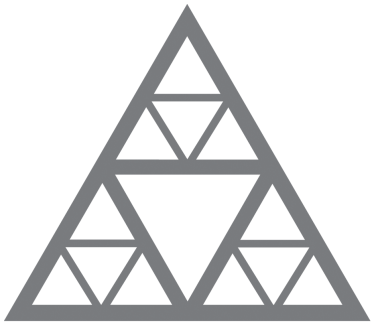
École des Ponts ParisTech



Techniques de développement logiciel

Projet : Implémentation d'un quiz

Maxime Bary, Julien Bernier, Mélisande Duhalde, Raniya El Jouhari

Encadré par Guillaume Desforges

20 janvier 2023

Table des matières

1 Objectif initial

Nous voulions implémenter un site proposant des quiz. Un quiz peut être un ensemble prédéfini de questions. Ou bien un ensemble de questions choisies aléatoirement. Nous avons choisi, lors de l'implémentation, la deuxième option.

L'utilisateur doit pouvoir : se connecter, créer une question, répondre à un quiz et connaître ses résultats, changer de compte. Ces fonctionnalités ont été implémentées dans cet ordre. Nous avions plusieurs idées, certaines ont pu être implémentées, d'autres pas, en raison des difficultés rencontrées (prise en main de Django notamment) lors de la durée du projet.

D'une part les idées implémentées : les questions sont rattachées à un et un seul thème, le nombre de questions ajoutées par l'utilisateur est visible depuis une page dédiée (*Mon compte*).

D'autre part, les idées en début de projet, non implémentées : autres statistiques concernant l'utilisateur (nombre de bonnes réponses à la fin d'un quiz, nombre de bonnes réponses sur tous les quiz, nombre de quiz effectués ou encore le temps de réponse à un quiz), statistiques concernant les quiz eux-mêmes (taux de réussite d'une question et proportions pour chacun de ses choix, note d'appréciation d'une question) et horloge affichée lors de la réalisation d'un quiz. Également, au tout début, nous pensions à la possibilité de créer un mode multi-joueurs : plusieurs utilisateurs sont connectés, le serveur étant alors sur une des machines.

Le nombre de questions par quiz est fixe (3 questions). Le nombre de choix par question est fixe également : 4 choix chacun ayant une valeur booléenne (vrai ou faux, en pratique dans l'implémentation 0 ou 1). Pour une même question, plusieurs choix peuvent être vrais. Lors de la réponse à un quiz, le mode de sélection utilisé est le bouton radio. Il est donc préférable qu'un des choix au moins soit de valeur vraie.

Parcours utilisateurs réalisés

Il est, à ce jour, possible de réaliser les parcours suivants :

Parcours 1 : Créer un compte
- L'utilisateur accède au site (écran d'accueil)
- L'utilisateur crée un compte (saisie d'un pseudo et d'un mot de passe)
- Retour à l'écran d'accueil après soumission des données

Parcours 2 : Se connecter
- L'utilisateur accède au site (écran d'accueil)
- L'utilisateur se connecte (saisie pseudo et mot de passe)
- Renvoi à l'écran menu après soumission des données

Parcours 3 : Créer une question
- L'utilisateur se connecte
- Cliquer sur le bouton *Nouvelle Question*
- Saisir le texte de la question. Cocher le thème. Cliquer sur le bouton *Submit*
- Saisir le premier choix : texte et valeur de vérité (vrai ou faux) puis soumettre les données
- Faire de même pour les trois choix suivants
- Renvoi à l'écran menu après soumission des données pour le quatrième choix

Parcours 4 : Répondre à un quiz
- L'utilisateur se connecte
- Cliquer sur le bouton *Quizz start*
- Choix du thème et cliquer sur le bouton *Commencer*
- Soumettre les réponses (boutons radio) puis cliquer sur le bouton *Confirmer*
- Affichage de la correction des réponses soumises (*True* ou *False*)
- Cliquer sur le bouton *Retour au menu principal*

Parcours 5 : Consulter son compte
- L'utilisateur se connecte
- Cliquer sur le bouton *Mon compte*
- Donnée affichée : nombre de questions ajoutées.
- Possibilité de changer de compte : cliquer le bouton *Changer de compte*

2 Réalisation

Pour implémenter notre site, nous avons utilisé le framework Django. c'est un framework Python open-source de haut niveau permettant un développement rapide de sites internet.

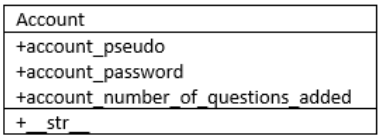
L'une des raisons pour lesquelles Django est particulièrement adapté à la création d'un site proposant des quiz est son support intégré pour les modèles de base de données. L'ORM (Object-Relational Mapper) de Django permet de définir le schéma de base de données à l'aide de classes Python, et fournit un système de gestion de modèles de données qui facilite la création, la modification et la suppression de tables dans la base de données. Cela facilite la création et la gestion des données d'un quiz, telles que les questions et les choix de réponses à ces questions.

Une autre raison pour laquelle Django est bien adapté à la création d'un quiz est son support pour la gestion des formulaires. Django propose une bibliothèque de gestion de formulaires intégrée qui facilite la création et la validation des formulaires, ce qui est utile lors de la création d'un quiz qui permet aux utilisateurs de soumettre leurs réponses.

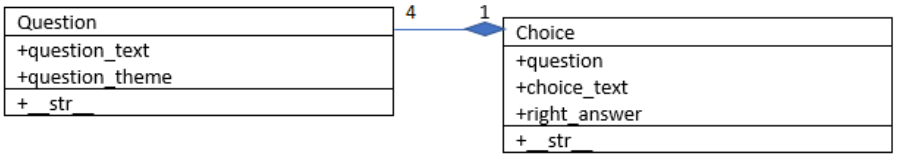
Enfin, Django inclut un panneau d'administration qui nous permet de gérer les données du site, comme les comptes des utilisateurs, les questions, les réponses etc. Il est simple à utiliser et permet d'effectuer rapidement les opérations d'ajout, de mise à jour et de suppression de données.

Notre application reposant sur une interface graphique, nous avons utilisé l'architecture MVC (Model View Controler).

Le modèle est constitué de trois classes :



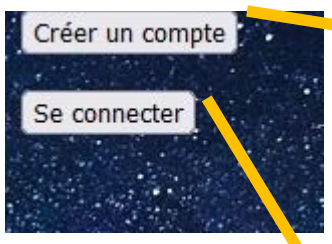
Account est une classe dont les instances sont les comptes des utilisateurs, avec pour attributs un pseudo, un mot de passe et le nombre de questions ajoutées par l'utilisateur.



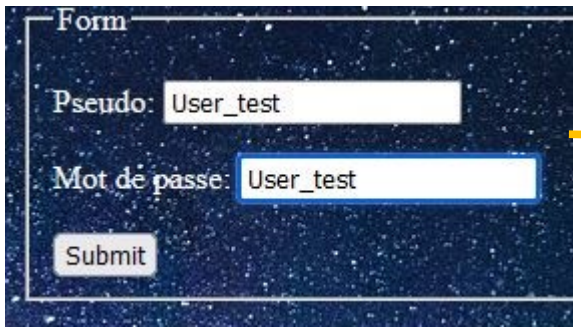
Question est une classe dont les instances sont les questions, avec pour attributs le texte de la question et le thème de la question. Choice est une classe dont les instances sont les choix de réponses des questions, avec pour attribut la question dont il est le choix, le texte du choix et un attribut right_answer indiquant si le choix est vrai ou faux.

Nous présentons l'architecture MVC utilisée à la page suivante.

base_screen



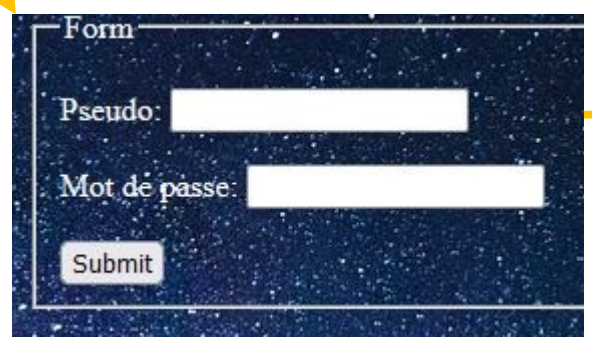
create_account



Créer un objet Account,

account_pseudo,
account_password
account_number_of_questions_added

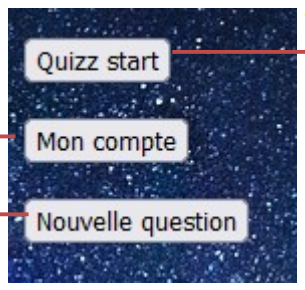
connect



Le controler vérifie si le pseudo et le mot de passe rentrés correspondent aux attributs d'un objet du type Account

Objet Account

main_menu

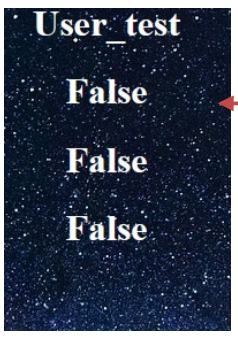


quizz_start

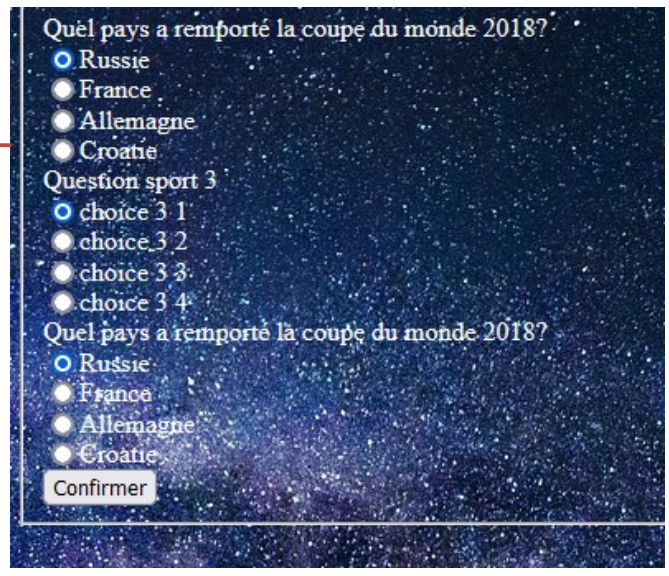
Objets Question | Objets Choice

Le controler choisit aléatoirement trois questions du thème choisi et accède aux choix qui leurs sont associés

results_sheet



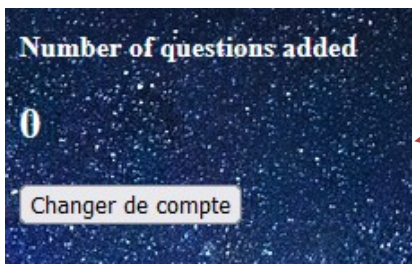
answer_question



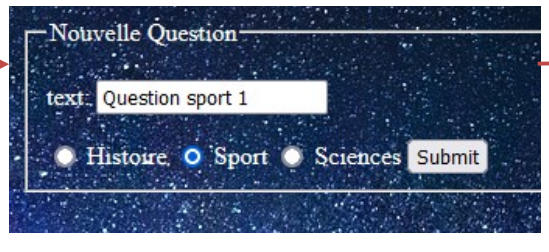
Objets Question

account_number_of_questions_added

my_account

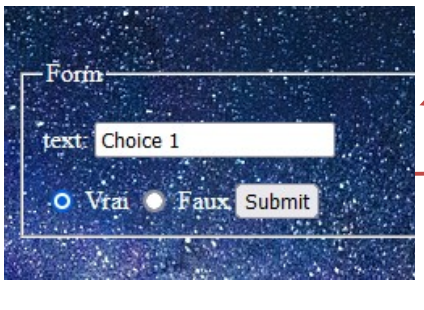


create_new_question



Créer un objet Question,

question_text,
question_theme



X 4

Créer un objet Choice,

Question,
choice_text,
right_answer

3 Difficultés rencontrées

Au cours de notre projet, nous avons rencontré différentes difficultés qui ont ralenti notre progression. Ces difficultés sont de natures différentes : certaines étaient relatives à l'architecture de l'application, et d'autres étaient relatives à l'implémentation de l'application sur Django.

3.1 Difficultés liées à l'architecture de l'application

Afin de réaliser des manipulations sur les instances de classe Choice et Question enregistrées dans la base de données, nous avons dû nous arrêter sur les relations à introduire entre ces deux types de données, et la manière de traduire ces relations dans la définition des classes Choice et Question. La difficulté a résidé dans le choix d'une solution parmi la multitude de solutions possibles et équivalentes, sachant que ce choix déterminerait la manière d'implémenter les différentes fonctionnalités prévues. Nous avons finalement décidé de lier ces deux types de données en introduisant l'attribut question à la classe Choice, ainsi pour une instance c de classe Choice, c.question renvoie l'instance de classe Question pour laquelle c constitue un choix possible.

3.2 Difficultés liées à l'implémentation de l'application sur Django

Avant ce projet, aucun membre du groupe n'avait utilisé Django. Ainsi, malgré le tutoriel et la documentation proposés, la majorité des problèmes que nous avons rencontrés concernent l'implémentation de nos idées sur Django.

Tout d'abord, au début du projet, beaucoup d'erreurs ont émergé d'une mauvaise compréhension des liens et interactions entre les URLs, les views, les forms et le model. Par exemple, comment sauvegarder une donnée dans la base de données via la méthode .save(). Avec la pratique, la structure générale et la marche à suivre sont devenues de plus en plus maîtrisées.

D'autres difficultés ont été spécifiques à l'utilisation du langage HTML, lui-aussi inconnu des membres du groupe. En particulier, faire passer des variables d'une URL à une autre a été problématique dans le cas d'une liste de variables. En effet, en faisant passer une liste en tant qu'argument dans une URL, nous n'avons pas réussi à accéder aux éléments de la liste dans le script HTML de la page. Notre solution, faire passer chaque élément séparément, n'est pas réalisable pour une liste de taille quelconque. Par ailleurs, nous avons eu du mal à accéder à l'option choisie dans un bouton radio en vue de l'enregistrer dans la base de données (point développé dans la partie 4. Focus sur un point technique).

Aux erreurs spécifiques à Django s'ajoutent enfin des pertes de temps à cause de problèmes de syntaxe (noms d'objets mal orthographiés, virgules oubliées, parenthésage, indentation, ...).

4 Focus sur un point technique

4.1 Définition du problème

Après avoir implémenté la fonctionnalité permettant à un utilisateur de créer une question, nous avons souhaité ajouter l'attribut *question_ theme* à la classe *Question*, en vue de générer plus tard des quiz selon un thème choisi par le joueur. Jusqu'alors, pour enregistrer une donnée saisie par le joueur dans la base de données, nous appelions dans la view un form précisant les attributs devant être renseignés. Cela propose au joueur de remplir une zone de texte, qui sera sauvegardée grace à la méthode form.save() appelée dans la view.

Cependant, si ce type de champ est adapté à des inputs créatifs (texte d'une question), il ne l'est pas pour un input sélectif (vrai ou faux), car d'une part le joueur ne connaît pas les différentes options, et d'autre part le joueur peut commettre une erreur d'orthographe en saisissant son choix.

4.2 Résolution

Afin de faciliter le choix du thème pour le joueur, nous avons décidé de remplacer le champ de texte proposé par défaut par une liste de boutons radio. Deux choix s'offraient à nous :

1. Intégrer les boutons radio dans le form directement ;

2. Masquer le champ de texte, introduire des boutons radio dans le HTML, puis, dans le résultat de la requête POST, récupérer le choix de l'utilisateur pour le placer dans l'emplacement réservé à la réponse saisie dans le champ de texte.

Nous avons finalement choisi la deuxième option, dont voici le détail de l'implémentation :

1. Afin de masquer le champ de texte, nous avons précisé le type de widget associé au champ *question_theme*.

```
class FormQuestion(forms.ModelForm):
    class Meta:
        model = Question
        fields = ["question_text", 'question_theme']
        labels = {'question_text': "text",'question_theme':'question_theme'}
        widgets = {'question_theme': forms.HiddenInput(), }
```

2. Les boutons radio ont ensuite été introduits dans le HTML. On précise notamment le nom donné à la donnée précisant le choix sélectionné (*theme*) et les différentes valeurs que cette donnée pourra prendre selon le choix du joueur (*histoire, sport et sciences*).

```
<input type="radio" id=Choix1 name="theme" value ="histoire">
<label for="Choix1">Histoire</label>

<input type="radio" id=Choix2 name="theme" value ="sport">
<label for="Choix2">Sport</label>

<input type="radio" id=Choix3 name="theme" value ="sciences">
<label for="Choix3">Sciences</label>
```

3. On récupère ensuite la valeur prise par *theme* avant de la placer comme réponse au champ *question_theme*.
   NB : le dictionnaire request.POST contient les réponses aux diférents champs mais n'est pas modifiable, on en fait donc une copie.

```
copy = request.POST.copy()
copy['question_theme'] = copy['theme']
```

Cette procédure a été utilisée à chaque fois que l'on demande au joueur de choisir entre certains choix prédéfinis (dont la création des choix possibles pour une question).

5 Retour d'expérience

Maxime Bary

Premièrement, j'ai apprécié de me familiariser avec Django, outil pour coder des applications. La pratique me permit de mieux comprendre ce que j'avais étudié dans le tutoriel de Django et d'éclaircir certains points que j'avais malgré tout laissés obscurs. Comprendre le principe du html, langage d'affichage, et sa syntaxe particulière, fut également intéressant. Toutefois, je regrette que nous n'ayons pas eu le temps de mettre l'application en réseau.

Deuxièmement, j'ai redécouvert l'aspect pratique de GitHub pour coder à plusieurs. Certes il peut être tentant de communiquer des modifications du code par mail et Messenger, mais la pratique devient vite insupportable comme nous l'avons constaté rapidement.

Troisièmement, ce qui m'a frappé fut le chemin restant à parcourir pour savoir coder, non pour l'algorithmie mais pour le simple codage. En particulier, j'ai découvert que repérer les erreurs dans le code était une tâche fastidieuse et chronophage mais nécessaire, qui pouvait être limitée par des tests réguliers.

Julien Bernier

En amont du projet, je n'avais pas d'attente particulière, autre qu'appliquer dans un cadre plus approfondi les concepts abordés lors des TPs et des quiz.

Habitué de manière générale à ne pas me soucier de la lisibilité et la propreté de mon code - me disant souvent "l'essentiel c'est que ça marche" -, ce projet m'a forcé à être plus soigneux et organisé dans ma rédaction, notamment dans un projet regroupant de nombreux fichiers et fonctions, le tout dans un framework nouveau pour tous les membres du groupe.

En effet, je ne m'attendais pas à découvrir de nouveaux langages et frameworks, en l'occurence le HTML et Django. L'apprentissage de ces nouveaux outils aura été le principal frein dans mon travail, mais aura élargi ma vision des différentes techniques utilisables en programmation.

Mélisande Duhalde

Ce projet a été enrichissant pour plusieurs aspects : le déroulement d'un projet, la prise en main de Django qui s'est avérée longue et peu intuitive, le passage de l'idée à l'implémentation.

Je retiens que dans ce type de projet la conception est revue ou ajustée au fur et à mesure que l'on progresse dans l'implémentation au sein du framework, lui-même se prêtant à certains choix de conception plutôt qu'à d'autres ainsi que l'importance de communiquer l'avancement et les idées au sein du groupe.

J'ai pu découvrir les concepts nécessaires à la création d'un site web : vue, formulaire, url et modèle d'une part, les éléments de style (CSS), de contenu et de sa structuration (HTML) d'autre part. Le projet m'a permis d'avoir une première vision globale de la création d'un site web.

Raniya El Jouhari

Cette première expérience de travail en groupe avec GitHub a été très enrichissante. J'ai appris à utiliser efficacement cet outil pour collaborer sur le développement de notre projet, ainsi que l'importance de communiquer régulièrement pour éviter les conflits de versions.

J'ai également compris l'importance de commenter le code pour faciliter la compréhension et la maintenance de celui-ci. Cela m'a aidé à mieux comprendre les décisions de conception prises par les autres membres de l'équipe, et à éviter les erreurs de codage en ayant une meilleure vue d'ensemble.

Enfin, ce projet m'a permis d'apprendre les concepts de base de Django, ainsi que les principaux éléments de HTML et CSS pour la mise en page. Cette expérience m'a donné une meilleure compréhension de ce qu'il faut pour créer un site web fonctionnel.