**SOMMAIRE**

[INTRODUCTION 3](#_Toc134981757)

[I. ARCHITECTURE DU PROJET 3](#_Toc134981758)

[I.1. Architecture globale 3](#_Toc134981759)

[I.2. Frontend 4](#_Toc134981760)

[I.3. Backend 5](#_Toc134981761)

[I.4. Base de données 5](#_Toc134981762)

[II. PRESENTATION DES TECHNOLOGIES UTILISEES 9](#_Toc134981763)

[III. MISE EN PLACE DE L’ENVIRONNEMENT DE TRAVAIL 12](#_Toc134981764)

[IV. REALISATION 12](#_Toc134981765)

**LISTE DES FIGURES**

[Figure 1 : Architecture globale du projet 3](file:///C:\Users\KOMISSA%20ZOTSU%20SHINER\Documents\GitHub\ProjetTO52\Documents\DOCUMENT%20TECHNIQUE%20Version%201.docx#_Toc134981729)

[Figure 2 : Architecture du frontend 4](file:///C:\Users\KOMISSA%20ZOTSU%20SHINER\Documents\GitHub\ProjetTO52\Documents\DOCUMENT%20TECHNIQUE%20Version%201.docx#_Toc134981730)

[Figure 3 : Architecture du backend 5](file:///C:\Users\KOMISSA%20ZOTSU%20SHINER\Documents\GitHub\ProjetTO52\Documents\DOCUMENT%20TECHNIQUE%20Version%201.docx#_Toc134981731)

[Figure 4 : Logo du langage Python 9](file:///C:\Users\KOMISSA%20ZOTSU%20SHINER\Documents\GitHub\ProjetTO52\Documents\DOCUMENT%20TECHNIQUE%20Version%201.docx#_Toc134981732)

[Figure 5 : Logo de Pycharm 10](file:///C:\Users\KOMISSA%20ZOTSU%20SHINER\Documents\GitHub\ProjetTO52\Documents\DOCUMENT%20TECHNIQUE%20Version%201.docx#_Toc134981733)

[Figure 6 : Logo de Postman 10](file:///C:\Users\KOMISSA%20ZOTSU%20SHINER\Documents\GitHub\ProjetTO52\Documents\DOCUMENT%20TECHNIQUE%20Version%201.docx#_Toc134981734)

[Figure 7 : Logo de Laravel 11](file:///C:\Users\KOMISSA%20ZOTSU%20SHINER\Documents\GitHub\ProjetTO52\Documents\DOCUMENT%20TECHNIQUE%20Version%201.docx#_Toc134981735)

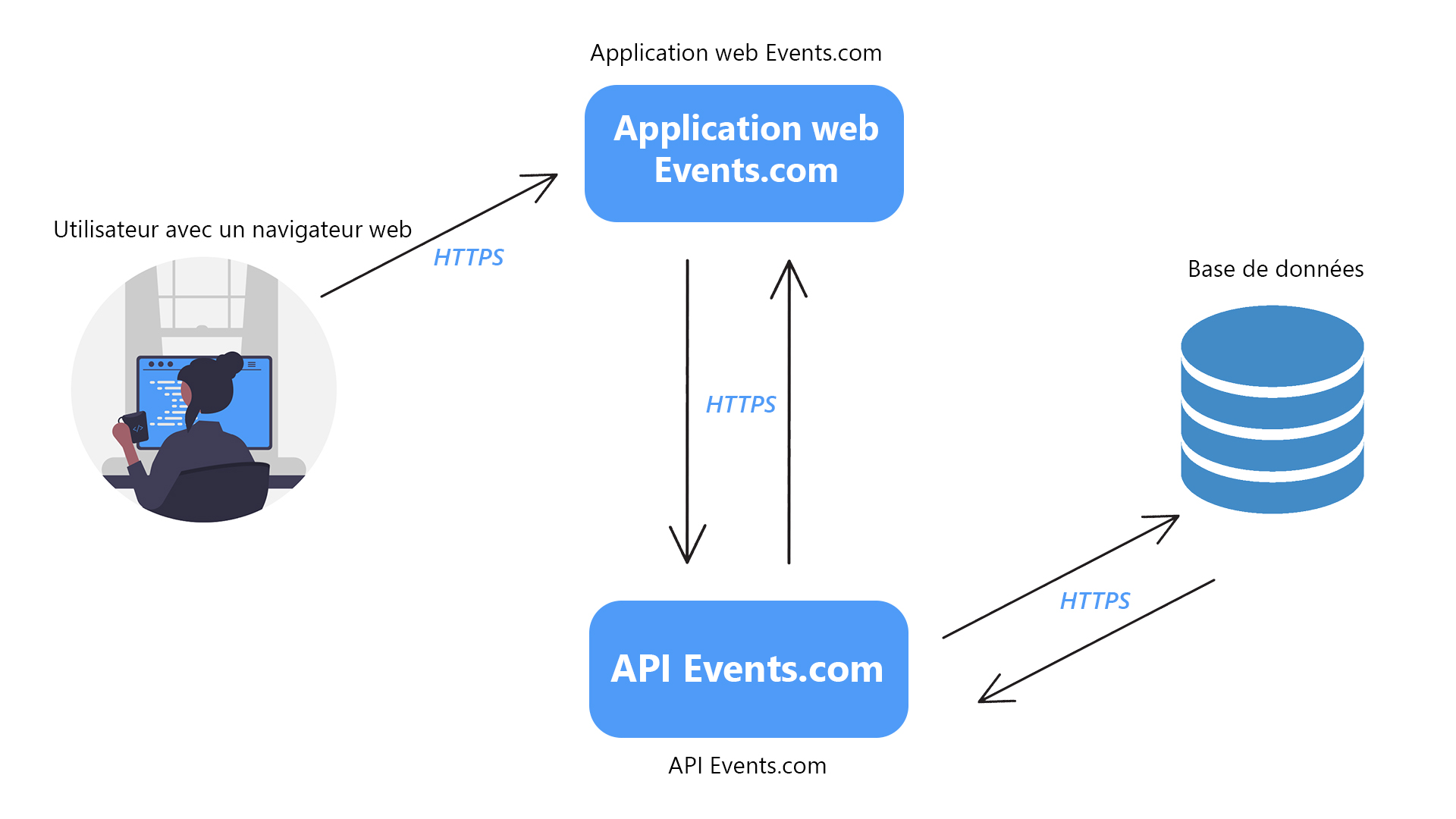
# INTRODUCTION

## 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

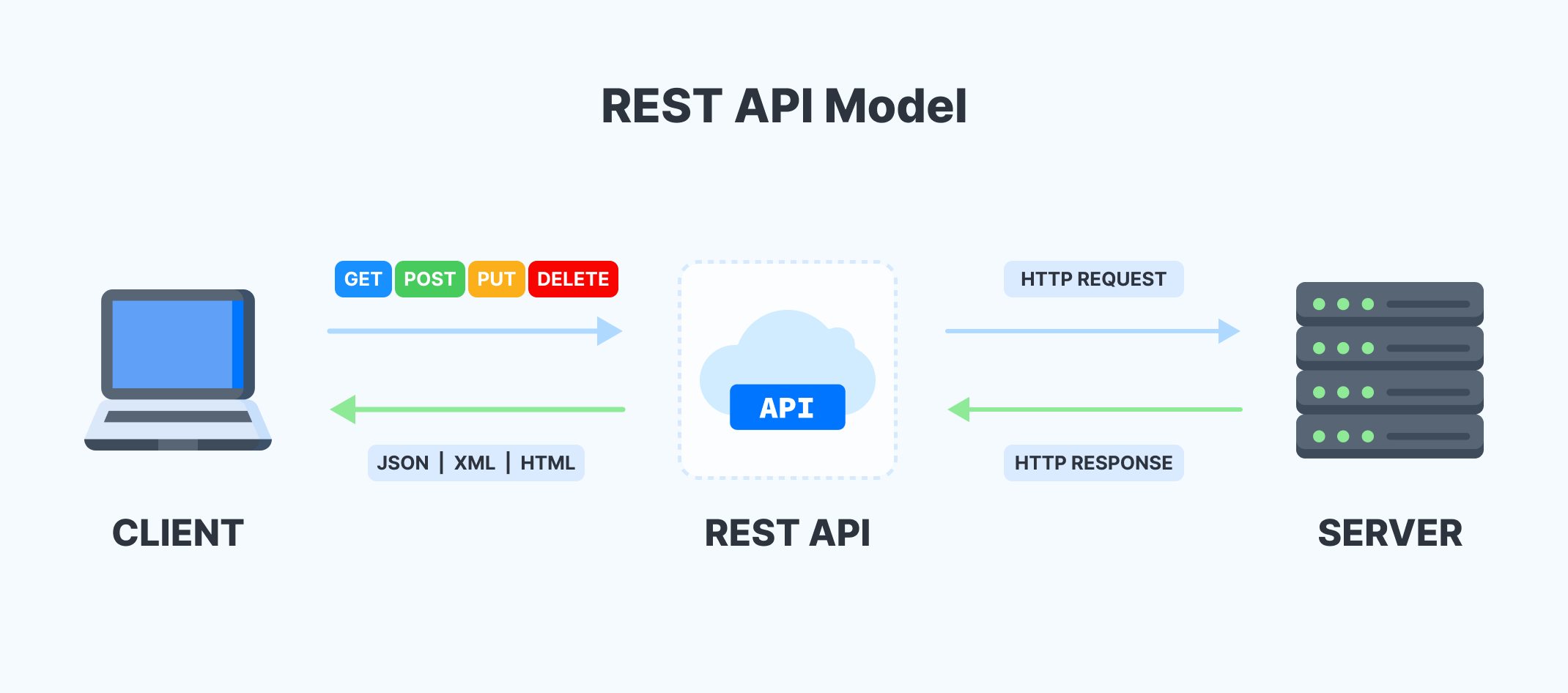
Figure 2 : Architecture du frontend



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

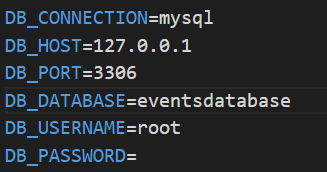


Pour notre backend, nous avons utilisé une architecture REST qui est un ensemble de contraintes permettant de créer des services web. Ces contraintes sont, entre autres, l’exécution de requêtes GET, POST, PUT et DELETE entre les différents composants du système.

### Base de données

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

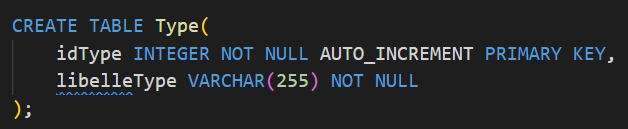
Connexion à la base de données :



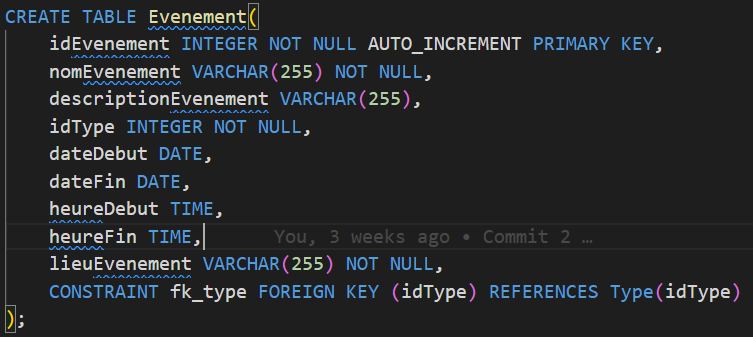
Script de création de la base de données :



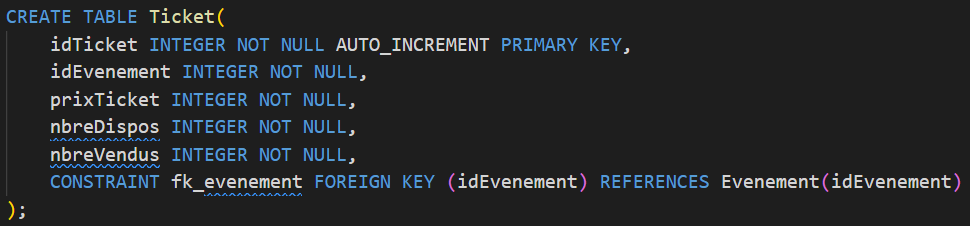
* Table Type



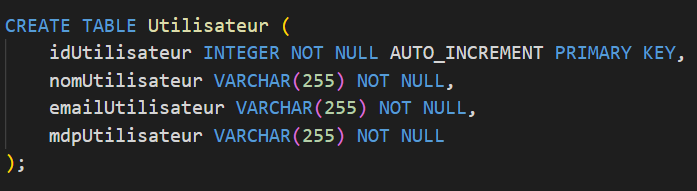
* Table Evènement



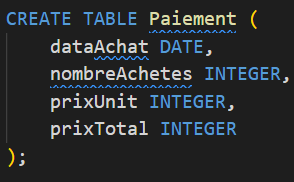
* Table Ticket



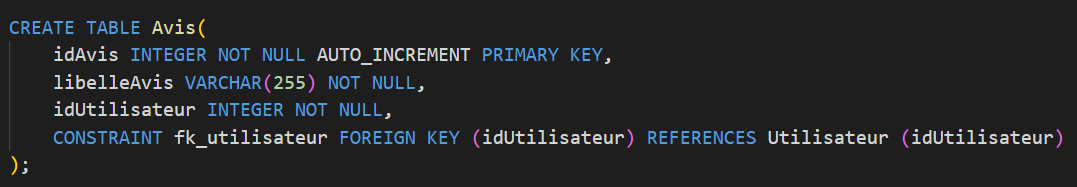
* Table Utilisateur



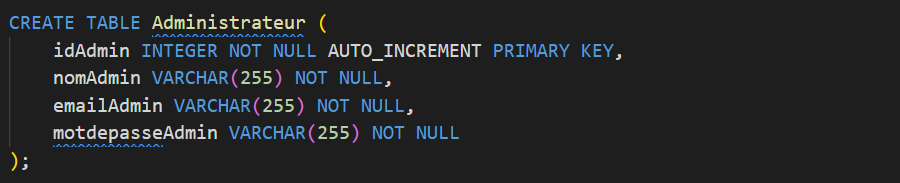
* Table Paiement



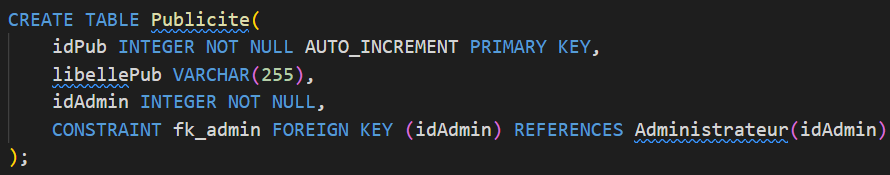
* Table Avis



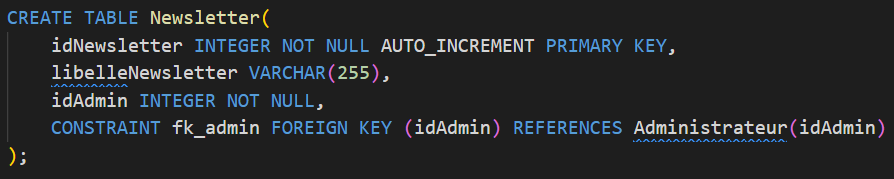
* Table Administrateur



* Table Publicité



* Table Newsletter

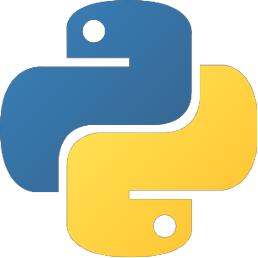


## PRESENTATION DES TECHNOLOGIES UTILISEES

#### Le langage de programmation Python

C’est aujourd’hui le langage de référence en programmation. Pour le développement de notre système 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 3.10

Figure 4 : Logo du langage Python

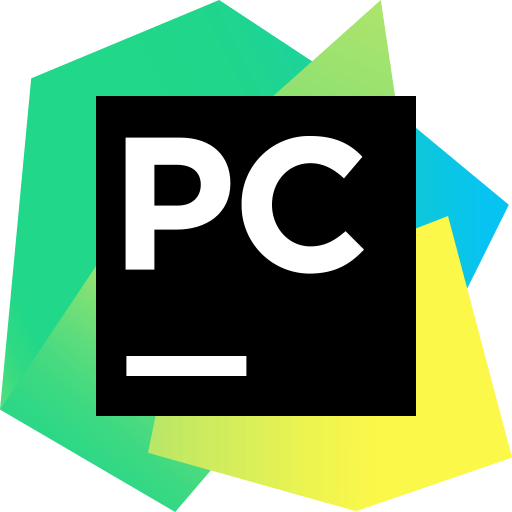


##### Le logiciel Pycharm

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 API Python.

Figure 5 : Logo de Pycharm



##### 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



##### 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 7 : Logo de Laravel



##### 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.

## MISE EN PLACE DE L’ENVIRONNEMENT DE TRAVAIL

## REALISATION