Rapport d’étudiants ingénieurs

Deuxième année

Filière 5 : Réseaux et sécurité informatique

Logiciel de gestion de patrimoine immobilier

*Présenté par* : **Sylvain BESSONNEAU, 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](#_Toc32179547)

[Table des illustrations 3](#_Toc32179548)

[Remerciements 4](#_Toc32179549)

[Introduction 5](#_Toc32179550)

[Etude générale du projet 5](#_Toc32179551)

[I. Introduction de l’étude 5](#_Toc32179552)

[1. Contexte et activité du client 5](#_Toc32179553)

[2. Analyse du besoin 5](#_Toc32179554)

[3. Objectifs du travail et attentes du client 5](#_Toc32179555)

[II. Matériel et méthodes 6](#_Toc32179556)

[1. Etude du problème et questionnements préliminaires 6](#_Toc32179557)

[2. Architecture de la solution et conception 7](#_Toc32179558)

[3. Le *backend* ou l’aspect fonctionnel 8](#_Toc32179559)

[4. Le *frontend* ou la relation directe avec l’utilisateur 9](#_Toc32179560)

[III. Résultats et perspectives 9](#_Toc32179561)

[1. Utilisation de la solution 9](#_Toc32179562)

[a) Gestion des locataires 9](#_Toc32179563)

[b) Gestion des locaux 9](#_Toc32179564)

[c) Gestion des contrats 9](#_Toc32179565)

[d) Gestion des documents 9](#_Toc32179566)

[2. Perspectives d’amélioration 9](#_Toc32179567)

[Conclusion 9](#_Toc32179568)

[Résumé 9](#_Toc32179569)

[Abstract 9](#_Toc32179570)

[Glossaire 9](#_Toc32179571)

[Références bibliographiques 9](#_Toc32179572)

[Annexes 9](#_Toc32179573)

# Table des illustrations

# 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

IL FAUDRAIT METTRE ICI UNE INTRODUCTION QUI PRESENTE LE PROJET DANS SA GLOBALITE TOUT EN EVITANT LA REDONDANCE AVEC L’INTRODUCTION DE L’ETUDE.

# 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 pose, dans une autre mesure, problème 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 Web1 car elles disposent d’une flexibilité notable et procurent une réelle segmentation entre l’architecture logicielle interne et la création de l’interface. 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

Au niveau du backend, côté serveur, nous utilisons le langage PHP avec le Framework Symfony. Symfony nous évite ainsi 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
* …..

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

## Résultats et perspectives

### Utilisation de la solution

#### Gestion des locataires

#### Gestion des locaux

#### Gestion des contrats

#### Gestion des documents

### 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