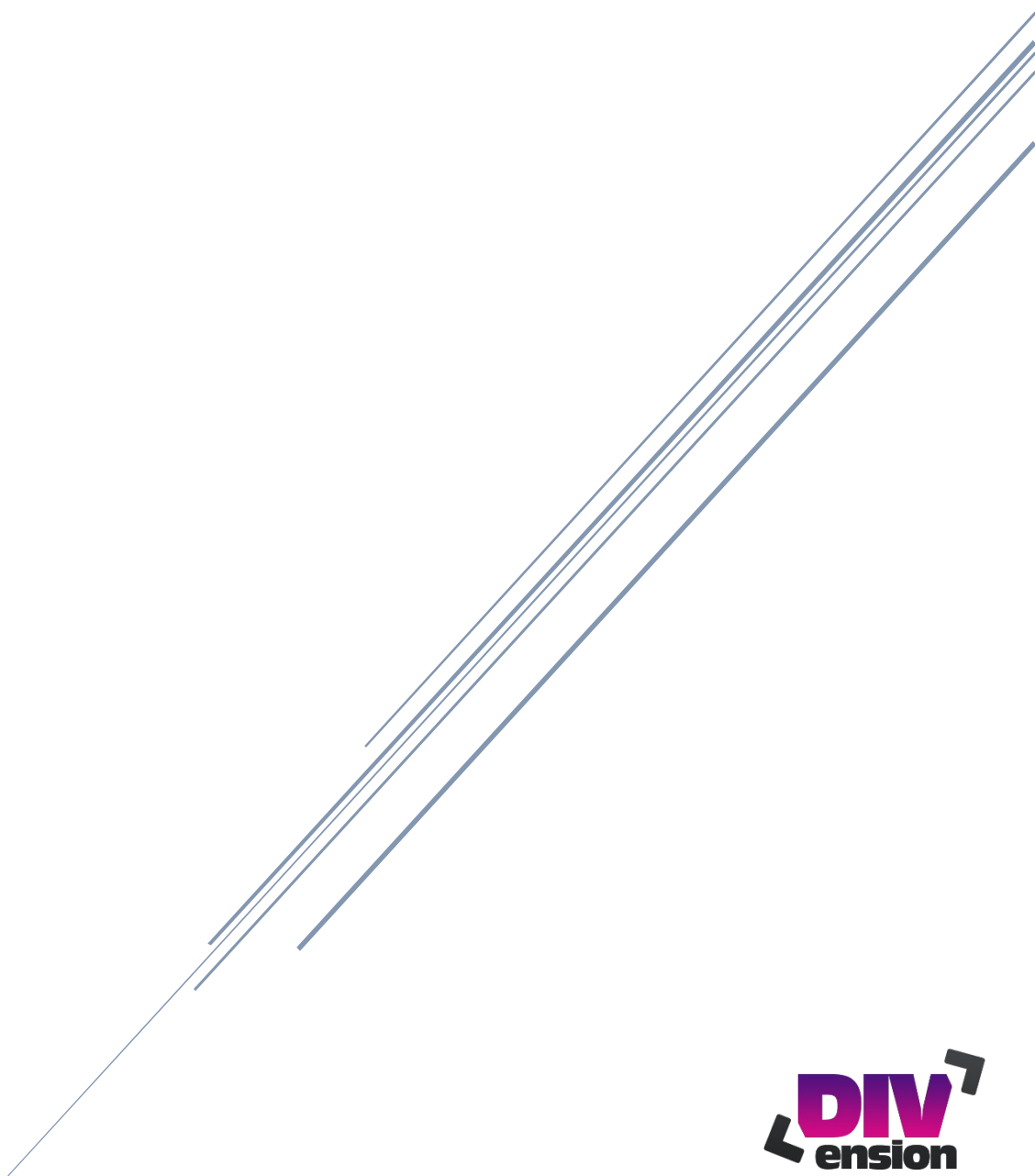


DOSSIER DE CONCEPTION

Open Tennis – Gestion des VIPs



Rédacteur : Bartholomé GILI
Superviseur : Ethan BOURCEREAU

Table des matières

Comportement du système	2
1. Cas d'utilisation	3
1.1 Acteurs.....	3
1.2 Fonctionnalités	3
1.3 Description du cas d'utilisation	4
2. Descriptions fonctionnelles	5
2.1 Authentification.....	5
2.2 Ajouter un VIP	6
2.3 Modifier un Suivi	7
Structure du système	8
3. États-transitions	9
4. Diagramme de classes	10
5. Structure de la base de données.....	11
6. Architecture.....	12
7. Technologies.....	14
Next.js.....	14
Symfony	14
PostgreSQL	14
Docker	14

Première partie

Comportement du système

1. Cas d'utilisation

1.1 Acteurs

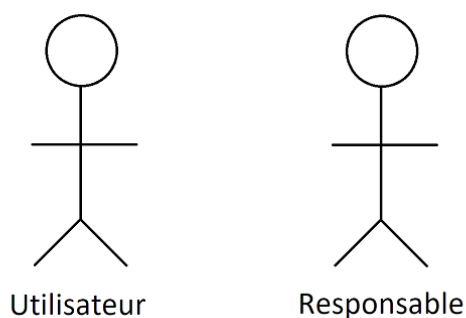


FIGURE 1.1 - ACTEURS DE L'APPLICATION

L'**utilisateur** n'a accès qu'à la page d'authentification.

L'acteur principal de l'application est le **responsable**. Ce dernier possède un rôle d'administrateur et est le seul à pouvoir accéder à l'interface de l'application web, sur lequel il possède tous les droits.

1.2 Fonctionnalités

Les fonctionnalités correspondent aux actions que peuvent réaliser les acteurs.

S'authentifier	<ul style="list-style-type: none">- Connexion à l'application- Déconnexion de l'application
Consulter la liste des VIPs	<ul style="list-style-type: none">- Consulter la liste des VIPs- Trier la liste des VIPs- Chercher un VIP
Modifier la liste des VIPs	<ul style="list-style-type: none">- Ajouter un VIP- Modifier un VIP- Supprimer un VIP
Afficher un VIP	<ul style="list-style-type: none">- Afficher un VIP
Consulter la liste des suivis	<ul style="list-style-type: none">- Consulter la liste des échanges- Consulter la liste des actions
Modifier la liste des suivis	<ul style="list-style-type: none">- Enregistrer un échange / action- Modifier un échange / action- Supprimer un échange / action
Afficher un suivi	<ul style="list-style-type: none">- Afficher un suivi

TABLEAU 1.1 - LISTE DE FONCTIONNALITES

1.3 Description du cas d'utilisation

Les cas d'utilisation relient les acteurs aux fonctionnalités. Ils correspondent aux thèmes de la section 1.2 et englobe les fonctionnalités correspondantes. Le système dans lequel on travaille est l'application web de Gestion des VIPs.

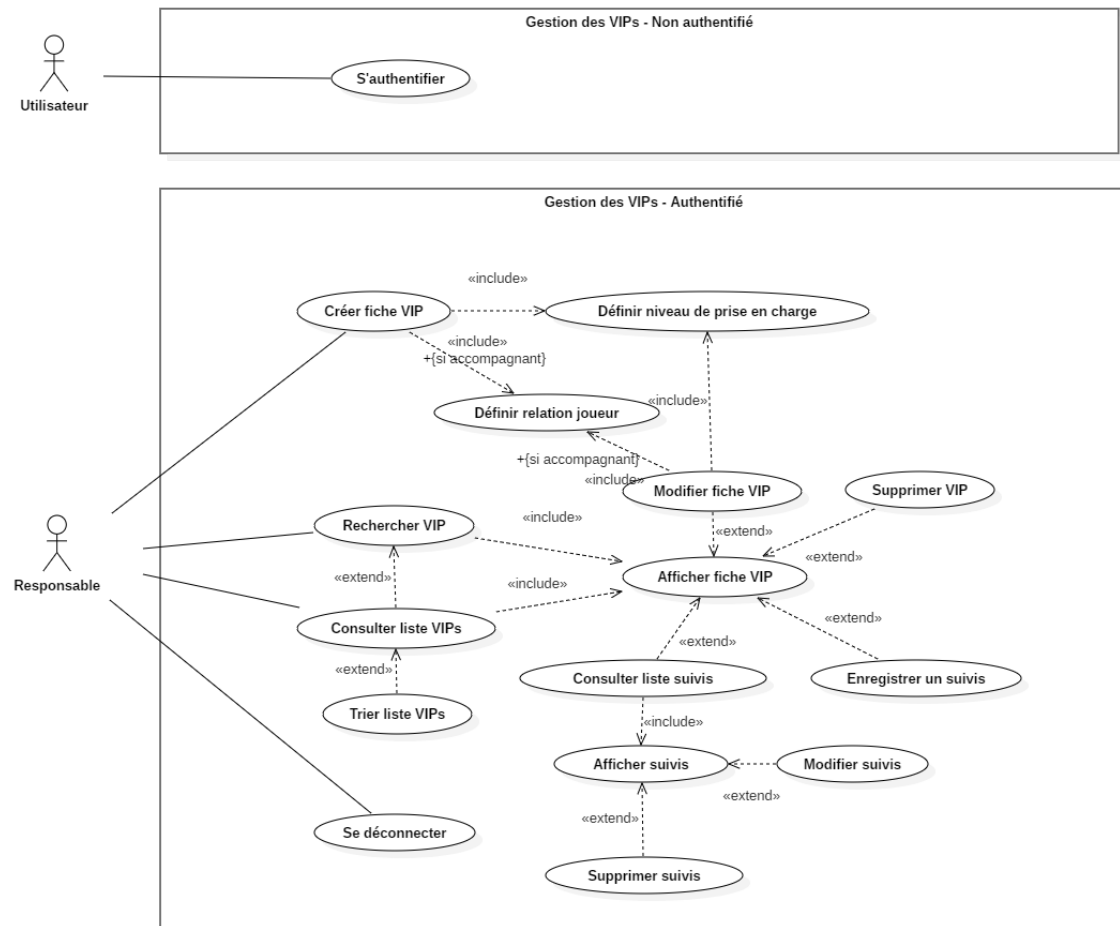


FIGURE 1.2 - DIAGRAMME CAS D'UTILISATION

L'authentification est obligatoire pour accéder à l'application en elle-même. Lorsqu'un utilisateur n'est pas authentifié, la seule page à laquelle il pourra accéder est celle d'authentification.

2. Descriptions fonctionnelles

Pour certains cas d'utilisation (cf. [1.3](#)), il est intéressant d'étudier son scénario d'utilisation.

2.1 Authentification

Authentification	
Acteurs	Utilisateur
Préconditions	L'utilisateur n'est pas authentifié
Entrées	
Séquence d'évènements	<ul style="list-style-type: none">- L'utilisateur renseigne son login et son mot de passe sur un formulaire prévu à cet effet- Lorsqu'il clique sur le bouton « envoyer », le formulaire va faire une requête à l'API pour valider les identifiants- Si les identifiants sont bons, l'utilisateur est authentifié en tant que responsable et est redirigé vers la page d'accueil- Sinon, un message d'erreur s'affiche
Postconditions	
Sorties	

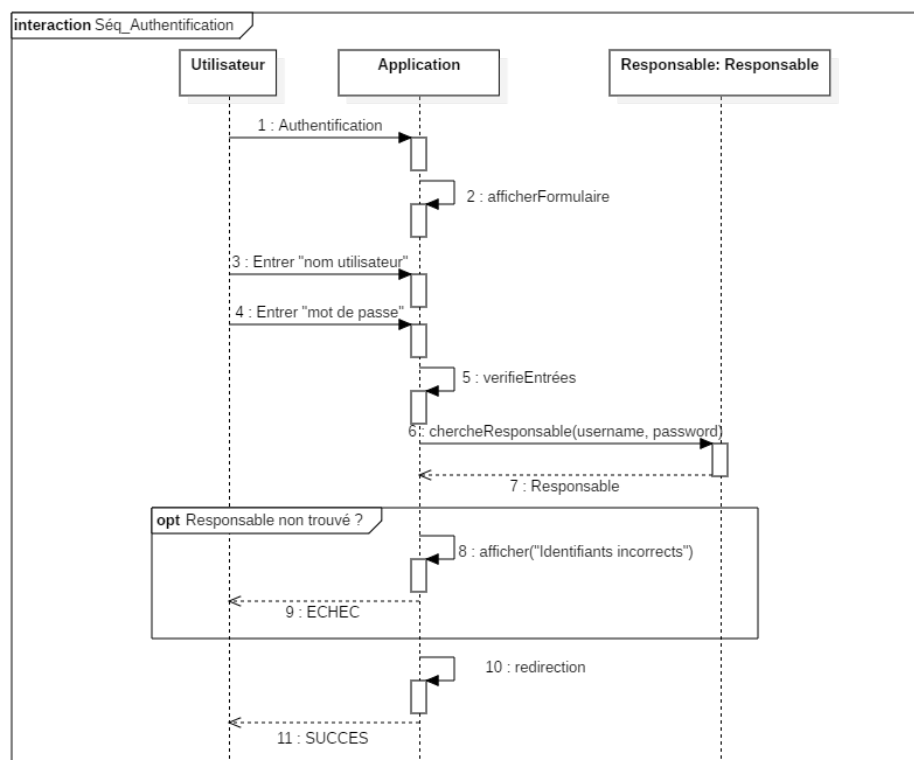


FIGURE 2.1 – DIAGRAMME DE SEQUENCE « S'AUTENTIFIER »

2.2 Ajouter un VIP

Ajouter un VIP	
Acteurs	Responsable
Préconditions	
Entrées	
Séquence d'évènements	<ul style="list-style-type: none"> - Le responsable remplit un formulaire avec les différents champs correspondants aux colonnes de la table VIP - Il doit choisir le type du VIP : Joueur ou Accompagnant - En fonction du type choisi, d'autres champs apparaissent - Une fois qu'il appuie sur le bouton « valider », le formulaire est envoyé et l'entité VIP est persistée dans la base de données
Postconditions	L'entité VIP a bien été créée
Sorties	

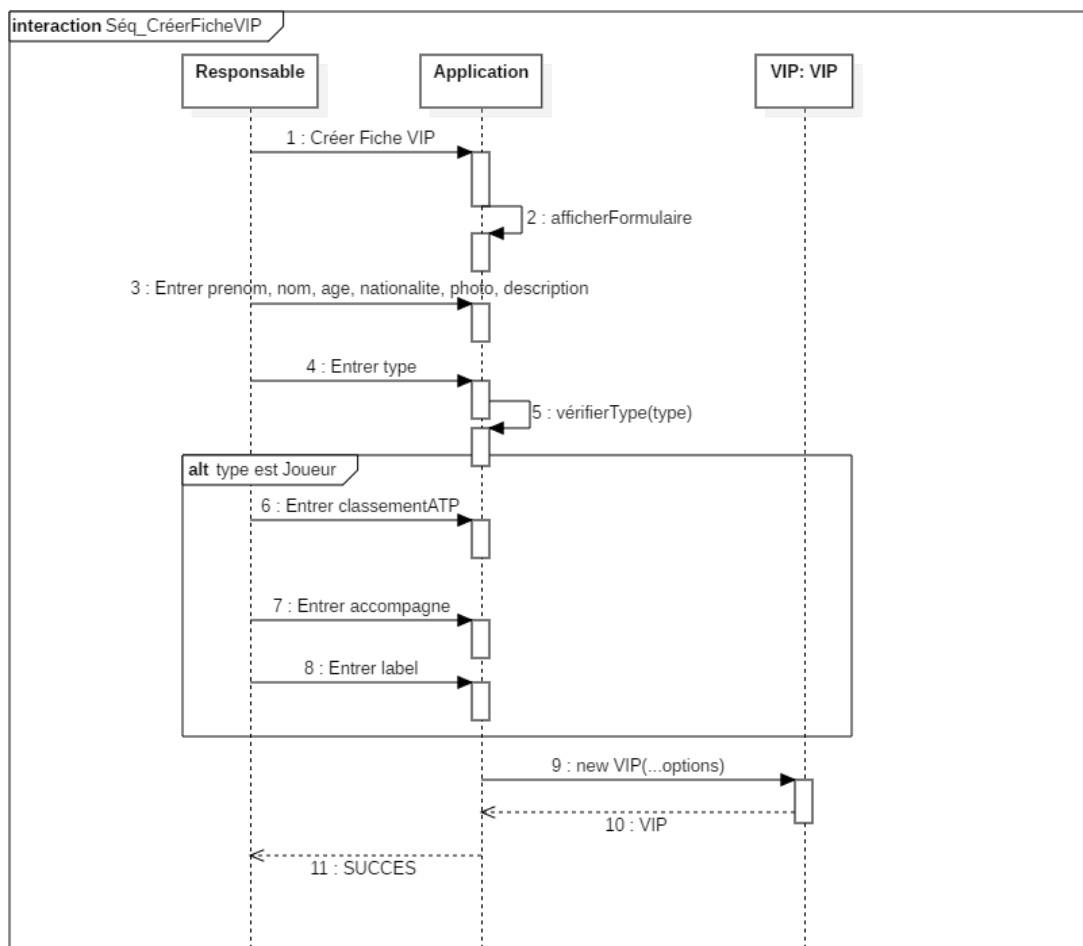


FIGURE 2.2 - DIAGRAMME DE SEQUENCE « AJOUTER UN VIP »

2.3 Modifier un Suivi

Modifier un Suivi	
Acteurs	Responsable
Préconditions	
Entrées	Identifiant du suivi à modifier
Séquence d'évènements	<ul style="list-style-type: none"> - L'application vérifie que l'identifiant du suivi correspond bien à un suivi, et renvoi ce dernier - Le responsable change ensuite les champs qu'il désire dans le formulaire affiché par l'application et prérempli avec les données actuelles du suivi - Une fois qu'il appuie sur le bouton « valider », le formulaire est envoyé et l'entité Suivi est modifiée dans la base de données
Postconditions	L'entité Suivi a bien été modifiée dans la base de données
Sorties	

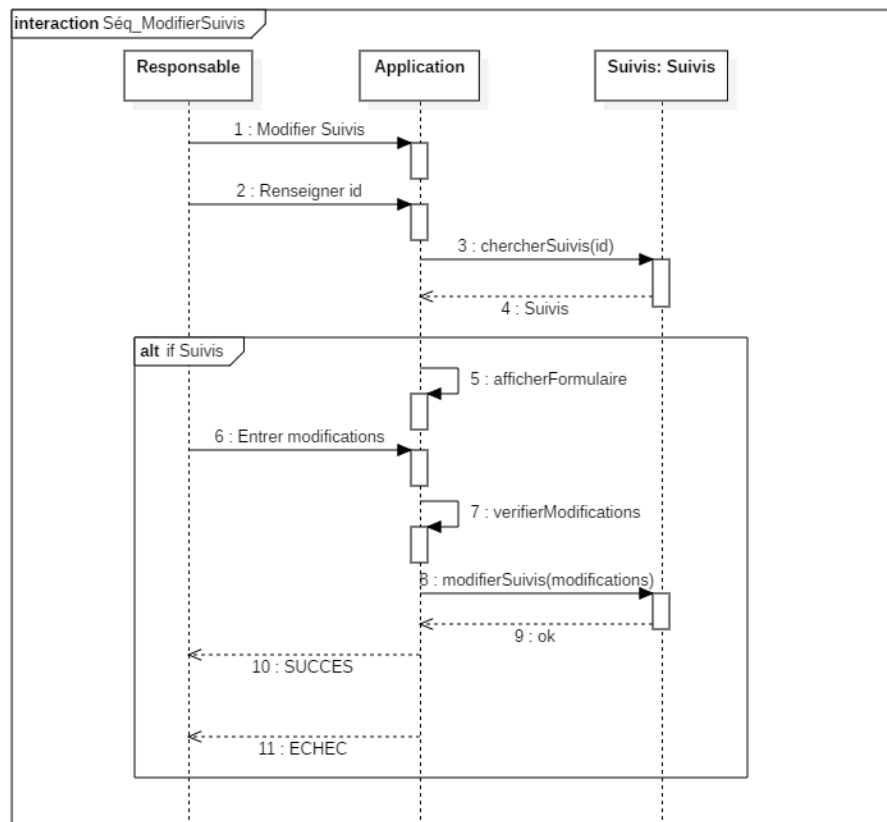


FIGURE 2.3 - DIAGRAMME DE SEQUENCE « MODIFIER UN SUIVI »

Deuxième partie

Structure du système

3. États-transitions

L'interface finale rendue pour l'utilisateur et les transitions globales d'états peuvent-être schématisées comme suit.

Chaque « état » représente une page, et chaque connexion une action que l'on effectue pour passer d'une page à une autre.

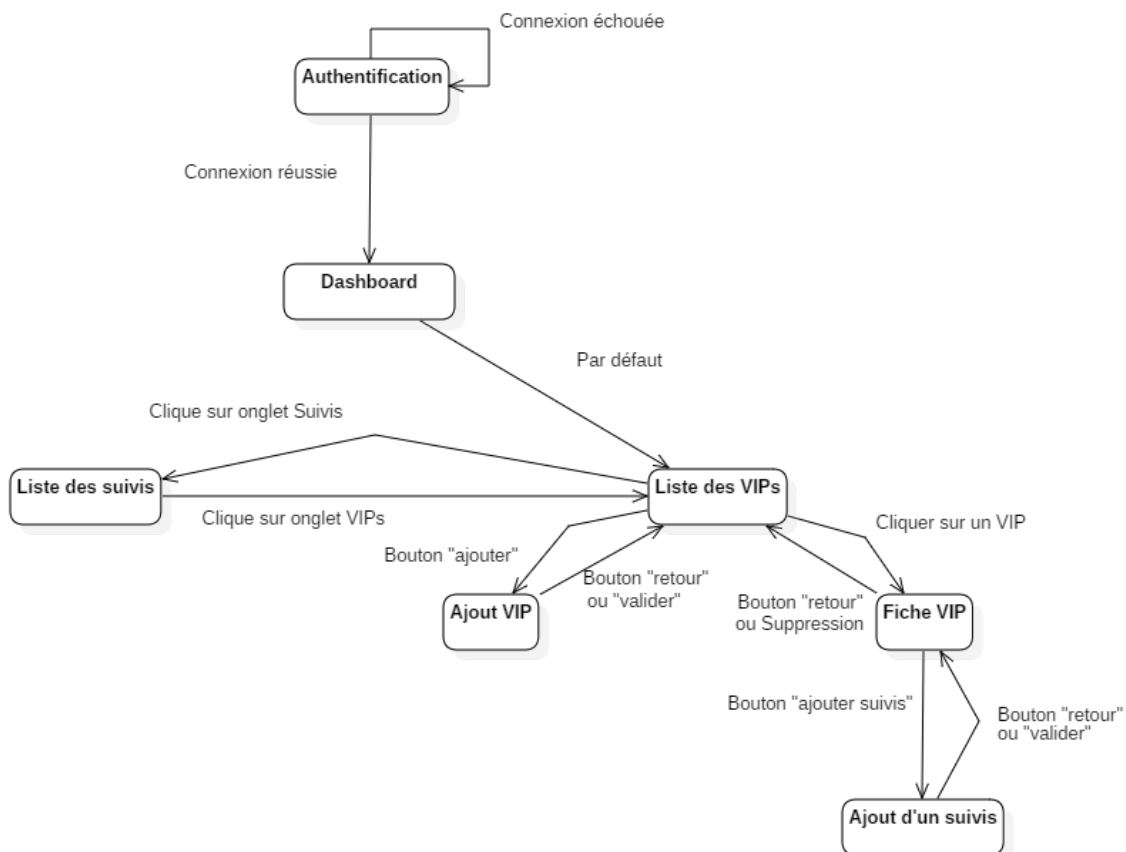


FIGURE 3.1 – DIAGRAMME D'ETATS-TRANSITIONS

4. Diagramme de classes

On présente ici le diagramme de classe du modèle, qui détaille l'articulation des classes du modèle avec les attributs :

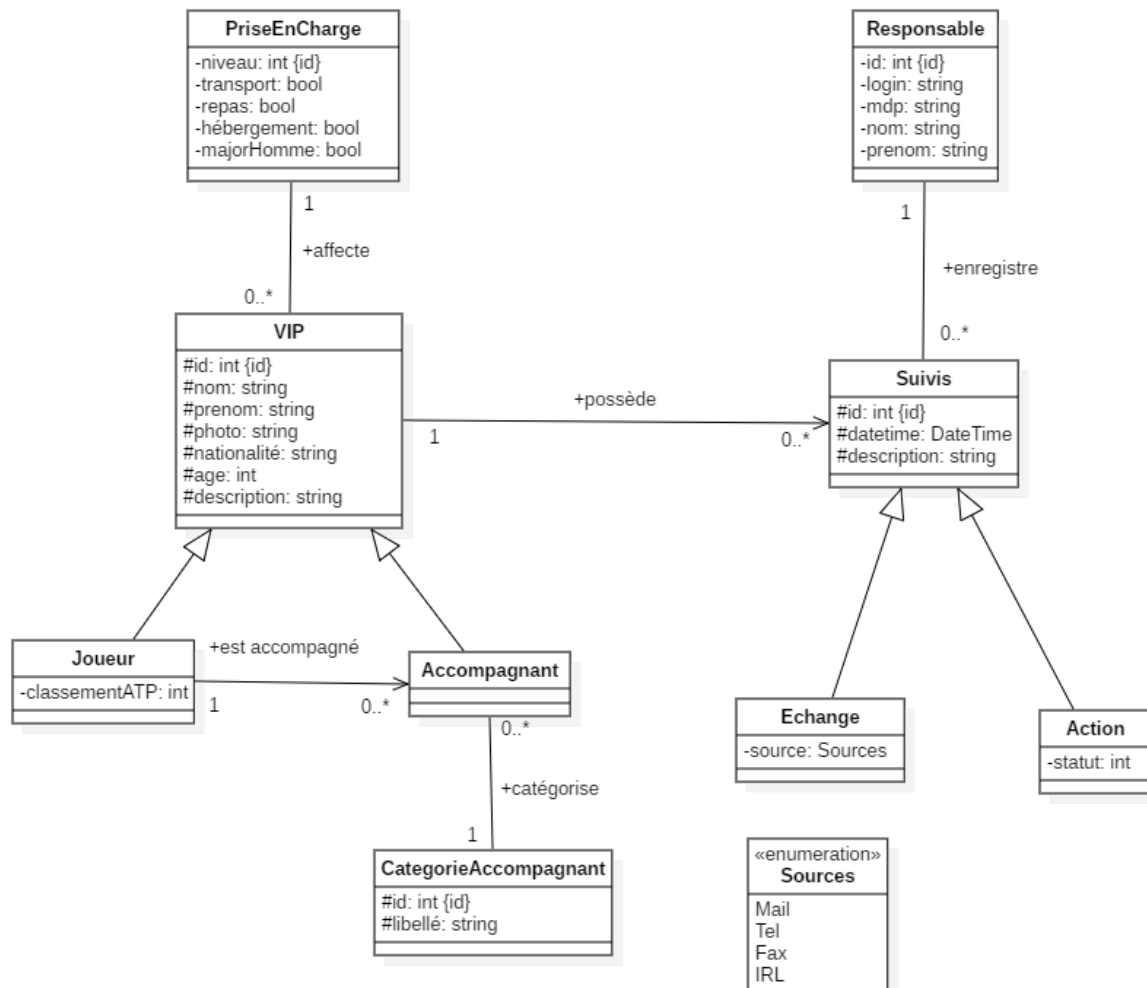


FIGURE 4.1 – DIAGRAMME DE CLASSES

5. Structure de la base de données

Étant donné que nous avons un diagramme de classe avec de l'héritage et une hiérarchie de classe concernant les VIPs et les Suivis, plusieurs options s'offrent à nous lors de sa conversion en diagramme de base de données avec de réelles tables. Nous avons opté pour celle de la « Table Simple ». En effet, une seule table va être créée, contenant à la fois les champs de la table mère et ceux des tables filles. Toutes ces colonnes peuvent être NULL, et on rajoute une colonne « type » afin de différencier les tables entre elles (dans le cadre des VIPs par exemple, la colonne type pourra prendre les valeurs : « Joueur » ou « Accompagnant »). Cela nous donne le schéma d'entités-relations suivant :

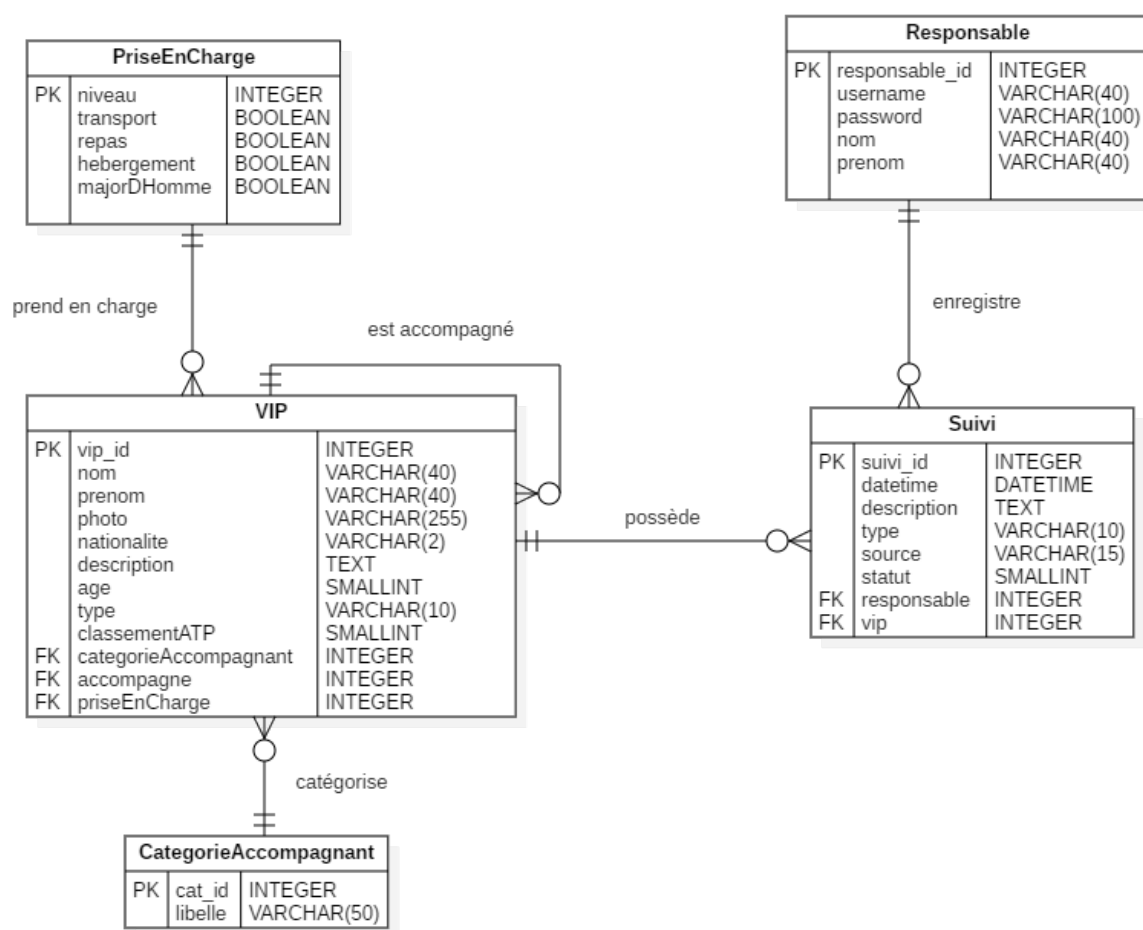


FIGURE 5.1 – DIAGRAMME D'ENTITES-RELATIONS REPRESENTATIF DE LA STRUCTURE DE LA BASE DE DONNEES

6. IHM

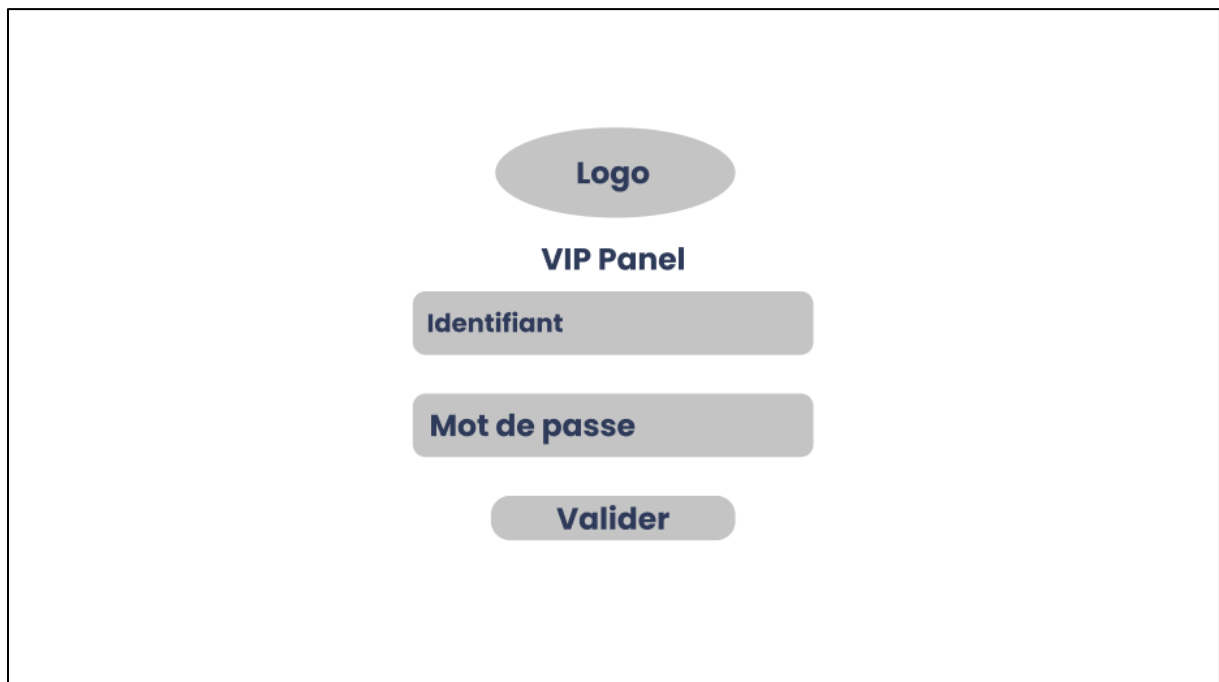


FIGURE 6.1 – PAGE DE CONNEXION



FIGURE 6.2 - DASHBOARD

[Dashboard](#) > [Liste VIP](#) > Ajouter VIP

Nom

Prénom

Date de naissance

Nationalité ▼



Description ▲




FIGURE 6.3 – AJOUT DE VIP

[Dashboard](#) > [Liste VIP](#) > NOM Prénom

Nom

Prénom



Date de naissance

Nationalité

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Sources ▼ 00/00/00

Description




FIGURE 6.4 – AFFICHAGE D'UN VIP

7. Architecture

Notre application est une application embarquée qui se connecte à un serveur de bases de données distant, via Internet, afin de récupérer les données. Ce qui nécessite aussi l'intégration d'un serveur web entre l'application client et le serveur de bases de données.

Il s'agit donc d'une architecture à **3 niveaux** :

- Le **client web** : conteneur de l'application, demandeur de ressource et produit final visible par l'utilisateur.
- Le **serveur web** : son rôle est d'interfacier le client et la base de données, au travers d'une API sécurisée permettant les différentes actions CRUD que l'on a besoin d'effectuer sur la base de données.
- Le serveur de **base de données**

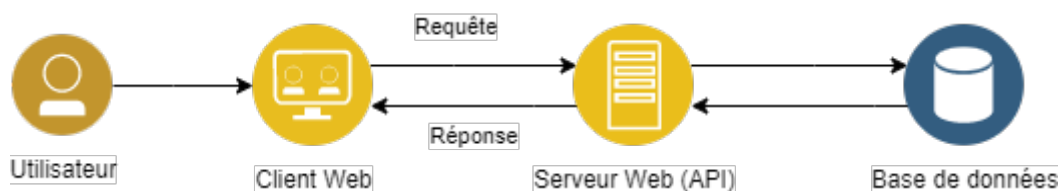


FIGURE 7.1 – DIAGRAMME DE L'ARCHITECTURE

8. Technologies



Next.js

Client

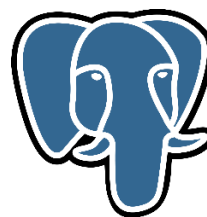
Next.js est une solution pour rendre des applications React côté serveur. Ainsi, nous créons une 4^{ème} couche dans notre architecture, à savoir le serveur web de Next.js, qui servira de « proxy » entre son serveur client et notre API afin d'améliorer la sécurité et la maintenabilité.



Symfony

Serveur (API)

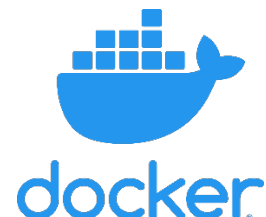
Symfony propose une solution simple quant à la réalisation d'APIs complexes et sécurisées.



PostgreSQL

Base de données

Nous avons choisi PostgreSQL comme base de données car elle est plus fiable, rapide et l'intégrité des données y est plus performante que d'autres SGBD comme MySQL ou Oracle.



Docker

Containerisation

Afin de fournir un environnement répliquable, stable, et scalable à l'infini, nous avons fait le choix de containeriser l'ensemble des composantes de l'application (Next.js, Symfony et PostgreSQL) dans des conteneurs Docker.