|  |
| --- |
| CFPT Informatique |
| GrinEat |
| Documentation Technique TPI 2022 |

|  |
| --- |
| BRIAN.GRN  02/05/2022 |

# Table des matières

[Table des matières 1](#_Toc103158595)

[1 Table des versions 3](#_Toc103158596)

[2 Introduction 3](#_Toc103158597)

[2.1 Généralités 3](#_Toc103158598)

[3 Analyse concurrentielle 4](#_Toc103158599)

[3.1 JustEat ou Eat.ch 4](#_Toc103158600)

[3.2 Smood 4](#_Toc103158601)

[3.3 UberEats 5](#_Toc103158602)

[3.4 Positionnement par rapport à la concurrence 6](#_Toc103158603)

[4 Rappel de l’énoncé 6](#_Toc103158604)

[4.1 Organisation 6](#_Toc103158605)

[4.2 Livrables 7](#_Toc103158606)

[4.3 Matériel et logiciels à disposition 7](#_Toc103158607)

[4.4 Description du site 7](#_Toc103158608)

[5 Méthodologie 8](#_Toc103158609)

[5.1 S’informer 8](#_Toc103158610)

[5.2 Planifier 8](#_Toc103158611)

[5.3 Décider 9](#_Toc103158612)

[5.4 Réaliser 9](#_Toc103158613)

[5.5 Contrôler 9](#_Toc103158614)

[5.6 Evaluer 9](#_Toc103158615)

[6 Planification 9](#_Toc103158616)

[6.1 Planning prévisionnel 9](#_Toc103158617)

[6.2 Planning effectif 9](#_Toc103158618)

[6.3 Bilan planning 9](#_Toc103158619)

[7 Analyse fonctionnelle 9](#_Toc103158620)

[8 Analyse organique 9](#_Toc103158621)

[8.1 Technologies utilisées 9](#_Toc103158622)

[8.2 Environnement 10](#_Toc103158623)

[8.3 Description de la base de données 10](#_Toc103158624)

[8.3.1 Modèle logique de données 10](#_Toc103158625)

[9 Plan de test 12](#_Toc103158626)

[9.1 Périmètre 12](#_Toc103158627)

[9.2 Description des tests 12](#_Toc103158628)

[9.3 Scénarios de tests 13](#_Toc103158629)

# Table des versions

|  |  |  |  |
| --- | --- | --- | --- |
| N° de version | Date | Auteur | Changements apportés |
| 0.1 | 02.05.2022 | Brian Grin | Structure de la documentation, Introduction, Cahier des charges, planification prévisionnelle et backlog |
| 0.2 |  |  |  |
| 0.3 |  |  |  |

# Introduction

## Généralités

Ce document a pour but de décrire le procédé et le raisonnement qui a servi à réaliser le projet « GrinEat » dans le cadre de mon travail de pratique individuel (TPI). Ce travail sert à vérifier ma légitimité à l’obtention du CFC Informaticien à l’issu de ce travail.

GrinEat est une application web permettant de consulter les restaurants autour d’un périmètre choisi autour d’une adresse fournie et de consulter les menus du restaurant sélectionné.

L’affichage se fait sous 2 formats :

* Une liste de restaurant dans l’ordre croissant de la distance par rapport à l’adresse renseignée
* Une carte interactive affichant des points sur la carte là où se situe les restaurants

L’application communique directement avec un serveur qui sera développé en même temps que le client et qui s’occupera de toute la partie de traitement l’application aura juste a affiché ce qu’on lui fournit.

J’ai choisi ce projet car il est plutôt simple à réaliser et à présenter avec ses petites touches de difficultés comme l’utilisation d’une carte interactive et la communication client-serveur. Je suis un consommateur occasionnel de service de livraison mais je trouvais la réalisation d’une application similaire intéressante. Malheureusement que 88h sont accordés pour réaliser ce travail j’aurais souhaité pousser plus loin et faire un affichage en temps réel du livreur sur la carte à l’aide d’une simulation mais cela demanderait plus de temps.

Néanmoins cela reste un bon projet et un très bon défi regroupant tout ce dont on attend d’un apprenti finalisant son apprentissage.

# Analyse concurrentielle

## JustEat ou Eat.ch

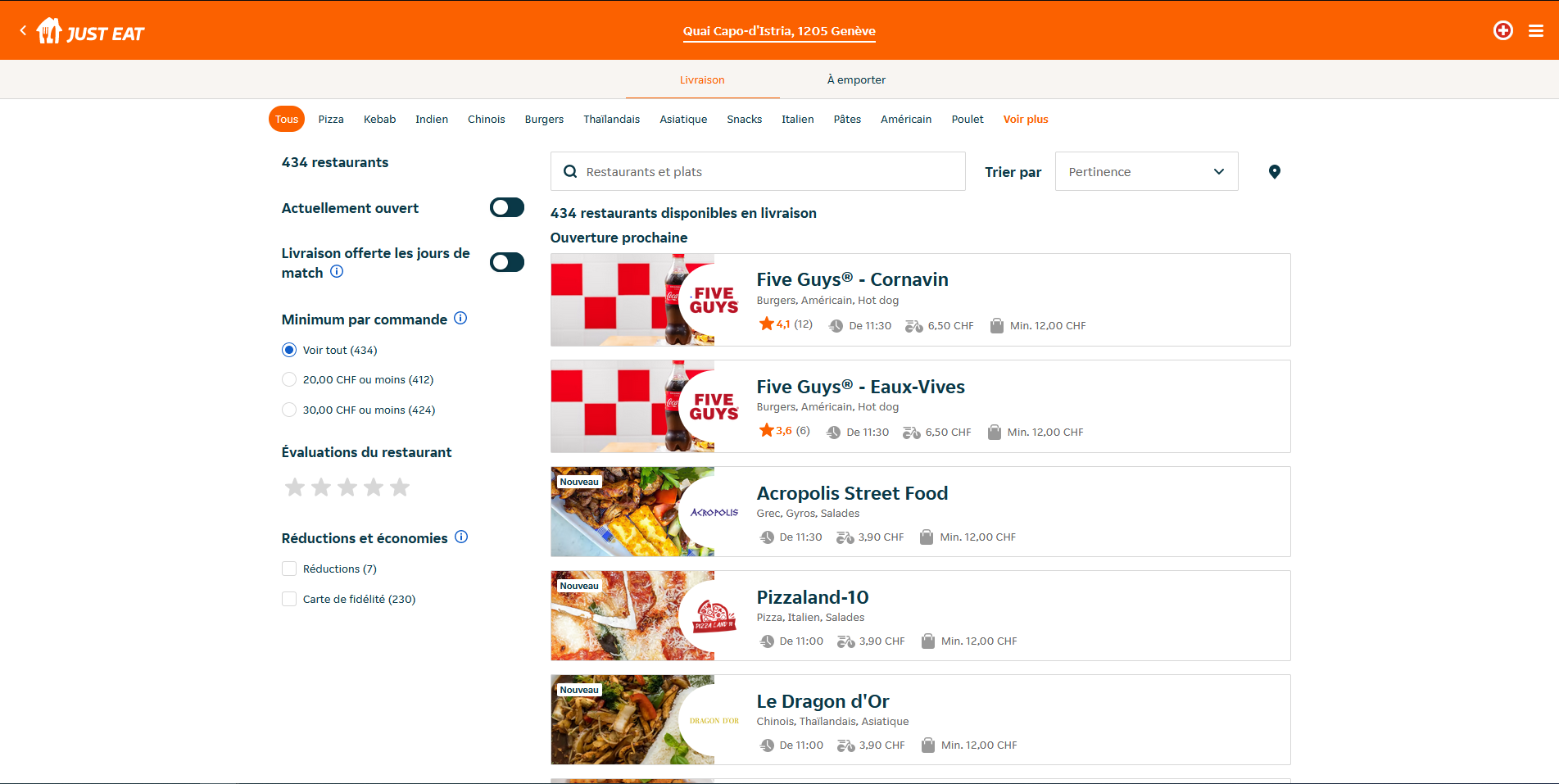


Figure : Page principale de eat.ch

Il est le premier apparu en suisse. Il a un design plutôt basique mais les informations sont bien regroupées on prend le site en main très rapidement. Il permet aussi d’effectuer des commandes à distance à l’emporter pas que de la livraison.

Il reste un bon concurrent mais selon moi pas le plus grand, le design reste très basique et il n’y a pas de carte interactive qui sera surement un plus pour GrinEat.

## Smood

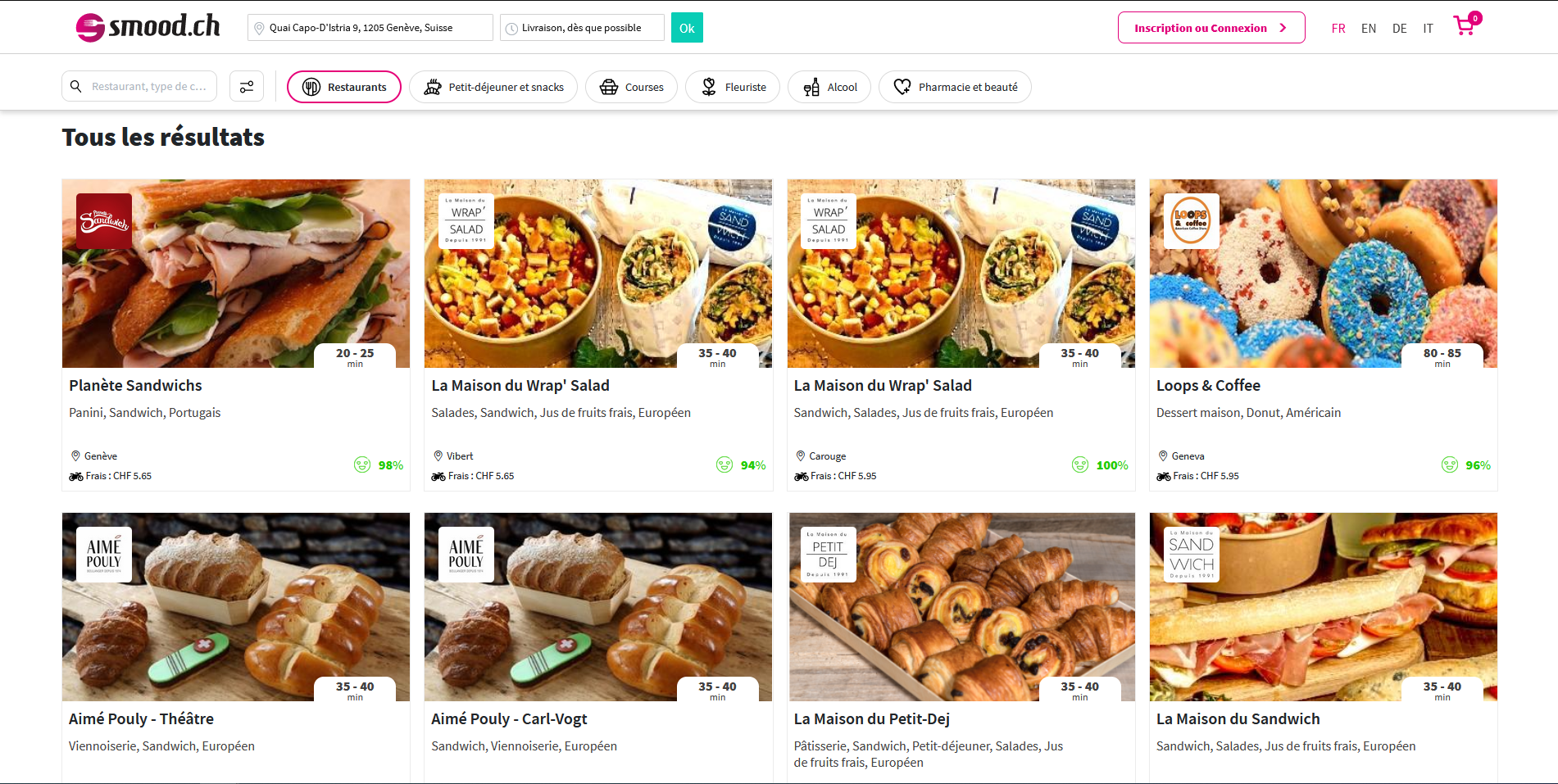


Figure : Page principale de smood.ch

Il est le dernier arrivé à Genève (mais créé avant Ubereats). Son design parait plus moderne qu’eat.ch avec des animations mais je pense même si ce n’est qu’un détail il est important selon moi il n’y a pas de carte interactive de nouveau ici mais les restaurants sont proposés en ligne, sur un écran 16:9 on a 4 restaurants par ligne comparé à eat.ch qui comportait juste une colonne de restaurant ce qui forçait à scroller plus pour consulter la liste des restaurants.

Il permet de planifier des commandes c’est un plus comparé au concurrent précédent et il permet aussi de faire des commandes à emporter depuis le site.

Surtout, il est celui avec le meilleur système de filtre séparant les filtres par cuisines et spécialités par exemple pour la cuisine thaï ou pour la spécialité Pad Thai.

En conclusion, il se place au-dessus d’eat.ch selon moi avec son design moderne et les petits détails qui font la différence.

## UberEats

Une image contenant texte

Description générée automatiquement

Figure : Page principale UberEats

Finalement j’en parle en dernier car juste par son nom il est surement déjà considéré pour le plus grand concurrent, qui ne connait pas UberEats quand l’on parle de service de livraison surement celui le plus présent dans le monde filial de l’entreprise américaine Uber.

La renommée c’est bien beau mais que donne le site ? Il est surement celui avec le design le plus attirant beaucoup d’images présentes, pour les restaurants comme les autres sites mais aussi pour les catégories. Il propose la liste des restaurants en ligne aussi.

Une image contenant carte

Description générée automatiquement

Figure : Carte interactive UberEats

Surtout il est le seul des trois à proposer une carte interactive qui était jusqu’à maintenant notre fonctionnalité clé mais par chance cette carte interactive ne présente que les lieux pour faire des commandes à emporter et pas les restaurants qui livrent.

Finalement, il est surement le plus grand concurrent ici pour le projet GrinEat dans un contexte lucratif. Bien que la carte interactive soit pas présentée directement mais une fonctionnalité en plus laissant le choix à l’utilisateur de l’utiliser ou non et ne soit pas utiliser dans le même but que GrinEat. Néanmoins j’aurais un reproche quand même à ce site qui est le fait que la navigation comportant la barre de recherche et les catégories ne soit pas statique quand l’on scroll ce qui force l’utilisateur à remonter si finalement il souhaite effectuer une recherche ou filtrer par une catégorie.

## Positionnement par rapport à la concurrence

Ce projet se réalisant dans un cadre scolaire pour l’obtention d’un diplôme, n’a donc pas de but lucratif réel. Cependant observer la concurrence et chercher à faire mieux reste une motivation il nous donne un bon sens de l’analyse pour juger au détail ce qu’on peut faire de mieux.

Des meilleurs projets similaires sont donc déjà présents et le temps accordé ne permet pas de réaliser un site aussi complet c’est pourquoi tout en restant conforme au cahier des charges. GrinEat regroupera dans la mesure du possible le meilleur de ses concurrents. Le point fort de GrinEat sera la carte interactive qui facilitera l’estimation du temps d’une livraison et l’usage pour l’utilisateur comparé à une liste qui donne juste l’adresse du restaurant on ne connait pas forcément toutes les adresses.

# Rappel de l’énoncé

## Organisation

|  |  |  |
| --- | --- | --- |
| Rôle | Prénom/Nom | Courriel |
| Elève | Brian Grin | brian.grn@eduge.ch |
| Maitre d’apprentissage | Antoine Schmid | antoine.schmid@edu.ge.ch |
| Experts | Pascal Court | pascal.court@hotmail.com |
| Francesco Foti | francesco.foti@devinfo.net |

## Livrables

Au maitre d’apprentissage :

* Accès au repository GIT avec droit de clone
* Un dump de la base de données contenant des données de test

Aux experts :

* Un exemplaire PDF du rapport TPI, contenant :
  + Documentation technique
  + Manuel utilisateur
  + Code source
  + Le journal des « commits »

## Matériel et logiciels à disposition

## Description du site

GrinEat est un site web permettant de trouver des restaurants et de consulter leurs menus. La recherche des restaurants est basée selon une adresse saisie manuellement, le rayon (distance à vol d’oiseau entre le restaurant et l’adresse), le type de cuisine ou le nom du restaurant. Les restaurants trouvés sont affichés sur une carte et sous forme de liste. La sélection d’un restaurant permet d’afficher le menu.

Voici les fonctionnalités qui doivent être implémentée :

* Pour le serveur :
* Retourner toutes les catégories de la base de données
* Retourner les restaurants
* Retourner le menu des restaurants
* Filtrer les restaurants par un rayon (distance à vol d’oiseau entre le restaurant et l’adresse)
* Filtrer par une ou des catégorie(s) de restaurants
* Recherche par nom de restaurant
* Pour le client :
* Rechercher un restaurant selon :
  + Une adresse saisie manuellement
  + Le rayon (distance à vol d’oiseau entre le restaurant et l’adresse)
  + La catégorie
  + Le nom du restaurant
* Consulter les restaurants trouvés sur une carte interactive
* Consulter les restaurants trouvés selon une liste
* Sélectionner un restaurant pour afficher sa page de menu

# Méthodologie



Figure : La méthode en six étapes

## S’informer

Au départ j’ai d’abord pris connaissance de ce qui m’était donné à faire dans l’énoncé. J’ai réfléchi un peu déjà à la manière dont je pourrais m’y prendre ce qui m’a mené à des incompréhensions dont j’ai pu demander l’explication par mail à mon maître d’apprentissage. J’ai aussi consulté les sites déjà présents du même type afin de les analyser et pouvoir m’en inspirer.

## Planifier

J’ai ensuite commencé la réalisation de mon planning prévisionnel afin d’estimer le temps nécessaire pour la réalisation des tâches et avoir un fil conducteur auquel je puisse me suivre et me rendre compte de mon avance ou retard. La réalisation du planning m’aura mené à une première réflexion des étapes et de la manière dont j’allais procéder pour réaliser ce travail j’ai pu éclaircir certains points dont je n’étais pas sûr. Pour la séparation des tâches j’ai regroupé l’administratif ensemble et pour ce qui est du projet je l’ai séparé en deux (Front et Back end) le serveur et le client avec en priorité la réalisation du serveur car sans lui aucunes données ne sera affichée sur le client. J’ai réalisé tout cela sur une feuille Excel mis à disposition pour les élèves passant leur TPI où l’on a deux feuilles (le planning prévisionnel et le planning effectif).

## Décider

Text

## Réaliser

Texte

## Contrôler

Texte

## Evaluer

Texte

# Planification

## Planning prévisionnel

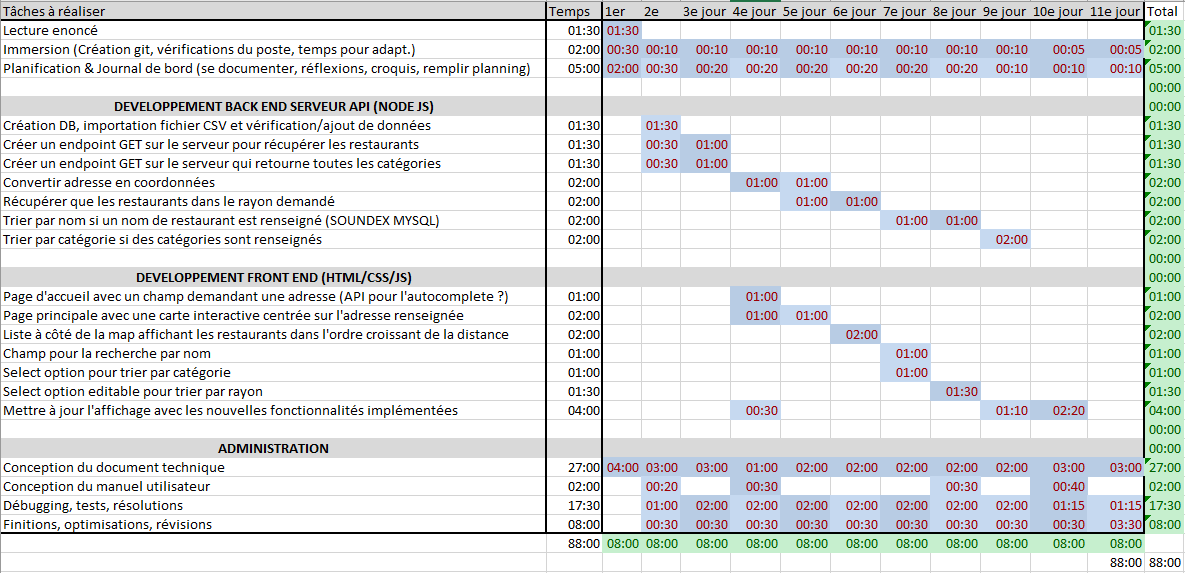


Figure : Planning prévisionnel

## Planning effectif

## Bilan planning

# Analyse fonctionnelle

# Analyse organique

## Technologies utilisées

* HTML
* CSS
* Javascript
* NodeJS
* PHP 7.4.28
* MariaDB
* Bootstrap 5
* Leaflet
* MapBox

## Environnement

Pour la rédaction de la documentation technique, du manuel utilisateur et la conception du planning prévisionnel et effectif je travaille sur le disque dur Windows 10 fourni par l’école dans la cadre de l’apprentissage, Windows donnant accès à la suite Office pour Excel et Word qui sont les logiciels les plus pratiques.

Pour ce qui est du développement je code directement depuis mon ordinateur personnel portable sous Manjaro qui est une distribution Linux basée sur Arch Linux. J’y ai installé LAMP en installant les paquets requis (style : apache, php, php-apache, mysql).

J’ai utilisé Visual Studio Code comme IDE pour le développement et phpMyAdmin pour les manipulations avec la base de données.

## Description de la base de données

L’encodage par défaut de la base de données est utf8mb4 et le moteur est InnoDB.

### Modèle logique de données

Un modèle de base de données était livré avec le cahier des charges je n’y ai apporté aucune grande modification à part le nommage des tables et champs et l’ajout d’un champ « nameEnglish » et « nameFrench » dans les tables « countries » et « categories », car lors de l’ajout j’avais ajouté les noms anglais en premier sans penser que l’affichage de l’application serait en français.

Une image contenant texte

Description générée automatiquement

Figure 7 - Modèle livré avec le cahier des charges

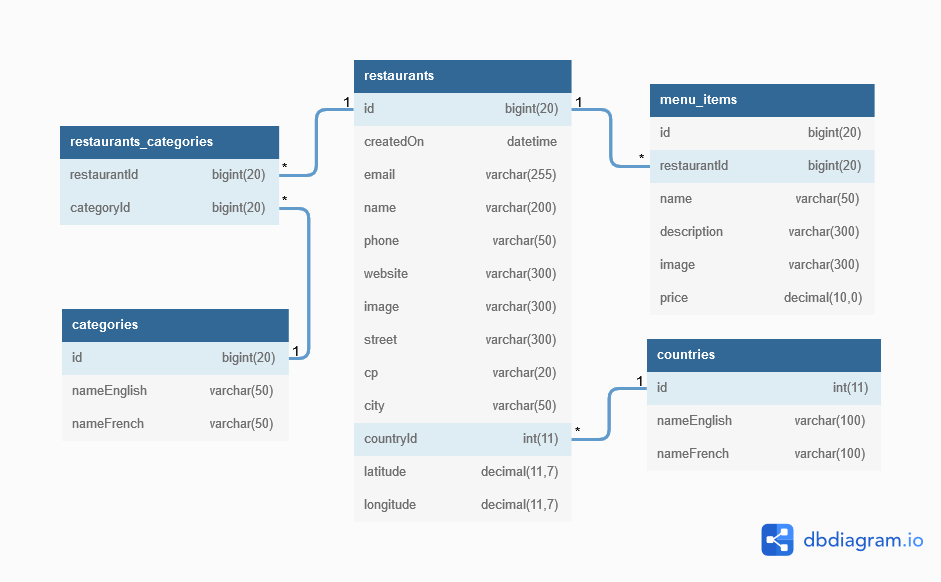


Figure 8 - Modèle utilisé avec l'application

#### Détails « restaurants »

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Colonne | Type | Null | Valeur par défaut | Commentaire |
| id | bigint (20) | Non |  | Primary Key / Auto Increment |
| createdOn | datetime | Non | current\_timestamp() |  |
| email | varchar(255) | Oui | NULL |  |
| name | varchar(200) | Non |  |  |
| phone | varchar(50) | Oui | NULL |  |
| website | varchar(300) | Oui | NULL |  |
| image | varchar(300) | Non |  |  |
| street | varchar(300) | Non |  |  |
| cp | varchar(20) | Non |  |  |
| city | varchar(50) | Non |  |  |
| countryId | int(11) | Non |  | Index / Foreign Key |
| latitude | decimal(11,7) | Non |  |  |
| longitude | decimal(11,7) | Non |  |  |

#### Détails « menu\_items »

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Colonne | Type | Null | Valeur par défaut | Commentaire |
| id | bigint (20) | Non |  | Primary Key / Auto Increment |
| restaurantId | bigint(20) | Non |  | Index / Foreign Key |
| name | varchar(50) | Non |  |  |
| description | varchar(300) | Non |  |  |
| image | varchar(300) | Non |  |  |
| price | decimal(10,0) | Non |  |  |

#### Détails « countries »

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Colonne | Type | Null | Valeur par défaut | Commentaire |
| id | bigint (20) | Non |  | Primary Key / Auto Increment |
| nameEnglish | varchar (100) | Non |  |  |
| nameFrench | varchar(100) | Non |  |  |

La table countries a été rempli à l’aide d’un script fait par moi-même avec lequel j’ai appelé l’API « restcountries » pour récupérer tous les pays et je les ai insérés dans la base de données.

#### Détails « restaurants\_categories»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Colonne | Type | Null | Valeur par défaut | Commentaire |
| restaurantId | bigint (20) | Non |  | Primary Key / Foreign Key |
| categoryId | bigint (20) | Non |  | Primary Key / Foreign Key |

#### Détails « categories »

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Colonne | Type | Null | Valeur par défaut | Commentaire |
| id | bigint (20) | Non |  | Primary Key / Auto Increment |
| nameEnglish | varchar (50) | Non |  |  |
| nameFrench | varchar(50) | Non |  |  |

# Plan de test

Dans le cadre du développement de mon application, le cahier des charges exige que je mette en place un protocole de test.

## Périmètre

Pour ce qui est du périmètre de mon protocole de test, le plus important sera la communication entre le client et le serveur puis l’affichage ensuite je vais donc séparer mes tests en deux points :

1. Les tests uniquement côtés serveurs (L’API reçoive bien les données envoyées par http, le traitement et qu’elle renvoie les bonnes informations Ex : je demande la liste des catégories le serveur me renvoie bien un JSON avec toutes les catégories) Cette partie sera testé avec Postman.
2. Les tests côtés client (On affiche les bonnes informations reçus par le serveur on envoie bien les bonnes informations Ex : Quand je cherche Mcdonald’s on m’affiche des Mcdonald’s sur la carte).

## Description des tests

|  |  |
| --- | --- |
| **Nom du test** | 1.1 | Récupérer les catégories |
| **Description** | On souhaite récupérer un JSON contenant les catégories existant dans la base de données. |
| **Comment le tester** | Effectuer une requête HTTP GET sur l’endpoint /categories du serveur |

|  |  |
| --- | --- |
| **Nom du test** | 1.2 | Récupérer les restaurants sans filtre |
| **Description** | On souhaite récupérer un JSON contenant les restaurants en envoyant juste notre adresse sans filtre il retourne les restaurants dans un périmètre de 5 km par défaut. |
| **Comment le tester** | Effectuer une requête HTTP POST sur l’endpoint /restaurants du serveur avec dans le body en JSON une propriété address contenant une addresse. |

|  |  |
| --- | --- |
| **Nom du test** | 1.3 | Erreur 500 pas d’adresse |
| **Description** | On teste un cas où l’on ne renseigne pas d’adresse et on s’attend à une erreur 500 renvoyé par le serveur. |
| **Comment le tester** | Effectuer une requête HTTP POST sur l’endpoint /restaurants du serveur avec rien dans le body |

|  |  |
| --- | --- |
| **Nom du test** | 1.4 | Récupérer les restaurants en filtrant par catégories |
| **Description** | On souhaite récupérer un JSON contenant les restaurants filtrés par les catégories qu’on envoie et notre adresse. |
| **Comment le tester** | Effectuer une requête POST sur l’endpoint /restaurants du serveur avec dans le body en JSON une propriété address contenant l’adresse et une propriété categories contenant un tableau de 1 ou plusieurs catégories. |

|  |  |
| --- | --- |
| **Nom du test** | 1.5 | Récupérer les restaurants en filtrant par nom |
| **Description** | On souhaite récupérer un JSON contenant les restaurants dont le nom ressemble au nom qu’on envoie et notre adresse. |
| **Comment le tester** | Effectuer une requête POST sur l’endpoint /restaurants du serveur avec dans le body en JSON une propriété address contenant l’adresse et une propriété name contenant le nom qu’on cherche. |

|  |  |
| --- | --- |
| **Nom du test** | 1.6 | Récupérer les restaurants en filtrant par rayon |
| **Description** | On souhaite récupérer un JSON contenant les restaurants dans un rayon autour de l’adresse qu’on envoie. |
| **Comment le tester** | Effectuer une requête POST sur l’endpoint /restaurants du serveur avec dans le body en JSON une propriété address contenant l’adresse et une propriété radius avec le nombre en kilomètre exemple : 5 (pour 5km). |

|  |  |
| --- | --- |
| **Nom du test** | 1.7 | Erreur 500 adresse chaine vide |
| **Description** | On teste un cas où l’on ne renseigne une adresse vide et on s’attend à une erreur 500 renvoyé par le serveur. |
| **Comment le tester** | Effectuer une requête HTTP POST sur l’endpoint /restaurants du serveur avec un JSON contenant la propriété adresse mais une chaine vide |

|  |  |
| --- | --- |
| **Nom du test** | 1.8 | Récupérer les éléments d’un menu d’un restaurant |
| **Description** | On souhaite récupérer les éléments du menu d’un restaurant, recevoir un JSON contenant les éléments quand on renseigne l’id d’un restaurant |
| **Comment le tester** | Effectuer une requête HTTP GET sur l’endpoint /restaurants/{id}/menu |

|  |  |
| --- | --- |
| **Nom du test** | 2.1 | Rentrer une adresse valide sur la page d’accueil |
| **Description** | On souhaite rentrer notre adresse et être rediriger sur la page des restaurants. |
| **Comment le tester** | Rentrer une adresse à l’aide de l’autocomplete séléctionnez l’adresse que vous souhaitez. |

|  |  |
| --- | --- |
| **Nom du test** | 2.2 | Entrer une adresse invalide sur la page d’accueil |
| **Description** | On souhaite rentrer une adresse invalide et recevoir une alerte indiquant que l’adresse est invalide. |
| **Comment le tester** | Rentrer une adresse aléatoire sans l’aide de l’autocomplete. |

|  |  |
| --- | --- |
| **Nom du test** | 2.3 | Tri par nom |
| **Description** |  |
| **Comment le tester** |  |

|  |  |
| --- | --- |
| **Nom du test** | 2.4 | Tri par rayon |
| **Description** |  |
| **Comment le tester** |  |

|  |  |
| --- | --- |
| **Nom du test** | 2.5 | Tri par catégories |
| **Description** |  |
| **Comment le tester** |  |

## Scénarios de tests

Tableau récapitulatif de l’état des tests au fur et à mesure des 11 jours.

**✗** = signifie que le test est raté.

✔️ = signifie que le test est passé

# = signifie que le test est passé partiellement

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test | J 1 | J 2 | J 3 | J 4 | J 5 | J 6 | J 7 | J 8 | J 9 | J 10 | J 11 |
| 1.1 | **✗** | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ |
| 1.2 | **✗** | **✗** | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ |
| 1.3 | **✗** | **✗** | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ |
| 1.4 | **✗** | **✗** | # | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ |
| 1.5 | **✗** | **✗** | **✗** | **✗** | **✗** | **✗** | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ |
| 1.6 | **✗** | **✗** | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ |
| 1.7 | **✗** | **✗** | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ |
| 1.8 | **✗** | **✗** | # | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ |
| 2.1 | **✗** | **✗** | **✗** | **✗** | **✗** | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ |
| 2.2 | **✗** | **✗** | **✗** | **✗** | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ |
| 2.3 | **✗** | **✗** | **✗** | **✗** | **✗** | # | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ |
| 2.4 | **✗** | **✗** | **✗** | **✗** | **✗** | # | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ |
| 2.5 | **✗** | **✗** | **✗** | **✗** | **✗** | # | ✔️ |  |  |  |  |