Pour décrire la structure générale, j'ai créé un diagramme de déploiement et un diagramme d'architecture.

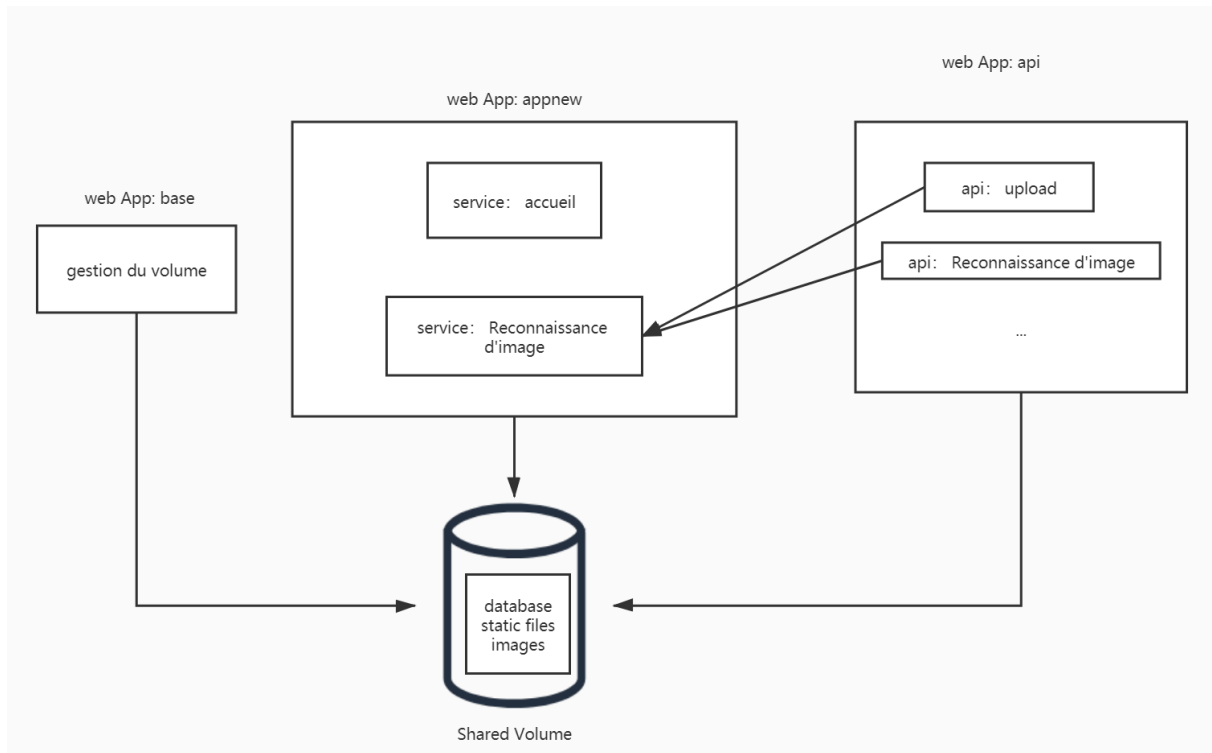


Figure 3 – – *Diagramme d'architecture*

Le système est déployé sur la base du conteneur Docker. Trois images sont actuellement déployées et dépendent les unes des autres pour fournir les services requis par le système.

L'image de base fournit des volumes partagés et d'autres fichiers partagés.

L'image api fournit différents apis, c'est aussi une image de support du système.

L'image appnew fournit le cadre principal du système, y compris l'authentification, la reconnaissance d'image, etc.

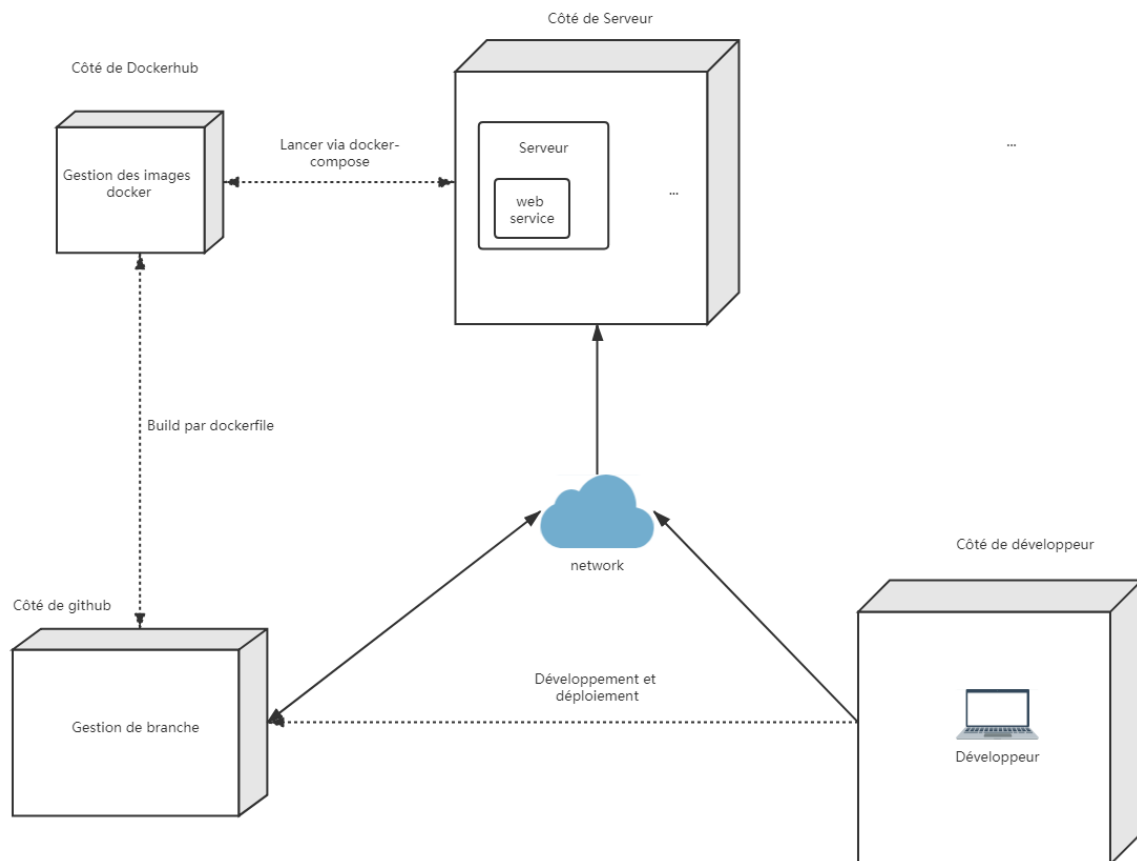


Figure 4 – – *Diagramme de déploiement*

Le processus de déploiement du système est illustré à la figure.

Je développe et teste localement.

Une fois le test réussi, j'ai téléchargé le service Web local sur github.

En raison de ma configuration dans dockerhub et de l'écriture du fichier docker local, la branche sur github créera automatiquement une image pour docker.

Enfin, via le fichier docker-compose, je peux facilement exécuter des services web sur le serveur.

3

Etat de l'art

Dans ce chapitre, on décrit des plateformes pour déployer service web et les exemples d'algorithmes d'IA.

1 Plateforme de déploiement

J'ai recherché deux plateformes : Cloud Foundry(CF) et Docker. Finalement, j'ai choisi d'utiliser Docker comme plateforme pour le déploiement de services Web.

1.1 Cloud Foundry

Cloud Foundry est une plateforme applicative multi-cloud open source en tant que service (PaaS) régie par la fondation Cloud Foundry. Le logiciel a été initialement développé par VMware, puis transféré à Pivotal Software, une entreprise commune d'EMC, VMware et General Electric.

Pour simplifier l'environnement de développement, j'ai étudié la possibilité de déployer Web service avec PaaS. En tant que plateforme open source, Cloud Foundry dispose de bonnes conditions de mise en oeuvre. Il supprime le coût et la complexité de la configuration de l'infrastructure pour les applications.



Figure 1 – – Cloud Foundry

Les développeurs doivent exécuter leurs applications sur n'importe quelle instance Cloud Foundry dans la langue et le framework de leur choix.

Il a donc besoin d'une plate-forme certifiée de Cloud Foundry. Toutes les offres certifiées utilisent le même logiciel Cloud Foundry de base et garantissent la portabilité des applications et des compétences entre les fournisseurs.

Voici quelques exemples :

- **Pivotal Cloud Foundry**

Pivotal Cloud Foundry est la plateforme native de cloud la plus puissante au monde pour créer et exécuter des logiciels.

On peut exécuter Pivotal Platform sur le cloud public et privé.

J'ai testé le déploiement en Pivotal et il est facile à utiliser.

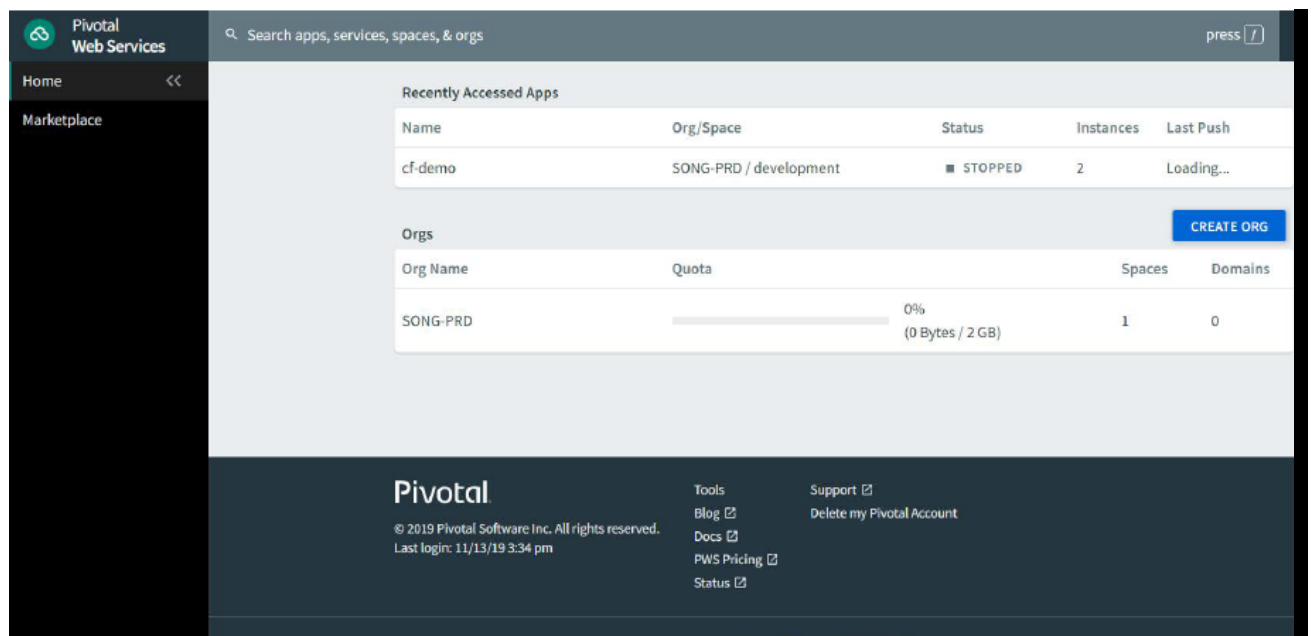


Figure 2 – Pivotal

- **IBM Cloud Foundry** La plateforme IBM Cloud Foundry ne consiste pas seulement à créer de nouvelles applications ou à migrer des applications existantes, des implémentations sur site ou hors site, ou à offrir des services cloud IaaS et PaaS. Il est conçu pour rassembler tous ces aspects pour vous aider à résoudre vos problèmes commerciaux réels et complexes dans le cloud.

Son compte gratuit n'a pas beaucoup d'espace de déploiement.

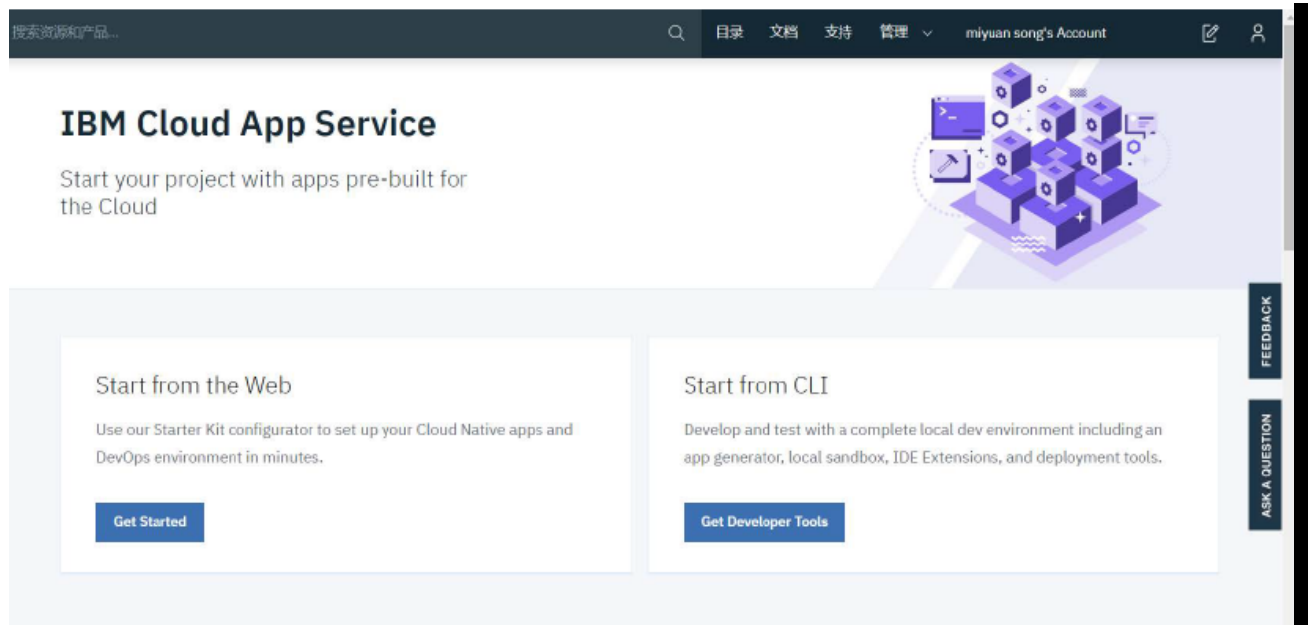


Figure 3 – IBM CF

- **Stratos**

Stratos est une interface de gestion basée sur le Web pour Cloud Foundry.

Il est une application de gestion Web moderne pour Cloud Foundry visant à répondre aux besoins des développeurs et des administrateurs. Il fournit une interface utilisateur visuelle pour compléter l'interface CLI, permettant aux utilisateurs finaux d'effectuer la plupart de leurs tâches quotidiennes directement à partir de leur navigateur Web.

Stratos est conçu pour permettre la gestion de plusieurs déploiements Cloud Foundry via une seule interface. Cela permet à l'utilisateur de travailler en toute transparence entre les déploiements, les organisations / espaces au sein d'un déploiement et de voir une vue unifiée des applications déployées dans plusieurs déploiements Cloud Foundry.

1.2 Docker

Docker est un outil conçu pour faciliter la création, le déploiement et l'exécution d'applications à l'aide de conteneurs. Les conteneurs permettent à un développeur de regrouper une application avec toutes les pièces dont il a besoin, telles que des bibliothèques et d'autres dépendances, et de les expédier en un seul package. Ce faisant, grâce au conteneur, le développeur peut être assuré que l'application s'exécutera sur toute autre machine Linux, quels que soient les paramètres personnalisés de cette machine qui pourraient différer de la machine utilisée pour écrire et tester le code.

D'une certaine manière, Docker est un peu comme une machine virtuelle. Mais contrairement à une machine virtuelle, plutôt que de créer un système d'exploitation virtuel complet, Docker permet aux applications d'utiliser le même noyau Linux que le système sur lequel elles s'exécutent et requiert uniquement que les applications soient livrées avec des éléments qui ne sont pas déjà en cours d'exécution sur l'ordinateur hôte. Cela améliore considérablement les performances et réduit la taille de l'application.

PyCharm est un outil évident pour le développement Python, car il prend en charge l'intégration avec Docker. PyCharm permet de déboguer et d'exécuter votre code dans un conteneur Docker directement depuis votre environnement de développement intégré (IDE). C'est extrêmement pratique car il n'est pas nécessaire de configurer votre environnement de travail pour commencer à travailler avec un projet. Il vous suffit de configurer votre IDE.

Dans le processus de mise en oeuvre, je considérerai également la possibilité de cette mise en oeuvre, et finalement, j'ai choisi d'utiliser Docker comme plate-forme pour le déploiement de services Web.

- **Docker hub**

Docker Hub est le plus grand référentiel d'images de conteneurs au monde avec un éventail de sources de contenu, y compris des développeurs de communauté de conteneurs, des projets open source et des éditeurs de logiciels indépendants (ISV) qui construisent et distribuent leur code dans des conteneurs.

Les utilisateurs ont accès à des référentiels publics gratuits pour stocker et partager des images ou peuvent choisir un plan d'abonnement pour les référentiels privés.

Mon projet a finalement été déployé sur dockerhub.

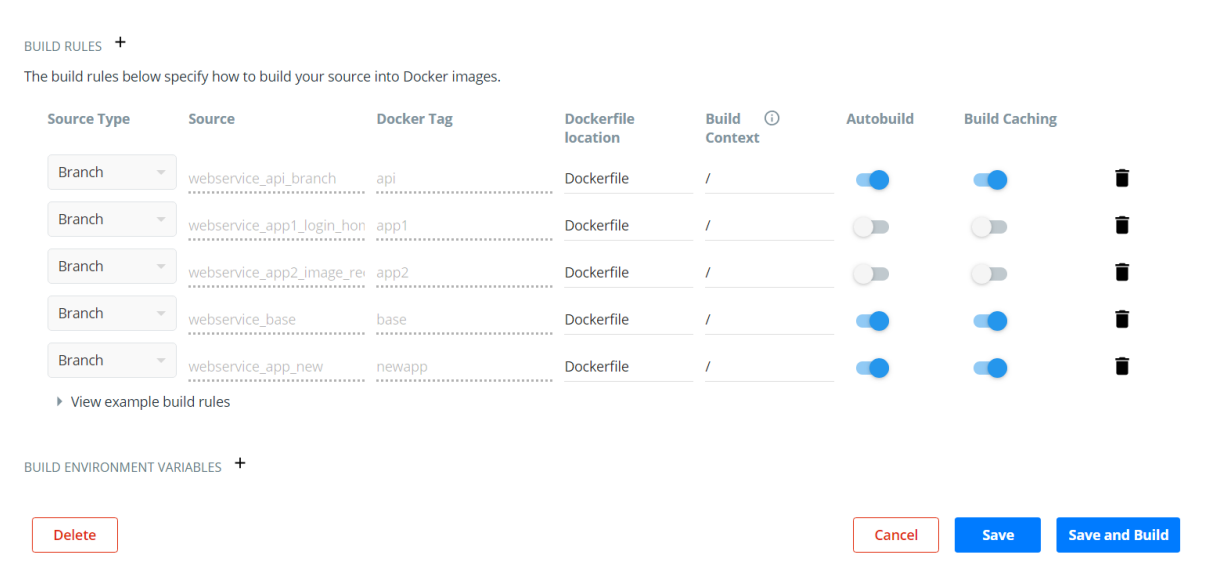


Figure 4 – *-dockerhub*

- **Portainer**

Portainer est une interface utilisateur de gestion d'environnement de docker légère qui peut être utilisée pour gérer les hôtes de docker et les clusters d'essaim de docker. Il est léger, suffisamment léger pour fournir des services aussi longtemps que moins de 100M de conteneur d'image Docker.

Portainer fournit des fonctions qui répondent pleinement à la plupart des besoins des grandes et petites entreprises. Les fonctions principales sont :

1. Fournir un panneau d'affichage d'état : affichez le nombre d'images, de conteneurs, etc. sur le cluster hôte ou essaim
2. Déploiement rapide du modèle d'application : vous pouvez utiliser des modèles préenregistrés ou des modèles personnalisés pour déployer rapidement
3. Affichage du journal des événements : il y a un enregistrement de toute opération et il y a une page pour afficher le journal d'audit
4. Opérations de la console de conteneur : afficher les conteneurs, gérer les conteneurs et afficher les performances occupées par les conteneurs (mémoire, processeur, etc.)
5. Gestion des grappes d'essaims : il peut gérer les grappes d'essaims, ce qui est le plus grand avantage
6. Gestion des utilisateurs de connexion : système utilisateur complet, contrôle des autorisations

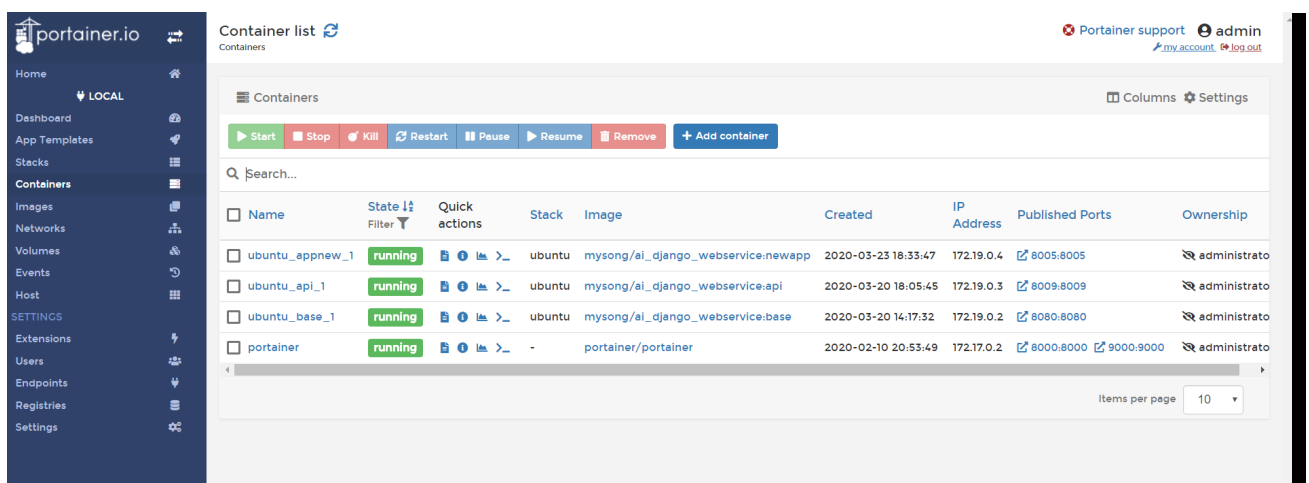


Figure 5 – *Portainer*

2 Cadre d'Algorithmes d'IA

Dans cette partie, je présenterai plusieurs cadres d'algorithmes d'IA.

Dans l'implémentation, je vais choisir un algorithme comme exemple, et le déployer sur service web.

Il faut noter que ce projet n'étudie pas les algorithmes d'IA, mais se concentre sur le déploiement de service web. Donc, on fait seulement une explication simple de l'algorithme d'IA existant.

2.1 H2O

H2O est une plate-forme d'apprentissage machine distribuée en mémoire entièrement open source avec une évolutivité linéaire.

Il prend en charge les algorithmes statistiques et d'apprentissage automatique les plus largement utilisés, y compris les machines à gradient amélioré, les modèles linéaires généralisés, l'apprentissage en profondeur et plus encore.

H2O possède également une fonctionnalité AutoML leader de l'industrie qui exécute automatiquement tous les algorithmes et leurs hyperparamètres pour produire un classement des meilleurs modèles.

Il contient différents algorithmes, comme des approches supervisées et non supervisées, y compris Random Forest, GLM, GBM, XGBoost, GLRM, Word2Vec et bien d'autres.

On peut utiliser Python pour utiliser ces algorithmes.

2.2 TensorFlow

Tensorflow est une plate-forme open source de bout en bout pour l'apprentissage de la machine. Il dispose d'un écosystème complet et flexible d'outils, de bibliothèques et de ressources communautaires qui permet aux chercheurs de faire évoluer l'état de l'art en ML et aux développeurs de créer et de déployer facilement des applications propulsées par ML.

TensorFlow a été initialement développé par des chercheurs et des ingénieurs travaillant au sein de l'équipe Google Brain au sein de l'organisation Machine Intelligence Research de Google pour mener des recherches sur l'apprentissage automatique et les réseaux de neurones profonds. Le système est suffisamment général pour être également applicable dans une grande variété d'autres domaines.

2.3 Api d'IA

De nombreuses entreprises proposent des API qui peuvent être utilisées directement, par exemple, Baidu AI Cloud et Google Cloud Platform.

Ce projet utilise l'api de Baidu AI Cloud. Il fournit un nombre limité de fonctions de reconnaissance d'image. Les API illimitées ne sont disponibles que pour les utilisateurs payants.

4

Analyse et conception

Dans ce chapitre, on détaille les problèmes qui ont existé lors du déploiement de service web. Et concevoir les composants spécifiques du système.

1 Compréhension de la plateforme serveur

J'utilise la technologie Docker pour simplifier l'environnement de déploiement. Plus précisément, j'ai déployé le service Web sur le serveur EC2 d'Amazon Web Services via des images Docker.

1.1 Amazon EC2

Amazon Elastic Compute Cloud ou EC2 est un service proposé par Amazon permettant à des tiers de louer des serveurs sur lesquels exécuter leurs propres applications web.

EC2 permet un déploiement extensible des applications en fournissant une interface web par laquelle un client peut créer des machines virtuelles, c'est-à-dire des instances du serveur, sur lesquelles le client peut charger n'importe quel logiciel de son choix. Un client peut créer, lancer, et arrêter des instances de serveurs en fonction de ses besoins, et paye en fonction du temps d'usage des serveurs, d'où le terme d'« Élastique » (Elastic en anglais).

1.2 Images Docker

Le déploiement traditionnel est l'installation, la configuration et le fonctionnement ; l'avènement de Docker a révolutionné le modèle traditionnel et le déploiement a été simplifié en deux étapes : copier et exécuter.

Par conséquent, de plus en plus d'entreprises utilisent Docker pour améliorer la construction et la livraison d'applications ;

2 Les composants du système

Ce projet utilise le framework django pour construire des services web.

L'idée de base de Django est MVT. MVT est Model-View-Template. L'idée centrale du MVT est le découplage.

M est Model, qui a la même fonction que M dans MVC, est responsable du traitement des données et possède un cadre ORM intégré.

V est View, qui a la même fonction que C dans MVC, reçoit `HttpRequest`, le traitement métier et renvoie `HttpResponse`.

T est Template, qui a la même fonction que V dans MVC, et est responsable de l'encapsulation du code HTML à renvoyer. Le moteur de template est intégré.

Des composants spécifiques sont présentés dans l'annexe «Cahier de développeur».

5

Mise en oeuvre

1 Test de faisabilité

Deux technologies principales sont importantes pour la réalisation du projet. Il est nécessaire de tester sa faisabilité avant de commencer le développement.

J'ai déployé une simple application helloworld utilisant le framework django pour vérifier sa faisabilité. Lorsque le service Web fonctionnait bien localement, j'ai écrit un fichier Dockerfile pour le déployer sur Docker, ce qui en faisait une image Docker.

1.1 Test du framework Django

Django est un framework d'application web open source écrit en Python. Le modèle de conception logicielle de MVT est adopté, c'est-à-dire le Model, la View et le Template.

J'ai écrit son framework de base dans «Cahier de développeur». Après avoir créé le cadre général du projet, j'ai créé une application web « App_helloworld » comme modèle de test. Il a finalement été intégré à la branche github « webservice_template », Cette branche peut être utilisée comme modèle pour développer de nouveaux services Web.

```
from django.shortcuts import render

# Create your views here.
def hello(request):
    #return HttpResponse("myapp2")
    return render(request, "hello.html")
```

Figure 1 – App_helloworld

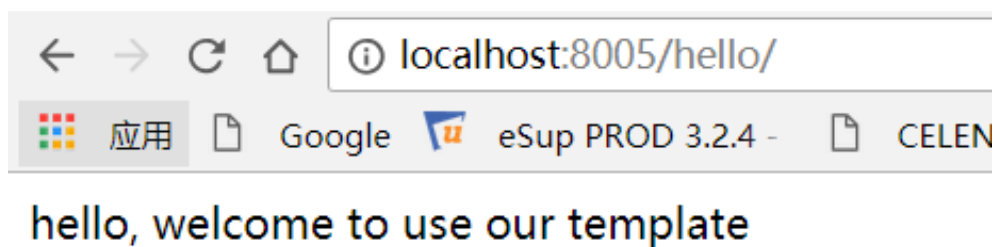


Figure 2 – *Page d'App_helloworld*

Comme indiqué, django fonctionne bien.

1.2 Test de déploiement de Docker

Au cours de mes recherches, j'ai appris à utiliser la plateforme Cloud Foundry comme moyen de déploiement. En raison de son haut degré de commercialisation, de nombreuses entreprises ont implémenté leur propre plate-forme Paas en utilisant son code open source. L'utilisation de ces plateformes n'est pas gratuite, ce qui rend difficile la poursuite du développement.

J'ai donc décidé d'utiliser la deuxième méthode de déploiement que j'ai apprise lors de mes recherches, Docker. Les fonctionnalités open source de Docker, en particulier le mécanisme Dockerhub, peuvent bien gérer les versions de projet et simplifier considérablement le travail du personnel d'exploitation et de maintenance.

Afin de générer une image dans docker, j'ai besoin d'écrire un fichier docker. J'ai donc appris la syntaxe pertinente et déployé avec succès la première image de dokeker.

```

FROM python:3.7
💡
RUN apt-get update
RUN apt-get install -y \
    postgresql-client \
    sqlite3 \
    && rm -fr /var/lib/apt/lists/* \
    && mkdir -p /usr/AIDjangoWebApp

WORKDIR /usr/AIDjangoWebApp
COPY . /usr/AIDjangoWebApp
RUN pip install --no-cache-dir -r requirements.txt

EXPOSE 8000

ENTRYPOINT ["python", "manage.py"]

CMD ["runserver", "0.0.0.0:8000"]

```

Figure 3 – *Dockerfile*

J'ai trié les connaissances liées à Docker et en ai fait un document séparé «L'apprentissage Docker.pdf», qui sera emballé et soumis avec le rapport PRD.

Table des matières

Getting Started (URL) 1

 Environment configuration..... 1

 virtual machine 1

 Linux containers 1

 Docker 1

 What Docker does..... 2

 Image file..... 2

 Image examples..... 2

 Container file..... 3

 Container data volume 3

 Dockerfile file 3

 Application software perspective: 4

 Example: Make your own Docker container 4

Microservices Tutorial (URL) 5

 Introduction 5

Docker compose 6

 Code into DockerHub by github 6

Figure 4 – *L’apprentissage Docker*

J’écris quelques commandes de docker courantes dans des documents séparés«docker cmd.pdf».

Après le test, l’image construite peut exécuter des services Web.

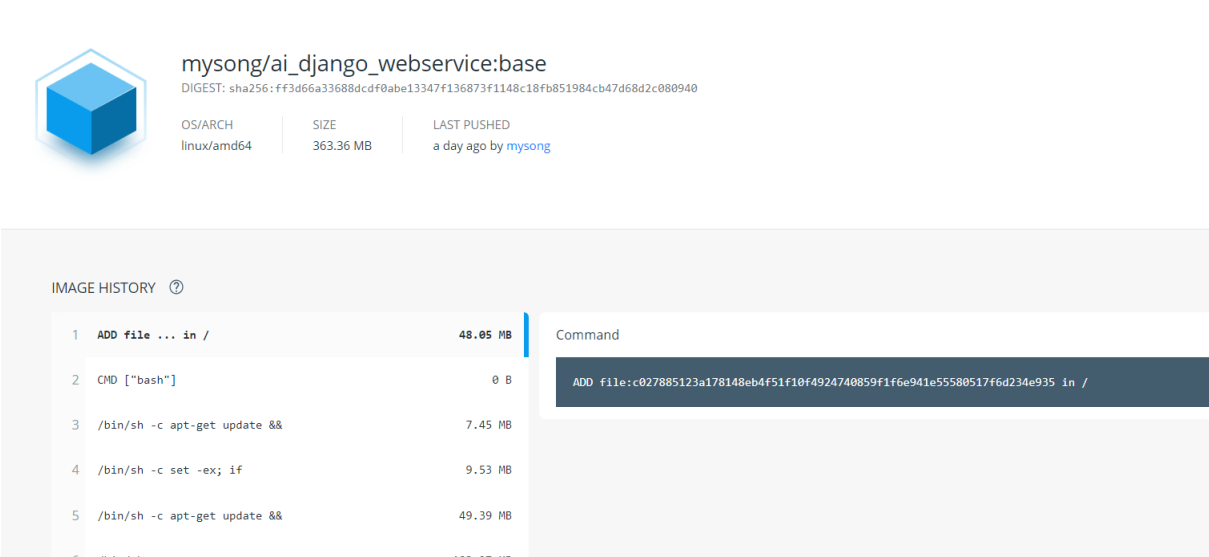


Figure 5 – *Test de Docker*

2 Implémentation

Selon la structure de la conception, la mise en œuvre de ce projet est divisée en trois parties principales, la conception de la structure globale, la mise en œuvre du support API et la construction de services spécifiques.

Les sous-sections suivantes sont basées sur ces trois grandes parties.

2.1 Technologies

La mise en œuvre de ce projet utilise principalement les technologies django et docker, et il existe d'autres technologies comme :

- Django REST framework
- Amazon Aws EC2 Cloud Server
- Git

2.2 Construction d'architecture

Pendant le processus de mise en œuvre, en raison des fonctionnalités de déploiement pratiques de l'environnement Docker, j'ai construit trois branches Git différentes du projet en tant qu'images Docker avec différentes balises.

L'architecture de chaque branche est similaire, les détails spécifiques sont dans « Cahier de développeur ».

2.3 Implémentation de la base de données

La base de données de ce projet n'est pas très compliquée, elle ne contient que des informations sur l'utilisateur et l'image, mais pour les informations sur l'image, j'ai ajouté l'attribut marques pour faciliter l'expansion.

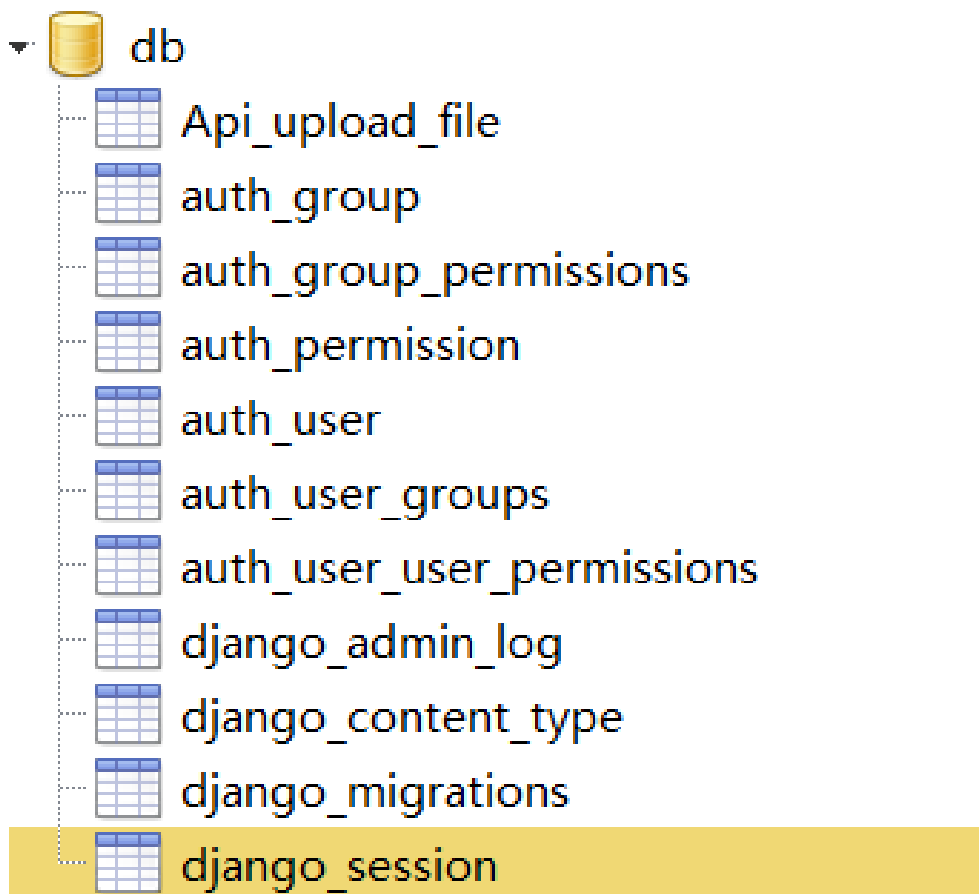


Figure 6 – La base de données

2.4 Implémentation front-end

Selon les caractéristiques de django, je mets les pages front-end dans le package "templates".

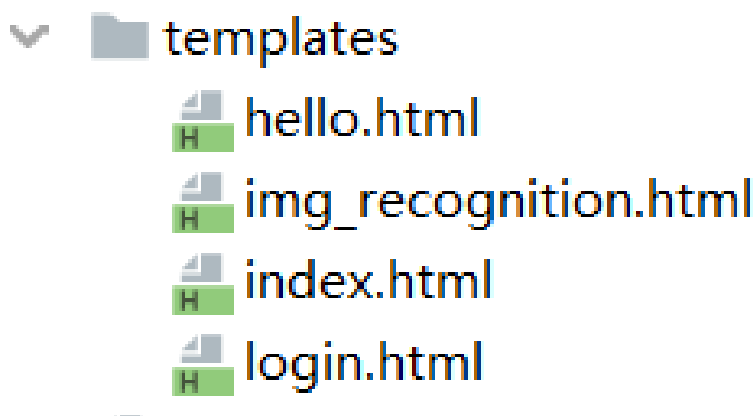


Figure 7 – Front-end

Comme vous pouvez le voir, il y a quatre pages Web, "hello" est utilisé pour les tests, "login" est utilisé pour la connexion.

J'utilise "index" comme page principale et "img_recognition" est la page de reconnaissance d'image.

Toutes les futures pages étendues seront placées ici, cela est nécessaire.

A propos du fichier statique de la page, je l'ai mis dans le package SharedVolume.

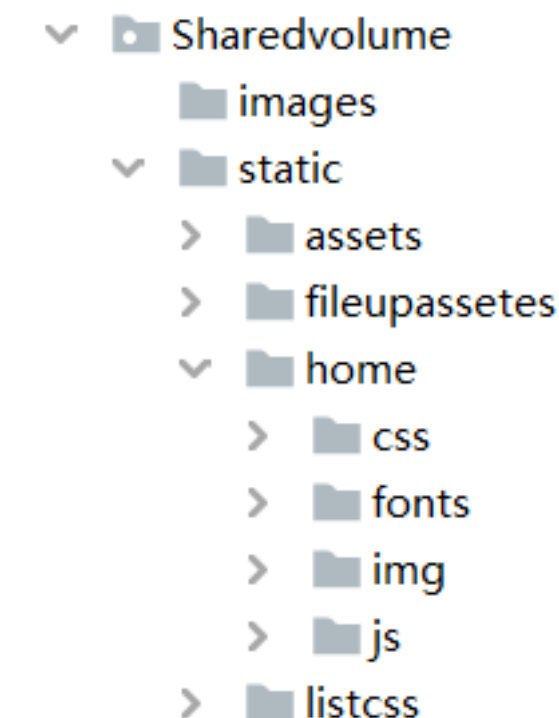


Figure 8 – *Fichier statique*

2.5 Implémentation de l'API

En tant que branche de support, l'api peut télécharger des photos, et j'ai réécrit baidu-api pour que son résultat de retour soit le résultat json attendu.

2.6 Implémentation de la logique métier

Le contrôle logique est implémenté dans le fichier de «view.py» de Django.

Selon le modèle conçu, j'ai implémenté le saut de la page de connexion et le saut orienté service de la page d'accueil.

2.7 Test local

Après avoir implémenté les fonctions principales, j'ai d'abord testé localement.

À des fins de test, j'ai installé Docker Desktop pour Windows en fonction de mon système.

En raison des caractéristiques de docker, l'environnement réel de ce projet est en fait dans le conteneur de docker, donc il n'a rien à voir avec le système du serveur.

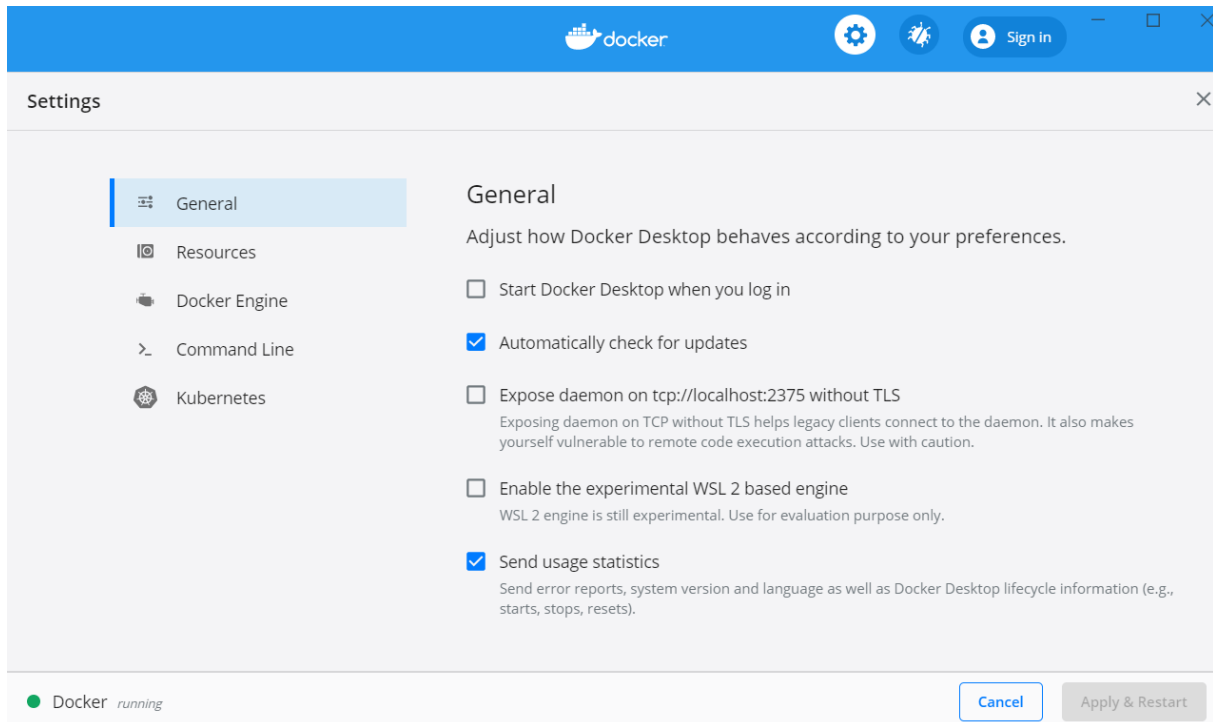


Figure 9 – –Docker Desktop

2.8 Déploiement de mise en ligne

Après les tests, j'ai déployé le service Web en ligne «Amazon Cloud Server».

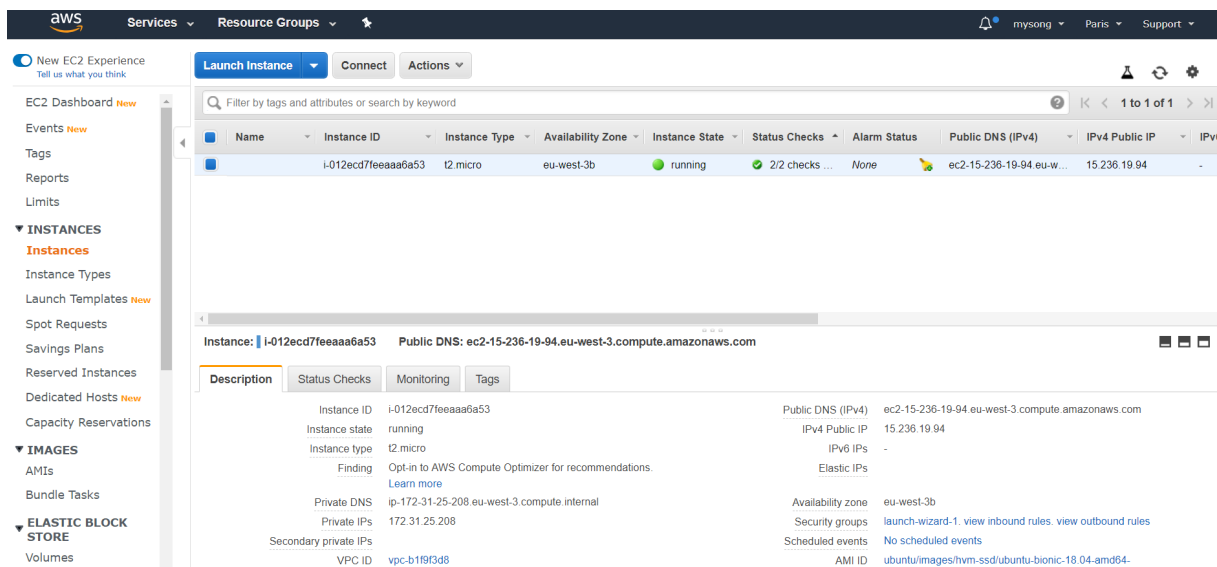


Figure 10 – –AWS EC2

Grâce au protocole ssh de l'outil Xshell, j'ai lancé docker-compose.yml dans la ligne de commande du serveur cloud.

```
root@ip-172-31-25-208:/home/ubuntu# ls
atexit.html  docker-compose.yml
root@ip-172-31-25-208:/home/ubuntu# docker-compose up
```

Figure 11 – *-Xshell*

Après les tests en ligne, le projet a été correctement déployé en ligne

3 Amélioration

Après avoir développé la version initiale, j'ai essayé d'optimiser certaines des fonctionnalités.

3.1 Amélioration de la remarque de reconnaissance d'image

Afin de faciliter l'expansion future des fonctions, j'ai ajouté la balise de remarque dans l'API de téléchargement, donc dans le développement initial, l'utilisateur doit entrer deux paramètres : fichier et remarque.

Compte tenu de la simplicité d'utilisation des utilisateurs, j'ai optimisé la logique métier. Maintenant, si l'utilisateur n'entre pas de remarque, le système peut également créer une remarque aléatoire.

```
if (request.FILES.get("file", None)) is not None: # and (request.POST.get("remark")) is not None:
    remark = ''
    if (request.POST.get("remark")) is not None:
        remark = request.POST.get("remark")
    randomkey = ''.join(random.sample(
        ['z', 'y', 'x', 'w', 'v', 'u', 't', 's', 'r', 'q', 'p', 'o', 'n', 'm', 'l', 'k', 'j', 'i', 'h', 'g',
        'f', 'e', 'd', 'c', 'b', 'a'], 6))
    sessionkey = request.session.session_key
    remark = remark + sessionkey + randomkey
```

Figure 12 – *-Remark*

Nous pouvons voir qu'après l'optimisation, la remarque est composée d'une clé de session, d'un nombre aléatoire et d'une entrée de remarque par l'utilisateur, ce qui garantit son unicité.

3.2 Mécanisme de session

Dans la version initiale, il n'y a pas de mécanisme de session pour les utilisateurs, la sécurité ne peut donc pas être garantie. Par conséquent, j'ai ajouté le mécanisme de session django, de sorte que toutes les pages doivent être en session, sinon il passera à la page de connexion.

```
# session param
SESSION_COOKIE_AGE = 60 * 20 # 20 mins
SESSION_SAVE_EVERY_REQUEST = True
SESSION_EXPIRE_AT_BROWSER_CLOSE = True # Close the browser, COOKIE fails
```

Figure 13 – *-SSession*

Pour la durée de la session, je la mets à 20min, bien sûr, vous pouvez aussi l'ajuster en fonction de vos besoins.

3.3 Positionnement du résultat de retour

Pendant le développement front-end, je mets tout le positionnement de la page en haut. Mais pour le saut après la reconnaissance d'image, le positionnement direct sur le résultat sera plus conforme aux habitudes d'utilisation, j'ai donc fait des ajustements.

3.4 Correction de bug

Il y a aussi quelques bugs dans le processus de développement.

La première est que sur une page, la balise url ne décrit pas précisément l'emplacement de l'écran actuel, j'ai donc fait une optimisation. J'ai également optimisé les balises d'accès côté mobile. Dans la version initiale, nous aurons une erreur de format json lors de la reconnaissance. Cela est dû à la non-unicité de la remarque, qui a été résolue après l'ajout du mécanisme aléatoire.

4 Contrôle de version

Dans ce projet, le contrôle de version est très important, car différentes branches créeront différentes images de docker. Notre projet fonctionne grâce à la coopération de ces différents images.

Bien sûr, le contrôle de version peut également afficher les résultats de mon travail, ce qui est pratique pour les développeurs pour comprendre mon projet.

4.1 Contrôle de version Github

Your branches					
<code>webservice_app_new</code>	Updated 3 hours ago by SIGESI	✓	2 46	New pull request	
<code>webservice_api_branch</code>	Updated yesterday by SIGESI	✓	2 39	New pull request	
<code>webservice_base</code>	Updated yesterday by SIGESI	✓	2 27	New pull request	
<code>webservice_app2_image_rec</code>	Updated 24 days ago by SIGESI	✓	2 28	New pull request	
<code>webservice_app1_login_home</code>	Updated 24 days ago by SIGESI	✓	2 19	New pull request	
<code>webservice_template</code>	Updated last month by SIGESI		2 12	New pull request	
<code>webservice_app2_algo1</code>	Updated 2 months ago by SIGESI	✓	2 20	New pull request	

Figure 14 – Branche de Github

Sur github, j'ai créé sept branches, le projet de déploiement nécessite «basa», «appnew», «api-branch». App1 et App2, ils peuvent être utilisés comme une autre méthode de déploiement, c'est-à-dire un déploiement séparé de chaque application Web.

Template, c'est un projet de modèle qui peut être étendu par des services.

Après près de deux cents soumissions, j'ai eu la version actuelle.

4.2 Contrôle de version Dockerhub

Docker Tag	Source	Latest Build Status	Autobuild	Build caching	
api	webservice_api_branch	SUCCESS	✓	✓	Trigger ▶
app1	webservice_app1_login_home	SUCCESS	×	×	Trigger ▶
app2	webservice_app2_image_rec	SUCCESS	×	×	Trigger ▶
base	webservice_base	SUCCESS	✓	✓	Trigger ▶
newapp	webservice_app_new	SUCCESS	✓	✓	Trigger ▶

Figure 15 – –Version sur Dockerhub

J'associe l'image construite par dockerhub à la branche github.

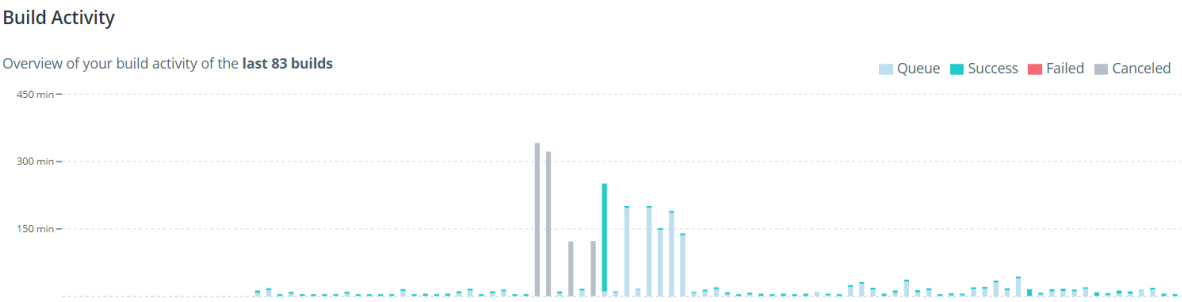


Figure 16 – –Build sur Dockerhub

6

Conclusion

Ce projet se passe en 2 semestres. Pour le semestre 9, je fais la partie de recherche. Et pour le semestre 10, je fais la partie de développement.

1 Bilan du semestre 9

Toutes les taches de S9 sont listées dans les cartes de Trello :

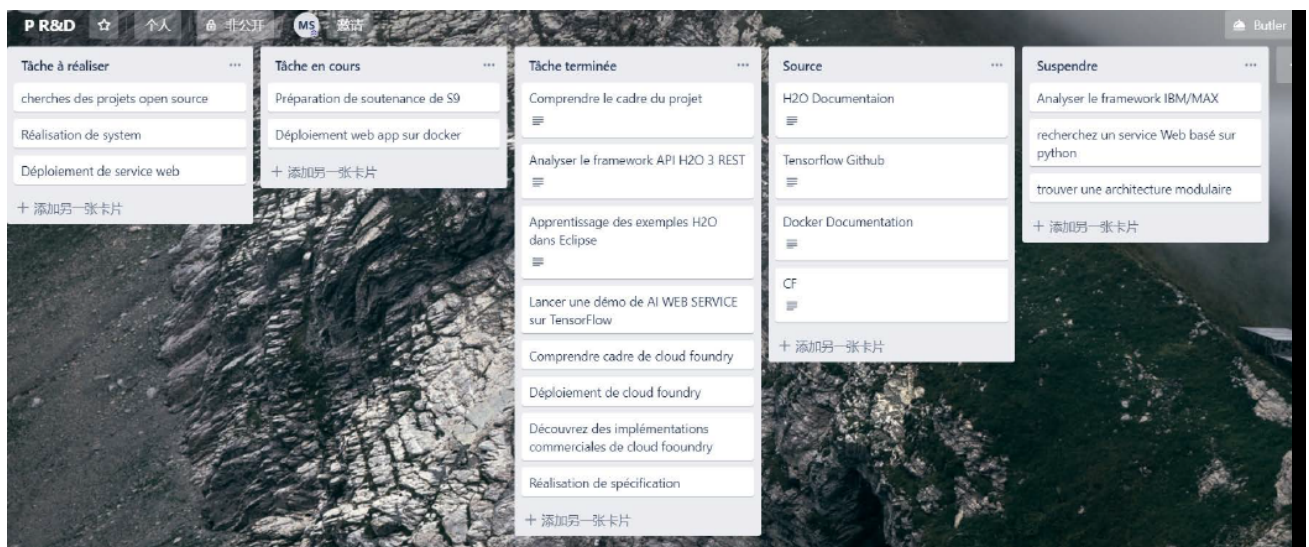


Figure 1 – Les tâche de S9 sur Trello 15

Les tâches finies Pendant la S9 :

- Comprendre le cadre du projet
- Analyser le framework API H2O 3 REST
- Apprentissage des exemples H2O dans Eclipse
- Lancer une démo de AI WEB SERVICE sur TensorFlow
- Comprendre cadre de cloud foundry
- Déploiement de cloud foundry
- Découvrez des implémentations commerciales de cloud fooundry

- Réalisation de spécification
- Ecrire le rapport et préparer la soutenance de S9.

La tâche en retard :

- Déploiement web app sur docker, jusqu'à présent, aucun déploiement de docker n'a été effectué.

Les tâches à faire :

- Chercher des projets open source
- Réalisation de system
- Déploiement de service web

2 Planning du semestre 10

Dans le diagramme de Gantt, on a fait le plan des taches pour le semestre 10. Ces taches suivantes sont à faire pendant le S10 :

- Implémenter le déploiement de l'exemple Web Service
- Modifier l'interface si nécessaire
- Développement et programmation
- Déploiement et test
- Rédaction du rapport
- Préparation de la soutenance

3 Bilan du semestre 10

En S10, j'ai utilisé la cartographie du cerveau pour gérer les tâches (gestion orientée développement).

L'implémentation spécifique est la suivante :

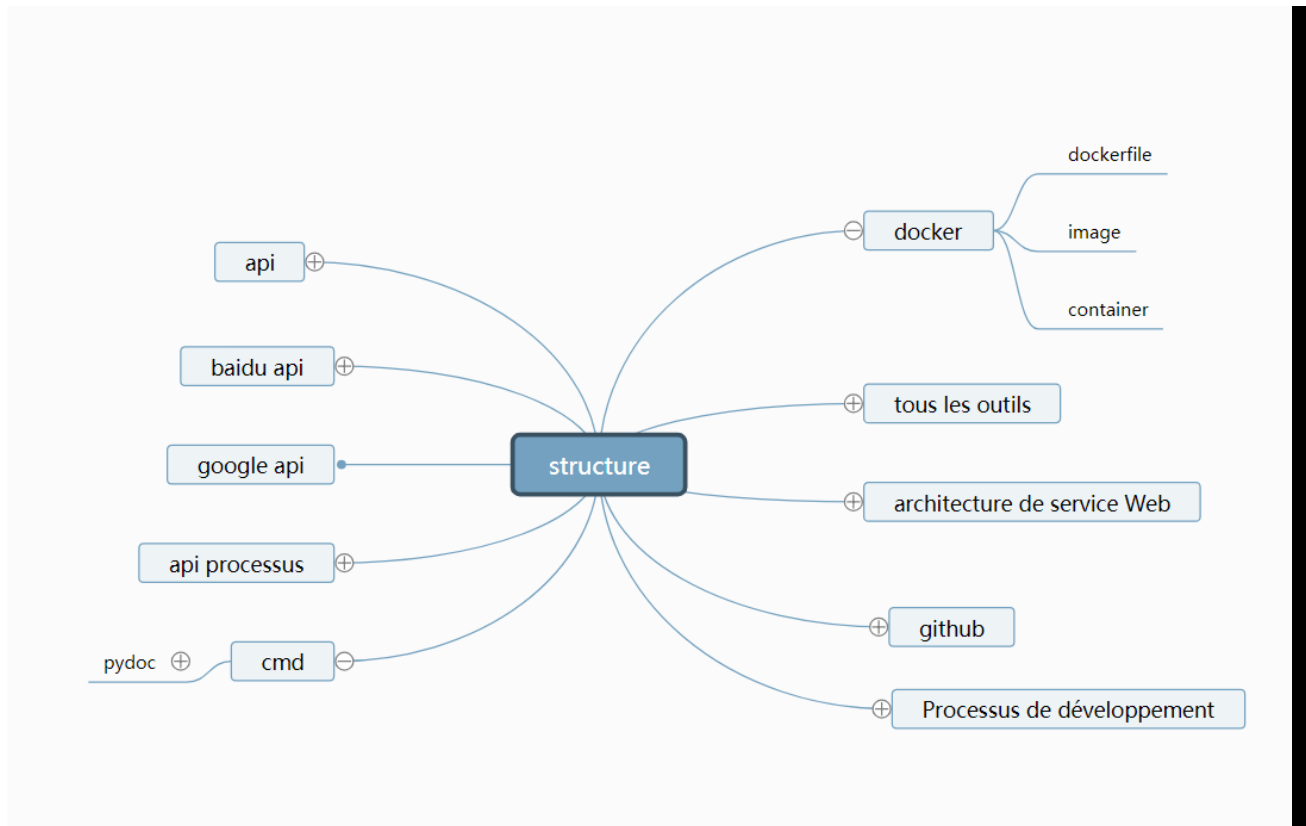


Figure 2 – Carte du cerveau 1

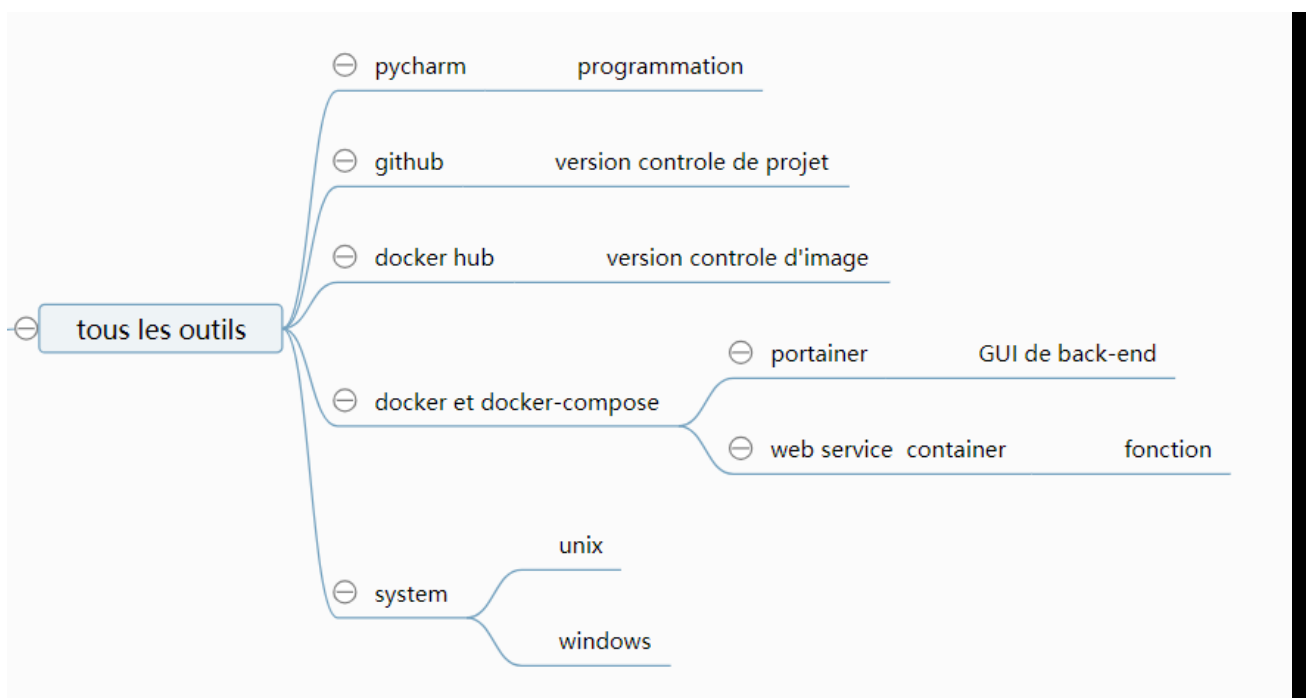


Figure 3 – Carte du cerveau 2

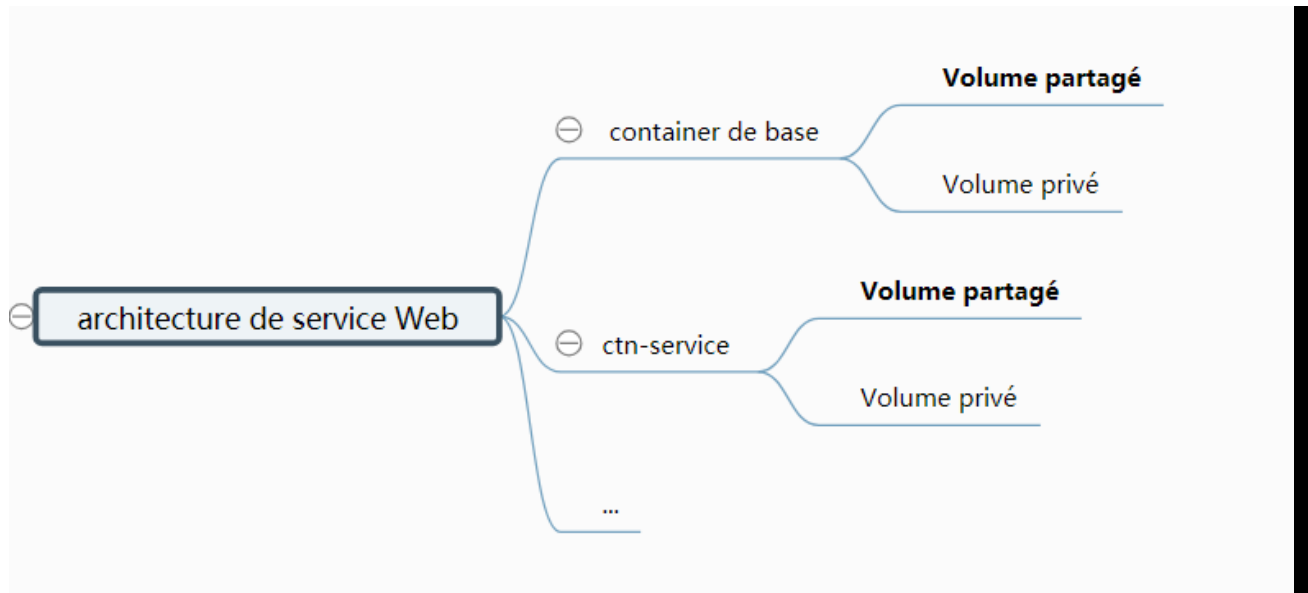


Figure 4 – Carte du cerveau 3

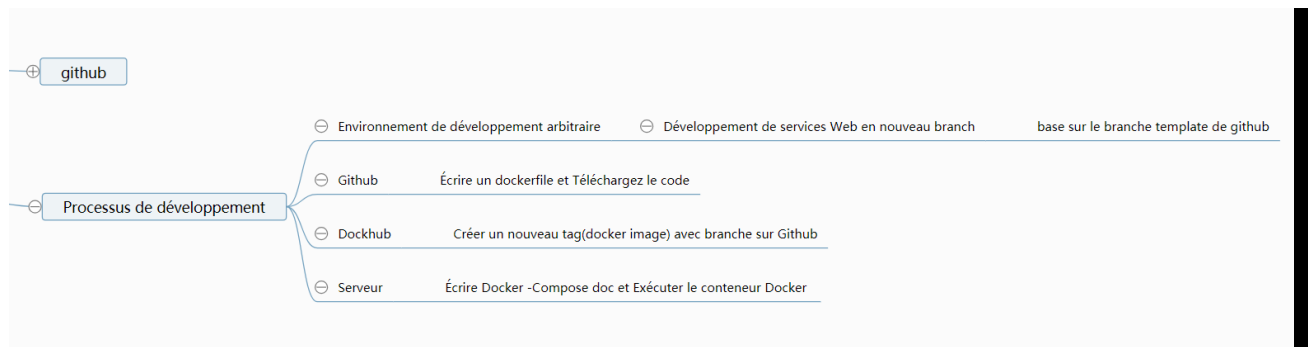


Figure 5 – Carte du cerveau 4

Les tâches globales sont les suivantes

1. Les tâches finies Pendant la S10

- Etudier la faisabilité d'utiliser docker ;
- Faire la modélisation logiciel ;
- Réaliser web service proposé la S9 ;
- Améliorer web service pour élever la performance ;
- Faire des tests ;
- Préparer la soutenance de S10.

2. La tâche en retard

- Ecrire le documentation ;

3. La tâche à faire :

- Livrer le projet

4 Bilan auto-critique

Mon projet est une architecture et une application basées sur un service Web. Il contient beaucoup de nouvelles connaissances et technologies, et je dois penser à toute l'architecture et quelques détails.

Mon avantage est que j'ai déjà expérimenté la programmation de services Web via le langage Java. Bien que j'aie utilisé python dans ce projet, l'expérience précédente m'a fait gagner beaucoup de temps. Et mon client est professeur d'école, il possède une expertise en informatique, je peux donc mieux lui transmettre certaines de mes idées et comprendre facilement certaines de ses exigences. L'utilisation de cartes cérébrales me permet de mieux organiser mes idées, et cela rend beaucoup de tâches plus détaillées, cela peut m'aider à terminer les tâches rapidement.

Je pense que ce qui doit être amélioré, c'est mon choix de technologie. J'ai utilisé beaucoup de nouvelles technologies dans ce projet (afin d'en tirer le meilleur parti), d'une part, j'ai gagné beaucoup, d'autre part, il a augmenté le risque du projet. Concernant le choix des outils de déploiement, j'avais au départ deux solutions, Docker et Cloud Foundry. Après beaucoup de recherches sur Cloud Foundry, j'ai dû l'abandonner car il est très commercialisé et a beaucoup plus de restrictions que docker. Même le docker a beaucoup de «pièges», mais après les avoir résolus un par un, ma technologie s'est considérablement améliorée.

Ce projet m'a donné une compréhension plus approfondie de «l'organisation». La réalisation d'un projet est en fait plus comme la construction d'un système. Il existe diverses dépendances et détails, par conséquent, il existe de nombreux «arts» dans la progression et la structure de la gestion. Ce sont la puissance douce, je dois explorer.

Annexes

A

Cahier de test

Les tests visent à garantir l'exactitude et la fiabilité du logiciel. Pour ce projet, les tests se concentrent sur les tests API et la mise en œuvre des fonctions de système.

Le test est divisé en deux parties : test unitaire et test fonctionnel. Pour les tests unitaires, nous utilisons Postman pour analyser les entrées et les sorties de l'API et les comparer avec le résultat attendu. Pour les tests fonctionnels, nous nous concentrerons sur la fonction du système et vérifierons chaque service

1 Typologie de test

Le test de ce projet comprend les deux parties suivantes :

	Type de test	Description
1	Test unitaire	Le test utilisera postman pour tester l'entrée et la sortie de l'api
2	Test fonctionnel	Les tests vérifieront la faisabilité des fonctions du système

Figure 1 – – Les Types de test

2 Tests unitaires

Le test se concentre sur quatre API Api-upload, Api-download, Api-database, Api-baidu-image. Ce test analysera leur entrée et sortie et l'implémentation des fonctions.

2.1 Tests unitaires de « Api-upload »

- **Description :**
 - Cette API implémente la fonction de upload des fichiers (images).
- **Données d'entrée :**

- File : file
- Text : remark
- **Sortie attendue :**
 - JSON : result
- **Résultat :**
 - Status : 201 Created

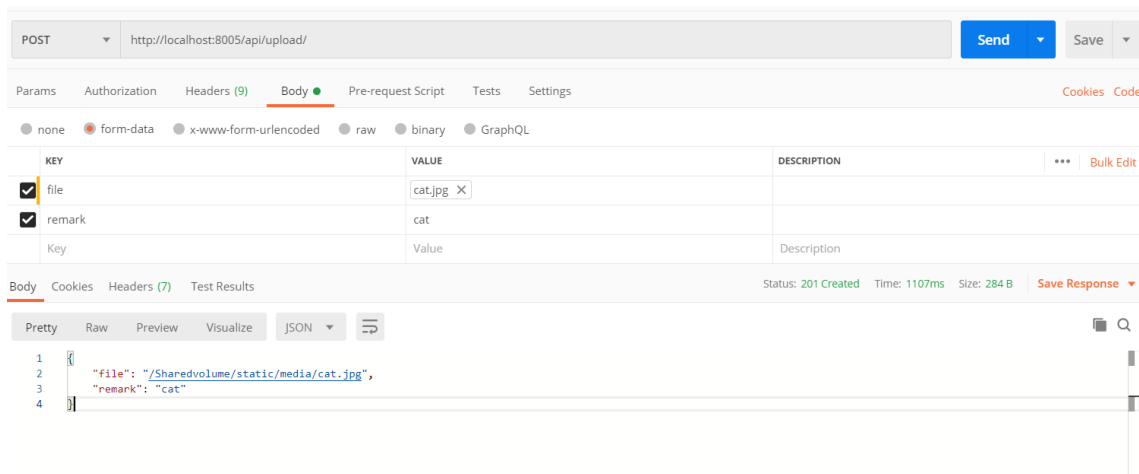
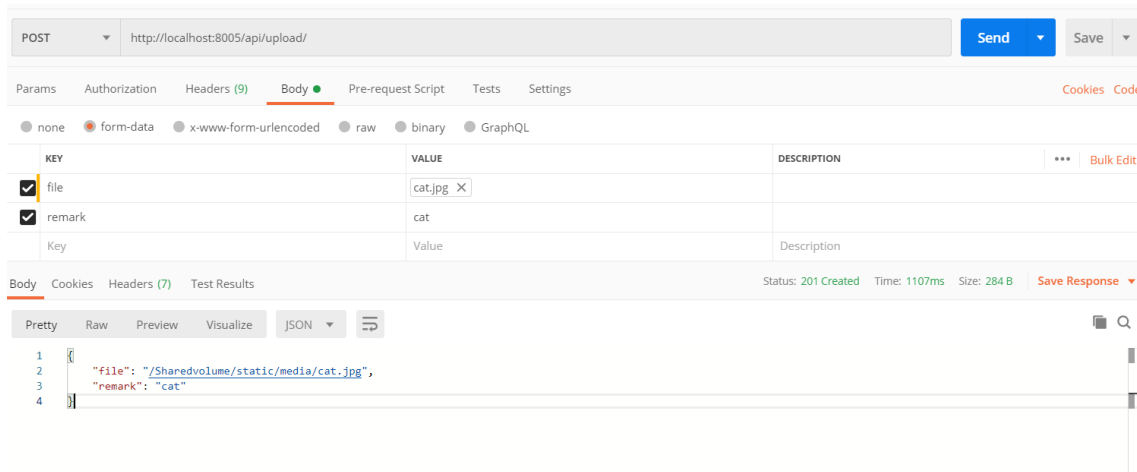


Figure 2 – – Api-upload :Résultat

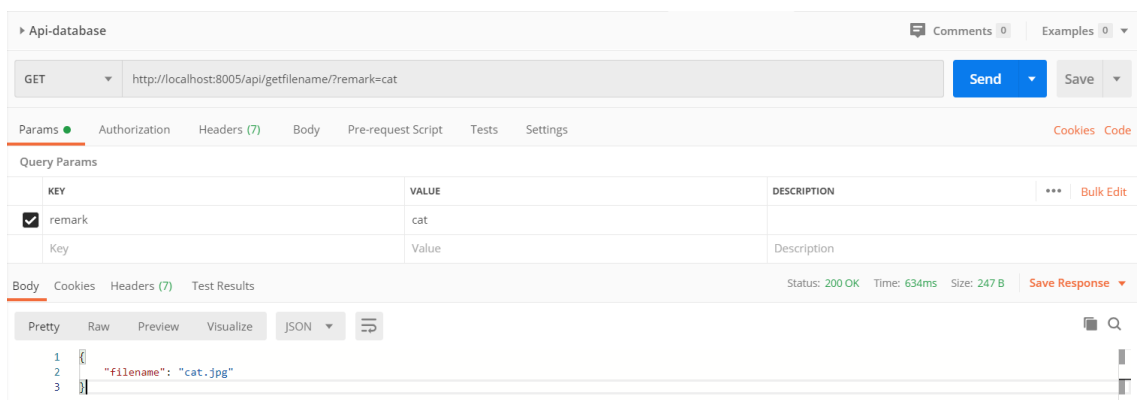
2.2 Tests unitaires de «Api-download»

- **Description :**
 - Cette API implémente la fonction de download des fichiers (images).
- **Données d'entrée :**
 - Param : filename
- **Sortie attendue :**
 - JSON : result
- **Résultat :**
 - Status : 200 ok

Figure 3 – – *Api-download :Résultat*

2.3 Tests unitaires de « Api-database »

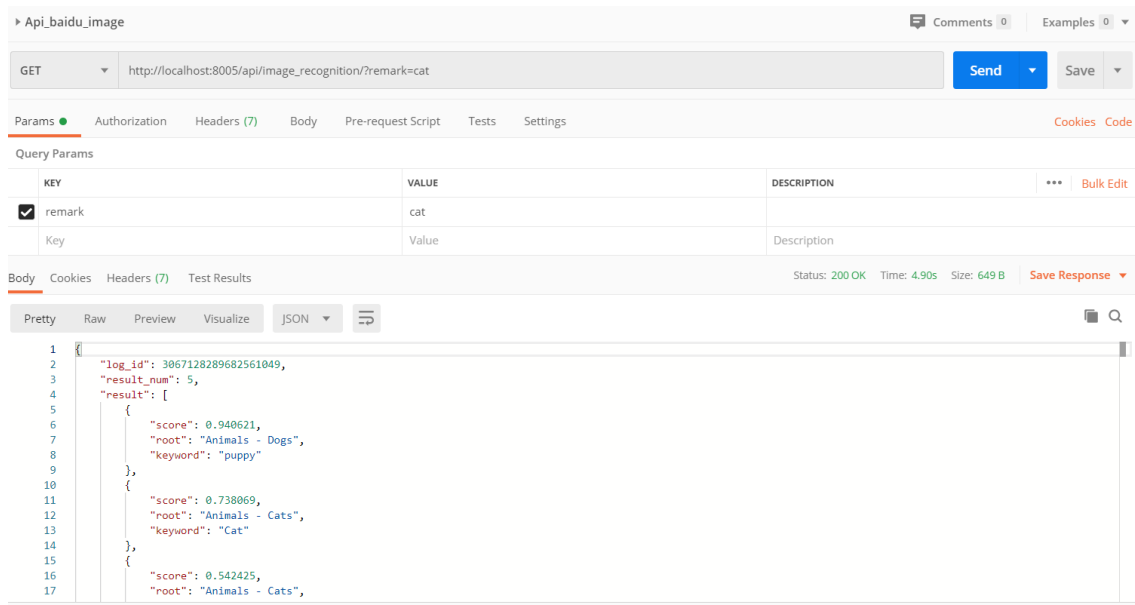
- **Description :**
 - Cette API implémente la fonction de obtenir le nom de fiches par remark. Il s’agit également d’une API de base de données extensible.
- **Données d’entrée :**
 - Param : remark
- **Sortie attendue :**
 - JSON : result
- **Résultat :**
 - Status : 200 Ok

Figure 4 – – *Api-database :Résultat*

2.4 Tests unitaires de « Api-baidu-image »

- **Description :**
 - Cette API implémente la fonction d’identifier le type d’image.

- **Données d'entrée :**
 - Param : remark
- **Sortie attendue :**
 - JSON : result
- **Résultat :**
 - Status : 200 OK

Figure 5 – – *Api-baidu-image :Résultat*

3 Test fonctionnel

Le test fonctionnel consiste à vérifier que le système fonctionne comme prévu. Ce projet implémentera les fonctions suivantes :

3.1 Test fonctionnel de «Authentification»

- **Description :**
 - L'authentification permettre aux utilisateurs de se connecter. Le test est effectué sur un service Web déjà implémenté en appelant l'application Web de l'utilisateur sous le framework django.
- **Données d'entrée :**
 - Données d'authentification saisies par l'utilisateur
- **Sortie attendue :**
- **Exceptions :**
 - Les données d'entrée n'existent pas.

- **Résultat :**
 - Test de connexion : OK
 - Test des données d'erreur : OK

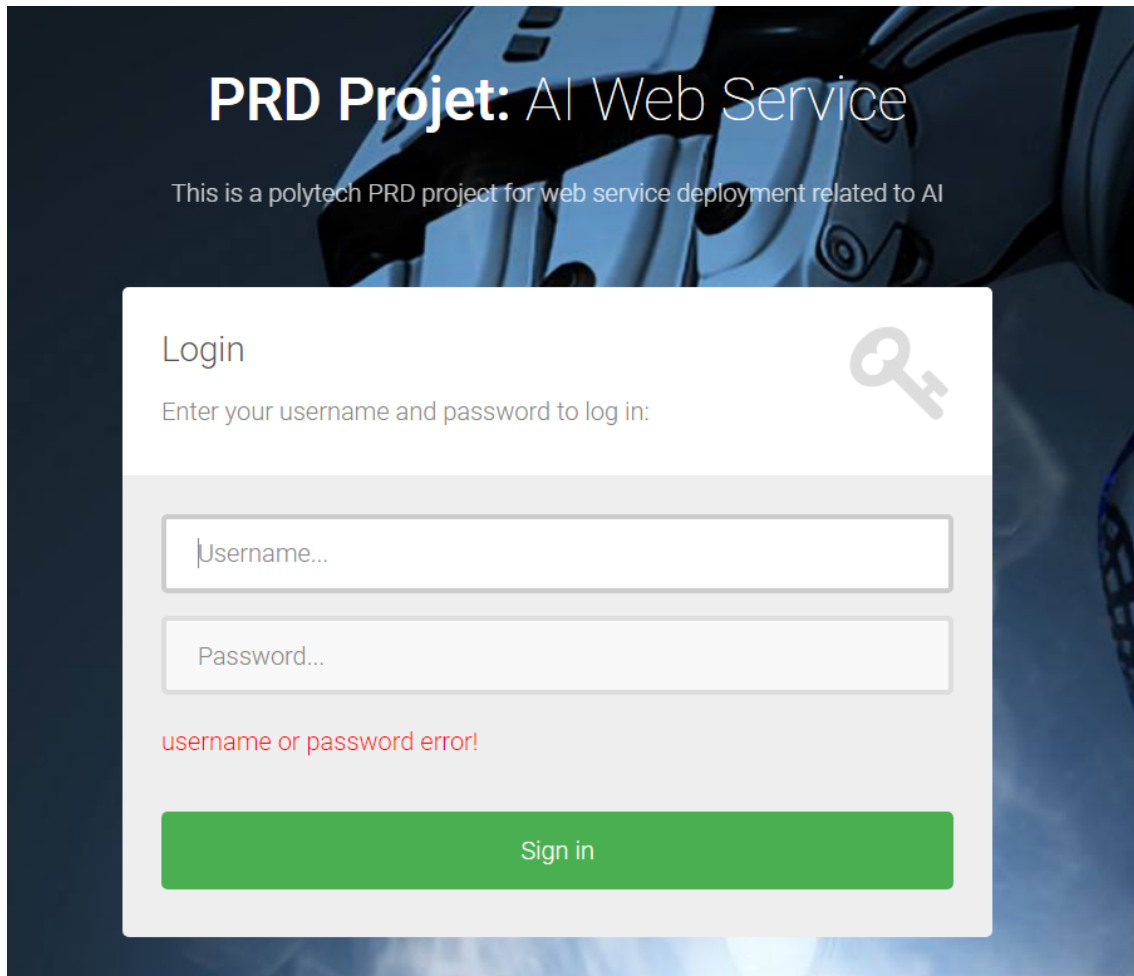


Figure 6 – – *Authentification : Résultat*

3.2 Test fonctionnel de «Index»

- **Description :**
 - L'index est la page principale du service Web, intégrant la navigation, pointant vers d'autres pages et d'autres fonctions.
- **Résultat :**
 - Test de Hyperlien : OK
 - Test d'affichage : OK

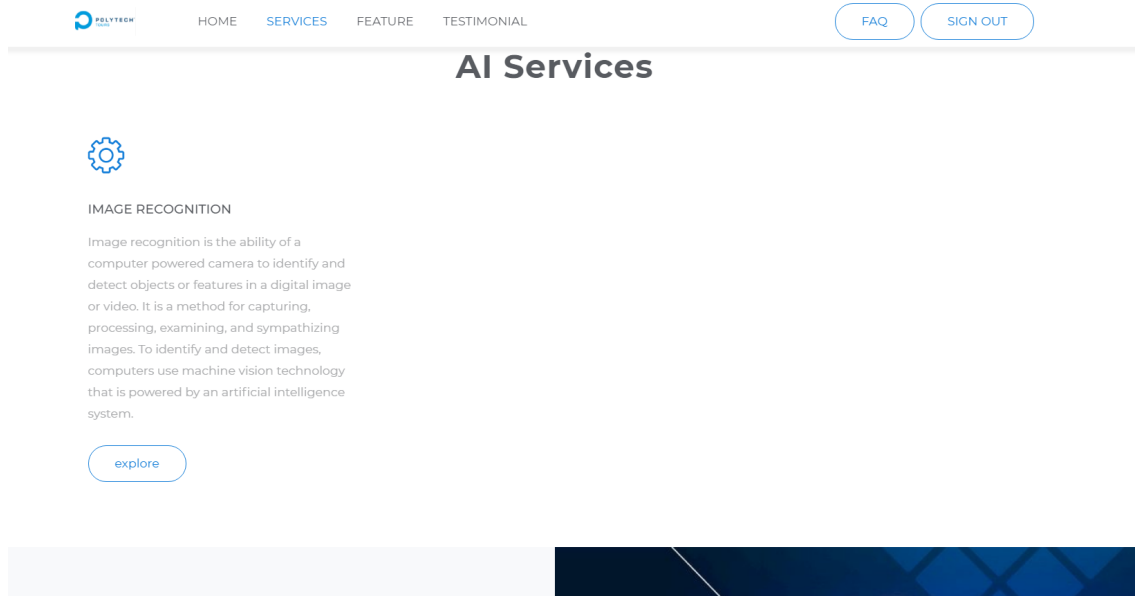


Figure 7 – – Index : Résultat


3.3 Test fonctionnel de «Reconnaissance d'image»

- **Description :**
 - La reconnaissance d'image est la fonction principale du système et, à titre d'exemple pour les développeurs, elle peut implémenter le téléchargement, la reconnaissance et le retour d'images.
- **Données d'entrée :**
 - Image
- **Sortie attendue :**
 - Résultat de reconnaissance
- **Exceptions :**
 - Erreur de téléchargement de fichier
- **Résultat :**
 - Test de upload : OK
 - Test d'affichage des résultats de la reconnaissance : OK

INTRODUCTION **START TO RECOGNIZE** RECOGNITION RESULT [RETURN](#) [SIGN OUT](#)

Input remark:

Select an image:

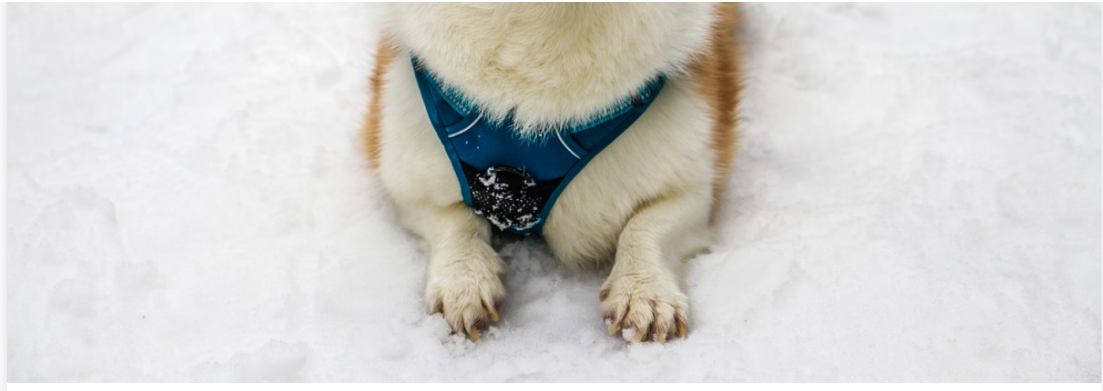


dog.jpg
(159.97 KB)

dog.jpg [Remove](#) [Cancel](#) [Upload](#) [Browse ...](#)

[Recognize](#) [Reset](#)

Figure 8 – – Reconnaissance d'image : Upload



keyword	root	score
Welsh Corgi	Animals - Mammals	0.850267
Welsh Corgi	Animals - Dogs	0.674785
Corgi	Animals - Dogs	0.497158
toy	Goods - Toys	0.27715
Japan Akita dog	Animals - Mammals	0.043151

Figure 9 – – Reconnaissance d'image : Affichage des résultats

3.4 Test fonctionnel de «Gestion des utilisateurs»

- **Description :**

- Système de gestion des utilisateurs construit par le framework django pour réaliser la gestion des utilisateurs.

- **Résultat :**

- Gestion des utilisateurs : OK



Figure 10 – – Gestion des utilisateurs : Résultat

3.5 Test fonctionnel de «Gestion de site Web»

- **Description :**

- Portainer est déployé en tant que système de gestion d'arrière-plan via Docker, qui peut gérer les conteneurs, les volumes et les journaux.

- **Résultat :**

- Gestion de site Web : OK

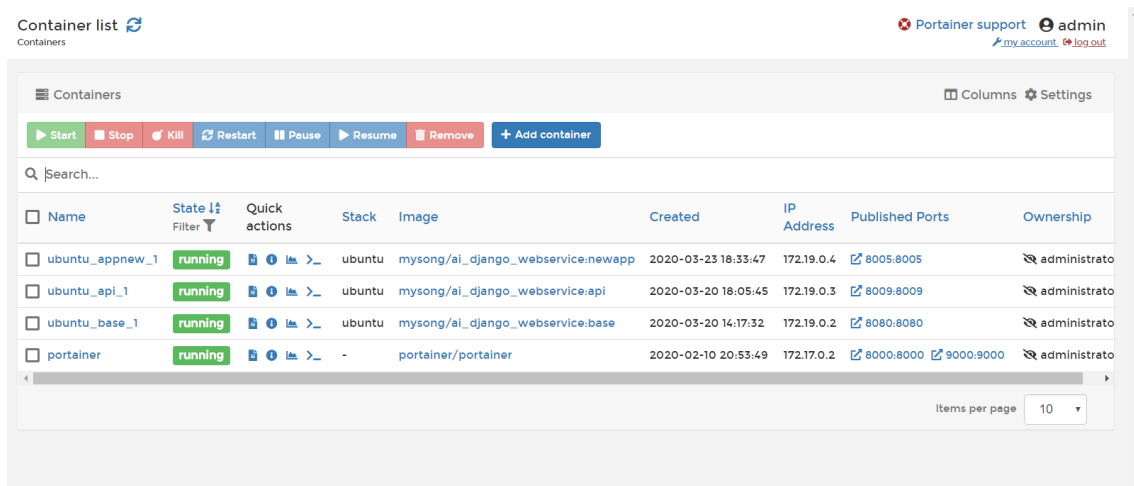


Figure 11 – – Gestion de site Web : Résultat

Container logs

Containers > ubuntu_appnew_1 > Logs

Log viewer settings

Auto-refresh logs Wrap lines Display timestamps Fetch All logsSearch Lines Actions Copy Copy selected lines Unselect

```
[24/Mar/2020 20:47:26] "GET /login/ HTTP/1.1" 200 4774
Not Found: /login/assets/img/backgrounds/1.jpg
[24/Mar/2020 20:47:27] "GET /login/assets/img/backgrounds/1.jpg HTTP/1.1" 404 2826
[24/Mar/2020 21:05:17] "GET /login/?next=/img_rec/ HTTP/1.1" 200 4774
[24/Mar/2020 21:05:18] "GET /Sharedvolume/static/assets/ico/favicon.png HTTP/1.1" 200 459
[24/Mar/2020 21:05:23] "GET /admin/ HTTP/1.1" 302 0
[24/Mar/2020 21:05:23] "GET /admin/login/?next=/admin/ HTTP/1.1" 200 1894
[24/Mar/2020 21:05:23] "GET /Sharedvolume/static/admin/css/base.css HTTP/1.1" 200 16106
[24/Mar/2020 21:05:23] "GET /Sharedvolume/static/admin/css/login.css HTTP/1.1" 200 1203
[24/Mar/2020 21:05:23] "GET /Sharedvolume/static/admin/css/responsive.css HTTP/1.1" 200 17894
[24/Mar/2020 21:05:23] "GET /Sharedvolume/static/admin/css/fonts.css HTTP/1.1" 200 423
[24/Mar/2020 21:05:24] "GET /Sharedvolume/static/admin/fonts/Roboto-Regular-webfont.woff HTTP/1.1" 200 80304
[24/Mar/2020 21:05:24] "GET /Sharedvolume/static/admin/fonts/Roboto-Light-webfont.woff HTTP/1.1" 200 81348
```

Figure 12 – – Gestion de site Web : Gestion de log

B

Cahier de développeur

1 Introduction

Ce projet implémente principalement un framework qui est facile à développer à nouveau, donc grâce au manuel du développeur, vous pouvez rapidement implémenter le déploiement de nouveaux services.

Les technologies utilisées dans le développement sont python, django, django restful framework, docker, git, etc. Le re-développement nécessite au moins la maîtrise du langage python, l'écriture simple de dockerfile et la documentation de docker-compose.

Les documents suivants expliqueront le cadre de développement et le processus de développement. Tout le code source est dans github et dockerhub, les développeurs peuvent rechercher par eux-mêmes :

- **Github** : <https://github.com/SIGESI/Ai-Django-WebService>
- **Dockerhub** : <https://hub.docker.com/r/mysong/ai-django-web-service/tags>

2 Aperçu de la diagramme de classe

Afin de déployer rapidement notre service, nous utilisons une technologie qui combine la branche de github avec le tag de dockerhub. Leur correspondance est :

- «webservice-api-branch» - «api»
- «webservice-app-new» - «newapp»
- «webservice-base» - «base»

Donc chaque branche est un projet complet, mais leur structure de diagramme de classe est similaire, ce sont tous des projets construits via le framework django.

Pendant le déploiement, ils sont configurés avec différentes ports pour communiquer via ces ports et le volume partagé.

Ce document expliquera le diagramme de classe de django(diagramme de classe intégré) et le diagramme de classe de l'API spécifique.

2.1 Diagramme de classe en branche de webservice-app-new

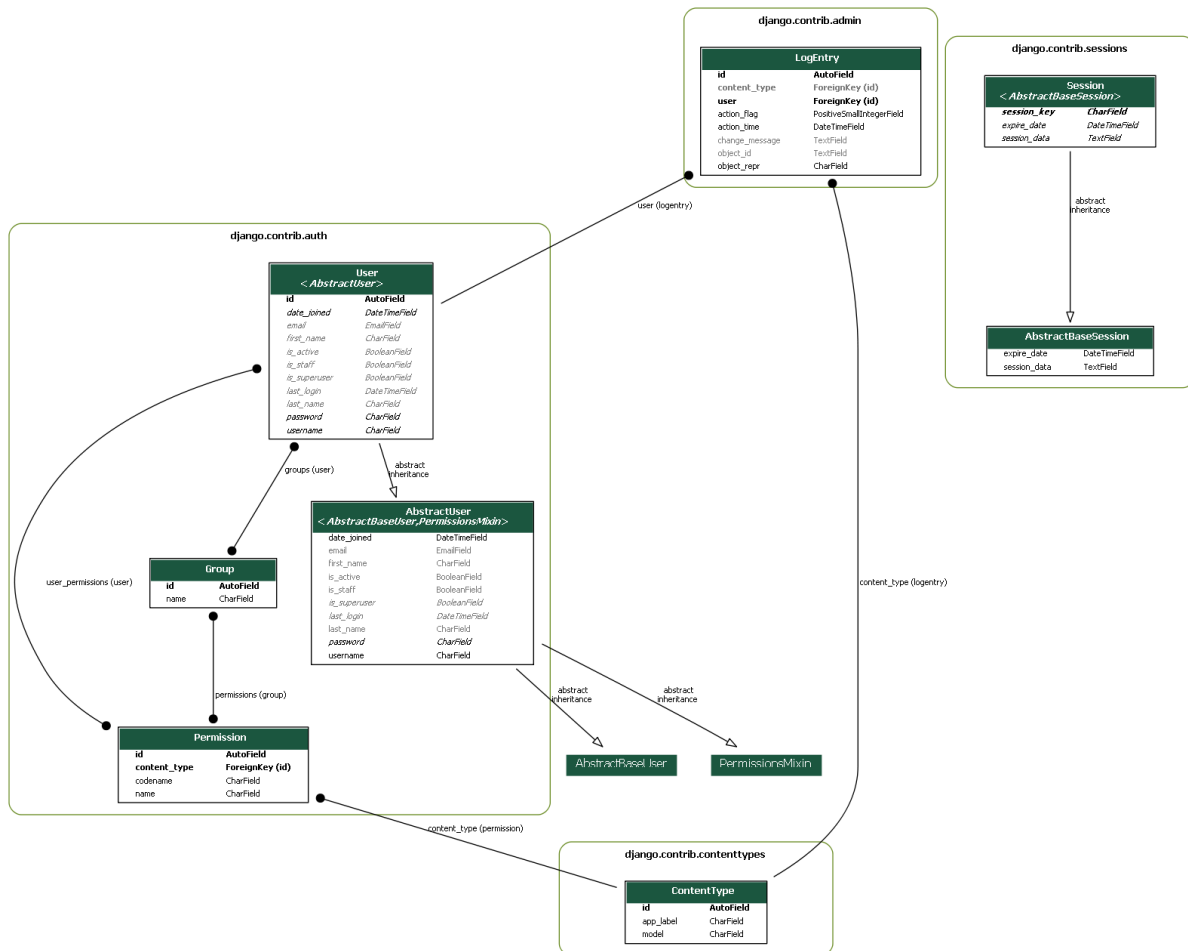


Figure 1 – Diagramme de classe en branche de webservice-app-new

Ceci est un diagramme de classes partagé par les trois branches, il est basé sur le framework django.

- **Package django.contrib.auth** : Ce package fournit principalement des fonctions d'utilisateur et d'authentification.
 - Classe de User : Il contient des attributs tels que le nom d'utilisateur et le mot de passe, fournissant à l'utilisateur de base similaire.
 - Classe de Group : Il fournit une fonction de regroupement d'utilisateurs
 - Classe de Permisson : Il permet aux utilisateurs d'avoir différentes autorisations en fonction de différents groupes.
- **Package django.contrib.admin** :
 - Classe de LogEntry : Le LogEntry enregistre les enregistrements d'opération de tous les utilisateurs. D'une part, il peut être utilisé pour la supervision et, d'autre part, il peut être utilisé pour la restauration.
- **Package django.contrib.session** :
 - Classe de session : Il fournit des fonctionnalités de session et utilise des cookies de navigateur pour enregistrer des informations.

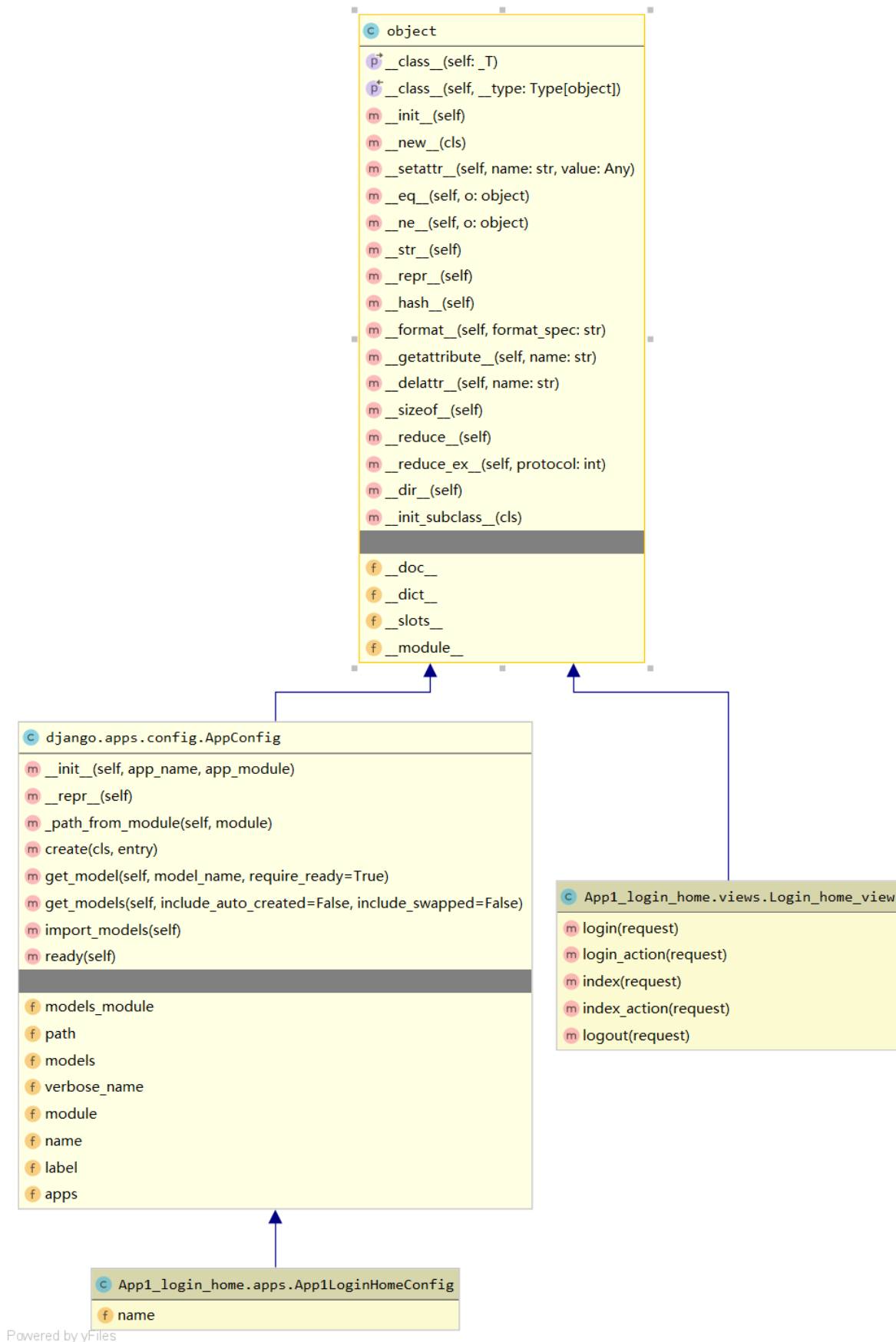


Figure 2 – Diagramme de classe App1-login-home

Le classe de AppConfig est généré par django lors de la création de l'application.

Le classe de login-home-view contient 4 méthodes. Ils forment la logique d'exécution des actions pour la connexion et l'index.

- **méthode de login et login-action** : Il remplit les fonctions d'une page de connexion et d'une connexion vérifiée.
- **méthode de index et index-action** : Il remplit les fonctions d'une page de index et son action.

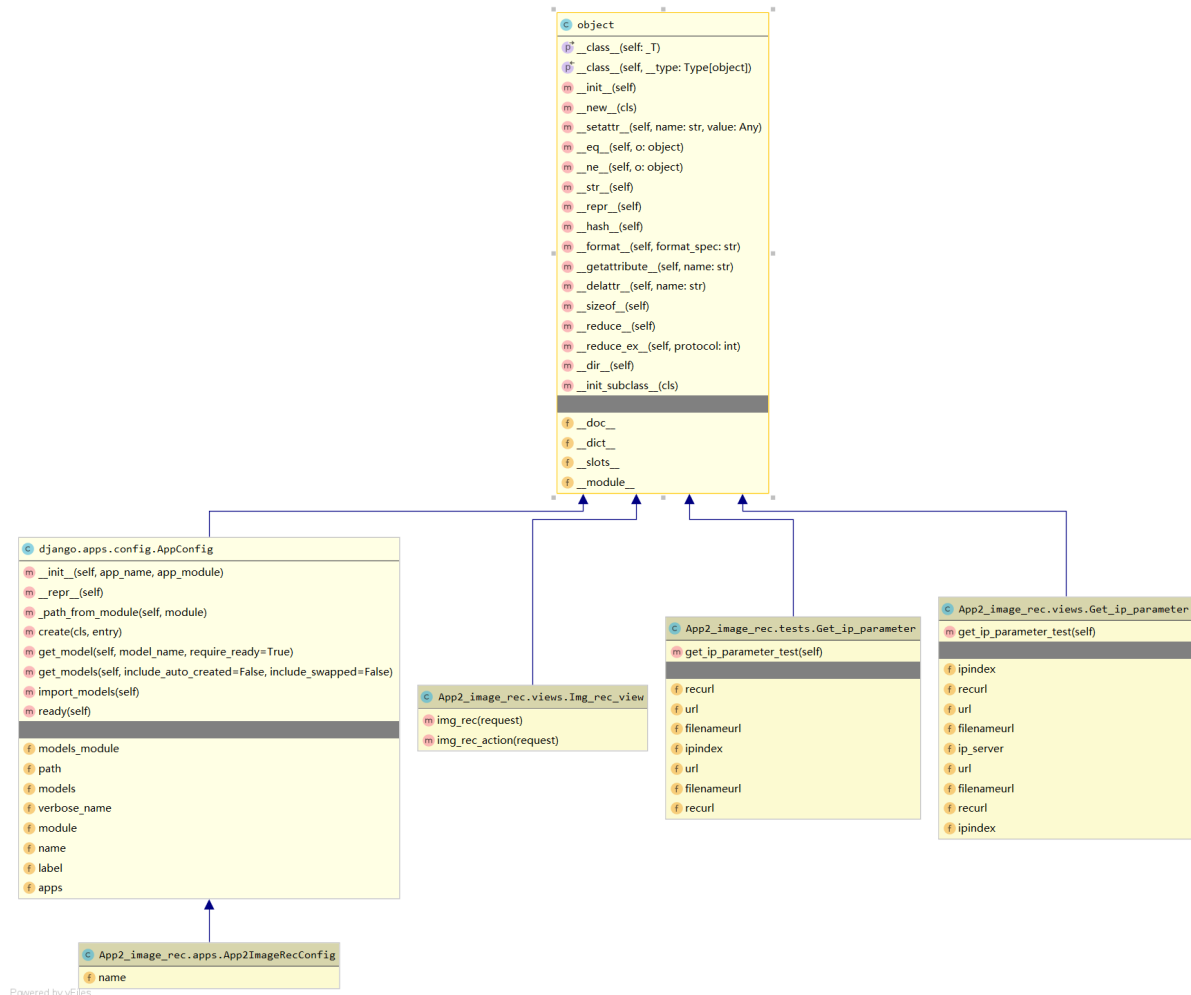


Figure 3 – Diagramme de classe App2-image-rec

Le classe de Get-ip-parameter Utilisez la méthode de test pour changer l'état de test de ip pour le test local.

Le classe d'img-rec-view propose des méthodes et des actions pour la reconnaissance d'image.

2.2 Diagramme de classe en branche de webservice-api

Le diagramme de classe de l'API est donné :

- **Api-upload** : Cette API implémente la fonction de upload des fichiers.

Figure 4 – – Diagramme de classe Api-upload

- **Api-download** : Cette API implémente la fonction de download des fichiers

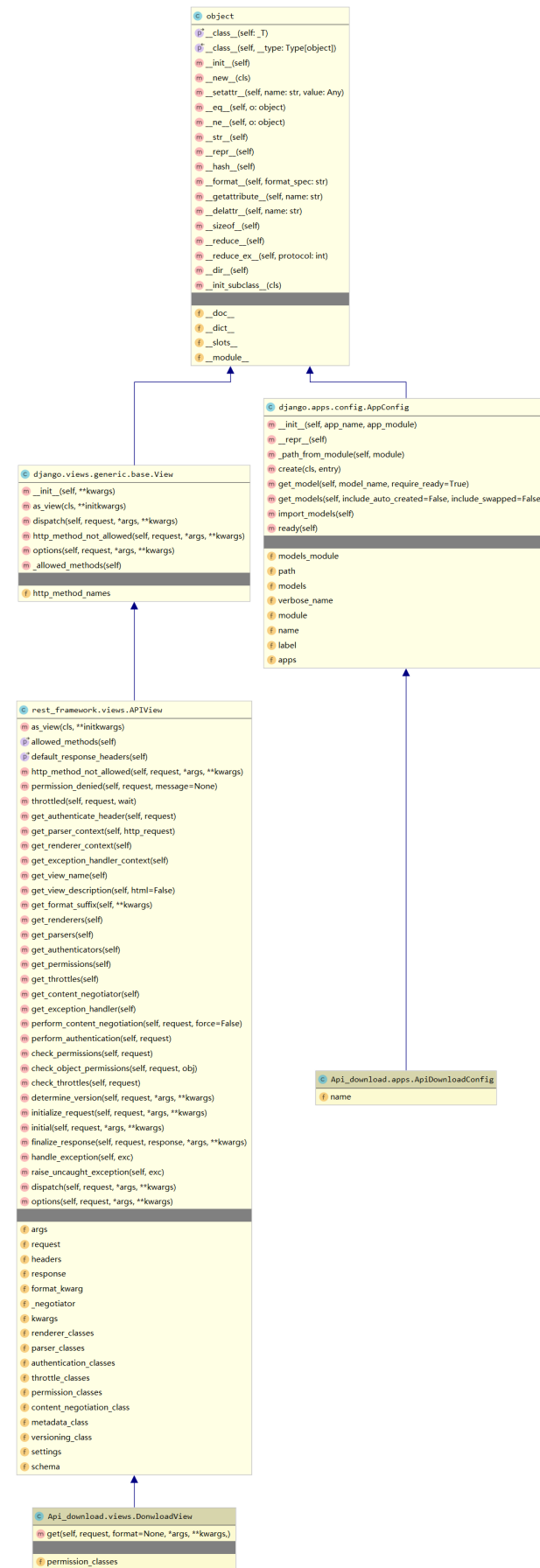


Figure 5 – – *Diagramme de classe Api-download*

- **Api-database** : Cette API implémente la fonction de obtenir le nom de fiches par remark. Il s'agit également d'une API de base de données extensible.

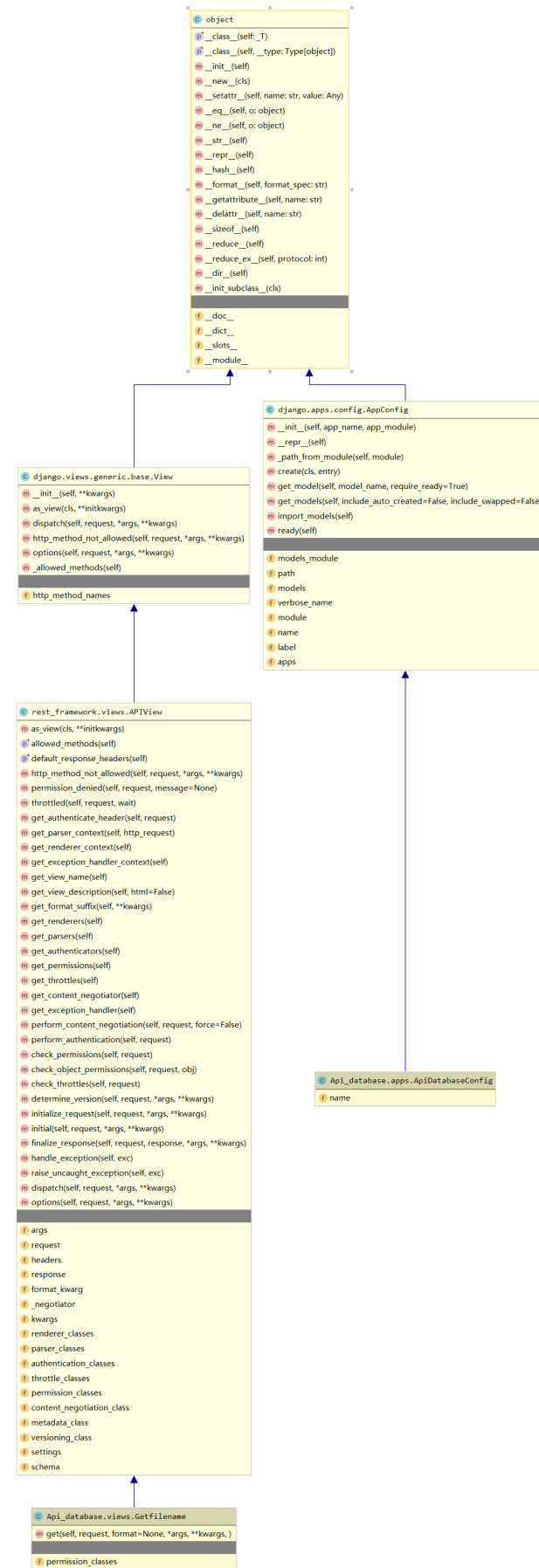


Figure 6 – – *Diagramme de classe Api-database*

- **Api-baidu-image** : Cette API implémente la fonction d'identifier le type d'image.

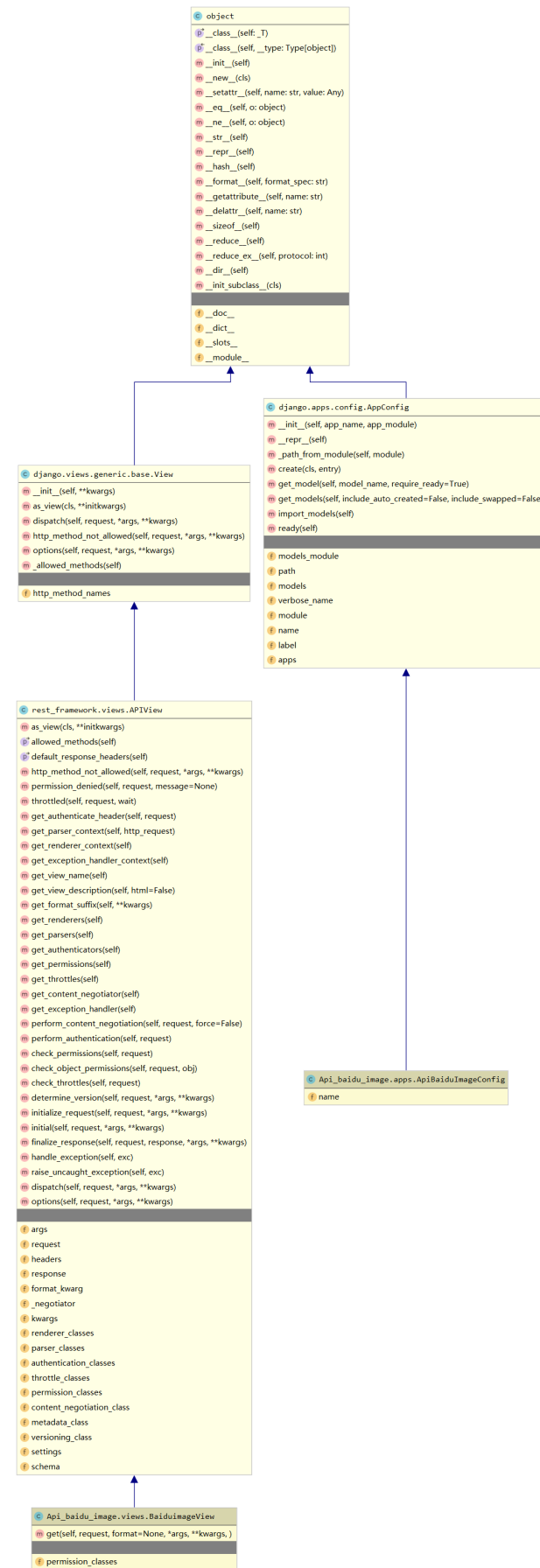


Figure 7 – – Diagramme de classe Api-baidu-image

3 Structure de projet

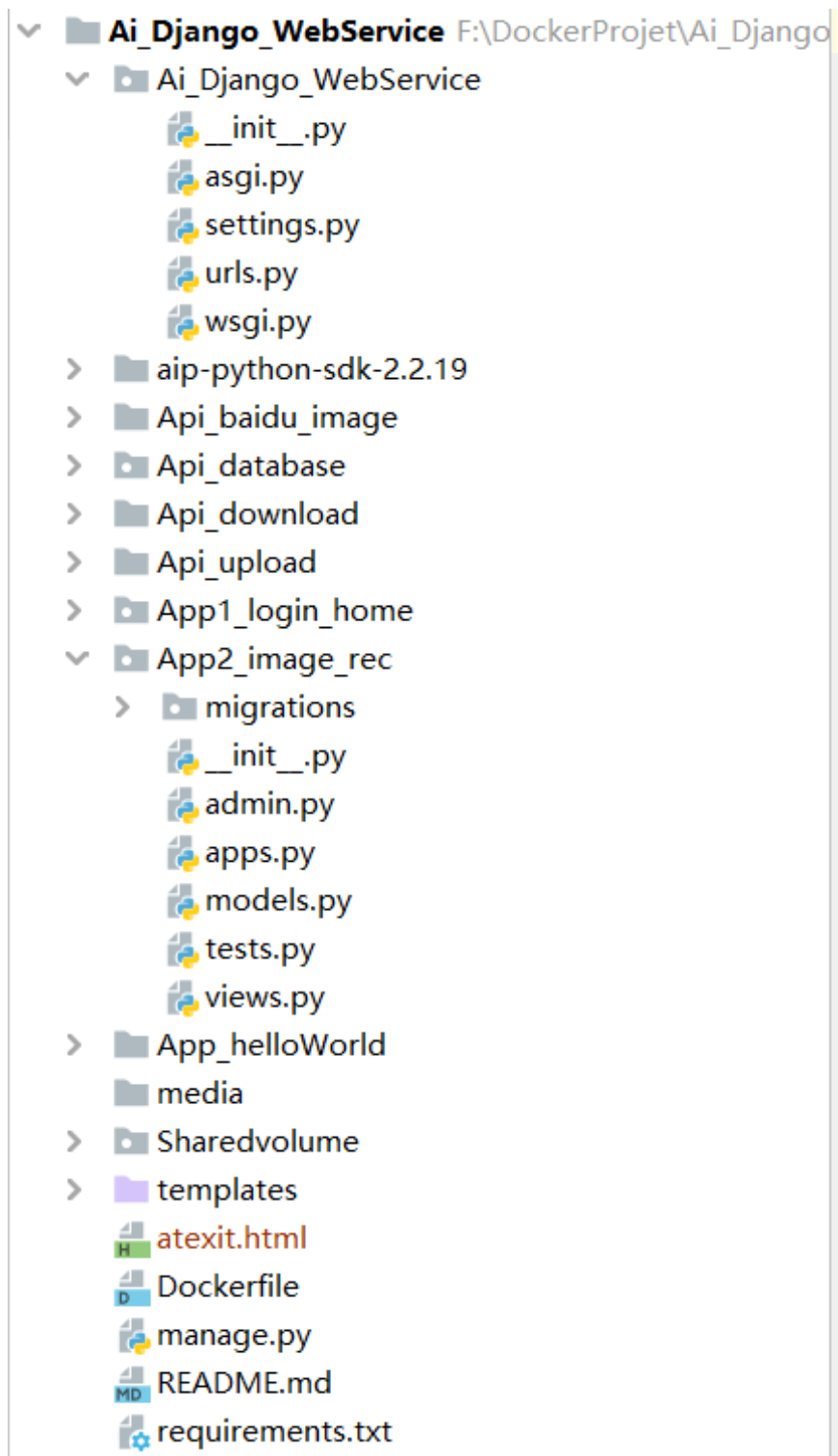


Figure 8 – – Diagramme de Structure

Le setting.py du package Ai-Django-WebService contient toutes les informations de configuration. Les packages Api et App sont des structures d'application Django standard.

SharedVolume formera un volume partagé spécial dans Docker. atexit.html est pydoc, il peut être exécuté sur un serveur local via la commande `python -m pydoc -p 5000`. Son adresse est `http://localhost:5000/`.



Figure 9 – – Diagramme de pydoc

Requirements.txt est configuré avec certaines dépendances du projet d'exécution et il sera construit automatiquement dans Docker.

4 Processus de développement

Ce projet simplifie le redéveloppement des développeurs grâce à l'utilisation de différentes technologies. Les développeurs peuvent se référer aux instructions suivantes pour développer :

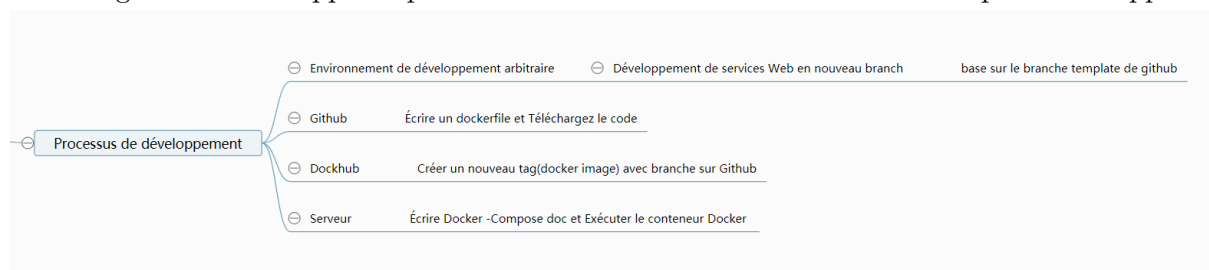


Figure 10 – – Processus de développement

4.1 Développement de services Web

Le développement de services Web peut être basé sur n'importe quelle plateforme ou environnement local. Il suffit d'extraire le code du template de la branche des webservice-template en github pour commencer le développement. Le readme de webservice-template a écrit quelques notes.

Ai_Django_WebService	add helloworld app to template	23 days ago
App2_algo1/images	setting '*'	last month
App_helloWorld	add helloworld app to template	23 days ago
Sharedvolume	add helloworld app to template	23 days ago
templates	add templates of html	last month
Dockerfile	change static(for .css .js) path to /Ai_Django_WebService(For reuse o...	last month
README.md	Create README.md	last month
docker-compose.yml	add docker-compose.yml template	last month
manage.py	Initial commit	last month
requirements.txt	change static(for .css .js) path to /Ai_Django_WebService(For reuse o...	last month

README.md

Webservice_template

This is the template for all new services.

In order to adapt to our deployment in docker, we changed the default file directory of django:

- We put the 'database' and 'static folders (.js, .cc, image)' in the Ai_Django_WebService folder.
- 'Ai_Django_WebService folder' will serve as a shared volume for the docker container.
- We added 'Dockerfile' and 'requirements.txt' to the root directory.

In each use, it is recommended to create a new branch according to the template to expand your own functions.

Figure 11 – – *webservice-template*

4.2 Écrire un dockerfile

Il existe déjà un modèle pour dockerfile dans webservice-template. Les développeurs ont besoin des connaissances de base de dockerfile pour créer leurs propres commandes d'exécution d'images.

18 lines (13 sloc) | 358 Bytes

```

1 FROM python:3.7
2
3 RUN apt-get update
4 RUN apt-get install -y \
5     postgresql-client \
6     sqlite3 \
7     && rm -fr /var/lib/apt/lists/* \
8     && mkdir -p /usr/AIDjangoWebApp
9
10 WORKDIR /usr/AIDjangoWebApp
11 COPY . /usr/AIDjangoWebApp
12 RUN pip install --no-cache-dir -r requirements.txt
13
14 EXPOSE 8000
15
16 ENTRYPOINT ["python", "manage.py"]
17
18 CMD ["runserver", "0.0.0.0:8000"]

```

Figure 12 – – *dockerfile*

4.3 Créer un docker image

Profitez de la combinaison de tag dockerhub et de branche github pour configurer des tag de construction automatique (en option).

BUILD RULES +

The build rules below specify how to build your source into Docker images.

Source Type	Source	Docker Tag	Dockerfile location	Build Context	Autobuild	Build Caching	
Branch	webservice_api_branch	api	Dockerfile	/	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Branch	webservice_app1_login_hon	app1	Dockerfile	/	<input type="checkbox"/>	<input type="checkbox"/>	
Branch	webservice_app2_image_rev	app2	Dockerfile	/	<input type="checkbox"/>	<input type="checkbox"/>	
Branch	webservice_base	base	Dockerfile	/	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Branch	webservice_app_new	newapp	Dockerfile	/	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

▶ View example build rules

BUILD ENVIRONMENT VARIABLES +

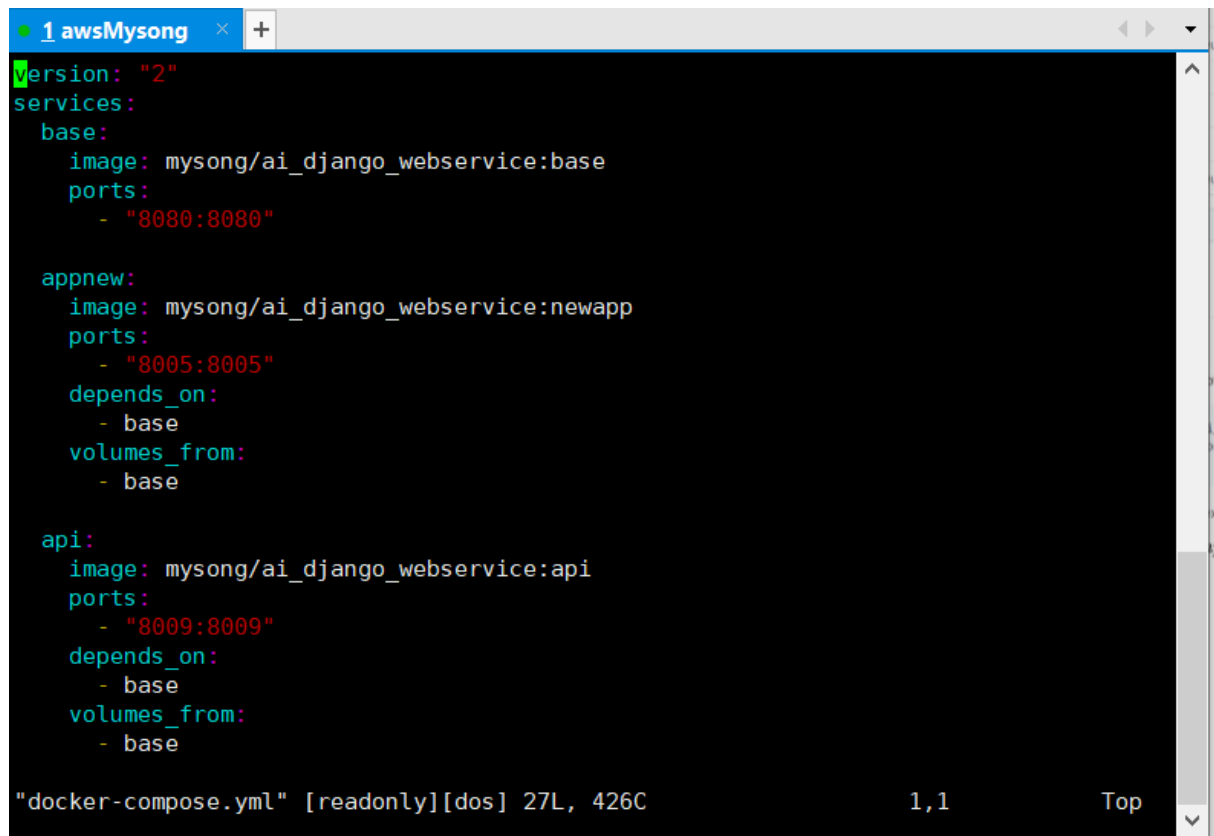
Delete
Cancel
Save
Save and Build

Figure 13 – – *dockerhub*

4.4 Exécuter en serveur

Ce projet utilise Docker-compose comme un outil pour créer et exécuter automatiquement des images Docker.

Par exemple, ce projet déploie des services Web sur Amazon AWS Cloud Server. Exécutez simplement docker-compose.yml avec la commande suivante pour chaque déploiement : -docker-compose up



The screenshot shows a code editor window titled "1 awsMysong". The editor displays a Docker Compose file with the following content:

```
Version: "2"
services:
  base:
    image: mysong/ai_django_webservice:base
    ports:
      - "8080:8080"

  appnew:
    image: mysong/ai_django_webservice:newapp
    ports:
      - "8005:8005"
    depends_on:
      - base
    volumes_from:
      - base

  api:
    image: mysong/ai_django_webservice:api
    ports:
      - "8009:8009"
    depends_on:
      - base
    volumes_from:
      - base
```

At the bottom of the editor, a status bar indicates: "docker-compose.yml" [readonly][dos] 27L, 426C. On the right side of the status bar, it shows "1,1" and a "Top" button.

Figure 14 – – *docker-compose*



Document d'utilisation

1 Installation et configuration

Pour tester ce projet, vous avez besoin d'un serveur avec Docker et Docker Compose installé.

1.1 Prérequis

La documentation d'installation de Docker dans différents environnements est :

- <https://docs.docker.com/install/>

La documentation d'installation de Docker Compose dans différents environnements est :

- <https://docs.docker.com/compose/install/>

Le serveur peut être :

- **Machine virtuelle basée sur Unix** : Vous pouvez utiliser une machine virtuelle lors de tests personnels, l'installation de Docker dans un environnement Unix est simple.
- **Serveur cloud** : Mon exemple de déploiement est en EC2 basé sur Amazon aws. Sa documentation d'utilisation est :
- <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>

Les nouveaux utilisateurs ont un an d'utilisation gratuite. Ceci est l'utilisation recommandée et peut également être utilisé et testé en tant qu'environnement de production.

- **PC Windows ou Mac** : Docker Engine est disponible sur Mac et Windows via Docker Desktop, Windows Server et en tant qu'installation binaire statique.

1.2 Démarrer le portainer

Vous pouvez lancer rapidement portainer via l'image docker.

Si vous utilisez Linux, le déploiement de Portainer est :

```
$ docker volume create portainer_data
```

```
$ docker run -d -p 9000 :9000 -p 8000 :8000 --name portainer --restart always -v /var/run/docker.sock :/var/run/docker.sock -v portainer_data :/data portainer/portainer
```

Si vous utilisez Linux, le déploiement de Portainer est :

```
$ docker run -d -p 9000 :9000 -p 8000 :8000 --name portainer --restart always -v /var/run/docker.sock :/var/run/docker.sock -v C :/data portainer/portainer
```

Vous pouvez maintenant faire la configuration utilisateur de portainer et la configuration d'entrepôt.

1.3 Démarrer le service

Ce projet utilise docker compose pour automatiser le démarrage du service. Un exemple de fichier docker-compose est :

- https://github.com/SIGESI/Ai_Django_WebService/blob/master/docker-compose.yml

Vous devez télécharger docker-compose.yml et le copier dans le répertoire du serveur.

Utilisez ensuite la commande suivante pour démarrer :

```
$ docker-compose up
```

```
root@ip-172-31-25-208:/home/ubuntu# ls
atexit.html  docker-compose.yml
root@ip-172-31-25-208:/home/ubuntu# docker-compose up
Pulling base (mysong/ai_django_webservice:base)...
base: Pulling from mysong/ai_django_webservice
f15005b0235f: Pull complete
41ebfd3d2fd0: Pull complete
b998346ba308: Pull complete
f01ec562c947: Pull complete
2447a2c11907: Pull complete
fdd2d569da3e: Pull complete
fe4ee39b9471: Pull complete
74ab1130643b: Pull complete
77e905403eac: Pull complete
e31295417d3a: Pull complete
13585072097a: Pull complete
ae07bb517de1: Pull complete
4807ebe598e8: Pull complete
Digest: sha256:ff3d66a33688dcdf0abe13347f136873f1148c18fb851984cb47d68d2c080940
Status: Downloaded newer image for mysong/ai_django_webservice:base
Pulling appnew (mysong/ai_django_webservice:newapp)...
newapp: Pulling from mysong/ai_django_webservice
f15005b0235f: Already exists
41ebfd3d2fd0: Already exists
b998346ba308: Already exists
f01ec562c947: Already exists
2447a2c11907: Already exists
fdd2d569da3e: Already exists
fe4ee39b9471: Already exists
74ab1130643b: Already exists
77e905403eac: Already exists
b9c2430e3c94: Pull complete
8371d37c43ef: Pull complete
2ec199dcfdbc: Pull complete
2db7a29556da: Pull complete
Digest: sha256:2f4f0e3d6c0b453279c2c7cc51ac776bb42cbc8395b2c8bc7925ea5acee0e63d
Status: Downloaded newer image for mysong/ai_django_webservice:newapp
Pulling api (mysong/ai_django_webservice:api)...
```

Figure 1 – – *docker-compose up*

Vous pouvez maintenant utiliser les services Web, consultez la section suivante pour plus de détails.

2 Guide d'utilisation général

2.1 Démarche

Lorsque vous testez localement, vous pouvez démarrer le service directement à partir de l'IDE local.

Vous pouvez également lancer un conteneur docker local via docker-compose pour simuler un environnement de production, tout comme les instructions de la section «Installation et configuration» .

Dans un environnement de production, vous devez commencer à configurer des règles entrantes et sortantes.

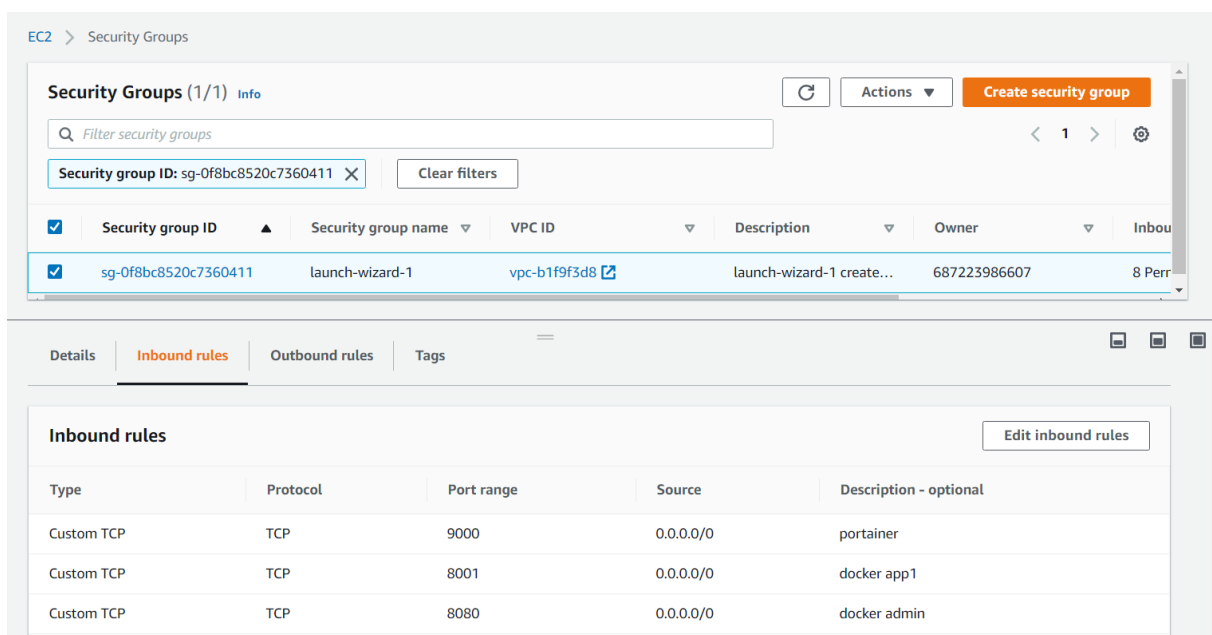


Figure 2 – Des règle d'aws

Adresse IP, « localhost » dans le test local, les autres conditions dépendent du réseau. Dans les sections suivantes, j'utilise «ip-prd» au lieu de véritable ip.

2.2 Log in et Log out

L'URL de connexion est :

- `https://ip-prd:8005/login/`

Notre système est configuré comme un modèle à usage interne, donc la fonction d'enregistrement n'est pas prise en compte. Les utilisateurs se connectent et se déconnectent via le compte attribué.

Le compte administrateur par défaut est :

-Username : admin

-Password : password

2.3 Reconnaissance d'image

Le service de reconnaissance d'image fournit deux types d'informations d'entrée : remark et image.

Parmi eux, la remarque est un élément facultatif, qui peut faciliter le marquage des images, et peut être utilisé comme moyen de gestion des images dans le développement futur.

INTRODUCTION START TO RECOGNIZE RECOGNITION RESULT RETURN SIGN OUT

Input remark: mydogimage

Select an image:

dog.jpg
(159.97 KB)

dog.jpg Remove Cancel Upload Browse ...

Recognize Reset

Figure 3 – – Reconnaissance d'image

2.4 Administration

L'URL d'administration est :

- <https://ip-prd:8005/admin/>

Nous utilisons le module d'administration du framework django pour implémenter la gestion des utilisateurs. Les utilisateurs peuvent se voir attribuer des groupes et des autorisations.

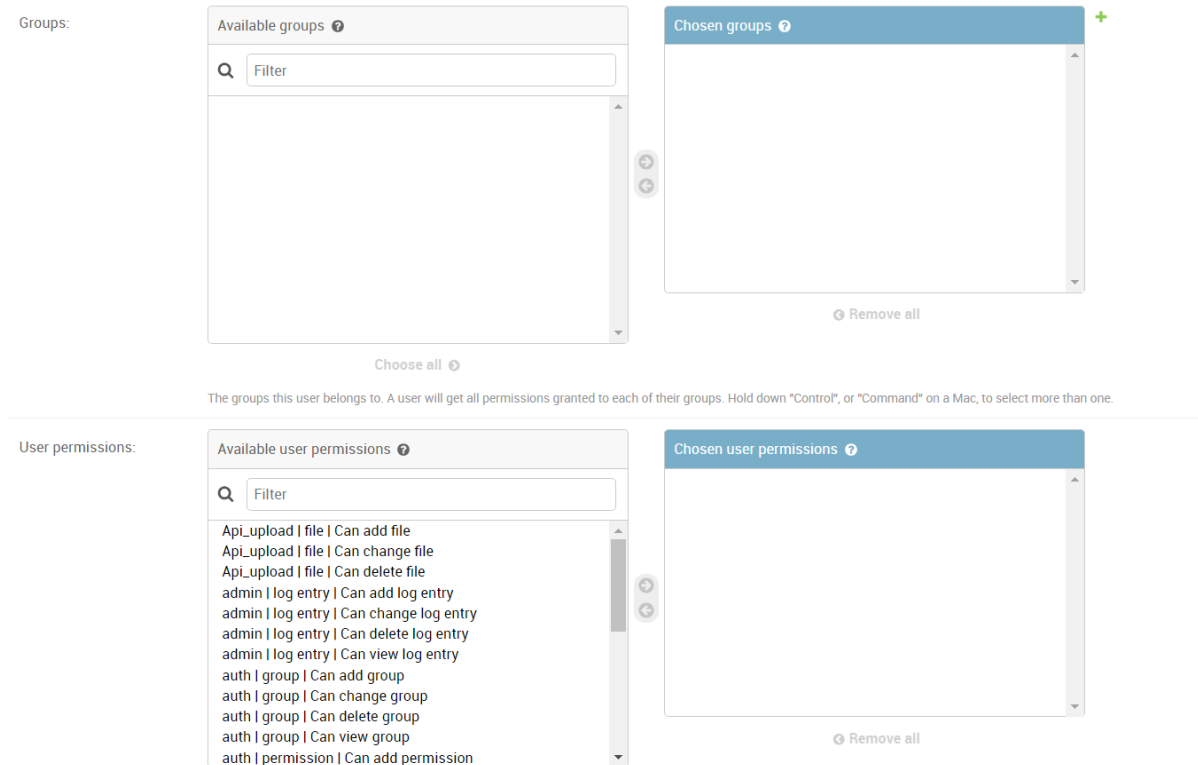


Figure 4 – Administration

Le compte administrateur est :

- Username : admin
- Password : password

2.5 Gestion de site Web

L'URL de gestion de site Web est :

- https ://ip-prd :9000

La gestion du site Web est basée sur la gestion des conteneurs. En utilisant l'image de gestion visuelle du conteneur de portainer pour obtenir la prise de contrôle des informations du site Web, etc.

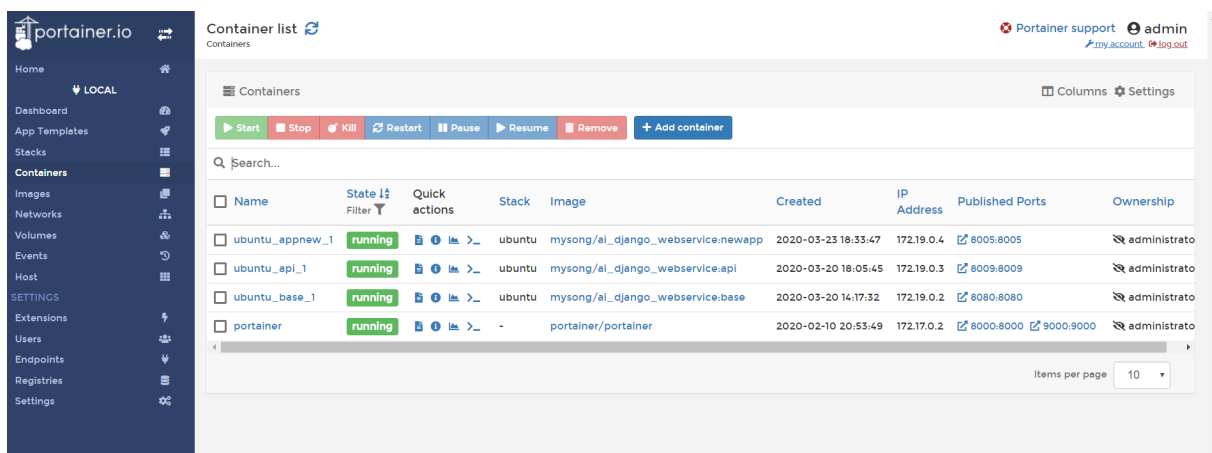


Figure 5 – Gestion de site Web

3 Guide du développeur

Les développeurs peuvent utiliser ce projet comme modèle pour développer leurs propres applications.

Les informations sur le développement du projet se trouvent dans le chapitre «Cahier de développeur», les étapes détaillées se trouvent dans sa quatrième section «Process de développement» .

D

Gestion de projet

Le plan de développement est un outil qui permet de formaliser des objectifs de développement de compétences, en matière de savoir, mais plus particulièrement de savoir faire et de savoir être. Le structure de découpage du projet est une décomposition hiérarchique des travaux nécessaires pour réaliser les objectifs d'un projet. Dans ce projet, je découpe le projet en plusieurs tâches afin de mieux organiser le projet.

En respectant le modèle en « cascade », on fait 3 cycles : l'étude de faisabilité, la spécification des besoins, et l'analyse pendant le semestre 9, et on fait les autres 4 cycles pendant le semestre 10 : la conception détaillée, l'implémentation, le test et la mise en place.

1 Aperçu de gestion de projet

Le diagramme de Gantt pour la planification de ce projet est :

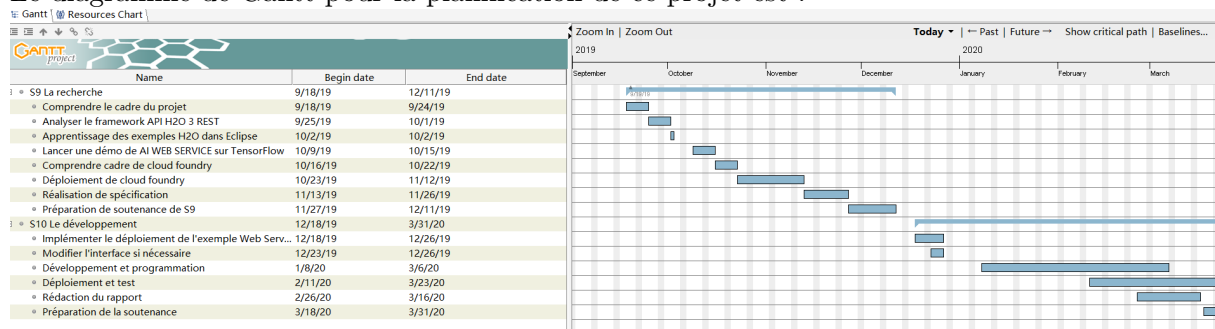


Figure 1 – La diagramme Gantt du planning

Pour le S9, le projet commence le 19/19/2019 par la compréhension du sujet et finit le 12/12/2019 par la soutenance de S9.

Pour le S10, le projet commence le 18/12/2019 en continuant à étudier et le projet finit le 01/04/2018 par la soutenance de S10.

1.1 Découpage des tâches

1. Comprendre le cadre du projet :

- Date de début : le 18 septembre 2019 ;
- Date de fin : le 24 septembre 2019 ;
- Durée : 6 jours ;

2. Analyser le framework API H2O 3 REST :

- Date de début : le 25 septembre 2019 ;
- Date de fin : le 1 octobre 2019 ;
- Durée : 6 jours ;

3. Apprentissage des exemples H2O dans Eclipse :

- Date de début : le 2 octobre 2019 ;
- Date de fin : le 8 octobre 2019 ;
- Durée : 6 jours ;

4. Lancer une démo de AI WEB SERVICE sur TensorFlow :

- Date de début : le 9 octobre 2019 ;
- Date de fin : le 15 octobre 2019 ;
- Durée : 6 jours ;

5. Comprendre cadre de cloud foundry :

- Date de début : le 16 octobre 2019 ;
- Date de fin : le 22 octobre 2019 ;
- Durée : 6 jours ;

6. Déploiement de cloud foundry :

- Date de début : le 23 octobre 2019 ;
- Date de fin : le 12 novembre 2019 ;
- Durée : 20 jours ;

7. Réalisation de spécification :

- Date de début : le 13 novembre 2019 ;
- Date de fin : le 26 novembre 2019 ;
- Durée : 13 jours ;

8. Préparation de soutenance de S9 :

- Date de début : le 27 novembre 2019 ;
- Date de fin : le 11 décembre 2019 ;
- Durée : 14 jours ;

9. Implémenter le déploiement de l'exemple Web Service :

- Date de début : le 18 décembre 2019 ;
- Date de fin : le 26 décembre 2019 ;
- Durée : 7 jours ;

10. Modifier l'interface si nécessaire :

- Date de début : le 23 décembre 2019 ;
- Date de fin : le 26 décembre 2019 ;
- Durée : 3 jours ;

11. Développement et programmation :

- Date de début : le 8 janvier 2020 ;
- Date de fin : le 17 janvier 2020 ;
- Durée : 9 jours ;

12. Déploiement et test :

- Date de début : le 20 janvier 2020 ;
- Date de fin : le 21 février 2020 ;
- Durée : 30 jours ;

13. Rédaction du rapport :

- Date de début : le 26 février 2020 ;
- Date de fin : le 16 mars 2020 ;
- Durée : 18 jours ;

14. Préparation de la soutenance :

- Date de début : le 18 mars 2020 ;
- Date de fin : le 31 mars 2020 ;
- Durée : 13 jours ;

2 Gestion des aléas

2.1 Le projet ne peut pas être achevé à temps

Il est très grave que le projet ne puisse être achevé à temps. Pour éviter cette situation, Il faut donc avoir un bon plan et le suivre strictement. Comme indiqué dans le plan ci-dessus, les livrables à chaque étape seront livrés à temps pour que le projet soit enfin terminé à temps. Dans le planning, j'ai également réservé du temps pour chaque tâche du plan afin de résoudre les imprévus.

Si les tâches de la phase de développement ont pris du retard et que je ne prévois pas de livrer à temps, je m'attacherai à terminer la livraison du document pour que les développeurs suivants puissent continuer à terminer la tâche conformément à mon document.

2.2 Progrès en retard à un certain stade

Étant donné que le PRD est un projet très important. Comme vous pouvez le constater sur la diagramme de Gantt, mon horaire ne comprend que les mercredis et les jeudis et ma division des tâches sont très détaillées. Par conséquent, si mon mentor et moi-même estimons que les progrès

sont à la traîne à un moment donné, je prendrai d'autres moments incluant les jours fériés pour promouvoir l'avancement des travaux.

2.3 Les livrables fournis ne répondent pas aux exigences

J'utiliserai des méthodes de développement agiles, je communiquerai donc fréquemment avec l'encadrant pour m'assurer que le projet avance.

E

Description des interfaces externes du système

1 Interfaces matériel/logiciel

Aucun matériel spécial n'est requis pour le développement de ce projet. Comme le diagramme de déploiement, nos serveurs sont déployés sur des plateformes cloud.

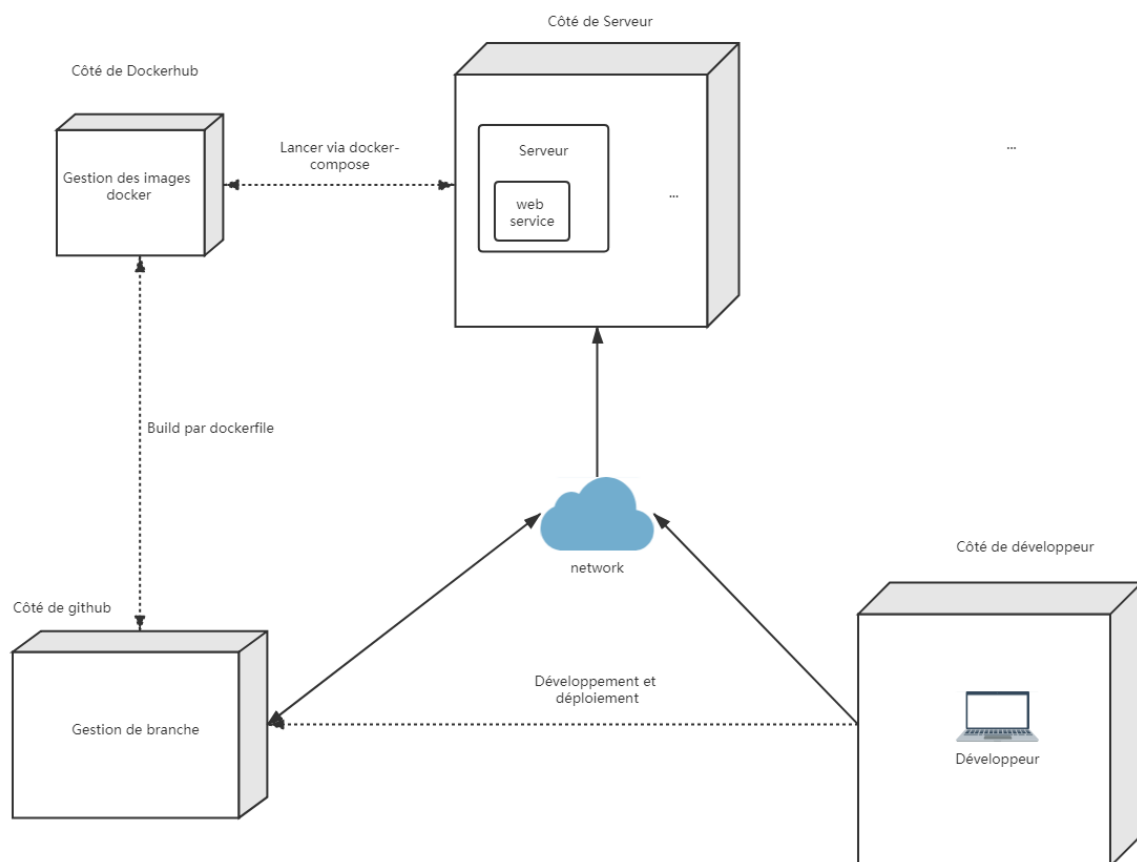
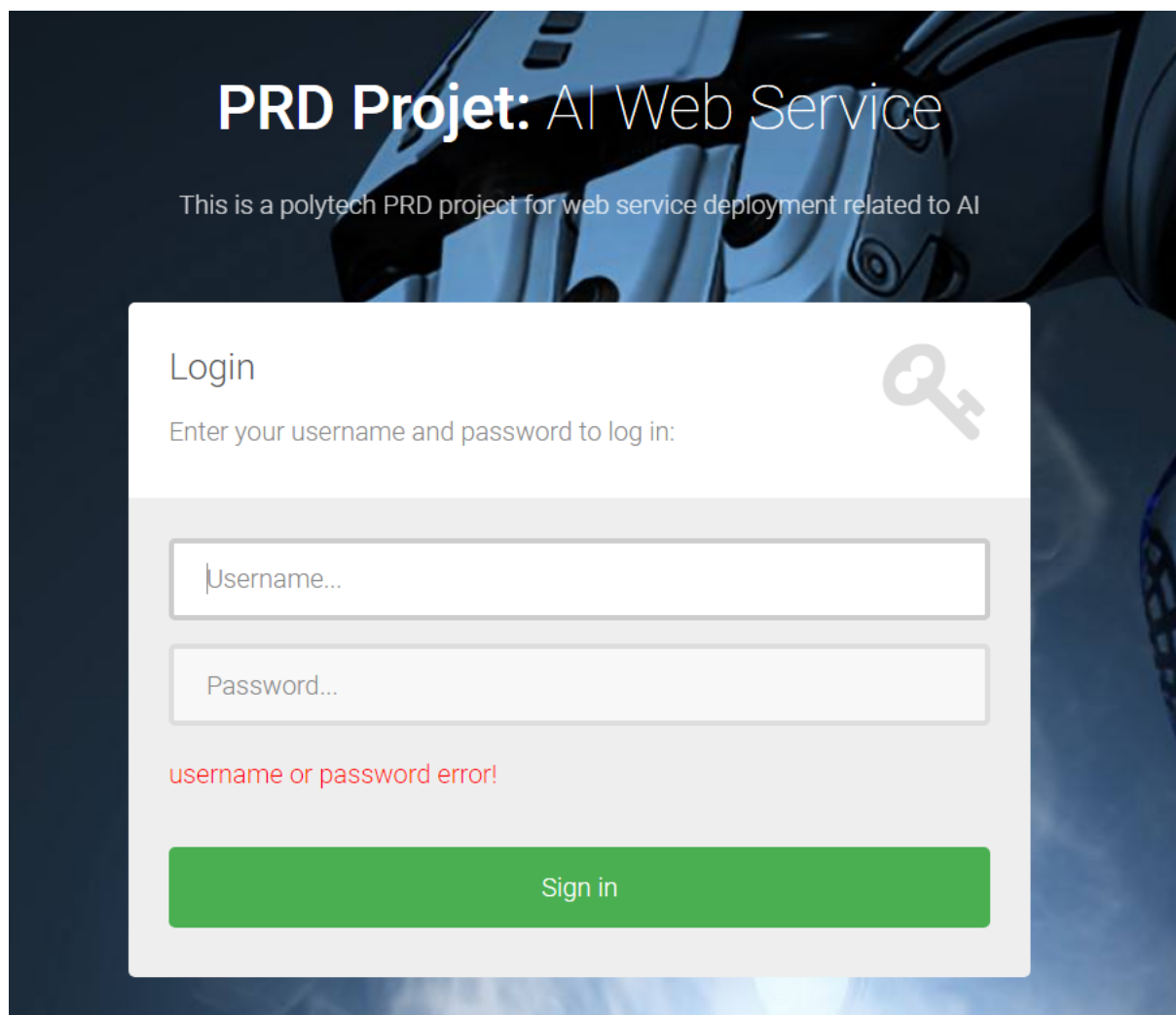


Figure 1 – – Diagramme de déploiement

2 Interfaces homme/machine

2.1 Page de login



The image shows a login page for 'PRD Projet: AI Web Service'. The background is a dark blue image of a robotic arm. The page has a white login form in the center. The form has a title 'Login' and a subtitle 'Enter your username and password to log in:'. There are two input fields: 'Username...' and 'Password...'. Below the password field, there is a red error message 'username or password error!'. At the bottom of the form is a green 'Sign in' button. A key icon is visible in the top right corner of the form area.

PRD Projet: AI Web Service

This is a polytech PRD project for web service deployment related to AI

Login

Enter your username and password to log in:

Username...

Password...

username or password error!

Sign in

Figure 2 – *Page de login*

2.2 Page d'accueil

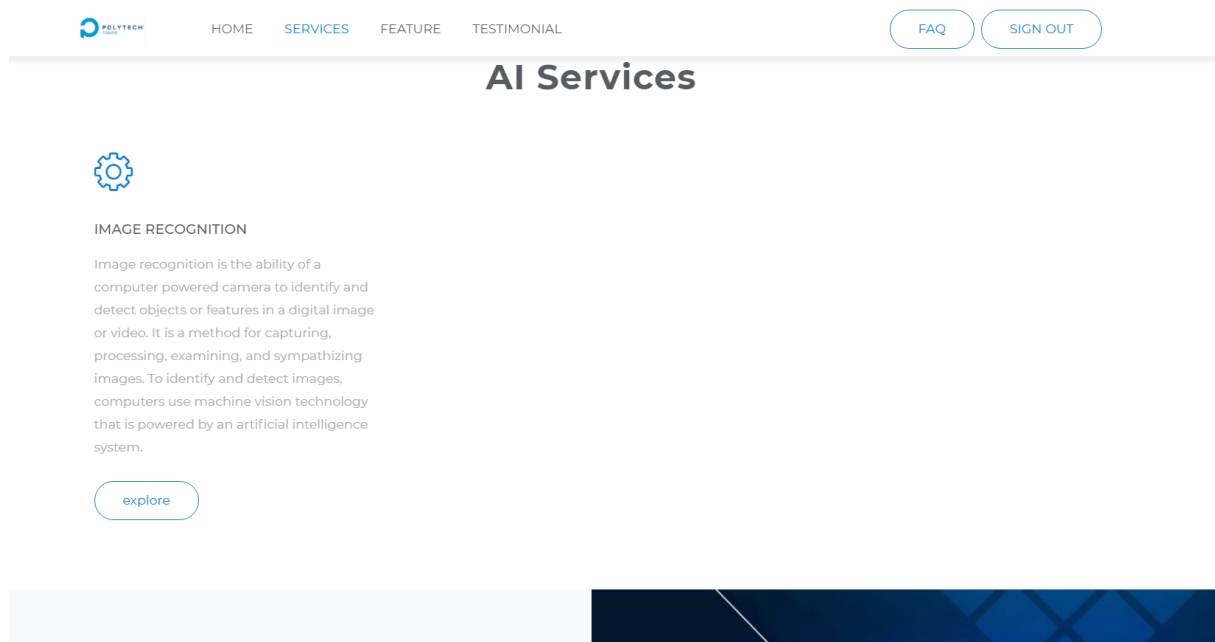


Figure 3 – Page d'accueil

2.3 Page de service

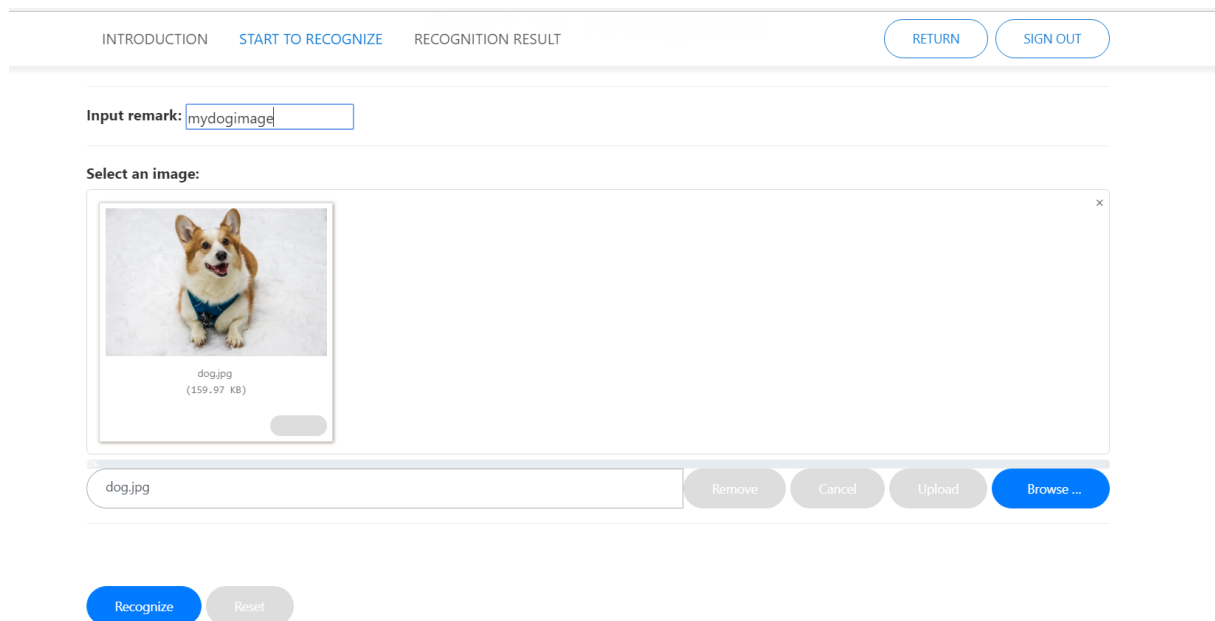


Figure 4 – Page de service

2.4 Page d'administration

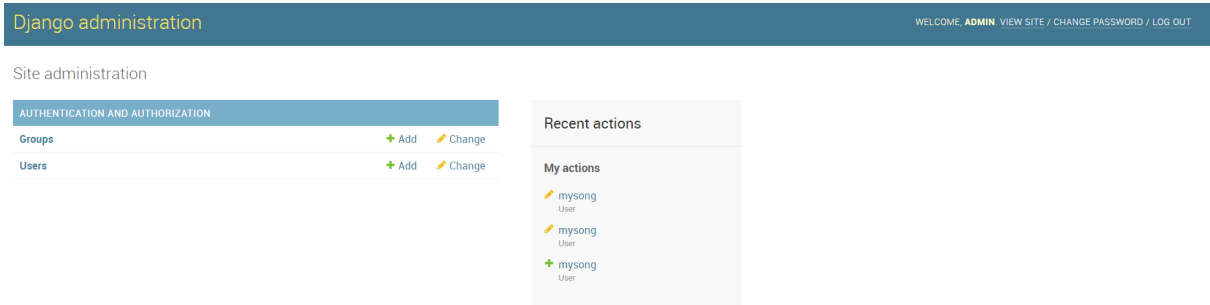


Figure 5 – Page d'administration

2.5 Page de gestion de web site

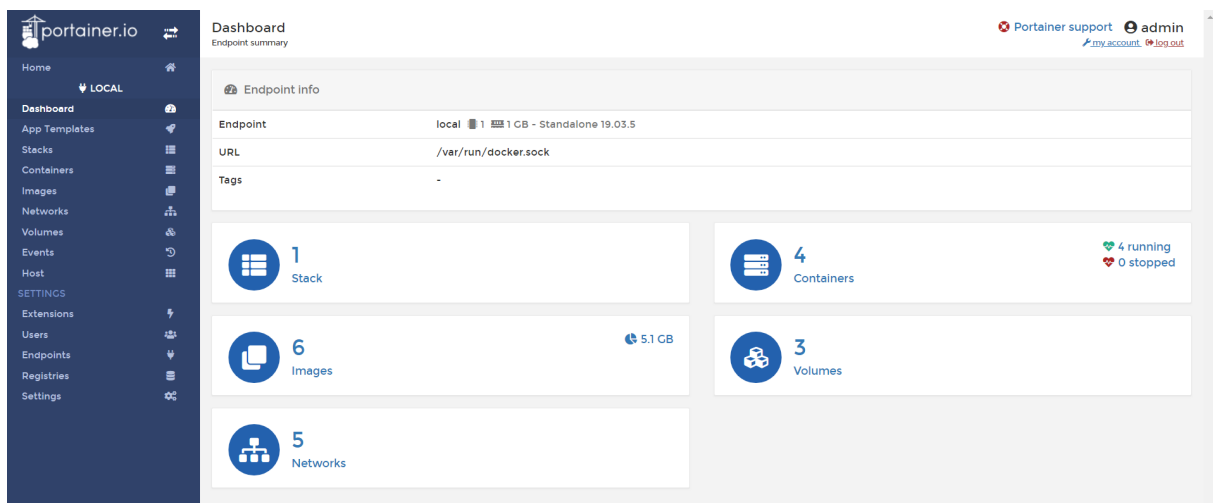


Figure 6 – Page de gestion de web site

F

Spécifications fonctionnelles

L'objectif de cette partie est de décrire l'ensemble des fonctions du système en précisant avec quels composants elles interagissent.

1 Fonction de Authentification

Présentation :

- Nom : Authentification
- Rôle : L'authentification permettre aux utilisateurs de se connecter.

Priorité : primordiale

Entrées : Données d'authentification saisies par l'utilisateur

Sorties : -

Exceptions : Les données d'entrée n'existent pas.

2 Fonction de Upload

Présentation :

- Nom : Upload
- Rôle : L'utilisateur télécharge l'image qu'il souhaite traiter.

Priorité : primordiale

Entrées : Données d'image

Sorties : -

Exceptions : Les données sont trop volumineuses

3 Fonction de Choix de service

Présentation :

- Nom : Choix de service
- Rôle : L'utilisateur choisit service qu'il souhaite utiliser.

Priorité : primordiale

Entrées : Options envoyées par l'utilisateur

Sorties : -

Exceptions : -

4 Fonction de reconnaissance d'image

Présentation :

- Nom : Traitement d'image.
- Rôle : Le système utilise l'algorithme pour traiter les images.

Priorité : primordiale

Entrées : Image envoyée par l'utilisateur

Sorties : Résultat de reconnaissance

Exceptions : Exception d'IO

5 Fonction de Download

Présentation :

- Nom : Download
- Rôle : L'utilisateur télécharge l'image traitée.

Priorité : primordiale

Entrées : -

Sorties : L'image

Exceptions : Délai d'attente de téléchargement.



Spécifications non fonctionnelles

1 Contraintes de développement et conception

Les contraintes de développement :

- Matériels : un IDE pycharm
- Langage de programmation : Python 3.7
- Logiciels et bibliothèques à utiliser pour le développement : le logiciel Anaconda3 pour la gestion de bibliothèques de python dans local.

2 Contraintes de fonctionnement et d'exploitation

Cette section décrit les dispositions qu'il est nécessaire de prendre en compte pour les différentes conditions de fonctionnement du système.

En raison de l'utilisation de Docker comme environnement d'exploitation, tout système avec Docker installé peut exécuter des services Web bien conçus.

2.1 Performances

La performance réside dans le nombre de déploiements de service web et la vitesse de reconnaissance des images.

2.2 Contrôlabilité

Ce projet utilise les outils de gestion et les interfaces attachés à la plateforme de serveur pour gérer le fonctionnement de Web service.

2.3 Sécurité

Ce projet utilise le framework de session django pour le contrôle de sécurité des utilisateurs.

- **25 Septembre 2019 :**

J'ai trouvé des informations pertinentes et j'apprends maintenant des technologies connexes. Je continuerai à travailler pour trouver des exemple.

- **26 Septembre 2019 :**

J'ai trouvé des projets liés. Mais en utilisant différents cadres IBM Model Asset eXchange (MAX). L'utilisation de cet exemple est L'utilisateur envoie l'image à l'API du modèle à l'aide de l'interface utilisateur Web. L'API du modèle renverra les données de l'objet et l'interface utilisateur Web affichera l'objet détecté. L'utilisateur interagit avec l'interface Web pour afficher et filtrer les objets détectés.

Je pense que cet exemple est assez représentatif. Je continuerai à comparer différents frameworks et à implémenter des fonctionnalités.

- **3 Octobre 2019 :**

J'ai essayé de déployer le framework MAX sur un système Linux, mais cela n'a pas fonctionné. Il est implémenté à l'image du docker. Je pense que c'était un problème avec mon utilisation de docker. De plus, je ne trouve toujours pas de meilleur exemple concernant H2O. Je viens maintenant d'étudier les exemples fournis par H2O. J'apprends encore à propos de l'API H2O REST. Je vais essayer de trouver plus d'exemples et de mener des recherches.

- **17 Octobre 2019 :**

J'ai trouvé une démo qui implémente le Web service et l'algorithme(yolo3) de détection de cible. Maintenant j'essaye de l'exécuter. Son noyau est une implémentation Keras de YOLOv3 (backend Tensorflow). En fait, j'ai trouvé que c'était un framework très populaire (Tensorflow + Keras). Le projet de démo est basé sur Python. Mais la configuration de l'environnement dont elle a besoin est un peu stricte. J'essaie de comprendre la relation, et de lancer un exemple simple d'implémentation de yolo3.

Ci-dessous quelques adresses de code source

<https://github.com/wizyoung/YOLOv3TensorFlow>

<https://github.com/qqwweee/keras-yolo3>

<https://github.com/sherlockchou86/vision-web-service>

- **24 Octobre 2019 :**

J'essaie de comprendre cloud foundry. Maintenant, terminez l'installation de la CLI. Et essayer de déployer l'application sur le cloud. En fait, j'ai passé trop de temps sur bosh C'est l'une des deux plateformes majeures de Cloud Foundry. C'est plus sur les applications distribuées. En comprenant, je pense que notre service Web est déployé sur la plateforme Application Runtime. J'ai trouvé qu'il y a des plugins qui peuvent l'utiliser. Et je peux également utiliser les outils fournis par cloud foundry il-même (CLI). J'ai rencontré des problèmes lors du déploiement de l'application. Je résous des problèmes d'installation du paquet d'environnement ruby....

J'ai quelques questions à vous poser Pour CdS, quelle partie devrais-je écrire maintenant ? À quoi devrais-je faire attention ?

- **28 Novembre 2019 :**

J'essaie de comprendre cloud foundry. Maintenant, terminez l'installation de la CLI. Et essayer de déployer l'application sur le cloud. En fait, j'ai passé trop de temps sur bosh C'est l'une des deux plateformes majeures de Cloud Foundry. C'est plus sur les applications distribuées. En comprenant, je pense que notre service Web est déployé sur la plateforme Application Runtime. J'ai trouvé qu'il y a des plugins qui peuvent l'utiliser. Et je peux également utiliser les outils fournis par cloud foundry il-même (CLI). J'ai rencontré des problèmes lors du déploiement de l'application. Je résous des problèmes d'installation du paquet d'environnement ruby....

J'ai quelques questions à vous poser Pour CdS, quelle partie devrais-je écrire maintenant ? À quoi devrais-je faire attention ?

- **28 Novembre 2019 :**

Pour ces deux jour, j'ai écrit le cahier de spécification. J'ai également fait un résumé des informations que j'ai appris. Pour le Cds, j'ai fait le diagramme de cas d'utilisation et le diagramme d'activité. Mais c'est version simple, je continuerai à modifier et à améliorer. Je vais utiliser ce week-end pour finir de finition des Cds.

Ce matin je suis allé au rdv de suivi de projet. La prof de l'entreprise m'a rappelé de préparer un plan pour la réalisation du projet. Je vais commencer à faire ces choses en même temps. Pour la semaine prochaine, je pense je peut finir les Cds et le plan de réalisation. Si vous avez le temps la semaine prochaine, puis-je prendre rendez-vous avec vous pour vous pouvez me donner quelques conseils ?

- **1 Décembre 2019 :**

Ce sont quelques images de mes idées actuelles concernant le projet à mettre en œuvre. Pour les utilisateurs, j'ai ajouté le rôle de administrateur pour effectuer certaines tâches de gestion des services Web.

Je ne suis pas sûr des détails de la fonction (diagramme de cas d'utilisation). Pouvez-vous me donner des conseils Que dois-je ajouter ou supprimer...

Pour la mise en œuvre, je veux adopter un méthode agile. Donc il peut y avoir des changements de ces diagrammes dans le futur.

- **19 Décembre 2019 :**

J'ai installé l'interface graphique de DOCKER sous Windows (avec linux vm) : Kitematic. Et elle marche bien avec page de gestion. Mais lorsque j'utilise l'interface graphique pour télécharger une nouvelle image, elle continue de s'afficher lorsqu'elle est connectée au docker hub. J'essaye de résoudre ça.

- **9 Janvier 2020 :**

Tout d'abord, bonne année

J'ai terminé le déploiement du projet dans docker et implémenté la fonction de connexion.

Maintenant, le projet utilise le langage python et est basé sur pycharm ide. J'ai utilisé le framework django pour le développement. Maintenant, après le lancement du projet, il fonctionnera sur docker comme conteneur, et je peux utiliser Kitematik (GUI) pour gérer. J'ai trouvé une image avec la configuration python et django en tant qu'environnement cloud en ligne.

Ensuite, je continuerai le développement du module de téléchargement de fichiers.

- **29 Janvier 2020 :**

Maintenant, j'ai déployé le UI de portainer. Il fait un bon travail de gestion de docker. (Il existe sous forme de container).

J'ai également déployé le service précédent sur docker. (pas par pycharm) et créé le volume de container.

J'ai encore des doutes sur le déploiement d'un nouveau service web (ajouté par d'autres), il s'agit principalement de l'implémentation utilisant la technologie docker. Parce que maintenant je peux implémenter la communication host et container via le volume, je peux également implémenter la communication container et container via le volume. Mais si d'autres ajoutent leur propre container à notre container (en tant que container parent), comment faire fonctionner leur service Web est un problème.

J'ai trouvé que Docker Compose pourrait être une solution au problème de coordination, je l'apprends maintenant.

Si vous avez le temps, puis-je demander un rdv pour déterminer avec vous les détails techniques ?

- **6 Février 2020 :**

Après quelque apprentissage et de pratique, j'ai optimisé l'ensemble du processus de développement.

Le processus actuel est 1. Programmation et écriture de fichiers tels que dockerfile sur notre propre IDE (comme pycharm ou autre outil). 2. Téléchargez le code sur github. 3. Associer un Docker hub à Github et nous pouvons construire notre image directement sur le docker hub. 4. Construisez notre container sur un serveur ou n'importe quel host machine. L'avantage est que l'ensemble du processus de développement s'organise et rend le processus de développement très clair.

À propos du déploiement de containers du service Web Django que nous avons discuté plus tôt, j'essaie d'utiliser certains fichiers publics comme volume partagé. C'est faisable, mais il y a aussi quelques problèmes : Lors de l'ajout d'un nouveau service Web, nous devons apporter quelques modifications à ces volumes partagés. Pour que les développeurs ajoutent plus facilement je souhaite que le volume partagé du nouveau conteneur écrase directement le volume partagé précédent. Mais le mécanisme du docker est l'héritage des volumes partagés cela signifie que si cette approche est adoptée, les développeurs devront aller dans le nouveau container et apporter des modifications au volume partagé. Une solution consiste à laisser l'ancien container hériter du nouveau conteneur pendant le déploiement j'essaie toujours s'il y a une nouvelle solution.

Pour automatiser portainer et notre service Web, j'ai trouvé que docker-compose était une bonne solution parce que notre portainer est également un container, l'utilisation de docker-compose peut coordonner l'ouverture de ces containers. (c'est comme un script de docker cmd) Dans django, il y a une gestion de fond django-admin qui vient avec le système. Je vais le pratiquer à l'avenir pour voir si cela peut aider les problèmes rencontrés avant.

Dans le même temps, j'approfondirai l'apprentissage et la programmation du framework django afin de faire plus d'optimisations sur l'architecture globale.

- **13 Février 2020 :**

Actuellement, j'ai terminé certains services Web : page de connexion, page d'accueil et un service qui peut télécharger des photos (il peut les réutiliser dans tous les algorithmes). Django est livré avec un service de gestion pour les utilisateurs. Je l'ai essayé, et c'est faisable.

Lorsque j'ai envisagé le déploiement de Docker sur le serveur cloud, j'ai constaté que l'AWS d'Amazon (service cloud) dispose d'une période d'essai gratuite d'un an. J'ai donc essayé de configurer le serveur cloud et d'installer Docker, et nous pouvons maintenant déployer avec succès notre service en ligne.

Je vais continuer à améliorer le déploiement de divers services dans docker.

- **16 Février 2020 :**

J'ai ajouté vos comptes github et dockerhub en tant que co-créateurs.

Le nom du projet est `Ai_Django_WebService`. Et mon compte est SIGESI en github, MYSONG en dockerhub.

Parfois, la page d'accueil de github peut ne pas recevoir d'invitations. Dans ce cas, vous pouvez vérifier l'adresse e-mail associée, il y aura un lien d'invitation.

Au cours des dernières tentatives, il y a eu quelques problèmes avec le volume partagé, et je continuerai à apporter des modifications.

- **5 Mars 2020 :**

Après avoir appris et pratiqué le `django restframework`, j'ai développé trois API utilisables pour implémenter la reconnaissance d'images, upload et download, et ils sont sur le branch de `webservice_api_branch`.

L'api pour la reconnaissance d'images, au début, je voulais utiliser le GCP de Google, mais mon compte Google a des problèmes de sécurité et est en cours de vérification. J'ai donc cherché d'autres API open source. Maintenant, utilisez l'API baidu AI, dans le test, il peut bien terminer la tâche de reconnaissance.

Je continuerai d'intégrer l'API dans l'application django et de la tester.

- **11 Mars 2020 :**

La vm sur amazon est à mon nom (Parce que il faut ajouter une carte bancaire pour créer un compte). Et je peux définir s'elle est toujours en ligne. Et je l'ai fermé hier.

En fait, pendant le processus de développement, j'ai utilisé la machine virtuelle locale pour les tests, et ils fonctionnent bien. Étant donné que tous les déploiements sont basés sur Docker, il n'a pas de problèmes de configuration complexes. Tant qu'un serveur dispose d'un Docker (VM et serveur locale ou serveur cloud. . .), il peut s'exécuter. J'ai testé sur mon ordinateur personnel basé sur Windows et sur vm basé sur Linux. Tous les déploiements peuvent s'exécuter.

J'ai rouvert le serveur Amazon l'adresse actuelle est 15.236.19.94. (avec port 9000 portainer, 8005 login/index, 8001 img_rec, 8009 api) Je n'ai pas utilisé Elastic IP donc, chaque fois que je ferme, l'IP change. (Amazon va redistribuer)

- **20 Mars 2020 :**

Pour mon projet j'ai ajouté la gestion de session (Les utilisateurs doivent se connecter pour accéder à d'autres pages) . Et j'ai aussi ajouté des invites d'erreur (Empêcher les utilisateurs d'oublier de saisir des remarques et des images). J'ai résolu le problème que la remark doit être différente (En utilisant le mécanisme de session).

J'ai également optimisé les résultats de reconnaissance (Afficher l'image, masquer le formulaire vierge). J'ai ajouté des hyperliens vers la page d'index (Dans la section Feature) pour

pointer vers différents référentiels.

J'ai redéployé des aws Amazon. Vous pouvez y accéder via l'URL suivante

- **23 Mars 2020 :**

J'ai amélioré certaines des erreurs que vous avez mentionnées. - ajouter sign out pour quitter session - La remarque n'est plus nécessaire maintenant - Erreur json corrigée (En fait, c'est toujours une erreur causée par la remarque. J'ai ajouté des nombres aléatoires pour le rendre unique.)

Et j'ai également écrit une nouvelle classe pour gérer toutes les adresses IP redirigées de manière unifiée. Il suffit maintenant d'annoter une méthode pour terminer les tests de l'environnement local et la transformation de l'environnement de production.

Je prépare maintenant ma soutenance Qualité mercredi. Si j'ai le temps, j'essaierai d'ajouter de nouveaux services au système tels que la reconnaissance faciale.

- **2 Avril 2020 :**

Après quelques débogages, j'ai optimisé ce que vous avez dit : - Lorsque l'on lance la reconnaissance, le retour sur la page se fasse directement sur la section "results". - J'ai changé l'URL en `http://15.236.19.94:8005/img_rec_action/result` J'ai également effectué des tests pour résoudre l'échec de connexion causé par la génération de pydoc.

À propos de la documentation, j'ai essentiellement terminé le rapport. Pour des instructions détaillées, je le mets dans les annexes C Document d'utilisation.

- **10 Avril 2020 :**

Pour mon PRD projet, afin de simplifier l'accès et les tests, j'ai utilisé le nom de domaine liée au ip. Vous pouvez maintenant accéder directement à l'adresse suivante pour accéder à la page de connexion : - `www.mysong.technology`

I

Bibliographie

- [1] <https://opensource.com/resources/what-docker>
- [2] <https://www.cloudfoundry.org/>
- [3] <https://www.apriorit.com/dev-blog/534-using-docker-developing-web-service>
- [4] <https://www.h2o.ai/products/h2o/>
- [5] <https://github.com/tensorflow/tensorflow>
- [6] <https://www.tensorflow.org/>

Une IA en Web service pour la connaissances d'image

Miyuan Song

Encadrement : Barthelemy Serres

Contexte

Afin de faciliter l'utilisation des open sources des IA et d'effectuer des recherches connexes, ce projet portera sur l'exploration d'un moyen de déployer rapidement des Web services liés à l'IA. L'accès à ce serveur se devra d'être simple et permettra aux utilisateurs de pouvoir mettre en oeuvre les bonnes pratiques de domaine d'AI.

Objectifs

Ce projet créera un ensemble de modèles de services Web qui utilisent des ressources d'IA pour les développeurs. Pour les utilisateurs ordinaires, ce projet crée des services Web et des applications pour la reconnaissance d'images basées sur des modèles

Mise en œuvre

J'ai construit des services web basés sur le framework django et l'ai déployé sur le serveur cloud en fonction de l'image docker. Et j'utilise l'API de reconnaissance d'image comme interface de service.

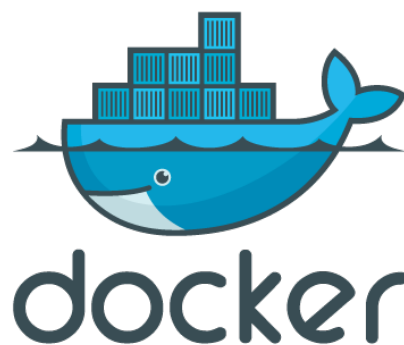


Figure 1 – – Docker

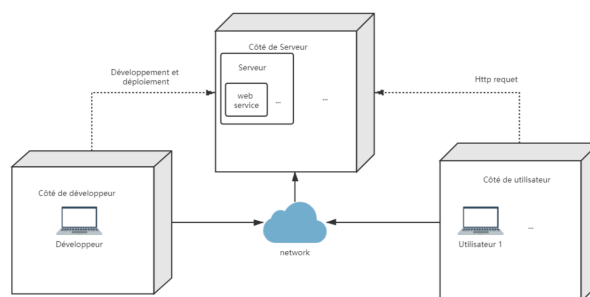


Figure 2 – – Diagramme de structure

IMAGE

api

Last updated 6 days ago by [mysong](#)

DIGEST

[dcc6faf1f99d](#)

Figure 3 – – Image de docker

Une IA en Web service pour la connaissances d'image

Miyuan Song

Encadrement : Barthélemy Serres

Contexte

Afin de faciliter l'utilisation des open sources des IA et d'effectuer des recherches connexes, ce projet portera sur l'exploration d'un moyen de déployer rapidement des Web services liés à l'IA. L'accès à ce serveur se devra d'être simple et permettra aux utilisateurs de pouvoir mettre en oeuvre les bonnes pratiques de domaine d'AI.

Objectifs

Ce projet créera un ensemble de modèles de services Web qui utilisent des ressources d'IA pour les développeurs. Pour les utilisateurs ordinaires, ce projet crée des services Web et des applications pour la reconnaissance d'images basées sur des modèles

Mise en œuvre

J'ai construit des services web basés sur le framework django et l'ai déployé sur le serveur cloud en fonction de l'image docker. Et j'utilise l'API de reconnaissance d'image comme interface de service.

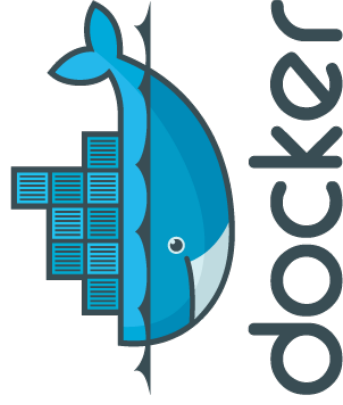


Figure 4 – – Docker

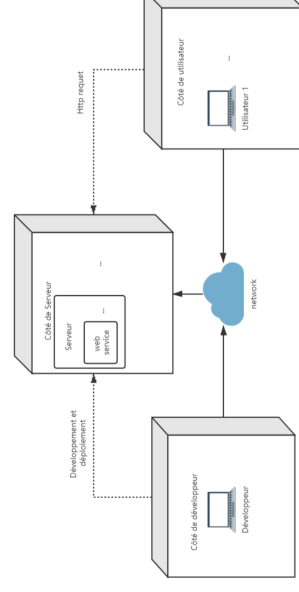


Figure 5 – – Diagramme de structure

IMAGE
api
Last updated 6 days ago by mysong
DIGEST
dcc6faf1f99d

Figure 6 – – Image de docker

Une IA en Web service pour la connaissances d'image

Résumé

Le sujet de Projet de Recherche et Développement est "Une IA en Web service pour la connaissances d'image". L'objectif de ce document est de présenter les documentations de ce projet. Ce projet utilise Docker pour déployer des services Web pour obtenir la reconnaissance d'image

Mots-clés

Projet PRD, Polytech, Web service, Connaissances d'image

Abstract

The subject of the Research and Development Project is "Une IA en Web service pour la connaissances d'image". The objective of this document is to present the documentation of this project. This project uses docker to deploy web services to achieve image recognition

Keywords

PRD Project, Polytech, AI, Web service, Image recognition