2024

QUIZ QUEST

[Nom de la société] |[Adresse de la société]

[Sous-titre du document]

Gautier CATTANEO

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 la mise en application de la méthode MoSCoW.

Trello est un site permettant de créer des tableaux pour un projet et d’y ajouter des cartes dans chaque colonne nommée Must Have, Should Have, Could Have, Wont Have, En cours, Terminé.

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

Description générée automatiquement

## Schéma du processus de développement

# Modélisation de la Base de Données

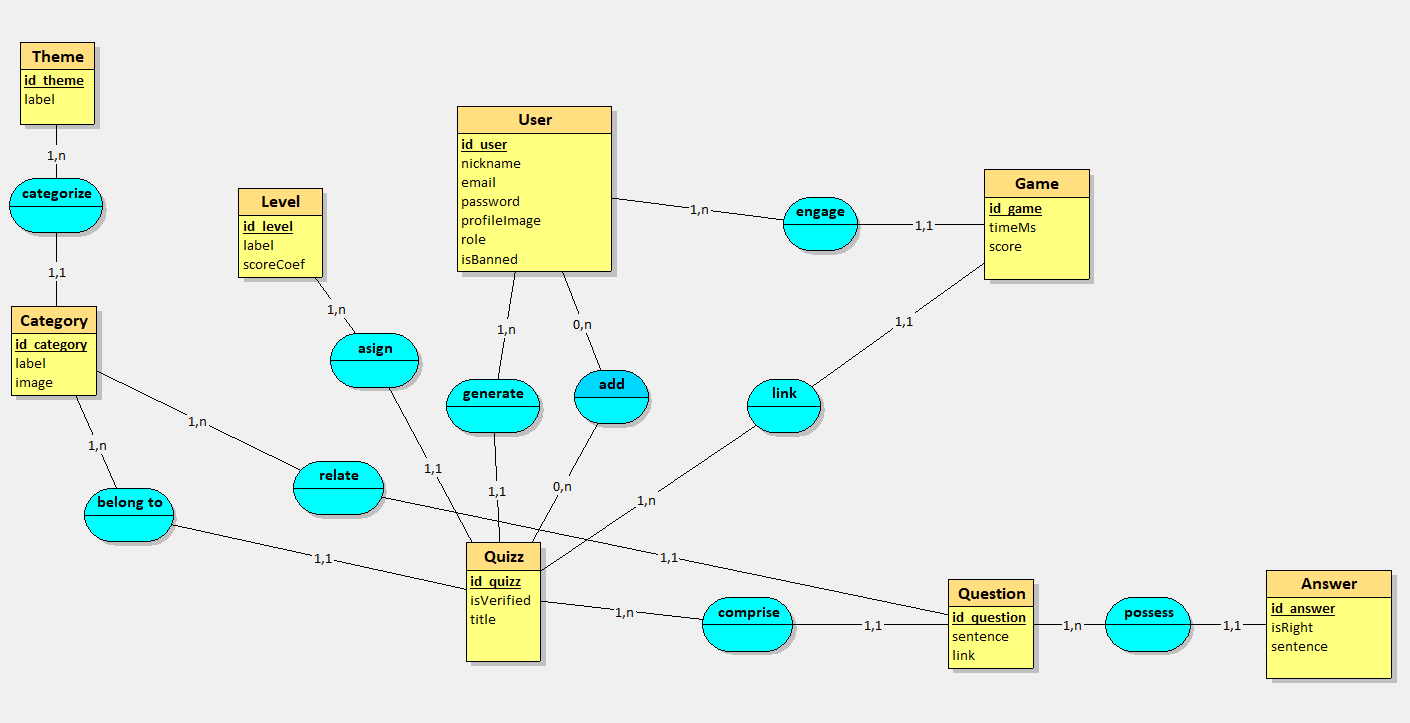
Après avoir décidé de mes fonctionnalités et organisé mon travail en priorisant les tâches, j’ai décidé de modéliser mon système d’information.

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 un élément crucial à la réalisation d’un site Web. Elle va permettre de visualiser schématiquement les entités et leurs propriétés qui constitueront nos Class et les tables de notre base de données de façon simple et visuelle. On peut les réaliser à l’aide de logiciels (looping pour mon projet, mais il en existe d’autres différents exemples : GitMind, Visual Paradigm...)



**Une entité** :

Dans mon projet, prenons l’exemple du User.

Un User est une personne préalablement enregistrée au site web voyons en détail ses **propriétés**:

* Id\_user : Son identifiant permettra de garantir le fait qu’il soit unique. Toute entité aura son identifiant unique. On les appelle les propriétés identifiantes.
* Username : Son pseudo qui sera affiché sur les classements des différents quiz auxquels il a participé
* Email : obligatoire et unique servira d’identifiant de connexion qui devra être vérifié après inscription via un lien envoyé dans un mail
* Password : Le mot de passe sans lui impossible de se connecter, il devra faire un minimum de 12 caractères avec au minimum 1 lettre majuscule, 1 lettre minuscule, 1 caractère spécial, et 1 chiffre pour être valide.
* Rôle : Son rôle sur le site par défaut, le rôle de base lui sera attribué celui du User.
* IsBanned : Un boolean qui sera à true ou à false. Pour définir son statut banni ou non ou cas où il ne respecterait pas les règles du site.

Les relations entre les entités.

Relation se fait à l’aide de cardinalités. Elle représente le type de lien entre chaque entité dans mon projet.

User est relié à :

Quiz : un User crée bulle bleue (son nom doit être un verbe à l’infinitif)1 ou plusieurs Quiz (cardinalité 1, n) et inversement un Quiz est créé par 1 seul et 1 seul User (cardinalité :1,1)

Topic : un User peut créer 0 ou plusieurs Topic (cardinalité 1, n) et 1 Topic est lié à 1 seul et unique User (cardinalité 1,1)

Game : 1 User joue 0 ou plusieurs Game (0, n) et une Game n’est lier qu’à un seul User (1,1)

Une fois toutes nos entités et leur propriété établies.

## Modèle Logique de Données (MLD)

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

Description générée automatiquement

Le MLD (modèle logique des données) décrit la structure de manière plus détaillé que le MCD en précisant le type des données utilisées.

Les entités du MCD deviennent des tables.

La propriété identifiante devient une clé primaire.

Une Clé primaire est un attribut ou un ensemble qui identifie de manière unique chaque enregistrement dans une table. Elle garantit l'unicité des données de cette table.

Dans le MLD on voit aussi l'apparitions de clé étrangère. Elles résultent des choix des cardinalités effectués dans le MCD.

Une clé étrangère fait référence à une clé primaire et permet de garantir l'intégrité référentielle entre les données de différentes tables.

Sur mon MLD nous constatons l'apparition de l'entité Favorites résultant de la relation Many To Many.

## Justification des choix de conception de la base de données

# Conception Graphique

Phrase de transition pour expliquer pourquoi vous avez un maquettage

Mettre en place visuellement les fonctionnalités que j’ai imaginées

Penser l’expérience utilisateur et l’ui

## Wireframe pour l’interface utilisateur

Un wireframe est une représentation visuelle de la conception des pages de notre site Web (sans couleur, sans image). On utilise les wireframes au début d’un projet pour établir une structure de base d’une page avant l’ajout de son visuel. Réalisable sur papier, en HTML et CSS directement ou à l’aide de logiciel ou site spécialisé (Figma pour mon projet)

## Maquettes réalisées avec Figma

# Réalisation du Projet

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

Mettre logos

### **HTML 5 :**

Le Html HyperText Markup Language, est un langage dit de balises, il est utilisé pour créer et représenter le contenu d’une page web et sa structure. S’exécute côté client.

### **CSS 3 :**

Cascading Style Sheets ou feuilles de style en cascade est le code utilisé pour mettre en forme une page Web. Couleurs et tailles du texte, taille des images, position des éléments, couleurs de fond. Il est associé directement avec les balises du HTML.

C’est en utilisant les sélecteur CSS qu’on met en forme les éléments HTML. On peut directement sélectionner une balise HTML (**div {}, .maClasse** {}, un id, **#monId** {}). S’exécute côté client.

### **Looping :**

Le logiciel de modélisation conceptuelle des données a été utilisé pour réaliser mon MCD/ MLD et schématiser la structure de ma base de données.

### **Visual Studio Code :**

C’est un éditeur de code. Son intégration avec Git est très simple et il dispose d’une bibliothèque d’extension très complète adaptée à tout type de langage.

C’est entièrement avec ce logiciel que j’ai réaliser tout le back-end et le front-end.

### **Javascript :**

Langage de script léger utilisé pour certaines parties dynamiques de mon application.

Tableau qui s’affiche en cliquant sur un bouton, Menu dynamique en version mobile

Et a été utilisé pour réaliser ma page play\_quiz.

Détaillé plus loin.

### **PHP 8 :**

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.

S’exécute côté serveur, générant ainsi le HTML qui sera envoyé au client.

### **JQuery :**

Utiliser pour gérer les CollectionType de Symfony, pour rendre les formulaires dynamiques.

Utiliser sur mon projet dans les formulaires de création du quiz de ses questions et de ses réponses.

### **Heidi SQL :**

Outil d’administration de base de données possédant un éditeur SQL et un constructeur de requête.

Il a été développé et optimisé avec le SGBD (Système de gestion de base de données) relationnel MySQL.

### **GitHub et Git :**

GitHub est un site Web et un service de cloud qui permet de stocker et de gérer son code, ainsi qu’à suivre les modifications qui lui sont apportées.

Git est un logiciel qui permet de créer différentes versions de son application et de garder un historique de son code en créant de nouvelles branches.

Toute stockées sur GitHub nous permettant de les comparer, de les fusionner ou même de revenir en arrière en cas de problème majeur.

### **Composer :**

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

### **Symfony CLI :**

Est un serveur local pour exécuter un projet sur son propre ordinateur.

### **Twig :**

Twig est le moteur de template utilisé par défaut sur Symfony. Il permet une interface simple et efficace 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 décidé d'utiliser un Framework.

Un Framework

Pour faire fonctionner Symfony, il faut installer une version de PHP, Composer et optionnellement Symfony CLI.

Mon projet est basé sur Symfony 6.3 Framework PHP, il assure la bonne structuration de l’application pour cela il utilise 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. Sur Symfony, l’affichage est géré par Twig

La couche **contrôleur**, comme son nom l’indique, est la tour de contrôle de l’application, c’est elle qui va gérer les actions des utilisateurs et appeler les bonnes fonctions pour renvoyer les bonnes informations à la vue.

Schéma représentant le MVC sous Symfony

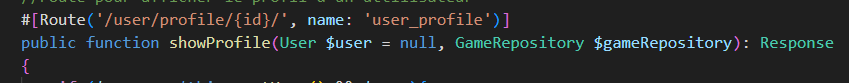
Une image contenant texte, capture d’écran, diagramme, conception

Description générée automatiquement

Quand un utilisateur clic sur une URL, cela envoie une requête HTTP au serveur, elle est traitée par le service de routing



Qui grâce aux informations contenues dans l'URL va appeler le bon contrôleur.



Le contrôleur, s'il en a besoin va appeler la couche modèle sous Symfony, le repository

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Et renvoyer les données dont il a besoin vers la vue

Une image contenant texte, Police, capture d’écran

Description générée automatiquement



Et une fois les données traitées et transmises l'afficher 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 comprendre son fonctionnement.

Tous débutent 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 et qui reçoit une réponse du serveur en HTML avec un code réponse (200 pour succès, 404 pour page non trouvée).



### Les avantages de Symfony

#### Les commandes :

Sur Symfony, certaines fonctions ont été automatisées à l’aide de commandes a exécutées dans une fenêtre PowerShell. Une fenêtre Windows existe sur les ordinateurs accessibles en recherchant PowerShell dans la barre de recherche, il existe aussi des logiciels, j’utilise personnellement la console intégrée à Visual Studio Code.

#### Créer un nouveau projet

Symfony new NomDuProjet –webapp

Permet de créer un dossier complet avec toutes les dépendances de base incluses.

#### Lancer le serveur

Symfony server::start / start -d (permet de garder la console ouverte serveur lancer en arrière-plan)

#### Créer une Entity ou la mettre à jour

Symfony console make:entity (ou m:e)

Permet de créer notre Entity défini dans notre MCD de façon automatisée et guider étapes par étapes la console nous demandant chaque propriété qui constituera notre objet et s’occupera de créer ses propriétés et méthodes et getter et setter.

Une commande spécifique existe pour la création d’un User qui intègre tout un système de sécurité inclus.

##### Création de l’entité User

symfony console make:user

Qui va créer une entité User avec certaines propriétés incluses dans le bundle Security de Symfony dont :

getUserIdentifier()

Qui sera l’identifiant utilisé lors de la connexion et choisi lors de la création à la question :

Enter a property name that will be the unique "display" name for the user (e.g. email, username, uuid) [email]:

> email

* Les getter et setter pour les rôles

**public** **function** getRoles(): array

    {

        $roles = **$this**->roles;

        // guarantee every user at least has ROLE\_USER

        $roles[] = 'ROLE\_USER';

**return** array\_unique($roles);

    }

**public** **function** setRoles(array $roles): **static**

    {

**$this**->roles = $roles;

**return** **$this**;

    }

* Toute la logique de vérification d’email si on en fait la demande

**public** **function** isVerified(): bool

    {

**return** **$this**->isVerified;

    }

**public** **function** setIsVerified(bool $isVerified): **static**

{

**$this**->isVerified = $isVerified;

**return** **$this**;

    }

Qui demande l’installation en plus du bundle verify-email

composer require symfonycasts/verify-email-bundle

Pour avoir accès à la partie Sécurité : SecurityController et AppAuthenticator, il faut aussi faire cette autre ligne de commande :

symfony console make:auth

Qui apporte toute la logique de connexion/déconnection à laquelle j’ai dû apporter quelque modification à la connexion pour la vérification du mail et de son statut banni ou non.

**public** **function** authenticate(Request $request): Passport

    {

        $email = $request->request->get('email', '');

        $request->getSession()->set(SecurityRequestAttributes::LAST\_USERNAME, $email);

        // On récupère le user grâce à son email

        $userEntity = **$this**->userRepository->findOneBy(['email' => $email]);

        // Et on Vérifie si l'email est vérifié ou si le user est banni

**if** (!$userEntity || !$userEntity->isVerified() || $userEntity->isIsBanned()) {

            $errorMessage = $userEntity && $userEntity->isIsBanned()

                ? 'Votre compte est banni.'

                : 'Votre adresse e-mail n\'est pas vérifiée.';

**throw** **new** CustomUserMessageAuthenticationException($errorMessage);

        }

#### Création de formulaire

##### symfony console make:registration-form

Pour la création du formulaire d’enregistrement du User ou là, aussi certaines modifications ont dû être appliquées au niveau du password.

->add('plainPassword', RepeatedType::class, [

                'attr' =>[

                    'class' => 'form-control'

                ],

                '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.'])

                ],

                'mapped' => false,

                'type' => PasswordType::class,

                'invalid\_message' => 'Les password doivent être identiques.',

                'options' => [

                    'attr' => [

                        'class' => 'password-field',

                        'class' => 'form-control'

                    ]

                ],

                'required' => true,

                'first\_options'  => ['label' => 'Password'],

                'second\_options' => ['label' => 'Repeat Password']

            ])

Avec l’ajout d’une regex qui oblige le User à enregistrer un mot de passe contenant au minimum 12 caractères dont 1 lettre majuscule, 1 lettre minuscule, 1 chiffre et 1 caractère spécial.

Pour être en accord avec la CNIL qui a mis à jour le 14 octobre 2022 pour tenir compte de l’évolution des connaissances et pour garantir un niveau de sécurité acceptable contre les attaques par force brute.

J’y ai ajouté une sécurité supplémentaire en limitant le nombre d’erreurs de mot de passe à 3 ensuite il faudra patienter 10 min avant de pouvoir réessayer.

##### Reset password

Autre bundle à installer le reset password qui permet l’ajout d’un lien au formulaire de login en cas d’oubli du mot de passe, qui fera parvenir un lien de réinitialisation sur l’adresse e-mail du user qui en fait la demande.

Commande pour installer le bundle

**composer require symfonycasts/reset-password-bundle**

Pour créer les formulaires et l’entité ResetPassworRequest et son contrôleur.

**symfony console make:reset-password**

L'utilisateur, en cliquant sur le lien mot de passe oublié, 

Va avoir accès à un formulaire pour rentrer son email.

Et seulement si l'email saisi est enregistré en base de données, alors un token unique sera généré d'une durée de vie de 1h, et il sera stocké à la session de l'utilisateur correspondant au mail et inclus dans le lien de réinitialisation envoyé dans l'email.

Se token sera comparé à la validation du formulaire de réinitialisation, et c'est seulement si le token stocké en session correspond au token avec lequel la réinitialisation a été soumise que le mot de passe sera modifié.

## 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 entrant, 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 Template. Toutes les données affichées sont échappées par défaut, ce qui empêche 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.

**Protection** :

* Limiter les 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.

Expliquer pourquoi vous changer le nom avec uniqid

Pourquoi les mettre dans un dossier séparés des autres images

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

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

                'mapped' => false,

                'required' => false,

                'constraints' => [

**new** File([

                        'maxSize' => '1024k',

                        'mimeTypes' => [

                            'image/jpeg',

                            'image/jpg',

                            'image/png'

                        ],

                        'mimeTypesMessage' => "Télécharger une image format valide s'il vous plait",

                    ])

                ],

            ])

* Renommer le fichier uploadé

Pour éviter la manipulation de l’extension de fichier et pour éviter que plusieurs fichier on le même nom, il est recommandé de renommer le fichier en utilisant un nom de fichier unique grâce à la fonction uniqid().

**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) {

            // ... handle exception if something happens during file upload

        }

**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 a avoir une meilleur organisation.

### 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 qu'il ne voulait pas soumettre.

Pour s'en protéger il existe une protection CRSF qui est l'ajout d'un champ masqué dans les formulaires qui contient une valeur(token) que seul l'administrateur du site et l'utilisateur peuvent connaitre, Cela garantit que c'est bien l'utilisateur connecter qui soumet les données fournies.



### Attaque par Brute force/ Attaque par dictionnaire

Les attaques par Brute force consiste à tester 1 par 1 toute les combinaisons possibles d'un mot de passe. Un ordinateur personnel est capable de tester plusieurs centaines de millier voire quelque million de combinaison par seconde.

Les attaques par dictionnaire fonctionnent dans le même schéma qu'une attaque par Brute force mais la seront testé une liste de mot passe, de mot de chaine de caractère, des phrases, dans le but de contourner la sécurité d'un système.

Protection : Sur mon projet j'ai limité le nombre de tentative de mot de passe erroné a 3 par minute ensuite un délai d'attente de 10 min devra être effectuer avant de pouvoir réessayé.

J'y est ajouter une REGEX qui oblige un utilisateur à avoir au minimum un mot de passe de 12 caractère contenant 1 minuscule, 1 majuscule, 1 chiffre et 1 caractère spécial. En obligeant j'augmente considérablement le nombre de combinaison possible.

# 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>

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

<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

**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

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

    {

        $builder

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

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

                'prototype' => 'answers',

                'label' => 'Réponses',

                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 Question n'a pas de setAnswer mais c'est Answer qui contient setQuestion Question est propriétaire de la relation. Pour éviter un mapping => false on est obligé de rajouter un 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 a 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', [

            'category' => $category,

            'quiz' => $quiz,

            'quizJson' => $quizJson,

            'formQuiz' => $formQuiz,

        ]);

}

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

// Récupération des données du quiz depuis l'attribut 'data-attribut'

**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 sélectionné 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 question

**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 sélection 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

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

**if** ($gamePlay) {

             $date = $gamePlay->getDateGame();

             $dateModify = date\_modify($date ,"+7 day");

        }

        $now = **new** DateTime();

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

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

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

                //si le formQuizulaire est remplie et valide

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

                    $recapData = $request->request->get('recapData');//récupère le tableau de récapitulatif du quiz en json

                    $recapDataArray = json\_decode($recapData, true);// convertit en une structure de données PHP dans notre cas un tableau associatif

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

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

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

                            $score = $data['score'];

                        } **else** {

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

                                $answer = $entityManager->getRepository(Answer::class)->findOneBy(['id' => $response['answerId']]);//récupère la question grace a son id contenu dans le tableau

                            }

                        }

                    }

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

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

                        $now = **new** DateTime();

                        $game->setDateGame($now);

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

                            $game->setScore($score);

                        }

                        $entityManager->persist($game);

                        // execute PDO(la requete Insert ou Update)

                        $entityManager->flush();

                    }**else**{

                        $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();

                        $game->setDateGame($now);

                        $game->setScore($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 ver le home qui est la liste des formation

                    }

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

                }

            }**else**{

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

**$this**->addFlash('error', 'Vous pourez rejouer le quiz dans '.$nbJour.' jours');

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

            }

        }**else**{

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

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

        }

vs

Fire un chapitre SEO

Faire un chapitre RESPONSIVE DESIGN

TERMINER PAR AXES D’AMELIORATIONS

TARTE AU CITRON DANS LE RGPD, expliquer ce que c’est un cookie et les recommandations de la cnil

POT de miel ;