Rapport Individuel

Introduction

Dans le cadre de mon projet scolaire dans le langage C#, j'ai utilisé le Framework **Blazor** (C#) pour créer une application interactive. Ce projet a permis de développer trois fonctionnalités majeures : un **morpion**, une **calculatrice** et une **application météo** qui utilise l'API OpenWeatherMap. Grâce à Blazor, j'ai pu créer des interfaces utilisateurs dynamiques et réactives, tout en exploitant les avantages du langage C# pour gérer la logique applicative.

Morpion (Tic-Tac-Toe)

Objectif :

L'objectif principal de cette fonctionnalité est de permettre à deux joueurs ou un joueur contre une IA de jouer au jeu de morpion (tic-tac-toe) dans une interface web simple et intuitive.

Technologies et Fonctionnalités :

- **Blazor Web App** : Utilisé pour créer une interface réactive et gérer les interactions de jeu.
- **Mode Multijoueur et IA** : Les joueurs peuvent jouer localement ou contre une intelligence artificielle.
- **Suivi de l'état de jeu** : Le jeu maintient et vérifie les états des cases pour déterminer le gagnant ou l'égalité.

Étapes de Développement :

- **Interface Utilisateur** : Création d'une grille 3x3 représentant le plateau de jeu, avec des boutons pour chaque case.
- **Logique de Jeu** : Mise en œuvre de la logique pour gérer les tours des joueurs, ainsi que les règles de victoire ou de match nul.
- **Mode IA** : (Optionnel) Ajout d'un mode de jeu où l'utilisateur peut jouer contre une IA qui prend des décisions basiques.
- **Affichage Dynamique** : Les cases sont mises à jour en fonction des actions des joueurs (X ou O).

Défis et Solutions :

- **Validation des mouvements** : Empêcher les joueurs de sélectionner une case déjà jouée.
- **Détection des Victoires** : Création d'une méthode pour vérifier les lignes, colonnes et diagonales pour déterminer un gagnant.

Calculatrice Web avec Blazor

Objectif :

Cette fonctionnalité permet de créer une calculatrice web interactive capable d'effectuer des opérations mathématiques de base (addition, soustraction, multiplication, division).

Technologies et Fonctionnalités :

- **Blazor Web App** : Utilisé pour créer une interface utilisateur réactive.
- **Opérations Mathématiques** : Support pour l'addition, la soustraction, la multiplication et la division.
- **Gestion des Erreurs** : Validation des entrées utilisateurs pour éviter des erreurs courantes telles que la division par zéro.

Étapes de Développement :

- **Création de l'Interface Utilisateur** : Une interface simple avec des boutons numériques et pour les opérations mathématiques.
- **Implémentation de la Logique** : En C#, gestion des opérations et des entrées de l'utilisateur pour afficher le résultat en temps réel.
- **Gestion des Événements** : Les événements des boutons capturent les clics et mettent à jour les résultats en temps réel.
- **Tests et Optimisations** : Tests approfondis pour garantir l'exactitude des calculs et optimisation de l'interface utilisateur pour une meilleure expérience.

Défis et Solutions :

- **Gestion des erreurs** : Validation des calculs pour prévenir les erreurs telles que la division par zéro.
- **Interaction utilisateur fluide** : Amélioration de l'expérience utilisateur en réagissant immédiatement aux clics de boutons.

Application Météo

Objectif :

La fonctionnalité permet à l'utilisateur de consulter les données météorologiques en temps réel pour une ville donnée, en utilisant l'API OpenWeatherMap.

Technologies et Fonctionnalités :

- **Blazor Web App** : Intégration des appels d'API et gestion des réponses en utilisant C#.
- **API OpenWeatherMap** : Récupération des données météo telles que la température, les conditions, l'humidité, etc.
- **Affichage Dynamique des Données** : Les utilisateurs peuvent entrer le nom d'une ville et recevoir instantanément les données météorologiques pertinentes.

Étapes de Développement :

- **Inscription à l'API** : Obtenir une clé API OpenWeatherMap pour effectuer des requêtes.
- **Intégration de l'API** : Utilisation de HttpClient pour faire des appels HTTP et récupérer les données météo.
- **Traitement des Données** : Conversion des réponses JSON en objets C# pour une utilisation facile.
- **Création de l'Interface Utilisateur** : Développement d'un formulaire permettant aux utilisateurs d'entrer une ville et d'afficher les résultats météo.
- **Gestion des Erreurs** : Affichage de messages d'erreur appropriés lorsque les données ne peuvent pas être récupérées.

Défis et Solutions :

- **Gestion des erreurs d'API** : Parfois, l'API peut ne pas renvoyer de résultats (ville incorrecte, problème réseau), et des messages d'erreur appropriés doivent être affichés à l'utilisateur.
- **Affichage réactif des données** : Mise à jour dynamique des informations météo dès qu'une ville est saisie, en garantissant une expérience utilisateur fluide.

Conclusion

Ce projet m'a permis d'acquérir une bonne maîtrise du Framework **Blazor** ainsi que des technologies associées telles que la gestion des **APIs** et le développement d'**interfaces utilisateur interactives** en **C#**. Le développement du **morpion**, de la **calculatrice** et de l'**application météo** a approfondi ma compréhension du **développement web moderne** en C#, tout en me préparant à des projets futurs dans le développement d'applications interactives