| **Kiddymap**  **Projet 12**  Dossier de conception technique  Version 1.0 |
| --- |
| **Auteur**  Charlotte VANHUYSE |

Table des matières

[**Versions**](#_heading=h.gjdgxs) **3**

[**Introduction**](#_heading=h.30j0zll) **4**

[Objet du document](#_heading=h.1fob9te) 4

[Références](#_heading=h.ew2c67vkyigz)

**Architecture technique 5**

Diagramme de composants *6*

[Les composants généraux](#_heading=h.17dp8vu) 6

Frontend 6

Backend 7

Base de données 7

Composants externes 7

**Architecture de déploiement 8**

[Diagramme d](#_heading=h.35nkun2)e déploiement 8

**Architecture logicielle 9**

Principes généraux 9

Les couches 9

Api Web 9

Les modules 9

Structure des sources 10

Api Web 12

Les modules 12

Structure des sources 12

**Points Particuliers 13**

Gestion des logs 13

Livraison 13

1. Versions

| **Auteur** | **Date** | **Description** | **Version** |
| --- | --- | --- | --- |
| Charlotte VANHUYSE | 15/03/2022 | Création du document | 1.0 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Introduction
   1. Objet du document

Le présent document constitue le dossier de spécifications techniques de l'application Kiddymap. Les éléments du présents dossiers découlent de l’analyse des besoins du projet et de la rédaction du dossier de conception fonctionnelle.

* 1. Références

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

1. **Projet 12 - Dossier de conception fonctionnelle** : Dossier de conception fonctionnelle de l'application
2. **Projet 12 - Bilan du projet** : Un bilan qui décrit rétrospectivement mon expérience durant le projet.
3. **Projet 12 - Note d’intention** : Une note d’intention expliquant les objectifs du projet.
4. Architecture technique
   1. Diagramme de composants

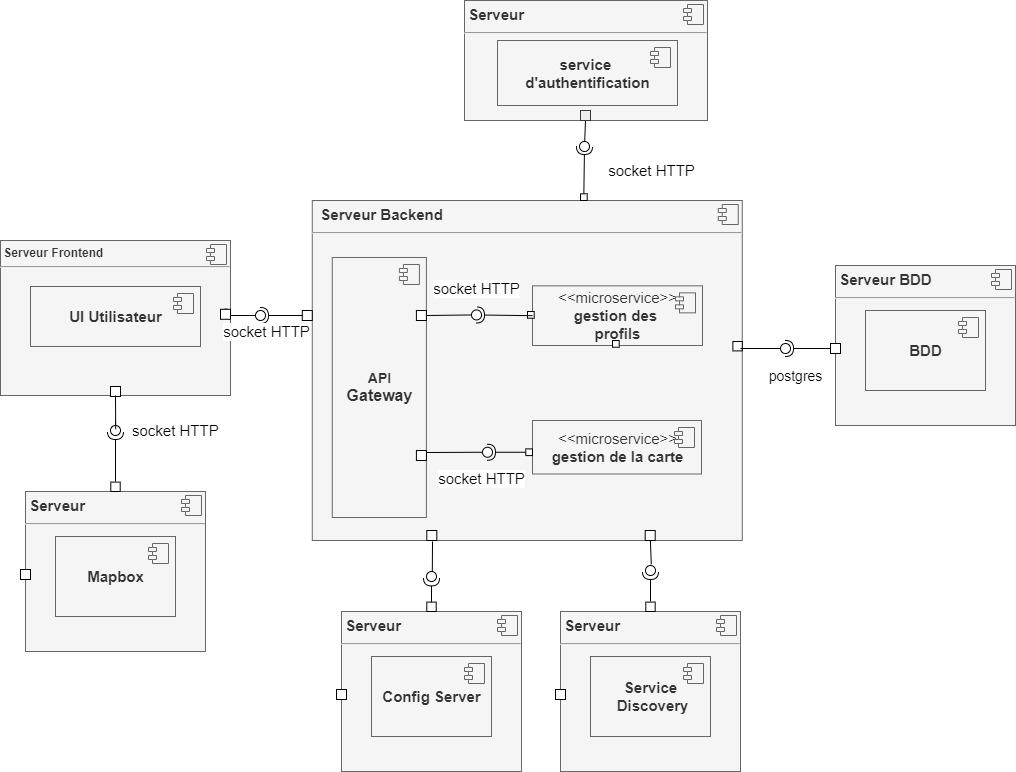
****

Diagramme de composants

* 1. Les composants généraux
     1. *Frontend*

La pile logicielle est la suivante : Angular

* + - 1. *Interface Utilisateur*

Ce composant propose une interface graphique dédiée aux clients.

* + 1. *Backend*

La pile logicielle est la suivante :

* Application web Java (JDK version 11)
* SpringBoot (2.3.0.RELEASE)
* Apache Maven (4.0.0)
* IntelliJ Ultimate (2020.2)
  + - 1. *Serveur de configuration*

Ce composant permet d’externaliser la configuration des microservices de l’application. On utilisera Config Server qui fait partie de la stack Spring Cloud Netflix.

* + - 1. *Discovery server*

Ce composant permet d’enregistrer les instances des microservices de l’application. On utilisera Eureka qui fait partie de la stack Spring Cloud Netflix.

* + - 1. *Api gateway*

Ce composant est l’unique point d’entrée de notre application. On utilisera Gateway qui fait partie de la stack Spring Cloud Netflix.

* + - 1. *gestion des profils*

Ce composant permet de gérer les profils des utilisateurs.

* + - 1. *gestion de la carte*

Ce composant permet de gérer les données des lieux présents sur la carte ainsi que les équipements associés.

* + 1. *Base de données*

La pile logicielle est la suivante : PostgreSQL

Ce composant correspond à la base de données de l’application qui communique avec les composants “ gestion des profils” et “ gestion de la carte”.

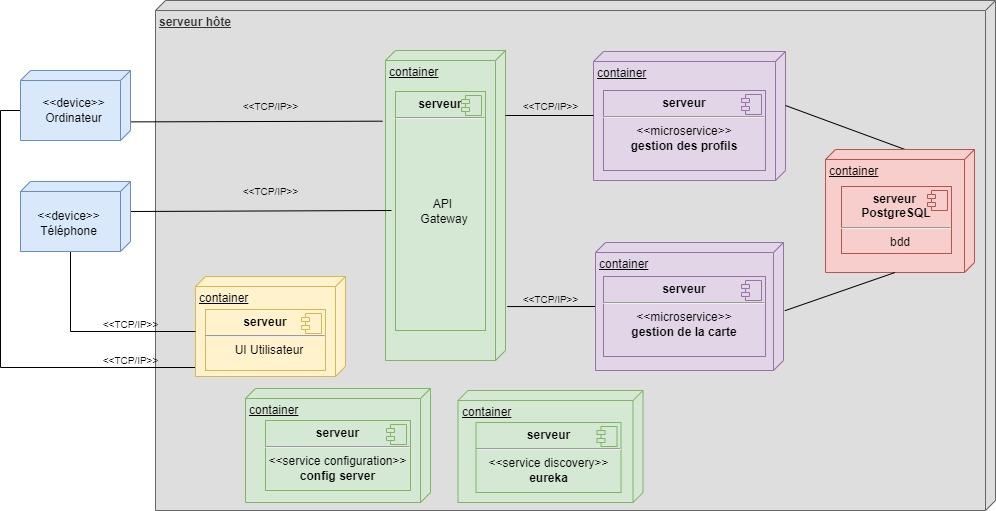
* + 1. *Composants externe*
       1. *Mapbox*

Ce composant externe permet d’initialiser la carte.

* + - 1. *Serveur d’authentification*

Auth0 permet de gérer l”inscription, l’identification et les permissions liées aux utilisateurs.

1. Architecture de déploiement
   1. Diagramme de déploiement

Diagramme de déploiement

1. Architecture logicielle
   1. Principe généraux

Les sources et versions du projet sont gérées par **Git**, Le code source de l’application se trouve sur **Github**, les dépendances et le packaging seront gérés par **Apache Maven**.

Le développement de l’application se fera en suivant une architecture microservice, on utilisera le framework Spring Boot pour le backend.

* + 1. *Les couches*

L'architecture applicative est la suivante :

* unecouche **business** : responsable de la logique métier du composant, elle est implémentée dans le package “service”.
* unecouche **model** : implémentation du modèle des objets métiers, elle est implémentée dans le package “entity”.
* Une couche **consumer** : responsable des interactions avec la base de données, elle est implémentée dans le package “repository” qui comporte les DAO.
* Une couche **présentation** : constitue l’interface entre le client et l’application, elle est implémentée dans le package "controller".
* Un package “**configuration**” sera également présent.
  1. Api Web
     1. *Les modules*

L’application suit une architecture multi-module Maven où chaque module est dédié à un microservice différent :

* Config-server
* Eureka-server
* Api-gateway
* microservice-profil
* microservice-location.
  + 1. *Structure des sources*

La structuration des répertoires du projet suit la logique suivante :

* les répertoires sources sont créés de façon à respecter la philosophie Maven (à savoir : « convention plutôt que configuration »)

kiddymap  
 ├─ *pom.xml*

├─ *docker-compose.yml*

├─ <eureka-server>  
 │ ├─ *pom.xml*

│ ├─ *.gitignore*  
 │ ├─ main  
 │ │ ├─ java

│ │ │ └─ EurekaServerApplication.java

│ │ ├─ resources

│ │ │ └─ application.properties  
 │ └─ test

│ ├─ java  
 │ └─ resources

├─ <api-gateway>  
 │ ├─ *pom.xml*

│ ├─ *.gitignore*  
 │ └─ src  
 │ ├─ main  
 │ │ ├─ java

│ │ │ ├─ configuration

│ │ │ └─ ApiGatewayApplication.java

│ │ ├─ resources

│ │ │ └─ application.properties  
 │ └─ test

│ ├─ java  
 │ └─ resources

├─ <microservice-profil>  
 │ ├─ *pom.xml*

│ ├─ *.gitignore*  
 │ └─ src  
 │ ├─ main  
 │ │ ├─ java

│ │ │ ├─ configuration

│ │ │ ├─ controller

│ │ │ ├─ entity

│ │ │ ├─ repository

│ │ │ ├─ service

│ │ │ └─ MicroserviceProfilApplication.java

│ │ ├─ resources

│ │ │ └─ application.properties  
 │ └─ test

│ ├─ java  
 │ └─ resources

├─ <microservice-location>  
 │ ├─ *pom.xml*

│ ├─ *.gitignore*  
 │ └─ src  
 │ ├─ main  
 │ │ ├─ java

│ │ │ ├─ configuration

│ │ │ ├─ controler

│ │ │ ├─ entity

│ │ │ ├─ repository

│ │ │ ├─ service

│ │ │ └─ MicroserviceLocationApplication.java

│ │ └─ resources

│ │ │ └─ application.properties  
 │ └─ test

│ ├─ java  
 │ └─ resources

├─ *README.md*

└─ .gitignore

* 1. Application Web
     1. *Les modules*

L’application web est cliente de l’API Web :

* UI-utilisateur : elle sera utilisée par les utilisateurs du site web.
  + 1. *Structure des sources*

<ui-utilisateur>  
 ├─

├─

├─ *.gitignore*  
 └─ src

└─ app  
 ├─ component

├─ service

├─ model

├─ app.module.ts

├─ app-routing.module.ts  
 └─ environment

1. Points Particuliers
   1. Gestion des log

**Logback** est nativement incorporé dans **Spring Boot** via la dépendance **spring-boot-starter**, elle-même comprise dans toutes les dépendances de type starter. LogBack utilise **SLF4J** comme interface native.

* 1. Livraison

Le code source de l’application est disponible sur Github :

Backend : <https://github.com/vancharlotte/projet12-kiddymap.git>

Frontend : <https://github.com/vancharlotte/projet12-front.git>