Rapport de projet
Capital Wars

Réalisé par

Andriamihaja Manu

Julien GARCIA

Lambert Raphael

Michaux-Kinet Alexis

Projet encadré par

Antoine CHOLLET

Semestre 5 Année Universitaire 2025-2026

Remerciements

Nous souhaitons exprimer notre profonde gratitude à M. Antoine Chollet pour son encadrement, ses conseils avisés et son soutien tout au long de ce projet. Son expertise nous a permis de conduire à son terme ce projet.

Enfin, nous adressons nos remerciements à toutes les personnes ayant, de près ou de loin, contribué à la bonne réalisation de ce projet.

Résumé et mots clés

Capitals wars est un jeu de gestion d'entreprise fonctionnant sur WEB conçue pour offrir à la fois aux débutants et aux experts en gestion d'entreprise une opportunité de s'amuser à travers des sessions multijoueur, tout en approfondissant leurs connaissances en gestion d'entreprise. Cette plateforme propose des fonctionnalités adaptées aux trois catégories principales d'utilisateurs identifiées : gestionnaire d'entreprise, étudiant et professeur.

Le projet repose sur le framework Symfony et s'appuie sur les langages de programmation PHP.

Mots clés : Capital Wars, Développement d'Application, Symfony, JavaScript, Gestion de Projet, Gestion d'entreprise, Jeux Vidéo.

Sommaire

Table des figures

Glossaire

Agile : Méthodologie de gestion de projet itérative et collaborative utilisée pour le développement du jeu.
Backlog : Liste hiérarchisée des fonctionnalités à développer ou des tâches à accomplir, gérée ici via GitHub.
DLC (Downloadable Content) : Contenu additionnel téléchargeable (payant ou non) pour étendre le jeu, mentionné dans le modèle économique.
Framework : Ensemble d'outils et de composants logiciels structurant le développement. Ici, le projet utilise **Symfony**.
GitHub : Plateforme d'hébergement de code et de gestion de versions utilisée par l'équipe.
Issue : Unité de travail ou signalement de bug sur GitHub, utilisée pour suivre les tâches individuelles.
Lobby : Salon d'attente virtuel où les joueurs se regroupent avant de lancer une partie.
Mêlée quotidienne (Daily Scrum) : Réunion brève et régulière de l'équipe pour synchroniser les activités du jour.
PEGI (Pan European Game Information) : Système de classification par âge des jeux vidéo que le projet respecte.
Période : Unité de temps dans le jeu (équivalent à un tour ou une session de gestion).
PHP : Langage de programmation utilisé côté serveur pour développer le jeu.
Plugin : Module d'extension permettant d'ajouter ou désactiver des fonctionnalités (ex: TVA, marché noir) sans toucher au cœur du jeu. C'est l'innovation clé de votre projet.
Rétrospective de sprint : Réunion interne à l'équipe après la revue pour analyser le fonctionnement du groupe et s'améliorer.
Revue de sprint : Réunion de fin de cycle pour présenter le travail accompli au client ou tuteur.
RGPD (Règlement Général sur la Protection des Données) : Règlement européen sur la protection des données personnelles, crucial car le jeu collecte des emails d'étudiants.
SAE (Situation d'Apprentissage et d'Évaluation) : Cadre pédagogique universitaire dans lequel ce projet est réalisé.
Serious Game : Jeu vidéo à vocation pédagogique ou sérieuse (apprentissage de la gestion ici), par opposition au divertissement pur.
Sprint : Cycle de développement court (itération) au terme duquel une partie fonctionnelle du produit est livrée.
Stack technologique : Ensemble des technologies (langages, frameworks, outils) utilisées pour construire l'application.
Symfony : Framework PHP utilisé pour garantir la robustesse et la modularité de l'application.
Webhook : Méthode permettant à une application d'envoyer des données en temps réel à une autre. Utilisé ici pour envoyer les formulaires de contact vers Discord.

Introduction

Dans le cadre de notre Situation d'Apprentissage et d'Évaluation (S.A.É.) du 5ème semestre, notre équipe a entrepris la conception et la réalisation d'un projet technologique d'envergure et ambitieux : la création de **Capital Wars**, un jeu de gestion multijoueur en ligne. Ce projet a été pensé non seulement comme une démonstration de nos compétences techniques, mais également comme une exploration approfondie des problématiques liées au développement de plateformes de jeux en ligne robustes et évolutives.

Fortement inspiré par la complexité stratégique et l'engagement à long terme qu'offrent des titres de gestion reconnus comme EXIGE, **Capital Wars** simule la concurrence acharnée et dynamique entre plusieurs entreprises virtuelles. Ces entités, gérées par les joueurs, sont spécialisées dans le secteur exigeant de la fabrication de produits manufacturés. Le cœur du jeu repose sur la prise de décisions stratégiques concernant la chaîne de production, l'innovation technologique, la gestion des ressources humaines, le marketing et la finance, le tout dans un environnement de marché volatile et influencé par les actions des autres participants.

L'objectif principal de ce projet dépassait la simple création d'un jeu fonctionnel. Il visait spécifiquement à développer une plateforme web de jeu de gestion conçue selon une architecture **modulaire**. Cette approche architecturale garantit un **noyau dur de fonctionnalités** absolument stable et performant (gestion des comptes, moteur de jeu de base, interface utilisateur principale). Plus crucial encore, elle offre la possibilité structurelle d'intégrer facilement des **modules supplémentaires appelés "DLC" ou extensions**. Cette modularité est la clé de l'**évolutivité** de l'application, permettant d'ajouter de nouvelles industries, des mécanismes de jeu inédits, des événements spéciaux ou des fonctionnalités sociales sans impacter le moteur principal. En d'autres termes, **Capital Wars** a été conçu pour être une plateforme vivante, capable de croître et de s'adapter aux retours utilisateurs et aux nouvelles idées de développement à long terme.

1. Analyse

1.1. Analyse de l'existant

Le projet **Capital Wars** vise à s'insérer sur le marché des jeux de gestion d'entreprise, en se positionnant comme une adaptation moderne et flexible du concept initialement popularisé par le jeu **Exige**, identifié comme notre référentiel direct et source d'inspiration principale. Notre objectif est de reproduire la profondeur de gestion l'Exige tout en résolvant ses lacunes techniques et structurelles en l'exploitant et créer une plateforme web.

1.1.1. Analyse de la Concurrence Actuelle

Pour définir notre proposition de valeur, nous avons analysé les leaders du marché des simulations économiques "sérieuses" (*Serious Games*) :

Concurrent	Points Forts	Limitations
Virtonomics	Simulation économique en ligne 'hardcore' très complète (RH, production, politique).	Interface austère, courbe d'apprentissage extrêmement raide, modèle parfois rigide.
Sim Companies	Accessibilité, économie dynamique (jeu sur navigateur).	Gameplay simpliste et répétitif, manque d'événements complexes ou scénarisés (grèves, incendies).
Capitalism Lab	Réalisme économique poussé (bourse, R&D).	Principalement solo et hors-ligne, ne permet pas les interactions sociales (alliances, sabotages) au cœur de notre projet.

1.1.2. Synthèse et Positionnement Stratégique

Le marché impose actuellement un compromis : une simulation extrêmement complexe et peu engageante (Virtonomics) ou une expérience accessible mais limitée en profondeur (Sim Companies).

L'innovation clé de notre projet réside dans son architecture modulaire. Contrairement à l'approche monolithique de ses compétiteurs, Capital Wars utilise un système de plugins permettant d'activer ou de désactiver des fonctionnalités spécifiques (Placement des usines, événements aléatoires, etc.).

Cette flexibilité pédagogique unique permet d'ajuster la complexité du jeu au niveau de l'utilisateur (notamment les étudiants), offrant ainsi une adaptabilité que les *Serious Games* traditionnels peinent à atteindre.

1.2. Contexte d'utilisation et environnement du logiciel

1.2.1. Public Cible et Objectifs Pédagogiques

Le projet **Capital Wars** est conçu pour s'adresser spécifiquement à un public d'étudiants en gestion d'entreprise. L'application répond à un besoin pédagogique précis : sensibiliser les utilisateurs à la gestion d'entreprise en conditions réelles à travers une simulation ludique. L'utilisateur est amené à bâtir un empire économique et à affronter des rivaux dans une économie virtuelle dynamique. L'objectif central est de placer les joueurs en situation de compétition pour la création et le développement d'entreprises virtuelles.

Le logiciel est prévu pour une utilisation flexible, utilisant des dynamiques tant compétitives que coopératives. L'expérience utilisateur est calibrée pour s'inscrire dans la durée, avec un cycle de vie du jeu s'étendant sur 2 à 30 sessions distinctes. Chaque session de jeu est dimensionnée pour durer entre 1 à 4 heures par année virtuelle, permettant une immersion suffisante pour gérer les aspects complexes de l'entreprise (ressources, résultats prévisionnels, alliances).

1.2.2. Environnement technique et déploiement

L'environnement logiciel repose sur une architecture Web, rendant l'application accessible via un navigateur standard sans nécessité d'installation lourde côté client. Le développement s'appuie sur le framework Symfony, garantissant robustesse et modularité. L'infrastructure réseau est constituée d’un serveur (celui de Raphaël) gérant deux services distincts pour assurer la fluidité des interactions en temps réel et la persistance des données de l'économie virtuelle

L'environnement d'utilisation intègre aussi des exigences strictes en matière d'accessibilité et de conformité légale. L'interface utilisateur est conçue pour inclure des options de lisibilité, des couleurs adaptées et des sous-titres, garantissant l'accès au plus grand nombre. Sur le plan normatif, le logiciel évolue dans un cadre respectant le Règlement Général sur la Protection des Données (RGPD), les droits d'auteur, ainsi que la classification PEGI, assurant un environnement sécurisé et adapté au public visé.

1.3. Présentation des besoins fonctionnels

1.3.1. Contraintes Techniques et Architecture

Le système se repose sur une architecture robuste et évolutive pour supporter la modularité qui constitue la principale valeur ajoutée du projet.

- **Plateforme cible :** L'application est exclusivement accessible via le WEB, garantissant une accessibilité multi-supports (PC, tablettes) sans installation lourde (installation de logiciels) pour les étudiants.
- **Stack technologique :** Le développement est réalisé avec le framework **Symfony**.
- **Modularité :** Le système est conçu nativement pour supporter l'ajout, l'activation et la désactivation de plugins sans altérer le cœur du jeu.

1.3.2. Ergonomie, Interface et Accessibilité

L'interface utilisateur doit être adaptée à un public étudiant et respecter des normes d'inclusivité. **Direction Artistique :** L'interface doit proposer un équilibre visuel entre un style **"professionnel"** (pour l'aspect gestion) et **"fun"** (pour l'aspect ludique). Les ambiances graphiques envisagées incluent le réaliste, le futuriste ou le cartoon. Néanmoins le jeu doit impérativement intégrer des fonctionnalités pour les utilisateurs en situation de handicap, notamment : une **lisibilité** optimale des textes ou des palettes de **couleurs adaptées**.

1.3.3. Contraintes Légales

Le déploiement du jeu dans un cadre public ou éducatif impose le respect strict des normes en vigueur.

- **Protection des Données :** Le jeu collectant des données utilisateurs (étudiants) et potentiellement des données de paiement DLC, il doit être strictement conforme au **RGPD**.
- **Propriété Intellectuelle :** Le projet gère et respecte les **droits d'auteur**, tant pour les ressources utilisées que pour le contenu créé.

1.3.4. Sécurité

Pour assurer le suivi et l'évolution du projet, des outils spécifiques doivent être intégrés au workflow.

- **Versionnage :** Le code source est géré via **GitHub**.
- **Collaboration :** La communication et le suivi de projet s'effectuent via **Discord**.
- **Disponibilité :** Le jeu est capable de maintenir l'état des parties sur le long terme, pour des durées de jeu s'étalant sur une année avec des sessions intermittentes.

1.3.5. Modèle Économique

Le système est conçu pour supporter techniquement les stratégies de monétisation envisagées la gestion d'un système d'**abonnement** ou la possibilité d'intégrer et de vendre des contenus additionnels sous forme de **DLC**.

1.4. Analyse des besoins non fonctionnels

1.4.1. Diagramme de cas d'utilisation

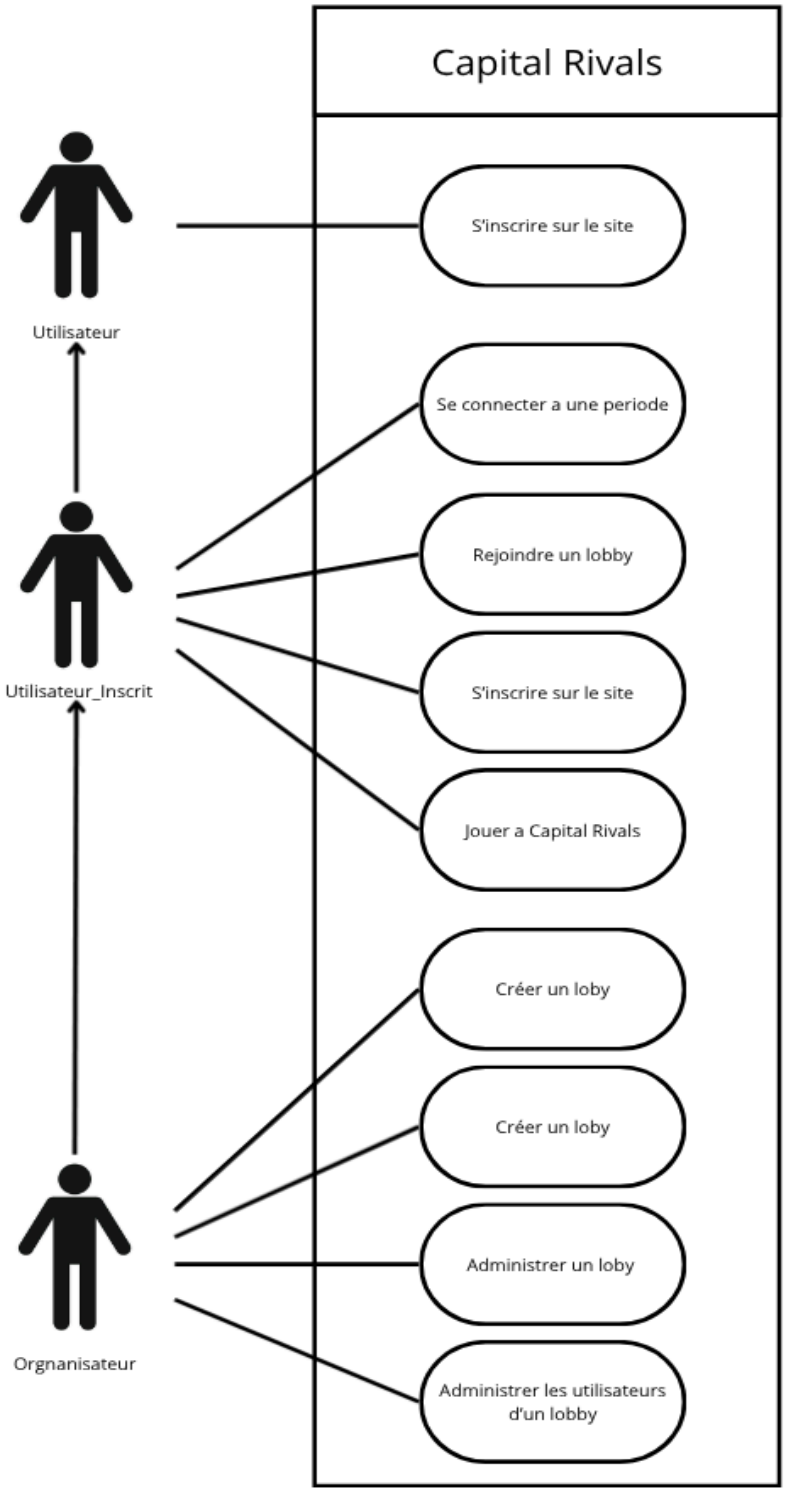


Figure 1: Diagramme de cas d'utilisation UML du système Capital Rivals

1.4.2. Cas d'utilisation

1.4.2.1. Les Acteurs

- **Utilisateur (Visiteur) :** L'acteur de base. Il ne peut faire que s'inscrire.
- **Utilisateur Inscrit :** Il hérite de "Utilisateur". Il peut faire tout ce que fait l'Utilisateur + se connecter, rejoindre, jouer.
- **Organisateur :** Il hérite de "Utilisateur Inscrit". Il a tous les droits précédents + les droits de création et d'administration.

1.4.2.2. Les cas (cf. Figure 1)

Cas n°1 : S'inscrire sur le site

- **Acteur principal :** Utilisateur (et par héritage, tout nouvel arrivant).
- **Pré-conditions :** L'utilisateur n'est pas connecté et ne possède pas de compte.
- **Scénario nominal :**
    1. L'utilisateur sélectionne l'option d'inscription.
    2. Le système demande les informations (pseudo, email, mot de passe).
    3. L'utilisateur remplit le formulaire.
    4. Le système valide les données et crée le compte.
- **Post-condition :** L'utilisateur possède un compte et devient un "Utilisateur Inscrit".

Cas n°2 : Se connecter à une période

- **Acteur principal :** Utilisateur Inscrit.
- **Pré-conditions :** L'utilisateur possède un compte valide.
- **Scénario nominal :**
    1. L'utilisateur demande à se connecter au système.
    2. Le système demande les identifiants.
    3. L'utilisateur saisit ses identifiants.
    4. Le système authentifie l'utilisateur.
    5. Le système affiche les périodes (sessions de jeu) disponibles.
    6. L'utilisateur sélectionne une période active.
- **Post-condition :** L'utilisateur est connecté et prêt à rejoindre un lobby.

Cas n°3 : Rejoindre un lobby

- **Acteur principal :** Utilisateur Inscrit.
- **Pré-conditions :** L'utilisateur est connecté. Un lobby doit exister.
- **Scénario nominal :**
    1. Le système affiche la liste des lobbies disponibles.
    2. L'utilisateur sélectionne un lobby dans la liste.
    3. (Optionnel) Si le lobby est privé, le système demande un code.
    4. L'utilisateur confirme sa volonté de rejoindre.
    5. Le système ajoute l'utilisateur à la salle d'attente du lobby.
- **Post-condition :** L'utilisateur est membre du lobby.

Cas n°4 : Jouer à Capital Wars

- **Acteur principal :** Utilisateur Inscrit.
- **Pré-conditions :** L'utilisateur a rejoint un lobby et la partie a commencé.
- **Scénario nominal :**
    1. La partie se lance.
    2. L'utilisateur interagit avec l'interface de jeu (prendre des décisions, faire des actions).
    3. Le système met à jour l'état du jeu en temps réel.
    4. La partie se termine selon les règles définies.
- **Post-condition :** Les scores sont enregistrés et l'utilisateur retourne au lobby ou au menu.

Cas n°5 : Créer un lobby

- **Acteur principal :** Organisateur.
- **Pré-conditions :** L'organisateur est connecté.
- **Scénario nominal :**
    1. L'organisateur demande la création d'un nouveau lobby.
    2. Le système demande les paramètres (Nom du lobby, nombre max de joueurs, règles spécifiques).
    3. L'organisateur configure les paramètres et valide.
    4. Le système génère le lobby.
- **Post-condition :** Le lobby est créé et ouvert aux autres utilisateurs.

Cas n°6 : Administrer un lobby

- **Acteur principal :** Organisateur.
- **Pré-conditions :** L'organisateur a créé un lobby ou en possède les droits.
- **Scénario nominal :**
    1. L'organisateur accède aux paramètres du lobby.
    2. Il peut modifier les règles, changer le nom, ou fermer le lobby.
    3. Le système enregistre les modifications instantanément.

Cas n°7 : Administrer les utilisateurs d'un lobby

- **Acteur principal :** Organisateur.
- **Pré-conditions :** Des utilisateurs sont présents dans le lobby de l'organisateur.
- **Scénario nominal :**
    1. L'organisateur consulte la liste des participants.
    2. Il sélectionne un utilisateur spécifique.
    3. Il choisit une action (Expulser, Bannir).
    4. Le système applique la sanction à l'utilisateur ciblé.

2. Rapport technique

2.1. Conception

2.1.1. Conception général

Le projet est en Symfony 6.4 et basé sur PHP 8.1 avec une architecture modulaire basée sur des plugins. Le système utilise Doctrine ORM pour la persistance des données et MySQL comme moteur de base de données. L'architecture suit le pattern Modèle-Vue-Contrôleur-Services (MVCS) avec une séparation claire entre les entités métier, la logique applicative et la présentation. Le projet est conçu comme un projet symfony classique cette a dire suivant cette architecture :

- `config/` : Contient la **configuration de l'application** (fichiers YAML pour les routes, services, Doctrine, sécurité, mailer, Twig, etc).
- `src/` : Héberge le **Code Source Principal** et la logique métier.
  - **Controller/** : Gère les requêtes HTTP (AccueilController, EntrepriseController, ApiController, AdminController, etc.).
  - **Entity/** : Définit les Modèles de domaine Doctrine (Utilisateur, Entreprise, Usine, Decision, Bilan, Pret, PeriodeDeJeu, etc.).
  - **Plugins/** : Regroupe les Modules Modulaires pour la logique spécifique du jeu (Bourse/, PersonnageRH/, ParisHipique/, Evenements/, etc).
  - **Repository/** : La couche d'accès aux données (EntrepriseRepository, BilanRepository, etc.).
  - **Service/ :** Contient la Logique métier (Services) (PdfService, PluginManager, UtilisateurManager, etc).
  - **Form/** : Les types de formulaires Symfony (UtilisateurType, EntrepriseType, PretType, etc).
  - **Twig/** : Les Extensions Twig personnalisées (TauxEmpruntExtension).
  - **Maker/ :** Les Commandes de génération de code (MakePluginCommand).
  - **Enum/** : Les Énumérations (TypeUsine).
  - **Attribute/** : Les Attributs personnalisés (RequiresPlugin).
- `templates/` : Les **Vues Twig (Rendu)** pour les templates HTML.
  - **emails/** : Templates d'emails.
  - **utilisateur/** : Templates utilisateur
  - **app/ :** Vues spécifiques de l'application
  - **regles/** : Vues des règles du jeu.
  - **admin/** : Vues du panneau d'administration.
  - etc…
- `public/` : Les **Assets Web Statiques** accessibles publiquement.
  - **css/, js/, img/ :** Feuilles de style, scripts JavaScript et images.
  - **uploads/ :** Fichiers utilisateurs (ex. avatars).
- `tests/` : Les **Tests PHPUnit** (fonctionnels et unitaires).
- `migrations/` : Les **Scripts Doctrine Migrations** (Versioning de la base de données).

2.1.2. Base de données

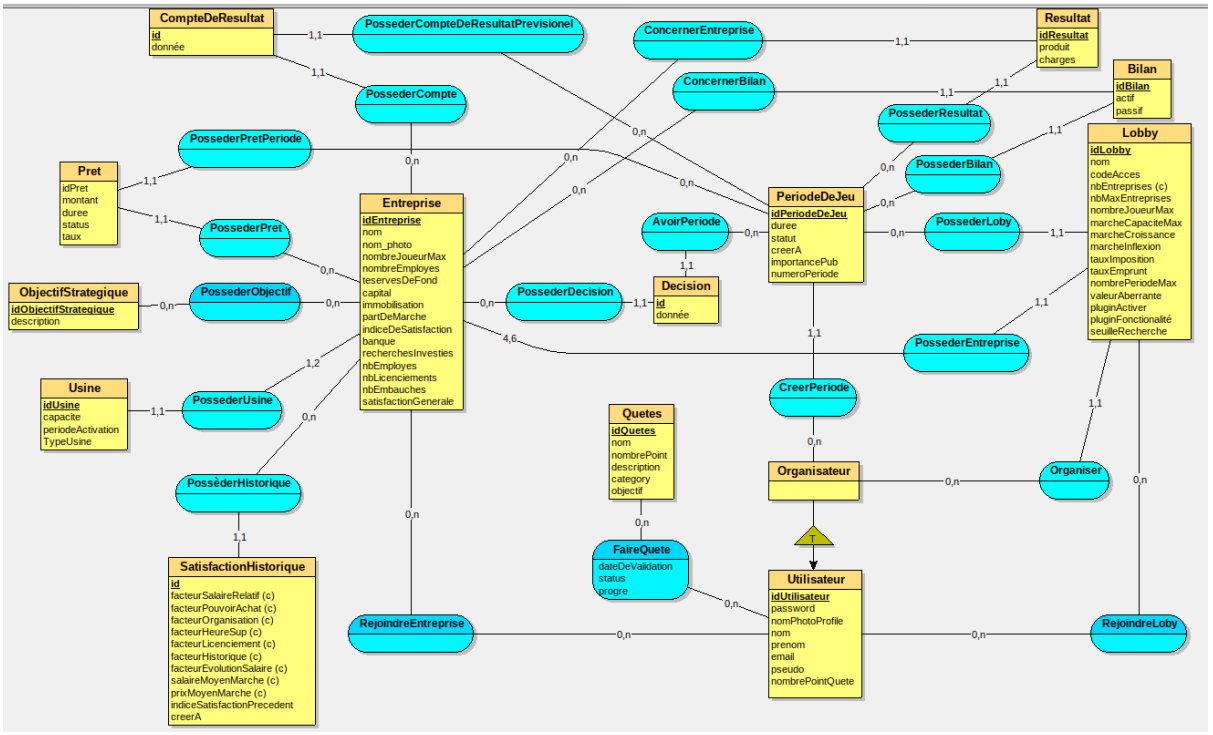


Figure 2: Modèle conceptuel de données - Architecture de la base de données

2.1.2.1. Connexion

L'entité **Utilisateur** est le socle de l'application. Elle centralise les informations de compte, les droits d'accès et la progression persistante.

1. Structure de l'entité Utilisateur

L'entité est conçue pour gérer à la fois l'authentification sécurisée et le profilage du joueur.

- Champs Clés :
  - **id** : Identifiant unique (UUID).
  - **username / password** : Identifiants de connexion (mot de passe haché).
  - **nom / prenom / mail** : Informations d'identité et de contact.
  - **roles (JSON)** : Définit les permissions (ex: ROLE_USER, ROLE_ADMIN).
  - **nb point quete** : Score global cumulé à travers toutes les parties jouées.
- Spécificité : L'Organisateur est un type d'utilisateur particulier (via héritage ou rôle spécifique) possédant des droits de création et de supervision sur les Lobbies.

2.1.2.2. Architecture principal

Le jeu se construit autour de trois entités principales du jeu (**Lobby, Entreprise, PeriodeDeJeu**) celle ci sont liées comme suit :

1. **Lobby et Entreprise** :
   - Un **Lobby** est le conteneur du jeu et **regroupe plusieurs Entreprises** (entre 4 et 6).
   - La relation est **Un à Plusieurs** : Un Lobby contient plusieurs Entreprises, et chaque Entreprise appartient à un seul Lobby.
2. **Lobby et PériodeDeJeu** :

    - Un **Lobby** gère la progression du jeu à travers plusieurs **PériodesDeJeu** successives (tours/cycles de simulation).
    - La relation est **Un à Plusieurs** : Un Lobby contient plusieurs PériodesDeJeu.
3. **Entreprise et autres éléments liés au jeu** :
    - Une **Entreprise** est liée à ses actions et résultats de jeu : elle a des relations Un à Plusieurs avec **Usine, Decision, Resultat, Bilan et Pret**. Ces éléments sont les conséquences et les outils de gestion de l'Entreprise au sein du Lobby et au fil des PériodesDeJeu.
    - **Relations** : **Un à PLusieurs** avec Décision, Resultat, Bilan, **Plusieurs à Un** avec Loby et Organisateur.

2.1.2.3. Entités Financières & Comptables

Les entités financières (Décision, Résultat, Bilan, Prêt) sont au cœur de la simulation économique du jeu et s'articulent de la manière suivante :

1. Liens entre les Entités Financières

- **Décision -> Résultat et Bilan :** Les choix stratégiques enregistrés dans l'entité **Décision** (Ventes_Usines, Achat_Usine, production_a_lancer, etc.) ont un impact direct sur les flux de produits et de charges du **Résultat** (ventes générées, coûts de production, amortissements) et sur la composition de l'actif et du passif du **Bilan** (achat/vente d'usines modifiant l'actif, emprunts modifiant le passif).
- **Prêt -> Résultat et Bilan :** L'entité **Prêt** gère les emprunts.
    - L'émission d'un emprunt (dans Décision) impacte le passif du **Bilan**.
    - Le remboursement annuel (via getAnnuite()) génère des charges financières (intérêts) qui apparaissent dans le Résultat et modifient le passif (diminution de la dette) du Bilan selon l'amortissement.
- **Résultat -> Bilan** : Le bénéfice ou la perte dégagée par le Résultat (getBenefice()) est reporté dans les capitaux propres au passif du Bilan. C'est le lien fondamental entre la performance de la période et la richesse cumulée de l'entreprise.

2. Connexion avec le Reste du Jeu

Les entités financières sont intimement liées aux structures principales du jeu :

- **Entreprise :**
    - Toutes les entités financières sont liées à l'entité **Entreprise** via une relation **Plusieurs à Un** (explicite pour Décision). Chaque entreprise possède son propre ensemble de **Décisions** prises, son **Résultat**, son **Bilan** et ses **Prêts** en cours. L'entreprise est l'acteur qui prend les décisions et subit les conséquences financières.
- **PeriodeDeJeu :**
    - L'entité **Décision** est liée à l'entité **PeriodeDeJeu** via une relation **Plusieurs à Un**. Chaque ensemble de décisions est daté et appartient à une période spécifique.
    - De même, le **Résultat** et le **Bilan** sont des instantanés ou des agrégations de flux financiers pour une période donnée. Ils sont généralement calculés à la fin de chaque **PeriodeDeJeu** en fonction des **Décisions** prises.
- **Lobby :**
    - L'entité **Lobby** est le conteneur de la session de jeu. Elle gère plusieurs

Entreprises et les **PeriodeDeJeu** successives.
    - Indirectement, les entités financières sont toutes contenues au sein du **Lobby** car elles sont attachées aux **Entreprises** qui sont elles-mêmes dans le **Lobby**. Le **Lobby** permet de consolider et de comparer les performances financières des différentes entreprises qui y participent.

2.1.2.4. Entités Financières & Comptables

L'entité **Usine** représente l'outil industriel de l'entreprise. C'est l'actif physique qui transforme les ressources financières et les matières premières en produits à assembler, générant ainsi la valeur qui sera enregistrée dans les entités financières.

1. Structure de l'entité Usine

Chaque usine est une unité distincte possédant ses propres caractéristiques de performance et de cycle de vie.

- **Champs Clés** :
    - type : Définit la puissance des usines.
    - capacité : Le volume maximal de production par PeriodeDeJeu.
    - Période d'activation : La période à laquelle l'usine devient opérationnelle après achat.
    - valeur achat : Le coût d'acquisition initial, inscrit à l'actif du Bilan.

2. Connexion avec le Reste du Jeu

L'Usine est le pivot entre la stratégie opérationnelle et la réalité comptable :
- Lien avec **Entreprise** : Une Entreprise possède plusieurs Usines (1 à 2).
- Lien avec **Décision** & **PériodeDeJeu** : L'achat d'une usine est une Decision prise lors d'une **PeriodeDeJeu** spécifique. L'impact n'est pas immédiat : il faut souvent attendre la période d'activation pour que la capacité de production augmente réellement.
- Lien avec le **Bilan** :
    - Actif : L'usine est immobilisée au **Bilan**. Sa VCN diminue à chaque période.

2.1.2.5. Gamification

Le système de Quêtes agit comme une couche transversale qui vient récompenser les actions des utilisateurs au sein de la plateforme. Il permet de transformer des indicateurs de performance économique en points d'expérience et en progression de compte.

1. Structure de l'entité Quête

L'entité **Quête** permet de stocker les objectifs globaux disponibles dans le jeu.
- Champs Clés :
    - **Nom** / description : Identification de l'objectif.
    - **catégorie** : Type de défi (ex: Économique, Social, Industriel).
    - **target** : Valeur cible à atteindre (ex: produire 1000 unités, atteindre un bénéfice de 50k€).
    - **nb point** : Récompense attribuée lors de la complétion.
- Relations : Relation **Plusieurs à Plusieurs** : Un utilisateur peut se voir attribuer plusieurs quêtes, et inversement, une quête peut être validée par plusieurs utilisateurs.

2. Dynamique de Progression : Quête Utilisateur

Pour suivre l'avancement individuel sans modifier la définition de la quête, une entité de liaison est nécessaire ainsi une fois la quête validée, le champ nb points quête de l'entité Utilisateur est incrémenté de la valeur nb point définie dans la quête.

2.2. Test

2.2.1. Tests

2.2.1.1. Github actions

Un job a été créé pour lancer automatiquement des tests sur les branches main et development. Il est associé à un système de notification (cf. 3.3.4 notification tests).

```yaml
jobs:
  test:
    runs-on: ubuntu-latest
    permissions:
      contents: read
      actions: read

    steps:
    - uses: actions/checkout@v4

    - name: Setup PHP
      uses: shivammathur/setup-php@v2
      with:
        php-version: '8.3'

    - name: Install dependencies
      run: cd Capital-Wars && composer install --no-progress --prefer-dist --optimize-autoloader

    - name: Run tests
      id: phpunit
      continue-on-error: true
      run: |
        cd Capital-Wars
        php bin/phpunit | tee phpunit.log
```

Figure 3: GitHub Actions - Configuration workflow tests (test.yml)

2.2.1.2. Tests

Nous avons réalisé des tests approfondis pour valider le bon fonctionnement de notre application, en utilisant une approche structurée. Cette phase d'assurance qualité a notamment inclus l'utilisation intensive de *mocks* et de *stubs* pour isoler les différentes couches de l'application et simuler des dépendances externes complexes.

Plus précisément, l'utilisation de *mocks* nous a permis de vérifier le comportement de nos composants en simulant les réponses et les interactions avec des services, entity, etc.. sans dépendre de leur disponibilité réelle ou de la complexité de leur configuration. Cette technique a été cruciale pour garantir la fiabilité et la reproductibilité des tests unitaires et d'intégration.

En résumé, la stratégie de test employée, basée sur l'isolation des composants via des *mocks* et l'automatisation des scénarios critiques, confirme la robustesse et la qualité du logiciel développé pour le projet SAE Capital Rivals. (cf. Figure 4)

```php
/**
 * Teste la manipulation complète du tableau 'actif'
 */
public function testManipulationActif(): void
{
    // 1. Set simple
    $this->bilan->setActifValue('stock', 100);
    $this->assertEquals(100, $this->bilan->getActifValue('stock'));

    // 2. Add (incrémentation)
    $this->bilan->addToActif('stock', 50);
    $this->assertEquals(150, $this->bilan->getActifValue('stock')); // 100 + 50

    // 3. SetTo (écrasement)
    $this->bilan->setToActif('stock', 200);
    $this->assertEquals(200, $this->bilan->getActifValue('stock'));

    // 4. Remove
    $this->bilan->removeFromActif('stock');
    $this->assertNull($this->bilan->getActifValue('stock'));
    $this->assertArrayNotHasKey('stock', $this->bilan->getActif());

    // 5. Test d'incrémentation sur une clé inexistante (doit commencer à 0)
    $this->bilan->addToActif('nouveau_champ', 10);
    $this->assertEquals(10, $this->bilan->getActifValue('nouveau_champ'));
}
```

Figure 4: Test unitaire PHPUnit - Manipulation du tableau actif du bilan

2.2.2. SonarQube

Nous avons lié le projet au serveur SonarQube de l'IUT afin d'avoir un meilleur aperçu du projet pour le niveau qualité.

2.2.2.1. Github actions

Nous avons lié le code a sonarQube grâce a un github action qui envoie le code sur le SonarQube de l'IUT. (cf. Figure 5)

```yaml
name: Build SonarQube Analysis

on:
  push:
    branches:
      - main

jobs:
  build:
    name: Build and analyze
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v4
        with:
          fetch-depth: 0

      - name: Setup PHP
        uses: shivammathur/setup-php@v2
        with:
          php-version: '8.3'
          extensions: ctype, iconv, pdo, pdo_mysql
          coverage: xdebug
```

Figure 5: GitHub Actions - Configuration workflow SonarQube (sonar.yml)

2.2.2.2. Coverage

Nous avons essayé de faire le plus grand nombre de tests afin d'avoir le plus grand line coverage possible. (cf. Figure 6)

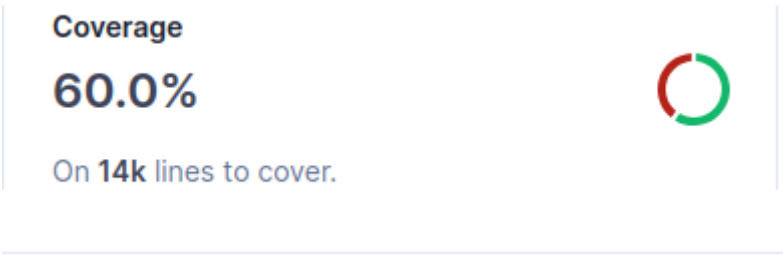


Figure 6: SonarQube - Indicateur de couverture de code (60.0%)

2.3. Réalisation

2.3.1. Connexion / Inscription

La conception de la page d'authentification a été pensée pour être la plus ergonomique possible. L'interface utilise un design bi-colonne qui sépare l'aspect visuel, situé à gauche, des formulaires interactifs situés à droite.

2.3.1.1. Interface de connexion

Une page de connexion a été réalisée cette page permet à l'utilisateur de se connecter, avec son email et un mot de passe. Et si l'utilisateur a oublié son mot de passe il peut utiliser le lien de navigation de secours vers la récupération de mot de passe (cf 2.3.9).

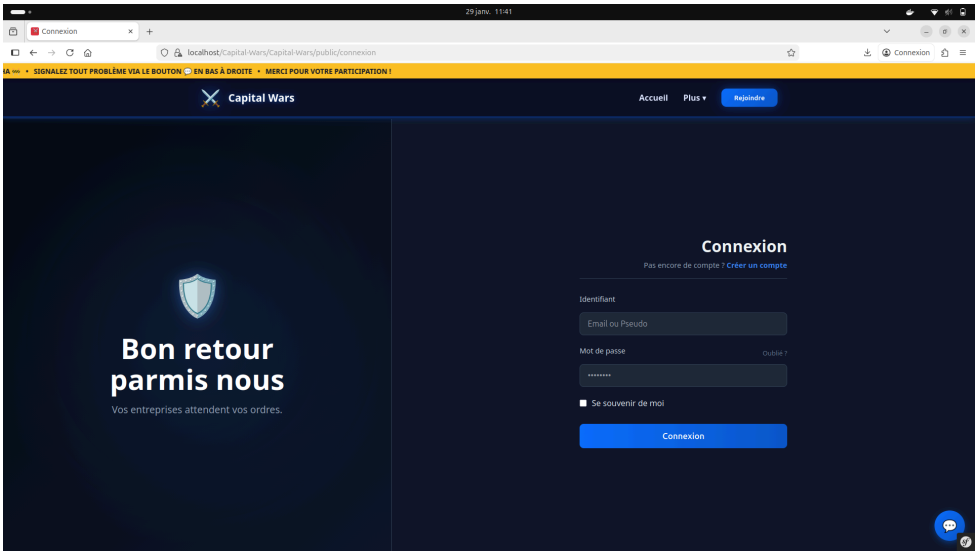


Figure 7 : Page de connexion

2.3.1.2. Le processus des inscriptions et l'onboarding

La page d'inscription a été conçue pour être la plus ergonomique possible. Elle intègre la collecte et la validation des champs de saisie (nom, prénom, e-mail) avec une vérification en temps réel de la conformité des données et de la correspondance des mots de passe. De plus, un utilisateur peut télécharger une photo de profil pour définir leur avatar, avec des contraintes techniques précises (format JPG/PNG, limite de 2 Mo). Nous avons aussi ajouté un bouton pour créer un « compte organisateur ».

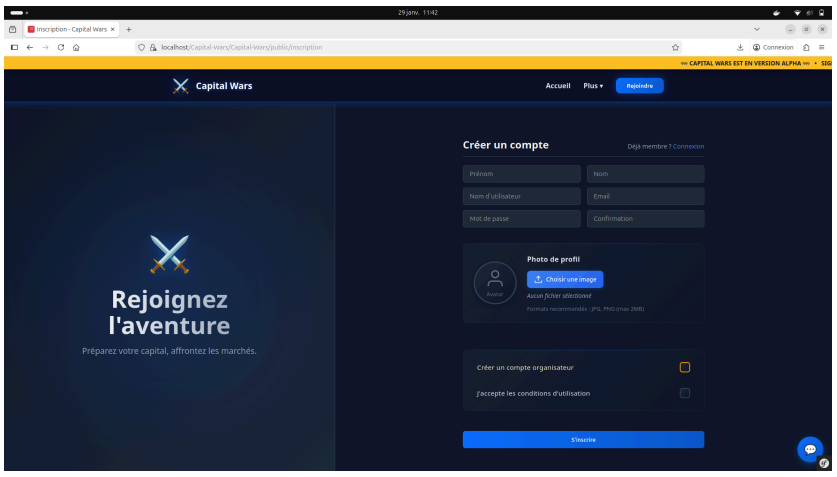


Figure 8 : Page d'inscription

2.3.2. Dashboard utilisateur

Une fois l'authentification réussie, l'utilisateur est dirigé vers un tableau de bord centralisé.

2.3.2.1. Barre latéral de gauche

La barre latérale persistante, située à gauche de l'écran, regroupe plusieurs éléments : un bloc « Profil Joueur » affichant l'identifiant et l'adresse e-mail de l'utilisateur, ainsi que les options permettant de créer ou de rejoindre une partie et d'accéder aux quêtes.

2.3.2.2. Gestion des sessions de jeu

Le cœur du dashboard est dédié à la gestion des parties. Le système distingue les sessions dont l'utilisateur est l'auteur de celles auxquelles il participe simplement, utilisant des icônes (couronne pour les créations, manette pour les participations). Chaque partie est représentée par une carte affichant les informations : code d'accès, période actuelle du jeu et taux de remplissage des joueurs (ex: 1/25). Un système de badges colorés permet de connaître instantanément le statut d'une partie, différenciant par exemple une "Période en cours" d'une "Attente d'une période". Pour les parties créées, l'organisateur dispose de boutons d'action rapide pour gérer, modifier ou supprimer la session.

2.3.2.3. Widgets latéraux et utilitaires

À droite de l'interface, des widgets complémentaires fournissent des données en temps réel grâce à des requêtes ajax. Ce qui permet de suivre en temps réel le décompte de la session en cours, renforçant l'aspect dynamique de la simulation.

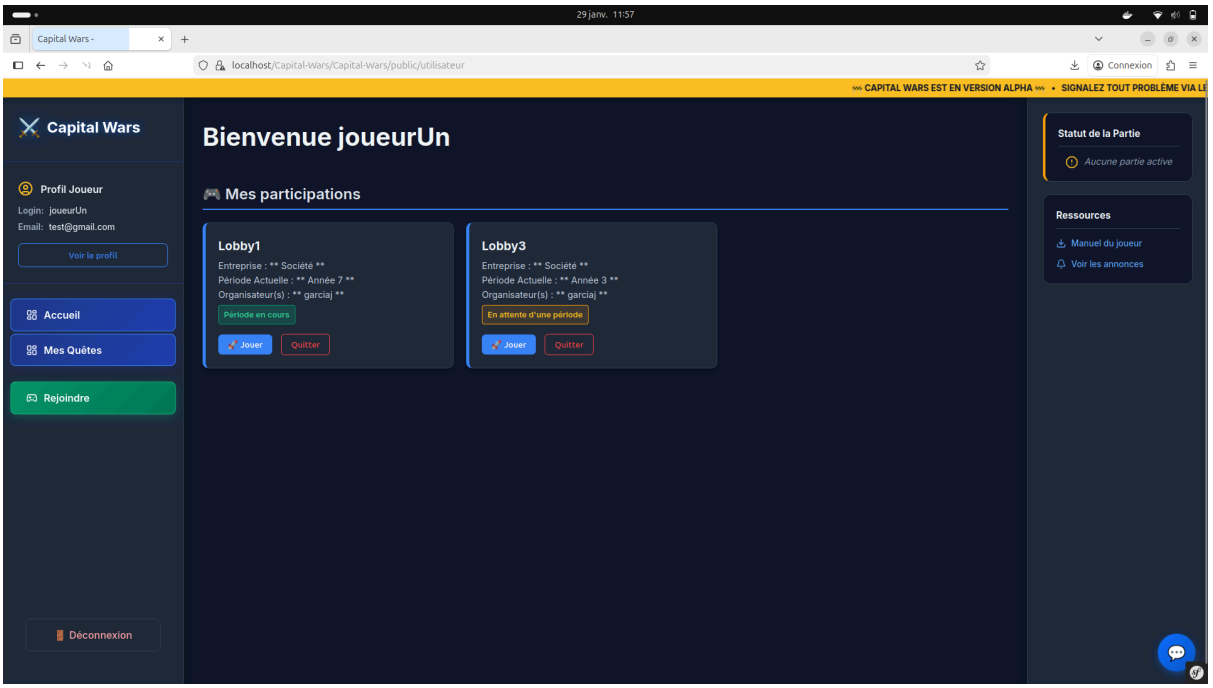


Figure 9 : Dashboard utilisateur

2.3.3. Rejoindre et Créer un lobby

Le système de lobbys constitue le pivot central du projet. Cette interface a été conçue pour offrir un parcours utilisateur intuitif, permettant soit de rejoindre une simulation existante via un système de code, soit de paramétrer intégralement une nouvelle session de jeu.

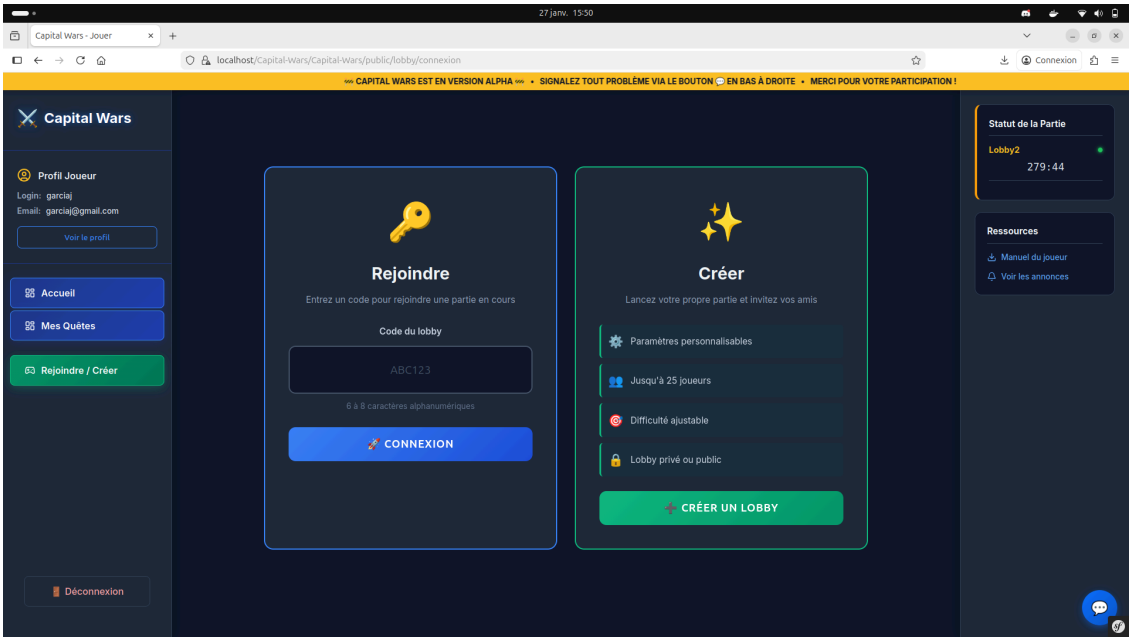


Figure 10 : Rejoindre ou créer un lobby

2.3.3.1. Configuration de la partie

Lorsqu'un organisateur choisit de créer un lobby, il accède à une interface de configuration. Cette page est le moteur de personnalisation de la simulation et permet à l'organisateur de définir le nom du lobby, de générer un code d'accès unique et de configurer la répartition des joueurs par entreprise (de 4 à 6 entreprises avec un nombre de joueurs ajustable). Ensuite l'organisateur peut régler des variables telles que la capacité du marché, le taux de croissance ou les paramètres fiscaux (impôts, emprunts). Ces données influencent directement le niveau de difficulté de la simulation. Et enfin une rangée de cases à cocher permet d'activer ou de désactiver les plugins (Bourse, Événements, etc.) cette partie est complètement modulaire et regarde en temps réel le dossier plugins et affiche les plugins disponibles.

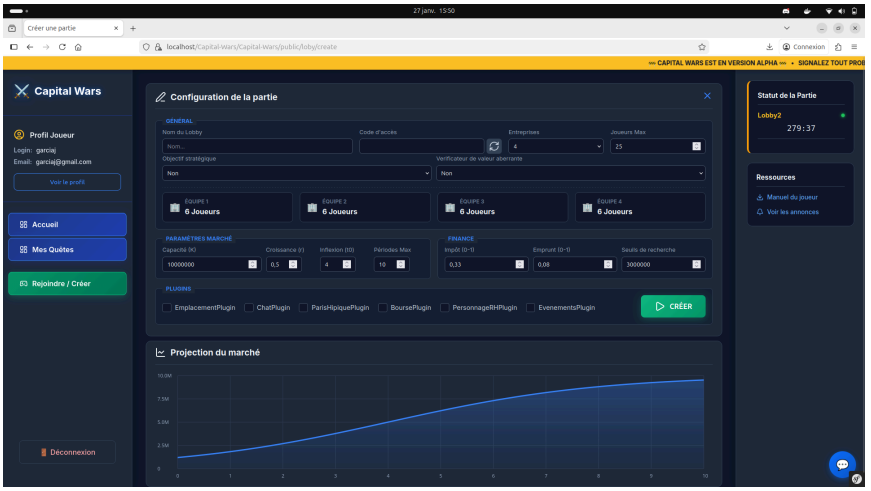


Figure 11 : Création d'un lobby

2.3.3.2. Projection du marché

À l'instar du jeu EXIGE, une projection du marché a été effectuée, soulignant son évolution au fil des périodes. L'organisateur a la possibilité de modifier et d'influencer la courbe de marché en ajustant quatre variables clés. (cf. Figure 7)

1. **Capacité Maximale du Marché (K - *marcheCapaciteMax*)** : Définit la quantité maximale théorique du marché (ex. : 1 000 000 d'unités par défaut, configurable).
2. **Croissance (r - *marcheCroissance*)** : Détermine le taux de croissance et, par conséquent, l'inclinaison de la courbe (ex. : 0,5 par défaut).
3. **Période d'Inflexion (t0 - *marcheInflection*)** : Indique la période où la croissance du marché atteint son pic et augmente le plus fortement (ex. : période 4 par défaut).
4. **Période Maximale** : Ajuste la durée totale (longueur) de la courbe de marché.

La quantité de marché à une période donnée (t) est calculée selon la formule logistique suivante :

$$Q(t) = \frac{K}{1 + e^{-r(t - t_0)}}$$

Figure 12 : Formule mathématique logistique - Calcul capacité du marché

Où t représente l'indice de la période actuelle (commençant à 1).

2.3.4. Administration d'un lobby

L'interface d'administration constitue le centre de contrôle pour l'organisateur de la partie. Une fois le lobby créé, cette page permet de piloter le déroulement de la partie en temps réel. Elle a été conçue pour offrir une visibilité sur l'état de la session et sur les interactions des participants, tout en regroupant les commandes de gestion critiques dans un espace structuré et réactif.

2.3.4.1. Pilotage de la simulation

Le bloc principal de gestion permet à l'organisateur de suivre l'avancement temporel du jeu. L'objectif est de fournir un état des lieux instantané sans avoir à naviguer dans des sous-menus complexes. Il permet un affichage clair de la période actuelle (ex: "Année 1"), du nombre maximum de périodes prévues et du nombre d'entreprises en compétition. Un bouton d'action principal permet aussi de générer une nouvelle période, déclenchant ainsi le calcul des résultats économiques et le passage à l'étape suivante pour tous les joueurs. Des raccourcis permettent aussi de consulter la liste des entreprises, l'historique complet des périodes passées ou de valider les demandes de prêt financier en attente.

2.3.4.2. Gestion des participants

À droite de l'interface, une section dédiée à la surveillance des joueurs du lobby a été créée pour un affichage en temps réel des utilisateurs connectés avec leur pseudonyme et leur adresse e-mail. L'organisateur dispose alors de la capacité d'exclure un participant via un bouton d'action dédiée, permettant une gestion rapide des erreurs d'inscription ou des comportements inappropriés. Aussi un indicateur visuel (ex: 1/25) permet de vérifier si tous les joueurs attendus ont bien rejoint la session avant de lancer la simulation.

2.3.4.3. Administration des modules (Plugins)

La partie inférieure de la page est consacrée à la gestion des plugins. Chaque module (Emplacement, Chat, Paris Hippiques, etc.) possède un badge indiquant s'il est actuellement actif ou inactif pour ce lobby. Des boutons de configuration individuels permettent d'accéder aux réglages spécifiques de chaque plugin sans quitter la page principale, assurant ainsi une administration fluide et centralisée. Chacun des plugins est récupéré comme lors de la création du lobby scannant le dossier plugins.

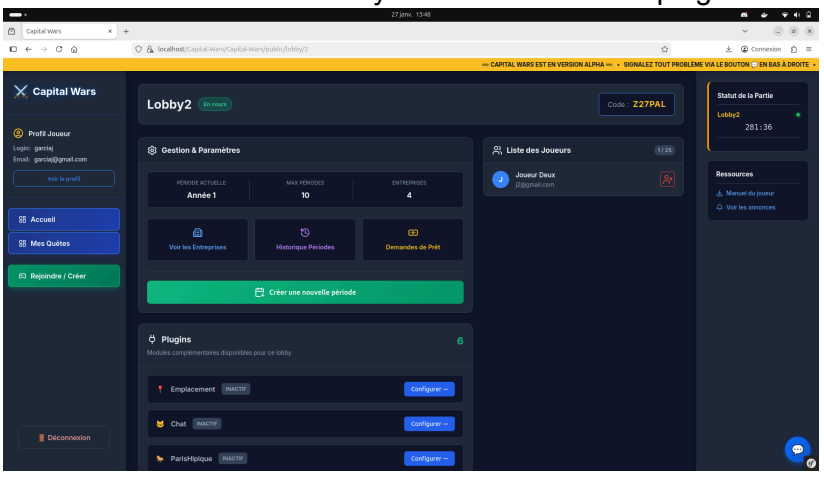


Figure 13 : Administration d'un lobby

2.3.5. Entreprise

L'interface Entreprises constitue le hub de répartition des participants au sein du lobby. Cette page a été conçue pour offrir une vision globale de la structure compétitive de la partie, tout en adaptant les interactions disponibles selon le rôle de l'utilisateur (Organisateur ou Joueur). L'ergonomie repose sur un système de cartes interactives et une légende de couleurs intuitive pour faciliter la compréhension immédiate de l'état du marché.

2.3.5.1. Interface de gestion pour l'organisateur

Pour le créateur du lobby, cette page sert d'outil de supervision et de configuration structurelle de la simulation. Il possède un contrôle total sur la composition des forces en présence, la section Actions Organisateur permet ainsi d'ajouter manuellement de nouvelles entreprises à la simulation si le besoin s'en fait sentir. Un panneau latéral Informations du Lobby affiche aussi en temps réel le nombre de joueurs connectés, les places disponibles et le décompte des entreprises affichant complet. Chaque carte d'entreprise indique le ratio actuel de joueurs (ex: 1/7) et permet à l'organisateur de visualiser la répartition des forces avant de valider le lancement de la période.

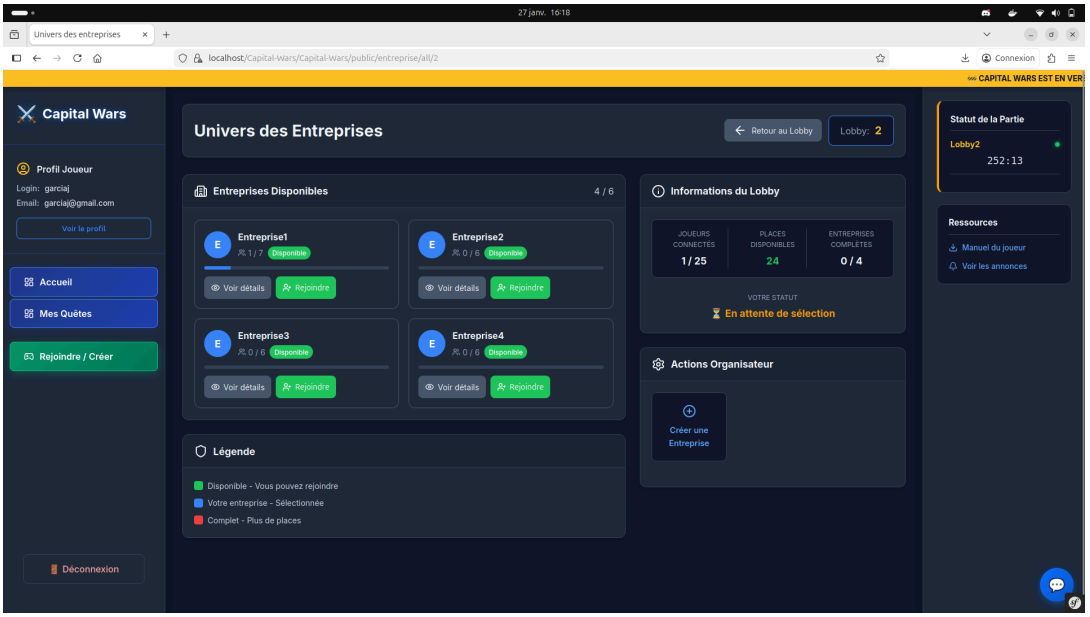


Figure 14 : Interface liste des entreprise (Organisateur)

2.3.5.2. Interface d'intéraction pour le joueur

Du côté du participant, l'interface est orientée vers l'action dans une équipe. Le parcours utilisateur est balisé pour garantir une intégration fluide dans la simulation. Le joueur peut parcourir les entreprises disponibles, consulter leurs détails ou cliquer sur "Rejoindre" pour intégrer une équipe, sous réserve de places libres. Un système de codes couleurs permet au joueur d'identifier son appartenance (Bleu pour son entreprise sélectionnée) et les opportunités restantes (Vert pour les places disponibles). Et enfin une fois une entreprise rejointe et qu'une période est lancée, une section "Jeu Actif" apparaît dynamiquement. Il permet au joueur d'accéder directement à son panel de gestion économique pour commencer à soumettre ses décisions.

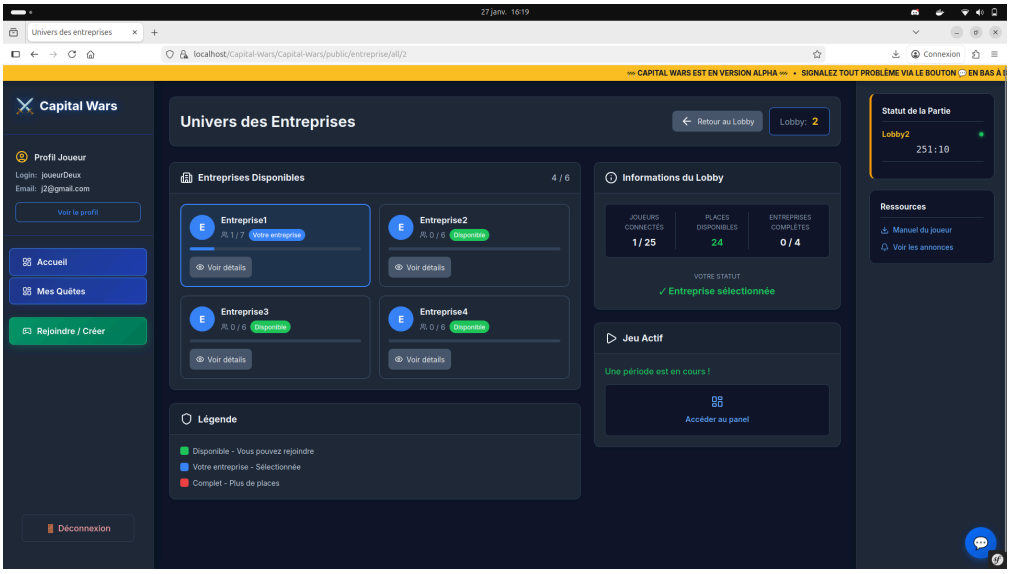


Figure 15 : Interface liste des entreprises (Joueur)

2.3.6. Période de jeu

Le jeu repose sur un découpage temporel par périodes, simulant des exercices comptables ou annuels comme pour EXIGE. Cette section du rapport présente les interfaces permettant de piloter l'avancement chronologique du lobby, d'orchestrer le lancement des nouveaux cycles et d'analyser les décisions prises par les entreprises avant le calcul des résultats finaux.

2.3.6.1. L'historique des périodes

L'interface de l'historique offre une vue rétrospective et une frise chronologique de la partie. Elle permet à l'organisateur de suivre l'évolution de la session de manière ordonnée. Chaque période est représentée par une carte indiquant la date et l'heure de création, ainsi que son statut actuel (ex: "En cours" ou "Inactive"). Pour les périodes passées, le système affiche précisément l'heure de fin, permettant une traçabilité complète du rythme de jeu. Cette page sert de pivot pour accéder aux détails spécifiques de chaque cycle ou pour initier le passage à l'étape suivante via le bouton de création. (cf. Figure 8)

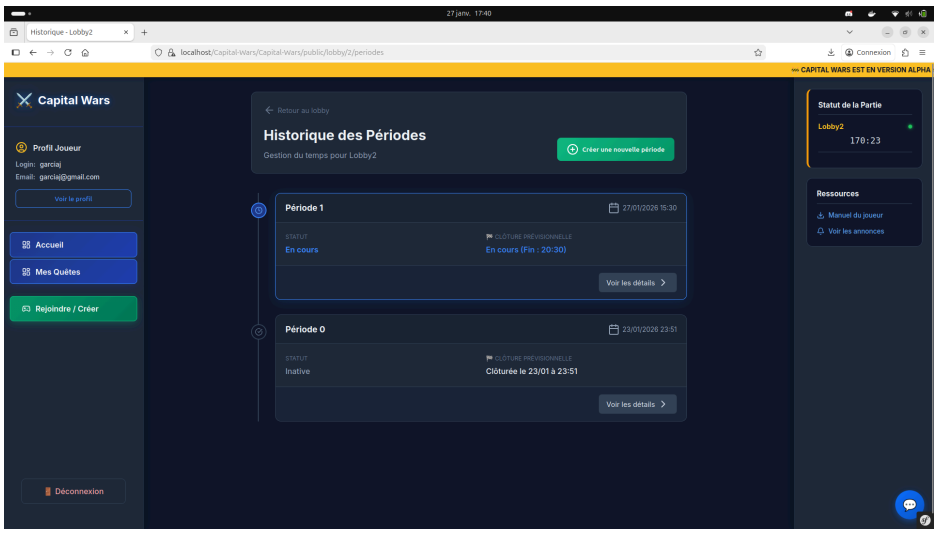


Figure 16 : Interface historique des périodes de jeu

2.3.6.2. Lancement d'une nouvelle période

Pour lancer une période, l'organisation doit établirent deux paramètres :

1. **Le temps alloué** : Le temps (en minutes) dont disposent les joueurs pour soumettre leurs décisions. Ce paramètre déclenche un compte à rebours affiché en temps réel dans le lobby.
2. **Le coefficient d'importance de la publicité** : Ce coefficient est utilisé pour ajuster l'influence de la publicité pour la période en cours. Ce réglage modifie de manière dynamique l'impact sur les ventes, en interférant avec leur calcul.

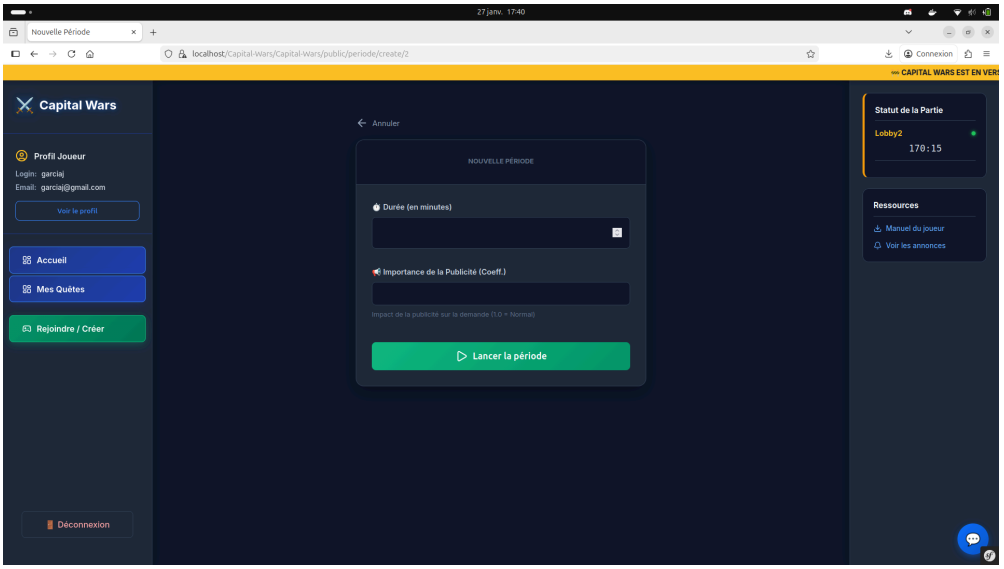


Figure 17 : Interface création d'une période

2.3.6.3. Monitoring des décisions et validation

Une fois la période lancée, l'organisateur accède à une page de détails qui agit comme un tableau de bord de surveillance en temps réel. Cette interface est cruciale pour afficher les données des joueurs en temps réel. Ainsi une barre de progression visuelle indique en temps réel le ratio d'entreprises ayant validé leurs décisions (ex: "4 / 4 entreprises - 100% validé"). Des cartes détaillées affichant les choix stratégiques de chaque équipe (Prix de vente, Budget Pub, etc). Cela permet à l'organisateur de détecter d'éventuelles valeurs aberrantes avant le calcul. Et ainsi une fois toutes les décisions rendues, l'organisateur dispose du bouton "Calculer Résultats". Cette action déclenche l'algorithmes de calcul de résultat qui transformeront ces entrées en données financières (chiffre d'affaires, part de marché, bénéfices) pour clore la période.

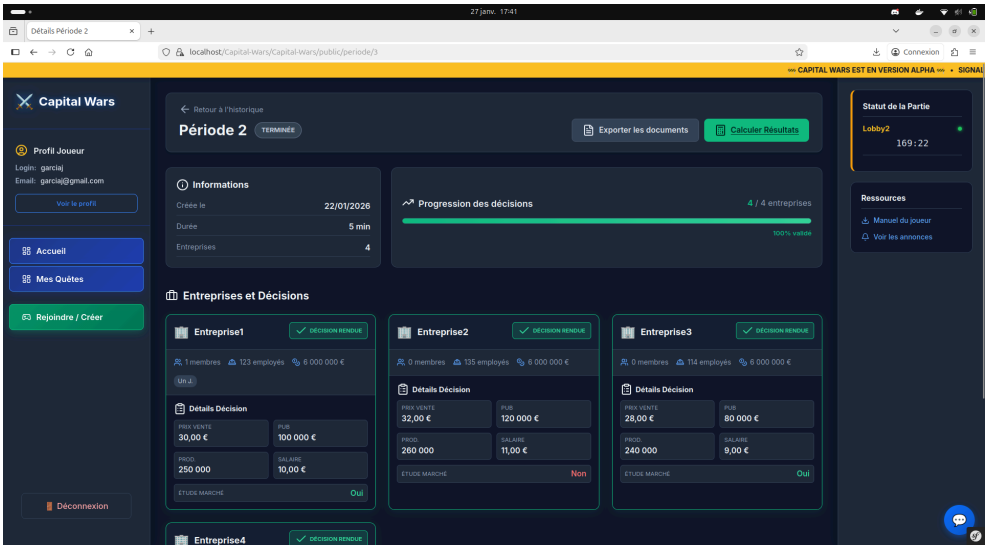


Figure 18 : Interface monitoring d'une période

2.3.7. Le panel de jeu

Le panel de jeu représente le cœur opérationnel pour l'utilisateur. C'est ici qu'il devrait insister et prendre des décisions. L'interface a été conçue pour centraliser tous les outils de gestion, de la saisie des ordres à l'analyse prévisionnelle, afin d'aider les joueurs à piloter leur entreprise de manière structurée.

2.3.7.1. Le panneau de contrôle

Avant d'accéder aux formulaires de saisie, le joueur arrive sur le Panneau de Contrôle. Ce hub centralise l'accès aux différentes dimensions de la gestion d'entreprise :

- **Indicateurs de soumission :** Des badges colorés ("Non soumis" / "Soumis") permettent au joueur de savoir instantanément quels documents il n'a pas encore remplis.
- **Accès aux outils :** Le panneau offre des raccourcis vers la feuille de décision, le plan de trésorerie, ainsi qu'un accès aux documents financiers des périodes précédentes (N−1) pour l'analyse comparative.
- **Accéder au prêt :** Ce panneau permet aussi au utilisateur d'accéder au demande et prêt en cours.
- **Accès aux plugins :** Ce panneau permet à l'utilisateur d'accéder au catalogue des plugins et d'effectuer les actions associées à ceux-ci.

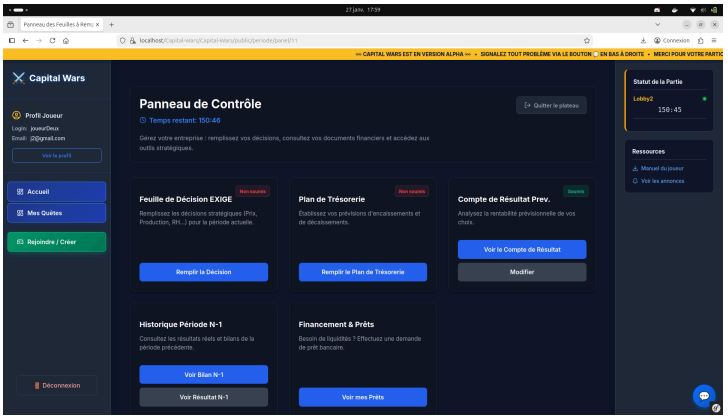


Figure 19 : Panneau de contrôle utilisateur

2.3.7.2. La feuille de décision EXIGE

La saisie des décisions stratégiques est organisée par section, reflétant les différents départements d'une entreprise réelle :

- **Gestion Usine et Production :** Les joueurs gèrent leurs capacités industrielles (achat/vente d'usines) et définissent le volume de production à lancer ainsi que le budget Recherche & Développement (R&D).
- **Gestion Commerciale :** Cette section permet de fixer le prix de vente unitaire, le budget publicitaire et de choisir d'activer ou non une étude de marché pour affiner la stratégie.
- **Gestion du Personnel :** Les décisions incluent les politiques salariales et les effectifs, influençant directement la productivité et le climat social de l'entreprise virtuelle.

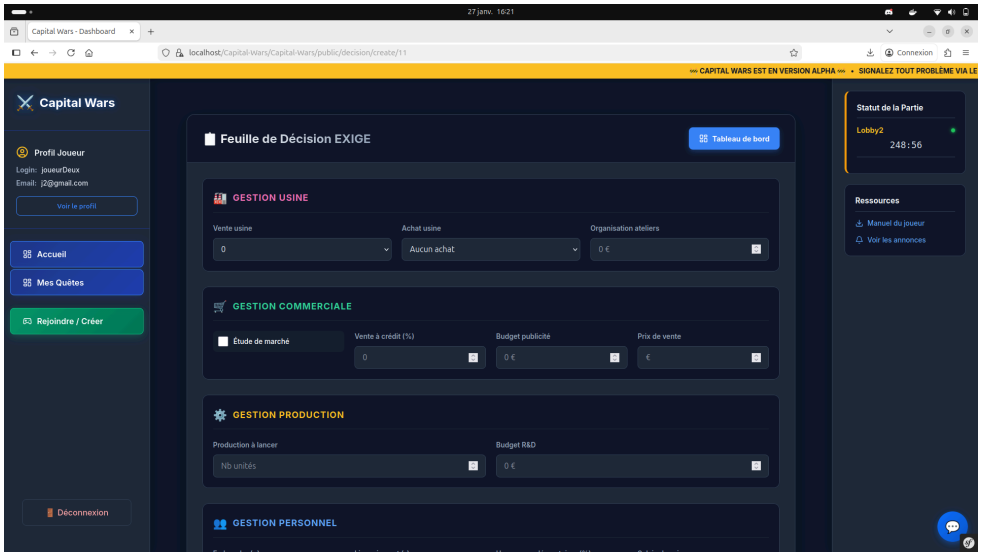


Figure 20 : Feuille de décision Exige

2.3.7.3. Outils de prévision et analyse

Pour limiter les risques d'erreurs stratégiques, le joueur dispose d'outils de simulation financière avant la validation finale :

- **Plan de Trésorerie Prévisionnel :** Une interface dédiée permet de lister les encaissements et décaissements prévus (achats de matières, charges de personnel, remboursements d'emprunts) afin d'anticiper le solde bancaire final.
- **Compte de Résultat Prévisionnel :** Ce compte de résultat prévisionnel permet à l'utilisateur de projeter la rentabilité de la période. Il détaille les produits d'exploitation face aux charges pour estimer le bénéfice ou la perte à venir.
- **Bilan, Compte de résultat, décision N-1 :** Accessible à tout moment offrant une base solide pour les décisions futures.

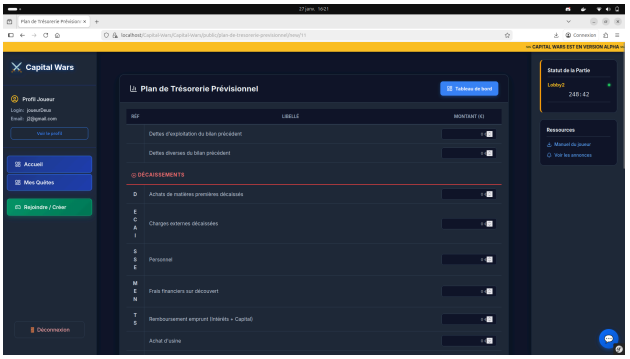


Figure 21 : Plan de trésorerie prévisionnel

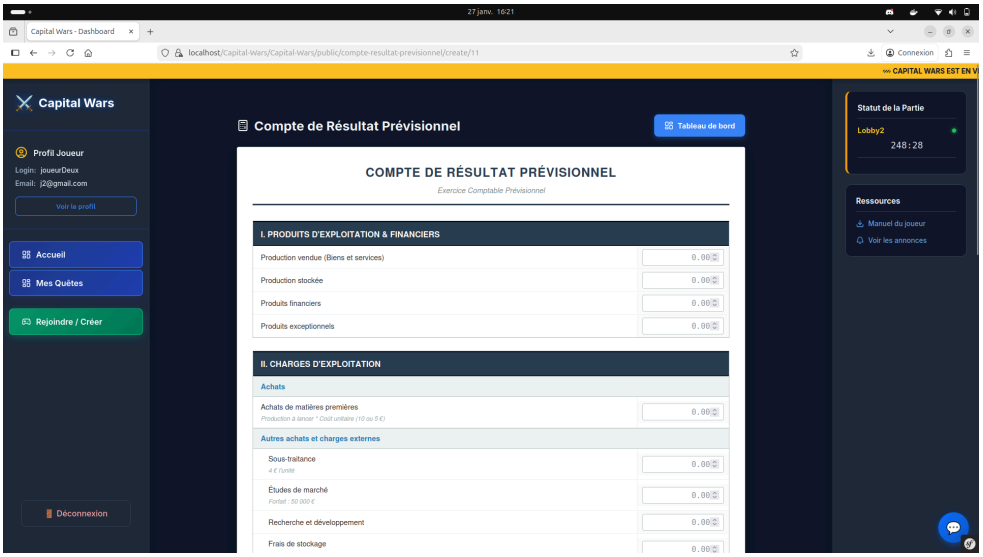


Figure 22 : Compte de résultat prévisionnel

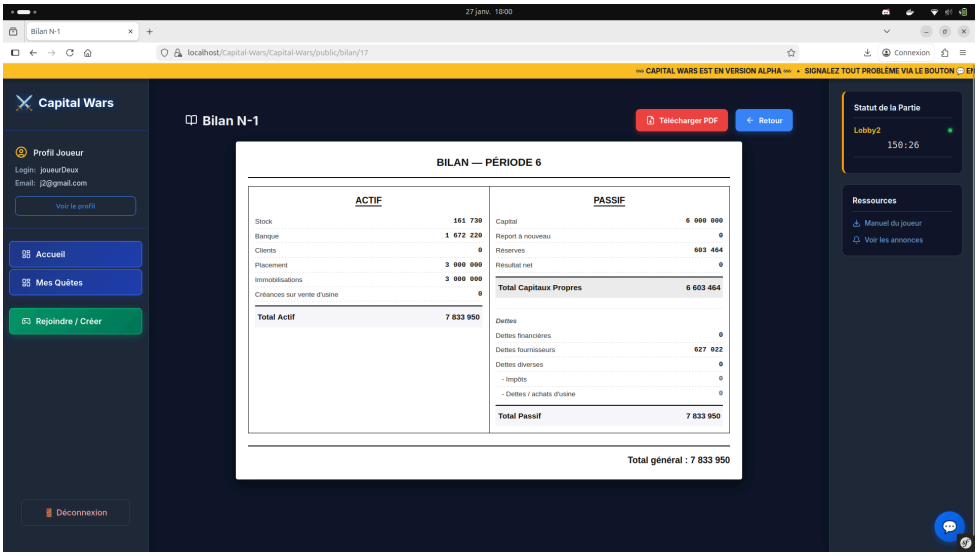


Figure 23 : Bilan N-1

2.3.7.4. Détection de valeurs aberrantes

Afin de prévenir les erreurs de saisie, un module de vérification intelligent a été intégré au processus de validation des décisions. Ce système analyse les données soumises par l'utilisateur et les compare en temps réel aux moyennes du marché pour identifier toute incohérence potentielle. (*Cette détection est activable ou non lors de la création du lobby)*.

- **Analyse Statistique :** Lors de la validation d'une feuille de décision, l'algorithme examine chaque champ pour détecter des écarts significatifs par rapport à la tendance du lobby.
- **Seuils d'Alerte :** Une valeur est jugée "aberrante" si elle dépasse deux fois la moyenne du marché, notamment pour des facteurs critiques tels que le prix de vente ou le budget alloué à la publicité.
- **Notification et Confirmation :**
    - Si une valeur anormale est détectée (par exemple, un prix ou un budget publicitaire démesuré), une icône d'avertissement s'affiche sur l'interface.

- Au moment de la validation finale, une fenêtre modale apparaît pour informer l'utilisateur de l'anomalie et exiger une confirmation de sa part.

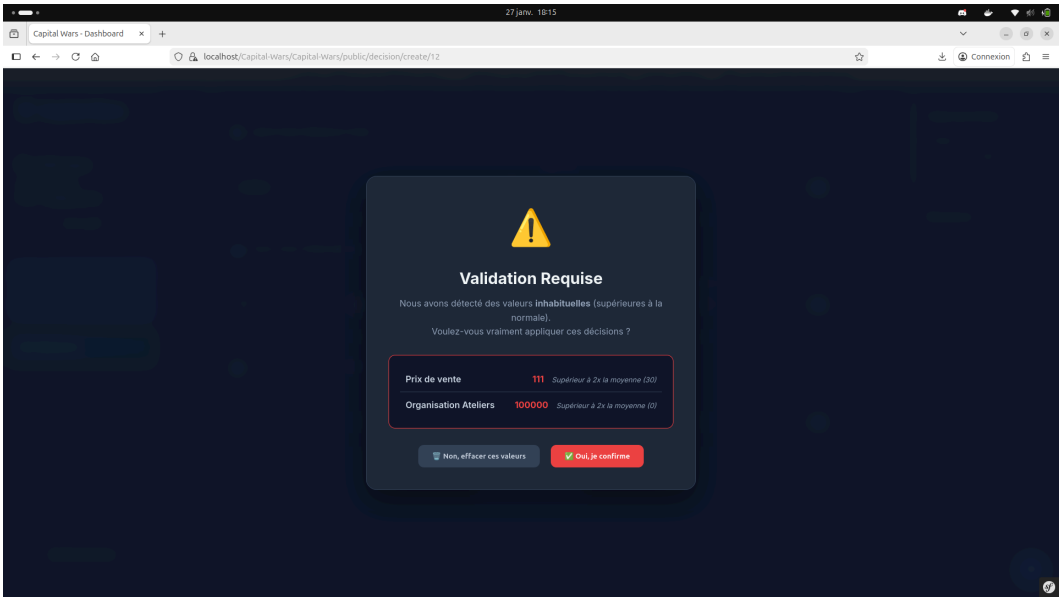


Figure 24 : Message détection de valeurs aberrantes

2.3.7.5. Étude du marché

L'étude de marché sert d'outil d'espionnage légal pour ajuster sa stratégie entre deux tours. Elle centralise les indicateurs clés du secteur, comme la demande totale ou le prix moyen, et propose un tableau comparatif pour observer les budgets pub et les parts de marché des concurrents. Un rapport PDF est également générable pour garder une trace de ces données et faciliter l'analyse hors ligne.

Figure 25 : Bouton accès à l'étude du marché

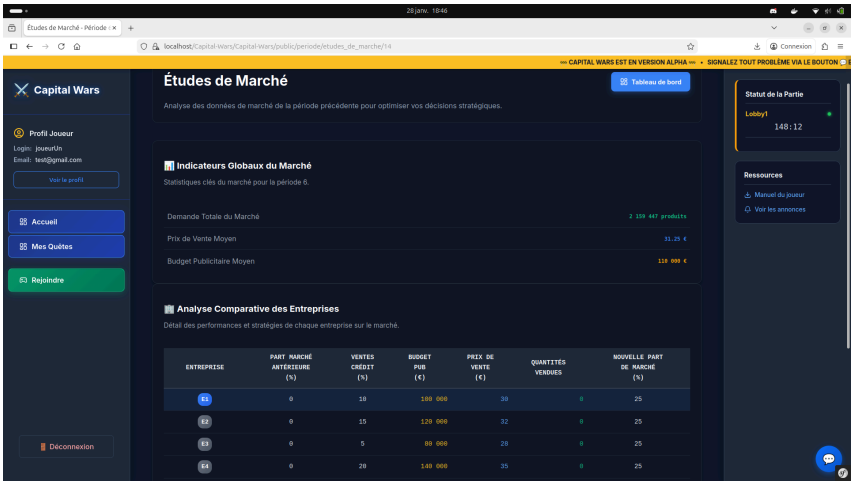
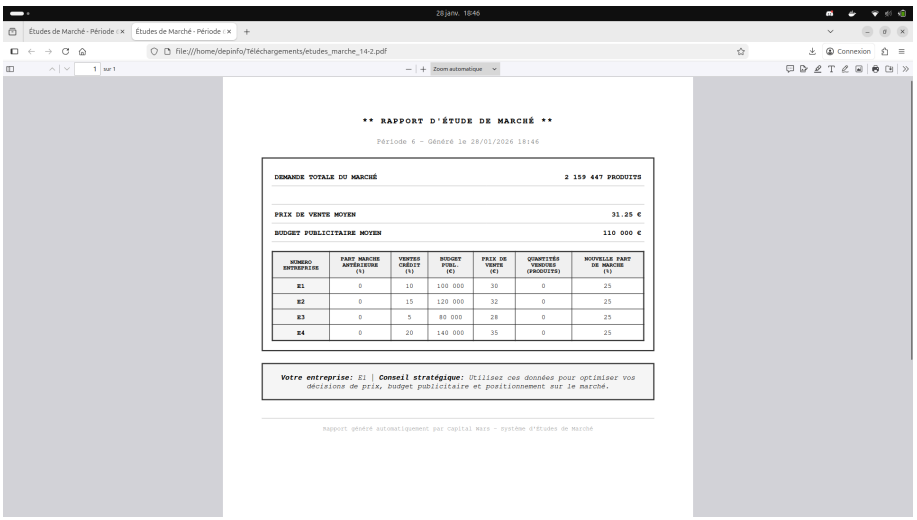


Figure 26 : Interface étude du marché



** RAPPORT D'ÉTUDE DE MARCHÉ **

Période 6 - Généré le 28/01/2026 18:46

DEMANDE TOTALE DU MARCHÉ 2 159 447 PRODUITS

PRIX DE VENTE MOYEN 31.25 €

BUDGET PUBLICITAIRE MOYEN 110 000 €

NUMERO ENTREPRISE	PART MARCHE ANTÉRIEURE (%)	VENTES CRÉDIT (%)	BUDGET PUBL. (€)	PRIX DE VENTE (€)	QUANTITÉS VENDUES (PRODUITS)	NOUVELLE PART DE MARCHE (%)
E1	0	10	100 000	30	0	25
E2	0	15	120 000	32	0	25
E3	0	5	80 000	28	0	25
E4	0	20	140 000	35	0	25

Votre entreprise: E1 | Conseil stratégique: Utilisez ces données pour optimiser vos décisions de prix, budget publicitaire et positionnement sur le marché.

Rapport généré automatiquement par Capital Wars - Système d'Études de Marché

Figure 27 : Document pdf étude du marché

2.3.8. Objectif Stratégique

Un système d'objectifs stratégiques, similaire à celui mis en place dans EXIGE, a été intégré au projet. Il prend la forme de quêtes à accomplir. Ces objectifs permettent aux organisateurs de visualiser la planification stratégique de chaque groupe pour sa partie. Actuellement, l'impact de ce système est limité, mais il est prévu pour le semestre prochain que ces objectifs stratégiques soient calculés et validés (ou non) automatiquement, offrant ainsi des bonus de points pour les quêtes. Cette fonctionnalité est optionnelle et activable ou non lors de la création du lobby.

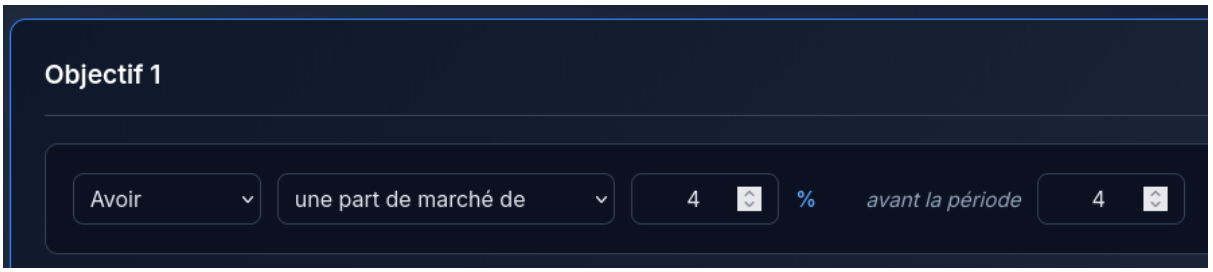


Figure 28 : Formulaire création d'objectifs stratégiques

Afin d'assurer la génération d'objectifs stratégiques sans erreur, un formulaire intelligent a été développé. Ce formulaire empêche les erreurs de syntaxe ou de formulation (par exemple, "réaliser une part de marche de"), en ne proposant à l'utilisateur que des phrases correctement formulées en français.

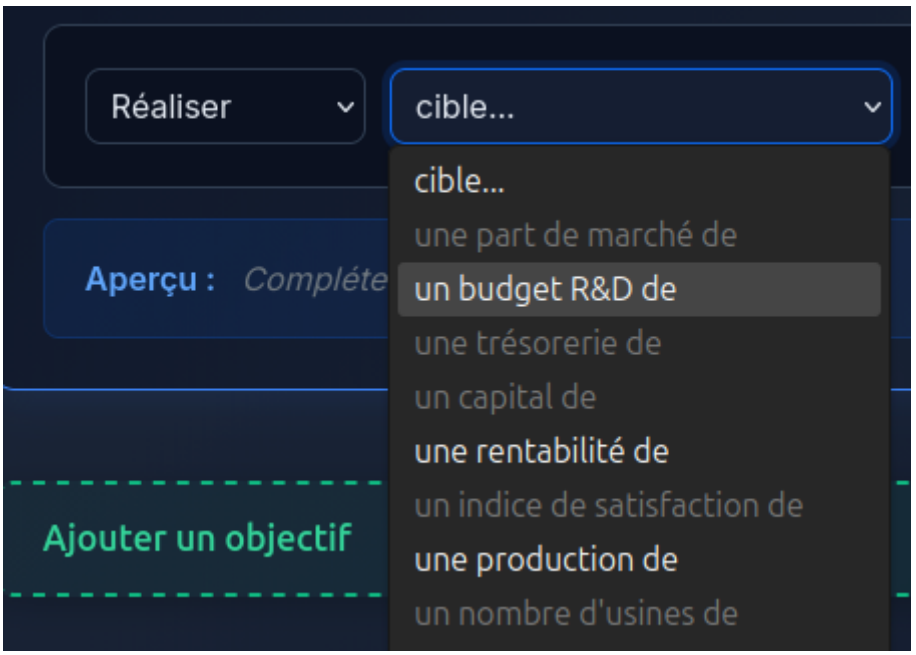


Figure 29: Menu déroulant - Définition d'objectifs

2.3.9. Prêt

Un système de prêt a été mis en place. Ce système fonctionne de manière similaire à celui de notre monde, où un lobby fixe un taux d'intérêt, permettant de calculer les annuités de remboursement. Processus de validation d'un prêt :

- **Demande de prêt** : Une entreprise soumet une demande de prêt.
- **Décision** : L'organisateur de la partie reçoit la demande et décide de l'accepter ou non.
- **Réalisation du prêt** : Si la demande est acceptée, le prêt est accordé. L'entreprise reçoit le montant demandé sur son compte bancaire et doit commencer à rembourser le prêt par le biais d'annuités.

2.3.10. Reset Mot de passe

Un système de réinitialisation de mot de passe a été mis en place. Lorsqu'un utilisateur oublie son mot de passe, il peut demander à recevoir un e-mail lui permettant de le modifier. Ce mécanisme repose sur le serveur SMTP de Google, qui est utilisé pour envoyer l'e-mail via une adresse Gmail dédiée.

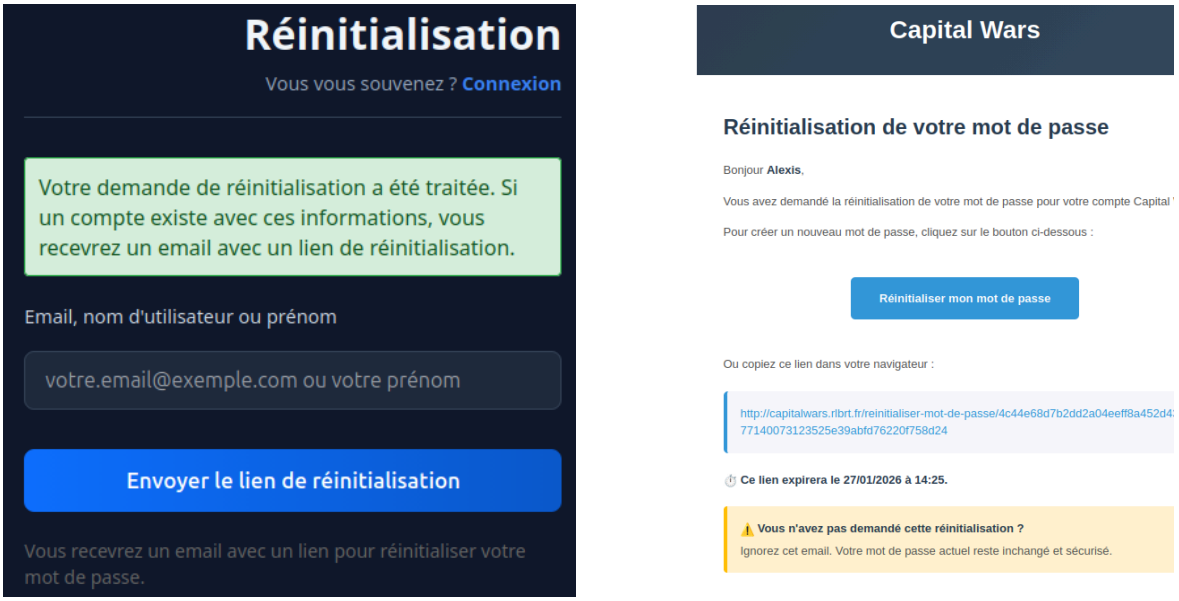


Figure 31: Interface demande de réinitialisation de mot de passe

2.3.11. Quêtes

Un système de quêtes a été réalisé, ce système de quêtes n'a pas d'impact réel actuellement. Mais au prochain semestre tout l'aspect gamification sera implémenté. Le système de quête fonctionne néanmoins de manière simple. Dans un commande des quêtes sont créer, chacune de ces quête possède plusieurs attributs qui peuvent être très facilement modifié. Les attributs importants sont le nombre de points que rapporte cette quête.

```
['nom' => 'Un debut a tous', 'points' => 10, 'description' => 'Rejoindre
pour la première fois un lobby', 'category' => self::CATEGORY_DEBUT,
'target' => 50 ],
```

2.3.11.1. Validation des quêtes

La validation des quêtes a été conçue pour être la plus modulaire et la plus simple possible à implémenter. (cf. Figure 12)

- **Quêtes simples :** Une seule ligne de code est nécessaire pour valider la quête de l'utilisateur. Le développeur n'a qu'à insérer cette ligne (avec le nom de la quête) au point approprié de la logique du jeu.
- **Quêtes à paliers (ex: "faire 10 parties") :** Un compteur est intégré. À chaque appel de la ligne de validation, le compteur est incrémenté. La quête n'est validée automatiquement que lorsque l'utilisateur atteint le nombre de paliers requis (par exemple, 10 parties).

```
$result = $this->queteValidationService->validateQuest('Un debut a tous', $user);
```

Figure 31: Exemple de code validation de quête

2.3.11.2. Affichage général

Cette page présente l'ensemble des quêtes, organisées par catégorie. L'utilisateur peut ainsi identifier facilement les quêtes qu'il souhaite accomplir. (cf. Figure 13)

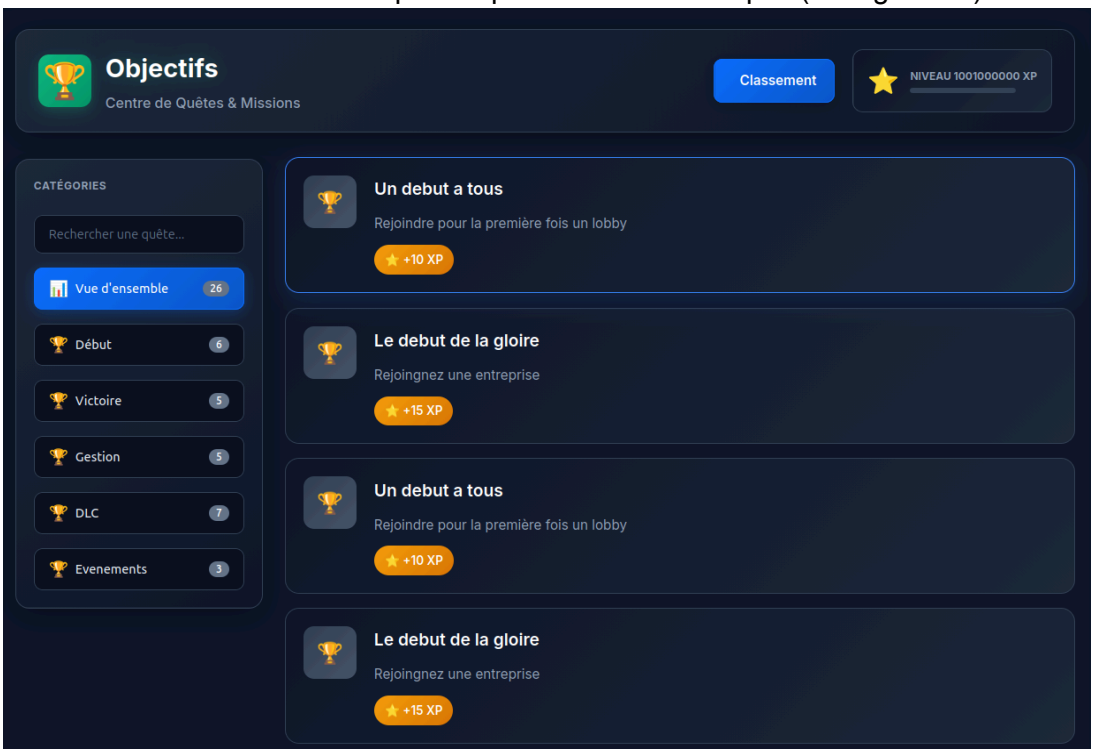


Figure 32: Centre de quêtes et missions avec filtres par catégories

2.3.11.3. Classement global

Un classement global a également été mis en place pour stimuler la compétition entre les joueurs. Ce **leaderboard** est un élément central de l'expérience Capital Wars, offrant une visibilité en temps réel sur les performances de chacun. (cf. Figure 14)

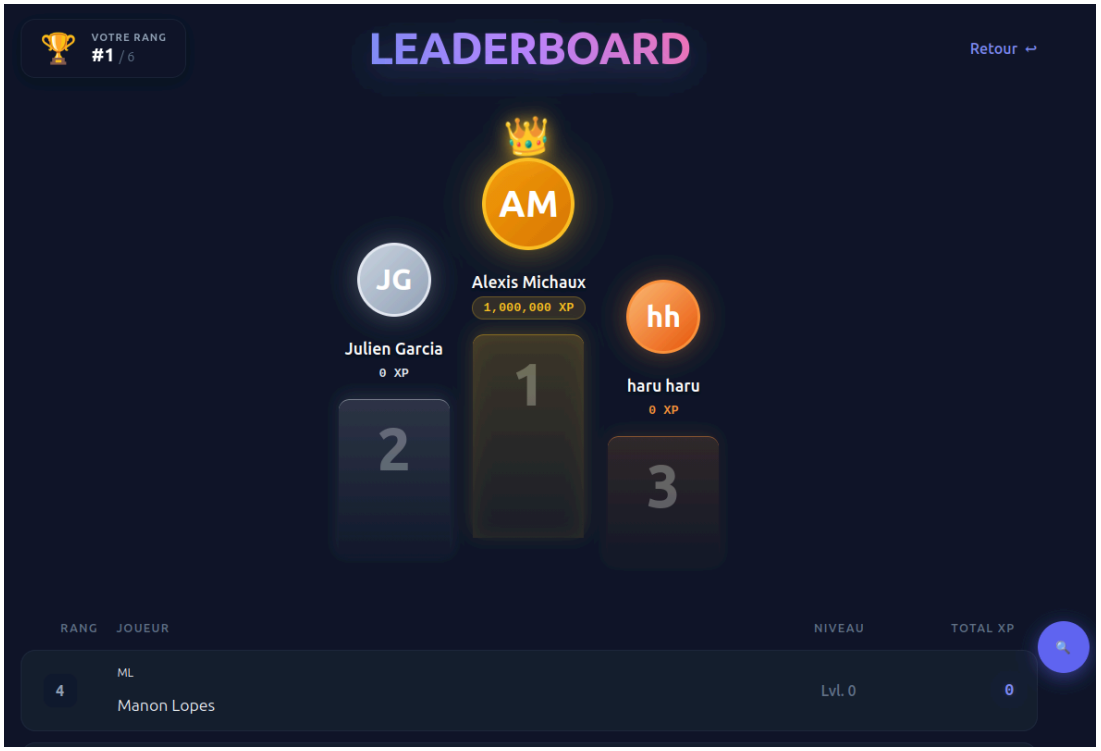


Figure 33: Leaderboard avec podium et tableau des scores

2.3.12. Page d'erreur

La robustesse d'une plateforme web ne se mesure pas seulement à ses fonctionnalités nominales, mais aussi à sa capacité à gérer les incidents de manière informative. Nous avons donc porté notre une attention particulière au design des pages d'erreurs (HTTP 403, 404 et 500) afin d'éviter la frustration de l'utilisateur et de le guider vers une solution de sortie.

2.3.12.1. Erreur 404 : Page non trouvée

L'erreur 404 intervient lorsqu'un utilisateur tente d'accéder à une URL inexistante ou déplacée. L'objectif est de transformer une impasse en une opportunité de navigation, ainsi deux boutons permettent soit de retourner directement à l'accueil du jeu, soit de revenir à la page précédente, garantissant que l'utilisateur ne reste jamais bloqué. (cf. Figure 15)

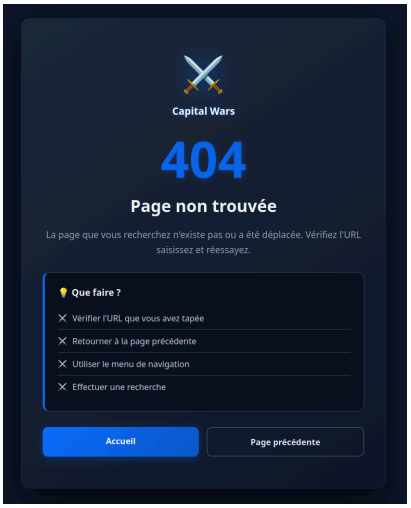


Figure 34: Page d'erreur 404 - Page non trouvée

2.3.12.2. Erreur 403 : Accès interdit

Cette page est un pilier de la sécurité du site. Elle s'affiche lorsqu'un utilisateur tente d'accéder à une ressource (comme un lobby privé ou un panel admin) sans en avoir les droits. (cf. Figure 16)

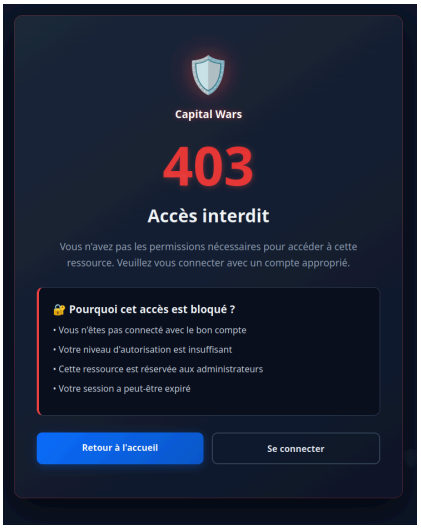


Figure 35: Page d'erreur 403 - Accès interdit

2.3.12.3. Erreur 500 : Erreur serveur

L'erreur 500 représente une défaillance technique inattendue du côté du serveur. La communication est ici orientée vers la réassurance et la technicité. (cf. Figure 17)

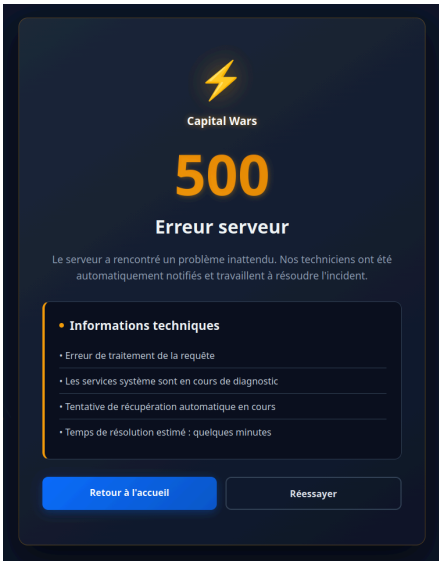


Figure 36: Page d'erreur 500 - Erreur serveur

2.3.13. Formulaire de contact

Un formulaire de contact a été réalisé afin de permettre à l' utilisateur de rapporter un bug, suggestion, ou autre. Le fonctionnement de ce formulaire est simple: il envoie les données à un WebHook Discord qui affichera ainsi le message. (cf. Figure 18 à 21)

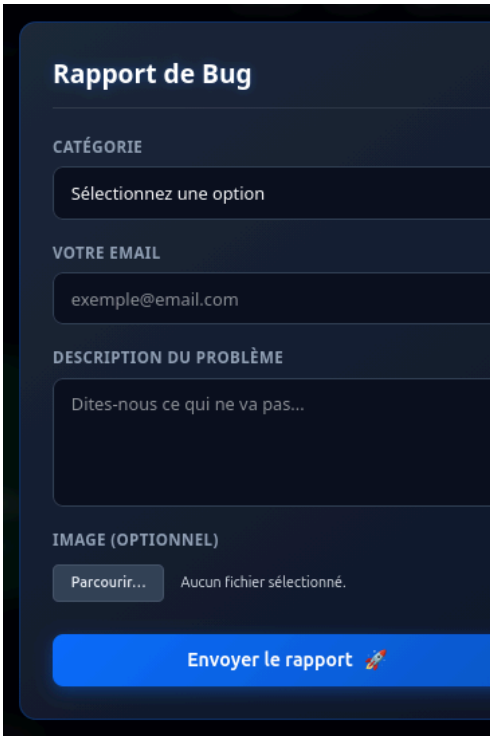


Figure 38: Formulaire de rapport de bug

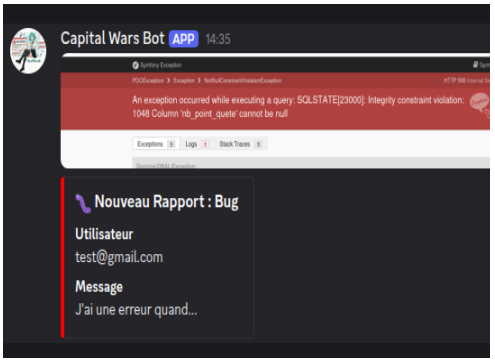


Figure 37: Notification Discord - Rapport de bug

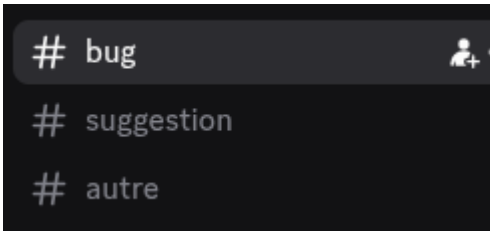


Figure 40: Organisation canaux Discord thématiques

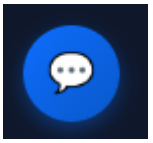


Figure 39: Bulle de chat flottante support

2.3.14. Usines

2.3.14.1. Général

Les usines sont le cœur des entreprises elles permettent la création de matières utilisables à partir de matière première. Ainsi pour ne pas avoir une trop grande production chaque entreprise peut avoir un maximum de 2 usines, chacune de ces usines possède un type qui ont des capacités de production et prix d'achat différents (*Ces statistiques sont basées sur les règles d'EXIGE)*.

Type d'usine	Capacité annuelle	Prix d'achat	Délai d'achat
6	600 000	14 000 000	1 an
4	400 000	10 000 000	
3	300 000	8 000 000	Disponible en P0

Ainsi, lorsqu'une entreprise veut acheter une usine pour augmenter sa production, elle doit respecter quelques conditions. La première ne possède pas déjà 2 usines, la seconde l'entreprise ne pourra acheter une usine que si l'entreprise dispose des fonds nécessaires en banque. Une fois, acheter le paiement d'une usine s'effectue en deux périodes. La première période, l'entreprise paie 50 % du prix d'achat. La deuxième période, elle paie le reste. L'usine sera alors en construction et elle commencera à produire l'année suivante. (cf. Figure 22)

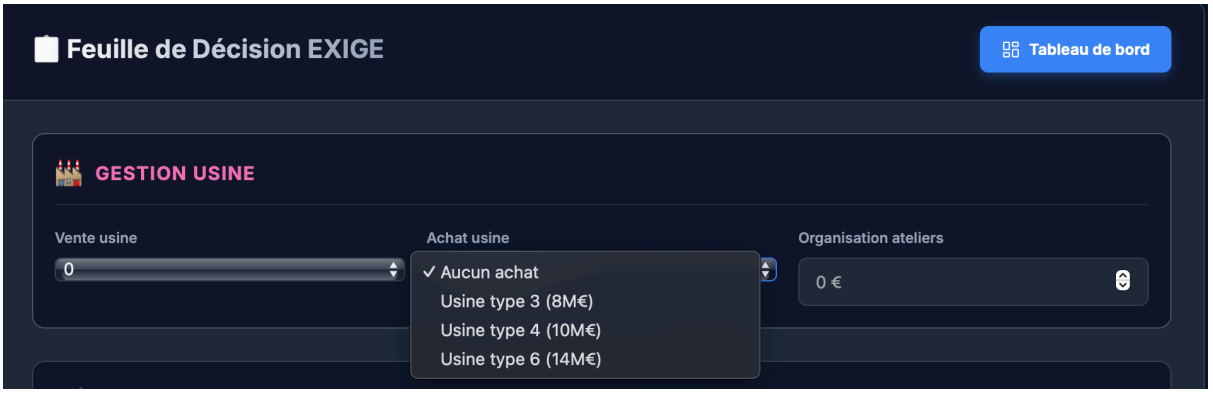


Figure 41: Interface gestion du parc industriel

Aussi la vente d'une usine se fait également via la feuille de décision mais comme pour l'achat elle doit respecter une condition, elle doit posséder au moins deux usines pour pouvoir en vendre une. Une fois l'usine vendue, l'entreprise reçoit immédiatement le prix actuel de l'usine, le solde étant versé l'année suivante.

Aussi comme dans le jeu exige les usines perdre chaque année une capacité de production qui s'élève à 20 000.

2.3.14.2. Conception

entreprise : Référence (relation Plusieurs-Un) vers l'entité Entreprise propriétaire.

periodeAchat : Référence (relation Plusieurs-Un) vers l'entité PeriodeDeJeu, utilisée pour calculer l'amortissement annuel de l'usine.

emplacementId : Attribut nullable (par défaut NULL), défini sur l'ID de l'entité Emplacement lorsque le plugin Emplacement est activé.

type : Énumération de type TypeUsine, avec valeurs possibles : "3", "4" ou "6". La classe TypeUsine expose des méthodes pour récupérer le type, le prix initial et la capacité de production initiale.

capacite : Entier représentant la capacité actuelle de production, diminuant de 20 000 unités par année.

periodeActivation : Référence (relation Many-to-One) vers l'entité PeriodeDeJeu, indiquant la période à partir de laquelle l'usine devient opérationnelle, facilitant le tri des usines inactives.

2.3.14.3. Affichage

Une interface d'affichage a ensuite été développée. Elle permet aux entreprises de visualiser à la fois les usines qu'elles possèdent et la capacité de production de ces dernières. (cf. Figure 23)

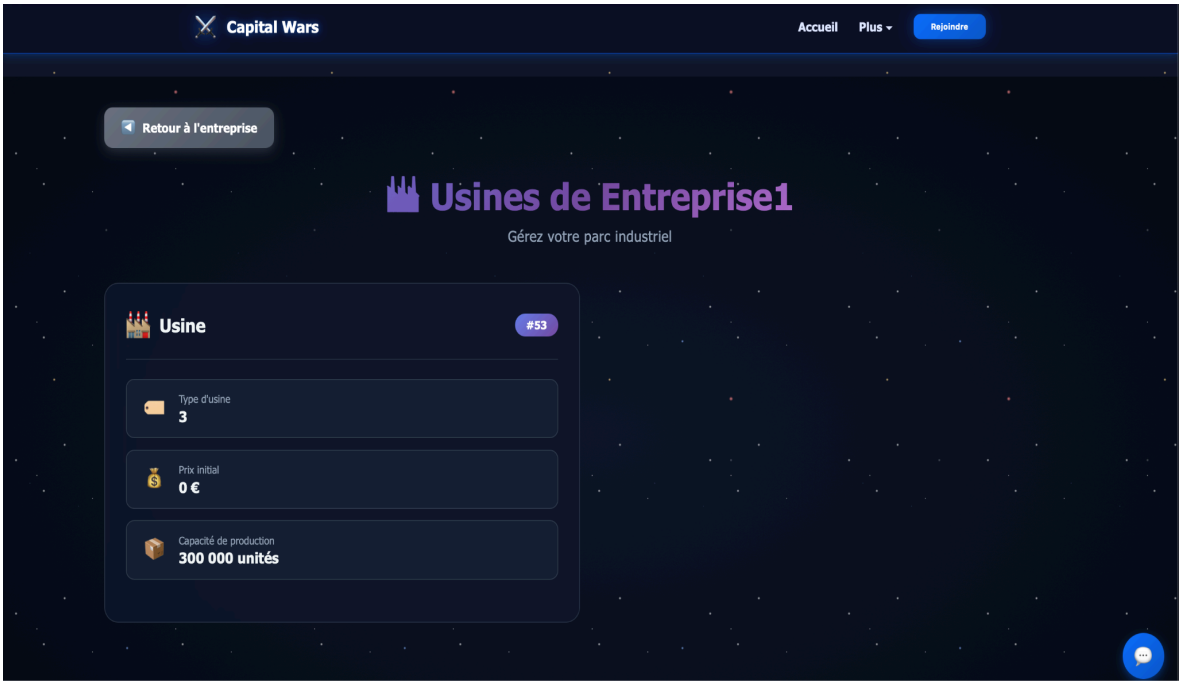


Figure 42: Module Gestion Usine - Feuille de décision EXIGE

2.3.15. Panneau d'administration

Au-delà de la gestion individuelle des lobbys, la plateforme dispose d'un système d'administration globale réservé aux utilisateurs possédant le rôle **Admin**. Ce panneau permet de superviser l'intégralité de l'écosystème de Capital Wars, d'analyser la santé de la plateforme en temps réel et d'intervenir sur n'importe quelle entité (utilisateurs, entreprises, sessions de jeu).

2.3.15.1. Tableau de bord et Statistiques en temps réel

La page d'accueil de l'administration offre une vision macroscopique de l'activité. Elle a été conçue comme un centre de pilotage rapide (cf. Figure 24) :

- **Indicateurs de performance (KPIs) :** Des cartes dynamiques affichent le nombre total d'utilisateurs inscrits, d'entreprises créées, de parties en cours et d'organisateurs actifs, avec des indicateurs de croissance en pourcentage.
- **Actions rapides :** Une grille de navigation permet d'accéder instantanément aux différents modules de gestion (utilisateurs, lobbys, entreprises, périodes) ou aux paramètres système.
- **Accessibilité :** Le menu d'administration est intégré directement dans la barre latérale sous forme de bloc distinct ("Panneau Admin") pour les comptes autorisés, facilitant le passage du mode joueur au mode superviseur.

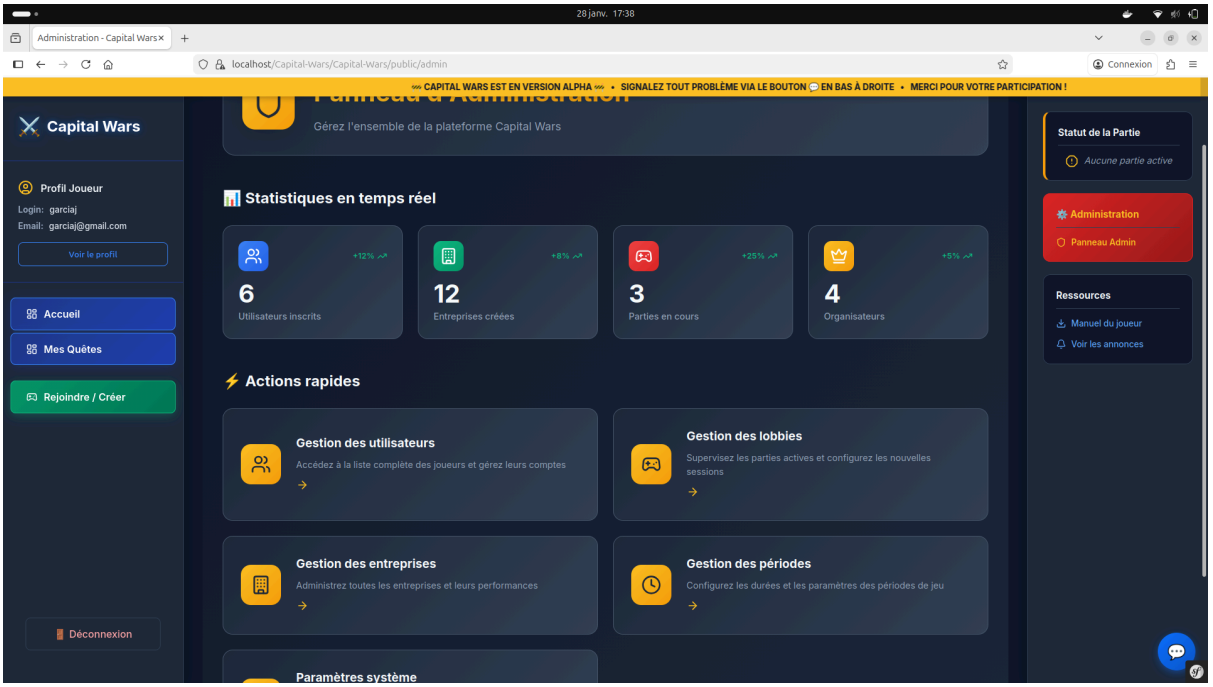


Figure 43: Tableau de bord administrateur - KPIs en temps réel

2.3.15.2. Gestion des entités

Chaque module de gestion a été réalisé sous forme de tableaux de données interactifs, permettant une recherche et un filtrage efficace (cf. Figure 25 à 28) :

- **Gestion des Utilisateurs :** Permet de consulter la liste complète des comptes, de vérifier leurs rôles (Admin, Organisateur, Joueur) et de voir leurs participations actives. Une fonction d'exportation est intégrée pour le suivi externe des données.
- **Supervision des Lobbies et Entreprises :** Ces vues centralisées permettent de monitorer toutes les sessions créées sur la plateforme. L'admin peut voir le statut de remplissage de chaque lobby et l'état de santé financier de chaque entreprise (capital, nombre d'employés, usines opérationnelles).
- **Contrôle des Périodes :** Un historique global de toutes les périodes générées permet de surveiller le rythme de jeu sur l'ensemble du serveur et d'identifier d'éventuels blocages.

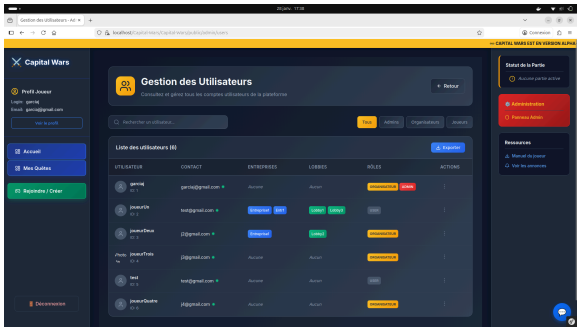


Figure 44: Panneau Gestion des Utilisateurs

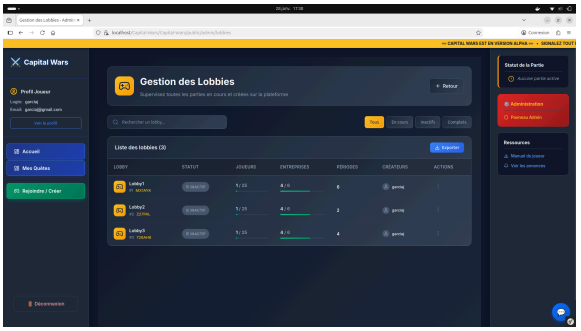


Figure 45: Panneau Gestion des Lobbies

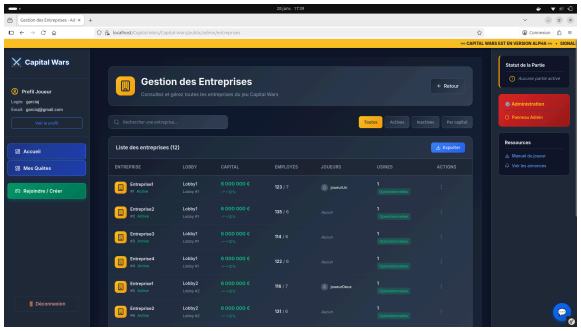


Figure 46: Panneau Gestion des Entreprises

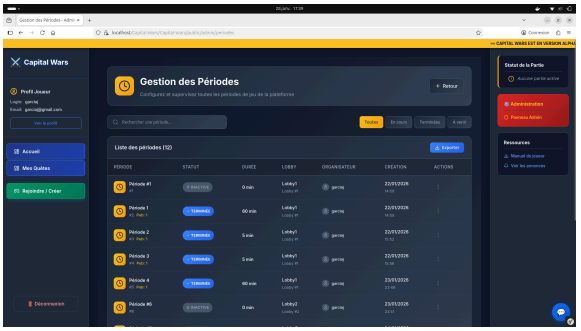


Figure 47: Panneau Gestion des Périodes

2.3.15.3. Paramètre du site et maintenance

L'interface de configuration permet de modifier le comportement global de l'application sans toucher au code (cf. Figure 29) :

- **Configuration générale :** Modification dynamique du nom du site et de la description SEO utilisée pour le référencement.
- **Mode Maintenance :** Un interrupteur permet d'activer instantanément une page d'attente pour tous les utilisateurs, bloquant les accès lors des mises à jour techniques.
- **Valeurs par défaut du Jeu :** Centralisation des variables globales, comme la durée standard d'une période ou le nombre maximum de joueurs autorisés par lobby.

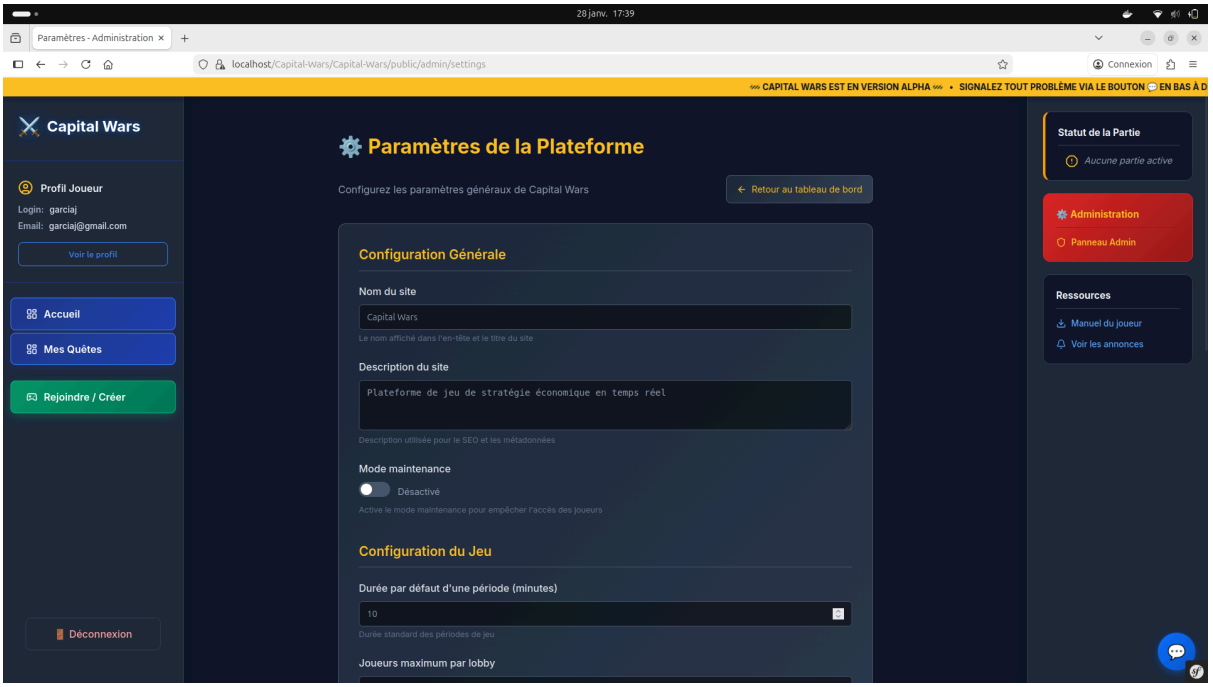


Figure 48: Interface Paramètres de la Plateforme

3. Gestion de projet

3.1. Réunion

3.1.1. Réunion de planification de sprint

Au début de chaque bloc de SAE, nous avons tenu une réunion (en présentiel ou par visioconférence) avec l'ensemble de l'équipe et M. Chollet. Ces réunions ont permis de définir les attentes et les tâches à accomplir pour le "sprint" concerné. Un compte-rendu de chaque réunion a ensuite été rédigé.

3.1.2. Mêlée quotidienne

Nous organisions une mêlée quotidienne (tous les jours ou tous les deux jours) afin de suivre l'avancement de chacun et d'identifier les problèmes potentiels. Ces réunions permettaient d'offrir de l'aide aux membres de l'équipe qui en avaient besoin, de surveiller la progression globale du projet et de faire le point sur les fonctionnalités développées par chacun.

3.1.3. Réunion de revue de sprint

À la fin de chaque itération, ou « sprint », une réunion de bilan d'une heure est systématiquement organisée avec M. Chollet. Cette rencontre a pour objectif principal de passer en revue l'ensemble des *issues* (tâches ou problèmes) qui ont été complétées et validées durant la période écoulée. C'est un moment crucial pour évaluer la charge de travail réalisée, identifier les réussites et les difficultés rencontrées par l'équipe, et confirmer la conformité des livrables aux attentes initiales. En outre, cette réunion permet de faire le point sur l'avancement global du projet.

3.1.4. Réunion de rétrospective de sprint

Finalement, après chaque revue de sprint avec M. Chollet, nous tenions une brève réunion. Celle-ci avait pour objectif d'identifier les problèmes potentiels rencontrés durant le sprint afin de mieux préparer la période d'SAE suivante.

3.2. Github

3.2.1. Issues

Nous avons mis en place une méthodologie rigoureuse pour la gestion de nos **issues**, chacune se voyant attribuer un niveau de priorité et d'effort matérialisé par un **poids (XS, S, M, L, XL, etc)**. L'ensemble de ces issues a été généré de manière semi-automatisée afin d'assurer une couverture exhaustive de nos besoins de développement. (cf. Figure 30)

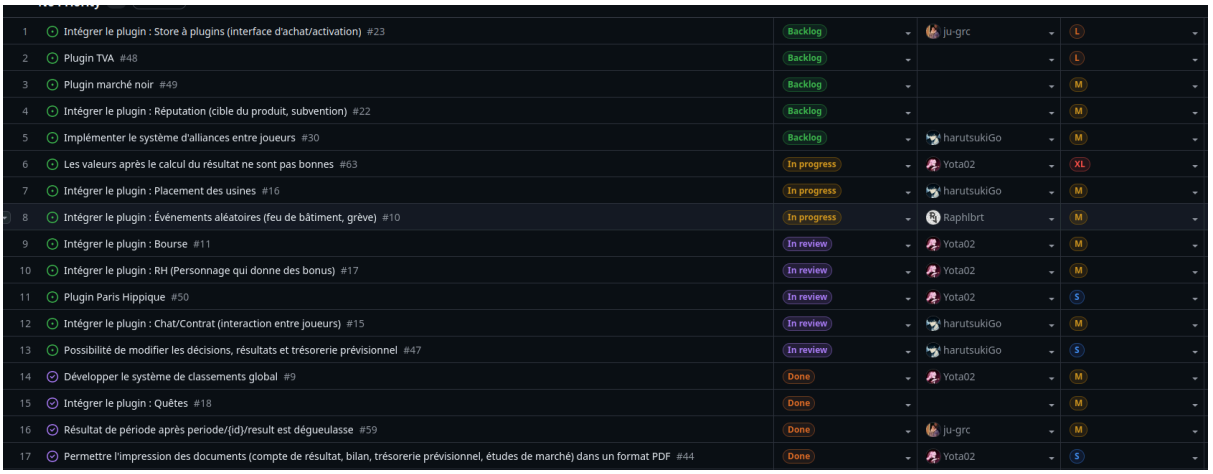


Figure 30: Backlog GitHub avec tags de priorité

Le processus de création d'une issue se déroulait en plusieurs étapes distinctes et optimisées (cf. Figure 31):

1. **Rédaction initiale de la fonctionnalité** : Dans un premier temps, nous définissions et articulions clairement la spécification de la fonctionnalité désirée. Cette étape consistait à rédiger le besoin en texte brut, en se concentrant sur le quoi et le pourquoi.
2. **Traitement et formalisation automatique** : Le texte en brut rédigé était ensuite transmis à un petit programme développé spécifiquement. Ce programme exploitait un modèle de langage exécuté en local garantissant rapidité, confidentialité et gratuité afin de générer une description d'issue complète, structurée, et bien formulée, intégrant les sections standard (titre, description détaillée, etc.).
3. **Intégration directe à la plateforme** : Une fois l'issue correctement formatée, le programme prenait en charge son push direct sur notre dépôt GitHub ou l'outil de gestion de projet associé.

Il s'agissait du même fonctionnement pour le processus de signalement et de suivi des **bugs**.

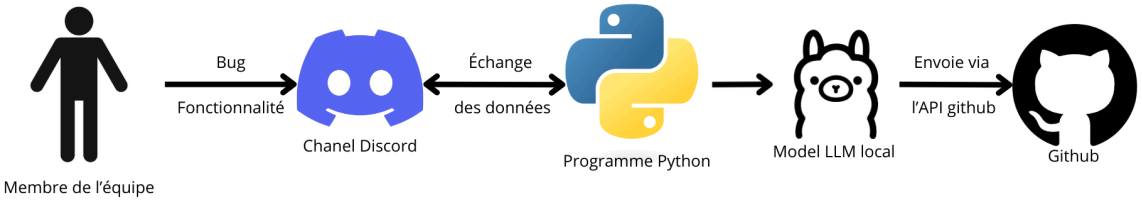


Figure 31: Workflow automatisé de création d'issues

3.2.2. Backlog

Nous avons géré les différentes problématiques et les tâches à réaliser au cours du projet en utilisant un backlog de manière rigoureuse. Cette organisation, essentielle pour le suivi et la priorisation, était divisée en plusieurs sections distinctes, permettant une vision claire de l'état d'avancement et des actions à venir. (cf. Figure 32)

- **Backlog** : Regroupe toutes les issues à réaliser pour le projet.
- **Ready** : Contient les issues à réaliser durant le sprint actuel.
- **In Progress** : Liste les issues en cours de développement.
- **In Review** : Identifie les issues en cours de test.
- **Done** : Indique les issues terminées.

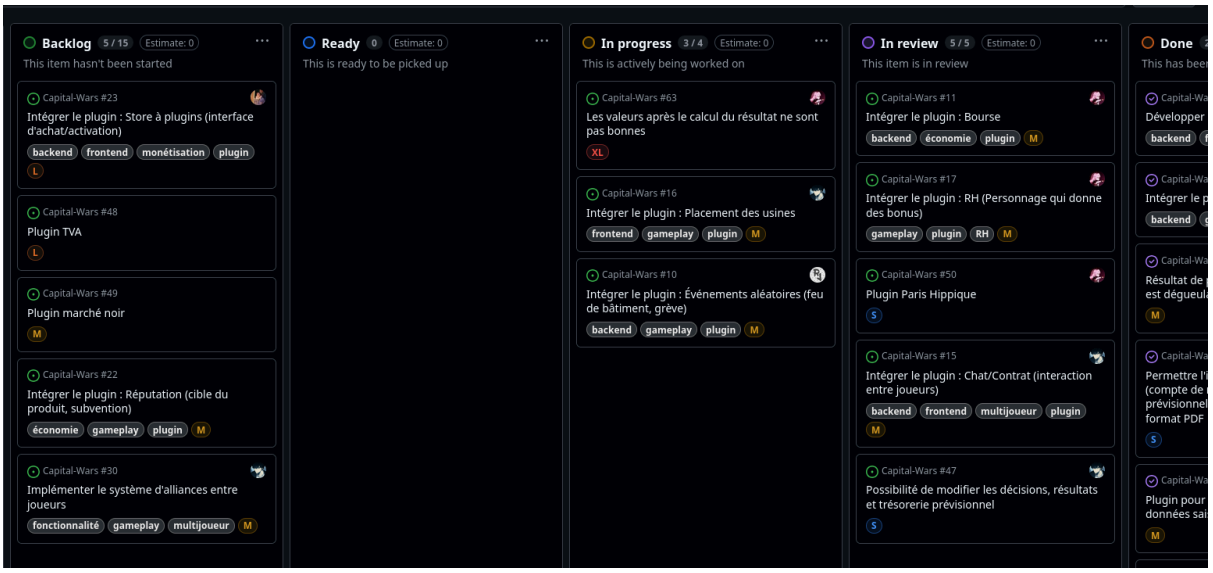


Figure 32: Tableau Kanban de gestion Agile

Chaque issue du backlog était assignée à un membre de l'équipe, datée, et son statut (À faire, En cours, En revue, Terminé).

3.3. Discord

3.3.1. Recap journalier

Afin d'assurer une transparence totale et un suivi rigoureux de l'avancement du projet, nous avons mis en place l'envoi quotidien d'un récapitulatif détaillé des activités. Ce rapport journalier permet non seulement de synthétiser les tâches accomplies et les livrables produits par chaque membre de l'équipe, mais il offre également une visibilité complète sur le travail de chacun. Cette pratique s'est avérée essentielle pour maintenir une coordination efficace, identifier rapidement les éventuels points de blocage ou les chevauchements de tâches, et s'assurer que tous les efforts sont alignés sur les objectifs globaux de Capital Rivals. De plus, la compilation de ces rapports quotidiens crée un historique précieux et facilement consultable, qui servira de référence pour l'évaluation des performances, l'audit du projet et la documentation des processus. (cf. Figure 33)

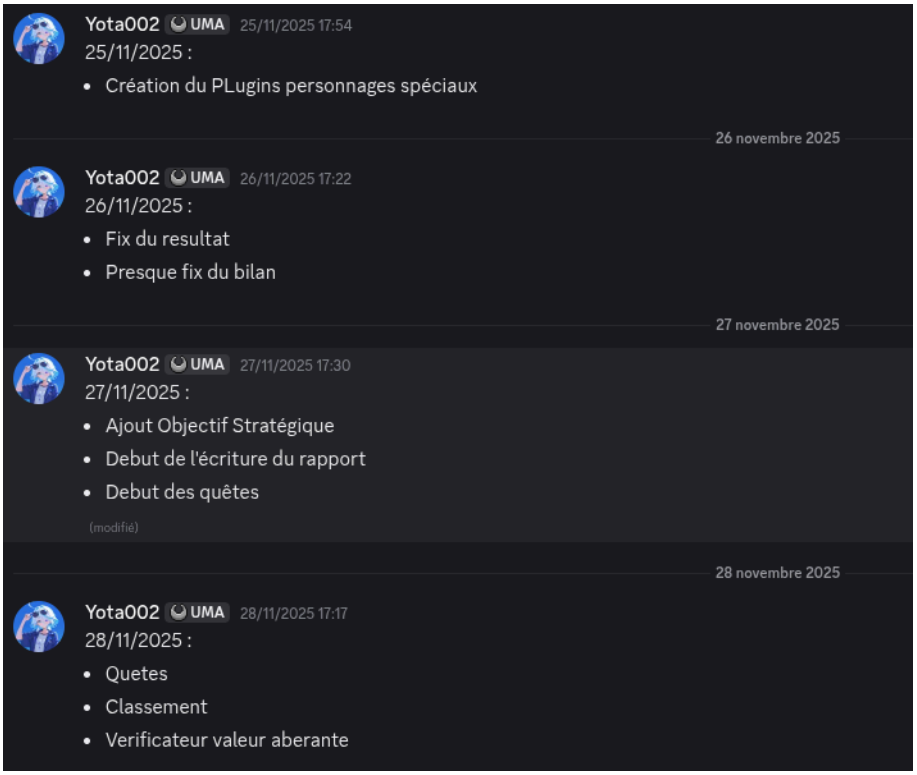


Figure 33: Récaps journaliers Discord - Messages quotidiens

3.3.2. Canal de communication

Pour structurer nos échanges et éviter la dispersion des informations, nous avons organisé le serveur Discord en salons thématiques distincts. Cette architecture sépare clairement les discussions techniques, la gestion administrative et les espaces créatifs.

Intérêts majeurs pour la gestion de projet :

- **Centralisation :** Le serveur agit comme un référentiel unique regroupant les discussions de l'équipe, les documents clés et les notifications techniques.
- **Traçabilité et Organisation :** Contrairement à un flux de conversation unique, cette segmentation permet de retrouver facilement les fichiers et décisions passées, assurant une meilleure accessibilité de l'information.
- **Communication ciblée :** Chaque membre peut se concentrer sur les canaux pertinents pour ses tâches, ce qui améliore l'efficacité globale.

3.3.3. Notifications de maintenance

Une connexion entre le serveur et Discord a été mise en place pour envoyer une notification via un webhook Discord vers un salon dédié. Cette notifications se déclenche à chaque fois qu'une maintenance est déclenchée. Cela assure un historique des maintenance. (cf. Figure 34)

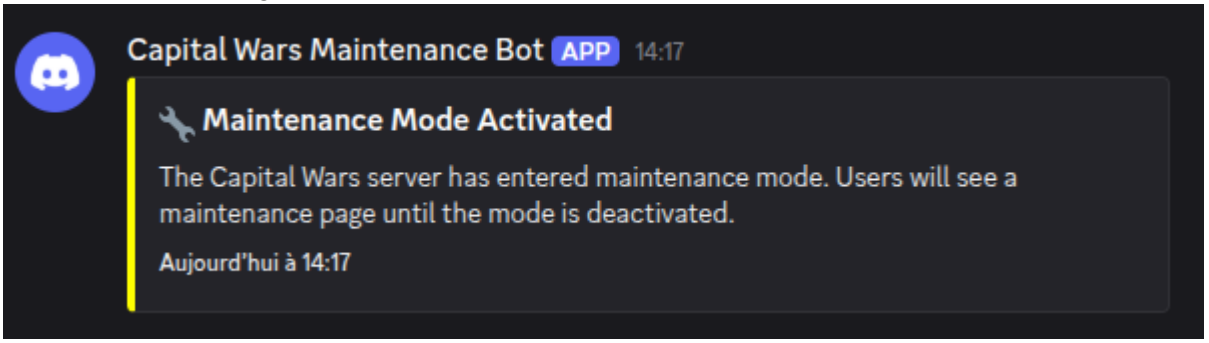


Figure 34: Notification Discord - Maintenance Mode Activated

3.3.4. Notifications d'inscription

Une connexion entre le serveur et Discord a été mise en place pour envoyer une notification via un webhook Discord vers un salon dédié. Cette notifications se déclenche à chaque fois qu'un nouveau compte est créé. Cela assure à la fois un historique des inscriptions et des informations sur le nombre de personnes s' inscrivant. (cf. Figure 35)

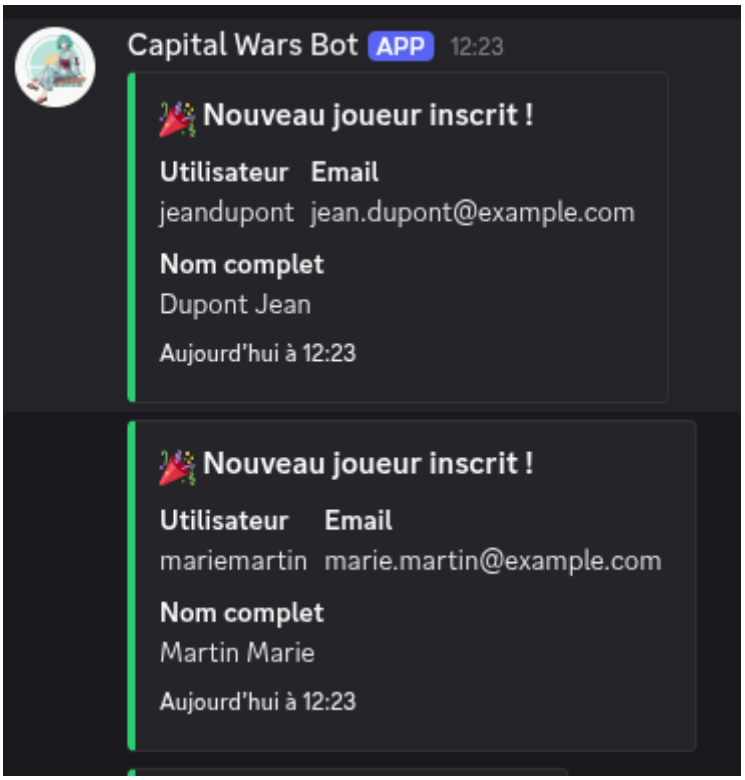


Figure 35: Notifications Discord - Nouveaux joueurs inscrits

3.3.5. Notifications de merges

Une GitHub Action a été mise en place pour envoyer une notification via un webhook Discord vers un salon dédié. Cette action se déclenche à chaque push sur les branches developpement ou main. Cela assure à la fois un historique des modifications et une visibilité sur les contributions individuelles. (cf. Figure 36)

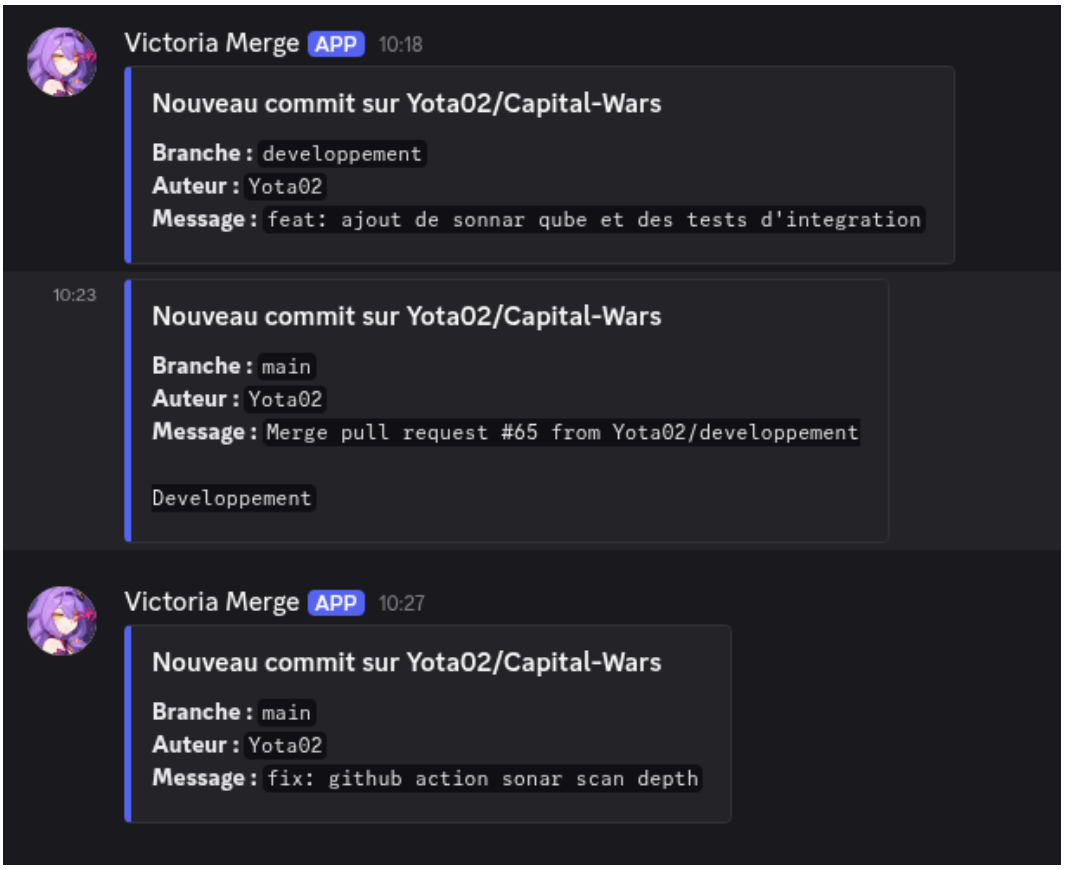


Figure 36: Notification Discord - Commit/Merge

3.3.6. Notification test

À chaque push sur les branches **main** ou **developpement**, les tests sont exécutés automatiquement. Une notification est ensuite envoyée via webhook sur un salon Discord, affichant toutes les informations nécessaires : l'auteur, la branche, les changements effectués, le nom du commit et le statut de l'action GitHub. (cf. Figure 37)

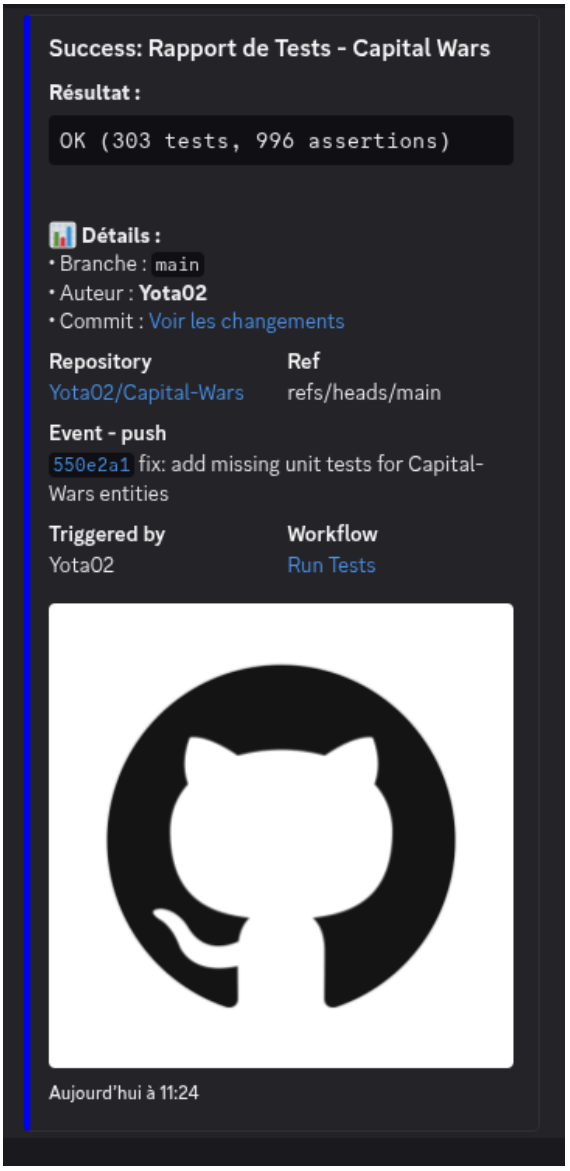


Figure 37: Rapport de tests automatisé Discord

3.3.7. Stockage

Pour centraliser l'information et faciliter la collaboration au sein de l'équipe, des salons thématiques Discord ont été créés. Ces salons servent de référentiel unique pour tous les documents de travail essentiels, incluant :

- **Les comptes rendus de réunion** : pour suivre les décisions, actions et la progression.
- **Les idées et brainstormings** : pour soumettre, discuter et affiner les concepts du jeu (mécaniques, scénarios, design).
- **Les fichiers de travail importants** : ébauches de design, spécifications techniques, plannings de production.

Cette structure Discord améliore la traçabilité, l'accessibilité de l'information et favorise une communication ciblée, contribuant à une gestion de projet plus efficace. (cf. Figure 38)

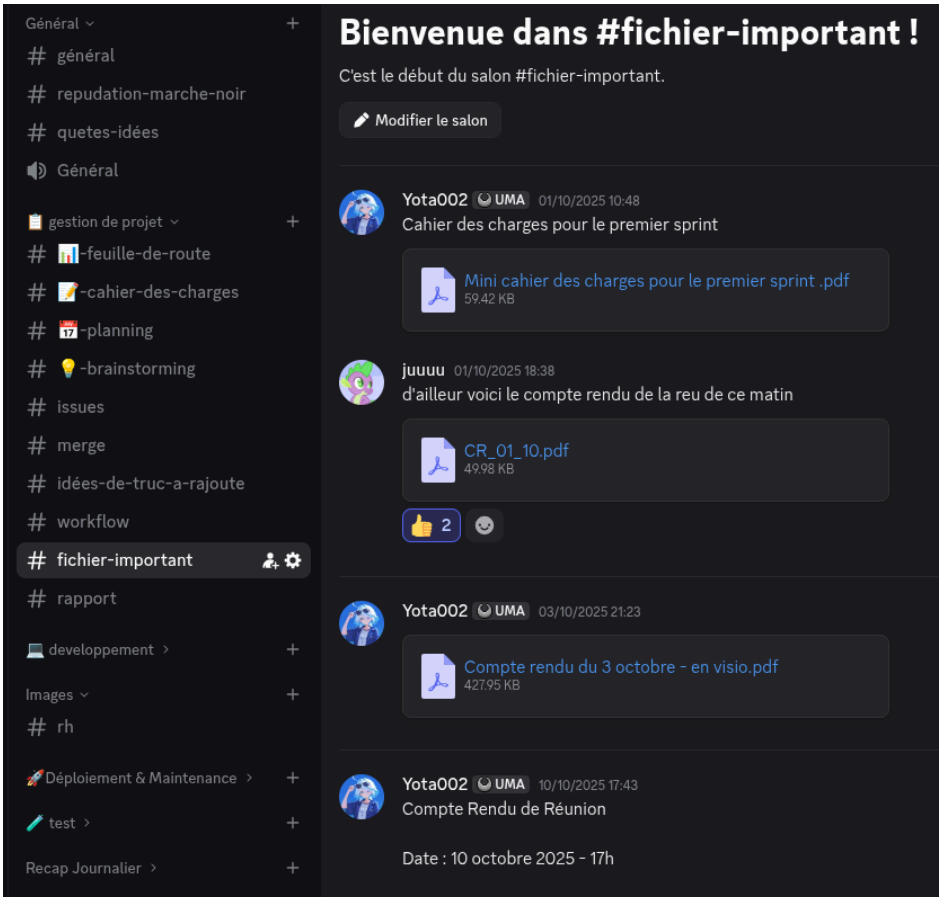


Figure 38: Canal Discord - Centralisation documents

3.4. Feuille de route

La planification du projet Capital Wars a été structurée sur une période d'un an, découpée en trois phases majeures. Cette feuille de route vise à assurer une transition maîtrisée entre le prototype technique (MVP) développé au Semestre 5 et une version commercialisable (V1.0) prévue après le Semestre 6. (cf. Figure 39)

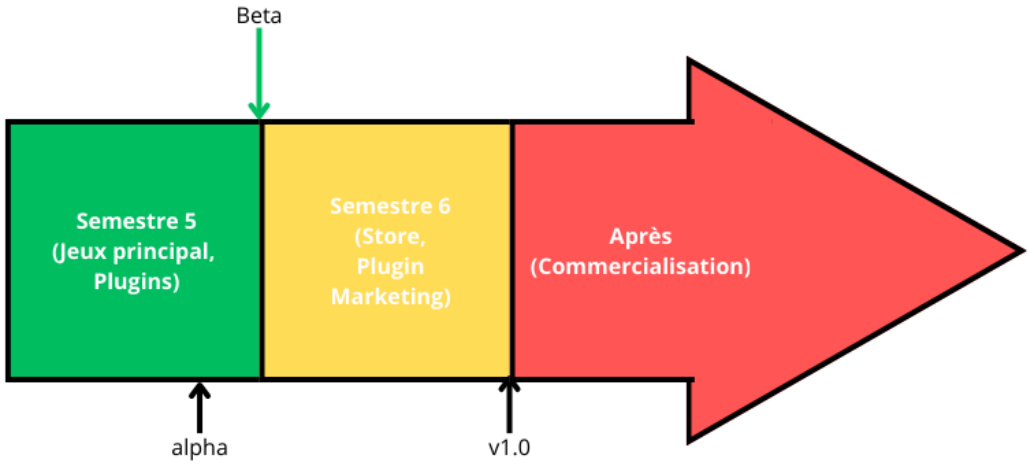


Figure 39: Diagramme feuille de route - Roadmap

3.4.1. Alpha public

Nous avons récemment franchi une étape cruciale dans le développement de Capital Rivals en lançant une **alpha publique**. Cette phase est fondamentale dans notre processus d'itération et d'amélioration continue. (cf. Figure 40)

Figure 40: Bandeau avertissement - Version Alpha

Objectifs de l'Alpha Publique :

1. **Test à Grande Échelle** : Rendre le projet accessible à un large éventail de joueurs permet de soumettre le jeu à des conditions réelles d'utilisation, bien au-delà de ce que nos tests internes peuvent simuler.
2. **Détection de Bugs et d'Anomalies** : L'afflux de nombreux joueurs est le moyen le plus efficace de découvrir des bugs (mineurs, critiques, d'interface, de gameplay, etc.) qui n'auraient pas été identifiés par notre équipe de développement. Chaque session de jeu contribue à un processus de Quality Assurance distribué et exhaustif.
3. **Recueil de Retours d'Expérience et d'Idées :** L'alpha publique est une source inestimable de suggestions et d'idées d'amélioration. Les retours des joueurs sur l'équilibrage du jeu, l'ergonomie de l'interface utilisateur, la clarté des mécaniques de jeu, et l'expérience utilisateur globale sont essentiels pour guider nos prochaines étapes de développement et assurer que le produit final réponde aux attentes de notre future communauté.

En somme, cette alpha publique est bien plus qu'un simple test : c'est une démarche proactive visant à identifier les points de friction et à capitaliser sur les opportunités d'optimisation avant de passer aux phases ultérieures (bêta, lancement).

4. Bilan des apprentissages

4.1. Général

La réalisation de Capital Wars a constitué une opportunité majeure de mettre en pratique l'ensemble des compétences acquises durant notre formation, tout en nous confrontant aux réalités d'un projet de développement complexe et ambitieux.

Au niveau technique, la maîtrise du framework Symfony a été approfondie, notamment à travers la mise en place d'une architecture modulaire non apprise en cours. Le défi principal réside dans la conception de ce système de plugins, permettant d'activer ou désactiver des fonctionnalités sans altérer le cœur du jeu. Par ailleurs, l'intégration d'outils d'industrialisation logicielle tels que GitHub Actions pour l'intégration continue et SonarQube pour l'analyse de la qualité du code nous a permis de maintenir un standard de code professionnel tout au long du développement.

Au niveau de la gestion de projet, l'application de la méthodologie Agile (Scrum) a structuré notre travail. L'organisation en sprints, rythmée par les mêlées quotidiennes et les revues de sprint, a favorisé une communication fluide et une détection rapide des points de blocage. L'utilisation de Discord pour les notifications automatisées (Webhooks pour les bugs, inscriptions et merges) a centralisé l'information, rendant le suivi du projet transparent et réactif.

4.2. Perspective d'amélioration

Bien que le projet ait atteint un stade fonctionnel avancé avec le lancement d'une Alpha publique, plusieurs axes d'amélioration ont été identifiés pour améliorer Capital Wars au prochain semestre :

- **Enrichissement du catalogue de Plugins :** Actuellement, le jeu propose des plugins comme Paris Hippique ou Catastrophes PESTEL. L'objectif futur serait de développer de nouveaux modules (espionnage industriel, marketing plus poussé, nouveau produit, etc) pour varier davantage les parties.
- **Optimisation de l'Expérience Utilisateur :** Les retours de l'Alpha sont cruciaux. Ils permettront d'affiner l'interface, notamment la lisibilité des graphiques financiers et l'accessibilité mobile, afin de rendre le jeu plus intuitif pour les néophytes en gestion.
- **Déploiement d'une Marketplace intégrée** : Pour soutenir le modèle économique du projet, nous envisageons la création d'un "Store" connecté nativement à l'application. Cette plateforme centralisée permettrait aux utilisateurs d'acquérir leurs plugins additionnels via une interface dédiée, fluidifiant ainsi l'expérience d'achat et de personnalisation des parties.
- **Gamification :** Afin de fidéliser les joueurs sur le long terme, nous souhaitons gamifié l'application. Cela inclut l'amélioration du système de progression persistant entre les parties (niveaux, expérience), des classements globaux (leaderboards) et des récompenses cosmétiques (skins d'interface, avatars). L'objectif est de créer un sentiment d'accomplissement au-delà de la simple victoire d'une session.
- **Immersion Narrative et Thématique (Lore)** Pour dépasser le cadre du simple tableau de gestion, nous prévoyons d'enrichir l'univers du jeu. Cela passera par l'ajout de **thèmes historiques ou futuristes** accompagnés d'un "Lore". Contextualiser les scénarios économiques apportera une plus-value immersive..
- **Intelligence Artificielle :** L'exploitation des données générées par les parties jouées ouvre la voie à l'entraînement d'un modèle d'**Intelligence Artificielle**. À terme, cette IA pourrait remplir deux rôles : un adversaire robuste capable de remplacer un joueur absent ou de challenger les meilleurs, et un assistant virtuel ("tutoriel intelligent") prodiguant des conseils stratégiques en temps réel.

- **Suivi et analyse des données (Général) :** La création d'une interface d'administration est envisagée pour le suivi et l'exploitation des données utilisateurs. Cet outil permettra de mieux comprendre l'usage des fonctionnalités en particulier les plus utilisées afin d'identifier des pistes d'amélioration. L'objectif est d'intégrer de nouveaux indicateurs de performance.
- **Valorisation et Promotion** : Enfin, le succès de *Capital Wars* passe par sa visibilité. Une stratégie de communication sera mise en place, incluant la réalisation d'un **site vitrine promotionnel** et de campagnes de diffusion (réseaux sociaux, démonstrations). Cela vise à attirer de nouveaux joueurs et à fédérer une communauté active autour du projet.

Conclusion

Le projet Capital Wars est né d'une double ambition, à la fois pédagogique et technique, visant à combler les lacunes des solutions existantes sur le marché des Serious Games de gestion d'entreprise. Notre objectif initial était de concevoir un jeu non seulement engageant et réaliste pour les apprenants, mais également de relever le défi d'une architecture web moderne, hautement flexible et pérenne. Face aux rigidités et au manque de modularité observés chez des concurrents bien établis tels que Virtonomics ou Sim Companies, nous avons opté pour une approche novatrice. Le résultat est une plateforme sophistiquée où la complexité du modèle économique et des scénarios de jeu est entièrement modulable grâce à l'implémentation d'un système de plugins unique et performant.

À l'issue de ce semestre intensif, nous sommes fiers de confirmer que tous les objectifs définis dans le cahier des charges initial ont été pleinement atteints et même dépassés. Le livrable final est une application non seulement fonctionnelle, mais qui témoigne également d'une qualité logicielle irréprochable. Cette qualité est attestée par la mise en place de processus rigoureux, incluant une couverture significative de tests unitaires et d'intégration, ainsi qu'une démarche d'amélioration continue validée par des outils d'analyse statique comme SonarQube, garantissant la maintenabilité et la robustesse du code.

Loin de se limiter à un simple exercice académique, la réalisation de Capital Wars nous a permis de simuler un environnement professionnel de développement complet et réaliste. Ce parcours a englobé toutes les étapes clés du cycle de vie d'un projet logiciel, allant de la conception détaillée de la base de données relationnelle (modélisation, optimisation des requêtes) à l'élaboration de l'interface utilisateur, en passant par l'intégralité du développement back-end et le déploiement continu sur une infrastructure cible. Cette phase s'est conclue par la livraison d'une version Alpha stable, soumise à des tests par des utilisateurs réels, dont les retours ont validé l'ergonomie et la pertinence pédagogique de la solution.

En conclusion, Capital Wars ne représente pas un point final, mais bien la fondation d'une plateforme solide et hautement évolutive. Grâce à sa conception intrinsèquement modulaire et découplée, il est parfaitement outillé pour accueillir de nouvelles fonctionnalités (ajout de modules de marché financier, gestion des ressources humaines avancée, etc.) sans nécessiter de refonte majeure. Ce projet valide non seulement l'efficacité et la performance de notre approche technique (architecture micro-services, utilisation de technologies modernes), mais également la pertinence de notre organisation et de notre gestion de projet (méthodes Agiles). Capital Wars est désormais prêt à servir d'outil d'apprentissage ludique et performant pour les futures promotions d'étudiants en gestion, confirmant sa vocation de référence dans le domaine des Serious Games.

1. Utilisation de l'IA

L'intelligence artificielle a été employée dans trois domaines distincts durant ce projet :

1. **Génération d'images :** L'IA a servi à la création d'éléments visuels tels que les images pour les plugins, les logos, etc.
2. **Aide à la rédaction :** Elle a assisté à la préparation de ce rapport.
3. **Développement de code :** L'utilisation de l'IA dans le codage s'est faite selon trois axes principaux :
    - **Correction de bugs :** L'IA a permis un gain de temps considérable pour l'identification et la correction de bugs, souvent dus à des variables mal instanciées ou nommées, ou à des oublis de points-virgules. Néanmoins, chaque correction a été validée et comprise par un membre de l'équipe.
    - **Génération du frontend** : L'outil Gemini de Google a fourni des templates permettant de générer des idées floues dans notre tête, facilitant grandement la conception des interfaces graphiques et accélérant le processus de développement.
    - **Génération de code répétitif/redondant :** L'IA a été mobilisée pour automatiser la création de code récurrent, notamment pour la logique des contrôleurs.

Il est important de souligner que, dans le domaine du code, les membres de l'équipe ont toujours supervisé et effectué des revues de code systématiques pour garantir la qualité et la pertinence des modifications apportées. L'IA n'a en revanche pas été utilisée pour définir l'architecture globale du projet.

Bibliographie

Technologies et Outils de Développement

[1] Symfony SAS, "Symfony: High Performance PHP Framework for Web Development," Symfony.com. [En ligne]. Disponible: https://symfony.com. (Consulté le: 26 janv. 2026).

[2] The PHP Group, "PHP: Hypertext Preprocessor Manual," PHP.net. [En ligne]. Disponible: https://www.php.net/docs.php.

[3] GitHub, "GitHub Actions Documentation," GitHub Docs. [En ligne]. Disponible: https://docs.github.com/en/actions.

[4] SonarSource, "Code Quality and Security with SonarQube," SonarQube.org. [En ligne]. Disponible: https://www.sonarqube.org/.

[5] Discord, "Discord Developer Portal — Webhooks," Discord.com. [En ligne]. Disponible: https://discord.com/developers/docs/resources/webhook.

[6] Sebastian Bergmann, "PHPUnit – The PHP Testing Framework," PHPUnit.de. [En ligne]. Disponible: https://phpunit.de/.

[7] M. Gasquet, "TD1 – Découverte du framework Symfony 1/2," R5.A.05 - Programmation Avancée Web. [En ligne]. Disponible: https://mgasquet.github.io/R5.A.05-ProgrammationAvancee-Web/tutorials/tutorial1.

Méthodologie et Gestion de Projet

[8] Atlassian, "Gitflow Workflow | Atlassian Git Tutorial," Atlassian.com. [En ligne]. Disponible: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow. [Impliqué par l'usage des branches 'main' et 'development']

Concurrents et Inspiration (État de l'art)

[9] Virtonomics, "Virtonomics: Business Simulation Game," Virtonomics.com. [En ligne]. Disponible: https://virtonomics.com/. [Analysé dans l'étude de la concurrence : source 75]

[10] Sim Companies, "Sim Companies: Online Business Simulator," SimCompanies.com. [En ligne]. Disponible: https://www.simcompanies.com/.

[11] Enlight Software, "Capitalism Lab," CapitalismLab.com. [En ligne]. Disponible: https://www.capitalismlab.com/.

[12] "Exige - Jeu de gestion d'entreprise,".

Dépendances et Environnement Technique

[13] N. Boggiano et J. Jordi, "Composer: A Dependency Manager for PHP," Getcomposer.org. [En ligne]. Disponible: https://getcomposer.org/.

[14] Oracle Corporation, "MySQL 8.0 Reference Manual," MySQL.com. [En ligne]. Disponible: https://dev.mysql.com/doc/refman/8.0/en/.

[15] Mozilla Developer Network, "JavaScript Guide," MDN Web Docs. [En ligne]. Disponible: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

[16] Derick Rethans, "Xdebug: Debugger and Profiler Tool for PHP," Xdebug.org. [En ligne]. Disponible: https://xdebug.org/.

Accessibilité et Conception Web

[17] W3C Web Accessibility Initiative (WAI), "Web Content Accessibility Guidelines (WCAG) 2.1," W3.org, juin 2018. [En ligne]. Disponible: https://www.w3.org/TR/WCAG21/.

[18] E. Gamma, R. Helm, R. Johnson, et J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994.

[19] R. C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall, 2008.

Infrastructure et Déploiement

[20] Shvam Mathur, "Setup PHP GitHub Action," GitHub Marketplace. [En ligne]. Disponible: https://github.com/shivammathur/setup-php.