

**Documentation technique: Projet de gestion de ticketing en ligne**



* Ange Jessica Djahan
* Franklin Essonno

**Table de matiéres**

[**1.Introduction 2**](#_8kta6mjcp1i0)

[**2.Architecture technique 2**](#_2qufpr9dr90k)

[2.1 Modèle (Model) 3](#_wmzuwvynemsy)

[2.2 Vue (View) 3](#_2sksfvsal13o)

[2.3 Contrôleur (Controller) 4](#_pqzn4lkvcz9j)

[**3.Structure du projet 4**](#_iia9vtrxycq7)

[3.1 /Classe 4](#_i45ku8ugsh83)

[3.2 /Controller 4](#_uqwcyj4ffj4i)

[3.3 /Model 5](#_7xw0oaxvhjy7)

[3.4 /Vendor 5](#_fvc2rvoq3was)

[3.5 /view 5](#_dlk2ak3rv3bi)

[**4.Base de données 6**](#_49pkkl4289fx)

[4.1 Schéma de la base de données 6](#_h7v4979inp8c)

[4.2 Relations entre les tables et de la façon dont elles sont utilisées dans le projet 6](#_dgndt0j6k5ty)

[**5.Spécifications fonctionnelles 7**](#_630orze3ae8h)

[**6.Environnement de travail 9**](#_69m8y1khhghh)

[6.1 Framework Bootstrap 9](#_2mwh8n668pwy)

[6.2 PDO (PHP Data Objects) 9](#_5zk0jignde4t)

[6.3 Outil de versioning Git/GitHub 9](#_azai7qgnjuna)

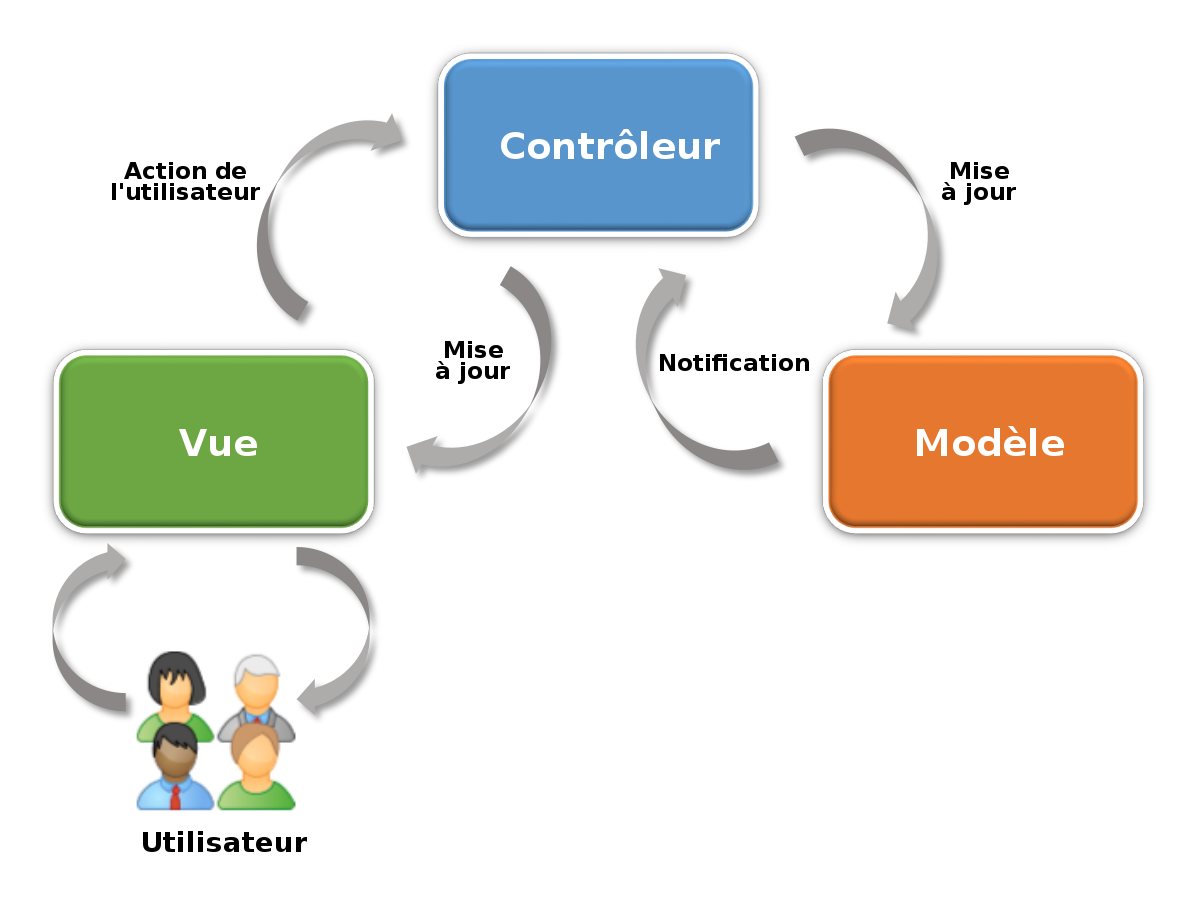
[6.4 Jira 10](#_3jw1ok60qbzp)

## 1.Introduction

La présente documentation technique fournit une vue d'ensemble du projet de gestion de ticketing. Ce document décrit l'architecture technique du projet et explique la structure du code en suivant le modèle MVC (Modèle-Vue-Contrôleur). L'objectif principal du projet est de créer une application web permettant aux utilisateurs de soumettre des tickets et de les gérer efficacement. Le développement sera réalisé en utilisant un serveur web local en PHP, avec l'utilisation du moteur de template Twig, une base de données MySQL et le framework Bootstrap pour un design responsive.

## 2.Architecture technique

L'architecture technique du projet repose sur le modèle MVC, qui est une approche couramment utilisée pour la conception et le développement d'applications web. Le modèle MVC divise l'application en trois composants principaux : le modèle, la vue et le contrôleur. Chaque composant a une responsabilité spécifique et interagit avec les autres de manière organisée.



### 2.1 Modèle (Model)

Le modèle représente la couche de données de l'application et gère la logique métier, y compris les interactions avec la base de données. Dans ce projet, le modèle sera responsable de la gestion des tickets, des utilisateurs, des catégories. Il utilisera la bibliothèque PHP PDO pour interagir avec la base de données MySQL.

### 2.2 Vue (View)

La vue est responsable de l'interface utilisateur et de l'affichage des données. Dans ce projet, les vues seront développées en utilisant le moteur de template Twig. Les vues seront conçues de manière à offrir une expérience utilisateur conviviale et intuitive. Le design sera de type responsive grâce à l'utilisation du framework Bootstrap, ce qui permettra une adaptation optimale sur différents appareils et résolutions d'écran.

### 2.3 Contrôleur (Controller)

Le contrôleur agit comme un intermédiaire entre le modèle et la vue. Il reçoit les requêtes de l'utilisateur à partir des vues, traite les données en utilisant le modèle approprié, puis renvoie les résultats à la vue correspondante pour affichage. Il assurera également la validation des données et l'exécution des actions appropriées.

## 3.Structure du projet

Le projet de gestion de ticketing en ligne est organisé selon une structure de fichiers et de dossiers bien définie, visant à assurer une organisation claire et efficace du code source. Voici la description de la structure du projet :

### 3.1 /Classe

Ce répertoire contiendra les fichiers des classes du projet.

* Administrateur.php : Classe pour gérer les fonctionnalités spécifiques aux administrateurs.
* Client.php : Classe pour gérer les fonctionnalités spécifiques aux clients.
* Database.php : Classe pour gérer la connexion et les opérations de base de données.
* Ticket.php : Classe pour gérer les fonctionnalités liées aux tickets.
* User.php : Classe pour gérer les fonctionnalités liées aux utilisateurs.

### 3.2 /Controller

Ce répertoire contiendra les fichiers des contrôleurs du projet.

* AdminController.php : Contrôleur pour gérer les actions liées à l'administration.
* AuthController.php : Contrôleur pour gérer les actions liées à l'authentification.
* ClientController.php : Contrôleur pour gérer les actions liées aux clients.
* ClientManagementController.php : Contrôleur pour gérer les actions de gestion des clients.
* TicketController.php : Contrôleur pour gérer les actions liées aux tickets.
* TicketCreationController.php : Contrôleur pour gérer la création de tickets.
* TicketManagementController.php : Contrôleur pour gérer la gestion des tickets.
* UserController.php : Contrôleur pour gérer les actions liées aux utilisateurs.

### 3.3 /Model

Ce répertoire contiendra les fichiers des modèles du projet.

* AdminModel.php : Modèle pour gérer les données des administrateurs.
* ClientModel.php : Modèle pour gérer les données des clients.
* TicketModel.php : Modèle pour gérer les données des tickets.
* UserModel.php : Modèle pour gérer les données des utilisateurs.

### 3.4 /Vendor

Ce répertoire contiendra les dépendances externes du projet.

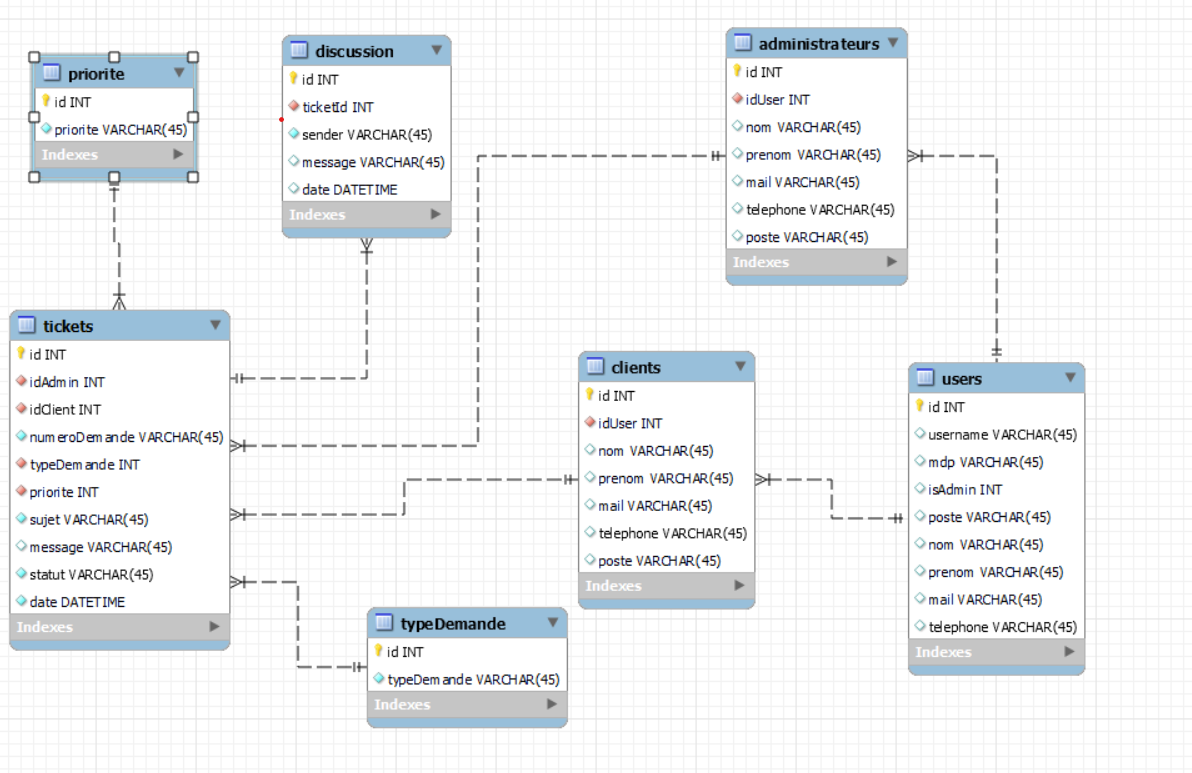
* /composer : Répertoire pour les dépendances gérées par Composer.
* /symfony : Répertoire pour les dépendances Symfony.
* /twig : Répertoire pour les dépendances Twig/Twig.

### 3.5 /view

Ce répertoire contiendra les fichiers de vue du projet.

## 4.Base de données

### 4.1 Schéma de la base de données



### 4.2 Relations entre les tables et de la façon dont elles sont utilisées dans le projet

La base de données est nommée ‘ticketing’ et contient huit tables : ‘administrateurs’, ‘clients’, ‘discussion’, ‘priorite’, ‘tickets’, ‘typedemande’ et ‘users’. Les relations entre ces tables sont les suivantes :

- ’administrateurs’ et ‘users’ : La table ‘administrateurs’ est liée à la table ‘users’ via la clé étrangère ‘idUser’, qui fait référence à la clé primaire ‘id’ de la table ‘users’. Chaque administrateur est donc un utilisateur.

- ’clients’ et ‘users’ : De la même façon que les administrateurs, la table ‘clients’ est également liée à la table ‘users’ via la clé étrangère ‘idUser’, qui fait référence à la clé primaire ‘id’ de la table ‘users’. Chaque client est donc aussi un utilisateur.

- ’tickets’ : Cette table est liée à plusieurs autres tables :

- ‘tickets’ est liée à ‘clients’ via la clé étrangère ‘idClient’, qui fait référence à la clé primaire ‘id’ de la table ‘clients’.

- ‘tickets’ est liée à ‘administrateurs’ via la clé étrangère ‘idAdmin’, qui fait référence à la clé primaire ‘id’ de la table ‘administrateurs’.

- ‘tickets’ est liée à ‘priorite’ via la clé étrangère ‘priorite’, qui fait référence à la clé ‘priorite’ de la table ‘priorite’.

- ‘tickets’ est liée à ‘typedemande’ via la clé étrangère ‘typeDemande’, qui fait référence à la clé ‘typeDemande’ de la table ‘typedemande’.

- ’discussion’ et ‘tickets’ : La table ‘discussion’ est liée à la table ‘tickets’ via la clé étrangère ‘ticketId’, qui fait référence à la clé primaire ‘id’ de la table ‘tickets’. Chaque discussion est donc associée à un ticket spécifique.

## 

## 5.Spécifications fonctionnelles

Notre application de gestion de ticketing vise à fournir un ensemble de fonctionnalités permettant de gérer efficacement les demandes de support et les problèmes signalés par les clients. Voici une description détaillée des principales fonctionnalités de l'application :

Réception des demandes : L'application permet aux utilisateurs de soumettre des tickets de support ou des demandes de service via une interface dédiée. Les utilisateurs peuvent fournir des informations détaillées sur leur problème, telles que la nature du problème, les symptômes rencontrés, les captures d'écran ou les fichiers joints pertinents.

Tri et catégorisation : Les tickets soumis sont automatiquement ou manuellement triés et catégorisés en fonction de leur urgence, de leur type ou de leur catégorie. Cela permet de faciliter le traitement des tickets en les regroupant de manière appropriée.

Attribution des tickets : Les tickets sont assignés aux membres de l'équipe de support compétents en fonction de leurs compétences, de leur charge de travail ou d'autres critères définis. L'attribution peut être effectuée automatiquement par le système en fonction du type de demande ou manuellement par un responsable de l'équipe.

Suivi et priorisation : Chaque ticket doit être suivi tout au long de son cycle de vie. L'application permet de connaître l'état d'avancement du ticket, de le mettre à jour et de le prioriser en fonction de l'urgence ou de l'importance. Les tickets en attente, en cours de traitement ou résolus sont clairement identifiés dans le système.

Communication et collaboration : L'application facilite la communication entre les utilisateurs et les membres de l'équipe de support. Les utilisateurs peuvent ajouter des commentaires supplémentaires aux tickets pour fournir des informations supplémentaires ou poser des questions. Les membres de l'équipe de support peuvent répondre aux commentaires des utilisateurs et fournir des mises à jour sur l'avancement de la résolution du ticket. Des notifications peuvent être envoyées aux utilisateurs et à l'équipe de support pour les tenir informés des mises à jour importantes.

## 6.Environnement de travail

L'environnement de développement du projet de gestion de ticketing comprend les éléments suivants, visant à assurer une mise en œuvre efficace et une collaboration facilitée

### 6.1 Framework Bootstrap

Le projet utilise le framework Bootstrap pour faciliter la conception et la mise en page de l'interface utilisateur. Bootstrap offre une collection de composants, de styles CSS et de scripts JavaScript prédéfinis, permettant de créer rapidement et facilement une interface responsive et esthétiquement agréable.

### 6.2 PDO (PHP Data Objects)

L'accès à la base de données MySQL est géré à l'aide de PDO. PDO est une extension de PHP qui fournit une interface abstraite pour interagir avec différentes bases de données, y compris MySQL. L'utilisation de PDO offre une approche sécurisée et portable pour la manipulation des données, protégeant contre les attaques par injection SQL et permettant une compatibilité facile avec d'autres bases de données si nécessaire.

### 6.3 Outil de versioning Git/GitHub

Git a été utilisé comme système de contrôle de version pour le suivi des modifications du code source. GitHub est utilisé comme plateforme de gestion de projets basée sur Git, permettant de partager et de collaborer sur le code source avec d'autres membres de l'équipe. Il nous a permis de fusionner les modifications et gérer les différentes versions de l'application.

### 6.4 Jira

C’est cette méthode agile qui a été adoptée pour la gestion de notre projet. Elle nous a permis de planifier, de suivre et de gérer les tâches et les fonctionnalités du projet. Cela nous a facilité la collaboration, le suivi et l'avancement du projet et la gestion des priorités.

