 MyMuffin

**Dossier d’Architecture Technique**

|  |  |
| --- | --- |
| Révision | 1.0.0 |
| Effectué par | Fofana Moussa |

Architecture applicative

# Objectifs

L’objectif de ce projet est de proposé une solution permettant la création de commande de muffins en ligne, du choix des muffins et de leurs ingrédients au paiement, ainsi que l’envoi au service de livraison local à la ville, le tout se faisant le jour même afin de garder des produits frais et succulent.

Il sera par la suite possible de proposer à des particuliers de créer leur « Marketplace » permettant de mettre en vente leurs propres muffins.

# Acteurs

## Externes

|  |  |  |
| --- | --- | --- |
| Acteur | Description | Localisation |
| Uber Serve | Service proposant des coursiers afin d’effectuer les livraisons | Méru, Lyon 03 |
| Stripe | Permet le règlement en ligne de la commande | Monde |

## Internes

|  |  |  |
| --- | --- | --- |
| Acteur | Description | Localisation |
| MyMuffins Fabric | Ateliers de conceptions des muffins | Esches, Lyon 05 |
| MyMuffins Devs | Équipe de developeur interne | Lyon 03 |

# Contraintes

## Budgétaire

Pas de contraintes budgétaires au moment de l’écriture mais limiter tant que possible les dépenses.

## Planning

Alpha de l’API REST fin janvier 2023, beta fin Mars 2023

Alpha Web view mi-mars 2023 beta fin mai 2023

Alpha application mobile fin juillet beta début novembre 2023

## Urbanisation

Il doit être impossible pour un client en dehors de la zone desservie par notre partenaire Uber Serve sous peine de frais supplémentaire à notre charge.

Une commande ne pourra être modifié que pendant les 5 premières minutes afin de garantir nos délais et limiter le gâchis des matières première.

Chaque MyMuffin factory doit préciser le nombre de commande simultané qu’elle peut gérer, de même pour les particuliers qui souhaiterais ouvrir une marketplace pour éviter leur surcharge et l’attente des clients.

# Exigences

L’API REST doit pourvoir changer de service de paiement avec le minimum de casse possible.

Prévoir en amont l’évolution concernant la Marketplace de particulier dans le design de l’API REST.

Il doit être possible de pouvoir communiqué avec plusieurs services de livraison a la fois afin d’étendre facilement notre couverture des villes françaises.

# Architecture cible

Notre application sera composée de 3 grandes parties :

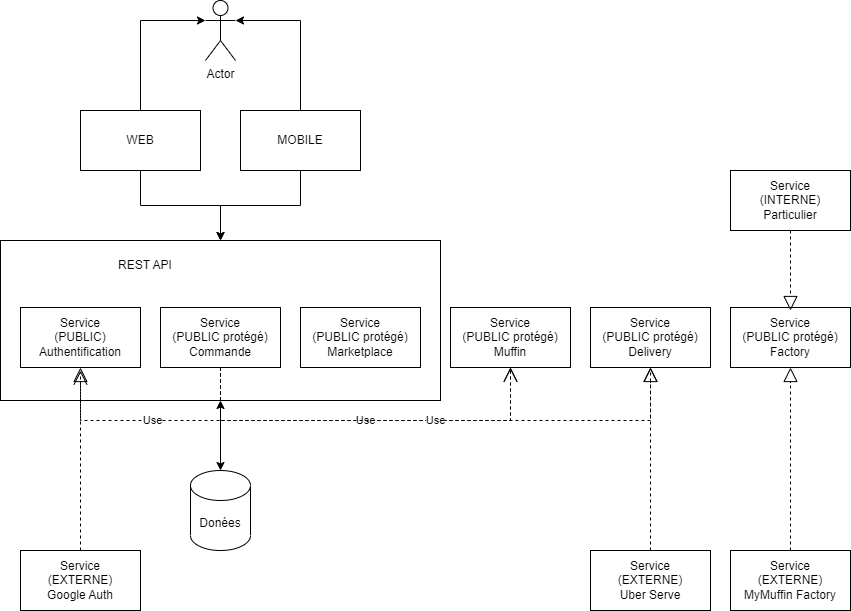
* L’API REST en micro-services
* La Web View
* La Mobile View

L’API REST est au centre du projet et doit être robuste elle permettra la communication entre les différents acteurs du projet, à des fins de scalabilité elle sera découpée en micro service :

* Service d’authentification (public)
* Service de commande (public protégé par l’authentification)
* Service de récupération des différentes marketplaces et de leurs produits (public)
* Service de modification du muffin (public protégé par l’authentification)
* Service d’envoi des commandes au coursiers (interne)
* Service d’envoi des commandes a une factory (interne)

### Schéma

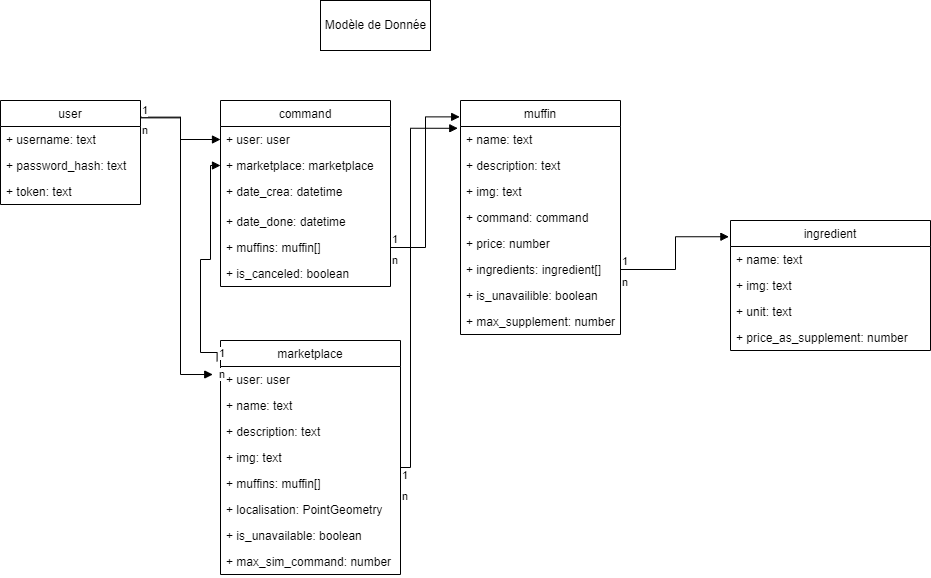
// TODO ajouter des couleurs pour plsu de lisibilité



L’api permettras à un client de s’authentifier via le service d’authentification puis de choisir une boutique via le service de récupération des différentes marketplaces une fois dans une boutique le service de commande prend le relais, celui-ci communiquera avec le service factory afin de savoir si on peut encore y passer commande et avec les services de livraison pour savoir si un coursier est disponible. Une fois toute ses condition réunie la commande est passé.

# Architecture en détail

## Modèle de données



La base de données sera développée à l’aide de [SurrealDB](https://surrealdb.com/) une base de données moderne qui nous permet une grande flexibilité et d’avoir le realtime sur les parties critiques du projet tel que les stocks.

## Backend

Nous devons limiter la logique customisée de notre backend au maximum, pour tout ce qui est algorithme métier mous utiliserons la fonctionnalité d’extension de SurrealDB en javascript ou webassembly.

L’api REST proposé par SurealDB devrais être suffisante pour mener à bien ce projet

Si besoin est nous opterons pour des cloud fonction pour des parties de code qui seras trop complexe à mettre en place directement sur SurrealDB.

## Frontend

### Mobile

Le frontend mobile seras réalisé à l’aide de flutter.

En premier lieu une librairie de composant réutilisable suivant notre charte graphique seras a réalisé avec le moins de dépendance possible afin de limiter la casse si une dépendance n’est plus utilisé.

Une fois terminé nous utiliserons [Riverpod](https://riverpod.dev/fr/) pour le state management et la navigation, qui nous permet une grande flexibilité et une très bonne granularité sur le rafraichissement de l’UI notamment sur le real time et le partage de données entre les différents écrans.

### Web

Le frontend web seras réaliser en Svelte, suivant le même modèle de développement que la version mobile, svelte possède une multitude de fonctionnalité très intéressante a note projet comme les stores permettant de s’abonner a des données en temps réel, ainsi que la gestion de l’asynchrone de manière native.

SvelteKit est a envisagé pour ce qui concerne la navigation, mais il n’est pas forcément nécessaire

Nous devons trouver une façon de créer des liens inter compatible entre la vue mobile et la vue web

# Architecture physique

Toute l’application sera dockerisé pour une facilité de reproduction, un docker compose seras mis a disposition des développeur avec les parties externe au projets mocké grâce au différents acteurs mettant à disposition ces mocks pour les développeurs.

Nous utiliserons AWS pour l’host des environnement staging et production qui aurons sensiblement la même configuration.

* L’environnement local permettras au développeur de travailler efficacement sans empiéter sur le travail des autres, il sera parfait pour tester des nouvelles fonctionnalités.
* Statging sera un environnement utilisé pour les tests partagés, des tests entre les développeurs ou à debugger des soucis arrivés jusqu’à la prod, il sera alimenté en données de production anonymisé toutes les fins de mois.
* Prod est l’environnement final utilisé par les différents acteurs du projet, principalement les clients de MyMuffins. Cet un environnement ou **il est strictement interdit de faire de l’écriture direct** en base de données. Il se doit d’être le plus stable possible de d’être surveiller autant que possible. La mise en place d’un service tel que [Sentry](https://sentry.io/welcome/) seras envisageable afin d’avoir une visibilité sur les problèmes présent sur la production.

Nous attendons les retours de notre expert devOps pour détailler cette partie. À l’heure actuel nous avons une base de docker compose pour le local. Les informations de connections et autre détail serons communiqué dans cette sections plus tard.

# Annexe

## Croquis de l’UI mobile

