GW2 CounterPicker

Documentation Technique Complète

Version 3.0 - 15/12/2025

L'outil d'intelligence WvW le plus puissant jamais créé pour Guild Wars 2

■ Table des Matières

1. Vue d'ensemble du projet

GW2 CounterPicker est une application web d'analyse de combats WvW (World vs World) pour Guild Wars 2. Elle permet aux joueurs et aux guildes d'analyser leurs performances, de comprendre les compositions ennemies et de recevoir des recommandations stratégiques personnalisées grâce à une intelligence artificielle.

Objectifs principaux

- Analyser les fichiers de combat EVTC générés par ArcDPS
- Fournir des statistiques détaillées par joueur et par escouade
- Détecter automatiquement les compositions ennemies
- Recommander des contre-compositions optimales via IA
- Suivre les performances individuelles et de guilde dans le temps
- Segmenter les données par contexte de combat (Zerg, Guild Raid, Roam)

Stack technique

Composant	Technologie
Backend	FastAPI 0.109+ (Python 3.11+)
Frontend	HTMX + Alpine.js + Tailwind CSS
Templates	Jinja2
Base de données	TinyDB (JSON)
IA	Ollama + Llama 3.2 8B
PDF	ReportLab
API externe	dps.report, API GW2 officielle

2. Architecture technique

Structure des fichiers

```
gw2-counterpicker/
■■■ main.py # Application FastAPI principale
■■■ counter_ai.py # IA de contre-attaque (Ollama)
■■■ counter_engine.py # Moteur de contre-pick statique
■■■ parser.py # Parser EVTC natif
■■■ models.py # Modèles Pydantic
■■■ role_detector.py # Détection automatique des rôles
■■■ translations.py # Support multilingue FR/EN
■
■■■ services/
■ ■■■ gw2_api_service.py # Intégration API GW2
■ ■■■ player_stats_service.py # Stats joueurs/guildes
■ ■■■ performance_stats_service.py # Comparaison gaussienne
■ ■■■ analysis_service.py # Service d'analyse
■ ■■■ file_validator.py # Validation fichiers
■
■■■ routers/
■ ■■■ gw2_api.py # Routes API GW2
■ ■■■ analysis.py # Routes d'analyse
■ ■■■ pages.py # Routes de pages
■
■■■ templates/ # Templates Jinja2
■■■ static/ # Assets statiques
■■■ data/ # Bases TinyDB (JSON)
```

Flux de données

1. L'utilisateur upload un fichier .evtc/.zevtc ou colle un lien dps.report
2. Le système vérifie que c'est un log WvW (filtre PvE/PvP)
3. Le fichier est parsé via dps.report (online) ou le parser local (offline)
4. Les données sont extraites : joueurs, stats, boons, composition
5. Le contexte de combat est détecté (Zerg/Guild/Roam)
6. Les stats sont enregistrées dans TinyDB pour l'historique
7. L'IA génère des recommandations de contre-composition

3. Fonctionnalités principales

3.1 Analyse rapide (Single File)

Permet d'analyser un seul fichier de combat. L'utilisateur peut soit uploader un fichier .evtc/.zevtc directement, soit coller un lien dps.report existant.

- Extraction des stats de tous les joueurs alliés
- Détection de la composition ennemie
- Calcul des DPS, soins, barrière, strips, cleanses, CC
- Génération de boons par joueur (Quickness, Stability, etc.)
- Détection automatique des rôles (DPS, Healer, Stab, Boon, Strip)
- Recommandation de contre-composition via IA

3.2 Analyse soirée (Multiple Files)

Permet d'analyser jusqu'à 100 fichiers simultanément pour obtenir une vue agrégée d'une session de jeu complète.

- Agrégation des statistiques sur tous les combats
- Comptage des victoires, défaites et nuls
- Déduplication des joueurs par nom de compte (account name)
- Calcul du nombre de joueurs uniques
- Composition moyenne de l'escouade
- Top 10 joueurs par dégâts
- Builds les plus joués par classe

3.3 Dashboard personnel

Accessible après connexion avec une clé API GW2. Affiche les statistiques personnelles du joueur et ses guildes.

- Informations du compte GW2 (nom, monde, rang WvW)
- Liste de toutes les guildes du joueur
- Statistiques de carrière (combats, victoires, temps de jeu)
- Import des combats depuis la base de données IA

3.4 Historique des combats

Affiche l'historique des combats du joueur avec des statistiques détaillées par spécialisation et une comparaison de performance.

3.5 Analytics de guilde

Vue dédiée aux statistiques d'une guilde spécifique, incluant les membres actifs, la répartition des rôles et les meilleures compositions de groupe.

4. Système de contexte de combat

Le système de contexte permet de segmenter les données WvW en 3 catégories distinctes, car les métas et stratégies diffèrent selon le type de combat.

4.1 Types de contexte

Contexte	Description	Critères
Zerg	Combats de masse blob vs blob	≥25 alliés OU ≥30 ennemis
Guild Raid	Combats de guilde structurés	10-25 alliés + cohésion guilde ≥50%
Roam	Petit comité / roaming	≤10 alliés ET ≤12 ennemis
Unknown	Cas ambigus	Zone grise entre les catégories

4.2 Algorithme de détection

La fonction guess_fight_context() utilise plusieurs paramètres pour déterminer automatiquement le contexte d'un combat :

- ally_count : Nombre de joueurs alliés dans le combat
- enemy_count : Estimation du nombre d'ennemis
- duration_sec : Durée du combat en secondes
- subgroup_count : Nombre de sous-groupes distincts
- main_guild_ratio : Ratio de joueurs de la guilde dominante (0.0-1.0)

4.3 Sélection manuelle

L'utilisateur peut également sélectionner manuellement le contexte via un sélecteur visuel sur la page d'analyse. Le contexte confirmé par l'utilisateur (context_confirmed) prend priorité sur le contexte détecté (context_detected).

4.4 Pages META par contexte

La page META est disponible en 4 versions filtrées :

- /meta - Tous les contextes combinés
- /meta/zerg - Statistiques Zerg uniquement
- /meta/guild_raid - Statistiques Guild Raid uniquement
- /meta/roam - Statistiques Roam uniquement

5. Analyse des performances

5.1 Comparaison gaussienne

Le système calcule et stocke des statistiques globales pour permettre aux joueurs de comparer leurs performances à l'ensemble de la communauté. Les métriques suivent une distribution gaussienne (normale).

5.2 Catégories de métriques

Catégorie	Indicateurs utilisés
DPS	Damage per second, Down contribution per second
Strip	Strips per second, CC per second
Boon	Quickness, Resistance, Aegis, Superspeed, Stability, Protection, Vigor, Might, Fury, Regeneration, Resolution
Stab	Aegis per second, Stability per second
Heal	Regeneration, Outgoing healing, Barrier, Cleanses, Resurrects (par seconde)

5.3 Algorithme de Welford

Pour calculer la moyenne et l'écart-type de manière incrémentale (sans stocker toutes les valeurs), le système utilise l'algorithme de Welford :

Pour chaque nouvelle valeur x :

- n = n + 1
- delta = x - mean
- mean = mean + delta / n
- M2 = M2 + delta * (x - mean)
- variance = M2 / n (si n > 1)

5.4 Calcul du percentile

Le percentile d'un joueur est calculé via la fonction de répartition cumulative (CDF) de la distribution normale :

```
z_score = (valeur - moyenne) / écart_type
percentile = Φ(z_score) × 100
```

5.5 Ratings

Percentile	Rating	Description
≥95%	Exceptionnel	Top 5% des joueurs
≥75%	Au-dessus	Meilleur que 75% des joueurs
≥25%	Moyenne	Dans la norme
<25%	En dessous	Marge de progression

6. Intégration API GW2

6.1 Connexion

L'utilisateur peut connecter son compte GW2 en fournissant une clé API générée sur le site officiel d'ArenaNet. La clé est stockée de manière sécurisée avec chiffrement Fernet.

6.2 Données récupérées

- Informations du compte (nom, monde, rang WvW)
- Liste des guildes du joueur
- Membres d'une guilde (si l'utilisateur a les permissions)
- Personnages du compte

6.3 Endpoints utilisés

- GET /v2/account - Informations du compte
- GET /v2/account/guilds - Liste des guildes
- GET /v2/guild/{id} - Détails d'une guilde
- GET /v2/guild/{id}/members - Membres d'une guilde
- GET /v2/characters - Liste des personnages

7. Intelligence Artificielle

7.1 Architecture

L'IA utilise Ollama avec le modèle Llama 3.2 8B pour générer des recommandations de contre-composition personnalisées. Elle apprend de chaque combat uploadé.

7.2 Base de connaissances

Chaque combat analysé est enregistré dans une base TinyDB avec :

- Composition ennemie détaillée
- Composition alliée avec builds détaillés
- Résultat (victoire/défaite/nul)
- Durée et statistiques de combat
- Contexte de combat (Zerg/Guild/Roam)

7.3 Prompts contextualisés

L'IA utilise des prompts système différents selon le contexte de combat :

- Zerg : Focus sur les AOE, le timing, la survie dans le blob, les skills de masse
- Guild Raid : Focus sur les synergies de composition, les rôles précis, la discipline
- Roam : Focus sur le 1v1, le burst, la mobilité, le disengage

7.4 Recherche de combats similaires

Avant de générer une recommandation, l'IA recherche des combats similaires dans la base de données en filtrant par :

- Contexte de combat identique
- Composition ennemie proche (specs similaires)
- Résultat (priorité aux victoires pour apprendre ce qui fonctionne)

8. Calculs et métriques

8.1 Détection des rôles

Le système détecte automatiquement le rôle de chaque joueur basé sur sa spécialisation élite et ses statistiques :

Rôle	Critères
DPS	Spécialisations offensives (Berserker, Reaper, Weaver...)
Healer	Spécialisations de soin (Druid, Tempest, Firebrand...)
Stab	Génération de Stability élevée
Boon	Génération de boons élevée (Quickness, Alacrity...)
Strip	Strips et CC élevés (Spellbreaker, Scourge...)

8.2 Calcul du résultat

Le résultat d'un combat est déterminé par le ratio de morts :

```
ratio = enemy_deaths / max(ally_deaths, 1)
```

- Victoire : ratio ≥ 1.5 (50% plus de morts ennemis)
- Défaite : ratio ≤ 0.67 (50% plus de morts alliés)
- Nul : entre 0.67 et 1.5

8.3 Filtre WvW

Le système filtre automatiquement les logs PvE et PvP pour n'accepter que les combats WvW. Les critères de détection sont :

- Présence de cibles avec enemyPlayer: true
- Absence de triggerID correspondant à des boss PvE connus
- fightName ne contenant pas de noms de boss (Vale Guardian, Dhuum...)
- isCM = false (pas de Challenge Mode)
- isTrainingGolem = false

8.4 Boon Generation

La génération de boons est calculée en pourcentage d'uptime sur la durée du combat. Le système supporte deux modes d'affichage :

- Group : Génération sur le groupe du joueur (5 joueurs)
- Squad : Génération sur l'escouade entière (jusqu'à 50 joueurs)

9. Sécurité et stockage

9.1 Rate limiting

Le système limite le nombre de requêtes par IP pour éviter les abus :

- 10 uploads par minute par IP

9.2 Validation des fichiers

Tous les fichiers uploadés sont validés :

- Taille maximale : 50 MB
- Extensions autorisées : .evtc, .zevtc, .zip
- Vérification du contenu (signature de fichier)

9.3 Chiffrement des clés API

Les clés API GW2 sont chiffrées avec Fernet (AES-128-CBC) avant stockage. La clé de chiffrement est générée au premier démarrage et stockée dans les variables d'environnement.

9.4 Bases de données TinyDB

- data/ai_fights.json - Combats pour apprentissage IA
- data/sessions.json - Sessions utilisateurs
- data/player_stats.json - Statistiques individuelles
- data/guild_stats.json - Statistiques de guilde
- data/performance_stats.json - Métriques globales de performance
- data/api_keys.json - Clés API chiffrées

10. Déploiement

10.1 Prérequis

- Python 3.11+
- pip (gestionnaire de paquets Python)
- Ollama (optionnel, pour l'IA)

10.2 Installation locale

```
git clone https://github.com/Roddygithub/gw2-counterpicker.git
cd gw2-counterpicker
python -m venv venv
source venv/bin/activate
pip install -r requirements.txt
python main.py
```

10.3 Variables d'environnement

- FERNET_KEY - Clé de chiffrement pour les API keys
- OLLAMA_URL - URL du serveur Ollama (défaut: http://localhost:11434)

10.4 Docker

L'image Docker embarque uvicorn pour la production.

```
docker build -t gw2-counterpicker .
docker run -p 8000:8000 gw2-counterpicker
```

11. Guide de contribution

11.1 Lancement des tests

```
pip install -r requirements-dev.txt
```

```
pytest
```

11.2 Organisation du code

Le projet suit une architecture claire pour faciliter les contributions :

Dossier/Fichier	Role
routers/	Endpoints HTTP (routes publiques et API)
services/	Logique metier (calculs, traitements)
counter_ai.py	IA et apprentissage (Ollama)
parser.py	Parsing natif des fichiers EVTC
models.py	Modeles Pydantic pour la validation
templates/	Templates Jinja2 pour le frontend

11.3 Conventions

- Ajouter une nouvelle route : Creer un endpoint dans routers/ correspondant
- Ajouter un nouveau service : Creer un fichier dans services/ avec la logique
- Logging : Utiliser logger.getLogger(__name__) dans chaque module
- Variables : snake_case pour Python, camelCase pour JavaScript
- Commits : Utiliser des messages clairs et descriptifs

12. Exemple de flow end-to-end

Scenario complet : Un joueur upload un fichier de combat

- 1. Le joueur upload 20251205-233553.zevtc sur /analyze
- 2. file_validator verifie la taille, l'extension et le contenu ZIP
- 3. analysis_service essaye d'abord d'appeler dps.report (online)
- 4. Si echec, fallback sur le parser local (offline)
- 5. Les stats sont envoyees a role_detector pour identifier les roles
- 6. performance_stats_service.record_player_performance enregistre les metriques
- 7. Si une cle API GW2 est liee, player_stats_service enregistre dans l'historique
- 8. counter_ai enregistre le combat dans ai_fights.json pour l'apprentissage
- 9. FastAPI rend dps_report_result.html avec l'IA et les onglets de stats

13. Recapitulatif des endpoints

13.1 Endpoints publics (HTML)

Methode	Route	Description
GET	/	Page d'accueil
GET	/analyze	Formulaire d'analyse
POST	/analyze	Upload de logs + analyse
GET	/meta	Meta globale
GET	/meta/zerg	Meta context Zerg
GET	/meta/guild_raid	Meta context Guild Raid
GET	/meta/roam	Meta context Roam
GET	/dashboard	Dashboard perso (cle API requise)
GET	/history	Historique perso
GET	/guild/{id}	Analytics guilde

13.2 Endpoints API (JSON)

Methode	Route	Description	Auth
GET	/api/gw2/stats	Stats carriere du compte	Session
GET	/api/gw2/stats/specs	Stats par specialisation	Session
GET	/api/gw2/fights	Historique des combats	Session
GET	/api/gw2/guilds	Liste des guildes	Session
GET	/api/gw2/guild/{id}	Stats detaillees guilde	Session
GET	/api/gw2/recommendations	Recommandations IA perso	Session

14. Limitations connues

- La detection de contexte (Zerg/Guild/Roam) repose sur des heuristiques + corrections utilisateurs
- Les stats meta peuvent etre biaisees vers les guildes/joueurs qui utilisent l'outil
- Les recommandations IA ne connaissent pas les builds/traits/equipements exacts des ennemis
- Seules les specialisations sont detectees, pas les traits ou runes specifiques
- L'IA depend de la disponibilite d'Ollama et du modele Llama 3.2

14.1 Avertissement IA

Les recommandations IA sont basees sur les combats enregistres via GW2 CounterPicker et sur un modele de langage generaliste. Elles ne sont pas garanties optimales et doivent etre utilisees comme aide a la decision, pas comme verite absolue.

15. Securite et RGPD

15.1 Donnees collectees

- Cles API GW2 (chiffrees avec Fernet)
- Nom de compte et personnages GW2
- Historique des combats analyses
- Statistiques de performance agregees
- Adresse IP (logs temporaires)

15.2 Retention des donnees

Les stats de combats sont conservees 6 mois. Les cles API sont stockees jusqu'a suppression par l'utilisateur. Les logs serveur sont conserves 30 jours.

15.3 Suppression des donnees

Pour demander la suppression de vos donnees, contactez l'administrateur via l'adresse email du projet. Un bouton de suppression automatique sera ajoute dans une future version du dashboard.