Une image contenant Police, Graphique, texte, logo

Description générée automatiquement

**Rapport de Projet Tutoré**

**ISIS – FIA5 – Promotion 2024**

**Année 2023 - 2024**

**L’Accessibilité Potentielle Localisée  
(APL)**

Table des matières

[I- Contexte et Analyse de l’existant 6](#_Toc169285753)

[1.1. Proposition envoyée aux étudiants 6](#_Toc169285754)

[1.2. Projet de recherche du laboratoire ICube 6](#_Toc169285755)

[2. Application existante hors projet 7](#_Toc169285756)

[3. Analyse du précédent livrable 9](#_Toc169285757)

[4. Méthode de Calcul 10](#_Toc169285758)

[5. Définition des objectifs 12](#_Toc169285759)

[II- Gestion de projet 13](#_Toc169285760)

[1. Choix technologique 13](#_Toc169285761)

[2. Outils de suivis 14](#_Toc169285762)

[3. Planning 15](#_Toc169285763)

[4. Coûts & Ressources 15](#_Toc169285764)

[5. Diagramme UML 16](#_Toc169285765)

[III- Réalisation des livrables 17](#_Toc169285766)

[1. Django 17](#_Toc169285767)

[2. Architecture de l’application 18](#_Toc169285768)

[3. Base de données 19](#_Toc169285769)

[4. Maquettes 21](#_Toc169285770)

[5. Développement de l’application 24](#_Toc169285771)

[5.1. Développement Backend 24](#_Toc169285772)

[5.2. Développement Frontend 27](#_Toc169285773)

[Annexe 1 : Proposition de projet tutoré 33](#_Toc169285774)

[Annexe 2 : Diagramme de classe de l’application 34](#_Toc169285775)

Table des Illustrations

[Figure 1 - Carte de densité médicale 8](#_Toc169285726)

[Figure 2 - Dossier Github & procédure de l'ancienne application 9](#_Toc169285727)

[Figure 3 - Correspondance entre le taux d’accessibilité et la distance qui sépare la commune de résidence du patient de la commune d’exercice du professionnel de santé 10](#_Toc169285728)

[Figure 4 - Tableau du facteur ETP 11](#_Toc169285729)

[Figure 5 - Modèle Kanban sur l'espace de travail Trello 14](#_Toc169285730)

[Figure 6 - Diagramme de Gantt 15](#_Toc169285731)

[Figure 7 - Diagramme de classe Client 16](#_Toc169285732)

[Figure 8 - Diagramme de cas d'utilisation du parcours de soin 16](#_Toc169285733)

[Figure 9 - Architecture MVC 17](#_Toc169285734)

[Figure 10 - Schéma d'architecture 18](#_Toc169285735)

[Figure 11 - Tableau présentant les sources de données principales du projet et leur réutilisation dans la base de données 19](#_Toc169285736)

[Figure 12 - Configuration de base de données dans Django 20](#_Toc169285737)

[Figure 13 - Exemple d’un objet de fixture de la table Profession 20](#_Toc169285738)

[Figure 14 - Maquette de la page d’accueil de la webapp 21](#_Toc169285739)

[Figure 15 - Maquette de la page de cartographie de la webapp 22](#_Toc169285740)

[Figure 16 - Maquette de la page de statistique de la webapp 22](#_Toc169285741)

[Figure 17 - Maquette de la page d’administration de la webapp 23](#_Toc169285742)

[Figure 18 - Class ProfessionnelDeSante représentant la table éponyme dans la base de données 24](#_Toc169285743)

[Figure 19 - Class Commune représentant la table éponyme dans la base de données 24](#_Toc169285744)

[Figure 20 - Visualisation de la table professionDeSante dans la base PostgreSQL 25](#_Toc169285745)

[Figure 21 - Visualisation de la contrainte entre la table ProfessionnelDeSante et la table Commune 25](#_Toc169285746)

[Figure 22 - Imports réalisés par le service calculAPL 26](#_Toc169285747)

[Figure 23 - Fonction calculAPL du service de calcul 26](#_Toc169285748)

[Figure 24 - Balise <head> du fichier base.html 27](#_Toc169285749)

[Figure 25 - Entête du fichier carte.html 27](#_Toc169285750)

[Figure 26 - Page d’accueil de la webApp 28](#_Toc169285751)

[Figure 27 - Carte de France présente sur la webapp avec le découpage par commune 28](#_Toc169285752)

Remerciements

En premier lieu, il nous semble naturel de présenter toutes les personnes qui nous ont aidés et appris au cours de ce projet tutoré, et qui ont aussi permis d’en faire un moment agréable et particulièrement formateur.

Tout d’abord, nous adressons nos remerciements à nos interlocuteurs de l’équipe CSIP d’Icube, Amadou COULIBALY et Virginie GOEPP-THIEBAUD qui nous ont proposé un sujet de projet tutoré galvanisant et parfaitement ancré dans les objectifs de la formation ISIS.

Nous tenons à remercier Imen MEGDICHE-BOUSARSAR et Elyes LAMINE, nos tuteurs école pour leur sollicitude, leur bienveillance ainsi que pour toute l’aide qu’ils nous ont apportée au cours du projet.

Introduction

Dans le paysage complexe de la santé publique française, l'accès aux soins demeure un défi majeur. Plus particulièrement dans les régions rurales où les déserts médicaux persistent. La méthodologie de l'Accessibilité Potentielle Localisée (APL) offre un outil précieux pour évaluer ces disparités en fournissant des données sur l'offre et la demande de soins à l'échelle locale. Cependant, l'application de cette méthodologie à tous les professionnels de santé libéraux, y compris les spécialistes, est encore incomplète, ce qui laisse un flou sur l'accessibilité aux soins spécialisés.

Afin de répondre à ce défi, le projet entrepris par l’équipe CSIP d’ICube en collaboration avec l'école ISIS vise à étendre l'utilisation de l'APL à l'ensemble des professionnels de santé libéraux en France. Son objectif est double :

* Identifier les zones géographiques nécessitant un renforcement de l'offre de soins et améliorer la prise en charge médicale.
* Intégrer des facteurs supplémentaires dans le calcul et l'analyse de l'APL.

En fournissant des données actualisées et détaillées sur l'accessibilité aux soins à travers le pays, ce projet permettra aux décideurs du secteur de la santé de mieux cibler leurs interventions et d'optimiser la localisation des services de santé. En outre, en mettant en place un suivi temporel de l'APL et des ressources de soins disponibles, il contribuera à évaluer l'impact des politiques de santé et à identifier les domaines nécessitant une attention particulière. Le projet s'inscrit dans une vision globale visant à améliorer la santé de la population en favorisant un accès plus équitable aux services médicaux, tout en restant fidèle aux principes de solidarité et d'universalité qui sous-tendent le système de santé français.

Dans un premier temps une remise en contexte et une analyse de la documentation, des données et du code à notre disposition en début de projet seront réalisé. Par la suite, nous ferons une présentation détaillée de la gestion de projet mise en place. Puis nous aborderons en détails la réalisation des livrables, de leur maquettage au produit final. Enfin, une rétrospective du déroulement de ce projet sera réalisée et fournira un regard critique sur notre travail.

# Contexte et Analyse de l’existant

Tout d’abord, il est important de mentionner que notre travail se base sur celui d’un autre groupe d'élèves. Ils ont réalisé une première analyse durant le dernier trimestre de 2023. Dans ce premier axe, nous allons poser et analyser l’existant afin de faire ressortir des objectifs et une feuille de route. Cette partie décrit nos recherches sur l’APL (son origine, ses objectifs et sa portée) mais aussi nos recherches sur les technologies utilisées par l’autre groupe d'élèves. Pour conclure, nous présenterons nos choix techniques et leurs implémentations.

1. Proposition de projet tuteuré

### Proposition envoyée aux étudiants

Le cahier des charges transmis à l’école ISIS pour le choix des sujets est assez succinct. La « Proposition de projet tuteuré » se trouve en Annexe 1. Voici ce que l’on peut en ressortir :

* Création d’une application web qui affiche une carte avec les contraintes suivantes :
  + Afficher les professionnels de santé, quelles que soient leurs spécialisations
  + Afficher les infrastructures de soins
  + Visualiser le calcul de l’APL
  + Permettre d’afficher les ressources et l’APL en fonction du temps (par année) et de la localisation (zoom sur les régions, départements et communes)
* L’application nécessite des connaissances en Python

Il est important de noter que cette proposition est la même que pour le groupe précédent.

### Projet de recherche du laboratoire ICube

Après nos premiers entretiens avec le client, nous avons appris que le projet APL faisait partie d’un projet plus global : le laboratoire ICube. Un projet porté par le CNRS, l'Université de Strasbourg, l’ENGEES et l’INSA de Strasbourg. Le laboratoire rassemble deux communautés scientifiques à l’interface entre le monde numérique et le monde physique, lui donnant ainsi une configuration unique. Fédéré par l'imagerie, ICube se concentre sur l'ingénierie pour la santé, l'environnement et le développement durable.

## Application existante hors projet

***Shiny***

Il existe aujourd’hui une application développée par la DREES assurant la cartographie de l’Accessibilité Potentielle Localisée (APL) en France. Cette application, nommée Shiny, offre un accès aux informations sur l’APL pour certaines catégories libérales de professionnels de santé, notamment les médecins généralistes, les sage-femmes, les infirmiers, les masseurs-kinésithérapeutes et les chirurgiens-dentistes. Elle permet aux utilisateurs d'observer l'évolution de l'indicateur de l'année 2015 à 2022, offrant ainsi une vue temporelle des disparités en matière d'accessibilité aux soins.

Shiny est intéressante sur sa manière de traiter et afficher les données, car elle permet de filtrer en fonction de l’âge des praticiens. Cependant, le plus gros point faible de cette application est la plage de données sur laquelle s’appuie le calcul de l’APL. En effet, Shiny base son calcul d’APL sur certaines professions libérales, mais ne prend pas en compte les structures de santé et les spécialités. On peut lire sur leur site que *« Les Ordres jouent le rôle de guichet principal pour les professionnels inscrits au tableau, le Service de Santé des Armées pour ceux qui ont le statut de militaires et l’État pour les fonctionnaires. Les données du RPPS sont élaborées à partir d’une confrontation des données des Ordres, de l’Assurance maladie (la CNAM) et du centre national de gestion (CNG) »* [[1]](#footnote-2). Les données provenant du répertoire RPPS, géré par l’ANS, possèdent une forte fiabilité en contrepartie d’un nombre réduit de profession.

Le deuxième répertoire mentionné, le répertoire ADELI recense les diplômes des professionnels de santé et est géré par les ARS, fut également utilisé sur les premiers calculs d’APL, mais il fut rapidement abandonné, car « *malgré l’obligation légale d’inscription, ce répertoire présente certaines fragilités. Tout d’abord, les professionnels de santé ne sont pas réellement incités à se désinscrire […] lorsqu’ils cessent leur activité temporairement ou définitivement, ce qui peut conduire à surestimer le nombre de professionnels âgés en exercice. À l’inverse, les nouveaux professionnels tardent parfois à faire enregistrer leur diplôme au répertoire […], ce qui tend à sous-estimer la part des plus jeunes. »1*

Comme l’a mentionné le précédent groupe de travail, Shiny a un fonctionnement inspirant, qui devrait cependant avoir une plage de données plus grande, afin d’élargir au spécialiste et aux infrastructures de soin, sans perdre en fiabilité. Il est important de conserver la possibilité de zoomer jusqu’à la commune pour les valeurs d’APL.

D’un point de vue plus critique, on pourrait reprocher à l'application d’être lente au chargement des données et de ne pas proposer un tri par tranche d’âge pour certaines professions.

***Urbi-Explore***

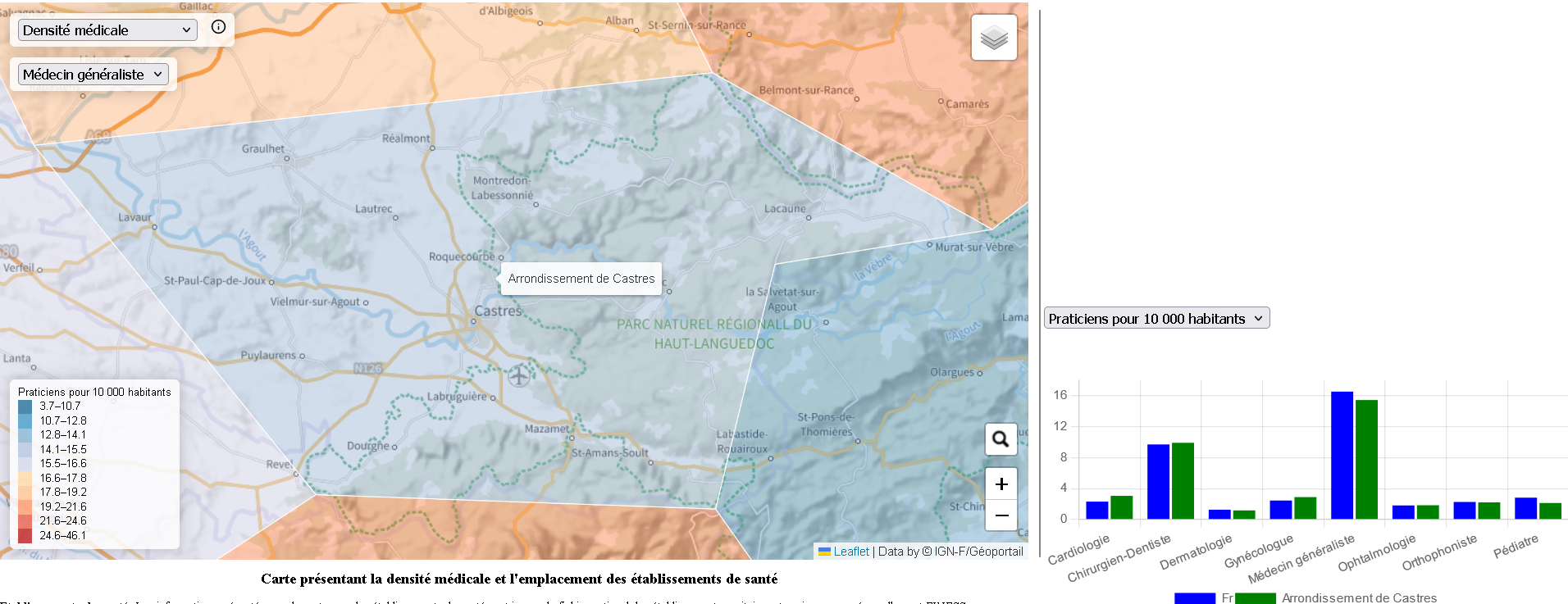
Urbi-Explore est une application web de cartographie permettant de visualiser divers points d’intérêt en France, dont les établissements et professionnels de santé. Développée par la société Domubi, cette application utilise les données officielles du FINESS et de l’annuaire de santé de l’assurance maladie. Elle permet un filtrage simple, mais efficace pour les établissements (par type) et traite les professionnels de santé par rapport à la densité de praticien pour 10 000 habitants (à l’échelle d’un arrondissement).

Figure - Carte de densité médicale

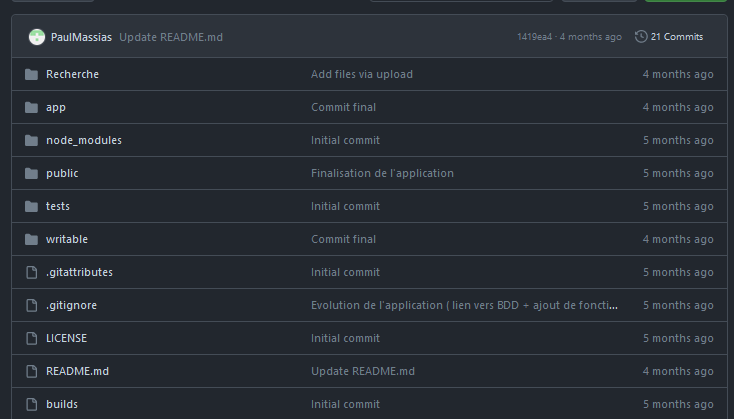
Urbi-Explore est moins précise sur la position des praticiens que Shiny et ne propose pas une vision claire de l’APL. Elle reste néanmoins plus complète sur le jeu de données totales en couvrant plus de professions et en permettant l’affichage des établissements de santé.

## Analyse du précédent livrable

Le groupe précédent a rendu deux livrables, un mémoire de projet comme celui-ci, et leur projet sur un dépôt GitHub, nous permettant d’accéder au code source de leur webapp et aux divers documents partagés dans le dossier.

Dans ce dépôt, on retrouve également une « procédure » nous expliquant comment déployer la solution et donnant des indications sur le fonctionnement de l’application. Le projet utilise le Framework PHP « CodeIgniter 4 ».

Figure - Dossier Github & procédure de l'ancienne application



Voici un résumé de ce qui se trouve dans ce dépôt :

* Les données de l’application, provenant d’un export de Shiny, stockées dans un fichier csv pour les informations lié à l’APL, et un fichier json pour les données liées à la localisation des villes.
* Un fichier de traitement en PHP servant à lire les fichiers de données et les charger.
* Un fichier front-end en PHP pour le visuel de l’application, il traite aussi les données chargées précédemment pour les afficher sur une carte.
* Un dossier de recherche contenant des fichiers Jupiter expliquant leur méthode de calcul d’APL, le mémoire du groupe, ainsi qu’un fichier servant de Webographie.

L’inconvénient de cette solution est l’absence d’une base de données qui rend le site statique. Actuellement, aucune rétention ou mise à jour de donnée n’est possible, ce qui rend le traitement des fichiers de données long et ralentit l’affichage de la page. Ceci est un point connu par l’ancien groupe, et un point d’amélioration important pour le client.

## Méthode de Calcul

Une image contenant texte, Police, blanc, ligne

Description générée automatiquementL'Accessibilité Potentielle Localisée (APL) est un indicateur développé pour mesurer l'accessibilité aux soins de santé dans différentes régions géographiques. Le calcul de l'APL prend en compte à la fois l'offre de soins (disponibilité des professionnels de santé) et la demande de soins (population locale), ainsi que la distance entre les patients et les professionnels de santé. La formule simplifiée de l'APL est la suivante :

**[[2]](#footnote-3)**

Les paramètres de la formule peuvent être décrits ainsi :

* ***Zi*** est la zone de proximité de la commune ***i***
* ***Pj*** est le nombre de professionnels de santé dans la commune ***j***
* ***W (dij)*** est un facteur de pondération fonction de la distance ***dij*** entre les communes ***i*** et ***j*** (Par exemple, ***W(dij)*** pourrait être **1/(1 + *dij*)** pour diminuer l'importance des professionnels plus éloignés)
* ***Di*** est la demande de soins de la commune ***i***, généralement proportionnelle à la population de ***i***.

Le **facteur de pondération** qui est fonction de la distance entre deux communes peut être exprimé soit en unités de longueur (kilomètres), soit en unités de temps (minutes). Cette distinction exerce une forte influence sur la valeur finale de l'APL surtout si les caractéristiques géographiques et l'accessibilité générale de la commune sont difficiles (zone de montagne, accès aux grands axes routiers, etc.). C’est pourquoi il est souvent préférable d'utiliser une unité de temps, car elle reflète plus précisément la facilité d'accès aux soins. La DRESS a défini un tableau de pondération par rapport au temps de trajet (Figure 3).

Une image contenant texte, nombre, capture d’écran, Police

Description générée automatiquement

Figure - Correspondance entre le taux d’accessibilité et la distance qui sépare la commune de résidence du patient de la commune d’exercice du professionnel de santé

Le **nombre de professionnels de santé** est évalué en fonction de l'équivalent temps plein (ETP), qui reflète l'activité d'un professionnel de santé basée sur son nombre d'heures de travail disponibles. Le ratio qui définit si un médecin vaut 1 ETP ou moins est déterminé par les déciles de distribution.

Les déciles de distribution sont des mesures statistiques qui divisent une série de données en dix parties égales. Chaque décile représente 10 % de la distribution totale. Par exemple, le premier décile comprend les 10 % de professionnels ayant le niveau d'activité le plus bas.

|  |  |
| --- | --- |
| Déciles de distribution de l'activité | Valeur ETP |
| Moins de 5 % | 0 |
| 5 % à 10 % | 0,2 |
| 10 % à 25 % | 0,5 |
| 25 % à 50 % | 0,7 |
| Plus de 50 % | 1 |

Figure - Tableau du facteur ETP

La **demande de soins** varie en fonction de l'âge médian de la population de la commune. Cette valeur de "consommation" est uniformisée à l'échelle nationale pour chaque tranche d'âge afin d'assurer une cohérence dans l'évaluation de la demande de soins.

L’APL peut être exprimée en différentes unités selon les paramètres pris en compte. Pour les médecins généralistes sur Shiny, l’APL est exprimée en "nombre de consultations accessibles par an et par habitant". Plutôt que de se baser sur l'ETP, une capacité de consultation est affectée.

* **ETP accessibles** : Mesure l'accessibilité en termes de disponibilité de professionnels de santé à temps plein, indépendamment de leur capacité de consultation (utilisé pour toutes les autres professions libérales).
* **Consultations accessibles par an et par habitant** : Mesure l'accessibilité en termes de capacité réelle de fournir des soins (consultations) à chaque habitant, en prenant en compte la capacité de consultation des professionnels de santé

## Définition des objectifs

Grâce à ces premières analyses, il est plus facile pour nous de définir des objectifs pour construire et gérer notre projet. Ces objectifs sont principalement fonctionnels et ils permettront par la suite de définir un cahier des charges complet qui sera présenté dans la partie suivante.

* L’application devra comprendre une carte, permettant l’affichage de l’APL à différents niveaux géographique (national, régional, départemental et communal)
* Cette carte devra être filtrable en fonction de divers critères :
  + L’année
  + Les professionnels de santé
    - Âge
    - Profession
    - Spécialisation (si disponible)
* L’application doit posséder une base de données, afin de rendre son utilisation plus rapide, simple et agréable pour les utilisateurs.
* L’application devra permettre l’importation de nouvelles données annuelles
* L’application devra avoir une unité d’APL adaptée en fonction du professionnel de santé affiché.
* Le calcul de l’APL devra être intégré à l’application, pour s’actualiser chaque année au moment de l’importation des données.

En plus de ces objectifs, le client nous a demandé d’intégrer plusieurs fonctionnalités supplémentaires :

* Ajout d'une page d’administration, accessible avec des identifiants spécifiques afin de pouvoir ajouter, modifier et supprimer des données.
* Permettre de visualiser les infrastructures de santé sur la carte de l’APL.
* L’application devra permettre de répertorier et d’afficher le nombre d’équipements disponible dans les infrastructures.
* L’application et la base de données devront être facilement modulables, car elles seront amenées à évoluer lors de futur projet de recherche.

Après nos recherches, nous avions également des propositions de nouvelles fonctionnalités/tâches :

* Lancer une demande d’accès aux données simplifié au SNIIRAM. Cette procédure est annoncée sur leur site comme courte, avec un délai de traitement d’un mois environ, contre 6 mois pour accès complet aux données.

Cet accès permettrait la construction d’un modèle avec des données officielles et fiables, rendant plus simple l’intégration de nouvelles données, le traitement de celle-ci, ainsi que leur maintenabilité.

* Ajout d’une page « Statistique », qui permettrait de visualiser l’évolution de l’APL sur plusieurs années.

Enfin, il a été décidé en accord avec le client de faire une refonte de l’application, afin de correspondre à toutes les spécifications évoquées, et plus particulièrement pour rendre l’application robuste et modulable sur le long terme. Cette décision a été motivée par de nombreux aspects, mais principalement pour des raisons de choix technologique et de maintenabilité de l’application. Tous les aspects de cette refonte vont être détaillés dans les parties suivantes, en se basant sur nos recherches et des outils de gestion de projet.

# Gestion de projet

Dans cette partie, nous aborderons tous les aspects pré-développement liés à la gestion de projet : les choix technologiques ainsi que la contextualisation de l’environnement technique dans lequel évolue le projet.

## Choix technologique

Le groupe précédent avait fait le choix d’utiliser le Framework PHP CodeIgniter pour développer leur application web. Bien que ce Framework ait de nombreux avantages comme sa courbe d’apprentissage douce et rapide, sa légèreté et sa modularité, il est aussi limité par son absence d’ORM et de fonctionnalités de sécurité ainsi que par la syntaxe et l’architecture de code vieillissante qu’il propose.

Nous avons fait le choix de travailler avec un Framework orienté python pour plusieurs points:

* L’utilisation de ce langage était une exigence client.
* Les calculs de l’APL avaient été réalisés par l’autre groupe en python à travers des fichiers Jupyter.
* De nombreuses bibliothèques de traitement de données performantes existent en python (numpy, pandas, etc)

Le Framework que nous avons choisi est Django. C’est un outil robuste offrant de nombreuses fonctionnalités intégrées, dont un module de sécurité, de gestion de compte et d'authentification, ainsi qu’un ORM puissant.

Les données utilisées pour générer une carte sur l’APL sont grandes et nécessitent beaucoup de puissance de calcul. C’est pourquoi nous avons fait le choix de stocker ces données ainsi que les résultats des calculs dans une base de données PostgreSQL. PostgreSQL offre de nombreux avantages : nombreuses fonctionnalités, implémentation des propriétés ACID, compatibilité avec de nombreuses plateformes, Framework, hébergeur de données. Le tandem Django - PG est souvent celui choisi par la communauté de Django et il est aussi très bien supporté par les deux entités.

Pour la partie frontend de notre application, nous avons choisi Bootstrap pour sa modularité et sa facilité d’accès. Ce Framework possède une très bonne compatibilité avec Django. En effet, Bootstrap possède une version dédiée aux applications utilisant Django « django-bootstrap-v5 ».

L’affichage de la carte de visualisation des APL est géré à l’aide de la bibliothèque Leaflet. Cette bibliothèque était déjà utilisée par le groupe précédent et est très performante. Les diagrammes de la page de statistique utilisent la bibliothèque chart.js, car elle est performante et facile d’utilisation.

## Outils de suivis

Afin de faciliter nos échanges avec le client, nous avons mis en place un suivi régulier par le biais de réunions et l’utilisation de divers outils. Les trois outils qui vont être présentés nous ont permis d’être transparent et de donner un visuel sur l’état d’avancement du projet.

***Groupe Teams***

Dès le début du projet, le client nous a créé un groupe de travail sur l’outil de messagerie Teams, nous permettant d’échanger, prévoir des réunions et avoir un espace de partage de document au sein d’une même application. C’est sur cet espace que nous avons pu partager les horaires prévus pour le projet au cours de l’année, au fur et à mesure que nos emplois du temps étaient saisis.

***Dossier GitHub***

Également au début du projet, nous avons invité M. Coulibaly et Mme. Goepp à rejoindre un répertoire GitHub, contenant l’entièreté du code de notre application. Tout le long du développement, nous avons régulièrement ajouter nos modifications du code sur ce répertoire et mis à jour une procédure permettant l’installation et le test de l’application. Ce dépôt fait partie des livrables que nous devons remettre à la fin du projet.

***Espace de travail Trello***

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquementAprès la définition des aspects fonctionnels et techniques, nous avons mis à disposition un espace de travail collaboratif sur le site Trello. Le modèle Kanban du projet (en figure 5) reprend ses aspects fonctionnels majeurs. Il définit les tâches en indiquant les sous-tâches associées (techniques ou fonctionnelles), la/les personne(s) responsables de cette tâche ainsi que son état d’avancement. Pour favoriser la compréhension, certaines tâches peuvent avoir des pièces jointes associés. Dans notre cas, les tâches liées au visuel du site étaient associées à une image de la maquette du site.

Figure - Modèle Kanban sur l'espace de travail Trello

## Planning

Au total, 90 heures dédiées spécifiquement à ce projet sont prévues dans notre emploi du temps. En complément de ces heures, nous avons également profité des créneaux de FOAD en entreprise lorsque nos charges de travail respectives le permettaient, ce qui représente environ 20 heures supplémentaires. Ces heures additionnelles ont été principalement réparties sur la première période en entreprise, permettant ainsi une meilleure répartition des tâches et une avance significative sur notre apprentissage des technologies choisies.

Le diagramme de Gantt ci-dessous présente le déroulement des différentes phases du projet. Les jalons en rose représentent les dates auxquelles ont eu lieu les réunions avec le client.

Une image contenant texte, nombre, ligne, Police

Description générée automatiquement

Figure - Diagramme de Gantt

Le projet a officiellement commencé le 11 janvier 2024. Nous avons organisé des réunions régulières pour discuter des progrès et ajuster les objectifs si nécessaire. Ces réunions sont essentielles pour assurer une bonne coordination et résoudre rapidement les éventuels problèmes.

Les étapes importantes incluent la définition des objectifs, l'étude de faisabilité et l'optimisation de l'environnement du projet, principalement en janvier et février. En mars, nous avons commencé à concevoir l'architecture du projet. La phase de développement, qui s'étend d'avril à juin, est dédiée à la programmation et aux tests pour s'assurer que toutes les fonctionnalités soient implémentées.

Le mémoire du projet doit être rendu le 15 juin 2024, suivi de la soutenance le 20 juin 2024. Ces dates marquent la fin du projet, où nous présenterons notre travail, les défis rencontrés et les solutions apportées.

## Coûts & Ressources

Aucuns frais ne sont à prévoir pour notre contribution au projet, qu’il s’agisse de dépenses matérielles (comme les serveurs) ou de coûts liés aux logiciels. En outre, aucune ressource supplémentaire ne sera fournie pour la réalisation de ce projet. La seule exception est l'accès au groupe Teams, comme mentionné dans la partie précédente, qui est mis à disposition par l’INSA Strasbourg. Cette solution collaborative facilitera la communication et le partage d’informations entre les membres de l’équipe, mais ne représente pas une ressource matérielle ou logicielle directe.

## Diagramme UML

Une image contenant texte, diagramme, Plan, ligne

Description générée automatiquementAu début du projet, le client nous a fourni un diagramme UML pour mieux comprendre sa vision et la portée de l’étude. Il sert aussi de socle pour construire un modèle de base de données soutenant le cahier des charges de notre webApp. Le diagramme de classe partiel en figure 7, représente le système de santé dans son ensemble :

Figure - Diagramme de classe Client

Les classes sont triées par couleur pour définir trois grandes catégories : les acteurs, les infrastructures et la gouvernance de santé. Ces familles ont plusieurs ramifications apportant de la précision et/ou des exemples d’entité : l’hôpital et la pharmacie sont des infrastructures, le médecin et le kinésithérapeute sont des professionnels de santé.

Le cahier des charges de notre projet restreint le nombre de donnes/notions nécessaires au bon fonctionnement du site web. C’est pourquoi la gouvernance qui n’est pas prise en compte dans l’offre de soin et le patient qui n’est pas utile au calcul de l’APL (à l’échelle de l’individu) ne seront pas intégré dans notre modèle de donnée. Le patient sera pris en compte à l’échelle d’un territoire, étant inclus dans la population de sa commune de résidence. En-dehors de ces deux points, on retrouve les classes importantes que nous voulons cartographier : les professionnels de santé (APL), les infrastructures et les équipements.

Une image contenant texte, diagramme, croquis, dessin

Description générée automatiquementLe deuxième diagramme présente les cas d’utilisation d’un parcours de soins. Notre application intervient donc au niveau de l’évaluation et de la gestion d’un territoire par l’ARS, comme un outil d’aide à la décision.

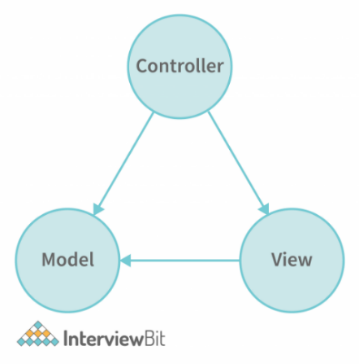
Figure - Diagramme de cas d'utilisation du parcours de soin

# Réalisation des livrables

Cette partie présente les livrables fournis ainsi que la manière dont ils ont été produits. Tout d’abords, nous détaillerons les caractéristiques du framework Django, puis l’architecture de notre projet, la base de données créées, les maquettes du produit, le développement de l’application (frontend et backend) et enfin le produit final.

## Django

Django est un framework web open-source orienté python. La dernière version accessible au public est la 5.0.6, le projet est stable, fonctionnel et facile à prendre en main. Il propose plusieurs modules déjà intégrés en adoptant la philosophie « Batteries-included » tels que la manipulation de bases de données, les modèles HTML, le routage d’URL et la gestion de session et des sécurités.

Il utilise l’architecture MVC ou Model, View, Controller qui peut être schématisé de la manière suivante :

* Un modèle contient les données à afficher
* Une vue contient la présentation de l’interface graphique
* Un contrôleur contient la logique concernant les actions effectuées par l’utilisateur.

Figure - Architecture MVC

***Model***

La partie modèle s’occupe de tout ce qui concerne la donnée : description, stockage, transformation, import, transfert, etc. Plus concrètement, pour notre application, cette partie s’occupe de définir les tables de notre base et leurs liens.

***View***

La partie vue permet d’afficher la donnée à l’utilisateur. Elle sert aussi d’interface d’envoi lorsque l’utilisateur doit remplir un formulaire ou envoyer un fichier. C’est ce module qui sert d’interface avec des bibliothèques ou Framework plus orienté front comme Bootstrap ou JQuery.

***Controller***

La partie Controller s’occupe de l’interfaçage, il définit quelle vue afficher sur quelle url en envoyant quelles données. La plupart des applications utilisent un Controller par bloc fonctionnel ce qui rend le code plus lisible.

## Architecture de l’application

Une image contenant texte, capture d’écran, diagramme, ligne

Description générée automatiquementLe schéma d’architecture de l’application permet de comprendre les interactions entre les différents modules du projet et Django.

Figure - Schéma d'architecture

L'utilisateur accède à l'application via un navigateur web. La communication entre l'utilisateur et le serveur Django se fait via des requêtes HTTP sur le localhost au port 8000.

En voulant accéder ou changer de page, le contrôleur Django va se servir des vues à sa disposition pour gérer le contenu dynamiquement coté client.

Les vues vont permettre de traiter les requêtes utilisateur transmises par le contrôleur, en récupérant les données auprès des modèles (en passant ou non par des algorithmes de calcul) puis en les renvoyant aux templates Django.

Les modèles vont permettre d’effectuer des opérations de lecture, écriture, mise à jour et suppression sur la base de données PostgreSQL. Elle est utilisée comme système de gestion de base de données relationnelle. La communication entre Django et PostgreSQL est paramétrée dans les fichiers de configurations du projet. Cela va permettre au model d’interagir grâce à des requêtes SQL sur le localhost au port 5432.

Les templates utilisent les données fournies par les vues et les contrôleurs pour générer dynamiquement le contenu des pages web. Par exemple, la carte Leaflet est intégrée dans un template Django et affiche les données récupérées par les vues.

## Base de données

La base de données utilisées dans le projet est inspirée à la fois du diagramme UML présenté en partie 2.5 et des jeux de données public trouvés au cours de nos recherches. L’objectif étant de créer un modèle robuste et modulable, fidèle aux données SNIIRAM.

***Recherche***

Les jeux de données publiques gouvernementales permettent d’obtenir la majorité des informations nécessaires au fonctionnement de l’application. Leurs modèles et architectures nous ont orientés sur la construction de notre propre modèle. L’ensemble des jeux utilisés, leurs sources et quelles tables ils ont permis de construire se trouve en figure 11. D’autres jeux de données nous ont permis d’avoir une vision plus globale sur la problématique des APL, mais ils n’ont pas été utilisés lors de la construction de base de données.

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom** | **Source** | **Table de notre base** | **Année** |
| Communes de France – Base des codes postaux | Data.gouv.fr | - Commune  - Département  - Région | Mars 2020 |
| Annuaire Santé de la CNAM | Data.gouv.fr | - Infrastructure  - Professionnels de santé  - Profession  - Specialite | Juin 2024 |
| Script python réalisé par nos soins accessible en Annexe X. | - Bibliothèque python Nominatim  - API OSRM | - Pondist | Juin 2024 |

Figure - Tableau présentant les sources de données principales du projet et leur réutilisation dans la base de données

***Schéma final***

Le schéma final et complet de notre base de données se trouve en Annexe 2. Il ne liste pas tous les champs des tables du modèle, les champs visibles permettent de comprendre les relations et fonctions de chaque table.

Tout d’abord, nous avons intégré les éléments nécessaires au calcul de l’APL. L’objet ProfessionnelDeSante comprend les informations du praticien ainsi que sa charge de travail hebdomadaire pour calculer l’ETP. Il est relié à deux sous-tables définissant la profession et la spécialité du praticien, cette vision correspond aux réalités organisationnelles et assure plus de flexibilité. Les professionnels sont associés à une commune et/ou à une structure selon leur domaine d’exercice.

L’objet Commune récupère les valeurs nécessaires au calcul de l’APL : notamment la localisation géographique et la population divisée par tranche d’âge de la commune, ce qui aide à estimer la demande de soins. Pour tenir compte de l’offre de soins des communes voisines, nous avons créé une table indiquant le temps de trajet entre deux communes (PonDist). Avec ces tables, nous avons toutes les variables nécessaires pour calculer et afficher l’APL sur notre application.

L’objet Infrastructure est relié à sa commune et possède donc des coordonnées géographiques permettant l’affichage sur une carte. Chaque équipement est lié à une infrastructure et peut s’afficher en complément d’information de l’infrastructure de santé.

Enfin, les objets Patient, Acte, Pathologie et Soin sont inclus dans le schéma bien qu’ils ne soient pas encore utilisés. Cette inclusion répond à une demande du client pour anticiper l’évolution future de l’application en intégrant le patient et ses relations avec les autres classes.

***Implémentation***

Pour gérer la partie base de données Django propose une connexion simplifiée avec PostgreSQL, MariaDB, MySQL, etc. La configuration de la base se fait dans le fichier *settings.py* de l’application :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure - Configuration de base de données dans Django

On peut voir en figure 12 que la configuration de la connexion à une base de données est simplifiée. Il suffit de définir le type de la base (postgresql), son nom (apldb), ses paramètres de connexion (host, port) et son utilisateur. Une fois la connexion établie, si les modèles de données sont déjà écrits, on peut lancer une commande administrateur : python manage.py makemigrations pour créer un fichier de migration contenant toutes les informations de notre base. La commande : python manage.py migrate joue cette migration et met à jour la base PostgreSQL.

***Fixture***

L’application web a pour objectif de fonctionner à l’aide des données du SNIIRAM. Nous n’avons pas pu les utiliser au cours du développement de l’application, car la procédure de demande est longue et complexe. Pour pallier ce problème nous avons fait le choix d’utiliser les données présentées dans la partie *Recherche* en comblant les manques par des données réalistes mais non-réelles. Les bases que nous avons trouvées proposent des données sur toute la France, mais pour des raisons de rapidité de traitement et de simplicité nous avons focalisé nos imports sur le département du Tarn. L’annexe 2 montre les données réelles et celles générées. Il est aussi important de préciser que ces données ne sont pas toutes mises à jour régulièrement et qu’elles ne sont pas forcément représentative d’une réalité.

Les données de test, ou *fixtures*, sont gérées habilement par Django. Une commande administrateur permet de charger des fichiers en json rapidement dans la base. Les fichiers json doivent présenter les occurrences en indiquant le modèle, l’index et les champs de l’objet. Un exemple d’une fixture de la table profession est présenté en figure 13.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure - Exemple d’un objet de fixture de la table Profession

## Maquettes

Dans cette section, nous présentons les maquettes détaillées du site web, elles illustrent la structure visuelle et fonctionnelle proposée pour notre projet. Ces prototypes bas niveau constituent une étape cruciale dans la validation de l'interface utilisateur et l'expérience de navigation avant le développement final.

***Page d’accueil :***

La figure 14 présente la maquette de la page d'accueil, elle a une disposition claire et intuitive. Au sommet de la page, on retrouve un espace dédié au titre "ACCUEIL" avec, à droite une icône cliquable pour accéder au compte de l’utilisateur. Sur le côté gauche de l'écran, un menu vertical répertorie les quatre modules du site : "Accueil", "Carte", "Statistiques" et "Mon compte". Chaque option est représentée par une icône et un texte, facilitant ainsi l'identification rapide des pages.

Les couleurs sobres et les icônes contribuent à créer une interface épurée et professionnelle. Le centre de la page est dédié au contenu principal, plusieurs sections sont réparties pour accueillir des zones de texte ou des images. Cette page d’accueil sert à présenter le site et le projet.

***Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement***

Figure - Maquette de la page d’accueil de la webapp

***Page de Cartographie :***

Similaire à la page d’accueil dans la structure, la page de cartographie constitue le module central de notre application. On peut voir en figure 15, une barre de filtres située en haut qui permet de trier les données affichées selon différents critères tels que l'âge, la profession, l'infrastructure, la spécialité, l'année et les équipements. Chaque critère est représenté par un menu déroulant à choix simple. Une carte de France occupe la majeure partie de l'écran avec à sa droite, une section de légende expliquant les différentes catégories ou éléments représentés.

Cette maquette de la page de cartographie se concentre sur l’expérience utilisateur (UX). La page est fonctionnelle, esthétique, et est conçue pour permettre une exploration facile et intuitive.

***Une image contenant texte, capture d’écran, carte, logiciel

Description générée automatiquement***

Figure - Maquette de la page de cartographie de la webapp

***Page de Statistique :***

La page de statistique en figure 16 ressemble beaucoup à celle de cartographie, car elle expose la même barre de filtre en haut de page. Elle a pour objectif d’afficher plusieurs graphiques comparant les données sur différents repaires spatio-temporelles.

***Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement***

Figure - Maquette de la page de statistique de la webapp

***Page d’administration :***

Enfin, la page d’administration en figure 17 sert à gérer les données (import, suppression, modification, etc) et les potentiels utilisateurs (création, suppression, modification, etc). En haut de la zone centrale, une section affiche les informations du compte connecté : adresse électronique, téléphone et autorisations. À droite de cette zone, deux boutons sont présents permettant respectivement la suppression ou la mise à jour des informations de compte.

Situé en dessous, on retrouve une zone dédiée à la gestion des données. Elle se compose d’un formulaire d’ajout de données qui peut prendre en charge un fichier csv ou json au bon format et l’intégrer dans la base actuelle. Ce système est un complément à l’interface pgAdmin plus riche en fonctionnalité, mais moins facile d’utilisation.

***Une image contenant texte, logiciel, Icône d’ordinateur, Page web

Description générée automatiquement***

Figure - Maquette de la page d’administration de la webapp

## Développement de l’application

Cette section est dédiée à la description du processus de développement de l'application APL en abordant à la fois les aspects backend et frontend. Pour chaque partie, une description du code, des difficultés rencontrées et des solutions apportées seront présentées.

### Développement Backend

Le développement backend de l’application se concentre sur la gestion des données et l’intégration de la base PostgreSQL ainsi que sur la mise en place de l’ORM (Object Relational Mapping) Django et son utilisation dans les communications back et front.

***Gestion de données avec PostgreSQL***

Cette partie est un complément à la partie 3.3 Base de données. Les modèles de données sont définis dans des fichiers python stockés dans un dossier référencé par la suite dans le fichier settings.py pour que Django puisse y accéder facilement. Un fichier de modèle définit une ou plusieurs classes représentant les tables de la base de données. Par exemple, pour représenter les professionnels de santé et les communes, nous avons créé les classes suivantes :

* La classe ProfessionnelDeSanté visible en figure 18 contient les champs nom, prenom, spécialité, etc. Chaque champ correspond à une colonne de la table des professionnels dans la base de données. La relation avec la table des communes est définie par une clé étrangère, commune, qui lie chaque professionnel à une commune spécifique.
* La classe Commune contient les champs nom, population, localisation, etc représentant respectivement le nom de la commune, sa population, et ses coordonnées géographiques.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure - Class ProfessionnelDeSante représentant la table éponyme dans la base de données

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure - Class Commune représentant la table éponyme dans la base de données

Comme vu dans la partie 3.3 : Base de données, ces modèles peuvent être ensuite utilisés pour créer une migration et importer ces tables dans la base PostgreSQL. La table *ProfessionnelDeSante* est visible en figure 20. En figure 21, on peut voir entouré d’orange la table *ProfessionnelDeSante* qui est associée à plusieurs contraintes, dont celle pointée avec une flèche verte sur la colonne commune\_id. Le cadre bleu foncé sur la droite indique bien que cette colonne est reliée à la table Commune.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Figure - Visualisation de la table professionDeSante dans la base PostgreSQL

*Une image contenant texte, capture d’écran, logiciel, Page web

Description générée automatiquement*

Figure - Visualisation de la contrainte entre la table ProfessionnelDeSante et la table Commune

***Mise en place de l’ORM***

Un ORM est un type de programme information qui se place en interface entre un programme applicatif et une base de données relationnelle pour simuler une base de données orientée objet. Ce genre de technologie est très utile pour les développeurs, car ils n’ont plus à rédiger à la main des requêtes SQL pour chaque interaction avec la base. Dans le cas de Django, l’ORM Django est nativement intégré et permet la création, la récupération, la mise à jour à la suppression d’objet de manière rapide et intuitive. L’application APL a besoin des objets de la base dans la partie Administrateur pour récupérer et/ou modifier les données de l’utilisateur et dans la partie de cartographie pour gérer le calcul de l’APL.

Par la suite, nous allons détailler le service de calcul de l’APL pour illustrer l’utilisation de l’ORM Django dans notre projet. Comme vu dans la partie 3.2 Architecture de l’application, le service calcul de l’APL communique avec la base de données (via l’ORM) et avec la vue. Comme on peut le voir en figure 22, le service importe les données de la base nécessaire au calcul.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure - Imports réalisés par le service calculAPL

La fonction calculAPL présente en figure 23 montre comment les objets de la table PonDist sont récupérés puis sont filtré sur la commune passé en paramètre pour récupérer ses communes voisines. En ligne 4, un filtrage est réalisé sur la table professionnelDeSante pour filtrer les professionnels en fonction de la commune et du code de profession passé en paramètre.

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Figure - Fonction calculAPL du service de calcul

***Difficultés rencontrées***

La partie Backend du projet nous a pris beaucoup de temps, car nous souhaitions avoir un socle stable et modulable sur lequel construire le frontend. De plus, pour coller au mieux au modèle de donnée du SNIIRAM nous avons réalisé beaucoup de recherche sur les bases de données des organismes publics. Une première base a été réalisée en début de projet puis nos recherches et les remarques de nos clients ont permis d’arriver à une deuxième proposition. Cette proposition a été retravaillée au moment de l’implémentation des calculs de l’APL pour mieux convenir à nos besoins.

### Développement Frontend

En parallèle, le développement frontend a porté sur la création d'interfaces utilisateur basées sur les maquettes vue en partie 3.4. Le Framework Django propose un moteur de template natif pour gérer le visuel d’une application. Un moteur de template est un outil utilisé par un Framework web pour séparer la logique de l’application de la présentation visuelle. Contrairement à un fichier HTML classique, un fichier de template peut contenir des variables fournies par le backend qui sont chargées dynamiquement lors du rendu de la page. Dans le cas de la page de cartographie, c’est grâce à ces variables que les filtres sont pris en compte.

L’utilisation d’un moteur de template permet aussi de construire des pages web de manière modulaire et organisée. On écrit un fichier de base qui contient les imports ainsi que la définition de la structure générale des pages. On peut voir en figure 24 la balise <head> du fichier base.html de l’application qui réalise les imports nécessaires au bon fonctionnement de l’application et définit un block titre ainsi qu’un bloc de styles.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure - Balise <head> du fichier base.html

En figure 25 on trouve l’entête du fichier carte.html qui hérite du fichier de base. En ligne 3 le template redéfinit le block title pour y ajouter le bon titre et en ligne 4, on utilise le bloc style pour ajouter des fichiers JS et CSS spécifiques à cette page.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure - Entête du fichier carte.html

L’utilisation de block peut aussi être utile pour découper son code en composant en créant un fichier menu qui est appelé directement dans le fichier base.html pour qu’il s’affiche sur toutes les pages au même endroit.

Comme vu précédemment dans le rapport, nous utilisons en plus le Framework Bootstrap pour gérer le rendu visuel des pages et assurer une navigation fluide et intuitive. On peut voir en figure 26 la page d’accueil de l’application sur laquelle on retrouve sur la gauche le menu, sur la droite le corps de la page et dans l’onglet le titre de la page « Accueil ».

Si on compare le rendu visuel de la page à la maquette réalisée en amont, on se rend compte que plusieurs choses ont été supprimées ou déplacées. En effet, le module en haut de page a été supprimé pour laisser place à des blocs flottants ce qui apporte plus de dynamisme à la page. L’icône mon compte a été supprimée, car elle était redondante avec le lien déjà présent dans le menu. Les blocs d’image ont été remplacés par une image de fond et les blocs de texte ont été fusionnés avec le titre de la page.

Une image contenant texte, carte, capture d’écran, logiciel

Description générée automatiquement

Figure - Page d’accueil de la webApp

Une des fonctionnalités centrales de l'application est la carte, implémentée avec la bibliothèque Leaflet. Cette carte permet de visualiser les différentes régions, départements et communes de France. Elle peut être manipulée avec la souris pour explorer d'autres zones en utilisant le zoom ou la souris. La carte propose une vision des communes du Tarn plus ou moins coloré selon la valeur de l’APL. Cette coloration change en fonction des filtres choisis par l’utilisateur. La carte affiche aussi les infrastructures de santé du département en les localisant dans leur commune. Le découpage par commune est permis grâce à l’utilisation d’un fichier Geojson lu par la bibliothèque Leaflet. La figure 27 montre à quoi ressemble la carte focalisée sur le département du Tarn avec le découpage par commune.

Une image contenant carte, texte

Description générée automatiquement

Figure - Carte de France présente sur la webapp avec le découpage par commune

***Difficultés rencontrées***

La partie frontend du projet n’a pas soulevée beaucoup de difficulté. Elle nous a permis de travailler sur des technologies que nous ne connaissions pas en utilisant Leaflet. C’était aussi un moyen de découvrir les fichiers Geojson qui sont dérivés du json, mais spécifique au partage de données géographiques.

Conclusion

Le projet tutoré que nous avons présenté dans ce rapport a été une vraie expérience pour nous en tant qu’alternant. C’est un exercice que nous réalisions pour la première fois, car il ne survient qu’en 5ème année pour les étudiants en cycle alternant. Le travail réalisé au cours de ce projet est particulièrement galvanisant, car il nous permet d’aborder des thématiques que nous ne voyons pas forcément à l’école ou en entreprise. Plus particulièrement d’un point de vue technique, car aucun de nous 2 ne connaissait le Framework Django, mais aussi au niveau de la gestion de projet, car nous ne sommes pas à des postes qui nous permettent d’avoir un contact aussi direct avec le client.

Cette expérience est aussi accompagnée de difficulté liée à l’alternance en elle-même : il est compliqué de créer une application complexe en la laissant de côté pendant plusieurs semaines ou mois durant les périodes entreprise. Nous déplorons également l’ampleur du projet, qui aurait mérité qu’on s’y attarde plus longtemps que les heures prévues par nos emplois du temps (100 - 120h). Dans ce sens, nous pensons avec le recul que le projet aurait pu être plus avancé si nous avions eu une personne de plus pour nous accompagner dans la recherche et dans le développement. Une autre possibilité aurait été de proposer ce projet de recherche comme stage pour un étudiant de 4ème ou 5ème année.

Le cahier des charges présenté en amont n’a pas été entièrement complété, l’application est fonctionnelle et propose une page d’accueil, une page de cartographie et une page d’administration. La page de cartographie permet l’affichage de l’APL à différents niveaux géographique (national, régional, départemental et communal) même si les données de fixtures sont focalisées sur le département du Tarn. La carte est filtrable sur les professionnels de santé tout particulièrement sur leur profession. Les filtres sur l’âge et sur la spécialisation n’ont pas pu être terminés sur la partie calcul, mais sont en théorie, fonctionnels. L’application possède une base de données PostgreSQL ce qui permet une utilisation rapide, simple et agréable pour l’utilisateur. La partie d’importation des données n’est pas terminée sur l’interface administrateur. Les calculs de l’APL sont intégrés à l’application et ne fonctionnent pas dans un script indépendant.

Nous n’avons malheureusement pas pu mettre en place un déploiement par conteneur, qui aurait pu faciliter la récupération et mise en place du projet par les ingénieurs qui continueront notre travail. Cependant, nous avons au maximum essayé de faciliter cette transmission, en préparant la base de données pour les derniers points à développer. L’ajout sur la carte des établissements de santé et de leurs équipements n’est pas lié aux calculs de l’APL. De cette manière les établissements de santé, déjà mis en fixture, peuvent être ajoutés facilement sur la carte à partir des coordonnées géographiques de leur commune de rattachement. Pour ce qui est des équipements, ils sont reliés à un établissement de santé et peuvent être ajoutés sur le marqueur définissant l’établissement sur la carte.

Pour finir sur une note plus critique, notre choix de reprendre de zéro le travail sur l’application nous a pris beaucoup de temps et nous finissons « au même stade » que le précédent groupe. Cependant, nous continuons à défendre que ce choix fût pertinent, car l’application actuelle est plus fonctionnelle, modulable et correspond plus aux attentes du client. Le chargement de la page de cartographie est rapide grâce à la base de données et les calculs de l’APL sont dynamiques.

Pour conclure, nous souhaitons à nouveau remercier les différents intervenants de ce projet : Amadou COULIBALY, Elyes LAMINE, Imen MEGDICHE-BOUSARSAR et Virginie GOEPP-THIEBAUD.

Webographie

***Article de Recherche :***

Application Shiny : <https://drees.shinyapps.io/carto-apl/> [09/05/2024]

Explication de l’indicateur d’accessibilité potentielle localisée (APL) :  
<https://drees.solidarites-sante.gouv.fr/sources-outils-et-enquetes/lindicateur-daccessibilite-potentielle-localisee-apl> [09/05/2024]

Méthode de calcul de l’APL : <https://drees.solidarites-sante.gouv.fr/sources-outils-et-enquetes/lindicateur-daccessibilite-potentielle-localisee-apl> [14/06/2024]

CodeIgniter vs Laravel : <https://www.redswitches.com/blog/codeigniter-vs-laravel/#pros-and-cons-of-codeigniter> [16/05/2024]

The Best Database for Django Web Apps : <https://www.nickmccullum.com/best-database-django-web-apps/> [14/06/2024]

***Source de données gouvernementale :***

La démographie des professionnels de santé : [https://drees.solidarites-sante.gouv.fr/sources-outils-et-enquetes/la-demographie-des-professionnels-de-sante](about:blank) [14/05/2024]

Les communes avec leur coordonnées géographiques : <https://www.data.gouv.fr/fr/datasets/communes-de-france-base-des-codes-postaux/> [14/06/2024]

L’annuaire santé de CNAM contenant les informations sur les professionnels de santé, profession, spécialité et les infrastructures : <https://www.data.gouv.fr/fr/datasets/annuaire-sante-de-la-cnam/#/resources> [14/06/2024]

Le projet mettant à disposition une API gratuite pour faire du calcul de distance entre 2 points : <https://project-osrm.org/> [14/06/2024]

Description d’ORM : <https://fr.wikipedia.org/wiki/Mapping_objet-relationnel> [14/06/2024]

GitHub du projet FranceGeoJson proposant un découpage de la France par région, département et communes : <https://github.com/gregoiredavid/france-geojson/blob/master/communes.geojson> [14/06/2024]

Glossaire

***Abréviations***

* **APL (Accessibilité Potentielle Localisée)** : Indicateur développé pour mesurer l'accessibilité aux soins de santé dans différentes régions géographiques.
* **ETP (Équivalent Temps Plein)** : Mesure de l'activité d'un professionnel de santé basée sur son nombre d'heures de travail disponibles.
* **CNRS (Centre National de la Recherche Scientifique)** : Principal organisme public de recherche en France.
* **ENGEES** : École Nationale du Génie de l'Eau et de l'Environnement de Strasbourg.
* **SNIIRAM (Système National d'Information Inter-Régimes de l'Assurance Maladie)** : Système destiné à centraliser et analyser des données sur l'assurance maladie en France.
* **RPPS (Répertoire Partagé des Professionnels de Santé)** : Base de données regroupant les informations des professionnels de santé en France.
* **ADELI (Automatisation DEs LIstes)** : Répertoire national des professionnels de santé et des autres professionnels intervenant dans le domaine sanitaire.
* **DREES (Direction de la Recherche des Études de l'Évaluation et des Statistiques)** : Direction du ministère de la Santé chargée des études et des statistiques en matière de santé.
* **ORM (Object Relational Mapping)** : Technique de programmation permettant de convertir des données entre des systèmes de types incompatibles en utilisant des objets de programmation.
* **JSON** : Format de données textuelles utilisé pour l'échange de données entre un serveur et une application web.
* **CSV** : Format de fichier texte utilisé pour stocker des données tabulaires où chaque ligne représente un enregistrement et les colonnes sont séparées par des virgules.
* **UML (Unified Modeling Language)** : Langage de modélisation graphique utilisé pour spécifier, visualiser, construire et documenter les composants d'un système logiciel.
* **MVC (Model View Controller)** : Modèle architectural utilisé pour le développement d'interfaces utilisateur qui divise une application en trois composants principaux : le modèle, la vue et le contrôleur.
* **HTML** : Langage de balisage utilisé pour créer des pages web.
* **CSS** : Langage de feuille de style utilisé pour décrire la présentation d'un document écrit en HTML ou XML.

***Termes Techniques***

* **Django** : Framework web open-source orienté Python qui suit le modèle architectural MVC.
* **Bootstrap** : Framework CSS destiné à faciliter le développement de sites web responsives et mobiles.
* **Leaflet** : Bibliothèque JavaScript permettant de créer des cartes interactives et de les intégrer dans des sites web.
* **Chart.js** : Bibliothèque JavaScript permettant de créer des graphiques interactifs sur des sites web.
* **Numpy** : Bibliothèque Python pour le calcul scientifique, utilisée pour manipuler des tableaux multidimensionnels et matrices.
* **Pandas** : Bibliothèque Python utilisée pour la manipulation et l'analyse des données, construite sur Numpy.
* **PostgreSQL** : Système de gestion de base de données relationnelle et objet, connu pour sa stabilité et ses fonctionnalités avancées.
* **CodeIgniter** : Framework PHP pour le développement web, connu pour sa simplicité et sa légèreté.
* **GitHub** : Plateforme de développement collaboratif et de gestion de code source utilisant Git pour le contrôle de version.
* **Trello** : Outil de gestion de projet basé sur la méthodologie Kanban, permettant de visualiser les tâches à réaliser, en cours et terminées.
* **Teams** : Outil de communication et de collaboration de Microsoft, intégrant la messagerie, les réunions et le partage de fichiers.
* **Jupyter** : Environnement interactif pour le calcul en Python, permettant de créer et de partager des documents contenant du code, des équations, des visualisations et du texte narratif.

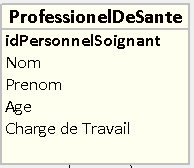
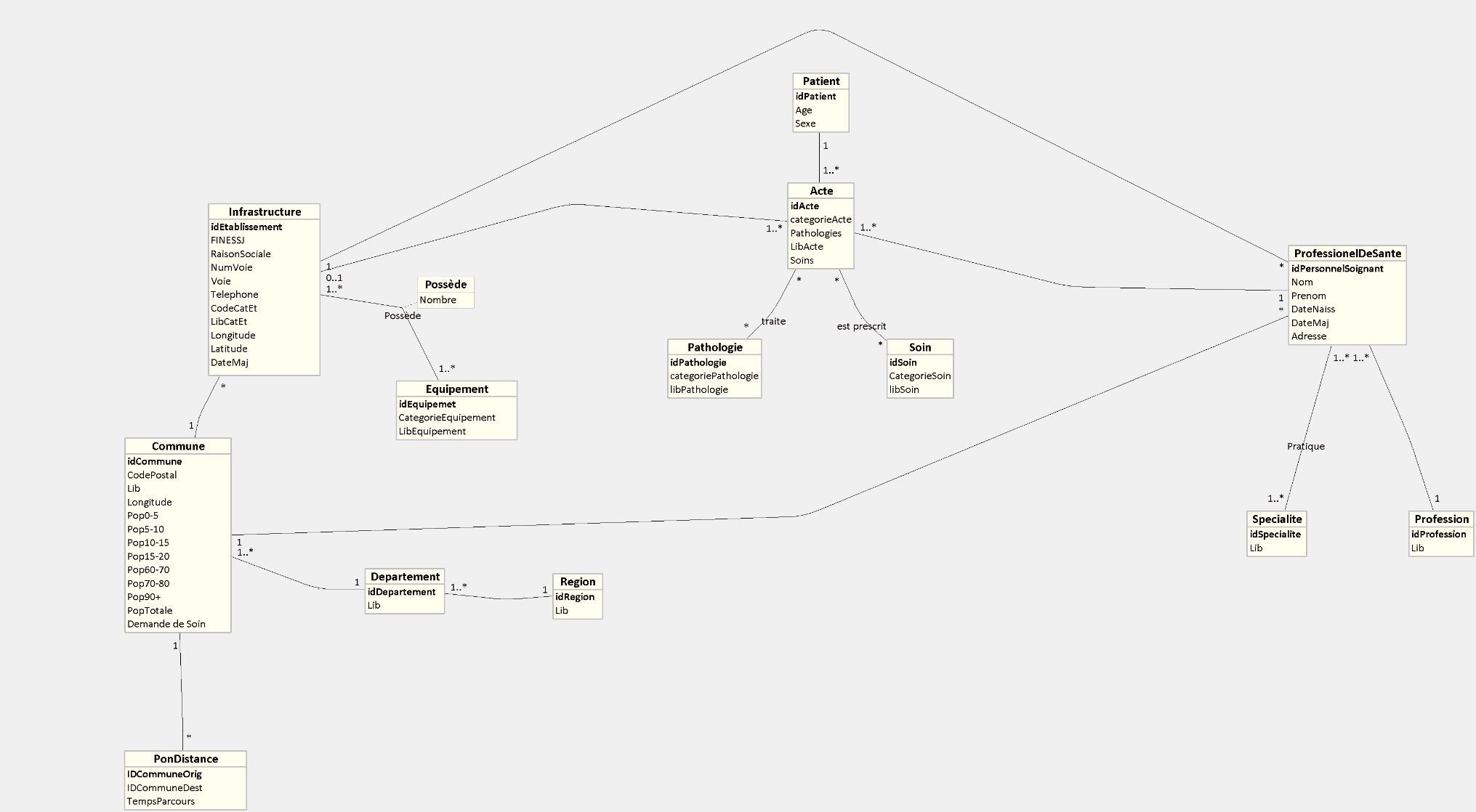
Annexes

# Annexe 1 : Proposition de projet tutoré

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

# Annexe 2 : Diagramme de classe de l’application



Champs générés encadrés en orange

1. *Article du DREES paru le 21/07/2020,*  [↑](#footnote-ref-2)
2. *L’indicateur d’accessibilité potentielle localisée (APL), paru le 20/08/2020* [↑](#footnote-ref-3)