



UNIVERSITÉ DE FRANCHE-COMTÉ

L3 CMI Informatique

PROJET TUTORÉ

Développement d'une application pour l'optimisation des rendez-vous médicaux par géolocalisation : Géo'doc

SMOLINSKI Marie, BORDE Corentin, SÉGARD Guillaume

Tuteur : MOUNTASSIR Hassan



Année universitaire 2018-2019

Remerciements

Nous souhaitons remercier chaleureusement M. Mountassir, enseignant chercheur à l'Université de Bourgogne-Franche-Comté, qui nous a accompagné et nous a fait des retours tout au long de cette année universitaire. Nous souhaitons également remercier M. Dadeau a su nous fournir de l'aide sur des points précis lors du développement.

Contents

1 Modélisation de l'application	4
1.1 Spécifications pour Géo'doc	4
1.2 Modélisation UML et autres	5
1.2.1 Représentation du modèle de données	5
1.2.2 Plan du site	8
1.3 Répartition du travail	8
2 Géo'doc : comment le site a été réalisé	9
2.1 Outils utilisés	10
2.1.1 Langages choisis	10
2.1.2 Outils complémentaires	10
2.2 Développement de l'application	11
2.2.1 Page d'accueil : le pivot de l'application	11
2.2.2 Planning des médecins : un vivier de fonctionnalités	14
2.2.3 Une identité graphique à créer	16
2.3 Difficultés rencontrées	16
3 Bilan : où en est le développement ?	18
3.1 Une application qui remplit les attentes	18
3.2 Un site voué à évoluer	18
3.2.1 Envoi de mails	18
3.2.2 Recherche de médecins	19
3.2.3 Compatibilité	19
4 Conclusion	20

Introduction

Le projet Géo'doc, supervisé par Hassan Mountassir, consiste au développement d'une application permettant l'optimisation de la prise de rendez-vous chez des médecins de diverses spécialités, à l'aide notamment de la géolocalisation.

Cette idée a été motivée par la tâche chronophage qu'est la gestion des rendez-vous pour les médecins. En effet, cette tâche est estimée à 30% de leur temps, temps qui pourrait être bien mieux investi, notamment dans les zones qui manquent de spécialistes. Un rendez-vous sur deux chez un généraliste est obtenu en moins de deux jours, cependant, lorsqu'il s'agit d'un spécialiste, les délais d'attente peuvent être de plusieurs mois, 80 jours en moyenne pour l'ophtalmologie par exemple. Bien entendu, les temps d'attentes sont plus longs dans les communes où l'accès géographique à un professionnel est faible. Il est donc important de ne pas perdre de temps.

Il est également important de permettre un retour sur les rendez-vous passés. En effet, un second problème important est celui des rendez-vous non honorés. Via l'application, on souhaite pouvoir obtenir des informations sur ces taux, mais également pouvoir influer dessus en dissuadant les annulations intempestives.

Nous décrirons donc la façon dont nous avons modélisé cette application pour répondre au mieux aux spécifications, puis nous expliquerons les étapes de développement par lesquelles nous sommes passées et les changements qu'elles ont engendré. Nous conclurons en tirant le bilan des actions réalisées et des améliorations à apporter.

1 Modélisation de l'application

Nous avons commencé par définir les spécifications et représenter un premier modèle avant de commencer le développement de l'application. Le modèle a ensuite évolué pendant toute une partie du développement; c'est cette évolution que nous décrirons dans cette partie.

1.1 Spécifications pour Géo'doc

Dans un premier temps, on cherche à produire une application qui réponde efficacement au besoin pour un client de prendre un rendez-vous le plus proche avec un spécialiste ou un généraliste dans un rayon géographique donné. Pour cela on demande qu'un client puisse :

- rechercher le médecin le plus proche de sa position géographique
- prendre rendez-vous chez le médecin de son choix en fonction de ses disponibilités
- annuler ou déplacer ses rendez-vous
- avoir une traçabilité sur les rendez-vous à venir ou sur les visites passées

Du côté des médecins, on souhaite pouvoir gérer efficacement et en minimisant la perte de temps. On doit pouvoir :

- accéder à son planning et le gérer : annuler ou déplacer des rendez-vous
- affecter des rendez-vous, juger et gérer les urgences

On souhaite bien évidemment une application ergonomique qui ne se limite pas strictement aux fonctionnalités essentielles :

- possibilité de renseigner et modifier ses informations personnelles, ainsi que, pour les médecins, des informations professionnelles
- on souhaite également stocker des informations dans le but de les transmettre à l'ARS (Agence Régionale de Santé), et ainsi lutter contre les déserts médicaux entre autres
- placement des éléments pratique et esthétique agréable

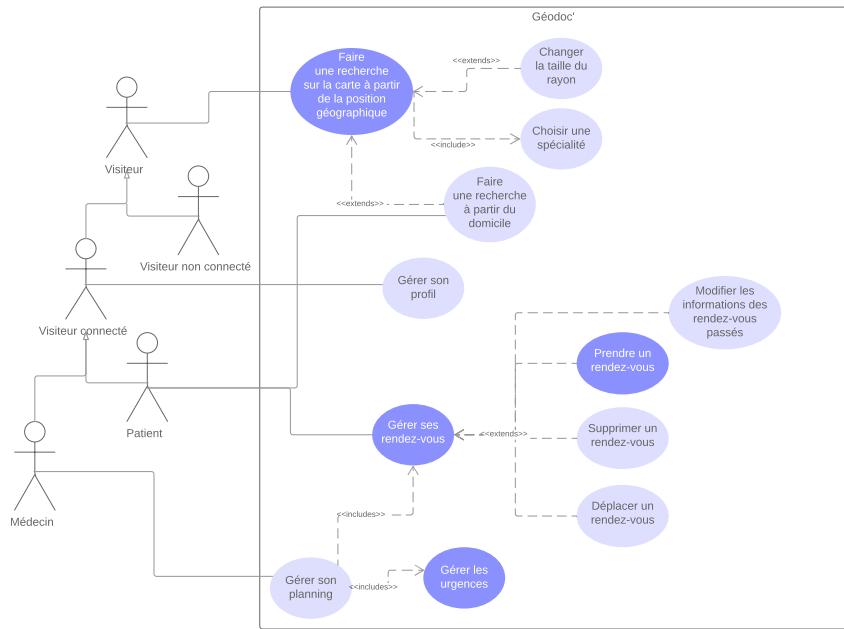


Figure 1: Diagramme de cas d'utilisation de Géo'doc

Nous avons réalisé en figure 1 le diagramme de cas d'utilisation qui nous a servi de base avant de commencer l'application. Il permet notamment d'identifier les acteurs et leurs spécialisations, ainsi que les actions qui leur sont réservées. On peut par exemple voir que seuls les médecins peuvent déplacer un rendez-vous, les clients pouvant seulement les prendre et les annuler. Nous avons identifié quelles sont les informations à conserver en fonction du statut du visiteur; en effet, un visiteur non connecté ne pourra avoir accès qu'aux informations publiques (celles des médecins, les rendez-vous occupés), et l'application ne retiendra aucune information de lui. En revanche il faudra stocker les informations de profil, de rendez-vous et de domicile d'un visiteur connecté, pour lui permettre d'accéder à plus de fonctionnalités, comme la localisation de médecins à partir du domicile.

1.2 Modélisation UML et autres

1.2.1 Représentation du modèle de données

Nous avons débuté par la modélisation théorique du modèle de données à l'aide de diagrammes de classe UML (figure 2).

Nous avons choisi un fonctionnement par créneaux pour correspondre au fonctionnement d'un médecin. En effet, le médecin indique souvent les créneaux

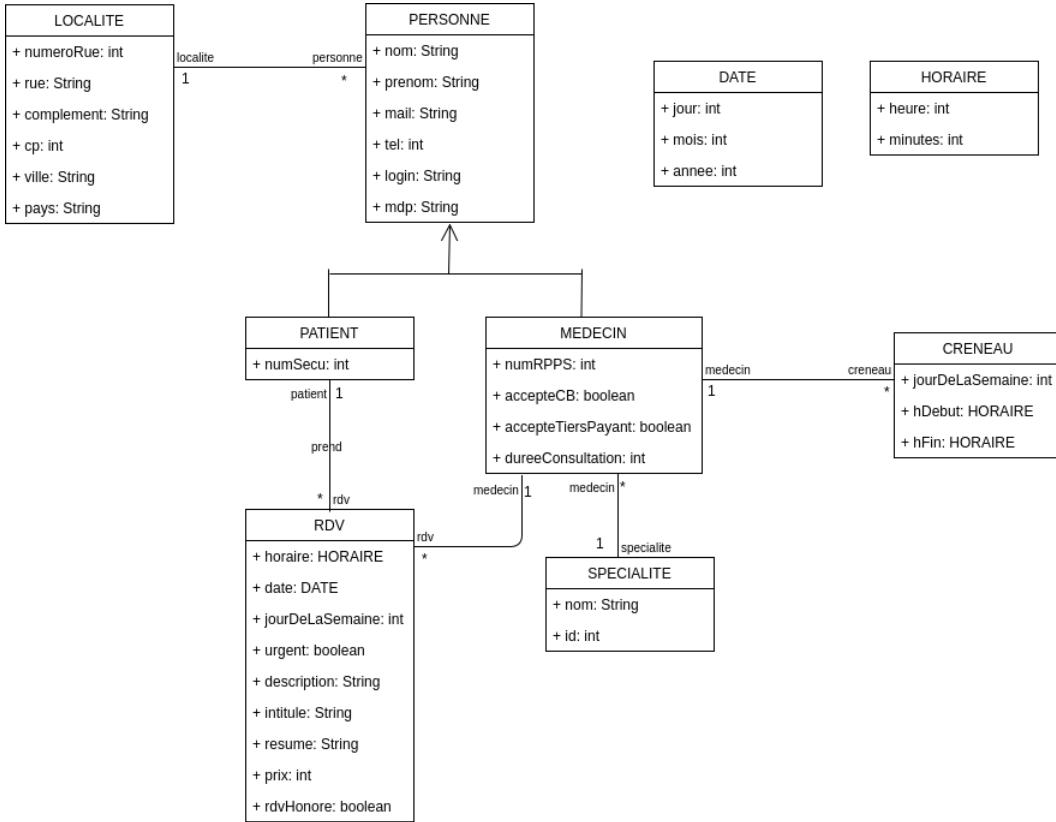


Figure 2: Diagramme de classes

dans la semaine ou la prise de rendez-vous est possible et ceux où la visite se fait sans rendez-vous, ces créneaux étant persistants d'une semaine à l'autre. Nous auront donc un planning constitué de créneaux et présenté par semaines, et la division en rendez-vous se fera par calcul en fonction de la durée générale des rendez-vous définie par chaque médecin, et de l'heure de début et de fin du créneau. Ce choix de modélisation rend les calculs plus complexes mais nous avons fait ce choix pour éviter de stocker trop d'informations. En effet, pour permettre l'affichage rapide à l'aide des seules informations contenues dans la base de données et sans calculs, il aurait fallu stocker tous les rendez-vous même les libres. Cela aurait été très lourd, mais nous aurions également été limités dans le temps pour l'affichage. Ici, seuls les rendez-vous occupés sont stockés dans la base.

Nous avons également fait le choix de regrouper en une classe les rendez-vous passés et futurs, malgré leur différence. Les informations auxquelles on souhaite accéder lorsque le rendez-vous est passé concernent le résumé, le prix, ce qui s'y est passé, tandis que pour un rendez-vous à venir on a besoin de l'heure, des

informations d'urgence. Cependant, les objets sont les mêmes dans le temps, dans le sens où tous les rendez-vous planifiés se transformeront en rendez-vous passé dont on souhaite l'accès, et certaines propriétés sont communes.

Lors de la transformation de ce modèle de données en base de données concrète, puis ensuite lors du début du développement, la modélisation a évolué pour atteindre le modèle de base de donnée décrit dans la figure 3. On constate que les rendez-vous ne sont pas liés avec les créneaux, ce qui est consistant avec le choix évoqué plus haut. Nous avons hésité à stocker les créneaux par semaine, ce qui aurait permis une plus grande liberté pour le médecin dans le placement et la suppression d'un créneau (on doit sinon s'assurer que le créneau est vide pour toutes les semaines avant de le supprimer), cependant cela aurait nécessité des manipulations pénibles au médecin pour programmer les créneaux à l'avance pour de longues périodes, et nous avons dit que les plages horaires sur lesquelles les médecins prennent des rendez-vous sont persistantes d'une semaine à l'autre.

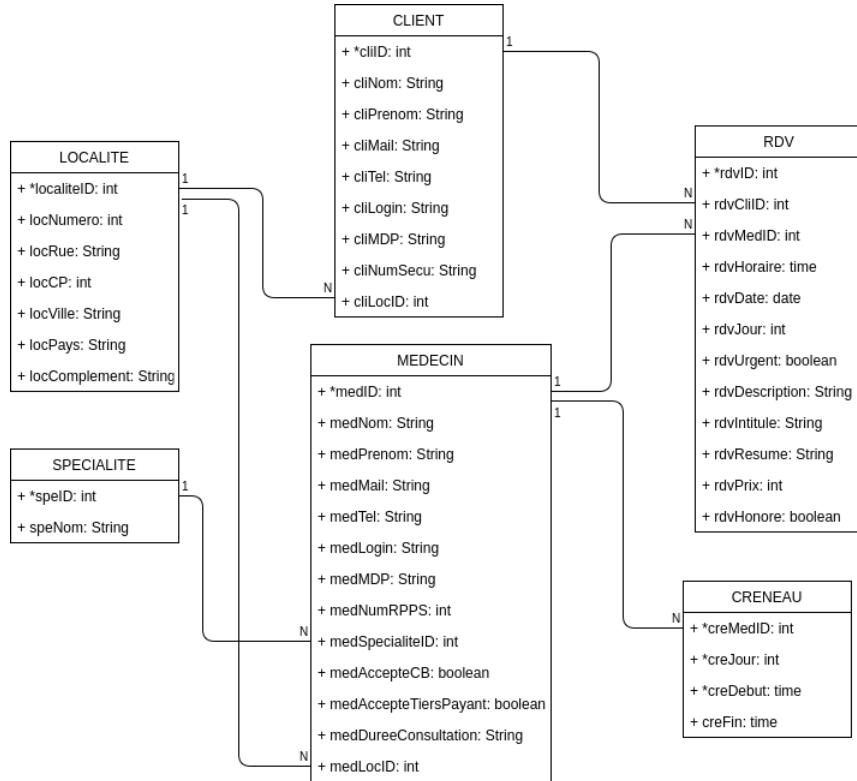


Figure 3: Schéma relationnel de la base de données

Nous avons donc fait le choix de créneaux qui n'ont pas d'informations de date mais seulement d'heure.

1.2.2 Plan du site

Dès le départ et avant de commencer le développement de l'application, nous avons réalisé des schémas des différentes pages du site, comme la page de présentation des médecins présente en figure 4, avec un premier placement des éléments, les différentes redirections, et une liste exhaustive des fonctionnalités et d'indications, afin d'avoir une base commune solide et visuelle.

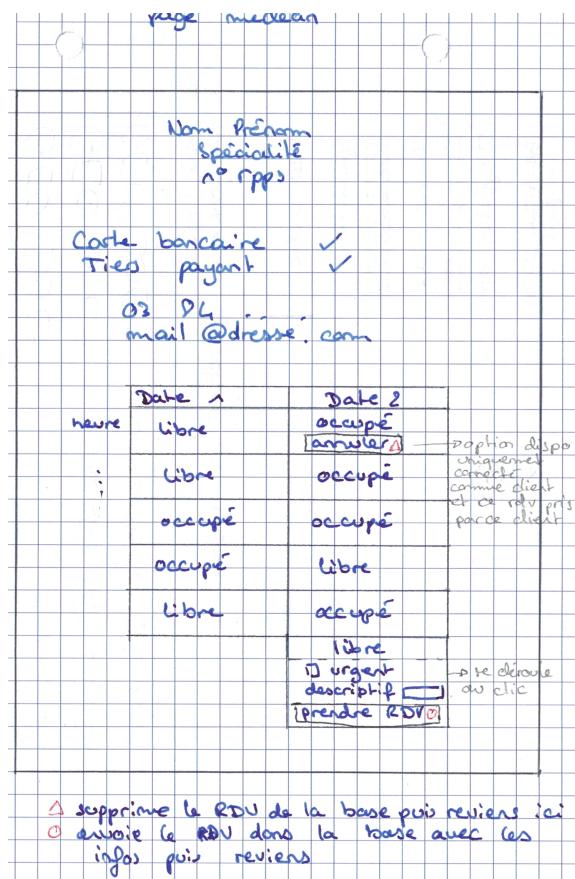


Figure 4: Schéma préliminaire guide de la page médecin

1.3 Répartition du travail

Le début du développement a été l'occasion de définir des rôles et des tâches à remplir pour chacun, qui ont également évolué, en fonction de la difficulté de ces tâches et des modifications en tous genres.

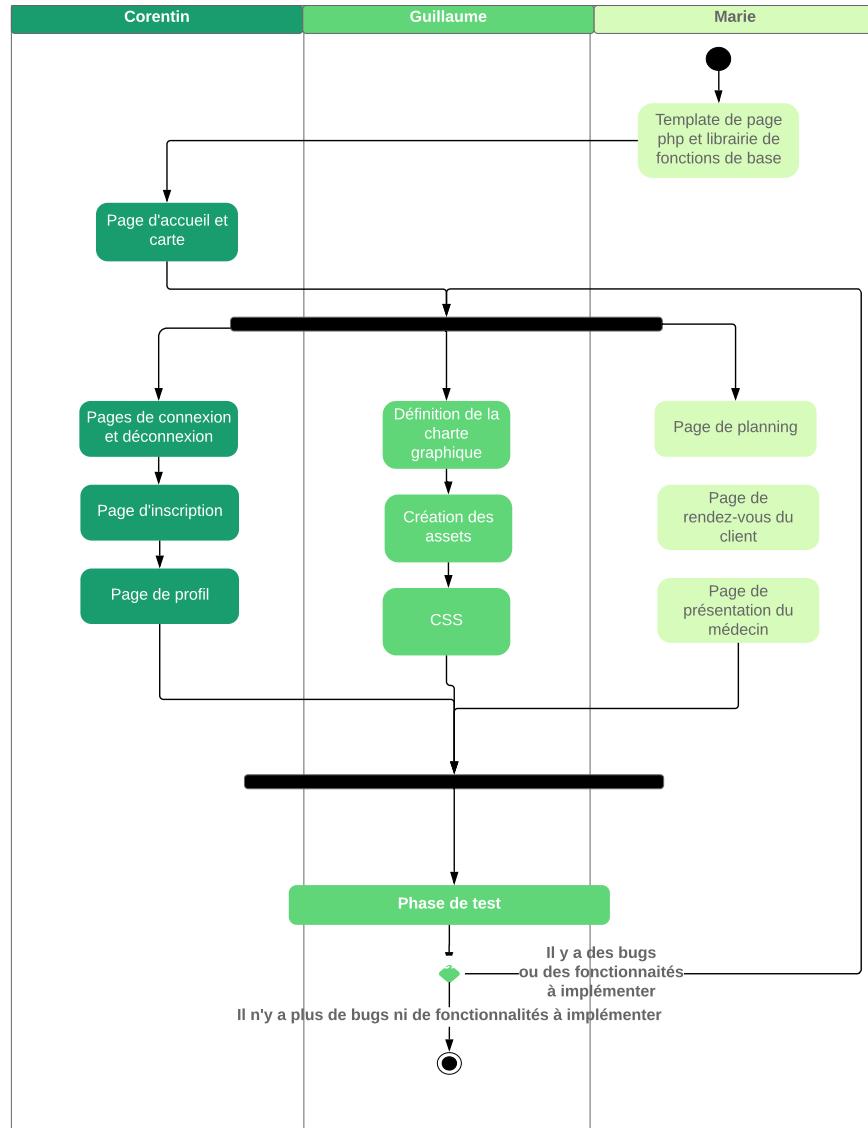


Figure 5: Diagramme d'activité du développement de Géo'doc

2 Géo'doc : comment le site a été réalisé

Dans cette partie nous expliquerons de façon plus concrète le déroulement du développement de Géo'doc via les choix d'outils et de technologies utilisées, ainsi

que dans les détails d'implémentation, les difficultés rencontrées et les solutions trouvées.

2.1 Outils utilisés

2.1.1 Langages choisis

Les spécifications et les attentes nous demandaient d'utiliser les langages du web et de développer un site internet. Les langages que nous avons donc utilisés sont HTML5, CSS3, PHP et JS (JavaScript).

Nous avons choisi de développer notre site web en langage PHP et de n'utiliser le langage JS que dans la page d'accueil pour afficher la carte et les médecins.

En effet, notre site se place du côté serveur et le langage PHP y est très adapté.

Nous avons tout de même pensé aux avantages et inconvénients de ce choix :

- l'utilisation de PHP permet de faire des requêtes SQL à notre base de données, alors qu'en JS, il faut utiliser une fonctionnalité intermédiaire, nommée AJAX (dont nous parlerons plus tard), ce qui peut avoir un impact sur l'optimisation du site et alourdir le code source;
- PHP ne permet pas de faire d'affichage dynamique. Par exemple, l'affichage des plannings et des rendez-vous aurait été plus facilement implémenté en JS.

2.1.2 Outils complémentaires

Pour développer certaines fonctionnalités, nous avons besoin de différents outils et de quelques API (interface de présentation) que nous présentons ici :

Géolocalisation Les navigateurs permettent de connaître la position du client, seulement s'il accepte d'être géolocalisé. Cette fonctionnalité nous permet de récupérer les coordonnées GPS (latitude et longitude).

Leaflet L'API *Leaflet* [1] est une API Open Source qui utilise les cartes de *OpenStreetMap*. Cette API nous permet d'afficher un cadre dans notre page, qui contient une carte. Il est possible de choisir la vue, le secteur et le niveau de zoom de la carte. Il est aussi possible de placer des repères sur la carte en connaissant les coordonnées GPS et d'y associer une petite fenêtre popup qui peut contenir des informations.

JavaScript L'API *Leaflet* s'utilise avec le langage JavaScript (JS). Un script JS peut être appelé depuis un document HTML ou PHP au moyen de la balise `<script>` placé dans l'en-tête du document. Il faut également prévoir une balise `<div>` dans le document HTML/PHP pour pouvoir y placer la carte.

AJAX Le langage JavaScript ne permet pas de faire des requêtes SQL à une base de données car ce langage s'exécute côté client. Les requêtes SQL doivent être traitées du côté serveur avec le langage PHP. Nous avons donc utilisé AJAX (Asynchronous Java Script XML).

Le principe d'AJAX est d'exécuter un script PHP qui va se charger de faire la requête SQL et de renvoyer une réponse sous format JSON (JavaScript Object Notation) qui contient les données renvoyées par la requête SQL.

JSON est un format de donnée textuel permettant de représenter de l'information structurée.

AJAX est asynchrone, c'est-à-dire que l'attente de la réponse ne bloque pas l'exécution du reste du script JS.

Adresse2Coordonnes Cette API [3] permet de convertir une adresse postale en coordonnées GPS. Nous en avons besoin pour le placement des repères sur la carte car nous avons fait le choix de ne pas stocker les coordonnées GPS des médecins ou des clients dans la base de données.

2.2 Développement de l'application

Nous nous focaliserons dans un premier temps sur des pages qui ont concentré le travail sur l'application de par leur complexité ou leur intérêt.

2.2.1 Page d'accueil : le pivot de l'application

En plus d'être la première page que l'on voit, et une de celles sur laquelle on revient le plus souvent, c'est également celle qui contient le plus d'éléments et de fonctionnalités. C'est sur cette page que l'on gère la géolocalisation, que l'on interagit avec la base de données, et que l'on donne des informations sur les disponibilités des médecins.

Description La page d'accueil permet d'afficher sur une carte la liste des médecins correspondant à une spécialité donnée. C'est l'utilisateur qui choisit la spécialité ainsi que la distance maximale. Cela se fait au moyen d'un formulaire affiché sur le côté de la carte.

Si le client est connecté, il a la possibilité d'être localisé à son domicile plutôt que sa position actuelle. Il faut bien entendu qu'il ait donné son adresse auparavant.

En cliquant sur l'icône d'un médecin sur la carte, le client a accès à des informations sur ce médecin :

- son nom et son prénom;
- son adresse;
- si il accepte la carte bancaire et le tiers payant;
- la distance qui le sépare de la position choisie;

- sa première disponibilité (date et heure);
- un lien sur la page de prise de rendez-vous.

La figure 6 montre une capture d'écran de la page d'accueil après avoir fait une recherche de médecins généralistes.

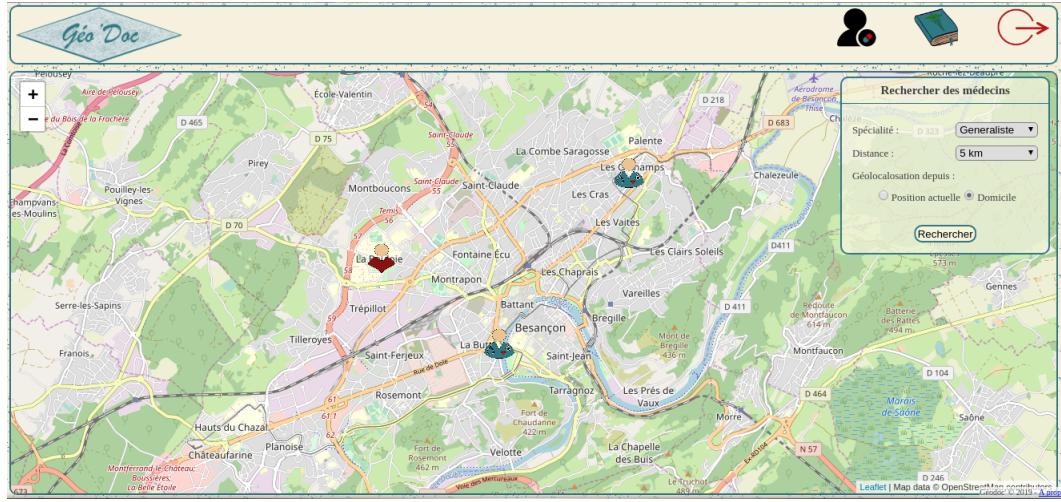


Figure 6: Page d'accueil de Géo'doc

Fonctionnement du script La première chose à faire est de récupérer la position du client. Dans un souci de légalité par rapport à confidentialité, les navigateurs ont l'obligation de demander l'autorisation au client. Si le client ne donne pas son accord, nous affichons la carte centrée sur Besançon et ne continuons pas les calculs.

Sinon, la carte est affichée centrée sur la position du client.

Lors de l'appui sur le bouton “Rechercher” du formulaire, une requête AJAX est lancée pour récupérer les informations et adresses des médecins de la spécialité sélectionnée. Parallèlement, si le client a coché la case pour faire la recherche depuis le domicile, une requête AJAX pour récupérer son adresse est lancée.

Une fois les réponses arrivées, on utilise l'API *Adress2Coordonnées* pour convertir les adresses obtenues en coordonnées GPS. On peut, dans la foulée, calculer la distance entre la position du client (ou le domicile) et les médecins. Un repère est placé sur la carte à l'adresse de chaque médecin, si la distance est bien inférieure à celle sélectionnée par le client dans le formulaire. Une dernière requête AJAX est lancée pour récupérer la date et l'heure de la première disponibilité de chaque médecin pour pouvoir l'afficher dans les popups associées à chaque repère.

Nous trouvons intéressant de présenter ici quelques lignes de code qui montrent l'utilisation d'AJAX et de l'API *Adress2Coordonnes*. La calcul de la distance entre deux points est aussi donnée.

La figure 7 montre comment AJAX s'utilise.

```
1 xhr_object = new XMLHttpRequest();
2
3 xhr_object.open("POST", url, false);
4 xhr_object.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
5
6 var data = "specialite="+specialite;
7
8 xhr_object.send(data);
9
10 if (xhr_object.readyState == 4){
```

Figure 7: Utilisation d'AJAX

Il faut créer un objet de type `XMLHttpRequest()`, le paramétrier en donnant, entre autre, l'url du script à exécuter, la méthode pour le passage des données, etc. Ce sont les lignes 3 et 4 qui le montrent.

Si on veux envoyer des données au script, il suffit de créer une variable qui contient une chaîne de caractère du type `clé=valeur`, et de l'envoyer à l'objet créé précédemment (ligne 8). Dans cet exemple, c'est la spécialité du médecin sélectionnée qui sera envoyée.

Quand le script donne la réponse, une propriété de l'objet change de valeur (`readyState`), et nous pouvons traiter les données renvoyées (ligne 10).

L'API *Adress2Coordonnes* utilise une forme simplifiée d'AJAX, comme le montre la figure 8.

```
var xmlhttp = new XMLHttpRequest();
var url = "https://nominatim.openstreetmap.org/search?format=json&limit=3&q=" + adresse;
xmlhttp.onreadystatechange = function(){
```

Figure 8: Utilisation de l'API *Adress2Coordonnes*

Il suffit de lui donner le lien du serveur qui calcule les coordonnées en fonction d'une adresse. Cette adresse est donnée à la fin du lien (on concatène le lien et la variable contenant l'adresse, ligne 2). Une fois la réponse arrivée, on peut exécuter une fonction.

Le calcul de la distance entre deux points sur une sphère est différent par rapport au calcul de la distance entre deux points sur un plan. Il nous faudra utiliser le rayon de la Terre, qui est d'environ 6378.137km.

Soient deux points de coordonnées (x_1, y_1) et (x_2, y_2) , après avoir convertit les

coordonnées en radian, la formule de calcul de la distance est la suivante :

$$d = r * \left(2 * \arctan \left(\frac{\sqrt{a}}{\sqrt{a-1}} \right) \right)$$

avec $r = 6378.137$, le rayon de la Terre

et

$$a = \left(\sin \left(\frac{x_2 - x_1}{2} \right) \right)^2 + \cos(x_1) * \cos(x_2) * \left(\sin \left(\frac{y_2 - y_1}{2} \right) \right)^2$$

2.2.2 Planning des médecins : un vivier de fonctionnalités

Figure 9: Page planning de Géo'doc

La page planning des médecins (figures 9 et 10) concentre un bon nombre de fonctionnalités concernant les rendez-vous, mais également sur le fonctionnement de la page. La figure 11 présente cette page. On constate 4 sections qui ont toutes nécessité des calculs différents :

- une section présentant les urgences à traiter ainsi que la description de l'urgence. Le médecin peut ensuite saisir le numéro de sécurité sociale affiché et l'affecter à un rendez-vous. Ici on récupère simplement dans

Rendez-vous passés

05/01/2019 à 14h30 Segard Guigui +

Numéro de sécurité sociale : 198082538825273

Intitulé

Résumé

Prix (€)

Le rendez-vous a-t-il été honoré ? oui non

Modifier

Figure 10: Rendez-vous passé

la base les rendez-vous à venir concernant le médecin et dont l'attribut "urgent" est à `true`.

- le planning en lui-même, pour la semaine en cours, ou bien pour une semaine sélectionnée. La figure 9 montre le pseudo-code de la fonction d'affichage du planning.

Chaque créneau y est récupéré dans la base puis affiché, puis on réalise un calcul permettant de connaître le nombre de rendez-vous, à l'aide des timestamp des heures récupérés grâce aux fonctions `date()` et `strtotime()`. On va donc, pour chaque rendez-vous du créneau, tester si l'heure de début du créneau correspond à l'heure de début d'un rendez-vous prévu dans la base.

Si oui on affichera un rendez-vous occupé, avec un formulaire permettant de le déplacer ou de l'annuler.

Si non on affichera un rendez-vous libre avec un formulaire permettant d'affecter un client.

- une section permettant la gestion des créneaux : l'ajout et la suppression (si aucun rendez-vous n'est pris pour toutes les semaines). Ici on récupère les créneaux du médecin dans la base et on affiche un formulaire pour chacun.

```

3   $creneaux = getCreneau($id_medecin);
4
5   foreach ($creneaux as $cre) {
6       $rdvs = getRdvsOfCreneau($id_medecin, $cre);
7       $nb_rdvs = ($timestamp_debut_creneau - $timestamp_fin_creneau) / $timestamp_duree_consultation;
8
9       for ($i = 0; $i < $nb_rdvs; ++$i) {
10           $debut_rdv = $timestamp_debut_creneau + ($timestamp_duree_consultation * $i);
11           $libre = true;
12
13           foreach ($rdvs as $rdv) {
14               if ($debut_rdv == $rdv['debut']) {
15                   display_rdv_occupe();
16                   $libre = false;
17               }
18           }
19
20           if ($libre) {
21               display_rdv_libre();
22           }
23       }
24   }

```

Figure 11: Pseudo-code de display_planning

- les rendez-vous passés. Pour chaque rendez-vous récupéré dans la base, on pourra cliquer sur un bouton pour dérouler les informations sur ce rendez-vous. On affichera un formulaire permettant de mettre à jour ces informations. Ce détail est codé à l'aide d'une petite fonction JavaScript qui permet de changer l'affichage de la section contenant les informations de `none` à `block`.

2.2.3 Une identité graphique à créer

La charte graphique est représentative des milieux hospitalier. Elle est composée majoritairement de blanc cassé et de bleu, couleurs que nous pouvons retrouver dans des hôpitaux par exemple. Ainsi, les utilisateurs ne sont pas dépayrés. Chaque asset a été créé de toute pièce, ou obtenu par la modification d'images étant libres de droits.

En ce qui concerne la présentation des pages, elle reste sobre et basique, afin de rendre l'utilisation du site la plus simple possible.

2.3 Difficultés rencontrées

Le projet s'est globalement bien déroulé, cependant nous avons été confrontés à quelques difficultés diverses. Ces difficultés restent minimes et n'ont à aucun moment mis en danger le bon déroulement du projet.

API de géolocalisation et affichage de la carte Nous avons cherché une API qui permette d'afficher une carte et de pouvoir y placer des repères en

fonction de coordonnées GPS ou d'adresses. Nous nous sommes renseigné pour utiliser l'API *Google Maps*, mais il s'avère que cette solution est payante. Nous avons donc cherché une autre API gratuite et nous avons trouvé l'API *Leaflet* qui répond parfaitement à nos attentes.

Apprentissage de nouveaux langages Au début du développement de l'application, nous ne connaissions pas le langage JavaScript. Nous avons donc commencé à apprendre les bases du langage en faisant quelques tutoriels en ligne. Il nous a également fallu apprendre à utiliser AJAX et les différentes API. Cependant une fois les connaissances acquises, le développement s'est déroulé sans encombre.

Problèmes de géolocalisation Il nous est arrivé d'avoir des problèmes avec la géolocalisation à partir du domicile, par exemple d'être localisé dans une autre région de France, à plusieurs dizaines de kilomètres de notre réelle position. Nous supposons que le problème vient de l'API *Adress2Coordonnes* qui ne référence pas toutes les adresses. Il est aussi possible que les clients renseignent mal leur adresse. Nous avons recherché un outil pouvant nous aider dans cette situation, cependant nous n'avons pas trouvé.

Modélisation Au départ nous avons réalisé une erreur de modélisation qui coûte cher à la complexité des algorithmes permettant d'afficher les plannings. En effet, nous n'avons pensé que très tard à lier la table créneau et la table rendez-vous par un attribut `rdvCreID` qui serait présent dans la table des rendez-vous. Cela aurait simplifié les algorithmes et permis une meilleure complexité.

Formulaires Nous l'avons évoqué plus haut, il n'est pas possible de faire de l'affichage dynamique en php, il a donc fallu passer essentiellement par formulaires, et cela a posé problème dans des pages comme le planning d'un médecin, sur laquelle sont présentes beaucoup de fonctionnalités. Cependant, une bonne organisation permet de venir à bout de ce problème.

Timestamp Les timestamp sont utilisés pour réaliser des calculs importants, notamment le timestamp 0. Cela nous a posé des problèmes de paramétrage lorsque nous nous sommes rendu compte que le timestamp 0 n'était pas le même sur toute les machines, et nous avons mis un certain temps à les résoudre, en harmonisant le fuseau horaire à l'aide d'une fonction.

Renseignement des disponibilités des médecins Sur la page d'index, les médecins sont affichés en premiers en fonction de leur distance. En effet, après plusieurs discussions durant lesquelles nous hésitions entre présenter les médecins en une liste triée par disponibilité, et sur une carte en fonction de la distance, nous avons choisi la deuxième solution puisque les spécifications mettent en avant la géolocalisation. Cependant, le fait de pouvoir visualiser rapidement les disponibilités manquait puisqu'il fallait quitter la carte, aller

sur la page médecin pour voir son planning et ses disponibilités. Nous avons réglé le problème à l'aide d'une fonction calculant la première disponibilité d'un médecin, dont le résultat est affiché dans les bulles d'informations qui apparaissent au clic sur le repère d'un médecin sur la carte.

3 Bilan : où en est le développement ?

3.1 Une application qui remplit les attentes

Les principales attentes (citées paragraphe 1.1) ont été traitées méticuleusement.

De plus, nous avons respecté les objectifs que nous nous sommes fixés en amont, comme on peut le constater sur la figure 9, page obtenue en respectant le schéma de la figure 4.

L'objectif de fournir des informations à l'ARS (Agence Nationale de Santé), en vue de faire évoluer les situations problématiques, est rempli, puisque de simples calculs permettent d'obtenir des informations comme le taux d'annulation des rendez-vous, ou le temps d'attente moyen pour un rendez-vous par spécialité et par région etc...Toutes les informations nécessaires sont stockées dans la base, et l'on peut théoriquement envoyer des rapports à l'ARS avec les résultats de ces calculs. Cependant ces données ne sont pas présentées sur le site, puisqu'elle n'ont d'intérêt que sous la forme d'un rapport envoyé à l'organisme qui pourra les diffuser, et nous n'avons pas trouvé le moyen de les rendre pertinentes. Cependant, on pourrait imaginer un système d'administrateur du site qui pourrait en un clic exécuter des fonctions pour récupérer des informations et les transmettre à l'ARS. Ainsi, l'envoi de ces données permettrait de lutter contre les désert médicaux, notamment en campagne.

Des améliorations ont été apportées à la base du projet, comme par exemple le fait que l'on puisse rechercher des médecins depuis son domicile, ou encore qu'un médecin puisse avoir une traçabilité de ses rendez-vous passés.

3.2 Un site voué à évoluer

Dans le cas où le projet viendrait à être repris par d'autres développeurs, nous avons simplifié au maximum le code par des phases de commentaires, par un nommage explicite des variables, ou encore par des fonctions, afin de le rendre le plus compréhensible possible.

De plus, comme dans n'importe quel site internet, nous pouvons implémenter quelques améliorations apportant un meilleur confort dans l'utilisation de l'application, pour un patient comme pour un médecin. Nous en avons relevé certaines.

3.2.1 Envoi de mails

En cas de changement d'horaire d'un rendez-vous (que ce soit pour une annulation, un déplacement, ou même une affectation de rendez-vous), on pourrait

envoyer un mail aux personnes concernées par cette modification. Cela éviterai aux patients et aux médecins de se connecter sur l'application tous les jours, simplement pour vérifier qu'il n'y a pas de changement dans leur rendez-vous.

3.2.2 Recherche de médecins

Il se peut que certaines adresses renseignées par les utilisateurs ne soient référencées par l'API *Adress2Coordonnes* (par exemple des petites ruelles). Dans l'idéal, pour ces cas là, on pourrait d'abord regarder si l'adresse est identifiée, puis si elle ne l'est pas, alors on déplacerait le positionnement à l'adresse référencée la plus proche.

Concernant également la recherche de médecins, une recherche par position ponctuelle pourrait être implémentée. C'est à dire que l'utilisateur pourrait trouver les médecins aux alentours de la position qu'il aurait pointée directement sur la carte.

3.2.3 Compatibilité

Pour des raisons esthétiques, le site n'est pas responsive : il n'est pas compatible sur téléphone. Il aurait fallu un design complètement différent pour utiliser l'application sur mobile, notamment pour l'affichage des plannings. Cela constitue donc une extension possible de l'application.

4 Conclusion

Nous sommes fiers d'avoir terminé ce projet et d'avoir rempli les spécifications. Le produit final remplit les tâches auxquelles il était destiné, tout en étant ergonomique et esthétique.

Ce projet nous aura permis de mener la réalisation d'un site web de bout en bout, de la conception aux tests, en partant de spécifications simples et définies. Nous avons également appris à utiliser de nouvelles technologies seuls, et nous avons bénéficié d'un espace dans lequel mettre en application les connaissances acquises au cours de notre licence. Le travail en trinôme s'est très bien déroulé, les tâches ont été réparties de manière équitable et notre collaboration fut fructueuse et agréable.

Nous espérons que l'application deviendra accessible à tous, ainsi nous pourrions recueillir un maximum d'informations et les remettre à l'ARS, pour pallier les déserts médicaux, ou simplement faciliter le travail des médecins. Dans le cas contraire, le site web pourra être remis à d'autres étudiants, comme nous, afin de lui apporter toutes les améliorations envisageables.

References

- [1] Leaflet. Leaflet, a javascript library, <https://leafletjs.com/>.
- [2] php.net. Php: date - manual, <http://php.net/manual/fr/function.date.php>.
- [3] StackOverflow. Get latitude and longitude per adress, <https://stackoverflow.com/questions/15919227/get-latitude-longitude-as-per-address-given-for-leaflet>, 2013.

Résumé

Ce rapport présente le travail effectué pour le projet annuel de licence de Corentin Borde, Guillaume Ségard et Marie Smolinski. Il s'inscrit dans le cadre de notre cursus de 3ème année de Licence Informatique label CMI (Cursus de Master en Ingénierie) à l'Université de Franche-Comté en 2018-2019. Encadré par M. Hassan Mountassir, enseignant chercheur, ce projet est une application web ayant pour but de permettre la prise de rendez-vous chez le médecin spécialiste ou généraliste le plus proche dans un rayon géographique donné, et la gestion ergonomique de leurs rendez-vous par les médecins.

Mots clés : CMI, licence, application, géolocalisation, médecin, prise de rendez-vous

Abstract

This report presents the work realised for the annual Bachelor's Degree project of Corentin Borde, Guillaume Ségard and Marie Smolinski. It's part of our Computer Science CMI Bachelor's Degree's 3rd year's program in Franche-Comté's university (years 2018-2019). Supervised by doctor Hassan Mountassir, this project is a web application that is meant to make appointments to the closest geolocated specialist or generalist doctor in a specified radius, and the doctor's appointments easy management.

Key words : CMI, Bachelor's Degree, app, geolocation, doctor, appointment making