

# Privacy statement

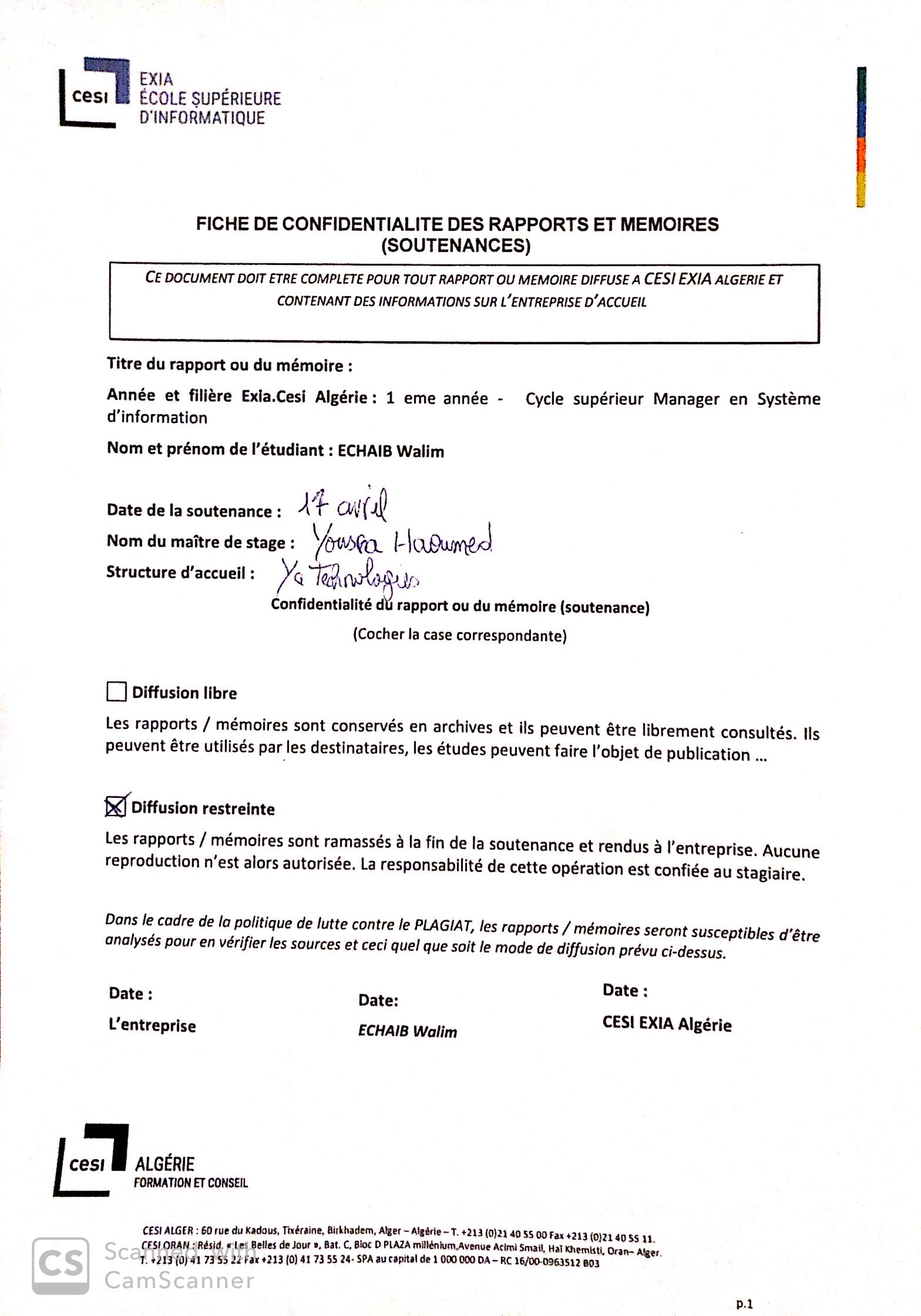


Table of content

[1. Fiche de confidentialité 3](#_Toc36224689)

[Table des figures : 5](#_Toc36224690)

[2. Remerciements : 6](#_Toc36224691)

[3. Introduction : 7](#_Toc36224692)

[4. Présentation de l’entreprise : 8](#_Toc36224693)

[5. Cahier des charges : 11](#_Toc36224694)

[6. Travail réalisé : 12](#_Toc36224695)

[I. Justification des choix techniques : 12](#_Toc36224696)

[a) Pourquoi Bootstrap ? 12](#_Toc36224697)

[b) Pourquoi React ? 13](#_Toc36224698)

[c) Pourquoi MERN ? 14](#_Toc36224699)

[d) Pourquoi SASS ? 15](#_Toc36224700)

[e) Pourquoi utiliser un Framework ? 16](#_Toc36224701)

[f) Socket IO 16](#_Toc36224702)

[g) NodeJS 17](#_Toc36224703)

[h) NPM 17](#_Toc36224704)

[i) ExpressJS 17](#_Toc36224705)

[j) Pourquoi MongoDB ? 18](#_Toc36224706)

[k) Pourquoi Redux ? 18](#_Toc36224707)

[II. Initialisation de l’environnement de travail : 19](#_Toc36224708)

[a) Familiarisation avec GitHub : 19](#_Toc36224709)

[b) Création du projet mobile : 19](#_Toc36224710)

[c) Utilisation de l’outil GIT : 20](#_Toc36224711)

[III. Base de données : 21](#_Toc36224712)

[a) Modélisation : 21](#_Toc36224713)

[b) Création des Schema : 22](#_Toc36224714)

[c) Les seeds: 23](#_Toc36224715)

[IV. Création de l’API : 24](#_Toc36224716)

[a) Création des Models : 24](#_Toc36224717)

[V. Controller : 24](#_Toc36224718)

[a) Le routage : 24](#_Toc36224719)

[b) Création des controller : 25](#_Toc36224720)

[c) La méthodologie utilisée pour la gestion : 27](#_Toc36224721)

[VI. Fonctionnalités : 28](#_Toc36224722)

[a) Mobile : 28](#_Toc36224723)

[1. Conception : 28](#_Toc36224724)

[2. Intégration des API (mobile) : 29](#_Toc36224725)

[3. Menu contextuel : 30](#_Toc36224726)

[4. Formulaires : 32](#_Toc36224727)

[5. Foire aux questions (côté mobile) : 34](#_Toc36224728)

[6. Conversations : 34](#_Toc36224729)

[7. L’application chat : 36](#_Toc36224730)

[8. Système de notation : 37](#_Toc36224731)

[1. English Summary: 39](#_Toc36224732)

[2. Résumé: 40](#_Toc36224733)

[3. Bilan du stage : 41](#_Toc36224734)

[4. Conclusion : 42](#_Toc36224735)

[5. Ressources : 43](#_Toc36224736)

# Table des figures :

[Figure 1: YA Technologies 9](#_Toc36224737)

[Figure 2: Organigramme YA Technologies 9](#_Toc36224738)

[Figure 3: Logo Yassir 10](#_Toc36224739)

[Figure 4: Application Yassir 10](file:///C:\Users\ASUS\Desktop\Exia.Cesi\Cycle%20sup\Troisième%20année\Stages\YA%20Technologies\Rapport\A3%20-%20Rapport%20de%20stage%20-%20Application%20d'une%20solution%20à%20un%20environnement%20existant.docx#_Toc36224740)

[Figure 5: Yassir Food Logo 11](#_Toc36224741)

[Figure 6: Yassir Food illustration 11](file:///C:\Users\ASUS\Desktop\Exia.Cesi\Cycle%20sup\Troisième%20année\Stages\YA%20Technologies\Rapport\A3%20-%20Rapport%20de%20stage%20-%20Application%20d'une%20solution%20à%20un%20environnement%20existant.docx#_Toc36224742)

[Figure 7 : React 14](#_Toc36224743)

[Figure 8: logo redux 18](#_Toc36224744)

[Figure 9: logo github 19](#_Toc36224745)

[Figure 10: repos GitHub 19](#_Toc36224746)

[Figure 11: création projet react-native 20](#_Toc36224747)

[Figure 12: cloner le projet git 20](file:///C:\Users\ASUS\Desktop\Exia.Cesi\Cycle%20sup\Troisième%20année\Stages\YA%20Technologies\Rapport\A3%20-%20Rapport%20de%20stage%20-%20Application%20d'une%20solution%20à%20un%20environnement%20existant.docx#_Toc36224748)

[Figure 13: synchroniser le dépôt local 21](#_Toc36224749)

[Figure 14: enregistrer les changements et les pousser sur le dépôt GIT 21](#_Toc36224750)

[Figure 15: logo draw.io 21](#_Toc36224751)

[Figure 16: schéma de la base de données 22](#_Toc36224752)

[Figure 17: logo mongoose 22](#_Toc36224753)

[Figure 18: Modèle de l'objet message 23](#_Toc36224754)

[Figure 19: seed de données à l'aide d'Insomnia 23](#_Toc36224755)

[Figure 20: modèle de l'objet agent 24](#_Toc36224756)

[Figure 21: route menant vers les fonctions de l'objet ticket 25](#_Toc36224757)

[Figure 22: controller de l'objet agent 26](#_Toc36224758)

[Figure 23: statistiques de tâches du premier sprint 27](#_Toc36224759)

[Figure 24: Kanban des tâches du deuxième sprint 28](#_Toc36224760)

[Figure 36: Logo Adobe XD 28](#_Toc36224761)

[Figure 37: Conception mobile 29](#_Toc36224762)

[Figure 38: menu de l'application 31](#_Toc36224763)

[Figure 39: navigation entre écrans 32](#_Toc36224764)

[Figure 40: fonction pour la navigation 32](#_Toc36224765)

[Figure 41: composant du formulaire 33](#_Toc36224766)

[Figure 42: foire aux questions côté mobile 34](#_Toc36224767)

[Figure 43: écran de conversations 35](#_Toc36224768)

[Figure 44: écran de chat 36](#_Toc36224769)

[Figure 45: page de notation 38](#_Toc36224770)

# Acknowledgements:

First of all, I would like to thank Mr and Mrs BENTALEB, CEO of Vibrand, for having integrated me into their team during these 2 months of internship where they volunteered to propose me a concrete project so that I could apply a good number of notions learned at CESI.

# Introduction :

Dans le cadre de ma première année de cycle supérieur du CESI Ecole D’Ingénieurs, j’ai choisi d’effectuer mon stage de 3 mois et demi au sein d’une entreprise qui propose divers services dans les domaines du transports et de la livraison sous la tutelle de madame HAOUMED Yousra, Project Development Manager à YA Technologies qui est la société éditrice de la célèbre application de VTC Yassir.

# Présentation de l’entreprise :



Figure 1: YA Technologies

YA Technologies, connue sous le nom de YA, est une nouvelle entreprise qui possède des bureaux à Palo Alto, dans la Silicon Valley, aux États-Unis et à Alger, en Algérie à Casablanca ainsi qu’à Tunis en Tunisie, qui compte changer la manière dont les services quotidiens sont fournis en Algérie et en Afrique grâce à des solutions innovantes numériques / mobiles de haute technologie, en établissant un pont entre les écosystèmes de la Silicon Valley et de l'Algérie.

Fondée en 2016, ils comptent une centaine d’employés dont des équipes de développeurs maitrisant les dernières technologies sur le marché et qui veillent à l’intégrité et au bon fonctionnement des services qu’ils proposent mais aussi de développer d’autres outils permettant à faciliter la vie des gens.

La structure Yassir comprend :

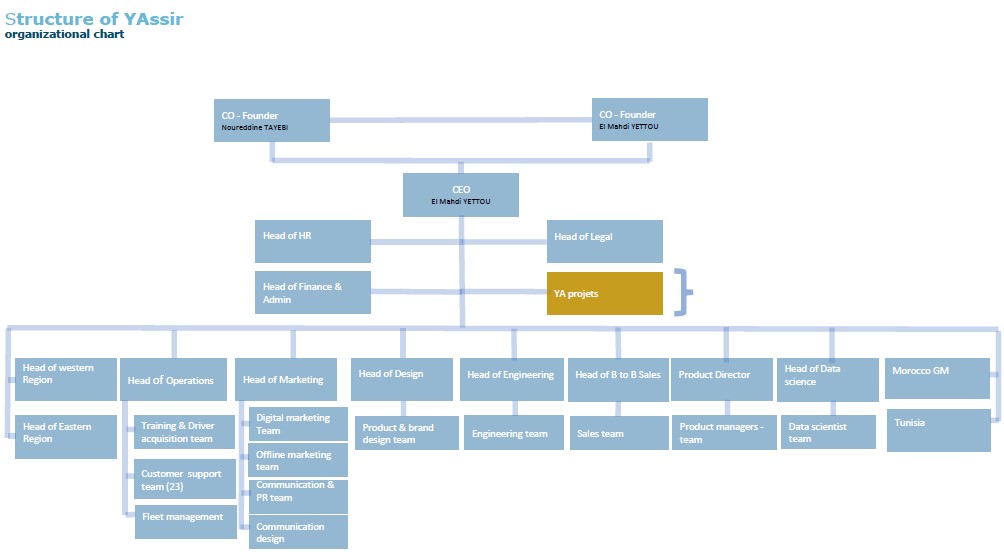


Figure 2: Organigramme YA Technologies

**Ils ont pour services connus :**

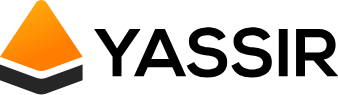


Figure 3: Logo Yassir

La célèbre application de VTC lancée en 2017, unique société à proposer ce service à cette époque ils prennent le monopole à l’aide d’une bonne communication, l’application pour ses débuts ciblait la capitale avant de se développer et desservir l’Algérie entière, mais ce n’est pas tout, l’étendue de cette application touchera exclusivement le Maghreb arabe pour rejoindre le Maroc et la Tunisie.

Figure : Application Yassir

Elle compte des milliers de chauffeurs mis à disposition des

citoyens et se distingua par la facilité d’utilisation de l’application mais aussi de la possibilité de réserver des courses pour un planning précis.

Donnant la chance à des particuliers d’arrondir leur fin de mois tout en tirant un bénéfice qui mettra tout le monde d’accord afin de satisfaire le client.



Figure 5: Yassir Food Logo



Figure : Yassir Food illustration

Un service récent proposé par YA Technologies, le concept existait déjà de telle sorte à ce que chaque restaurant proposait la livraison avec ses propres chauffeurs et étaient débordés d’appels téléphoniques car ils n’avaient pas de système pour répertorier les commandes des clients, ce qui créa la confusion chez le client car dans certains cas il y avait des oublis, c’est pour cela que Yassir Food a été mis en place, il permet aux clients de réserver dans un des restaurants de leur choix la nourriture de leur choix, le restaurant de son côté recevra les commandes ordonnées dans une liste de commandes qu’il pourra lancer avec des identifiants prédéfinis de manière à ne pas se tromper.

# Cahier des charges :

**Objectif du projet**

Réaliser une plateforme permettant aux agents du call center (centre d’appels) de réceptionner les demandes d’aide des utilisateurs (via l’application mobile Yassir) directement sur le portail web, mais aussi de créer le module qui permettra à l’utilisateur de pouvoir communiquer avec cette plateforme en temps réel.

**Audience ciblée**

Les agents du call center, ainsi que les utilisateurs de l’application.

**1. Ce qu’ils attendent de l’application**

L’objectif étant d’optimiser la communication entre le support et le client qui se fait actuellement via téléphone sans aucune trace en :

* Mettant en place un système de chat en temps réel.
* Mise en place d’un système de FAQ.
* Mise en place d’un système de tickets pour assurer le suivi des demandes d’aide.
* La gestion des agents.
* Système de réponse automatique.

**2. Spécifications fonctionnelles**

**Front-end :**

Ayant une charte prédéfinie, nous avions dû nous l’intégrer dans nos calculs à la conception de l’interface réalisée en React.

**Maquettes**

N’ayant pas d’exigences précises en ce qui concerne la disposition des menus et du contenu, nous nous contenterons d’une interface « standard » de notre choix.

**Back-end :**

**Fonctionnalités**

* Envoi de messages en temps réel.
* Attribution de tickets de façon automatique.
* Création et édition de catégories, questions et réponses.
* Gestion des utilisateurs.
* Services reliés aux différents pays.
* Paramètres liés aux tickets.
* Paramètres liés à l’application mobile.
* Envoi de formulaire depuis l’application.
* Notifications push.

# Travail réalisé :

## Justification des choix techniques :

### Pourquoi Bootstrap ?



Eh bien, car c’est un Framework c’est-à-dire orchestre de composants mis à disposition, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d’une partie l’architecture d’un site internet, logiciel ou application. Par exemple, nous devons faire du routage pour un site, nous prenons un composant déjà tout prêt et qui a fait ses preuves et nous l'utilisons : gain de temps et fiabilité. L'utilité d'un Framework est d'éviter de passer du temps à développer ce qui a déjà été fait par d'autres, souvent plus compétents.

Avec les avancées technologiques et les différents types d’appareils disponibles sur le marché, nous avons été obligés d’adapter nos interfaces web aux différentes tailles afin de correspondre à ces derniers, c’est possible de faire cela avec du simple CSS (Media Queries) mais cela prendrait beaucoup plus de temps, c’est là qu’intervient Bootstrap, a grille Bootstrap comporte par défaut 12 colonnes. Cependant, il faut savoir qu’elle est flexible et extensible dans la mesure où l’on peut réduire le nombre de colonnes ou en insérer de nouvelles dans chaque colonne existante et cela juste en modifiant certaines class de notre HTML.

Le système Bootstrap possède quatre tailles de grilles prédéfinies pour que la page se module suivant la taille de l’écran selon deux hypothèses : soit les éléments se redimensionnent en restant positionnés, soit ils s'empilent quand la fenêtre devient plus étroite et se positionne côte à côte quand elle s'élargit.

### Pourquoi React ?

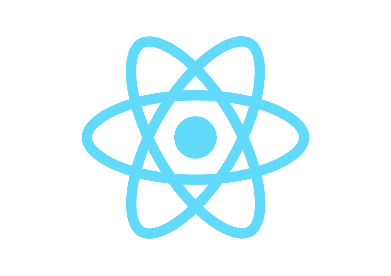


Figure 7 : React

**Efficacité du développement**

Moins de temps de développement et une meilleure qualité

React nous permet d'écrire un code modulaire et propre, en décomposant le projet en composants séparés. Cela permet également de réutiliser le code, ce qui peut vraiment stimuler la productivité des développeurs.

**Flexible et facile à entretenir**

Contrairement aux grands projets monolithiques, le code React est facile à maintenir et à mettre à jour grâce à sa structure modulaire. Les produits React sont extrêmement flexibles et peuvent être facilement mis à l'échelle. Cela peut permettre de gagner beaucoup de temps et d'argent à long terme.

**Haute performance de votre application**

Conçues dans un souci de performance, les fonctionnalités de base de la bibliothèque, telles que le DOM virtuel et le rendu côté serveur, permettent de créer des applications à grande échelle très rapidement.

**Optimisation des moteurs de recherche - plus de visibilité pour votre site**

Grâce au rendu du backend, les projets React sont mieux optimisés pour les moteurs de recherche. En outre, la vitesse et les performances élevées, que nous avons énumérées ci-dessus, contribuent à un meilleur référencement et offrent également une meilleure expérience globale à l'utilisateur.

**Un ensemble d'outils ouvert et riche**

Étant une technologie à source ouverte, React est gratuit et ouvert à tous. De plus, elle dispose d'un riche écosystème, comprenant des outils incontournables tels que Flux ou Redux ainsi que Node.js sur le backend ou encore un grand nombre de bibliothèques open source spécifiques aux tâches, répertoriées sur npm.

**Une communauté forte**

Construite initialement sur Facebook, la bibliothèque est toujours soutenue et entretenue par la société ainsi que par plus de 1 000 contributeurs indépendants dans le monde entier. Grâce à la communauté dynamique et au solide soutien de l'entreprise, React est une technologie fiable et moderne.

**Transition facile vers React Native**

React Native est un cadre de développement mobile, basé sur React. Ainsi, il est facile pour les développeurs de React JS de passer à React Native et de créer des applications mobiles avec une apparence et une compatibilité native. De plus, certaines parties de votre application web React peuvent être réutilisées dans une application mobile React Native, ce qui améliore la vitesse de développement.

### Pourquoi MERN ?



Une pile est la combinaison des technologies utilisées pour créer une application web. Toute application web sera réalisée en utilisant plusieurs technologies (Framework, bibliothèques, bases de données, etc.).

La pile MERN est une pile JavaScript conçue pour rendre le processus de développement plus fluide. MERN comprend quatre composants open-source : MongoDB, Express, React et Node.js. Ces composants fournissent une approche end-to-end pour les développeurs.

Le principal avantage pour les développeurs qui utilisent la pile MERN est que chaque ligne de code est écrite en JavaScript. C'est un langage de programmation qui est utilisé partout, tant pour le code côté client que pour le code côté serveur. Avec un seul langage pour tous les niveaux, il n'est pas nécessaire de changer de contexte.

Pour les piles techniques comportant plusieurs langages de programmation, les développeurs doivent trouver le moyen de les interfacer ensemble. Avec la pile JavaScript, les développeurs n'ont besoin de maîtriser que JavaScript et JSON.

Dans l'ensemble, l'utilisation de la pile MERN permet aux développeurs de créer des applications web très efficaces et maintenables.

### Pourquoi SASS ?



Comme vous le savez, SASS est un préprocesseur CSS c’est-à-dire qu’il permet de générer dynamiquement des fichiers CSS et a pour objectif d'améliorer l'écriture de ces fichiers, en apportant plus de flexibilité.

Cet outil nous apporte beaucoup à notre manière d’écrire du CSS en nous facilitant la stylisation à l’aide de ces différentes fonctionnalités je cite :

**Les variables**  
Cela nous servira plus quand nous voudront respecter la charte graphique en les stockant dans nos variables, ce qui facilitera la tâche quand nous voudrions les appeler, pareil pour les polices d’écriture.

**Les mixins**  
Les mixins sont des fragments de CSS qui peuvent s’utiliser comme des fonctions.

**Une facilité d’écriture, de lecture, et un code optimisé**L’écriture avec sass est mieux organisée, plus lisible, moins répétitive. Ce qui facilitera la lecture de notre code.

### Pourquoi utiliser un Framework ?

**Une organisation à notre projet**

Se basant sur le modèle MVC (Model – View – Controller), cela clarifie notre code et le rend plus logique.

Le modèle va s’occuper de récupérer les données qu’il fournira au contrôleur, ce dernier va exécuter un certain nombre d’opérations et faire afficher le résultat sur la vue.

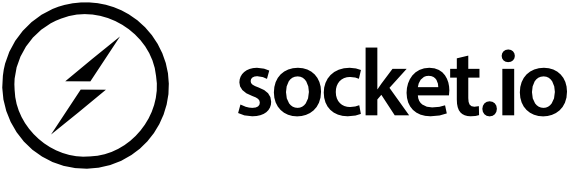
**Des composants et bibliothèques réutilisables**

Grâce aux Framework nous pourront faire appel à un certain nombre de packages contenant des fonctions prédéfinies nous permettant d’éviter d’écrire plus de lignes de code, et d’utiliser des fonctionnalités qui nous permettront d’avancer plus rapidement dans la conception de notre projet.

**Une base régulièrement mise à jour**

En choisissant un Framework, on choisit également une communauté active qui détectera et corrigera des failles ou des manques du Framework. En quelque sorte, on mutualise les moyens de développements. Vous profiterez ainsi des mises à jour du Framework avec le lot d’améliorations qu’elles comportent.

### Socket IO



Socket.IO permet une communication bidirectionnelle entre le client et le serveur. Les communications bidirectionnelles sont activées lorsqu'un client dispose de Socket.IO dans son navigateur et qu'un serveur a également intégré le progiciel Socket.IO. Si les données peuvent être envoyées sous plusieurs formes, JSON est la plus simple.

Pour établir la connexion, et pour échanger des données entre le client et le serveur, Socket.IO utilise Engine.IO. Il s'agit d'une implémentation de niveau inférieur utilisée sous le capot. Engine.IO est utilisé pour l'implémentation du serveur et Engine.IO-client est utilisé pour le client.

Socket.IO fait penser aux Web Sockets. Les Web Sockets sont également une implémentation de navigateur permettant une communication bidirectionnelle, mais Socket.IO ne l'utilise pas comme norme. Tout d'abord, Socket.IO crée une connexion de sondage longue durée en utilisant le xhr-polling. Ensuite, une fois cette connexion établie, il passe à la meilleure méthode de connexion disponible. Dans la plupart des cas, il en résultera une connexion Web Socket.

### NodeJS



Node.js a été initialement construit pour Google Chrome, et plus tard en open-source par Google en 2008. Il est construit sur le moteur JavaScript V8 de Chrome. Il est conçu pour construire des applications réseau évolutives et peut exécuter du code JavaScript en dehors d'un navigateur.

Node.js fonctionne sans page HTML, mais utilise son propre système de modules basé sur CommonJS, pour assembler plusieurs fichiers JavaScript.

### NPM



npm est l'abréviation de node package manager. Il permet une gestion transparente des paquets de node.js. Avec cet outil nous pouvons installer, partager et gérer les paquets node.js, dans notre projet il nous permettra d’utiliser des librairies déjà prêtes de composants nous facilitant le développement et l’intégration de certaines fonctionnalités.

### ExpressJS



ExpressJS est un Framework d'application web pour Node.js, un autre composant de MERN. Au lieu d'écrire le code complet du serveur web à la main sur Node.js directement, les développeurs utilisent Express pour simplifier la tâche d'écriture du code du serveur. Il n'est pas nécessaire de répéter le même code encore et encore, comme vous pourriez le faire avec le module HTTP de Node.js.

Le Framework Express est conçu pour construire des applications web et des API robustes. Il est connu pour sa rapidité et sa structure minimaliste, avec de nombreuses fonctionnalités disponibles sous forme de plugins.

### Pourquoi MongoDB ?



MongoDB est une base de données NoSQL (non relationnelle).

Alors que les bases de données relationnelles conventionnelles ont une conception de schéma typique basée sur des colonnes et des tables, MongoDB est sans schéma. Les données sont stockées dans des documents flexibles avec un langage de requête basé sur JSON (JavaScript Object Notation). Le contenu, la taille et le nombre de champs dans les documents peuvent varier d'un document à l'autre. Cela signifie que la structure des données doit être modifiée au fil du temps.

MongoDB est connu pour être flexible et facile à mettre à l'échelle.

### Pourquoi Redux ?



Figure 8: logo redux

Redux est un outil de gestion de l'État. Bien qu'il soit principalement utilisé avec React, il peut être utilisé avec n'importe quel autre Framework ou bibliothèque JavaScript. Il est léger (2 Ko, dépendances comprises).

Avec Redux, l'état de notre application est conservé dans un stockage, et chaque composant peut accéder à n'importe quel état dont il a besoin à partir de ce stockage.

1. Le central store, tout composant peut accéder à n'importe quel état depuis le store, il n'y a pas besoin de passer des props dans les deux sens.
2. Une autre façon de voir le store centralisé, il persiste l'état d'un composant même après que celui-ci ait été démonté.
3. Cela évite les rendus inutiles, car lorsque l'état change, il renvoie un nouvel état qui utilise une copie peu profonde.
4. Les tests seront faciles car l'interface utilisateur et la gestion des données sont séparées.
5. L'historique de l'état est conservé, ce qui permet d'implémenter très facilement des fonctionnalités telles que l'annulation.

## Initialisation de l’environnement de travail :

### Familiarisation avec GitHub :



Figure 9: logo github

GitHub est un outil de gestion de version, il permet donc que plusieurs développeurs travaillent sur le même projet, sur les mêmes fichiers sans qu’il n’y ait de conflit par rapport aux modifications apportées au projet.

Pour la structure du projet, 3 répertoires GIT ont été créés :

YAchat-frontend : qui contiendra la plateforme web.

YAchat-mobile : qui comptera l’application mobile.

YAchat-backend : qui comptera l’api assurant la liaison entre les interfaces et les opérations serveur.

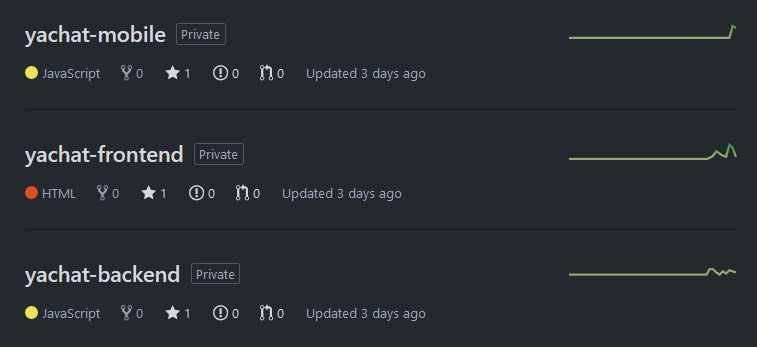


Figure 10: repos GitHub

### Création du projet mobile :

Avant d’installer le projet React Native (qui est en réalité un boiler plate) il faut d’abord créer un dossier en local, ce n’est pas une nécessité mais c’est une bonne pratique afin de garder son bureau bien rangé.

L’installation s’est faite à l’aide de npm :



Figure 11: création projet react-native

### Utilisation de l’outil GIT :

Une fois notre projet créé, nous allons effectuer un cloner notre projet.

Afin de relier le fichier que nous avons créé au répertoire créé sur GitHub comme vous pouvez le voir ci-dessous :

Figure : cloner le projet git

Mais dans notre cas, étant donné que le maître de stage a pris soin de créer le répertoire du projet nous allons nous contenter d’un pull :

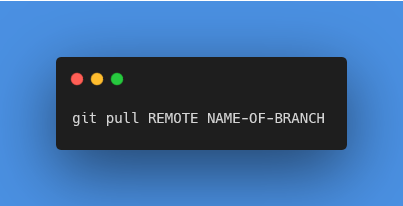


Figure 13: synchroniser le dépôt local

Une fois le projet récupéré, il faudra fréquemment le mettre à jour afin de conserver les modifications apportées en cas de perte de fichiers en local ou d’accident.

Avant d’envoyer nos données, il faudra les lister afin de savoir ce que nous avons comme changements opérés, le point « . » servira à lister tous les changements en une seule opération.

Ensuite il faudra envoyer les fichiers dans notre dépôt avec un message (D’où l’utilisation du paramètre -m) :

Ensuite il faudra envoyer nos informations dans notre dépôt avec la commande push :



Figure 14: enregistrer les changements et les pousser sur le dépôt GIT

## Base de données :

### Modélisation :



Figure 15: logo draw.io

Pour la modélisation de notre base de données nous avons utilisé la solution Draw.IO, un outil qui nous permettra de créer un diagramme nous permettant de définir le comportement des données recueillies.

Voici un schéma de notre modèle de base de données :

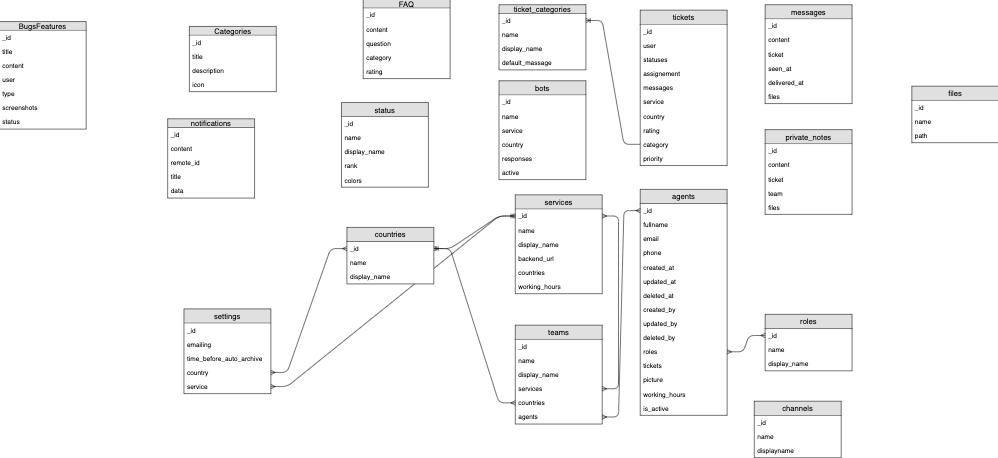


Figure 16: schéma de la base de données

* Utilisant une base de données non relationnelle, le principe du raisonnement est de type objet.
* Chaque objet représente un élément, l’équivalent d’une entité, possédant des attributs.

### Création des Schema :



Figure 17: logo mongoose

Un ODM est un modèle de données d'objet, utilisant le mappage de documents d'objet. Il présente les données de l'application sous forme d'objets Javascript qui sont ensuite analysés et mis en correspondance avec la base de données.

Mongoose & MongoDB vont de pair et sont le choix incontournable de la communauté Node, car le système de stockage de documents et de requêtes est, à première vue, presque exactement comme JSON.

Dans MongoDB, une base de données NoSQL, nous utilisons la modélisation des données orientée vers les documents. Nous créons un modèle, un squelette de ce que chaque création doit contenir et ressembler. Nous appelons la création d'un modèle un document et un groupe de documents une collection. Il est similaire aux rangées de tables pour une base de données relationnelle.

Pour créer un schéma on utilise généralement ce type de modèle :

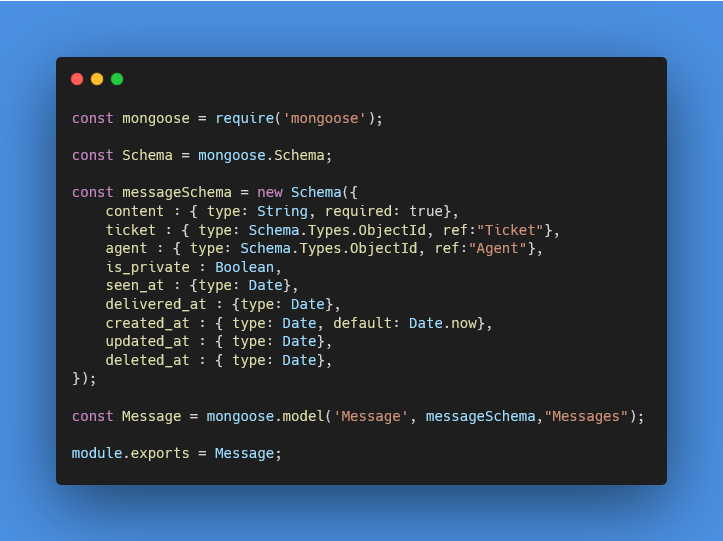


Figure 18: Modèle de l'objet message

### Les seeds:

Un seeder permet d’ajouter une donnée dans la base de données sans qu’il y ai un traitement sur la vue mais directement grâce à une requête http effectuée par un client comme Insomnia, n’ayant pas les données des utilisateurs et n’ayant pas besoin d’instaurer un système d’utilisateurs car les données seront reçues directement via les serveurs de Yassir, par exemple pour générer le token d’un utilisateurs (Nous en reparlerons plus tard dans le rapport) il nous suffit simplement d’envoyer une requête au lieu dédié à la génération de token pour l’authentification prévu à cet effet :

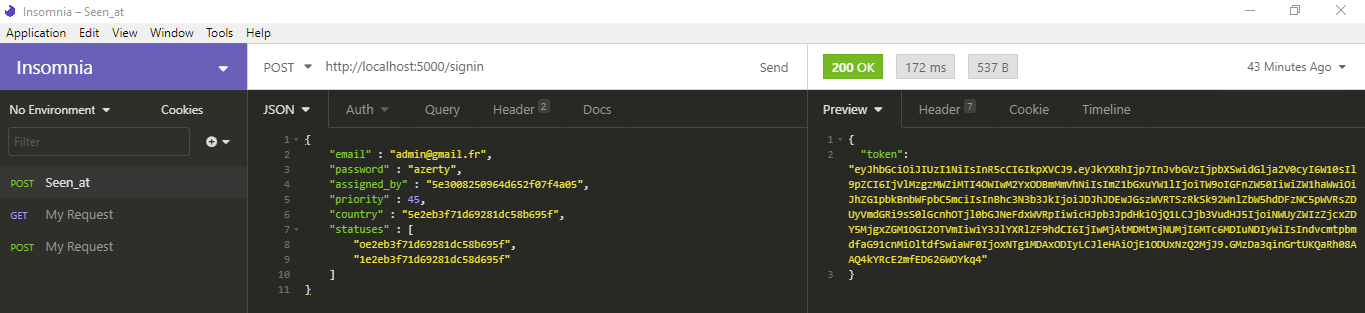


Figure 19: seed de données à l'aide d'Insomnia

## Création de l’API :

La création de l’API se fait à l’aide d’Express, chaque objet possède un Schema qui permettra de générer un objet dans notre base de données non-relationnelle, mais pour contrôler cet objet il nous faudra un Modèle, c’est ce qui nous permettra de mettre en relation certains attributs d’objets qui les lient à d’autres.

### Création des Model :

Pour créer un model il faut suivre un schéma prédéfini :

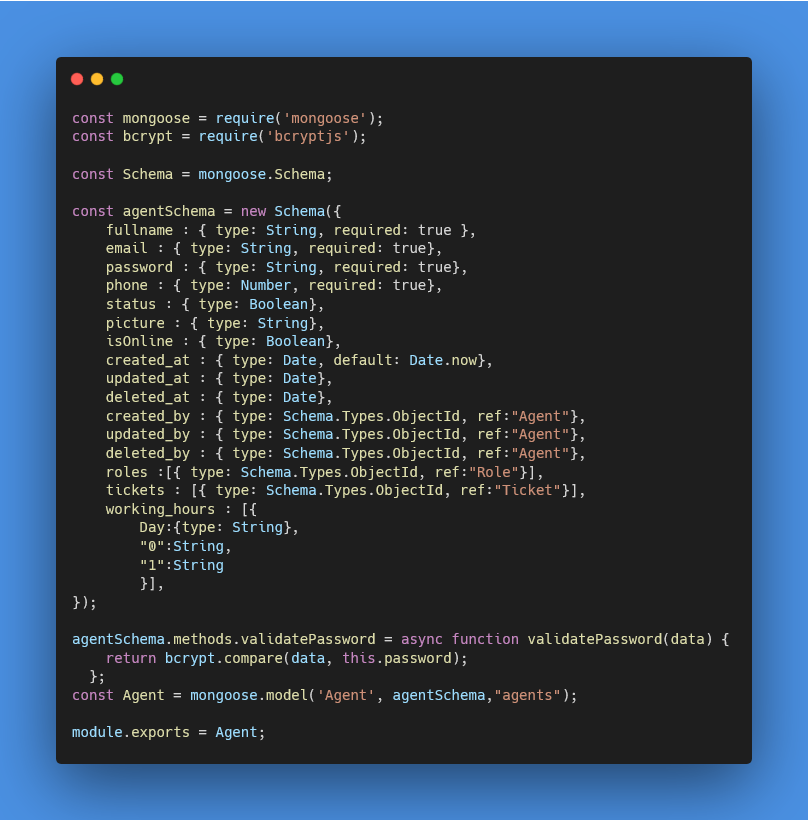


Figure 20: modèle de l'objet agent

Dans ce modèle on peut constater que notre objet « Agent » a plusieurs attributs, mais aussi que nous utilisons une fonction de cryptage pour crypter les identifiants de l’agent pour des raisons de sécurité.

## Controller :

### Le routage :

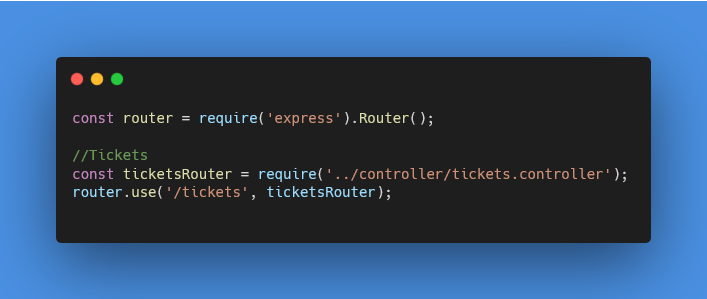
Une route c’est comme un chemin qu’on donne à une URL qu’on crée, en lui indiquant ce qu’il effectuera une fois que cette URL est utilisée, par exemple dans notre cas, nous allons y appeler un controller.. 

Figure 21: route menant vers les fonctions de l'objet ticket

### Création des controller :

La tâche d'un contrôleur est de réceptionner une requête (qui a déjà été triée par une route) et de définir la réponse appropriée à celle-ci.



Figure 22: controller de l'objet agent

Comme vous pouvez le voir, un controller contient les opérations effectuées concernant l’objet actuel traité qui est l’agent (par exemple l’authentification mais nous y reviendrons plus tard).

### La méthodologie utilisée pour la gestion :

Pour la gestion de ce projet, nous avons utilisé la méthode Scrum, c’est une méthodologie ou un cadre de gestion de projet agile utilisé principalement pour les projets de développement de logiciels dans le but de fournir de nouvelles capacités logicielles toutes les 2 à 4 semaines, 2 semaines dans notre cas, qui consiste à lancer des sprints qui sont des rush de 2 semaines où des tâches sont assignées aux membre de l’équipe comprenant une réunion pour déterminer les tâches, des daily meeting (une petite réunion chaque matin pour déterminer l’avancement des tâches)

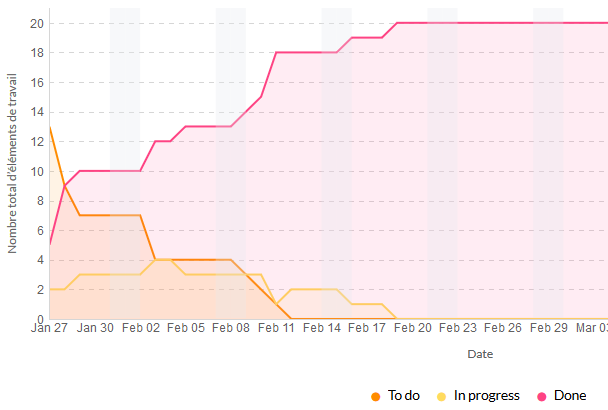


Figure 23: statistiques de tâches du premier sprint

Mais aussi un tableau kanban qui est un outil de travail permettant d'appliquer la méthode kanban à un flux de travail. Les tableaux kanban utilisent des fiches kanban pour visualiser l'évolution des unités de travail dans un processus de production, qui permettra d’assigner des tâches précises à un ou plusieurs membres de l’équipe de développement

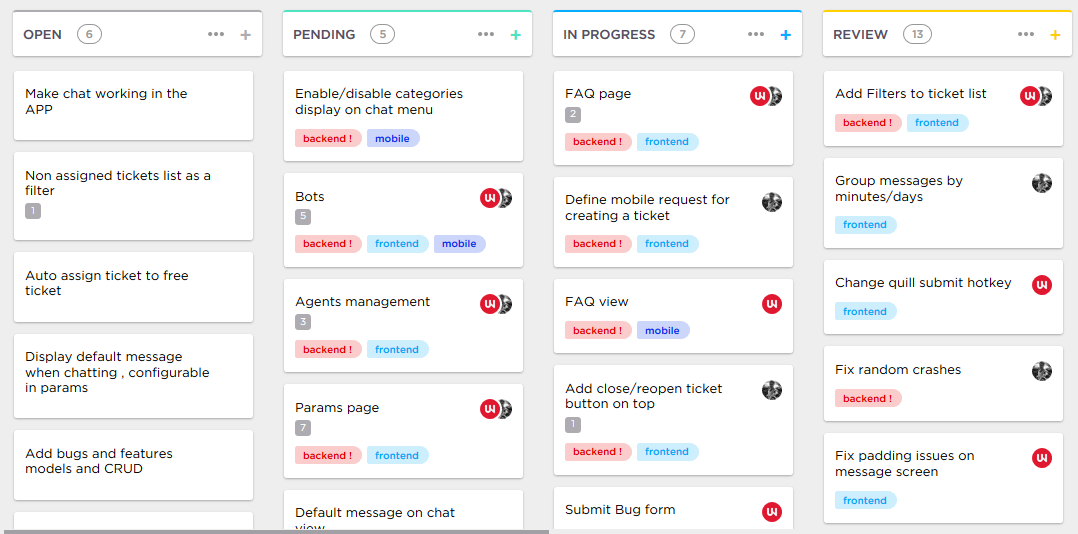


Figure 24: Kanban des tâches du deuxième sprint

## Fonctionnalités :

### Mobile :

L’application mobile est la partie client, elle consiste à permettre au client de dialoguer avec les agents du support en créant un ticket, il sera ensuite assigné à un agent qui pourra lui répondre depuis la plateforme, pour cette partie nous avons utilisé React Native, une façon de développer des applications compatibles multi OS React Native est comme React, mais il utilise des composants natifs au lieu de composants web comme éléments de base. Pour comprendre la structure de base d'une application React Native, nous devons donc comprendre certains des concepts de base de React, comme JSX, les composants, l'état et les accessoires. Si vous connaissez déjà React, vous devez encore apprendre quelques trucs spécifiques à React-Native, comme les composants natifs.

### **Conception :**

La conception de l’application mobile a été faite par moi-même étant donné que la Project Manager n’aimait pas le design proposé par l’équipe de conception, elle a été faite à l’aide d’Adobe XD (XD pour Experience Design)



Figure 36: Logo Adobe XD

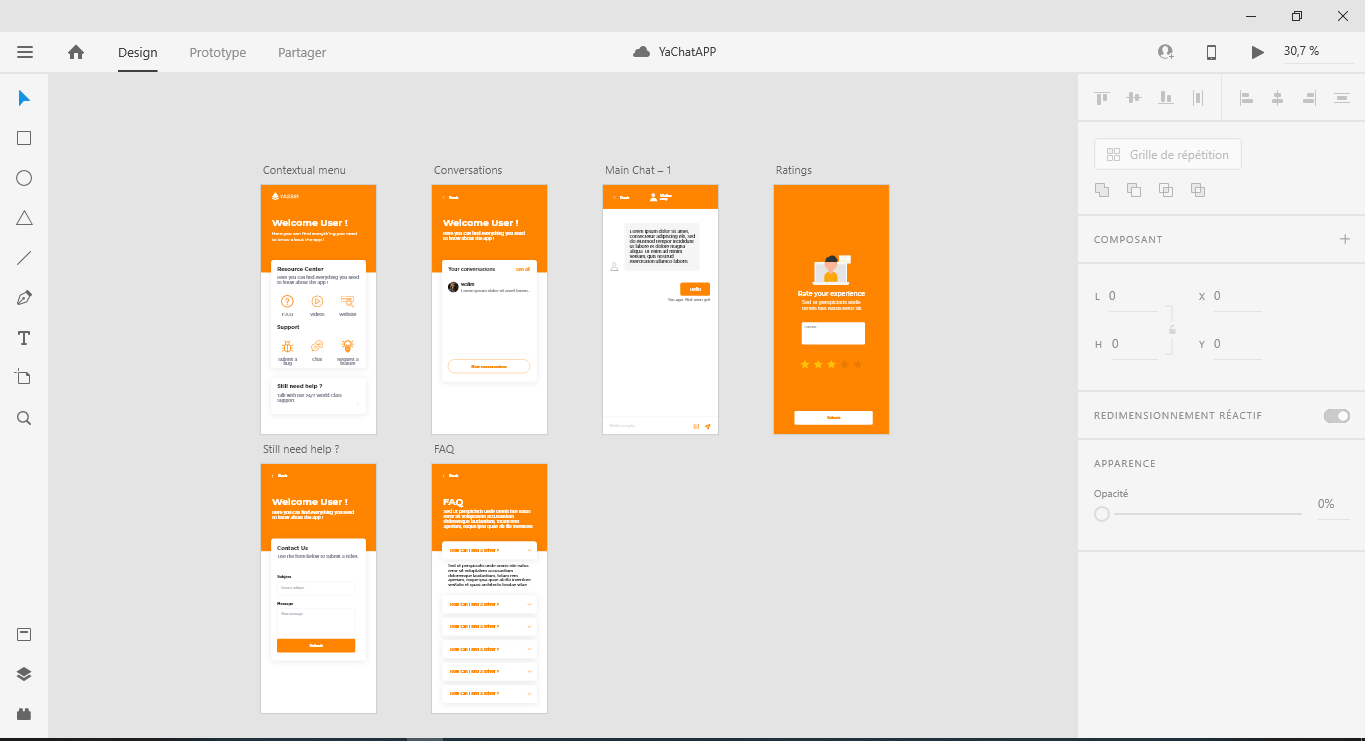


Figure 37: Conception mobile

### **Intégration des API (mobile) :**

Après la conception du module mobile je suis passée à l’étape de l’intégration des API’s qui vont permettre la communication entre le module et le serveur.

Les communications se feront grâce à AXIOS : une bibliothèque JavaScript fonctionnant comme un client HTTP. Elle permet de communiquer avec des API en utilisant des requêtes de tous types (POST, GET…).

Le module mobile sera intégré sur des service des établis de Ya technologies tell que Yassir cela veut dire que chaque utilisateur de ces services dispose d’un ID unique qui si envoyés à notre server pourra l’identifier et récupère des informations spécifiques à lui :



### **Menu contextuel :**

Ce menu assure la liaison vers toutes les pages de l’application mobile.

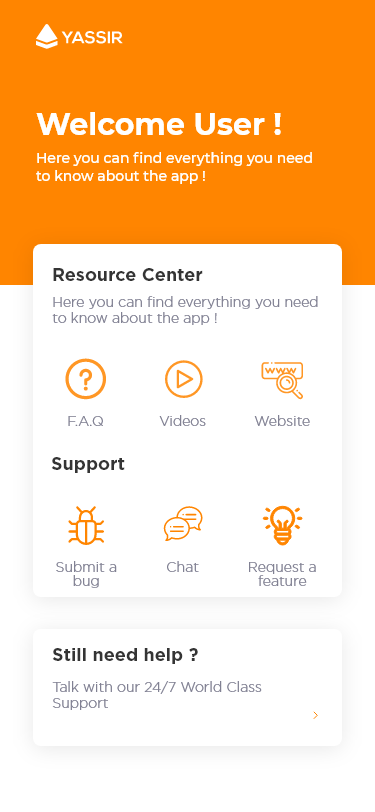


Figure 38: menu de l'application

La navigation se fait à l’aide de React Native Navigation qui est composé de quelques utilitaires de base et ceux-ci sont ensuite utilisés par les navigateurs pour créer la structure de navigation dans notre application :



Figure 39: navigation entre écrans

« Navigation Container » représente le container qui va englober tous nos éléments concernés par la navigation.

L’élément « Stack Navigator » permet d’établir une pile d’écrans avec des attributs spécifiques qui nous permettrons de naviguer entre les fenêtres à l’aide de la méthode navigate :

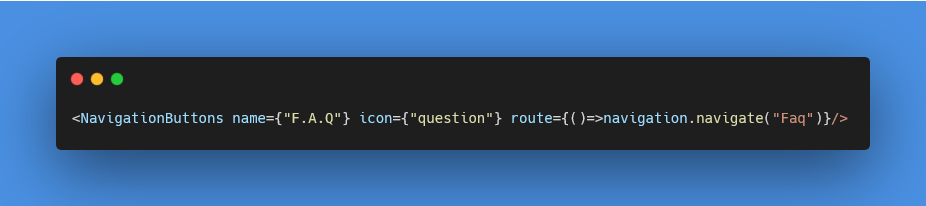


Figure 40: fonction pour la navigation

### **Formulaires :**

Pour les formulaires nous avons opter pour un seul composant permettant d’envoyer le même type de données mais dans différents champs.

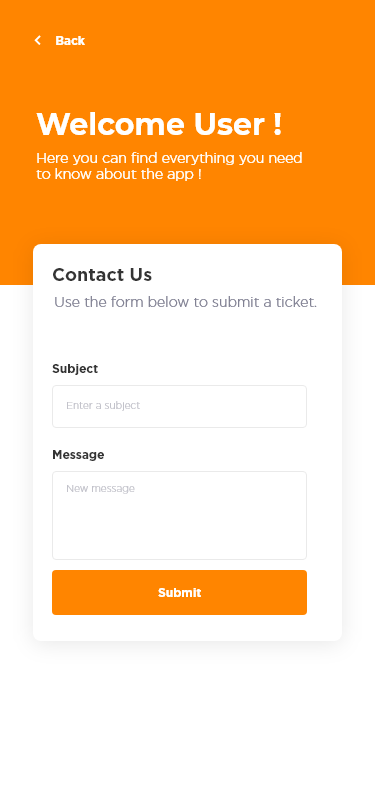


Figure 41: composant du formulaire

**Côté serveur :**

Ces deux interfaces disposent de d’une logique de fonctionnement semblable, une fois les champs remplis une requête POST est envoyée vers les API concernées contenant les données insérées par le client ainsi que son ID afin que le serveur puisse vérifier et grader trace de lui. Leurs demandes seront traitées par des agents ensuite.

### **Foire aux questions (côté mobile) :**

La rubrique à l’aide d’un accordéon va nous permettre d’afficher les réponses cachées sous les cartes de questions à l’aide d’un bouton qui fera effet de toggle :

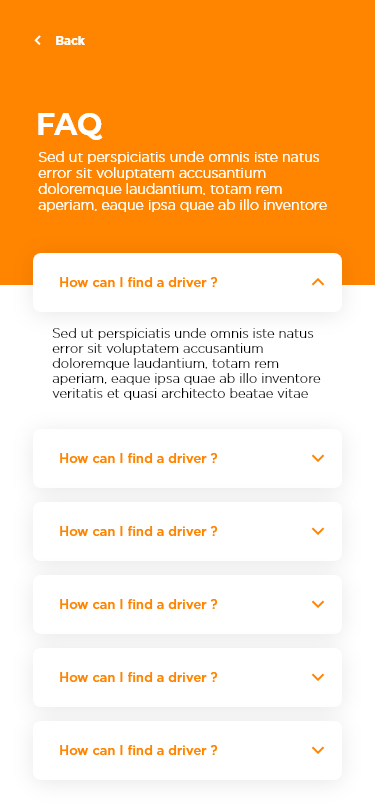


Figure 42: foire aux questions côté mobile

### **Conversations :**

L’onglet conversation nous permettra de répertorier les tickets ouverts par l’utilisateur qu’il pourra consulter une fois fermés.

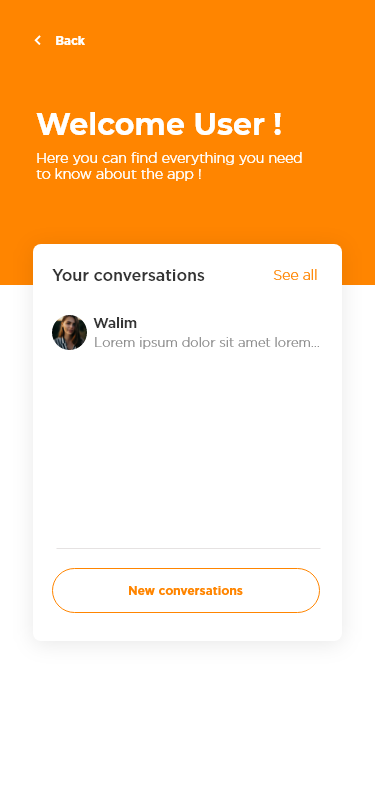


Figure 43: écran de conversations

Parmi les fonctionnalités disponibles sur cette page, la création d’un ticket, car nous partons dans une optique où dès que le client enverra le premier message un ticket se créera et sera assigné à un agent.

**Côté serveur :**

En utilisant Axios on envoie une requête à L’api responsable du renvoie des tickets en fournissant l’ID de l’utilisateur « client », le serveur vérifie la validité des informations et renvoie les tickets du client qui sont affiches comme sur l’illustration si dessous

### **L’application chat :**

L’application de chat permet au client de dialoguer avec un opérateur du call center, au premier message envoyé il recevra un message automatique du robot programmé depuis la plateforme, il pourra envoyer un message ou une photo qui pourra illustrer le problème qu’il encourt et depuis la plateforme l’opérateur aura la possibilité de lui répondre.

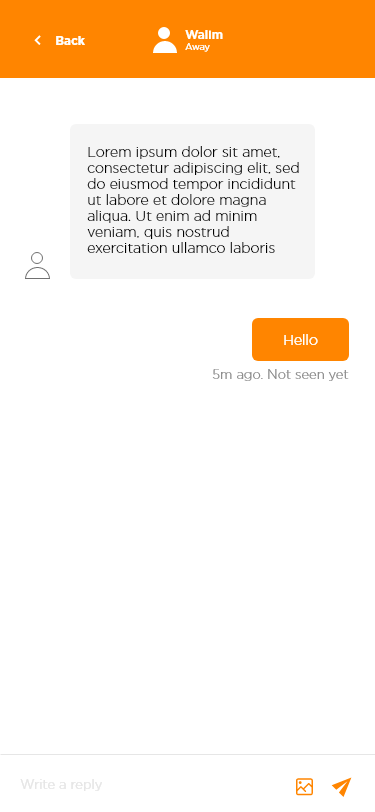


Figure 44: écran de chat

**Côté serveur :**

C’est la fonctionnalité principale du module elle permet la communication entre les agents et les clients, lors du choix d’un ticket à afficher une requête initiale est envoyée afin de récupérer la conversation qui est ensuite affichée, à chaque clique sur le bouton d’envoi une autre requête est envoyée afin d’ajouter d’insérer le message dans la conversation, la requête contient le message saisi par le l’utilisateur ainsi que son ID de confirmer son identité sur le serveur.



### **Système de notation :**

Une fois que le ticket sera clos, l’utilisateur aura la possibilité de noter la prestation de l’opérateur qui s’est occupé de lui afin d’établir une moyenne et déterminer la note finale qu’aura l’opérateur, elle sera cruciale pour déterminer son efficacité au sein de l’équipe du call center.

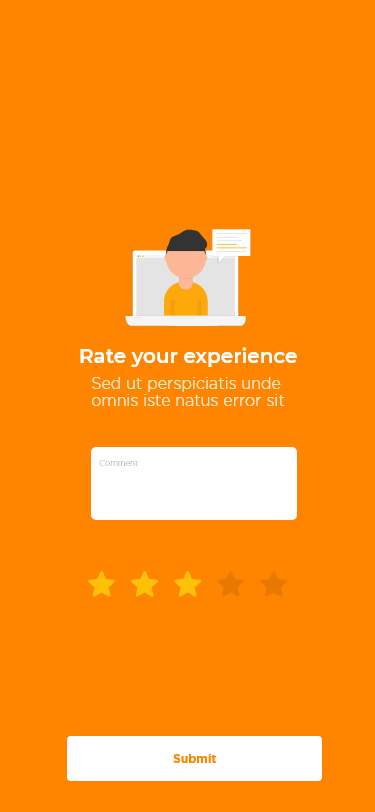
****

Figure 45: page de notation

# English Summary:

**Introduction:**

I’m a student in first year of engineering cycle in an IT engineering school at CESI Engineering School, I did this internship as part of my formation in a big startup company named YA Technologies (Yassir).

This internship allowed me to discover a little more the several web and mobile architecture technologies by learning them with practicing.

During this internship, my main mission was to learn some JavaScript technologies and make them in practice.

**Summary of my internship work:**

In order to run an application, it needs a source from which to extract data to display which is a given database, in order to design it is necessary to create a schema that will identify as needed the objects we will need as well as the attributes that will allow us to establish links between our data.

From this schema we will be able to create what we call "Schema" that will allow us to create our database architecture, in order to create it we will need a first sending of data, the controller will allow the API to perform various operations that we will need on our platform and the mobile application.

Then will come our web platform made up of various components with instantaneous loading, it will allow the support agents to perform various operations to meet the needs of the user and thus control the call center.

As for the mobile application, it will establish a direct link between users (driver or customers) and operators so that they can be taken care of as soon as possible and stay informed using the various links present and the robot assistant.

**Conclusion**

These three months internship allowed me to learn more about a lot of new technologies that are in demand on the market while my main mission was to create a solution to put the customer in touch with technical support.

# Résumé:

**Introduction:**

Je suis étudiant en première année de cycle d'ingénieur dans une école d'ingénieurs en informatique à CESI Ecole d’Ingénieurs, j'ai fait ce stage dans le cadre de ma formation dans une grande startup nommée YA Technologies (Yassir).

Ce stage m'a permis de découvrir un peu plus les différentes technologies d'architecture web et mobile en les apprenant par la pratique.

Pendant ce stage, ma principale mission a été d'apprendre quelques technologies JavaScript et de les mettre en pratique.

**Résumé de mon travail durant le stage :**

Pour faire fonctionner une application, il lui faut une source où soutirer des données à afficher qui est une base donnée, afin de la concevoir il est nécessaire de créer un schéma qui identifiera selon le besoin les objets dont nous auront besoin ainsi que les attributs nous permettant d’établir des liaisons entre nos données.

À partir de ce schéma nous pourront créer ce que l’on appelle des « Schema » qui nous permettrons de créer notre architecture de base de données, afin qu’elle soit créée il faudra un premier envoi de données, le contrôlleur quant à lui permettra à l’API d’effectuer divers opérations dont nous aurons besoin sur notre plateforme et l’application mobile.

Puis viendra notre plateforme web constituée de divers composants à chargement instantané, elle permettra aux agents du support d’effectuer diverses opérations afin de répondre aux besoins de l’utilisateur et ainsi contrôler le call center.

Quant à l’application mobile, elle établira un lien direct entre utilisateurs (chauffeur ou clients) et les opérateurs afin qu’ils soient pris en charge dans les meilleurs délais, et rester informer à l’aide des différents liens présents et l’assistant robot.

**Conclusion**

Ces trois mois de stage m'ont permis d'en apprendre plus sur un grand nombre de nouvelles technologies qui sont demandées sur le marché alors que ma mission principale était de créer une solution pour mettre le client en contact avec le support technique.

# Bilan du stage :

Après environ trois mois de stage, je peux en conclure que j’ai pu apprendre beaucoup de choses dont des technologies dont nous n’avons pas encore évoqué le sujet au CESI, mais qui dans le futur pourront m’aider dans ma carrière professionnelle.

# Conclusion :

Ce stage de courte durée m’a été bénéfique pour mon apprentissage, il m’a permis de découvrir le travail dans au sein d’une équipe de développeurs et les bonnes pratiques à adopter.

Je peux dire que ce stage m’a appris à développer de façon plus professionnelle et aux standards des grandes entreprises.

# Ressources :

* <https://www.linguee.fr/>
* <https://react.semantic-ui.com/>
* <https://fr.reactjs.org/docs/getting-started.html>
* <https://reactnative.dev/>
* <https://www.npmjs.com/>
* <https://medium.com/>
* <https://software-mansion.github.io/react-native-gesture-handler/docs/getting-started.html>
* <https://reactnavigation.org/>
* <https://www.mongodb.com/fr>
* https://www.adobe.com/mena\_en/products/xd.html