République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A/Mira de Béjaïa

Faculté des Sciences Exactes

Département d'Informatique

MÉMOIRE DE MASTER EN INFORMATIQUE

Thème

Conception et mise en œuvre d'une plateforme de gestion des machines à café

Cas d'études : Machines installées au sein d'une entreprise

Présenté par:

Adel RAHMANI

Rayan RAMDANI

Encadreur Pr Zineb TAHAKOURTH

Promotion : 2020/2021

Remerciements

Nous tenons tout d'abord à remercier Dieu, le tout puissant de nous avoir accordé la force, la volonté et le courage d'achever ce modeste travail.

Un grand merci à nos parents, qui nous ont soutenus, encouragés, offert les moyens, assisté durant tout notre cursus et leurs confiances en nous.

Nos remerciements à notre encadrant Mme Zineb TAHAKOURTH pour sa disponibilité, ses idées et ses conseils durant cette année de travail.

Nos remerciements aux membres du jury qui nous font honneur d'évaluer notre modeste travail.

Nous tenons enfin à remercier nos frères et sœurs pour leurs soutiens, nos camardes de tous les niveaux en particulier notre promotion 5e année informatique (2020/2021) et tous ceux qui ont contribué de près ou de loin dans la réalisation de ce travail.

Dédicaces

Nous dédions ce mémoire, à nos très chers parents qui ont toujours été là pour nous, ont pu créer le climat affectueux et adéquat à la poursuite de nos études et à la réalisation de ce travail.
Aucune dédicace, aucun remerciement ne pourrait présenter un grain de notre reconnaissance encore moins du sentiment d'amour envers vous, prunelles de nos yeux.

À nos frères et nos sœurs,

à nos familles,

Et à nos amis.

Table des matières

Liste des figures

Liste des tableaux

Liste des abréviations

DB = DataBase (Base de données).

SGBDR = Système de Gestion de Base de données relationnelle.

SQL = Structured Query Language.

UML = Unified Modeling Language.

UP = Unified Process.

CRUD = Create, Read, Update, Delete : Créer, lire, mettre à jour, Supprimer

UML = Unified Modeling Language : Langage de modélisation unifié.

MVC = Modèle-Vue-Contrôleur.

HTTP = Hypertext Transfer Protocol : Protocole de transfert hypertexte**.**

IoT = Internet of things.

Introduction générale

L'informatique est un domaine d'activité scientifique, technique et industriel concernant le traitement automatique de l'information par l'exécution de programmes informatiques par des machines.

De nos jours, ce domaine d'activité est devenu indispensable dans notre vie quotidienne, privée ou professionnelle, il est utilisé à tous les niveaux et dans diverses spécialités : économique, social, culturel, médical, etc. Il est aussi utilisé dans la gestion des entreprises, hôpitaux, instituts, bureaux administratifs, ainsi que des cabinets d'avocats, médicaux, etc.

Dans le monde, le café est la deuxième boisson la plus consommée après l'eau. Cette boisson omniprésente est devenue indispensable : plus de 90% de la population boit en effet du café. En entreprise, lorsque le café est offert, un collaborateur boit en moyenne 2 cafés par jour.

Les producteurs de café et les cafés à grand volume méritent des ventes et des marges élevées. Grâce à notre solution IoT brevetée, nous sommes en mesure de connecter la plupart des machines à café du marché pour extraire, en temps réel, le nombre de cafés distribués, la manière dont la machine a été utilisée, remplie avec les bons grains comme recommandé.

Nous avons tenu compte de ce dernier fait, pour choisir le thème de notre projet, qui a pour but la réalisation un ensemble de composants visant à aider à la mise en place d'une plateforme de gestion de machine à café permettant à la fois au personnel de gérer les stocks, mais aussi aux responsables d'accéder à une partie administration ainsi qu'à une partie « Dashboard », la plateforme est capable de gérer plusieurs machines à café possédant une version est un type.

Ce rapport est organisé comme suit :

- Dans le chapitre 1, nous aborderons les aspects théoriques et les spécifications du système.
- En ce qui concerne le chapitre 2, nous présenterons la conception et les scénarios envisagés, ainsi que l'environnement matériel et logiciel du développement de l'application.
- Quant à la conclusion, elle établira un bilan du travail et dressera les perspectives du projet.

Chapitre I
Cahier des charges

Chapitre 1

1 Cahier des charges

De nos jours, plusieurs institutions, sociétés, entreprises commerciales …, rencontrent des difficultés de gestion de leurs machines à café en suivant des méthodes très classiques, ce qui peut provoquer des pertes d'informations et de temps, par conséquent nous allons, à travers ce projet, mettre en place, une plateforme informatique complète, tout en veillant à ce qu'elle soit rapide et simple d'utilisation.

1.1 Présentation du projet

1.1.1 Présentation du domaine de travail

Dans notre cas on a opté pour une entreprise commerciale quelconque, ou nos machines à café sont installées dans plusieurs coins de l'entreprise.

Cette entreprise a pour but :

- Fournir des services d'achats de haute qualité à tous ses clients.
- Investissez ses employés pour offrir un meilleur service et croissance de l'entreprise.
- La sécurité est la priorité absolue pour assurer des procédures de travail sécuritaires.
- Investir dans la technologie pour fournir un service rapide, précis et rentable

1.1.2 Organisation et structure de la société

1.1.2.1 Organisation

L'organisation de l'entreprise s'articule autour du Directeur général qui assure la coordination de cinq organes qui se représentent comme suit :

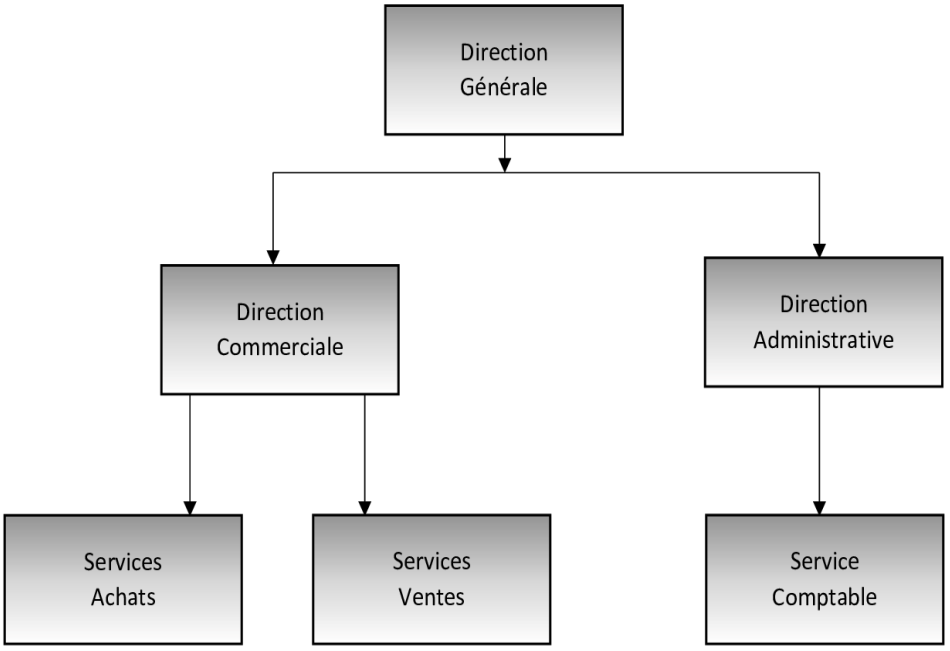


Figure 1 – Organisation de l'entreprise.

1.1.2.2 Objectifs

Comme le système actuel de gestion des machines à café installées au sein de l'entreprise se fait manuellement, c'est-à-dire que c'est le personnel lui-même qui s'occupe de tout en utilisant des fiches, des blocs-notes…, dont risque de perdre lors de la manipulation, nous avons eu l'idée de créer une plateforme, qui a pour but :

- La plateforme de gestion des machines à café va permettre à la fois au personnel de gérer les stocks, mais aussi aux responsables d'accéder à une partie administration ainsi qu'à une partie « Dashboard », la plateforme est capable de gérer plusieurs machines à café possédant une version est un type.

1.2 Besoins fonctionnels et non fonctionnels

1.2.1 Besoins fonctionnels

Besoins	Fonctionnalités
Un espace d'accueil	Principale interface qui permet l'accès aux fonctionnalités primaires de la plateforme.
Un espace de connexion	Permets l'autorisation d'accès à la plateforme.
Un espace d'affichage	Permet d'avoir une vue globale sue le contenu desfonctionnalités de l'application.

Tableau 1 – Besoins fonctionnels.

1.2.2 Besoins non fonctionnels

Ce sont des exigences qui ne concernent pas spécifiquement le comportement du système, mais plutôt identifient des contraintes internes et externes du système. Les principaux besoins non fonctionnels de notre application se résument dans les points suivants :

- **Graphisme :**
- Les principales couleurs utilisées sont le gris, blanc, et ceci parce qu'elles n'agressent pas les yeux et correspondent parfaitement au domaine médical.
- Le choix de la police s'est porté sur (Calibri), car elle est simple et facilement lisible en plus d'être prise en charge par la quasi-totalité des navigateurs.

- **Ergonomie :** La plateforme doit offrir une interface simple et facile d'utilisation.

- **Fiabilité :** La possibilité de mettre à jour des informations.

1.2.3 Gabarit d'écran type

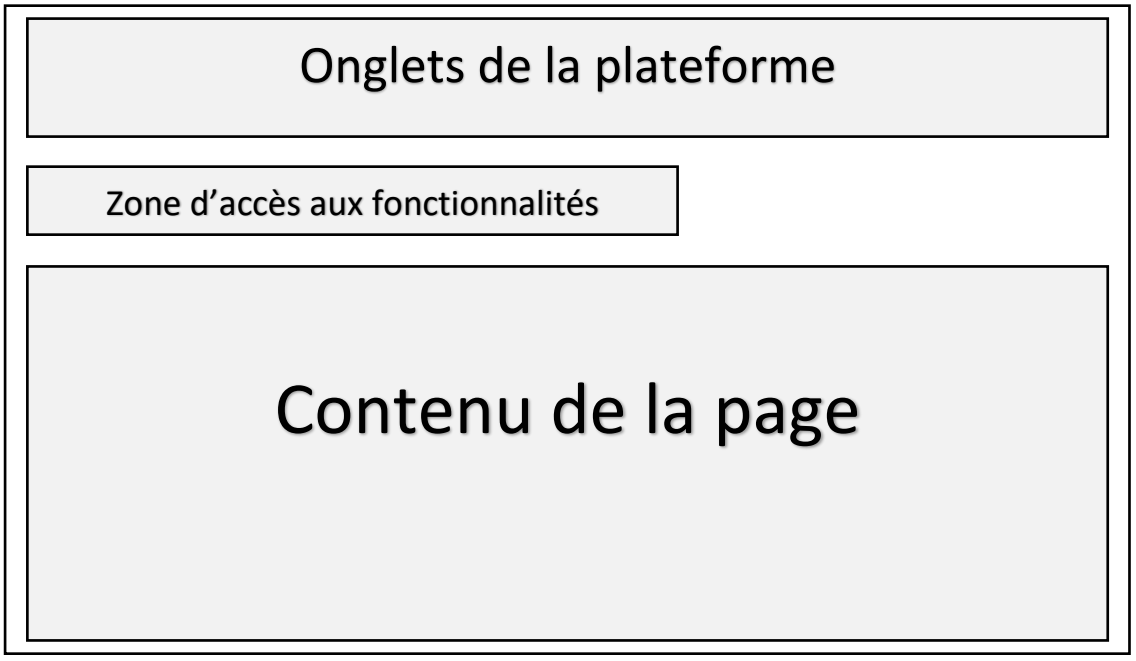


Figure 2 – Maquette de l'application.

1.2.4 Nom de l'application

En ce qui concerne le nom de l'application, nous avons choisi : **Coffee App Coffee,** comme référence à l'objectif principal de notre application, qui est la machine à café.

App, comme référence à application.

En rassemblant le tout, nous obtenons un nom explicite et en rapport avec le thème del'application, qui est de gérer les machines à café au sein de la société.

1.3 Conclusion

Après l'étude du cahier des charges présenté ci-dessus, nous avons pu définir les besoinset fonctionnalités nécessaires à la gestion des machines à café (au sein d'une société quelconque, La Belle dans notre cas). En effet, la réalisation d'une application nécessite une étude conceptuelle détaillée.

Chapitre II
Analyse et conception de la plateforme Coffee App

Chapitre 2

2 Conception

Le choix de la méthode de conduite de notre projet est indispensable, en effet nous avons besoin d'une méthodologie d'analyse et de conception qui a pour objectif de permettre de formaliser les étapes préliminaires du développement. Pour cela, nous avons choisi d'utiliser le langage de modélisation UML (Unified Modeling Language) en suivant le processus de développement UP (Unified Process), que nous présenterons à travers ce chapitre.

2.1 Identification des acteurs

Acteurs	Description
Administrateur	**Une fois connecté, l'administrateur peut accéder, via la barre de navigation, aux différentes interfaces degestion : utilisateurs, machines et produits.**
Utilisateur	**une fois connecté, l'utilisateur peut accéder, via la barre de navigation, aux interfaces permettant de gérerses machines, ses produits, et son compte.**

Tableau 2 - Description des acteurs.

2.2 Identification des cas d'utilisation

N.°	Cas D'utilisation		Acteurs
1	Authentification		Acteur
2	Gestion des utilisateurs	Rechercher un utilisateur	Administrateur
		Ajouter un utilisateur	Administrateur

		Éditer un utilisateur	**Administrateur**
		Supprimer un utilisateur	**Administrateur**
		Exporter la liste des utilisateurs en format PDF	**Administrateur**
3	**Gestion des machines**	**Rechercher une machine**	**Administrateur / Utilisateur**
		Ajouter une machine	**Administrateur**
		Éditer une machine	**Administrateur / Utilisateur**
		Affecter une machine	**Administrateur**
		Supprimer une machine	**Administrateur**
4	**Gestion des produits**	**Rechercher un produit**	**Utilisateur / Administrateur**
		Ajouter un produit	**Utilisateur**
		Éditer un produit	**Utilisateur**
		Supprimer un produit	**Utilisateur**
		Exporter la liste des produits en format PDF	**Administrateur**
5	**Gestion des comptes**	**Éditer un compte**	**Utilisateur**

Tableau 3 – Identification des cas d'utilisations.

2.3 Modèle conceptuel de données

La relation OnetoMany entre Machine et Utilisateur signifie qu'un utilisateur peut être affecté à une ou plusieurs machines. La relation OnetoMany entre Produit et Utilisateur signifie qu'un utilisateur peut déposer plusieurs produits dans une machine ou en supprimer un ou plusieurs produits d'une certaine quantité. La relation OneToMany entre Machine et Produit signifie qu'une seule machine peut contenir un seul ou plusieurs produits.

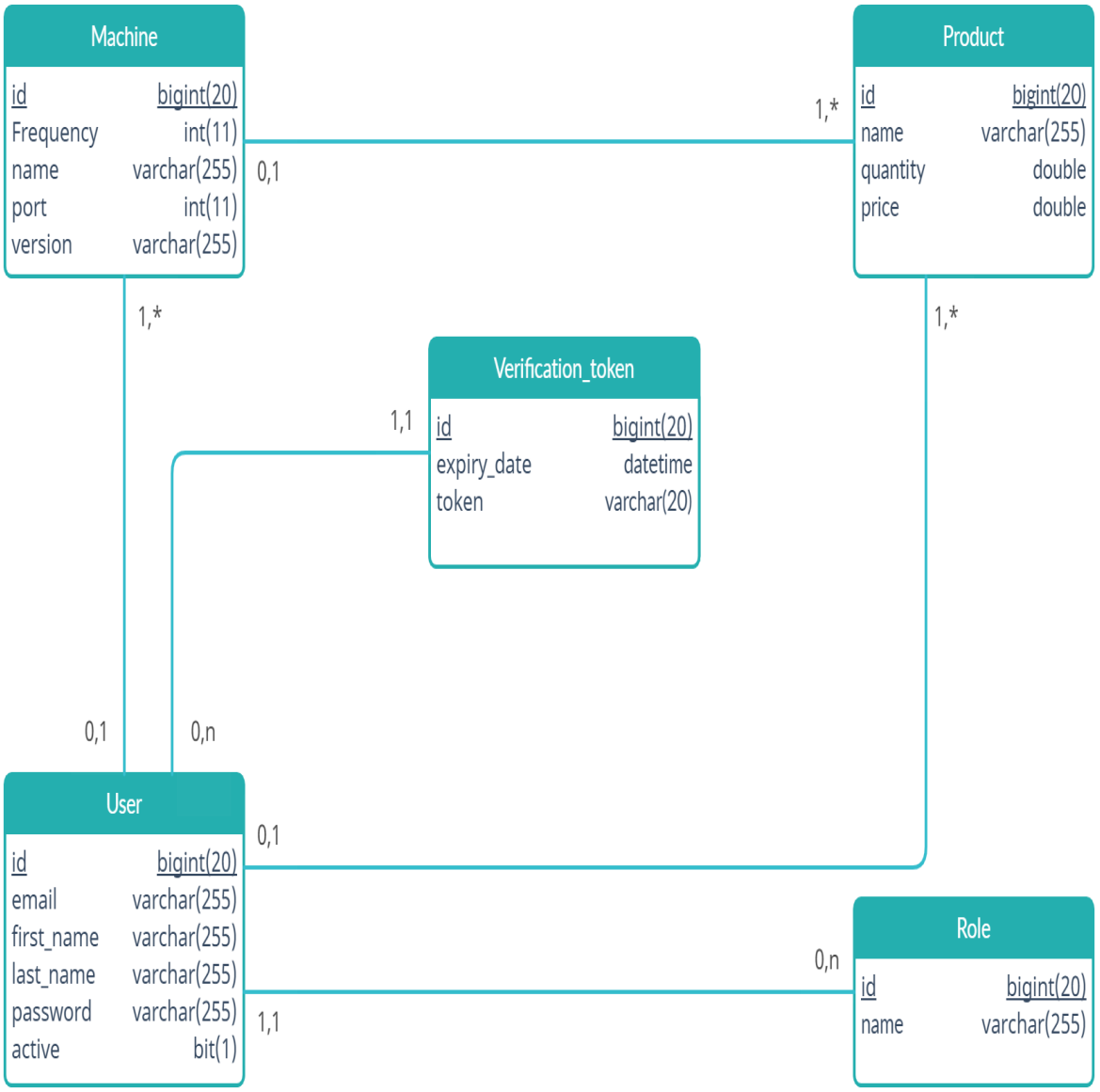


Figure 3 – Modèle conceptuel de données.

2.4 Diagramme de classe

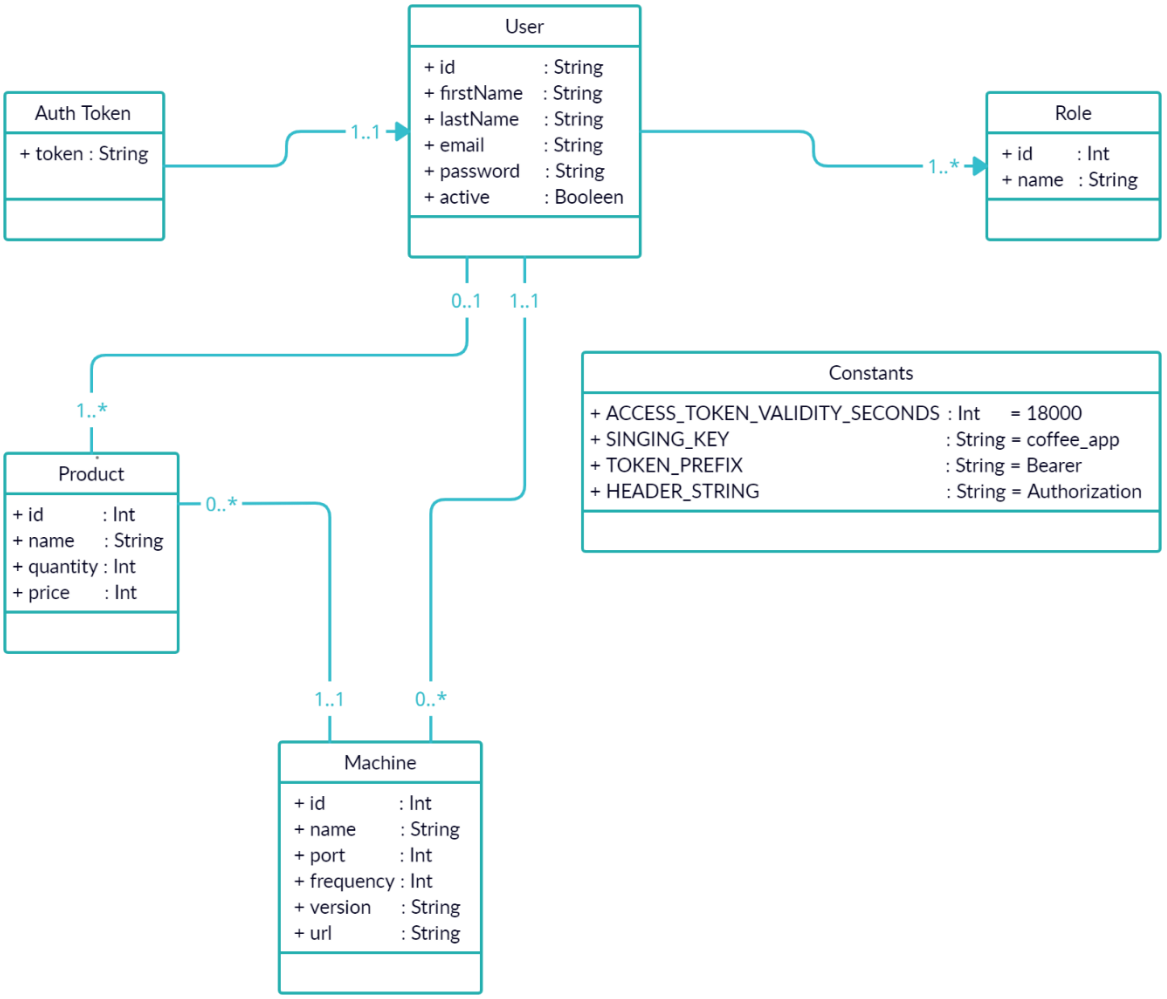


Figure 4 - Diagramme de classe.

2.5 Conclusion

Ce chapitre clôture la partie conception, il nous a permis de tirer avantage des différents diagrammes UML afin de structurer notre application. En effet, les entités représentées parles classes d'analyse entrent, en plus des contrôles et des dialogues, dans l'élaboration de diagrammes de classes participantes, qui ont servi ensuite à détailler les opérations internesdu système dans les diagrammes de séquence. Ces derniers ont mis en avant les interactionsentre les différents objets constituant notre application. Par la suite, en définissant les relations entre les entités, nous sommes parvenues à concevoir le diagramme de classes de conception donnant ainsi une vue plus structurée des éléments qui formeront la base de données liée à notre application.

Chapitre III
Implémentation du système

Chapitre 3

3 Implémentation du système

Ce chapitre représente la dernière partie de ce rapport, il traite la phase qui a pour objectif l'implémentation de notre application. Nous commençons, tout d'abord, par la description des technologies utilisées pour développer notre application. Ensuite nous mentionnons les outils technologiques utilisés. Finalement nous donnons un aperçu sur le travail réalisé en termes de code.

3.1 Réalisation technique

3.1.1 Méthode d'authentification

La méthode d'authentification utilisée est l'authentification avec token, qui est bien adaptée à Angular.

Jwt (JSON Web Token) fonctionne comme suit:

1. Le client envoie une requête pour obtenir un token (Login).
2. Le serveur vérifie si le client est autorisé et lui donne un token avec date d'expiration bien et des privilèges.
3. Le client stocke le token en session, et ajoute ce dernier à l'entête de toutes ses requêtes (l'entête est préfixé par : bearer).
4. Le serveur vérifie à chaque requête (vers les ressources privées) si le token est valide et sile client a les privilèges pour accéder à la ressource demandée et délivre une réponse.

3.1.1.1 Côté back (AuthController)

Le gestionnaire d'authentification (**AuthenticationManager**) vérifie l'email et le mot de passe. Si les credentials ne sont pas bon, il renvoit un code d'erreur 400 au client avec un message d'erreur"Email/Mot de passe invalide".
Si les credentials sont bons, il génère un token (avec **JwtTokenUtil**) et renvoie le token dans la réponse.

La classe **JwtAuthentificationFilter** se charge d'intercepter les tokens et d'extraire l'utilisateur correspondant avec son rôle. Les Endpoints privés marqués par l'annotation @PreAuthorize avec les méthodes isAuthenticated (), hasRole ("...") utiliseront ces données.

3.1.1.2 Côté front

Après authentification, si on obtient un token, le client le sauvegarde en session (cookie).
Chaque requête HTTP exécutée est interceptée par la classe Interceptor. (app/core/interceptor) qui s'ajoute un entête Authorization contenant le token préfixé par "Bearer ".

3.1.2 Back end

3.1.2.1 Qu'est-ce qu'un Endpoint ?

Un Endpoint est ce qu'on appelle une extrémité d'un canal de communication.
Autrement dit, lorsqu'une API interagit avec un autre système, les points de contact de cette communication sont considérés comme des Endpoints.

Ainsi, pour les API, un Endpoint peut inclure une URL d'un serveur ou d'un service. Chaque Endpoint est l'emplacement à partir duquel les API peuvent accéder aux ressources dont elles ont besoin pour exécuter leur fonction.

En effet, les API fonctionnent à l'aide de « demandes » et de « réponses ».

C'est donc lorsqu'une API demande des informations à une application Web ou à un serveur Web, qu'elle recevra une réponse.

Enfin, un Endpoint représente l'endroit où les API envoient les demandes et où réside la ressource.

3.1.2.2 Les Endpoints (Plateforme)

URL	Méthode	Niveau d'accès	Body	Description
/api/auth/login	**POST**		**{ "email": "string", "password": "string" }**	**Authentifie l'utilisateur,renvoie un Token**
/api/auth/ logout	**GET**	**User, Admin**		**Renvoie l'username del'utilisateur courant**

/api/auth/ **logout**	**GET**	**User, Admin**		**Déconnecte l'utilisateur**
/api/users	**GET**	**Admin**		**Retourne tous les utilisateurs**
	POST	**Admin**	**{** **"email": "string",** **"firstName": "string",** **"lastName": "string" ,** **"password" : "string"** **}**	**Crée un utilisateur**
/api/users/{id}	**PUT**	**User, Admin**	**{** **"email": "string",** **"firstName": "string",** **"lastName": "string"** **}**	**modifie l'utilisateur avec l'id donné**
	GET	**User, Admin**		**Retourne l'utilisateuravec l'id donné**
/api/users/{id}	**DELETE**	**Admin**		**Supprime l'utilisateuravec l'id donné**
/api/users/ password forgotten	**GET**	**User, Admin**	**"email"**	**Envoie un mail de réinitialisation de mdp**
/api/users/ change_ password	**PUT**	**User,** **Admin**	**{** **"email": "string",** **"password": "string"**	**Permets de changer lemot de passe de l'utilisateur courant.**

			}	
/api/users/ change_ password /forgotten/ {token}	PUT	User	{ "email": "string", "password": "string" }	Changer le mot de passe d'un utilisateurqui a oublié son mot de passe
/api /machines	GET	User, Admin		Retourne toutes les machines
	POST	Admin	{ "frequency": 0, "name":"string ", "port": 0000, "userId": 0, "version": "string" }	Crée une machine

/api/machines /config/{id}	PUT	User, Admin	{ "frequency": 0, "name":"string", "port": 0000, "userId": 0, "version": "string" }	Configure une machine
/api/machines {id}	GET	User, Admin		Retourne une machine
	DELETE	Admin		Supprime une machine
/api/machines /assign/{id}	PUT	Admin	"userId"	Assigne une machineà un utilisateur

/api/products	GET	User, Admin		Retourne tous les produits
	POST	User	{ "machineId": 0, "name":"string", "price": 0, "quantity": 0 }	Crée un produit dansun machine
/api/products/ update/{id}	PUT	User	{ "machineId": 0, "name":"string", "price": 0, "quantity": 0 }	Mets à jour un produit
/api/products /{id}	GET	User		Retourne un produit
	DELETE	User	xss	Supprime un produit

Tableau 4 - Les Endpoints (Plateforme).

3.1.2.3 Les Endpoints (Machine à café)

URL	Méthode	Body	Description
/api/version	GET		Récupérer la versionde l'IoT (machine à café)
/api/config	POST	{ "serverUrl": "string", "serverPort": "long", "agentPort" :"long", "frequency": "int","version": "string" }	Modifier la configuration de lamachine à café
/api/products /{id}	GET		Récupérer la liste desproduits

	POST	**{ "id": "long", "name": "string", "quantity" :"long", "price": "int" }**	**Insérer un nouveau produit**
	PUT	**{ "id": "long", "name": "string", "quantity" :"long", "price": "int" }**	**Modifier un produit**
	DELETE		**Supprimer un produit**
/api/products / consume	**POST**	**{ "id": "long", "quantity": "int" }**	**Consommer x unité du produit id**
/api/products /synchronize	**POST**	**{ "machineId": 0,"products": [] }**	**Synchronise les données sur les produits de la machine avec ceux du serveur**

Tableau 5 - Les Endpoints (Machine à café).

3.2 Environnement de développement

3.2.1 Le Back office et le Front office

3.2.1.1 Back office

En informatique, le **back-office** est une partie d'un site Internet ou d'un système informatique.

Elle concerne la partie qui permet à l'entreprise d'administrer et de gérer son site. On pourra retrouver comme service par exemple :

- Ajout des produits et des services ;
- Modification des services ;
- Modifications de paramètres ;
- Administration d'un forum ;
- Gestion des utilisateurs ;
- ...

Cette partie est opposée au front office qui est la partie dédiée à l'interaction avec l'utilisateur. [1]

3.2.1.2 Front office

En informatique, le terme **front office** est un terme d'architecture logicielle. Il désigne la partie qui prend en charge l'interface d'une application, par opposition au back-office qui lui regroupe la partie gestion qui, par rapport à une architecture (architecture trois tiers regroupent la partie métier et données).

On retrouve typiquement le **front office** dans les sites web commerciaux qui permettent de commander en ligne ou celui des banques permettant de gérer des comptes en ligne. Dans ce cas, le **front office** correspond à la partie hypertexte (web) de l'application, donc ce qui est visible par le *consommateur*. Elle lui permet d'interagir avec le reste du site, par exemple, pour effectuer des achats, un virement bancaire, etc.

Par analogie, on peut comparer le **front office** à la partie d'un magasin qui est fréquenté par les clients, tandis que le **back-office** correspond davantage à l'administration, aux entrepôts, aux échanges avec les partenaires, etc. [1]

3.3 API Rest

Une API est un ensemble de définitions et de protocoles qui facilite la création et l'intégration de logiciels d'applications. API est un acronyme anglais qui signifie « Application Programming Interface », que l'on traduit par interface de programmation d'application.

3.3.1 À quoi servent les API ?

Les API permettent à votre produit ou service de communiquer avec d'autres produits et services sans les détails de leur mise en œuvre. Elles simplifient le développement d'applications et vous font ainsi gagner du temps et de l'argent. Lorsque vous concevez de nouveaux outils et produits, ou que vous assurez la gestion de ceux qui existent déjà, les API vous offrent plus de flexibilité, simplifient la conception, l'administration et l'utilisation, et vous donnent les moyens d'innover.

Les API sont parfois considérées comme des contrats, avec une documentation qui constitue un accord entre les parties : si la partie 1 envoie une requête à distance selon une structure particulière, le logiciel de la partie 2 devra répondre selon les conditions définies.

3.3.2 Pourquoi les API sont-elles importantes ?

Parce que les API simplifient la façon dont les développeurs intègrent de nouveaux composants d'applications dans une architecture existante, elles facilitent la collaboration entre les équipes informatiques et métiers. Souvent, les besoins des entreprises changent rapidement face à l'évolution constante des marchés numériques, où de nouveaux concurrents peuvent bouleverser tout un secteur avec une nouvelle application. Afin de conserver leur compétitivité, il est important pour ces entreprises de soutenir le développement et le déploiement rapides de services novateurs. Le développement d'applications cloud-native est un moyen évident d'augmenter la vitesse de développement. Il repose sur la connexion d'une architecture d'applications de type micro services via des API.

Les API constituent un moyen simplifié de connecter votre propre infrastructure au travers du développement d'applications cloud-native. Elles vous permettent également de partager vos données avec vos clients et d'autres utilisateurs externes. Les API publiques offrent une valeur métier unique parce qu'elles peuvent simplifier et développer vos relations avec vos partenaires, et éventuellement monétiser vos données (l'API Google Maps en est un parfait exemple).

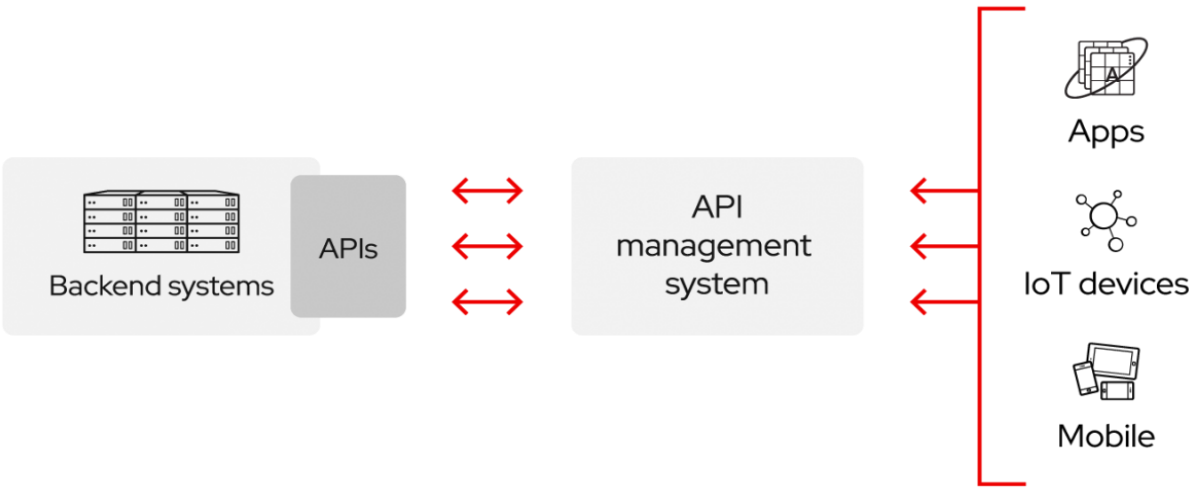


Figure 5 - API

3.3.3 Les API sont-elles sûres ?

Les API vous permettent d'ouvrir l'accès à vos ressources sans sacrifier le contrôle et la sécurité. C'est vous qui choisissez les ressources que vous souhaitez partager, et avec qui. La sécurité des API dépend avant tout de leur bonne gestion. La connexion des API et la création des applications qui utilisent les données ou fonctionnalités exposées par les API peuvent se faire par l'intermédiaire d'une plateforme d'intégration distribuée qui connecte tout, y compris les systèmes existants et l'Internet des objets.

Il existe trois approches d'accès aux API.

3.3.3.1 API privées

L'API n'est utilisable qu'en interne. Cette approche permet de garder un contrôle total sur l'API.

3.3.3.2 API partenaires

L'API est partagée avec certains partenaires de l'entreprise. Cette approche peut générer de nouveaux flux de revenus sans compromettre la sécurité.

3.3.3.3 API publiques

L'API est accessible à tous. Cette approche autorise les tiers à développer des applications qui interagissent avec votre API et peut devenir source d'innovations.

3.3.4 Comment les API favorisent-elles l'innovation ?

En rendant vos API accessibles à vos partenaires ainsi qu'aux tiers, vous pouvez :

- Générer de nouveaux canaux de revenus ou étendre ceux qui existent déjà ;
- Étendre la portée de votre marque ;
- Stimuler l'innovation Open-Source ou améliorer l'efficacité grâce au développement et à la collaboration externe.

Intéressant, non ? Mais quel est le rôle des API dans tout ça ?

Reprenons notre exemple du distributeur de livres.

Imaginez qu'un partenaire de l'entreprise développe une application qui permet aux clients de localiser les livres qu'ils cherchent dans les rayons. En améliorant l'expérience client, cette application va attirer d'autres clients dans la librairie, elle-même cliente du distributeur, qui en fin de compte aura étendu son canal de revenus.

Une entreprise tierce peut aussi utiliser une API publique pour développer une application afin que les clients puissent acheter des livres directement auprès du distributeur, sans passer par la librairie. Dans ce cas, le distributeur aura ouvert un nouveau canal de revenus.

Une entreprise peut tirer parti du partage de ses API avec certains de ses partenaires ou avec le monde entier. Chaque partenariat vous permet d'étendre la notoriété de

votre marque en parallèle de vos campagnes marketing. En rendant vos technologies publiques, avec une API publique par exemple, vous encouragez les développeurs à créer un écosystème d'applications autour de votre API. Plus les gens utilisent vos technologies, plus ils sont susceptibles de faire affaire avec vous.

Vous pouvez ainsi obtenir des résultats aussi inattendus qu'intéressants en ouvrant l'accès à vos technologies et ces résultats peuvent parfois bouleverser un secteur tout entier. Dans le cas de notre distributeur de livres, de nouvelles entreprises, comme un service de location de livres, peuvent provoquer un changement fondamental dans son fonctionnement. Les API publiques et partenaires vous permettent de profiter des innovations d'une communauté de développeurs plus large. Les idées novatrices peuvent germer n'importe où. Les entreprises doivent se tenir au courant des changements qui s'opèrent sur leur marché et se préparer à y faire face. C'est là que les API interviennent.

3.3.5 Les API, une histoire toute récente

Les API sont apparues à l'aube de l'informatique, avant même les ordinateurs personnels. À cette époque, elles étaient surtout utilisées en tant que bibliothèques pour les systèmes d'exploitation. Elles résidaient presque toutes en local sur les systèmes sur lesquels elles s'exécutaient, même si elles transféraient parfois des messages entre les mainframes. Presque 30 ans après, les API sont sortis de leurs environnements locaux. Au début des années 2000, elles sont devenues importantes pour l'intégration des données à distance.

3.3.6 API distantes

Les API distantes sont conçues pour interagir via un réseau de communication. Ici, « distant » signifie que les ressources manipulées par l'API ne se trouvent pas sur l'ordinateur qui formule la requête. Le réseau de communication le plus fréquemment utilisé étant Internet, la plupart des API sont conçues sur la base des normes Web. Toutes les API distantes ne sont pas des API Web, mais on peut supposer que toutes les API Web sont distantes.

Les API Web utilisent en général le protocole HTTP pour leurs messages de requête et fournissent une définition de la structure des messages de réponse. Les messages de réponse se présentent la plupart du temps sous la forme d'un fichier XML ou JSON. Ces deux formats sont les plus courants, car les données qu'ils contiennent sont faciles à manipuler pour les autres applications.

3.3.7 Améliorations apportées aux API

Les API n'ont cessé de se développer sur le Web aujourd'hui omniprésent, et plusieurs initiatives ont tenté de simplifier leur conception et d'augmenter leur utilité.

3.3.8 Un peu de SOAP et beaucoup de REST

Pour standardiser l'échange des informations entre les API toujours plus nombreuses, il a fallu développer un protocole : le « Simple Object Access Protocol », ou SOAP. Les

API conçues d'après le protocole SOAP utilisent le format XML pour leurs messages et reçoivent des requêtes via HTTP ou SMTP. SOAP a pour objectif de simplifier l'échange des informations entre les applications qui s'exécutent dans des environnements différents ou qui ont été écrites dans des langages différents.

Le « Representational State Transfer », ou REST, sont une autre tentative de normalisation. Les API Web qui respectent les contraintes de l'architecture REST sont appelées API RESTful. Ces deux éléments diffèrent sur un point fondamental : SOAP est un protocole, alors que REST est un style d'architecture. Cela signifie qu'il n'existe aucune norme officielle qui régit les API Web RESTfulCes contrainte peuvent sembler difficiles à appliquer, mais dans les faits, elles le sont moins qu'un protocole. C'est pour cette raison que les API RESTful sont en train de prendre le pas sur les API SOAP.

- Ces dernières années, la spécification open API s'est imposée comme la norme commune pour définir les API REST. La norme open API permet aux développeurs de créer des interfaces d'API REST de manière à ce que les utilisateurs puissent les comprendre avec un minimum d'approximation.

3.3.9 Architectures SOA et de micro services

Les deux approches architecturales qui utilisent le plus les API distantes sont l'architecture orientée service (SOA) et l'architecture de microservices. La SOA est l'approche la plus ancienne. Elle visait à l'origine à améliorer les applications monolithiques. Par définition, une application monolithique fait tout. Mais certaines fonctions peuvent être fournies par d'autres applications faiblement couplées via un modèle d'intégration, comme une ESB (Entreprise Service bus).

Si l'architecture SOA est plus simple qu'une architecture monolithique sous bien des aspects, elle peut potentiellement provoquer des changements en cascade au sein de l'environnement si les interactions entre les différents composants ne sont pas parfaitement comprises. En complexifiant l'environnement, l'architecture SOA réintroduit une partie des problèmes qu'elle cherchait à résoudre.

Les architectures de microservices fonctionnent d'une manière très similaire, dans le sens où elles utilisent des services faiblement couplés. Par contre, elles poussent la déstructuration de l'architecture classique encore plus loin. Les services qui composent l'architecture de microservices utilisent une structure de messagerie commune, telle que les API RESTful. Ils se servent des API RESTful pour communiquer entre eux simplement, sans convertir leurs données ni recourir à des couches d'intégration supplémentaires. L'utilisation des API RESTful permet et favorise même l'accélération de la distribution des nouvelles fonctions et mises à jour. Chaque service est distinct. Vous pouvez remplacer, améliorer ou supprimer chacun d'entre eux sans affecter les autres services de l'architecture. Cette architecture légère vous aide à optimiser les ressources distribuées ou cloud et à faire évoluer chaque service de façon dynamique. [2]

3.4 Bases de données Mysql

MySQL est un serveur de bases de données relationnelles SQL développé dans un souci de performances élevées en lecture, ce qui signifie qu'il est davantage orienté vers le service de données déjà en place que vers celui de mises à jour fréquentes et fortement sécurisé. Il est multithread et multi-utilisateur.

C'est un logiciel libre7, open source8, développé sous double licence selon qu'il est distribué avec un produit libre ou avec un produit propriétaire. Dans ce dernier cas, la licence est payante, sinon c'est la licence publique générale GNU (GPL) qui s'applique. Un logiciel qui intègre du code MySQL ou intègre MySQL lors de son installation devra donc être libre ou acquérir une licence payante. Cependant, si la base de données est séparée du logiciel propriétaire qui ne fait qu'utiliser des API tierces (par exemple en C# ou PHP), alors il n'y a pas besoin d'acquérir une licence payante MySQL. Ce type de licence double est utilisé par d'autres produits comme le Framework de développement de logiciels Qt (pour les versions antérieures à la 4.5). [3]

3.5 Interface utilisateur UI

UI est un sigle à la mode, entre autres parce que le métier d'**UI designer** est vogue et que c'est un peu le nouveau nom "trendy", un "buzzword" pour dire "graphiste" (à tort et à raison).

Mais pourquoi est-ce qu'on en entend autant parler ?

Et bien parce que les évolutions technologiques ont amené à repenser les usages et donc les formes d'interfaces en général ; cela est notamment lié au **mobile**, mais l'**interface utilisateur** ne date pas du mobile.

Une **interface utilisateur**, c'est tout simplement le point de rencontre entre un utilisateur et un objet (ou une machine) ; c'est ce qui permet d'interagir avec le produit. Pour retenir, découpez le mot "Inter-face" : c'est la "face" (la sur**face**) qui fait l'**inter**médiaire entre vous et un produit.

Prenons l'exemple de la télévision : la télécommande représente l'interface. Sans elle, vous ne pouvez pas interagir avec la télévision, la regarder, changer de chaine, bref : l'utiliser. Autre exemple, la voiture : le tableau de bord + le volant + les pédales + le levier de vitesse forme ensemble l'interface de la voiture.

3.5.1 Sachez à quoi sert une interface

Wikipédia vous dira que l'interface permet d'effectuer une tâche, une action, une commande (en bref d'établir un contrôle) via un aller-retour entre l'utilisateur et la machine :

Une interaction est initiée/stimulée/déclenchée par l'humain (de quelque façon que ce soit : via le toucher dans la plupart du temps, mais vous verrez, pas toujours) === input

À laquelle la machine répond ("feeds back" en anglais = donne quelque chose en retour / rend) de façon simultanée en donnant de l'information, ou un résultat === output

Visualisez cet aller-retour comme un lancer de boomerang…

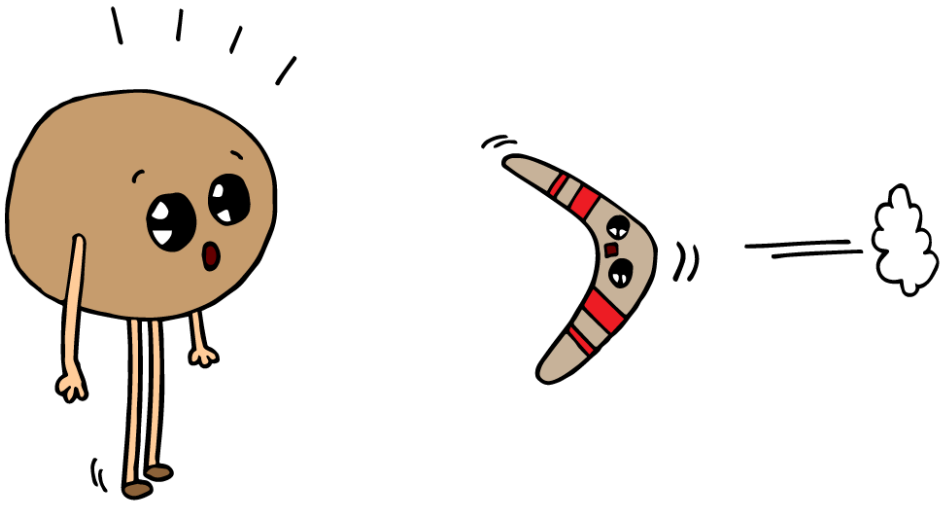


Figure 6 - Rôle d'interface.

Si l'on reprend l'exemple de la télécommande :

L'interface (qu'est la télécommande) entre vous et la télévision se comporte de telle façon que si vous appuyez sur un bouton, il aura l'effet escompté.

Plus précisément, si vous appuyez sur le bouton marqué "2", vous faites le premier pas, vous lancez le boomerang. Vous donnez un "input" (en anglais).

Cela aura comme résultat la chaîne de télévision numéro 2 qui va s'afficher sur l'écran de télévision ; c'est le feedback, le retour du boomerang, la réponse de la machine à votre impulsion. Vous récupérez un "output" (en anglais).

Et tout cela forme l'interaction ! [4]

3.6 Dashboard

En tant que Data Analyst, votre rôle est de présenter les données qui intéressent vos collaborateurs.

Et comme il est bien plus simple d'interpréter des données de manière visuelle qu'en observant une multitude de nombres sur une feuille de calcul, il vous faut trouver un moyen de rendre ces données *visualisables* !

C'est le principal intérêt d'un **tableau de bord**, ou **dashboard** en anglais. Un dashboard permet de garder un œil sur les métriques importantes pour piloter un projet, ou même une entreprise. Il est une sorte de boussole permettant d'atteindre des objectifs. [5]

3.7 Services externes (IoT / machines à café)

3.7.1 IoT | L'Internet des Objets

L'Internet des Objets (IoT – Internet of Things) compte probablement parmi les phénomènes dont on parle le plus de nos jours. Mais qu'est-ce que l'IoT ? Gartner, leader mondial des sociétés de conseil et de recherche dans le domaine des technologies, le définit ainsi :

« L'Internet des Objets est un réseau d'objets physiques dédiés qui intègrent des technologies pour détecter où interagir avec leurs états internes ou leurs environnements externes. L'IoT constitue un écosystème d'objets, de communications, d'applications et d'analyses des données ».

Avec l'introduction de compteurs, de capteurs, de systèmes de gestion des bâtiments et autres dispositifs qui mesurent les comportements réels, communiquent et interagissent avec d'autres systèmes, l'IOT offre une large palette d'applications dans un bâtiment et pour l'exploitation des services. Certains analystes estiment que 50 % des fonctions de contrôle par un être humain disparaîtront au cours des cinq prochaines années. La gestion de l'immobilier et le Facility Management seront donc confrontés à des changements drastiques dans leurs modèles et leurs processus d'exploitation. [6]

4 Les technologies utilisées

4.1 Spring Boot

Si vous êtes amenés à travailler sur une application développée autour de l'architecture Microservices en Java, vous aurez tôt ou tard affaire à Spring Boot. Dans ce chapitre, nous allons donc prendre le temps de comprendre ce que fait ce framework et surtout lever l'ambiguïté sur la différence avec les autres frameworks Spring, qui à première vue font la même chose.

Spring Boot est un framework qui facilite le développement d'applications fondées sur Spring en offrant des outils permettant d'obtenir une application packagée en jar, totalement autonome. Ce qui nous intéresse particulièrement, puisque nous essayons de développer des Microservices !

Si vous avez du mal à comprendre l'ensemble des notions à venir, ne vous inquiétez pas ! Nous reprendrons tous ces concepts dans le chapitre prochain avec un exemple concret.

4.1.1 Ce que propose Spring Boot

Spring et Spring MVC sont de formidables outils quand on essaye de développer une application web. Néanmoins, un de leurs plus gros problèmes est la configuration. Si vous avez déjà développé une application avec ces outils, vous avez dû remarquer que votre application est bardée de fichiers XML qui indiquent les configurations des servlets, des vues, des contenus statiques, etc. Ces fichiers de configuration deviennent un vrai challenge lorsque vous avez une application complexe.

Comment Spring Boot permet-il de créer une application sans écrire une seule ligne de configuration, ou presque ?

Pour simplifier cette configuration, Spring Boot propose 2 fonctionnalités principales :

- L'auto-configuration.
- Les starters.

4.1.2 L'auto-configuration

Cette fonctionnalité est la plus importante de Spring Boot. Elle permet de configurer automatiquement votre application à partir des jars trouvés dans votre Classpath. En d'autres termes, si vous avez importé des dépendances, Spring Boot ira consulter cette liste puis produira la configuration nécessaire pour que tout fonctionne correctement.

Prenons l'exemple d'une application web dans laquelle vous avez les dépendances : Hibernate et Spring MVC. Normalement, vous devez créer les fichiers de configuration suivants :

- appconfig-mvc.xml
- web.xml

persitence.xml.

Comme vous ne connaissez pas nécessairement la syntaxe de ces fichiers par coeur, il vous faut consulter la documentation ou vous inspirer d'un ancien projet. Vous devez ensuite écrire le code Java qui permet de lier ces fichiers XML à ApplicationContext de Spring.

Si vous n'êtes pas familier avec ApplicationContext de Spring, je vous recommande de lire ces explications.

Ensuite, vous adaptez et testez, puis réadaptez et rétestez encore pendant un bon moment avant que tout fonctionne... Bien sûr, dès que vous faites un changement dans votre application, il ne faut pas oublier de mettre à jour tous les fichiers de configuration, puis débugger de nouveau. Cela devient très vite très fastidieux !

Voici l'équivalent de l'ensemble de ces étapes avec Spring MVC : **@EnableAutoConfiguration**

C'est tout ! Avec cette annotation, Spring Boot ira scanner la liste de vos dépendances, trouvant par exemple Hibernate. Ayant constaté que vous n'avez défini aucun autre data source, il créera la configuration nécessaire et l'ajoutera à ApplicationContext.

Bien entendu, vous pouvez très facilement personnaliser ces configurations, en créant vos Beans ou vos propres fichiers de configuration. Spring Boot utilisera alors en priorité vos paramètres.

4.1.3 Les Starters

Les starters viennent compléter l'auto-configuration et font gagner énormément de temps, notamment lorsqu'on commence le développement d'un Microservice. Un starter va apporter à votre projet un ensemble de dépendances, communément utilisées pour un type de projet donné. Ceci va vous permettre de créer un "squelette" prêt à l'emploi très rapidement.

L'autre énorme avantage est la gestion des versions. Plus besoin de chercher quelles versions sont compatibles puis de les ajouter une à une dans le pom.xml ! Il vous suffit d'ajouter une simple dépendance au starter de votre choix. Cette dépendance va alors ajouter, à son tour, les éléments dont elle dépend, avec les bonnes versions.

Prenons l'exemple où vous souhaitez créer un Microservice. En temps normal, vous aurez besoin des dépendances suivantes :

- Spring ;
- Spring MVC ;
- Jackson (pour json) ;
- Tomcat ; [7]

4.2 C'est quoi Spring security ?

Spring Security est l'un des modules de sécurité de Spring Framework. Il s'agit d'une infrastructure de sécurité Java SE / Java EE qui fournit une authentification, une autorisation, une authentification unique et d'autres fonctionnalités de sécurité pour les applications Web ou les applications d'entreprise. On peut aussi s'authentifier avec Spring Security en utilisant **un Token** ou **le Framework Ionic.** [8]

4.3 Framework Angular

Angular est un **framework très populaire** et puissant dans le monde du JavaScript. Développé par les équipes de Google, il apporte **efficacité, rapidité et organisation** pour un meilleur partage de votre travail.

4.3.1 Comparaison des technologies InterfaceUtilisateur

4.3.1.1 Les critères de choix

Pour sélectionner le framework sur lequel baser le développement des applications, faut considérer les facteurs suivants :

- Sa courbe d'apprentissage,
- Son efficacité, sa capacité à développer rapidement et simplement,
- La facilité avec laquelle on peut maintenir ses développements,
- Son extensibilité et la richesse de son écosystème,

- Son aptitude à s'intégrer à d'autres technologies front pour permettre une transition en douceur
- Ses capacités d'intégration dans un environnement technique existant.
- Sa popularité et la vivacité de sa communauté,
- La qualité de sa documentation,
- Sa performance.

On peut distinguer deux grandes familles de frameworks :
Vue et **React** sont souvent considérés comme des librairies proposant un noyau dur de fonctionnalités relatives à la couche de présentation, noyau auquel on peut ajouter au besoin des modules complémentaires.
Angular et **Angular.js** sont quant à eux plutôt considérés comme des frameworks tout-en-un, qui mettent donc potentiellement à disposition bien plus que ce dont nous pouvons avoir besoin. En contrepartie ils assurent la compatibilité totale entre toutes les composantes.

4.3.2 Analyse des frameworks

4.3.2.1 La courbe d'apprentissage

La courbe d'apprentissage de Vue.js est clairement plus facile à suivre que celle des autres frameworks proposés. Vue a tendance à utiliser simplement JavaScript en ajoutant quelques idées, tandis que React, Angular ou Angular.js imposent d'adapter des pratiques éculées deprogrammation JavaScript à une nouvelle façon de faire, particulière au framework.

4.3.2.2 L'efficacité

Vue.js est le moins verbeux ce qui le place donc en tête pour l'efficacité et la capacité à développer rapidement et simplement.

4.3.2.3 La maintenance

Concernant la facilité avec laquelle on peut maintenir ses développements :

- Vue.js, à travers l'utilisation de Webpack, incite à structurer un projet et permet de gérer un unique fichier par composant regroupant sa représentation graphique, sa logique et son style. Mais aucun élément constituant l'application n'est soumis à un cadre obligé. Il en va de même pour Angular (malgré la notion de modules par exemple) et React qui n'imposent pas de règle particulière. La simplicité de syntaxe de Vue lui donne néanmoins une longueur d'avance sur sa capacité à être facilement maintenable.

- Bien que le processus d'Angular ait tardé à se stabiliser sur ce sujet, la gestion des versions de tous ces frameworks est aujourd'hui basée sur la spécification Semantic Versioning. Ce qui donne une visibilité plus claire sur les impacts d'une nouvelle version d'un framework. Des maintenances correctives et de

sécurité d'une version N-1 sont également assurées sur plusieurs mois après la sortie officielle de la version N.

- L'avantage des solutions "tout-en-un" (angular, angular.js) est que l'intégration de toutes les composantes se fera de manière naturelle et homogène, sans avoir besoin de gérerla compatibilité de leurs différentes versions, ce qui n'est pas forcément le cas des "librairies" (vue et react) pour lesquelles les composantes additionnelles peuvent ne fonctionner que pour des versions précises de la librairie principale.
- Quelle que soit la solution utilisée, même si elle impose un cadre important, une revue de code régulière s'avère un passage obligé pour garantir la qualité du produit fini.

4.3.2.4 L'écosystème

À propos de l'extensibilité et de la richesse de l'écosystème

- Toutes ces solutions proposent des plugins pour les navigateurs web afin de faciliter les développements.

  Les librairies techniques sont trop chargées pour React comme pour Vue, ce qui peut sembler rebutant au premier abord ; il faut cependant relativiser et bien comprendre qu'une fois ajoutées les deux ou trois librairies incontournables (bien documentées, bien maintenues et facilement intégrables), on pourra couvrir généralement 90% de nos besoins applicatifs. Les 10% restants étant couverts par des librairies additionnelles que pour l'essentiel on ajoutera également à nos frameworks "tout-en-un".
- Les composants unitaires "tout prêts" se multiplient, mais il faut préférer les librairies de composants déjà fonctionnels et uniformes. Angular (avec Bootstrap notamment) dispose de librairies abouties et matures. React.js semble un peu plus limité à ce niveau. Les librairies pour Vue.js ont souffert un temps d'un certain manque de stabilité, mais certaines sont très prometteuses et déjà riches et bien documentées, comme Vuetify.

4.3.2.5 Les capacités d'intégration

Sur la possibilité de s'intégrer à d'autres technologies front pour permettre une transition en douceur :

- Angular est clairement réservé à la construction de (grosses) applications SPA et même s'il est théoriquement possible de l'intégrer à des applications web utilisant un autre framework, c'est à proscrire; il en est de même pour Angular.js (qui n'est presque plus utilisé comme on le verra dans la partie popularité).
- À l'inverse Vue.js et React.js offrent cette possibilité de très facilement s'intégrer à n'importe quelle application web puisqu'il suffit d'importer la

librairie (très légère) correspondante.

- Concernant la capacité des frameworks à supporter les web components, et plus particulièrement la spécification **Custom Elements** considérée comme l'une des plus importantes puisqu'elle permet de définir de nouveaux types d'éléments HTML, il s'avère que tous ne proposent pas aujourd'hui le même niveau de compatibilité : Angular, Vue.jset Angular.js respectent totalement la norme définie, mais ce n'est pas le cas de React.js.

4.3.2.6 La popularité

En production :

- React.js est utilisé par Facebook, AirBnB, Netflix, Paypal ou encore Twitter.
- Vue.js est utilisé par Alibaba, Nintendo, Gitlab, Expedia.
- Angular/AngularJs par Google, Ryanair, Bose ou Rockstar Games.

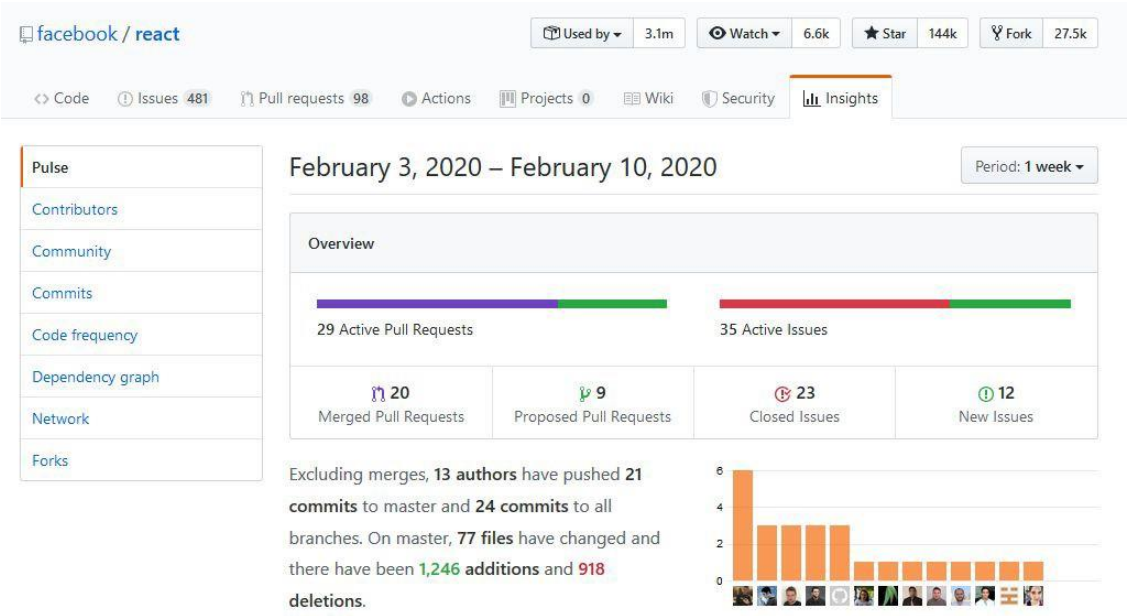
On constate de plus en plus de remplacements d'AngularJS/Angular par Vue.js ou par React.js.

Sur GitHub, Vue.js (v1 sortit en octobre 2015, v2 en septembre 2016) cumule actuellement plus de 157k d'étoiles, soit 2,6 fois plus que Angular.

En jetant un coup d'oeil sur github, on peut constater:

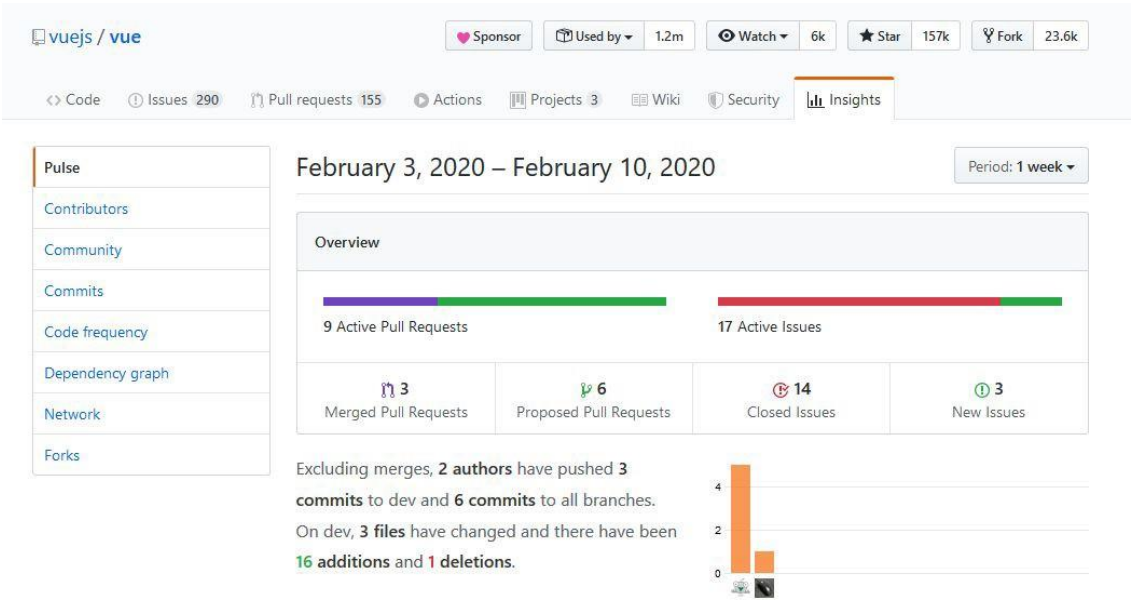
Pour React.js, 46 commiteurs avec pour les 10 plus actifs :

- Un total de 191 commits répartis de 5 à 62 commits par personne.
- 4 sont employés Facebook et totalisent 133 commits.
- 2 principaux intervenants (employés Facebook) à 62 et 45 commits, le suivant à 17 (non Facebook).



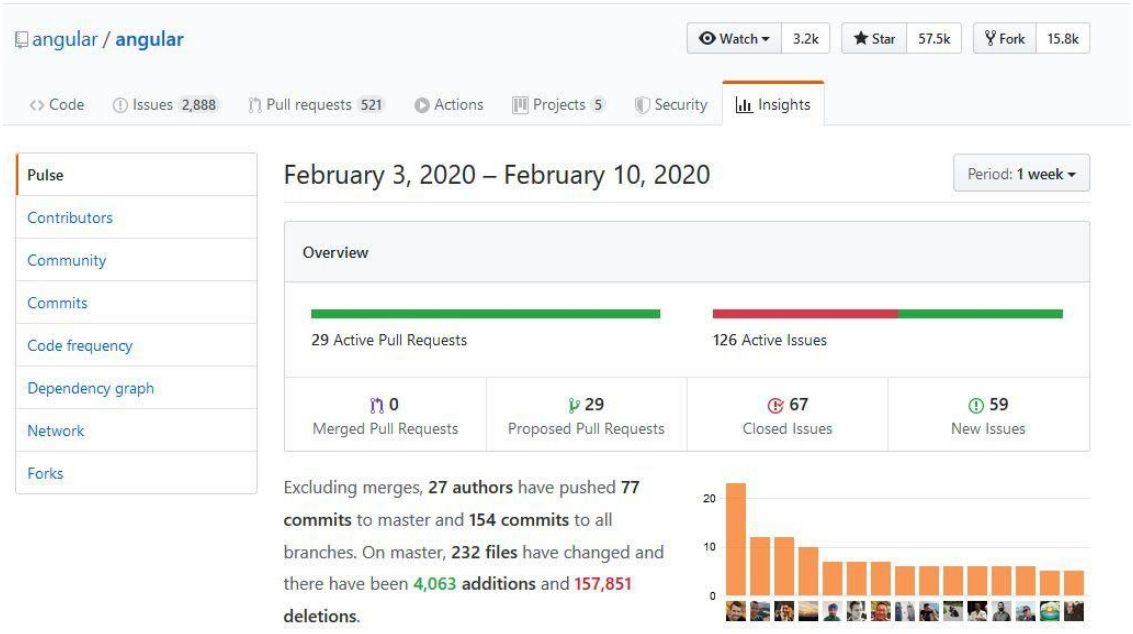
Pour Vue.js, 25 commiteurs avec pour les 10 plus actifs :

- Un total de 98 commits répartis de 1 à 63 commits par personne.
- 1 principal intervenant à 63 commits (Evan You), le suivant à 17.

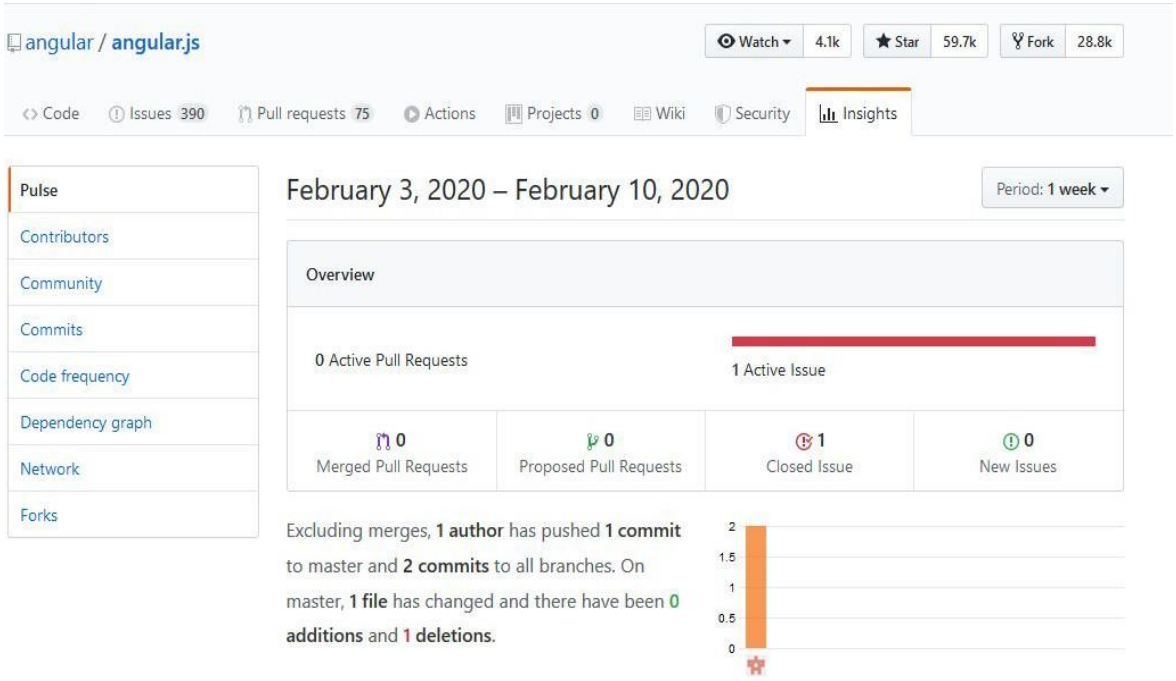


Pour Angular, 39 commiteurs avec pour les 10 plus actifs :

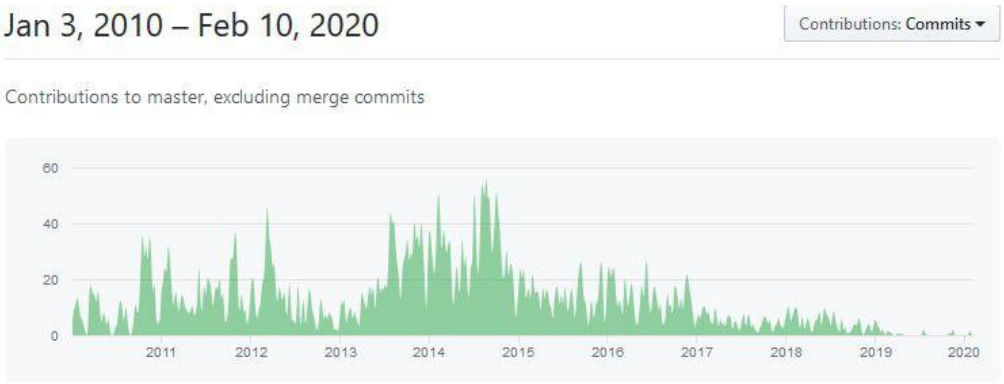
- Un total de 202 commits répartis de 10 à 44 commits par personne.
- 6 sont employés chez Google et totalisent 100 commits.
- Le premier (non Google) à 44, le second à 33.



Par contre on peut bien voir que angularJs un seul commiteur avec un seul commit en 7 jours



On peut bien voir sur le graphe qui suit qu'à partir de 2015 sa popularité a chuté drastiquement et en 2020 il n'y a presque plus de contributeurs



4.3.2.7 Les soutiens de poids

Contrairement à Angular et React.js, Vue.js n'est pas officiellement soutenu par de grossesentités. Mais comme ce framework est utilisé par d'importantes sociétés.
Mais c'est surtout l'actuel débat autour du framework JavaScript qui sera intégré à Wordpress qui peut faire entrer Vue.js, en concurrence avec React.js, dans une autre dimension, surtout s'il n'est pas simplement intégré au noyau du moteur de blog, mais également "imposé" comme solution officielle pour le développement des modules additionnels.

4.3.2.8 La documentation

La qualité de la documentation est indéniable pour l'ensemble des solutions considérées ; on peut toutefois donner un léger avantage aux solutions orientées "framework tout-en-un" puisqu'elles garantissent de fait une documentation

centralisée et de qualité homogène ; et parmi les solutions de cette catégorie, Angular se démarque par sa documentation qui apporte des exemples plus détaillés et plus proches de la complexité des cas de figure réellement rencontrés au quotidien.

4.3.2.9 Les performances

Afin de comparer la performance des 4 frameworks, nous avons utilisé un benchmark, pour tester le temps d'exécution sur différentes opérations.

Et on peut clairement voir que vue.js se démarque bien des autres frameworks pour la plupart des benchmarks. [9]

Temps d'exécution en milliseconde +- déviation standard (ralentissement = durée / la durée la plus courte)

Name	vue-v2.5.16-keyed	angular-v6.1.0-keyed	react-v16.4.1-keyed	angularjs-v1.7.2-keyed
create rows Duration for creating 1000 rows after the page loaded.	182.1 ±7.6 (1.0)	185.2 ±10.2 (1.0)	180.5 ±7.3 (1.0)	225.5 ±10.7 (1.2)
replace all rows Duration for updating all 1000 rows of the table (with 5 warmup iterations).	158.8 ±2.7 (1.0)	161.2 ±2.7 (1.0)	157.3 ±2.0 (1.0)	201.4 ±6.4 (1.3)
partial update Time to update the text of every 10th row (with 5 warmup iterations) for a table with 10k rows.	156.4 ±9.8 (2.3)	68.8 ±3.7 (1.0)	81.9 ±2.7 (1.2)	90.0 ±5.8 (1.3)
select row Duration to highlight a row in response to a click on the row. (with 5 warmup iterations).	10.6 ±2.0 (1.0)	7.9 ±4.3 (1.0)	10.3 ±2.1 (1.0)	9.7 ±1.3 (1.0)
swap rows Time to swap 2 rows on a 1K table. (with 5 warmup iterations).	20.0 ±2.9 (1.0)	105.8 ±1.8 (5.3)	106.5 ±1.9 (5.3)	106.8 ±1.1 (5.3)
remove row Duration to remove a row. (with 5 warmup iterations).	54.2 ±2.2 (1.2)	47.1 ±3.0 (1.0)	49.6 ±0.8 (1.1)	50.4 ±0.8 (1.1)
create many rows Duration to create 10,000 rows	1,603.2 ± 34.8 (1.0)	1,693.9 ± 70.1 (1.1)	1,935.4 ± 33.6 (1.2)	2,126.4 ± 71.4 (1.3)
append rows to large table Duration for adding 1000 rows on a table of 10,000 rows.	342.5 ±6.0 (1.4)	243.3 ±6.3 (1.0)	268.6 ±6.9 (1.1)	305.7 ±11.6 (1.3)
clear rows Duration to clear the table filled with 10.000 rows.	191.9 ±6.1 (1.1)	263.9 ±3.0 (1.5)	175.4 ±4.1 (1.0)	419.2 ±6.9 (2.4)
slowdown geometric mean	1.17	1.27	1.27	1.54

Métriques de démarrage

Name	vue-v2.5.16-keyed	angular-v6.1.0-keyed	react-v16.4.1-keyed	angularjs-v1.7.2-keyed
consistently interactive a pessimistic TTI - when the CPU and network are both definitely very idle. (no more CPU tasks over 50ms)	2,252.7 ± 0.2 (1.0)	3,278.5 ± 4.0 (1.5)	2,477.6 ± 0.6 (1.1)	2,854.0 ± 0.7 (1.3)
script bootup time the total ms required to parse/compile/evaluate all the page's scripts	55.1 ±1.5 (1.0)	235.2 ±8.5 (4.3)	65.6 ±2.3 (1.2)	124.6 ±2.0 (2.3)
main thread work cost total amount of time spent doing work on the main thread. Includes style/layout/etc.	420.6 ±67.5 (1.0)	679.3 ±5.3 (1.6)	466.0 ±3.9 (1.1)	530.6 ±2.5 (1.3)
total byte weight network transfer cost (post-compression) of all the resources loaded into the page.	215,445.0 ± 0.0 (1.0)	365,497.0 ± 0.0 (1.7)	251,915.0 ± 0.0 (1.2)	328,568.0 ± 0.0 (1.5)

Allocation mémoire en Megabytes

Name	vue-v2.5.16-keyed	angular-v6.1.0-keyed	react-v16.4.1-keyed	angularjs-v1.7.2-keyed
ready memory Memory usage after page load.	2.7 ±0.2 (1.0)	6.0 ±0.0 (2.2)	2.8 ±0.2 (1.0)	3.4 ±0.2 (1.2)
run memory Memory usage after adding 1000 rows.	7.1 ±0.0 (1.1)	9.8 ±0.0 (1.5)	6.7 ±0.0 (1.0)	10.9 ±0.0 (1.6)
update eatch 10th row for 1k rows (5 cycles) Memory usage after clicking update every 10th row 5 times	7.2 ±0.0 (1.0)	9.8 ±0.0 (1.4)	7.6 ±0.0 (1.1)	10.9 ±0.0 (1.5)
replace 1k rows (5 cycles) Memory usage after clicking create 1000 rows 5 times	7.2 ±0.0 (1.0)	10.1 ±0.0 (1.4)	7.9 ±0.0 (1.1)	11.4 ±0.1 (1.6)
creating/clearing 1k rows (5 cycles) Memory usage after creating and clearing 1000 rows 5 times	3.0 ±0.0 (1.0)	6.4 ±0.0 (2.1)	3.8 ±0.0 (1.3)	3.9 ±0.0 (1.3)

5 Les outils utilisés

Parmi les outils que nous avons pu utiliser, on peut citer :

5.1 MySql Workbench

MySQL Workbench (anciennement *MySQL administrator*) est un logiciel de gestion et d'administration de bases de données MySQL créé en 2004. Via une interface graphique intuitive, il permet, entre autres, de créer, modifier ou supprimer des tables, des comptes utilisateurs, et d'effectuer toutes les opérations inhérentes à la gestion d'une base de données. Pour ce faire, il doit être connecté à un serveur MySQL.

5.2 Fonctionnalités

5.2.1 Visualisation graphique des performances

L'une des spécificités de ce client de gestion de base de données réside dans sa possibilité de visualiser en temps réel, via des indicateurs graphiques, la charge du serveur MySQL, le pourcentage d'occupation en mémoire, les connexions courantes, le nombre de requêtes continue sur une base de données. Il permet aussi de voir l'occupation disque, ainsi que l'espace alloué aux tables et aux fichiers de logs.

Enfin, les tables sont disponibles sous forme de diagramme, permettant ainsi la modélisation des données.

5.2.2 Sauvegarde et restauration des données

Le logiciel permet, comme les autres clients de gestion de base de données, de créer facilement des sauvegardes de tables et de bases. Il permet aussi de restaurer rapidement des données, par simple sélection des tables via l'interface de gestion.

5.2.3 Transfert entre SGBD

Une fonction permet d'exporter des bases de Microsoft SQL Server vers MySQL. [10]

5.3 Visual Paradigm for UML

Visual Paradigm for UML est un logiciel de création de diagrammes, dans le cadre d'une programmation tout-en-un, il possède plusieurs options, permettant une large possibilité de modélisation en UML. [12]

6 Organisation du code

L'architecture utilisée afin que le code soit clair et précis est le modèle MVC, son but est de séparer la logique du code en trois parties que l'on retrouve dans des fichiers distincts : Model, view et Controller.

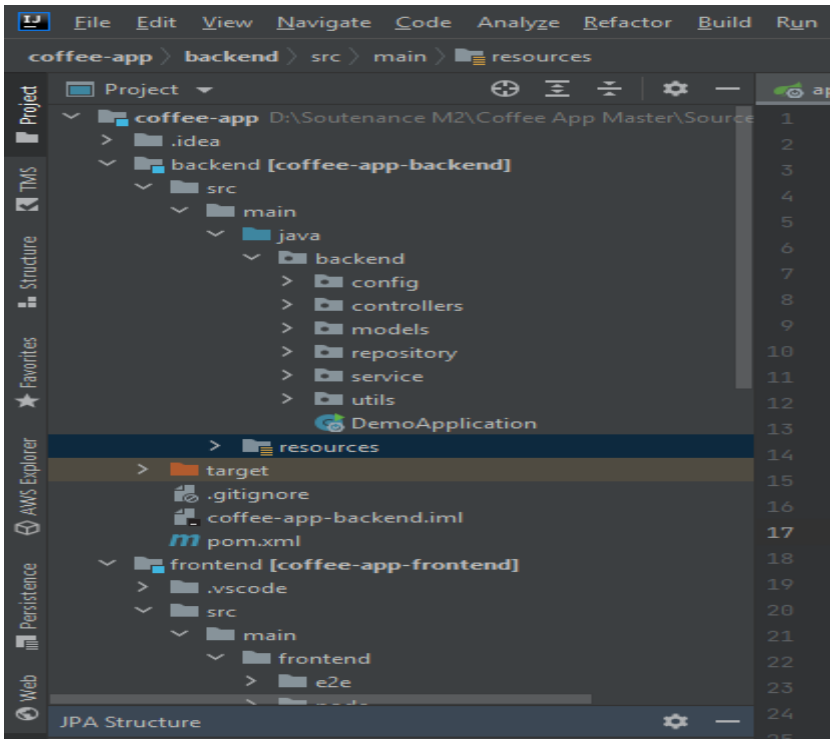


Figure 7 - Organisation du code.

6.1 Model (modèle)

Le rôle de cette partie est de récupérer les données dans la base de données, de les organiser et de les assembler pour qu'elles puissent être traitées par le contrôleur. [13]

6.2 View (vue)

Cette partie se focalise plus particulièrement sur la partie graphique de l'application, elle ne fait pratiquement aucun calcul, mais se contente de récupérer des variables pour savoir ce qu'elle doit afficher. [14]

6.3 Controller (Contrôleur)

Cette partie s'occupe de gérer le code qui prend des décisions, c'est en quelque sorte l'intermédiaire entre le modèle et la vue. [14]

6.4 Méthodologie de conception

Le Processus Unifié (UP pour UnifiedProcess) est un processus générique de développement logiciel construit sur UML. Générique décrit qu'il est nécessaire d'ajuster UP au contexte du projet à réaliser. C'est un patron de processus pouvant être adapté à une large classe de systèmes logiciels et à différents domaines d'applications. [15]

6.5 Le formalisme

UML est l'acronyme anglais pour « Unified Modeling Language ». On le traduit par

« Langage de modélisation unifié ». La notion UML est un langage visuel constitué d'un ensemble de schémas appelés des diagrammes qui donnent chacun une vision différente du projet à traiter.UML nous fournit donc des diagrammes pour représenter le logiciel à développer : son fonctionnement, sa mise en route et les actions susceptibles d'être effectuées par le logiciel. [16]

6.6 Avantages de l'UML

L'enthousiasme d'utiliser le langage UML est dû à ses nombreux avantages. En effet, il favorise l'utilisation d'outils, constituant à cet effet un gage de stabilité, permet le bénéfice d'un travail précis et facilite la compréhension de représentations abstraites complexes. [17]

Toutefois, ce langage reste très complexe et n'est pas facile à assimiler, surtout lorsque nous souhaitons obtenir rapidement un gain de productivité.

6.7 Présentation de l'UML

UML dans sa 2e version (UML 2.0) propose treize (13) diagrammes pouvant être utilisés dans la description d'un système. Ces diagrammes sont regroupés dans deux grands ensembles. [18]

- Les diagrammes structurels
- Les diagrammes de comportement

7 L'architecture MVC de la plateforme Coffee App

Le Framework Spring se base sur une architecture MVC comme nous indique la figure suivante. Le contrôleur est responsable de la logique de contrôle de l'application, il sert à gérer les demandes des utilisateurs et à récupérer des données, en tirant parti des modèles, les modèles servent à interagir avec la base de données et récupérer les informations des objets, les vues pour afficher des pages.

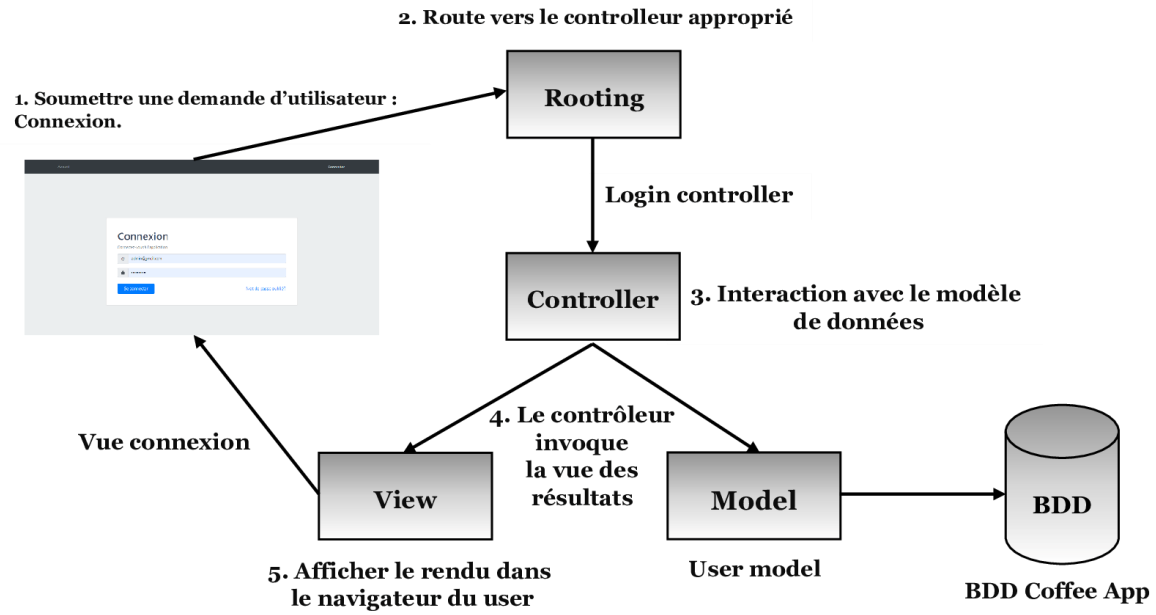


Figure 8 - Architecture MVC de Coffee App.

8 L'implémentation de la plateforme Coffee App

8.1 Manuel d'installation/exécution

- Veuillez à avoir Angular et l'IDE Intellij installé.
- Veuillez à avoir Node js et apache maven installé.
- Placez-vous sur votre dossier : **coffee-app-master > Sources > coffee-app**
- Exécutez les deux commandes : **mvn install** et **npm install** sur le Shell
- Placez-vous sur votre dossier pour lancer l'application angular : **coffee-app-master > Sources > coffee-app > frontend > src > main > frontend**
- Exécutez la commande : **ng build**

8.1.1 Installation de la plateforme

1. Créer un utilisateur avec un username admin et un password coffee sur MySQL.
2. Créer une base de données MySql et la nommer coffee_bd.
3. Affecter tous les privilèges à l'utilisateur admin sur cette base de données.
4. Exécuter le fichier CoffeeApp.bat pour lancer la plateforme.
5. D'abord lancer le serveur, sous intelliJ cliquez sur le triangle près de DémoApplication, ou sur Run -> Run 'DémoApplication'
6. Se rendre sur localhost:8080/home.

8.1.2 Intégration de la machine à café

1. Créer un compte utilisateur avec le compte admin (admin@gmail.com, P@sser1234),
2. Créer une machine en lui assignant comme utilisateur, l'utilisateur créé dans 2,
3. Récupérer l'id de la machine en question,
4. Modifier le fichier de configuration(conf.properties) de l'IoT (CoffeeIot) en mettant iot.machine le id de la machine (étape 4), le port de la machine (iot.agentPort) et la fréquence de synchronisation avec la plateforme (iot.frequency) en millisecondes.
5. Changer dans le fichier de configuration (conf.properties) l'URL et le port de la plateforme(localhost et 8080 par défaut),
6. Exécuter le fichier coffeeIoT.bat pour lancer la machine à café.

8.1.2.1 Exemple de scénario

1. Ajouter un produit dans la plateforme
2. Exécuter le endpoint GET /api/products de l'iot (localhost:<iot.agentPort>): Le endpoint affichera le produit ajouté dans l'étape 1.
3. Consommer une quantité du produit en sollicitant le endpoint POST /api/products/consume
4. Vérifier que la quantité a été mise à jour dans la plateforme.

Remarque: L'état d'une machine à café est synchronisé avec la plateforme tous les x temps (iot.frequency).

8.2 Maquettage du système

Le maquettage reste une des étapes les plus importantes à définir afin d'avoir un rendu visuel de notre application, nous sommes passés par un maquettage sur papier (Annexe A) avant de pouvoir définir les interfaces qui suivent.

Pour la page d'accueil, nous rendons disponibles quelques informations utiles :

- **Un encart indiquant si c'est le week-end :**

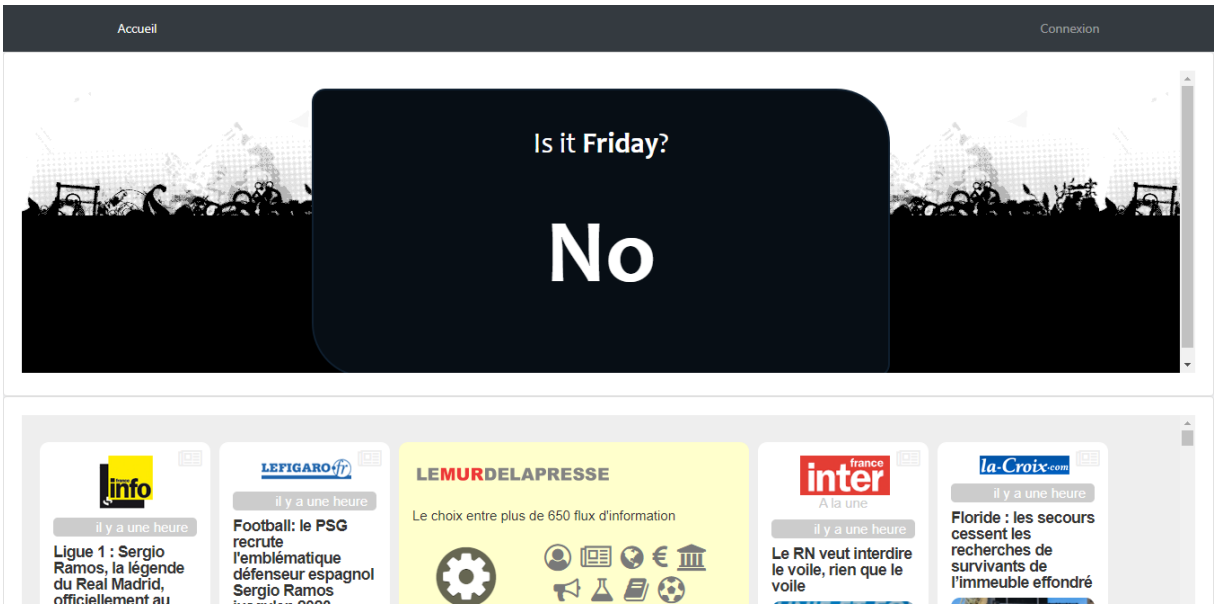


Figure 9 - Encart week-end.

- **Un encart presse :**

Figure 10 - Encart presse.

- **Un "widget" météo**

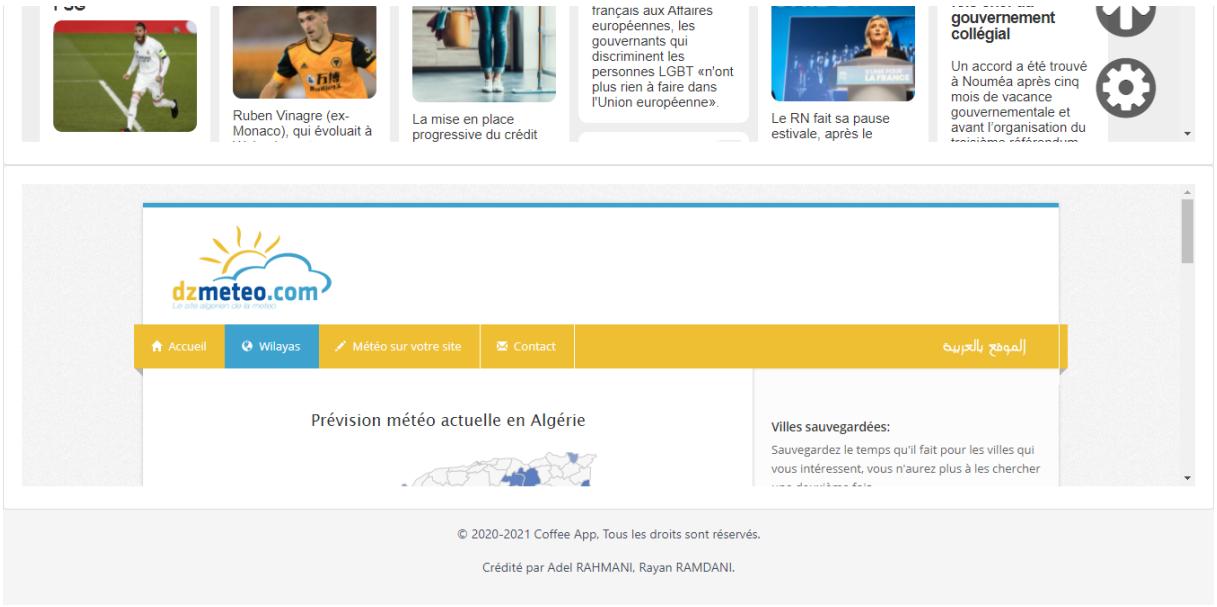


Figure 11 - Widget météo.

8.3 Manuel d'utilisation

8.3.1 Manuel d'administrateur

8.3.1.1 Interface d'authentification

Sur cette interface l'utilisateur doit inscrire son mail ainsi que son propre mot de passe afin de pouvoir y accéder à l'ensemble de fonctionnalités que propose l'interface.

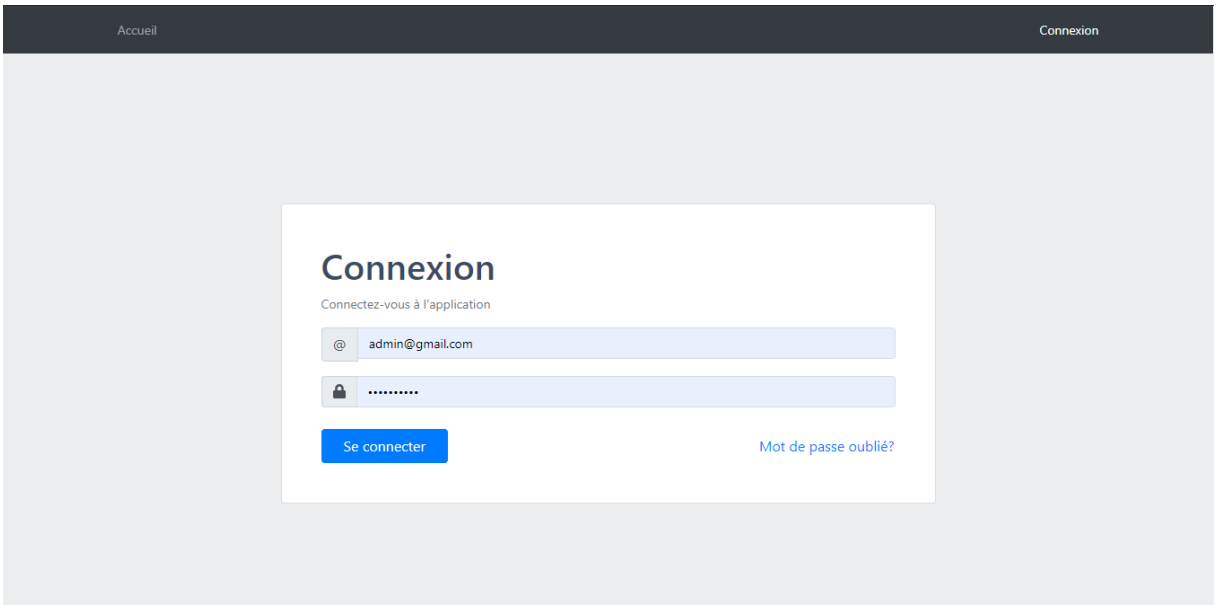


Figure 12 - Interface d'authentification.

Une fois connecté, l'administrateur peut accéder, via la barre de navigation, aux différentes interfaces degestion : utilisateurs, machines et produits.

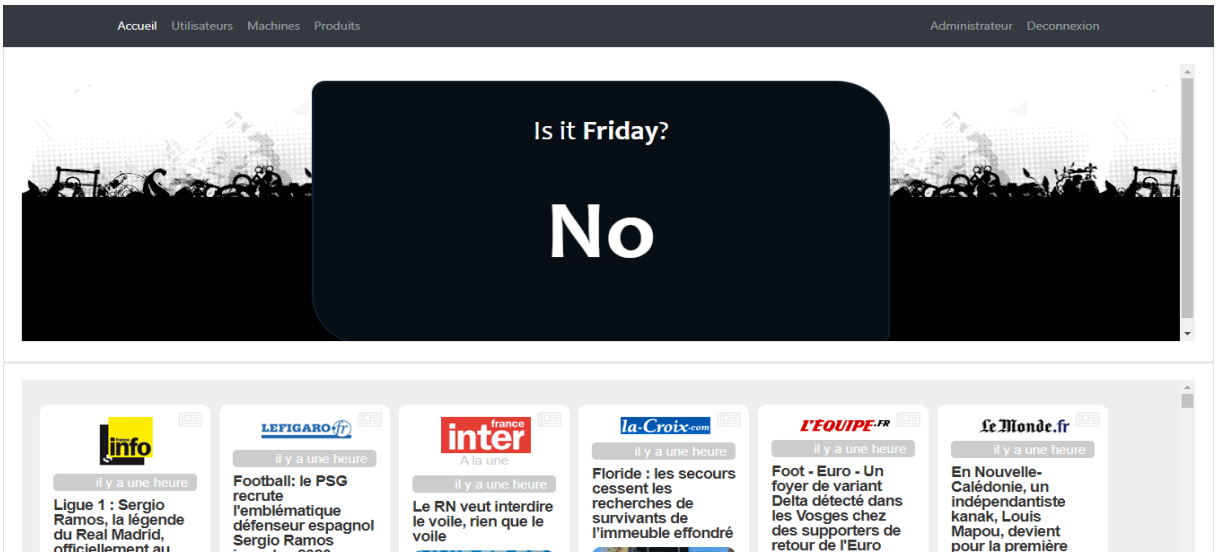


Figure 13 – Manuel administrateur Coffee App.

8.3.1.2 Gestion des utilisateurs

L'administrateur peut, via cette interface, ajouter, modifier et supprimer un utilisateur, il peut aussiexporter la liste des utilisateurs sous format PDF. Quand un utilisateur est créé, son compte n'est pas directement actif. Il reçoit un email qui lui permettra de choisir son mot de passe, son compteest directement activé.

NB: Cette fonctionnalité d'envoi de mail ne fonctionnera pas en présence d'un serveur proxy quibloque l'envoi d'email non sécurisé. Dans les autres cas, il faut également consulter le dossier des emails indésirables.

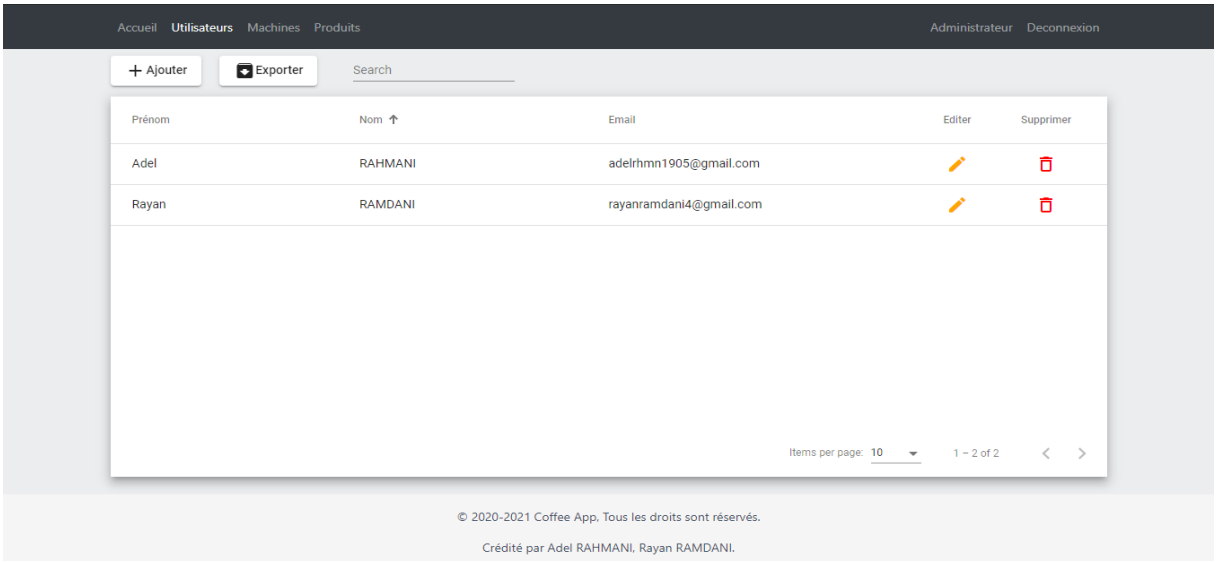


Figure 14 - Gestion des utilisateurs admin.

8.3.1.3 Gestion des machines

Via cette interface, l'administrateur peut : créer, configurer et supprimer une machine, il peut aussiassigner un utilisateur à une machine

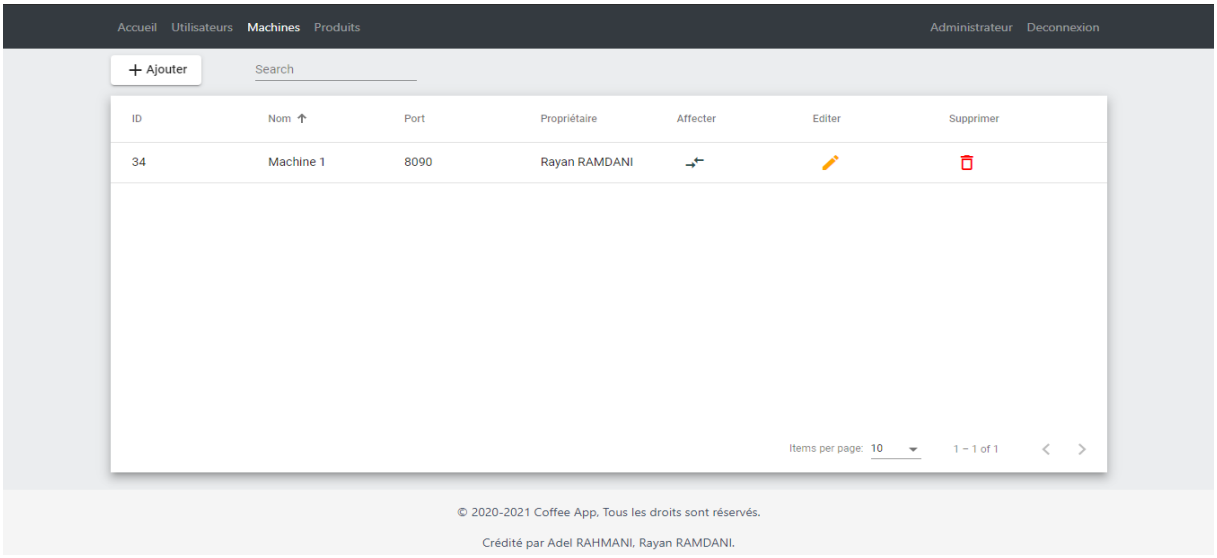


Figure 15 - Gestion des machines admin.

8.3.1.4 Gestion des produits

Sur cette interface, l'administrateur peut seulement lister les produits et les exporter en PDF.

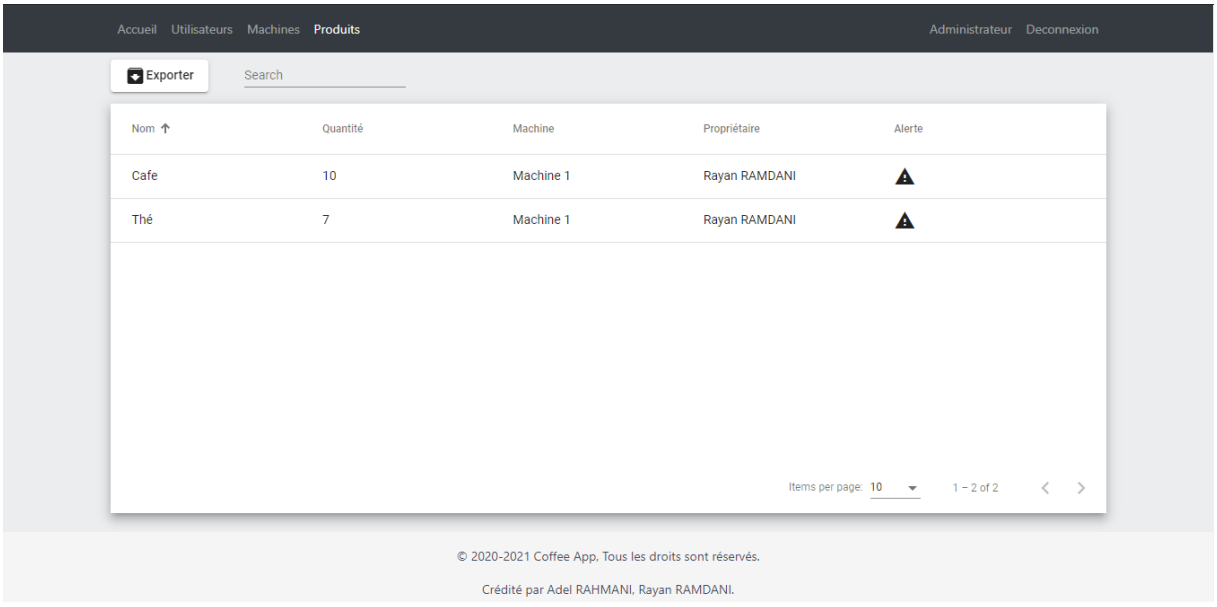


Figure 16 - Gestion des produits admin.

8.3.2 Manuel d'utilisateur

Une fois connecté, l'utilisateur peut accéder, via la barre de navigation, aux interfaces permettant de gérer ses, ses produits, et son compte.

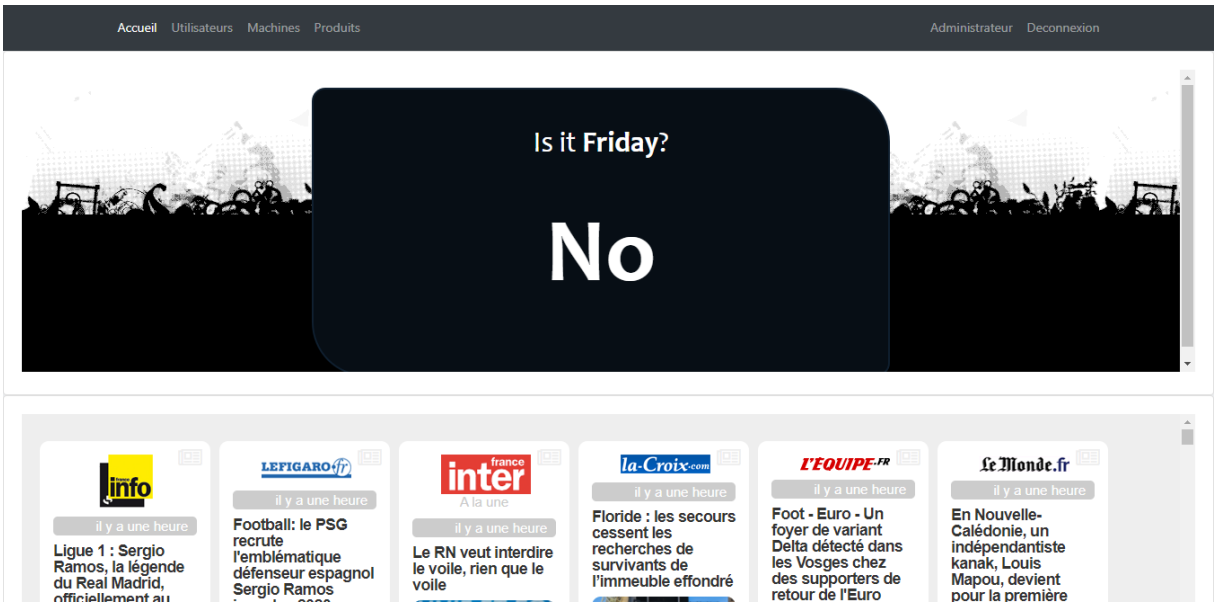


Figure 17 - Manuel utilisateur Coffee App.

8.3.2.1 Gestion des machines

L'utilisateur peut lister les machines qui lui sont assignées et les configurer.

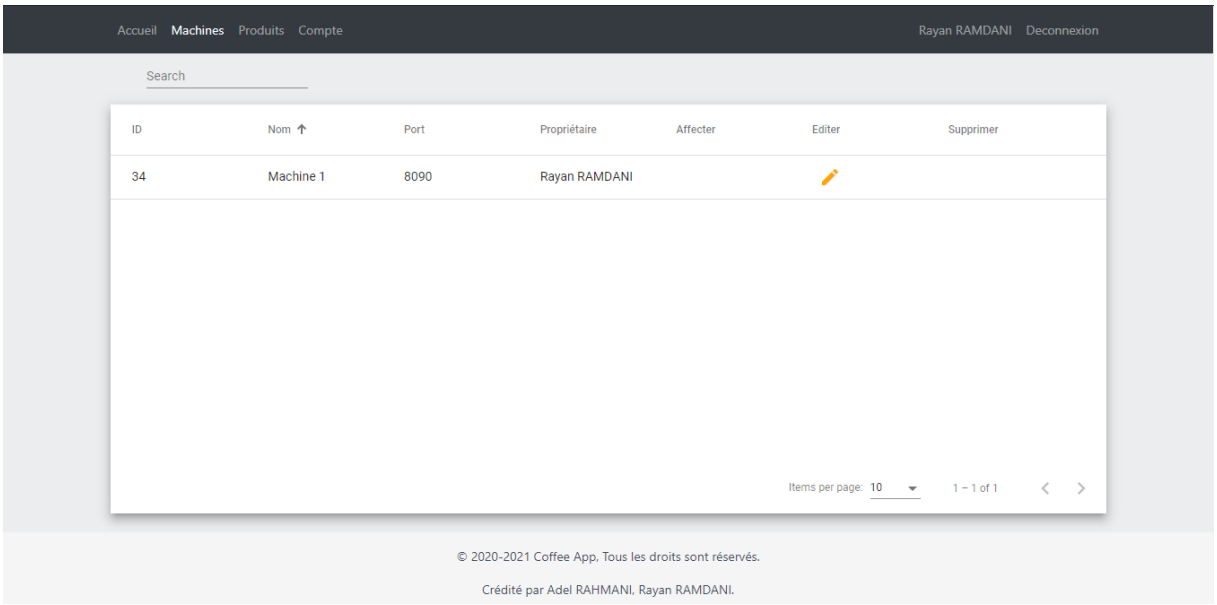


Figure 18 - Gestion des machines utilisateur.

8.3.2.2 Gestion des produits

L'utilisateur ajouter, modifier et supprimer ses produits.

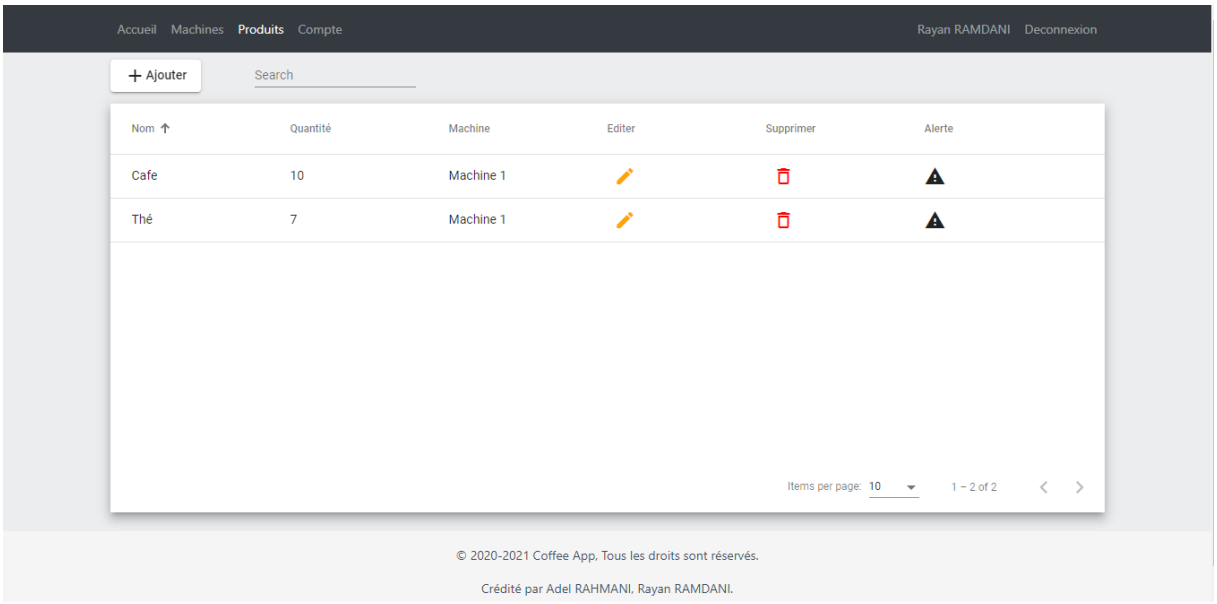


Figure 19 - Gestion des produits utilisateur.

8.3.2.3 Gestion du compte

Ici, l'utilisateur peut modifier son compte (modifier nom et prénom, changer le mot de passe).

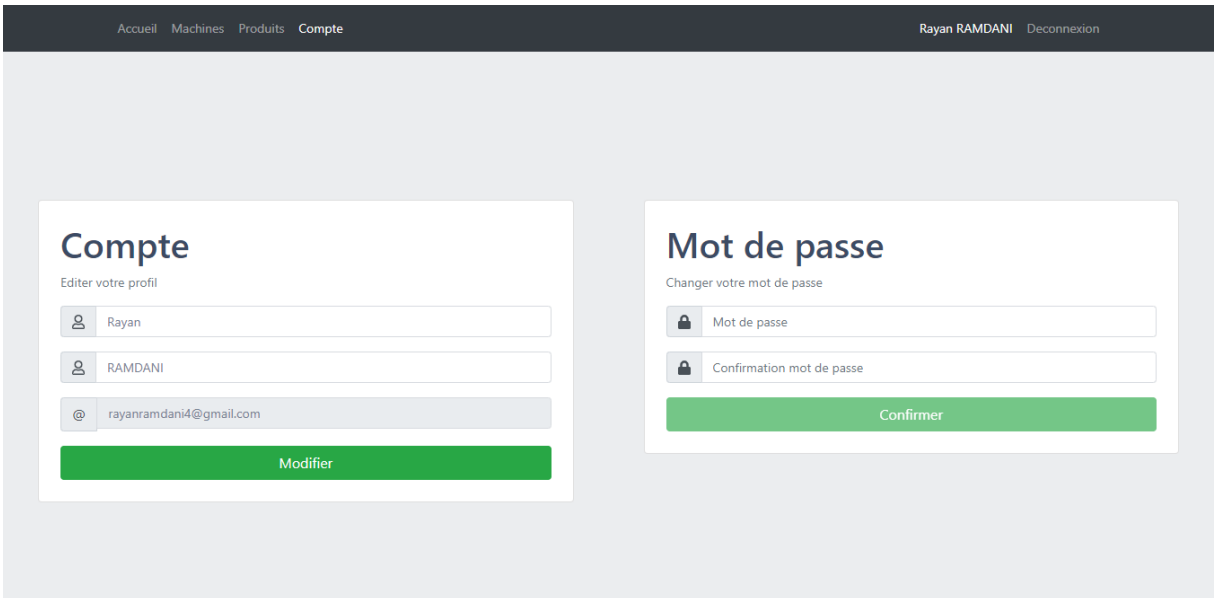


Figure 20 - Gestion de compte utilisateur.

Conclusion générale

Ce projet de fin d'études avait pour ambition d'établir une plateforme de gestion de machines à café innovante, en se demandant si le besoin réel existait ou pas et faire face à l'absence de ce genre de technologies adéquates pour la meilleure utilisation de ces machines.

Le rapport mentionne toutes les étapes traversées pour arriver au résultat attendu. Il a fallu dans un premier temps recenser les différents besoins existants, nous avons pu aussi donner un contexte général à notre projet et identifier les différentes exigences du futur système.

Nous avons établi ensuite une étude des systèmes existants qui nous permettent de connaitre les fonctionnalités primordiales. Par la suite nous avions entamé la phase d'analyse et de conception du système.

Pour conclure le dernier chapitre, notre projet d'étude atteint sa fin. Tout au long de ce chapitre, nous avons abordé notre environnement de travail. Par la suite, nous avons expliqué notre architecture d'application afin de présenter finalement les différentes principales parties d'implémentation de notre application réalisée.

Ce travail nous a été très formateur, puisqu'il a permis de découvrir une nouvelle technologie innovante, et nous a permis également de nous confronter à plusieurs contraintes à la fois : contraintes de temps, contraintes d'expérience et de technologie, en outre, ce projet nous a permis d'approfondir nos connaissances dans les bonnes pratiques de l'ingénierie génie logiciel.

Perspectives

Notre réalisation est encore d'actualité et ne s'arrête pas à ce niveau. En effet plusieurs perspectives s'offrent à ce projet :
Pour rendre l'application plus expansive en termes de base de données, nous proposons d'utiliser MySQL workbench.

Vu l'accomplissement de projet, nous souhaitons très fortement qu'il soit le fruit du progrès, de l'évolution et qu'il reste à la hauteur des exigences des entreprises.

9 Références bibliographiques et webographiques

[1] https://www.techno-science.net/definition/11021.html

[2] https://www.redhat.com/fr/topics/api/what-is-a-rest-api

[3] https://fr.wikipedia.org/wiki/MySQL

[4] https://openclassrooms.com/fr/courses/3936801-composez-des-interfaces-utilisateurs-en-material-design/3936808-comprenez-la-notion-dinterface-utilisateur

[5] https://openclassrooms.com/fr/courses/4525361-realisez-un-dashboard-avec-tableau

[6] https://planonsoftware.com/fr/glossaire/l-internet-des-objets/

[7] https://openclassrooms.com/fr/courses/4668056-construisez-des-microservices/5122425-decouvrez-le-framework-spring-boot

[8] https://intellitech.pro/fr/tutorial-0-comment-sauthentifier-avec-spring-security/

[9] https://insights.stackoverflow.com/survey/2019#technology- -most-loved-dreaded-and-wanted-web-frameworks

[10] https://fr.wikipedia.org/wiki/MySQL Workbench

[11] Visual Paradigm for UML. Https://www.clubic.com/telecharger-fiche384046-visual-paradigm-for-uml.html. 2019. Consulté le 14/05/2019.

[12] AdobePhotoshop. Https ://fr.wikipedia.org/wiki/Adobe Photoshop.2019. Consulté le 13/03/2021.

[13] Le modèle. Http ://dictionnaire.sensagent.leparisien.fr/Mod C3 A8le-Vue-Contr C3 4leur/fr-fr/. 2019. 12/05/2021

[14] La vue. Http ://dictionnaire.sensagent.leparisien.fr/Mod C3 A8le-Vue-Contr C34leur/fr-fr/. 2019. 19/04/2021

[15] Le contrôleur. Http ://dictionnaire.sensagent.leparisien.fr/Mod C3 A8le-Vue-ContrC3 B4leur/fr-fr/. 2019. 02/05/2021

[16] Laurent PIECHOCKI Frédéric DI GALLO. Cours UML, ENITA de Bordeaux, 2005.

[17] Joseph GABAY David GABAY. UML 2 analyse et Conception. DUNOD, 1re Édi-tion, 2008.

[18] Michel GRIMALDI. Modélisation UML Diagrammes structurels, 2007.

[18] Pascal ROQUES. UML 2 par la pratique. EYROLLES, 7e édition, 2009.

Annexes

Annexe A : maquettes de la plateforme Coffee App à l'étape 1 dessinées.

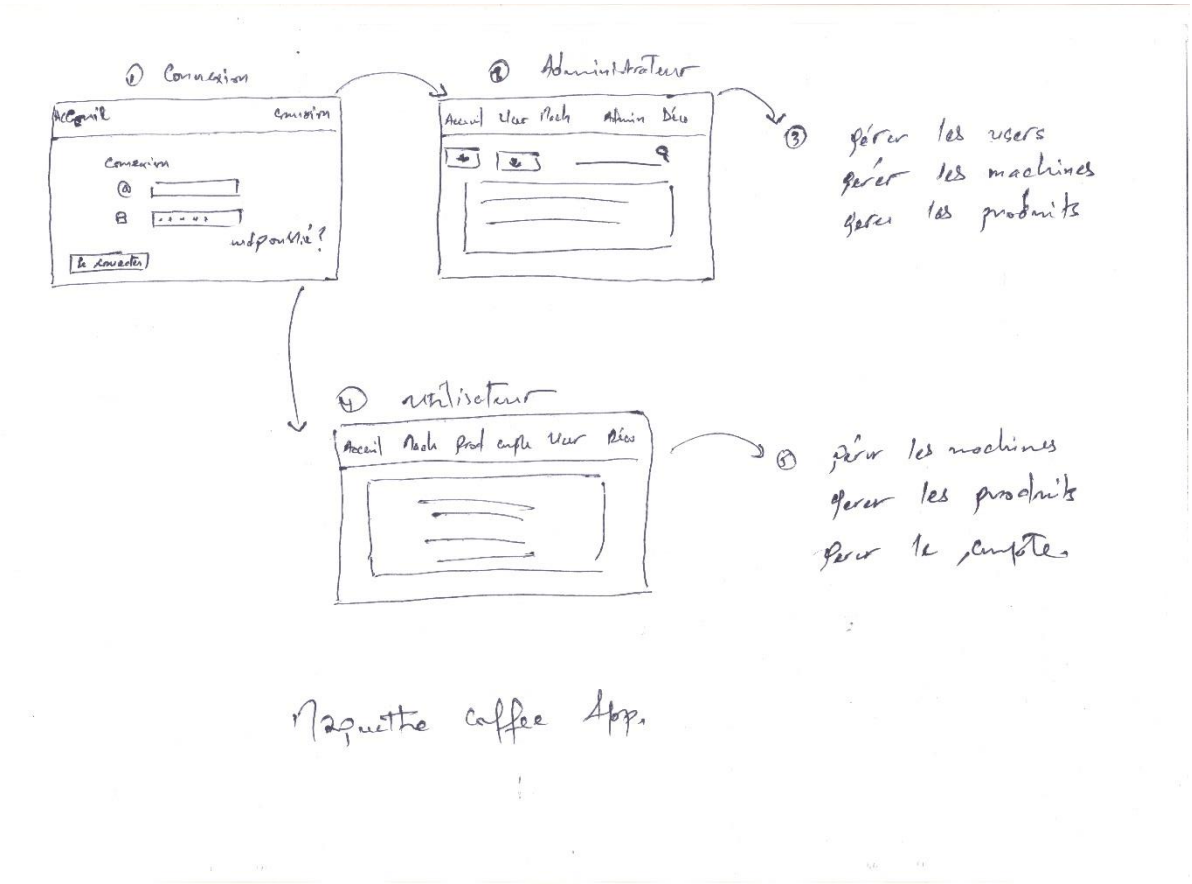


Figure 21 - Maquette Coffee App.

Résumé

Ce travail a été réalisé dans le cadre d'un projet de fin de cycle, il a pour objectif l'obtention du diplôme de master.

Il contient l'étude, la conception et la réalisation d'une plateforme, qui permet la gestion des machines à café au sein d'une entreprise donnée.

La réalisation de ce projet a nécessité une bonne analyse conceptuelle, nous avons donc approfondi nos connaissances en UML, suivant le processus UP, afin d'aboutir à un meilleur résultat lors de la réalisation.

Mots clés : Conception, réalisation, application, micro services, gestion, entreprise, Spring boot, UML, UP.

Abstract

This work was carried out within the framework of a final cycle project, with the objective of obtaining the master's degree.

It contains the study, the conception and the realization of a platform, which allows the management of coffee machines within a given company.

The realization of this project required a good conceptual analysis, so we deepened our knowledge in UML, following the UP process, in order to achieve a better result during the realization.

Keywords: Design, implementation, application, microservices, management, enterprise, Spring boot, UML, UP.