**Gautier CATTANEO**

**DEV QUIZ**

Une image contenant symbole, Graphique, Bleu électrique, capture d’écran

Description générée automatiquement



68350 BRUNSTATT-DIDENHEIM

10 avenue de Bruxelles  
68350 BRUNSTATT-DIDENHEIM

**DOSSIER DE SYNTHESE**

Titre Développeur web et web mobile

15 janvier 2024

Table des matières

[1 Introduction 4](#_Toc151995921)

[1.1 Présentation du projet 4](#_Toc151995922)

[1.2 Description du projet 4](#_Toc151995923)

[2 Conformité au RGPD 6](#_Toc151995924)

[2.1 Présentation du RGPD et son importance dans le projet 6](#_Toc151995925)

[2.1.1 Réglementation Général(européen) sur la protection des données (en anglais General Data Protection Regulation ou GDPR) 6](#_Toc151995926)

[2.1.2 Mon projet et le Rgpd 7](#_Toc151995927)

[3 Conception et Gestion de Projet 7](#_Toc151995928)

[3.1 Méthode Agile 7](#_Toc151995929)

[3.1.1 Explication détaillée de la méthode MoSCoW pour la gestion des exigences 8](#_Toc151995930)

[La méthode MoSCoW est une alternative de choix en matière de priorisation agile des éléments à aborder, des taches à effectuer. Diviser en 4 niveaux : 8](#_Toc151995931)

[3.2 Utilisation de Trello pour la gestion des tâches 9](#_Toc151995932)

[3.3 Schéma du processus de développement 9](#_Toc151995933)

[4 Modélisation de la Base de Données 9](#_Toc151995934)

[4.1 Modèle Conceptuel de Données (MCD) 9](#_Toc151995935)

[4.2 Modèle Logique de Données (MLD) 11](#_Toc151995936)

[4.3 Justification des choix de conception de la base de données 12](#_Toc151995937)

[5 Conception Graphique 12](#_Toc151995938)

[5.1 Wireframe pour l’interface utilisateur 12](#_Toc151995939)

[5.2 Maquettes réalisées avec Figma 12](#_Toc151995940)

[6 Réalisation du Projet 12](#_Toc151995941)

[6.1 Présentation détaillée de la mise en œuvre de chaque technologie 12](#_Toc151995942)

[6.1.1 **Html 5 :** 12](#_Toc151995943)

[6.1.2 **CSS 3 :** 12](#_Toc151995944)

[6.1.3 **Looping :** 12](#_Toc151995945)

[6.1.4 **Visual Studio Code :** 12](#_Toc151995946)

[6.1.5 **Javascript :** 13](#_Toc151995947)

[6.1.6 **PHP 8 :** 13](#_Toc151995948)

[6.1.7 **JQuery :** 13](#_Toc151995949)

[6.1.8 **Heidi SQL :** 13](#_Toc151995950)

[6.1.9 **GitHub et Git :** 13](#_Toc151995951)

[6.1.10 **Composer :** 14](#_Toc151995952)

[6.1.11 **Symfony CLI :** 14](#_Toc151995953)

[6.1.12 **Twig :** 14](#_Toc151995954)

[6.2 Explication du rôle de Symfony dans le développement 14](#_Toc151995955)

[6.2.1 HyperText Transfert Protocol (http) 14](#_Toc151995956)

[6.2.2 Les avantages de Symfony 15](#_Toc151995957)

[6.3 Gestion de la sécurité (faille XSS, faille upload /mime type, injection SQL) 20](#_Toc151995958)

[6.3.1 Faille xss 20](#_Toc151995959)

[6.3.2 Injection SQL 21](#_Toc151995960)

# Introduction

## Présentation du projet

J’ai choisi de réaliser pour mon projet une web -app sur les quiz dans le thème du développement web, ce choix a été fait dans le but de me lancer un défi personnel en associant les quiz que j’aime beaucoup et en même temps de tester mes connaissances sur mes apprentissages tous au long de la formation, devant réfléchir à la réalisation des quiz, de leurs questions et de leurs réponses, mais aussi de pouvoir servir de support de révision à de futurs apprenants qui feraient la même formation que moi ou qui souhaiterais découvrir les bases du développement web.

## Description du projet

Pour la réalisation de ce projet j’ai beaucoup réfléchi à comment mettre en place une application qui soit simple d’utilisation.

Pour cela, j’ai pris le choix de limiter une grande partie de l’accessibilité de l’application seulement à l’utilisateur enregistré.

**Un visiteur pourra :**

* S’inscrire.
* Voir la liste des quiz qui sont disponibles.
* Se connecter
* Voir la liste des catégories
* Le pourcentage moyen de réussite au quiz.

**Un utilisateur**

* Une fois enregistré, il faudra qu’il valide son email avant de pouvoir se connecter.
* Une fois connecté, il aura accès à tous les quiz disponibles et qui auront été validés par la modération.
* Accéder à son profil
* Peux modifier son profil, email et pseudo.
* Un quiz une fois joué sera indisponible durant 7 jours.
* Ils pourront créer et modifier leurs propres quiz qui seront soumis à vérification avant d’être jouables par les autres utilisateurs.
* Aura une page de profil avec un récapitulatif de ses parties jouées.
* Un quiz sera considéré comme réussi à partir de 80 % de bonnes réponses, mais restera accessible et rejouable dans les mêmes conditions que les autres.
* Un maximum de 5 parties par quiz sera enregistré en base de données, au-delà le score de la 6ème partie sera comparé avec le score minimum obtenu sur les anciennes et ne sera enregistré que s’il est supérieur, sinon seule la date de la partie sera mise à jour.

**Modérateur**

* À tous les droits d’un utilisateur
* Peux modifier, créer et valider un quiz, question et réponse.
* Peux créer et modifier une catégorie.
* Peux créer et modifier un thème
* Peux jouer un quiz sans limitation de durée

**Administrateur**

* A tous les droits du modérateur
* Peux modifier le pseudo d’un utilisateur
* Peux attribuer et enlever le rôle de modérateur à un utilisateur
* Peux bannir un modérateur/utilisateur

# Conformité au RGPD

## Présentation du RGPD et son importance dans le projet

### Réglementation Générale(européenne) sur la protection des données (en anglais General Data Protection Regulation ou GDPR)

Signée en avril 2016 et applicable depuis le 5 mai 2018, 173 considérants, 11 chapitres pour 99 articles. Le RGPD encadre le traitement des données personnelles sur le territoire de l’Union Européenne.

#### Pour qui ?

Toute organisation public et privée, qui traite des données personnelles pour son compte ou non dès lors :

* Qu’elle est établie sur le territoire de l’Union Européenne.
* Ou que son activité cible directement des résidents européens.

#### Qu’est-ce qu’une donnée personnelle ?

Une donnée personnelle est une information se rapportant à une personne physique identifiée ou identifiable :

* Directement (nom, prénom)

Ou indirectement (identifiant, n° client, n° de téléphone), donnée biométrique mais aussi la voix et l’image.

* À partir d’une seule donnée (ex : nom)
* À partir du croisement d’un ensemble de données (une femme vivant à telle adresse, née tel jour, abonnée à tel opérateur…)

#### Qu’est-ce que le traitement de données personnelles ?

Une ou un ensemble d’opérations portant sur les données personnelles, quel que soit le procédé utilisé (collecte, enregistrement, conservation, adaptation, modification, extraction, consultation, utilisation, communication par transmission diffusion), les fichiers papier sont également concernés et doivent être protégés dans les mêmes conditions.

Un traitement de données doit avoir un objectif : on ne demande pas une adresse postale si on n’en a aucune utilité (envoi de courrier, livraison …).

#### Mon projet et le RGPD

###### Minimisation des données

Pour respecter le RGPD et la Minimisation des données qui consiste à ne demander que les données nécessaires

N’ayant aucun système de paiement, ni besoin d’une identité réelle, ou d’information spécifique, je ne demande à un utilisateur souhaitant s’enregistrer que :

* 1 email
* 1 pseudo
* 1 mot de passe (haché)

Je ne demande aucune donnée sensible**.**

##### Finalités déterminées

Pour respecter la transparence je collecte ses données pour ses raison

1. **Email :**
   * Envoyer des notifications importantes concernant le compte (par exemple, réinitialisation du mot de passe, confirmation d'inscription).
   * Communiquer des mises à jour sur les services ou produits, mais seulement si l'utilisateur a explicitement donné son consentement pour recevoir de telles communications.
   * Faciliter la récupération du compte en cas de perte d'accès.
2. **Pseudo :**
   * Identifier l'utilisateur dans le système.
   * Affichage public (si applicable) pour permettre aux autres utilisateurs de vous reconnaître dans le cadre de l'utilisation du service.
3. **Mot de passe haché :**
   * Assurer la sécurité de l'accès au compte de l'utilisateur.
   * Vérifier l'identité lors de la connexion.

#### Droit à l'oublie

Toute personne enregistrée sur mon site bénéficie du droit a l'oublie et peux demander la suppression d'une information ou la totalité de son compte qui le concerne.

#### Droit d'accès

Conformément aux recommandations de la CNIL toute Utilisateurs enregistrés aura le droit d'accéder à ses données sur simple demande et devras les récupérer dans un format compréhensible. De ce faite un mail de contact est accessible sur le site.

Consentement (case à cocher)

A l'inscription une case a coché est disponible pour accepter les termes et condition avec un lien qui redirige vers ces termes.

# Conception et Gestion de Projet

## Méthode Agile

J’ai décidé de mettre en place une méthode de gestion de projet pour travailler efficacement, j'ai choisi la méthode Agile est une méthode de gestion de projet qui consiste à décomposer vos projets en une suite de petits objectifs atteignables.

Cette approche a été développée dans les années 2000 par des personnes qui n’étaient pas satisfaites des méthodes de gestion de l’époque jugées trop lourdes, lentes et contraignantes.

En mode agile, vous travaillez sur des cycles courts que l’on appelle **sprint** ou **itérations** (durent en général entre 1 semaine et 1 mois) contrairement à la méthode type diagramme de Gantt (plan sur 12 à 24 mois).

Le manifeste Agile s’axe autour de 4 valeurs :

* Les individus et leurs interactions plutôt que les processus et les outils
* Des logiciels opérationnels plutôt qu’une documentation exhaustive
* La collaboration avec les clients plutôt que la négociation contractuelle
* L’adaptation au changement plutôt que le suivi d’un plan

### Explication détaillée de la méthode MoSCoW pour la gestion des priorités

Il m’a été important de prioriser les tâches afin de m’organiser. Pour cela j’ai utilisé la méthode MOSCOW.

La méthode MoSCoW est une alternative de choix en matière de priorisation agile des éléments à aborder, des tâches à effectuer. Divisé en 4 niveaux :

* Must Have (Vital):

Le M de MoSCoW. Les tâches impératives à la réussite du projet.

Elles sont cruciales. Sans ces fonctionnalités, pas de produit final.

Elles seront réalisées en premier.

* Should Have (Essentiel):

Le S. représente des tâches importantes mais pas vitales, elles seront réalisées une fois le must have terminer. Elles sont constituées de vraies valeurs ajoutées au produit final.

Exemple : des bugs mineurs,

Amélioration de performance.

Amélioration de certaines fonctionnalités.

* Could have (confort) :

Le C. Les tâches dites de fonctionnalité de confort ne pouvant être réalisées que si les deux étapes précédentes ont été réalisées.

Un coût moindre, un design plus sophistiqué.

* Wont Have (luxe) :

Les fonctionnalités futures.

Initiative pouvant être intéressante, mais longue

Cout budgétaire plus élevé.

## Utilisation de Trello pour la gestion des tâches

J’ai utilisé Trello pour appliquer la méthode MoSCoW.

Trello est une plateforme qui permet de créer des tableaux pour un projet et d’y ajouter des cartes correspondant à chaque colonne définie comme Must Have, Should Have, Could Have, Wont Have, En cours et Terminé.

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

# Modélisation de la Base de Données

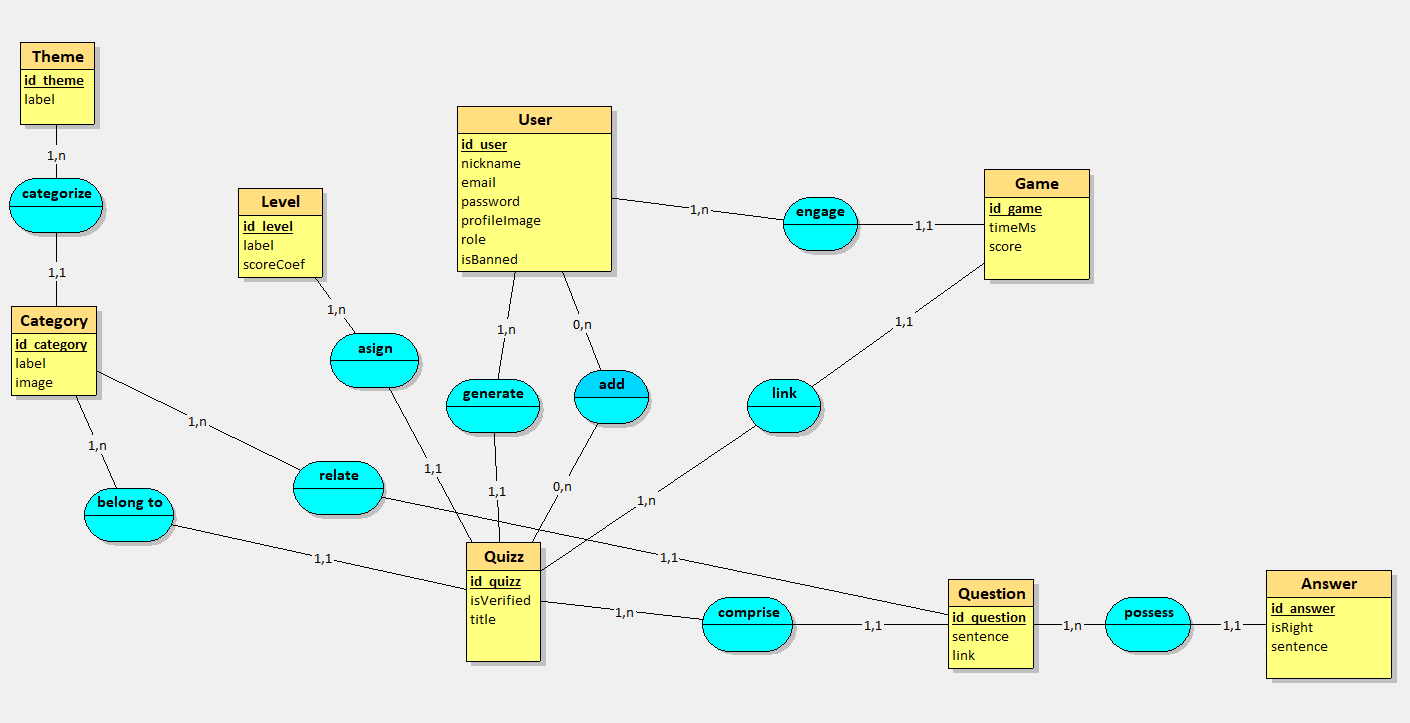
Après avoir décidé de mes fonctionnalités et organisé mon travail en priorisant les tâches, j’ai décidé de modéliser mon système d’information.

Pour la modélisation de mes données qui est une représentation visuelle de la structure de comment mes données seront organisées. Pour se faire j’ai utilisé la méthode Merise.

Merise (Méthode d’étude et de réalisation informatique pour les systèmes d’entreprise) est une méthodologie de modélisation dans le domaine du développement de système d’information en utilisant des outils graphiques pour représenter les différentes étapes de conceptions pour faciliter la compréhension et la communication entre tous les acteurs d’un projet.

## Modèle Conceptuel de Données (MCD)

Le MCD (Modèle conceptuel des données) est essentiel pour la conception d’un site Web, permettant de visualiser schématique les entités et leurs propriétés, formant ainsi les classes et les tables de notre base de données de manière simple et visuelle. Ces modélisations peuvent être réalisées via des logiciels tels que Looping (utilisé dans mon projet), mais il en existe d’autres outils, comme GitMind ou Visual Paradigm.



**Une entité** :

Prenons l'exemple de l'entité 'User' dans mon projet.

Un 'User' représente une personne préalablement enregistrée sur le site web.

Voyons en détail ses **propriétés**:

* Id\_user : Identifiant unique garantissant son unicité, appelé propriété identifiante.
* Username : Pseudo affiché dans les classements des différents quiz auxquels il a participé et sur sa page de profil.
* Email : Identifiant de connexion, obligatoire et unique, devant être vérifié via un lien envoyé par e-mail après inscription.
* Password : Mot de passe nécessaire pour se connecter, devant comporter au minimum de 12 caractères avec au moins 1 lettre majuscule, 1 lettre minuscule, 1 caractère spécial, et 1 chiffre pour être valide.
* Rôle : Rôle attribué par défaut à l'utilisateur, initialement défini en tant que 'User.
* IsBanned : Un booléen indiquant le statut banni ou non, en cas de non-respect des règles du site. Seul l'admin peut bannir un utilisateur.

Les relations entre les entités sont établies grâces aux cardinalités. Par exemple,

'User' est relié à :

* Quiz : un User peut créer un ou plusieurs Quiz (cardinalité 1 à n) tandis qu'un Quiz est créé par un seul et un seul User (cardinalité 1 à 1).

Game : un 'User' peut jouer zéro ou plusieurs 'Game' (cardinalité 0 à n) et chaque Game est lier à un seul 'User' (cardinalité 1 à 1)

Une fois toutes nos entités et leur propriété établies sur le MCD, je suis ensuite passé à une phase plus détaillée de modélisations le MLD.

## Modèle Logique de Données (MLD)

Une image contenant texte, diagramme, capture d’écran, Plan

Description générée automatiquement

Le MLD (Modèle Logique des Données) décrit de manière plus détaillée la structure des données par rapport au MCD, en précisant le type des données utilisées.

Les entités du MCD se transforment en tables dans le MLD.

La propriété identifiante devient une clé primaire, qui est un attribut ou un ensemble garantissant l'unicité de chaque enregistrement dans une table.

Dans le MLD, apparaissent également des clés étrangères, résultant des choix des cardinalités effectués dans le MCD.

Les clés étrangères font référence aux clés primaires d'autre tables, assurant ainsi l'intégrité référentielle entre les données de différentes tables.

Dans mon MLD, on remarque l'apparition de l'entité "Favorites", résultant de la relation Many-To-Many. Cette entité est une table associative contenant les clés étrangères du quiz et de l'utilisateur.

# Conception Graphique

En tant que débutant dans le domaine du développement web et web mobile, mon premier projet personnel a été une opportunité précieuse pour plonger dans la phase de conception graphique. Cette étape s'avère essentielle pour concrétiser mes idées et mettre en place l'expérience utilisateur (UX) et l'interface utilisateur (UI) que j'ai imaginées.

## Wireframe pour l’interface utilisateur

Les wireframes ont marqué mes premiers pas dans la conception de mon site web. Leur simplicité m'a permis de poser les bases, de structurer les pages sans être distrait par les détails visuels. En les utilisant, j'ai pu me concentrer sur l'ergonomie et la navigation, apprenant ainsi à organiser efficacement les éléments pour offrir une expérience utilisateur fluide et intuitive. (j'ai utilisé Figma pour mon projet.)

## Maquettes réalisées avec Figma

Au cours de ce processus, j'ai rencontré des défis, notamment lors de la création de la maquette pour la page d'accueil. Cela m'a conduit à retravailler la maquette de la page d'accueil à plusieurs reprises, explorant différentes itérations pour trouver l'équilibre parfait entre esthétique et fonctionnalité. Cette expérience m'a appris l'importance de la persévérance, de la créativité et de l'itération continue pour aboutir à un résultat final répondant à mes attentes et aux besoins des utilisateurs.

## **Responsive Design**

Le Responsive Design, intégré dans la conception de mon projet, revêt une importance capitale pour offrir une expérience utilisateur optimale sur une gamme varié d'appareil

### Introduction au Responsive Design :

Le Responsive Design est une approche de conception web qui garantit que les sites web s'affichent de manière optimale sur une variété d'appareils et de tailles d'écran, en s'adaptant dynamiquement à la résolution de l'appareil utilisé.

C'est une approche fondamentale pour offrir une expérience utilisateur homogène, quel que soit l'appareil utilisé.

### Principes clés du Responsive Design :

1. **Conception flexible :** Utilisation de grilles flexibles, images et contenus fluides pour s'adapter à différentes tailles d'écran.
2. **Media Queries :** Les Media Queries en CSS sont essentielles pour définir des styles spécifiques en fonction des caractéristiques de l'appareil, permettant ainsi de personnaliser l'affichage en fonction de la taille de l'écran.
3. **Priorisation du contenu :** Mettre en avant le contenu essentiel et le rendre accessible sur tous les appareils.
4. **Tests multi-appareils :** Vérifiez régulièrement l'aspect et la convivialité de votre site sur différents appareils pour garantir une expérience utilisateur optimale.

# SEO (Optimisation pour les moteurs de recherche)

Le SEO, connu sous le nom de Search Engine Optimization, est un ensemble de techniques visant à améliorer la visibilité d'un site web dans les résultats des moteurs de recherche tels que Google, Bing, et d'autres. Un bon référencement SEO est essentiel pour attirer du trafic organique et accroître la présence en ligne de mon projet.

## Introduction au SEO :

Le SEO vise à optimiser mon site web pour répondre aux critères de pertinence et de qualité des moteurs de recherche. Son objectif principal est de positionner mon site parmi les premiers résultats de recherche pour des requêtes spécifiques, ce qui accroît la visibilité et le nombre de visiteurs.

## Points clés du SEO :

1. **Recherche de mots-clés :** L'identification de mots-clés pertinents est cruciale. Utilisez des outils de recherche de mots-clés pour trouver les termes les plus adaptés à mon contenu et intégrez-les naturellement dans mes pages.
2. **Contenu de qualité :** La création de contenu pertinent et informatif est primordiale. Les moteurs de recherche favorisent le contenu original, actualisé et pertinent pour répondre aux besoins des utilisateurs.
3. **Optimisation technique :** S'assurer que mon site est techniquement optimisé pour les moteurs de recherche. Cela implique l'amélioration de la vitesse de chargement, la structure des URL, les balises HTML, etc.
4. **Adaptation mobile :** Un site mobile-friendly est désormais incontournable. Les moteurs de recherche privilégient les sites adaptés aux appareils mobiles pour offrir une expérience utilisateur optimale.

# Réalisation du Projet

## Présentation détaillée de la mise en œuvre de chaque technologie

### **Une image contenant Graphique, orange, rouge, conception Description générée automatiquement HTML 5 :**

Le Html HyperText Markup Language, est un langage de balisage utilisé pour créer et structurer le contenu d’une page web. Il s’exécute côté client.

### **Une image contenant Bleu électrique, symbole, Graphique, Rectangle Description générée automatiquementCSS 3 :**

Cascading Style Sheets ou feuilles de style en cascade est un langage utilisé pour formater et styliser une page Web, incluant les couleurs, tailles de texte, positionnement des éléments et fonds. Il est associé directement aux balises HTML.

Les sélecteur CSS permettent de formater les éléments HTML. Il s’exécute côté client.

### **Une image contenant Police, logo, Graphique, symbole Description générée automatiquement** **Looping :**

Looping est un logiciel de modélisation conceptuelle des données utilisé pour réaliser mon MCD/ MLD et schématiser la structure de ma base de données.

### **Une image contenant symbole, Bleu électrique, conception Description générée automatiquement Visual Studio Code :**

Visual Studio Code est un éditeur de code avec intégration Git simplifiée et une vaste bibliothèque d’extension adaptée à tous les langages.

J'ai utilisé exclusivement ce logiciel pour développer le back-end et le front-end de mon apllication.

### **Une image contenant logo, symbole, jaune, conception Description générée automatiquementJavascript :**

Javascript est un langage de script léger utilisé pour des éléments dynamiques de mon application, tels qu'un menu dynamique en version mobile, et a été utilisé pour réaliser ma page 'play\_quiz'.

Détaillé plus loin.

### **Une image contenant logo, Police, Graphique, symbole Description générée automatiquement PHP 8 :**

PHP (HyperText Preprocessor) est un langage de script généraliste et Open source, conçu pour le développement d’applications Web. Il peut être facilement intégré en HTML et s’exécute côté serveur, générant ainsi le HTML envoyé au client.

### **Une image contenant logo, Graphique, Police, graphisme Description générée automatiquement JQuery :**

Jquery a été utiliser pour gérer les CollectionType de Symfony et rendre les formulaires dynamiques dans la création du quiz, ses questions et ses réponses.

### **Une image contenant logo, Graphique, Police, symbole Description générée automatiquement Heidi SQL :**

Heidi SQL est un outil d’administration de base de données avec un éditeur SQL et un constructeur de requête.

J'ai travaillé avec ce logiciel optimisé pour le SGBD (Système de gestion de base de données) relationnel MySQL.

### **Une image contenant clipart, Graphique, conception Description générée automatiquement GitHub et Git :**

GitHub est un service de cloud et un site Web pour stocker et gérer son code, ainsi que suivre les modifications.

Git, un logiciel de gestion de versions, permet de créer différentes versions de l'application et de conserver un historique du code en créant de nouvelles branches.

Tous les fichiers sont stockées sur GitHub permettant la comparaison, la fusion et le retour en arrière en cas de besoin.

### **Une image contenant dessin, croquis, clipart, illustration Description générée automatiquementComposer :**

Composer est un outil permettant de gérer les dépendances des projets PHP.

### **Une image contenant Graphique, symbole, clipart, cercle Description générée automatiquement Symfony CLI :**

Est un serveur local permettant d'exécuter un projet sur son propre ordinateur.

### **Une image contenant plante, fleur Description générée automatiquement Twig :**

Twig est le moteur de template par défaut sur Symfony, facilitant l'interface entre le front et le back-end ainsi que l’utilisation de logique au sein du HTML.

## Explication du rôle de Symfony dans le développement

J'ai opté pour l'utilisation d'un Framework.

Un Framework est une boite à outils pour les développeurs web, contenant des composants autonomes facilitant le développement d'un site web ou d'une application. Le mot 'Framework' est une combinaison des termes 'Frame' qui signifie cadre et 'Work' se traduit par travail.

Pour faire fonctionner Symfony, l'installation d'une version de PHP, de Composer et éventuellement de Symfony CLI est nécessaire.

Mon projet est basé sur Symfony 6.3, un Framework PHP qui assure une structuration optimale de l’application en utilisant le design pattern MVC (Modèle-Vue-Contrôleur).

### MVC

Le MVC est une architecture logicielle utilisée pour séparer la logique de présentation de l’interface utilisateur de la logique de gestion des données.

La couche **modèle** représente les données et les règles métier.

La **vue** gère l’affichage de ces données sur l’écran du client. Dans Symfony, l’affichage est géré par Twig

La couche **contrôleur**, comme son nom l’indique, est la tour de contrôle de l’application. Il gère les actions des utilisateurs et appelle les bonnes fonctions pour renvoyer les informations adéquates à la vue.

Schéma représentant le MVC sous Symfony

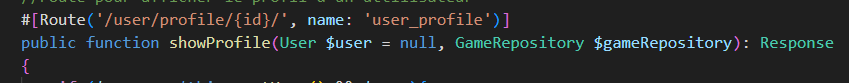
Une image contenant texte, capture d’écran, diagramme, conception

Description générée automatiquement

Lorsqu'un utilisateur clic sur une URL, cela envoie une requête HTTP au serveur, qui est traitée par le service de routing.



Ce service, en utilisant les informations de l'URL, appelle le bon contrôleur.



Le contrôleur, si nécessaire, fait appelle à la couche modèle sous Symfony, le repository

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Et renvoie les données nécessaires à la vue.

Une image contenant texte, Police, capture d’écran

Description générée automatiquement



Une fois les données traitées et transmises, elles sont affichées dans le navigateur(la réponse HTTP)

Une image contenant texte, logiciel, Police, Bleu électrique

Description générée automatiquement

### HyperText Transfert Protocol (HTTP)

Pour comprendre le design pattern il m'a tout d'abord fallu saisir son fonctionnement.

Tous commence par une requête HTTP.

La requête HTTP est un message du client au serveur contenant un URL et un verbe HTTP (GET, POST, PUT, DELETE) + un document (Get layout, GET image.png …) + le type du navigateur et sa version. En retour, le serveur répond avec du HTML et un code réponse (200 pour succès, 404 pour page non trouvée).



### Les avantages de Symfony

#### Les commandes :

Symfony propose plusieurs commandes qui automatisent certaines fonctions lors du développement. Ces commandes sont exécutées dans une fenêtre PowerShell, disponible sur les ordinateurs Windows en recherchant 'PowerShell' dans la barre de recherche. Personnellement, j'ai utilisé la console intégrée à Visual Studio Code.

#### Création d'un nouveau projet

La commande 'Symfony new NomDuProjet –webapp' permet de créer un dossier complet avec toutes les dépendances de base incluses.

#### Lancement du serveur

'Symfony server::start / start -d' permet de démarrer le serveur en arrière-plan tous en gardant la console ouverte.

#### Création ou mise à jour d'une Entity

La commande 'Symfony console make:entity (ou m:e)' facilite la création d'une défini dans le MCD. Cette commande guide étape par étape en demandant chaque propriété qui constituera l'objet, générant automatiquement ses propriétés, méthodes, getters et setters.

Pour créer une entité User avec des fonctionnalités de sécurité incluses, la commande '**Symfony console make:user'** est utilisée. Elle génère une entité User avec des propriétés spécifiques, telles que 'getUserIdentifier**()'** **,**les rôles, et la logique de vérification d'e-mail

#### **Création de formulaires**

La commande 'symfony console make:registration-form' permet de générer le formulaire d'enregistrement de l'utilisateur. Des modifications peuvent être apporter au niveaux du mot de passe, telles que l'ajout d'une regex exigeant un mot de passe contenant au moins 12 caractères avec une lettre majuscule, une lettre minuscule, un chiffre et un caractère spécial.

                'constraints' => [

**new** Regex([

                    'pattern' => '/^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[@$!%\*?&])[A-Za-z\d@$!%\*?&]{12,64}$/',

                    'message' => 'Votre mot de passe doit contenir au minimum 12 caractères dont 1 lettre majuscule, 1 lettre minuscule, 1 chiffre et 1 caractère spécial.'])

                ],

##### Reset password

Pour gérer la réinitialisation du mot de passe en cas d'oubli, le bundle 'reset password' est installé à l'aide de la commande 'composer require symfonycasts/reset-password-bundle'. Cette fonctionnalité permet à l'utilisateur de demander un lien de réinitialisation sur son adresse e-mail enregistrée.

Lorsque l'utilisateur clique sur le lien 'mot de passe oublié', 

Il accède à un formulaire pour saisir son adresse e-mail.

Si cette adresse l'email est enregistré dans la base de données, un token unique avec une durée de vie de 1h est généré. Ce token est alors stocké dans la session de l'utilisateur lié à son adresse e-mail et est inclus dans le lien de réinitialisation envoyé dans l'email.

Lors de la validation du formulaire de réinitialisation, le token stocké en session est comparé au token utilisé pour la réinitialisation. Le mot de passe ne sera modifié que si ces deux tokens correspondent garantissant ainsi la sécurité du processus de réinitialisation du mot de passe.

## Gestion de la sécurité (faille XSS, faille upload /mime type, injection SQL)

### Faille xss

La faille xss ou **Cross-site Scripting** est une **injection de code** malveillant, souvent en JavaScript **dans le contenu des sites Web**. Elle peut se produire dans diverses parties d’une application, telles que :

* Espace commentaire
* Barre de recherche
* Formulaire
* Tous autres champs ou des données peuvent être écrites par un utilisateur.

**Risques associés** :

Ces scripts malveillants peuvent **récupérer les cookies de session** ou d’autres informations stockées dans le navigateur, pouvant être utilisés pour des attaques telles que le phishing ou le vol de données.

**Méthodes de protection :**

* Utilisation des fonctions PHP comme `**htmlspecialchars`** et `**htmlentities`** pour échapper les caractères spéciaux en HTML et éviter l’exécution de code :
* **Htmlspecialchars** remplace les caractères qui ont une signification spéciale en HTML par leurs équivalents dans la chaine string.
* `**htmlentities`** remplace tous les caractères par leurs équivalents en HTML

Utilisation des filtres de validation côté serveur, tels que :

* + **FILTER\_SANITIZE\_FULL\_SPECIAL\_CHARS**: Convertit les caractères spéciaux en entités HTML
  + **FILTER\_SANITIZE\_NUMBER\_INT** : Supprime tous les caractères sauf les chiffres, et les signe ‘**+**’ et ‘**-**’.
* Symfony fournit des outils de validation des données côté serveur pour garantir un format spécifique des données entrantes, réduisant les risques d’injections malveillantes.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Tous une liste de Type qui serviront à filtrée les données entrantes.

* Le moteur de template de Symfony, Twig, effectue un échappement automatique des données affichées dans les Templates. Toutes les données affichées sont échappées par défaut, ce qui prévient l'exécution involontaire de code JavaScript malveillant.

### Injection SQL

L’injection SQL est une faille de sécurité qui survient lorsque des données non filtrées ou mal échappées sont incluses dans les requêtes SQL de manière incorrecte. Le code SQL malveillant est injecté via des champs de saisie comme les formulaires, et les paramètres d’URL, ou d’autres entrées utilisateurs.

Risque :

Une telle vulnérabilité permettrait à un utilisateur malveillant de récupérer toutes les données enregistrées en base de données.

**Protection** :

* Validation et filtrage des entrées utilisateur :

Validez et filtrez soigneusement toutes les données entrantes pour vous assurer qu’elles correspondent au format et au type attendus.

* Utiliser des requêtes préparées et des paramètres liés :

**Exemple :**

Les requêtes préparées sont des requêtes qui vont être créées en trois temps : la préparation, la compilation et l’exécution.

La préparation : Création d'un Template de la requête sans préciser les valeur réelles mais en utilisant des marqueur nommés (exemple :title ) ou des marqueur interrogatif ( ? ). Ces marqueurs seront remplacés par les vrais valeur lors de l'exécution de la requête.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

### Faille upload/Mime Type

La faille upload est une faille de sécurité courante dans les applications Web ou les utilisateurs peuvent télécharger des fichiers sur un serveur.

**Risque encouru :**

* **Téléchargement de fichier malveillant :**

Il est important de vérifier le type du fichier uploadé par l’utilisateur, car il pourrait contenir du code malveillant ou un script PHP. Il suffirait ensuite au pirate d’appeler son fichier pour que celui-ci s’exécute. (Exemple : les webshells permettent de prendre le contrôle du serveur)

* **Contournement de l’extension de fichier :**

Les pirates peuvent renommer leurs fichiers malveillants avec une extension de fichier autorisée pour contourner les restrictions ou avec une double extension.

Pour prévenir de telles attaques plusieurs mesures de protection sont mises en place:

**Protection** :

* Restriction des types de fichiers acceptés :

Vérification de l’extension du fichier et de son type MIME (Multipurpose Internet Mail Extensions) qui est une étiquette utilisée pour identifier le type de contenu d'un fichier. Il indique le format ou le type de données contenu dans le fichier.

->add('picture', FileType::class,[

                'label' => 'Image de la catégorie (jpeg, png, jpg)',

                'mapped' => false,

                'required' => false,

                'constraints' => [

**new** File([

                        'maxSize' => '1024k',

                        'mimeTypes' => [

                            'image/jpeg',

                            'image/jpg',

                            'image/png'

                        ],

                        'mimeTypesMessage' => "Veuillez télécharger une image au format valide s'il vous plait",

                    ])

                ],

            ])

* **Renommer le fichier uploadé:**

Pour éviter la manipulation de l’extension de fichier, les fichier sont renommés à l'aide de noms uniques générés par **'uniqid()'.** Cette méthode assure l'unicité des noms de fichiers et limite les risques de collision.

**public** **function** upload(UploadedFile $file): string

    {

        $originalFilename = pathinfo($file->getClientOriginalName(), PATHINFO\_FILENAME);

       Function Php unique(id)

        $fileName = uniqid().'.'.$file->guessExtension();

**try** {

            $file->move(**$this**->getTargetDirectory(), $fileName);

        } **catch** (FileException $e) {

        }

**return** $fileName;

    }

Les fichiers télécharger sont stocké dans un dossier séparer afin d'isoler les fichiers d'origine inconnu du reste de l'application, et garantir une meilleure organisation, ainsi qu'une sécurité renforcée.

### CRSF (Cross-Site Request Forgery)

Le CRSF est une méthode par laquelle un utilisateur malveillant tente de forcer un utilisateur légitime à soumettre sans le savoir des données ou des requêtes non désirées sur un site web.

Pour se protéger contre se type d'attaque, une protection CRSF est mise en place. Elle consiste à intégrer un champ masqué dans les formulaires contenant une valeur spécifique (token) que seul l'administrateur du site et l'utilisateur peuvent connaitre. Ce token garantit que seuls les utilisateurs autorisés peuvent soumettre des données, empêchant ainsi les requêtes malveillantes d'être exécutées.

Lorsqu'un utilisateur soumet un formulaire, le serveur vérifie ce token CSRF. Si le token est manquant ou incorrect, la requête est refusée, évitant ainsi toute soumission de données indésirables par des tiers malveillant.



### Attaque par Brute force/ Attaque par dictionnaire

Les attaques par Brute force consiste à tester méthodiquement toutes les combinaisons possibles d'un mot de passe, souvent en utilisant des ordinateurs capables d'essayer plusieurs centaines de milliers, voire des millions de combinaisons par seconde.

Les attaques par dictionnaire suivent un schéma similaire à celles par force Brute, mais elles utilisent une liste prédéfinie de mots, de chaînes de caractères ou de phrases, pour contourner la sécurité d'un système.

Pour sécuriser mon projet contre de telles attaques, j'ai mis en place des mesures limitant le nombre de tentatives de mot de passe incorrects à 3 par minute. En cas de dépassement, un délai d'attente de 10 min est imposé avant toute nouvelle tentative. De plus j'ai intégré une REGEX exigeant que les utilisateurs créer des mots de passe d'au moins 12 caractères comprenant au moins une minuscule, une majuscule, un chiffre et un caractère spécial. Ces critères de complexité augmentent considérablement le nombre de combinaison potentielles, renforçant ainsi la sécurité du système.

En plus des mesures de sécurité déjà mentionnées, j'ai mis en place Google reCAPTCHA dans sa version 3 pour renforcer davantage la sécurité de la plateforme. Contrairement aux versions précédentes qui nécessitaient une interaction de l'utilisateur, la version 3 opère de manière invisible pour les utilisateurs, analysant le comportement des visiteurs tout au long de leur session sur le site pour distinguer les humains des robots.

Cette intégration permet d'évaluer en arrière-plan le comportement des utilisateurs afin de déterminer leur légitimité, contribuant ainsi à réduire les risques liés aux activités malveillantes sans perturber l'expérience de navigation de nos utilisateurs authentiques.

# Fonctionnalité représentative

## Formulaire de création de Quiz

Une fonctionnalité qui ma demande un gros travail de recherche et de lecture de la documentation Officiel de Symfony pour la gestion des collectionType de plusieurs niveaux

<https://symfony.com/doc/current/reference/forms/types/collection.html>

<https://symfony.com/doc/6.3/form/form_collections.html#allowing-new-tags-with-the-prototype>

### Extrait de la documentation

Allowing "new" Tags with the "Prototype" Previously you added two tags to your task in the controller. Now let the users add as many tag forms as they need directly in the browser. This requires a bit of JavaScript code. But first, you need to let the form collection know that instead of exactly two, it will receive an unknown number of tags. Otherwise, you'll see a "This form should not contain extra fields" error. This is done with the allow\_add option:

Une des sources qui m’a beaucoup aidé a été trouvé sur un forum de discussion en ligne et du partage du gitHub de la personne qui ma envoyer vers la pite des écouteurs de mutation du Dom

Traduction :

Permettre l'ajout de "nouveaux" tags avec le "Prototype" Précédemment, vous avez ajouté deux tags à votre tâche dans le contrôleur. Maintenant, laissez les utilisateurs ajouter autant de formulaires de tags qu'ils le souhaitent directement dans le navigateur. Cela nécessite un peu de code JavaScript.

Mais tout d'abord, vous devez informer à la collection de formulaire qu'au lieu de recevoir exactement deux tags, elle recevra un nombre inconnu de tags. Sinon, vous verrez une erreur "Ce formulaire ne devrait pas contenir de champs supplémentaires". Cela se fait avec l'option allow\_add .

<https://openclassrooms.com/forum/sujet/formulaire-imbrique-3-niveaux-symfony-4>

<https://github.com/Ymox/blades/blob/master/public/js/entity_adder.js#L26>

Le formulaire de quiz QuizType qui contient la collection de question.

**public** **function** buildForm(FormBuilderInterface $builder, array $options): void

    {

        $builder

//…………….

            ->add('questions', CollectionType::class,[

                'entry\_type' => QuestionType::class,

                'label' => false,

                'prototype' => true,

Autorise l'ajout de nouveau élément qui seront persiter grâce au cascade persit sur l’élément Question ça va activer un data prototype qui sera un attribut html qu'on pourra manipuler en js.

                'allow\_add' => true, //autorise l'ajout

                'allow\_delete' => true, //autorise la suppression

                'by\_reference' => false,// il est obligatoire car Quiz n'a pas de setQuestion mais c'est Question qui contient setQuiz. Question est propriétaire de la relation. Pour éviter un mapping => false on est obligé de rajouter un by\_reference => false.

            ])

Le formulaire de Question (QuestionType ) qui contient lui la collection de Answer

**public** **function** buildForm(FormBuilderInterface $builder, array $options): void

    {

        $builder

            ->add('answers', CollectionType::class,[

                'entry\_type' => AnswerType::class,

                'prototype' => 'answers',

                'label' => 'Réponses',

                'allow\_add' => true, //autorise l'ajout

                'allow\_delete' => true, //autorise la suppression

                'by\_reference' => false

            ])

        ;

    }

Et enfin le formulaire de réponse AnswerType

**public** **function** buildForm(FormBuilderInterface $builder, array $options): void

    {

        $builder

        ->add('sentence', TextType::class,[

            'attr' =>[

                'class' => 'form-control',

                'label' => false

            ]

        ])

        ->add('isRight',ChoiceType::class,[

            'label' => false,

            'choices' => [

                'Bonne réponse' => '1',

                'Mauvaise réponse' => '0'

            ],

            'expanded' => true, // Affiche les réponses sous forme de checkboxes

            'multiple' => false, // Permet de n'en sélectionner qu'un seul

        ])

Leur affichage en Twig

{{ form\_start(formNewQuiz) }}

    {{ form\_row(formNewQuiz.title) }}

    {{ form\_row(formNewQuiz.level) }}

    <div class="formNewQuestion-group row ">

        <div class="col-formNewQuestion-label col-sm-2">&nbsp;</div>

Le prototype de question et de réponses .

        <div id="questions-fields-list" class="col-sm-10 remove-collection-widget"

        data-prototype="{{ form\_widget(formNewQuiz.questions.vars.prototype)|e }}"

        data-quiz="{{quizId}}"

        data-widget-tags="{{ '<p></p>'|e }}"

        data-widget-counter="{{ formNewQuiz.questions|length }}">

Boucle pour les questions.

            {% **for** questionField in formNewQuiz.questions %}

                <p>

                    {{ form\_errors(questionField) }}

                    {{ form\_widget(questionField, {'attr': {'class': 'borders'}}) }}

                </p>

Boucle pour les réponses

                {% **for** answerField in questionField.answers %}

                <p>

                    {{ form\_errors(answerField) }}

                    {{ form\_widget(answerField, {'attr': {'class': 'borders'}}) }}

                </p>

                {% **endfor** %}

            {% **endfor** %}

Le bouton pour ajouter les questions dynamiquement

    <div class="formNewQuestion-group row flex-action">

        <button type="button"

            class="add-another-collection-widget button"

            data-list-selector="#questions-fields-list">

            Ajouter une Question

        </button>

    </div>

{{ form\_end(formNewQuiz) }}

La gestion des ajouts dynamique en jquerry

Fonction déclenchée lors du clic sur le bouton ajouter question ajout d’un nouveau bloc "question" au sein d'un quiz

 $('.add-another-collection-widget').click(**function** (e) {

**let** list = $($(**this**).attr('data-list-selector'));

Récupération du nombre actuel d'élément de l'entity concerné dans la collection (à défaut, utilisation de la longueur de la collection)

**let** counter = list.data('widget-counter') || list.children().length;

Récupération de l'identifiant du quiz concernée, en cours de création/modification

**let** quiz = list.data('quiz');

Extraction du prototype complet du champ (que l'on va adapter ci-dessous)

**let** newWidget = list.attr('data-prototype');

Remplacement des séquences génériques "\_\_name\_\_" utilisées dans les parties "id" et "name" du prototype par un numéro unique au sein de la collection de "question" ou de "answer"  : ce numéro sera la valeur du compteur courant (équivalent à l'index du prochain champ, en cours d'ajout).

Finalement, l'attribut ressemblera à "quiz[questions][n°]"

        newWidget = newWidget.replace(/\_\_name\_\_/g, counter);

        newWidget = newWidget.replace(/><input type="hidden"/, ' class="borders"><input type="hidden" value="'+quiz+'"');

Incrémente le counter à chaque ajout de question

        counter++;

        list.data('widget-counter', counter);

**let** newElem = $(list.attr('data-widget-tags')).html(newWidget);

Ajout d'un bouton de suppression pour ce nouveau bloc de question

        addDeleteLink($(newElem).find('div.borders'));

        newElem.appendTo(list);

    });

Ensuite il ma fallu faire en sorte d’ajouter un bouton réponse de façon dynamiques à l’intérieur de ses nouveau bloc question.

Fonction qui va créer le bouton et lui ajouter un écouteur d’évènement au clic.

**function** addAnswerButton($element) {

Création d'un nouveau bouton "Ajouter une réponse"

**let** addAnswerButton = $('<button>', {

        'type': 'button',

        'class': 'add-answer-another-collection-widget',

        'text': 'Ajouter une réponse'

    });

Ajout du bouton nouvellement créé au container de question

    $element.append(addAnswerButton);

En observant la boite d’inspection du navigateur lors de l’ajout d’une nouvelle question j’ai récupéré l’élément qui m’intéresser pour en observer les modifications du DOM pour détecter l'ajout d'éléments dans la div avec questions-fields-list

**let** observer = **new** MutationObserver(**function**(mutationsList) {

        mutationsList.forEach(**function**(mutation) {

**if** (mutation.type === 'childList' && mutation.target.id === 'questions-fields-list') {

                $(mutation.addedNodes).each(**function**() {

**let** $addedElement = $(**this**);

                    addAnswerButton($addedElement);

Appel de la fonction addAnswerButton "Ajouter une réponse" pour chaque nouveau bloc de question créer

Options pour observer les modifications du DOM

**let** observerConfig = { childList: **true**, subtree: **true** };

* **chidList: true :**

Cette propriété indique à l’observateur de surveiller les modifications apportées à la liste des enfants de l’élément cible (ajout, suppression)

* **subtree :true :**

Observe aussi les modifications de tous les enfants de l’élément cible.

Commencer à observer les modifications sur la div avec questions-fields-list

observer.observe(document.getElementById('questions-fields-list'), observerConfig);

Ajout du bouton "Ajouter une réponse" pour les container des question déjà présents

$('#questions-fields-list > div').each(**function**() {

addAnswerButton($(**this**));

});

Et enfin j’ai dû ajouter la logique sur ce bouton pour ajouter des nouveaux bloc réponses sur chaque question

Ajout des écouteurs d’évènement sur le bouton ‘Ajouter réponses’

addAnswerButton.on('click', **function**() {

Récupération de l’élément question lié au bouton cliqué

**let** $question = $(**this**).prev('div[id^="quiz\_questions\_"]');

Récupération du conteneur des réponses associé à la question

**let** $answersContainer = $question.find('[id$="\_answers"]');

Récupération du prototype des réponses depuis les attributs du conteneur de réponses

**let** prototype = $answersContainer.attr('data-prototype');

Vérification si le prototype existe

**if** (**typeof** prototype !== 'undefined') {

Comptage du nombre actuel de réponses dans le conteneur

**let** counterAnswer = $answersContainer.children().length;

Vérification du nombre maximum de réponses (limite à 4 réponses)

**if** (counterAnswer < 4) {

Récupération de l'index de la question actuelle pour l’utiliser dans la création des id de réponse associer à chaque question

**let** questionIndex $question.attr('id').split('\_')[2];

Utilisation de RegExp pour construire dynamiquement l'expression régulière qui sera utilisé pour changer les id de réponses

**let** regex2 = **new** RegExp('quiz\_questions\_' + questionIndex + '\_answers\_' + questionIndex + '', 'g');

prefix créer ou le deuxième questionIndex sera remplacer par count dans le prototype de réponses pour avoir les bon id de réponses à chaque question

**let** prefix2 = ('quiz\_questions\_' + questionIndex + '\_answers\_' + count);

On remplace avec la nouvelle valeur

**let** newForm = prototype.replace(regex2, prefix2)

Second préfix créer pour remplacer le l’id de réponse dans l’attribut name du label de réponse

**let** prefix1 = 'quiz[questions][' + questionIndex + '][answers][' + count + ']';

On effectue le replace avec le second préfix

newForm = newForm.replace(/quiz\[questions\]\[\d+\]\[answers\]\[\d+\]/g, prefix1);

Ajout de la nouvelle réponse au conteneur

$answersContainer.append(newForm);

Appel de la fonction checkMinimumAnswer() qui vérifie que chaque container de question existant ou nouvellement créer contient bien chacun au minimum 2 réponse et qu’il y est au moins 10 questions si ce n’est pas le cas le bouton valider est désactiver

checkMinimumAnswers();

Sélection des boutons radio pour une question spécifique pour limiter le choix a une seule bonne réponse par question

$radioAnswer = $('[name^="quiz[questions]['+ questionIndex + '][answers]"][type="radio"]')

 Ajout d’un écouteur d’événement sur tous les bouton radio

$radioAnswer.on('change', **function**() {

Vérifie si la réponse sélectionnée est une bonne réponse

**if** ($(**this**).val() === '1') {

Si c'est une bonne réponse, coche toutes les autres réponses de la même question sur mauvaise

$('[name^="quiz[questions]['+questionIndex+'][answers]"][type="radio"][value="0"]').prop('checked', **true**);

 $(**this**).prop('checked', **true**);

  }

});

Incrémentation du compteur pour maintenir les identifiants uniques

count++;

} **else** {

**let** errorMessage = $('<p>', {'text': 'Limite de 4 réponses atteinte pour cette question.',

           'class': 'error-message'

     });

count = 0

      // Ajout du message d'erreur

$element.append(errorMessage);

}

} **else** {

console.error("La valeur de prototype est undefined");

            }

        });

La fonction qui check qu’il y a au minimum 10 question avec au moins 2 réponses

**function** checkMinimumAnswers() {

Déclaration de la variable a true par défaut

**let** allQuestionsValid = **true**;

Le nombre total des questions

**let** totalQuestions = $('.borders').length;

$('.borders').each(**function**() {

On extrait l'index des questions

**let** questionIndex = $(**this**).attr('id').split('\_')[2];

On récupère le conteneur des réponses

**let** answersContainer = $('#quiz\_questions\_' + questionIndex + '\_answers');

On regarde le nombre de réponse dans le conteneur

**let** counterAnswer = answersContainer.children().length;

Si il y a au minimum de 2 réponses pour chaque question

**if** (counterAnswer < 2) {

Alors allQuestion est faux

                allQuestionsValid = **false**;

**return** **false**;

            }

        });

Activer le bouton Valider si au moins 10 questions ajouté et que toutes les questions ont au moins deux réponses

**if** (totalQuestions >= 10 && allQuestionsValid) {

            button.attr('disabled', **false**); // si les deux sont vrais le bouton est actif

        } **else** {

            button.attr('disabled', **true**);// sinon il est désactiver

Et voilà mon formulaire me permettant de créer le quiz avec ses questions et ses réponses finaliser.

## Play Quiz

Une seconde partit très importante, réaliser toutes les fonctionnalités pour jouer un quiz.

Plusieurs option s’offrait à moi

1. Afficher le quiz complet et ses question réponse en 1 seule fois et laisser l’utilisateur remplir le formulaire complet et le valider.
2. Cacher les questions sauf la première et afficher les autres à chaque validation.
3. Utiliser Javascript et créer un quiz dynamique qui affichera les questions les 1 après les autres.

J’ai opté pour la troisième option, pour se faire il a fallu que je réfléchisse à comment envoyer les données qui me serait utile pour les récupérer en js.

J’ai choisie de récupéré les données utiles de mes entité de les stocker dans un tableau que j’ai convertit en Json(Javascript Objetc Notation).

$quizData =

[

   'titre' => $quiz->getTitle(),//ajoute le titre du quiz

  'questions' => [], // Initialise le tableau des questions

];

**foreach** ($quiz->getQuestions() as $question) {

Boucle qui récupère les questions appartenant au quiz

**if** ($question->getLink()) {

Si question a un lien

    $link = $question->getLink();// on le récupère et le stock dans la variable $link

    }**else**{

//Sinon $link sera un string vide

                $link = '';

            }

//On stock les donnée utiles dans un tableau

            $questionData = [

                'id' => $question->getId(),//ajoute l'id de la question

                'question' => $question->getSentence(),//ajoute l'intitulé de la question

                'link' => $link, //contiendra le lien de la question ou un string vide

                'reponses' => [], // Initialise le tableau des réponses

            ];

        //boucle pour récupérer les réponses de chaque question

**foreach** ($question->getAnswers() as $reponse) {

                // Ajoutez les données de chaque réponse dans le tableau des réponses

                $questionData['reponses'][] = [

                    'id' => $reponse->getId(),//ajoute l'id de la réponse

                    'intitulle' => $reponse->getSentence(),//ajoute l'intitulé de la réponse

                    'isRihgt' => $reponse->isIsRight(),//ajoute le bolean isRight pour savoir si la réponse est juste au mauvaise

                ];

            }

            // Ajoute les données de la question et de ses réponses au tableau des questions initialisé au début dans quizData

            $quizData['questions'][] = $questionData;

        }

        $quizJson = json\_encode($quizData); //transforme le tableau en Json ((JavaScript Object Notation))

Puis je me suis envoyer le Json vers le Template de ma page

**return** **$this**->render('quiz/playQuiz.html.twig', [

            'quizJson' => $quizJson,

Et l’est intégré dans un data-attribut d’un input en hidden pou qu’il soit caché

{# fichier json en hidden récupérer par le fichier scrip.js et traité pour afficher les question et les réponse du quiz de façon dynamique  #}

<input hidden id="quiz" type="text" data-attribut="{{ quizJson }} ">

On récupère le fichier Json stocker dans le data-atribut

Utilisation de **JSON.parse** pour transformer le JSON en un objet javascript utilisable

**let** quizData = JSON.parse(document.getElementById('quiz').getAttribute('data-attribut'));

J’ai ensuite créé une fonction pour sélectionner les questions au hasard a l’intérieur de se tableau en étant sur qu’il n’affichera pas deux fois la même question

**let** askedQuestions = []; //tableau vide qui stockera les index des question déjà sortit

**let** min = 0;  // déclaration d'une variable min a 0 qui sera utilisé pour l'affichage des question au hasard

**let** max = quizData.questions.length; // variable max qui est set avec le nombre total de question du quiz

**let** currentQuestionIndex = Math.floor(Math.random() \* (max - min)) + min; //Créer un index de question aléatoire pour la première question

 // Fonction pour obtenir un index unique aléatoire

**function** getRandomUniqueIndex(min, max) {

**let** newIndex = Math.floor(Math.random() \* (max - min));// créer un nouvel index de question et tant que l'index nouvellement créer est déjà présent dans le tableau askedQuestion

**while** (askedQuestions.includes(newIndex)) {

            //il va en créer un autre

            newIndex = Math.floor(Math.random() \* (max - min));

        }

        //et retournera le nouvel index

**return** newIndex;

    }

Puis j’ai créé la fonction d’affichage des question et réponses :

// Fonction pour afficher une question

**function** displayQuestion() {

Utilisation de la fonction getRamdomUnique pour stocker le nouvel index de la question

**let** newIndex = getRandomUniqueIndex(0, quizData.questions.length

Et remplace celui de currentQuestion pour la question suivante

        currentQuestionIndex = newIndex;

Stock les index de question qui ont déjà été afficher pour garantir qu'une question ne sera posé qu'une seul fois

        askedQuestions.push(newIndex);

Stock la question a affiché

**let** currentQuestion = quizData.questions[currentQuestionIndex];

Affiche la question

quizQuestion.textContent = currentQuestion.question;

Désactive le bouton a l'affichage de chaque nouvelle question pour qu’on ne puisse pas valider une question sans réponse

        valider.disabled = **true**;

        // Efface les réponses précédentes

        quizAnswer.innerHTML = '';

Affichage des réponses possibles pour la question

        currentQuestion.reponses.forEach(**function** (reponse) {

**let** label = document.createElement('label'); //créer un élément label

**let** input = document.createElement('input'); // créer un élément input

            input.type = 'radio'; // Ajoute un type"radio" sur l'élément input

            input.name = `answer`; // Créer un name 'answer pour tous les bouton radio pour limiter le choix de réponse a 1 seul

            input.id =  `${reponse.id}`; // Ajoute l'id de la réponse dans l'id du input

            label.appendChild(input); // Place le input en élément enfant de l'élément label

            label.appendChild(document.createTextNode(reponse.intitulle)); //affiche les réponses

            quizAnswer.appendChild(label);// Place les réponses dans la div du conteneur de réponses

Ajoute un écouteur d'événements aux bouton radio qui permettra de savoir quand une réponse a été sélectionné et appel la fonction validateAnswers

    input.addEventListener('change', validateAnswers);

Création de la fonction validateAnswers pour activer désactiver le bouton valider en fonction qu'une réponse soit coché ou non

**function** validateAnswers() {

Récupère tous les bouton radio contenu dans une question

**const** radios = quizAnswer.querySelectorAll('input[type="radio"]');

**let** atLeastOneChecked = **false**; // initialise a false pour que le bouton valider ne sois pas cliquable au lancement du quiz

        radios.forEach(**function** (radio) {

**if** (radio.checked) { //si une réponse sélectionné

                atLeastOneChecked = **true**; // le met a true

            }

        });

Condition qui fait que tant que la variable est à false (aucune réponse sélectionné) le bouton valider est désactiver et dès qu’une est sélectionner le réactive

**if** (atLeastOneChecked) { // si atLeastOneChecked = true

            valider.disabled = **false**; // le bouton valider est cliquable

        } **else** {

            valider.disabled = **true**; // sinon il est désactiver jusqu'a qu'on coche une réponse

        }

    }

Ajout d’un écouteur d'événement sur le bouton valider

    valider.addEventListener('click', **function**() {

        validateAnswers();

Déclaration d’un tableau pour stocker les réponses sélectionnées pour chaque question

**let** checkedAnswers = []

La question actuellement affiché

**let** currentQuestion = quizData.questions[currentQuestionIndex];

Récupère tous les bouton radio

**let** radios = quizAnswer.querySelectorAll('input[type="radio"]');

        radios.forEach(**function**(radio) {

Récupère le bouton radio qui est chek donc la réponse sélectionnée

**if** (radio.checked) {

**let** answerId = radio.id; // récupère l'id de la réponses sélectionné

**let** answerIntitulle = radio.nextSibling.textContent; // Récupérez l'intitulé de la réponse sélectionné

Cherche la propriété isRight de la réponse sélectionnée dans le tableau d'après son id

**let** answerIsRight = currentQuestion.reponses.find((reponse) => reponse.id == answerId ).isRihgt

Ajoute dans le tableau l'id de la question son intitule, l'id de la réponse sélectionnée et son intitulé et si c'est une bonne réponse ou non

checkedAnswers.push({

questionId: currentQuestion.id,

questionIntitulle: currentQuestion.question,

link: currentQuestion.link,

      answerId: answerId,

answerIntitulle: answerIntitulle

      answerIsright: answerIsRight

                });

            }

        });

         // Si 10 questions ont été posées

**if** (count === 9 ) {

                valider.style.display = 'none' //cache le bouton valider après la validation de la dernière question

                document.getElementById('quiz-container').style.display = 'none';//cache le contenu du quiz-container contenant les question et les réponse

            }

Stocke les réponses sélectionnées dans le tableau selectedAnswers en s’assurant que chaque question aura son propre tableau de réponses

        selectedAnswers[currentQuestion.id] = selectedAnswers[currentQuestion.id] || [];

        selectedAnswers[currentQuestion.id].push(...checkedAnswers);

Tant que count est inférieur à 9 continue d’afficher des questions

**if** (count < 9) {

            displayQuestion();

        } **else** {

Sinon ajoute le recap dans le html pour le récupérer dans le contrôleur le stocké en session à l'afficher sur une autre page et affiche le message du quiz terminer

            displayRecap()

            message.innerHTML = 'Bravo vous avez Terminer ce quiz appuyer sur le bouton pour découvrir votre résultat';

        }

        count++

    });

Et pour finir la fonction qui va me créer un récapitulatif du quiz avec les Intitulé de question la réponse sélectionnée et le calcul du score

**function** displayRecap() {

**let** scoreTotal = 0

Cache le contenu du quiz-container contenant les questions et les réponses

        document.getElementById('quiz-container').style.display = 'none';

Fait apparaitre le bouton de soumission

        submit.style.display = 'block';

**const** recapContainer = document.getElementById('recap-container');

Cache l’élément ou sera envoyer le tableau de recap

document.getElementById('recap-container').style.display = 'none';

askedQuestionsOrder.forEach((questionIndex, index) => {

**const** questionData = selectedAnswers[quizData.questions[questionIndex].id];

**if** (questionData) {

questionData.forEach(answerData => {

Si la question sélectionnée est vrais on incrémente la variable scoreTotal de 10

**if** (answerData.answerIsright) {

**const** score = 10 ;

scoreTotal = scoreTotal + score

Ajoute le score dans un tableau

tableScore = []

      tableScore.push({

      'score': scoreTotal //score total

Création de l'objet combiné avec les réponses et le score total

**const** selectedAnswersScore = {

                    ...selectedAnswers,

                    ...tableScore

                };

Transforme le tableau en JSON

**const** recapDataJSON = JSON.stringify(selectedAnswersScore);//

JSON.stringify converti une variable Javascript/ un objet ou un tableau en un string JSON prend en premier paramètre la value, peux accepter un second paramètre une fonction de remplacement et un 3ème paramètre pour l'indentation

Ajoute le Json dans un input caché dans le formulaire

**const** recapDataField = document.createElement('input');//créer un élément input

                recapDataField.type = 'hidden';// met le input en type hidden(non visible)

                recapDataField.name = 'recapData'; // lui ajoute le name recapData

                recapContainer.appendChild(recapDataField);// place le input en enfant de la div recapContainer

                recapDataField.value = recapDataJSON; // Ajoutez le Json a la value du input pour pouvoir le récupérer dans le contrôler

### Gestion du quiz et de la sauvegarde en Base de données

Récupération des données de jeux existantes :

$gamePlay = $gameRepository->findOneBy(['userId'=>**$this**->getUser()->getId(), 'quiz' => $quiz->getId()]);

);

Validation et traitement des réponses du formulaire de quiz

**if** ($formQuiz->isSubmitted() && $formQuiz->isValid()) {

Récupère le tableau de récapitulatif du quiz en json.

$recapData = $request->request->get('recapData')

Convertit en une structure de données PHP dans notre cas un tableau associatif

$recapDataArray = json\_decode($recapData, true)

Stockage des données récapitulatif en session pour une utilisation ultérieur

$session->set('recap', $recapDataArray);

On récupère le score stocké dans le tableau

**foreach** ($recapDataArray as $data) {

**if** (isset($data['score'])) {

      $score = $data['score'];

       }

Si l'uttilisateur a 5 partit jouer pour ce quiz

**if (count($allGameUser) == 5) {**

On récupère la game avec le plus petit score

**$game = $gameRepository->findOneBy(['userId'=>$this->getUser()->getId(), 'quiz' => $quiz->getId()],['score'=> 'ASC']);**

Variable créer qui stocker la date du jours

**now = new DateTime();**

Remplace la date de la partie avec le plus petit score par la nouvelle date de la partie

**$game->setDateGame($now);**

Si le score de la nouvelle partie est supérieur au plus petit score déjà enregistré

**if ($game->getScore() < $score) {**

**//alors on écrase le score avec le nouveau**

**$game->setScore($score);**

**}**

Sinon le score reste inchangé seul la date change.

**$entityManager->persist($game);**

**// execute PDO(la requete Insert ou Update)**

**$entityManager->flush();**

**}else{**

Si aucune ne partit jouer sur ce quiz

**$game = new Game; // nouvelle instance de Game**

**$game->addAnswer($answer);//ajoute les question a la game**

**$quiz->addGame($game);// ajout du quiz dans Game**

**$user = $this->getUser(); // on récupère l'user en session**

**$game->setUserId($user); // on rajoute l'user en session a la Game**

**$now = new DateTime();//varaible avec la date du jours**

**$game->setDateGame($now);// on stock la date du jours dans la nouvelle partie**

**game->setScore($score);//ajoute le score a la game**

**// prepare PDO(prepare la requete Insert ou Update)**

**$entityManager->persist($game);**

**// execute PDO(la requete Insert ou Update)**

**$entityManager->flush();**

**}**

Redirige vers la page de récapitulatif

**$this->redirectToRoute('app\_recap');**

        }**else**{

**$this**->addFlash('error', "Ce quiz n'est pas encore disponible");

**return** **$this**->redirectToRoute('app\_home\_quiz');

        }

### Gestion des redirections

J'ai ajouté plusieurs conditions dans mon contrôleur pour gérer toutes les situations

1. Si le quiz est vérifié ou si le rôle est modérateur ou admin

**if** ($quiz->isIsVerified()|| **$this**->isGranted('ROLE\_MODERATOR'))

Si la condition n'est pas vérifiée, renvoyez vers le home avec un message d'erreur

  }**else**{

**$this**->addFlash('warning', "Ce quiz n'est pas encore disponible");

**return** **$this**->redirectToRoute('app\_home\_quiz');

       }

1. S'il n'y a pas de partie jouée ou si la date du jours est égal à la date modifier ou si le rôle est modérateur ou admin

**if** (!$gamePlay || $now == $dateModify || **$this**->isGranted('ROLE\_MODERATOR'))

$dateModify est égal à la date de la dernière partie jouer + 7j.

Si les conditions ne sont pas respectées, redirigez vers le home avec un message d'erreur indiquant le nombre de jours restant avant de pouvoir rejouer.

}**else**{

$nbJour = $dateModify->diff($now)->format("%d");

**$this**->addFlash('warning', 'Vous pourrez rejouer au quiz dans '.$nbJour.' jours');

**return** **$this**->redirectToRoute('app\_home\_quiz');

}

# Axe d'amélioration

1. **Optimisation du Responsive Design :** Continuer à améliorer l'adaptabilité et la convivialité du site sur divers appareils pour offrir une expérience utilisateur optimale.
2. **Optimisation du SEO :** Approfondir les techniques d'optimisation pour améliorer la visibilité du site dans les résultats de recherche et augmenter le trafic organique.
3. **Optimisation des Performances :** Travailler sur l'optimisation de la vitesse de chargement du site, la réduction du temps de réponse du serveur, et la gestion des ressources pour garantir des performances optimales.
4. **Optimisation du Code :** Examiner le code existant pour identifier les parties pouvant être factorisées ou améliorées, favorisant ainsi une structure plus propre, plus efficiente et plus maintenable.
5. **Implémentation d'un Espace Communautaire (style Forum) :** Ajouter une section où les utilisateurs peuvent interagir, partager des informations, discuter et poser des questions pour créer une communauté active et engagée.
6. **Mode Multijoueur pour les Quiz :** Permettre aux utilisateurs de jouer à des quiz en multijoueur avec un système de chronométrage des questions et un mécanisme de score adapté à cette version collaborative.
7. **Fonctionnalité d'Ajout d'Amis :** Offrir la possibilité d'ajouter d'autres utilisateurs en tant qu'amis, facilitant ainsi l'invitation à jouer ensemble et favorisant l'interaction sociale sur la plateforme.
8. **Système de Récompenses (Trophées) :** Mettre en place un système de récompenses ou de trophées pour reconnaître les réalisations des utilisateurs, ce qui peut stimuler l'engagement et la fidélité à la plateforme.