

ECOLE POLYTECHNIQUE DE L'UNIVERSITÉ FRANÇOIS RABELAIS 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 2017-2018

Création d'une plateforme pour la reconnaissance d'insectes

Tuteur académique
Maxime MARTINEAU

Étudiant
Pierre JUILLIE (DI5)

2 avril 2018



Liste des intervenants

Nom	Email	Qualité
Pierre JUILLIE	pierre.juillie@etu.univ-tours.fr	Étudiant DI5
Maxime MARTINEAU	maxime.martineau@univ-tours.fr	Tuteur académique, Département Informatique



Avertissement

Ce document a été rédigé par Pierre JUILLIE susnommé l'auteur.

L'Ecole Polytechnique de l'Université François Rabelais de Tours est représentée par Maxime MARTINEAU susnommé 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 assument 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

Pierre JUILLIE, *Création d'une plateforme pour la reconnaissance d'insectes*, Projet Recherche & Développement, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2017-2018.

```
@mastersthesis{
  author={JUILLIE, Pierre},
  title={Création d'une plateforme pour la reconnaissance d'insectes},
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université François Rabelais de Tours},
  address={Tours, France},
  year={2017-2018}
}
```

Table des matières

Liste des intervenants	a
Avertissement	b
Pour citer ce document	c
Table des matières	i
Table des figures	iv
1 Introduction	1
1 Contexte et Acteurs	1
2 Objectifs	1
3 Hypothèses	2
4 Bases méthodologiques	2
2 Description générale	3
1 Environnement du projet	3
2 Caractéristiques des utilisateurs	4
3 Fonctionnalités du système	4
4 Structure générale du système	6
3 Étude de l'existant et vieille technologique	8
1 Description de l'existant.....	8
2 Comparatif des frameworks Python	9
2.1 Intérêt d'un framework	9
2.2 Procédure	10
2.3 résultat	11

3	Django.....	11
4	Fonctionnement HTML et rendu	13
4	Analyse et conception	14
1	Formulation définitive du projet.....	14
2	Description Algorithmme.....	14
3	Critiques et limites	15
3.1	Technologiques	15
3.2	Conceptuelles	15
4	Déroulement et prévisions	15
5	Mise en oeuvre	17
1	Environnement	17
2	Implémentation	17
3	Création du contenu	19
4	Mise en place des Tests.....	20
5	Déploiement serveur	20
6	Bilan et conclusion	22
1	Ce qui est fait et reste à faire	22
2	Analyse qualité	22
3	Analyse gestion de projet	22
	Remerciements	24
	Annexes	25
A	Conception	26
B	Document d'installation	30
1	Contraintes fonctionnelles	30
2	Installation de l'environnement.....	30
3	Installation des librairies.....	31
4	Configuration de la base de données	31
5	Lancement du projet	31
C	Cahier du développeur	33
1	Avant-propos	33
2	Langage, framework, librairies et convention de nommage.....	33
3	Documentation du code	33
4	Structure du système	34
5	Autres informations importantes.....	35

D	Cahier de tests	36
1	Avant Propos.....	36
2	Où trouver les tests.....	36
3	Fonctionnalités testées explicitement	36
4	Lancer les tests.....	37
	Webographie	38

Table des figures

2 Description générale

1	Schéma de l'application.....	4
2	Diagramme des cas d'utilisation	5
3	Tableau des fonctionnalités par ordre de priorité	6
4	Structure du projet	7

3 Étude de l'existant et vieille technologique

1	Diagramme de l'application existante	8
2	Exemple de la page Ajout Image.....	9
3	Tableau du comparatif des frameworks.....	11
4	Fonctionnement Django	12
5	Fonctionnement WSGI	12

4 Analyse et conception

1	Exemple image d'entrée	15
2	Diagramme de Gantt	16
3	Diagramme de Gantt 2.....	16

5 Mise en oeuvre

1	Structure du projet	18
2	Structure de la BDD	19

A Conception

1	Première version du diagramme des cas d'utilisation.....	27
---	--	----

2	Mockup de la page d'accueil	28
3	Mockup de la page d'ajout d'images	29
 C Cahier du développeur		
1	Structure du projet	34

1

Introduction

Ce document a pour but de dresser l'ensemble des spécifications ainsi que d'expliquer le déroulement du Projet de Recherches et Développement visant à créer une application qui hébergera un algorithme de reconnaissance des insectes.

1 Contexte et Acteurs

Dans le domaine du vivant et par extension, dans celui des insectes, un des enjeux pour les biologistes est souvent de pouvoir reconnaître un individu. De ce fait, il est dans leur intérêt et dans celui de nombreuses personnes d'être capable de mettre à disposition de tous un algorithme permettant la reconnaissance des êtres vivants pris en photo. Ainsi, un algorithme de reconnaissance d'insectes a été développé mais il doit maintenant être hébergé pour faciliter sa distribution et sa maintenance.

La demande a été formulée suite à un partenariat au sein de l'université de Tours et le client est représenté par Damien Munier, travaillant au CETU innophyt. Ce projet est encadré par Maxime Martineau et mis en place par moi-même, Pierre Juillié. Gilles Venturini et Romain Raveaux ont aussi participé dans la conception du projet et son encadrement.



Une première application pré-existante m'a été fournie au début du projet. Elle a fait office d'exemple et nous aborderons dans la suite son intérêt.

2 Objectifs

L'objectif principal est de créer une application web accessible à partir de différentes plateformes. Pour cela, une exploitation de la première application est prévue mais aussi une amélioration suivant des critères et des objectifs définis dans ce document.

Le site final devra accueillir la fonctionnalité principale qu'est la reconnaissance d'insectes, mais proposera plusieurs autres fonctionnalités liées à la reconnaissance et à l'utilisateur le tout s'articulant de manière logique afin de garder un contenu simple et efficace pour tout nouvel utilisateur.

Un des objectifs finaux de ce projet est de pouvoir intégrer des parties payantes afin d'auto-financer l'hébergement et le temps accordé à ce site par les scientifiques.

3 Hypothèses

Plusieurs hypothèses ressortent dans la spécification de ce projet :

- Le projet étant structuré par une méthode agile, des sprints vont être créés pour implémenter les différentes fonctions définies avec le client. De ce fait, si celui-ci souhaite corriger ou changer une fonctionnalité, ou implémenter une nouvelle fonctionnalité qui remet en cause le modèle précédent, ceci peut impacter le délai du projet.
- La conception initiale prévoyait la construction d'une application web. Il a été évoqué par la suite qu'une utilisation possible de l'application se fera par mobile. L'application sera donc développée pour soutenir les deux possibilités, cependant il n'est pas convenu dans le temps initiale de construire une application mobile par la même occasion.
- L'intégration de services payants est prévue dans la construction de ce site. Cependant la gestion des transactions bancaires et/ou des autres moyens de paiement n'étant pas de mon ressort, il se peut que ces fonctionnalités ne soient pas entièrement opérationnelles dans la dernière version de la plateforme.
- Il n'est pas prévu dans le développement de modifier ou comprendre le principe de reconnaissance appliquée à l'image. L'algorithme ne sera pas modifié par moi-même durant ce projet, il sera simplement appelé avec une image et le résultat sera récupéré.

4 Bases méthodologiques

Le site ayant été découpé en plusieurs fonctionnalités claires et simples, il paraît plus intelligent et pratique de pouvoir effectuer les tâches une par une et proposer des livrables au fur et à mesure. Une méthode agile paraît donc plus appropriée.

Des outils de gestion de projet seront utilisés comme un Trello, un logiciel de versioning qui fera aussi office de dépôt, tel que Git.

Les tests de l'application auront lieu soit un serveur local grâce à un Wampserver, ou alors via un environnement virtuel de python ou même via Vagrant.

2

Description générale

1 Environnement du projet

Comme précisé auparavant, une application assez primaire avait été développée dans le même langage. Cependant la structure du logiciel ne permet pas sa réutilisation totale et seulement quelques parties pourront être conservées dans notre application. Même si ces parties aideront l'avancement de notre application, elles ne constituent pas un point de départ pour celle-ci.

Le site sera une application web accessible sur toutes les plateformes (téléphone, tablette, ordinateur, ..). Il assurera la liaison entre les utilisateurs et l'algorithme de reconnaissance qui est la fonctionnalité principale du site. Pour cela l'utilisateur pourra déposer des images et lancer la reconnaissance dessus. Des fonctionnalités payantes pourront être greffées pour améliorer les résultats d'un point de vu qualitatif et informatif. Les interactions avec l'utilisateur resteront simples et efficaces.

Tout le développement de l'application se détache de l'algorithme de reconnaissance d'images qui sera simplement intégré et appelé au sein du code lorsqu'il sera nécessaire. Il n'y a pas de changements d'algorithme de prévu, les seules interventions seront éventuellement au niveau des entrées et sorties de l'algorithme si nécessaire.

On peut en revanche tenir compte de certains travaux en parallèles d'étudiants qui travailleraient sur du traitement d'images tant au niveau de la reconnaissance que sur l'amélioration de la qualité d'une image (cadrage, netteté, ...).

On peut visualiser l'environnement simplifié de notre programme d'un point de vu utilisateur dans le schéma suivant :

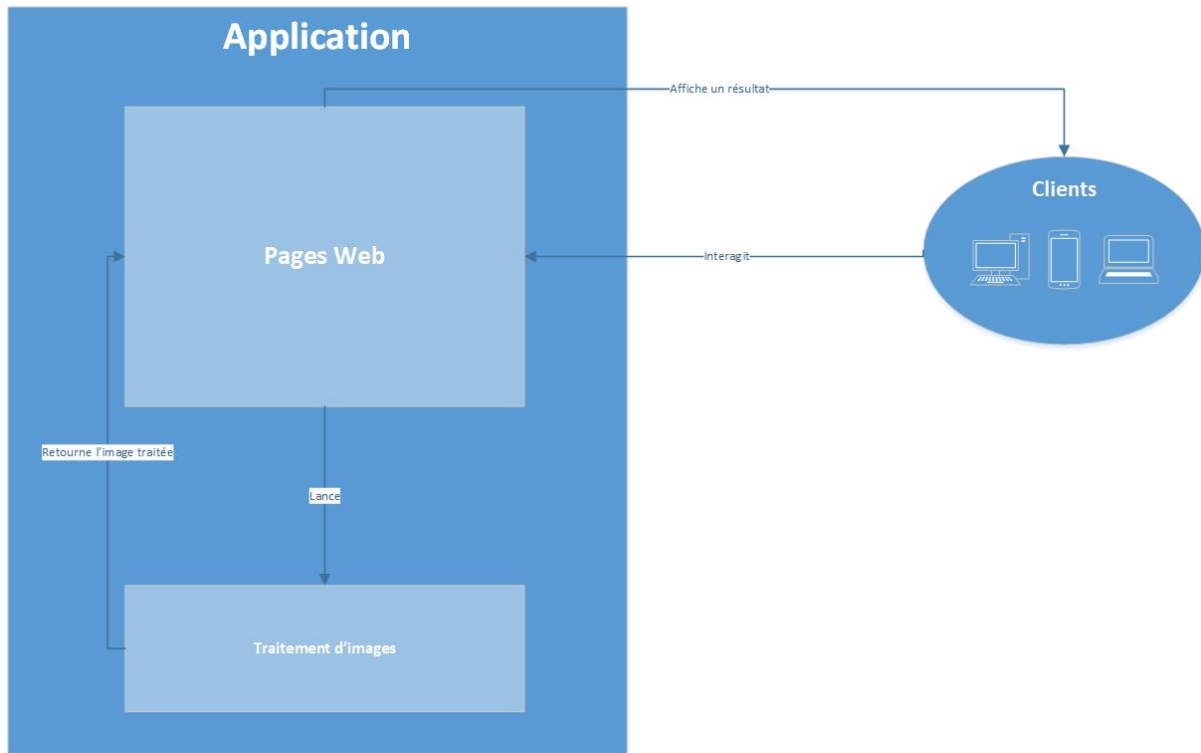


Figure 1 – Schéma de l'application

2 Caractéristiques des utilisateurs

Trois types d'utilisateurs ont été identifiés :

- **Simple** : Peuvent être des personnes sans aucune expérience du site ou de l'informatique. Ils peuvent naviguer sans difficulté sur les différentes pages et accéder aux fonctionnalités de base définies par le diagramme utilisateur.
- **Payant** : Correspond à un utilisateur simple ayant payé l'évolution de son statut. Il est donc nécessairement passé par un profil simple. Il possède toutes les capacités d'un utilisateur simple mais débloque des informations supplémentaires sur les résultats. Les différentes pages de l'application ne seront que peu ou pas impactées en terme de rendu entre un utilisateur simple et payant.
- **Administrateur** : Demande un niveau de compréhension minimum du site. Il faut bien connaître son intérêt, la notion d'utilisateur et éventuellement le fonctionnement de l'algorithme mais surtout ses résultats. Cet utilisateur garde un contrôle général sur l'application et sera de ce fait responsable du bon fonctionnement de celle-ci et notamment des fonctionnalités payantes. Celui-ci se verra accéder l'accès à des pages supplémentaires inaccessibles par les autres types d'utilisateurs.

3 Fonctionnalités du système

Une première version d'un diagramme des cas d'utilisation avait été effectuée avant la consultation définitive avec le client. Elle est observable ici 1 (Annexe A)

Les différentes fonctionnalités du système sont décrites pour chaque utilisateur au travers le diagramme de cas d'utilisation suivant :

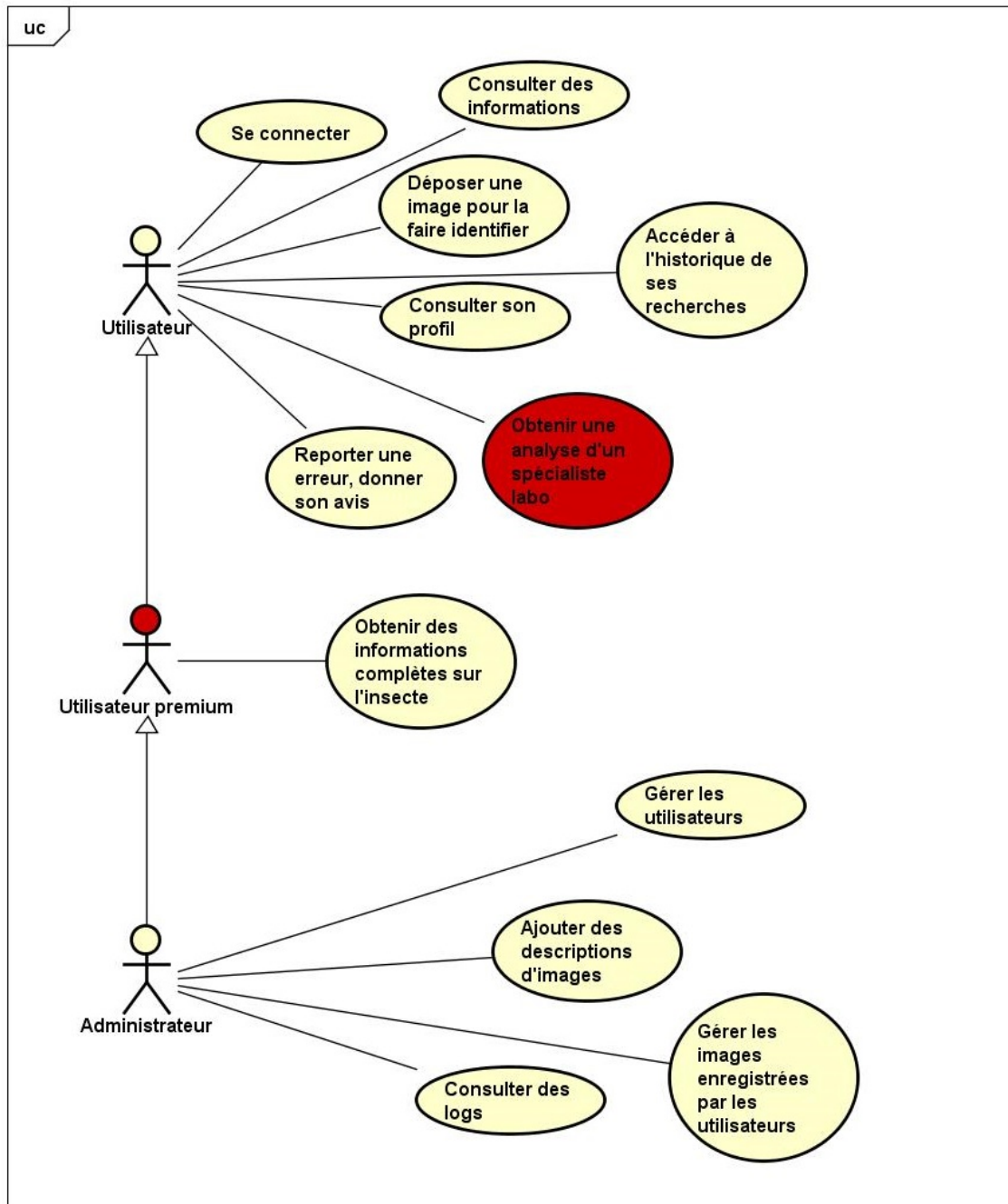


Figure 2 – Diagramme des cas d'utilisation

Il est à noter que les parties payantes ont été représentées en rouge sur ce diagramme.

On peut alors dresser une priorité sur certaines fonctionnalités (notamment utile pour une méthode agile) :

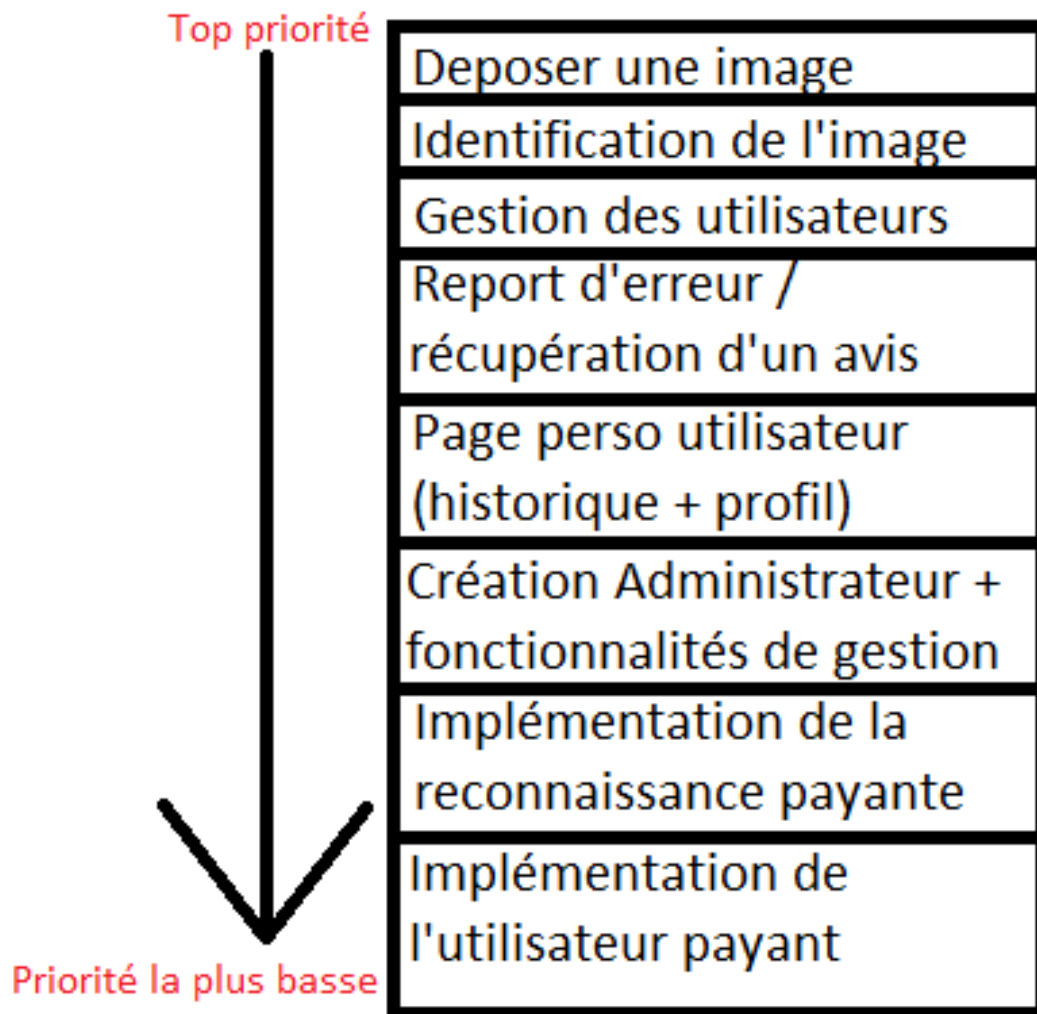


Figure 3 – Tableau des fonctionnalités par ordre de priorité

4 Structure générale du système

Même si les framework web de python encouragent et facilitent la simplicité du code, le développement doit s'articuler autour d'une structure assez forte et suffisamment évolutive.

Comme de nombreuses structures d'application web, nous verrons une organisation assez proche de celle-ci :

Comme on peut le voir sur le schéma, un modèle travaille en même temps qu'une base de données pour garder en mémoire les informations importantes de l'application. Les classes du modèle ne sont pas toutes écrites. En effet, le développement déterminera combien de classes et de fichiers il sera nécessaire de mettre en place dans le modèle.

En parallèle, les vues seront regroupées entre elles et construites en utilisant des fichiers statics tel que des fichiers CSS ou Javascript. De la même manière que le modèle, le nombre de vues ne sera déterminé que lors du développement.

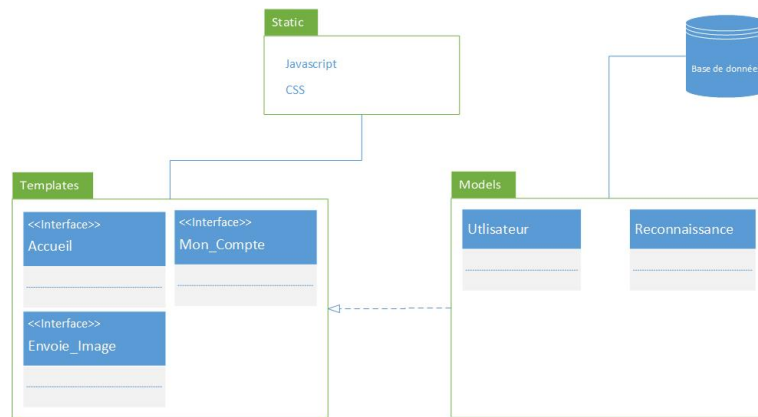


Figure 4 – *Structure du projet*

Les packages observés correspondent à des modules. De nombreux modules peuvent apparaître durant le développement. Ceux-ci ne possèdent pas obligatoirement un sens concret mais peuvent intervenir pour faciliter le découpage de fichiers trop importants.

Les fichiers de configuration du framework sont volontairement omis de ce schéma, vu qu'ils n'appartiennent pas à la structure élémentaire de notre projet.

Des mockups des pages web principales sont jointes en annexe 2 (Annexe A) et 3 (Annexe A)

3

Étude de l'existant et vieille technologique

1 Description de l'existant

Une première version du programme a été conçue avec un but similaire mais en version très simplifiée. L'attention était mise sur la capacité à envoyer une image sur le serveur et pouvoir lancer l'algorithme à partir de celle-ci. Pour ce faire, un serveur web a été développé en python avec un microframework nommé Flask. [WWW5] Un framework web est essentiel pour structurer l'application et permettre facilement l'utilisation et la programmation de certaines fonctionnalités comme la redirection par exemple. Un microframework est donc un concept similaire dont le but est de condenser ces fonctionnalités en gardant l'essentiel afin d'alléger les petits projets. De ce fait, le programme résultant est simple. Voici un diagramme résumant la structure de l'ancien système :

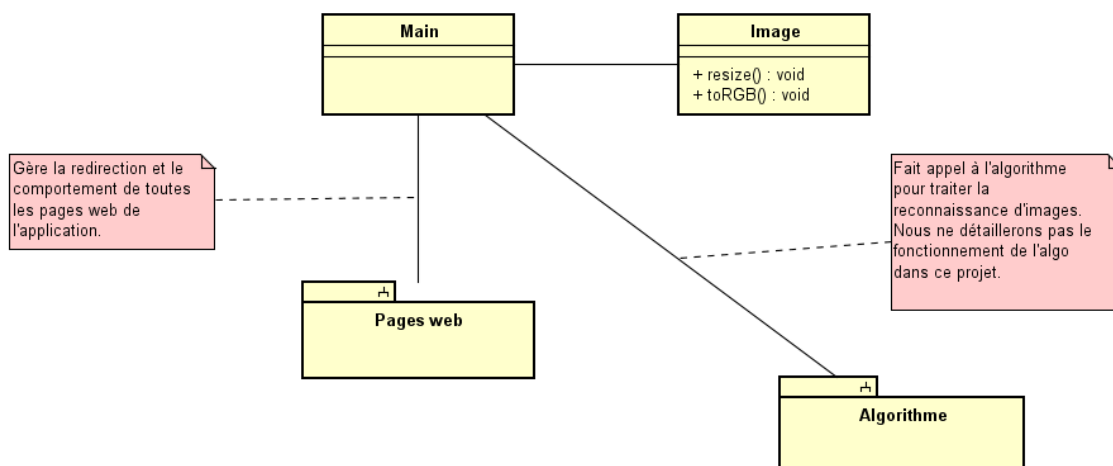


Figure 1 – Diagramme de l'application existante

Un fichier main gère complètement l'application (redirection, traitements, etc). Seul le traitement d'image est gardé dans un fichier à part. Son but est de s'assurer que l'image passée en paramètre corresponde aux restrictions de taille et de format avant d'être passé à l'algorithme. Peu de modifications sont cependant effectuées, l'algorithme n'étant pas très restrictif.

L'utilisateur navigue à travers différentes pages html que l'on peut résumer à une page d'accueil, une page pour déposer son image et une page pour visualiser le résultat. Le serveur modifie et récupère le contenu de ces pages facilement.

Il n'y a pas de gestion d'utilisateurs dans cette version, il y a donc toute une dimension manquante pour la suite de notre développement, ce qui nous pousse donc à remettre en question la réutilisation de cette version. De plus, l'utilisation d'un microframework rend restrictif l'ajout de nouvelles options et l'amélioration globale de l'application.

Voici un exemple d'une page de l'ancienne application :

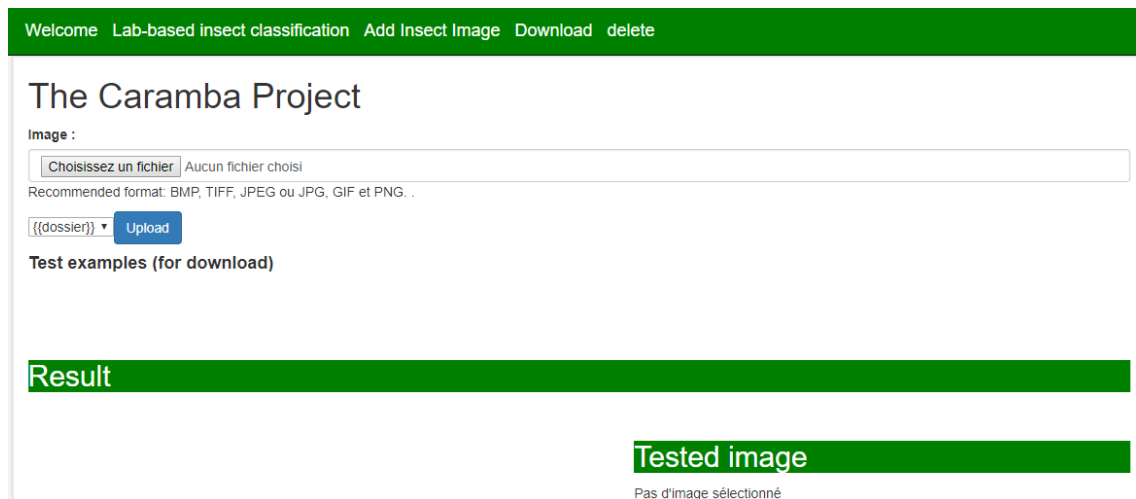


Figure 2 – Exemple de la page Ajout Image

2 Comparatif des frameworks Python

2.1 Intérêt d'un framework

Afin de créer une application, il est très important d'avoir une structure propre (une fondation) et des liens bien construits entre nos entités. Ainsi pour faciliter ces étapes et ainsi que pour automatiser de nombreux processus dans notre programme, il est recommandé d'ajouter une couche à notre programme qui s'appelle un framework. Ce dernier possède aussi très souvent des composants qui ne seront donc pas à développer et nous permettent de gagner un temps précieux dans l'avancement d'un projet.

Le choix du framework a un impact conséquent pour la suite du développement puisqu'il influence drastiquement nos choix et notre façon d'implémenter certaines parties de l'application. En effet, chaque framework est développé dans un sens et pour un but précis, même si ce but peut être large, certaines contraintes peuvent rapidement être imposées et nous sommes dépendants de ses fonctionnalités. Il est par exemple, difficile d'attacher plusieurs frameworks sur un même projet, surtout quand ceux-ci ont un but similaire mais un moyen différent d'y parvenir. Si l'utilisation d'un framework conduit à des résultats non voulus par nos spécifications, il est peut-être sage de ne pas le choisir au préalable. Cependant, il faut s'attendre qu'un framework très flexible soit possiblement un framework dur à prendre en main.

Il est donc important de prendre le temps de bien le choisir tout en connaissant les limites et les avantages que fournit le framework utilisé. Le choix de celui-ci n'est pas automatique et dépend bien entendu des spécifications de notre projet, de l'importance qu'on lui porte et de nombreux autres critères que nous allons évoquer par la suite.

2.2 Procédure

J'ai décidé de comparer 4 frameworks suffisamment différents pour couvrir les différentes variétés de frameworks existants[WWW6].

Le point de départ est de sélectionner des candidats à notre comparaison. Pour cela, je me suis basé sur plusieurs critères simples[WWW4] :

- J'ai tout d'abord pris en considération le framework utilisé de base pour l'application développée au préalable (c'est à dire Flask).
- J'ai ensuite cherché rapidement les frameworks Python recensés pour voir si certains apparaissaient de manière récurrente mais suffisamment différents pour justifier une comparaison. J'en ai ici sorti 3 autres Django(très répandu), Web2py (qui se présente comme un framework classique), et Cyclone (très complet et minutieux).

Une fois qu'on a les candidats, on fait un premier inventaire de ce qui peut être vraiment utile dans la comparaison d'un framework[WWW3] :

1. Rapidité d'apprentissage : Python étant un langage assez récent pour moi, je devrai quoi qu'il arrive apprendre le framework que je choisirai, l'important est donc de déterminer la rapidité d'apprentissage de celui-ci pour pouvoir estimer le temps qu'il me faudra pour le prendre en main.
2. Difficulté de mise en œuvre : en prenant en compte la rapidité d'apprentissage et les tâches à effectuer pour ce projet, on peut en déduire ceci.
3. Type de licence : il est toujours important de garder un oeil sur le type de licence qu'un produit peut posséder. On doit s'assurer que son utilisation n'est pas limitée ou alors d'en connaître ses limites.
4. Profondeur du code(niveau) : une des principales différences reste la profondeur du code, c'est-à-dire, à quel point le langage demande des configurations, un nombre plus ou moins important de lignes de code pour effectuer une tâche. Cela impacte directement plusieurs aspects d'un framework(sécurité, rapidité d'apprentissage, efficacité, ...)
5. Documentation : pour pouvoir apprendre un framework et l'utiliser au fur et à mesure du projet, il faut s'assurer que sa documentation sera suffisante, mais aussi qu'on puisse facilement trouver des réponses à ses problèmes. Cet aspect est souvent lié à la popularité du framework.
6. Evolutivité/Reprise : un point important dans la création d'applications, c'est de faire en sorte que d'éventuelles améliorations futures puissent être considérées. Mais aussi que ces modifications puissent être effectuées par une autre personne, de manière simple.
7. Sécurité : un aspect souvent négligé. Notre application doit être efficacement encapsulée tant pour son fonctionnement que pour sa sécurité et celle de ses utilisateurs (étant donné qu'on va devoir créer des comptes).
8. Popularité : je ne mettrai pas spécialement la popularité en avant, mais il peut être intéressant de la prendre en considération car un framework populaire l'est forcément pour une raison.

Pour certains de ces critères, j'ai placé une note sur 5 pour chaque framework afin de pouvoir les

différencier. Pour les autres critères, ils sont présentés sous forme de commentaires/annotations afin de faciliter la mise en évidence d'approches différentes de ces frameworks.

2.3 résultat

Voici le tableau obtenu, en prenant en compte que lorsqu'un chiffre est donné, cela représente une note sur 5 :

	Flask	Web2py	Django	Cyclone
Propreté du code	3	5	5	4
Profondeur du code(niveau)	3	4	4	5
Facilité de la prise en main	4	3	3	2
Restriction d'utilisation	non	non	non	non
Temps d'apprentissage	Rapide	Correct	Correct	Long (apprendre twister + cyclone)
Documentation	Correct	Correct	Forte	Correct
Facilité d'évolutivité	3 (facile pour un nouveau mais code pas évolutif)	4 (similaire a Django mais moins utilisé)	5 (très utilisé, une fois mis en place : facile)	3 (très dur pour un débutant)
Remarques :	Microframework : Compact, efficace, peu évolutif	Pas de version supportée pour python 3.X	Template & model gérés entièrement, ne passe pas par un support différent	-

Figure 3 – Tableau du comparatif des frameworks

Pour mon application j'ai choisi de mettre en place un framework Django[[WWW2](#)]. Mon choix s'est orienté vers sa popularité, son efficacité et sa bonne documentation. De nombreuses applications connues se basent sur ce framework, j'ai donc une certitude de qualité. De plus pour avoir comparé les ressources et les fonctionnalités apportées par les différents framework, j'ai pu constater que Django était souvent très supérieur aux autres, j'ai donc de grandes chances de réussir l'implémentation de tous les aspects de l'application que je souhaiterai mettre en place.

3 Django

Même si l'intérêt et les qualités/défauts de Django ont été abordés juste avant, il me semble bon de repréciser quelques points liés à son fonctionnement.

Autre sa particularité d'être un framework web Python, Django vient avec une panoplie de composants utiles pour la création de sites. Ces composants peuvent être dans la simplification du développement ou dans la sécurisation du système essentiellement.

Si on regarde en détails la structure de Django, on s'aperçoit que c'est un framework complet qui prend en charge tous les aspects d'un site web et les fait fonctionner ensemble. Il s'occupe aussi de gérer une bonne partie des fonctionnalités d'un serveur classique d'une manière assez simple qui suit le schéma suivant :

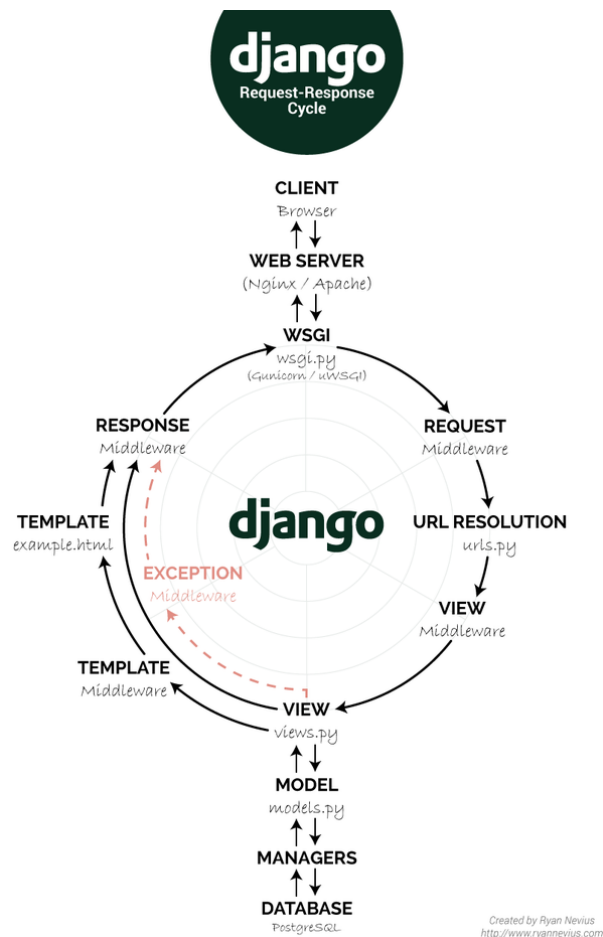


Figure 4 – Fonctionnement Django

On retrouve en partie la structure initiale du projet avec les différents éléments du schéma. On voit cependant intervenir le mot WSGI.

WSGI (Web Server Gateway Interface) est le langage de communication utilisé entre Django et le client. Il permet de faciliter la communication entre un langage python et les navigateurs web. Voici un schéma pour aider à situer l'impact de WSGI [WWW1] :

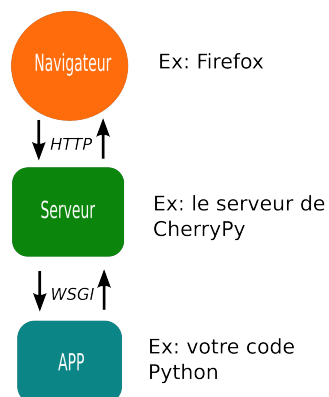


Figure 5 – Fonctionnement WSGI

Cette structure a notamment un impact lors du déploiement du serveur puisqu'il peut obliger certains réglages spécifiques et limite le nombre de serveur publique et gratuit capable d'héberger ce genre d'application.

4 Fonctionnement HTML et rendu

L'affichage d'une page HTML pour notre site passe par plusieurs étapes qu'il est essentiel de bien connaître. Les premières étapes ont pu être décrites en partie dans les paragraphes précédents dans lesquels on explique notamment le fonctionnement de Django et de la gestion des requêtes au sein d'un serveur. Une fois ces principes bien en tête, on comprend que notre serveur va retourner une page web qu'il va falloir afficher du côté client. Le design de nos pages doit pouvoir correspondre avec justesse à un panel complet d'appareils et de formatages différents. Il est pour cela important de bien comprendre le principe de rendu d'une page.

1. La requête : Il existe plusieurs requêtes possibles, mais rappelons que selon le type de la requête la réponse est complètement différente voir inexistante.
2. La réponse : Le serveur répond à la requête et très souvent sous forme d'une page web. Cette page web n'est rien d'autres qu'un message contenant notre code prêt à être lu par un navigateur. Il est donc important que tous les pseudo-codes ou autres balises liées à un langage non compris par le navigateur doivent avoir disparu au moment de l'envoi. C'est le cas ici, toutes les balises du type `{{ }}` ou `{% %}` et leur contenu doivent être remplacées auparavant.
3. La compilation : Le navigateur ayant reçu le message il peut maintenant interpréter la réponse et plus particulièrement construire la page reçue. Pour cela il effectue 3 actions successives toujours dans l'ordre précis :
 - a- Compilation du DOM(Document Object Map) : cela correspond à construire toute la structure de la page, c'est-à-dire l'html.
 - b- Compilation du CSSOM(CSS Object Map) : même principe mais en suivant la structure du CSS.
 - c- Création d'un arbre de rendu : Le but est de fusionner les résultats des compilations précédentes.
4. Le rendu : Le navigateur n'étant pas conscient de manière permanente de l'appareil qui s'en sert, il doit ensuite se charger de faire correspondre l'affichage de la page à celui souhaité pour l'appareil. Il ne rajoute pas de code, et se contente simplement de chercher la taille de l'écran et la manière dont celle-ci va affecter l'affichage final.

Même si ces étapes restent fondamentalement non modifiables, il devient de plus en plus possible via des attributs de moduler intelligemment le rendu de la compilation sur certains aspects.

Il ne reste plus qu'à afficher la page pour le navigateur.

4

Analyse et conception

1 Formulation définitive du projet

Notre projet se résume donc à créer une plateforme web en python à l'aide du framework Django afin de donner accès à des utilisateurs à l'algorithme de reconnaissance d'insectes.

Cette plateforme se distinguera majoritairement de la précédente en incluant une gestion d'utilisateurs qui suivra le diagramme de cas d'utilisation donné auparavant **Figure 2** (Chapitre 2).

En découlera la création de nombreuses vues différentes liées à l'utilisateur telles que du paramétrage de compte, des historiques d'analyse d'images ou encore des pages de paiements.

L'hébergement définitif du site et les transactions bancaires ne sont pas compris dans ce projet et représentent une étape clef dans l'avancement de celui-ci pour le client.

Le projet étant construit sur une méthode agile, il va devoir se découper en tâches qui seront réalisées au fur et à mesure des différents sprints. Ces tâches sont classées par ordre de priorité **3** (Chapitre 2), de ce fait les bases de la plateforme seront rapidement mises en place.

2 Description Algorithme

L'algorithme utilisé pour la reconnaissance d'images utilise des fichiers de modèles au nombre de trois stockés dans le serveur. Ces fichiers seront utilisés par la librairie *Tensorflow* afin de calculer le résultat final.

Il est important de bien comprendre que l'image d'entrée de l'algorithme subira quelques traitements définis dans la classe Algorithm d'application, il est donc pas nécessaire de forcer l'utilisateur à donner une image dite "parfaite" mais simplement de l'encourager. Voici un exemple d'image qui correspond au besoin de l'algorithme :

Cette image possède l'avantage d'être prise en labo, ajoutant la facilité de la couleur uniforme en arrière-plan, cependant ce paramètre n'est pas essentiel dans la reconnaissance.



Figure 1 – Exemple image d'entrée

3 Critiques et limites

3.1 Technologiques

Un des premiers points à débattre est sur le choix des langages. Python étant imposé, il n'est pas possible de modifier ce paramètre même si cette technologie reste relativement récente dans mes compétences. Cependant, c'est un langage assez simple à prendre en main et avec suffisamment de documentation à jour pour éviter un blocage significatif.

L'autre point crucial pourrait être l'utilisation de Django qui représente une partie importante de ma documentation sur les deux semestres de travail. Même si la comparaison des différents frameworks [Figure 3](#) (Chapitre 3) m'a conduit à choisir celui-ci, il est évident que le temps nécessaire à son apprentissage et ma capacité à le prendre en main de manière complète pourront impacter la production finale. Même si une transition de framework peut se considérer, je ne pense pas qu'elle sera nécessaire.

3.2 Conceptuelles

La structure exacte du site est difficile à définir, notamment à cause du point précédent. L'allure générale peut être amenée à changer même s'il y a de faibles chances que cela arrive. Ceci n'aura cependant que peu ou pas d'impact sur le résultat et les préconditions. Ce type de langage et de frameworks présente une grande adaptabilité aux changements et l'encourage même parfois lorsque le site évolue. De ce fait il restera délicat jusqu'à la fin de déterminer la structure exacte du projet mais garantira une plus grande facilité dans l'implémentation de nouvelles fonctionnalités.

De plus, face à la conception définie auparavant, les fonctionnalités sont faites pour être amenées au fur et à mesure que l'application se développe et non pas pour revenir en arrière dans le développement.

4 Déroulement et prévisions

Lors des premières prévisions, il était question de passer un certains temps sur la recherche puis d'enchaîner rapidement sur une série de sprints de développement des différentes fonctionnalités de l'application.

Le défaut principal de ce modèle restait dans la possibilité d'intégrer le client à cette conception et donc d'obtenir un retour sur le développement. De plus, le temps donné pour ces sprints seraient très probablement juste pour arriver à un développement "propre" et agrémenté de tests.

Le prévisionnel a donc été revu notamment pour intégrer le déploiement du serveur et un temps plus important sur le premier sprint :

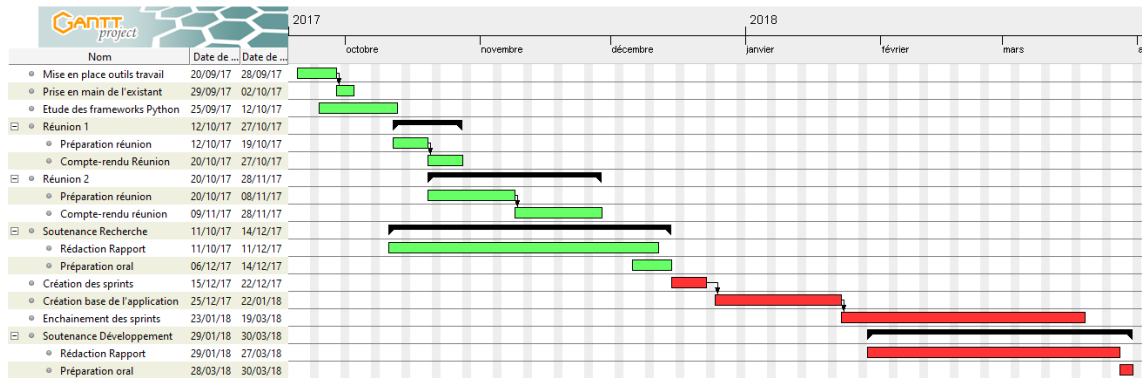


Figure 2 – Diagramme de Gantt

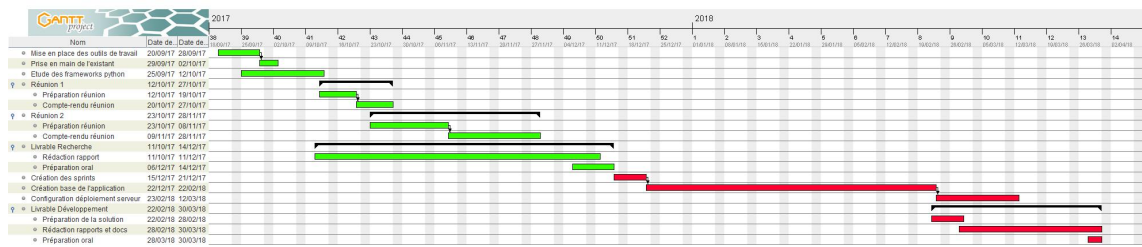


Figure 3 – Diagramme de Gantt 2

Le déploiement serveur étant une donnée assez nouvelle, il a été nécessaire de travailler en partie dessus pour pouvoir le mettre en place au sein de mon projet.

5

Mise en oeuvre

1 Environnement

Comme précisé en annexe **B** et dans les chapitres précédents, nous utilisons Python 3.6.X pour développer notre application à l'aide d'un framework web Django.

Plusieurs librairies ont été utilisées dans ce développement :

- Des librairies liées à Django afin de faciliter le développement
- Des librairies liées à l'algorithme (Tensorflow, keras, ..)
- D'autres librairies diverses nécessaires

Ces librairies sont compatibles entre-elles et s'installent facilement grâce au package *pip* de Python. Il est cependant recommandé d'utiliser un environnement virtuel de Python afin de faciliter l'installation et le lancement de l'application.

L'ensemble des documents d'installation, de prise en main et de tests sont en annexe afin d'expliquer les détails concernant l'environnement du projet.

2 Implémentation

Afin de mettre en place notre solution il a tout d'abord fallu décider quelle était la meilleure structure pour notre application. Après avoir étudié le fonctionnement de Django, il était évident que celle-ci devait posséder une structure assez conséquente pour accueillir l'ensemble des fonctionnalités voulues.

Après avoir créé un repository et un environnement virtuel, j'ai pu mettre en place une application simple de Django.

Par la suite il fallait faire un choix entre tout mettre naturellement dans l'application ou la segmenter. J'ai donc choisi de la segmenter en utilisant des "apps" qui sont simplement des modules du serveur. Ces modules sont transparents pour l'utilisateur du site et sans grande conséquence pour le développeur, cependant il peut naturellement découper le programme par fonctionnalités tout en s'assurant le bon fonctionnement du programme.

Nous sommes arrivés à la structure suivante :

Comme on peut le voir, l'application **mainapp** est l'endroit où se trouve 90% de notre code, on retrouve une explication de ce schéma dans le cahier du développeur à la fin de ce rapport.

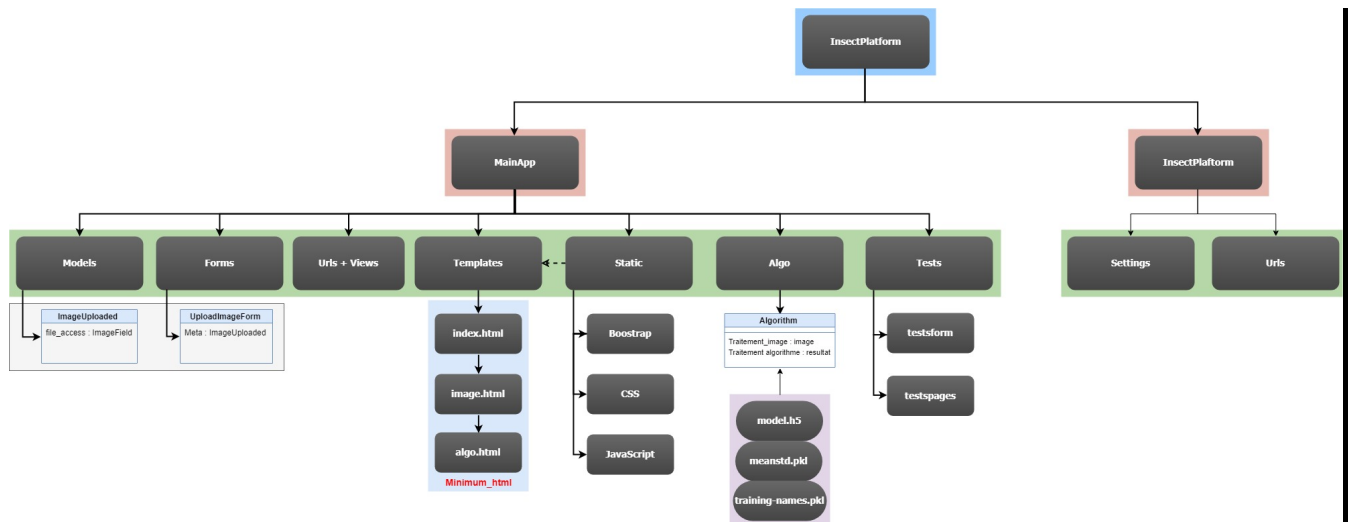


Figure 1 – Structure du projet

La structure étant mise en place, il faut jongler en permanence entre les modèles, les vues et les contrôleurs pour produire l'ensemble de l'application.

Un des processus important de création de cette application se trouve dans le choix et la mise en place de la base de données. J'ai décidé de prendre une base de données suffisamment conséquente et stable toujours dans l'optique de créer une application complète et efficace. Je me suis donc orienté vers un modèle de base de données mysql hébergeable très facilement en parallèle.

Pour le moment l'application possède une base de données peu remplie dû au développement actuel. Les principales tables sont celles générées par Django et seul notre table *imageUploaded* est présente.

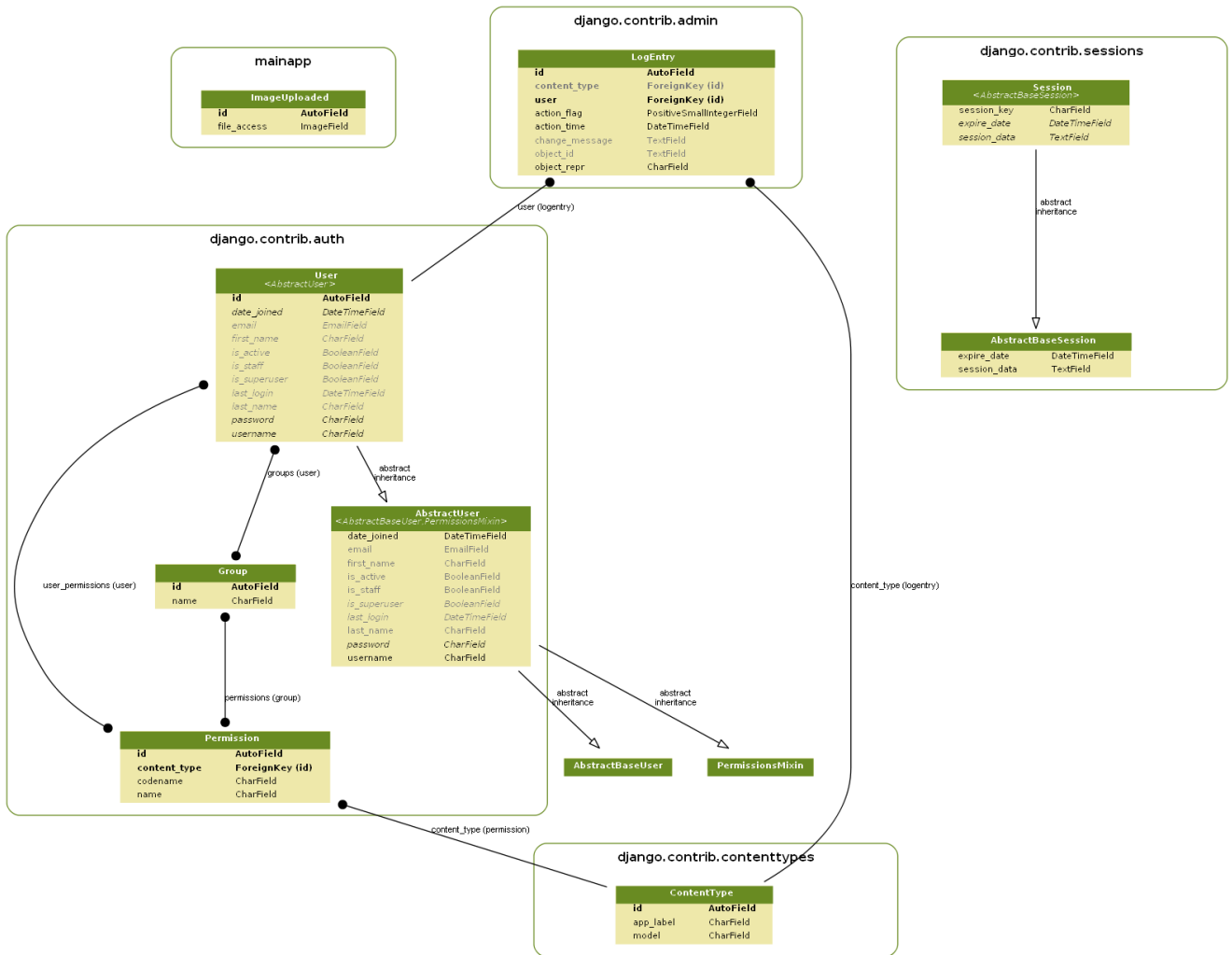


Figure 2 – Structure de la BDD

On comprend facilement l'importance de la bonne configuration du framework à sa création si on souhaite qu'il fonctionne correctement au vu de son utilisation de la base de données.

3 Création du contenu

Toujours en se basant sur le schéma, trois vues ont été développées pour correspondre au premier développement :

- *index.html* qui correspond à notre accueil
- *image.html* qui correspond à la page où l'utilisateur va mettre en ligne son image.
- *algo.html* qui permettra l'affichage du traitement de l'algorithme sur l'image passé dans la page précédente.

Les deux premières pages ne nécessitent pas de traitements particuliers, cependant de la deuxième à la troisième le serveur reçoit l'image de l'utilisateur et doit lancer la reconnaissance. Il a été choisi de conserver les images en bdd et dans les fichiers du serveur notamment pour des raisons d'évolutivité de l'application. Même si l'implémentation telle qu'elle est sera très certainement amenée à être modifiée légèrement, elle reste un avantage pour la future intégration des utilisateurs telle qu'elle a été conçue plus haut dans ce document.

Lorsque l'image est sauvegardée elle est envoyée grâce au contrôleur correspondant dans une suite de traitements définis par la classe *Algorithm* du projet. Ces traitements mènent au lancement de l'algorithme de reconnaissance d'image et à la formulation du résultat sous forme d'un tuple contenant les insectes et un pourcentage : `(insecte1, pourcentage), (insecte2, pourcentage), ...`

Il est à noter que le formulaire de récupération d'images est assez pratique pour automatiser la sauvegarde de l'image dans la base de données avec le modèle correspondant.

Ce résultat est ensuite interprété et envoyé dans la vue finale du processus.

Pour la réalisation des vues, il a été décidé d'utiliser **Bootstrap** pour faciliter la structuration des pages et les rendre plus agréables. En parallèle, nous avons utilisé une page html de base pour contenir tout le code d'en-tête et de pied de page pour éviter de répéter le code dans les différentes vues et simplifier leur contenu. Cette page de base est travaillée pour faciliter le référencement et le chargement des pages.

4 Mise en place des Tests

Les tests sont un point important de toute application. Cependant il est difficile de concevoir correctement des tests lorsqu'on utilise un framework qui possède déjà des tests intégrés. Il était donc important de faire un point sur les différentes sécurités déjà existantes afin d'éviter de re-tester ces mêmes points.

Voici quelques sécurités déjà développées :

- Test de validation de formulaires avec *form.is_valid*.
- Sécurité des formulaires avec un token CSRF automatiquement généré.
- Protection contre les injections SQL.
- Contrôle des paramètres avant déploiement du serveur, avec existence d'une commande pour faire un inventaire des paramètres défectueux.
- Et plein d'autres encore ...

Pour mettre en place les tests personnels, j'ai encore une fois suivi le principe d'une large structure en découpant des fichiers de tests selon les fonctionnalités testées. Cela permet de facilement retrouver les tests souhaités.

Les tests sont explicitement détaillés dans les annexes de ce document.

5 Déploiement serveur

Un des challenges de ce projet est de pouvoir mettre en place le serveur qui hébergera notre application. Il est un avantage de pouvoir rapidement mettre en place ne serait-ce que réussir une installation même temporaire de notre projet sur un serveur pour plusieurs raisons :

- Tests de du serveur en mode release et non en mode debug
- Possibilité de montrer le rendu au client en temps réel et lui laisser le temps de tests.
- Anticiper les problèmes de déploiement qui pourraient apparaître à la fin du projet

Même si Django fournit un environnement de test rapidement exécutable, il est cependant impossible d'effectuer de déployer complètement le serveur seulement en une commande. Il est donc nécessaire de prendre le temps de réfléchir à cette problématique.

Le déploiement du site doit se faire en plusieurs étapes :

1. Préparation du site :

Nous devons nous assurer d'effectuer de nombreuses vérifications sur notre site avant sa publication sur un serveur accessible par tous :

- Choix de la version : la version du site hébergée doit être la dernière version stable développée et non une version encore en développement. L'utilisation d'un logiciel de versioning permet justement de s'assurer que cette contrainte puisse être respectée sans soucis.
 - Sécurité : aucune donnée privée doit être accessible de l'extérieur, ainsi certaines données doivent être configurées sur le serveur et non plus codées en dure dans notre code.
 - Nettoyage du code : aucune partie de code inutile ne doit être conservée, on peut éventuellement prévoir de compacter certaines parties à l'aide de logiciels afin de créer une version seulement faites pour être hébergée sur le serveur.
 - Dernières vérifications : d'autres problématiques doivent éventuellement être pensées avant le déploiement, que ce soit la gestion d'espace serveur, revues des dépendances (test de l'existence si on change de logiciel d'exploitation, comptabilité,...), et de nombreux autres paramètres.
2. Mise en ligne du site :
- Ensuite nous devons mettre en ligne nos fichiers tout en s'assurant de garder la structure mise en place auparavant. On s'assure d'installer toutes les dépendances puis d'effectuer les configurations nécessaires sur le serveur.
- Selon le framework principal utilisé pour développer le site, plusieurs instructions peuvent être nécessaires pour lancer correctement le site. Il faut donc s'assurer des instructions présentes sur les documentations des frameworks correspondants. Par exemple pour Django, il est nécessaire d'exporter les fichiers static d'une manière spécifique pour garantir leur accès de l'extérieur.
- Il sera important aussi de créer/importer la base de données soit manuellement, soit automatiquement avec les outils mis à disposition soit par le framework soit par le serveur en lui-même.
- Nous ne considérons pas ici toutes les étapes propres au serveur qui peuvent être l'acquisition d'un nom de domaine, de la configuration des DNS aux autres réglages de la configuration de base d'un serveur.
3. Tests et analyse :
- Après avoir mis en ligne correctement votre site, il est assez normal de vérifier que tout marche normalement. Pour cela rien de plus efficace que de contrôler toutes les pages et les tester si possible ainsi que leur comportement. L'utilisation de certains logiciels de tests peut se refaire ici aussi afin de valider le résultat obtenu.
- On pourra par la suite envisager d'apporter une autre dimension à la configuration d'après-déploiement tel que la gestion du référencement et des analyses liées au site (tel que Google Analytics propose par exemple)

6

Bilan et conclusion

1 Ce qui est fait et reste à faire

Suite aux spécifications du premier semestre, une première application a donc été créée afin de mettre en place les bases de la structure finale du projet.

L'essentiel était donc de rendre accessible dans un premier temps l'algorithme et ses résultats, puis dans un second temps de valider cette base à l'aide de configuration, tests et un déploiement de l'application sur un serveur.

La base étant maintenant correctement mise en place, il est possible d'ajouter désormais les fonctionnalités initialement prévue et de les sécuriser proprement. L'application a été normalement structurée pour accueillir les prochains changements tant d'un point de vu organisation des fichiers que d'un point de vu organisation du code.

Les prochaines étapes seront d'intégrer la gestion des utilisateurs, améliorer les vues selon le retour du client, compléter les fonctionnalités liées à la reconnaissance (si nécessaire), puis de partir sur les fonctionnalités payantes.

2 Analyse qualité

Même si l'orientation du projet à facilement permis la réflexion et la mise en place des tests, il aurait été souhaitable de pouvoir en faire plus. Cependant la base de données actuelle ne permet pas beaucoup de test vu son faible remplissage, et les contraintes structurelles de Django sécurisent déjà un bon nombre d'aspects de l'application réduisant ainsi le nombre de tests à implémenter.

Dans l'ensemble je pense que l'application actuelle possède de bons atouts en terme de qualité pour continuer d'évoluer dans cette voie.

3 Analyse gestion de projet

La gestion du projet avait été orienté sur une méthode agile afin d'intégrer au fur et à mesure la solution finale tout en garantissant un minimum de sécurité et de recul sur l'avancement du projet.

La première solution a été correctement intégrée mais malheureusement la gestion s'étant réorienté, les autres sprints ont été mis en pause et au final pas effectués. Cet aspect a été plus frustrant puisque je n'ai pas pu continuer sur les fonctionnalités prévues initialement et donc présenter un site plus évolué. Cependant, j'ai obtenu la satisfaction de pouvoir le déployer sur un serveur de Polytech et de savoir que le projet va pouvoir continuer avec de bonnes bases.

Si l'organisation initiale prévoyait le déploiement serveur, je pense que j'aurais passé un peu plus de temps sur la recherche à ce sujet lors du premier semestre, afin d'être plus confiant et efficace au deuxième semestre.



Remerciements

Je tiens à remercier Maxime Martineau pour l'aide apportée durant toute la période du projet.

Je remercie aussi Damien Munier pour avoir participé aux spécifications clients.

Je remercie enfin Gilles Venturini ainsi que Romain Raveaux pour leurs différentes interventions et conseils ainsi que Barthelemy Serres pour l'aide fourni lors de la mise en place du serveur qui héberge l'application.

Annexes



Conception

Attention, certains schémas sont présentés en anglais, ce qui ne signifie pas que les pages finales de la plateforme le seront.

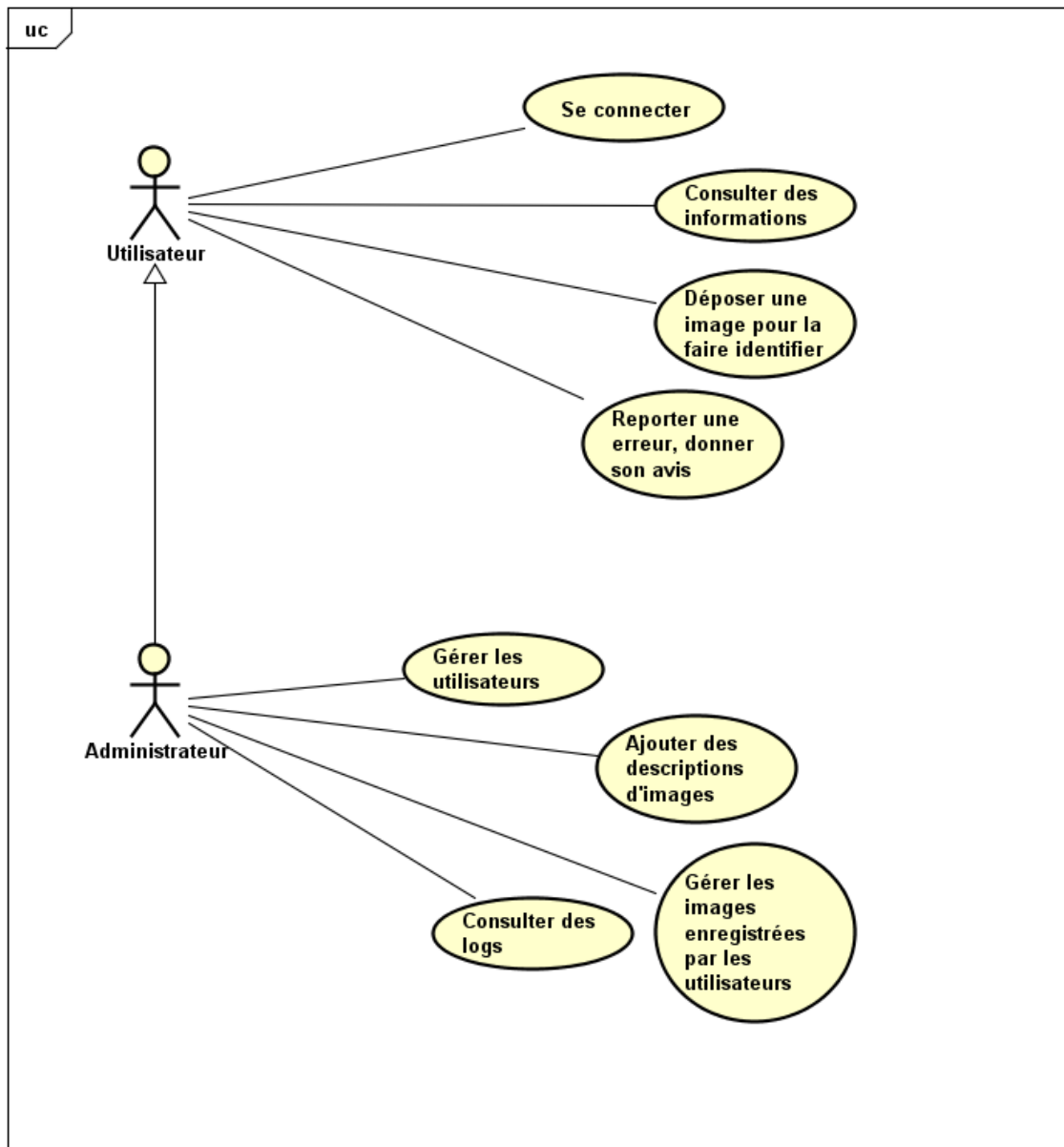


Figure 1 – Première version du diagramme des cas d'utilisation

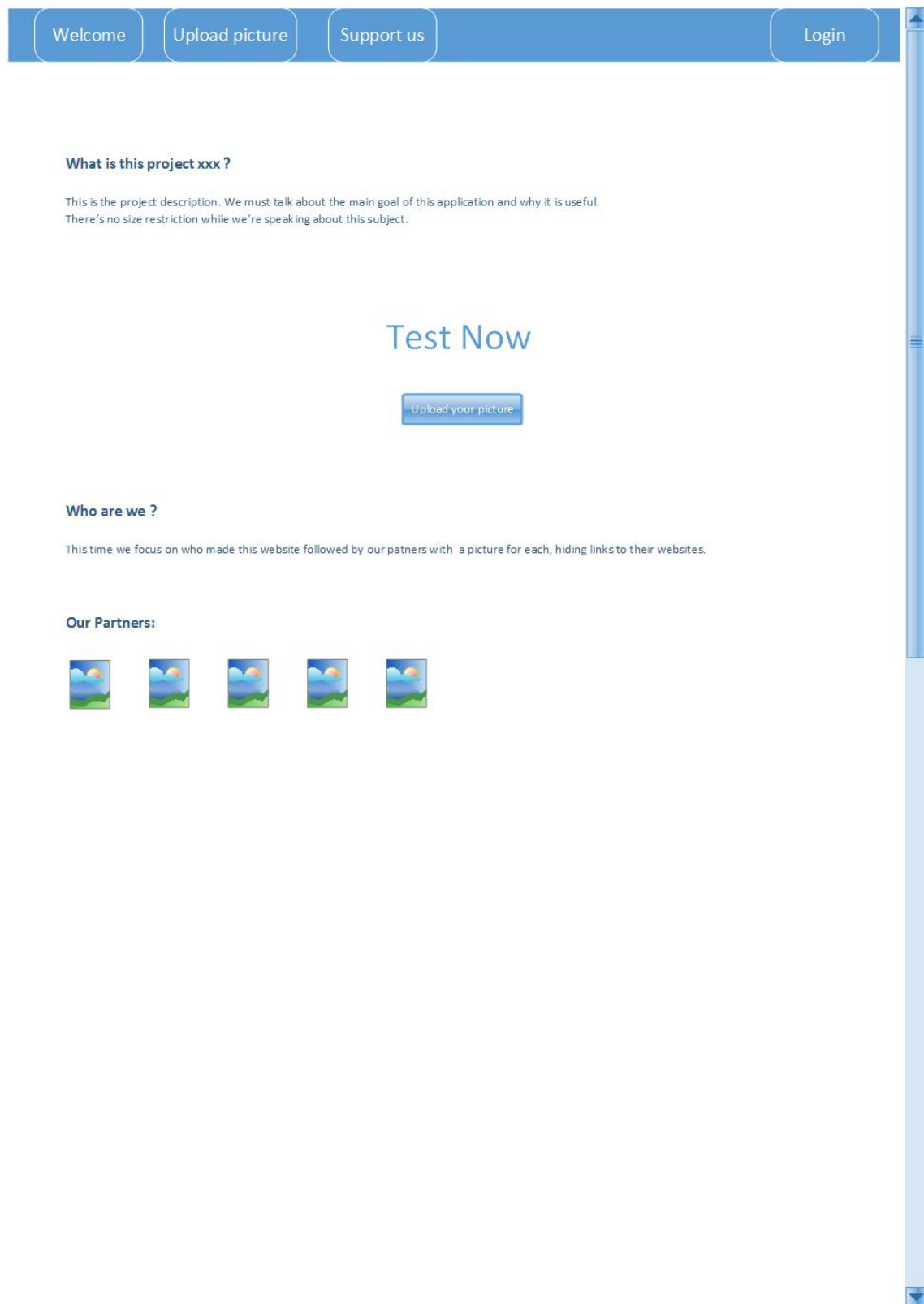


Figure 2 – Mockup de la page d'accueil




1- Upload a Picture

No Picture Selected
→ No idea how your picture should be taken ?

Accepted Files Types: BMP, TIFF, JPEG or JPG, GIF and PNG

2- See the result + feedback




How accurate is the result ?

☐ ☐ ☐

☐ Novice
 ☐ Professional

What was the problem ? (only if needed)

3- Résultat complet + incitation à le confirmer



Final Result: You took a picture of a *RESULT*

Here is a description of what is the result. This description can be set by admins in a specific part.

→ Do you want your result to be examined by a specialist ?

→ Restart with an another picture ?

Figure 3 – Mockup de la page d'ajout d'images

B

Document d'installation

1 Contraintes fonctionnelles

Afin d'installer l'application il est nécessaire d'avoir une machine qui tourne sous un système d'exploitation où python est installable (Windows, Linux, ...) Si vous souhaitez déployer sur un réseau le serveur, assurez vous aussi d'avoir accès à internet et les droits suffisants sur le réseau. Cette documentation n'explique pas la mise en place d'un serveur WSGI dans les détails.

Tout d'abord, assurez vous que python version 3.6 ou supérieur (si compatibilité) est installé. Certaines librairies comme pip et virtualenv sont nécessaires, vérifiez qu'elles sont présentes à la fin de l'installation. Le code est présent sur un dépôt gitlab dont les droits sont limités à Pierre Juillié et Maxime Martineau, pour obtenir les droits et récupérer le code merci de voir avec ces personnes.

2 Installation de l'environnement

Après avoir mis en place votre gestionnaire de repository (git, svn, ..) et téléchargé le projet **InsectPlatform**, placez vous à la racine du projet, ouvrez une fenêtre de commandes en administrateur et créez un environnement virtuel python à l'aide de la commande :

```
> python -m venv NomChoisi
```

- Le nom choisi n'a que peu d'importance.
- Il est conseillé d'exclure de votre repository ce dossier créé.
- Si plusieurs versions de python sont installées sur votre ordinateur, assurez vous d'utiliser python3 à la place de python.

Avant de continuer, on va lancer le script de l'environnement virtuel. Celui-ci appliquera toutes vos commandes sur l'environnement seulement et non pas sur l'ordinateur, il sera donc important de ne pas lancer le projet à partir d'une fenêtre de commandes sans avoir lancé le script au préalable.

```
1 > cd NomChoisi/Scripts
2 > ./activate
3 ----- ou -----
4 > activate
```

Si tout se passe bien, cette fenêtre de commande est maintenant dans le script.

3 Installation des librairies

Tout en restant dans l'environnement, replacez-vous dans le projet à l'endroit où se trouve le fichier *requirements.txt*.

Lancez maintenant cette commande :

```
> pip install -r requirements.txt
```

Toutes les librairies ainsi que leurs dépendances devraient s'installer.

Il est important que l'invité de commandes soit ouvert en administrateur.

Rappelez vous qu'en dehors de l'environnement virtuel, ces librairies n'existeront pas.

Si une librairie était déjà présente sur votre ordinateur (même en dehors de l'environnement virtuel), il se peut que le processus ne l'installe pas. Vérifiez alors que la version installée soit compatible ou mettez la à jour.

4 Configuration de la base de données

Le serveur nécessite une base de données pour tourner. Il est totalement possible de changer le type de base de données de départ. Pour cela il faudra éditer le fichier *InsectPlatform/InsectPlatform/settings.py* au niveau des databases et spécifier celle que vous souhaitez utiliser. Par défaut, la configuration choisie est une BDD de type mysql (vous pouvez utiliser WampServer sur windows) avec ces identifiants :

```
1 DATABASES = {
2     'default': {
3         'ENGINE': 'django.db.backends.mysql',
4         'NAME': 'insectdb',
5         'USER': 'root',
6         'PASSWORD': '',
7         'HOST': 'localhost',
8     }
9 }
```

Si cette configuration vous convient, créez une base de données avec pour nom "insectdb" et assurez vous que cette base de données soit hébergée sur le localhost avec un utilisateur *root*. Il est conseillé de définir un mot de passe puis d'éditer ce fichier avec le mot de passe choisi (différent de *password*).

Pour créer les tables, il est nécessaire d'effectuer une commande spécifique. Assurez vous cette fois d'être à l'intérieur de votre projet, au niveau du fichier *manage.py* :

```
> python manage.py migrate
```

5 Lancement du projet

Maintenant que tout est prêt, vous pouvez lancer le serveur Django :

```
> python manage.py runserver
```

Plusieurs points sont à noter ici :

- Par défaut, la configuration du projet le laisse en développement. Pour assurer son déploiement en mode release, il est important de suivre quelques étapes spécifiées dans le rapport du projet et trouvable dans la documentation Django.
- Il est possible de spécifier une adresse IP et un port dans la commande, cependant si vous n'en spécifiez pas, le serveur sera lancé sur : 127.0.0.1 :8000/

Pour plus d'informations, veuillez consulter la documentation Django ou le cahier du développeur de ce projet.

C

Cahier du développeur

1 Avant-propos

Il est fortement recommandé de lire au préalable la notice d'installation du projet, de prendre connaissance de la documentation Django et de se référer au rapport complet de ce projet pour plus d'informations. La partie concernant les tests sera plus détaillée dans le cahier de tests. Les commandes commençant par ">" sont des commandes Bash à effectuer dans un invité de commandes sans ce symbole.

2 Langage, framework, librairies et convention de nommage

Ce projet est développé en python 3.6 en utilisant le framework web Django. Les versions des librairies et du framework sont incluses dans le fichier *requirements.txt*. Ce fichier est auto généré par le processus pip de python, il est déconseillé de le modifier manuellement. En cas d'ajout de bibliothèques, utilisez la commande suivante après l'installation de la librairie :

```
> pip freeze > requirements.txt
```

Les processus d'installation sont détaillés dans la notice d'installation.

Concernant la convention de nommage, elle est telle qui suit :

- les classes commencent toujours par une majuscule
- les fonctions commencent toujours par une minuscule
- si le nom est composé de plusieurs mots, chaque mot prend une majuscule à la première lettre dans la mesure où la deuxième règle n'est pas perturbée.
- les noms de variables sont en minuscules et en cas de plusieurs mots dans le nom, on les sépare par un tiret.

Exceptions possibles : les noms de fichiers et des images des utilisateurs.

3 Documentation du code

Tous les fichiers, classes, fonctions et certains attributs possède une structure de commentaires unique.

Chaque fichier contenant une en-tête de type `## Explication du fichier` est un fichier commenté et modifiable. Les autres fichiers sont très souvent des fichiers auto-générés par Django et donc leur modification est rarement utile.

Chaque déclaration de classes ou de fonctions est suivie d'un bloc de code structuré de cette façon :

```

1  '''
2      Explication courte de la fonction / classe
3
4      Explication détaillée / précisions
5  # param : les paramètres d'entrée (facultatif)
6  # retourne : les paramètres de sortie (facultatif)
7  # condition : exprime une condition structurelle ayant un impact sur le comportement ←
8      (facultatif)
9  # exit : équivalent de # retourne pour les contr leurs (facultatif)
10 '''

```

4 Structure du système

Le projet est construit de telle façon que le système soit le plus détaillable possible. L'application générale s'appelle **InsectPlatform**. Pour faciliter son développement et sa sécurité, cette application est elle-même découpée en applications. Cela n'a aucun impact sur la manière dont l'application générale est compilée et fonctionne, seulement sur sa structure propre.

Pour le moment une seule sous-application est développée : **mainapp**. Elle gère essentiellement les fonctionnalités de base, c'est-à-dire les pages web Index, Image, et Algo qui sont les pages coeurs du projets.

Voici la structure détaillée de l'application :

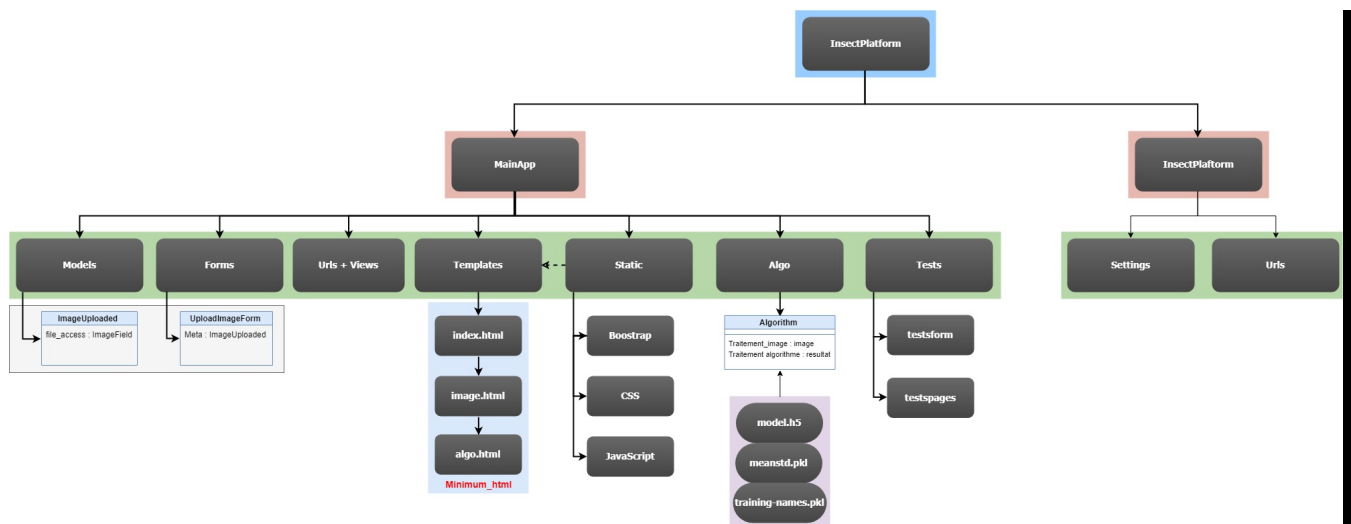


Figure 1 – Structure du projet

Comme on peut voir, les modèles sont stockés dans un fichier `models.py`, les formulaires dans un fichier `forms.py`, les vues dans un dossier **templates**, les fichiers statiques dans le dossier **static**, la gestion de l'algorithme dans le fichier `algorithm.py`, les fichiers modèles d'algorithme sont stockés dans le dossier **reconnaissance** et enfin les tests sont placés dans le dossier **tests**. Il est à noter que si le fichier des modèles et celui des formulaires deviennent trop imposants, il est possible de créer des dossiers à la place et de découper en plusieurs fichiers le contenu.

Chaque application créée par la suite peut suivre à la lettre cette structure en changeant les fonctionnalités si nécessaire. Si toutefois le contenu brut venait à se factoriser, il est possible de le faire en plaçant ce qui est en commun dans l'application principale.

Les autres fonctionnalités tels que l'ajout de la gestion des utilisateurs ou autres parties importantes doivent se faire dans une nouvelle sous-application :

```
> python manage.py startapp NomSousApp
```

Attention tout de même, il sera nécessaire de correctement éditer et configurer les fichiers de l'application principale, notamment le fichier *settings.py* pour réussir proprement la création de la sous-application (cf documentation Django).

Si toutefois vous souhaitez simplement ajouter de petites modifications à l'application, il est parfaitement possible de le faire dans l'application déjà en place dans la limite de la cohérence.

5 Autres informations importantes

Par défaut, le projet est configuré pour tourner en mode 'Test' et il est conseillé de garder cette configuration pour le développement. En cas de déploiement du serveur pour le public, il est conseillé de lire plus attentivement le rapport de ce projet, de se renseigner sur la documentation Django et des serveurs WSGI.

Remarque : Dans le fichier *views.py* vous trouverez une classe *ImageView*. Cette classe permet d'afficher le contenu d'une table dans la base de données. Ce type d'affichage est à enlever dans le déploiement du serveur par sécurité et protection des données (pensez à enlever la vue correspondante dans les templates ainsi que la redirection dans le fichier *urls.py*).

Il est possible que lors des différentes compilations vous soyez confrontés à plusieurs avertissements du type "deprecated". C'est un comportement normal. Il sera normalement prévu une mise à jour des bibliothèques liées à l'algorithme qui permettra de faire disparaître ces avertissements.

D

Cahier de tests

1 Avant Propos

Il est fortement conseillé de lire attentivement la notice de Django concernant les tests et d'avoir lu de manière générale le fonctionnement de l'application ainsi que la documentation d'installation et le cahier du développeur. Certains tests sont déjà implémentés par le framework et ne sont pas refaits explicitement dans les fichiers de tests.

2 Où trouver les tests

Tous les tests sont répartis par type de tests dans des fichiers stockés au niveau du dossier **Tests** de chaque application. Concernant notre application **mainapp**, le dossier **Tests** contient 2 fichiers :

- *testform.py* : permet de tester les formulaires de l'application **mainapp**.
- *testspages.py* : permet de tester les pages web et notamment les redirections au niveau des contrôleurs.

3 Fonctionnalités testées explicitement

testforms.py : Fonctionnement du formulaire `UploadImageForm` dans le cas d'une image classique, pour vérifier sa cohérence avec le modèle. Le test vérifie que *form.is_valid == true*.

testspages.py : Redirection correcte dans les cas :

- requête : GET, chemin : `"/`
- requête : GET, chemin : `"/image"`
- requête : POST, chemin : `"/image"`, contenu : `'test' : 'test'`

La réponse générée doit être 200.

Attention cependant, le dernier test n'est possible que dans ces conditions. D'une api extérieure à l'environnement de test, la requête est rejetée notamment pour un problème de token CSRF (comportement normal, voulu, et testable grâce à des clients comme POSTMAN)

4 Lancer les tests

Si on souhaite lancer ces tests, assurez vous d'être placé dans le répertoire de base du projet **InsectPlatform/** au niveau du fichier *manage.py*. Puis lancer cette commande : > `python manage.py test mainapp.tests`

Ceci devrait lancer tous vos tests concernant l'application **mainapp**. Si vous avez ajouté votre propre application avec la même structure, la commande restera la même en remplaçant `mainapp.tests` par `LeNomDeVotreApp.tests`.



Webographie

- [WWW1] Sam et Max. *Qu'est-ce que WSGI et à quoi ça sert?* 3 juil. 2013. URL : <http://sametmax.com/quest-ce-que-wsgi-et-a-quoi-ca-sert/>.
- [WWW2] LES TEACHERS DU NET. *Tutoriel Django*. 4 mar. 2017. URL : <https://www.youtube.com/watch?v=SjRlmyEVXq4&feature=youtu.be>.
- [WWW3] SAM. *Blog comparaison des framework python*. 22 juin 2014. URL : <http://sametmax.com/un-petit-tours-des-frameworks-web-python/>.
- [WWW4] SEBSAUVAGE. *Programmation - Critères de choix d'un langage/framework*. 1^{er} déc. 2017. URL : <http://www.commentcamarche.net/faq/3964-programmation-criteres-de-choix-d-un-langage-framework>.
- [WWW5] THENEWBOSTON. *Tutoriel Flask*. 15 déc. 2015. URL : <https://www.youtube.com/watch?v=ZVGwqnj0Kjk>.
- [WWW6] WIKIPEDIA. *Liste des frameworks python*. 27 déc. 2015. URL : https://fr.wikipedia.org/wiki/Liste_de_frameworks_Python.

Création d'une plateforme pour la reconnaissance d'insectes

Pierre JUILLIE

Encadrement : Maxime MARTINEAU

Le CETU Innophyt

Ce laboratoire de l'université François Rabelais travaille sur les insectes et particulièrement sur la lutte anti-parasitaire. Dans le cadre de leur activité, ils sont amenés à reconnaître des insectes sur des photos et souhaitent automatiser ce système.



Ce que propose Polytech

Polytech a donc proposé un algorithme pour permettre cette identification de manière automatique des insectes en utilisant un algorithme de reconnaissance d'images. Seul problème, il faut maintenant lui donner une vie dans une application.



Conséquence

Une application web va être développée afin d'héberger cet algorithme. Elle sera créée en Python avec un framework Django et permettra de rendre autonome la reconnaissance tout en apportant des fonctionnalités supplémentaires.



Création d'une plateforme pour la reconnaissance d'insectes

Pierre JUILLIE

Encadrement : Maxime MARTINEAU

Le CETU Innophyt

Ce laboratoire de l'université François Rabelais travaille sur les insectes et particulièrement sur la lutte antiparasitaire. Dans le cadre de leur activité, ils sont amenés à reconnaître des insectes sur des photos et souhaitent automatiser ce système.

Ce que propose Polytech

Polytech a donc proposé un algorithme pour permettre cette identification de manière automatique des insectes en utilisant un algorithme de reconnaissance d'images. Seul problème, il faut maintenant lui donner une vie dans une application.

Conséquence

Une application web va être développée afin d'héberger cet algorithme. Elle sera créée en Python avec un framework Django et permettra de rendre autonome la reconnaissance tout en apportant des fonctionnalités supplémentaires.



Création d'une plateforme pour la reconnaissance d'insectes

Résumé

Dans le cadre des Projets de Recherche et de développement de 5ème année, un algorithme de reconnaissance d'insectes m'a été mis à disposition dans le but de créer une application web qui propose plusieurs fonctionnalités autour de cet algorithme. Ce document résume les spécifications de cette application.

Mots-clés

Plateforme, insectes, reconnaissance, site web, Python, Django

Abstract

For our 5th year project, an algorithm to recognize insects was given to me in order to produce a website to host this algorithm and handle different requests around this subject. This document details the functional specifications.

Keywords

Website, recognition, insects, Python, Django