Arnaud Roudsovsky
Christophe Frezier

Jean Luc Le Gual

Analyste développeur au sein de la cellule d'expertise technique

Lainé Louis

Université de Bordeaux.

1 — Remerciements

En premier lieu, je souhaite adresser mes plus sincères remerciements à l'ensemble de l'équipe SQLI Bordeaux.

J'ai particulièrement apprécié le professionnalisme et la bonne humeur de chacun, qualités qui ont fait de ce stage une réussite tant sur le plan humain que professionnel.

Je tiens à remercier chaleureusement, **Béatrice Fichera** et **Emmanuel Uria** qui m'ont accordés leur confiance et permis d'intégrer l'agence de Bordeaux pour ce stage.

Je souhaite également remercier **Arnaud Roudsovsky**, mon maitre de stage, ainsi que toute l'équipe d'experts techniques avec laquelle j'ai été amené à travailler.

Merci à **Miguel Diez Garcia**, **Frédéric Gracia**, **Romain Ballan**, **Christian Alonzo Chavez Ley**, **Rodolphe Baron** pour leurs conseils.

Un merci en particulier à **Christophe Frezier** qui m'a guidé, aidé et conseillé tout particulierement durant ce stage.

Un merci à Mr **Jean-Luc Le Gual** pour sont suivi pendant le stage et ses conseils ainsi que l'équipe pédagogique de la MIAGE Bordeaux, qui m'a permis par le biais de ce stage, d'enrichir mon experience profesionnelle.

Table des matières

1 Remerciements **1**

2 Summary **4**

3 Introduction **5**
3.1 Le groupe SQLI . . . 5
3.1.1 Présentation et histoire . . . 5
3.1.2 Un groupe européen . . . 5
3.1.3 Les valeurs et le positionnement de l'entreprise . . . 6
3.1.4 Les métiers . . . 7
3.1.5 Clients . . . 8
3.1.6 Evolution et positionnement . . . 8
3.2 Cellule d'expertise technique . . . 9

4 Seating Plan **10**
4.1 Analyse des besoins . . . 10
4.1.1 Problèmatiques . . . 10
4.2 Conception et développement . . . 11
4.2.1 Conception . . . 11
4.2.2 Développement . . . 13
4.2.3 Gestion de projet . . . 14
4.2.4 Capacity planning et interface gestion avancée . . . 16
4.3 Continuous integration . . . 18
4.4 Valeur ajoutée du projet . . . 20
4.4.1 Bilan . . . 20
4.4.2 Enseignements . . . 20
4.4.3 Recette . . . 20

5 Pantaire **21**
5.1 Analyse des besoins . . . 21
5.1.1 Problèmatiques . . . 21
5.1.2 Specification . . . 21
5.2 Réflexion sur le NoSQL (Not Only SQL) . . . 23
5.2.1 Pourquoi les bases NoSQL sont elles intéressantes ? . . . 24
5.3 Développement . . . 24
5.3.1 Build pattern . . . 26
5.3.2 GIT . . . 27
5.4 Valeur ajoutée du projet . . . 28
5.4.1 Bilan . . . 28
5.4.2 Enseignements . . . 28

6 Une expérience réussie **29**

7 Annexes **30**
7.1 Sources 30
7.2 A propos 30

Glossary **31**

2 — Summary

To validate my bachelor degree in computing science i had to make an intership for 3 month which a decided to perform in a IT service company named SQLI located near Bordeaux at Pessac.
The main goal of the internship is to enforce all the courses seen during the year like project management or application developpement.
During 3 month i worked in the technical expertise center under the direction of the CTO Mr Roudsovsky Arnaud.
He give me those internship topics.

- Analysis and developpement of a management tool in order to manage the agency seating-plan.
- Analysis and developpement of an application in order to aggregate project key's perfomance indicator in an decision support application.

Those topic, were perfectly in harmony with my professionnal project which is working with a very heavy technical aspect.

Now i'm going to introduce the followings parts.

- Firstly i'm going to introduce the firm.
- Secondly i'm going to explain how i led my first internship topic
- Thridly i'm going to describe how i led the second internship topic
- And finally i'm going to present the teachings learned.

3 — Introduction

3.1 Le groupe SQLI

3.1.1 Présentation et histoire

Le groupe SQLI a une expertise reconnue dans la conception et la mise en œuvre de systèmes d'informations reposant sur les nouvelles technologies. Il accompagne ses clients quotidiennement à divers niveaux : du conseil en amont du projet au transfert de compétences en aval, en passant par la réalisation, l'intégration et la maintenance des applications.

SQLI est également la première SSII européenne à avoir mis en pratique le modèle CMMI (Capability Maturity Model Integration) et la première à avoir été certifiée CMMI niveau 3. Ce modèle permet au groupe de garantir à ses clients un certain niveau de qualité dans ses projets. Cette double orientation d'innovation et de qualité a permis à SQLI de devenir un leader européen dans le domaine des services utilisant Internet et les nouvelles technologies informatiques.

La société créee en 1990 et le schéma ci dessous rappelle les principales étapes de l'évolution de l'entreprise lui ayant permis d'aboutir au stade actuel de développement.

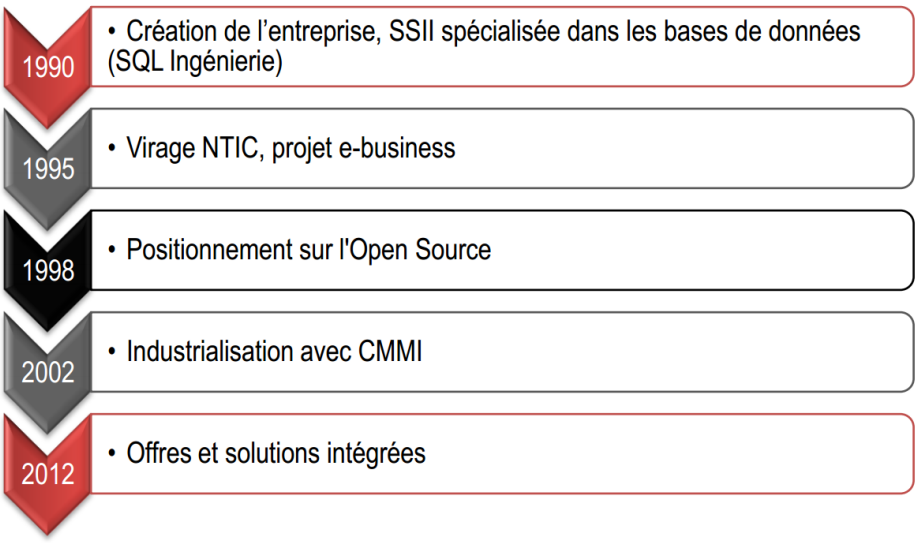


FIGURE 3.1 – Historique de la société

3.1.2 Un groupe européen

Le groupe SQLI emploie aujourd'hui plus de 1800 collaborateurs dans 17 sites répartis en France, en Suisse, dans les pays du Benelux et au Maroc. Fondée il y a onze ans, l'agence SQLI de Bordeaux a connu une forte croissance ces dernières années pour compter aujourd'hui environ 150 collaborateurs. L'agence de Bordeaux est donc de taille respectable, et comporte tous les pôles du groupe que sont l'intégration, le consulting, l'agence web

et la cellule d'expertise technique au sein duquel j'ai effectué mon stage.

3.1.3 Les valeurs et le positionnement de l'entreprise

Pour se démarquer de la concurrence, SQLI se développe autours de 4 axes majeurs.

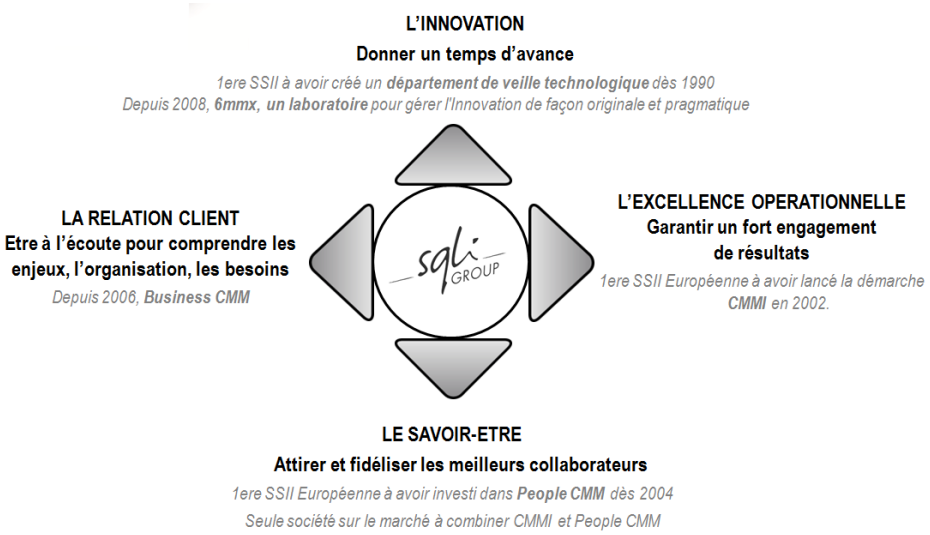


FIGURE 3.2 – Les valeurs de la société

Une des ambitions du groupe est de renforcer ses compétences par des acquisitions ou en nouant des partenariats industriels et commerciaux afin de développer des solutions métiers e-business innovantes dans des secteurs très porteurs.
Ainsi parmi les entreprises ayant rejoint récemment le groupe SQLI :

INLOG Secteur de la santé

IconeWeb Secteur de l'immobilier

Clear Value Spécialiste français de SAP

3.1.4 Les métiers

SQLI souhaite pouvoir être un interlocuteur pertinent quelque soit les besoins du client, c'est pourquoi elle regroupe de nombreuses activitées.

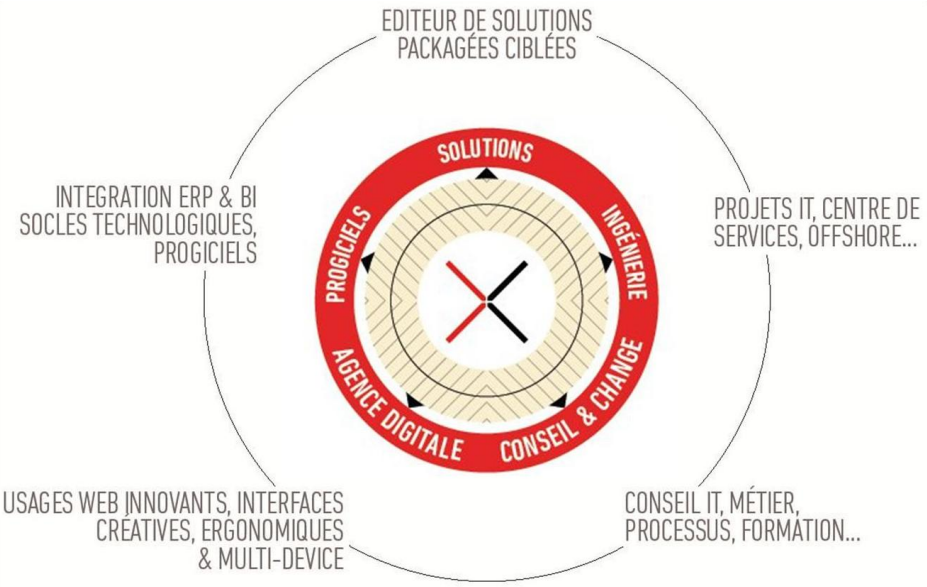


FIGURE 3.3 – Les métiers de SQLI

Le pôle solutions

Le groupe vend ses propres applications de logistique, finance, management de projet, dont la suite SteeringProject, orientée gestion de projet et maintenance applicative.

Le pôle ingénierie

Le groupe affiche une expertise reconnue dans le développement d'applications reposant sur les nouvelles technologies (J2EE 3 , PHP, .Net, Oracle, SQL Server, et autres). Ceci permet à ses ingénieurs d'améliorer leurs expertises dans ces domaines en mettant en œuvre essentiellement des projets web.

Le pôle conseil et change

Le conseil en système d'information permet d'offrir des services d'aide pour la mise en place de systèmes d'informations. Ces conseils peuvent porter autant sur le plan décisionnel ou technologique que sur le plan fonctionnel. Ce pôle apporte au client une aide opérationnel qui vise à définir des solutions personnalisées en intégrant les différentes contraintes du système d'information. Il va aussi donner des préconisations et des aides quant au choix technique pour la mise en place de nouvelles solutions.

Le pôle agence digitale

SQLI Agency est spécialisé dans le design d'apllication web. Elle propose à ses clients des réponses à forte valeur ajoutée : conseil en performance ergonomique, connaissance utilisateur, ainsi que des prestations de création, de conception, d'accompagnement et de formation au secteur internet.

Le pôle progiciels

Le groupe veut améliorer la performance de l'entreprise par le pilotage des finances, le suivi de production, des ventes, par une meilleure connaissance de ses clients et usagers, à l'aide des progiciels intégrés les plus pertinents selon le contexte.

3.1.5 Clients

SQLI compte plus de 800 clients, grands comptes et PME, issus de tous les secteurs d'activité. L'agence de Bordeaux travaille notamment en étroite collaboration avec les entreprises Réseau de Transport de l'Electricité, La Poste, Cofinoga, et plus récemment, les magasins Cultura, Sanofi- Aventis et Dekkra.

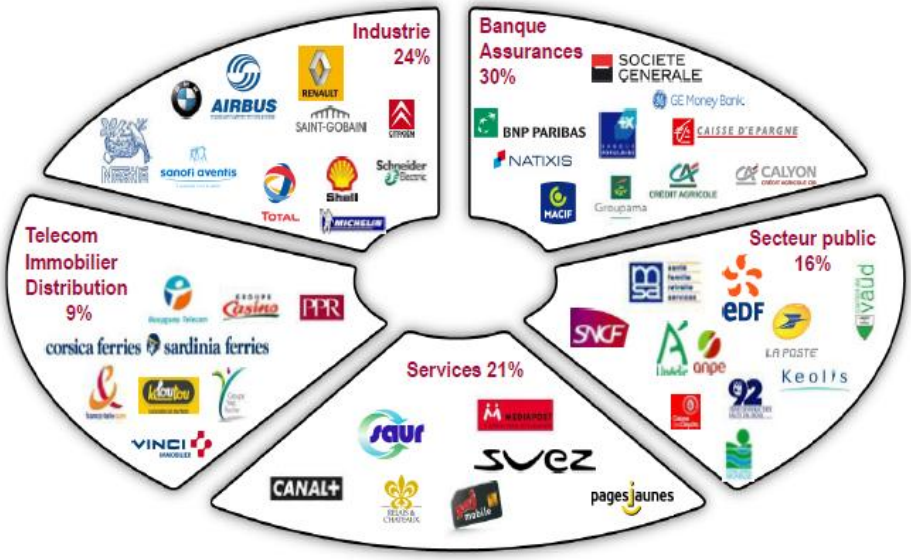


FIGURE 3.4 – Répartition des clients par secteur

3.1.6 Evolution et positionnement

Le groupe a affiché un chiffre d'affaires de 158.1M d'euros en 2012, pour un résultat net en progression à 4.5M d'euros. Ces résultats placent l'entreprise dans les 30 premières SSII en France en termes de chiffre d'affaire. Si l'entreprise n'a pas le volume d'activité des très grandes SSII telles CapGemini, Atos ou CGI (ex-logica), son positionnement technologique et sa volonté de fidélisation des clients lui ont permis au cours des dernières années de remporter des marchés de la région bordelaise devant ces mêmes entreprises. La diminution du chiffre d'affaires des deux années passées s'explique principalement par un recentrage des activités du groupe et doit être relativisé par une progression du résultat net.

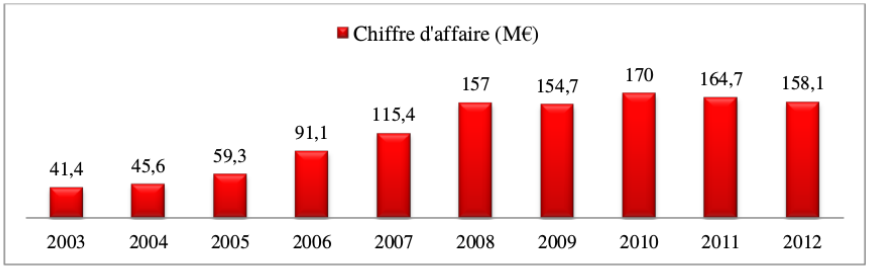


FIGURE 3.5 – Evolution du chiffre d'affaire

3.2 Cellule d'expertise technique

Dès le début de mon stage, j'ai été integré à la cellule d'expertise technique. Cette cellule unique dans chaque Agence, étant composé d'une poignée d'ingénieurs est utile à l'ensemble de l'agence pour :

1. La mise en place et maintien des environnements de développement, l'intégration continue
2. La mise en place de référents techniques pour les développements
3. Les revues de code
4. Les revues de conception et modélisation d'applications
5. Dans le cadre d'intégration continue : audit de la performance du SI, analyse du temps de réponse et de la capacité de tenue de charge, identification des points de contention.
6. Le transfert de compétences, par le biais de formations, workshops, entretiens de recrutements, etc.
7. Assure la veille technologique.

C'est donc avec cette équipe que j'ai travaillé pendant les 3 mois de stage.

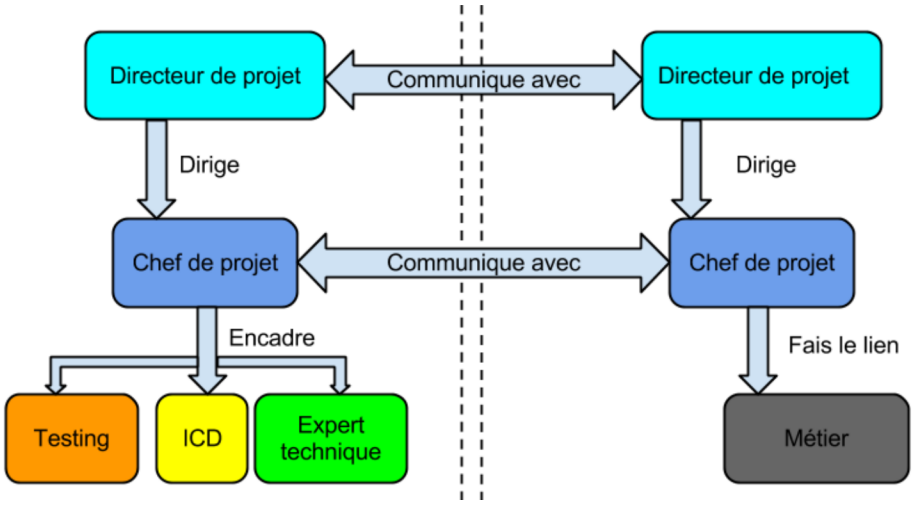


FIGURE 3.6 – Rôle de l'expertise technique sur un projet. La partie droite représente le client.

4 — Seating Plan

L'application Seating-plan à traduire par plan officiel ou plan des bureaux, est un applicatif permettant d'avoir la répartition des bureaux, employés et téléphones sur les différents batiments de l'agence de Pessac.
Cet outil est utilisé par les directeurs de projets, directeur des ressources humaines pour leur permettre de gérer le capital humain.

4.1 Analyse des besoins

Cette application déjà déployée en production, présente une ergonomie assez archaïque ainsi qu'une conception assez peu maintenable.

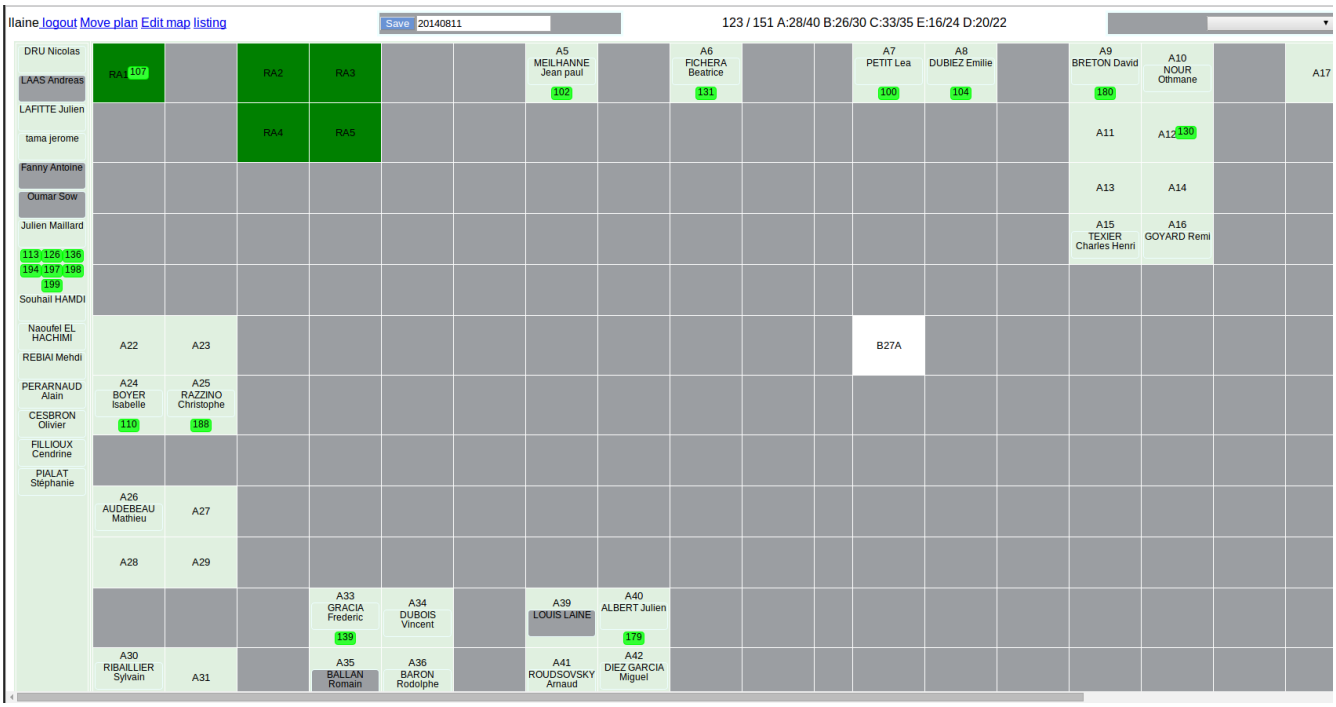


FIGURE 4.1 – L'ancien seating

En réalisant l'état des lieux de l'application existante, j'ai pu déterminer qu'elle était écrite en **PHP**, herbergée en local sur l'intranet de la société par le biais d'un server **Apache** tournant sur une virtual Machine **Debian**. Aussi aucune base de données n'était présente pour effectuer la persistence. L'ensemble des données provenait de fichiers textes, aggrémentés à la main par le CTO.

4.1.1 Problèmatiques

Mon maitre de stage désireux d'améliorer cette application m'a demandé si j'avais dans un premier temps, la possibilité de corriger les quelques bugs présents et dans un second temps, de réaliser des améliorations sur

l'existant. Etant donné l'état de l'applicatif déjà présent, j'ai décidé de repartir de zéro.

Il fallait déterminer quelles allaient être les contraintes pour ce projet. Toute la difficulté résidait dans la compréhension des attentes, pour réaliser un applicatif répondant au mieux aux besoins et ne pas livrer un outil en recul par rapport à la version précédente.

J'ai donc demandé à mon maitre de stage pour avoir quelques spécifications.

- L'application doit être écrite en PHP.
- Elle doit être accessible par tout le monde, mais en fonction de la personne se connectant octroyer un certain nombre de droits plus ou moins permissif sur les actions possible à réaliser.
- L'application doit être en mesure de gérer les différents plans par une interface de gestion avancée coté client.
- Elle doit permettre de gérer le listing des employés, téléphones et bureaux.
- Elle doit permettre de gérer dynamiquement les différents bâtiments, pouvoir les activer/désactiver dynamiquement.
- L'applicatif doit gérer un niveau de rôle spécifique à chaque utilisateur/groupe d'utilisateur.
- En fonction du rôle octroyé par l'application, l'utilisateur doit avoir la possibilité de réaliser des sauvegardes des plans (pour tester par exemple une disposition particulière). En revanche, seule la dernière sauvegarde réalisée par un administrateur prévaudra sur l'ensemble des autres et c'est celle qui sera affiché par défaut.
- L'applicatif, devra persister les données dans une base MySQL et à partir de celle-ci fournir un ensemble de statistiques.
- L'application devra embarquer une fonctionnalitée dite de **capacity-planning** (gestion de la capacité), qui permettra à SQLI de garantir la qualité de service en alignement avec les besoins métiers. Cette activité de gestion de capacité sera ainsi composé d'une surveillance sur le long terme de l'ensemble du parc physique afin d'éviter le dépassement d'effectif d'une manière globale sur l'agence ou d'une manière détaillée sur un étage en particulier.

4.2 Conception et développement

4.2.1 Conception

Etant donné le besoin de réactivité, j'ai donc choisi de réaliser une application se basant sur une architecture client-serveur.
Le dialogue se fait donc de manière asynchrone entre les deux couches et l'utilisateur n'est à aucun moment "bloqué" par le chargement d'une page. Une interface très réactive peut ainsi être mise en place par le biais de cette architecture.
Pour uniformiser les transferts entre le client et le serveur le W3C préconise l'utilisation d'une API dite REST qui est un style d'architecture.
Cette API REST, me permet de faire l'interface entre la partie cliente et serveur.

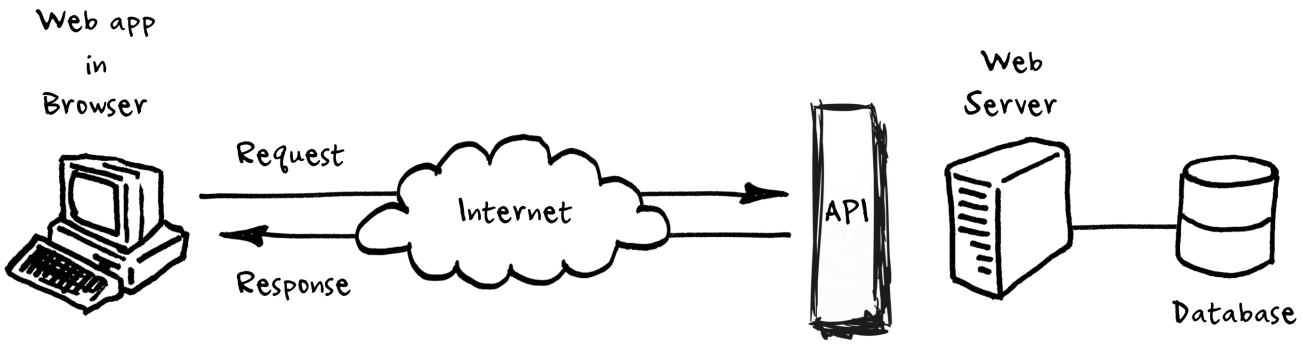


FIGURE 4.2 – Fonctionnement d'une API

D'une manière générale ce type d'architecture permet de
- Découper les responsabilités entre le client et le serveur, pour permettre au deux d'évoluer indépendamment.
- De libérer l'état de chaque requête. Celle-ci doit contenir toutes les informations nécessaires pour permettre au serveur de comprendre la requête, sans dépendre d'un contexte en particulier, ce qui permet d'économiser la puissance serveur nécessaire.
- D'uniformiser l'interface avec des règles prédifinies simplifiant largement les échanges.

Le framework utilisé coté serveur utilise, une architecture spécifique, le **MVC**. C'est un standard pour la création d'applicatif web, le **M**odèle **V**ue, **C**ontroller.

Ce modèle est destiné à répondre à des problèmatiques d'architecture d'application web. En effet, lors du développement d'une application web, ou d'une autre application, il est possible de ne pas suivre de pattern pour développer l'architecture. Seulement, l'application va très vite devenir instable car aucune séparation des responsabilités ce qui est assez compliqué à débuguer et à maintenir.

C'est ainsi que pour créer des applications web le MVC s'est imposé comme standard de fait en une architecture permettant de séparer distinctement chaque logique d'une application en trois catégories.

Le modèle : Representant le coeur véritable de l'application, il va décrire les données manipulées par l'application. Il dialogue avec la base de données. Cette partie ne s'occupe pas de la présentation

La vue : Cette partie a comme tâche de présenter les résultats renvoyés par le modèle. Cette couche interagit également avec l'utilisateur et les différents évenements qu'il peut provoquer.

Le controller : Le controller prend en charge l'ensemble des traitements entre la vue et les modèles. Il va se charger de l'ensemble des évenements, déclenchés par la vue.

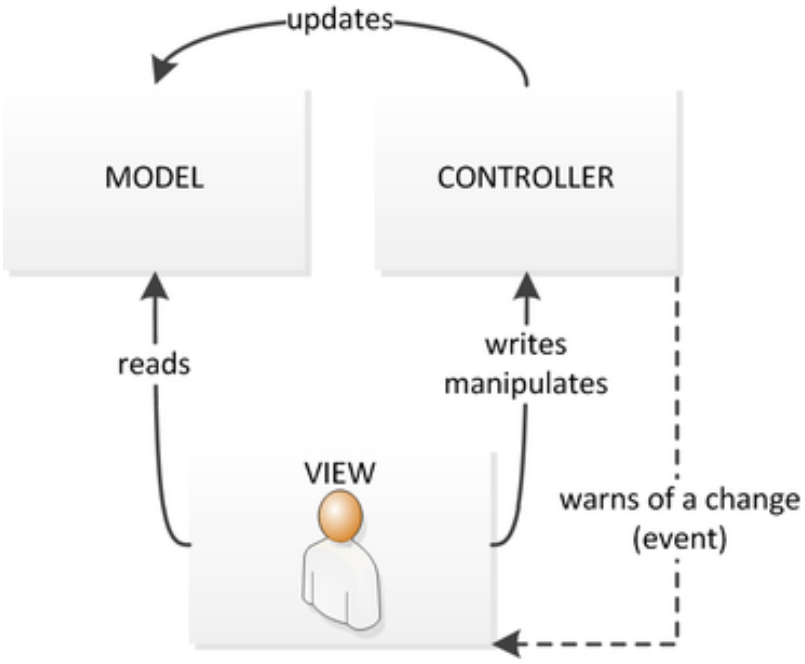


FIGURE 4.3 – Fonctionnement du MVC

Cette architecture très claire permet également, aux autres développeurs qui vont reprendre mon application, de comprendre plus facilement son fonctionnement.

4.2.2 Développement

Pour mettre en oeuvre cette architecure, j'ai du choisir les outils à utiliser pour chaque partie.

FIGURE 4.4 – Logo de Symfony

Coté serveur, mon dévolu se jeta sur **Symfony**, qui est un framework MVC écrit en PHP 5 ayant largement fait ses preuves, permettant de faciliter et d'accéler le processus de développement, se basant sur un ensemble de bonne pratique et de surcroit embarquant en natif le principe d'architecture REST.

FIGURE 4.5 – Logo de AngularJS

Coté client, je décidais de choisir le très à la mode framework JavaScript **AngularJS** porté par Google, pour son architecture MVVM, spécialement conçue pour des "web-app" ultra réactives, ce qui est en parfaite adéquation avec le futur seating-plan.

La première difficulté rencontrée lors du développement, fut de faire communiquer les deux parties. En effet, Symfony en natif embarque un système de vue propre au framework, qui au premier abord semble difficile de ne pas utiliser.

Après une recherche approfondie sur internet, je décide donc de chercher une solution pour faire dialoguer les deux couches d'une manière propre et évolutive.

Au bout de quelques heures, j'ai trouvé une solution et j'ai pu commencer à développer.
Ma découverte potentiellement qualifiée de "hacking" dans le sens ou l'on détourne le fonctionnement de quelque chose (et non pas l'autre définition parfois utilisé à tort comme synonyme de "piratage informatique"), j'ai également décidé en parallèle du stage, d'écrire un article sur un site spécialisé pour diffuser ma trouvaille, ce qui pourra potentiellement aider d'autre personnes qui comme moi ont recherché une solution sur internet, en vain.
Lien vers **l'article** (rédigé en Anglais)

4.2.3 Gestion de projet

Agile

SQLI étant certifié CMMI niveau 3, l'ensemble des pratiques utilisées pour gérer des projets suivent une guideline bien définie. Concernant donc le développement de projet informatique, l'entreprise utilise une approche *Agile*. Cette méthode s'oppose aux méthodes classiques qui spécifie et planifie avec le client dans les détails l'intégralité du produit avant de le développer. Le client va plutôt élaborer sa vision du produit à réaliser ainsi que la liste des fonctionnalités et des exigences à développer. L'équipe de développement va par la suite sélectionner une portion des exigences à réaliser dans une portion de temps courte appellée **itération**. Chaque itération inclut des travaux de conception, de spécification fonctionnelle et technique quand c'est nécessaire, de développement et de test. A la fin de chacune de ces itérations, le produit partiel mais utilisable est montré au client. Ce dernier peut alors se rendre compte par lui même très tôt du travail réalisé, de l'alignement sur le besoin. L'utilisateur final quant à lui peut se projeter dans l'usage du produit et émettre des feedbacks précieux pour les futures itérations. La visibilité ainsi offerte est clef. Cette transparence peut également apporter davantage de confiance et de collaboration dans la relation client/fournisseur. Les risques quant à eux sont levés très tôt.

En opposition à la méthode Agile, on peut trouver par exemple le schéma de développement en V.

Cycle en V

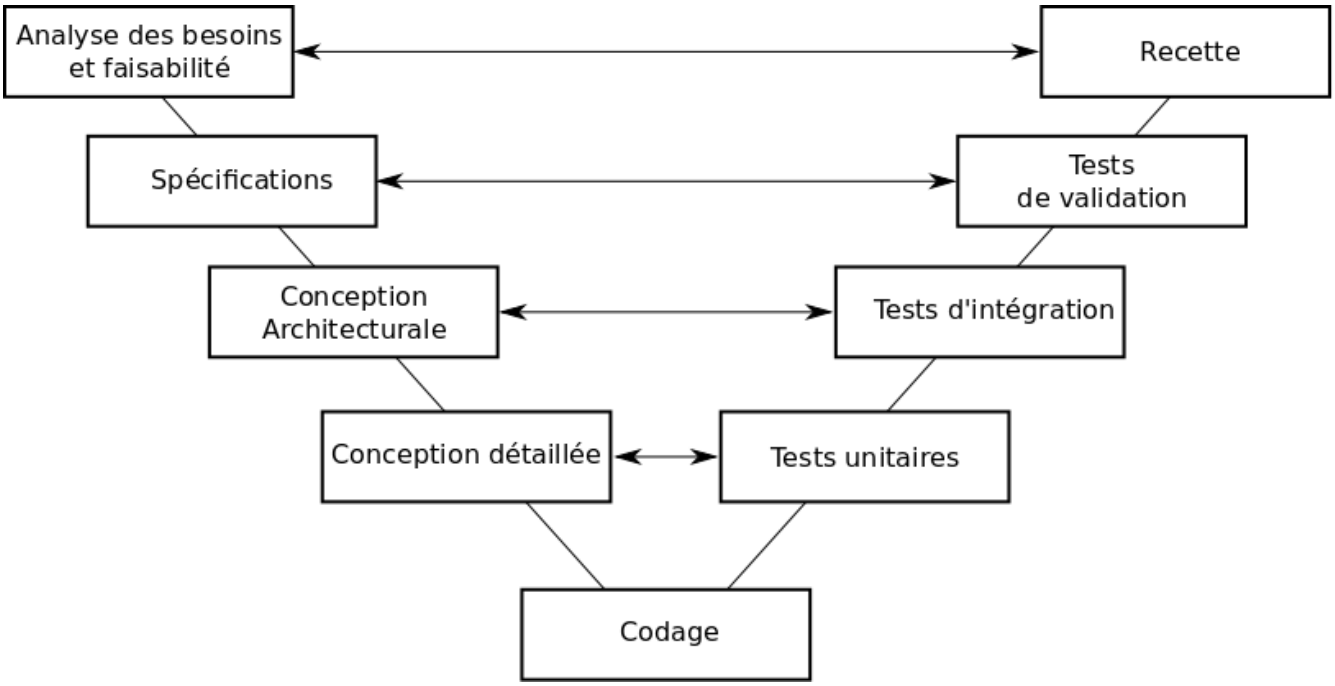


FIGURE 4.6 – Cycle en V

Ce modèle, est utilisé pour limiter au maximum la regression du projet en cas d'anomalie. On peut y distinguer trois grandes parties.

- Phase de conception
- Phase de codage
- Phase de validation

Chaque étape découle de la précédente ce qui diminue les risques sur le projet.
C'est en suivant l'approche *Agile* que j'ai réalisé le seating-plan.

Modèle de données

Après avoir configuré et mis en place l'environnement, il fallu déterminer un modèle de données pour la persistence de celle-ci.

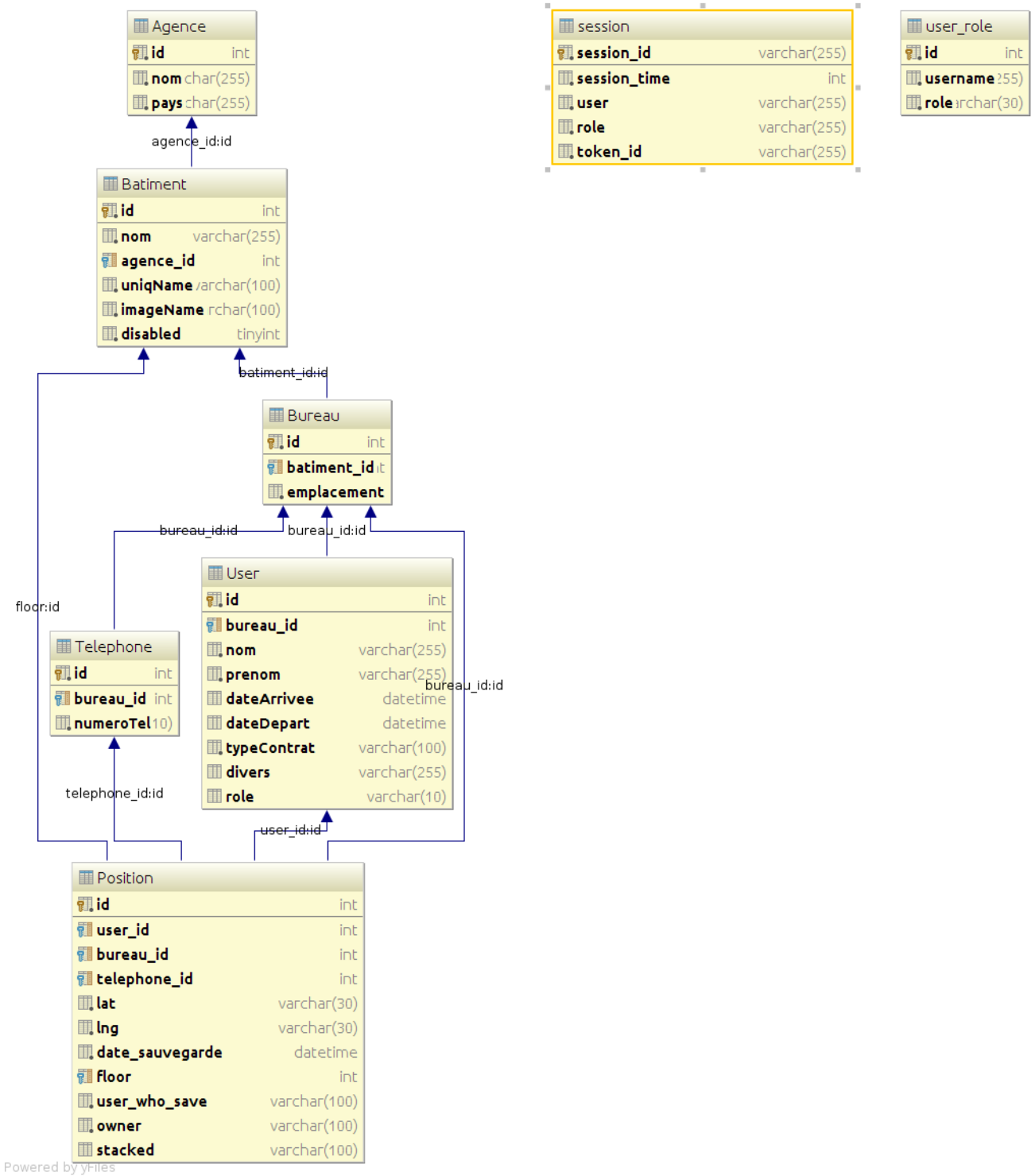


FIGURE 4.7 – Modèle de base de données

Etant donné que Symfony se base sur un ORM pour faire le lien avec la base de données, le modèle objet sera exactement le même que celui de la base de données.

L'ORM qui signifie en anglais **object relational mapping** est une technique de programmation, qui permet d'avoir un modèle objet à partir d'une base de données relationnelle, en définissant des correspondances entre cette base de données et les objets du langage utilisé. Par exemple, les requêtes ne s'effectue plus en SQL mais en DQL (Doctrine Query Language) et non plus sur des tables contenant des champs, mais sur des objets contenant des propriétés.
Une fois le framework configuré, j'ai pu commencer la phase de développement, utilisant GIT comme serveur de versionning pour m'aider à sauvegarder mon code.

4.2.4 Capacity planning et interface gestion avancée

Capacity planning

Une des fonctionnalités intéressantes à implémenter fût la mise en place du capacity-planning.

En français gestion de la capacité, qui permet à SQLI de garantir la qualité de service en alignement avec les besoins métiers. Cette activité de gestion de capacité est ainsi consitutée d'une surveillance sur le long terme de l'ensemble du parc physique afin d'éviter le dépassement d'effectif d'une manière globale sur l'agence ou d'une manière détaillé sur un floor en particulier.

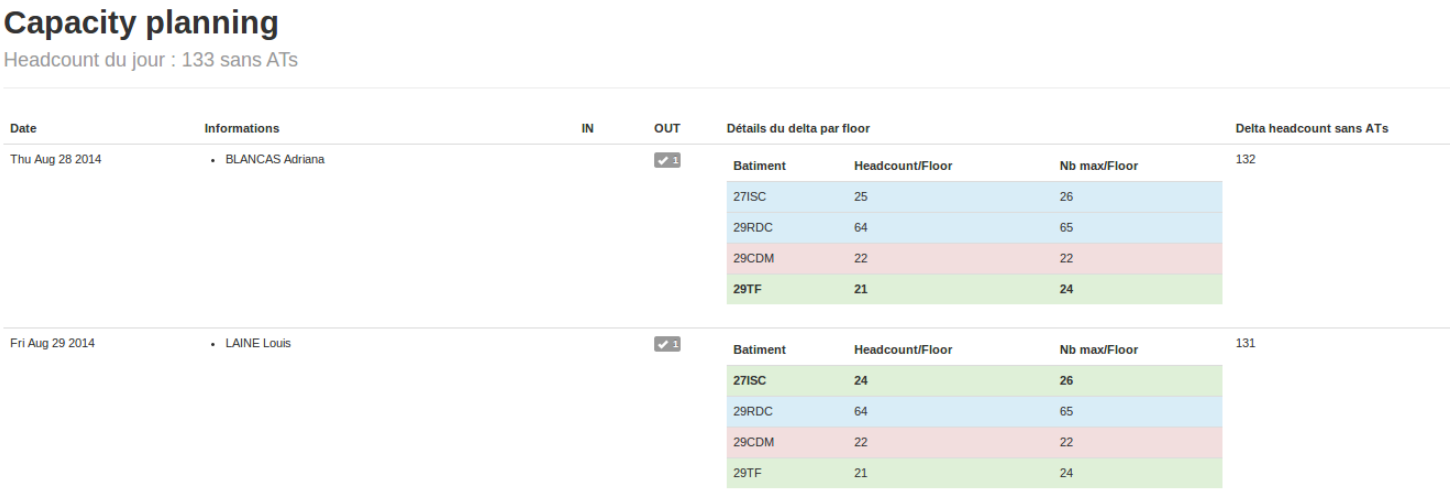


FIGURE 4.8 – Capacity planning actuellement en production

Ainsi, pour chaque date d'arrivée/départ d'une personne de l'entreprise le système est capable de donner la variation du nombre total de personnes présentes physiquement sur l'agence et de détailler ce chiffre sur l'ensemble des bâtiments activés dans l'application.
Sur l'exemple ci-dessus on peux constater, qu'au 28 août 2014, une personne s'en va du batiment **29TF**. De même pour le 29 ou une personne part du **27ISC**, libérant un bureau ramenant le nombre de places libre à un chiffre plus large.
L'utilisation d'un code couleur, permet de savoir rapidement si la marge restante est acceptable ou non.

Les managers vont pouvoir ainsi avoir une vision sur les mois à venir des différents bâtiments, qu'ils doivent gérer.

Interface de gestion avancée

La clé de voute de l'application, étant basée sur la réactivité, la gestion des différents plans ne pouvait se faire sans incorporer un système intuitif et agréable pour l'utilisateur.
C'est ainsi que j'ai développé une sorte de mini framework, basé sur jQuery UI et le drag'n'drop pour gérer toutes les interactions avec l'utilisateur. A l'aide d'une barre d'outils affichée sur le coté, l'utilisateur peux ajouter dynamiquement des objets sur les plans, qu'il va pouvoir faire glisser à un endroit spécifique. De même le système

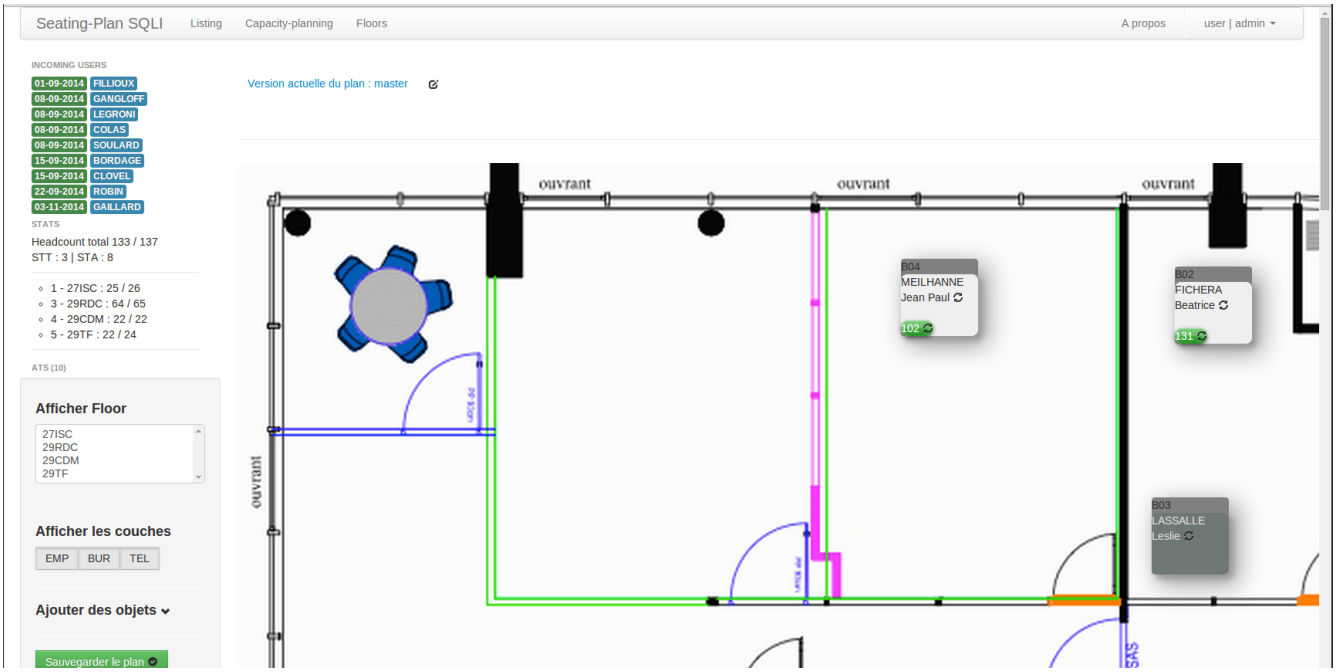


FIGURE 4.9 – Design du nouveau seating-plan

reconnaît les propriétés des différents objets qu'il est possible de manipuler. On pourra donc assigner un utilisateur à un bureau, avec des téléphones par exemple ce qui va afficher un plan complet permettant d'avoir une idée du placement des employés et s'ils disposent de téléphones à proximité.
Une fois le plan défini, l'utilisateur peut alors sauvegarder l'état actuel de ses modifications, figeant ainsi l'ensemble des objets. On peut ainsi arriver à gérer des étages, contenant beaucoup d'objets assez simplement

FIGURE 4.10 – étage contenant beaucoup d'objets

4.3 Continuous integration

Une fois l'application au stade d'une première version stable, j'ai procédé à la mise en production où mettre l'application en ligne.

Au début, j'ai tenté le déploiement à la main, par le biais d'un terminal, d'une session SSH sur la machine serveur et de quelques script Shell.
Mais ce procédé n'étant pas en règle avec les bonnes pratiques de l'entreprise, j'ai vite changé pour utiliser une méthode différente, vivement recommandée par les experts techniques autours de moi. **L'intégration continue**.

En effet dans le cadre d'une certification CMMI niveau 3 impliquant des méthodes d'extrême programming, SQLI utilise en interne une plateforme d'intégration continue, se basant sur Jenkins. Cette plateforme va permettre aux développeurs de pouvoir déployer leurs applications, d'une manière automatisée et industrielle en incorporant un certains nombre de règles que le projet doit respecter pour être mis en production.

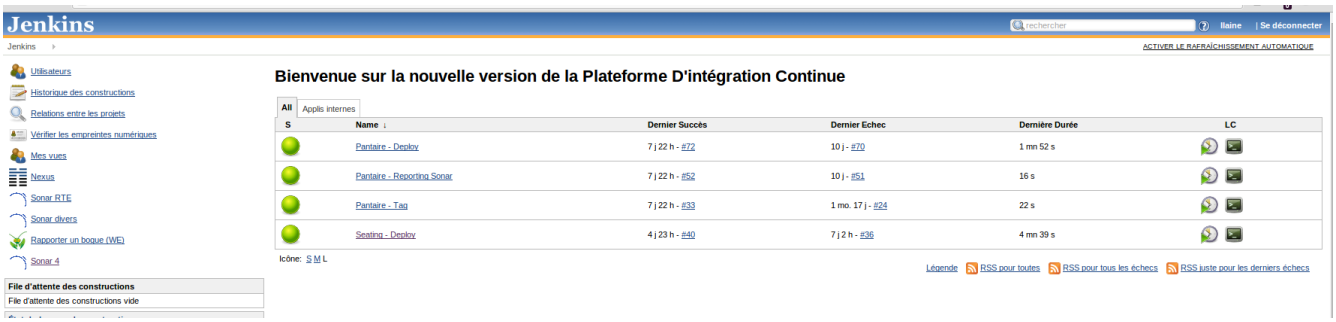


FIGURE 4.11 – Capture d'écran du logiciel

A l'aide de Jenkins, j'ai pu apprendre à déployer mon application de manière industrielle, par le biais d'un processus relativement simple et d'un script en Shell Unix faisant tout le travail à ma place me permettant d'avoir une solution automatique, reproductible et assez simple à éffectuer même pour une personne n'étant pas développeur.

Voici un exemple du script que j'ai utilisé pour effectuer la mise en production.

Listing 4.1 – deploy.sh

```
#!/bin/bash
...

setupRight(){
        cd ${APP_DIR}
        #checkCache
        rm -rf app/cache/*
        # Set right to cache/log folders
        chown mantis.www-data -R app/cache/
        chown mantis.www-data -R app/logs/
        chmod 775 -R app/cache/
        chmod 775 -R app/logs/
        echo "[i] Setup right ... "
}

# Clear the cache
clearCache(){
    cd ${APP_DIR}
    echo "Clearing cache for ${TARGET} env"
    if [[ ${TARGET} = "prod" ]]; then
        # Clear cache
        php app/console cache:clear --env=prod
```

```
24          # Setting right
            setupRight
26      else
            php app/console cache:clear
28      fi
    }
30
    fetchLastVersion(){
32      echo "[i] Fetching last version .."
        cd ${APP_DIR}
34      git archive --format=tar --remote=git@10.33.33.15:llaine/seating.git ${VERSION} > archive.tar
        checkArchive
36      tar -xf archive.tar
    }
38
    cleanUp(){
40      echo "[i] Cleanning old repo ... "
        cd ${APACHE_DIR}
42      sudo rm -rf ${APP_DIR}
        #echo "cd $APACHE_DIR && rm $APP_DIR"
44      mkdir ${APP_DIR}
    }
46

48  echo "[i] Deployement Seating.SQLI V ${VERSION}:${TARGET}"

50  usage # Checkin usage

52  checkEvt # Checking environnement

54  cleanUp # Cleaning folder in order to receive the new version

56  fetchLastVersion # Fetching last version

58  #setupRight # Setup good rights for the cache && logs folder

60  clearCache # Clear cache

62  echo "[i] Restarting server ... "

64  sudo service apache2 restart

66  ...
```

Après lecture du code, on constate que le script éxecute toute les commandes qu'une personne fait de manière classique.

En utilisant cette méthode, j'apprend d'une part de nouveaux outils qui me permettront d'être plus efficace sur le terrain dans le cadre de mon futur métier et d'autre part me permet de livrer un travail plus profesionnel, sur lequel il est très facile de faire des mise à jours, patch et autres correctifs. Les autres experts techniques présent dans la cellule peuvent également reprendre mon code, dans le cadre d'évolution ou de correctifs et déployer de nouvelles versions facilement sans être bloqué par une quelconque manipulation dont moi seul aurait eu la connaissance.

4.4 Valeur ajoutée du projet

4.4.1 Bilan

Ce premier projet chez SQLI, m'a permis tout d'abord de prendre mes marques avec la société, l'environnement autour de moi. Ayant été seul à développer cette application, j'ai mis en place ma propre méthode de travail et développé les compétences de programmation apprises durant l'année. La méthode utilisé s'articule en deux étapes.
Dans un premier temps, j'ai récolté des informations pouvant être utile à mon projet par le biais de petite réunion avec mon tuteur. Et dans un second temps une fois l'ensemble des spécifications en ma possession j'ai pu mettre en oeuvre ma méthode de travail, alternant des phases de développement intense, réalisant des objectifs fixés au jour le jour et des phases de revue de code ou des réunions. Celle-ci ajustée tous les jours pour arriver à une productivité optimale.

4.4.2 Enseignements

Réaliser ce projet m'a beaucoup appris, que ce soit dans la technique pure ou sur le côté fonctionnel.

Du coté technique, j'ai pu apprendre sur la réalisation de projet en SS2I, j'ai pu me former sur des technologies nouvelles, très intéressantes et très utilisées sur le marché actuel. Etre plus au fait des standards, bonne pratiques et convention qu'il faut utiliser dans le monde du développement.
D'un point de vue fonctionnel, j'ai appris sur le fonctionnement de projet en SSII. En faisant partie de la cellule d'expert technique j'étais au coeur du fonctionnement de l'agence de Pessac. En assistant aux réunions hebdomadaires de suivi projets, j'ai pu mieux comprendre les notions étudiés durant l'année sur la gestion d'un projet puisque appliquées sur des situations réelles.
D'un point de vue organisationnel, étant au contact au jour le jour avec les futurs utilisateurs de mon application, j'ai instauré un dialogue constant, me permettant de comprendre au mieux leurs attentes et de les traduire dans l'applicatif.
J'ai donc écouté d'une oreille attentive, leurs directives en assistant à des réunions diverses avec les différents protagonistes de l'agence Pessacaise. Avant de faire ce projet, je n'avais pas idée de l'importance qu'il pouvait y avoir à gérer constamment l'ensemble du parc physique dans une entreprise. Après avoir réalisé ce projet, je dirais que c'est indispensable, notamment étant donné la valeur très importante que représente le facteur humain dans une SSII.

4.4.3 Recette

Avant de déployer l'application, j'ai procédé avec mon maître de stage à une phase de recette. En amont du projet, une phase d'analyse des besoins et de faisabilité avait fourni des spécifications précises. Toute la difficulté fut de répondre par le biais de l'application au mieux à ces attentes. A la lumière des résultats obtenus ainsi que de l'appréciation de mon tuteur tout au long du projet, je pense que les objectifs ont été atteints pour l'entreprise. Il s'agissait en effet d'avoir un outil permettant de gérer consciemment les seating des différents bâtiments de l'agence de Pessac.

Après avoir mis en production le seating-plan, mon tuteur de stage m'a orienté sur une autre mission, un projet en interne centré sur la performance des projets.

5 — Pantaire

5.1 Analyse des besoins

Pantaire est une application permettant de présenter sous une interface plus intuitive, l'ensemble des informations concernant les serveurs gérés par SQLI. Cette application, est déployée en interne ou chez les gros clients de la société.

En plus d'avoir accès aux informations serveurs, l'équipe d'expert désire pouvoir accéder aux indicateurs de performances des différents projets développés par l'entreprise. L'idée serait de présenter d'une manière condensée ces indicateurs, sur l'application Pantaire.

5.1.1 Problèmatiques

Pour monitorer l'ensemble des projets qui sont développés, SQLI utilise un outil appelé SonarQube qui permet de mesurer la qualité du code source en continu. Ce logiciel permet entre autres d'avoir un reporting précis sur différentes métriques comme :

- Analyse statique de code
- Le taux de duplication de code dans un projet
- Le niveau de documentation
- Le respect de certaines règles de programmation
- etc.

L'agence de Pessac, peut ainsi garantir dans son développement et dans sa prise de décision, une certaine qualité de services directement chiffrable et tangible se basant sur des métriques (appelés KPI) précises directement prises sur les différents projets.

Cependant, certains indicateurs fourni par Sonar ne sont pas assez précis et pertinent et peuvent être faussés.

Par exemple, lorsqu'on définit un certain niveau de contrainte à propos des règles de programmation à suivre durant un développement, il se peut que celle-ci soient sciement non-suivies pour des raisons propres à un projet. De plus concernant le code coverage (ou couverture de code, décrivant le taux de code source testé d'un programme permettant de mesurer la qualité des tests effectués), le taux peut atteindre une valeur illusoire, faussé par un nombre trop important de tests réalisés sur des getters et setters, n'ayant aucune valeur concernant la qualité du code.

Ma seconde mission pour ce stage, sera de créer un outil permettant de présenter d'une manière condensée et différente l'ensemble des métriques fournies par Sonar.

5.1.2 Specification

Toute la difficulté réside dans la pertinence de la représentation de données. Les différents interlocuteurs clients auront accès à cet outil, et l'utiliseront pour avoir une idée de l'état des projets. L'applicatif doit être simple, intuitif et aller à l'essentiel.

Pantaire est une application, full stack JavaScript. Elle utilise le même schéma d'architecture que le seating-plan,

à savoir une architecture client-serveur ou les deux couches sont indépendantes et discutent entre elles.

Après quelques réunions avec les experts techniques m'indiquant leurs attentes j'ai pu réaliser une liste de spécification.

- L'outil doit être integré à un applicatif existant appelé *Pantaire* et suivre le même modèle de développement que celui-ci.
- L'outil doit permettre d'avoir l'ensemble des projets, pouvant être regroupés selon plusieurs propriétés.
- Il doit permettre l'accès à l'historique de chaque métrique sur chaque projet.
- Il doit intégrer une fonctionnalité de gestion de profil. Un profil correspond à un ensemble de métriques sélectionnées parmi lesquelles l'utilisateur va pouvoir appliquer une valeur plafond. Ainsi en sélectionnant un profil, l'utilisateur ne verra que les métriques qu'il a choisi sur les projets.
- L'outil doit intégrer une vision "de groupe", ainsi en fonction d'un profil et d'un groupe sélectionné avoir une vision globale de l'ensemble des projets et sur les métriques sélectionnées par le profil.
- Il doit aller chercher de manière quotidienne les données dans SonarQube et les persister dans une base de données, pour pouvoir les exploiter par la suite.

Concernant la partie front-end, comme dans le projet précédent j'ai utilisé **AngularJS**. Cependant concernant la partie back-end, j'ai utilisé **NodeJS** déjà présent sur l'application Pantaire.

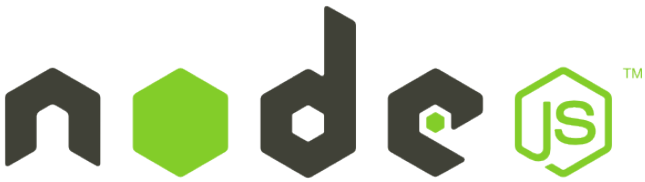


FIGURE 5.1 – Logo NodeJS

NodeJS est une plateforme basée sur le moteur v8 javascript de Google, qui permet d'executer du Javascript coté serveur. Il a été créée pour répondre à des problématiques de montée en charge. Entièremment développé en C++, il commence à être beaucoup utilisé depuis quelques années.

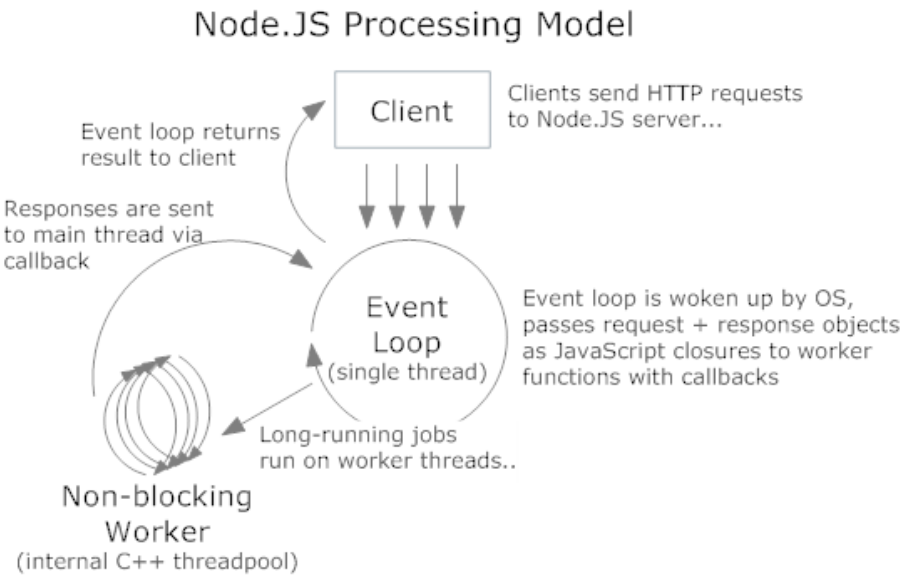
Il embarque en natif une bibliothèque HTTP, lui permettant de se comporter comme un serveur classique du type Apache, NGiNX. Cependant, l'une des particularités de NodeJS est qu'il est un "langage" asynchrone.

Asynchronous I/O

D'une manière générale, nous sommes habitués à travailler de manière synchrone. Un programme, va appeler une instruction, qui va s'éxecuter et retourner une valeur. Une fois celle-ci terminée et une autre instruction s'effectuera et ainsi de suite.

Ce différent paradigme permet à la seconde instruction de s'éxecuter avant que la première soit terminée. Ainsi plusieurs instructions sont lancées sans se bloquer les unes aux autres. Le développement en suivant ce modèle est au début assez déroutant mais par la suite très intéressant, notamment grâce à sa rapidité d'éxecution.

Ce genre de modèle est très utilisé dans les système de gestion de base de données, comme Oracle ou SQL Server.



5.2 Réflexion sur le NoSQL (Not Only SQL)

Concernant la persistence, vue la nature des données à aggréger et manipuler, une base de données MySQL n'aurait pas été assez efficace en terme de performance, c'est donc que mon choix, dans le contexte de l'application Pantaire, c'est tout naturellement porté vers le NoSQL.

NoSQL, ce terme désigne l'ensemble des bases de données qui s'opposent à la notion relationnelle des SGBDR classiques. Ces bases de données, proposent une alternative ou un complément aux fonctionnalités des bdd classiques. Les géants tels Google ou Facebook, ont mit au point ces solutions pour répondre à leurs besoins de scalabilité.

Le paradigme NoSQL, très populaire ces dernières années, redéfinit totalement les bases de données. L'unité logique n'y est plus la table, les données ne sont pas manipulées avec SQL. Il rompt également avec les différents avantages propres au SGBD tels quel la cohérence des données de la base, les mécanismes de jointures, etc. En revanche là où le NoSQL est intéressant c'est concernant la rapidité d'éxecution et le temps de réponse qui reste constant peu importe le nombre de données sauvegardées. C'est également, concernant l'accès aux données qui s'effectue d'une manière assez simple, ce qui permet de manipuler assez facilement la masse de données persistée.

Il existe plusieurs types de base de données NoSQL, nous allons nous intéresser ici uniquement à celle orientées documents.

L'ensemble des données, sont stockés sous forme de collections (l'équivalent des tables en SQL). Chaque collection, contient des documents (l'équivalent des enregistrement en SQL).

```
{
  "Subject": "Why NoSQL"
  "Author": "Lainé"
  "PostedDate": "19/01/2014"
  "Tags": ["NoSQL", "database", "scalability"]
  "Body": "Today i'm going to talk about NoSQL ..."
  "Comments": [
       {
           'date': '20/01/2014'
           'text' : 'Great post ! '
           'authorId': '02356'
       }

  ]
```

```
16      }
```

Listing 5.1 – Exemple d'un document JSON

La base de données utilisée dans ce programme s'appelle **MongoDB**, l'une des plus populaire base de données NoSQL, sous licence open source actuellement en version 2.6.

5.2.1 Pourquoi les bases NoSQL sont elles intéressantes ?

— La **productivité**. Le temps passé à concevoir et à maintenir les bases de données relationnelles, avec le mapping objet correspondant, peut-être assez long en fonction du projet. Le NoSQL peut proposer un modèle de données qui correspondra peut être mieux aux contraintes de l'application, impliquant moins de code à écrire et à débuguer simplifiant grandement les interactions avec la base de données.
— La **scalabilité**. Les bases de données relationnelles sont à la base conçues pour tourner sur une machine unique, cependant il est plus économique d'utiliser des clusters de petites machines, plus économiques, pour gérer beaucoup de données. La plupart des bases de données NoSQL sont conçues pour être déployées sur des cluster exploitant au mieux le big data.

5.3 Développement

Une fois l'ensemble des outils mis en place, j'ai pu commencer à développer le projet.
Sous la tutelle de Christophe Frézier, j'ai au fil des semaines implémenté des fonctionnalités, suivant l'architecture REST et les bonnes pratiques mise en place pour ce projet.

En plus d'utiliser **NodeJS** coté serveur, j'ai utilisé un module appelé **Restify**, qui m'a permis de construire très facilement une API REST.

```
(function(){
2     var restify = require('restify');

4     // Création d'un instance serveur
      var server = restify.createServer();
6
      // Pour une resource
8     server.get('/resource', function);
      server.post('/resource', function);
10    server.put('/resource', function);
      server.delete('/resource', function);
12
      server.listen(8081, function(){
14        console.log('Server listening at %s', server.url);
      });
16  })();
```

Listing 5.2 – Exemple d'utilisation avec restify

Le Javascript n'étant pas un language très contraignant concernant l'architecture à mettre en place, j'ai pu donc facilement créer ma propre architecture, déplacant d'un côté les modèles et de l'autre les logiques métiers dans un module à part.

Pour assurer les transactions avec la base de données noSQL, j'ai utilisé un librairie appelé **Mongoose** qui facilite largement l'utilisation avec **NodeJS**.

```
var mongoose = require('mongoose'),
    Schema = mongoose.Schema;

module.exports = (function(){

    /**
     * @type {Schema}
     */
    exports.schema = new Schema({
        nom: String,
        metriques: [{
            key: String,
            value: String
        }]
    });


    return mongoose.model('Profile', exports.schema);

})();
```

Listing 5.3 – Exemple de définition d'un schema NoSQL

```
/**
 * GET all the profile from the DB
 * @param req
 * @param res
 */
exports.getAllProfile = function(req, res){

    res.setHeader('Access-Control-Allow-Origin', '*');

    modelDispatcher.Profile.find().exec(function(err, lesProfiles){

        handleErrors(err);

        displayDebug("Getting all the profiles");

        res.send(lesProfiles);

    });

};
```

Listing 5.4 – Exemple d'accès aux données

On constate ici, qu'aucune requête de type SQL n'est utilisé, en effet MongoDB pour accéder à ses données va utiliser des fonctions du type **find** ou **findOne**. Ce type de requêtage est très semblable à celui d'une liste ou d'une collection ce qui permet de très facilement accéder aux données.

Tout au long de cette seconde mission, j'ai utilisé plusieurs outils plus ou moins nouveaux à ma connaissance me permettant d'enrichir ma méthode de travail.

5.3.1 Build pattern

Dans un soucis de gestion d'automatisation de production, le projet existant utilise des outils sur lesquels j'ai du me former pour être en conformité avec les conventions de développement.

Grunt

Pour automatiser les tâches j'ai utilisé **Grunt**, qui m'a beaucoup aidé à améliorer ma rapidité de travail. Nombre de tâches sont très répétitives dans le monde du développement logiciel, par exemple sur Pantaire, en fonction de l'environnement de déployement on désire pouvoir accéder à plus ou moins de contenu. Par le biais d'un fichier de configuration, on va assigner des variables qui seront en fonction de l'environnement de destination valorisées différement.

```
// Environnement de développelement
"dev" : {
    "serverPort" : "8081",
    "serverHost" : "127.0.0.1",
    "zabbixUrl" : "http://10.33.33.9/zabbix/api_jsonrpc.php",
    "zabbixUser" : "zefi436nfez",
    // ...
}
"prod : {
    "serverPort" : "8085",
    "serverHost" : "127.0.0.1",
    "zabbixUrl" : "http://10.33.33.9/zabbix/api_jsonrpc.php",
    "zabbixUser" : "aa234grehix",
    // ...
}
```

Listing 5.5 – Ex de configuration en fonction d'evt

Ces variables de configuration, sont utilisées dans l'ensemble de l'application. Changer l'ensemble de celles-ci dans tous les fichiers à chaque fois qu'on désire changer d'environnement serait une perte de temps énorme et les risques d'erreur seraient beaucoup plus grands.

C'est là, ou Grunt va nous aider En définissant une tâche particulière, l'outil va faire le travail à notre place et générer nos fichiers de configuration. Voici un exemple de définition de tâche.

```
// Exemple de définition
module.exports = function(grunt) {

  grunt.initConfig({
    replace: { // Nom de la tache
        dist: {
            options: { // Options
                patterns: [
                    { match: 'version', replacement: '<%= pkg.version %>' },
                    { match: 'title', replacement: '<%= pkg.name %>' },
                    { match: 'timestamp', replacement: '<%= grunt.template.today() %>' },
                    { match: 'serverPort', replacement: environment.serverPort },
                    { match: 'serverHost', replacement: environment.serverHost },
                    { match: 'zabbixUrl', replacement: environment.zabbixUrl },
                    { match: 'zabbixUser', replacement: environment.zabbixUser },
                    // ...
                ],
                force: true
            }, // Fichier cibles
            files: [
                {
                    src: [ './frontend/conf/confSkeletonFO.js' ],
                    dest: './frontend/conf/generated_conf.js'
                },
                {
                    src: [ './backend/modules/conf/confSkeletonBO.js' ],
                    dest: './backend/modules/conf/generated_conf.js'
                }
                ]
            }
```

```
        }
32  })

34  // Redéfinition de la tache défaut, pour qu'elle prenne en compte notre replacement.
    grunt.registerTask('default', ['replace'])
36  }
```

Listing 5.6 – Tâche de concaténation des fichiers

Désormais en lancant la commande *grunt replace*, et en indiquant l'environnement cible mes fichiers vont tout simplement se modifier.

Il est possible de créer des tâches très différentes, comme par exemple celle de lancer des tests unitaires, créer une nouvelle version, etc.

Bower

Dans la panoplie de l'automatisation de production Javascript, Bower est un autre module très intéressant.

Là où **Grunt**, nous permet d'automatiser, **Bower**, nous permet de gérer l'ensemble dépendances d'un projet. Dans un souci de gain de temps et de factorisation, on évite de recoder à chaque fois tout ce dont on a besoin et on utilise des librairies tierce. Ainsi, lorsqu'on commence un nouveau projet on a tendance à copier-coller toutes les librairies qu'on désire utiliser à la main. On perd donc du temps et on risque de se retrouver face à des dépendances dépréciées ou manquantes.

C'est la ou Bower entre en jeu. En créant un fichier appelé **bower.json**, contenant l'ensemble des dépendances et en lancant une commande d'installation, le petit script va télécharger automatiquement celles-ci dans la version désirée.

```
{
2     "name": "projet", // Nom du projet
      "dependencies": { // Les dépendances utilisées.
4         "knacss": "latest",
          "html5shiv": "latest",
6         "box-sizing-polyfill": "latest",
          "modernizr": "latest",
8         "jquery": "1.10.2"
      }
10 }
```

Listing 5.7 – Exempe-type de fichier JSON

Ces outils, sont semblable à **Apache Maven** ou **Gradle** dans le monde Java.

Couplé à l'utilisation d'une plateforme d'intégration continue (utilisée également dans ce projet), on peux ainsi arriver beaucoup plus facilement à gagner en productivité, automatiser l'ensemble des tâches ingrâtes que peut apporter le développement et se concentrer uniquement sur la programmation.

5.3.2 GIT

Dans le monde du développement informatique, lorsqu'on travaille à plusieurs sur le même projet et sur le même code source, on fait face à de nombreux problèmes. Les modifications provoquant des bugs, une version sur une machine qui ne fonctionne plus sur une autre, l'historisation des versions et le code correspondant à chaque patch, etc. En somme, beaucoup de problèmes, qu'il est possible de contourner à l'aide d'un logiciel de gestion de versions. Il va permettre de suivre l'évolution d'un fichier texte ligne par ligne. Il permet par exemple de sauvegarder en plus des modifications, la personne ayant modifié chaque fichier ainsi que la raison de cette modification. Il peut également fusionner deux versions d'un même fichier en évitant que l'une des deux versions soit écrasée.

Aussi bien dans le projet précédent que dans celui-ci, j'ai utilisé Git qui est un logiciel très puissant pour permettre le travail colaboratif. Chaque nouvelle fonctionnalité était enregistrée puis assignée à un commit ou une branche spécifique. De même lorsque je désirais pousser en production mon code, mergeant mon travail sur la branche master le travail était largement simplifié pour l'administrateur du code source qui pouvait très simplement voir chaque ligne de code que j'avais écrite y apposer un commentaire, qui pouvait être discuté par la suite.

5.4 Valeur ajoutée du projet

5.4.1 Bilan

Le bénéfice d'avoir pu effectuer deux missions distinctes lors d'un même stage, m'a permis de développer compétences apprise durant l'année, devenant plus polyvalent dans ma démarche professionnelle et renforcer la compréhension du développement en SSII.

Ce second projet m'a également permis de prendre du recul avec le travail précédemment éffectué. Ayant progréssé dans la mise en oeuvre des outils, je comprend après coup les méthodes que j'aurai pu mettre en place dans la réalisation de ma première mission et le temps que j'aurai gagné en utilisant celles-ci.

5.4.2 Enseignements

Sur le plan technique, en me formant à de nouvelles technologies, j'ai pu accroitre mes connaissances que ce soit en matière de base de données en utilisant le NoSQL ou encore sur la programmation serveur avec NodeJS. Concernant la méthodologie de travail, j'ai beaucoup apprécié m'exercer à l'ensemble des opérations annexes à la programmation. Avant ce stage, je n'avais pas idée des avantages à utiliser les outils d'automatisation, d'intégration continue. C'est maintenant quelque chose d'incontournable que je vais m'efforcer d'utiliser dès que j'en aurais l'occasion.

De plus dans la manière de développer, ayant été sous la tutelle d'experts techniques durant toute la durée de ce second projet, ils m'ont transmit les bonnes manières, les conventions à respecter dans le développement me permettant de beaucoup progresser sur ce point.

Et bien sûr d'un point de vue fonctionnel, j'ai pu m'instruire encore plus sur le fonctionnement d'une société et la gestion de projet. En participant à des réunions différentes du premier projet, avec des personnages différentes, j'ai agrandi mes découvertes sur les différentes synérgies qu'il peut y avoir entre les différentes entités constituantes de l'agence. Sur la manière dont SQLI utilise la méthode Agile ou le cycle de développement en V pour gérer ses projets, chaque étape me semble désormais plus simple, claire et précise.

6 — Une expérience réussie

Ce stage de fin de licence très orienté technique, me permit d'aborder un certain nombre de technologies. La liberté que j'ai eu pour remplir mes fonctions que ça soit dans le projet seating-plan ou bien Pantaire est appréciable dans la mesure ou j'ai travaillé en interne et pas directement intégré à une équipe sur une ligne de production. Les délais à respecter y sont souvent courts et des résultats plus précis y sont attendus. A contrario, le manque de structure peut s'avérer être un obstacle. Il faut une bonne autonomie, de la rigueur de la régularité pour parvenir à mener de tels projets à bien. Le plus difficile fut le premier projet, ou il a fallu tout en s'adaptant au tout nouvel environnement, répondre au mieux aux attentes de mon maître de stage pour lui donner une application qui soit au plus proche de ses attentes. A la lumière des résultats obtenus ainsi que de l'appréciation de mon tuteur entreprise tout au long du projet, je pense que les objectifs ont été atteints pour l'entreprise. Il s'agissait en effet d'avoir un outil permettant de gérer consciemment le seating des différents bâtiments. Pantaire fût tout aussi intéressant, apportant son lot de surprise, de joie et de frustration. La mission était beaucoup plus précise que sur le seating plan et nécessitait une compréhension plus appropriée du problème. Ce second projet me permit d'élargir mon champ de compétence techniques, fonctionnelles et méthodologiques.

D'un point de vue plus personnel, j'ai tout particulièrement apprécié le cadre du stage. SQLI Bordeaux est un site dynamique où la communication est aisée et l'ambiance détendue. Les différentes missions ont été intéressantes, exposant au maximum les compétences techniques tout en s'appuyant sur des besoins fonctionnels. Les outils sur lesquels j'ai pu me former, AngularJS, NodeJS et le NoSQL, diversifiant mon panel d'outil à disposition. Les notions plus gestion de projets mettant en oeuvre la méthode Agile ou le schéma développement en V, ou l'extrême programming, s'intéressant aux notions d'intégration continue. Enfin la proximité des experts techniques, présents pour m'aider lorsque j'en avais le besoin et présent pour m'apprendre de nouvelles notions, partageant leurs savoir.
Une expérience enrichissante tant sur le plan personnel que professionnel qui m'a permis de gagner en maturité dans mes méthodes de développement, d'avoir une approche différente sur les notions de gestion de projet ou de développement apprisent à l'université.

7 — Annexes

7.1 Sources

— Wikipedia d'une manière générale
— Site d'entraide pour le langage LaTeX
— Documentation MongoDB
— Documentation NodeJS
— Documentation Mongoose
— Documentation Grunt
— Documentation Jenkins
— Documentation Bower
— Documentation AngularJS
— Page wikipedia sur le NoSQL

7.2 A propos

Rapport écrit sur un système libre GNU/Linux à l'aide de LaTeX.

Code source du rapport.

Glossary

Apache Le logiciel libre Apache HTTP Server (Apache) est un serveur HTTP créé et maintenu au sein de la fondation Apache. Il permet de servir des requêtes HTTP développé pour le web. 10

API En informatique, une interface de programmation (souvent désignée par le terme API pour Application Programming Interface) est un ensemble normalisé de classes, de méthodes ou de fonctions qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels. 11, 24

architecture client-serveur L'environnement client-serveur désigne un mode de communication à travers un réseau entre plusieurs programmes ou logiciels : l'un, qualifié de client, envoie des requêtes ; l'autre ou les autres, qualifiés de serveurs, attendent les requêtes des clients et y répondent. Par extension, le client désigne également l'ordinateur sur lequel est exécuté le logiciel client, et le serveur, l'ordinateur sur lequel est exécuté le logiciel serveur. 11, 22

asynchrone L'asynchronisme désigne le caractère de ce qui ne se passe pas à la même vitesse, que ce soit dans le temps ou dans la vitesse proprement dite, par opposition à un phénomène synchrone. 11, 22

big data Les big data, littéralement les « grosses données », parfois appelées données massives, est une expression anglophone utilisée pour désigner des ensembles de données qui deviennent tellement volumineux qu'ils en deviennent difficiles à travailler avec des outils classiques de gestion de base de données ou de gestion de l'information. 24

capacity-planning Gestion de la capacité. Surveillance sur le long terme de l'ensemble du parc physique afin d'éviter le dépassement d'effectif d'une manière globale sur l'agence ou d'une manière détaillé sur un floor en particulier. 11, 16

cluster Un cluster est une grappe de serveurs (ou « ferme de calcul ») constituée de deux serveurs au minimum (appelés aussi nœuds) et partageant une baie de disques commune. 24

CTO Le directeur de la technologie (CTO, pour Chief Technology Officer en anglais) est un employé chargé de s'occuper de la direction des questions scientifiques et techniques au sein d'une organisation. 10

Debian Distribution Linux basée sur le noyau Linux et le système GNU. 10

drag'n'drop Drapg'n'drop ou glisser-déposer est dans une interface graphique une méthode consistant à utiliser une souris, pavé ou écran, pour déplacer d'un endroit à un autre un élément graphique présent sur l'écran d'un smartphone, tablette ou ordinateur. 16

framework En programmation informatique, un framework est un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel (architecture). 12, 13, 16

full stack Full Stack désigne le fait d'utiliser l'ensemble des couches d'un applicatif métier utilisant la même logique. Ici on parle de Full Stack dans le sens ou toutes les couches de l'application utilise le langage JavaScript. 21

guideline Une convention, une directive, une ligne de conduite. 14

HTTP L'HyperText Transfer Protocol, plus connu sous l'abréviation HTTP — littéralement « protocole de transfert hypertexte » — est un protocole de communication client-serveur développé pour le World Wide Web. 22

Jenkins Jenkins est un outil open source d'intégration continue, fork de l'outil Hudson après les différends entre son auteur, Kohsuke Kawaguchi, et Oracle. Écrit en Java, Jenkins fonctionne dans un conteneur de servlets tel qu'Apache Tomcat, ou en mode autonome avec son propre serveur Web embarqué. 18

jQuery UI Bibliothèque de widgets graphiques, d'animations visuelles Javascript. 16

KPI Les indicateurs clefs de performance (ICP), ou en anglais Key Performance Indicators (KPI) , sont des indicateurs mesurables d'aide décisionnelle. 21

MongoDB MongoDB (de l'anglais humongous qui peut être traduit par « énorme ») est un système de gestion de base de données orientée documents, répartissable sur un nombre quelconque d'ordinateurs et ne nécessitant pas de schéma prédéfini des données. 24, 25

moteur v8 javascript V8 JavaScript engine est un moteur JavaScript open source développé par Google au Danemark. Il est notamment utilisé dans les navigateurs Internet Google Chrome et Chromium. 22

MVC Le patron modèle-vue-contrôleur (en abrégé MVC, de l'anglais model-view-controller), tout comme les patrons modèle-vue-présentation ou Présentation, abstraction, contrôle, est un modèle destiné à répondre aux besoins des applications interactives en séparant les problématiques liées aux différents composants au sein de leur architecture respective. 12, 13

MVVM Le Modèle-Vue-VueModèle (en abrégé MVVM, de l'anglais Model View ViewModel) est une architecture et une méthode de conception utilisée dans le génie logiciel. Cette méthode permet, tel le modèle MVC, de séparer la vue de la logique et de l'accès aux données en accentuant les principes de binding et d'événement. 13

MySQL MySQL est un moteur de base de données relationnel écrit en PHP et open-source. 11, 23

NGiNX Nginx [engine x] est un logiciel libre de serveur Web (ou HTTP) ainsi qu'un proxy inverse écrit par Igor Sysoev, dont le développement a débuté en 2002 pour les besoins d'un site russe à très fort trafic. 22

NoSQL NoSQL (Not only SQL en anglais) désigne une catégorie de systèmes de gestion de base de données (SGBD) qui n'est plus fondée sur l'architecture classique des bases relationnelles. L'unité logique n'y est plus la table, et les données ne sont en général pas manipulées avec SQL. 23, 24

ORM Un mapping objet-relationnel (en anglais object-relational mapping ou ORM) est une technique de programmation informatique qui crée l'illusion d'une base de données orientée objet à partir d'une base de données relationnelle en définissant des correspondances entre cette base de données et les objets du langage utilisé. On pourrait le désigner par « correspondance entre monde objet et monde relationnel ». 15

patch Un patch ou correctif est une section de code que l'on ajoute à un logiciel, pour y apporter des modifications : correction d'un bug, traduction, crack. 19, 27

PHP Hypertext Preprocessor3, plus connu sous son sigle PHP (acronyme récursif), est un langage de programmation libre4 principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP3, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale. PHP est un langage impératif orienté objet. 10

REST REST (REpresentational State Transfer) est un style d'architecture pour les systèmes hypermédia distribués, créé par Roy Fielding en 2000 dans le chapitre 5 de sa thèse de doctorat. 11, 13, 24

Seating-plan Plan de table. Dans le contexte du stage : plan des bureaux. 10

SGBDR Un système de gestion de base de données (abr. SGBD) est un logiciel système destiné à stocker et à partager des informations dans une base de données, en garantissant la qualité, la pérennité et la confidentialité des informations, tout en cachant la complexité des opérations. 23

SonarQube SonarQube (précédemment Sonar) est un logiciel libre permettant de mesurer la qualité du code source en continu. 21, 22

SSH Secure Shell (SSH) est à la fois un programme informatique et un protocole de communication sécurisé. Le protocole de connexion impose un échange de clés de chiffrement en début de connexion. 18

versionning Un logiciel de gestion de versions (ou VCS en anglais, pour Version Control System) est un logiciel qui permet de stocker un ensemble de fichiers en conservant la chronologie de toutes les modifications qui ont été effectuées dessus. Il permet notamment de retrouver les différentes versions d'un lot de fichiers connexes. 16

virtual Machine machine virtuelle est une illusion d'un appareil informatique créée par un logiciel d'émulation. 10

W3C Le World Wide Web Consortium, abrégé par le sigle W3C, est un organisme de normalisation à but non lucratif, fondé en octobre 1994 chargé de promouvoir la compatibilité des technologies du World Wide Web. 11