Documentation Technique du Projet : Gameforge

Ce document détaille l'architecture et la configuration du projet Django "Gameforge". L'analyse couvre à la fois la partie frontend (statique) et la configuration du backend (Django), en expliquant le rôle et le fonctionnement de chaque fichier et de chaque paramètre de configuration pertinent.

Frontend (Fichiers Statiques)

Cette section couvre les fichiers responsables de l'apparence et du comportement de l'interface utilisateur.

static/css/app.css

Ce fichier définit le style visuel de votre application. Il utilise une approche moderne avec des variables CSS (:root) pour une gestion centralisée des couleurs et des polices, ce qui facilite grandement la maintenance du design.

- **:root** : Définit une palette de couleurs globale (par exemple, --bg pour l'arrière-plan, --primary pour la couleur principale, --danger pour les messages d'erreur). Cela permet de changer le thème du site en modifiant seulement quelques lignes.
- **Général** : Un style de base est appliqué (html, body) avec un dégradé en arrière-plan et une police système moderne (Inter). Les liens (a) sont stylisés pour correspondre à la palette de couleurs.
- **Structure & Layout** : Des classes comme .container, .grid, .grid-2, et .page sont utilisées pour organiser le contenu de manière responsive. Le système de grille (display: grid) est utilisé pour créer des mises en page complexes et flexibles pour les listes de cartes et les formulaires.
- **Composants** : Le fichier définit des styles pour des composants réutilisables :
    - .btn : Des boutons avec des états (normal, survol) et des variantes (.primary, .ghost).
    - .card : Des cartes pour afficher des éléments (probablement des jeux) avec une image (.thumb) et un corps (.card-body).
    - .panel, .form-grid, .table : Des styles pour les panneaux, les formulaires et les tableaux, assurant une cohérence visuelle dans les pages d'administration ou de profil.

    - .auth-card : Un conteneur stylisé spécifiquement pour les formulaires d'authentification (connexion, inscription).
- **Animations & Effets** :
    - .loader et .spinner : Une animation de chargement qui s'affiche pendant les opérations asynchrones. L'animation spin est définie avec @keyframes.
    - backdrop-filter: blur(8px) sur .topbar crée un effet de "verre dépoli" pour la barre de navigation, une touche visuelle moderne.
- **Responsive Design** : La règle @media (max-width:900px) ajuste la mise en page sur les écrans plus petits, en passant les grilles de plusieurs colonnes à une seule colonne pour une meilleure lisibilité sur mobile.

static/js/app.js

Ce script gère les interactions dynamiques de base sur les pages web. Il s'exécute une fois que le contenu HTML de la page est entièrement chargé (DOMContentLoaded).

- **Loader Animation** :
    - **Fonction** : Sélectionne tous les éléments avec la classe .loader.
    - **Comportement** : Il utilise setTimeout pour masquer chaque loader après 5 secondes (5000 ms). C'est une simulation simple qui garantit que le loader disparaît même si une opération attendue ne se termine pas.
    - **Limite** : Cette méthode est une simulation. Dans une application réelle, il serait préférable de masquer le loader en réponse à un événement spécifique (par exemple, la fin d'un chargement de données) plutôt qu'après un délai fixe.
- **Boutons "Favoris"** :
    - **Fonction** : Sélectionne tous les boutons avec la classe .fav-btn.
    - **Comportement** : Un écouteur d'événement (click) est ajouté à chaque bouton. Au clic, il bascule la classe .active. En fonction de la présence de cette classe, le texte du bouton change entre "★ Favori" (actif) et "☆ Favori" (inactif).
    - **Limite** : Cette interaction est purement visuelle et côté client. Elle ne sauvegarde pas l'état des favoris. Pour que cela soit persistant, il faudrait ajouter un appel à une API backend (via fetch ou axios) pour enregistrer le choix de l'utilisateur dans la base de données.

Backend (Projet Django)

Cette section analyse en profondeur la configuration de votre projet Django.

gameforge/settings.py

Ce fichier est le cœur de la configuration de votre projet Django. Il définit comment tous les composants (base de données, applications, fichiers statiques, etc.) interagissent.

Importations et Configuration de Base

- import os, from pathlib import Path, from dotenv import load_dotenv: Ces lignes importent les bibliothèques nécessaires pour interagir avec le système d'exploitation (os), gérer les chemins de fichiers de manière moderne (Path), et charger des variables d'environnement à partir d'un fichier .env (load_dotenv).
- BASE_DIR = Path(__file__).resolve().parent.parent: Cette ligne définit le chemin absolu vers la racine de votre projet. C'est une pratique standard et robuste pour s'assurer que les chemins de fichiers fonctionnent sur n'importe quelle machine.
- load_dotenv(BASE_DIR / ".env"): Charge les variables (comme les clés d'API secrètes) depuis un fichier .env situé à la racine du projet. Cela permet de séparer les informations sensibles du code source.

Paramètres de Sécurité et de Déploiement

- SECRET_KEY = os.environ.get("DJANGO_SECRET_KEY", "..."): Récupère la clé secrète de Django depuis les variables d'environnement. C'est crucial pour la sécurité en production. Une valeur par défaut est fournie pour faciliter le développement.
- DEBUG = os.environ.get("DJANGO_DEBUG", "1") == "1": Active ou désactive le mode débogage. Le récupérer depuis les variables d'environnement permet de le désactiver facilement en production (où il ne doit **jamais** être activé) en changeant la variable à "0".
- ALLOWED_HOSTS = ["127.0.0.1", "localhost"]: Définit les noms de domaine autorisés à servir ce site. En production, vous devriez y ajouter le nom de domaine de votre site.

Applications Installées (INSTALLED_APPS)

C'est la liste de toutes les applications qui composent votre projet.

- **Apps Django par défaut** : django.contrib.admin, auth, contenttypes, sessions, messages, staticfiles sont des applications standards fournissant des fonctionnalités essentielles (interface d'administration, authentification, etc.).
- **Vos applications personnalisées** :
    - ai: Suggère une application dédiée à la gestion des fonctionnalités d'intelligence artificielle (probablement les appels à l'API Hugging Face).
    - accounts: Indique une application pour gérer les comptes utilisateurs (inscription, connexion, profil, etc.).
    - core: Généralement utilisée pour la logique principale du site (pages d'accueil, vues principales, etc.).

Middleware (MIDDLEWARE)

Les middlewares sont des couches de traitement qui interceptent les requêtes et les réponses. L'ordre est important. Ceux listés sont des middlewares standards de Django pour la sécurité (SecurityMiddleware, CsrfViewMiddleware), la gestion des sessions (SessionMiddleware), l'authentification (AuthenticationMiddleware), etc.

Configuration des URLs et des Templates

- ROOT_URLCONF = "gameforge.urls": Indique à Django que le fichier gameforge/urls.py est le point d'entrée pour le routage des URLs.
- TEMPLATES: Configure le moteur de templates de Django.
    - "DIRS": [BASE_DIR / "templates"]: Indique à Django de chercher les templates dans un dossier templates à la racine du projet, en plus des dossiers templates de chaque application.

Base de Données (DATABASES)

- "ENGINE": "django.db.backends.sqlite3" et "NAME": BASE_DIR / "db.sqlite3": Configure le projet pour utiliser une base de données **SQLite**. C'est simple et parfait pour le développement ou les petites applications, mais pour une production à grande échelle, vous pourriez envisager de passer à une base de données plus robuste comme PostgreSQL.

Internationalisation (LANGUAGE_CODE, TIME_ZONE, etc.)

- LANGUAGE_CODE = "fr-fr": Définit le français comme langue par défaut pour le projet.

Gestion des Fichiers Statiques et Média

- STATIC_URL = "/static/": L'URL sous laquelle les fichiers statiques (CSS, JS, images) seront servis.
- STATICFILES_DIRS = [BASE_DIR / "static"]: Indique à Django de chercher des fichiers statiques dans un dossier static à la racine du projet.
- MEDIA_URL = "/media/" et MEDIA_ROOT = BASE_DIR / "media": Configure le stockage et le service des fichiers téléversés par les utilisateurs (par exemple, les images de profil).

Paramètres Personnalisés (Spécifiques à Gameforge)

- **Variables Hugging Face** :
  - HUGGINGFACE_API_TOKEN: Récupère votre clé d'API Hugging Face depuis les variables d'environnement.
  - HF_TEXT_MODEL et HF_IMG_MODEL: Définit les modèles d'IA par défaut pour la génération de texte et d'images. Les rendre configurables via des variables d'environnement est une excellente pratique.
- **Limite d'utilisation de l'IA** :
  - DAILY_GENERATION_LIMIT: Fixe une limite sur le nombre de générations par IA qu'un utilisateur peut effectuer en 24h. C'est une mesure très importante pour contrôler les coûts de l'API et prévenir les abus.
- **Redirections d'Authentification** :
  - LOGIN_URL, LOGIN_REDIRECT_URL, LOGOUT_REDIRECT_URL: Configurent les URLs de redirection pour le système d'authentification de Django, offrant une meilleure expérience utilisateur après la connexion, la déconnexion ou lors d'un accès à une page protégée.

gameforge/urls.py

Ce fichier est le routeur principal de votre projet. Il dirige les requêtes HTTP vers la vue Django appropriée en fonction de l'URL demandée.

- from django.urls import path, include: Importe les fonctions nécessaires pour définir les routes.
- urlpatterns = [...]: C'est la liste qui contient toutes les routes principales du projet.
  - path("admin/", admin.site.urls): Inclut les URLs prédéfinies pour l'interface d'administration de Django.
  - path("", include(("core.urls", "core"), namespace="core")): Inclut toutes les URLs définies dans le fichier urls.py de l'application core. Le

namespace="core" permet de référencer ces URLs de manière unique dans les templates (par exemple, {% url 'core:home' %}). C'est la route principale du site (page d'accueil, etc.).
    - path("accounts/", include(("accounts.urls", "accounts"), namespace="accounts")): Fait de même pour l'application accounts, gérant les URLs comme /accounts/login/, /accounts/register/, etc.
- if settings.DEBUG:: Ce bloc conditionnel ajoute une route spéciale pour servir les fichiers médias (MEDIA_URL) uniquement lorsque le projet est en mode DEBUG. En production, cette tâche doit être déléguée à un serveur web dédié comme Nginx ou Apache pour de meilleures performances.

gameforge/wsgi.py et gameforge/asgi.py

Ces deux fichiers sont des points d'entrée pour les serveurs web afin de communiquer avec votre application Django. Vous les avez probablement obtenus en lançant django-admin startproject et n'avez généralement pas besoin de les modifier.

- **wsgi.py (Web Server Gateway Interface)** : C'est le standard pour les serveurs web Python synchrones.
- **asgi.py (Asynchronous Server Gateway Interface)** : C'est une évolution de WSGI qui prend en charge les applications asynchrones (par exemple, pour gérer des WebSockets ou des requêtes longues durées).