Dossier Projet

VETTESE Giovanni

Sommaire

I. Présentation....3
II. Présentation du projet....4
III. Cahier des charges....5
1. présentation d’entreprise....5
2. Utilisateurs du projet....6
3. Structure de l’application....6
4. Objectifs du projet....7
5. La cible adressée....8
6. Contraintes techniques....8
7. planning du projet....8
IV. Organisation du projet....9
1. Méthode de gestion de projet....9
2. Outils de versionning....10
V. Conception UX/UI....11
1. Charte graphique....11
2. Zoning et Wireframe....11
3. Maquette graphique....12
4. Responsive Design....13
VI. Conception base de données....14
1. Dictionnaire de données....14
2. MCD....15
3. MLD....18
VII. Conception application UML....21
1. Use Case....21
2. Diagramme de classe....22
VIII. Conception Multi couche....25
1. Architecture MVC....25
2. Architecture MVVM....26
3. Architecture 3 tiers....27
IX. Sécurité....28
1. Injections SQL....28
2. Injections XSS....29
3. Attaques CSRF....30
4. Politique de Mot de passe....31
X. Démonstration de L’application....32
1. Récupération de l’équipe du joueur....33
XI. Politique de test....37
XII. Veille technologique....38
XIII. Difficultés rencontrées....39
XIV. Conclusion....39

I. Présentation

I’m Giovanni VETTESE, I’m 24yo.
I live in Viry-Châtillon and I work in Giga-concept in Avrainville.

I work as software developers and work on different applications, the company is based on selling IOT solutions and sensors, the goal of one of the applications I worked on is to collect data from cars’ sensors, and add the sensor themselves to the database, and see if any information on movement is returned or if one sensor lost connection. There is some alarms if the sensors don’t send any information for a certain duration. Because the customer is Canadian we had to make the application multi language for english and french people.

I always love video games. I play since I’m young, I play different games big or little. I love big licenses with lot of history in there games, the more there is his history the more I like the game.

I’m in this school because I want to be a better developer.

I discover the computer science in high school and I knew that I wanted to do that. First of all, I wanted to work in big video games company like Ubisoft, Bethesda, CD Projeckt Red or Nintendo, but I learned the truth behind there beautiful games, all the problems of burn out and developers who we ask the impossible in a minimal time and the difficulty to reach the job of developers in these company. It’s really what I have in mind when I think of the work of my life, I like to be chill and programming peacefully.

So I searched for another type of development and found that hardware and software were interesting too and I begin to learn it. I begin with C and C++ and already love it since the beginning, I struggle because they’re difficult language but always love what I was doing. And later I learn some others programming language like JavaScript that I really like too, and learn Angular that I use in my project.

And now I’m going to introduce you my project.

II. Présentation du projet

Le projet est une plateforme de jeu local basée sur la licence bien connue Pokémon. Son objectif principal est de permettre aux joueurs de créer leur propre équipe de Pokémon et de les faire progresser en gagnant de l'expérience lors de combats tour par tour. Les joueurs ont également la possibilité d'acheter des Pokémon supplémentaires à l'aide d'une monnaie virtuelle disponible dans un magasin intégré à la plateforme.

L'un des aspects clés de ce projet est le système de combat tour par tour. Les joueurs peuvent affronter d'autres équipes de Pokémon contrôlées par un autre joueur en local ou par l'ordinateur. Pendant les combats, les joueurs peuvent choisir différentes actions pour leurs Pokémon, comme attaquer, se soigner, ou changer de Pokémon actif. Chaque Pokémon possède ses propres statistiques et types d'attaques ce qui ajoute de la stratégie et de la diversité aux combats.

Pour obtenir des fonds virtuels, les joueurs doivent remporter des victoires dans l'arène. Plus ils remportent de batailles, plus ils gagnent d'argent virtuel, ce qui leur permet d'acheter de nouveaux Pokémon dans le magasin intégré. Le magasin propose une variété de Pokémon avec des caractéristiques différentes, ce qui permet aux joueurs de personnaliser leur équipe en fonction de leur stratégie de jeu et de leurs préférences.

La plateforme de jeu offre également d'autres fonctionnalités pour améliorer l'expérience des joueurs. Cela peut inclure des fonctionnalités telles que la personnalisation des avatars des joueurs, des classements pour voir les meilleurs joueurs, des défis quotidiens ou hebdomadaires, des événements spéciaux, des récompenses et des réalisations à débloquer. Toutes ces fonctionnalités visent à rendre le jeu plus interactif, compétitif et addictif.

En résumé, ce projet est une plateforme de jeu local basée sur la licence Pokémon qui permet aux joueurs de créer leur équipe de Pokémon, de les faire progresser lors de combats tour par tour et d'acheter des Pokémon supplémentaires à l'aide d'une monnaie virtuelle gagnée en remportant des victoires dans l'arène. La plateforme offre également diverses fonctionnalités pour rendre l'expérience de jeu plus engageante et compétitive.

III. Cahier des charges

1. présentation d’entreprise

Bienvenue dans le monde passionnant de Game and Fans, une entreprise dynamique et créative qui a été fondée au début de l'année 2023 avec une vision claire en tête : créer des jeux indépendants innovants en utilisant des licences existantes. Dans un paysage vidéoludique où les grandes entreprises se concentrent de plus en plus sur des titres pour le grand public. Game and Fans se démarque en donnant priorité aux joueurs passionnés des grandes franchises.

L'une des valeurs fondamentales de Game and Fans est de raviver l'enthousiasme des joueurs nostalgiques en réinventant les jeux basés sur des licences classiques. Que ce soit un titre culte des années 80 ou une franchise bien-aimée des années 90, ou même des années 2000, l'équipe talentueuse de Game and Fans s'engage à préserver l'essence de ces univers tout en y apportant une touche fraîche et innovante.

Les créateurs de Game and Fans sont des passionnés de jeux vidéo. Ils ont grandi en jouant à des classiques intemporels qui ont marqué l'industrie et ont façonné leur amour pour cet art interactif. Conscients des attentes des joueurs chevronnés, ils sont déterminés à offrir des expériences de jeu authentiques, stimulantes et satisfaisantes.

L'équipe de développement de Game and Fans est composée d'artistes talentueux, de programmeurs expérimentés et de concepteurs de jeux visionnaires. Ensemble, ils travaillent en étroite collaboration pour capturer l'essence des licences existantes et les adapter de manière à ce qu'elles résonnent avec le public d'aujourd'hui. Leur passion pour les détails et leur engagement envers l'excellence transparaissent dans chaque aspect de leurs créations, des graphismes époustouflants aux mécaniques de jeu captivantes.

Game and Fans comprend également l'importance de la communauté des joueurs. Ils encouragent activement les retours des fans et créent un dialogue ouvert pour façonner leurs jeux. Cette approche collaborative permet à l'entreprise de s'assurer que chaque titre répond aux attentes des joueurs tout en proposant des surprises agréables et des innovations uniques.

En tant qu'entreprise indépendante, Game and Fans a la liberté de repousser les limites créatives et d'explorer de nouvelles idées audacieuses. Ils ne sont pas liés par des contraintes de marketing ou des pressions financières excessives, ce qui leur permet de prendre des risques calculés et d'offrir des expériences de jeu originales et captivantes.

L'objectif ultime de Game and Fans est de faire revivre la magie des jeux vidéo classiques, de susciter des souvenirs nostalgiques chez les joueurs et de créer de nouvelles expériences qui émerveilleront les générations actuelles et futures. Ils sont convaincus que leur passion pour l'industrie et leur engagement envers la qualité se traduiront par des jeux qui résonneront profondément auprès des fans du monde entier.

En conclusion, Game and Fans est une entreprise dynamique et passionnée qui s'efforce de raviver la flamme des anciens joueurs en créant des jeux indépendants basés sur des licences existantes. Leur approche centrée sur les joueurs, combinée à leur créativité sans limite, fait de Game and Fans un acteur unique dans l'industrie du jeu vidéo. Attendez-vous à être transporté dans des mondes fantastiques, à vivre des aventures inoubliables et à redécouvrir les joies intemporelles du jeu grâce aux titres exceptionnels de Game and Fans.

2. Équipe du projet

Les équipes du projet sont les développeurs/administrateurs de l’entreprise

Il existe une équipe de deux développeurs endossant également le rôle d'administrateur de leurs applications. Ils travaillent chacun sur un projet différent du mien, mais doivent être en mesure de répondre aux requêtes des utilisateurs émises sur mon projet.

3. Structure de l’application

dans un 1er temps l’application sera divisé en différentes parties :

- La page d’accueil
- La page de connexion
- un menu vers les autres pages ci dessous
- La page de gestion de l’équipe et visualisation de tout les pokémons de l’utilisateur
- La page de profil
- La page de combat
- La page du magasin
- La page de choix des pokémons pour le mode 2 joueurs

4. Objectifs du projet

Le projet à pour objectif de créer un jeu pokémon de stratégie où le joueur peut se battre contre des ordinateurs ou en local avec un 2ème joueur, et pourra acheter des autres pokémons depuis une boutique.

Le combat se déroule en tour par tour et les actions disponibles sont :

- attaquer (choix entre 4 attaques selon le pokémon)
- changer de pokemon
- se soigner (utilisable une fois tout les 3 tours)

Mais pour arriver jusque là un nouvel utilisateur aura plusieurs étapes avant son premier combat. Il devra se créer un compte en accepter les conditions générales d'utilisation, puis s'y authentifier. Ensuite pour sa 1ère connexion il devra choisir son 1er pokemon entre 3. Et enfin il pourra faire son 1er combat où il sera contre un adversaire avec comme lui 1 seul pokémon.

Après son 1er combat le joueur sera libre de faire ce qu'il souhaite dans l'application

Le joueur pourra accéder à son équipe et la gérer comme il la souhaite avec tout ses pokémons à disposition

Il pourra aussi accéder au magasin de pokémon où il pourra acheter des pokémons selon les 30 aléatoires pokémons disponibles et selon son argent.

Le joueur aura aussi un profil où il pourra voir ses résultat des combats précédents.

Tant qu'un joueur n'a pas si 6 pokémons à disposition il sera obligés de tous les mettre dans son équipe et les ordinateur en auront autant que lui.

Si un pokémon bat un pokémon adverse il gagnera un certain nombre d'expérience selon le niveau de son adversaire. Et si un joueur bat tout les pokémons d'un adversaire il gagnera de l'argent, sinon il ne se passe rien.

Si durant le combat un pokémon gagne assez d'expérience pour monter en niveau et que ce niveau lui permet d'évoluer, à la fin du combat ,gagner ou non, une demande sera faite au joueur s'il veut faire évoluer son pokémon ou s'il veut attendre et le pokémon évoluera ou non suite au choix.

Chaque pokémon aura un ou plusieurs type (par exemple normal, feu, eau, etc …..) et chaque type est plus fort ou plus faible par rapport à un autre

5. La cible adressée

L’application vise un public de tout âge intéressé par les jeux de stratégie. Une connaissance de la licence Pokémon n’est pas forcément nécessaire, toutes personnes peut jouer peut importe ses connaissances en jeux de stratégie ou jeux pokémons, mais un connaisseur aurait plus de facilités à comprendre les subtilités du jeu (le rapport de puissances entre les types par exemple).

6. Contraintes techniques

- Le projet sera fait en spa (Single Page Application) avec Angular pour le front et un back en nodejs .

- L'application doit être compatible avec les navigateurs Google Chrome, Firefox.

- L'application doit être sécurisée et ne pas contenir de failles susceptibles d'introduire des injections malveillantes (SQL, XSS...).

7. planning du projet

différentes versions seront vouées à sortir:

la v1 sortira le 21/08/23

le 20/09/23 pour la v2

la v3 sortira plus tard courant mars 2024

IV. Organisation du projet

1. Méthode de gestion de projet

j’ai utilisé la méthode kanban pour ce projet.

Qu’est-ce que la méthode kanban ?

La méthode Kanban est un système de gestion visuelle qui aide à améliorer la gestion des flux de travail et la productivité. Elle a été développée initialement par Toyota dans le cadre de son système de production, connu sous le nom de Toyota Production System (TPS). Depuis, la méthode Kanban a été adoptée et adaptée par de nombreuses entreprises dans divers domaines.

Pourquoi je l’ai utilisé ?

J’ai utilisé la méthode kanban afin de limiter le flux de travail et de ne pas commencer des tâches quand d’autres ne sont pas finis et définir les tâches ‘A faire’ ‘En cours’ ‘Terminé’

Je me suis appuyé sur l’application Trello pour appliquer cette méthode :

2. Outils de versionning

J’ai aussi utilisé l’outil de versioning Git pour les dossiers de mon projet afin de pouvoir travailler depuis différents ordinateurs et de pouvoir toujours garder une version récente de mon projet.

J’ai séparé mon projet en 3 grands dossiers, à savoir :

- le dossier de conception comportant tous les documents de la conception,
- le dossier front pour toute la partie Angular de l’application
- et le dossier API pour l’API utilisé dans la partie Angular

V. Conception UX/UI

Pour la conception graphique de mon projet j'ai utilisé des éléments de la licence pokémon comme la police du titre ou les images des pokémons et de terrain des jeux et ai accordé les couleurs

1. Charte graphique

J'ai donc commencé par établir une charte graphique.

C'est l'élément de conception graphique qui montre les différentes polices d'écritures et couleurs utilisées sur toute l'application

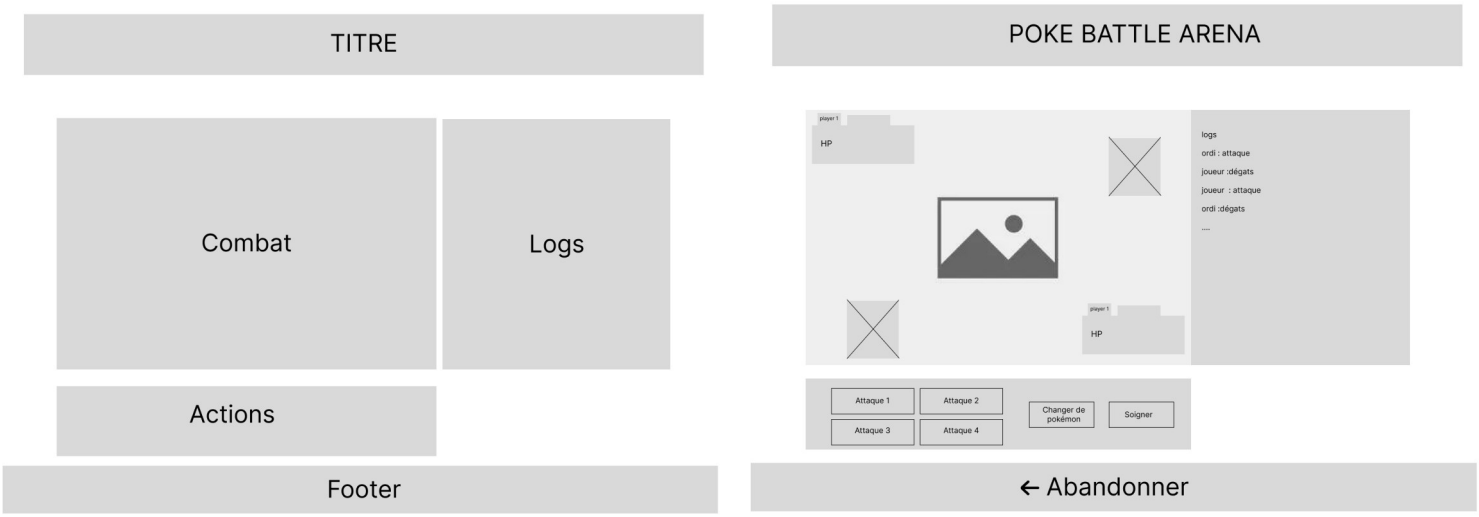
Voici donc ce que donne la charte graphique :

2. Zoning et Wireframe

J'ai continué avec le zoning et le wireframe.

Le zoning fait référence à la création d'une représentation visuelle préliminaire de l'agencement et de la disposition des éléments d'interface utilisateur sur une interface ou un écran. Tandis que le wireframe lui détailles les éléments décrits dans le zoning en rajoutant les textes et les emplacements des différents boutons et autres éléments à l'intérieur de l'élément principal.

Pour le design de la page de combat (qui reste la fonction principale du projet), j'ai mis l'écran de combat centrer sur la gauche avec les logs du combat collés à droite pour suivre en détail le combat. Les actions en dessous de la fenêtre de combat pour que le joueur accède rapidement à sa prochaine action. Et en pied de page un bouton de retour qui permettra d'abandonner le combat.



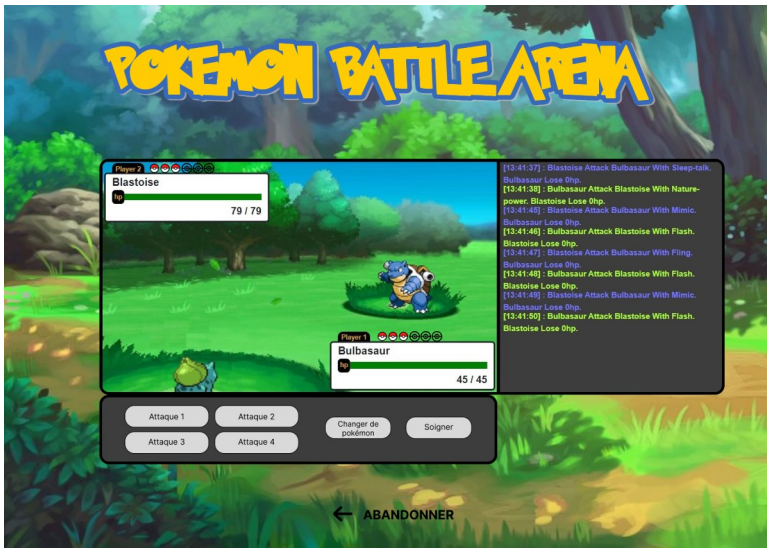
3. Maquette graphique

Ensuite, j'ai créé la maquette graphique :

La maquette graphique est l'étape finale de la conception graphique d'une interface, c'est la représentation statique d'une interface. Elle détaille les couleurs, la typographie et met en scène une version réaliste de ce que pourra visualiser un utilisateur.

Voici donc à quoi ressemble mon interface de combat avec les pokémons affichés de face ou de dos selon leur emplacement, leur barre de vie à côté d'eux avec au-dessus le nom du joueur et à droite son nombre de pokémon.

L'image du fond de l'ensemble de l'application et l'image du terrain pour le combat. Ainsi que l'ajout de la typographie du titre des jeux pokémon (voir annexe 1 pour l'image en grand)



4. Responsive Design

J’ai aussi créé une vue responsive de ce design

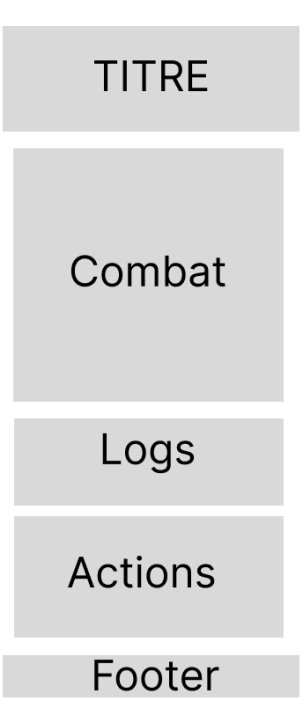
Une vue responsive est une vue adaptée à la taille de différents écrans comme ordinateur ou tablette ou smartphone. Elle permet une meilleure navigation selon le type d’écran.

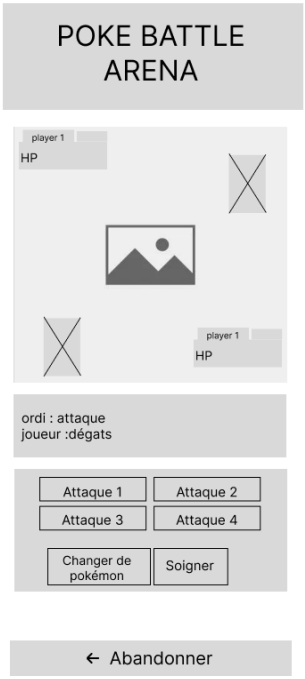
Après la vue pour ordinateur et tablette voici la vue pour smartphone de l’interface de combat.

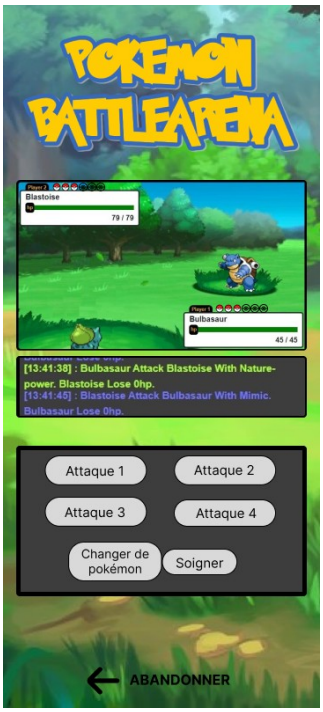
On peut remarquer tout d’abord que sur le zoning les logs du combat sont passés en dessous du cadre du combat pour que le cadre puisse prendre la largeur de l’écran et rester visible pour l’utilisateur. Le bloc des actions s’est, lui aussi, raccourci et décollé du cadre de combat, les attaques ont été centrée sur le bloc et les actions de « changer de pokémon » et « soigner » ont été mises en dessous.

Le titre lui a été coupé en 2 pour permettre un affichage plus propre.

L’interface prend la longueur de l’écran seulement pour éviter le scroll (faire dérouler l’écran) entre les actions.







VI. Conception base de données

Après le design je me suis mis à la conception de la bdd(base de données) et ai utilisé la méthode Merise.

Alors qu'est-ce que la méthode Merise ?

MERISE (Méthode d'Étude et de Réalisation Informatique pour les Systèmes d'Entreprise) est une méthode d'analyse, de conception et de gestion de projet informatique qui permet de conceptualiser et modéliser le fonctionnement de notre futur bdd en se concentrant sur la structuration des données et leurs relations.

Dans ce projet seront présentés un Dictionnaire de Données, un MCD et un MLD.

1. Dictionnaire de données

Le dictionnaire de données est un répertoire de tous les attributs utilisées dans le MCD

Num	Nom	Code	type	taille	decim...	Util...
1	user_id	USER_ID	Auto_increment			
2	username	USERNAME	Varchar	50		
3	poke_id	POKE_ID	Auto_increment			
4	name	NAME	Varchar	50		
5	money	MONEY	Int			
6	level	LEVEL	Int			
7	is_team	IS_TEAM	Bool			
8	is_shop	IS_SHOP	Bool			
9	price	PRICE	Int			
10	exp	EXP	Int			
11	poke_id_shop	POKE_ID_SHOP	Auto_increment			
12	shop_id	SHOP_ID	Auto_increment			
13	id_poke	ID_POKE	Int			
14	team_id	TEAM_ID	Auto_increment			
15	heal	HEAL	Int			
16	damage	DAMAGE	Int			
17	winner	WINNER	Varchar	50		
18	role	ROLE	Varchar	50		
19	password	PASSWORD	Varchar	50		
20	logs	LOGS	Text			
21	log_id	LOG_ID	Auto_increment			

2. MCD

Alors qu’est-ce qu’un MCD (modèle de données conceptuel) ?

Un MCD est un schéma conceptuel qui permet de définir les entités, les attributs et les relations entre les entités d'une manière indépendante de la technologie de base de données utilisée. Il s'agit donc d'une représentation haut niveau des concepts de données avant de les traduire en une structure concrète dans une base de données réelle.

Quels sont les différents éléments d’un MCD ?

- Entités : Ce sont les objets ou les concepts de l'application. Chaque entité possède des attributs qui décrivent les caractéristiques de cette entité.
- Attributs : Les attributs sont les propriétés ou les caractéristiques des entités. Par exemple, pour une entité "Client", les attributs pourraient inclure le nom, l'adresse et le numéro de téléphone.
- Relations **:** Les relations représentent les associations entre différentes entités. Elles définissent comment les entités interagissent les unes avec les autres.
- Cardinalités : Les cardinalités spécifient le nombre d'instances d'une entité qui peuvent être liées à une ou plusieurs instances d'une autre entité à travers une relation.

J’ai donc fait un MCD pour créer mes différentes entités et leurs relations :

- **User**
  - Description : Représente les utilisateurs du système.
  - Attributs :
    - user_id : Identifiant unique de l'utilisateur.
    - username : Nom d'utilisateur.
    - password : Mot de passe de l'utilisateur.
    - role : Rôle de l'utilisateur ("admin" ou "user").
    - money : Montant d'argent de l'utilisateur.
    - team_id : Référence à l'équipe à laquelle l'utilisateur appartient.
  - Cardinalités :
    - Chaque utilisateur peut posséder un ou plusieurs Pokémon.
    - Chaque utilisateur peut appartenir à une équipe (Team).
    - Chaque utilisateur peut avoir des logs de combat (BattleLogs) associés.
    - Chaque utilisateur peut avoir un contenu de boutique propre (Shop).

- **Team**
  - Description : Représente les équipes de Pokémon.
  - Attributs :
    - `team_id` : Identifiant unique de l'équipe.
    - `name` : Nom de l'équipe.
  - Cardinalités :
    - Chaque équipe peut avoir un ou plusieurs Pokémon.
    - Chaque équipe appartient à un utilisateur spécifique.

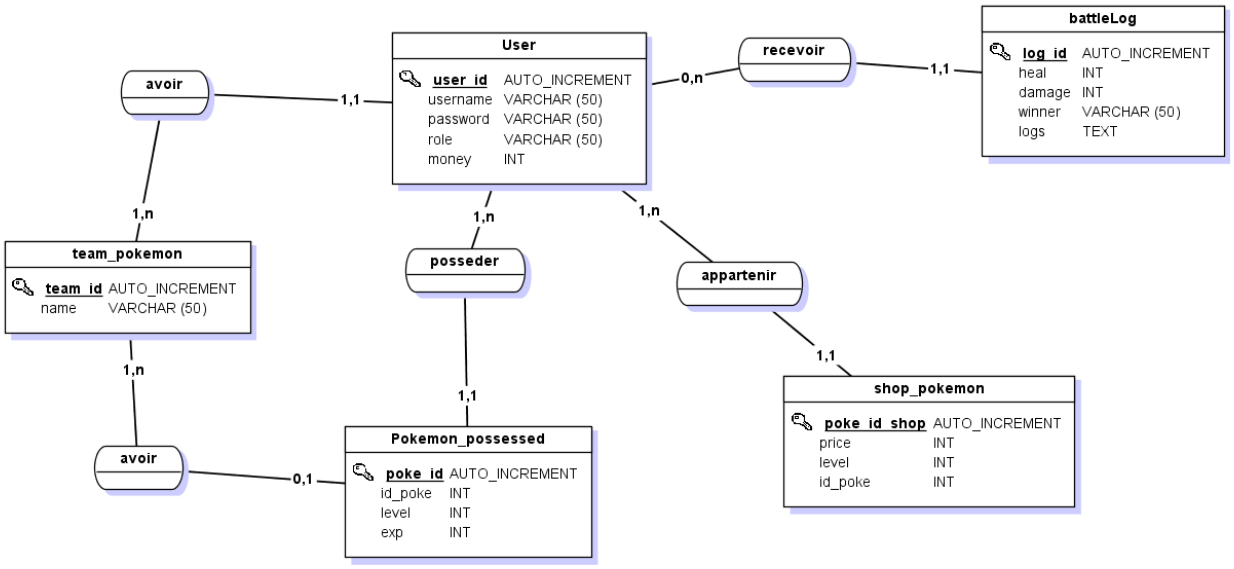
- **Pokemon**
  - Description : Représente les Pokémon possédés par les utilisateurs.
  - Attributs :
    - poke_id : Identifiant unique du Pokémon possédé.
    - id_poke : Identifiant du Pokémon.
    - level : Niveau du Pokémon.
    - exp : Points d'expérience du Pokémon.
    - user_id : Référence à l'utilisateur possédant le Pokémon.
    - team_id : Référence à l'équipe à laquelle le Pokémon appartient.
  - Cardinalités **:**
    - Chaque Pokémon peut être associé à une équipe ou ne pas en avoir.
    - Chaque Pokémon est lié à un utilisateur.

- **Shop**
  - Description : Représente les Pokémon disponibles à l'achat dans la boutique.
  - Attributs :
    - poke_id_shop : Identifiant unique du Pokémon en vente.
    - price : Prix du Pokémon.
    - level : Niveau du Pokémon en vente.
    - id_poke : Identifiant du Pokémon en vente.

- user_id : Référence à l'utilisateur responsable de la boutique.

- Cardinalités **:**
  - Chaque boutique est attribuée à un utilisateur spécifique.
  - Les boutiques contiennent un contenu spécifique lié aux objets à vendre.

- **BattleLogs**
  - Description : Représente les équipes de Pokémon.
  - Attributs :
    - log_id : Identifiant unique du log de combat.
    - heal : Montant de soins dans le combat.
    - damage : Dommages infligés dans le combat.
    - winner : Vainqueur du combat.
    - logs : Texte des journaux de combat détaillés.
  - Cardinalités **:**
    - Chaque logs est lié à un utilisateur spécifique.
    - Les logs enregistrent les détails des combats de l'utilisateur.



3. MLD

J’ai aussi fait un MLD (Modèle Logique de Données)

MLD (Modèle Logique de Données) est un concept utilisé en ingénierie logicielle et en conception de bases de données pour décrire la structure logique d'une base de données. Il s'agit d'une représentation schématique des entités, des attributs, des relations et des contraintes qui régissent les données à l'intérieur d'une base de données. Le MLD permet de modéliser de manière abstraite et indépendante des détails de mise en œuvre les relations et les propriétés des données, fournissant ainsi un guide pour la création d'une base de données relationnelle cohérente et normalisée. En d'autres termes, le MLD est une étape intermédiaire entre le modèle conceptuel de données (MCD) et la mise en œuvre réelle de la base de données.

Les entités sont devenues des tables et il a fallu définir définir les différentes clés étrangères qui seront dans la bdd (base de données)

De ce fait nous avons :

- Table : team_pokemon
  - Description : Cette table représente les équipes de Pokémon.
  - Colonnes :
    - team_id (Clé primaire, auto-incrémentée) : Identifiant unique de l'équipe.
    - name : Nom de l'équipe.

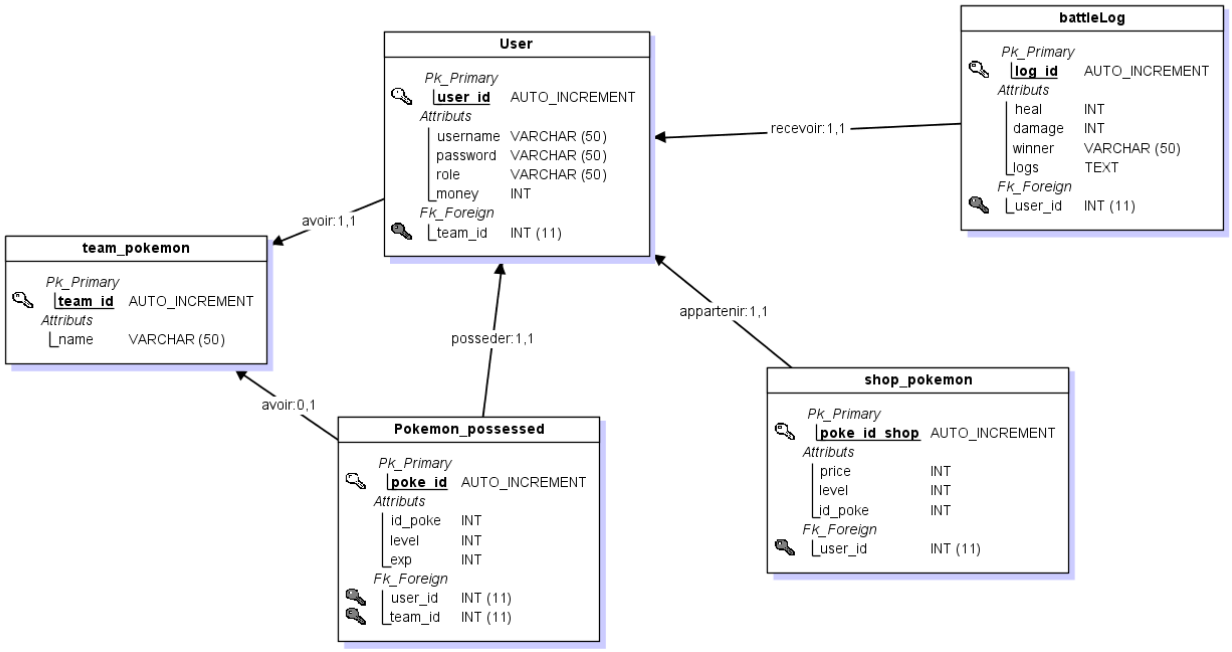
- Table : User
  - Description : Cette table représente les utilisateurs du système.
  - Colonnes :
    - user_id (Clé primaire, auto-incrémentée) : Identifiant unique de l'utilisateur.
    - username : Nom d'utilisateur.
    - password : Mot de passe de l'utilisateur.
    - role : Rôle de l'utilisateur (peut être "admin", "player", etc.).
    - money : Montant d'argent de l'utilisateur.
    - team_id (Clé étrangère) : Référence à l'équipe à laquelle l'utilisateur appartient (Cardinalité : 1 utilisateur - 1 équipe).

- Table : Pokemon_possessed
  - Description : Cette table représente les Pokémon possédés par les utilisateurs.
  - Colonnes :
    - poke_id (Clé primaire, auto-incrémentée) : Identifiant unique du Pokémon possédé.
    - id_poke : Identifiant du Pokémon selon les jeux originaux
    - level : Niveau du Pokémon.
    - exp : Points d'expérience du Pokémon.
    - user_id (Clé étrangère) : Référence à l'utilisateur possédant le Pokémon (Cardinalité : 1 utilisateur - N Pokémon).
    - team_id (Clé étrangère) : Référence à l'équipe à laquelle le Pokémon appartient (Cardinalité : 1 équipe - N Pokémon, peut être nul).

- Table : shop_pokemon
  - Description : Cette table représente les Pokémon disponibles à l'achat dans la boutique.
  - Colonnes :
    - poke_id_shop (Clé primaire, auto-incrémentée) : Identifiant unique du Pokémon en vente.
    - price : Prix du Pokémon.
    - level : Niveau du Pokémon en vente.
    - id_poke : Identifiant du Pokémon en vente.
    - user_id (Clé étrangère) : Référence à l'utilisateur responsable de la boutique (Cardinalité : 1 utilisateur - N Pokémon en vente).

- Table : battleLog
  - Description : Cette table enregistre les journaux de combat.
  - Colonnes :
    - log_id (Clé primaire, auto-incrémentée) : Identifiant unique du log de combat.
    - heal : Montant de soins dans le combat.
    - damage : Dommages infligés dans le combat.
    - winner : Vainqueur du combat.

- logs : Texte des journaux de combat détaillés.
- user_id (Clé étrangère) : Référence à l'utilisateur lié au log de combat (Cardinalité : 1 utilisateur - N journaux de combat).



VII. Conception application UML

UML (Unified Modeling Language) est un langage de modélisation visuelle utilisé dans le domaine du génie logiciel pour représenter graphiquement des systèmes logiciels. Il propose des notations standardisées pour décrire les différentes perspectives d'un système, telles que sa structure, son comportement, ses interactions et ses processus, en utilisant des diagrammes tels que les diagrammes de classes, de séquence, d'activités, etc. UML facilite la communication entre les membres d'une équipe de développement en fournissant un moyen commun de représenter et de comprendre la conception et le fonctionnement d'un système logiciel.

1. Use Case

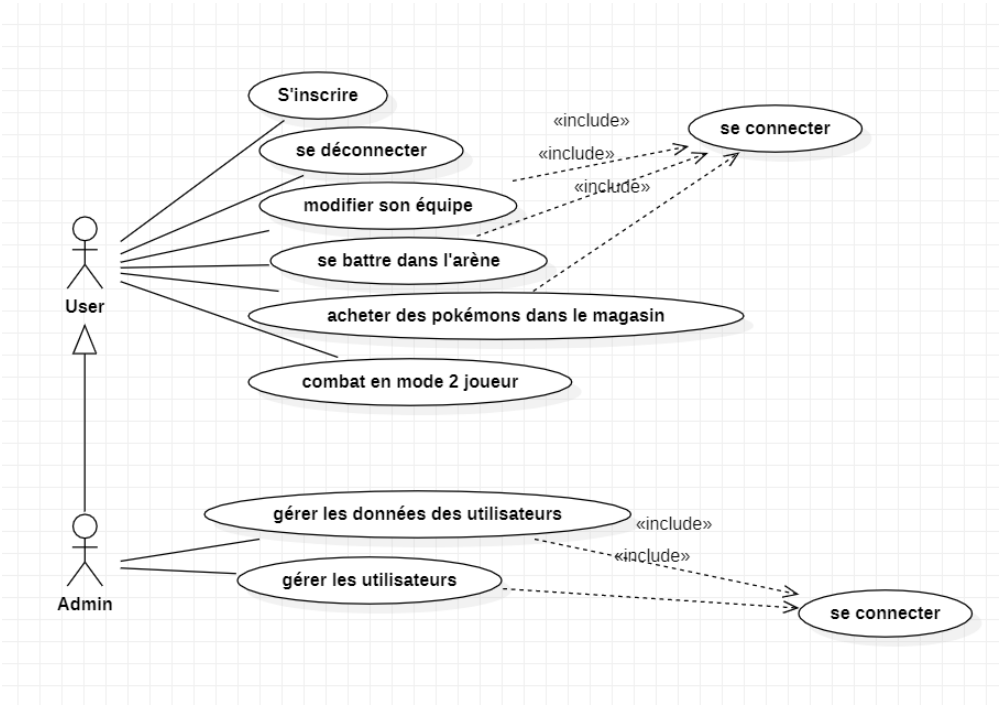
Le diagramme de use case, ou cas d'utilisation, représente toutes les actions possibles pour un utilisateur selon son rôle.

Alors tout d'abord qu'est-ce qu'un utilisateur, il représente n'importe quel visiteur qui arrive sur l'application. Les rôles sont administrateur et user, un user est un utilisateur quelconque tandis que l'administrateur est une personne est ayant des codes d'accès supérieur à un utilisateur normal connecté.

Ici, un utilisateur normal pourra s'inscrire, se connecter et se déconnecter. De plus, il pourra aussi accéder au mode de combat à 2 joueurs sans se connecter. Le reste des fonctionnalités : gérer son équipe depuis la page « Team », se battre dans l'arène avec le combat et acheter des pokémons depuis le magasin, nécessiteront à l'utilisateur de se connecter avant.

Un administrateur, lui, en plus d'avoir accès à toutes les fonctionnalités utilisateurs aura aussi accès à des informations sur les utilisateurs et pourra les gérer

Le « include » signifie que le cas est inclut t à l'initialisation de la fonctionnalité.



2. Diagramme de classe

Un diagramme de classe sert à définir les différentes classes qui seront utilisé dans un projet orienté POO (programmation orienté objet) et le lien qu'il y a entre elles.

Commençons par expliquer ce qu'est une classe. Une classe est un modèle ou un plan utilisé pour créer des objets. Elle définit la structure et le comportement d'un type d'objet particulier. Une classe agit comme un modèle à partir duquel on peut créer des instances, également appelées objets.

Une classe contient généralement des attributs (variables) et des méthodes (fonctions) qui définissent les caractéristiques et les comportements des objets créés à partir de cette classe. Les attributs représentent les données associées à l'objet, tandis que les méthodes définissent les actions que l'objet peut effectuer.

Sur le diagramme sont représentés différentes propriétés de visibilités :

- private les « - »
- public les « + »

Il en existe un 3ème « Protected » que je n'ai pas utilisé, car il est sans intérêt pour les classes utilisées. Cette visibilité est surtout utile parce qu'elle permet l'accès aux données ayant ce champ entre une classe mère sur une classe fille lors d'un héritage, mais reste inaccessible pour toutes autres utilisations en dehors de l'héritage.

Un attribut ou une méthode privée est un élément qui n'est pas accessible depuis l'extérieur de la classe contrairement à un attribut ou une méthode publique qui sera lui accessible à chaque instance (ou objet) de la classe.

Les différentes classes ont chacune leurs attributs et pour certaines des méthodes.

- **La classe User** représente l'utilisateur connecté peu importe son rôle et à donc les propriétés utiles à un utilisateur à savoir un id :
  - un nom (username),
  - un mot de passe, tous les trois de type String (chaîne de caractère),
  - un tableau de pokémon pour les pokémons qu'il possède
  - une équipe (team)

- un porte-monnaie virtuel pour le magasin.

Ainsi que ses méthodes, « login » pour se connecter et « logout » pour se déconnecter et gainMoney pour lui faire gagner de l'argent.

- **La classe Pokemon** indique les différents pokémons présents dans l'application qui a pour attributs :
  - un id,
  - un nom,
  - ainsi que les différentes statistiques d'un pokémon d'après les jeux originaux: hp, hp max, attaque, défense, attaque spéciale, défense spécial, vitesse, type, compétences (moves), niveau, expérience, expérience max, et son évolution s'il en a une.

  Elle a aussi des méthodes comme :
  - battle pour se combattre contre d'autres pokémons
  - leveling pour le faire monter de niveau si son expérience atteint l'expérience max
  - evolution pour le faire évoluer s'il à atteint un niveau d'évolution
  - isDead pour vérifier si les point de vie (hp) ont atteint 0 et donc qu'il ne peut plus combattre
  - randomMove pour choisir une compétence aléatoire parmi les siennes.

- La classe Move pour les capacités d'un pokémon à :
  - un nom
  - une précision
  - une puissance
  - un nombre d'utilisation (pp)
  - un type

- La classe Team qui désigne l'équipe d'un joueur utilisateur ou ordinateur, elle a :
  - un nom
  - un id
  - un tableau de pokémon

  Les méthodes de cette classe sont :

- addPoke afin d’ajouter un pokémon à l’équipe s’il y a de la place
- deletePoke pour supprimer un pokémon de cette équipe.

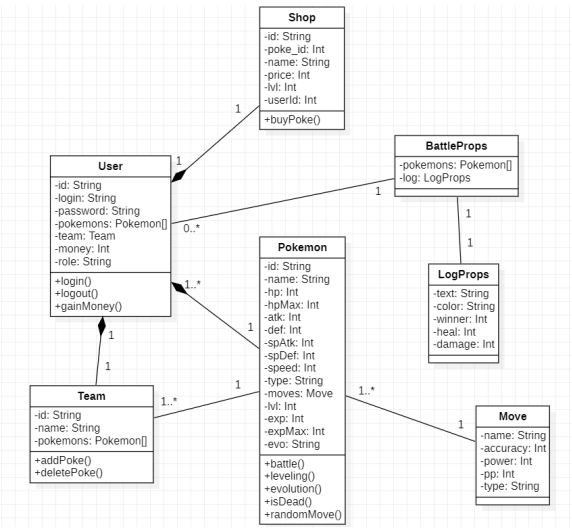
- La classe Shop pour un pokémon du magasin qui à des attributs différents d’un pokémons pour un joueur, elle a :
  - un id
  - un id de pokémon
  - le nom du pokémon
  - son prix
  - son niveau

  Cette classe à une seule méthode buyPoke pour acheter un pokémon si l’utilisateur à assez d’argent.

- La classe BattleProps regroupe les log de combats et les pokémons de ce même combat, il a donc :
  - une tableau de pokémon
  - des log

- La classe LogsProps représente les log de combat composé :
  - d’un texte
  - du gagnant
  - de son taux de soin
  - de son taux de dégâts à l’adversaire.

(voir annexe 2 pour l’image en grand)



VIII. Conception Multi couche

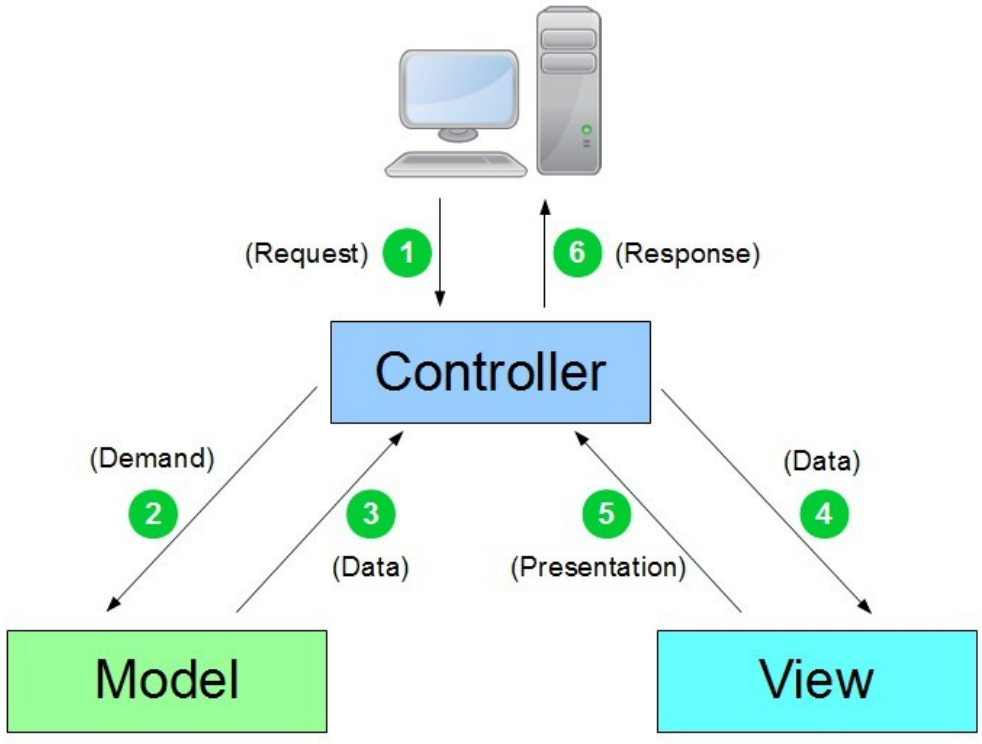
1. Architecture MVC

Commençons par expliquer ce qu'est une architecture MVC (Modèle-Vue-Contrôleur) est un concept utilisé dans le développement logiciel pour organiser les composants d'une application de manière à faciliter la gestion des données, les interactions utilisateur et la logique métier.

Un modèle représente les données de l'application ainsi que la logique associée à leur manipulation. Il peut être considéré comme une représentation des données sous-jacentes, telles que les informations d'une base de données. Le modèle gère l'accès, la validation et la mise à jour des données. Il communique avec le contrôleur pour notifier les changements de données et avec la vue pour fournir les données à afficher.

La vue gère l'interface utilisateur et l'affichage des données. Elle est responsable de la présentation visuelle des informations aux utilisateurs. La vue reçoit les données du modèle et les affiche de manière appropriée. Elle ne contient généralement pas de logique métier complexe, se concentrant plutôt sur la mise en forme des données pour une expérience utilisateur optimale.

Le contrôleur agit comme un intermédiaire entre la vue et le modèle. Il reçoit les entrées de l'utilisateur depuis la vue, interprète ces actions et décide comment réagir en conséquence. Le contrôleur coordonne les actions de l'application en interagissant avec le modèle pour effectuer les opérations nécessaires. Il met à jour le modèle en fonction des interactions de l'utilisateur et détermine quelle vue doit être affichée.



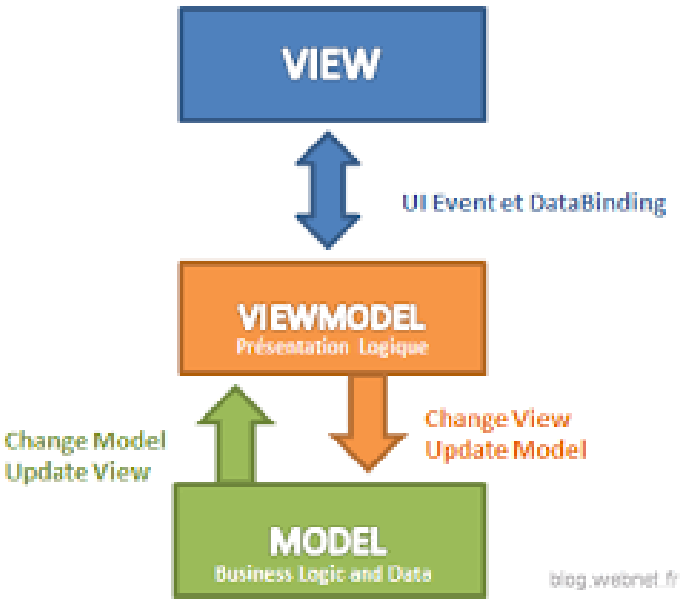
2. Architecture MVVM

L'architecture MVVM (Modèle-Vue-VueModèle) est un modèle de conception utilisé dans le développement logiciel pour organiser les composants d'une application en séparant les responsabilités liées aux données, à l'interface utilisateur et à la logique métier. Voici une définition en français de chaque composant de l'architecture MVVM :

Le modèle représente les données et la logique associée à leur manipulation. Il s'agit généralement d'une représentation des informations sous-jacentes, telles que celles stockées dans une base de données ou récupérées depuis des services. Le modèle gère l'accès aux données, les opérations de lecture/écriture et la validation.

La vue est responsable de l'interface utilisateur et de l'affichage des données. Elle affiche les informations aux utilisateurs et réagit aux interactions de l'utilisateur. Contrairement à l'architecture MVC, la vue MVVM est généralement plus légère en termes de logique métier. Elle se concentre principalement sur l'affichage des données et sur la liaison avec la VueModèle pour récupérer les données à afficher.

La VueModèle agit comme un intermédiaire entre la vue et le modèle. Il contient la logique métier et les informations nécessaires pour rendre les données du modèle compatibles avec l'affichage dans la vue. La VueModèle prépare et formate les données spécifiquement pour l'interface utilisateur. Il gère également les interactions de l'utilisateur avec les données, comme les actions et les commandes. Dans certaines implémentations, la VueModèle peut également gérer la communication avec le modèle et les services.



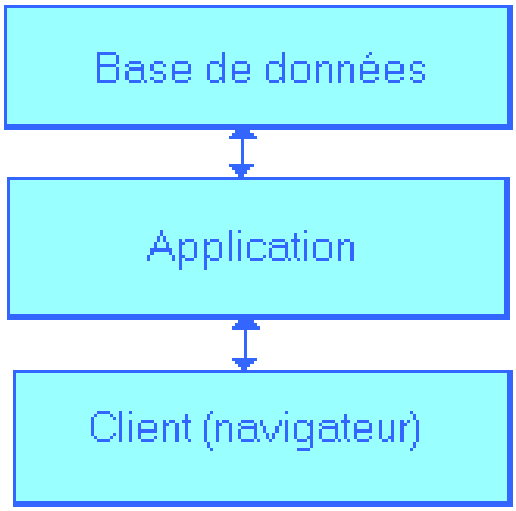
3. Architecture 3 tiers

L'Architecture 3 tiers est une architecture qui présente 3 couches distinctes. Une couche de présentation, une couche de traitement et une couche d'accès aux données.

La couche de présentation est la partie qui voit un client sur son navigateur, le front, dans mon projet elle représente la partie Angular. Elle est responsable du visuel et de l'interface de l'application. Elle communique avec l'API, l'application, pour envoyer et récupérer les informations entre l'application et l'interface client, mais ne gère pas directement l'accès aux données avec la base de données. Elle s'occupe simplement de transformer les informations de sortes à être compréhensible par l'utilisateur.

La couche de traitement, ici mon API (NodeJS Express), s'occupe de la partie back-end, la logique métier de l'application. Elle gère les requêtes envoyées par la couche de présentation et exécute le traitement de différentes opérations comme l'authentification, validation de la requête ou la validation d'accès pour un utilisateur à certaines données privées. C'est elle qui renvoie ensuite les données au client.

La couche d'accès aux données est la partie qui gère l'accès à la base de données, dans mon projet ça correspond à l'ORM (Sequelize) que j'ai utilisé. Elle remplit les requêtes validées par l'API et renvoi ou modifie les données de la base de données selon la requête du client.



IX. Sécurité

1. Injections SQL

Qu’est-ce qu’une injection SQL ?

Une injection SQL consiste à exploiter une faille dans l’application pour modifier les requêtes SQL et manipuler des informations de la BDD autres que les manipulations accessibles normalement.

Il existe différent types d’ injections SQL :

- Accès non autorisé aux données : Un attaquant peut exploiter une injection SQL pour accéder aux données sensibles de la base de données, telles que les informations d'identification, les données financières, etc.
- Divulgation d'informations : L'attaquant peut extraire des informations confidentielles de la base de données, ce qui peut compromettre la sécurité des utilisateurs et de l'application elle-même.
- Altération des données : Les attaquants peuvent modifier, supprimer ou altérer les données stockées dans la base de données, causant des dommages et des perturbations.
- Prise de contrôle du système : Dans les cas graves, une injection SQL réussie peut permettre à un attaquant de prendre le contrôle du système sur lequel l'application est exécutée.

Alors comment s’en protéger ?

On peut se protéger des injections SQL en faisant des requêtes préparées de sorte qu'aucune autres informations que celles voulues ne soient dans la requête.

Mon ORM (Sequelize) prends les requêtes SQL préparé par défaut pour éviter à la base les failles d’injections

Comme le montre cette capture d écran d’une de mes requêtes, elle va uniquement chercher dans la table user et lui envoi uniquement l’information d’un user préalablement validé.

2. Injections XSS

Une faille XSS (Cross-Site Scripting) est une injection permettant de rajouter du contenu à une page web pour provoquer des actions malveillantes sur le client et ses informations

Il existe trois types principaux d'injections XSS :

- Stocké (Stored) XSS : Dans cette variante, le script malveillant est injecté dans une application et stocké dans la base de données ou sur le serveur. Lorsque d'autres utilisateurs consultent la page contaminée, le script est chargé et exécuté dans leurs navigateurs.
- Réfléchi (Reflected) XSS : Dans ce scénario, le script malveillant est injecté dans une URL ou un formulaire et renvoyé au navigateur de l'utilisateur ciblé via une redirection ou une réponse du serveur. L'utilisateur clique sur un lien ou soumet un formulaire contenant le code malveillant, déclenchant ainsi son exécution.
- Dom-Based (Document Object Model) XSS : Cette variante exploite les manipulations du DOM, qui est une représentation en mémoire de la structure d'une page web. Le script malveillant modifie le DOM de la page pour exécuter du code côté client sur le navigateur de la victime.

Comment s’en protéger ?

Il faut échapper ses données en entrée et en sortie

Comme fait ici par exemple

```
<input type="text" name="userName" placeholder="userName" [formControlName]="'userName'" required>
```

les « [] » autour de formControlName sont des directives de liaison d'Angular et permettent que les données soientt correctement encodées pour éviter les attaques XSS.

Ou comme ceci

```
<p>name: {{pokemon.name}} </p>
```

les « {{}} » sont des interpolations sécurisées dans Angular et elles encodent automatiquement les données insérées dans les templates HTML pour empêcher l'exécution de scripts malveillants

3. Attaques CSRF

Une attaque CSRF (Cross-Site Request Forgery) est une vulnérabilité de sécurité dans les applications web où un attaquant exploite la confiance d'un utilisateur authentifié pour effectuer des actions non autorisées en son nom. Cette attaque se produit lorsque des requêtes non sollicitées et malveillantes sont exécutées à partir du navigateur d'une victime, généralement sans que cette dernière en soit consciente.

Le scénario typique d'une attaque CSRF implique les étapes suivantes :

- L'utilisateur authentifié se connecte à un site web A.
- Pendant la session active de l'utilisateur sur le site A, celui-ci visite un site malveillant B (contrôlé par l'attaquant) ou clique sur un lien ou une image malveillante.
- Le site malveillant B contient des requêtes HTTP (comme des formulaires) destinées au site A. Ces requêtes visent à effectuer des actions spécifiques au nom de l'utilisateur sans son consentement.
- Les requêtes sont envoyées au site A à partir du navigateur de l'utilisateur, avec les cookies d'authentification attachés. Puisque l'utilisateur est déjà authentifié sur le site A, le serveur du site A accepte les requêtes et effectue les actions demandées.

Comment s’en protéger ?

Ajouter un token CSRF généré pour protéger mon token JWT lors de ma connexion afin d’avoir une connexion plus sécurisé

4. Politique de Mot de passe

La CNIL recommande des mot de passe de 14 caractères minimum avec majuscule, minuscule et des chiffres.

Il est aussi préférable de hacher son mot de passe plutôt que de le crypter.

Un mot de passe haché (ou hash de mot de passe) est une représentation cryptographique d'un mot de passe en tant que résultat d'une fonction de hachage. Une fonction de hachage est un algorithme qui prend une entrée (dans ce cas, un mot de passe) et génère une sortie fixe de longueur fixe, souvent sous forme d'une série de caractères alphanumériques. L'objectif principal de l'utilisation d'une fonction de hachage est de protéger les mots de passe en stockant des valeurs illisibles et difficiles à inverser.

Contrairement au chiffrement, où il est possible de revenir à la valeur d'origine (décrypter) en utilisant une clé, les fonctions de hachage ne sont pas conçues pour être inversées. Lorsque vous hachez un mot de passe, vous ne pouvez pas récupérer le mot de passe d'origine à partir du hachage. Cela rend les mots de passe hachés plus sécurisés en cas d'exposition accidentelle ou de violation de données, car les attaquants ne peuvent pas simplement décrypter les hachages pour obtenir les mots de passe réels.

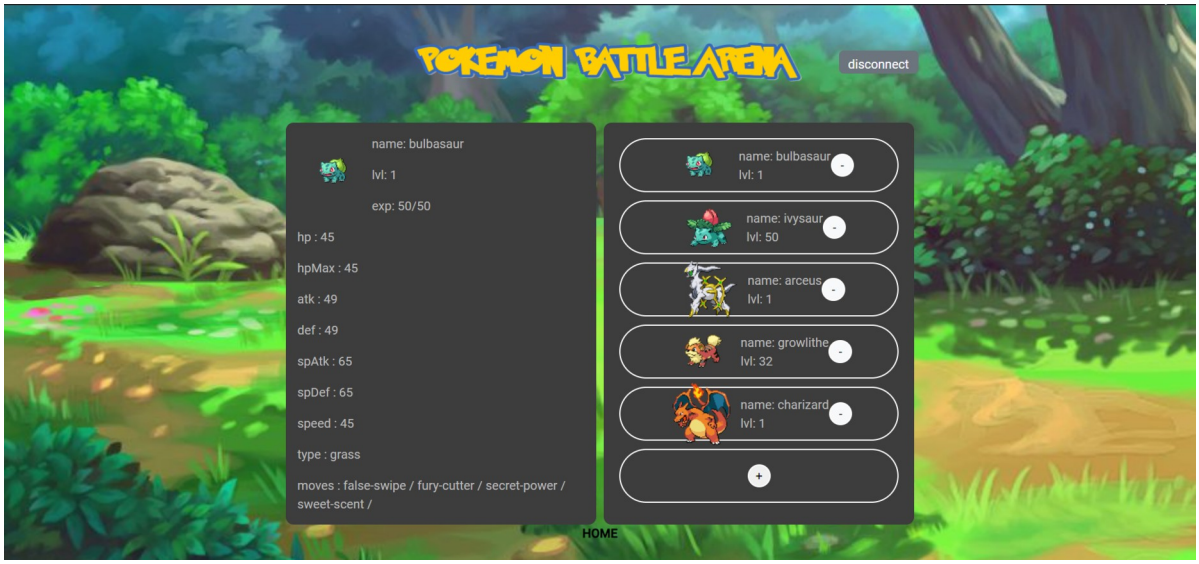
Mon mot de passe est donc haché à l’aide du package JS Bcrypt

```
const saltRounds = 8;
user.password = await bcrypt.hash(user.password, saltRounds);
```

X. Démonstration de L’application

Prenons l’exemple de la Team en montrant tout ce qu’il se passe depuis le front jusqu’à la requête en BDD.

Alors tous d’abord nous commençons sur la vue de l’équipe :




```
<div class="d-flex flex-row" style="width: 100%;">
    <div class="d-flex flex-column box" *ngIf="pokemon">
        <div class="d-flex flex-row">
            <div>
                <img src="https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/{{pokemon.id}}.png"
                    alt="">
            </div>
            <div>
                <p>name: {{pokemon.name}} </p>
                <p>lvl: {{pokemon.lvl}}</p>
                <p>exp: {{pokemon.exp}}/{{pokemon.expMax}}</p>
            </div>
        </div>
        <div style="display: flex; flex-direction: column; flex-wrap: wrap;">
            <p *ngFor="let stat of stats"> {{stat.name}} : {{stat.key}}</p>
            <span>moves :
                <span *ngFor="let m of pokemon.moves"> {{m.name}} /</span>
            </span>
        </div>
    </div>

    <div class="d-flex flex-column box">
        <div class="poke" *ngFor="let poke of [].constructor(nbpoke); let i=index">
            <div *ngIf="pokeTeam.pokemons[i]" (click)="onClick(i)" class="d-flex flex-row align-items-center">
                <div>
                    <img src="https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/{{pokeTeam.pokemons[i].id}}.png"
                        alt="">
                </div>
                <div class="d-flex flex-column justify-content-center">
                    <span>name: {{pokeTeam.pokemons[i].name}} </span>
                    <span>lvl: {{pokeTeam.pokemons[i].lvl}}</span>
                </div>
                <button class="btn btn-light btnadd" (click)="onDelete(i,pokeTeam.pokemons[i])">
                    -
                </button>
            </div>
        </div>
        <div (click)="onAdd()" class="poke" *ngFor="let poke of [].constructor(6-nbpoke)">
            <button class="btnadd">+</button>
        </div>
    </div>

</div>
```

1. Récupération de l’équipe du joueur

Afin de pouvoir de récupérer l’équipe qui est affiché je fais appelle à la fonction ngOnInit des composants Angular

C’est une fonction qui s’exécute au lancement d’un composant en l’occurrence le composant Team

```
ngOnInit(): void {
  this.user = JSON.parse(localStorage.getItem('user')!);
  console.log(this.user);

  this.pokeTeam.id = this.user.teams[0].id;
  this.teamService.getTeam(+this.user.id).subscribe((v) => {
    // console.log(v.pokemons);
    let teamPoke: any[] = v.pokemons;
    teamPoke.forEach((poke) => {
      this.pokeService.getPokemonById(poke.id_poke).subscribe((pokemon) => {
      this.pokeTeam.pokemons.push(pokemon);
      this.pokeTeam.pokemons.find((x) => x.id == poke.id_poke)!.lvl =
        poke.lvl;
      this.pokeTeam.pokemons.find((x) => x.id == poke.id_poke)!.exp =
        poke.exp;
      this.pokeTeam.pokemons.find((x) => x.id == poke.id_poke)!.expMax =
        poke.expMax;
      // console.log(this.pokeTeam.pokemons);
      this.nbpoke = this.pokeTeam.pokemons.length;
      this.pokemon = this.pokeTeam.pokemons[0];
    });
  });
  });
```

Dans cette fonction je fais appel au service de la team pour envoyer une requête HTTP à mon api qui va elle lancer une fonction pour envoyer une requête à ma BDD.

Service Angular :

```
getTeam(id: number) {
  return this.http.get<any>(environment.apiUrl + 'team/getById/' + id);
}
```

Route Api :

```
app.use("/team", teamrouter);
```

```
teamrouter.get("/getById/:id", getTeamById);
```

Controller Api :

```
export const getTeamById: RequestHandler = async (req, res, next) => {
  const id = req.params.id;
  const team = await teamServices.getTeamById(id);
  return res.status(200).json(team).end();
};
```

Service Api :

```
export async function getTeamById(id: string) {
  try {
    return await Team.findOne({ where: { id }, include: {model: Pokedb} });
  } catch (error) {
    throw error;
  }
}
```

2. Ajout d’un pokémon à l ‘équipe

Ensuite nous voulons ajouter un pokémon à l’équipe en appuyant sur la case vide avec le « + »

S’il n’y a pas de case vide, c’est que l’équipe est au complet

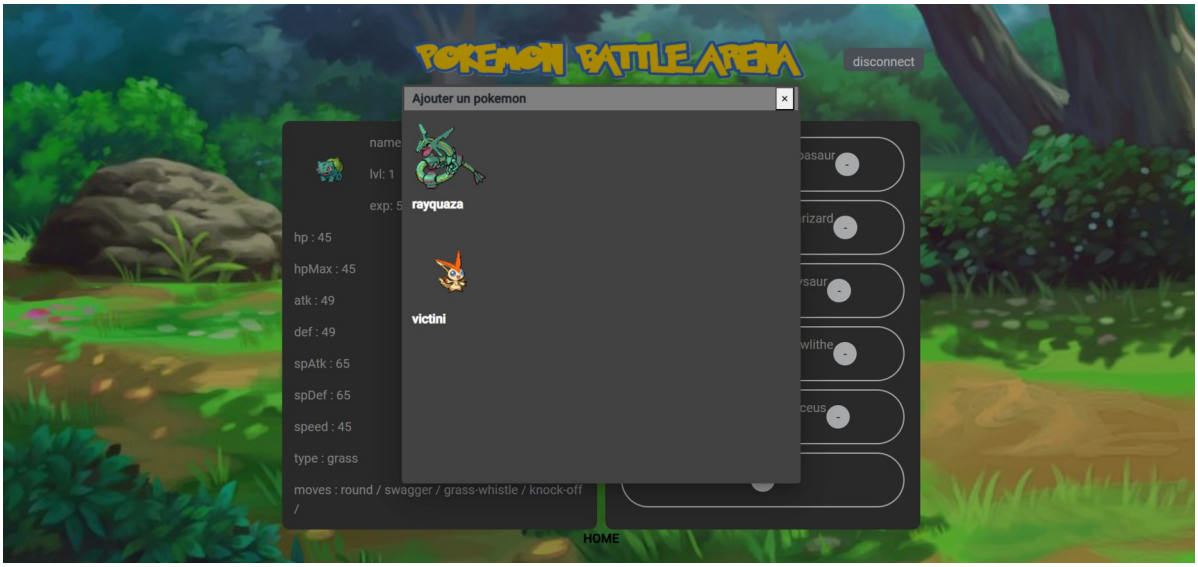
une fois qu’on a cliqué sur la case vide un fonction se lance

```
onAdd() {
  const dialogRef = this.dialog.open(DialogPokeComponent, {
    data: this.pokeCardFilter,
  });

  // console.log('pokecard ', this.pokeCard);

  dialogRef.afterClosed().subscribe((result) => {
    // console.log('The dialog was closed', result);
    if (result) {
      this.pokedb = result;


      this.pokedb.team_id = this.pokeTeam.id;
      this.pokeService.getPokemonById(result.id_poke).subscribe((poke) => {
        this.pokeTeam.pokemons.push(poke);
        // console.log('poke ', poke);

        this.pokeTeam.pokemons.find((x) => x.id == poke.id)!.lvl =
          this.pokedb.lvl;
        this.pokeTeam.pokemons.find((x) => x.id == poke.id)!.exp =
          this.pokedb.exp;
        this.pokeTeam.pokemons.find((x) => x.id == poke.id)!.expMax =
          this.pokedb.expMax;
      });
      this.teamService.addPokeTeam(this.pokedb);
      this.nbpoke++;
    }
  });
}
```

Cette fonction prend en charge l'ouverture et la fermeture de la pop up. Ainsi que les informations renvoyées pour la pop-up.

Elle appelle ensuite une autre fonction du service de la team pour ajouter le pokémon choisi en bdd

La fenêtre pop-up :



3. Supression d'un pokemon dans la team

En appuyant sur le bouton « - » à coté du pokémon nous pouvons supprimer un pokémon de la team. Cependant l'utilisateur possédera toujours ce pokémon.

Voici donc la fonction déclenche lors de la suppression

```
onDelete(index: number, poke: Pokemon) {
  if (this.pokeTeam.pokemons.length > 1) {
    // console.log(this.pokeTeam.pokemons[index]);
    let pokedel = {
      id_poke: this.pokeTeam.pokemons[index].id,
      team_id: this.pokeTeam.id,
    };
    this.pokeTeam.pokemons = this.pokeTeam.pokemons.filter(
      (x) => x.id != poke.id
    );
    this.teamService.delPokeTeam(pokedel);
    index = index <= 0 ? index : index - 1;
    this.pokemon = this.pokeTeam.pokemons[index];
    this.nbpoke--;
  }
  this.isdel = true;
}
```

XI. Politique de test

1. Test unitaire

Un test unitaire est une procédure automatisée visant à vérifier individuellement la fonctionnalité correcte et isolée d'une petite partie d'un programme ou d'un module logiciel, appelée unité.

Pour mes test unitaire j’ai utilisé Jest et pour simuler les requêtes j’ai utilisé SuperTest.

```
describe("post pokemon", () => {
  test("It should respond with a 200 status code", async () => {
    const response = await request(app).post("/pokedb/add").send({
      "id_poke": "6",
      "name": "charizard",
      "exp": "50",
      "expMax": "50",
      "lvl": "1",
      "user_id":"2"
    });
    expect(response.statusCode).toBe(201);
  });
});

describe("update pokemon", () => {
  test("It should respond with a 200 status code", async () => {
    const response = await request(app).put("/pokedb/update/6&2").send({
      "lvl" : "42"
    });
    expect(response.statusCode).toBe(200);
  });
});

describe("Get all pokemon", () => {
  test("It should respond with a 200 status code", async () => {
    const response = await request(app).get("/pokedb/getAll");
    expect(response.statusCode).toBe(200);
  });
});

describe("get pokemon by id", () => {
  test("It should respond with a 200 status code", async () => {
    const response = await request(app).get("/pokedb/get/6&2");
    expect(response.statusCode).toBe(200);
  });
});
```

2. Test fonctionnelle

Un test fonctionnel est une méthode de vérification d'un système, d'une application ou d'un logiciel pour s'assurer qu'il fonctionne conformément aux spécifications et aux attentes définies. Ces tests évaluent les fonctionnalités, les interactions et les résultats d'une application du point de vue de l'utilisateur final. Les tests fonctionnels sont axés sur la validation des exigences fonctionnelles du logiciel et sont généralement effectués en simulant des scénarios d'utilisation réelle pour garantir que le logiciel répond aux besoins prévus.

J’ai donc utilisé Cypress pour tester le bon fonctionnement de l’application

```
describe('Tests Cypress pour la page Pokemon', () => {
  beforeEach(() => {
    cy.mount(TeamViewComponent)
  });

  it('Vérifie la présence d\'informations de Pokémon', () => {
    cy.get('.box').first().should('be.visible');
    cy.get('.box').first().find('p').should('have.length', 3); // Vérifie les 3 paragraphes de données
    cy.get('.box').first().find('img').should('be.visible');
  });

  it('Vérifie la présence d\'équipe de Pokémon', () => {
    cy.get('.box').last().should('be.visible');
    cy.get('.poke').should('have.length', 6); // Vérifie le nombre de blocs de Pokémon
  });

  it('Ajoute et supprime un Pokémon de l\'équipe', () => {
    cy.get('.poke').eq(0).click(); // Clique sur le premier Pokémon dans l'équipe
    cy.get('.btnadd').should('be.visible');
    cy.get('.btnadd').click(); // Ajoute un Pokémon dans l'équipe
    cy.get('.poke').should('have.length', 6); // Vérifie que l'équipe a maintenant 6 membres
    cy.get('.btnadd').last().click(); // Supprime le dernier Pokémon de l'équipe
    cy.get('.poke').should('have.length', 5); // Vérifie que l'équipe a maintenant 5 membres
  });

  // Ajoutez plus de tests selon vos besoins...
});
```

XII. Veille technologique

La veille technologique en informatique se réfère au processus continu de surveillance, de collecte et d'analyse des informations relatives aux nouvelles technologies, aux tendances émergentes, aux innovations et aux évolutions dans le domaine de l'informatique et des technologies de l'information. L'objectif principal de la veille technologique est de rester à jour avec les avancées technologiques, les nouvelles méthodes de développement, les changements dans les langages de programmation, les frameworks, les outils, les meilleures pratiques de sécurité, et tout autre élément pertinent pour les professionnels de l'informatique.

OWASP (Open Web Application Security Project) est une organisation à but non lucratif axée sur la sécurité des applications web. Elle fournit des ressources, des outils et des guides pour aider les développeurs à créer des applications web plus sûres. En utilisant le site OWASP pour ma veille technologique, j’ai pu accéder à des informations sur les vulnérabilités de sécurité courantes telles que les injections SQL, les failles XSS, la gestion incorrecte des sessions, etc. Le site propose également des listes de contrôle, des recommandations et des conseils pour sécuriser vos applications. En suivant les mises à jour d'OWASP, j’ai pu me tenir informé des dernières tendances en matière de sécurité des applications web et adopter les meilleures pratiques pour protéger mon développement.

XIII. Difficultés rencontrées

Mon parcours dans ce projet n'a pas été sans difficultés. Dès les premières étapes, j'ai dû faire face à des difficultés. Tout d'abord, la conception de la base de données s'est avérée être un problème. J'ai dû la retravailler à plusieurs reprises avant d'atteindre le résultat qui répondait à mes attentes.

Ensuite, l'étape de mise en place de l'API a présenté son propre lot de défis. Plus spécifiquement, l'intégration de Sequelize avec TypeScript s'est avérée être une partie peu documentée. J'ai dû consacrer un pas mal de temps à la recherche et à l'expérimentation pour parvenir à mettre en place cette partie de manière fonctionnelle et conforme à mes besoins.

Cependant, la plus grande problématique a été le développement de la fonctionnalité de combat Pokémon, un aspect crucial de mon application. Cette partie a exigé une attention particulière, car elle devait fonctionner correctement pour faire marcher le reste de mon application. Les multiples itérations, les tests et les ajustements m’ont pris une grande partie de mon temps de développement.

En somme, ce projet m'a offert un apprentissage approfondi à travers la résolution de ces difficultés. Chacune de ces étapes a contribué à mon développement en me confrontant à des problèmes concrets et en me poussant à trouver de nouvelles solutions.

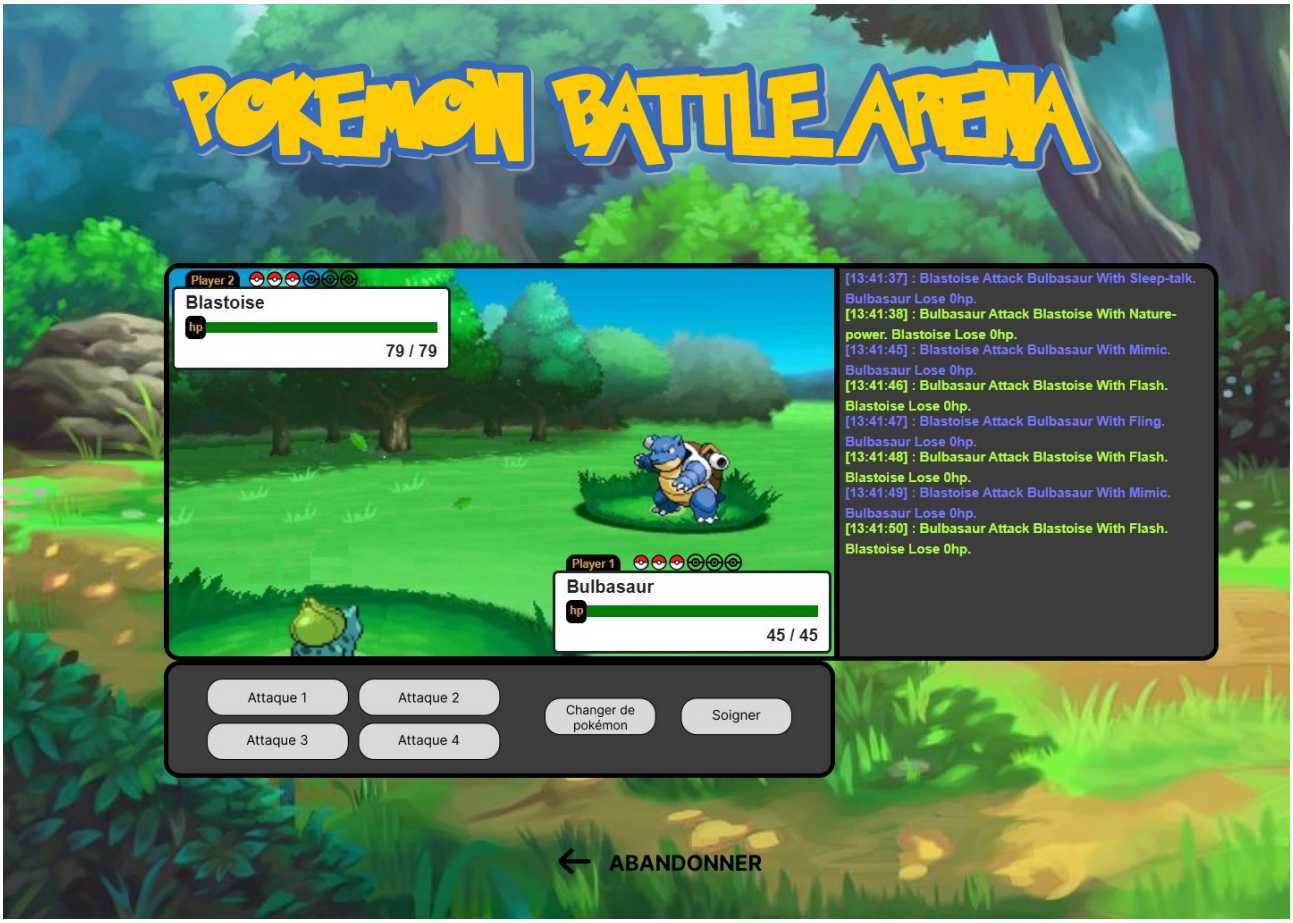
XIV. Conclusion

Ce projet a été une expérience enrichissante qui m'a permis d'acquérir de nouvelles connaissances approfondies concernant les différentes parties de conception. J'ai également développé une meilleure compréhension des compétences nécessaires pour une gestion efficace de projet.

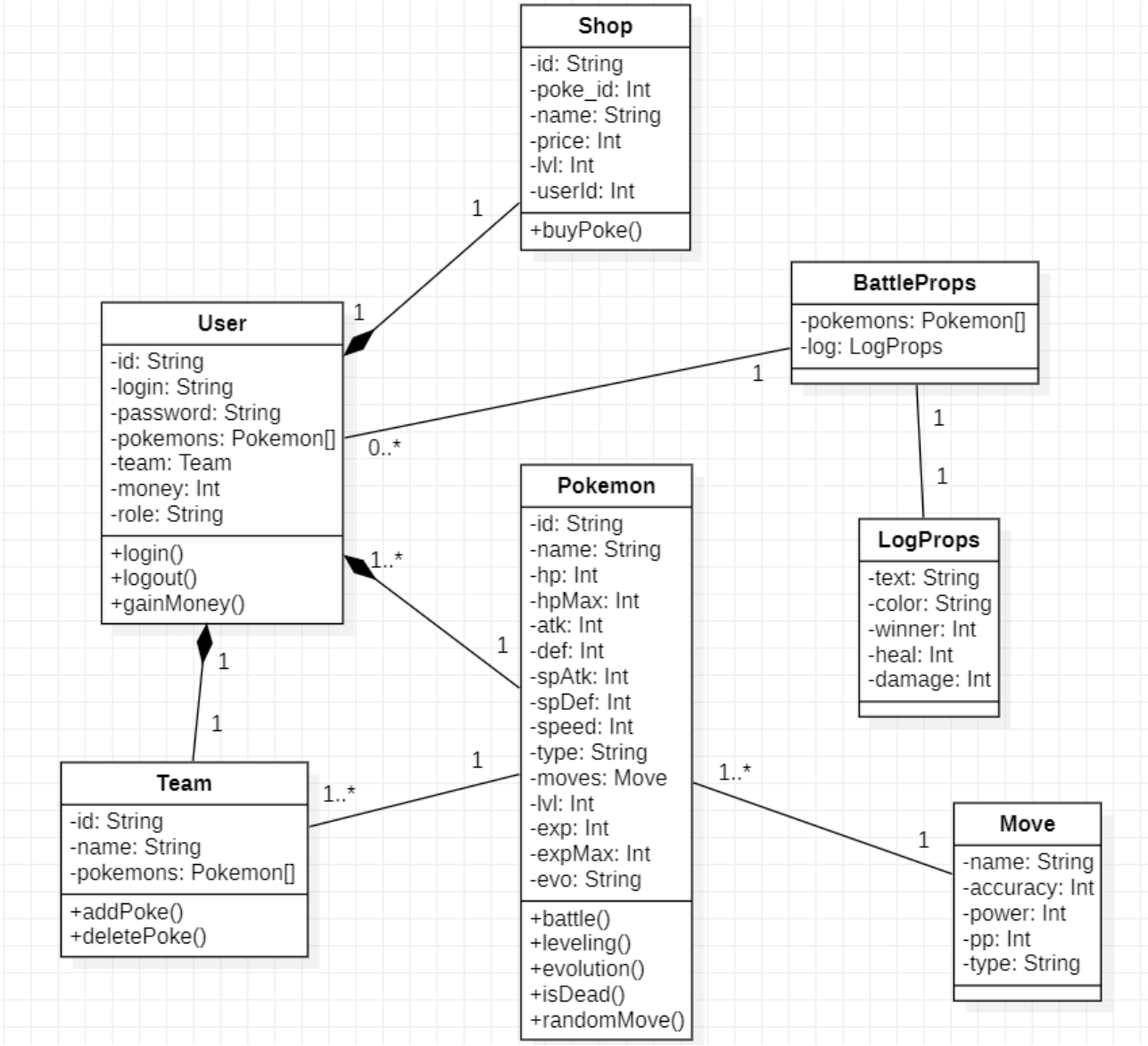
Cette immersion m'a non seulement permis de renforcer mes compétences techniques, mais aussi de perfectionner mes capacités à m’organiser. En résumé, ce projet a été une opportunité précieuse pour élargir mes horizons professionnels en intégrant des aspects clés de la conception et de la gestion de projet.

XV. Annexes

1. Annexe 1



2. Annexe 2


Shop
-id: String
-poke_id: Int
-name: String
-price: Int
-lvl: Int
-userId: Int
+buyPoke()
BattleProps
-pokemons: Pokemon[]
-log: LogProps
User
-id: String
-login: String
-password: String
-pokemons: Pokemon[]
-team: Team
-money: Int
-role: String
+login()
+logout()
+gainMoney()
Pokemon
-id: String
-name: String
-hp: Int
-hpMax: Int
-atk: Int
-def: Int
-spAtk: Int
-spDef: Int
-speed: Int
-type: String
-moves: Move
-lvl: Int
-exp: Int
-expMax: Int
-evo: String
+battle()
+leveling()
+evolution()
+isDead()
+randomMove()
LogProps
-text: String
-color: String
-winner: Int
-heal: Int
-damage: Int
Team
-id: String
-name: String
-pokemons: Pokemon[]
+addPoke()
+deletePoke()
Move
-name: String
-accuracy: Int
-power: Int
-pp: Int
-type: String
1
1
1
0..*
1
1
1..*
1
1
1
1..*
1
1..*
1