Projet Industriel - Cahier des charges

Application présentant le tableau de bord relatif à l'activité d'une officine

Najla SABBOURI - Yoann RAUCOULES- Anthony YOUSSEF - Haroun ZIDANE

Année universitaire 2016-2017

Sommaire

1 - Introduction

1.1 A propos de ce document

Ce document a pour but de définir la démarche de la réalisation du projet industriel. L'objectif de ce projet est de mettre en place un tableau de bord présentant l'activité d'une officine avec les nouvelles technologies du web. Ce projet a été proposé par l'entreprise Pharmagest à des élèves de Telecom Nancy. Il s'inscrit dans la volonté de Pharmagest de fournir toujours plus de services à ses clients, l'outil à réaliser permet d'apporter une réelle valeur ajoutée à une solution déjà existante appelée LGO. Ce dernier contient diverses informations concernant les clients, les produits ou encore les résultats comptables. L'intérêt d'un tableau de bord est de mettre en forme ces données afin que le client puisse les consulter de manière simple et dynamique dans un outil web.

Ce cahier des charges présente dans un premier temps un aperçu global du produit, puis les besoins fonctionnels du projet et enfin les besoins techniques.

1.2 Portée du produit

L'application web développée est un tableau de bord relatif à l'activité d'une officine. Elle doit mettre à disposition de multiples fonctions aux utilisateurs tout en respectant les droits et les libertés de chacun. Elle gère également les restrictions d'accès de chaque utilisateur. Dans un premier temps, il est nécessaire que l'application dispose d'un module d'authentification, de type nom d'utilisateur et mot de passe, afin d'identifier l'utilisateur. Cet utilisateur pourra alors effectuer diverses actions qui sont la gestion de son compte, la consultation et la personnalisation de sa page d'accueil ainsi que la consultation des différentes pages mettant en valeur les données du LGO. Grâce à ces données et à l'aide de graphiques, l'utilisateur sera guidé dans sa prise de décision ou pourra simplement avoir des informations statistiques relatives à l'activité de l'officine. Les accès aux pages contenant les données peuvent être différentes selon les restrictions d'accès de l'utilisateur. En effet, il est possible qu'un opérateur ne doive pas avoir accès aux mêmes données que le gérant.

1.3 Public concerné et vue d'ensemble de ce document

L'application est principalement destinée aux gérants des pharmacies mais elle peut aussi être destinée aux divers opérateurs intervenant dans l'activité de l'officine. Elle a pour but d'aider et de conseiller les gérants dans leurs choix stratégiques concernant la gestion de leur activité. Grâce aux informations mises en valeur par le tableau de bord, le gérant sera à même de prendre des décisions sur sa gestion de produits, la définition de ses objectifs de vente ou encore ses relations avec les fournisseurs et les clients. Concernant les opérateurs, l'application pourra leur permettre d'avoir différentes informations sur les produits ou encore sur leurs résultats personnels.

1.4 Définitions, acronymes et abréviations

- GDR (Générateur De Requêtes) : outil utilisé afin de générer des requêtes vers la base de données
- LGO (Logiciel de Gestion aux Officines) : logiciel mis à disposition des officines par Pharmagest afin de gérer l'activité de ces dernières
- BD : base de données
- IDE : *Integrated Development Environment*, Environnement de développement

1.5 Conventions de rédaction du document

- **Titre chapitre :**

fonte : Times New Roman gras 36 points

- **Sous-titre :**

fonte : Arial gras 14 points

- **Paragraphe :**

fonte : Arial 11 points

1.6 Références et remerciements

Type	Référence

2 - Description globale

2.1 Perspective du produit

Le produit sera utilisé dans un contexte professionnel, en parallèle d'un produit qui est un logiciel de gestion aux officines déjà utilisé par de nombreuses pharmacies et fourni par Pharmagest. Actuellement la société vend un système d'informations qui utilise les données de la pharmacie pour générer des comptes-rendus d'activité. Le tableau de bord a donc pour objectif d'utiliser ces données et de les mettre à disposition dans un cadre agréable pour l'utilisateur. Pharmagest veut proposer un produit de consultation d'indicateurs de performances simple d'utilisation, ergonomique et respectant la confidentialité des données. Les utilisateurs finaux envisagés sont : les gérants de pharmacies ainsi que leurs collaborateurs.

2.2 Fonctionnalités du produit

La fonctionnalité principale du produit est de permettre la consultation d'informations relatives à l'activité de l'officine. Néanmoins il est nécessaire de prendre en compte d'autres actions possibles concernant la gestion du tableau de bord et des utilisateurs.

Parmi les fonctionnalités du produit, on retrouve :

- l'authentification des utilisateurs
- la déconnexion des utilisateurs du système
- la gestion du compte utilisateur
- la modification des restrictions d'accès des opérateurs par le gérant des officines
- la personnalisation de l'écran d'accueil
- l'utilisation d'une fonction de recherche (notamment pour la recherche d'indicateurs)
- la consultation d'informations relatives aux clients (fidélité, nombre de clients par unité de temps...)
- la consultation d'informations relatives aux fournisseurs (fidélité, nombre de commandes par unité de temps...)
- la consultation d'informations relatives aux opérateurs (rentabilité, nombre de ventes réalisés par unité de temps…)

- la consultation d'informations relatives aux produits (meilleur produit, nombre de ventes par unité de temps, gestion des stocks…)
- la consultation d'informations relatives aux ventes (sur ordonnance, remboursables, mode de paiement…)
- la consultation d'informations comptables et financières

2.3 Utilisateurs

Caractéristiques de chaque utilisateur :

- **Gérant** : nom, prénom, date de naissance, identifiant, mot de passe, email, numéro de téléphone
- **Opérateurs** : nom, prénom, date de naissance, identifiant, mot de passe, email, numéro de téléphone

2.4 Environnement d'exécution

La volonté de Pharmagest est d'offrir une interface graphique utilisable sur mobile, tablette et ordinateur. La solution choisie est donc d'utiliser un navigateur web comme environnement d'exécution. Cette solution permet une grande portabilité, puisque la majorité des terminaux visés possèdent un navigateur web intégré (*Safari pour Apple, Android Browser pour Android, IE ou Edge pour Windows*…), et d'autres sont disponibles en téléchargement (*Chrome*, *Firefox*, *Opera*...). De plus, elle est indépendante du système d'exploitation de l'utilisateur. Cependant, il faut prendre en compte les problèmes entre les différentes versions des navigateurs ainsi que les différences d'interprétation des différents navigateurs.

2.5 Contraintes de conception et d'implémentation

Nous allons différencier différents types de contraintes :

- **Contraintes d'organisation**

L'environnement de développement choisi est imposé, afin de s'adapter à celui déjà mis en place par Pharmagest. L'IDE utilisé pour le développement *Java* sera donc *Eclipse Neon* auquel a été ajouté le plug-in *Spring IDE*. Pour ce qui est de la partie web, le choix de l'environnement de développement est laissé à libre appréciation, la décision n'impactant pas le reste de l'équipe, il est préférable que chacun des développeurs utilisent l'éditeur avec lequel il

ou elle a le plus d'affinités. Pour ce qui est de l'organisation du développement nous allons, une fois encore, nous adapter à la structure déjà mise en place par Pharmagest. Le code sera déposé sur un dépôt *git* fournit par nos encadrants industriels.

- **Contraintes interface Homme-Machine**

L'application est destinée à des professionnels non-initiés à l'informatique, il est donc impératif que le produit fournisse une interface simple d'utilisation et intuitive pour ses utilisateurs. Il est donc important qu'une réflexion soit portée sur l'ergonomie du produit afin de réaliser un tableau de bord dans lequel les utilisateurs peuvent trouver l'information qu'ils souhaitent sans difficulté et en effectuant le moins d'actions possibles. Pour cela, une maquette d'une vue de l'application a été fournie par l'entreprise Pharmagest dans le but d'aiguiller les réflexions autour de l'interface à mettre en place. De plus, l'interface doit respecter un code couleur fourni par les encadrants industriels dans le but de garder une cohérence visuelle entre les différentes applications destinées aux clients.

L'application sera décomposée entre trois parties :

- La première est une barre d'actions contenant des informations sur l'utilisateur ainsi que des fonctionnalités comme par exemple l'action de déconnexion de l'application. Elle se trouvera en haut de la page et sera présente sur toutes les pages.

- La seconde est un menu qui contient différents onglets afin de passer d'un contenu de l'application à un autre. La page active sera mise en valeur dans la liste des onglets afin que l'utilisateur puisse se rappeler de la page qu'il est en train de consulter. Le menu se trouvera dans la partie gauche de l'application, sera présent sur toutes les pages et pourra être masquer/afficher par l'utilisateur. Il est important que ce menu s'adapte en fonction de la taille de l'appareil. Par exemple sur plate-forme mobile, le menu se présentera sous la forme d'une fenêtre dynamique s'ouvrant depuis la gauche de l'écran, tandis que sur PC le menu se trouvera à gauche et sera statique.

- La dernière partie occupera le reste de l'écran et contiendra le contenu sélectionné par l'utilisateur, ce contenu est dynamique et s'adapte à l'onglet sélectionné dans le menu par l'utilisateur.

- **Contraintes d'exploitation et de développement**

La maintenance et la gestion de l'application ne doivent pas générer de coûts de fonctionnement supplémentaires. Les services informatiques pourront prendre la relève sans aucune difficulté. Pour faciliter la maintenance du système par les services informatiques, il faudra commenter le code source un maximum. L'administrateur doit pouvoir avoir accès à tout moment à l'historique de l'application et voir toutes les maintenances effectuées ainsi que tous les problèmes rencontrés.

2.6 Manuel utilisateur

Un manuel utilisateur papier et numérique doit être fourni auprès du gérant et de tous les opérateurs de l'officine puisqu'ils sont susceptibles également d'utiliser le tableau de bord. Il expose toutes les fonctionnalités de l'application en détail. De plus, il explique la procédure de connexion et de déconnexion et fournit les coordonnées du support technique en cas de problème. Tous les utilisateurs de l'application pourront trouver dans ce manuel les informations qu'ils peuvent consulter selon leur restriction d'accès.

2.7 Hypothèses et dépendances

Comme clairement indiqué lors des premières réunions de conception, les utilisateurs finaux n'ont pas, ou très peu, d'expérience en informatique. De plus, il est possible qu'aucun système de gestion de profil ne soit déjà implémenté. Nous supposons aussi que chaque pharmacie possède une connexion haut débit à internet et que les navigateurs web utilisés sont à jour. Enfin, le tableau de bord s'appuie sur un serveur qui est le seul à communiquer avec la base de données. Ce système est déjà implémenté et fonctionnel et ne nécessitera aucun développement annexe de notre part.

3 - Besoins spécifiques

3.1 Besoins externes

3.1.1 Interface utilisateur

L'interface utilisateur se décompose en trois parties (fig. 1). La première située en haut de l'écran est la barre d'actions, sur la gauche se trouve le menu de sélection du contenu et au centre de l'écran, on retrouve le contenu sélectionné par l'utilisateur.

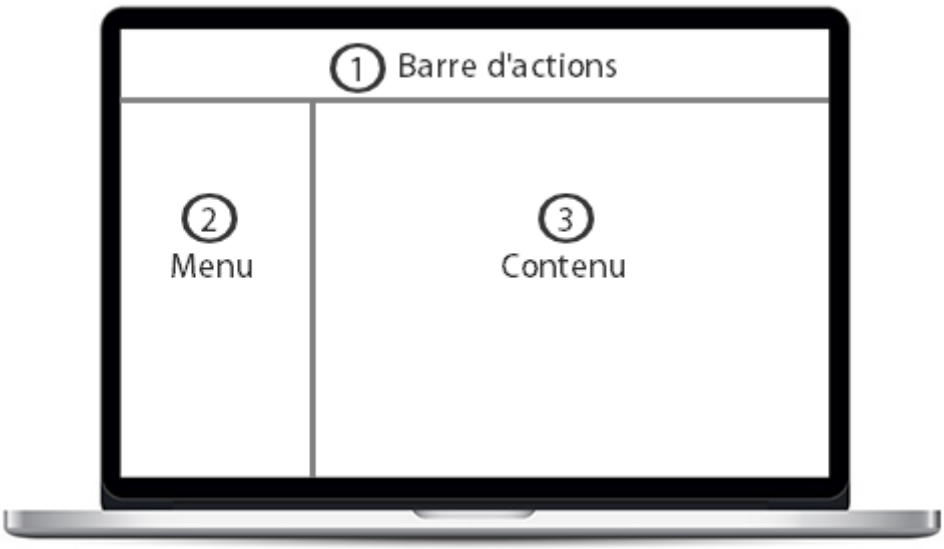


Figure 1. Schéma de l'interface utilisateur

1) **Barre d'actions** :

La barre d'actions affiche les informations de l'utilisateur authentifié, elle permet aussi à l'utilisateur de gérer son compte et ses paramètres (fig. 2).

Figure 2. Barre d'actions

2) Menu :

Le menu permet à l'utilisateur de choisir le contenu à afficher dans le centre de la page. Il contient plusieurs sections cliquables afin d'afficher dynamiquement le contenu (fig 3.).

Figure 3. Menu de navigation de l'application

3) **Contenu** :

Le contenu est la partie de l'interface dans laquelle les informations seront disponibles et présentées à l'utilisateur (fig. 4). Le contenu est dynamique et change en fonction de la section sélectionnée dans le menu. L'utilisateur peut interagir avec les différentes informations présentées dans la vue.

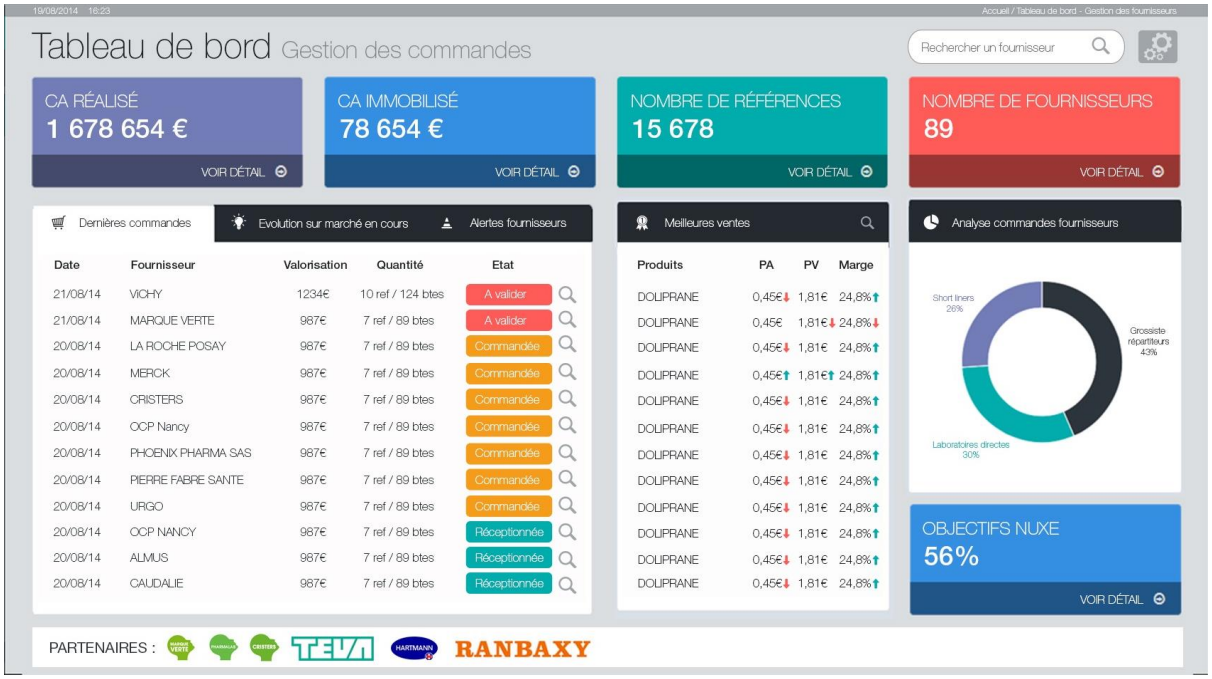


Figure 4. Visualisation du contenu de l'application

3.1.2 Interfaces matérielles

L'utilisateur utilise l'application via le navigateur Web. L'application fait ensuite le lien avec le serveur qui lui-même accède à la base de données.

Il s'agit d'une architecture *3-tier* qui permet de diviser l'ensemble de l'application en 3 couches :

- la partie client auquel l'utilisateur a accès via le navigateur (qui est la seule partie sur laquelle nous travaillerons)
- la partie traitement, aussi appelée « couche métier », qui correspond au serveur et qui assure le lien entre les couches supérieure et inférieure du modèle
- la partie qui permet d'accéder aux données (la BD), la dernière couche du modèle.

Schématiquement, le client envoie une requête au serveur, qui la traite pour pouvoir extraire les données nécessaires depuis la base de données, puis calcule et renvoie le résultat au client.

Il est important de noter que seul le serveur communique avec le client et la base de données, l'intérêt principal de ce modèle est justement la séparation de ces deux couches pour pouvoir assurer l'intégrité et la validité des données qui sont échangées. De plus, les SGBD utilisés peuvent être multiples (jusqu'à maintenant, Oracle, MySQL et PostgreSQL mais il est possible que cela évolue), et ce modèle nous permet de ne pas nous préoccuper de cette variation. En effet, le serveur, avec un conteneur Web ou un serveur d'application, est le seul à interagir avec la BD et échange les requêtes sous forme normalisée et accède aux données de manière uniforme : le couplage entre serveur et BD est faible.

3.1.3 Interfaces logicielles

La première, et aussi l'une des seules contraintes imposée sur ce projet, est le développement sous AngularJS 1. C'est un framework front-end JavaScript intuitif et facile à prendre en main qui permet de développer des applications Web dynamiques. Il permet aussi de gérer les authentifications qui sont des *session cookie* inclus dans le header de chaque requête HTTP. De plus, le tableau de bord doit être *responsive design*, d'où l'utilisation de Bootstrap 3 pour assurer la portabilité entre les différents navigateurs et appareils. Enfin, il faudra faire apparaître des éléments graphiques pour mettre en valeur les données sur le tableau de bord, comme par exemple des diagrammes. Il sera donc nécessaire d'utiliser une librairie qui apporte cette fonctionnalité.

L'ensemble des données utilisées se trouvent dans une BD mise en place et gérée par Pharmagest. Afin de rendre le tableau de bord utilisable nous devons donc faire en sorte de récupérer ces données à l'aide de services REST.

3.1.4 Interfaces de communication

Les données seront échangées entre le client et le serveur via le protocole HTTP en passant par Internet. Les requêtes HTTP contiendront les informations concernant l'utilisateur sous forme de jeton de connexion et permettront de charger le contenu du serveur et de communiquer avec ce dernier. Les données contenues dans les requêtes HTTP seront au format JSON. De plus, le système utilisera une architecture REST pour la communication client-serveur.

3.2 Besoins fonctionnels

Gérant:

- **Pouvoir saisir ses identifiants pour se connecter:**

L'authentification et la déconnexion de l'utilisateur sont assurées par le produit.

- **Pouvoir gérer le compte utilisateur:**

L'utilisateur a la possibilité de modifier ses informations personnelles (en cas de changement d'adresse, photo de profil, etc..).

- **Pouvoir modifier les restrictions d'accès des différents opérateurs:**

Le gérant doit pouvoir modifier les droits d'accès aux différents opérateurs de l'officine.

- **Personnaliser l'écran d'accueil:**

Le gérant doit pouvoir personnaliser son écran d'accueil et choisir les éléments qu'il souhaite voir apparaître en priorité dans sa page d'accueil.

- **Pouvoir utiliser la fonction de recherche:**

Le gérant doit pouvoir effectuer facilement des recherches sur l'ensemble des indicateurs de performance de son officine, par exemple à l'aide d'une barre de recherche.

- **Consulter les informations relatives aux clients:**

Le gérant peut consulter les informations relatives à tous ses clients à tout moment telles que :

- leur fidélité
- le panier moyen
- les médecins consultés
- l'évolution de la fréquentation du client par rapport à l'année passée.

- **Consulter les informations relatives aux fournisseurs:**

Le gérant doit pouvoir accès à toutes les informations relatives à ses fournisseurs à tout moment telles que :

- les commandes passées
- le produit le plus vendu
- le délai moyen et la fréquence de livraison.

- **Consulter les informations relatives aux opérateurs de l'officine:**

Le gérant peut visualiser à tout moment les informations relatives aux opérateurs de son officine telles que :

- le nombre de ventes réalisés par unité de temps pour chaque employé
- le chiffre d'affaires par opérateur
- la comparaison de la rentabilité des opérateurs.

- **Consulter les informations relatives aux produits:**

Le gérant doit pouvoir accès à toutes les informations concernant les produits de son officine telles que :

- le nombre de ventes du produit par unité de temps
- la gestion des stocks
- le produit le plus vendu
- le type de produits (médical, paramédical, matériel)
- la quantité par panier.

- **Consulter les informations relatives aux ventes:**

Le gérant peut visualiser à tout moment les informations relatives aux ventes de son officine telles que :

- le mode de paiement
- les ventes sur ordonnance ou non
- les ventes de produits remboursables ou non
- les ventes de produits génériques / princeps
- les ventes de produits de parapharmacie / pharmacie

- **Consulter les informations comptables et financières de l'officine:**

Seul le gérant doit pouvoir accès aux informations comptables et financières de son officine (chiffre d'affaires, résultat net comptable…).

Opérateurs :

- **Pouvoir saisir leurs identifiants pour se connecter:**

L'authentification et la déconnexion de l'utilisateur sont assurées par le produit.

- **Pouvoir gérer le compte utilisateur:**

L'utilisateur a la possibilité de modifier ses informations personnelles (en cas de changement d'adresse, photo de profil, etc..).

- **Personnaliser l'écran d'accueil:**

Les opérateurs doivent pouvoir personnaliser leur écran d'accueil et choisir les éléments qu'ils souhaitent voir apparaître en priorité dans leur page d'accueil.

- **Pouvoir utiliser la fonction de recherche:**

Les opérateurs doivent pouvoir effectuer facilement des recherches sur l'ensemble des indicateurs de performance auxquels il a accès, par exemple à l'aide d'une barre de recherche.

- **Consulter les informations relatives aux clients:**

Le gérant peut consulter les informations relatives à tous ses clients à tout moment telles que :

- leur fidélité
- le panier moyen
- les médecins consultés
- l'évolution de la fréquentation du client par rapport à l'année passée.

- **Consulter les informations relatives à leurs résultats et à ceux de l'officine:**

Les opérateurs peuvent visualiser à tout moment les informations de leur activité au sein de l'officine telles que :

- le nombre de ventes réalisés par unité de temps
- le chiffre d'affaires réalisé par unité de temps

- **Consulter les informations relatives aux produits:**

Les opérateurs doivent pouvoir accès à toutes les informations concernant les produits de l'officine telles que :

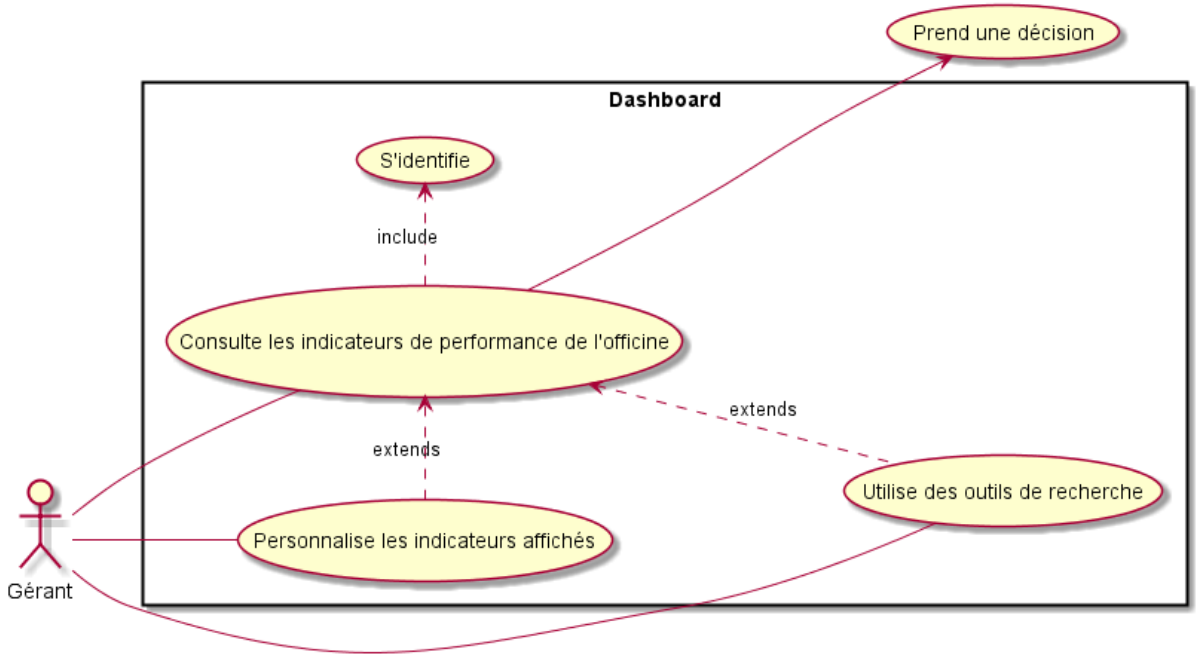
- le nombre de ventes du produit par unité de temps
- la gestion des stocks
- le produit le plus vendu
- le type de produits (médical, paramédical, matériel)
- la quantité par panier.

3.3 Besoins comportementaux

Diagramme des cas d'utilisation

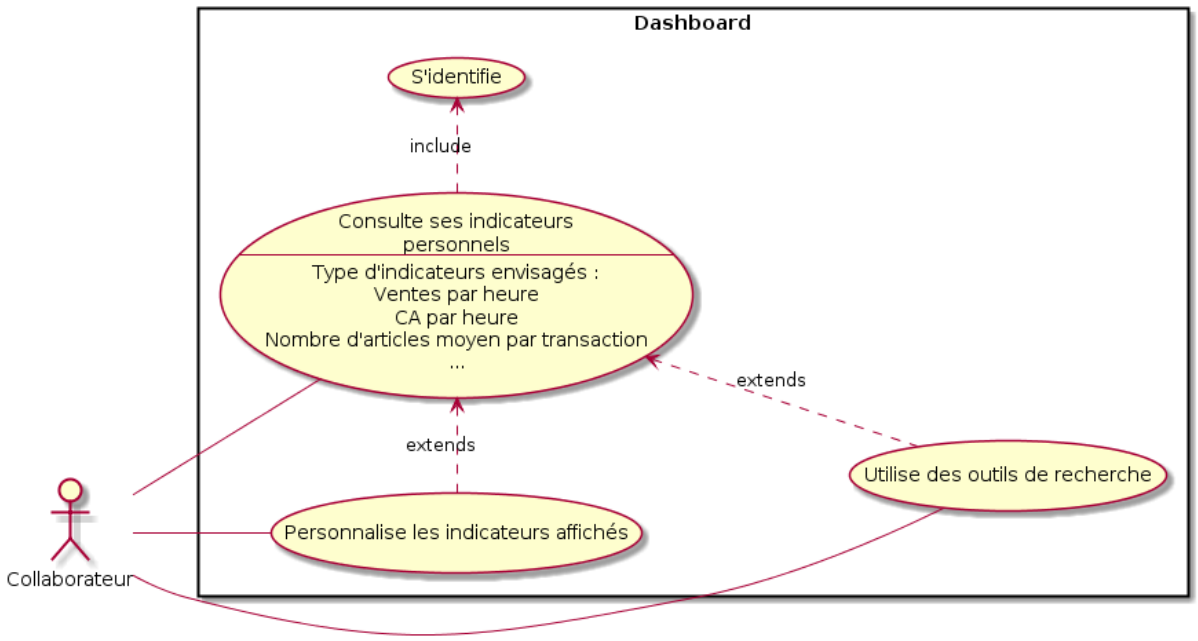
Use case n°1

Utilisation du tableau de bord par le gérant de la pharmacie. Consultation des indicateurs économiques, de performances des collaborateurs et des informations clients



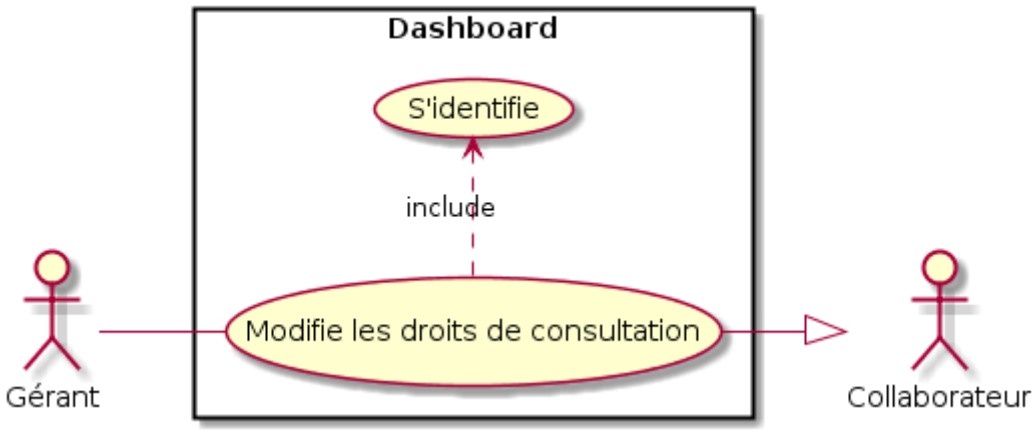
Use case n°2

Utilisation du tableau de bord par un collaborateur de la pharmacie. Consultation de ses performances de vente.



Use case n°3

Paramétrage des profils. Modification des droits de consultation par le gérant.



4- Autres besoins non fonctionnels

4.1 Besoins liés aux performances

L'application doit pouvoir répondre à un besoin spécifique et spontané de l'utilisateur, depuis son domicile mais aussi à l'officine lors d'une vente à un patient. Pour cette raison, l'accès aux informations sur les stocks du commerce et sur les dossiers des patients doit pouvoir être effectué en un temps non contraignant pour l'utilisateur : la réponse doit être renvoyée moins de 3 secondes après la requête.

Les pannes du système et les maintenances doivent être minimisés, le service doit être disponible à 99%. Toutefois, la stabilité du système repose entièrement sur la fiabilité des bases de données et des serveurs déjà mis en place. La partie *front-end* doit donc prévoir les cas possibles d'échecs des requêtes en cas de défaillance des serveurs.

4.2 Besoins liés à la sécurité

Le gérant et les opérateurs se connectent à l'application grâce à leur identifiant et leur mot de passe. Au vu de la sensibilité des données, aucune connexion « invité » ne pourra être faite, seuls les utilisateurs enregistrés peuvent accéder à l'application.

De plus, différents niveaux de droits sont à prévoir pour permettre la restriction de certaines données. Les opérateurs n'ont accès qu'aux informations concernant les clients, les produits ainsi que les statistiques de leurs ventes personnelles. Le gérant quant à lui a accès à la totalité des données disponibles : clients, produits, ventes par opérateur ou à l'échelle de l'officine, fournisseurs et informations comptables sur son entreprise.

4.3 Besoins liés à la qualité du produit.

- L'application doit être *responsive design* et permettre une prise en main facile et rapide. Lorsqu'un utilisateur découvre l'application, son fonctionnement doit lui sembler intuitif c'est-à-dire qu'il ne faut pas surcharger l'interface graphique avec trop de boutons et d'options. Le manuel utilisateur doit répondre à toutes les interrogations ou problèmes d'utilisation du produit.
- Le système doit éviter toute erreur bloquante.
- L'application doit être capable de s'adapter à tous types d'écrans, aussi bien de PC, tablette ou smartphone afin de permettre à chaque utilisateur de la consulter quand il le souhaite.
- En cas de saisie invalide d'une URL sur l'application, l'utilisateur doit être soit redirigé vers la page d'accueil, soit être redirigé vers une page l'informant que la page n'est pas disponible.
- L'application a une disponibilité quasi-permanente, dans le cas où celle-ci n'est plus disponible de manière imprévue ou non, les clients doivent en être informés le plus rapidement possible.

Appendix A

Gestion de projet

Afin de gérer au mieux ce projet, nous avons mis en place un drive (Google Drive) où chaque membre peut y déposer ces travaux. Nous utilisons également un outil de gestion de projet en ligne “Trello” afin de mieux gérer l’organisation du groupe et la répartition des tâches. Pour le compte-rendu, nous avons décidé d’utiliser un google document afin de faciliter le partage et la répartition de la rédaction des parties.

Nous avons passé environ 40 heures sur la recherche et le développement des besoins, 3 heures sur l’élaboration des diagrammes de cas d’utilisation et de séquences et 20 heures sur la rédaction du rapport.

Appendix B

Etat de l'art JavaScript Framework

Présentation d'AngularJS

AngularJS fait partie des frameworks front-end Javascript qui sont apparus sur le marché ces dix dernières années. Parmi ces frameworks, on peut citer Backbone.js, Ember.js ou Knockout.js, qui sont similaires à Angular.js. AngularJS est développé par Google et permet de développer facilement des applications web dynamiques grâce à une ergonomie simple et une prise en main intuitive. Sa première version a été publiée en 2009 et le framework est diffusé sous une licence MIT qui est une licence de logiciel libre, open-source mais non copyleft. On peut donc y inclure des modifications sous d'autres licences même non libres. La licence donne à toute personne recevant le logiciel le droit illimité de l'utiliser, le copier, le modifier, le fusionner, le publier, le distribuer, le vendre et de changer sa licence. La seule obligation est de mettre le nom des auteurs avec la notice de copyright. AngularJS est compatible avec tous les navigateurs, à l'exception des versions d'Internet Explorer inférieur à la version 8.

Utilisation d'AngularJS

De manière générale, l'utilisation d'un framework permet de structurer le code et impose une architecture à respecter. Il doit être capable de s'adapter à des projets subissant des évolutions continues, être facile à prendre en main et à maintenir, permettre un gain de temps au niveau du développement et avoir un support conséquent en cas de problèmes.

AngularJS est une librairie écrite en Javascript et est distribuée sous forme d'un fichier Javascript. Elle peut être ajouter à une page HTML en utilisant la balise <*script*>. Le framework a deux principales fonctionnalités. La première est qu'il étend les balises et attributs HTML grâce à l'utilisation de directives (*ng-directives*). Les directives sont la liaison entre le DOM et le contrôleur et permettent d'identifier un élément afin que le contrôleur y applique sa logique. La seconde fonctionnalité est qu'il lie les données au code HTML de manière bidirectionnelle grâce aux expressions (*{{ expression }}* ou *ng-bind*).

AngularJS est basé sur le patron d'architecture MVC (parfois appelé MVVW pour model-view-viewmodel pour ce type de framework) dans lequel la vue est représentée par le code HTML étendu, le modèle par les *scopes* et des contrôleurs écrits en Javascript permettant de définir les actions de la vue sur le modèle et inversement. Le framework a pour intérêt de surveiller les objets Javascript afin d'en détecter les modifications ce qui lui permet de réaliser un *data-binding* bidirectionnel permettant la synchronisation automatique des modèles et des vues. Le principe est que les données saisies par le client engendrent une mise à jour du contrôleur qui est lui-même lié à la vue. AngularJS inclut une partie de la librairie jQuery appelée jQLite, il est néanmoins possible d'utiliser la librairie jQuery dans son intégralité en la chargeant. Ce framework a aussi la spécificité de pouvoir ajouter facilement des modules développées par des tierces afin d'ajouter des fonctionnalités au framework. Parmi ces modules, on trouve des modules de traduction de pages web, de redirection entre les pages, de composants graphiques adaptés à l'utilisation du framework et bien d'autres. AngularJS a pour rôle de charger les modules, l'utilisateur n'a pas à gérer les dépendances entre les modules, il doit simplement les appeler et le framework se charge de les instancier et de les injecter. C'est le même mécanisme qui est utilisé pour déclarer des services dans une application. Ce concept s'appelle l'injection de dépendances.

AngularJS dans le projet industriel

L'utilisation d'AngularJS nous a été imposée par l'entreprise Pharmagest afin de développer un tableau de bord pour la gestion des officines qui sera une application web dynamique. La version utilisée dans le projet est la première version d'AngularJS, en effet, au moment de la définition du sujet la deuxième version n'était pas disponible dans sa version finale. De plus, Pharmagest a déjà utilisé la première version pour d'autres projets. L'utilisation d'un framework tel que AngularJS va nous permettre d'appréhender un projet complètement front-end en se basant une architecture MVC déjà définie par le framework et en déclarant des vues dynamiques de manière simple. L'intérêt du framework est de décomposer efficacement les différentes parties du projet que sont les vues, contrôleurs et modèles afin d'obtenir un environnement lisible et rapide à mettre en place. De plus, AngularJS apporte une abstraction du DOM qui permet de le manipuler plus facilement et donc facilite la maintenabilité et la testabilité du code.

AngularJS et les autres frameworks Javascript

Dans ce paragraphe sont exposés les avantages et les inconvénients des principaux frameworks Javascript.

Backbone.js :

Backbone.js est un framework plus verbeux qu'AngularJS. Il est simple, rapide, léger et bien documenté.

Malgré tout, il éprouve quelques lacunes. Il atteint vite ses limites, en effet il devient rapidement nécessaire d'ajouter des modules externes afin de compléter le framework. Dans le même temps, il n'est pas assez structurant car très modulaire.

Ember.js :

Ember.js est un framework proche d'AngularJS, notamment en terme de performance. Il contient des fonctionnalités telles que la gestion des données ou le routage entre les vues. Comme AngularJS, Ember inclut le data-binding bidirectionnel.

Néanmoins, ce framework reste complexe et donc difficile à prendre en main. Il est aussi relativement lourd, ce qui est un gros désavantage pour une utilisation sur plate-forme mobile. De plus, ce framework a subi quelques instabilités ce qui a provoqué une diminution de ses utilisateurs et de sa communauté.

Knockout.js :

Knockout est simple, pratique et intuitif. Il se trouve que ce n'est pas réellement un framework à part entière, il est plutôt considéré comme une librairie de data-binding.

Les fonctionnalités que propose Knockout sont néanmoins très limitées particulièrement concernant les appels aux serveurs. Il est possible de gérer des données au format JSON mais en comparaison au service $http d'AngularJS, le service de Knockout est pauvre.

React.js :

ReactJS est un framework développé par Facebook. C'est le framework qui se rapproche le plus d'AngularJS, même si ce dernier reste le plus en vogue chez les développeurs. La grande différence avec AngularJS est que React ne suit pas le patron d'architecture MVC, il a malgré cela d'autres avantages. Il permet de créer facilement des composants UI et offre un mécanisme très efficace pour le rendu HTML que l'on appelle le virtual DOM. Il peut s'interfacer avec d'autres outils comme Backbone.js par exemple. Ce

framework a connu une ascension très forte dans le monde du web ces deux dernières années et est de plus en plus utilisé par les professionnels.

Angular.js :

AngularJS a la particularité d'être très complet. Ces deux principales fonctionnalités que sont le 2-way data-binding, qui permet le rafraîchissement des données, et les directives, qui permettent une manipulation propre du DOM et une grande réutilisabilité du code, sont des atouts majeurs vis-à-vis de ces concurrents. De plus, il structure l'application grâce à l'utilisation de *modules*, de *contrôleurs*, de *services* et de *factories*. Ces avantages rendent les applications extensibles et très modulaires. Les templates s'écrivent entièrement en HTML, ce qui n'est pas le cas des autres frameworks qui nécessitent d'apprendre d'autres syntaxes. Un autre avantage se trouve dans l'injection de dépendances que propose AngularJS, il permet à l'utilisateur de se détacher de l'instanciation des modules et des services qui est alors gérée par le framework. Enfin, AngularJS est le framework le plus populaire des frameworks Javascript, on trouve donc une documentation détaillée, un support complet ainsi qu'une communauté très active.

Le data-binding proposé par AngularJS est très puissant mais est très gourmand en ressources, ce qui peut être problématique lors de son utilisation sur une plate-forme mobile, bien que des frameworks de création d'applications mobiles hybrides sont basés sur AngularJS, on citera par exemple Ionic. Concernant les directives, elles sont très pratiques, mais une utilisation avancée des directives peut s'avérer complexe.

Liens utiles

- **http://docs.angularjs.org/api/ : Documentation officielle**
- **http://books.google.fr/books/about/AngularJS.html?id=-PvAMgEACAAJ&redir_esc=y : Livre "AngularJS", par Brad Green, l'un des fondateurs d'AngularJS**
- **http://www.frangular.com/ : Communauté française d'AngularJS**
- **http://angular-ui.github.io/ : Regroupe des modules AngularJS**
- **http://egghead.io/ : Vidéos sur les nouvelles technologies front end.**

Appendix C

Charting Libraries

Nom	Lien	Plate-formes	Données en entrée	Personnalisation	Charts dispo	Performance	Export	Interactivité	Prix / Licence	Support
amCharts	https://www.amcharts.com/	Firefox, Internet Explorer 6+, Google Chrome, Safari, Opera, iPhone, iPad	JSON	Zoom, Annotations, Combination of charts, Data labels, Date-time axis, Dynamic charts, Export files, Interactive, Print, Text Rotation for Labels	Line, Timeline, Scatter, Area, Pie, Donut, Bullet, Radar, Funnel, Gantt, Bar, Grouped	size = 190 KB, SVG or VML for old IE	Yes PNG, JPG, SVG, PDF	Legends, Mouse Over, onClick	$900 / Single App or SaaS website license	Support service (tickets) / amPlus (priority support additional payment)
CanvaJS	http://canvasjs.com/	Firefox, Internet Explorer 9+, Google Chrome, Safari, Opera, iPhone, iPad	JSON	Zoom, Combination of charts, Data labels, Date-time axis, Dynamic charts, Export files, Interactive, Text Rotation for Labels	Line, Timeline, Scatter, Area, Pie, Donut, Ba	size = 27 KB, CanvasJS	Yes	Legends, Mouse Over, onClick	$999 / Team License (upto 5 developers) / CC BY-NC 3.0	Docs / Forum
Chart.js	http://www.chartjs.org/	Firefox, Internet Explorer, Google Chrome, Safari, Opera, iPhone, iPad Need to use Sencha Touch for mobile web apps.	JavaScript API	Annotations, Combination of charts, Date-time axis, Interactive	Line, Timeline, Scatter, Area, Pie, Donut, Radar, Grouped, Bar	size = 188 KB including all dependencies, HTML5 Canvas	No	Legends, Mouse Over, onClick	Free / MIT License	Personalized tech support and big community forum
Flot	http://www.flotcharts.org/	Firefox, Internet Explorer 6+, Google Chrome, Safari, Opera, iPad	JSON and XML	Zoom, Annotations, Combination of charts, Data labels, Dynamic charts, Interactive	Timeline, Scatter, Area, Pie, Donut	size = 95 KB, Canvas	No	Legends, Mouse Over, onClick	Free / MIT License	Discussions group
FusionCharts	http://www.fusioncharts.com/	Firefox, Internet Explorer, Google Chrome, Safari, Opera, iPhone, iPad	JSON and XML	Zoom, Annotations, Combination of charts, Data labels, Date-time axis, Dynamic charts, Export files, External Data Loading, Interactive, Print, Text Rotation for Labels	Line, Scatter, Area, Pie, Donut, Bullet, Radar, Funnel, Gantt, Network, Grouped, Bar	size = 553 KB, SVG / VML	Yes	Legends, Mouse Over, onClick	$999 / Team License (upto 5 developers)	Personalized tech support and community forum with 20,000 active members
Google Charts	https://developers.google.com/chart/	Firefox, Internet Explorer, Google Chrome, Safari, Opera, iPhone, iPad	JavaScript API	Annotations, Combination of charts, Date-time axis, Interactive	Line, Timeline, Scatter, Area, Pie, Donut, Grouped, Bar	size = 24 KB (only through google link, no offline mode), SVG	No	Legends, Mouse Over, onClick	Free / Google controls API (Creative Commons Attribution 3.0 License), code samples Apache 2.0 License	A lot of discussions group / Docs
highcharts	http://www.highcharts.com/	Firefox, Internet Explorer 6+, Google Chrome, Safari, Opera, iPad	JSON	Zoom, Annotations, Combination of charts, Data labels, Date-time axis, Dynamic charts, Export files, External Data Loading, Interactive, Print, Text Rotation for Labels	Line, Timeline, Scatter, Area, Pie, Donut, Radar, Funnel, Grouped, Bar	size = 45 kb, HTML5 Canvas / SVG / VML	Yes	Legends, Mouse Over, onClick	$1,755 / Highcharts 5 Developer	Personalized tech support and community forum options
NVD3	http://nvd3.org/	Firefox, Internet Explorer, Google Chrome, Safari, Opera, iPhone, iPad	JavaScript API	Zoom, Annotations, Combination of charts, Data labels, Dynamic charts, Interactive	Line, Scatter, Area, Pie, Donut, Bullet, Grouped, Bar	size = 153 KB, SVG	No	Legends, Mouse Over, onClick	Free / Apache 2.0 License	small community
ZingChart	https://www.zingchart.com/	Firefox, Internet Explorer 6+, Google Chrome, Safari, Opera, iPad	JSON	Zoom, Annotations, Combination of charts, Data labels, Date-time axis, Dynamic charts, Export files, External Data Loading, Interactive, Print, Text Rotation for Labels	Line, Scatter, Area, Pie, Donut, Bullet, Radar, Funnel, Network, Grouped	HTML5 Canvas / SVG / VML	Yes	Legends, Mouse Over, onClick	$1,999 / ZingChart Annual Software as a Service (SaaS) License	Help center (chat, stack overflow, email)