**SOMMAIRE**

[INTRODUCTION 4](#_Toc136598321)

[I. ARCHITECTURE DU PROJET 4](#_Toc136598322)

[I.1. Architecture globale 4](#_Toc136598323)

[I.2. Frontend 6](#_Toc136598324)

[I.3. Backend 7](#_Toc136598325)

[I.4. Base de données 8](#_Toc136598326)

[II. PRESENTATION DES TECHNOLOGIES UTILISEES 11](#_Toc136598327)

[II.1. Le backend 11](#_Toc136598328)

[II.2. Le frontend React 14](#_Toc136598329)

[II.3. Le frontend Laravel 18](#_Toc136598330)

[II.4. La base de données 18](#_Toc136598331)

[III. MISE EN PLACE DE L’ENVIRONNEMENT DE TRAVAIL 19](#_Toc136598332)

[III.1. Processus d’installation des Framework 19](#_Toc136598333)

[IV. REALISATION 24](#_Toc136598334)

[IV.1. Arborescence globale du projet 24](#_Toc136598335)

[IV.2. Présentation des interfaces 28](#_Toc136598336)

**LISTE DES FIGURES**

[**Figure 1 : Architecture globale du projet** 5](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598293)

[**Figure 2 : Architecture du frontend** 7](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598294)

[**Figure 3 : Architecture du backend** 8](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598295)

[**Figure 4 : Logo du langage Python** 12](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598296)

[**Figure 5 : Logo de Pycharm** 13](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598297)

[**Figure 6 : Logo de Postman** 13](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598298)

[**Figure 7 : Logo de Flask** 14](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598299)

[**Figure 8 : Logo de JSON** 14](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598300)

[**Figure 9 : Logo de React** 15](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598301)

[**Figure 10 : Logo de Javascript** 16](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598302)

[**Figure 11 : Logo de Node.js** 16](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598303)

[**Figure 12 : Logo de npm** 17](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598304)

[**Figure 13 : Logo de React-Bootstrap** 18](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598305)

[**Figure 14 : Logo de Visual Studio Code** 18](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598306)

[**Figure 15 : Logo de Apache** 19](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598307)

[**Figure 16 : Logo de MySQL** 20](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598308)

[**Figure 17 : Etape 1 : Installation de Xampp** 21](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598309)

[**Figure 18 : Etape 2 : Installation de Xampp** 22](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598310)

[**Figure 19 : Etape 3 : Installation de Xampp** 22](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598311)

[**Figure 20 : Etape 1 : Installation de Composer** 23](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598312)

[**Figure 21 : Etape 2 : Installation de Composer** 24](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598313)

[**Figure 22 : Arborescence du backend** 25](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598314)

[**Figure 23 : Arborescence du frontend** 26](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598315)

[**Figure 24 : Détails du dossier app** 27](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598316)

[**Figure 25 : Détails du dossier public** 28](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598317)

[**Figure 26 : Détails du dossier resources** 28](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598318)

[**Figure 27 : Détails du dossier routes** 29](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598319)

[**Figure 28 : Les autres fichiers** 29](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%202.docx#_Toc136598320)

# INTRODUCTION

## ARCHITECTURE DU PROJET

### Architecture globale

L’architecture globale de notre projet de décline comme suit :

Une image contenant texte, capture d’écran, diagramme, Police

Description générée automatiquement

Figure 1 : Architecture globale du projet

L’architecture globale de notre système se décline en trois principales parties :

* Le frontend est basé sur la bibliothèque React.js, qui est un Framework JavaScript populaire pour la construction d'interfaces utilisateur. Il s'agit d'une bibliothèque JavaScript open-source qui facilite la création d'interfaces utilisateur réactives et interactives. Il utilise également le serveur d’applications NodeJS ainsi que le gestionnaire de paquets npm. Pour envoyer des requêtes HTTP depuis le frontend React.js, le code utilise généralement la fonction **fetch**, qui est une API JavaScript intégrée aux navigateurs modernes. **fetch** permet d'effectuer des requêtes HTTP vers des ressources distantes et de traiter les réponses. Cette API prend en charge les méthodes HTTP telles que GET, POST, PUT, DELETE ainsi que la gestion des en-têtes et des données. Lorsqu’une action est déclenchée dans l’interface utilisateur, par exemple un clic sur un bouton, le code JavaScript dans le frontend utilise fetch pour envoyer une requête http au backend. La requête peut contenir des données supplémentaires, telles que des paramètres de requête ou des données de formulaire, qui sont encodées au format JSON. Le backend traite ensuite la requête, effectue les opérations nécessaires et renvoie une réponse, qui est reçue par le frontend via fetch au format JSON.
* Notre backend est basé sur le Framework Flask. Il permet de gérer les requêtes HTTP, traiter les données et renvoyer des réponses au frontend. Dans Flask, nous avons défini des routes pour spécifier les URL auxquelles notre API répondra. Lorsque le frontend envoie une requête HTTP à notre backend Flask, la route correspondante est déclenchée. Nous avec spécifié pour chaque route le type de requête et dans la fonction de vue correspondante, il est possible d’accéder aux données de la requête, tels que les paramètres de requête, les données de formulaire, les en-têtes.

Avant de pouvoir interagir avec la base de données MySQL, nous avons établi une connexion grâce au package mysql.connector de Python.

* Une fois la connexion établie entre la base de données et le backend, nous exécutons des requêtes SQL sur la base de données. Cela inclut des opérations telles que la sélection (SELECT), l'insertion (INSERT), la mise à jour (UPDATE), la suppression (DELETE).

### Frontend

Une image contenant texte, capture d’écran, diagramme, Police

Description générée automatiquement

Figure 2 : Architecture du frontend

Le frontend est basé sur la bibliothèque React.js, qui est un Framework JavaScript populaire pour la construction d'interfaces utilisateur. Il s'agit d'une bibliothèque JavaScript open-source qui facilite la création d'interfaces utilisateur réactives et interactives. Il utilise également le serveur d’applications NodeJS ainsi que le gestionnaire de paquets npm. Pour envoyer des requêtes HTTP depuis le frontend React.js, le code utilise généralement la fonction fetch, qui est une API JavaScript intégrée aux navigateurs modernes. fetch permet d'effectuer des requêtes HTTP vers des ressources distantes et de traiter les réponses. Cette API prend en charge les méthodes HTTP telles que GET, POST, PUT, DELETE ainsi que la gestion des en-têtes et des données. Lorsqu’une action est déclenchée dans l’interface utilisateur, par exemple un clic sur un bouton, le code JavaScript dans le frontend utilise fetch pour envoyer une requête http au backend. La requête peut contenir des données supplémentaires, telles que des paramètres de requête ou des données de formulaire, qui sont encodées au format JSON. Le backend traite ensuite la requête, effectue les opérations nécessaires et renvoie une réponse, qui est reçue par le frontend via fetch au format JSON.

### Backend

Une image contenant texte, capture d’écran, Police, diagramme

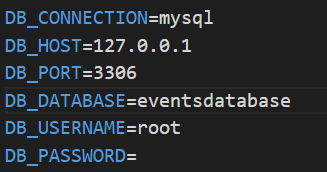
Description générée automatiquement

Figure 3 : Architecture du backend

Notre backend est basé sur le Framework Flask. Il permet de gérer les requêtes HTTP, traiter les données et renvoyer des réponses au frontend. Dans Flask, nous avons défini des routes pour spécifier les URL auxquelles notre API répondra. Lorsque le frontend envoie une requête HTTP à notre backend Flask, la route correspondante est déclenchée. Nous avec spécifié pour chaque route le type de requête et dans la fonction de vue correspondante, il est possible d’accéder aux données de la requête, tels que les paramètres de requête, les données de formulaire, les en-têtes.

### Base de données

Notre base de données est une base MySQL, avec 9 tables. Les paramètres de connexion à notre base de données sont :



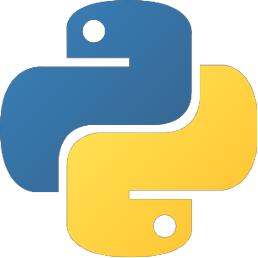
## PRESENTATION DES TECHNOLOGIES UTILISEES

### II.1. Le backend

#### II.1.1. Le langage de programmation Python

C’est aujourd’hui le langage de référence en programmation. Pour le développement de notre API nous avons opté pour l’utilisation du langage de programmation python car c’est celui qui nous a été imposé pour la réalisation de ce projet. La version de Python utilisée est la version 3.11

Figure 4 : Logo du langage Python

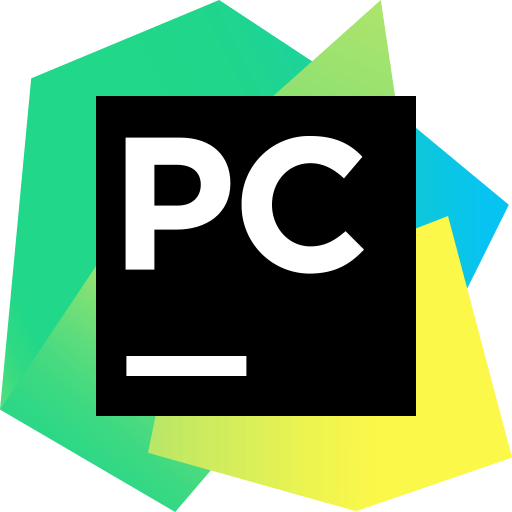


#### II.1.2. Le logiciel Pycharm

PyCharm est un environnement de développement intégré utilisé pour programmer en Python. Il permet l'analyse de code et contient un débogueur graphique. Il permet également la gestion des tests unitaires et l'intégration de logiciel de gestion de versions. Développé par l'entreprise tchèque JetBrains, c'est un logiciel multi-plateforme qui fonctionne sous Windows, Mac OS X et GNU/Linux. Il est décliné en édition professionnelle, diffusé sous licence propriétaire, et en édition communautaire diffusé sous licence Apache.

Nous avons utilisé le logiciel PyCharm pour coder notre API Python.

Figure 5 : Logo de Pycharm



#### II.1.3. Le logiciel Postman

Postman est une application permettant de tester des API, « créée en 2012 par Abhinav Asthana, Ankit Sobti et Abhijit Kane à Bangalore pour répondre à une problématique de test d'API partageable. D'abord module complémentaire de Google Chrome, puis client lourd, et finalement client léger, elle est à présent utilisée par plus de 500 000 entreprises dans le monde et a son siège à San Francisco ». Postman regroupe chaque test d'API dans une collection, permettant de mutualiser leurs URLs et authentifications.

Nous avons utilisé Postman pour réaliser les différents tests sur notre API. La version utilisée est la 9.25.0

Figure 6 : Logo de Postman



#### II.1.4. Le micro Framework Flask

Flask est un micro Framework open-source de développement web en Python. Il est classé comme micro Framework car il est très léger. Flask a pour objectif de garder un noyau simple mais extensible. Il n'intègre pas de système d'authentification, pas de couche d'abstraction de base de données, ni d'outil de validation de formulaires. Cependant, de nombreuses extensions permettent d'ajouter facilement des fonctionnalités. Flask nous a permis de créer l’application Python qui communique avec l’interface en servant de web services.

Une image contenant noir, obscurité

Description générée automatiquement

Figure 7 : Logo de Flask

#### II.1.5. Le format de données JSON

JSON est un format d'échange de données indépendant conçu pour représenter des structures de données simples. JSON est limité aux valeurs textuelles et numériques. Les valeurs binaires ne sont pas supportées. Les structures de données en JSON sont basées sur des paires clé / valeur.

Nous nous sommes basés sur ce format d’échange pour effectuer certaines requêtes d’insertion vers notre base MySQL et pour fournir des services à partir de notre application Python.

Figure 8 : Logo de JSON

Une image contenant cercle, Graphique, capture d’écran, art

Description générée automatiquement

### II.2. Le frontend React

#### II.2.1. La bibliothèque React

React est une bibliothèque JavaScript libre développée par Facebook (maintenant Meta) depuis 2013. Le but principal de cette bibliothèque est de faciliter la création d'application web monopage, via la création de composants dépendant d'un état et générant une page HTML à chaque changement d'état. Nous avons utilisé React pour coder notre frontend.

Une image contenant Graphique, cercle, art, symbole

Description générée automatiquement

Figure 9 : Logo de React

#### II.2.2. Le langage de programmation Javascript

JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives et à ce titre est une partie essentielle des applications web. Avec les langages HTML et CSS, JavaScript est au cœur des langages utilisés par les développeurs web. Une grande majorité des sites web l'utilisent, et la majorité des navigateurs web disposent d'un moteur JavaScript pour l'interpréter. JavaScript est aussi employé pour les serveurs Web avec l'utilisation de Node.js.

Figure 10 : Logo de Javascript

Une image contenant Police, Graphique, logo, conception

Description générée automatiquement

#### II.2.3. Le serveur d’applications Node.JS

Node.js est une plateforme logicielle libre en JavaScript, orientée vers les applications réseau évènementielles hautement concurrentes qui doivent pouvoir monter en charge. Parmi les modules natifs de Node.js, on retrouve http qui permet le développement de serveur HTTP. Ce qui autorise, lors du déploiement de sites internet et d'applications web développés avec Node.js, de ne pas installer et utiliser des serveurs web tels que Nginx ou Apache. Concrètement, Node.js est un environnement bas niveau permettant l'exécution de JavaScript côté serveur ce qui fait de Node.js notre serveur d’applications.

Figure 11 : Logo de Node.js

Une image contenant capture d’écran, Graphique, conception

Description générée automatiquement

#### II.2.4. L’outil npm

npm est le gestionnaire de paquets par défaut pour l'environnement d'exécution JavaScript Node.js. npm se compose d'un client en ligne de commande, également appelé npm, et d'une base de données en ligne de paquets publics et privés payants, appelée le registre npm. Le registre est accessible via le client, et les paquets disponibles peuvent être parcourus et recherchés via le site Web de npm. Le gestionnaire de paquets et le registre sont gérés par npm, Inc.

Une image contenant Graphique, rouge, capture d’écran, Police

Description générée automatiquement

Figure 12 : Logo de npm

#### II.2.5. Le package React-Bootstrap

Bootstrap est une collection d'outils utile à la création du design (graphisme, animation et interactions avec la page dans le navigateur, etc.) de sites et d'applications web. Il se base sur un système de grille simple et efficace pour mettre en ordre l'aspect visuel d'une page web. Il apporte du style aux différents composants d’une page web. Nous utilisé Bootstrap v5 à cause de sa simplicité d’utilisation. Il nous a été utile pour le design de notre plateforme web. Le package React-Bootstrap permet d’adapter les composants Bootstrap aux composants React.

Une image contenant symbole, Graphique, logo, Police

Description générée automatiquement

Figure 13 : Logo de React-Bootstrap

#### II.2.6. Le logiciel Visual Studio Code

Visual Studio Code est un éditeur de code extensible développé par Microsoft pour Windows, Linux et MacOs. Les fonctionnalités incluent la prise en charge du débogage, la mise en évidence de la syntaxe, la complétion intelligente du code, les snippets, la refactorisation du code et Git intégré. Nous avons utilisé le logiciel Visual Studio Code pour le code de notre plateforme web d’administration.

Figure 14 : Logo de Visual Studio Code

Une image contenant symbole, Bleu électrique, logo, ligne

Description générée automatiquement

### II.3. Le frontend Laravel

#### II.3.1. Le Framework Laravel

Laravel est un Framework écrit en PHP respectant le principe modèle-vue-contrôleur (MVC) et entièrement développé en programmation orientée objet. Laravel a été créé par Taylor Otwell en Juin 2011. Il a permis de créer l’application web d’administration car il est facile à prendre en main et permet un développement rapide d’applications grâce à son vaste écosystème d’outils. La version utilisée est Laravel 8.

Figure 15 : Logo de Laravel

Une image contenant motif, art, Symétrie, conception

Description générée automatiquement

#### II.3.2. Le client http Guzzle

Guzzle est un client HTTP PHP qui facilite l'envoi de requêtes HTTP et l'intégration aux services Web. La version de Guzzle utilisée lors de notre développement est la 7.0 ; installée à partir du gestionnaire de dépendance « Composer » de PHP. Nous avons utilisé Guzzle pour faire des requêtes HTTP vers notre web service à partir de notre plateforme d’administration.

Figure 16 : Logo de Guzzle

Une image contenant Police, Graphique, logo, graphisme

Description générée automatiquement

#### II.3.3. le Framework JQuery

jQuery est une bibliothèque JavaScript libre et multiplateforme créée pour faciliter l'écriture de scripts côté client dans le code HTML des pages web. La première version est lancée en janvier 2006 par John Resig. JQuery facilite la manipulation du code HTML, la gestion des évènements, l’animation ainsi que les fonctions AJAX grâce à sa bibliothèque compatible avec la plupart des navigateurs.

Figure 17 : Logo de JQuery

Une image contenant Graphique, Police, logo, graphisme

Description générée automatiquement

### II.4. La base de données

#### II.4.1. Le serveur web Apache

Apache HTTP Server (Apache) est un serveur HTTP créé et maintenu au sein de la fondation Apache. Jusqu'en avril 2019, ce fut le serveur HTTP le plus populaire du World Wide Web. Il est distribué selon les termes de la licence Apache. La version d’apache utilisée lors de notre développement est la 2.4.53; installée à partir du logiciel XAMPP. Il a permis de déployer en local notre application web d’administration.

Figure 18 : Logo de Apache

Une image contenant stylos et plumes, outil d’écriture, Graphique, conception

Description générée automatiquement

#### II.4.2. Le système de gestion de base de données MySQL

MySQL est un système de gestion de bases de données relationnelles (SGBDR). Il est distribué sous une double licence GPL et propriétaire. Il fait partie des logiciels de gestion de base de données les plus utilisés au monde. Il permet de gérer des bases de données relationnelles.

Figure 19 : Logo de MySQL

Une image contenant Graphique, Police, graphisme, logo

Description générée automatiquement

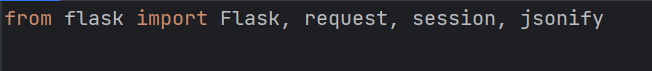
## MISE EN PLACE DE L’ENVIRONNEMENT DE TRAVAIL

### III.1. Processus d’installation des Framework

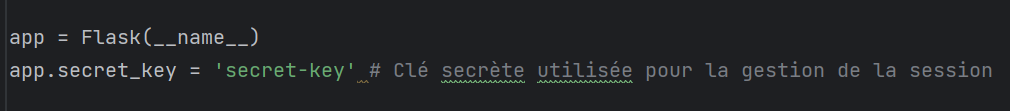
#### III.1.1. Le micro Framework Flask

Le micro Framework Flask se comporte comme un package Python, ce qui simplifie son installation. Il peut s’installer directement dans Pycharm dans l’onglet Python Packages ou dans le terminal python grâce à la commande : $ pip install Flask

Après l’installation, nous importons le package Flask ainsi que les composants dont nous avons besoin grâce à la ligne :



Nous créons ensuite notre application Flask avec la ligne :



Ensuite nous configurons la base de données dans notre application python. Pour ce faire, nous installons le package ***mysql-connector*** qui est le pilote MySQL de Python. Ce package s’installe grâce à la commande : $ pip install mysql-connector

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

L’importation du package se fait grâce à la ligne :

Enfin, nous allons établir la connexion à la base de données grâce à la commande :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Il suffit maintenant de créer nos routes et d’écrire les fonctions correspondantes à ces routes.

#### III.1.2. Le Framework Laravel

Le Framework Laravel s’installe de multiples manières. Nous avons opté la manière la plus simple qui est de passer par le logiciel Composer. La dernière version disponible est la v2.5.5. Le logiciel Composer peut être téléchargé via le lien : <https://getcomposer.org/download/> . Cependant, avant d’être installé, il nécessite que PHP soit installé. Alors on peut installer PHP ainsi que le serveur Apache directement en installant le logiciel Xampp via le lien : <https://www.apachefriends.org/fr/download.html>

##### Processus d’installation de Xampp

Au lancement du fichier .exe, nous avons la fenêtre ci-dessous qui apparaît :

Figure 20 : Etape 1 : Installation de Xampp

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

Nous cliquons sur « Next » pour continuer. La fenêtre suivante apparaît et nous sélectionnons les serveurs Web que nous désirons installer grâce à Xampp. Pour ce projet, nous pouvons sélectionner uniquement Apache et MySQL mais il est recommandé de cocher tous les serveurs pour ne pas avoir à procéder à une autre installation lorsque nous aurons besoin des autres serveurs pour un autre projet. Nous cochons donc toutes les cases comme l’indique l’image ci-dessous :

Figure 21 : Etape 2 : Installation de Xampp

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

Dans la capture suivante, nous sélectionnons le chemin d’installation de xampp. Pour notre projet, nous avons utilisé le chemin par défaut proposé par le logiciel.

Figure 22 : Etape 3 : Installation de Xampp

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

La suite de l’installation se poursuit en cliquant sur le bouton « Next ».

##### Processus d’installation de Composer

Après avoir installé le logiciel Xampp avec les serveurs Apache, MySQL et PHP, nous allons maintenant installer Composer. Au lancement du fichier exécutable, nous avons la fenêtre ci-dessous qui apparaît :

Figure 23 : Etape 1 : Installation de Composer

Une image contenant texte, capture d’écran, logiciel, Page web

Description générée automatiquement

En cliquant sur le bouton « Next » nous arrivons sur la deuxième capture où nous indiquons le chemin du PHP que nous avons installé grâce à Xampp. Dans notre projet, il s’agit du chemin par défaut.

Figure 24 : Etape 2 : Installation de Composer

Une image contenant texte, Appareils électroniques, capture d’écran, logiciel

Description générée automatiquement

##### Processus d’installation de Laravel

Avec Composer et PHP installés, nous allons maintenant procéder à l’installation du logiciel Laravel en créant notre application Laravel grâce à la commande :

composer create-project laravel/events

Dans le cas où le projet (l’application) existe déjà et quelle a été clonée de GitHub par exemple, il suffit de faire composer require laravel pour installer Laravel.

Nous allons ensuite configurer notre environnement de développement. Pour cela il faut créer une copie du fichier ***.env.example*** en exécutant la commande : cp .env.example .env.

Nous allons ensuite nommer la copie ***.env*** puis nous allons remplir ce nouveau fichier avec nos paramètres de configuration de la base de données :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Laravel exige que une clé d’encryptions pour chaque application, ceci est généralement généré de manière aléatoire et stocké dans le fichier ***.env***. Laravel utilisera cette clé d’encryptions pour encoder divers éléments de notre application web, tels que les cookies, le hachage de mots de passe et bien d’autres éléments. Pour générer cette clé il nous faut exécuter la commande : php artisan key :generate

Enfin, nous pouvons exécuter notre application Laravel grâce à la commande : php artisan serve

#### III.1.3. Le Framework React

##### Installation de NodeJS

##### Création de l’application React

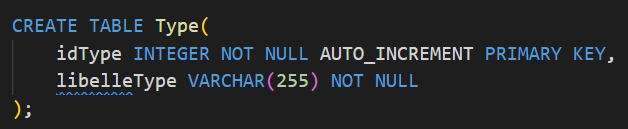
#### III.1.4. La base de données MySQL

Pour créer notre base de données MySQL, nous exécutons les commandes suivantes :

* Script de création de la base de données



* Création de la table Type

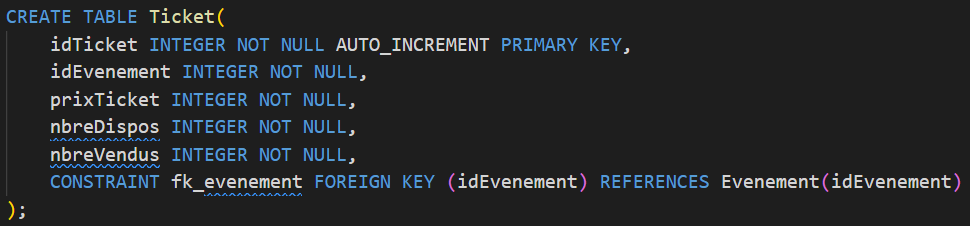


* Création de la table Evènement

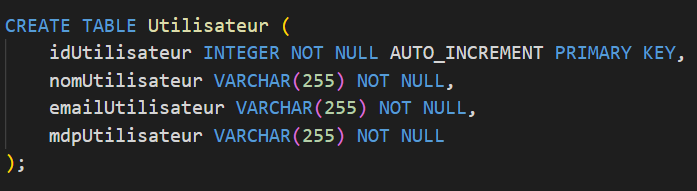
Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

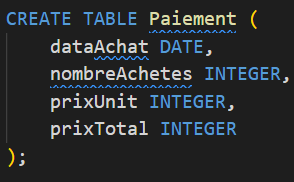
* Création de la table Ticket



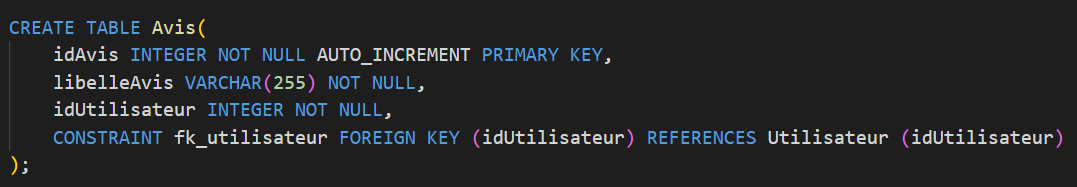
* Création de la table Utilisateur



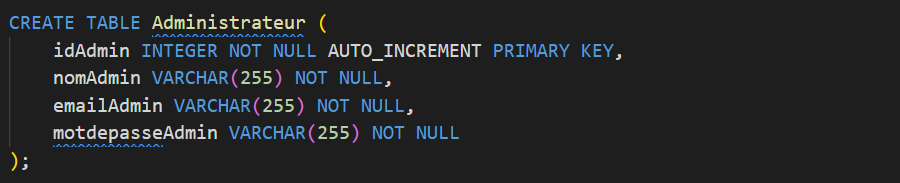
* Création de la table Paiement



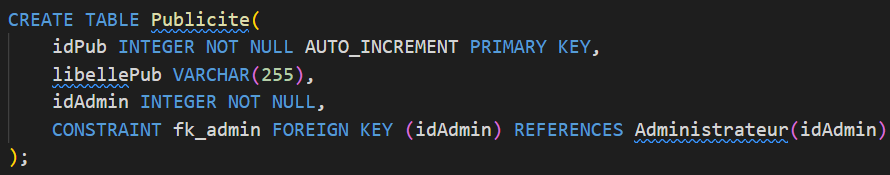
* Création de la table Avis



* Création de la table Administrateur



* Création de la table Publicité



* Création de la table Newsletter

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

## REALISATION

### IV.1. Arborescence globale du projet

#### IV.1.1. Arborescence du backend

Figure 25 : Arborescence du backend

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Le fichier ***main.py*** est le fichier principal de notre application Python. Il contient toutes les fonctions nécessaires au bon déroulement de notre application.

#### IV.1.2. Arborescence du frontend

Figure 26 : Arborescence du frontend

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Dans notre frontend nous avons plusieurs dossiers dont les plus importants sont : app, public, resources, routes, storage.

##### Le dossier app

Il contient les dossiers : Console, Exceptions, Http, Models et Providers. Le dossier http contient les dossier Controllers et Middleware.

Figure 27 : Détails du dossier app

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Dans le dossier Controllers nous avons les différents controllers de notre projet. Les controllers contiennent les fonctions qui permettent aux composants de notre plateforme web de communiquer avec notre application Web.

##### Le dossier public

Il contient plusieurs dossiers dont les dossiers build, dist et plugins. Le dossier dist contient les dossiers css, js et img qui contiennent respectivement les fichiers de mise en forme css et javascript et les images de notre application web. Le dossier build contient les configurations javascript, scss, npm utilisées dans tout notre projet.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Figure 28 : Détails du dossier public

##### Le dossier resources

Il contient les dossiers css, js, lang et views. Le dossier lang contient toutes les configurations linguistiques telles que les messages d’erreur ou les notifications de l’application web. Le dossier views contient toutes les vues de notre application web.

Figure 29 : Détails du dossier resources

Une image contenant texte, capture d’écran, Police, conception

Description générée automatiquement

##### Le dossier routes

Le dossier routes contient quatre fichiers dont le fichier web. Ce fichier contient toutes les routes nécessaires pour les communications entre notre plateforme web et l’application Python.

Figure 30 : Détails du dossier routes

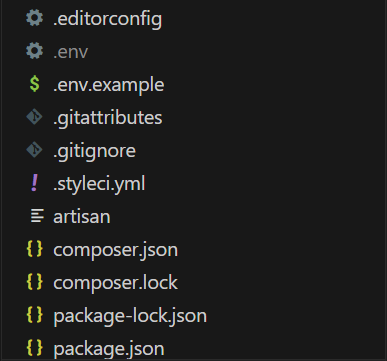
Une image contenant texte, capture d’écran, Police, violette

Description générée automatiquement

##### Les autres fichiers

Les autres fichiers importants sont le fichier .env qui contient les configurations de la base de données ainsi que les configurations de base de notre application web ; le fichier artisan qui permet d’exécuter notre application web et le fichier composer.

Figure 31 : Les autres fichiers



### IV.2. Présentation des interfaces

#### IV.2.1. Page d’accueil du site web

##### Page d’accueil REACT

##### Page d’accueil Laravel

#### IV.2.2. Tableau de bord de l’administrateur

##### Tableau de bord REACT

##### Tableau de bord Laravel