Projet PHP/MySQL : Mini Jeu d'enquête - "Le Mystère du Manoir"

Consignes :

Une documentation détaillée *est demandée, celle-ci devra* ***impérativement*** *:*

- *Présenter votre projet de façon globale, son but et son fonctionnement, vous veillerez à apporter un soin particulier à la grammaire et à l'orthographe !*
- *Des captures d'écran des pages dans différents environnements pourront illustrer cette présentation.*
- *Expliquer comment fonctionne techniquement votre projet de façon détaillée, une présentation du modèle de données et un diagramme de classes sera présenté dans le document.*
- *Un storyboard présentant l'ensemble de la navigation sur le site sera réalisé et mis en annexe, pour chaque page il devra mettre en avant :*
    - *La navigation entre les pages*
    - *Les fonctionnalités utilisées*
- *Une liste de toutes les méthodes développées figurera dans la documentation, pour chacune il faudra la présenter (nom de la méthode, son utilité, les arguments qu'elle prend en entrée, ce qu'elle renvoie).*

Le développement respectera ces règles :

- *Le développement sera basé sur le modèle MVC.*
- *Le développement sera effectué en PHP, mySQL, HTML et CSS*
- *Les pages seront en HTML5/CSS3 et conformes aux standards du W3C*
- *Les pages seront conformes aux normes d'accessibilité numérique (https://accessibilite.numerique.gouv.fr/)*
- *Le site sera fonctionnel dans un navigateur web sur ordinateur et sur mobile, il faudra faire des tests de fonctionnement sur Firefox, Safari, Chrome, Edge, Safari pour iOS, Chrome pour Android. Le fonctionnement devra être le même sur toutes les plateformes mais adapté à chaque écran.*
- *Le code sera commenté de façon utile.*
- *Toutes les informations légales seront présentes conformément à la législation française en vigueur.*

Consignes à respecter
Vos noms de propriétés doivent être en anglais et explicites. Les noms de vos méthodes doivent être en anglais et basées sur un verbe et décrire de façon explicite ce que fait la méthode.
Vous utiliserez la notation standard suivante :
Classes : camelCase avec première lettre en majuscule.
Propriétés : camelCase avec première lettre en minuscule
Méthodes : camelCase avec première lettre en minuscule

L'ensemble de votre code devra être en mesure de ***gérer les exceptions*** *!*

Description :

Le projet consiste à développer un jeu narratif interactif dans lequel le joueur incarne un détective chargé de résoudre une enquête dans un manoir. L'objectif est de découvrir qui est le coupable parmi plusieurs suspects en interrogeant des personnages, en collectant des indices et en déduisant la vérité.

Le jeu repose sur des choix qui influencent la progression et les informations disponibles. Les suspects, indices et le coupable sont générés dynamiquement à chaque partie pour rendre le jeu rejouable.

Objectifs :

1. **Apprentissage du modèle MVC :**

- Gestion de l'état du jeu (progression, collecte d'indices).

- Organisation claire entre le back-end (logique du jeu) et le front-end (affichage des scènes et des choix).

2. **MySQL :**
- Stockage dynamique des personnages, indices, et dialogues.
- Création d'une structure relationnelle pour les interactions du joueur avec les données.

3. **PHP :**
- Gestion de la logique conditionnelle (évolution en fonction des choix).
- Génération dynamique de contenu en fonction des actions du joueur.

4. **Narration et interactivité :**
- Créer une histoire immersive tout en gérant les aspects techniques.

Fonctionnement du jeu :

1. Début de la partie :
- Le joueur arrive dans un manoir où un meurtre a eu lieu.
- Plusieurs suspects sont présents, chacun ayant un alibi et un mobile.
- L'histoire commence avec une courte introduction décrivant la scène du crime.

2. Progression du jeu :
- Le joueur peut explorer différentes pièces du manoir (salon, cuisine, bibliothèque, etc.).
- Dans chaque pièce, il peut :
- **Interroger des suspects** pour obtenir des alibis.
- **Trouver des indices** qui peuvent innocenter ou incriminer certains suspects.
- **Analyser les preuves** (par exemple : comparer des empreintes ou des objets).
- À tout moment, le joueur peut accuser un suspect. Mais attention, une accusation erronée met fin à la partie.

3. Objectif principal :
- Identifier correctement le coupable en rassemblant suffisamment de preuves pour justifier votre accusation.

Architecture MVC :

1. Modèle (Model) :
- Gestion de la logique du jeu :
- Générer les suspects, leurs alibis, mobiles et rôles.
- Stocker les indices collectés par le joueur.
- Mettre à jour la progression de l'enquête.
- **Exemple de tables MySQL :**

Table suspects :

id	**name**	**alibi**	**mobile**	**is_guilty**
1	Victor	"J'étais à la cuisine"	"Héritage"	1
2	Élise	"Je lisais au salon"	"Conflit financier"	0

Table rooms :

id	**name**	**description**
1	Salon	Une pièce élégante
2	Bibliothèque	Remplie de vieux livres

Table clues :

id	description	room_id	suspect_id
1	Un couteau ensanglanté	1	NULL
2	Une lettre de menace	2	1

- **Exemple de méthodes PHP dans le modèle :**
  - generateGame() : Initialise une nouvelle partie avec des suspects et des indices aléatoires.
  - getRoomDetails($roomId) : Récupère les informations d'une pièce (suspects et indices).
  - accuseSuspect($suspectId) : Vérifie si le suspect accusé est le coupable.

2. Vue (View) :

- Affichage des différentes scènes du jeu :
- Une page pour chaque pièce avec les actions disponibles.
- Une page pour interroger les suspects (affichage des dialogues).
- Une page de résumé pour consulter les indices collectés.
- Une page de victoire ou d'échec après une accusation.
- **Exemple d'éléments dynamiques :**
  - Génération des options de dialogue en fonction du suspect interrogé.
  - Affichage des indices trouvés dans la pièce visitée.

3. Contrôleur (Controller) :

- Gestion des actions du joueur :
- Navigation entre les pièces du manoir.
- Gestion des choix (interroger, analyser, accuser).
- Mise à jour des informations en fonction des décisions.
- **Exemple de contrôleurs :**
  - GameController :
  - Méthode startGame() : Initialise une nouvelle partie.
  - Méthode exploreRoom($roomId) : Affiche les détails de la pièce visitée.
  - Méthode interrogateSuspect($suspectId) : Gère les dialogues avec un suspect.
  - ClueController :
  - Méthode findClue($clueId) : Ajoute un indice à l'inventaire du joueur.

Fonctionnalités à implémenter :

Phase 1 : Fonctionnalités de base

- Génération aléatoire des suspects et du coupable.
- Exploration des pièces et collecte d'indices.
- Interrogatoire des suspects avec des réponses fixes.

Phase 2 : Améliorations

- Dialogues interactifs (choix multiples dans les questions).
- Ajout de mini-jeux pour analyser certains indices.
- Suivi des erreurs ou des incohérences dans les alibis.

Phase 3 : Extensions

- Gestion de plusieurs niveaux de difficulté (plus de suspects, indices moins évidents).
- Rejouabilité avec des scénarios générés aléatoirement.
- Système de score basé sur la rapidité et l'efficacité de l'enquête.

Options :

Si vous le souhaitez vous pouvez implémenter les éléments suivants

- **Gestion de l'inventaire :** Ajouter une interface pour voir et analyser les indices collectés.
- **Narration dynamique :** Ajouter des événements aléatoires qui influencent le déroulement de l'enquête (par exemple, un suspect qui ment ou fuit).