

Rapport de stage

Réalisation d'une application pour le domaine
médical

Thomas Grassellini

Année 2017–2018

Stage de deuxième année réalisé dans l'entreprise Aliae

Maître de stage : Philippe Jolivet

Encadrant universitaire : Rémi Badonnel

Déclaration sur l'honneur de non-plagiat

Je soussigné,

Nom, prénom : Grassellini, Thomas

Élève-ingénieur régulièrement inscrit en 2^e année à TELECOM Nancy

Numéro de carte de l'étudiant : 2014041022

Année universitaire : 2017–2018

Auteur du document, mémoire, rapport ou code informatique intitulé :

Réalisation d'une application pour le domaine médical

Par la présente, je déclare m'être informé sur les différentes formes de plagiat existantes et sur les techniques et normes de citation et référence.

Je déclare en outre que le travail rendu est un travail original, issu de ma réflexion personnelle, et qu'il a été rédigé entièrement par mes soins. J'affirme n'avoir ni contrefait, ni falsifié, ni copié tout ou partie de l'œuvre d'autrui, en particulier texte ou code informatique, dans le but de me l'accaparer.

Je certifie donc que toutes formulations, idées, recherches, raisonnements, analyses, programmes, schémas ou autres créations, figurant dans le document et empruntés à un tiers, sont clairement signalés comme tels, selon les usages en vigueur.

Je suis conscient que le fait de ne pas citer une source ou de ne pas la citer clairement et complètement est constitutif de plagiat, que le plagiat est considéré comme une faute grave au sein de l'Université, et qu'en cas de manquement aux règles en la matière, j'encourrai des poursuites non seulement devant la commission de discipline de l'établissement mais également devant les tribunaux de la République Française.

Fait à Nancy, le 12 septembre 2018

Rapport de stage

Réalisation d'une application pour le domaine
médical

Thomas Grassellini

Année 2017–2018

Stage de deuxième année réalisé dans l'entreprise Aliae

Thomas Grassellini
27 allée des chaufourniers
54600, Villers-les-Nancy
téléphone non communiqué
thomas.grassellini@telecommancy.eu

TELECOM Nancy
193 avenue Paul Muller,
CS 90172, VILLERS-LÈS-NANCY
+33 (0)3 83 68 26 00
contact@telecommancy.eu

Aliae
2 Rue Jacques Villermieux
54000, Nancy
téléphone non communiqué

Maître de stage : Philippe Jolivet
Encadrant universitaire : Rémi Badonnel

Aliaé

Remerciements

“Je souhaiterais remercier Philippe Jolivet ainsi qu’Alexandre Durand-Salmon pour m’avoir proposé ce sujet et pour m’avoir accompagné tout au long du projet. De plus je souhaiterais remercier Rémi Badonnel pour la visite qu’il nous a rendue au cours de notre stage et Christophe Boutier pour les conseils qu’il nous a fournis lors du développement de l’application. Enfin, je tiens également à remercier Guillaume Koenig et Graziella Husson pour leur coopération et la bonne ambiance qu’il y a eu tout au long de ce stage.”

– Thomas Grassellini

Avant-propos

Ce stage étant la continuité de mon PIDR et des travaux qu'a effectués Graziella Husson à ce sujet, il peut être utile de lire ces rapports en amont avant de commencer la lecture de celui-ci.

Table des matières

Remerciements	v
Avant-propos	vii
Table des matières	ix
1 Introduction	1
2 Présentation de l'entreprise	2
3 Définition d'une étude clinique	3
4 Choix de conception	4
4.1 PostgreSQL	4
4.2 Django et Python	5
4.3 L'environnement de développement : serveur Scaleway	5
5 L'étude de l'existant	6
5.1 Recherche sur OpenClinica	6
5.2 Etude du cahier des charges et du mockup	6
6 Etapes de conceptions et de préparations	8
6.1 Réalisation du schéma entité association	8
6.2 Prise en main de Django	10
6.3 Initialisation du serveur Scaleway	11
7 Mise en place de la base de données et du projet	13
7.1 Création des modèles Django et implémentation sur PostgreSQL	13
7.2 L'accès à l'interface d'administration et mise en place d'un jeu d'essai	14
7.3 Extension du modèle utilisateur	15
7.4 Organisation de notre projet sous Django	16
8 Réalisation de l'application web	17
8.1 Méthode de travail	17
8.2 Réalisation d'un site statique de démonstration	17
8.3 Création des premières pages	17
8.4 Récupération des données dans la base	17
8.5 Réalisation des pages d'ajout de données	18
8.6 Réalisation des filtres et d'une fonction de recherche	18
8.7 Pagination, ordonnancement et URL avec regex	19
8.8 Implémentation des différents rôles	19
8.9 Questionnement sur la sécurité du site	21

8.10	Les tests unitaires	22
8.11	Réalisation des fonctions d'insertions pour le chatbot	22
8.12	La dataviz	23
8.13	L'audit trail	23
8.14	Réalisation des différents manuels de documentation	24
9	Bilan du stage	25
9.1	Principales difficultés	25
9.2	Evolutions futures	25
10	Conclusion	26
Bibliographie / Webographie		27
Liste des illustrations		29
Annexes		32
A	Annexe 1 : Note de cadrage	32
B	Annexe 2 : Gantt du projet	37
C	Annexe 3 : Mockup de l'application	38
D	Annexe 4 : Schéma entité associations	39
E	Annexe 5.1 : Table de permission des rôles	40
F	Annexe 5.2 : Table de permission des rôles	41
Résumé		43
Abstract		43

1 Introduction

L'entreprise Aliae est une jeune poussée qui propose des solutions digitales permettant un meilleur suivi des patients inclus dans des études cliniques.

Tout au long de l'étude clinique, le patient va exprimer ce qu'il ressent avec un Patient Reporting Outcome (PRO). Ce procédé va permettre de savoir si le patient est fatigué, ou s'il ressent de la douleur par exemple.

À l'heure actuelle, cette collecte d'information est effectuée en utilisant des Electronique Data Capture (EDC) et font des Electronique Case Report Form (ECRF). Pour chaque étude, le patient doit remplir régulièrement des formulaires papiers qu'il remet au centre investigator lors de chaque visite (espacé de plusieurs semaines ou mois).

Ce procédé (questionnaire papier) est très répétitif et le patient peut être amené à repousser le remplissage du questionnaire au dernier moment de sa visite sur site. Dans ces conditions, le patient peut avoir oublié de mentionner des symptômes ressentis plusieurs semaines auparavant, ce qui peut fausser le résultat et l'efficacité de l'étude.

L'application développée a pour intérêt de faire remonter ces informations le plus régulièrement possible, et en temps réel.

Les données que l'application devra gérer étant liées au domaine médical, le système devra répondre aux critères propres au milieu comme une sauvegarde de mot de passe antérieur, la sauvegarde des données pendant 20 ans ce qui amènera plus tard, à des choix de conception particuliers. De plus l'intégralité des modifications faites sur les données devront être inscrites dans un fichier pour savoir qui fait quoi, sur quoi et pour quelle raison.

Le but de ce stage dans la continuité du PIDR sera de réaliser une application pour permettre au patient de discuter avec un chatbot, de collecter ses informations et de permettre à des utilisateurs de créer des études cliniques et de les suivre.

Dans un premier temps, ce rapport présentera l'entreprise d'accueil puis parlera des choix de conception pour ce stage. Ensuite, il parlera de l'étude de l'existant et des étapes de conceptions et de préparations du projet. Enfin, il traitera la mise en place de la base de données et la réalisation complète de l'application web.

2 Présentation de l'entreprise

Aliae est une start-up qui a été créée le 16 novembre 2016 à Paris. Il s'agit d'une Société par action simplifiée (SAS) à capital variable. L'entreprise a été fondée par JOLIVET Philippe et DURAND-SALMON Alexandre. Elle ne compte pas de salariés. L'organigramme de l'entreprise est présenté figure 2.1. Aliae est actuellement implantée dans une pépinière d'entreprises appelée le MédiaParc situé dans le quartier des entrepreneurs et donc les locaux appartiennent à la métropole du Grand Nancy.

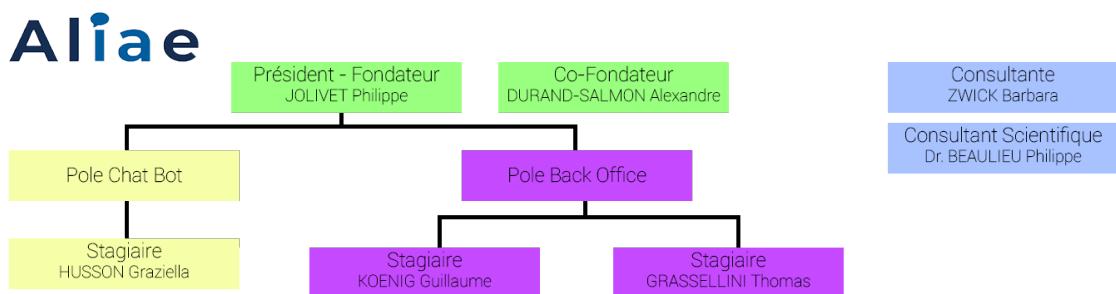


FIGURE 2.1 – Organigramme ALIAE

À l'heure actuelle, la start-up réalise une preuve de concept via un prototype dans le but de rechercher des financements et de développer son projet. Initialement, Aliae souhaitait développer un agent conversationnel appliqué à l'étude du sommeil. Après des recherches, le cadre du projet est élargi au domaine des études cliniques. En effet, avant de fonder Aliae, M. JOLIVET a travaillé au sein d'une grande entreprise pharmaceutique et connaît bien ce domaine ce qui lui permet d'en connaître les problématiques et les contraintes. Le projet de la start-up est de réaliser un outil permettant de récupérer des informations via un agent conversationnel auprès des patients dans une étude clinique pour augmenter le taux de réponse et améliorer les résultats.

3 Définition d'une étude clinique

Les études cliniques sont les phases précédant la commercialisation d'un nouveau médicament. Elles sont initiées et développées par les laboratoires pharmaceutiques qui identifient et mandatent des sites investigateurs (comme des hôpitaux) pour recruter des patients et les suivre tout au long des différentes phases de test.

Le laboratoire peut faire appel à des Contact Research Organization (CRO) ou organisation de recherche clinique par contrat pour assurer l'exécution de l'étude clinique pour son compte.

Pour débuter une étude clinique, le laboratoire doit réaliser un protocole clinique, qui correspond à un cahier des charges avec les mesures et résultats attendus (end point principal et des end point secondaires). Cela peut par exemple être d'augmenter la durée de survie. De plus, le laboratoire doit spécifier comment les patients vont être traités, comment seront recrutés en fonction de critères tels que le sexe, l'âge, etc. On définit ensuite la façon de récupérer les données pour démontrer le end point (par exemple image médical avec des scanners tout au long de l'étude, ou marqueurs sanguins).

Ces études cliniques se décomposent en quatre phases pour une durée totale de 10 à 12 ans sur des êtres humains après un pré étude de trois ou quatre ans sur les animaux. Pour les êtres humains, les phases se déroulent comme suit :

- *Phase 1 :*

Cette phase consiste à tester la molécule ou le produit médicamenteux sur un ensemble d'une trentaine de patients sains pendant quelques mois pour regarder qu'il n'y ait pas d'effet indésirable comme l'apparition d'autres maladies ou la réduction de la qualité de vie. Pour faire ces tests, un seul site est utilisé. Cette phase a pour but de valider la safety de la molécule.

- *Phase 2 :*

Pour cette phase, on regarde encore la safety, mais surtout l'efficacité de la molécule. Cette phase se fait sur un groupe de 200 à 300 patients et dure un an et demi. On passe désormais en multicentrique (plusieurs sites) pour arriver à une dizaine de centres recrutant ses patients pour l'étude clinique.

- *Phase 3 :*

Cette étape est globalement similaire à la phase précédente sauf que l'on augmente la durée de suivi des patients pour les suivre pendant trois à quatre ans. On passe également sur un nombre de patients entre 4 000 et 20 000.

- *Phase 4 :*

Enfin, cette dernière est dite phase post marketing. Une fois la molécule mise sur le marché, on continue de suivre les patients suivant le traitement pour être bien sûr la molécule ne provoque pas d'effet indésirable sur le très long terme.

4 Choix de conception

4.1 PostgreSQL



FIGURE 4.1 – Logo de Postgresql

Pour réaliser notre base de données nous avons dû faire un choix entre plusieurs Système de Gestion de Base de Données (SGBD). Ce choix a été motivé suite à des recherches sur les différents SGBD et langages associés en comparant les performances de ceux-ci ainsi que leur coût pour l'entreprise. Connaissant déjà le langage Structured Query Language (SQL), vu en Diplômes Universitaires de Technologie (DUT) mais également à Telecom Nancy, le choix s'est principalement fait entre PostgreSQL et MySQL car Oracle par opposition aux deux premiers est payant et utilise un langage différent pour les requêtes.

PostgreSQL est plus efficace dans la gestion des grosses bases de données de plusieurs centaines de giga contenant de nombreuses tables. Il propose également des fonctions pour répliquer les bases des données ou répartir la charge de travail entre les différents serveurs si besoin.

Durant les premières années de l'application, on peut penser qu'un seul serveur suffira, mais au fur et à mesure du temps, la base de données grossira et ses fonctionnalités pourront être utilisées.

Contrairement au premier, MySQL est plus adapté aux petites bases et propose moins de fonctionnalités. MySQL est également plus simple d'utilisation pour des utilisateurs non-informaticiens ou débutants.

Les bases de données étant amenées à stocker des données médicales, elles gagneront en taille au fur et à mesure du temps et devront pouvoir gérer des volumes de données toujours plus importants. C'est pour ces raisons que nous avons décidé de porter notre choix sur PostgreSQL.

4.2 Django et Python



FIGURE 4.2 – Logo de Django avec celui de Python

En ce qui concerne le langage de programmation, le choix s'est porté sur python. Ce choix semblait s'imposer pour ne pas avoir à mettre en relation du code écrit dans plusieurs langages. En effet, l'application devra être reliée à un chatbot utilisant la Rasa stack, une Application Programming Interface (API) utilisant Rasa Nlu et Rasa Core toutes deux écrites en python pour ses avantages dans le domaine de l'intelligence artificielle. Enfin, python est également le langage qui permet d'utiliser un framework très connu pour le développement d'applications web : Django

Django est un framework python pour le web. Il a été créé et utilisé par des groupes comme Instagram, Pinterest ou encore la Nasa. Ce framework est utile pour gagner du temps avec une gestion en modèle Modèle Vu Template (MVT) permettant l'utilisation de templates.

Hypertext Markup Language L'utilisation de ce framework par rapport à un trio Hypertext Markup Language (HTML)/Cascading Style Sheets (CSS)/Hypertext Preprocessor (PHP) classique ou à Javascript sera un énorme gain de temps et une architecture de projet plus propre et beaucoup plus portable. Ce gain de temps s'explique par la complexité des langages comme PHP ou Javascript par rapport à python.

De plus un gros avantage est que Django et Python sont assez simples d'utilisation et bien documentés.

4.3 L'environnement de développement : serveur Scaleway

Pour pouvoir réaliser nos tests, l'entreprise a mis à notre disposition un serveur Scaleway pour pouvoir lancer le chatbot et discuter avec lui et pour nous permettre de créer nos bases de données et notre application web.

5 L'étude de l'existant

5.1 Recherche sur OpenClinica



FIGURE 5.1 – Logo de Openclinica

Pour réaliser cette application d'études cliniques, l'entreprise ayant identifié un logiciel open source (Open Clinica) et l'avait installé sur ses serveurs. Nous avons été chargés de faire son étude pour évaluer la faisabilité de s'appuyer sur ce logiciel, et son architecture de base de données (contenant 110 tables) pour l'adapter à notre besoin. Cette étude nous a rapidement permis de mettre en évidence les éléments suivants :

- L'application étant écrite en Java, il aurait fallu faire un lien entre ce code Java et le code Python utilisé pour réaliser le chatbot.
- L'application est basée sur une licence Lesser General Public License (LGPL), variante de la licence General Public License (GPL). La licence GPL est une licence virale disant que toutes modifications d'un code GPL en quelque quantité que ce soit rend automatiquement l'application entière GPL. La licence LGPL est quant à elle moins virale que la précédente, elle permet, si on rajoute du code par-dessus du code LGPL de rendre propriétaire cette petite partie du code. Autrement dit, la licence LGPL ne contamine pas le code rajouté.

De plus, ces licences demandent de redonner gratuitement toutes modifications qui auront été faites sur du code LGPL pour quiconque le demanderait. Ces aspects liés à la licence ont été déterminants dans la réflexion de ne pas utiliser le logiciel pour notre application. l'entreprise Aliae est une entreprise commerciale qui ne souhaite pas adopter une position risquée sur le plan juridique tout particulièrement.

En outre, il reste encore la possibilité de n'utiliser que la partie base de données d'OpenClinica et de rajouter notre code sans modifier la base pour ne pas avoir à rendre notre code GPL. Toutefois, le modèle possède 110 tables et contient de nombreuses choses qui ne sont pas utiles pour l'application que l'on souhaite développer et ne possède aucune table pour gérer notre future partie avec le chatbot.

Nous avons toutefois regardé leurs modèles de données afin d'avoir une base de comparaison avec les spécifications fonctionnelles détaillées fournis par Aliae.

5.2 Etude du cahier des charges et du mockup

En complément de ce que l'on peut trouver chez les concurrents, Aliae nous a fourni un cahier des charges et un mockup (une maquette montrant l'agencement des différents boutons, champs

et liens) de l'application web que l'entreprise souhaitait réaliser. Un extrait de ce mockup est disponible en annexe C

Le cahier des charges présente les spécifications que l'entreprise souhaite avoir dans son application et le mockup nous montre comment l'entreprise veut que cela soit fait pour nous aider à réaliser l'application.

Ces spécifications et le mockup associé ont constitué une base documentaire pour mieux comprendre le besoin utilisateur et échanger avec les membres de l'entreprise. Ces discussions nous ont permis de préciser certains points et d'en modifier d'autres en accord avec l'entreprise.

À l'aide de toutes ces informations, une note de cadrage a pu voir le jour pour résumer le but du projet, ses acteurs, ses objectifs et son organisation. Cette note de cadrage est disponible en annexe A.

6 Etapes de conceptions et de préparations

6.1 Réalisation du schéma entité association

Une fois le besoin de l'entreprise cerné, nous nous sommes lancés dans la réalisation d'un schéma entité association pour pouvoir mettre en forme toutes les tables que nous allions devoir créer plus tard dans notre base de données. Pour commencer, nous avons créé les entités principales pour la gestion de l'application à savoir la gestion des projets (ici une étude clinique), des sites, des patients et des entités. Au fur et à mesure des jours nous avons étoffé ce schéma avec d'autres entités contenant des statuts pour un projet (en cours, terminé ...), des types d'entité ou d'utilisateurs, etc.

Globalement ce schéma est assez simple à comprendre mais nécessite quelques explications à certains endroits particuliers :

- *L'association OWN pour les mots de passe (pas présent sur le schéma ci-dessous) :*
Cette association entre un utilisateur et un patient vers leurs tables de mots de passe respectifs se fait avec des cardinalités 1-1 et 0-N. Le mot de passe est stocké dans une table de mot de passe et pas directement dans un champ de la table patient ou de la table utilisateur pour permettre le respect de 2 contraintes. La première est le fait qu'un utilisateur ne peut pas mettre un mot de passe identique au précédent. La seconde est le fait de devoir stocker les 5 derniers mots de passe d'un utilisateur ou d'un patient. Cette fonctionnalité n'est pas encore implémentée à ce jour mais reste une fonctionnalité future importante d'où son explication ici.
- *Association is_patient_in et l'association posses :*
L'association is_patient_in nous permet d'affecter des patients dans un groupe donné pour un projet donné et l'association posses permet d'affecter un groupe à un projet. Dans un premier temps on voit qu'il y a une redondance au niveau de la donnée sur le numéro de groupe, d'une part l'id du projet sera lié à un groupe et d'autre part il sera remis lors de l'affectation de patient.

Cette redondance est en contradiction avec la forme normale de Boyce Codd, car on peut retrouver le projet du groupe en passant par deux associations distinctes. Toutefois, cette redondance sera essentielle plus tard dans l'application pour faire comme suit. Dans un premier temps une entité crée un projet, des groupes et spécifie quel groupe est associé à quel groupe. Dans un second temps, on crée les patients et on les affecte à un groupe pour un projet donné. C'est ici que cette redondance prend tout son sens, cela permettra dans l'application de faire un premier filtre sur les groupes possibles du projet quand on affectera un patient à ce projet. Si l'on ne fait pas cette redondance, on va pouvoir, lors de l'affectation d'un patient à un projet, choisir un groupe qui n'appartient pas au dit projet.

- *Association command et l'association is_organisation_in :*
Cette fois-ci, on peut encore avoir une redondance de données si l'entité qui commande l'étude est aussi une des organisations qui participent au projet. Ces deux associations permettent à un laboratoire de commander une étude clinique et de la faire réaliser par

un CRO. Pour des raisons de simplicité en base et pour avoir accès plus facilement aux données, une organisation qui commande un projet y participe automatiquement même si elle peut ne faire rien concrètement.

Le Modèle Logique de Données (MLD) ci-dessous est une version minimaliste qui ne contient pas la totalité des tables de la base de données finales mais qui permet d'avoir une bonne idée de comment est structurée la plus grosse partie de l'application. Le schéma entité association partiel est disponible sous un format plus visuel en annexe D.

User_type	(<u>user_type_id</u> , <u>user_type_lab</u> , <u>user_type_desc</u>)
User_account	(<u>user_id</u> , <u>user_login</u> , <u>user_first_name</u> , <u>user_last_name</u> , <u>user_email</u> , <u>user_date_created</u> , <u>user_date_updated</u> , <u>user_date_lastvisit</u> , <u>user_phone</u> , # <u>user_type_id</u> , # <u>organization_id</u> , # <u>user_status_id</u>)
Project_type	(<u>projet_type_id</u> , <u>projet_type_lab</u> , <u>project_type_desc</u>)
Organization_status	(<u>entity_status_id</u> , <u>entity_status_lab</u> , <u>entity_status_desc</u>)
User_status	(<u>user_status_id</u> , <u>user_status_lab</u> , <u>user_status_desc</u>)
Organization	(<u>organization_id</u> , <u>organization_name</u> , <u>organization_city</u> , <u>organization_zipcode</u> , <u>organization_country</u> , <u>organization_date_created</u> , <u>organization_date_updated</u> , <u>organization_comment</u> , <u>organization_main_clinical_name</u> , <u>organization_main_clinical_email</u> , <u>organization_main_clinical_phone</u> , <u>organization_main_finance_name</u> , <u>organization_main_finance_email</u> , <u>organization_main_finance_phone</u> , # <u>entity_status_id</u> , # <u>organization_type_id</u>)
User_role	(<u>user_role_id</u> , <u>user_role_lab</u> , <u>user_role_desc</u> , # <u>user_id</u> , # <u>project_id</u> , <u>is_staff_in_date_participe</u> , <u>is_staff_in_date_update</u>)
Project_status	(<u>project_status_id</u> , <u>project_status_lab</u> , <u>project_status_desc</u>)
Is_organization_in	(# <u>organization_id</u> , # <u>project_id</u>)
Organization_type	(<u>organization_type_id</u> , <u>organization_type_lab</u> , <u>organization_type_desc</u>)
Patient_status	(<u>patient_status_id</u> , <u>patient_status_lab</u> , <u>patient_status_desc</u>)
Patient	(<u>patient_id</u> , <u>patient_date_of_birth</u> , <u>patient_gender</u> , <u>patient_date_created</u> , <u>patient_date_updated</u> , <u>patient_date_last_visit</u> , <u>patient_comment</u> , # <u>patient_status_id</u> , # <u>site_id</u> , <u>follow_date_follow</u>)
Site	(<u>site_id</u> , <u>site_name</u> , <u>site_address</u> , <u>site_city</u> , <u>site_zip_code</u> , <u>site_country</u> , <u>site_phone</u> , <u>site_comment</u> , <u>site_date_created</u> , <u>site_date_updated</u> , # <u>organization_id</u>)
Is_site_in	(# <u>site_id</u> , # <u>project_id</u>)
Project	(<u>project_id</u> , <u>project_name</u> , <u>project_unique_identifier</u> , <u>project_summary</u> , <u>project_date_created</u> , <u>project_date_updated</u> , <u>project_date_start_expected</u> , <u>project_date_end_expected</u> , <u>project_protocol_type</u> , <u>project_protocol_desc</u> , <u>project_protocol_date_verif</u> , <u>project_gender</u> , <u>project_expected_total_enrolment</u> , <u>project_phase</u> , <u>project_age_min</u> , <u>project_age_max</u> , <u>project_healthy_volunteer_accepted</u> , <u>project_acronym</u> , <u>project_therapeutic_area</u> , <u>project_outcome_measure</u> , <u>project_authority_id</u> , <u>project_date_start_real</u> , <u>project_date_end_real</u> , <u>project_comment</u> , # <u>projet_type_id</u> , # <u>organization_id</u> , # <u>project_status_id</u>)
Group	(<u>group_id</u> , <u>group_lab</u> , <u>group_desc</u> , # <u>patient_id</u> , # <u>project_id</u> , <u>is_patient_in_date_enrollement</u> , <u>is_patient_in_date_update</u> , <u>is_patient_in_unique_identifier</u> , # <u>project_id</u> .1)

6.2 Prise en main de Django

En parallèle de la création du schéma entité association je me suis renseigné sur Django pour voir ce qu'il était possible de faire avec. Je me suis aidé du tutoriel disponible sur le site Open Classroom qui couvre tout ce dont nous aurons besoin pour réaliser l'application [1].

Ce tutoriel nous fait réaliser des vues à l'aide de templates, de créer des modèles qui seront directement convertis en requêtes SQL pour être implémentés directement dans PostgreSQL. Cette gestion des modèles est un exemple de l'avantage d'utiliser ce framework. Pour réaliser cette application, il n'y aura aucune requête SQL à écrire nous-mêmes, l'accès aux données se fera facilement via l'accès aux modèles.

Django fournit également 2 outils pour la gestion complète des bases de données.

Le premier outil est une interface complète pour la gestion des bases de données créées avec postgreSQL. Une fois un super utilisateur créé on peut accéder à une interface montrant toutes les bases de données créées.

On peut à partir de cette fenêtre se rendre dans les différentes tables, pour y voir les différents champs et ajouter, modifier ou supprimer de nouvelles données. À partir du fichier admin.py de Django, il est possible de personnaliser cette interface administrateur. Il est possible de définir quel champ seront affichés dans le tableau montrant le contenu de la base de données. On peut également choisir les champs sur lesquels il sera possible de filtrer les données du tableau et de choisir ceux sur lesquels il sera possible de faire une recherche.

Le deuxième outil est la possibilité d'utiliser une interface shell pour créer des objets du même type que ceux crées dans les modèles et pouvoir spécifier chaque champ avant d'ajouter les données à la base, ce que Django fait seul en créant la requête SQL associé automatiquement. On privilégiera le premier outil pour le confort apporté.

Une fois encore cette interface complète est un gain de temps considérable car elle nous évitera de recoder une interface super admin pour l'entreprise.

6.3 Initialisation du serveur Scaleway



FIGURE 6.1 – Logo de Scaleway

L'entreprise ayant pris un serveur Virtual Private Server (VPS) chez Scaleway, il a fallu choisir un système d'exploitation pour le faire fonctionner et d'y installer tous les modules que nous voulions faire tourner dessus. Ce serveur aura pour vocation de permettre aux différents membres de l'entreprise d'effectuer des tests sur la version en cours de développement qui sera régulièrement mise à jour mais ne constitue en aucun cas un serveur de production.

En effet, le gros du développement se fera en local sur nos ordinateurs personnels et une fois les différentes fonctions réalisées un push sera fait sur le git du projet et le serveur pourra récupérer ce contenu sur le git pour se mettre à jour.

La première chose qui a été installée sur ce serveur est une version d'OpenClinica pour pouvoir faire différents tests sur ce logiciel et pour pouvoir bien comprendre ce que ce logiciel permet de

faire pour pouvoir produire un logiciel reprenant les possibilités offertes par celui-ci mais en rajoutant des nouvelles.

Dans un second temps, nous avons installé Django avec d'autres librairies complémentaires pour pouvoir utiliser le code que nous allons réaliser. Il a également fallu installer Rasa pour permettre la discussion avec les différentes versions du Chatbot.

7 Mise en place de la base de données et du projet

7.1 Crédation des modèles Django et implémentation sur PostgreSQL

Une fois le modèle validé, nous avons décidé de créer les modèles sous Django pour nous permettre de générer la base de données sous postgresql. Les modèles sont une fonctionnalité de Django qui permet de gérer une base de données sans avoir à réaliser ses propres requêtes SQL. De ce fait, Django va automatiquement gérer la création des différentes requêtes comme les CREATE TABLE ou la gestion des clés primaires et étrangères.

```
----- Begin ORGANIZATION models -----#
class ORGANIZATION_TYPE(models.Model):
    """
    ORGANIZATION_TYPE : id, lab, descriptption
    Contains existing type of organization.
    Existing type :
    - pharmaceutical laboratory
    - contract research organisation
    - hospital
    - clinic
    - safety authority
    """
    # ID
    organization_type_id = models.AutoField(primary_key = True)
    # Label
    organization_type_lab = models.CharField(max_length = 30)
    # Description
    organization_type_desc = models.TextField(max_length = 256, blank = True)

    def __str__(self):
        return self.organization_type_lab

    class Meta:
        verbose_name = 'ORGANIZATION TYPE'
        verbose_name_plural = 'ORGANIZATION TYPE'|
```

FIGURE 7.1 – Exemple d'un modèle Django

On peut voir dans l'exemple ci-dessus class organization_type qui deviendra une table du même nom. Cette table permettra de lister les différents types qu'une organisation peut avoir comme un laboratoire ou une autorité de contrôle. On peut voir que cette table il n'y a que 3 champs. Un id qui va nous servir de clé primaire, un label qui sera l'équivalent du nom que l'on donnera au type d'une organisation et une description sommaire de ce type.

On trouve ensuite un fonction str qui permet de retourner un certain champ de la table quand on doit faire référence à celle-ci. Par exemple ici si l'on doit afficher le type d'une organisation

ce sera son label qui sera affiché et non pas son ID. Enfin la class Méta permet, elle de donner un nom de table au singulier et au pluriel pour l'interface d'administration de Django.

Pour notre base de données nous avons réalisé de nombreuses classes python comme celle-ci pour arriver à la fin à une trentaine de tables avec parmi elles, les tables "de base" pour la gestion d'un projet, d'un site ou encore d'un utilisateur ainsi que celles pour la gestion de l'audit et celles qui permettront une utilisation correcte du chatbot.

Suite à cela une simple migration va permettre de créer toutes les bases sur postgre à partir de ce fichier et rendre le tout accessible à partir d'une interface d'administration.

7.2 L'accès à l'interface d'administration et mise en place d'un jeu d'essai

Django permet d'accéder à une interface qui permet de voir les différentes tables créées précédemment comme on peut le voir ci-dessous. Il suffit au départ de créer un premier superuser qui aura accès au site d'administration. Une fois la connexion sur le site avec le nom d'utilisateur et le mot de passe du superuser, il est possible d'ajouter des données dans les tables, de les modifier ou d'en supprimer. Cette interface d'administration permet également d'ajouter des utilisateurs et de créer des groupes avec des permissions spécifiques sur la base de données.

Administration du site

The screenshot shows the Django Admin interface with the following sections:

- APPLI** section:

	Ajouter	Modifier
ALTERNATIVE LANGUAGE		
ENTITY		
GROUP		
IS ENTITY IN		
IS PATIENT IN		
IS SITE IN		
IS STAFF IN		
PATIENT		
PROJECT		
SITE		
USER ACCOUNT		
- AUTHENTIFICATION ET AUTORISATION** section:

	Ajouter	Modifier
Groupes		
Utilisateurs		
- Actions récentes** sidebar:

Mes actions
phil Utilisateur
barbara Utilisateur
barbara Utilisateur
barbara Utilisateur
alex Utilisateur
alex Utilisateur
philipe Utilisateur
philipe Utilisateur
4 - Studdy 552-D Feasibility study of doliprane 5k - 552-D-UC152-O - Phase 1 - set-up - 2 - KIN Ara - Umbrella Corporation - Data Manager - 2018-06-14 13:02:42.052968+00:00 IS STAFF IN
4 - Studdy 552-D Feasibility study of doliprane 5k - 552-D-UC152-O - Phase 1 - set-up - 1 - WOLF Black - Umbrella Corporation - Study Manager - 2018-06-14 13:02:36.253545+00:00 IS STAFF IN

FIGURE 7.2 – Interface d'administration de Django

Cette interface a permis aux membres de l'entreprise de créer un premier jeu d'essai minimal en ajoutant des organisations, des projets, des patients et de relier ces différentes données entre elles. Naturellement, ce site étant accessible à une super admin, il est possible de passer outre les contraintes d'intégrités de la base sans que celle-ci ne renvoie d'erreur.

Par exemple, il est possible ici dans la table is_patient_in d'affecter un patient à un projet et de le mettre dans un groupe même si ce groupe ne fait pas partie des groupes du projet. L'interface nous affiche, lors de la sélection d'un groupe pour l'affectation d'un patient à un projet la liste complète des groupes disponibles sans regarder le champ id_project du groupe qui montre l'id du projet auquel un groupe appartient.

Ce premier test aura permis de mettre en évidence quelques erreurs comme dans le nommage de champs notamment sur la notion d'entité qui pouvait être confondue avec le terme entité utilisé par Rasa et par le chatbot.

7.3 Extension du modèle utilisateur

Suite à ces tests, la question de la sécurité des données dans les modèles s'est posée. Au départ nous voulions réutiliser des fonctions que l'on sait performantes pour ce qui est du hachage de mot de passe ou de la gestion des cookies. Après quelques recherches, nous avons découvert que Django crée automatiquement des tables supplémentaires lorsque l'on crée une base de données. Ces tables contiennent notamment, les groupes, les utilisateurs et les permissions.

La table qui nous intéresse tout particulièrement est la table user. En effet cette table user utilise des fonctions particulières notamment une fonction de hachage en SHA256 qui hash le mot de passe 10000 fois avec bien entendu l'utilisation d'un salt pour complexifier ce hashage. L'utilisation de cette table et de ces fonctions aura pour but de nous faire gagner en temps et en sécurité car on peut facilement penser que les fonctions de Django sont plus fiables que celle que l'on pourrait prendre ou réaliser nous-mêmes.

Pour réaliser cette extension il nous a fallu supprimer les tables user et patients et mettre ces données dans la table user de Django. Désormais la différence entre les deux se fera en attribuant à un utilisateur un groupe d'utilisateurs Django spécifique comme user, patient ou administrateur.

Comme la table user de django est fixe et que l'on souhaite rajouter des informations supplémentaires pour un utilisateur comme un numéro de téléphone, un mail il nous faut modifier nos modèles pour créer une table user_détails qui aura une relation OneToOne (équivalant à une cardinalité 1 1) avec la table user de Django.

Ce choix a été fait pour gagner en sécurité mais implique un moins grand contrôle dans les contraintes d'intégrité en mode superuser. En effet, un patient doit être anonyme dans la base donc les champs qui permettent de remonter jusqu'à lui ne devront pas être remplis comme le nom, le prénom un email ou un numéro de téléphone. Il y aura donc une différence de champ à remplir entre un utilisateur et un patient le tout dans la même table ce qui aura pour but de parfois laisser des champs vides. De plus désormais les patients et les utilisateurs étant stocké dans la même table, le problème évoqué plus haut sur les groupes dans les projets se retrouve ici aussi. Un patient pourra désormais être ajouté à un projet en tant que staff car à la proposition de choix Django ne regarde pas qu'une entrée dans la table user et bien un user et non pas un patient.

Malgré quelques points négatifs en superuser, on voit bien que les avantages compensent les

inconvénients surtout dans le domaine médical et que ces inconvénients pourront être réglés en applicatif lors de la réalisation du site web.

7.4 Organisation de notre projet sous Django

Pour réaliser un projet le mieux construit possible nous avons décidé de suivre le modèle MVT avec Django qui gère seul la partie contrôleur et une notion de template, c'est-à-dire de fichiers html qui seront appelés par la vue.

Le site Openclassroom indique ce qu'il reste à réaliser à savoir :

- le routage des requêtes, en fonction d'une Uniform Resource Locator (URL)
- la représentation des données dans l'application, avec leur gestion (ajout, édition, suppression...), c'est-à-dire les modèles
- l'affichage de ces données et de toute autre information au format HTML, c'est-à-dire les templates.
- le lien entre les deux derniers points : la vue qui récupère les données et génère le template selon celles-ci.

On construit donc une architecture avec en racine le dossier qui contient toutes les applis réalisables avec Django notamment celle que l'on doit développer.

📁	decorators	08/08/2018 10:08	Dossier de fichiers
📁	fixtures	08/08/2018 10:08	Dossier de fichiers
📁	Medibot	09/08/2018 10:41	Dossier de fichiers
📁	migrations	08/08/2018 10:08	Dossier de fichiers
📁	static	04/07/2018 14:52	Dossier de fichiers
📁	templates	08/08/2018 10:08	Dossier de fichiers
📁	templatetags	08/08/2018 10:08	Dossier de fichiers
📁	views	08/08/2018 10:08	Dossier de fichiers
📄	_init_.py	13/06/2018 12:19	Python File 0 Ko
📄	admin.py	18/06/2018 11:57	Python File 5 Ko
📄	apps.py	13/06/2018 12:19	Python File 1 Ko
📄	forms.py	20/06/2018 15:08	Python File 1 Ko
📄	middleware.py	08/08/2018 08:49	Python File 5 Ko
📄	models.py	18/06/2018 13:59	Python File 27 Ko
📄	tests.py	13/06/2018 12:19	Python File 1 Ko
📄	urls.py	22/06/2018 10:36	Python File 1 Ko

FIGURE 7.3 – Architecture de l'application

Dans le dossier de cette application visible ci-dessus, on trouve un dossier statique pour les images de notre site, un dossier templates pour nos fichiers HTML, un dossier templatetag pour réaliser nos propres tags par template, un dossier decorators pour le code factorisé avec le pattern du même nom et un dossier view qui contiendra tous nos scripts python appelant les vues.

On trouve également dans ce dossier, le fichier url pour faire le routage, le fichier model.py contenant les modèles, le fichier admin pour personnaliser l'interface d'administration et un fichier de test pour réaliser des tests unitaires. Le dossier Medibot contient lui tous les fichiers qui permettent de faire fonctionner le chatbot que l'on associe à notre application. Ce dossier est le plus gros dossier de l'application.

8 Réalisation de l'application web

8.1 Méthode de travail

Pour réaliser l'application demandée par l'entreprise il va falloir réfléchir à une charte graphique mais également à une structure plus technique pour faire les différentes pages. Cette réalisation d'application suivra une méthode se rapprochant d'une méthode agile en réalisant des pages complètes une par une au cours du temps pour pouvoir petit à petit ajouter de nouvelles fonctionnalités au site.

La nouvelle fonctionnalité mise en place sera directement intégrée à l'application sur le serveur pour pouvoir être testée par les membres de l'entreprise pour voir certains défauts, des incohérences ou des manquements par rapport à ce qui est attendu.

8.2 Réalisation d'un site statique de démonstration

Dans un premier temps, pour avoir une bonne idée de ce que l'entreprise désire en matière de rendu graphique, nous avons décidé de réaliser un site statique en HTML. Ce site, réalisé en dur avec des liens statiques et des données remplies dans des tableaux non dynamiques permettra de valider en plus du mockup définitivement le design graphique de l'application.

Ce site a permis de repositionner certains boutons ou revoir certaines fonctionnalités pour rendre le site plus ergonomique et pour faciliter son utilisation par un utilisateur non-informaticien. De plus cela a également permis de soulever des questions sur d'éventuels panneaux ou fonctionnalités à rajouter ou à changer par rapport à ce que l'on pouvait voir sur le mockup.

8.3 Crédit des premières pages

Une fois cela fait, il fallait donc réaliser les premières pages de l'application. Les premières pages sont celles qui permettront de faire la visualisation de données dans la base ainsi que leur ajout simplement sans pour le moment avoir de lien entre elles. Parmi les pages se trouvant dans la base du site il y a :

- La page de listing et d'ajout d'une organisation
- La page de listing et d'ajout d'un site
- La page de listing et d'ajout d'un utilisateur
- La page de listing et d'ajout d'un patient
- La page de listing et d'ajout d'un projet

Ces pages au fonctionnement plus ou moins similaire permettront d'avoir un bon début d'application pour faire différents tests et vérifier que tout est bien conforme avant d'aller plus loin.

8.4 Récupération des données dans la base

Une fois avoir rempli la base, il nous fallait récupérer ces données dans la base pour pouvoir les afficher sur nos différentes pages. Cette récupération se fait en utilisant les modèles précédemment créés sans avoir à faire des requêtes SQL.

Le modèle Object-Relational Mapping (ORM) de Django nous permet de récupérer facilement des données. Pour ce faire, il suffit de citer le nom du modèle et de récupérer les objets contenus dans celui-ci en y appliquant éventuellement des filtres. On peut choisir ainsi de récupérer des objets entiers, un ensemble d'objets appelés queryset ou une valeur d'un objet c'est-à-dire une des colonnes de la table.

De plus, Django fournit une possibilité très pratique pour le développement qui est la possibilité de remonter des cardinalités 0 N grâce à la commande set. Par exemple, on peut voir dans notre schéma entité association qu'une intent peut avoir plusieurs entités mais qu'une entité n'est associée qu'à une et une seule intent. Ainsi grâce à la cardinalité 1 1 on retrouve immédiatement l'intent d'une entité mais l'inverse est plus compliqué et c'est ici qu'intervient le set. On pourra donc à l'aide d'une entité__set remonter la cardinalité 0 N et obtenir toutes les intents d'une entité donnée.

Une fois ces données récupérées, le template HTML se charge de les afficher dans un tableau bien structuré pour que l'utilisateur puisse facilement voir tout ce dont il a besoin.

8.5 Réalisation des pages d'ajout de données

Pour pouvoir ajouter de nouvelles données sans passer par l'interface d'administration, nous avons réalisé de nombreuses pages pour pouvoir ajouter des données. Ces pages se basent une nouvelle fois sur les modèles de données Django.

Pour ce faire on crée dans le fichier form.py un ensemble de formulaires ModelForm, c'est-à-dire des formulaires qui font créer automatiquement les différents champs à saisir pour correspondre au modèle auquel ils sont associés en faisant un simple model = ORGANIZATION par exemple.

Une fois ceci fait, il suffit de créer le formulaire et de faire des contrôles sur la validité de celui-ci lors de sa soumission et d'appeler la méthode save() pour pouvoir enregistrer ou modifier une donnée dans la base.

Du côté du template le formulaire a été détaillé champ par champ pour pouvoir lui donner un style particulier avec bootstrap et pour pouvoir plus tard lors de la réalisation des champs, pouvoir griser certains champs en masquer d'autres ou encore en désactiver certains.

8.6 Réalisation des filtres et d'une fonction de recherche

Une fois les pages de données réalisées il fallait ensuite proposer des solutions pour permettre à l'utilisateur de trouver rapidement ce qu'il cherche même dans un grand volume de données. Pour ce faire nous avons implémenté une fonction de recherche et une fonction de filtre. Nous avons décidé de rendre la possibilité de filtrer sur nos données pour les champs qui possèdent un nombre fini de choix possible.

Ainsi il est possible de filtrer sur des statuts comme il n'y a généralement que deux à quatre statuts possibles pour un objet. Pour les autres champs comme les noms ou des projets ou des sites par exemple il est plus favorable de procéder à une recherche pour enlever les résultats non souhaités et ne pas surcharger la page.

8.7 Pagination, ordonnancement et URL avec regex

Pour ne pas non plus surcharger l'utilisateur avec beaucoup de données d'un coup, nous avons introduit un système de pagination sur nos tableaux de données. Cette pagination permet de forcer le tableau à n'afficher qu'un certain nombre d'objets et de permettre de changer de "page" du tableau pour voir les X données suivantes. Cette méthode permet également de conserver une page relativement petite car le tableau étant généré automatiquement avec une boucle, il peut potentiellement être très grand avec un grand nombre de données.

Dans la même optique, il est également possible d'ordonner les données du tableau par ordre alphabétique classique ou inverse pour faciliter la lecture des données pour un utilisateur en utilisant un `order_by` sur les modèles.

Enfin, nous avons rendu possible d'accéder à une même page avec différents paramètres dans l'URL. Pour ce faire dans le fichier `url.py` on spécifie l'url d'accès à une page avec ou sans expression régulière. Cela nous permet de ne pas avoir à refaire certaines pages. L'exemple le plus parlant est celui des pages de modifications de données et de visualisation par rapport aux pages d'ajout des données. Au lieu de faire des pages différentes, on utilise la même page et on vérifie ou non la présence d'un `requested_id` dans l'URL. Si cet id est présent cela signifie que l'utilisateur cherche à accéder à une donnée particulière pour en voir le détail et éventuellement modifier une donnée car tout n'est pas affiché dans le tableau toujours pour ne pas surcharger. Par contre, si cet id n'est pas présent cela signifie que l'utilisateur souhaite ajouter une nouvelle donnée.

Dans le deuxième cas, les champs du formulaire seront grisés car si l'on souhaite voir le détail, on ne peut entreprendre aucune action. Dans le premier cas, les `requeste_id` font apparaître des boutons supplémentaires pour pouvoir modifier ou supprimer une donnée et ne grisent pas les champs pour effectuer lesdites modifications. Ces doubles utilisations d'une même page nous ont permis de gagner du temps de travail et permettent de ne pas surcharger l'application de fichier python et html.

8.8 Implémentation des différents rôles

L'application ayant pour but d'être utilisée par de nombreuses personnes venant de plusieurs organisations différentes, il a fallu définir des rôles d'utilisateurs ainsi que leurs droits. Pour rappel, lorsqu'un utilisateur est ajouté à un projet, on lui associe un rôle pour ce projet ce qui va lui donner différents droits.

Après analyse des besoins des différents rôles, les rôles suivants ont été déterminés :

- *Le rôle administrateur :*

Ce rôle n'est pas un rôle à proprement parler car on ne pourra pas affecter un admin à un projet mais représente la personne qui devra avoir un accès complet à toutes les données de l'application. Les personnes généralement de l'entreprise qui possèdent ce rôle peuvent ajouter de nouvelles données mais ces personnes ont quelques restrictions au niveau du patient.

Ils ne peuvent pas importer de patient via un fichier Excel par exemple ou en créer de nouveau à la main car ils n'ont pas vocation à le faire tout comme les personnes ayant le

rôle de project manager ou de sponsor.

De plus il leur est impossible de faire la gestion des sites médicaux au sein des projets créés par les utilisateurs de l'application.

– *Le rôle Project Manager :*

Ce rôle ainsi que le suivant sont des rôles qu'il est possible d'affecter à un utilisateur lorsqu'il rentre dans un projet. Ce rôle se rapproche d'un rôle administrateur mais beaucoup plus recentré sur le projet auquel il est affecté. Il s'agit de la personne qui devra gérer le projet et le superviser. Ce rôle comme ceux en dessous n'ont pas la possibilité de créer ou d'importer de nouveau projet dans l'application ni même d'en supprimer car cela doit rester propre à un administrateur.

De la même façon, la gestion complète des organisations, que ce soit l'ajout de nouvelles organisations souhaitant participer à des projets, leurs mises à jour, suppression, import et export ne sont pas disponibles pour ce rôle et pour les rôles suivants. Cette gestion des organisations n'est pas autorisée que ce soit au niveau global ou au niveau d'un projet, même pour le projet pour lequel l'utilisateur est project manager.

– *Le rôle Sponsor :*

Ce rôle est affecté à une personne de l'organisation qui a commandé le projet. Cette organisation n'étant pas obligée de participer au projet elle peut avoir des personnes de chez elles qui garderont un contrôle sur ce qui peut se passer dans ce projet et qui souhaitent voir les résultats.

Ce rôle a pour but principal de faire de la visualisation de données c'est pour cela que ce soit au niveau global ou au niveau du projet, un sponsor ne peut rien créer ou modifier et ne peut pas importer non plus. Toutefois il a accès en read only à l'intégralité des données du projet même s'il ne peut pas les exporter.

– *Le groupe de rôle Monitor (Data Manager et CRA) :*

Les deux rôles de ce groupe sont des utilisateurs qui sont là pour gérer les patients parmi les organisations qui participent aux projets mais qui peuvent quand même consulter les données du projet sans pour autant pouvoir en ajouter, modifier ou en supprimer.

– *Le groupe de rôle Site (Nurse et Investigator) :*

Enfin ce dernier groupe et le groupe le plus restrictif, ils n'ont accès qu'aux données du projet et au patient pour les gérer mais ne peuvent rien faire de plus.

Ces rôles nous servent à pouvoir ajouter des restrictions d'accès au page avec des redirections si la personne ne possède pas le ou les rôles qui permettent d'accéder à cette page. Ils permettent également de bloquer les boutons ou les champs avec lesquels un utilisateur ne peut pas interagir.

Lors de la réalisation de ces rôles, un problème s'est posé. En effet, un rôle est défini pour un utilisateur lors de son affectation à un projet. Il y a donc une possibilité même si très faible qu'un utilisateur puisse être affecté à plusieurs projets avec des rôles différents.

Pour éviter cette situation on peut facilement faire en sorte de créer un seul compte utilisateur pour un rôle donné de manière à n'avoir qu'un et un rôle par compte. Toutefois comment faire

lorsque cette situation arrive ?

À ce problème vient également s'ajouter celui concernant les pages générales qui ne concernent pas le projet. Lorsqu'on se trouve dans un problème, il est très facile de requérir la base pour connaître quel est le rôle de l'utilisateur pour ce projet mais comment faire lorsqu'il n'est pas dans un contexte projet et qu'il faut déterminer ses droits d'accès ?

La solution choisie a été de regarder lorsqu'un utilisateur arrive sur une page ; toujours dans un contexte hors projet et dans le cas où il posséderait des rôles différents ; de regarder l'ensemble des rôles qu'il possède et de voir s'il possède au moins un rôle qui lui donne l'accès à la fonctionnalité auquel l'utilisateur souhaite accéder.

Ainsi si un utilisateur est à la fois project manager et nurse, il cumulera les droits d'accès de ces deux rôles et pourra donc ajouter des patients via son rôle de nurse mais également consulter des sites via son rôle de project manager.

Cet aspect de gestion des rôles est un des gros piliers de l'application et sera sûrement amené à évoluer avec le temps. En effet pour le moment le choix des droits pour un rôle a été fait celui de coder en dur les différents accès par rapport à la matrice rôle/droit. Sur le long terme on peut imaginer réaliser une partie qui viendra s'ajouter à la base de données pour pouvoir spécifier un ensemble de droit d'accès pour un rôle donné et pour pouvoir les modifier avec le temps.

Un résumé graphique de cet ensemble de permission est accessible en annexe E de ce rapport.

8.9 Questionnement sur la sécurité du site

Une fois toutes ces pages réalisées, le plus gros du site est alors créé et il fallait donc s'assurer que cette base était solide en termes de sécurité du moins concernant les failles les plus faciles à exploiter. Encore une fois on peut compter sur Django pour empêcher l'utilisation de certaines failles classiques.

Parmi elles, on peut citer les injections HTML et SQL qui sont évitées en faisant un échappement automatique de tout ce qui est rentré dans les formulaires pour éviter ces problèmes. Django protège également le site contre les failles Cross-Site Scripting (XSS) où un éventuel pirate voudrait envoyer du code pour y être exécuté. Il gère également les failles Cross-Site Request Forgery (CSRF) en donnant un identifiant à un formulaire créé pour vérifier que c'est bien le bon formulaire voulu qui est envoyé.

De plus en ayant choisi de laisser Django se charger de la gestion des mots de passe, ceux-ci sont sécurisés par Django et on n'a plus besoin de se soucier la sécurité de ceux-ci. En effet Django utilise ses fonctions de hashage pour hasher le mot de passe un grand nombre de fois, il ne permet pas de rentrer des mots de passe trop fragile qui pourraient facilement être trouvés ou encore qui ressemblent trop au nom d'utilisateur entré précédemment.

Une autre grosse faille à corriger a été celle instaurée lors de la création des différents rôles et des restrictions associés. En effet pour réaliser les rôles, les différents templates HTML se sont vus complétés par des éventuels attributs disabled si le rôle ne possède pas le droit d'accès à une fonctionnalité. Le problème rencontré vient quand quelqu'un décide de regarder le code HTML côté client et de supprimer cet attribut disabled de sa page. Cette suppression d'attribut côté client

permet donc à la personne d'accéder à toutes les fonctionnalités peu importe son rôle et lui donne donc un accès plus grand sur les pages à laquelle il a accès.

Ce problème ayant lieu côté client, il est impossible de faire en sorte que le client ne puisse pas modifier sa page ; c'est pourquoi nous avons rajouté des sécurités lors de la validation de formulaire pour vérifier si en plus d'envoyer un formulaire valide, l'utilisateur a le droit d'envoyer un formulaire sur cette page(c'est-à-dire si son rôle le permet).

Pour le reste des sécurités, le manque de connaissances dans le domaine et le temps imparti pour le faire fait que nous n'avons pas cherché à ajouter plus de sécurité que celle nécessaire à résoudre les problèmes que nous pouvions constater. On peut notamment penser à passer le serveur sur un protocole Hypertext Transfer Protocol Secure (HTTPS) plutôt qu'en Hypertext Transfer Protocol (HTTP) classique pour sécuriser les données médicales.

Cet aspect sécurité devra donc être réalisé par quelqu'un ayant les connaissances avant que l'outil ne puisse vraiment être utilisé.

8.10 Les tests unitaires

Pour ce stage, il a fallu faire un choix entre développer de nouvelles fonctionnalités ou s'assurer que les fonctionnalités existantes font parfaitement ce que l'on attend d'elles. Le but du stage étant avant tout de réaliser une preuve de concept, il fallait qu'un maximum de fonctions soit disponible pour pouvoir montrer l'application à d'éventuels clients.

De ce fait nous n'avons réalisé aucun test unitaire lors de ce stage bien que nous ayons conscience de leur importance et ces tests devront également être réalisés par quelqu'un d'autre dans le futur.

Selon notre outil de review de code, on peut estimer à environ 430 tests unitaires à réaliser pour couvrir l'ensemble des fonctions qui ont été réalisées.

8.11 Réalisation des fonctions d'insertions pour le chatbot

À ce stade, l'application possède la majeure partie des fonctions souhaitées et il reste à réaliser une des grosses parties à savoir la fusion de notre travail avec celui de Graziella.

Cette fusion consiste à intégrer un chatbot fonctionnel à l'application pour que le patient puisse parler et que ses dires soient enregistrés dans la base de données. Pour se faire, nous avons ajouté des fonctions d'insertions dans la base en compléments de certaines actions réalisées par le chatbot. Ces fonctions consistent en deux choses, la première est une fonction d'insertion dans la table conversation quand le bot dit quelque chose ou bien quand le patient s'exprime pour pouvoir conserver l'intégralité de ce qui a pu être dit dans une conversation. La deuxième fonction est celle qui insère les informations à sauvegarder pour le patient. Une fois que le patient a fini par exemple de parler de la douleur, le bot lui fait un récapitulatif de ce qu'il a dit pour voir si toutes ces informations sont correctes. Si elles le sont, le bot va tout enregistrer dans la table answer. Un exemple serait sur les informations de la douleur où le bot va sauvegarder le lieu de la douleur, son intensité, sa durée etc.

N'ayant pas le temps de réaliser l'interface patient, nous avons fait le choix de connecter le chatbot à l'application Slack et d'utiliser nos comptes Slack comme des comptes patients pour réaliser les tests et avoir une communication fonctionnelle.

Un problème s'est posé lors de cette implémentation car dans son fonctionnement, le bot possédait des variables globales qui étaient communes à tous les patients et donc dans notre cas à tous les comptes Slacks. Il a donc fallu pour régler ce problème créer des tables supplémentaires dans la base de données en créant de véritables patients et leur associer l'identifiant Slack de chacun de nos comptes.

Cette table permet de stocker un ensemble de valeur comme par exemple un booléen pour savoir si c'est la première conversation entre le bot et l'utilisateur ou encore les différentes dates.

Il est évident que cette situation est temporaire et que, à terme, l'application possèdera sa propre application de conversation sans passer par Slack.

8.12 La dataviz

Une fois cette gestion de la conversation entre le bot et le patient faite, il nous fallait un moyen dans l'application de pouvoir voir les données que le patient a communiqué ainsi que la sauvegarde des résultats.

Le choix s'est porté sur la possibilité de choisir une date pour laquelle on souhaite voir ce qui a été dit par un patient. En choisissant la date, l'ensemble de la conversation pour cette date est affiché dans un format de chat et en dessous on trouve un tableau récapitulatif des informations que le bot a sauvégardé pour ce patient.

On a également la possibilité pour une lecture plus claire, de sélectionner qu'une seule intention pour avoir un ciblage sur la douleur ou sur les aspects sociaux exprimés par le patient. Si aucun filtre n'est sélectionné, alors l'intégralité des intentions sera affichée par défaut.

En plus de cela il y a la possibilité d'afficher des graphiques pour des données qui peuvent être traitées de cette façon. Par exemple on peut réaliser un graphique qui montre l'évolution du niveau de douleur en fonction du temps.

8.13 L'audit trail

L'application étant liée au domaine médical, tout ce qui se fait dans la base de données doit être enregistré pour que l'on puisse facilement retracer tout ce qu'une personne a pu faire dans l'application. Ce suivi de la personne se déroule en trois étapes.

La première étape consiste à enregistrer toutes les tentatives de connexion d'un utilisateur et d'indiquer dans la base si cette tentative a échoué ou réussi en indiquant de quel IP la tentative de connexion a été réalisée.

La seconde consiste à enregistrer toutes les demandes d'accès à une page d'un utilisateur pour savoir exactement sur quelles pages il s'est rendu et ainsi suivre ses actions.

Enfin, la dernière, la plus importante est de pouvoir savoir quand une personne ajoute, modifie ou supprime une donnée. Pour réaliser cela j'ai dû modifier la méthode de sauvegarde de Django pour lui faire faire un enregistrement supplémentaire à chaque fois qu'une des tables de la base

est modifiée suite à un ajout, une modification ou une suppression.

Lors d'une modification, la nouvelle méthode de sauvegarde va regarder la valeur actuellement stockée dans la base et la comparer avec la valeur souhaitant être insérée. S'il y a une différence alors Django fera une insertion supplémentaire dans la table d'audit en signifiant la valeur avant et la valeur après, la date de mise à jour, la personne responsable de cette mise à jour le tout pour chaque champ des tables nécessitant d'être suivi.

Ces modifications permettent de faire les comparaisons en base au lieu de devoir les faire au sein de chaque fichier mettant à jour les différents champs de la base de données.

8.14 Réalisation des différents manuels de documentation

Pour préparer au mieux notre succession sur ce projet, une documentation a été faite. Pour cette documentation je me suis chargé de toute la partie commentaire de code et explication tandis que Guillaume a réalisé les manuels qui accompagneront l'application.

La première documentation consiste à réaliser un manuel utilisateur pour que toutes les personnes qui utilisent l'application puissent accéder à toutes ses fonctionnalités de la façon la plus simple possible. Cette documentation a été réalisée en se plaçant du point de vue utilisateur en évitant les mots techniques et en expliquant le plus simplement possible les possibilités offertes par l'application.

La deuxième documentation est une documentation technique pour aider la ou les personnes qui devront reprendre le développement derrière nous. Ce document expliquera l'architecture de l'application, celle de la base et les dépendances nécessaires à installer pour faire fonctionner l'application en local ou sur un serveur.

Enfin les commentaires ajoutés sur le code permettront à la personne qui reprend de bien comprendre notre code et de pouvoir facilement le compléter ou le modifier.

9 Bilan du stage

9.1 Principales difficultés

Parmi les difficultés, la première fut de constamment devoir faire évoluer le Modèle Conceptuel de Données (MCD). Cette constante évolution nous a poussé à faire des modifications des modèles et donc par extension de devoir changer à chaque fois l'intégralité des pages du site utilisant les modèles qui ont été modifiés. Une validation définitive plus rapide nous aurait permis de gagner du temps et de ne pas avoir à refaire du travail déjà effectué auparavant.

Ensuite, comme nous avons une méconnaissance du domaine médical, cela a engendré certaines incompréhensions. Le besoin du domaine des études cliniques étant très strict cela a également provoqué des ralentissements dans le développement de l'application.

Enfin, la dernière difficulté fut liée au manque de personnes compétentes en informatique qualifiée au sein du projet. Au cours de ce stage, nous n'avons pas eu la possibilité d'être supervisés par quelqu'un qui aurait pu nous faire profiter de son expérience dans le développement pour nous signifier certaines erreurs ou certaines façons de penser incorrectes pour le développement d'une application.

Ce problème nous a donc obligé à prendre des décisions tout au long du développement sans être sûr que nous n'en prenions pas de mauvaises. De plus, lors de la confrontation à un souci technique nous ne pouvions compter que sur nous-mêmes pour le résoudre sous peine de rester bloqués indéfiniment ou de devoir laisser tomber.

9.2 Evolutions futures

Pour continuer le développement de l'application, on peut imaginer un grand nombre de possibilités d'amélioration ou nouvelles fonctions qu'il sera possible d'ajouter.

Parmi ces améliorations, on peut penser à rajouter des tests unitaires et des fonctions de sécurité. En amélioration, on peut ajouter à corriger le reste des erreurs détecté dans notre outil de review de code pour enlever le code mort, les importations inutiles ou pour factoriser du code. Il serait également bien de faire en sorte de baisser si possible la complexité du code notamment dans les grandes boucles imbriquées en voyant s'il n'est pas possible de faire mieux.

Dans les nouvelles fonctionnalités, il faudra gérer les anciens mots de passe ainsi que la sauvegarde des 5 derniers mots de passe pour des contraintes propres au domaine des études cliniques. Il faudra également améliorer sans cesse le bot dans le but de maximiser sa qualité de détection d'intention pour avoir une application de plus en plus qualitative et performante.

Enfin la grande partie restante pour avoir un projet fonctionnel est la réalisation de toute l'interface côté patient. Celle-ci contiendra comme dit précédemment, la fenêtre de chat pour que le patient puisse discuter avec le bot pour donner son état de santé général.

10 Conclusion

Pour conclure, ce stage aura permis la création non pas d'un prototype mais d'une version alpha d'un futur logiciel commercialisable par l'entreprise. Cette version alpha permet à l'heure actuelle de gérer toutes les parties côté utilisateur ainsi que celle côté admin la quasi-totalité des fonctionnalités souhaitées de ce côté.

Cette version sera donc présentée à des personnes souhaitant s'intéresser au projet et pour pouvoir également obtenir quelques subventions extérieures pour aider le développement futur de cette application.

Bien évidemment, il reste de nombreux points à améliorer et de nouvelles fonctionnalités à développer pour rendre cette application totalement opérationnelle.

Ce stage aura été pour moi l'occasion de parfaire mes connaissances en python tout en apprenant à utiliser un nouveau framework qui est à mon sens beaucoup plus simple d'utilisation et beaucoup plus efficace que google app engine utilisé en cours de web. Cela aura été également un stage qui m'aura permis de voir comment se déroule un projet au sein d'une start-up qui est assez différent de l'ambiance d'une entreprise plus classique.

De plus, cela m'a permis de pouvoir bien progresser dans le domaine du travail collaboratif avec 2 collègues de l'école avec qui j'ai pu échanger et résoudre les problèmes que nous avons rencontrés tout au long de ce développement et ce malgré l'absence d'un référent en ce qui concerne les connaissances en informatique.

Enfin, le plus important à mes yeux et ce que je retiendrai de ce stage est que j'ai, avec mon PIDR en amont de ce stage, pu suivre le développement d'un projet important en informatique de l'idée de base avec la notion de chatbot à intégrer au domaine médical pour faciliter les besoins du patient jusqu'à la réalisation quasi complète de l'application comprenant tout ce qu'il faut pour gérer les essais cliniques et avec l'intégration de ce bot.

Cela m'aura donc permis de voir comment se passe le développement d'un projet de l'idée jusqu'à sa réalisation concrète dans un environnement avec de réels enjeux par rapport aux projets que j'ai pu réaliser dans le cadre de mes études.

Bibliographie / Webographie

- [1] Mathieu Xhonneux Maxime Laurent. Développez votre site web avec le framework django.
10

Liste des illustrations

2.1	Organigramme ALIAE	2
4.1	Logo de Postgresql	4
4.2	Logo de Django avec celui de Python	5
5.1	Logo de Openclinica	6
6.1	Logo de Scaleway	11
7.1	Exemple d'un modèle Django	13
7.2	Interface d'administration de Django	14
7.3	Architecture de l'application	16
D.1	Schéma entité association partiel	39
E.1	Table de permission	40
F.1	Table de permission	41

Glossaire

API Application Programming Interface. 5

CRO Contact Research Organization. 3, 9

CSRF Cross-Site Request Forgery. 21

CSS Cascading Style Sheets. 5

DUT Diplômes Universitaires de Technologie. 4

ECRF Electronique Case Report Form. 1

EDC Electronique Data Capture. 1

GPL General Public License. 6

HTML Hypertext Markup Language. 5, 16, 17, 21

HTTP Hypertext Transfer Protocol. 22

HTTPS Hypertext Transfer Protocol Secure. 22

LGPL Lesser General Public License. 6

MCD Modèle Conceptuel de Données. 25

MLD Modèle Logique de Données. 9

MVT Modèle Vu Template. 5, 16

ORM Object-Relational Mapping. 18

PHP Hypertext Preprocessor. 5

PRO Patient Reporting Outcome. 1

SAS Société par action simplifiée. 2

SGBD Système de Gestion de Base de Données. 4

SQL Structured Query Language. 4, 11, 13, 17, 21

URL Uniform Resource Locator. 16, 19

VPS Virtual Private Server. 11

XSS Cross-Site Scripting. 21

Annexes

A Annexe 1 : Note de cadrage

Note de cadrage

1. Les acteurs du projet

Pour ce projet, les acteurs sont :

- Guillaume KOENIG : développeur étudiant en 2e année à TELECOM Nancy
- Graziella HUSSON : développeuse étudiante en 3e année à TELECOM Nancy
- Thomas GRASSELLINI : développeur étudiant en 2e année à TELECOM Nancy
- Philippe JOLIVET : responsable management de l'équipe
- Alexandre DURAND SALMON : responsable management de l'équipe
- Barbara ZWICK

2. Utilité du projet

Le projet consiste en la réalisation d'une application permettant d'effectuer un suivi de patients lors d'études cliniques et post-cliniques en utilisant un chatbot. Le projet comprend plusieurs parties distinctes :

- Le chatbot permettant de converser avec les patients et de les comprendre
- Le front-office permettant au patient de converser avec le chatbot de manière textuel
- Le back-office permettant la gestion des informations concernant les différents aspects d'une étude clinique ou post-clinique.

3. Objectifs du projet

Les objectifs prioritaires sont :

- Concevoir une base de données fiable et robuste dans le temps répondant aux exigences et réglementations du domaine et assurant la sécurité des données.
- Réaliser un back-office web comprenant les fonctionnalités critiques du projet :
 - Gestion des utilisateurs
 - Gestion des rôles (administrateur, sponsor, site et patient)
 - Gestion des sites (hôpitaux ou cliniques)
 - Gestion des patients
 - Gestion des projets (étude clinique ou post-clinique)
 - Gestion du chatbot
 - Visualisation des données
 - Enregistrement des logs
 - Export des données
- Réaliser un front-office en utilisant une page web

À ce stade, le projet comporte les bases permettant de le présenter afin de le commercialiser. Il doit comporter une documentation précise afin de poursuivre son développement dans le temps.

5. Organisation du projet

Pour favoriser la réussite de ce projet, il a été décidé de travailler en suivant le principe de méthodes agiles avec des sprints réguliers sur des périodes de 1 à 2 semaines.

Ces différents sprints seront référencés dans un diagramme de Gantt et devront aboutir à des livrables en fin de période.

Les différents acteurs du projet travailleront ensemble dans un même espace pour maximiser la communication entre eux. De plus, un outil de gestion de version sera utilisé pour permettre un travail collaboratif et répartis entre les membres de l'équipe projet.

Pour chacun des sprints, l'équipe construira une to-do list sur un tableau blanc avec des papillons adhésifs pour pouvoir respecter au mieux les délais imposés.

En outre, des réunions se tiendront régulièrement entre les membres de l'équipe pour permettre de faire le point sur l'avancement du projet par rapport au planning et pour pouvoir gérer d'éventuels problèmes. Ces réunions donneront aux membres la possibilité de prendre des notes personnelles pour les aider dans la gestion de leur projet.

Au bout de chaque sprint, les livrables seront testés par les membres de l'entreprise pour garantir que tout est conforme à leur demande.

6. Technologies utilisées

Le système de base de données choisis est **PostgreSQL**. Nous avons choisi PostgreSQL, car il est adapté à de grands volumes de donnée. Même si au début le nombre de tables sera relativement faible il y a un fort risque d'augmentation du nombre de celle-ci avec le temps (ajouts de fonctionnalités, etc.). De plus, certains services existants au sein de l'entreprise fonctionnent déjà en utilisant PostgreSQL ce qui permet de centraliser toutes les bases au même endroit. Lors de nos phases de recherche, nous avons retenu plusieurs SGBDR. Le premier, MySQL est celui que nous avions déjà utilisé. Le second est PostgreSQL et enfin le troisième est Oracle que nous avons utilisé à l'école en première année. Nous avons directement écarté Oracle, car il représente un coût important en termes de budget. Nos recherches entre MySQL et PostgreSQL montrent que PostgreSQL est adapté aux grandes bases de données. Celui-ci garantit aussi une cohérence des données et effectuer des tests d'intégrité garantissant la fiabilité des données et de la base de données.

L'application web sera développé en **Python** et utilisera le Framework **Django**. Python et Django sont intransigeant sur les bonnes pratiques, le fait de ne pas se répéter, l'utilisation du MVT (Modèle Vue Template), les tests unitaires, les migrations de base de données et l'utilisation d'un ORM (pour gérer les requêtes à la BDD). Tout cela contribue à obtenir un code sain, maintenable, facilement récupérable par un autre. En effet, le projet n'ayant pas vocation à s'arrêter après le stage, il est important de fournir un code propre et lisible. Autre avantage, Python est assez simple d'utilisation, il existe une grande variété des bibliothèques permettant de couvrir un large champ de demande. Il existe aussi une grande

quantité de documentation Python sur Internet. À la place de Python, nous avons envisagé PHP et Javascript, mais ils sont complexes à utiliser et lourd. Une autre proposition a été de développer une application web en Java EE. N'ayant jamais développé d'application web en Java EE, nous avons préféré utiliser Python.

Afin de rendre le site Web Responsive et faciliter le travail sur la partie graphique, nous utiliserons le framework CSS **Bootstrap**. Il fournit un large choix de classe CSS évitant ainsi de re développer de zéro certaines choses.

7. Gestion du temps

a. Liste des livrables :

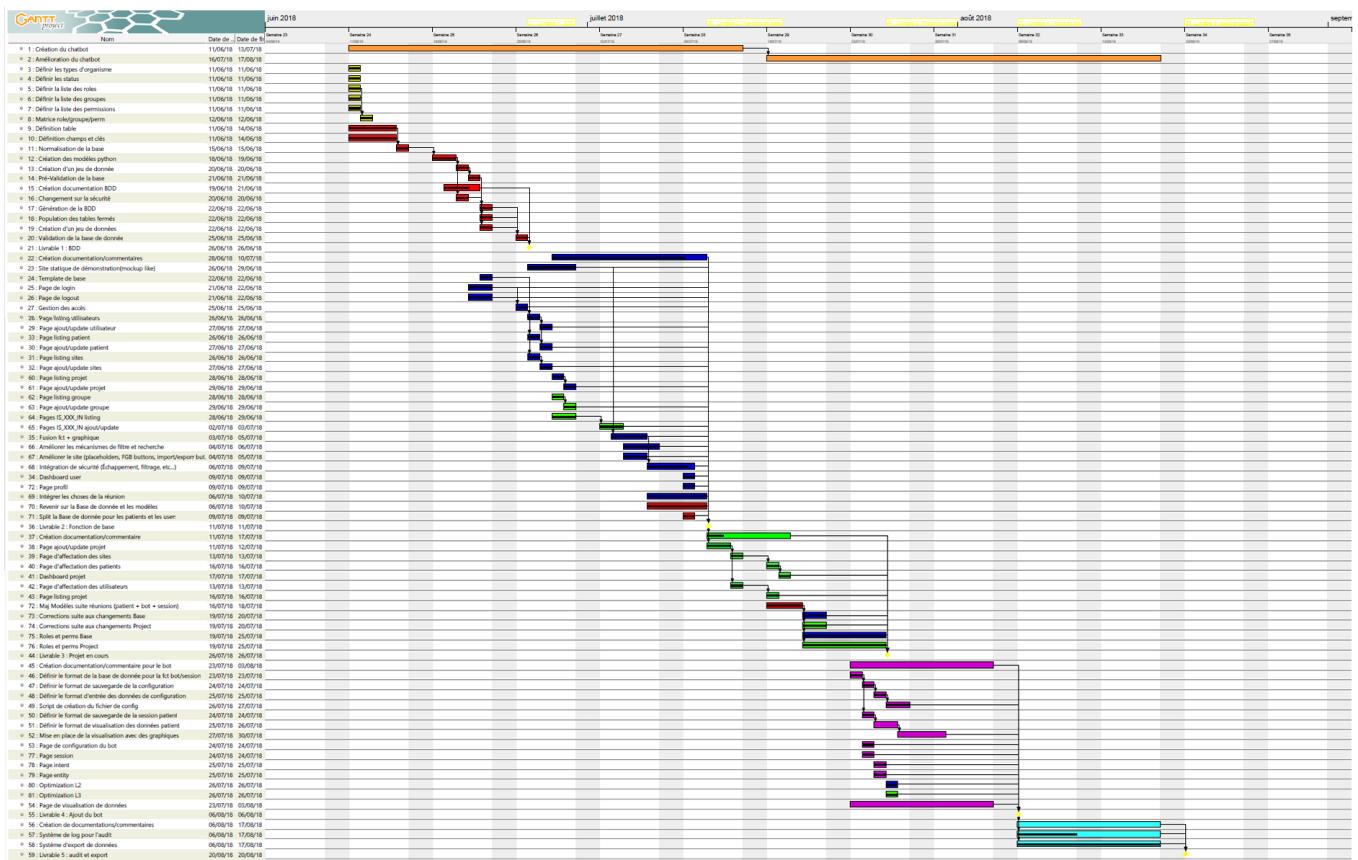
- Livrable 1 : Définition de la base de données. La définition de la base de données consiste en la création d'un schéma fiable et robuste dans le temps prenant en compte les différents aspects du projet et n'empêchant pas son futur développement. La base de données doit suivre un format de donnée permettant sa portabilité. De plus, celle-ci doit répondre aux contraintes réglementaires imposées dans le domaine des études cliniques.
- Livrable 2 : Fonctionnalités de base. Les fonctionnalités de base comportent à minima :
 - La gestion des utilisateurs et des rôles
 - Une page permettant de lister les utilisateurs
 - Une page permettant de créer, modifier ou supprimer un utilisateur
 - La gestion des sites
 - Une page permettant de lister les sites existants
 - Une page permettant de créer, modifier ou supprimer un site
 - La gestion des patients
 - Une page permettant de lister les patients existants
 - Une page permettant de créer, modifier ou supprimer un patient
- Livrable 3 : Gestion des projets. La gestion des projets permet de gérer les études cliniques ou post-cliniques en y ajoutant des informations, des utilisateurs ayant un rôle précis, des sites et des patients. Elle comporte à minima :
 - Une page permettant de lister les projets en cours
 - Une ou plusieurs pages permettant de créer un projet
 - Une page permettant d'obtenir les détails d'un projet
- Livrable 4 : Gestion du chatbot et visualisation. Le chatbot permet de récupérer des données. Ces données sont stockées dans une base de données. Ce livrable doit permettre le lien entre le chatbot et le back-office. Il comporte à minima :
 - Une page de configuration pour le bot du projet.
 - Un système de visualisation des données par patient.
- Livrable 5 : Gestion des audits, et l'export de données avec à minima :
 - Un système pour enregistrer les activités faites par les utilisateurs sur l'application pour savoir qui a modifié les données, quand et pourquoi.
 - Une page permettant l'export des données de la base selon différents formats.

b. Échelonnage des livrables

- Le premier livrable concernant la création du schéma relationnel devra être réalisé en 2 semaines, commencé à la date du 11/06/2018 et rendu à la date du 22/06/2018
- Le second livrable concernant les fonctionnalités de base devra être réalisé en 1 semaine, commencé à la date du 25/06/2018 et rendu à la date du 29/06/2018
- Le troisième livrable sur la gestion des projets devra être réalisé en 2 semaines, commencé à la date du 02/07/2018 et rendu à la date du 13/07/2018
- Le quatrième livrable sur la gestion du chatbot et sa visualisation dans l'application devra être réalisé en 3 semaines, commencé à la date du 16/07/2018 et rendu à la date du 03/08/2018
- Le cinquième livrable l'audit et l'export de données devra être réalisé en 2 semaines, commencé à la date du 06/08/2018 et rendu à la date du 17/08/2018

Le temps restant pour la durée de ce stage sera alloué pour prévoir d'éventuels retards.

B Annexe 2 : Gantt du projet



C Annexe 3 : Mockup de l'application

Cette image représente une des nombreuses captures d'écran du mockup défini par l'entreprise. Le mettre dans sa totalité aurait surchargé le rapport donc en voici seulement un exemple pour donner l'idée générale.

Customer Options 3 / 31

A Web Page http://

The mockup shows a web browser window with a header bar and a sidebar menu. The sidebar contains icons for Dashboard, Customers, Projects, Conversations, Notifications, Reports, and Help. The main content area is titled 'Customer details' and shows a customer record for 'C1'. The record includes fields for Customer name, Main Contact Clinical, Main Contact Finance, Address, Zip Code, City, Country, Status (Active or Closed), and Comments. There are also 'Insights', 'Tools', and 'Delete' buttons above the form. At the bottom are 'Cancel' and 'Save' buttons.

D Annexe 4 : Schéma entité associations

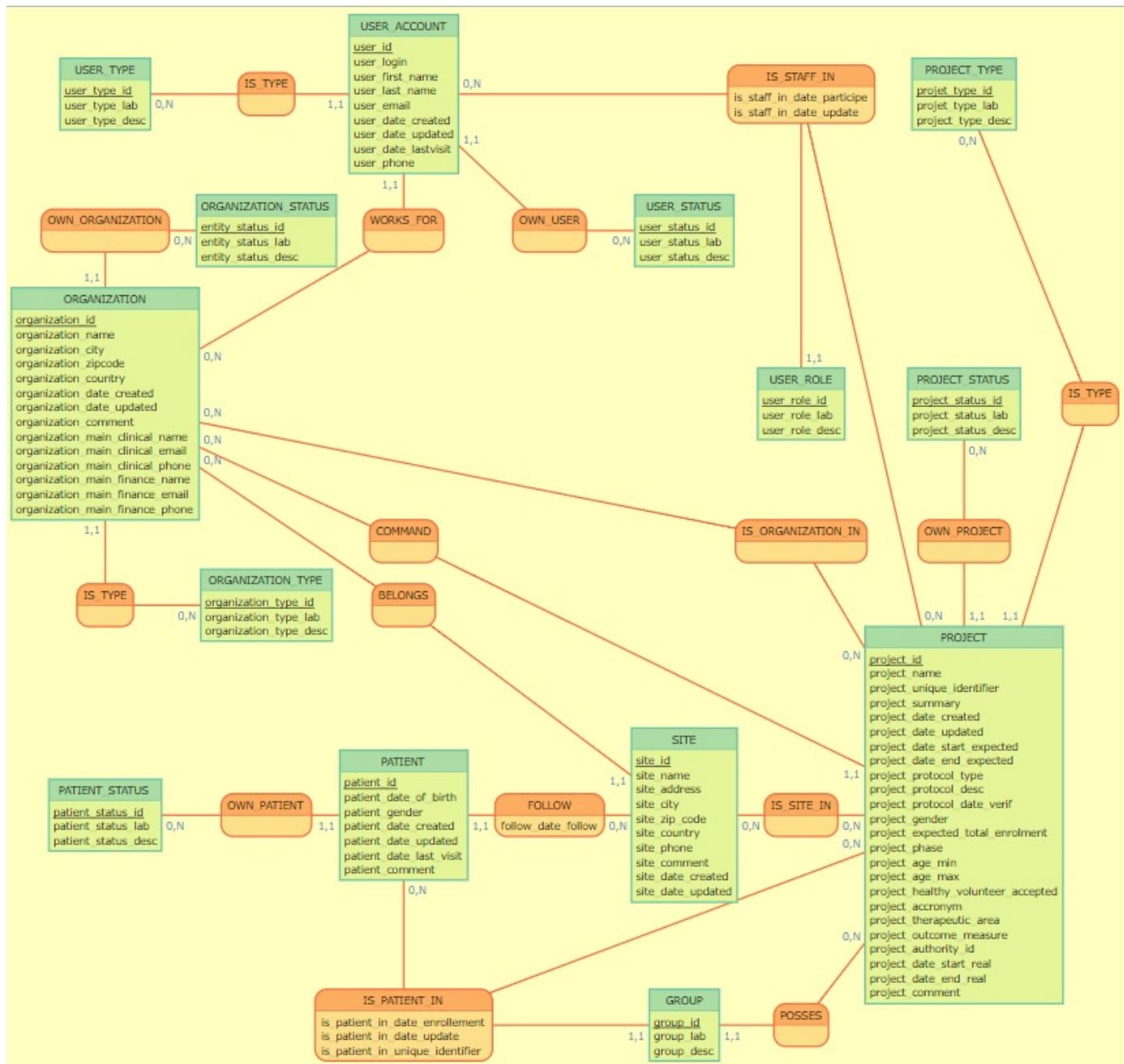


FIGURE D.1 – Schéma entité association partiel

E Annexe 5.1 : Table de permission des rôles

		Administrator <i>full access on projects DB</i>	Project Manager <i>can access full data at project(s) level (only project assigned by Admin)</i>	Sponsor <i>can access full data at project(s) level (read only mode) (only project assigned by Admin)</i>	Monitor (=DM, =CRA) <i>can access data of a limited number of assigned site(s)</i>	Site (=inv, nurse) <i>can access data of the assigned site only</i>
Dashboard	View standard widget	yes	yes	yes	yes	yes
Project	View Project List	yes	yes	yes	yes	yes
done	Import Project List	yes	no	no	no	no
done	Export Project List	yes	yes	yes	yes	no
done	Add Project to DB	yes	no	no	no	no
done	Update Project to DB	yes	yes	no	no	no
done	Delete Project to DB	yes	no	no	no	no
Organization	View Organization List	yes	yes	no	no	no
done	Import Organization List	yes	no	no	no	no
done	Export Organization List	yes	no	no	no	no
done	Add Organization to DB	yes	no	no	no	no
done	Update Organization to DB	yes	no	no	no	no
done	Delete Organization to DB	yes	no	no	no	no
done	Add Organization to Projec	yes	no	no	no	no
Site	View Site List	yes	yes	yes	yes	no
done	Import Site List	yes	yes	no	no	no
done	Export Site List	yes	yes	no	no	no
done	Add Site to DB	yes	yes	no	no	no
done	Update Site to DB	yes	yes	no	no	no
done	Delete Site to DB	yes	yes	no	no	no
done	Add Site to Project	no	yes	no	no	no

FIGURE E.1 – Table de permission

F Annexe 5.2 : Table de permission des rôles

		Administrator <i>full access on projects DB</i>	Project Manager <i>can access full data at project(s) level (only project assigned by Admin)</i>	Sponsor <i>can access full data at project(s) level (read only mode) (only project assigned by Admin)</i>	Monitor (=DM, =CRA) <i>can access data of a limited number of assigned site(s)</i>	Site (=inv, nurse) <i>can access data of the assigned site only</i>
Staff	View Staff List	yes	yes	yes	yes	no
done	Import Staff List	yes	yes	no	no	no
done	Export Staff List	yes	yes	no	no	no
done	Add Staff to DB	yes	yes	no	no	no
done	Update Staff to DB	yes	yes	no	no	no
done	Delete Staff to DB	yes	yes	no	no	no
done except	Add Staff to Project	yes	except for PM of pro	no	no	no
	Add Staff to Site	yes	yes	no	no	no
Group	View Group List	yes	yes	yes	yes	yes
done	Import Group List	yes	yes	no	no	no
done	Export Group List	yes	yes	no	no	no
done	Add Group to project	yes	yes	no	no	no
Patient	View Patient List	yes	yes	yes	yes	yes
done	Import Patient List	no	no	no	yes	yes
done	Export Patient List	yes	yes	yes	yes	yes
done	Add Patient to G/S/P	no	no	no	yes	yes

FIGURE F.1 – Table de permission

Résumé

Ce stage a pour but de réaliser une application pour la gestion des études cliniques. Ce domaine faisant partie du domaine médical, ce projet doit respecter toutes les contraintes imposées par ce domaine.

Le développement de cette application a été réalisé avec le langage python et le framework de développement web Django. Ce framework nous permet de pouvoir réaliser une base de données PostgreSQL sans avoir à réaliser nos requêtes via sa couche intermédiaire. Cette application est mise en ligne sur un serveur scaleway sur lequel se trouve également notre agent conversationnel nécessaire au fonctionnement du projet.

La conception a commencé par une étude de l'existant et de la concurrence ainsi qu'à l'étude des besoins de l'entreprise grâce à un cahier des charges et à un mockup.

Le développement s'est déroulé avec une approche proche des méthodes agiles et chaque nouvelle fonctionnalité a été implémentée au fur et à mesure sur le serveur. Il a commencé par la structure principale de l'application avec les différentes pages de gestion ainsi que par la création des modèles, essentiel pour la création et le maintien de la base de données. Il s'est poursuivi par des ajouts mineurs pour faciliter le confort de l'utilisateur et s'est fini par l'intégration du Chatbot en relation avec Slack.

Enfin, la question des tests et de la sécurité s'est posée pour pouvoir fournir un produit le plus robuste possible dans le but d'obtenir une version alpha présentable à de futurs clients. De plus, une documentation complète a été réalisée pour aider les futurs développeurs dans la continuité du projet ainsi que les utilisateurs non-initiés souhaitant se servir de l'application.

Mots-clés : Etude clinique, Web, Python, Rasa, Chatbot, Django

Abstract

For this internship, the main goal was to develop a web application to suit the domain of clinical trial. The clinical trial, as part of the medical domain contain a lot of restriction that we have to consider during the development.

To develop this application we chose to use Python and his framework name Django along with the PostgreSQL database language. Those language are those who fit our needs in term of quality of development and in term of scalability for the application.

The creation start with some research on what already exist for the clinical trial in the other firm and on what our firm really want to do with their future application. We determine all that stuff by making some study of other software and on the specifications made by the firm before we arrive.

After those research we start to develop the web application step by step starting with the main structure of the application. This main structure contain the pages related to a project who allow us to add, update or delete data and the pages who allow to see those data with some ease of use for a user like research field, filter or ordering.

Then we work on the way to add a entire IA to our application. This IA is made to discuss with the patient of clinical trial to replace the old way to get the patient symptoms which are still

based on a paper questionnaire today.

At the end our work ended with a alpha version with the function for the admin, the user and for the patient that the firm can show to future client but it still have a lot of point to improve or some new function to develop to really get a 1.0 version.

Keywords : Clinical trial, Web, Python, Rasa, Chatbot, Django