Rapport d’étudiants ingénieurs

Deuxième année

Filière 5 : Réseaux et sécurité informatique

Logiciel de gestion de patrimoine immobilier

*Présenté par* : **Sylvain BESSONNEAU et Clément DIF**

Tutrice ISIMA : Myoung-Ah KANG **03 février 2020**

Campus des Cézeaux. 1 rue de la Chébarde. TSA 60125. 63178 Aubière CEDEX

# Table des matières

[Table des matières 2](#_Toc32840056)

[Table des illustrations 3](#_Toc32840057)

[Remerciements 4](#_Toc32840058)

[Introduction 5](#_Toc32840059)

[Etude générale du projet 5](#_Toc32840060)

[I. Introduction de l’étude 5](#_Toc32840061)

[1. Contexte et activité du client 5](#_Toc32840062)

[2. Analyse du besoin 5](#_Toc32840063)

[3. Objectifs du travail et attentes du client 6](#_Toc32840064)

[II. Matériel et méthodes 6](#_Toc32840065)

[1. Etude du problème et questionnements préliminaires 6](#_Toc32840066)

[2. Architecture de la solution et conception 7](#_Toc32840067)

[3. Mise en œuvre de la solution 9](#_Toc32840068)

[a) Ressources et organisation du travail 9](#_Toc32840069)

[b) Le *backend* ou l’aspect fonctionnel 11](#_Toc32840070)

[c) Le *frontend* ou la relation directe avec l’utilisateur 12](#_Toc32840071)

[III. Résultats et perspectives 12](#_Toc32840072)

[1. Utilisation de la solution 12](#_Toc32840073)

[a) Gestion des locataires 12](#_Toc32840074)

[b) Gestion des locaux 13](#_Toc32840075)

[c) Gestion des contrats 13](#_Toc32840076)

[d) Gestion des documents 14](#_Toc32840077)

[2. Perspectives d’amélioration 15](#_Toc32840078)

[Conclusion 16](#_Toc32840079)

[Résumé 16](#_Toc32840080)

[Abstract 16](#_Toc32840081)

[Glossaire 16](#_Toc32840082)

[Références bibliographiques 16](#_Toc32840083)

[Annexes 16](#_Toc32840084)

# Table des illustrations

[Figure 1 : Représentation schématique de la base de données utilisée au sein du projet 8](file:///G:\REAM\french%20report.docx#_Toc32840085)

[Figure 2 : Logo officiel du logiciel libre Apache HTTP Server créé et maintenu au sein de la fondation Apache. 9](file:///G:\REAM\french%20report.docx#_Toc32840086)

[Figure 3 : MySQL, système de gestion de bases de données relationnelles. 10](file:///G:\REAM\french%20report.docx#_Toc32840087)

[Figure 4 : Représentation schématique de la séparation frontend et backend dans un projet de développement Web. 10](file:///G:\REAM\french%20report.docx#_Toc32840088)

# Remerciements

Nous tenons à remercier notre tutrice de projet Mme Myoung-Ah KANG pour l’aide qu’elle a su nous fournir lors de nos différentes rencontres. Elle aura pu notamment nous guider et nous orienter lors de la création de bases de données et de l’agencement des fichiers de l’utilisateur au sein de notre projet.

Par ailleurs nous portons une attention particulière à M. Vincent MAZENOD et M. Rémi DUBOURGNOUX avec qui nous avons pu échanger au sujet de multiples questionnements que nous avons pu avoir au cours de l’élaboration de notre projet.

Enfin, adressons nos remerciements à Mme Murielle MOUZAT pour l’encadrement de la partie expression et communication et pour nous avoir donné les directions pour la rédaction de notre rapport.

# Introduction

Notre projet est un logiciel de gestion de gestion de patrimoine immobilier. Il comprend ainsi la gestion de contrats, de logements, de locataires ainsi que des documents qui en découlent. Celui-ci a été développé dans le but de répondre au besoin d’un particulier qui gère un parc d’appartements.

Dans une première partie, nous présenterons le projet en détail en analysant le besoin du client ainsi que les objectifs. Puis, nous parlerons des problèmes auxquels nous avons dû faire face, aux solutions trouvées ainsi que du fonctionnement complet du logiciel. Enfin, nous évoquerons les résultats obtenus avec notre logiciel et également les éventuelles perspectives que nous pouvons effectuer pour l’améliorer.

# Etude générale du projet

## Introduction de l’étude

### Contexte et activité du client

Le client est un particulier gérant la location de plusieurs appartements et hangars au sein d’un parc immobilier. Pour mener son activité, il administre les contrats de bail et les formalités administratives manuellement. Cependant, cela lui nécessite actuellement un temps précieux car il lui est obligatoire de remplir ces documents de manière manuscrite à la venue d’un potentiel locataire.

La création de contrats étant également cause de nombreux échanges, le nombre élevé de courriels envoyés pour le partage d’informations entre les deux parties et la gestion qu’il incombe forme, dans une autre mesure, contrainte au propriétaire.

### Analyse du besoin

Lorsque l’on examine le contexte dans sa globalité, l’on peut en effet noter qu’il pourrait être particulièrement judicieux d’offrir l’automatisation du remplissage des documents au propriétaire. Cette complétion informatisée serait alors mise en place grâce à une saisie de toutes les informations nécessaires par l’utilisateur du logiciel.

L’idée de notre projet et la solution qu’il présente est donc effectivement de proposer un tel service à partir de l’entrée d’informations sur le locataire mais également sur le bien destiné à être loué et sur le type de contrat liant le propriétaire et son client.

Evidemment, des solutions similaires existent déjà sur le marché mais celles-ci sont malheureusement trop coûteuses ou non adaptées à une gestion de patrimoine à échelle non professionnelle.

### Objectifs du travail et attentes du client

Dans un souci de gain de temps, le client a émis le souhait d’avoir un logiciel capable de gérer des locataires, des locaux et des contrats de location de manière détaillée et d’informatiser l’ensemble des données pour une dématérialisation complète.

Le logiciel que nous devons mettre en place doit en conséquence être capable de créer des locataires à partir d’un formulaire, de les modifier, de les rechercher, de les afficher et de leur assigner un contrat tout cela au travers d’une interface graphique. Par ailleurs, la solution doit également permettre une gestion similaire de locaux en autorisant leur ajout et leur suppression au besoin.

Egalement, notre programme se doit d’apporter un service équivalent concernant les baux de locations. Cela se traduit d’une part par leur création en assignant un local et un locataire et d’autre part, par la génération automatique de documents à partir de patrons fournis par l’utilisateur du logiciel, propriétaire des biens immobiliers.

Enfin, une des principales attentes de l’utilisateur, bien que non directement liée aux fonctionnalités propres du logiciel, était la présence d’une interface propre et surtout rapide d’utilisation. Le client souhaite effectivement se dispenser de tout type de manuel et pouvoir accéder et prendre en main le logiciel de manière tout à fait intuitive et apprécier par ailleurs son esthétisme.

## Matériel et méthodes

### Etude du problème et questionnements préliminaires

Les objectifs étant fixés et déterminés, nous avons dû par la suite rechercher l’ensemble des technologies possibles afin de sélectionner celles permettant de réaliser une solution viable et durable.

Premièrement, le réflexe que nous avions eu a été d’opter pour une solution « application de bureau » ou « desktop application » en anglais c’est-à-dire un logiciel applicatif affichant son interface graphique dans un environnement de bureau. Une telle solution aurait ainsi pu être implémentée à l’aide de langages fournissant cette possibilité comme le C++ ou le Java.

Néanmoins, bien que satisfaisante de prime abord, nous nous sommes rapidement rendu compte qu’une telle idée n’était pas à conserver et qu’il était d’ailleurs plus judicieux d’aliéner notre intérêt pour ces dernières. En effet, l’une des contraintes les plus importantes était, comme nous l’avons mentionné en introduction d’étude, la présence d’une interface esthétique et intuitive. En dépit du fait que le C++ et le Java offrent des cadres d'applications et bibliothèques pour un affichage graphique, les solutions apportées n’égalaient pas les exigences de départ. Par ailleurs, la qualité de réflexion nécessaire à la mise en place d’une application à l’aide du langage C++ formait un inconvénient considérable car, bien que très complet, sa complexité intrinsèque aurait risqué de ralentir considérablement le développement de la solution sans pour autant combler une nécessité de performance.

C’est donc à juste titre que nous avons préféré éviter les bibliothèques relevant de ces deux langages comme GTKmm ou Qt pour le premier et JavaFX pour le second, pour nous concentrer sur des technologies plus adaptées à notre problème.

Nous nous sommes alors rapprochés de techniques employées au sein du World Wide Web[[1]](#endnote-1) car elles disposent d’une flexibilité notable et procurent une réelle segmentation entre l’architecture logicielle interne et la création de l’interface. Cette occasion de fractionner le développement a d’ailleurs formé un atout notable qui nous a largement permis de mettre en place une organisation solide et robuste du travail d’équipe comme nous le verrons dans la suite de notre étude. En outre, du fait de leur présence récurrente dans l’univers numérique, la possibilité de créer des interfaces « familières » a été un avantage non négligeable pour la suite du projet et a donc, par la même occasion, confirmé notre sélection.

Ensuite, dès lors que nous avions déterminé une piste de développement et de création, il nous a été obligatoire, comme nous allons l’expliquer en sous-partie suivante, de conceptualiser la solution et de déterminer un plan d’implémentation afin de cadrer notre travail.

### Architecture de la solution et conception

La phase de conception, essentielle au bon déroulement d’un projet a eu lieu en lien direct avec le client final. Effectivement, il était important, dès les prémisses de la création de la solution, d’être en accord avec l’utilisateur afin de proposer un résultat adéquat à ses besoins que nous avons développés précédemment.

La première étape a été de comprendre quelles fonctionnalités souhaitait le client et de quelle manière pouvaient-elles être affichées à sa convenance. La maquette ainsi réalisée de l’interface dans son ensemble donnait une idée de la représentation globale du logiciel et de son interface générale. C’est alors ici que le client s’est montré relativement souple dans la création de chaque cas d’utilisation et nous a offert la liberté de la création des interfaces dans ces cas spécifiques. Evidemment la contrainte d’une utilisation abordable et autodidacte du logiciel restait maîtresse de nos ambitions.

**[ INSERER UNE IMAGE DE LA MAQUETTE + L’EXPLIQUER]**

Néanmoins, afin de pouvoir générer des documents facilement et rapidement, il était évident de devoir coucher sur le papier l’ensemble des données devant être renseignées au sein du logiciel. Il a donc été nécessaire de recenser une liste d’informations pour chaque entité prenant part à la création d’un contrat comme, à titre d’exemple, le locataire ou le local.

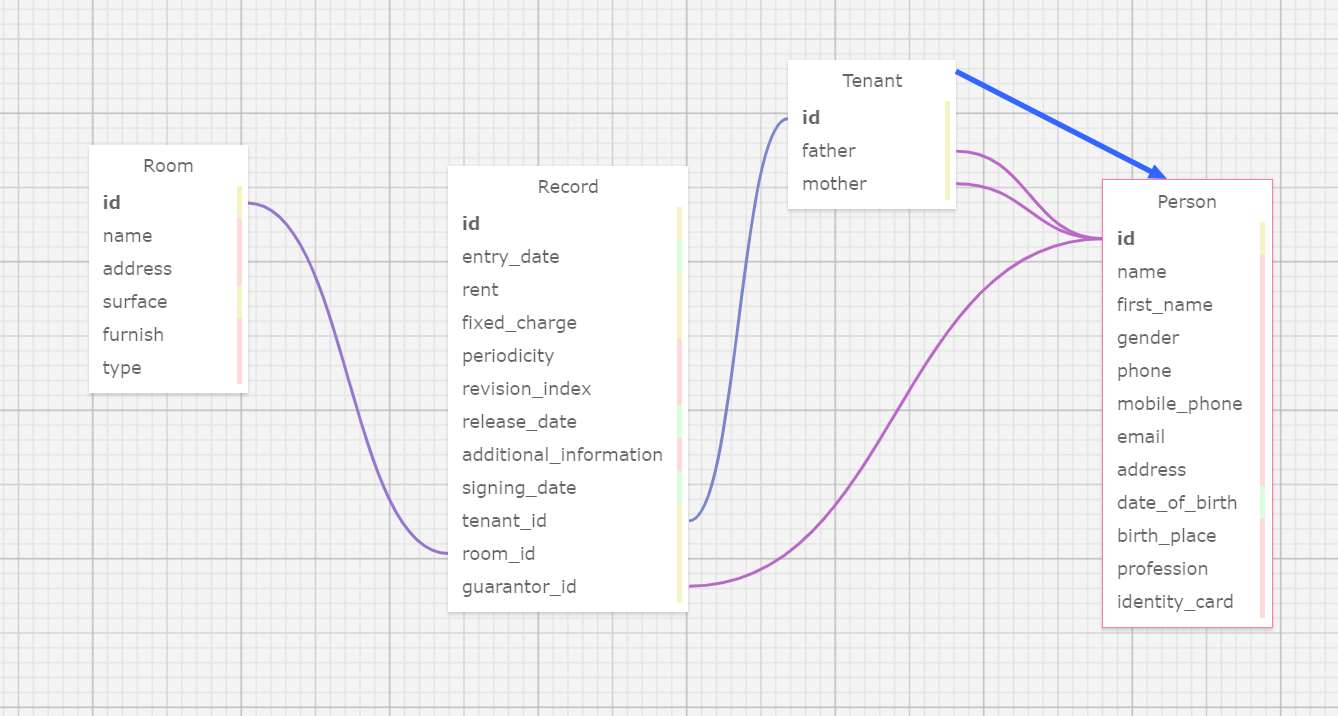
Subséquemment, nous avons dû nous pencher sur la manière qui pourrait nous permettre de stocker et garder ces données de façon structurée. Immanquablement, la création d’une base de données prévue à cet effet à de suite été étudiée. Nous avons alors pris soin d’échanger au cours d’une réunion avec notre tutrice afin de pouvoir agencer, avec la disposition la plus logique et rigoureuse possible, l’ensemble des données mentionnées. De cette rencontre productive a émergé le schéma suivant qui a de suite été adopté et qui d’ailleurs nous aura suivi jusqu’à la fin du projet.

Figure 1 : Représentation schématique de la base de données utilisée au sein du projet

Comme cela peut être constaté aisément, l’ensemble de la base de données et d’ailleurs, par extension, l’ensemble du code de notre application est en anglais. Ce choix est justifié de par le fait que même si l’interface est en français, l’anglais est la langue utilisée en informatique dans le milieu professionnel. Nous avons en conséquence souhaité reprendre cette pratique usuelle et courante et l’appliquer à notre projet.

Dans le prolongement de la conception de notre solution, s’est alors posé le problème de comment récupérer les patrons de documents, comment les interpréter et surtout comment les classer. En premier lieu, suivant les conseils et explications du client, il était particulièrement judicieux et commode d’élaborer une stratégie pour catégoriser les documents. Nous avons alors décidé de considérer ces modèles textuels sous forme de trois catégories que l’on retrouve classées ci-dessous du cas le plus spécifique cas le plus général :

* Les patrons uniques ou concernant un local particulier

Ils regroupent les modèles liés à un seul local présent en base de données. L’on retrouve par exemple les documents relatifs aux risques sismiques et technologiques d’un appartement ou hangar.

* Les patrons typés ou relatif à un type de local

Cette catégorie rassemble les documents relatifs aux appartements seulement ou aux hangars uniquement. L’attestation de consentement d’HADOPI est par exemple un modèle correspondant aux appartements et n’est jamais présente lors de la location des hangars car ces derniers ne prévoient pas un accès internet.

* Les patrons globaux

Les modèles figurant dans cette catégorie sont les plus généraux et ne sont en conséquence pas liés à un type de contrat en particulier, ils sont toujours présents et doivent en conséquence être toujours générés dans n’importe quel type de location.

Connaissant alors, de par notre étude conceptuelle, parfaitement les objectifs du projet et la manière par laquelle celui ici devait être façonné, nous devions alors amorcer le développement de ce dernier. Nous avons alors entrepris de mettre en place l’architecture définie à l’aide de technologies Web comme expliqué antérieurement. Nous verrons, en sous-partie suivante, dans quelle mesure nous avons mis en œuvre la solution, avec quelles ressources et dans quelle perspective d’organisation.

### Mise en œuvre de la solution

#### Ressources et organisation du travail

Avant de débuter l’écriture du code à proprement parlé, nous avons décidé de nous organiser grâce à *GitHub*[[2]](#endnote-2), service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions *Git*[[3]](#endnote-3). Celui-ci permet en effet de planifier les étapes du projet grâce à un *« project board »* et de garder à l’esprit les bugs et les divers disfonctionnements à corriger au travers des *« issues »*. L’outil *Git* nous a également permis de faciliter le déroulement du projet par la mise en commun de nos travaux et la gestion des conflits inhérente à la fusion des codes sources.

Notre dépôt est accessible à l’URL suivante : <https://github.com/Clem9963/REAM> et recense l’historique de notre projet que l’on peut donc qualifier *« d’open source »*.

Successivement à la mise en place de notre plan d’organisation, nous avons pris le temps de choisir les outils que nous allions utiliser pour créer la solution. Comme nous l’avons précisé précédemment, nous avions décidé de nous concentrer sur des technologies Web. C’est donc alors que nous avons souhaité mettre en place un cadre de travail avec un serveur HTTP[[4]](#endnote-4) et une base de données en local c’est-à-dire sans hébergement sur internet.

La préparation d’un environnement n’a pas été tâche ingrate car de nombreux outils libres de droits existent. Nous nous sommes alors décidés d’opter pour les services d’un serveur Apache (confère figure 2 ci-contre) et d’un système de gestion de base de données MySQL (confère figure 3 ci-dessous). Malgré leur ampleur, leurs fonctionnalités nous ont été très utiles dès les premières ébauches de notre projet. En effet, leur configuration rapide nous a permis de nous concentrer rapidement sur le code source et d’obtenir ainsi aussitôt quelque chose de fonctionnel à discuter avec le client.

Figure 2 : Logo officiel du logiciel libre Apache HTTP Server créé et maintenu au sein de la fondation Apache.

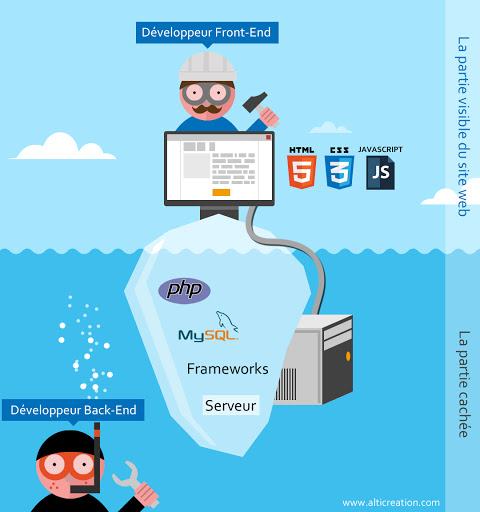
L’environnement étant installé et parfaitement fonctionnel, nous avons alors décidé de nous séparer la charge de travail pour éviter, au cours du développement, de produire des codes divergents qui auraient nécessité une mise en commun plus que chronophage.

Figure 3 : MySQL, système de gestion de bases de données relationnelles.

Cela a amplement été facilité par la qualité intrinsèque du développement Web qu’est la séparation de la programmation côté serveur de celle côté client. Il a donc été tout à fait judicieux d’attribuer l’une et l’autre à chacun des membres du binôme. Sylvain BESSONNEAU s’est alors tenu responsable d’étudier et de mettre en place l’architecture *« backend[[5]](#endnote-5) »* tandis que Clément DIF s’est concentré sur la partie frontale du projet plus communément dénommée *« frontend [[6]](#endnote-6)»*.

Concernant les langages, nous avons d’une part décidé d’utiliser le PHP pour le côté *« backend »* et le JavaScrit, CSS et HTML d’autre part pour son alter ego frontal.

Figure 4 : Représentation schématique de la séparation frontend et backend dans un projet de développement Web.

*Crédits :* [*www.alticreation.com*](http://www.alticreation.com)

Du fait de sa flexibilité et de sa compatibilité avec la plupart des systèmes de gestion de base de données, le PHP était en effet une technologie pertinente compte tenu des objectifs du projet. Par ailleurs, il dispose d’un vaste panel de *« frameworks »* aidant à la création de sites Web et très largement utilisés dans le milieu professionnel. Nous avons alors considéré qu’utiliser la version 4 du cadriciel Symfony pouvait être très adapté à la situation car il s’agit d’un cadre d’applications fournissant de nombreux outils. L’on retrouve par exemple la génération et le remplissage de documents, la gestion des données en base et également le puissant moteur de templates Twig.

Au niveau du backend, côté serveur, nous utilisons le langage PHP avec le Framework Symfony. Symfony nous évite ains i de passer du temps sur des parties ennuyantes du logiciel, pour nous concentrer sur la partie métier.

Au niveau du frontEnd, côté client, nous utilisons les langages Javascript, CSS et HTML afin d’avoir un rendu agréable à regarder pour l’utilisateur.

Entre le front et le back, nous passons les paramètres et éléments dans des fichiers Twig qui sont ensuite transformés en fichier html interprétables par les navigateurs actuels.

Au niveau des données, nous les stockons dans une base de données, afin que celles-ci ne se perdent pas. Au début nous avons utilisés le server de base de donnée MySQL. Puis, nous voulions que notre projet se suffise à lui-même, c’est-à-dire que lorsque nous téléchargeons celui-ci, il n’y a pas besoin d’utiliser d’autres outils ou librairies externes au projet. Ainsi, nous sommes passé sur une base de donnée SqLite, dont le fichier se trouve directement dans notre projet. Ceci permet un meilleur déploiement de notre solution.

Au niveau du serveur, nous avions besoin d’un serveur web afin d’exécuter le code PHP. Pour se faire, nous avions commencé par utiliser le serveur Apache. C’est un serveur très connu et qui fonctionne très bien. Cependant, toujours dans un souci de déploiement, nous préférions un serveur qui soit intégré à notre dossier de projet. Comme le logiciel est utilisé en local, c’est-à-dire qu’il n’est pas connecté à Internet, nous avons pu nous servir du server web intégré à Symfony. Ce qui nous a permis de nous passer d’Apache et ainsi supprimer une dépendance de taille au projet.

De plus, au niveau de la gestion des documents génériques, nous pensions utiliser des fichier PDF remplissable. Cependant, cette solution nous contraignait à devoir acheter une license Adobe afin de pouvoir remplir les fichiers PDF. Nous avons donc cherché d’autres solutions, et nous avons décidé d’utiliser plutôt des fichiers word (DOCX). Ceux-ci sont en effet remplissable directement par le code PHP et les modules Symfony.

#### Le *backend* ou l’aspect fonctionnel

Le backend, c’est tout ce que l’utilisateur du logiciel ne voit pas. Il contient le code métier.

C’est lui qui va fournir les différents éléments au frontend. Par exemple, les locataires, les contrats, les locaux ou les documents.

Ces differents éléments sont accessibles grâce à des chemins, que l’on appelle des routes. (par exemple : /locataires/12 ou /contrats/1/addDocument)

C’est aussi le backend qui va permettre d’envoyer des emails automatiquement lorsqu’un contrat est ajouté et c’est aussi lui qui va permettre de remplir les documents templates avec les informations fournies lors de la création d’un contrat.

Les documents ainsi que les templates sont stockés selon une arborescence de dossiers bien précise :

* users
* default
* generated Dossier des templates générés automatiquement
* record1 Document templates pour le contrat numéro 1
* record2 Document templates pour le contrat numéro 2
* ……
* templates Dossier des contenant les templates vides
* general Contient les templates généraux
* typed Contient les templates typés
* apartments Contient les templates pour les appartements
* hangars Contient les templates pour les hangars
* unique Contient documents pour les locaux
* room1 Contient les documents relatifs au local numéro 1
* room2 Contient les documents relatifs au local numéro 2
* …..
* tenants Dossier des contenant les documents des locataires
* tenant1 Contient les documents relatifs au locataire numéro 1
* tenant2 Contient les documents relatifs au locataire numéro 2
* …

#### Le *frontend* ou la relation directe avec l’utilisateur

## Résultats et perspectives

### Utilisation de la solution

#### Gestion des locataires

Grace à notre interface « user friendly » il est très facile de gérer les locataires. On peut en ajouter avec le formulaire d’ajout d’un locataire. On peut par ailleurs renseigner son père, sa mère ou les deux. Ceux-ci sont ainsi sauvegardés en base de données lors de la soumission du formulaire.

Lors de l’ajout d’un locataire, plusieurs champs sont à renseignés :

Au niveau du locataire :

Nom,

Prénom,

Sexe,

Téléphone fixe

Téléphone mobile,

Email,

Adresse postale,

Date de naissance,

Lieu de naissance

Au niveau du/des parents :

Nom,

Prénom,

Sexe,

Téléphone fixe

Téléphone mobile,

Email,

Adresse postale,

Date de naissance,

Lieu de naissance,

Profession

Tous ces champs sont obligatoires et nécessaires à la création d’un contrat par la suite.

Une page permet aussi de visionner tous les locataires enregistrés : on peut ainsi les modifier, les supprimer ou les consulter de manière fluide pour l’utilisateur.

Sur la page de consultation d’un locataire, il est possible de rajouter des documents en lien avec le locataire. Par exemple, il est possible d’ajouter la carte d’identité de celui-ci. Celle-ci fait d’ailleurs parti des documents obligatoires que doit posséder l’utilisateur du logiciel. Mais ces documents peuvent être rajoutés plus tard, par exemple, si au moment de la création du contrat, le locataire n’est pas en possession de sa carte d’identité. L’utilisateur est notifié, sur la page principale de l’application, des locataires dont la carte d’identité n’a pas encore été renseignée. Cela lui permet ainsi de toujours savoir quels locataires ne sont pas tout à fait en règle.

#### Gestion des locaux

Notre logiciel permet aussi la gestion des locaux. Ceux-ci comprennent les appartements et les hangars. En effet, il est possible de louer soit un appartement, soit un hangar, soit les deux.

Pour ce faire, tout une partie du logiciel est consacrée aux locaux.

On peut ajouter de nouveaux locaux grâce au formulaire d’ajout approprié.

Plusieurs champs obligatoires sont à renseignés :

Le nom du local, son adresse, sa surface, son type (appartement / hangar) et s’il est meublé ou non.

Après avoir créé un local, l’utilisateur peut associer des documents patrons au local. Comme ceci, chaque local possède ses propres documents patrons qui seront remplis automatiquement lors de la création d’un contrat.

Comme pour les locataires, il est possible de lister les locaux, de les modifier ou de les supprimer.

#### Gestion des contrats

Une fois que le local et le locataire ont été renseigné, il est alors possible de créer un contrat.

Pour ce faire, nous avons élaborer un formulaire de création de contrat, dans lequel le logiciel va demander de remplir certains champs :

Date de début du contrat,

Montant du loyer,

Montant des charges fixes,

Périodicité du loyer,

Indice de révision

Date de fin de contrat,

Date de signature du contrat,

Informations additionnelles (champ facultatif),

Le locataire relatif au contrat,

Le local relatif au contrat

Le champ permettant de choisir le locataire est en fait une liste de sélection qui contient tous les locataires renseignés. Il suffit ainsi de sélectionner celui-ci à qui l’utilisateur veut attribuer le contrat. C’est la même chose pour la sélection du local.

Lors de la création du contrat, il faut aussi renseigner un garant. Celui-ci peut être le père, la mère ou une autre personne.

Si c’est une autre personne, il faut remplir les champs relatifs au garant : ce sont les mêmes que pour ceux pour l’ajout du père ou de la mère lors de la création d’un locataire.

Si c’est le père ou la mère qui est renseignée, une vérification est effectuée lors de la soumission du formulaire. En effet, par exemple, on ne peut pas choisir un garant qui est la mère si, lors de la création du locataire, on a renseigné le père. Du coup, ce qu’il faudrait faire, c’est soit changer le garant, soit modifier le locataire afin de rajouter un parent.

Comme pour les locataires et les locaux, il est tout à fait possible de visualiser la liste des contrats générés et de les supprimer. Il n’est par contre pas possible de les modifier car sinon ce n’est plus le même contrat.

De plus, lors de la création d’un contrat, des documents sont générés automatiquement grâce aux informations fournies par le formulaire. En effet, l’un des buts principal de ce logiciel est de pouvoir remplir automatiquement des documents lors de la création d’un contrat, afin de faire gagner du temps à l’utilisateur. Il faut juste que l’utilisateur fournisse les documents templates aux logiciels et c’est le logiciel qui s’occupe de les remplir. C’est le cas notamment pour le contrat de bail, pour l’attestation ADOPI ou encore pour l’inventaire.

#### Gestion des documents

D’autres documents sont utilisés :

* Les documents relatifs à un local : ce sont des documents qui diffèrent d’un local à un autre. Ces documents sont ainsi gérés directement dans la partie du local en question

On peut en ajouter, en supprimer et les télécharger. Le format accepté est le format Word (docx). C’est obligatoirement ce format car il faut pouvoir le lire et le remplir automatiquement. C’est un format qui nous permet de remplir facilement le document à partir du code PHP.

* Les documents générés automatiquement : Ce sont les documents qui sont générés lors de la création d’un contrat. Ceux-ci sont accessible directement à partir du contrat auxquels ils sont rattachés. Pour ce faire, nous utilisons une librairie PHP qui prend en entrée un fichier Word (docx) et qui va le remplir grâce à des balises présentes dans le fichier. En sortie, nous obtenons un fichier PDF qui a donc été généré automatiquement.
* Les documents relatifs à un locataire : ce sont tous les documents obligatoires ou non relatifs à un locataire en particulier. Ils sont ainsi accessibles à partir de la page du locataire en question. Le logiciel accepte des fichiers PDF, JPEG et PNG. Il est possible d’en ajouter, d’en supprimer ou d’en télécharger un. De plus, un bouton permet de télécharger tous les fichiers d’un locataires sous forme d’archive.

Bonne idée de parler des documents, de leur classement et de leur génération mais il serait peut-être plus approprié d’en parler dans la partie sur le backend…

Ici, il faudrait plutôt placer une capture d’écran et expliquer l’utilisation du logiciel en lien avec les objectifs attendus.

### Perspectives d’amélioration

Utilisation de Jhipster afin de bien séparer les parties frontend et backend, avec l’utilisation d’API REST. Jhiipster est bien adpaté pour ce genre de projet qui utilise des formulaires et des entités. De plus, il gère aussi l’intégration et le déploiement continu. Et il est très facile de lancer le site sur le cloud.

# Conclusion

# Résumé

# Abstract

# Glossaire

# Références bibliographiques

# Annexes

1. World Wide Web [↑](#endnote-ref-1)
2. GitHub [↑](#endnote-ref-2)
3. Git [↑](#endnote-ref-3)
4. Serveur HTTP [↑](#endnote-ref-4)
5. Backend [↑](#endnote-ref-5)
6. Frontend [↑](#endnote-ref-6)