|  |  |  |
| --- | --- | --- |
| EPCF1 | 08 septembre  2023 | |
| Dossier projet – Plateforme WEB pour la gestion des tickets (Plateforme Ticketing) | | EXPERNET |

Table des matières

[I. Liste des compétences du référentiel qui sont couvertes par le projet 3](#_Toc144887140)

[II. Résumé du projet en anglais d'une longueur d’environ 5 à 10 lignes soit 100 à 120 mots, ou environ 600 caractères espaces non compris 4](#_Toc144887141)

[III. Cahier des charges, expression des besoins, ou spécifications fonctionnelles du projet 5](#_Toc144887142)

[IV. Spécifications techniques du projet, élaborées par le candidat, y compris pour la sécurité 7](#_Toc144887143)

[V. Réalisations du candidat comportant les extraits de code les plus significatifs, et en les argumentant, y compris pour la sécurité 8](#_Toc144887144)

[Modèle conceptuel des données 8](#_Toc144887145)

[Modèle physique des données 9](#_Toc144887146)

[Conception et développement de la plateforme 11](#_Toc144887147)

[Connexion à la plateforme 11](#_Toc144887148)

[Conception 11](#_Toc144887149)

[Développement 12](#_Toc144887150)

[Page de connexion finale 13](#_Toc144887151)

[Dashboard pour les administrateurs 14](#_Toc144887152)

[Conception 14](#_Toc144887153)

[Développement 15](#_Toc144887154)

[VI. Présentation du jeu d’essai élaboré par le candidat de la fonctionnalité la plus représentative (données en entrée, données attendues, données obtenues) 17](#_Toc144887155)

[VII. Description de la veille, effectuée par le candidat durant le projet, sur les vulnérabilités de sécurité 18](#_Toc144887156)

[VIII. Description d’une situation de travail ayant nécessité une recherche, effectuée par le candidat durant le projet 19](#_Toc144887157)

# Liste des compétences du référentiel qui sont couvertes par le projet

|  |  |
| --- | --- |
| **Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité** | Développer la partie back-end d’une interface utilisateur web |
| Développer des composants d’accès aux données |
| Développer une interface utilisateur de type desktop |
| Développer la partie front-end d’une interface utilisateur web |
| Maquetter une application |

Le projet couvre obligatoirement les compétences suivantes :

* Maquetter une application
* Développer des composants d’accès aux données
* Développer la partie front-end d’une interface utilisateur web
* Développer la partie back-end d’une interface utilisateur web

# Résumé du projet en anglais d'une longueur d’environ 5 à 10 lignes soit 100 à 120 mots, ou environ 600 caractères espaces non compris

A company requires an online ticket management platform that enables various departments to create tickets for tasks and monitor their progress.

Each ticket can encompass multiple processes, each with its own status, effective dates, and the user who treat.

Users are divided into two groups: 'regular' users and administrators. Administrators have access to a dashboard that compiles statistics on pending, in-progress, and closed tickets per department, as well as user account management. 'Regular' users can view and create tickets, take ownership of ongoing tickets, and close those they are currently working on.

The platform is secured through an authentication system and developed in PHP using the Symfony framework for the back-office, along with daisyUI and Tailwind CSS utilities for the front-office.

# Cahier des charges, expression des besoins, ou spécifications fonctionnelles du projet

Le présent cahier des charges définit les exigences et les spécifications pour le développement d'une plateforme de gestion des tickets en ligne pour une entreprise.

* Contexte

Une entreprise souhaite mettre en place une solution permettant à ses différents services de créer des tickets pour effectuer des tâches et de suivre leur progression.

* Fonctionnalités
  + Les fonctionnalités permettent de savoir ce qui est fonctionnel pour utilisateur ‘régulier’ et un administrateur

1. La plateforme a une page de connexion
   * + Identification de l’utilisateur connecté
       - La plateforme identifie l’utilisateur et permet de savoir le type d’utilisateur (régulier ou administrateur).
       - L’utilisateur peut se déconnecter.
       - L’utilisateur peut demander à réinitialiser son mot de passe depuis la plateforme (il doit recevoir un mail contenant son nouveau mot de passe fort qui est généré aléatoirement).
2. Fonctionnalités utilisateur ‘régulier’
   * + Création de ticket
       - L’utilisateur peut créer un ticket avec un raison de création et le service concerné.
     + Suivi de ticket
       - Chaque ticket peut contenir plusieurs processus, chacun avec son propre statut, des dates effectives, et un utilisateur en charge.
     + Prise en charge de ticket
       - L’utilisateur peut ouvrir un traitement pour un ticket de son service qui a le statut ‘EN ATTENTE’.
       - L’utilisateur peut prendre le relais sur le traitement d’un ticket de son service qui a le statut ‘EN COURS’.
       - L’utilisateur peut clore son traitement sur le ticket et ainsi clore le ticket ce qui marque le statut comme ‘Fermé’.
       - L’utilisateur peut transférer le ticket à un autre service ce qui marque le statut comme ‘TRANSFÉRÉ // EN ATTENTE’.
3. Fonctionnalités administrateur

**L’administrateur doit avoir les mêmes fonctionnalités que l’utilisateur.**

* + - Tableau de bord
      * L’administrateur doit avoir accès depuis un tableau de bord aux statistiques des états de tickets par service (nombre de tickets ‘EN ATTENTE’, nombre de tickets ‘EN COURS’, nombre de tickets ‘Fermé’.
      * L’administrateur doit pouvoir gérer les comptes des autres utilisateurs (création d’utilisateur, modification d’un utilisateur, suppression d’un utilisateur).
      * L’administrateur doit avoir accès aux statistiques de chaque utilisateurs (nombre de tickets ‘Crée’, nombre de ticket ‘Pris en charge’, nombre de ticket ‘Fermé’.
* Technologies utilisées
  + Conception de la plateforme
    - Modèle conceptuel de données avec Looping.
    - Diagramme d’utilisation avec Draw.io.
    - Maquettes de la plateforme avec Figma.
  + Développement de la plateforme
    - Back-office en PHP 8.2.4
    - Front-office en CSS 3
  + Persistance des données
    - La base de données avec MySQL 8.1.0 (serveur local avec WampServer).
* Livrable et délais
  + La plateforme complète et fonctionnelle doit être livrée conformément aux spécifications.
  + La date de livraison prévue pour la plateforme est le 08/09/2023.

# Spécifications techniques du projet, élaborées par le candidat, y compris pour la sécurité

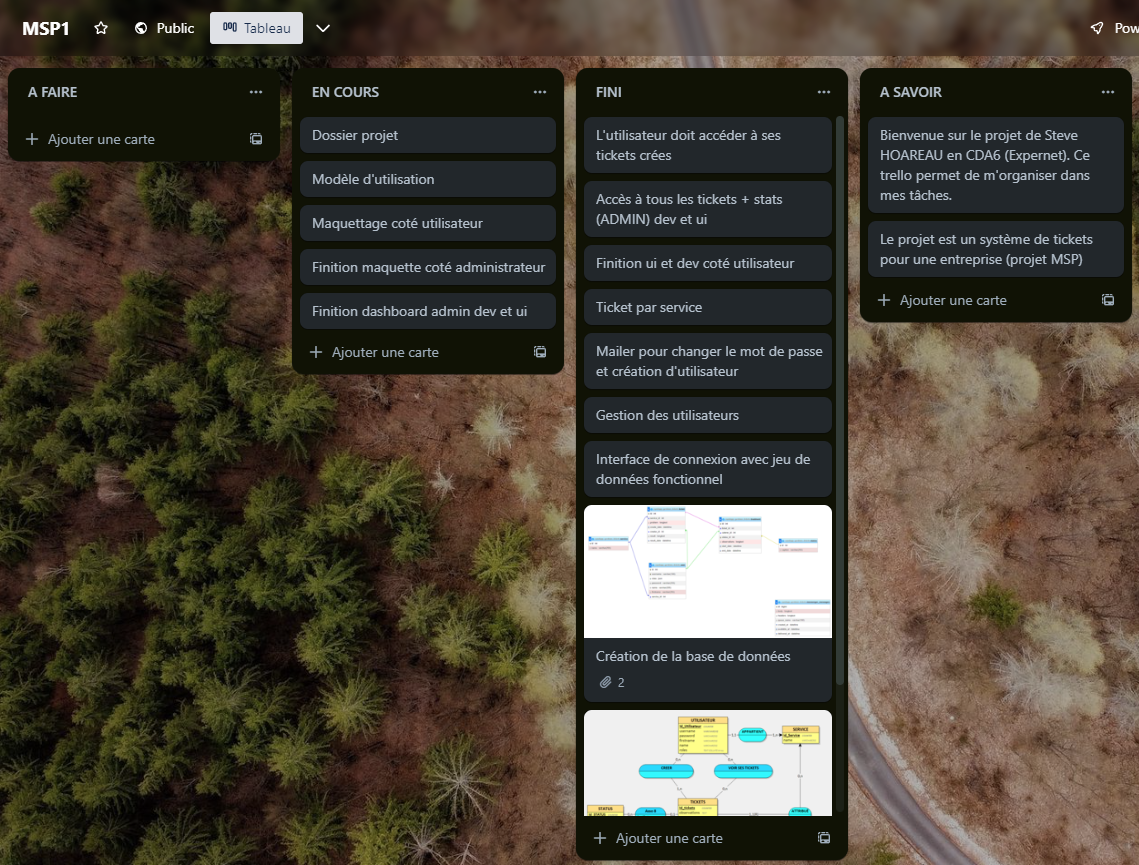
1. Développement de la plateforme

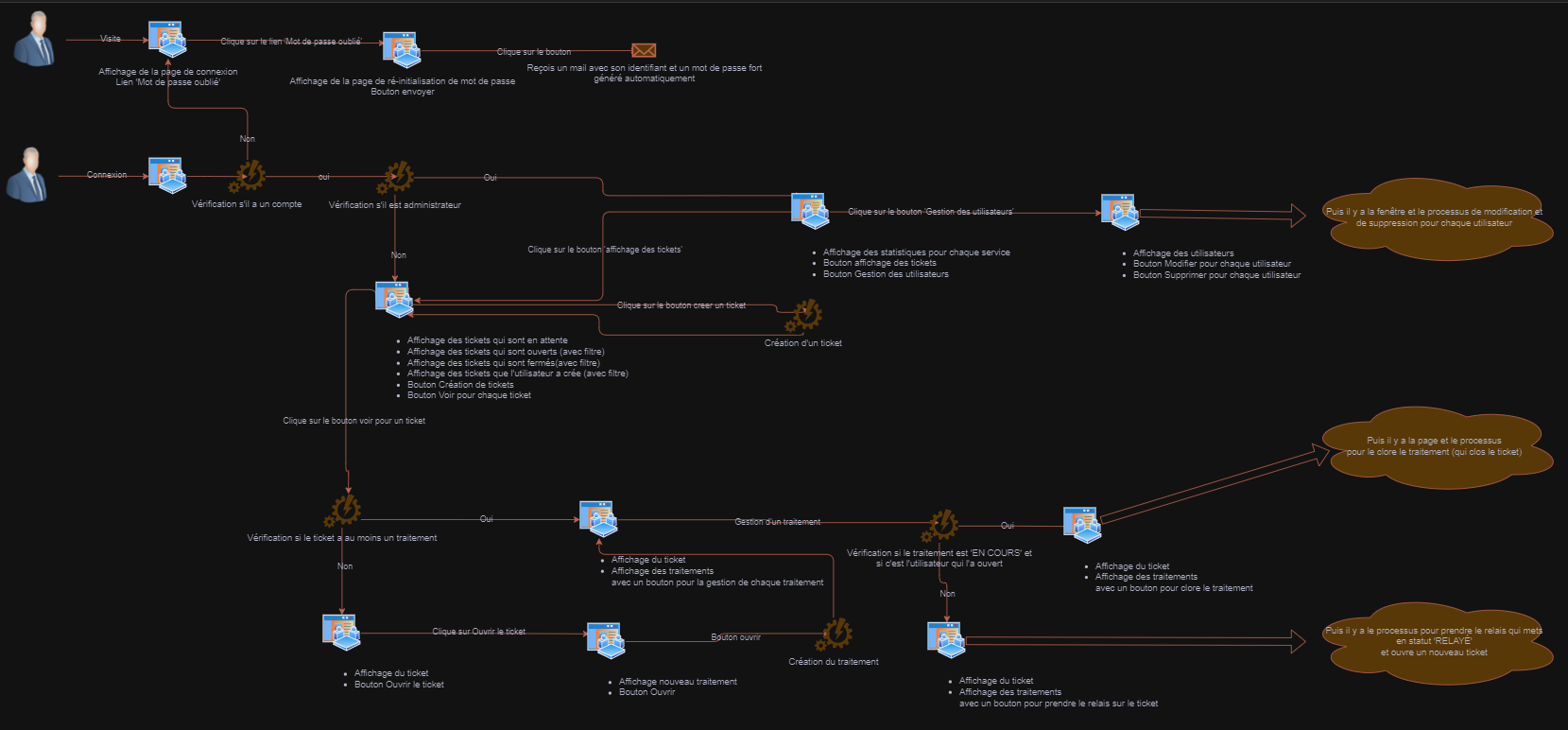
* Le back-office a été codé en PHP avec le Framework Symfony 6.3.1.
* Le front-office a été codé en CSS 3 avec la librairie daisyUI 3.6.4 et l’utilitaire TailwindCSS 3.3.3.
* La gestion des composants de développement s’est faite avec Composer 2.6.2 et NodeJS 18.16.0.
* Les ressources (assets) utilisées sont traitées par Webpack et PostCSS.
* Outil de version avec Git 2.42.0.
* Requêtes AJAX avec JQuery 3.7.1.
* Quelques rendus graphiques avec ChartJS 4.3.2.

1. Sécurité de la plateforme

* Un système d’identification est mis en place sur la plateforme avec une page de connexion.
* Les mots de passes sont générés automatiquement depuis l’application et ils sont forts (mot de passe long et contenant des lettres majuscules et minuscules, des symboles et des chiffres).
* Les mots de passes des utilisateurs sont cryptés depuis la plateforme en utilisant bCrypt.
* Les routes dans le code sont sécurisées en vérifiant si l’utilisateur est connecté.
* La manipulation de la base de données dans le code est sécurisée avec des jetons d’accès.
* Les classes dans le code sont encapsulées.

1. Gestion de projet

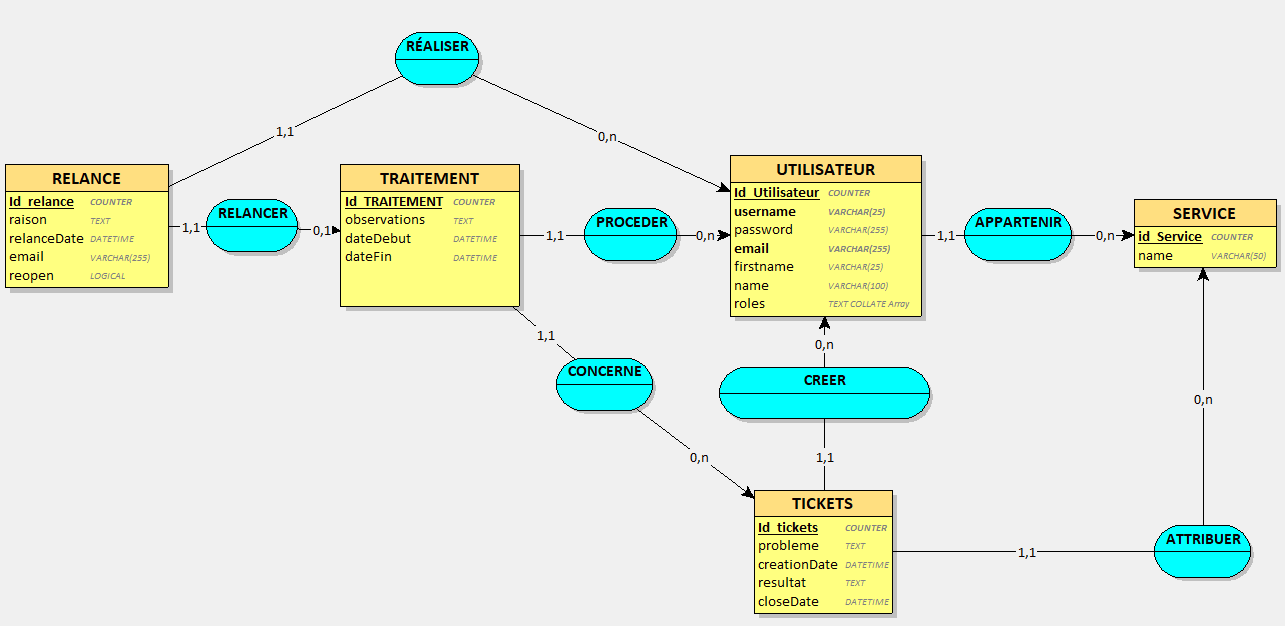
* Gestion des tâches avec Trello ([MSP1 | Trello](https://trello.com/b/bTeXd9jD/msp1))



↪ Diagramme d’utilisation

# Réalisations du candidat comportant les extraits de code les plus significatifs, et en les argumentant, y compris pour la sécurité

## Modèle conceptuel des données



Dans ce MCD, on retrouve 5 entités et 6 associations.

L’entité « utilisateur » possède un identifiant « id\_utilisateur ».

L’entité « service » possède un identifiant « id\_service » et est associé à 0 ou plusieurs utilisateurs, l’utilisateur appartient à 1 et 1 seul service.

L’entité « tickets » possède un identifiant « id tickets » et est créer par à un et un seul utilisateur, l’utilisateur peut créer de 0 à n tickets. Un ticket est aussi attribué à 1 et 1 seul service et un service peut avoir 0 à n tickets.

L’entité « traitement » possède un identifiant « id\_traitement » et est associé à un et un seul utilisateur, l’utilisateur peut procéder 0 à n traitement. Un traitement est aussi concerné par 1 et 1 seul ticket et un ticket peut être concerné par 0 à n traitements.

L’entité « relance » possède un identifiant « id\_relance » et est relancer sur un 1 et 1 seul traitement et un traitement peut avoir une relance. Une relance est réalisée par 1 et 1 seul utilisateur et un utilisateur peut réaliser 0 à n relance.

## Modèle physique des données

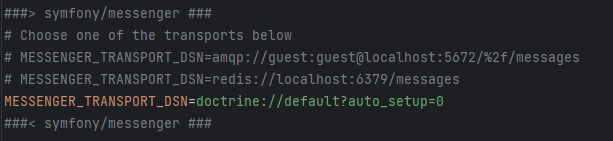
On observe 7 tables.

Les 5 entités du modèle conceptuel des données se sont transformées en tables dans le modèle physique. Aucune association a été transformée en table.

On retrouve les anciennes entités :

* « service » qui est devenue la table « service » qui possède une clé primaire « id »
* « utilisateur » qui est devenue la table « user » qui possède une clé primaire « id » et une clé étrangère « service\_id » qui fait référence à la table « service »
* « tickets » qui est devenue la table « ticket » qui possède une clé primaire « id », une clé étrangère « service\_id » qui fait référence à la table « service » et une clé étrangère « creator\_id » qui fait référence à la table « user »
* « traitement » qui est devenue la table « treatment » qui possède une clé primaire « id », une clé étrangère « ticket\_id » qui fait référence à la table « ticket » et une clé étrangère « caterer\_id » qui fait référence à la table « user »
* « relance » qui est devenue la table « relance » qui possède une clé primaire « id », une clé étrangère « treatment\_id » qui fait référence à la table « treatment » et une clé étrangère « user\_id » qui fait référence à la table « user »

On retrouve aussi 2 nouvelles tables :

* « messenger\_message » qui possède une clé primaire. Cette table va servir à la plateforme pour enregistrer les messages des Publishers pour être traités avec un système de queue.

En l’occurrence, dans mon application, ce module est configuré dans mon fichier d’environnement «.env».

* « mail\_configuration » qui possède une clé primaire. Cette table va servir à la configuration mail depuis la plateforme et enregistrer celle-ci dans la base de données. Il existera une seule configuration mail dans le cas de l’utilisation de cette plateforme mais ça peut être utile si la plateforme doit évoluer.

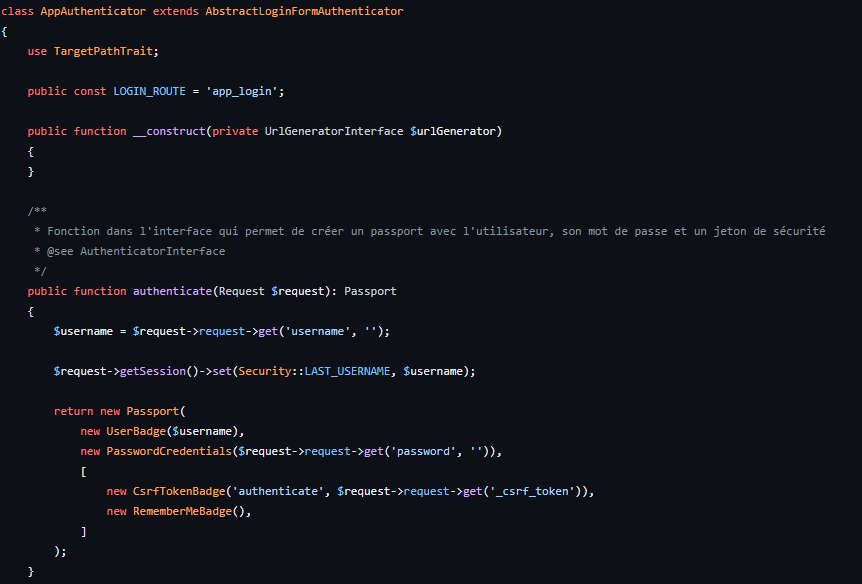
## Conception et développement de la plateforme

### Connexion à la plateforme

#### Conception

↪ Maquette conçue sur Figma. L’utilisateur est redirigé vers la page de connexion si l’utilisateur n’est pas connecté.

#### Développement



↪ Méthodes qui permettent d’exécuter un traitement après une authentification.

↪ Route par défaut, on redirige l’utilisateur vers la page de connexion s’il n’est pas connecté.

#### Page de connexion finale

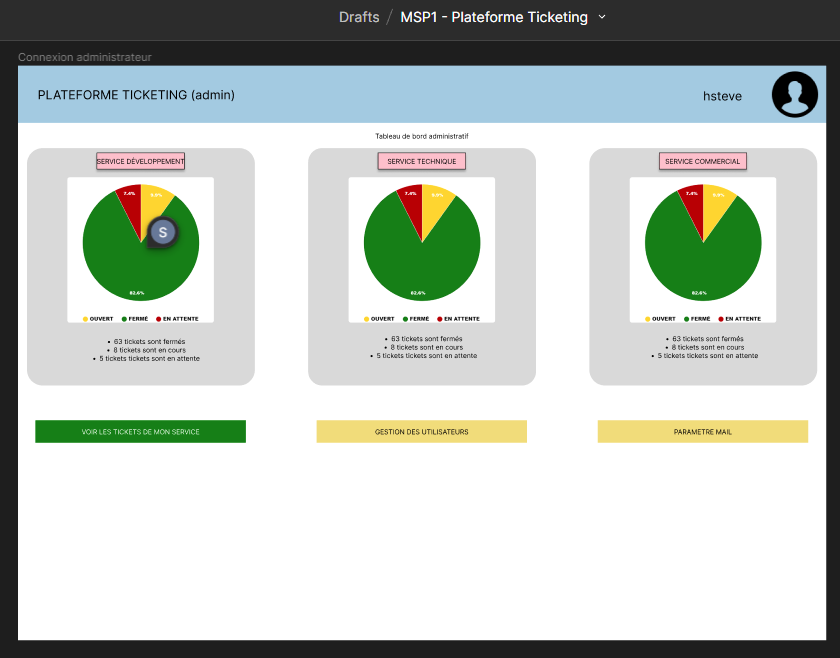
↪ L’identifiant est envoyé par mail avec le mot de passe lors de la création du compte.

Spécificité :

* Un compte administrateur par défaut est créé et envoyé par un email par défaut lors du premier lancement de la page de connexion.
* Un lien « Mot de passe oublié ? » est disponible pour avoir un mot de passe fort générer automatiquement et envoyé par mail si le compte existe.

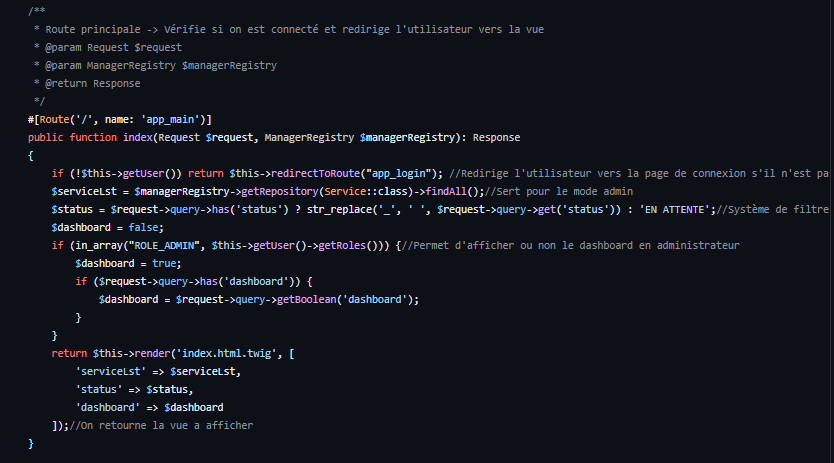
### Dashboard pour les administrateurs

#### Conception

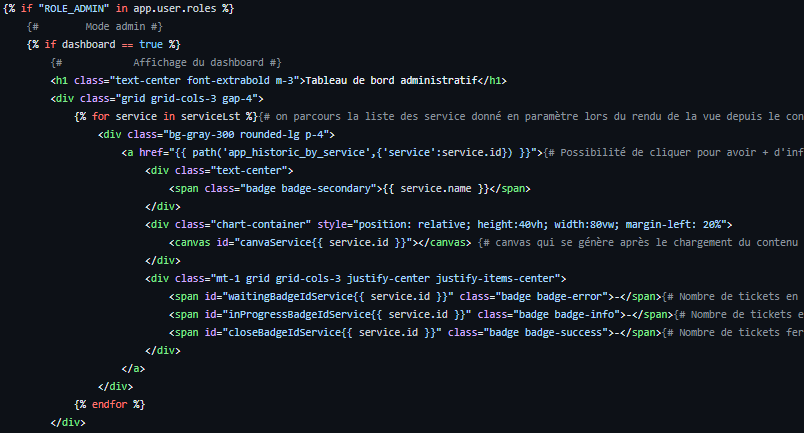


↪ Maquette conçue sur Figma

#### Développement



↪ Route par défaut, retourne une vue avec des paramètres spécifique si c’est un administrateur



↪ Vérification dans la vue lorsqu’un utilisateur est connecté (index.html.twig) qui montre le tableau de bord si le paramètre dashboard est activé (dashboard == true) et les statistiques par service.

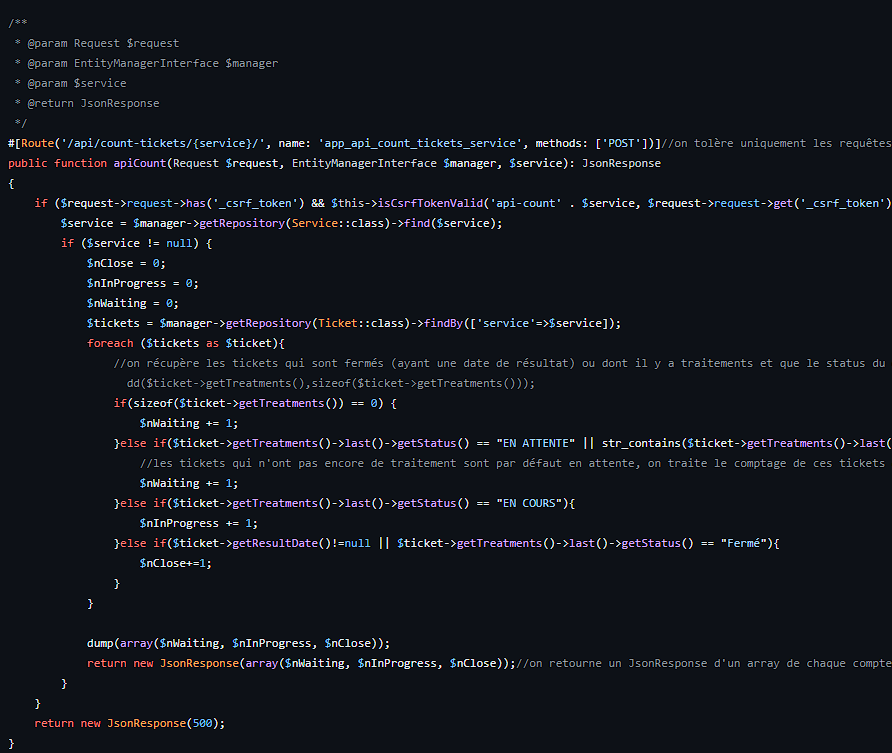
La balise « canvas » qui fera apparaitre l’élément graphique d’un camembert généré à partir de ChartJS grâce à son identifiant « canvaService{{ service.id}} » (voir ci-dessous)



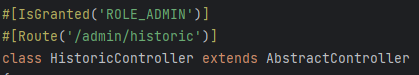
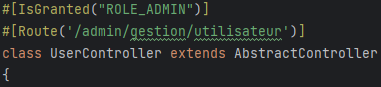
↪ Pour mettre à jour les balises « canvas », on utilise l’évènement du chargement du DOM et on met à jour, pour chaque service, le camembert de ChartJS associé en faisant des appels AJAX à l’aide de JQuery à la route « app\_api\_count\_tickets\_service »

Spécificité :

* La requête AJAX à l’aide de JQuery est sécurisée car elle utilise la méthode POST et un jeton « \_csrf\_token »



↪ la route « app\_api\_count\_tickets\_service » qui tolère uniquement les requêtes POST et qui sert à avoir sous forme de JsonResponse, chaque compteur pour les traitements de tickets.

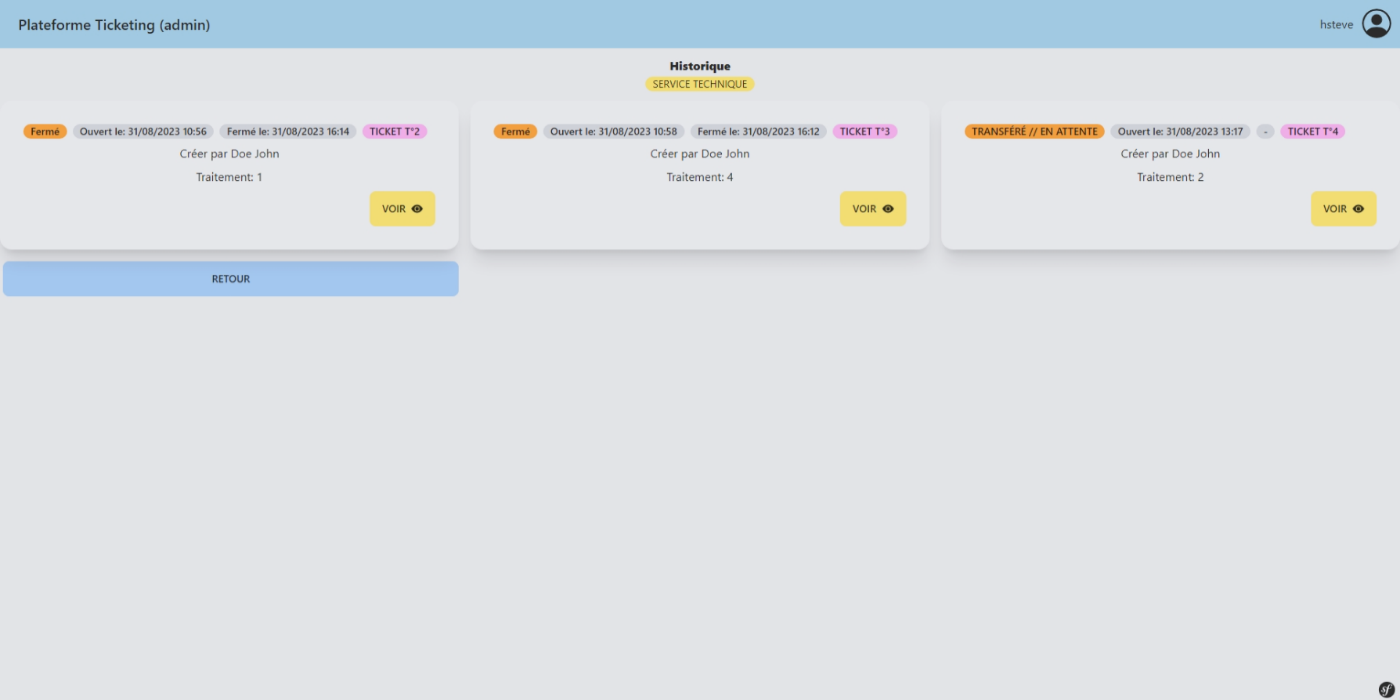


↪ Certaines routes sont protégées ↩

#### Dashboard final

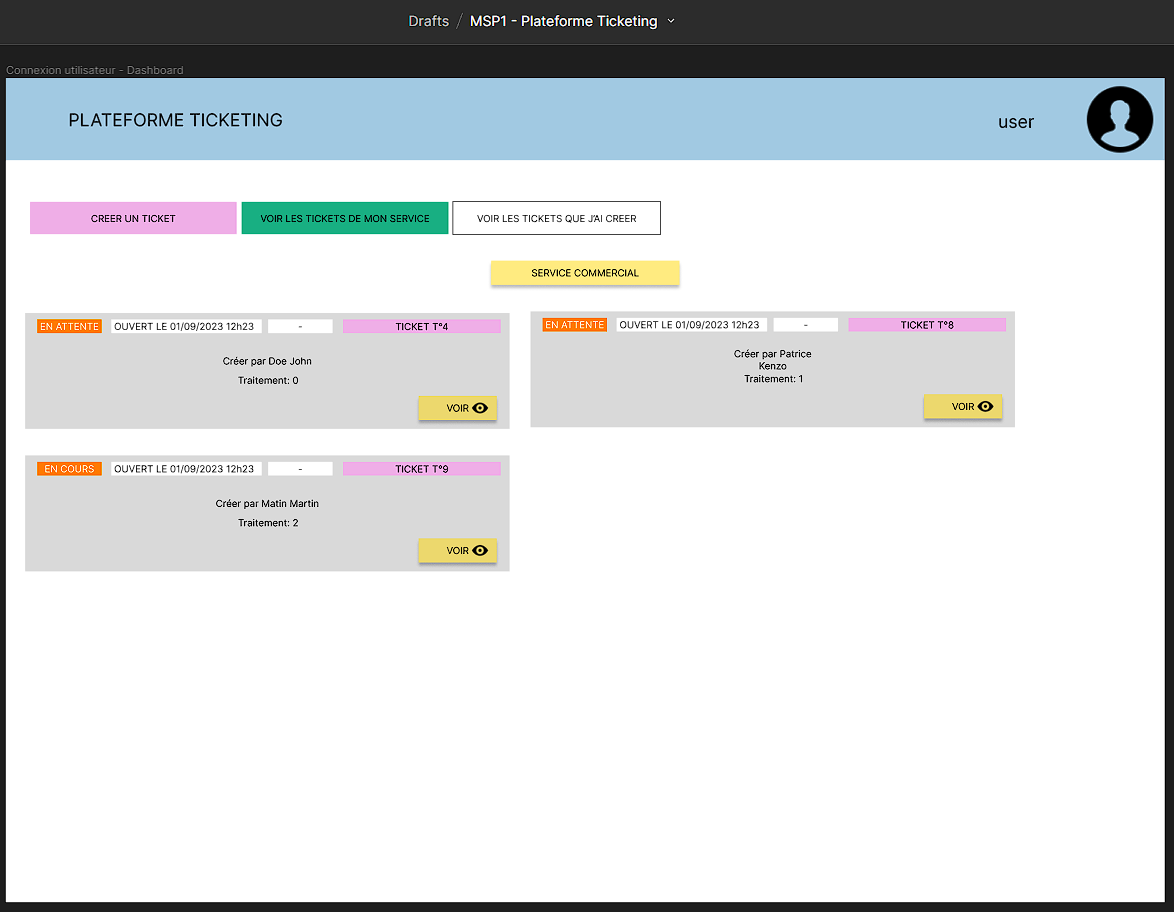


↪ En cliquant sur un service l’administrateur aura un historique des tickets par rapport au service(voir-ci-dessous).



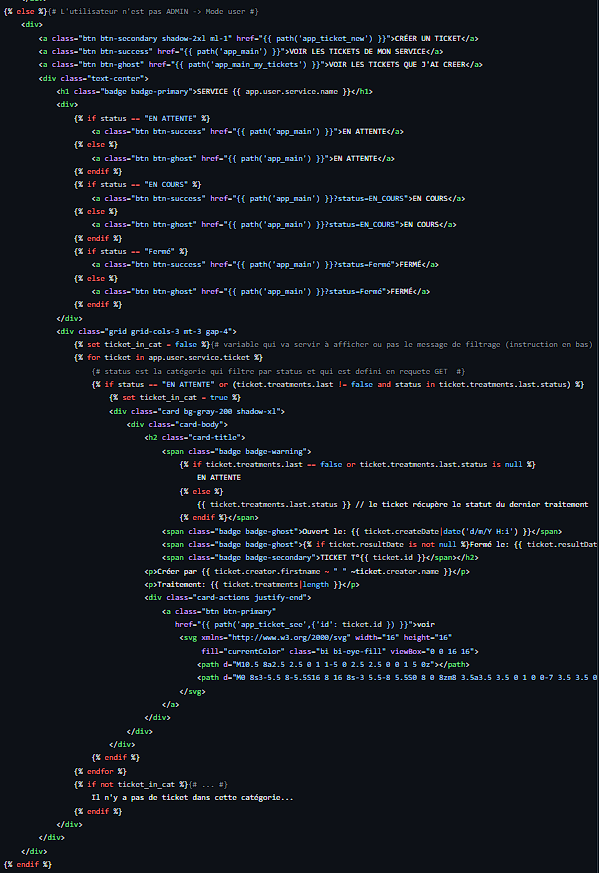
### Page utilisateur

#### Conception



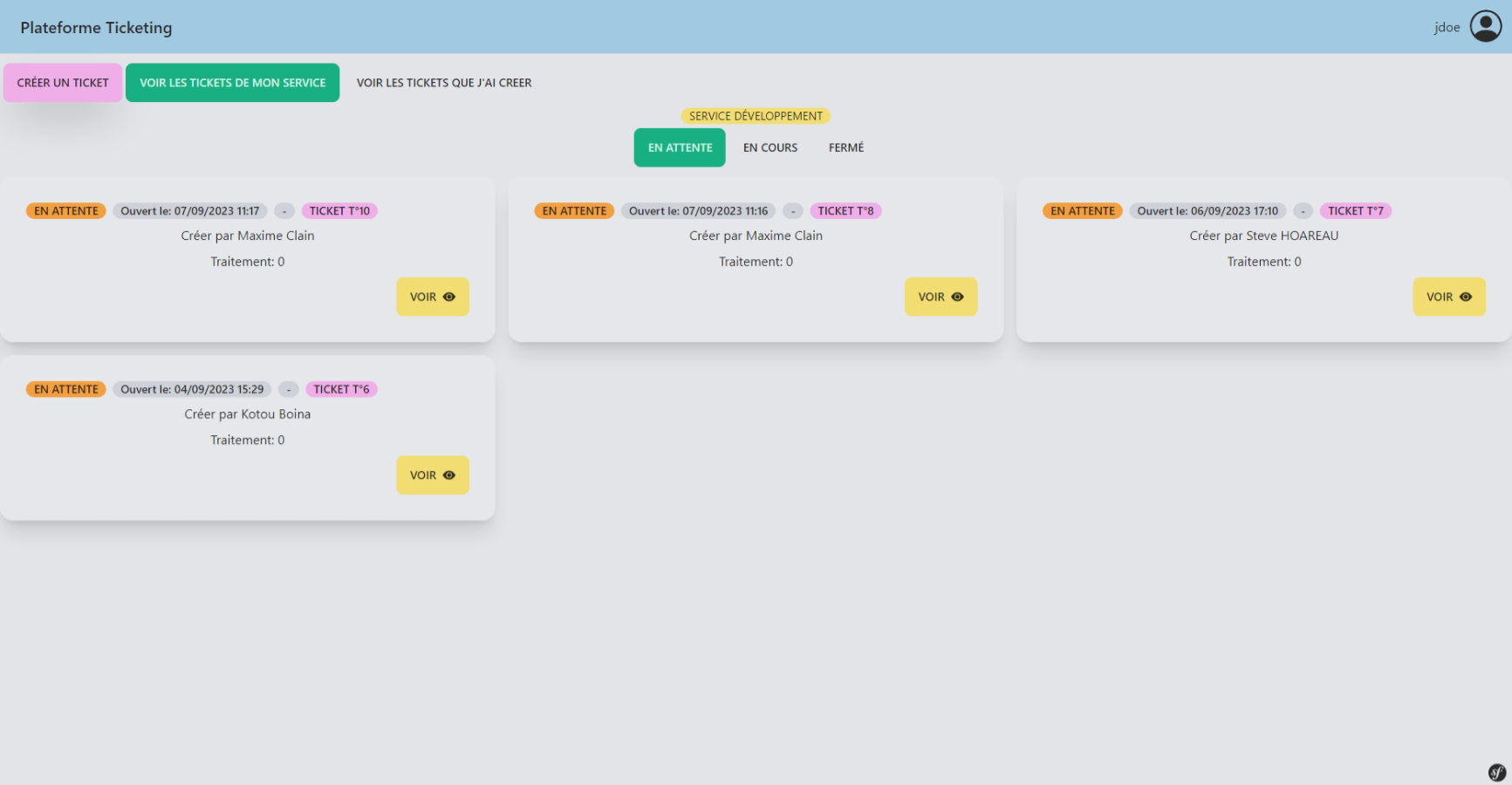
↪ Maquette conçue sur Figma.

#### Développement

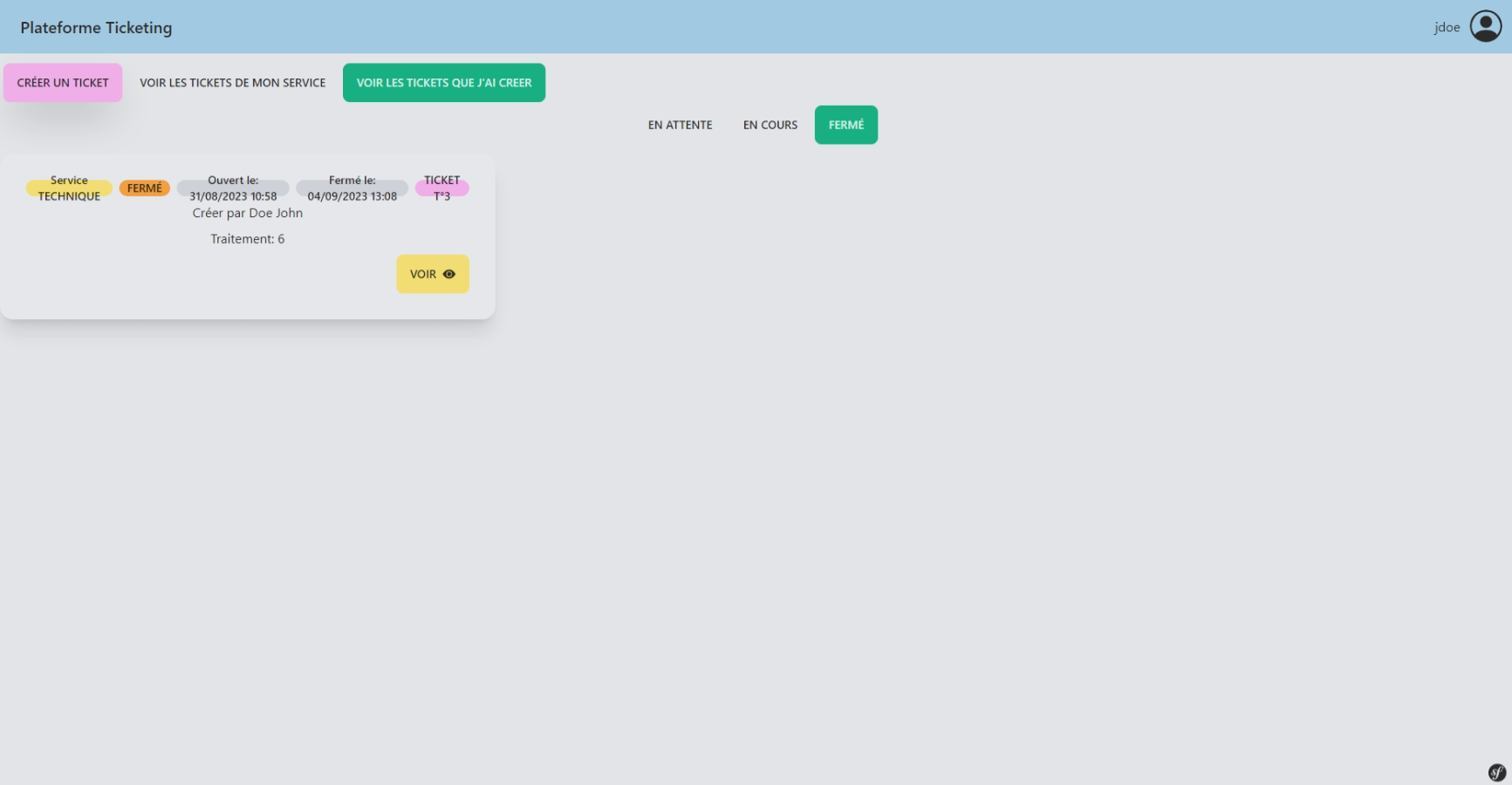


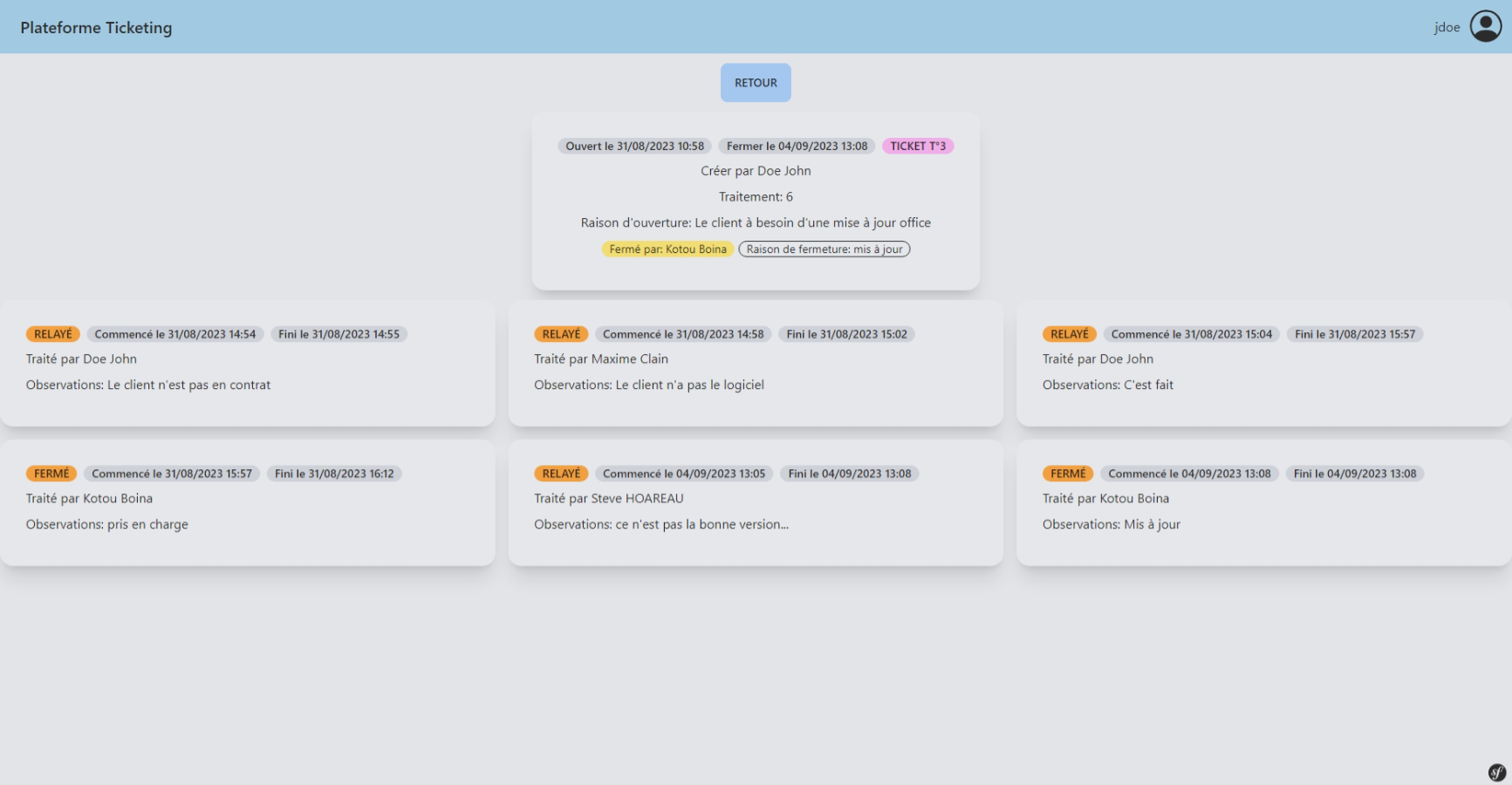
↪ Vue appelée depuis la route par défaut qui a été montrer plus haut. Chaque ticket récupère par défaut le statut ‘EN ATTENTE’ ou le statut de son dernier traitement dans le cas échéant.

#### Page utilisateur finale



↪ L’utilisateur a accès à ses tickets avec un système de filtre par statut. Il peut aussi voir les tickets qu’il a créé(voir-ci-dessous).



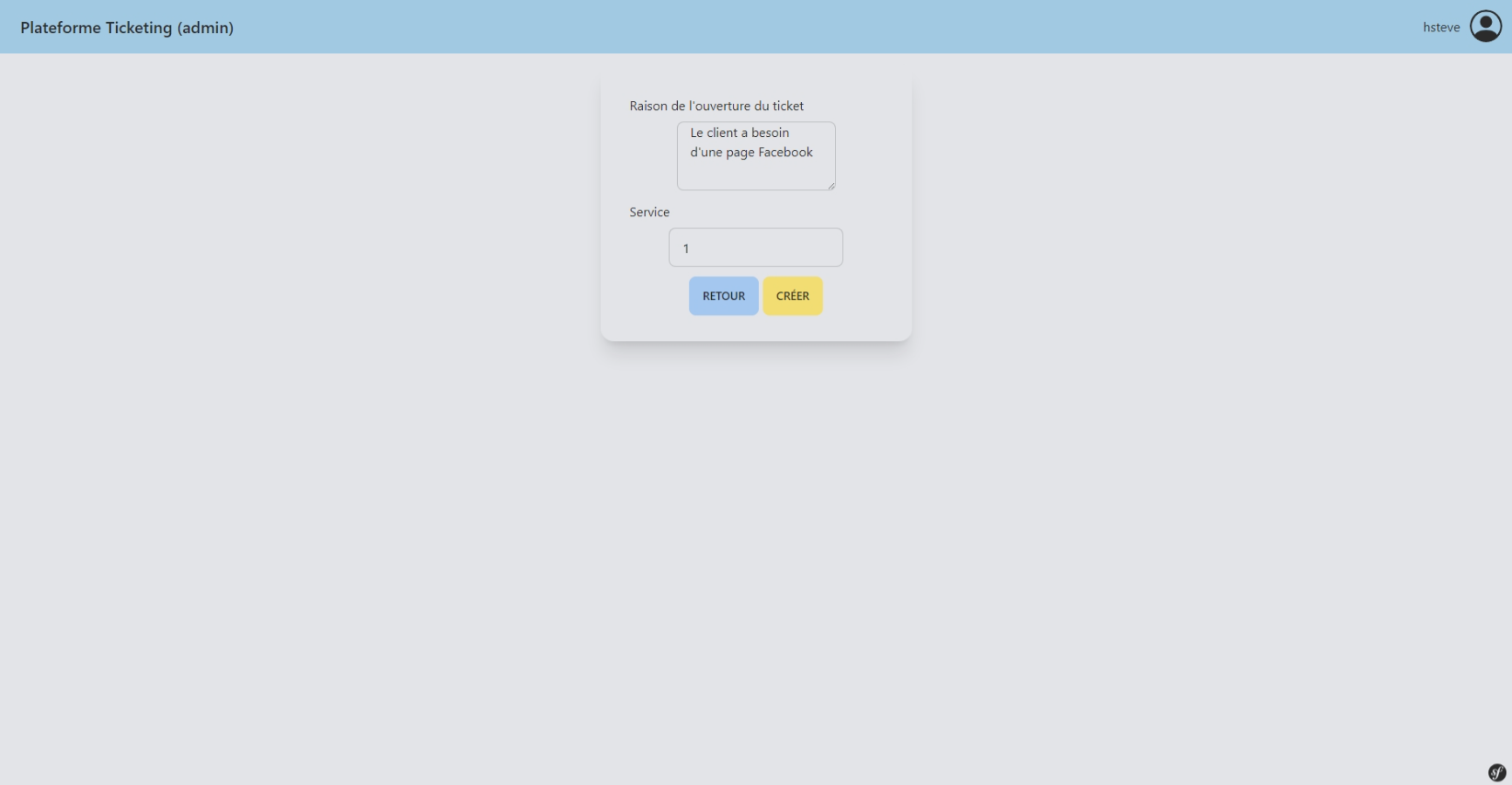


↪ L’utilisateur a accès aux traitements du ticket.

# Présentation du jeu d’essai élaboré par le candidat de la fonctionnalité la plus représentative (données en entrée, données attendues, données obtenues)

## Création de ticket

### Données en entrées



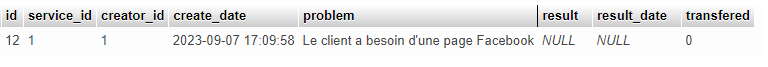
L’identifiant d’un service (obligatoire)

Une raison d’ouverture (obligatoire)

### Données attendues

* Une raison d’ouverture non vide
* Un identifiant d’un service non vide et existant
* L’utilisateur connecté qui est remonté automatiquement dans le code

### Données obtenues



↪ Le ticket a été crée et ajouté dans la base de données

Le champ « id » est l’identifiant du ticket.

Le champ « service\_id » est l’identifiant du service non vide et existant que l’utilisateur a renseigné

Le champ « create\_date » est la date quand l’utilisateur a crée qui est définie automatique dans le code par la date du jour.

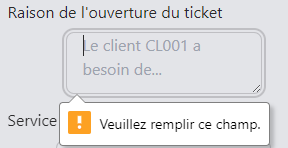
Le champ « problem » est la raison d’ouverture non vide que l’utilisateur a définie quand il a créé le ticket.

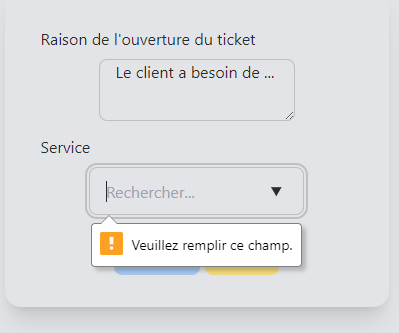
Le champ « result » et « result\_date » sont NULL par défaut et sert pour la fermeture du ticket.

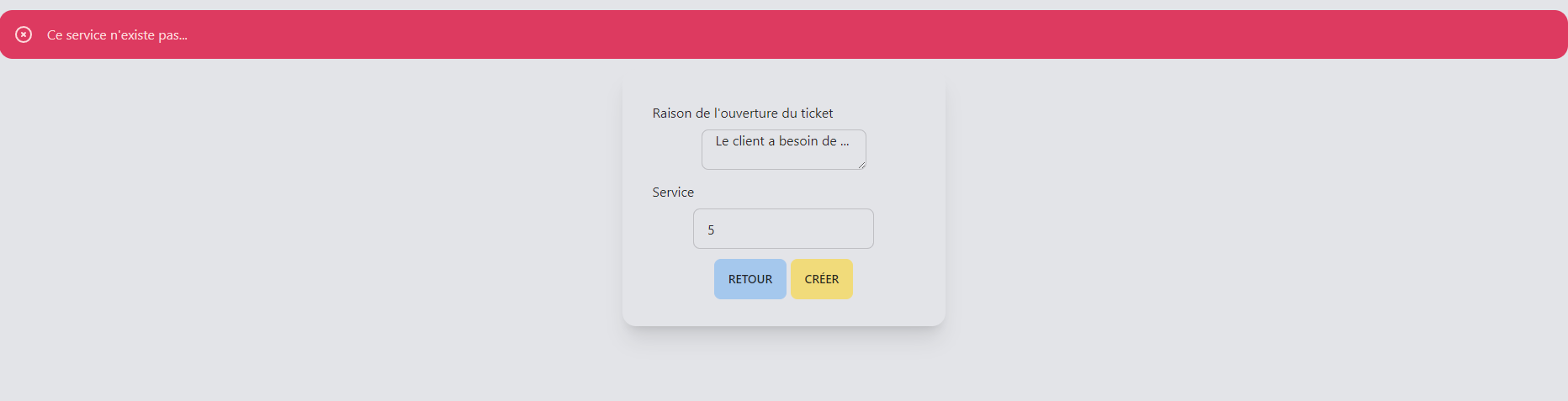
Le champ « transfered » est false par défaut (0) et sert à définir quand le ticket est transféré.

En cas d’erreur :

* Pour une raison d’ouverture vide :



* Pour un service vide :
* Pour un service non existant :



# Description de la veille, effectuée par le candidat durant le projet, sur les vulnérabilités de sécurité

# Description d’une situation de travail ayant nécessité une recherche, effectuée par le candidat durant le projet