

Documentation des Modifications

Étape 6 du Projet

Projet : My Little RPG equipe

Lien GitHub : https://github.com/cegep-chicoutimi/My_Little_RPG_Jordy_Aurelien.git

Branche : remise-etape6

Réalisé par :
Aurélien et Jordy

Table des matières

Documentation des Modifications	1
1. Backend - API.....	3
1.1. Program.cs.....	3
Modification : Ajout de la conversion des enums en string.....	3
1.2. ControleurMonster-APIv1.csproj	3
Modification : Ajout de la librairie BCrypt.....	3
1.3. Controllers/AuthController.cs	3
Fonction modifiée : PostUtilisateur() - Méthode d'inscription	3
Fonction modifiée : Login() - Méthode de connexion	3
1.4. Models/Tuile.cs.....	3
Propriété supprimée : imageURL.....	3
1.5. Models/Dto/ResultMoveDto.cs	3
Modification : Ajout de données lors d'un combat avec un monstre	4
1.6. Services/TuileService.cs.....	4
Fonction modifiée : GenererTuile()	4
Fonction modifiée : ChoisirType().....	4
2. Frontend - Web	5
2.1. js/config/config.js (nouveau fichier)	5
Fichier créé : Configuration centralisée	5
2.2. js/services/authService.js	5
Modification 1 : Utilisation du fichier de configuration.....	5
Modification 2 : Amélioration de la gestion d'erreurs - login()	5
Modification 3 : Amélioration de la gestion d'erreurs - register()	5
2.3. js/services/PersonnageService.js	5
Modification : Utilisation du fichier de configuration	5
2.4. js/services/gameService.js	5
Modification : Utilisation du fichier de configuration	5
Suppression de la gestion de imageURL.....	6
2.5. js/managers/GrilleManager.js (nouveau fichier).....	6
Nouveau fichier : Gestionnaire de grille	6
2.6. js/managers/GameManager.js	6
Refactoring majeur : Délégation à GrilleManager	6
Simplification de la position du joueur	6
Modification de notifications de combat visuelles	7
Simplification de updateSelectedTileInfo()	7
Amélioration de movePlayer()	7
Fonction supprimée : recenterGrid()	7
2.7. index.html.....	7
Modification 1 : Suppression des éléments dupliqués	7
Modification 2 : Mise à jour des chemins d'images dans la légende	8
Modification 3 : Réorganisation des scripts	8
3. Ressources.....	9
3.1. Images renommées	9
Renommage des fichiers d'images dans Web/images/.....	9

1. Backend - API

1.1. Program.cs

Modification : Ajout de la conversion des enums en string

Enjeu : Les enums étaient envoyés au frontend sous forme de nombres (0, 1, 2, etc.), ce qui rendait le code difficile à comprendre et à déboguer.

Justification : En convertissant les enums en string (ex : "HERBE", "EAU", "MONTAGNE"), le code devient plus lisible côté frontend, facilite le débogage et il ne fallait plus convertir les numéros en enum.

1.2. ControleurMonster-APIv1.csproj

Modification : Ajout de la librairie BCrypt

Enjeu : Les mots de passe étaient stockés en clair dans la base de données, ce qui représente un risque de sécurité majeur.

Justification : BCrypt permet de hasher les mots de passe de manière sécurisée.

1.3. Controllers/AuthController.cs

Fonction modifiée : PostUtilisateur() - Méthode d'inscription

Ajout 1 : Vérification d'unicité de l'email

Enjeu : Plusieurs utilisateurs pouvaient s'inscrire avec le même email.

Justification : La vérification d'unicité garantit qu'un email ne peut être utilisé qu'une seule fois.

Ajout 2 : Hachage du mot de passe

Enjeu : Les mots de passe étaient stockés en clair.

Justification : Le hachage avec BCrypt protège les mots de passe des utilisateurs.

Fonction modifiée : Login() - Méthode de connexion

Vérification du mot de passe avec BCrypt

Enjeu : La comparaison directe des mots de passe ne fonctionne plus maintenant qu'ils sont hashés.

Justification : BCrypt.Verify() permet de comparer le mot de passe en clair fourni avec le hash stocké en base.

1.4. Models/Tuile.cs

Propriété supprimée : imageURL

Enjeu : L'URL des images était stockée en base de données de manière redondante.

Justification : L'URL peut être déterminée côté frontend en fonction du type de tuile. Cela améliore la séparation des responsabilités et la performance.

1.5. Models/Dto/ResultMoveDto.cs

Modification : Ajout de données lors d'un combat avec un monstre

Enjeu : Permettre au front d'avoir accès à plus de données d'un monstre au lieu de seulement les HP

Justification : Permet au front d'accéder à toute les données du monstre sans aller chercher dans son Local Storage

1.6. Services/TuileService.cs

Fonction modifiée : GenererTuile()

Justification : Adaptation à la nouvelle structure du modèle Tuile sans imageURL.

Fonction modifiée : ChoisirType()

Justification : Simplification de la méthode qui se concentre uniquement sur la logique métier. Le frontend gère ses propres chemins d'images.

2. Frontend - Web

2.1. js/config/config.js (nouveau fichier)

Fichier créé : Configuration centralisée

Enjeu : Les URLs de l'API étaient codées en dur dans chaque service, rendant difficile le changement entre environnements (MacOS/Windows, dev/prod).

Justification :

- **Centralisation :** Une seule source de vérité pour l'URL de l'API
- **Maintenabilité :** Changement facile de l'environnement en modifiant un seul fichier
- **Flexibilité :** Support de différentes configurations (MacOS vs Windows)
- **Bonnes pratiques :** Séparation de la configuration du code métier

2.2. js/services/authService.js

Modification 1 : Utilisation du fichier de configuration

Justification : Utilisation de la configuration centralisée pour faciliter les changements d'environnement.

Modification 2 : Amélioration de la gestion d'erreurs - login()

Enjeu : Les messages d'erreur étaient génériques et peu informatifs pour l'utilisateur.

Justification :

- Messages d'erreur plus clairs
- Gestion spécifique des différents codes HTTP (401, 400, etc.)
- Meilleure expérience utilisateur avec des messages explicites

Modification 3 : Amélioration de la gestion d'erreurs - register()

Enjeu : Messages d'erreur non traduits et peu informatifs, notamment pour le cas de l'email déjà utilisé.

Justification :

- Gestion spécifique du code 409 (Conflict) pour informer l'utilisateur que l'email est déjà utilisé
- Distinction claire entre les différents types d'erreurs

2.3. js/services/PersonnageService.js

Modification : Utilisation du fichier de configuration

Justification : Utilisation de la configuration centralisée pour faciliter les changements d'environnement.

2.4. js/services/gameService.js

Modification : Utilisation du fichier de configuration

Justification : Utilisation de la configuration centralisée pour faciliter les changements d'environnement.

Suppression de la gestion de imageURL

Enjeu : Le backend ne retourne plus cette propriété.

Justification : Adaptation aux changements du backend. Les URLs d'images sont maintenant déterminées côté frontend en fonction du type de tuile.

2.5. js/managers/GrilleManager.js (nouveau fichier)

Nouveau fichier : Gestionnaire de grille

Description : Ce nouveau fichier encapsule toute la logique de gestion de la grille de jeu qui était auparavant dans GameManager.

Responsabilités principales :

- **Gestion des limites du monde :** Définition et validation des positions (0-50 sur X et Y)
- **Conversion de coordonnées :** Conversion entre coordonnées de grille (5x5) et coordonnées mondiales
- **Création de la grille :** Génération de l'interface visuelle avec récupération des données de l'API
- **Affichage des tuiles :** Gestion de l'affichage des tuiles explorées, monstres, et position du joueur
- **Gestion des images :** Détermination des URLs d'images basée sur le type de tuile

Enjeu : GameManager était surchargé avec trop de responsabilités (gestion du jeu + gestion de la grille), dépassant 1000 lignes de code.

Justification :

- **Principe de responsabilité unique :** Séparation des préoccupations entre logique de jeu et affichage de grille
- **Maintenabilité :** Code plus facile à comprendre, modifier et déboguer
- **Réutilisabilité :** Le GrilleManager peut être réutilisé ou testé indépendamment
- **Lisibilité :** Chaque classe a un rôle clair et bien défini

2.6. js/managers/GameManager.js

Refactoring majeur : Délégation à GrilleManager

Code supprimé : Environ 600 lignes de code liées à la gestion de la grille

Enjeu : GameManager était trop volumineux (>1000 lignes) avec des responsabilités multiples.

Justification :

- **Architecture plus propre :** Séparation des responsabilités
- **Meilleure maintenabilité :** Chaque classe a un rôle clair
- **Code plus lisible :** GameManager se concentre sur la logique du jeu

Simplification de la position du joueur

Enjeu : La gestion de la position était complexe avec des conversions entre grille et monde.

Justification :

- Initialisation de la position depuis l'API
- Stockage direct des coordonnées mondiales
- Simplification de la logique de déplacement

Modification de notifications de combat visuelles

Fonction ajoutée : `showCombatPopup()`

Enjeu : Les résultats de combat n'étaient pas clairement communiqués à l'utilisateur.

Justification :

- Popup visuel en haut à droite qui affiche le résultat du combat
- Affichage des statistiques détaillées (PV, force, défense, niveau, XP)
- Bordures colorées selon le résultat (vert pour victoire, rouge pour défaite)
- Disparition automatique après 5 secondes
- Meilleur feedback visuel que l'ancien système qui modifiait le header

Simplification de `updateSelectedTileInfo()`

Enjeu : Conversion complexe des types avec emojis inutiles et redondants et suppression de vérification inutile.

Justification :

- Utilisation directe du type string (ex : "HERBE") retourné par l'API
- Suppression des conversions d'enum (plus nécessaires grâce à `JsonStringEnumConverter`)
- Suppression des emojis pour un affichage plus simple et professionnel
- Code plus court et plus maintenable

Amélioration de `movePlayer()`

Enjeu : Méthode complexe avec plusieurs appels de recentrage et affichage peu clair des résultats de combat.

Justification :

- Remplacement de `displayCombatResult()` par `showCombatPopup()` pour un meilleur feedback visuel
- Remplacement de `recenterGrid()` par `createGrid()` direct (plus simple)
- Mise à jour des données de tuile si un monstre a été battu
- Appels explicites à `updatePlayerInfo()` et `updateMapInfo()` pour rafraîchir l'UI
- Sauvegarde unique dans `localStorage` (évite les doublons)
- Code plus lisible et plus facile à maintenir

Fonction supprimée : `recenterGrid()`

Justification : Indicateur de chargement supprimé pour simplifier le code. Les déplacements sont suffisamment rapides pour ne pas nécessiter d'indicateur visuel. Cela évite aussi de modifier le header inutilement.

2.7. index.html

Modification 1 : Suppression des éléments dupliqués

Suppression du doublon du bouton de changement de thème qui apparaissait deux fois dans l'interface.

Enjeu : Le bouton de changement de thème apparaissait deux fois dans l'interface.

Justification : Suppression du doublon pour une interface propre et cohérente.

Modification 2 : Mise à jour des chemins d'images dans la légende

Enjeu :

- URLs absolues avec localhost codées en dur

Justification :

- **URLs relatives :** Fonctionnent quel que soit l'environnement
- **Nommage cohérent :** Noms de fichiers identiques aux types de tuiles
- **Maintenabilité :** Plus facile de déplacer le projet

Modification 3 : Réorganisation des scripts

Ordre corrigé des scripts :

Enjeu :

- Scripts dupliqués (Bootstrap, ThemeManager)
- Ordre incorrect causant des erreurs de dépendances

Justification :

- **Ordre de dépendances :** config.js en premier, puis services, puis managers
- **Suppression des doublons :** Chaque script chargé une seule fois
- **Ajout des nouveaux fichiers :** Support de la nouvelle architecture

3. Ressources

3.1. Images renommées

Renommage des fichiers d'images dans Web/images/

Enjeu : Incohérence entre les noms de fichiers (anglais) et les types de tuiles (français).

Justification :

- **Cohérence :** Noms de fichiers identiques aux valeurs de l'enum TypeTuile
- **Simplicité :** Permet de construire dynamiquement les URLs : images/\${type}.png
- **Maintenabilité :** Plus facile de comprendre quelle image correspond à quel type