**DOSSIER DE SYNTHESE**

**OsteoClic**





Thomas ROESS

Développeur WEB/WEB Mobile

Elan 2020

Table des matières

[A propos de moi 3](#_Toc52374153)

[Le projet 4](#_Toc52374154)

[Pourquoi ce projet ? 4](#_Toc52374155)

[Présentation du projet 4](#_Toc52374156)

[Le cahier des charges 5](#_Toc52374157)

[La charte graphique 6](#_Toc52374158)

[Les compétences couvertes par le projet 8](#_Toc52374159)

[Activités type 1 : FRONT END 8](#_Toc52374160)

[Activités type 2 : BACK END 8](#_Toc52374161)

[Le RGPD 9](#_Toc52374162)

[Le développement du projet 11](#_Toc52374163)

[Technologies utilisées 11](#_Toc52374164)

[Le MCD et MLD 13](#_Toc52374165)

[Explication du MCD 14](#_Toc52374166)

[Les maquettes 15](#_Toc52374167)

[La liste des patients 15](#_Toc52374168)

[Les détails du patient et consultation 16](#_Toc52374169)

[Design pattern 17](#_Toc52374170)

[Qu’est-ce qu’un MVP ? 17](#_Toc52374171)

[Le MVP dans mon application 18](#_Toc52374172)

[La sécurité 21](#_Toc52374173)

[Gestion des consultations 22](#_Toc52374174)

[Axes d’amélioration 27](#_Toc52374175)

[Traduction 28](#_Toc52374176)

[Pour conclure 30](#_Toc52374177)

[Difficultés rencontrées 30](#_Toc52374178)

[Remerciements 30](#_Toc52374179)

[Conclusion 31](#_Toc52374180)

[Elan 31](#_Toc52374181)

# A propos de moi

Je m’appelle Thomas, j’ai 31 ans et j’ai décidé de reprendre ma vie en main en venant suivre la formation de développeur web/web mobile chez Elan Formation.

J’ai toujours été dans l’informatique depuis que je suis petit, j’ai découvert mon amour pour la programmation et la logique durant mon cursus de Bac Pro ELEEC où l’on a appris les organigrammes et les algorithmes de base pour programmer des machines.

Plusieurs années après, je me suis redirigé vers un BTS informatique de gestion et j’ai commencé un cycle d’ingénieur en informatique. Malheureusement je n’ai pas pu le finir à cause de problèmes de santé.

Depuis que je me suis remis sur pied je n’ai qu’une envie, de rentrer dans la vie active en faisant quelque chose qui me plaise. J’ai trouvé cette formation chez Elan qui m’a permis de repartir sur de bonnes bases et d’approfondir mes connaissances dans les langages web.

J’ai désormais plusieurs projets à présenter aux entreprises pour leurs prouver ma motivation, grâce à cette formation chez Elan.

# Le projet

## Pourquoi ce projet ?

Ce projet m’est venu lors d’échanges avec un utilisateur de de logiciels gratuits d’ostéopathie type « Libreosteo». En effet, cette personne rencontrait de nombreux problèmes lors de l’utilisation de ces outils. Il lui était impossible de le personnaliser, les fonctionnalités ne correspondaient pas à ses besoins spécifiques (certaines lui étaient totalement inutiles alors que d’autres lui manquaient cruellement. Et last but not least, la sécurité était déficiente -base SQLite sans mot de passe-).

C’est pour répondre à ces problématiques que j’ai commencé à réfléchir à une application sur mesure. Il s’agissait de construire un projet qui comprendrait une sécurité renforcée grâce à MySQL et Symfony, où seuls les renseignements nécessaires au client alimenteraient la base de données. A cela, je pensais proposer la possibilité de modifier ou d’ajouter des fonctionnalités à la demande sans devoir attendre une éventuelle mise à jour d’un logiciel global.

## Présentation du projet

L’objectif de mon projet est de créer une application web personnalisée avec le client concerné. Les avantages d’une application sur mesure pour un client sont nombreux, à l’instar d’une intervention rapide en cas de problèmes informatiques ou encore l’évolution du logiciel en fonction des demandes.

Dans ce projet, la demande de ma tutrice était de disposer d’un outil lui permettant de gérer sa patientèle. Elle me demandait la possibilité d’enregistrer tous ses anciens et nouveaux patients, l’accès rapide à la liste de l’ensemble de ses patients mais également la gestion des dossiers de consultation. Pour finir, elle souhaitait introduire une gestion des droits d’utilisateur pour masquer certaines informations sensibles à d’autres utilisateurs tels que des remplaçants.

### Le cahier des charges

Pour mettre en place le cahier des charges, ma tutrice et moi avons réfléchi à toutes les fonctionnalités dont elle pourrait avoir besoin en nous inspirant du logiciel qu’elle utilisait actuellement. Nous avons établi une liste des taches les plus importantes en y ajoutant de nouvelles fonctionnalités petit à petit.

* L’utilisateur doit pouvoir accéder à la liste des patients
* L’utilisateur peut faire une recherche par nom, prénom, adresse, numéro de téléphone
* L’utilisateur peut voir le détail d’un patient
* L’utilisateur peut ajouter de nouveaux patients
* L’utilisateur peut créer une nouvelle consultation
* L’utilisateur peut accéder aux anciennes consultations
* Le remplaçant ne peut pas voir les données sensibles, telles que l’adresse et le numéro de téléphone
* L’utilisateur peut choisir dans quel cabinet il se trouve
* L’utilisateur principal peut modifier les utilisateurs et leurs droits
* L’utilisateur principal peut modifier, ajouter, ou supprimer les informations des cabinets

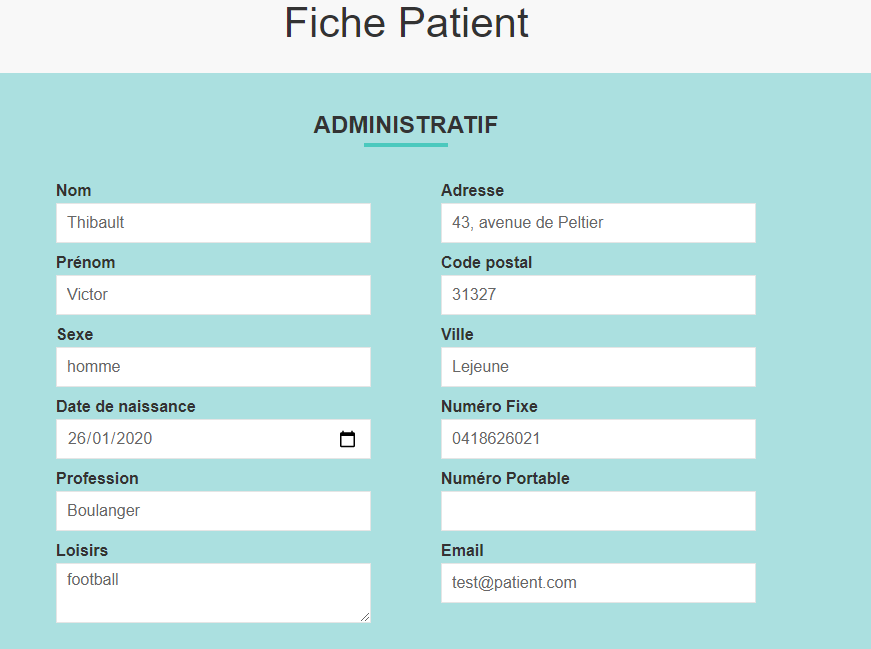
## La charte graphique





Ayant déjà créé un site vitrine pour ma cliente, j’ai voulu réutiliser la charte graphique que j’avais mise en place.

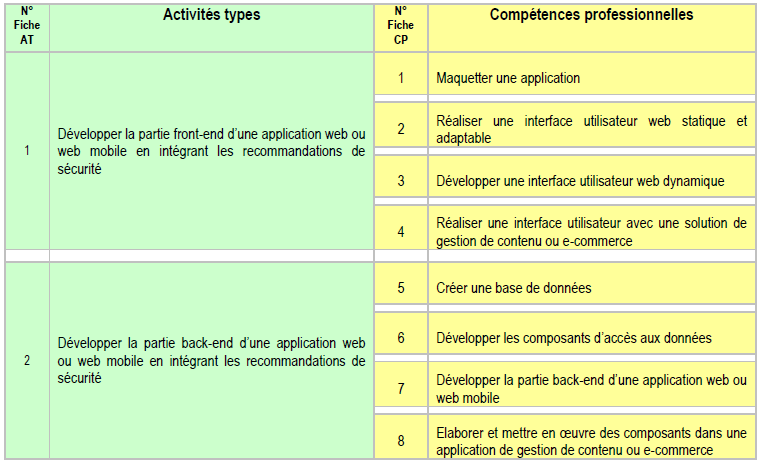
Nous avions retenu dès la création du site vitrine une combinaison d’orange (#f05f40) et de sarcelle (#12a09c), que je reprends et adapte dans la page du patient. L’application est déclinée dans des variantes plus claires : pour l’orange (#f1ab9c) et le bleu (#abe0e0). Le bleu représente souvent le médical et correspond actuellement à la couleur de sa table de soin. L’orange est symbole de vitalité, ce qui nous a semblé cohérent par rapport à la philosophie d’un site à vocation médicale. Il est de plus complémentaire par rapport au bleu.





Je réutilise également la même structure en section et une mise en forme équivalente. J’ai pour projet de lier l’application et le site en créant une prise de rendez-vous personnalisée.

## Les compétences couvertes par le projet



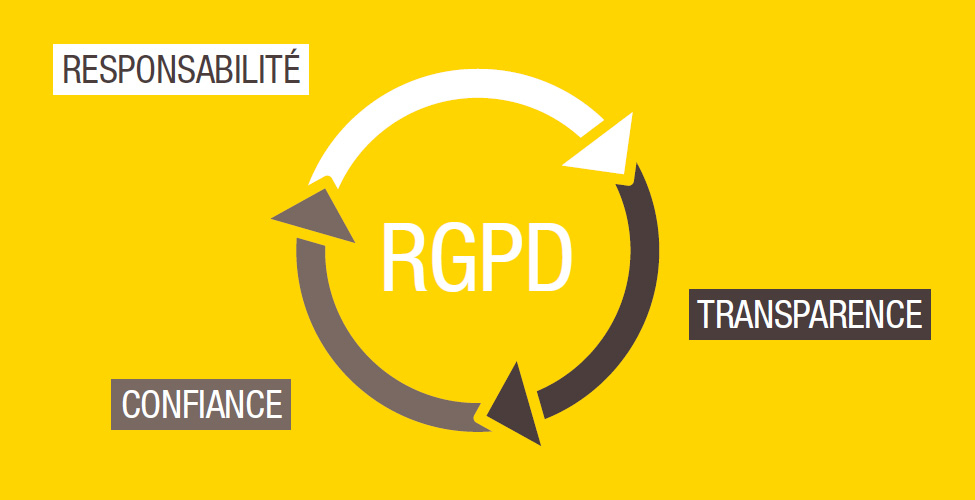
### Activités type 1 : FRONT END

1. **Le maquettage** a été effectué au début du projet selon la demande du client et a ensuite évolué avec le temps en ajoutant de nouvelles fonctionnalités. Il est la base d’une bonne mise en forme et permet de gagner du temps durant les premières parties de la confection du site.
2. Une **Interface adaptable** permet l’utilisation via l’ordinateur ou la tablette du praticien, afin de pouvoir gérer ses consultations depuis ses cabinets ou le domicile de ses patients.
3. Une **interface dynamique** s’adapte à l’utilisateur, un remplaçant n’aura pas accès à toutes les données telles que l’adresse ou le numéro de téléphone, par exemple. Un champ de recherche permet aussi de filtrer des patients selon certains critères (nom, prénom, numéro de téléphone, adresse, cabinet de rattachement).

### Activités type 2 : BACK END

1. La **base de données** MySQL permet un stockage plus sécurisé des données médicales, elle a évolué durant le projet pour refléter les nouvelles demandes du client.
2. Les **accès à la base de données** sont gérés grâce à l’ORM Doctrine qui utilise le DQL (Doctrine Query Langage), il permet d’utiliser les entités pour créer les requêtes et fournir les données sous forme d’objet.
3. La partie **back-end** du projet est développée à l’aide de Symfony 5 et utilise différents langages comme PHP7, JS, TWIG. Différents bundles ont aussi été utilisés pour gagner du temps sur le développement, en particulier Knp\_Paginator et FullCalendar.js

## Le RGPD



Le **RGPD** est le « Règlement Général sur la Protection des Données » mis en place par le Parlement Européen pour protéger les données personnelles des utilisateurs. Il permet d’encadrer le traitement et la circulation des données sur le territoire européen.

Le RGPD s’applique à tout organisme, quelle que soit sa taille, traitant des données personnelles pour son compte ou non, dès lors qu'il est établi sur le territoire de l'Union Européenne ou que son activité cible directement des résidents européens.

Ce projet se déroule dans un cadre médical et comporte donc des spécificités relatives au domaine médical. Les données collectées sur les patients doivent être**adéquates, pertinentes et limitées**à ce qui est strictement nécessaire à la prise en charge du patient au titre des activités de prévention, de diagnostic et de soins. Les données doivent également être stockées pendant une période déterminée, le Conseil National de l’Ordre des Médecins préconise de s’aligner sur les délais de conservation prévus pour les dossiers médicaux :

* 20 ans à compter de la date de la dernière consultation du patient
* Si le patient est mineur et que ce délai de 20 ans expire avant son 28ème anniversaire, la conservation des informations le concernant doit être prolongée jusqu'à cette date
* Dans tous les cas, si le patient décède moins de 10 ans après sa dernière consultation, les informations le concernant doivent être conservées pendant 10 ans à compter de la date du décès
* En cas d’action tendant à mettre en cause la responsabilité du médecin, il convient de suspendre ces délais de conservation.

La CNIL estime légitime de collecter certaines catégories de données personnelles :

* Les données d’identification : nom, prénom, date de naissance, adresse, numéro de téléphone.
* Selon les contextes, la vie professionnelle : profession, conditions de travail.
* La santé : historique médical, historique des soins, diagnostics médicaux, traitements prescrits, nature des actes effectués, résultats d’examens de biologie médicale et tout élément de nature à caractériser la santé du patient et considéré comme pertinent par le médecin.
* Informations relatives aux habitudes de vie : si collectées avec l’accord du patient et dans la stricte mesure où elles sont nécessaires au diagnostic et aux soins.

# Le développement du projet

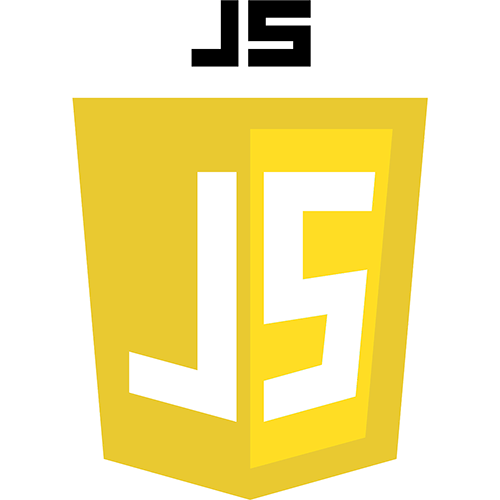
## Technologies utilisées

**PHP** Hypertext Preprocessor est un langage de scripts généraliste et Open Source, spécialement conçu pour le développement d'applications web. Il a été utilisé pour la principale partie du back end du projet.



**HTML** signifie « *HyperText Markup Language* ». Il est utilisé afin de créer et de représenter le contenu d'une page web et sa structure.



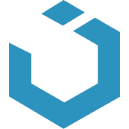
**CSS** signifie « *Cascading Style Sheets »*. C’est un langage de styles. Il permet d'apporter du design au contenu HTML intégré. CSS est capable de changer tous les aspects de design d'une page Web.

**JavaScript**est un langage de programmation qui permet de créer du contenu mis à jour de façon dynamique. Il a principalement été utilisé pour la gestion dynamique des formulaires de consultation.

**Twig**est un moteur de templates pour le langage de programmation PHP, utilisé par défaut par le framework Symfony.



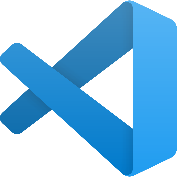
**Symfony** est un puissant Framework PHP développé par une société française : SensioLabs. Il permet de réaliser des sites internet dynamiques de manière rapide, structurée, et avec un développement clair.

****

**UIKit** est un framework front-end pour faciliter le développement



**MySQL** est un serveur de bases de données relationnelles SQL développé dans un souci de performances élevées en lecture

**VSCode** est un environnement de développement multi-langages et multi-plateformes, intégrant de nombreux outils facilitant l’écriture d’une application quel(s) que soi(en)t le(s) langage(s) employé(s).

******Looping** est un logiciel gratuit et d’usage libre créé par l’Université de Toulouse III, Looping offre des outils de modélisation conceptuelle de données et a permis de réaliser les MCD et MLD présentés dans ce dossier.

**Github** est une plate-forme communautaire d’archivage et de partage de projets, intégrant une gestion de version basée sur l’outil Git.



**Laragon** est un outil qui permet créer un serveur virtuel pour pouvoir développer avec du PHP, il s’occupe aussi de lancer la base de données dans le serveur virtuel.

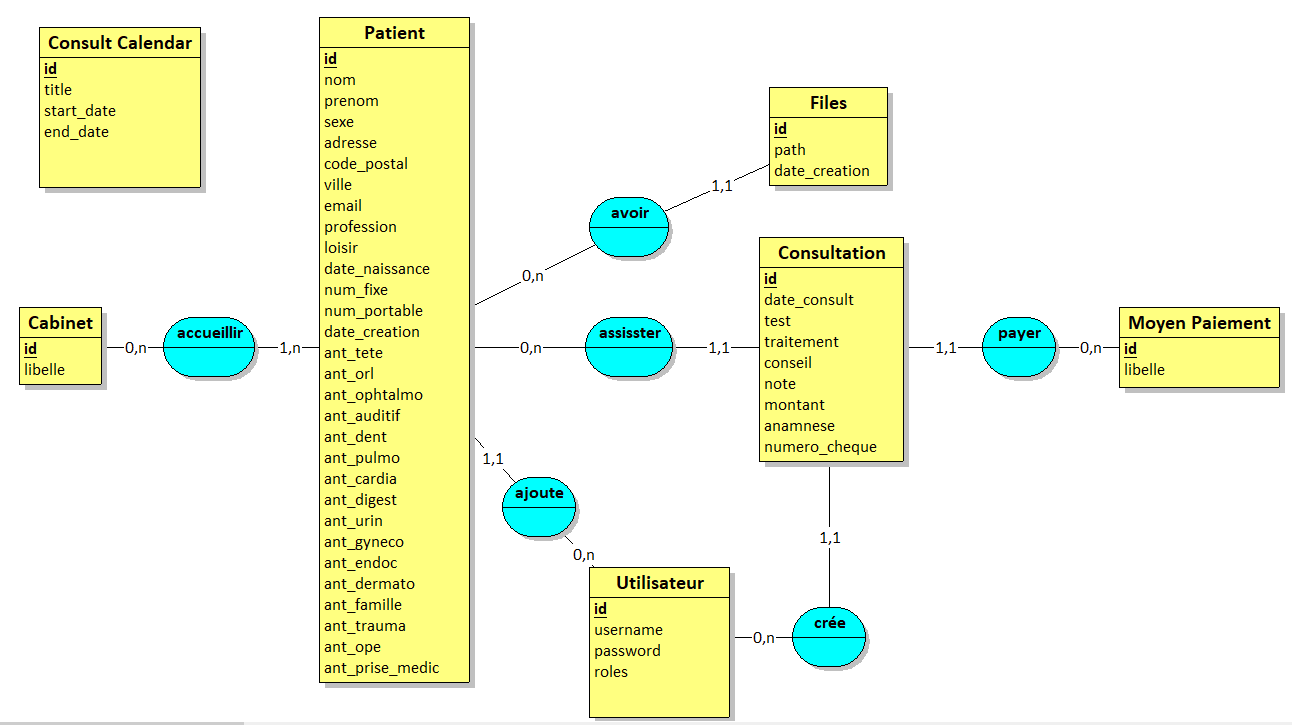


**Composer** est le manageur de package, il s’occupe de gérer et d’installer les bundles dans symfony.



**Doctrine** est un ORM (couche d'abstraction à la base de données) pour PHP. Il s’occupe de traduire les entités dans le code en données lisible pour la base de données, et inversement il transforme les données de la DB en objet.

## Le MCD et MLD



**Utilisateur = (id *COUNTER***, username *VARCHAR(30)*, password ***VARCHAR(255)***, roles *TEXT***);**

**Patient = (id *COUNTER***, nom *VARCHAR(50)*, prenom *VARCHAR(50)*, sexe *VARCHAR(100)*, adresse *VARCHAR(150)*, code\_postal *VARCHAR(10)*, ville *VARCHAR(200)*, email *VARCHAR(50)*, profession *VARCHAR(50)*, loisir *TEXT*, date\_naissance *DATE*, num\_fixe *VARCHAR(20)*, num\_portable *VARCHAR(20)*, date\_creation *DATETIME*, ant\_tete *TEXT*, ant\_orl *TEXT*, ant\_ophtalmo *TEXT*, ant\_auditif *TEXT*, ant\_dent *TEXT*, ant\_pulmo *TEXT*, ant\_cardia *TEXT*, ant\_digest *TEXT*, ant\_urin *TEXT*, ant\_gyneco *TEXT*, ant\_endoc *TEXT*, ant\_dermato *TEXT*, ant\_famille *TEXT*, ant\_trauma *TEXT*, ant\_ope *TEXT*, ant\_prise\_medic *TEXT, #Utilisateur.id***);**

**Files = (id *COUNTER***, path *VARCHAR(150)*, date\_creation *DATETIME, #Patient.id***);**

**Cabinet = (id *COUNTER***, libelle *VARCHAR(50)***);**

**Moyen\_Paiement = (id *COUNTER***, libelle *VARCHAR(100)***);**

**Consult\_Calendar = (id *COUNTER***, title *VARCHAR(255)*, start\_date *DATETIME*, end\_date *DATETIME***);**

**Consultation = (id *COUNTER***, date\_consult *DATETIME*, test *TEXT*, traitement *TEXT*, conseil *TEXT*, note *TEXT*, montant *DECIMAL(5,2)*, anamnese *TEXT*, numero\_cheque *VARCHAR(50), #Patient.id, #Utilisateur.id, #Moyen\_paiement.id***);**

**accueillir = (*#Patient.id, #Cabinet.id*);**

### Explication du MCD

Les **utilisateurs** (ostéopathe principal ou ostéopathe remplaçant) sont liés aux **consultations** pour avoir une trace de qui a mené la consultation.

Un **cabinet** peut accueillir plusieurs **patients** tout comme un **patient** peut aller dans différents **cabinets** en fonction de ses disponibilités.

Le **patient** peut bien entendu avoir plusieurs **consultations** tandis que la **consultation** est liée à un **patient**, un **utilisateur,** et un **moyen de paiement**.

Le **patient** est lié à un **utilisateur** pour savoir qui l’a ajouté à la BDD, pour l’affichage des données du remplaçant.

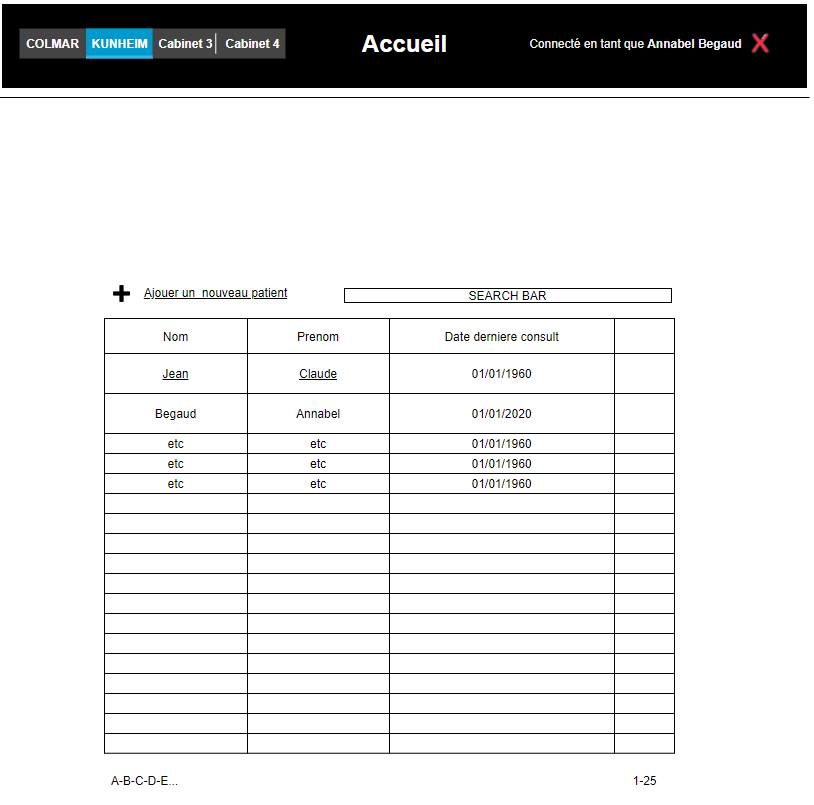
Le **moyen de paiement** est lié à la **consultation**, principalement pour stocker les numéros de chèques et plus tard inclure une comptabilité possible.

Les **fichiers** seront intégrés plus tard, il s’agit de tout ce qui touche aux radio ou comptes rendus médicaux.

La seule table sans relation actuellement est Consult\_Calendar, la table qui gère les évènements de l’agenda. Plus tard, elle sera liée aux patients pour préparer les futures consultations en attente.

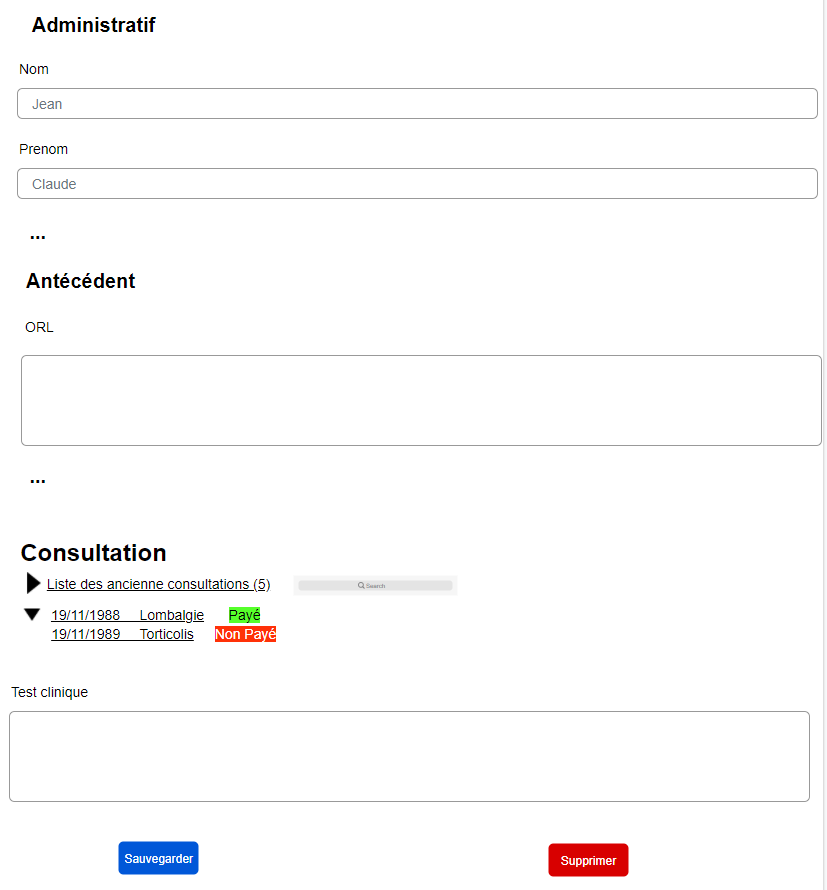
## Les maquettes

### La liste des patients



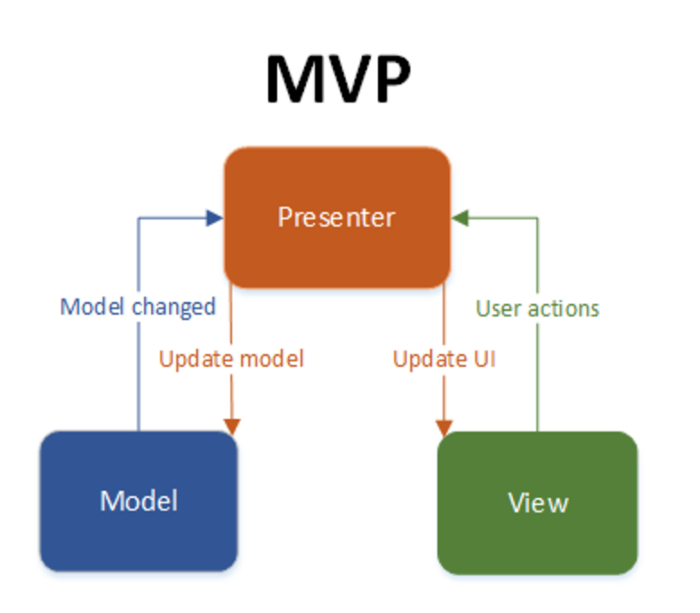
La liste des patients permet de trouver rapidement un patient qui revient en consultation et d’accéder rapidement à ses informations pour lancer une nouvelle séance selon le cabinet où l’on se trouve.

### L**es détails du patient et consultation**



Plusieurs fonctionnalités ont été changées durant la programmation, après discussions avec le client, plusieurs idées apportées n’étaient pas forcément réalisables ou simplement ne rendaient pas l’application ergonomique. Les emplacements de plusieurs champs ont été modifiés, de nouveaux champs ont été ajoutés pour correspondre à la réalité de l’application.

## Design pattern



### Qu’est-ce qu’un MVP ?

Le MVP (Model View Presenter) est un patron d'architecture, considéré comme un dérivé du patron d'architecture Modèle-Vue-Contrôleur. Il garde les mêmes principes que le MVC sauf qu'il élimine l'interaction entre la vue et le modèle. En effet, elle sera effectuée par le biais de la présentation, qui organise les données à afficher dans la vue.

* Le **Model** représente la base de données, il envoie les données nécessaires au presenter
* La **View** représente la vue, c’est l’affichage des données et des pages HTML
* Le **Presenter**, présentation en français, contient la logique de l’application. C’est lui qui régit toutes les interactions entre le modèle et la vue.

### Le MVP dans mon application

Le **Model** interagit avec la base de données, son rôle consiste à envoyer ou à chercher les données demandées pour les renvoyer sous un format configurable et exploitable par le **Presenter**.

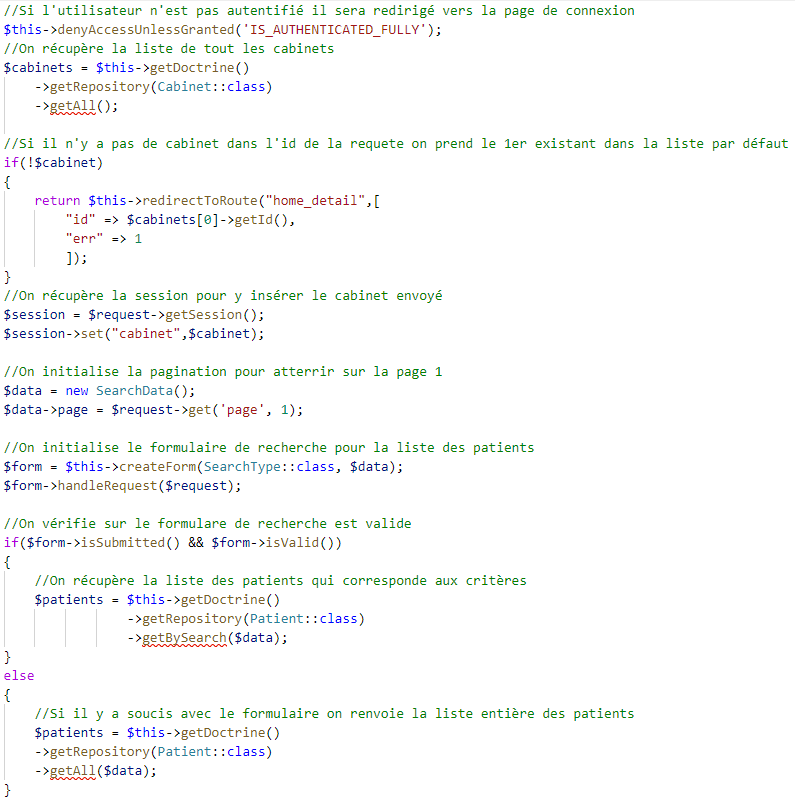
Une fois les données récupérées depuis la BDD, l’ORM (Object Relational Mapping) les traduit en objets correspondant aux configurations de l’entité en question. Il en fait de même pour l’écriture, il traduit les objets reçus pour créer une requête SQL.

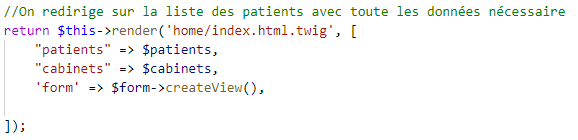
Dans le cas de mon application, j’utilise Doctrine et le DQL (Doctrine Query Language) pour utiliser mes entités.



On peut voir dans la copie d’écran ci-dessous une partie de l’entité Patient et les annotations qui seront utilisées par l’ORM pour la traduction en SQL :

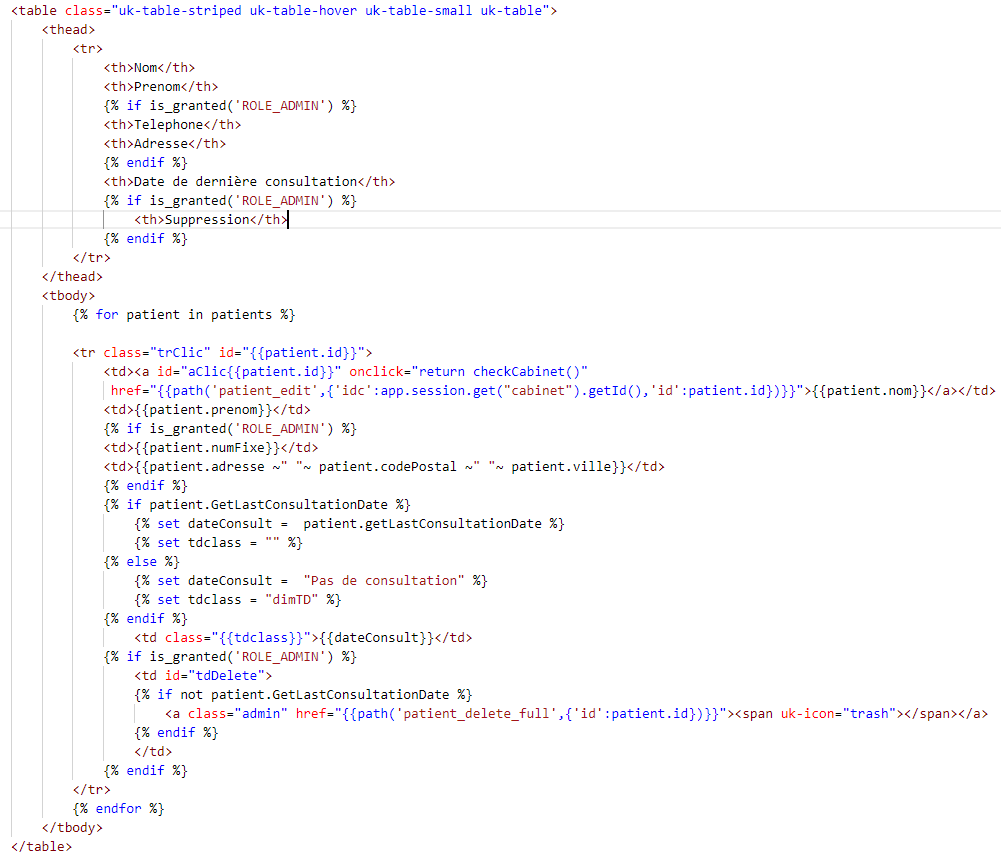
Le **Presenter** sert d’intermédiaire entre les **Views** et les **Models**, c’est là que les vérifications de sécurité sont faites, que les données de la BDD sont récupérées ou envoyées. Puis il s’occupe de les envoyer dans la vue correspondante à la requête de l’utilisateur.

Ici le HomeController se charge d’afficher la liste des patients et de gérer les recherches, ensuite il retourne les données demandées dans la vue correspondante.



La **View** s’occupe d’organiser les données à l’écran. Son rôle est d’afficher toutes les informations récupérées par le **Presenter** et de les mettre en forme pour l’utilisateur. La **View** est principalement formée par du code HTML aidé par TWIG pour traiter les données reçues du **Presenter**.

Dans le cas de l’application, le **Presenter** renvoie la liste des patients et la **View** s’occupe de la traiter sous la forme d’un tableau HTML. Grâce à TWIG, nous pouvons utiliser les boucles pour la mise en forme et gérer l’affichage des données en fonction des droits d’administration.

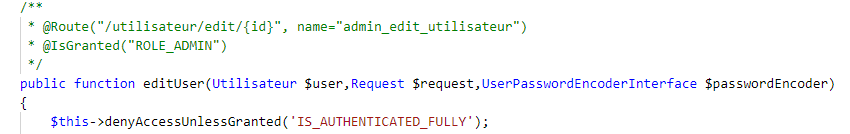


## La sécurité

La sécurité est un point très important dans ce projet, pour protéger les données personnelles et médicales des patients. J’ai donc appliqué plusieurs couches de protection en plus de celles intégrées par Symfony.

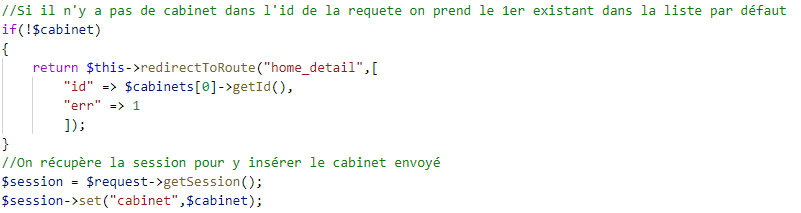
La première couche rencontrée est le filtrage IP dans le fichier security.yaml, si vous n’êtes pas dans la liste des IP autorisées, vous ne pourrez pas accéder à l’application. Dans le même fichier se trouve la gestion des rôles, il y en a deux principaux : ADMIN et USER. L’ostéopathe principal sera géré comme ADMIN, il pourra accéder à la gestion des cabinets et des utilisateurs, tandis que l’USER pourrait être un remplaçant et ne pourra pas y accéder. Il ne pourra ainsi pas voir les informations de contacts des patients déjà présents dans la base, pour ne pas se les approprier.

La détection de rôle se passe dans les **Controllers** grâce aux annotations de routes, si l’utilisateur n’est pas administrateur, il tombera sur une page 403 personnalisée.



Sur la copie d’écran ci-dessus, on peut aussi voir la protection d’accès si l’utilisateur n’est pas connecté ou si la session a expiré. Il sera automatiquement redirigé sur la page de connexion pour rafraîchir sa session ou simplement se connecter.

De même, si l’utilisateur tape une adresse qui ne correspond à rien dans l’application, il tombera sur une page 404 personnalisée avec possibilité de retourner à l’accueil.



Grâce à ce système qui se trouve dans le HomeController, je fais en sorte qu’il y ait toujours un cabinet valide de sélectionné. Si l’utilisateur tape un id de cabinet inexistant, il sera redirigé sur la page principale et pourra choisir le bon cabinet où il fera sa consultation. J’enregistre aussi le cabinet sélectionné dans la session pour un accès rapide (par ex. les liens de retour à l’accueil pour rester dans le même cabinet). Par contre, si l’utilisateur tape un numéro de patient inexistant, il arrivera simplement sur la page de création d’un nouveau patient.

Symfony intègre des protections pour plusieurs failles connues du PHP : les failles CSRF, les failles XSS, et les injections SQL.

Les failles **CRSF** (Cross-Site Request Forgery) consistent à faire effectuer une action par un utilisateur du site sans qu’il ne s’en aperçoive. Ces failles sont gérées grâce à l’utilisation d’un jeton généré à la création de la session. Il est différent pour tous les utilisateurs. Le jeton est présent dans chaque formulaire qui contiennent des informations importantes (absent dans un formulaire de recherche par ex) et le compare à celui de l’utilisateur en session.

Les failles **XSS** (Cross Site Scripting) consistent, elles, à injecter du code HTML ou Javascript dans une page en passant par les formulaires ou l’url pour atteindre des variables mal protégées. Grâce à TWIG, les champs sont échappés et ignorent le code injecté.

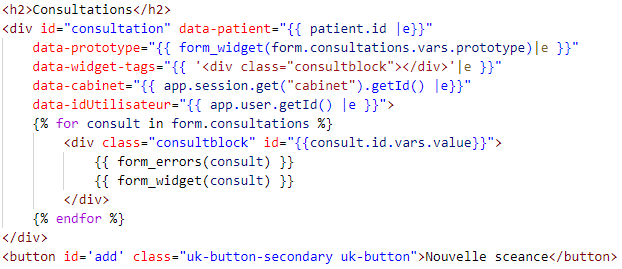
Les **Injection SQL** consistent à modifier la requête envoyée à la base de données pour accéder à des données non autorisées ou bien pour supprimer simplement la base. Elles sont gérées grâce à Doctrine qui permet d’invalider le code SQL via la fonction « prepare »

## Gestion des consultations

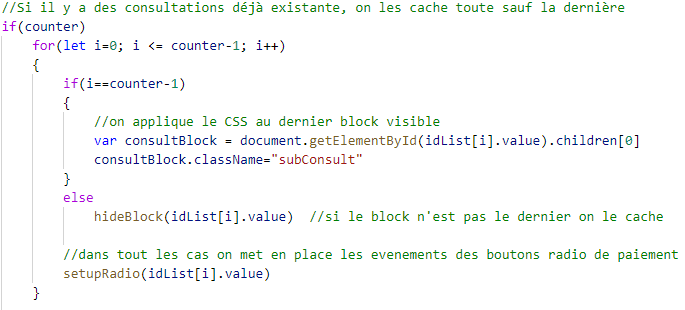
La gestion des consultations a été un axe important de mon projet et est principalement gérée en Javascript.

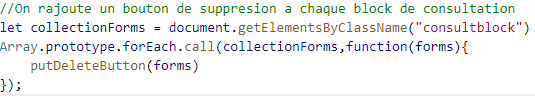
Le bloc de consultation est un ensemble de champs géré en tant que CollectionType sous Symfony qui arrive dans la fiche « patient ». Le principe est simple, on presse le bouton « Nouvelle séance ».

Le nouveau bloc de consultation apparaît avec tous les champs et la séance peut commencer.

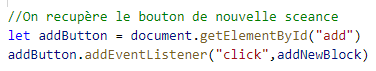


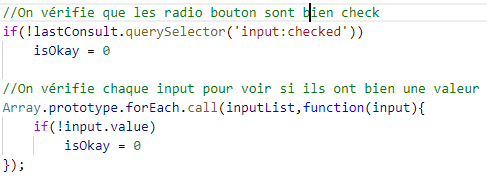
Dans le HTML est enregistré le prototype et toutes les informations nécessaires à la création d’un nouveau bloc. Le prototype représente l’ensemble du formulaire à afficher, il est généré par le formType lié aux consultations. Le widget tag sera le conteneur de chaque consultation, le prototype sera inséré à chaque nouveau bloc. On retrouve également l’id du patient pour le récupérer dans le Javascript qui, lui, gèrera le champ Select présent pour la relation entre le patient et la consultation. L’idUtilisateur est utilisé pour relier la consultation à l’utilisateur (titulaire ou remplaçant). La boucle charge ensuite tous les blocs de consultation présents en base de données pour cet utilisateur.

Le Javascript entre en scène et commence à traiter les blocs pour les rendre plus interactifs, il s’occupe de placer les bons ID aux bons endroits puis rajoute un bouton de suppression à la fin de chaque bloc.

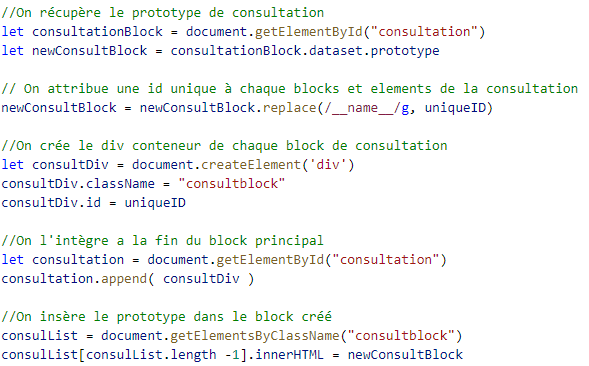


On vérifie ensuite s’il y a des blocs déjà existants et on les cache sauf le dernier pour pouvoir visionner rapidement la dernière consultation et les soins appliqués. La fonction *hideBlock* s’occupe également de mettre en place le bouton *show/hide* et lui ajouter les événements de *click*, d’assigner une classe pour le CSS, et enfin d’ajouter la date et l’auteur de la consultation. On met en place les *eventListeners* dans les boutons radios du moyen de paiement pour cacher ou afficher le champ « numéro de chèque ».

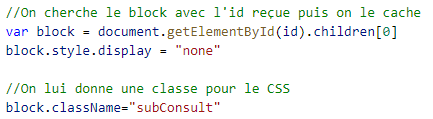




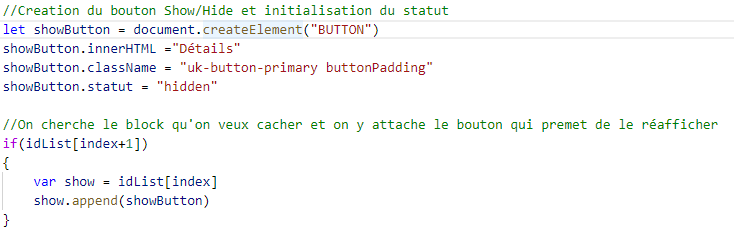
Et finalement, on récupère le bouton « nouvelle séance » et on lui attache la fonction d’ajout de nouveau bloc de consultation. Cette fonction a été énormément modifiée et travaillée au cours du développement. A chaque nouvelle idée, nouvelles options d’optimisation, elle a été modifiée pour rendre l’utilisation adéquate pour le praticien. Bien sûr, il reste toujours de nouveaux moyens de l’améliorer !

A l’intérieur de la fonction, j’ai ajouté une vérification pour ne pas ouvrir plusieurs nouvelles consultations tant que tous les champs obligatoires ne sont pas remplis, pour éviter, par exemple, de cliquer plusieurs fois sur le bouton « nouvelle séance » par erreur et ne pas pouvoir valider la consultation car l’ancien bloc vide est caché. J’ai également mis en place une vérification pour ne pas avoir de soucis d’affichage quand on crée plusieurs consultations de suite sans enregistrer, principalement pour la date de création qui est générée via les entités.

Grâce au prototype généré par Symfony qu’on a stocké dans les *datasets* nous pouvons recréer facilement de nouveaux blocs de consultation ! Un ID unique est généré pour éviter les différents conflits de nommage que j’ai pu rencontrer tout le long de la programmation de ces fonctions en Javascript. Il est inséré dans chaque élément du prototype. Puis, je crée le conteneur de la consultation et lui attribue l’id unique, une fois cela fait, je l’attache à la fin du bloc principal de consultation et intègre le prototype modifié.

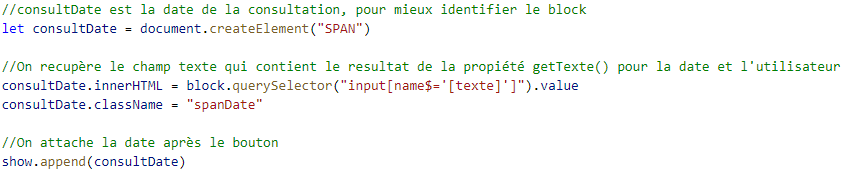


Je lui ajoute ensuite le bouton de suppression, la classe pour le CSS, j’insère le bon patient et le bon utilisateur, le setup des bouton radio est également fait. Le nouveau bloc est prêt à l’emploi, mais il faut encore cacher l’ancien bloc et mettre en place le système pour l’afficher ou le cacher.

J’ai choisi de cacher le premier *ChildNode* plutôt que le bloc entier en raison de la méthode d’ajout des boutons de suppression et d’affichage. De cette manière, je peux afficher facilement une ligne comprenant l’intitulé, la date, l’auteur de la consultation, ainsi que les deux boutons.



Une fois le bouton d’affichage créé, on lui insère toutes les informations nécessaires au bon fonctionnement et à l’affichage puis on l’ajoute à la fin du block de consultation. Je procède ensuite à l’ajout de la fonction évenementielle qui permet l’alternance entre l’affichage et la dissimulation du bloc.



On crée ensuite le réceptacle destiné à recevoir la date de création de la consultation et l’auteur, le texte est enregistré dans un input rempli par une fonction dans l*’entity* et affiché par le formulaire ConsultationType , on lui attribue une classe et on l’attache après les boutons.



Voilà à quoi ressemble la consultation cachée après le traitement avec un délimiteur pour bien différencier chaque bloc.

# Axes d’amélioration

Il reste plusieurs fonctionnalités que je souhaite encore ajouter ou améliorer à l’application

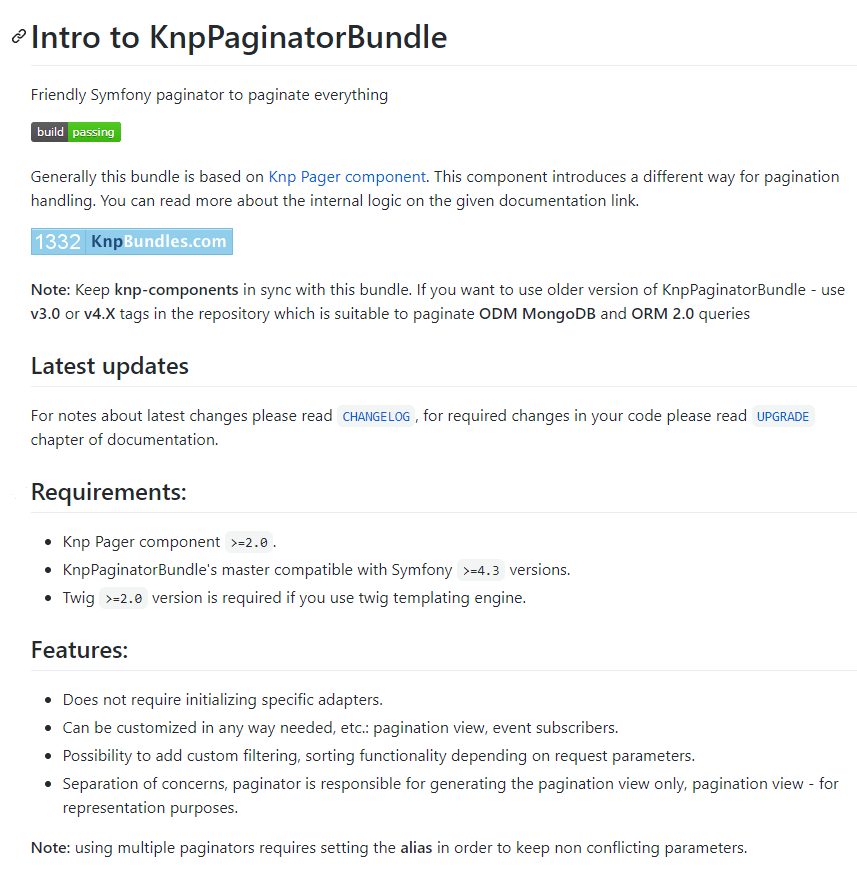
**Ajout :**

* L’agenda n’est pour l’instant lié à aucun patient et sert uniquement à avoir un aperçu des séances pour préparer ses consultations à l’avance. J’aimerais plus tard créer la prise de rendez-vous avec création automatique de l’évènement dans le calendrier avec la gestion des absences
* La gestion des fichiers de type « radio » ou autre sera ajoutée dans la page du patient
* Une possible aide à la comptabilité avec un résumé par cabinet qui pourrait être ajouté dans une section dédiée
* Un ajout de statistiques d’afflux de patients par cabinet sur la durée
* Archivage automatique des données en fonction du RGPD, les données médicales du patient doivent être archivée après 20 ans depuis la dernière consultation

**Modification :**

* Toutes les pages ne sont pas encore responsives, petit à petit je ferais en sorte qu’elles soient toutes utilisable depuis une tablette, pour les visites à domicile
* Je compte améliorer le CSS des pages secondaires, c'est-à-dire toutes les pages qui ne sont pas visitée à chaque utilisation, comme la gestion des cabinets ou des compte utilisateurs
* Le CSS des pages principales sera modifié petit a petit pour améliorer l’ergonomie et rendre les infos rapides à trouver au premier coup d’œil
* Je compte aussi améliorer le CSS des consultations pour être sûr que l’utilisation est optimale et les infos rapide à trouver

# Traduction



**Introduction à KnpPaginatorBundle**

C’est un système de pagination facile d’utilisation pour tout paginer.

Généralement ce bundle est basé sur « Knp Pager component ». Ce composant introduit une façon différente de manipuler la pagination. Vous pouvez en apprendre plus à propos de la logique interne dans le lien de documentation donné.

**Note :** Gardez **knp-components** synchronisé avec ce bundle. Si vous voulez utiliser une version antérieure de KnpPaginatorBundle – utilisez les tags **V3.0** ou **V4.X** dans le dépôt correspondant à la pagination des requêtes **ODM MongoDB** et **ORM 2.0**.

**Dernières mises à jour**

Pour les notes à propos des dernières mises à jour, veuillez lire le **CHANGELOG**, pour les changements obligatoires dans votre code, veuillez lire le chapitre **UPGRADE** de la documentation.

**Prérequis**

* Composant Knp Pager >= 2.0
* La branche master de KnpPaginatorBundle compatible avec des versions >=4.3 de Symfony.
* Une version >=2.0 de Twig est requise si vous utilisez le moteur de rendu de Twig.

**Fonctionnalités**

* Pas besoin d’un adapteur spécifique pour l’initialisation
* Peut être personnalisé de n’importe quelle façon, etc : vue de la pagination, créateurs d’évènements
* Possibilité d’ajouter des filtres personnalisés, fonction de tri et des paramètres de la requête.
* Séparation des préoccupations, Paginator est responsable uniquement de la génération de la vue de la pagination et de présentation

Note : L’utilisation de plusieurs systèmes de pagination demande de configurer des alias afin d’éviter tout conflit de paramètres.

# Pour conclure

Difficultés rencontrées

Le point positif de ce projet est d’avoir réussi à quasiment tout faire par moi-même pendant le stage, ce projet m’a apporté énormément de connaissances dans plusieurs domaines, notamment sur l’utilisation de Symfony et du Javascript.

La plus grosse difficulté rencontrée durant la programmation de cette application était le manque de référent technique durant une grande partie du développement. Le manque d’expérience s’est fait énormément ressentir sur le temps passé sur certaines fonctions qui auraient pu être réalisées bien plus rapidement.

De même, ce manque d’expérience m’a handicapé sur la planification des tâches et l’évaluation de la durée de programmation global. En effet, il est très difficile de juger sa vitesse de développement et la sous-évaluation du temps de travail nécessaire est très fréquente.

Malgré tout, je suis parvenu au terme de ce projet, et j’en retire à présent un peu plus d’expérience et une certaine fierté.

## Remerciements

Tout d’abord, je souhaiterais remercier Monsieur Virgile GIBELLO et Monsieur Mickaël MURMANN, mes formateurs à Strasbourg, pour avoir réussi à nous enseigner et avoir su s’adapter aux conditions imposées pendant cette période de pandémie, en inventant une nouvelle manière de nous faire les cours en jonglant entre le présentiel et le distanciel.

Je tiens également à remercier Elan Formation de m’avoir permis de participer à cette formation, et merci à l’équipe pédagogique que l’on a malheureusement que très peu vue.

Je suis également reconnaissant envers ma tutrice, Annabel BEGAUD, pour m’avoir soutenu et proposé de participer à des projets enrichissants en matière de nouvelles connaissances. C’est avec enthousiasme que j’ai pu les réaliser et ainsi approfondir mes acquis de la formation.

Un dernier remerciement à ma promotion DL7 pour la bonne ambiance de travail et l’entraide fournie durant toute la formation.

## Conclusion

J’ai été très heureux de pouvoir réaliser ce projet qui m’a énormément apporté de connaissances. C’était un projet qui me tenait à cœur et j’espérais pouvoir le réaliser depuis plusieurs mois. Grâce à Elan formation, j’ai pu acquérir les compétences techniques nécessaires pour pouvoir enfin travailler dessus et sécuriser l’application que ma compagne utilisera pour son travail. Il me reste encore plusieurs fonctionnalités et améliorations que j’aimerais ajouter au projet, et sans doute d’autres se rajouteront à la liste après une longue période d’utilisation.

## Elan

ELAN Formation, c’est plus de 25 ans d’expérience en Intra ou en Inter dans les domaines de la Bureautique, la PAO, le Multimédia, d’Internet, des Techniques de secrétariat.

C’est des formations sur mesure et totalement individualisées. Possibilité de monter un dossier en partenariat avec des OPCA (Organismes Paritaires Collecteurs Agréés) et optimiser ainsi les recherches de financement.

ELAN est un organisme de formation local. A ce titre, Elan dispose de locaux sur [STRASBOURG](http://elanweb.info/elanformation/centres/strasbourg/), [SELESTAT](http://elanweb.info/elanformation/centres/selestat/), [HAGUENAU](http://elanweb.info/elanformation/centres/haguenau/), [SAVERNE](http://elanweb.info/elanformation/centres/saverne/), [COLMAR](http://elanweb.info/elanformation/centres/colmar/), [MULHOUSE](http://elan-formation.eu/centres/mulhouse/), METZ et NANCY pour dispenser des formations individualisées en Inter.