



Documentation Technique

Le jeu Age of War est constitué de 3 parties importantes assurant son bon fonctionnement, une base de donnée mysql, une API en JS et un client lourd (non compilé) en Python.

Pour modifier le jeu vous aurez besoins d'avoir des bases en JS, SQL et Python.

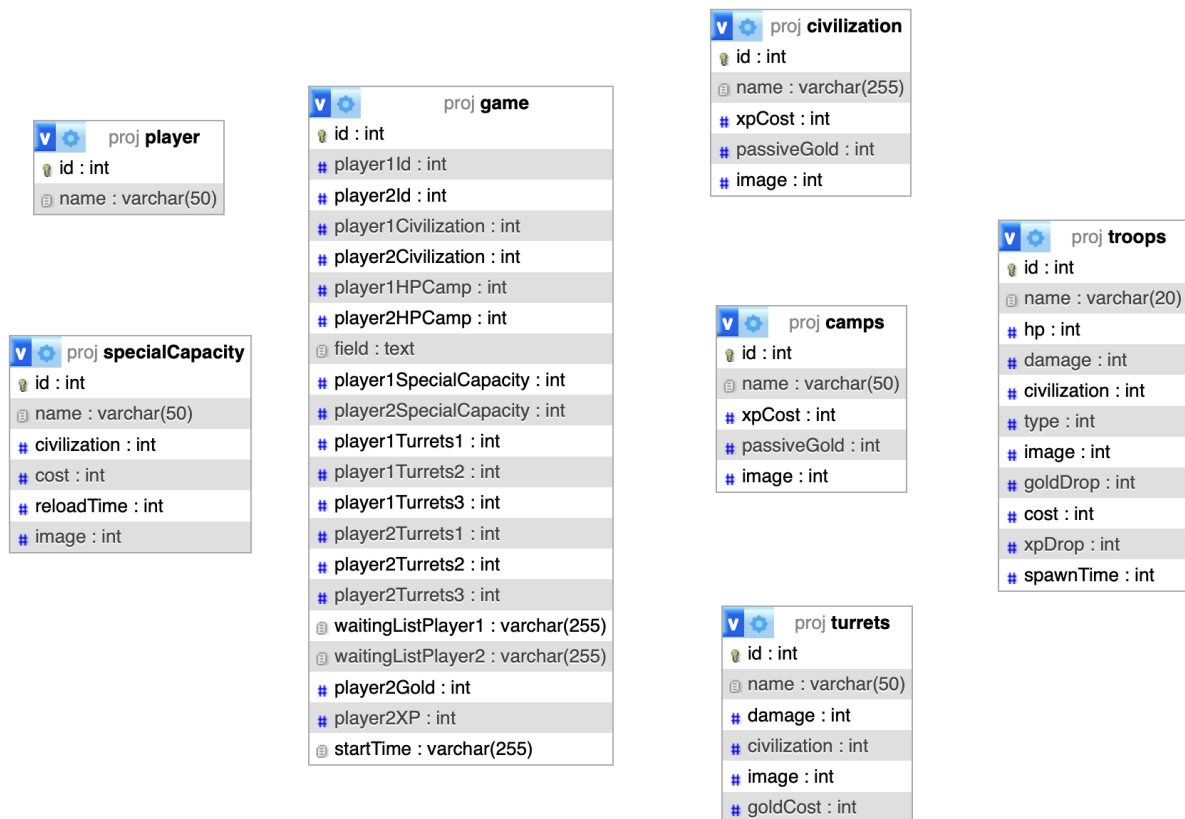
Sommaire

1. Base de donnée
2. API
3. Backend
 - a. Démarrage d'une partie
 - b. Gestion d'une partie
4. Frontend

1. Base de donnée

La base de donnée tourne tous le temps sur un serveur eliascastel.ddns.net et sur le port 3306.

Voici la structure de la base de donnée:



2. API

L'api à été faite en JS à l'aide de expressJS voici le lien de la documentation de l'api: <http://eliascastel.ddns.net:3001/doc/>

Celle ci permet le lien entre les joueurs et l'enregistrement des données

Cette API est conçue pour gérer le jeu en ligne, la gestion des parties et la communication entre les joueurs à l'aide de la base de donnée. Voici un résumé de ce que chaque endpoint permet de faire :

Endpoints principaux

- **/:** Vérification de l'état du serveur (GET).
- **/game/:**
 - **GET:** Obtenir des informations générales sur le jeu.
 - **POST:** Créer un nouveau jeu en spécifiant le joueur via un paramètre de requête.
 - **DELETE:** Supprimer un jeu en spécifiant l'ID du jeu via un paramètre de requête.

- **/game/get/{id}**: Obtenir les détails d'un jeu spécifique en fournissant son ID (GET).
- **/game/joinGame**: Rejoindre un jeu existant en spécifiant le joueur, l'ID du jeu et l'heure de début via des paramètres de requête (PUT).
- **/game/data**: Mettre à jour les données d'un jeu (ex. capacités spéciales des joueurs, tourelles, liste d'attente, ressources) en utilisant divers paramètres de requête (PUT).
- **/game/camps**: Obtenir des informations sur les camps dans le jeu (GET).
- **/game/civilization**: Obtenir des informations sur les civilisations disponibles dans le jeu (GET).
- **/game/specialCapacity**: Obtenir des informations sur les capacités spéciales des joueurs (GET).
- **/game/troops**: Obtenir des informations sur les troupes disponibles dans le jeu (GET).
- **/game/turrets**: Obtenir des informations sur les tourelles disponibles dans le jeu (GET).

Endpoints relatifs aux images

- **/image/background**: Obtenir l'image de fond du jeu (GET).

Endpoints relatifs aux joueurs

- **/player/**:
 - **GET**: Obtenir des informations générales sur les joueurs.
 - **POST**: Créer un nouveau joueur en spécifiant son nom via un paramètre de requête.
 - **DELETE**: Supprimer un joueur en spécifiant son ID via un paramètre de requête.
- **/player/{id}**: Obtenir des informations détaillées sur un joueur spécifique en fournissant son ID (GET).

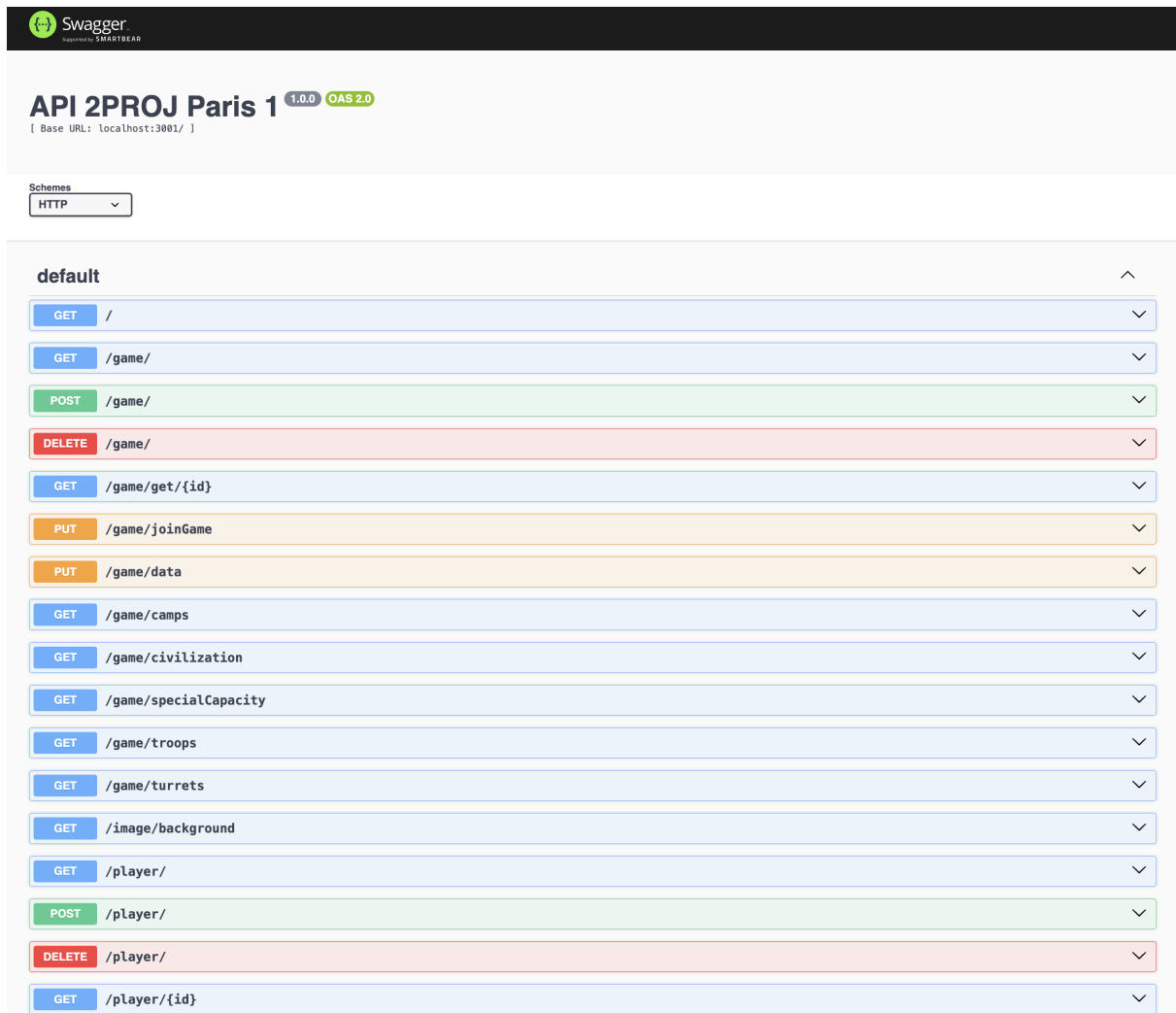
Codes de réponse

- **200**: La requête a été traitée avec succès.
- **201**: Une nouvelle ressource a été créée.

- **400:** La requête était incorrecte.
- **404:** La ressource demandée n'a pas été trouvée.
- **500:** Une erreur interne du serveur s'est produite.

Cette API offre donc des fonctionnalités complètes pour gérer les aspects du jeu en ligne, y compris la création et la gestion de jeux, la manipulation de données de jeu, ainsi que la gestion des informations sur les joueurs.

Au final nous obtenons un swagger qui correspond à cela:



The screenshot displays the Swagger UI for the API 2PROJ Paris 1. The interface includes a header with the Swagger logo and the API title. Below the header, there is a dropdown menu for selecting a scheme (currently set to HTTP). The main content area lists the API endpoints, grouped by HTTP method. The endpoints are as follows:

Method	Endpoint
GET	/
GET	/game/
POST	/game/
DELETE	/game/
GET	/game/get/{id}
PUT	/game/joinGame
PUT	/game/data
GET	/game/camps
GET	/game/civilization
GET	/game/specialCapacity
GET	/game/troops
GET	/game/turrets
GET	/image/background
GET	/player/
POST	/player/
DELETE	/player/
GET	/player/{id}

a. BackEnd

Démarrage d'une partie

Ce code initialise un jeu en utilisant Pygame pour la gestion de la fenêtre et l'affichage des éléments graphiques, ainsi que Pyglet pour les animations. Voici les étapes de l'initialisation :

1. **Importation des bibliothèques nécessaires** : Les bibliothèques Pygame, Requests, Array, PIL et Pyglet sont importées pour différentes fonctions du jeu.
2. **Définition de la fonction d'initialisation `init`** : Une fonction `init` est définie pour regrouper tout le code d'initialisation.
3. **Définition de l'URL** : Une URL est définie mais elle n'est pas utilisée dans la fonction.
4. **Initialisation de Pygame** : La bibliothèque Pygame est initialisée pour préparer la gestion de la fenêtre de jeu et des graphismes.
5. **Obtention des dimensions de l'écran** : Les dimensions actuelles de l'écran sont obtenues pour configurer correctement la fenêtre de jeu.
6. **Définition de la légende de la fenêtre** : Le titre de la fenêtre de jeu est défini.
7. **Calcul des dimensions de la fenêtre tout en maintenant le ratio d'aspect** : Les dimensions de la fenêtre sont calculées pour s'adapter à l'écran tout en maintenant le ratio d'aspect correct.
8. **Création de la fenêtre en mode plein écran** : Une fenêtre de jeu en plein écran est créée avec les dimensions calculées.
9. **Chargement et affichage de l'image de fond** : Une image de fond est chargée, redimensionnée et affichée pour couvrir toute la fenêtre de jeu.
10. **Affichage d'un rectangle gris pour les informations** : Un rectangle gris est affiché en haut à gauche de la fenêtre pour contenir les informations de jeu.
11. **Affichage des textes d'expérience et d'or** : Les textes indiquant l'expérience et l'or du joueur sont affichés dans le rectangle gris.
12. **Chargement et affichage de l'image des paramètres** : Une image représentant les paramètres du jeu est affichée en haut à droite de la

fenêtre.

13. **Affichage des images de base** : Des images représentant les bases du jeu sont affichées à gauche et à droite de la fenêtre.
14. **Affichage des barres de vie** : Des barres de vie sont affichées en dessous des images de base pour indiquer l'état de vie des bases.
15. **Affichage des textes de vie** : Des textes indiquant les points de vie sont affichés en dessous des barres de vie.
16. **Chargement et affichage d'une animation** : Une animation de guerrier squelette est chargée et affichée à une position spécifique sur l'écran.
17. **Affichage des boutons d'action pour les troupes et les tourelles** : Plusieurs boutons d'action pour les troupes et les tourelles sont affichés sur l'écran. Chaque bouton est représenté par un texte et encadré par un rectangle.
18. **Affichage du bouton d'ère et de l'attaque spéciale** : Des boutons pour changer d'ère et pour une attaque spéciale sont affichés sur l'écran.

En résumé, le code configure l'environnement de jeu en plein écran, charge et affiche diverses images, crée des zones pour les informations de jeu et configure des éléments interactifs tels que les boutons d'action et les barres de vie.

b. Gestion d'une partie

Ce code gère la partie du jeu en orchestrant les différentes étapes nécessaires pour initialiser et lancer le jeu en fonction de la difficulté choisie par le joueur. Voici une explication détaillée de son fonctionnement :

1. **Importation des modules nécessaires** : Les fonctions nécessaires pour créer un joueur, démarrer le jeu, initialiser l'environnement de jeu, gérer les modes de jeu solo et multijoueur, et afficher le menu principal sont importées.

2. **Définition de la fonction `main`** : La fonction principale `main` gère le flux du jeu.
3. **Affichage du menu et obtention de la difficulté** : La fonction `menu` affiche le menu principal et retourne le niveau de difficulté choisi par le joueur.
4. **Gestion des différents niveaux de difficulté et des modes de jeu** : Une structure de correspondance (match) est utilisée pour exécuter différentes actions en fonction de la difficulté choisie :
 - **Facile, moyen, difficile, impossible** : Pour chaque niveau de difficulté, le joueur est créé, le jeu est démarré, la fenêtre de jeu est initialisée, et le mode solo est lancé avec la difficulté correspondante.
 - **Multijoueur** : Le joueur est créé, le jeu est démarré, la fenêtre de jeu est initialisée, et le mode multijoueur est créé en attendant un second joueur.
 - **Difficulté numérique** : Si la difficulté est un entier, cela signifie que le joueur souhaite rejoindre un jeu multijoueur existant. Le joueur est créé, la fenêtre de jeu est initialisée, et le mode multijoueur est rejoint.
 - **Cas par défaut** : Si la difficulté n'est pas reconnue, un message d'erreur est affiché.
5. **Bloc d'exécution conditionnelle** : Ce bloc s'exécute si le script est lancé directement. Il initialise le jeu avec une difficulté par défaut (moyenne).

En résumé, ce code gère la partie en orchestrant la création du joueur, le démarrage du jeu, l'initialisation de l'environnement de jeu, et en appelant les fonctions appropriées pour lancer le jeu en mode solo ou multijoueur en fonction de la difficulté choisie par le joueur.

Front End

Ce projet de jeu utilise Pygame pour créer une interface graphique interactive. Voici une description détaillée de l'interface utilisateur et des éléments de la fenêtre de jeu :

Initialisation de la fenêtre de jeu

- **URL du Serveur** : Le jeu se connecte à une URL spécifique pour récupérer des données.
- **Initialisation de Pygame** : Pygame est initialisé pour créer la fenêtre de jeu.
- **Taille de l'écran** : La taille de l'écran est récupérée pour adapter l'affichage du jeu.

Fenêtre de jeu

- **Dimensions de la fenêtre** : Les dimensions de la fenêtre sont ajustées en fonction de la taille de l'écran tout en maintenant le ratio d'aspect.
- **Mode Plein Écran** : La fenêtre de jeu est affichée en plein écran.

Interface Utilisateur

1. Image de Fond :

- Une image de fond est chargée et redimensionnée pour s'adapter à la taille de la fenêtre.

2. Zone d'Informations :

- Une zone rectangulaire grise est dessinée en haut à gauche de l'écran pour afficher des informations sur le jeu, comme l'expérience (XP) et l'or (GOLD) du joueur.

3. Textes d'Informations :

- **XP** : Affiché en blanc avec une taille de police de 48.
- **GOLD** : Affiché en blanc avec la même taille de police.

4. Icône de Paramètres :

- Une icône de paramètres est affichée en haut à droite de l'écran pour permettre l'accès aux options de configuration du jeu. Elle est redimensionnée pour s'adapter à l'interface.

5. Images de Base :

- Des images représentant les bases du jeu sont placées à gauche et à droite de l'écran, près du bas de la fenêtre.

6. Barres de Vie :

- Deux barres de vie, l'une pour la base de gauche et l'autre pour la base de droite, sont affichées juste en dessous des images de base. Elles montrent la santé des bases avec une partie rouge (vide) et une partie verte (pleine).
- Des textes indiquant la quantité de points de vie (hp) sont également affichés au centre de chaque barre de vie.

Boutons et Contrôles

- **Boutons de Troupes :**
 - Des boutons numérotés (1, 2, 3) pour sélectionner les différentes troupes sont alignés horizontalement près du bas de l'écran. Chaque bouton est entouré d'un cadre.
- **Boutons de Tourelles :**
 - Des boutons pour sélectionner des tourelles sont également alignés près du bas de l'écran.
- **Bouton d'Époque :**
 - Un bouton pour changer d'époque dans le jeu.
- **Bouton d'Attaque Spéciale :**
 - Un bouton pour une attaque spéciale.