OC Pizza

IT CONSULTING
& Development

OC Pizza

Système de gestion de pizzérias

Dossier de conception technique

Version 1.0

Auteur
Vincent Caronnet
Software Engineer

Table des matières

1 - VERSIONS DU DOCUMENT

Auteur	Date	Description	Version
Vincent Caronnet	29/05/2021	Création du document	1.0

2 - INTRODUCTION

2.1 - Objet du document

Le présent document constitue le dossier de conception technique du système de gestion de pizzérias développé par IT Consulting & Development pour OC Pizza découlant :

1. des besoins d'OC Pizza recueillis le 3/05/2021 par IT Consulting & Development.
2. et du Dossier de conception fonctionnelle produit suite à ce recueil des besoins.

Dans le cadre des méthodes Agile utilisées pour le développement, la mise en production et la maintenance du système, ce document devra être mis à jour en fonction de l'évolution du projet, de ses nouveaux enjeux et des nouvelles solutions qui seront mises en œuvre dans le futur.

Chaque mise à jour de ce document par le prestataire devra être à nouveau vérifiée et approuvée par le client.

Les versions successives de ce document seront conservées et versionnées dans ce dépôt GitHub : https://github.com/centvingt/OCPizzaRedaction.

2.2 - Références

Pour de plus amples informations, se référer également aux éléments suivants :

1. Dossier de conception fonctionnelle du système
2. Dossier d'exploitation du système

3 - Architecture Technique

3.1 - Backend

Le backend se limite à une simple API GraphQL développée en Node.js avec le *framework* Express.

Elle traite peu de logique métier pour limiter les temps de réponse du serveur tout en réduisant les ressources nécessaires au serveur pour exécuter cette API.

Le SGBD utilisé est PostgreSQL, un outil libre et gratuit sous licence BSD.

Certains composants de l'API sont couplés à des *web sockets* pour transmettre au client des données en temps réel lors de leur mise à jour dans la base de données.

3.2 - Frontend

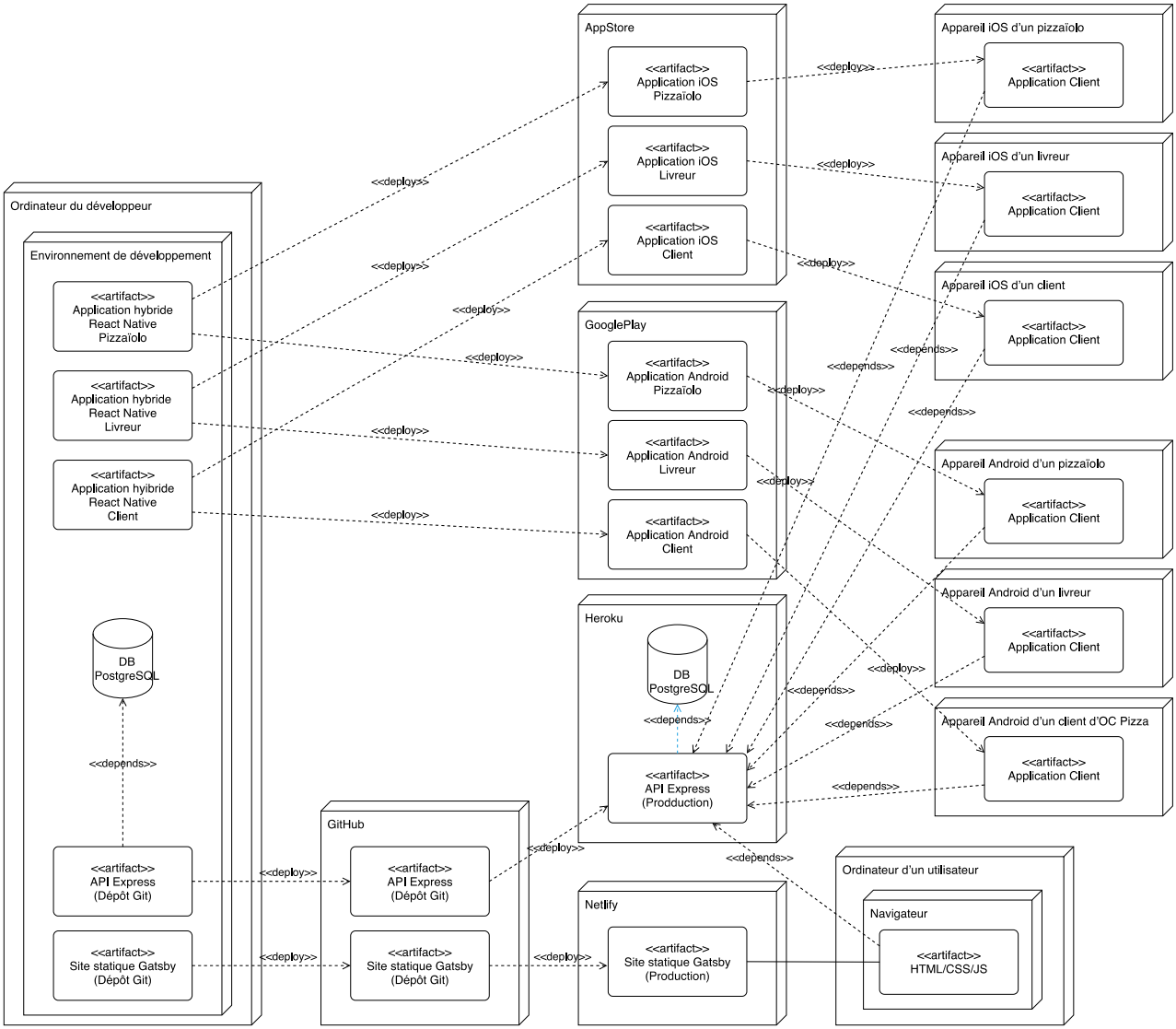
Les applications clientes (site internet et applications mobiles et tablettes pour le client, le livreur et le pizzaïolo) traitent elles-mêmes la majeure partie de la logique métier.

Elles s'appuient sur des requêtes GraphQL de l'API dont elles consomment les données pour construires les interfaces utilisateurs qu'elles affichent. Ainsi, la plupart du travail de l'application se fait du côté du client, l'API se contentant de lui fournir les données dont il a besoin.

Les applications clientes seront développées en React avec le *framework* Gatsby pour le site internet et en React Native pour les applications mobiles et tablettes.

4 - Architecture de Déploiement

Les applications web et mobiles sont déployées en continu pour fournir aux utilisateurs finaux des services augmentés de nouvelles fonctionnalités au fur et à mesure de leur développement. Le déploiement continu de l'applications web consistera à *pusher* la branche de production du dépôt Git présent sur l'ordinateur du développeur vers un dépôt sur GitHub auquel les services d'Heroku (pour l'API) et de Netlify (pour le site statique) sont connectés. L'utilisateur final (le client, l'employé, le livreur ou le pizzaïolo) accède ainsi au site statique hébergé par Netlify et mis à jour via GitHub. Ce site consomme l'API Express GraphQL hébergé par Heroku et lui aussi mis à jour via GitHub. Les trois applications mobiles (pour le client, le livreur et le pizzaïolo) sont chacune développées en React Native à la fois pour iOS et Android. Elles sont déployées sur l'App Store et le Google Play, les stores respectifs des appareils iOS et Android.



5 - ARCHITECTURE LOGICIELLE

5.1 - Principes généraux

Les diverses interfaces utilisateurs web et mobiles d'OC Pizza consomment l'API avec des requêtes HTTP pour présenter les données de cette API et lui transmettre les interactions des utilisateurs.

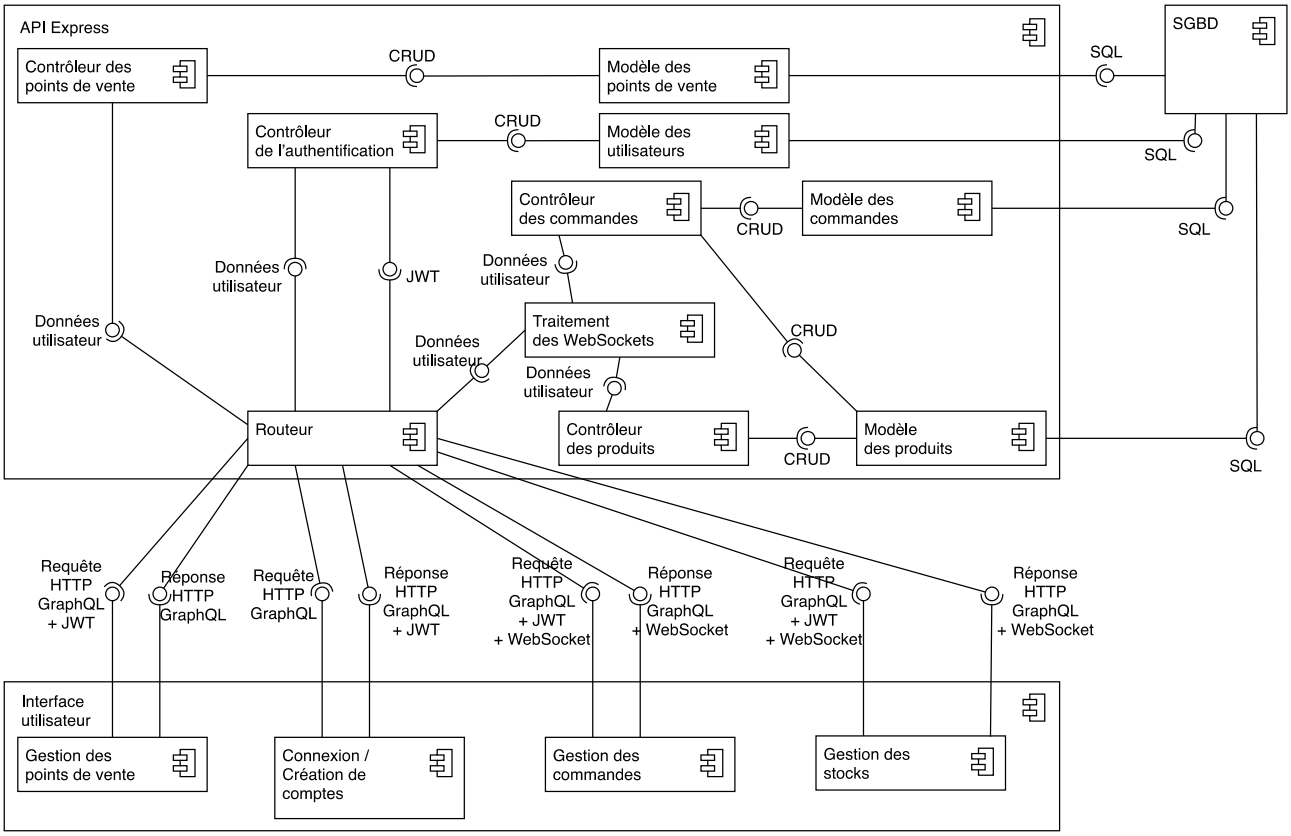
Lors de la connexion d'un utilisateur, l'API fournit au client un JSON *web token* qui lui permet par la suite d'accéder aux ressources protégées de l'API.

Les requêtes HTTP sont traitées ainsi par l'API :

1. le routeur les intercepte,
2. il transmet chaque requête au Contrôleur concerné,
3. et celui-ci utilise le Modèle pour requêter la base de données.

Ensuite, l'API envoie sa réponse au client.

Les requêtes et les réponses sont toutes structurées en GraphQL.



5.2 - API

Les sources et versions de l'API sont :

1. Sauvegardées par Git sur l'ordinateur du développeur
2. Poussées vers un clone Git sur GitHub
3. Déployées sur Heroku

5.2.1 - Les couches

L'architecture applicative est la suivante :

- Le serveur HTTP
- Le routeur
- Les contrôleurs qui traitent les *end ponts* leurs requêtes et leurs réponses
- Les modèles

5.2.2 - Les dépendances

Les dépendances de l'API sont installées et mises à jour avec NPM. Ce sont les Node Modules suivants :

- express v4.17.1 qui sert l'API
- helmet v4.4.1 qui protège l'API
- jsonwebtoken v8.5.1 qui traite les JWT
- sequelize v6.6.2 qui traite les requêtes SQL
- pg v8.6.0 qui permet d'utiliser sequelize avec PostgreSQL
- graphql v15.5.0 qui traite les requêtes et les réponses formulées en GraphQL
- socket.io v4.1.2 qui traite les *web sockets*

5.2.3 - Structure des sources

La structuration des répertoires du projet permet de séparer le routeur des contrôleurs et des modèles.

5.3 - Application Web

L'application web est un site statique réalisé avec le *framework* Gatsby. Ses sources sont :

1. Sauvegardées par Git sur l'ordinateur du développeur
2. Poussées vers un clone Git sur GitHub
3. Déployées sur Netlify

5.3.1 - Les couches

L'architecture applicative est la suivante :

- Une *single page application* (SPA)
- Les composants React chargés par cette SPA

5.3.2 - Les dépendances

Les dépendances de l'application web sont installées et mises à jour avec NPM. Ce sont les Node Modules suivants :

- babel-plugin-styled-components v1.12.0
- gatsby v2.24.66
- gatsby-image v2.4.20
- gatsby-plugin-manifest v2.4.32
- gatsby-plugin-offline v3.2.29
- gatsby-plugin-react-helmet v3.3.12
- gatsby-plugin-styled-components v3.10.0
- gatsby-source-filesystem v2.3.31
- prop-types v15.7.2
- react v16.12.0
- react-dom v16.12.0
- react-helmet v6.1.0
- styled-components v5.2.1
- socket.io v4.1.2

5.3.3 - Structure des sources

Les sources du projet sont regroupées dans le répertoire *src.*

5.4 - Applications mobiles

Les applications mobiles pour le client de la pizzéria, le pizzaïolo et le livreur sont développées avec la librairie React Native pour les *stores* d'iOS et d'Android.

5.4.1 - Les couches

L'architecture applicative de ces applications est la suivante :

- Un routeur qui traite la navigation
- Des composants qui traitent l'affichage des écrans
- Un *store* qui traite les données de l'application pour tous les composants

5.4.2 - Les dépendances

Les dépendances de l'application web sont installées et mises à jour avec NPM. Ce sont les Node Modules suivants :

- @react-native-async-storage/async-storage v1.13.2
- @react-native-community/async-storage v1.12.0
- @react-native-community/masked-view v0.1.10
- @react-navigation/native v5.7.6

- @react-navigation/stack v5.9.3
- @react-navigation/stack v5.9.3
- react v16.13.1
- react-domv 16.13.1
- react-native v0.63.4
- react-native-gesture-handler v1.7.0
- react-native-reanimated v1.13.0
- react-native-safe-area-context v3.1.4
- react-native-screens v2.10.1
- react-native-svg v12.1.0
- react-native-web v0.13.12

5.4.3 - Structure des sources

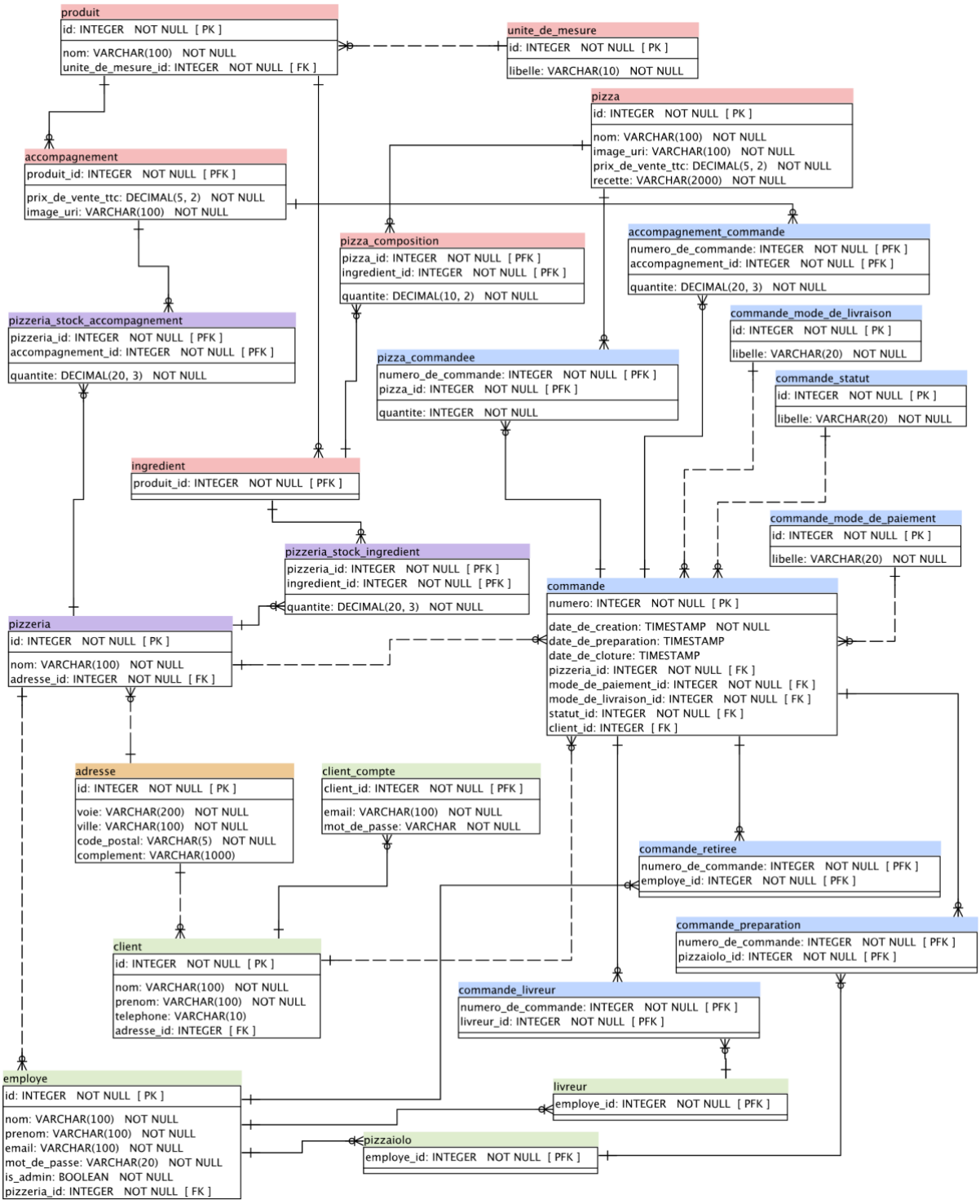
5.5 - Base de données

5.5.1 - SGBD utilisé

Le SGBD utilisé est PostgreSQL, un outil libre et gratuit sous licence BSD. L'ORM Sequelize utilisé par l'API permettrait de migrer vers un autre SGBD SQL si c'était nécessaire.

5.5.2 - Modèle physique de données

Ce modèle physique de données reflète le diagramme de classes de l'application détaillé dans le Dossier de conception fonctionnelle réalisé au préalable. Ce modèle permet de construire la base de données.



6 - POINTS PARTICULIERS

6.1 - Gestion des logs

Les logs de l'API sont consultable dans le *backoffice* du compte Heroku d'OC Pizza, tandis que ceux du site le sont dans celui du compte Netlify d'OC Pizza.

6.2 - Procédure de packaging / livraison

Les branches de développement, de test et de production des applications du projets sont partagées avec le compte GitHub d'OC Pizza.