**SOMMAIRE**

[INTRODUCTION 5](#_Toc135769071)

[I. ARCHITECTURE DU PROJET 5](#_Toc135769072)

[I.1. Architecture globale 5](#_Toc135769073)

[I.2. Frontend 6](#_Toc135769074)

[I.3. Backend 7](#_Toc135769075)

[I.4. Base de données 7](#_Toc135769076)

[II. PRESENTATION DES TECHNOLOGIES UTILISEES 11](#_Toc135769077)

[II.1. Le langage de programmation Python 11](#_Toc135769078)

[II.2. Le logiciel Pycharm 11](#_Toc135769079)

[II.3. Le logiciel Postman 12](#_Toc135769080)

[II.4. Le micro Framework Flask 13](#_Toc135769081)

[II.5. Le Framework Laravel 13](#_Toc135769082)

[II.6. Le client http Guzzle 14](#_Toc135769083)

[II.7. Le format de données JSON 14](#_Toc135769084)

[II.8. Le framework CSS Bootstrap 15](#_Toc135769085)

[II.9. Le framework JQuery 15](#_Toc135769086)

[II.10. La librairie ApexCharts 16](#_Toc135769087)

[II.11. Le logiciel Visual Studio Code 16](#_Toc135769088)

[II.12. Le serveur web Apache 17](#_Toc135769089)

[II.13. Le système de gestion de base de données MySQL 17](#_Toc135769090)

[II.14. L’outil de versionnage Git 18](#_Toc135769091)

[II.15. Le Paas Github 18](#_Toc135769092)

[III. MISE EN PLACE DE L’ENVIRONNEMENT DE TRAVAIL 19](#_Toc135769093)

[III.1. Processus d’installation des Framework 19](#_Toc135769094)

[IV. REALISATION 24](#_Toc135769095)

[IV.1. Arborescence globale du projet 24](#_Toc135769096)

[IV.2. Présentation des interfaces 28](#_Toc135769097)

**LISTE DES FIGURES**

[**Figure 1 : Architecture globale du projet** 5](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769098)

[**Figure 2 : Architecture du frontend** 6](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769099)

[**Figure 3 : Architecture du backend** 7](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769100)

[**Figure 4 : Logo du langage Python** 11](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769101)

[**Figure 5 : Logo de Pycharm** 12](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769102)

[**Figure 6 : Logo de Postman** 12](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769103)

[**Figure 7 : Logo de Flask** 13](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769104)

[**Figure 8 : Logo de Laravel** 13](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769105)

[**Figure 9 : Logo de Guzzle** 14](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769106)

[**Figure 10 : Logo de JSON** 14](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769107)

[**Figure 11 : Logo de Bootstrap** 15](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769108)

[**Figure 12 : Logo de JQuery** 15](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769109)

[**Figure 13 : Logo de ApexCharts** 16](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769110)

[**Figure 14 : Logo de Visual Studio Code** 16](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769111)

[**Figure 15 : Logo de Apache** 17](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769112)

[**Figure 16 : Logo de MySQL** 17](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769113)

[**Figure 17 : Logo de Git** 18](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769114)

[**Figure 18 : Logo de GitHub** 18](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769115)

[**Figure 19 : Etape 1 : Installation de Xampp** 20](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769116)

[**Figure 20 : Etape 2 : Installation de Xampp** 21](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769117)

[**Figure 21 : Etape 3 : Installation de Xampp** 21](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769118)

[**Figure 22 : Etape 1 : Installation de Composer** 22](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769119)

[**Figure 23 : Etape 2 : Installation de Composer** 22](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769120)

[**Figure 24 : Arborescence du backend** 24](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769121)

[**Figure 25 : Arborescence du frontend** 25](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769122)

[**Figure 26 : Détails du dossier app** 26](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769123)

[**Figure 27 : Détails du dossier public** 26](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769124)

[**Figure 28 : Détails du dossier resources** 27](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769125)

[**Figure 29 : Détails du dossier routes** 27](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769126)

[**Figure 30 : Les autres fichiers** 28](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%201.docx#_Toc135769127)

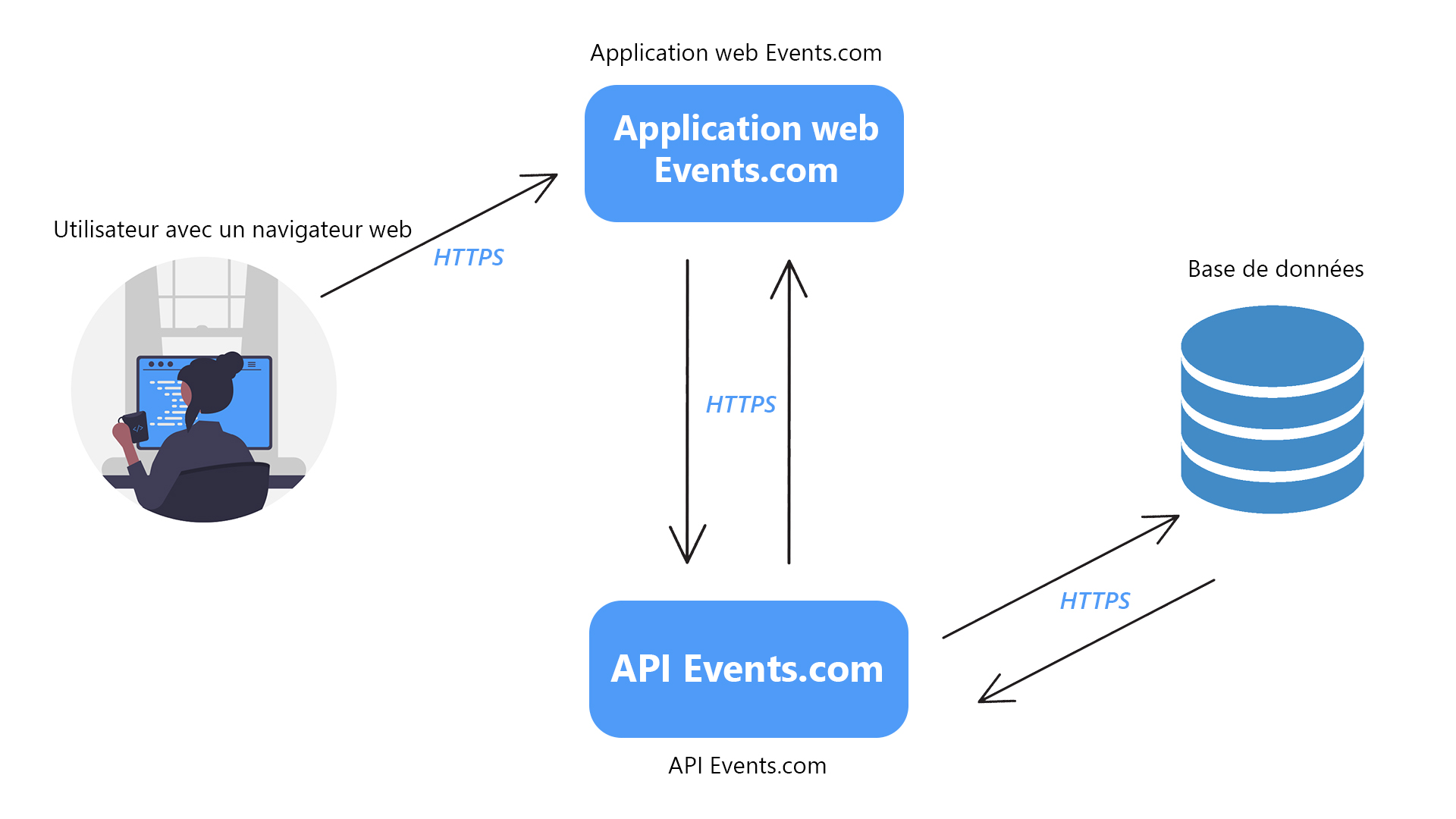
# INTRODUCTION

## ARCHITECTURE DU PROJET

### Architecture globale

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

Figure 1 : Architecture globale du projet



Pour notre développement, nous utilisons l’architecture distribuée de type trois tiers. L’architecture trois tiers, également appelée architecture à trois niveaux ou à trois couches, est une architecture client-serveur dans laquelle coexistent et sont maintenus des modules indépendants permettant le rendu d'une interface utilisateur, les « process » logiques, fonctionnels et métiers ainsi que l'accès aux données. Notre choix s’est donc porté sur ce type d’architecture car nous voulons rendre notre projet très modulable et facilement maintenable. L’architecture se dessine donc en trois principales parties :

* **La présentation des données** : correspond à l’affichage, la restitution sur le poste de travail, le dialogue avec l’utilisateur. Elle est assurée par un poste client et un navigateur web.
* **Le traitement métier des données** : correspond à la mise en œuvre de l’ensemble des règles de gestion et de la logique applicative. Le traitement métier est géré par un serveur applicatif ;
* **L’accès aux données persistantes** : correspond aux données qui sont destinées à être conservées sur une durée voire de manière définitive. L’accès aux données se fait grâce au serveur de base de données MySQL.

### Frontend

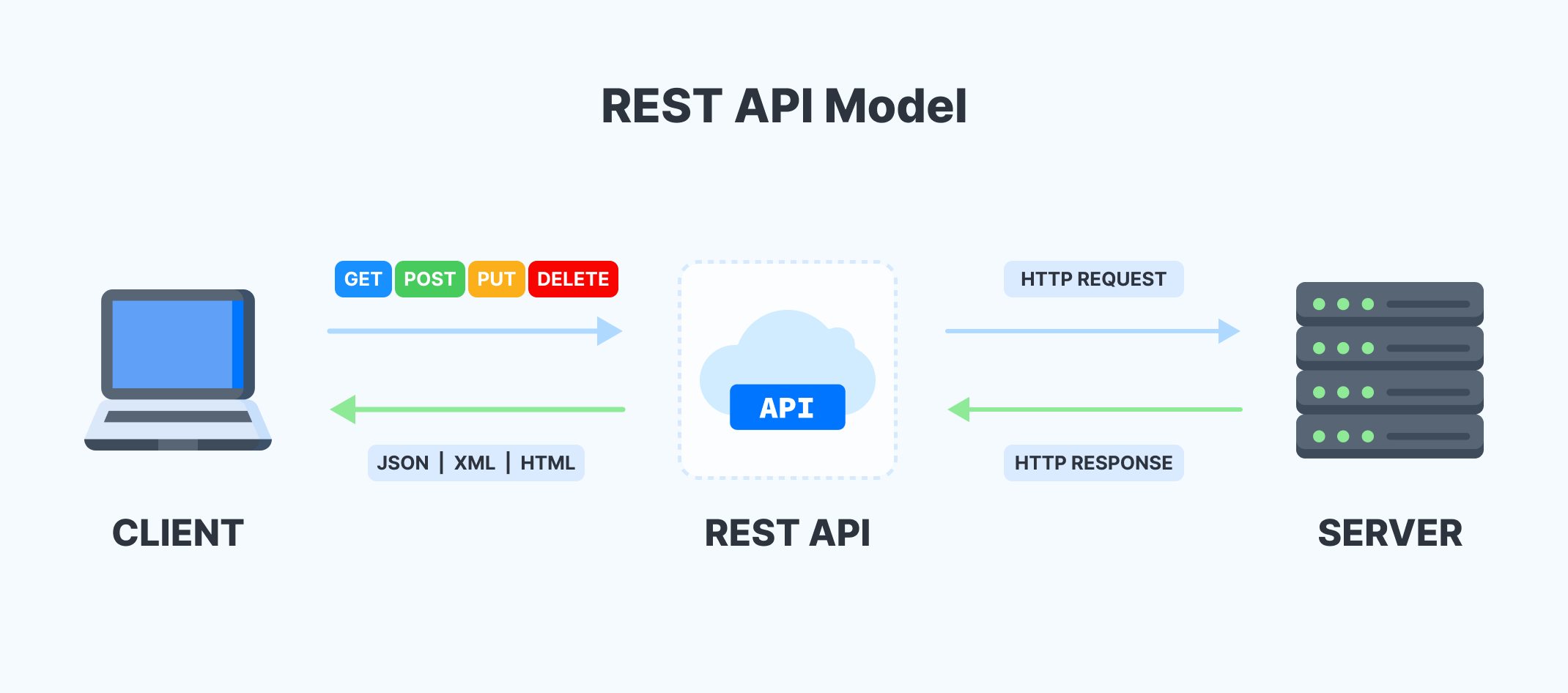
Figure 2 : Architecture du frontend



Le frontend de notre projet consiste en une plateforme web basée utilisant le Framework Laravel. L’architecture utilisée est l’architecture MVC (modèle-vue-contrôleur) qui est un modèle destiné à répondre aux besoins des applications interactives en séparant les problématiques liées aux différents composants au sein de leur architecture respective.

### Backend

Figure 3 : Architecture du backend

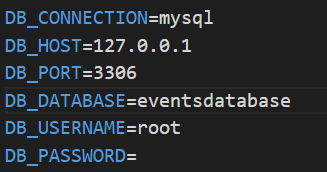


Pour notre backend, nous avons utilisé une architecture REST qui est un ensemble de contraintes permettant de créer des services web. Ces contraintes sont, entre autres, l’exécution de requêtes GET, POST, PUT et DELETE entre les différents composants du système.

### Base de données

Notre base de données est une base MySQL, avec 9 tables.

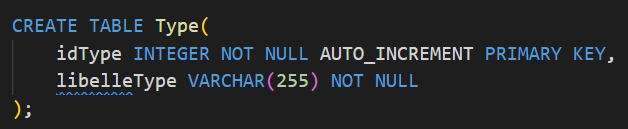
Connexion à la base de données :



Script de création de la base de données :



* Table Type

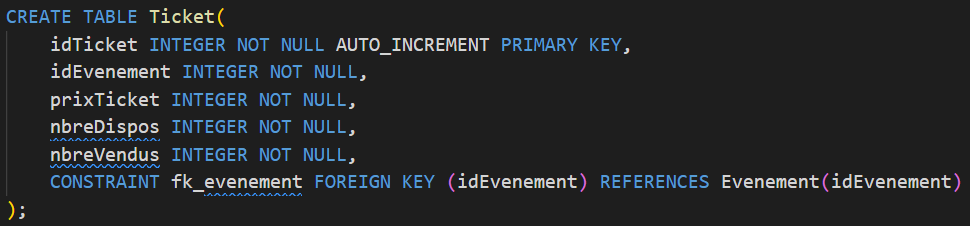


* Table Evènement

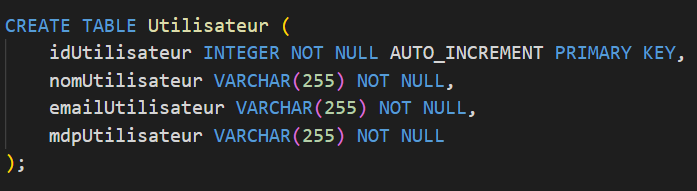
Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

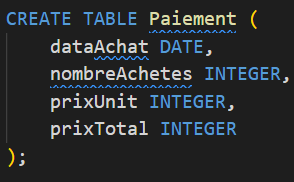
* Table Ticket



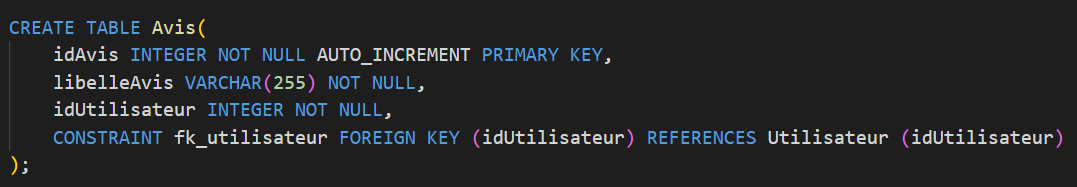
* Table Utilisateur



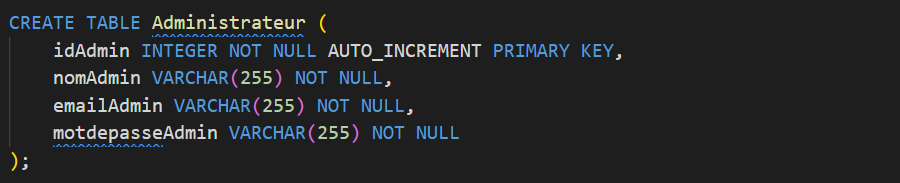
* Table Paiement



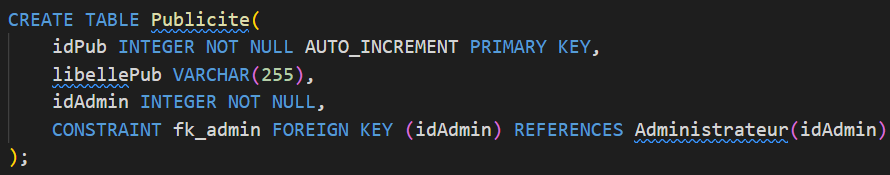
* Table Avis



* Table Administrateur



* Table Publicité



* Table Newsletter

Une image contenant texte, capture d’écran, Police

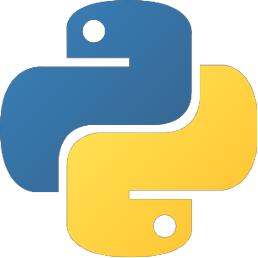
Description générée automatiquement

## PRESENTATION DES TECHNOLOGIES UTILISEES

### II.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

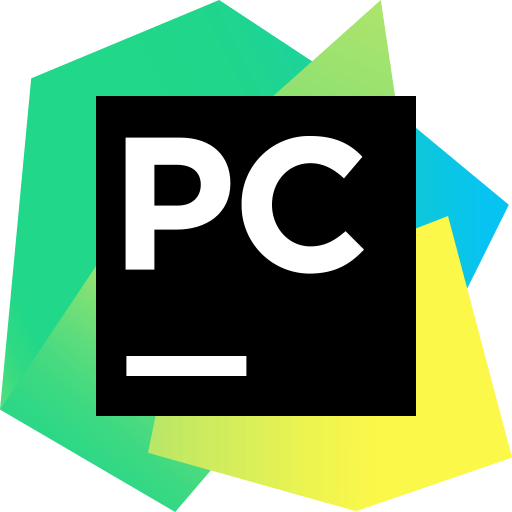


### II.2. Le logiciel Pycharm

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, l'intégration de logiciel de gestion de versions, et supporte le développement web avec Django. 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 application Python.

Figure 5 : Logo de Pycharm



### II.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.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.5. 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 8 : Logo de Laravel

### II.6. 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.

Une image contenant Police, Graphique, logo, graphisme

Description générée automatiquement

Figure 9 : Logo de Guzzle

### II.7. 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 10 : Logo de JSON

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

Description générée automatiquement

### II.8. Le framework CSS 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.

Figure 11 : Logo de Bootstrap

Une image contenant Graphique, graphisme, symbole, Police

Description générée automatiquement

### II.9. 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.

Une image contenant Graphique, Police, logo, graphisme

Description générée automatiquement

Figure 12 : Logo de JQuery

### II.10. La librairie ApexCharts

ApexCharts est une bibliothèque de graphiques moderne qui aide les développeurs à créer de belles visualisations interactives pour les pages Web. Il s'agit d'un projet open source sous licence MIT et est libre d'utilisation dans des applications commerciales. Nous utilisons ApexCharts 3.21.0 pour mettre en forme des graphiques pour l’administrateur.

Figure 13 : Logo de ApexCharts

Une image contenant triangle, Caractère coloré, ligne

Description générée automatiquement

### II.11. 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.12. 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 15 : Logo de Apache

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

Description générée automatiquement

### II.13. 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 16 : Logo de MySQL

Une image contenant Graphique, Police, graphisme, logo

Description générée automatiquement

### II.14. L’outil de versionnage Git

Git est un logiciel décentralisé de gestion de versions. C'est un logiciel libre créé par Linus Torvald, auteur du noyau Linux, et distribué selon les termes de la licence publique générale GNU version 2. En 2016, il s’agit du logiciel de gestion de versions le plus populaire qui est utilisé par plus de douze millions de personnes. Il nous a permis de mettre en place les versions des codes sources et documents de notre application.

Figure 17 : Logo de Git

Une image contenant Panneau de signalisation, symbole, signe

Description générée automatiquement

### II.15. Le Paas Github

GitHub est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git. GitHub est centré vers l'aspect social du développement. Le site assure également un contrôle d'accès et des fonctionnalités destinées à la collaboration comme le suivi des bugs, les demandes de fonctionnalités, la gestion de tâches et un wiki pour chaque projet. Il nous a permis d’effectuer le versionnage et de faciliter le suivi et le développement du projet.

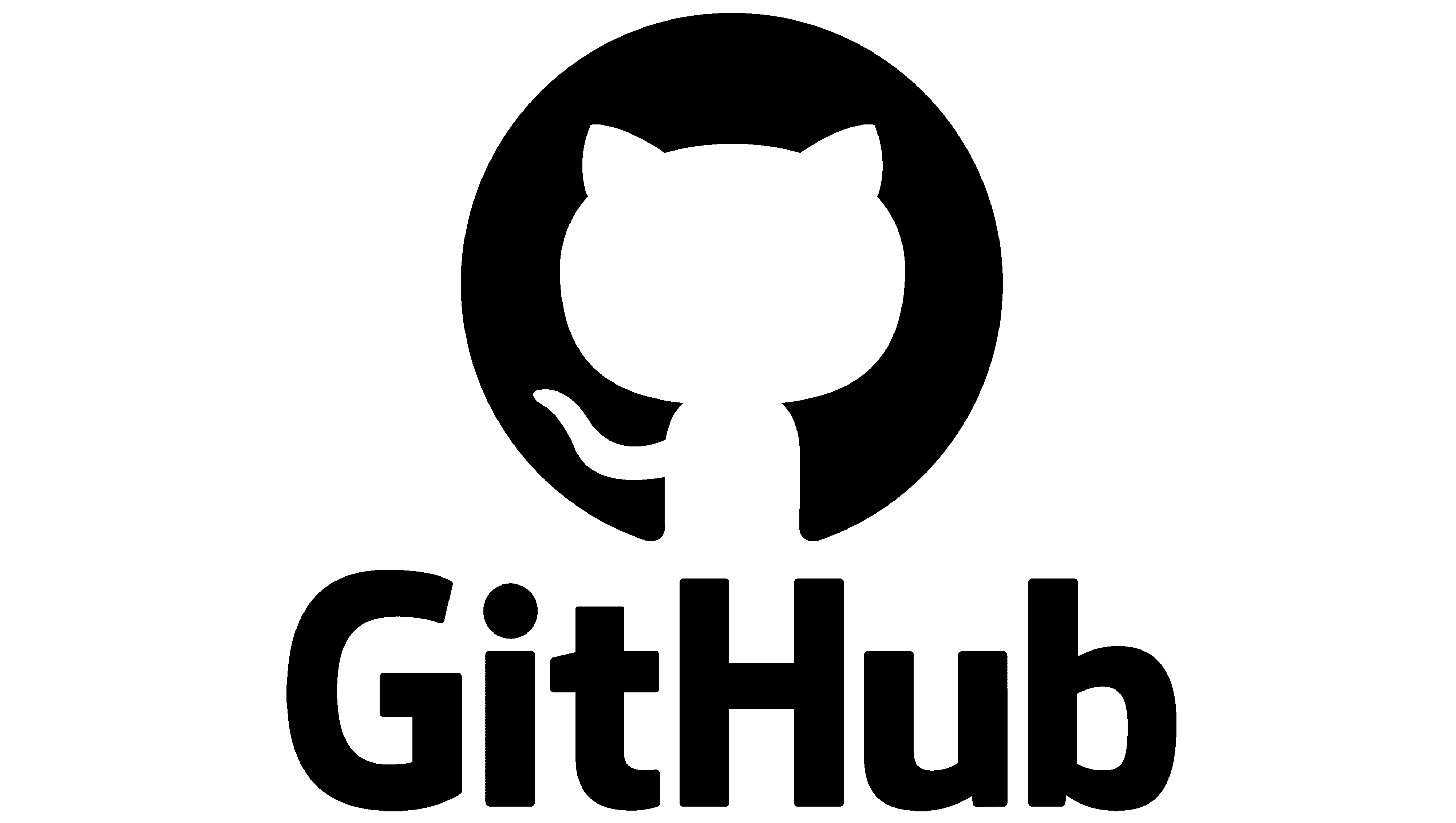


Figure 18 : Logo de GitHub

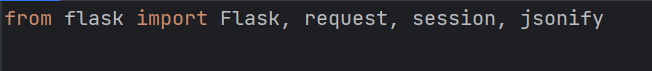
## MISE EN PLACE DE L’ENVIRONNEMENT DE TRAVAIL

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

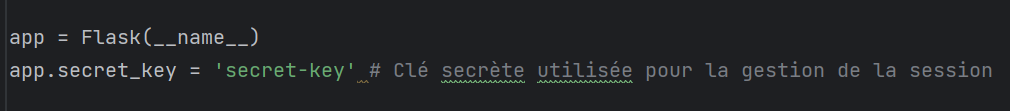
#### III.1.1. Le micro Framework Flask

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 19 : 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 20 : 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 21 : 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 22 : 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 23 : 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. La base de données MySQL

Pour créer notre base de données MySQL, nous exécutons les commandes précédemment citées dans la partie I.4 de ce document.

## REALISATION

### IV.1. Arborescence globale du projet

#### IV.1.1. Arborescence du backend

Figure 24 : 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 25 : 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 26 : 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 27 : 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 28 : 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 29 : Détails du dossier routes

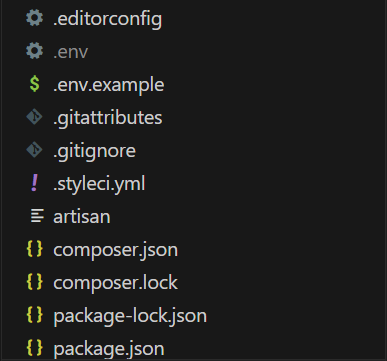
Une image contenant texte, capture d’écran, Police, violette

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 30 : Les autres fichiers



### IV.2. Présentation des interfaces

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