Projet fil rouge CDA – « Pokemort »

Table des matières

Table des matières 1

Problématique 2

A qui s'adresse le projet ? 3

Fonctionnalités principales 3

Détail des Fonctionnalités 3

- Initialiser l'aventure 3
- Consulter la liste des pokémons du dresseur 3
- Consulter la liste des pokémons déjà capturés (Pokédex) 4
- Phase de combat 4
  - Démarrer et charger les données d'un combat 4
  - Décider d'une attaque sur le dresseur adverse 4
  - Estimer la priorité des actions dans un combat 5
  - Evaluer les dommages d'une attaque d'un dresseur 6
  - Changer de pokémon au cours d'un combat 7
  - Tenter de capturer un pokémon (uniquement un pokémon sauvage) 8
  - Prendre la fuite face à un pokémon sauvage 9
  - Suggérer des stratégies par infobulle pour celui qui débute réellement le jeu 9
  - Calculer l'expérience et l'argent gagné lorsque le dresseur a battu un pokémon adverse ou un dresseur 9

Design et ergonomie 10

- Introduction 10
- Maquettes 11

Conception merise 12

- Base de données 12
  - Entités 12
  - Dictionnaire des données 14
  - Relations 16
  - MCD Graphique 19
  - MLD 20
  - MPD 21

UML 29
Classes 29
Cas d'utilisation 31
Contraintes Techniques 32
Backend (J2EE) : 33
Frontend (React) 33
Base de données (MySQL) 33
Performance et Scalabilité 33

Problématique

De nombreux fans ou anciens fans du jeu Pokémon ont investi des dizaines voire des centaines d'heures sur le jeu Pokémon. Beaucoup ont passé des moments à jouer en mode solo. D'autres, sur des versions plus récentes, ont commencé à affronter leurs adversaires en mode multijoueur. Cet engouement a été international et a même séduit la France, dans des communautés de type « Pokémon generation ».

Aujourd'hui, beaucoup de fans ne jouent plus à ce jeu. Ces fans ont pris d'autres directions dans la vie que de jouer à des jeux vidéo. C'est pourquoi il serait intéressant de leur proposer une expérience dans laquelle ils seraient à l'aise, sans pour autant avoir besoin de s'engager sur une trop longue durée.

Un « nuzlocke challenge » est un mode de jeu pokémon où les restrictions suivantes s'appliquent au jeu :

- Un pokémon KO est « mort » et disparaît de la liste des pokémons du dresseur
- Il n'est possible de capturer qu'un unique Pokémon par séquence de jeu

De plus, le jeu ne comprend pas des séquences où on se promène sur des cartes, afin que le joueur ne se concentre que sur les combats.

Ainsi, tous les combats contre les monstres sauvages ou les dresseurs vont aller jusqu'à lui. Il va enchaîner des combats qu'il devra toujours gagner. S'il en perd un seul, la partie se termine.

Au contraire, s'il gagne de plus en plus, des objets bonus lui seront offerts et il pourra soigner ses pokémons blessés !

A qui s'adresse le projet ?

Tous les anciens fans du jeu pourront jouer à ce jeu, sans aucune difficulté. Quant aux autres, ils pourront apprendre les bases du jeu, grâce à un guide.

Fonctionnalités principales

- Initialiser l'aventure
- Consulter la liste des pokémons que le dresseur détient
- Consulter la liste des pokémons déjà rencontrés
- Démarrer et charger les données d'un combat
- Décider d'une attaque sur le dresseur adverse
- Evaluer les dommages d'une attaque d'un dresseur
- Estimer la priorité des actions dans un combat
- Calculer les effets de l'action d'un joueur ou d'un opposant
- Changer de pokémon en cours de combat, que ce soit pour le dresseur ou l'attaquant
- Tenter de capturer un pokémon (uniquement un pokémon sauvage)
- Prendre la fuite face à un pokémon sauvage
- Suggérer des stratégies par infobulle pour celui qui débute réellement le jeu
- Calculer l'expérience gagné lorsque le dresseur a battu un pokémon adverse
- Décider de la séquence d'apparition des dresseurs ou des pokémons

Détail des Fonctionnalités

Initialiser l'aventure

Un écran d'introduction présente brièvement le défi Nuzlocke et ses règles principales (perte définitive des Pokémon KO et capture possible d'un unique pokémon par zone). Le joueur nomme ensuite son personnage. Un Pokémon de départ lui est attribué (Pikachu). Après une explication des règles du défi, l'aventure débute immédiatement avec un premier combat contre un Pokémon sauvage ou un dresseur. Le jeu sauvegarde automatiquement l'aventure en cas de problème de réseau. Cependant, plus d'une heure d'inactivité réinitialise la partie.

Consulter la liste des pokémons du dresseur

Consulter la liste des Pokémon permet au joueur d'accéder à une vue d'ensemble de ses Pokémons capturés et en équipe. Cette fonctionnalité affiche les Pokémon sous forme

de miniatures, avec leurs noms, niveaux, et statistiques de base (PV, attaque, défense, etc.). Le joueur peut consulter les détails de chaque Pokémon, comme ses attaques, son état (santé actuelle), et les objets qu'il porte. En fonction du système de jeu, le joueur peut également effectuer des actions comme soigner ses Pokémon, changer leur position dans l'équipe. Cette liste permet de suivre l'évolution et la gestion de son équipe au fur et à mesure de l'aventure.

Consulter la liste des pokémons déjà capturés (Pokédex)

Consulter la liste des Pokémon déjà rencontrés permet au joueur de visualiser tous les Pokémon qu'il a croisés au cours de son aventure, qu'ils aient été capturés ou non. Cette liste présente les Pokémon sous forme de vignettes, avec leurs noms, leurs types et un indicateur de leurs statuts (capturés ou non, vus ou non).

Phase de combat

Démarrer et charger les données d'un combat

Démarrer et charger les données d'un combat est l'action qui prépare le jeu avant le début d'un affrontement entre Pokémon. L'aventure Nuzlocke challenge n'est composé que de combats qui vont s'enchaîner un par un, jusqu'à la défaite du dresseur ou la fin de l'aventure, quand il aura battu tous les champions de la ligue des 4. Lorsqu'un combat commence, le jeu récupère toutes les informations nécessaires pour les deux participants : les Pokémon du joueur et ceux de l'adversaire. Cela inclut leurs niveaux, statistiques (telles que les points de vie, l'attaque, la défense, la vitesse), ainsi que leurs attaques disponibles et les objets équipés.

Ensuite, l'écran de combat s'affiche, montrant les portraits des Pokémon, leurs barres de vie, et d'autres informations essentielles comme leurs noms et niveaux. Si le joueur est confronté à un Pokémon sauvage, ses statistiques sont affichées, tandis que pour un dresseur, l'équipe de ce dernier est chargée.

Le jeu initialise également les mécaniques de combat, comme la gestion du tour de jeu (qui commence en premier, si des effets de statut sont actifs, etc.). Le joueur peut alors choisir une action (attaquer, utiliser un objet, fuir, etc.), tandis que l'adversaire (l'ordinateur) réagit en fonction des règles définies et des données déjà connues.

Tout est ainsi prêt pour que le joueur puisse prendre ses décisions stratégiques et commencer à combattre.

Décider d'une attaque sur le dresseur adverse

Décider d'une attaque sur le dresseur adverse implique que le joueur choisisse une action pour son Pokémon pendant son tour de combat. Lorsque le joueur sélectionne son Pokémon, une liste d'attaques disponibles s'affiche, basée sur les capacités de ce

dernier. Chaque attaque est associée à un type (par exemple, feu, eau, plante), et peut avoir des effets supplémentaires comme des statuts altérés (brûlure, gel, etc.) ou des modificateurs de dégâts en fonction des forces et faiblesses des types.

Le joueur doit aussi prendre en compte la puissance de chaque attaque et son PP (Points de Pouvoir) restant, qui dépendent du nombre d'utilisations restantes pour chaque attaque. Une fois l'attaque choisie, le joueur confirme sa sélection et un calcul des priorités est effectué. Le joueur qui a le Pokémon le plus rapide (ou l'attaque la plus rapide) effectuera l'attaque en premier. En fonction de la précision de l'attaque, cette attaque pourrait être annulée car échouée. Cependant, si l'attaque se lance, le système calculera les dégâts en fonction de l'attaque choisie, des statistiques de chaque Pokémon, et de la défense de l'adversaire.

Une fois que l'attaque est effectuée, l'écran de combat se met à jour pour refléter les changements de barres de vie et les effets de l'attaque. Le procédé sera identique pour le deuxième pokémon (le moins prioritaire) qui attaquera à son tour.

Estimer la priorité des actions dans un combat

Dans un jeu Pokémon, estimer la priorité des actions dans un combat implique de déterminer l'ordre dans lequel les attaques et actions des Pokémon sont exécutées. Ce processus est essentiel pour la stratégie de combat, car il peut influencer le résultat d'un affrontement. La priorité des actions est déterminée par plusieurs facteurs clés :

1. **Statistique de Vitesse** : Dans la plupart des jeux Pokémon, l'action du Pokémon avec la **vitesse** la plus élevée sera généralement exécutée en premier. Si les vitesses des deux Pokémon sont égales, un autre facteur, comme la **chance**, peut être utilisé pour déterminer l'ordre.

2. **Priorité des attaques** : Certaines attaques ont un niveau de **priorité** qui les fait agir avant d'autres, indépendamment de la statistique de vitesse. Par exemple, des attaques comme **Vitesse Extrême** ou **Aiguisage** ont une priorité plus élevée, permettant au Pokémon d'attaquer avant même si sa vitesse est plus faible que celle de l'adversaire.

3. **Effet de statut** : Les statuts comme la paralysie ou le sommeil peuvent affecter la vitesse ou la capacité d'un Pokémon à agir. Un Pokémon paralysé, par exemple, a une probabilité réduite d'agir en premier.

4. **Ordre de sélection des actions** : Les actions sont choisies simultanément par les deux combattants, mais l'ordre de leur exécution est souvent déterminé après, en fonction de la vitesse des Pokémon et de la priorité des attaques. Les Pokémon ayant un statut de **priorité** agissent toujours avant les autres, même si leur vitesse est inférieure.

5. **Critères de priorité dans les capacités spécifiques** : Certaines attaques ou capacités peuvent modifier directement l'ordre d'exécution. Par exemple, des attaques comme **Feu Follet** ou **Soin** peuvent être affectées par la priorité de l'attaque, tandis que des effets comme **Explosion** ou **Destruction** peuvent se déclencher en dernier après que l'attaque a infligé des dégâts.

En résumé, la priorité des actions est déterminée par une combinaison de la vitesse des Pokémon, des priorités spécifiques des attaques, des effets de statut et de la manière dont les capacités interagissent entre elles. Ce système permet de créer des combats dynamiques et stratégiques.

Evaluer les dommages d'une attaque d'un dresseur

Évaluer les dommages d'une attaque d'un dresseur consiste à calculer l'impact d'une attaque sur le Pokémon adverse, en prenant en compte plusieurs facteurs de gameplay. Lorsque le joueur choisit une attaque, le système effectue un calcul basé sur les statistiques de chaque Pokémon (attaque du lanceur, défense du défenseur, niveau, etc.). Voici comment les dommages sont évalués :

1. Calcul des dégâts de base : Le système commence par déterminer une valeur de dégâts de base à partir de la puissance de l'attaque et des statistiques d'attaque du Pokémon lanceur. Les attaques de type spécial (comme des attaques de type feu ou eau) utilisent l'attaque spéciale et les attaques physiques utilisent la statistique d'attaque.

2. Facteurs de type : Le jeu évalue les types des Pokémon impliqués (par exemple, le feu est fort contre la plante mais faible contre l'eau) et applique un modificateur de type qui peut augmenter ou réduire les dégâts en fonction de la

   relation entre les types des attaquants et des défenseurs •

3. Niveau et statistiques : Le niveau des Pokémon influence le calcul des dégâts. Un Pokémon de niveau plus élevé infligera généralement plus de dégâts, à condition que ses statistiques d'attaque et de défense soient également favorables.

4. Randomisation : Les dégâts ne sont pas constants et sont affectés par une valeur aléatoire pour rendre chaque combat un peu plus dynamique et imprévisible. Cela crée une plage de valeurs possibles pour les dégâts.

5. Effets supplémentaires : Certains mouvements peuvent avoir des effets secondaires, comme réduire les statistiques du défenseur (par exemple, baisser

sa défense), ou infliger des statuts comme la paralysie, le gel ou la brûlure, qui influencent les dégâts.

6. Affichage des résultats : Après le calcul, les dégâts finaux sont appliqués à la barre de vie du Pokémon adverse, et l'animation de l'attaque est affichée à l'écran pour que le joueur voit l'impact de son attaque.

En résumé, évaluer les dommages d'une attaque implique de prendre en compte les statistiques des Pokémon, leurs types, des facteurs aléatoires et les effets des attaques pour calculer l'impact précis sur l'adversaire.

Changer de pokémon au cours d'un combat

Changer de Pokémon en cours de combat permet au joueur de remplacer un Pokémon actif par un autre de son équipe pendant un combat. Cette fonctionnalité offre une dimension stratégique importante, permettant de s'adapter aux attaques de l'adversaire ou de gérer la santé de ses Pokémon. Voici les étapes et éléments clés de cette fonctionnalité :

1. Accès à l'option de changement : Pendant le combat, le joueur peut choisir de changer de Pokémon en sélectionnant l'option appropriée dans le menu de combat, généralement pendant son tour. Cette option est disponible tant pour le dresseur que pour l'attaquant adverse (si le combat est contre un dresseur).

2. Sélection d'un Pokémon de remplacement : Une liste des Pokémon restants dans l'équipe du joueur est affichée, avec leurs noms, niveaux et barres de vie. Le joueur peut sélectionner un Pokémon qu'il souhaite faire entrer dans le combat.

3. Conséquences du changement : Lorsqu'un Pokémon est changé, l'action de changement peut entraîner une perte de tour. En fonction des règles du jeu, il peut être nécessaire de subir une attaque de l'adversaire après que le Pokémon remplaçant entre dans le combat. Certaines attaque peuvent , au contrainte, s'en prendre au Pokémon adverse, avant qu'il soit échangé.

4. Stratégie de type : Le changement de Pokémon permet de profiter des forces de type. Par exemple, si le Pokémon adverse utilise un pokémon de type Eau, le joueur pourrait choisir une attaque de type électrique pour maximiser les dégâts.

5. Changement durant un combat contre un dresseur : Dans un combat contre un dresseur, ce dernier peut également changer de Pokémon pendant son tour. Cette fonctionnalité crée un jeu de stratégie et de réaction, car le joueur doit anticiper les changements de son adversaire.

6. Limites et restrictions : Selon les règles du défi (comme un Nuzlocke Challenge), certains changements peuvent être restreints par des conditions particulières (par exemple, un Pokémon KO ne peut pas être remplacé). Par ailleurs, certains

combats peuvent interdire ou restreindre l'option de changement, par exemple, lors de combats de boss ou dans des situations spécifiques.

En résumé, changer de Pokémon en cours de combat est une fonctionnalité qui permet au joueur d'ajuster son équipe en fonction de la situation de combat, en utilisant les forces et les faiblesses des types et en prenant en compte la santé de ses Pokémon. Cette action ajoute de la profondeur stratégique aux combats.

Tenter de capturer un pokémon (uniquement un pokémon sauvage)

Tenter de capturer un Pokémon (uniquement un Pokémon sauvage) est une action qui permet au joueur de tenter d'ajouter un Pokémon rencontré dans la nature à son équipe. Cette action se déroule en plusieurs étapes essentielles :

1. **Rencontre avec un Pokémon sauvage** : Lorsqu'un Pokémon sauvage apparaît en combat, le joueur a l'option de tenter de le capturer au lieu de simplement le combattre.

2. **Affaiblir le Pokémon sauvage** : Avant de tenter la capture, il est généralement nécessaire de réduire les **points de vie (PV)** du Pokémon sauvage sans le mettre KO. Le joueur peut utiliser des attaques ou changements de types qui affaiblissent lentement le Pokémonou bien utiliser des attaques modifiant son **statut**, comme la paralysie ou le sommeil, ce qui augmente les chances de succès.

3. **Choisir une Pokeball** : Une fois le Pokémon suffisamment affaibli, le joueur peut sélectionner une **Pokéball** à lancer. Différentes Pokéballs (standard, Super Pokéball, Hyperball, etc.) ont des **chances de capture** variées.

4. **Lancer la Pokéball** : Le joueur lance la Pokéball et attend de voir si le Pokémon sera capturé. L'issue est déterminée par un calcul basé sur plusieurs facteurs, tels que la **santé** du Pokémon, les **effets de statut** appliqués, et le type de Pokéball utilisée.

5. **Réussite ou échec de la capture** : Si le Pokémon est capturé, il est ajouté à l'équipe du joueur ou envoyé dans le **PC** si l'équipe est déjà complète. Si la capture échoue, le joueur peut essayer de lancer une nouvelle Pokéball, mais la tentative sera limitée par le nombre de Pokéballs en possession.

6. **Règles du Nuzlocke Challenge** : Dans le cadre d'un **Nuzlocke Challenge**, le joueur ne peut capturer qu'un seul Pokémon par zone. Si un Pokémon s'échappe ou que la capture échoue, cette opportunité est perdue pour la zone.

Ainsi, cette action repose sur une bonne gestion des ressources (Pokéballs, attaques, et effets de statut) et sur une stratégie pour maximiser les chances de capturer un Pokémon tout en minimisant les risques.

Prendre la fuite face à un pokémon sauvage

Prendre la fuite face à un Pokémon sauvage permet au joueur de quitter un combat sans perdre de Pokémon. Lorsqu'un joueur est confronté à un Pokémon sauvage qu'il préfère éviter, il peut choisir l'option de **fuir** dans le menu de combat. La fuite n'est pas garantie et dépend souvent de la **vitesse** du Pokémon du joueur par rapport à celui de l'adversaire. Si la fuite échoue, le combat continue. Le Pokémon adverse pourra jouer son tour, sans pour autant que le dresseur puisse riposter pendant ce tour. La fuite est une option pratique lorsqu'un combat est trop risqué ou lorsque le joueur manque de ressources.

Suggérer des stratégies par infobulle pour celui qui débute réellement le jeu

Calculer l'expérience et l'argent gagné lorsque le dresseur a battu un pokémon adverse ou un dresseur

La victoire d'un dresseur face à un Pokémon sauvage ou un dresseur se produit lorsqu'un joueur réussit à battre tous les Pokémon de son adversaire, que ce soit un Pokémon sauvage ou un dresseur. Voici les étapes clés qui suivent la victoire :

1. **Affichage du résultat** : Après avoir mis K.O. tous les Pokémon de l'adversaire, un message de victoire apparaît à l'écran, indiquant que le joueur a triomphé du combat.
2. Récompenses : En cas de victoire contre un Pokémon sauvage, le joueur reçoit de l'expérience pour ses Pokémon et parfois des objets comme des Pokéballs ou des Baies. Contre un dresseur, des récompenses peuvent inclure des pokédollars et des objets spéciaux.
3. **Gains d'expérience** : Les Pokémon ayant participé au combat reçoivent des points d'expérience, ce qui peut leur permettre d'augmenter de niveau et d'améliorer leurs statistiques. Dans certains cas, un Pokémon peut même évoluer après avoir gagné suffisamment d'expérience.
4. **Fin de combat** : Après la victoire, le combat se termine et le joueur est directement envoyé sur un autre combat. Si un Pokémon a été blessé pendant le combat, il pourra nécessiter des soins, comme des **potions**. Dans la version du jeu, il ne sera pas possible d'aller en centre Pokémon.
5. **Enregistrement de la victoire** : Le jeu sauvegarde automatiquement la progression du joueur, permettant de reprendre l'aventure à tout moment. Puisque la victoire a lieu dans un défi Nuzlocke, il est essentiel de gérer les conséquences de la victoire, comme l'ajout de nouveaux Pokémon à l'équipe ou l'optimisation de l'équipe actuelle.

En résumé, une victoire contre un Pokémon sauvage ou un dresseur permet au joueur de progresser, de gagner des récompenses et de faire évoluer son équipe pour les futurs défis.

Design et ergonomie

Introduction

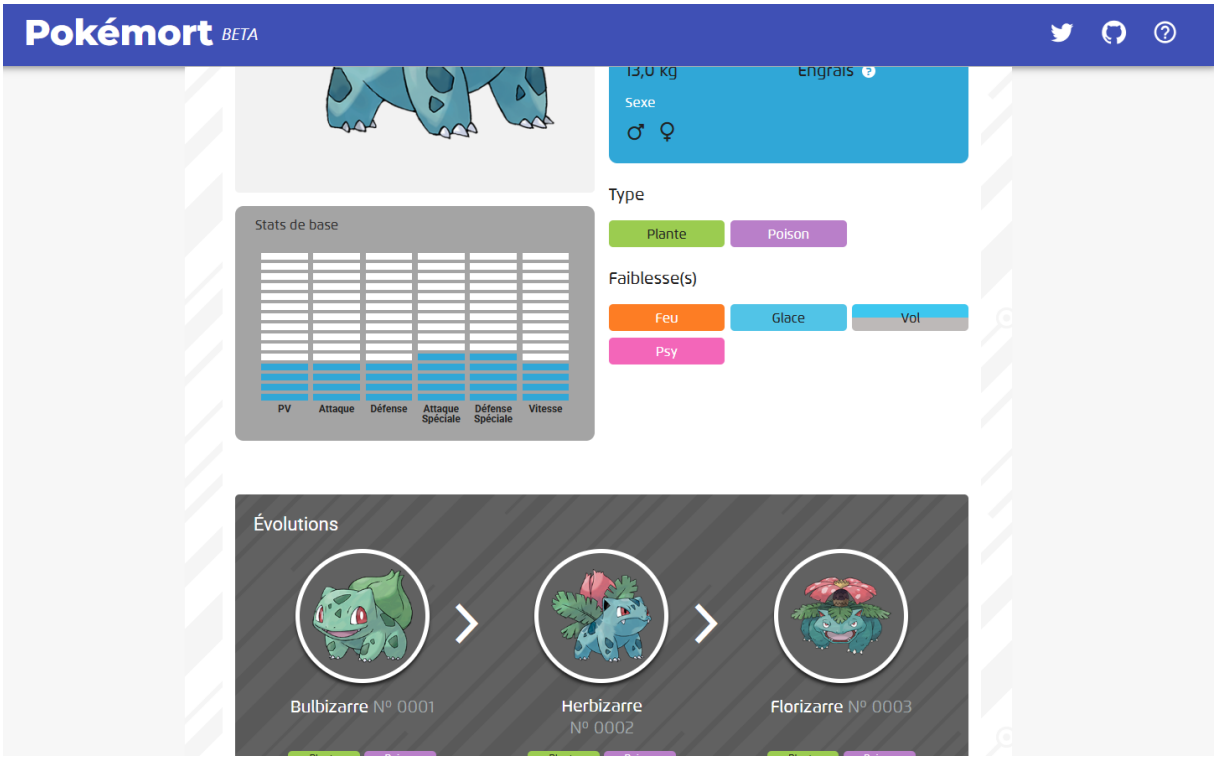
Le jeu « Pokémort » aura un design qui est adapté à l'accessibilité de ses joueurs, aux origines variées et avec des spécificités variées.

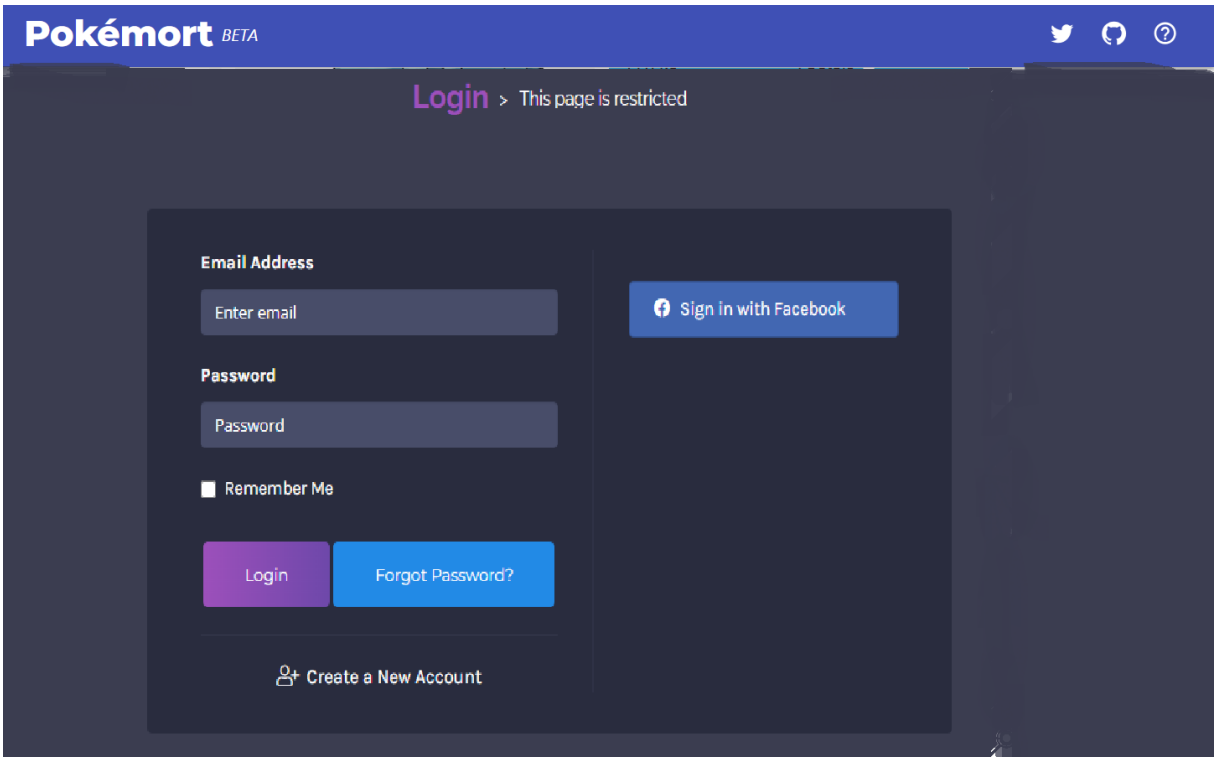
Le jeu sera sonorisé et aura un design qui respectera les fondamentaux des règles WCAG et RGAA.

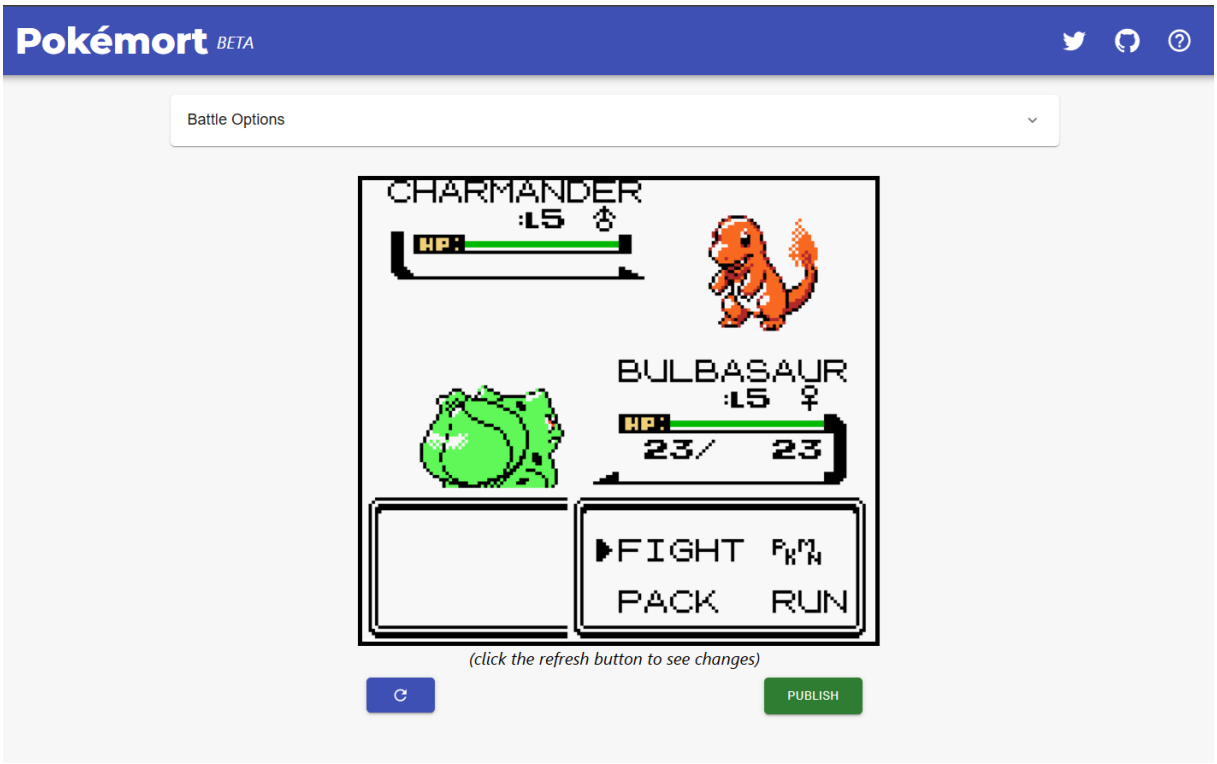
Toute action sera totalement décrite avec une audiodescription.

De plus, le jeu se jouera sur n'importe quel navigateur, tout comme sur mobile, grâce à une programmation d'une interface responsive.

Maquettes


Pokémort BETA
13,0 kg
Engrais
Sexe
Type
Plante
Poison
Faiblesse(s)
Feu
Glace
Vol
Psy
Stats de base
PV
Attaque
Défense
Attaque Spéciale
Défense Spéciale
Vitesse
Évolutions
Bulbizarre Nº 0001
Herbizarre Nº 0002
Florizarre Nº 0003


Pokémort BETA
Login > This page is restricted
Email Address
Enter email
Password
Password
Remember Me
Login
Forgot Password?
Create a New Account
Sign in with Facebook



Conception merise

Base de données

Entités

- Partie
- Joueur
- Type
- Badge
- Objet
- Pokemon
- Talent
- Attaque
- BaseEffet
- Effet
- Strategie

- Ennemi
- Climat

-

Dictionnaire des données

Code/Libellé	Désignation	Type	Taille
PARTIE			
par_id	Identifiant de la partie	D	11
par_actif	La partie est-elle active ?	D	1
par_score	Score total de la partie en cours ou terminé	D	11
par_statut	Statut de la partie (terminée ou en cours)	D	1
par_debut	Heure du début de la partie	D	1
par_fin	Heure de fin de la partie	D	17
MEMBRE			
mem_id	Identifiant du joueur	D	11
mem_pseudo	Pseudonyme du membre	AN	20
mem_mdp	Mot de passe du membre	AN	255
mem_nom	Nom choisi pour le membre	AN	255
mem_photo	Photo du membre	AN	500
PERSONNAGE			
per_id	Identifiant du personnage jouable	D	11
per_nom	Nom du personnage jouable ou non jouable	AN	500
per_photo	Photo du personnage jouable ou non jouable	AN	500
per_jouable	Le personnage représente un personnage non-jouable ?	D	1
per_copie	S'agit-t-il version temporaire d'un personnage non jouable, destiné à être attaquable en combat ?	D	1
TYPE			
typ_id	Identifiant du type de pokémon	D	
typ_nom	Nom du type pokémon	AN	
typ_logo	Lien image vers le logo du type	AN	
BADGE			
bad_id	Identifiant du badge	D	11
bad_image	Image du badge	AN	200
bad_ordre	Ordre d'obtention du badge	D	11
OBJET			
obj_id	Identifiant de l'objet	D	11

obj_nom	Nom de l'objet	AN	100
obj_image	Image de l'objet	AN	255
STATUT			
sta_id	Identifiant du statut	D	11
sta_nom	Nom du statut (pour Pokémon)	AN	50
POKEMON			
pok_id	Identifiant du pokémon	D	11
pok_nom	Nom du pokémon	AN	50
pok_image	Lien de l'mage principale du pokémon	AN	500
pok_miniature	Lien de l'image de miniature du pokémon	AN	500
Pok_base_pv	IV des PV du pokémon	D	11
pok_base_attaqie	IV de l'attaque du pokémon	D	11
pok_base_defense	IV de la défense du pokémon	D	11
pok_base_attaque_speciale	IV de l'attaque spéciale du pokémon	D	11
pok_base_defense_speciale	Iv de la défense spéciale du pokémon	D	11
pok_base_vitesse	IV de la vitesse du pokémon	D	11
TALENT			
tal_id	Identifiant du talent	D	2
tal_nom	Nom du talent	AN	10
tal_fois	Nombre de fois qu'un talent peut être lancé	D	11
ATTAQUE			
att_id	Identifiant de l'attaque	D	11
att_nom	Nom de l'attaque	AN	11
att_puissance	Points de dégâts infligés par l'attaque	D	3
att_precision	Précision de l'attaque, en pourcentage	D	2
BASEEFFET			
bas_id	Identifiant de l'effet de base	D	11
bas_colonne	Colonne du pokémon affecté par l'effet (table dresser)	AN	50
bas_precision	Probabilité de la survenue de l'effet	N	3
bas_degats	Dégâts infligés par l'effet de base, en pourcentage ou en unités	N	4
bas_affecte_ennemi	Cet effet s'applique-t-il à l'ennemi ?	D	1
bas_affecte_pokemon	Cet effet s'applique sur le pokémon du dresseur	D	1
EFFET			
eff_id	Identifiant de l'effet de l'attaque	D	11

eff_nom	Nom de l'effet pouvant être déclenché	D	2
eff_description	Description de l'effet	AN	100
STRATEGIE			
str_id	Identifiant de la stratégie	D	11
str_nom	Nom de la stratégie	AN	100
ENNEMI			
enn_id	Identifiant de l'ennemi	D	11
enn_ordre	Ordre d'apparition de l'ennemi, en valeur relative	D	11
enn_type	S'agit-t-il d'un pokémon ou d'un humain	D	3
enn_nom	Nom de l'ennemi	AN	50
enn_phrase_debut	Phrase de début de l'ennemi	AN	200
enn_phrase_victoire	Phrase de victoire de l'ennemi	AN	200
enn_phrase_defaite	Phrase de défaite de l'ennemi	AN	200
CLIMAT			
Cli_id	Identifiant d'un climat	D	11
Cli_nom	Nom du climat	AN	200
COMBAT			
com_id	Identifiant d'un climat	D	11
com_nom	Nom du climat	AN	200

Relations

Récompenser (rec_date)

- Badge : 1,n
- Partie : 1,n

Initier

- Partie : 1,1
- Membre : 0,1

Incarner

- Partie : 1,1
- Personnage : 1,n

Coordonner

- Strategie : 1,n
- Ennemi : 1,1

Delivrer

- Ennemi : 0,1

- Badge : 0,1

Offrir (off_quantite)

- Ennemi : 1,n
- Objet: 1,n

Copier

- Ennemi : 1,1
- Personnage : 1,n

Combattre (com_statut)

- Personnage : 1,n
- Personnage : 1,n
- Climat : 1,n
- Ennemi : 1,n

SortCombattre

- Combattre : 1,n
- Effet : 1,n

Décomposer

- Effet : 1 ;n
- BaseEffet : 1,n

Appliquer

- Effet : 1,n
- Objet : 1,n

Exprimer

- Talent : 1,n
- Effet : 1,n

Déclencher

- Attaque : 1,n
- Effet : 1,n

Engendrer

- Statut : 1 ;n
- Effet : 1,n

Craindre (effet, signe)

- Type : 1,n

- Type : 1,n

EtreType

- Pokemon : 1,n
- Type : 1,n

PeutApprendre (niveau)

- Pokemon : 1,n
- Attaque : 1 ;n

VaEvoluer (niveau)

- Pokemon : 0,1
- Pokemon : 0,1

Exprimer

- Pokémon : 1,n
- Talent : 1,n

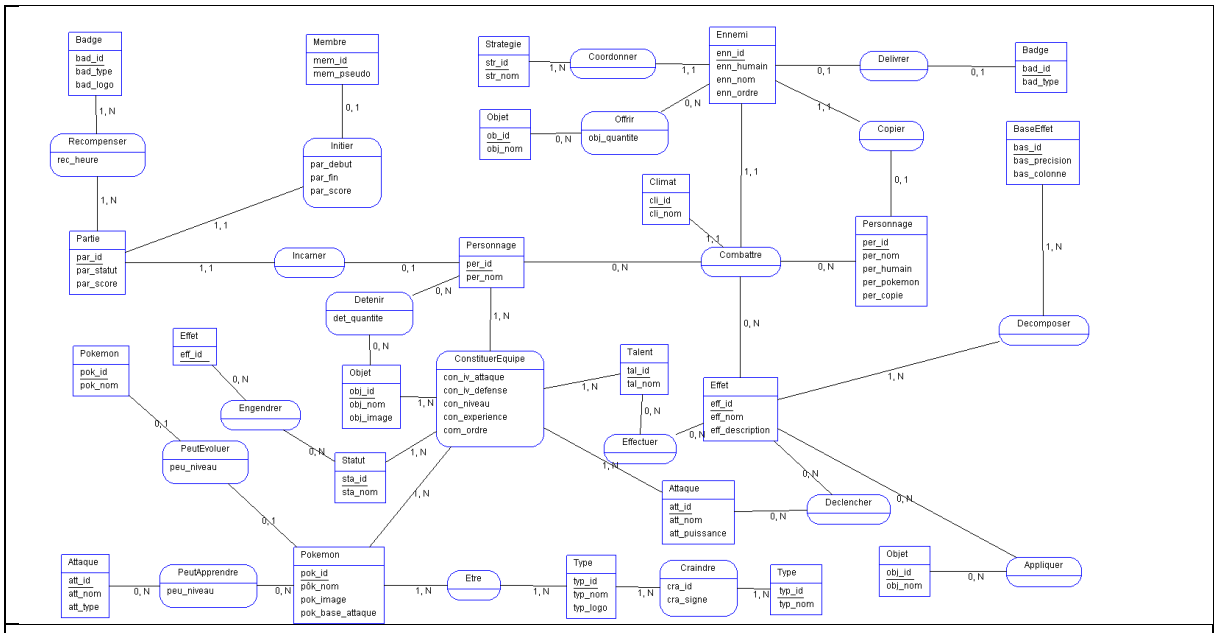
Détenir (quantite)

- Personnage : 1,n
- Objet : 1,n

ConstituerEquipe (con_iv_attaque, con_iv_defense, con_iv_attaque_specialecon_iv_defense_specialecon_iv_vitesse, con_niveau, con_pv_restants, con_ordre, con_talent, con_experience, con_shiny, con_statut)

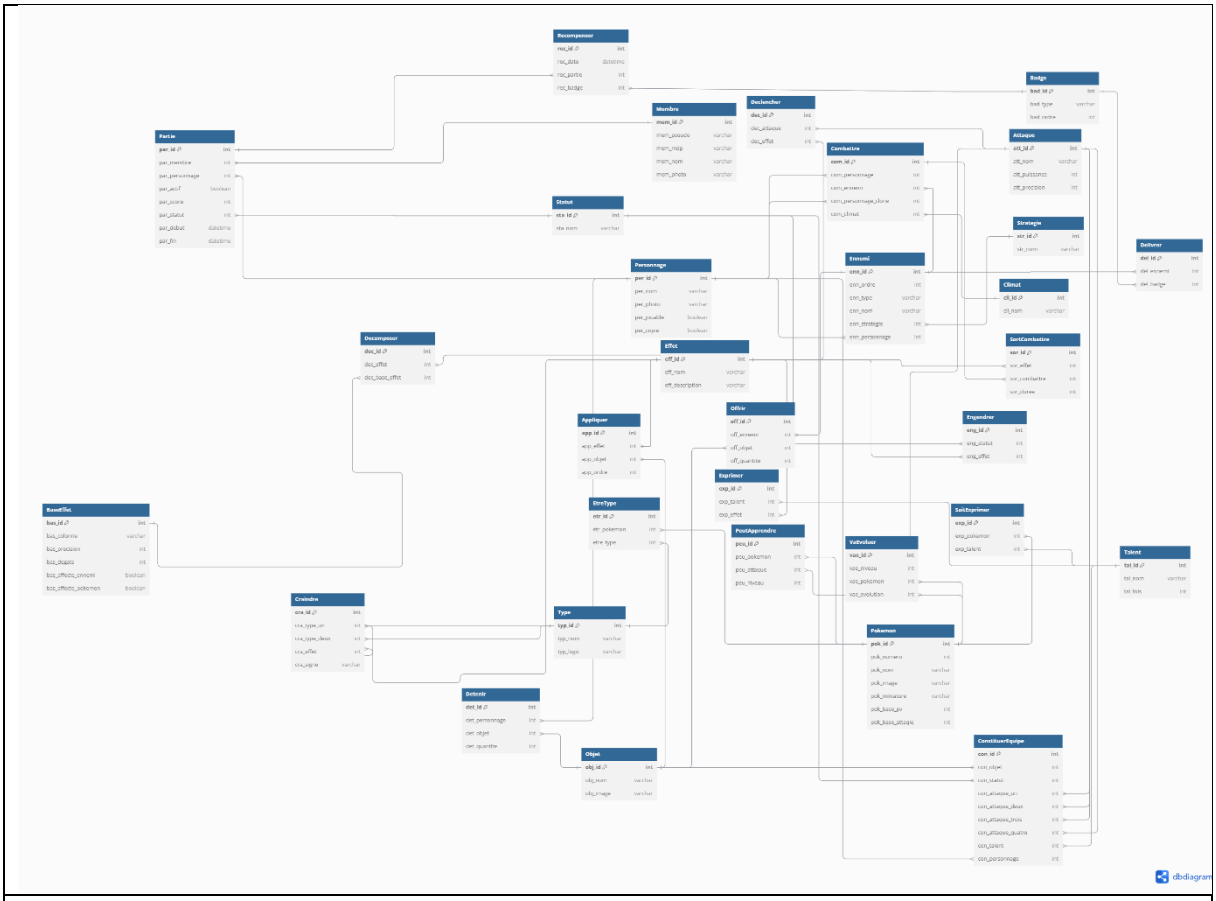
- Personnage : 1,n
- Objet : 1,n
- Pokemon : 1,n
- Talent : 1,n
- Statut : 1,n
- Attaque : 1,n
- Attaque : 1,n
- Attaque : 1,n
- Attaque : 1,n

MCD Graphique



La question de faire jouer plusieurs personnes à la fois sur l'application a été étudiée

MLD



MLD graphique de la base de données

MPD

```sql
DROP DATABASE IF EXISTS pokemort;

CREATE DATABASE pokemort;

USE pokemort;

CREATE TABLE membre (
    mem_id INT PRIMARY KEY,
    mem_pseudo VARCHAR(255) NOT NULL,
    mem_mdp VARCHAR(255) NOT NULL,
    mem_nom VARCHAR(255) NOT NULL,
    mem_photo VARCHAR(255)
);

CREATE TABLE statut (
    sta_id INT PRIMARY KEY,
    sta_nom VARCHAR(255) NOT NULL
);

CREATE TABLE personnage (
    per_id INT PRIMARY KEY,
    per_nom VARCHAR(255) NOT NULL,
    per_photo VARCHAR(255),
    per_jouable BOOLEAN NOT NULL,
    per_copie INT
);

CREATE TABLE type (
    typ_id INT PRIMARY KEY,
    typ_nom VARCHAR(255) NOT NULL,
    typ_logo VARCHAR(255)
);

CREATE TABLE badge (
    bad_id INT PRIMARY KEY,
    bad_type INT NOT NULL,
    bad_ordre INT NOT NULL
);

CREATE TABLE objet (
    obj_id INT PRIMARY KEY,
    obj_nom VARCHAR(255) NOT NULL,
    obj_image VARCHAR(255)
```

```sql
);

CREATE TABLE pokemon (
    pok_id INT PRIMARY KEY,
    pok_numero INT NOT NULL,
    pok_nom VARCHAR(255) NOT NULL,
    pok_image VARCHAR(255),
    pok_miniature VARCHAR(255),
    pok_base_pv INT NOT NULL,
    pok_base_attaqie INT NOT NULL,
    pok_base_defense INT NOT NULL,
    pok_base_attaque_speciale INT NOT NULL,
    pok_base_defense_speciale INT NOT NULL,
    pok_base_vitesse INT NOT NULL
);

CREATE TABLE talent (
    tal_id INT PRIMARY KEY,
    tal_nom VARCHAR(255) NOT NULL,
    tal_fois INT NOT NULL
);

CREATE TABLE attaque (
    att_id INT PRIMARY KEY,
    att_nom VARCHAR(255) NOT NULL,
    att_puissance INT NOT NULL,
    att_precision INT NOT NULL
);

CREATE TABLE baseeffet (
    bas_id INT PRIMARY KEY,
    bas_colonne VARCHAR(255),
    bas_precision INT,
    bas_degats INT,
    bas_affecte_ennemi BOOLEAN,
    bas_affecte_pokemon BOOLEAN
);

CREATE TABLE effet (
    eff_id INT PRIMARY KEY,
    eff_nom VARCHAR(255) NOT NULL,
    eff_description TEXT
);

CREATE TABLE strategie (
    str_id INT PRIMARY KEY,
    str_nom VARCHAR(255) NOT NULL
);
```

```sql
CREATE TABLE ennemi (
    enn_id INT PRIMARY KEY,
    enn_ordre INT NOT NULL,
    enn_type VARCHAR(255),
    enn_nom VARCHAR(255) NOT NULL,
    enn_phrase_debut TEXT,
    enn_phrase_victoire TEXT,
    enn_phrase_defaite TEXT,
    enn_strategie INT,
    enn_personnage INT
);

CREATE TABLE climat (
    cli_id INT PRIMARY KEY,
    cli_nom VARCHAR(255) NOT NULL
);

CREATE TABLE partie (
    par_id INT PRIMARY KEY,
    par_membre INT NOT NULL,
    par_personnage INT NOT NULL,
    par_actif BOOLEAN NOT NULL,
    par_score INT NOT NULL,
    par_statut INT NOT NULL,
    par_debut DATETIME NOT NULL,
    par_fin DATETIME NOT NULL
);

CREATE TABLE recompenser (
    rec_id INT PRIMARY KEY,
    rec_date DATETIME NOT NULL,
    rec_partie INT NOT NULL,
    rec_badge INT NOT NULL
);

CREATE TABLE delivrer (
    del_id INT PRIMARY KEY,
    del_ennemi INT NOT NULL,
    del_badge INT NOT NULL
);

CREATE TABLE offrir (
    off_id INT PRIMARY KEY,
    off_ennemi INT NOT NULL,
    off_objet INT NOT NULL,
    off_quantite INT NOT NULL
);

CREATE TABLE combattre (
```

```
    com_id INT PRIMARY KEY,
    com_personnage INT NOT NULL,
    com_ennemi INT NOT NULL,
    com_climat INT NOT NULL
);

CREATE TABLE sortcombattre (
    sor_id INT PRIMARY KEY,
    sor_effet INT NOT NULL,
    sor_combattre INT NOT NULL,
    sor_duree INT NOT NULL
);

CREATE TABLE decomposer (
    dec_id INT PRIMARY KEY,
    dec_effet INT NOT NULL,
    dec_base_effet INT NOT NULL
);

CREATE TABLE appliquer (
    app_id INT PRIMARY KEY,
    app_effet INT NOT NULL,
    app_objet INT NOT NULL,
    app_ordre INT NOT NULL
);

CREATE TABLE exprimer (
    exp_id INT PRIMARY KEY,
    exp_talent INT NOT NULL,
    exp_effet INT NOT NULL
);

CREATE TABLE declencher (
    dec_id INT PRIMARY KEY,
    dec_attaque INT NOT NULL,
    dec_effet INT NOT NULL
);

CREATE TABLE engendrer (
    eng_id INT PRIMARY KEY,
    eng_statut INT NOT NULL,
    eng_effet INT NOT NULL
);

CREATE TABLE craindre (
    cra_id INT PRIMARY KEY,
    cra_type_un INT NOT NULL,
    cra_type_deux INT NOT NULL,
    cra_effet INT NOT NULL,
```

```
    cra_signe VARCHAR(255)
);

CREATE TABLE etretype (
    etr_id INT PRIMARY KEY,
    etr_pokemon INT NOT NULL,
    etre_type INT NOT NULL
);

CREATE TABLE peutapprendre (
    peu_id INT PRIMARY KEY,
    peu_pokemon INT NOT NULL,
    peu_attaque INT NOT NULL,
    peu_niveau INT NOT NULL
);

CREATE TABLE vaevoluer (
    vae_id INT PRIMARY KEY,
    vae_niveau INT NOT NULL,
    vae_pokemon INT NOT NULL,
    vae_evolution INT NOT NULL
);

CREATE TABLE saitexprimer (
    sai_id INT PRIMARY KEY,
    sai_pokemon INT NOT NULL,
    sai_talent INT NOT NULL
);

CREATE TABLE detenir (
    det_id INT PRIMARY KEY,
    det_personnage INT NOT NULL,
    det_objet INT NOT NULL,
    det_quantite INT NOT NULL
);

CREATE TABLE constituerquipe (
    con_id INT PRIMARY KEY,
    con_pokemon INT,
    con_objet INT,
    con_talent INT,
    con_statut INT,
    con_attaque_un INT,
    con_attaque_deux INT,
    con_attaque_trois INT,
    con_attaque_quatre INT,
    con_iv_attaque INT,
    con_iv_defense INT,
    con_iv_attaque_speciale INT,
```

```
    con_iv_defense_speciale INT,
    con_iv_vitesse INT,
    con_niveau INT,
    con_pv_restants INT,
    con_ordre INT,
    con_experience INT,
    con_shiny BOOLEAN,
    con_personnage INT
);

-- Constraints d'intégrité

ALTER TABLE badge ADD FOREIGN KEY (bad_type) REFERENCES type(typ_id) ON DELETE CASCADE;

ALTER TABLE ennemi ADD FOREIGN KEY (enn_strategie) REFERENCES strategie(str_id) ON DELETE CASCADE;
ALTER TABLE ennemi ADD FOREIGN KEY (enn_personnage) REFERENCES personnage(per_id) ON DELETE CASCADE;

ALTER TABLE partie ADD FOREIGN KEY (par_membre) REFERENCES membre(mem_id) ON DELETE CASCADE;
ALTER TABLE partie ADD FOREIGN KEY (par_personnage) REFERENCES personnage(per_id) ON DELETE CASCADE;
ALTER TABLE partie ADD FOREIGN KEY (par_statut) REFERENCES statut(sta_id) ON DELETE CASCADE;

ALTER TABLE recompenser ADD FOREIGN KEY (rec_partie) REFERENCES partie(par_id) ON DELETE CASCADE;
ALTER TABLE recompenser ADD FOREIGN KEY (rec_badge) REFERENCES badge(bad_id) ON DELETE CASCADE;

ALTER TABLE delivrer ADD FOREIGN KEY (del_ennemi) REFERENCES ennemi(enn_id) ON DELETE CASCADE;
ALTER TABLE delivrer ADD FOREIGN KEY (del_badge) REFERENCES badge(bad_id) ON DELETE CASCADE;

ALTER TABLE offrir ADD FOREIGN KEY (off_ennemi) REFERENCES ennemi(enn_id) ON DELETE CASCADE;
ALTER TABLE offrir ADD FOREIGN KEY (off_objet) REFERENCES objet(obj_id) ON DELETE CASCADE;

ALTER TABLE combattre ADD FOREIGN KEY (com_personnage) REFERENCES personnage(per_id) ON DELETE CASCADE;
ALTER TABLE combattre ADD FOREIGN KEY (com_ennemi) REFERENCES ennemi(enn_id) ON DELETE CASCADE;
ALTER TABLE combattre ADD FOREIGN KEY (com_climat) REFERENCES climat(cli_id) ON DELETE CASCADE;
```

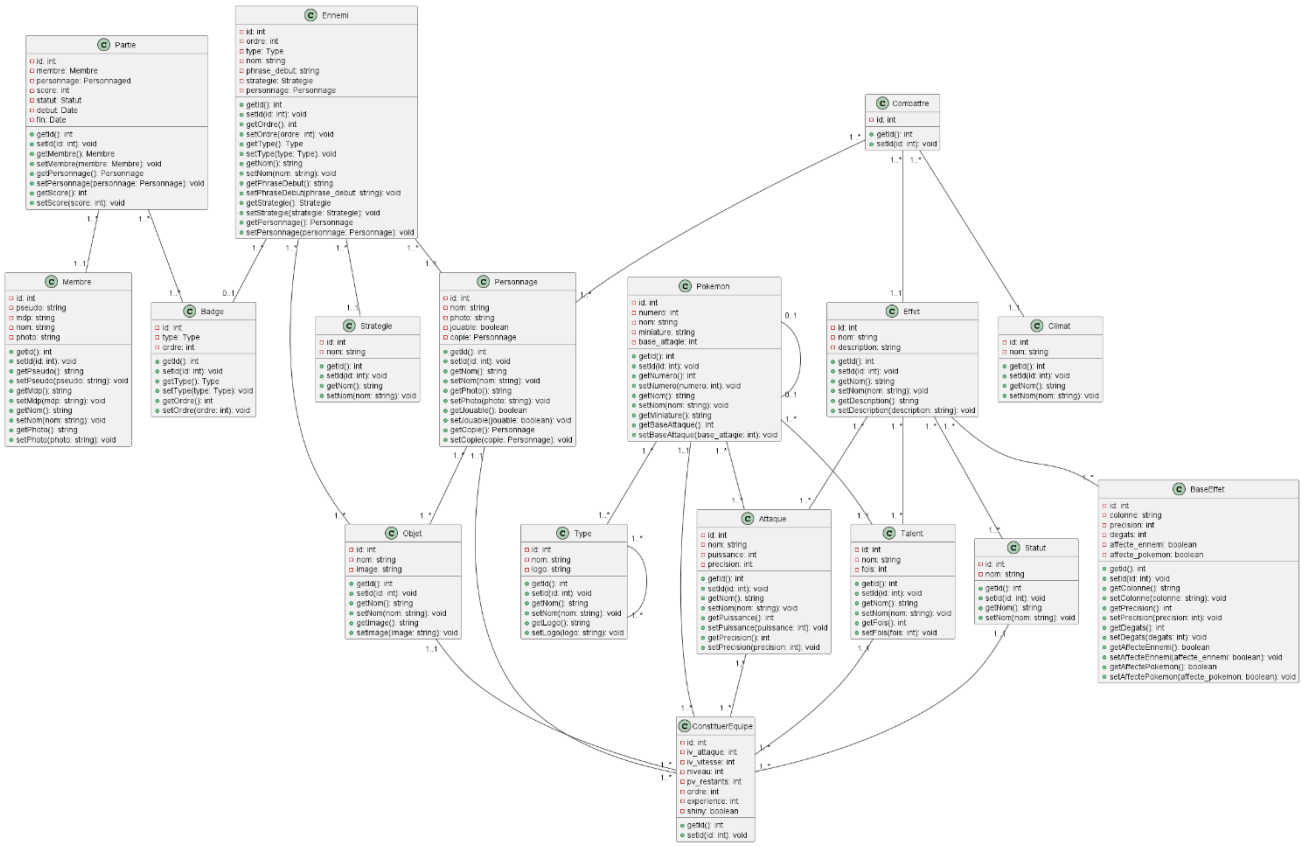
```
ALTER TABLE sortcombattre ADD FOREIGN KEY (sor_effet) REFERENCES effet(eff_id)
ON DELETE CASCADE;
ALTER TABLE sortcombattre ADD FOREIGN KEY (sor_combattre) REFERENCES combat-
tre(com_id) ON DELETE CASCADE;

ALTER TABLE decomposer ADD FOREIGN KEY (dec_effet) REFERENCES effet(eff_id)
ON DELETE CASCADE;
ALTER TABLE decomposer ADD FOREIGN KEY (dec_base_effet) REFERENCES baseef-
fet(bas_id) ON DELETE CASCADE;

ALTER TABLE appliquer ADD FOREIGN KEY (app_effet) REFERENCES effet(eff_id) ON
DELETE CASCADE;
ALTER TABLE appliquer ADD FOREIGN KEY (app_objet) REFERENCES objet(obj_id) ON
DELETE CASCADE;

ALTER TABLE exprimer ADD FOREIGN KEY (exp_talent) REFERENCES talent(tal_id)
ON DELETE CASCADE;
ALTER TABLE exprimer ADD FOREIGN KEY (exp_effet) REFERENCES effet(eff_id) ON
DELETE CASCADE;

ALTER TABLE declencher ADD FOREIGN KEY (dec_attaque) REFERENCES at-
taque(att_id) ON DELETE CASCADE;
ALTER TABLE declencher  ADD FOREIGN KEY (dec_effet) REFERENCES effet(eff_id)
ON DELETE CASCADE;

ALTER TABLE engendrer ADD FOREIGN KEY (eng_statut) REFERENCES statut(sta_id)
ON DELETE CASCADE;
ALTER TABLE engendrer ADD FOREIGN KEY (eng_effet) REFERENCES effet(eff_id) ON
DELETE CASCADE;

ALTER TABLE craindre ADD FOREIGN KEY (cra_type_un) REFERENCES type(typ_id) ON
DELETE CASCADE;
ALTER TABLE craindre ADD FOREIGN KEY (cra_type_deux) REFERENCES type(typ_id)
ON DELETE CASCADE;

ALTER TABLE etretype ADD FOREIGN KEY (etr_pokemon) REFERENCES pokemon(pok_id)
ON DELETE CASCADE;
ALTER TABLE etretype ADD FOREIGN KEY (etre_type) REFERENCES type(typ_id) ON
DELETE CASCADE;

ALTER TABLE peutapprendre ADD FOREIGN KEY (peu_pokemon) REFERENCES
pokemon(pok_id) ON DELETE CASCADE;
ALTER TABLE peutapprendre ADD FOREIGN KEY (peu_attaque) REFERENCES at-
taque(att_id) ON DELETE CASCADE;

ALTER TABLE vaevoluer ADD FOREIGN KEY (vae_pokemon) REFERENCES pokemon(pok_id)
ON DELETE CASCADE;
ALTER TABLE vaevoluer ADD FOREIGN KEY (vae_evolution) REFERENCES
pokemon(pok_id) ON DELETE CASCADE;
```

```
ALTER TABLE saitexprimer ADD FOREIGN KEY (sai_pokemon) REFERENCES pokemon(pok_id) ON DELETE CASCADE;
ALTER TABLE saitexprimer ADD FOREIGN KEY (sai_talent) REFERENCES talent(tal_id) ON DELETE CASCADE;

ALTER TABLE detenir ADD FOREIGN KEY (det_personnage) REFERENCES personnage(per_id) ON DELETE CASCADE;
ALTER TABLE detenir ADD FOREIGN KEY (det_objet) REFERENCES objet(obj_id) ON DELETE CASCADE;

ALTER TABLE constituerquipe ADD FOREIGN KEY (con_pokemon) REFERENCES pokemon(pok_id) ON DELETE CASCADE;
ALTER TABLE constituerquipe ADD FOREIGN KEY (con_objet) REFERENCES objet(obj_id) ON DELETE CASCADE;
ALTER TABLE constituerquipe ADD FOREIGN KEY (con_talent) REFERENCES talent(tal_id) ON DELETE CASCADE;
ALTER TABLE constituerquipe ADD FOREIGN KEY (con_statut) REFERENCES statut(sta_id) ON DELETE CASCADE;
ALTER TABLE constituerquipe ADD FOREIGN KEY (con_attaque_un) REFERENCES attaque(att_id) ON DELETE CASCADE;
ALTER TABLE constituerquipe ADD FOREIGN KEY (con_attaque_deux) REFERENCES attaque(att_id) ON DELETE CASCADE;
ALTER TABLE constituerquipe ADD FOREIGN KEY (con_attaque_trois) REFERENCES attaque(att_id) ON DELETE CASCADE;
ALTER TABLE constituerquipe ADD FOREIGN KEY (con_attaque_quatre) REFERENCES attaque(att_id) ON DELETE CASCADE;
ALTER TABLE constituerquipe ADD FOREIGN KEY (con_personnage) REFERENCES personnage(per_id) ON DELETE CASCADE
```

Modèle physique de données

UML

Classes

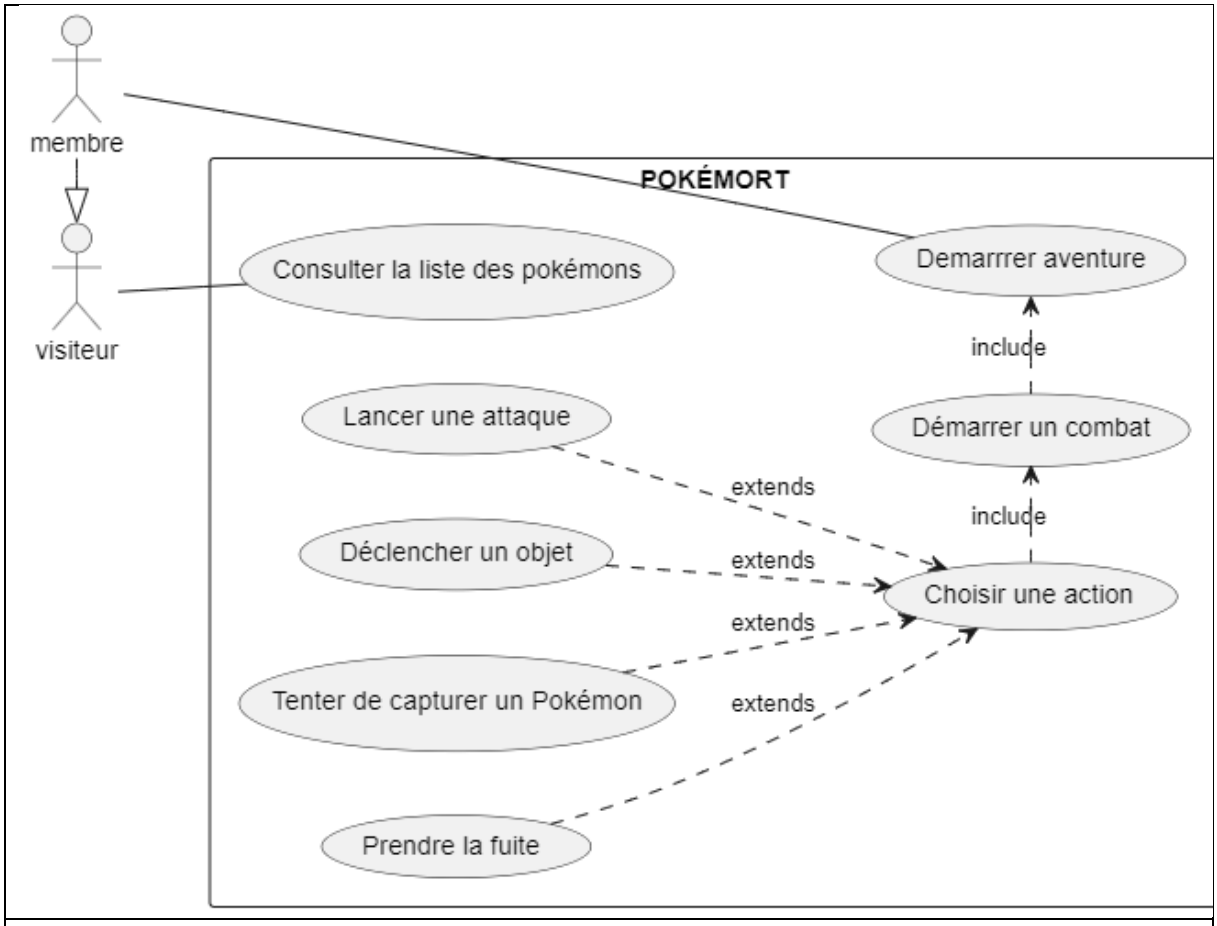


Cas d’utilisation

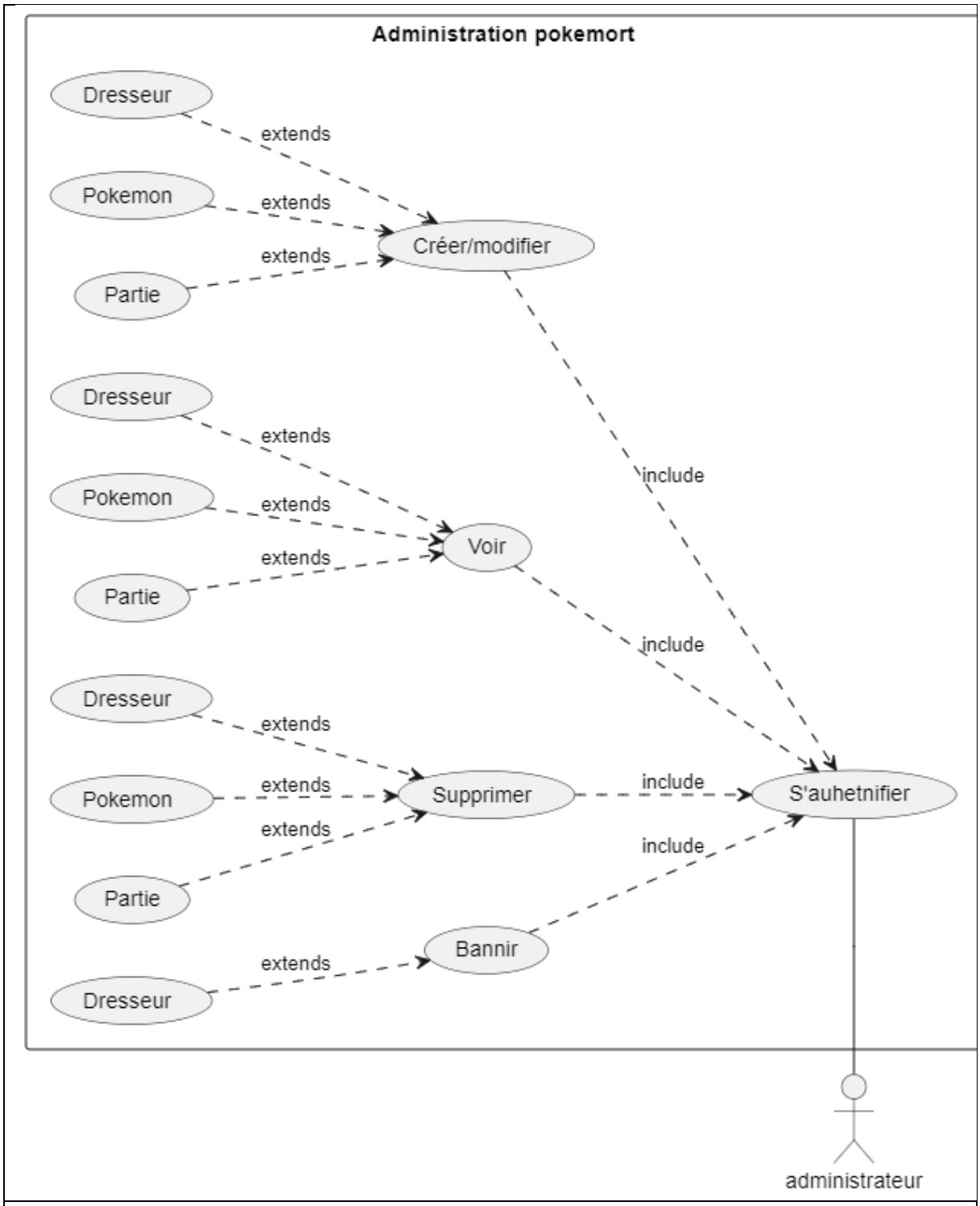
Liste

- Consulter la liste des Pokémons
- Démarrer l’aventure
- Démarrer un combat
- Choisir une aciton
- Lancer une attaque
- Lancer un objet
- Tenter de capturer un pokémon
- Prendre la fuite
- Ajouter, éditer ou supprimer des données de la base de données
- Bannir un utilisateur

Représentation visuelle



Cas d'utilisation pour les membres de l'application



Cas d'utilisation pour les administrateurs

Contraintes Techniques

Technologies utilisées : J2EE, React, MySQL

Backend (J2EE) :

Le backend du projet est basé sur **J2EE**, avec des services RESTful pour la communication entre le frontend React et le backend. Le serveur d'applications doit être configuré pour gérer les sessions utilisateurs et l'intégrité des données (captures, pertes de Pokémon). La sécurité sera assurée par des mécanismes d'authentification sécurisée (OAuth ou JWT).

Frontend (React)

Le frontend, développé en **React**, doit être interactif, fluide et responsive. La gestion de l'état du jeu (niveau, vie, évolution des Pokémon) sera effectuée via Redux ou Context API. Les appels API RESTful permettront d'interagir avec le backend. L'interface doit être optimisée pour différents appareils.

Base de données (MySQL)

La base de données **MySQL** stocke les informations sur les utilisateurs, les Pokémon et les événements du jeu. La modélisation doit prendre en compte la gestion des captures et des pertes conformément aux règles du Nuzlocke Challenge. Les requêtes doivent être performantes et les données sensibles doivent être sécurisées avec des techniques de chiffrement.

Performance et Scalabilité

La gestion des performances est essentielle pour garantir une bonne expérience utilisateur. La base de données et le backend doivent être conçus pour supporter un grand nombre d'utilisateurs et d'événements en parallèle. Des solutions comme la mise en cache et la scalabilité horizontale seront mises en place pour optimiser la réactivité du jeu.

Le projet **POKEMORT** nécessite une architecture robuste pour garantir une expérience de jeu fluide et sécurisée, avec une gestion efficace des données et une performance optimale.