PROJET ISEN BREST - DÉVELOPPEUR IA

Miskatonic : Générateur de Quiz

> *Ce projet vise à créer une plateforme web robuste et flexible pour la création et la gestion automatisée de quiz pédagogiques, répondant aux besoins spécifiques des enseignants et des administrateurs.*

Objectif Principal

Développer une plateforme web fonctionnelle pour la création et la génération de quiz.

L'Équipe

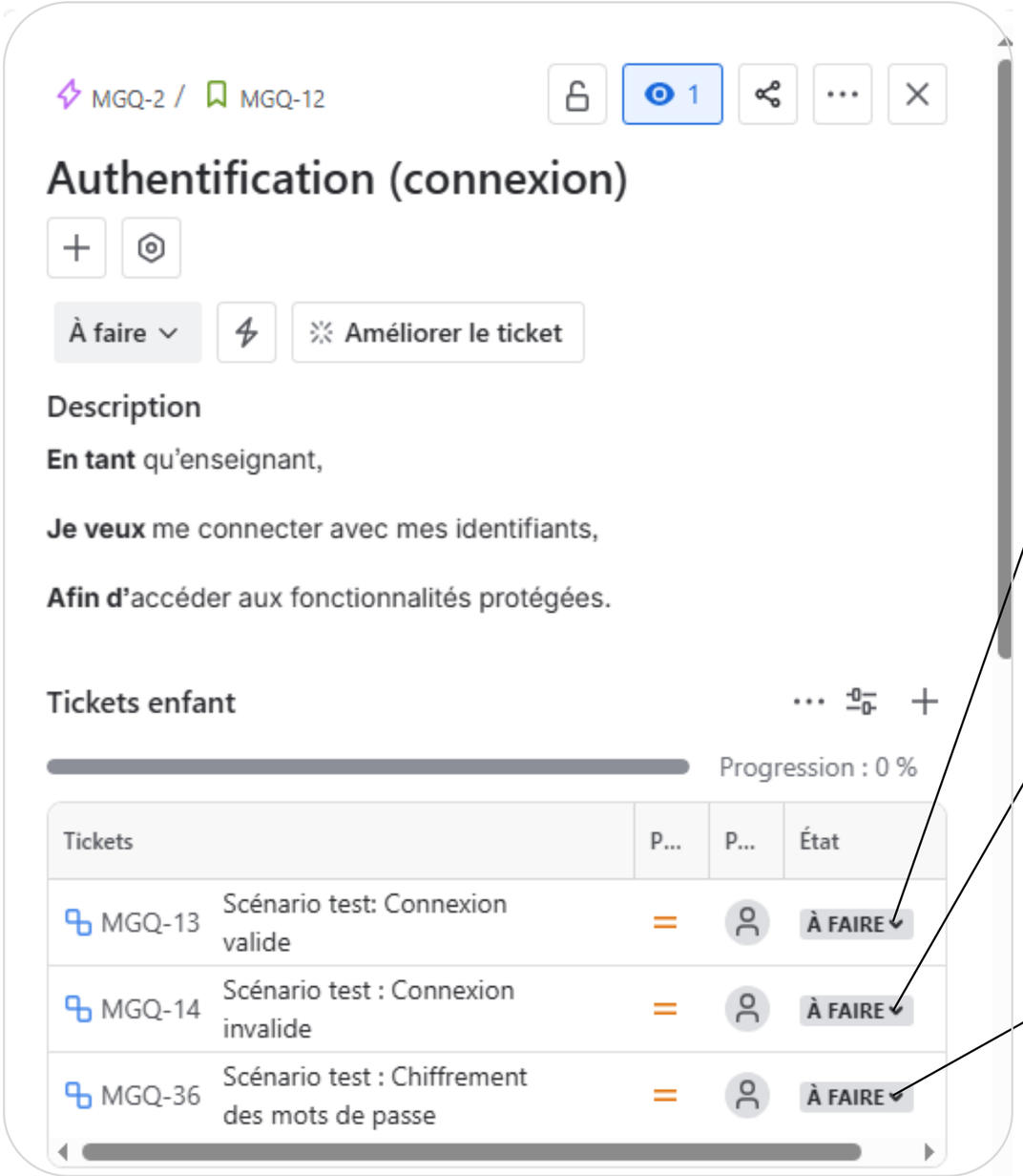
Nathalie Bédiée et Hugo Babin

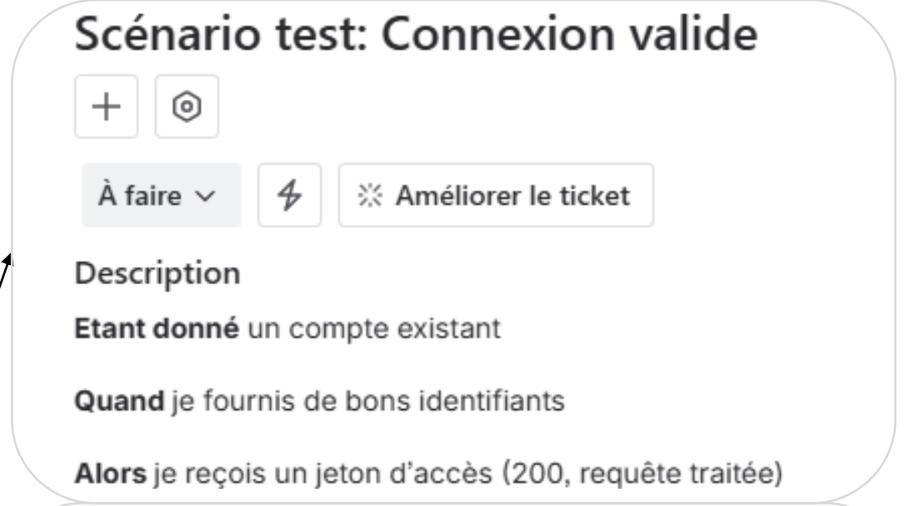
Méthodologie

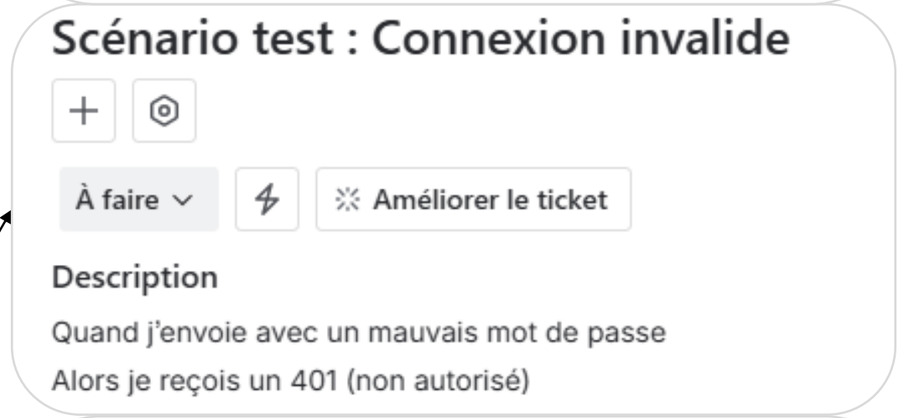
Utilisation de la méthode Agile (Jira, sprints, backlog INVEST)

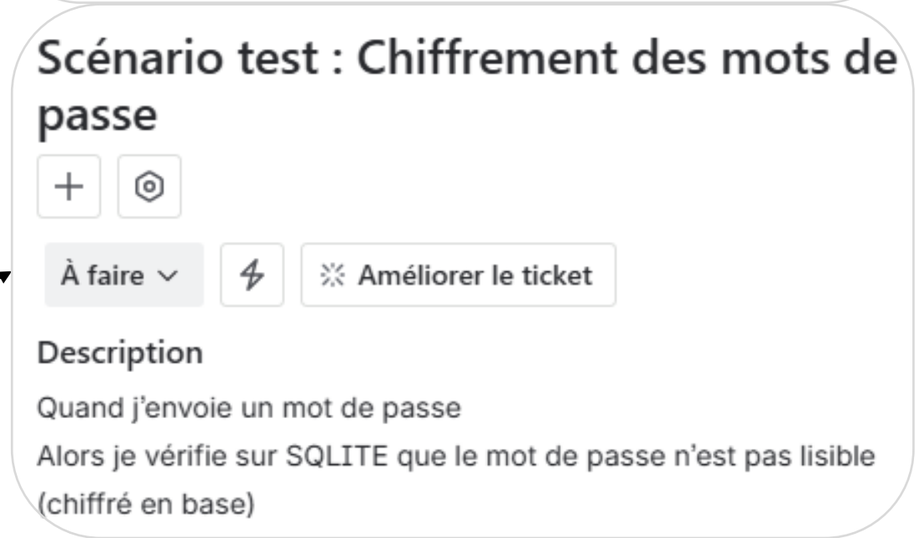
[2]Organisation US sous Jira

Toutes nos US suivent ce format









Les scénarios de test sont nos critères d'acceptation

FEUILLE DE ROUTE PRODUIT

Les Impératifs Fonctionnels de Miskatonic

Afin de garantir une solution complète et utilisable, nous avons défini un ensemble d'objectifs fonctionnels clés, ciblant à la fois la sécurité, la gestion des données et l'expérience utilisateur.

Sécurité et Rôles

Implémentation d'une authentification sécurisée distinguant les rôles d'enseignant (teacher) et d'administrateur (admin).

Importation de Masse (ETL)

Développement d'un pipeline ETL automatisé permettant l'importation massive et structurée de questions via des fichiers CSV.

Stockage NoSQL

Utilisation de MongoDB pour le stockage flexible et évolutif des données de questions et de quiz.

Génération Aléatoire

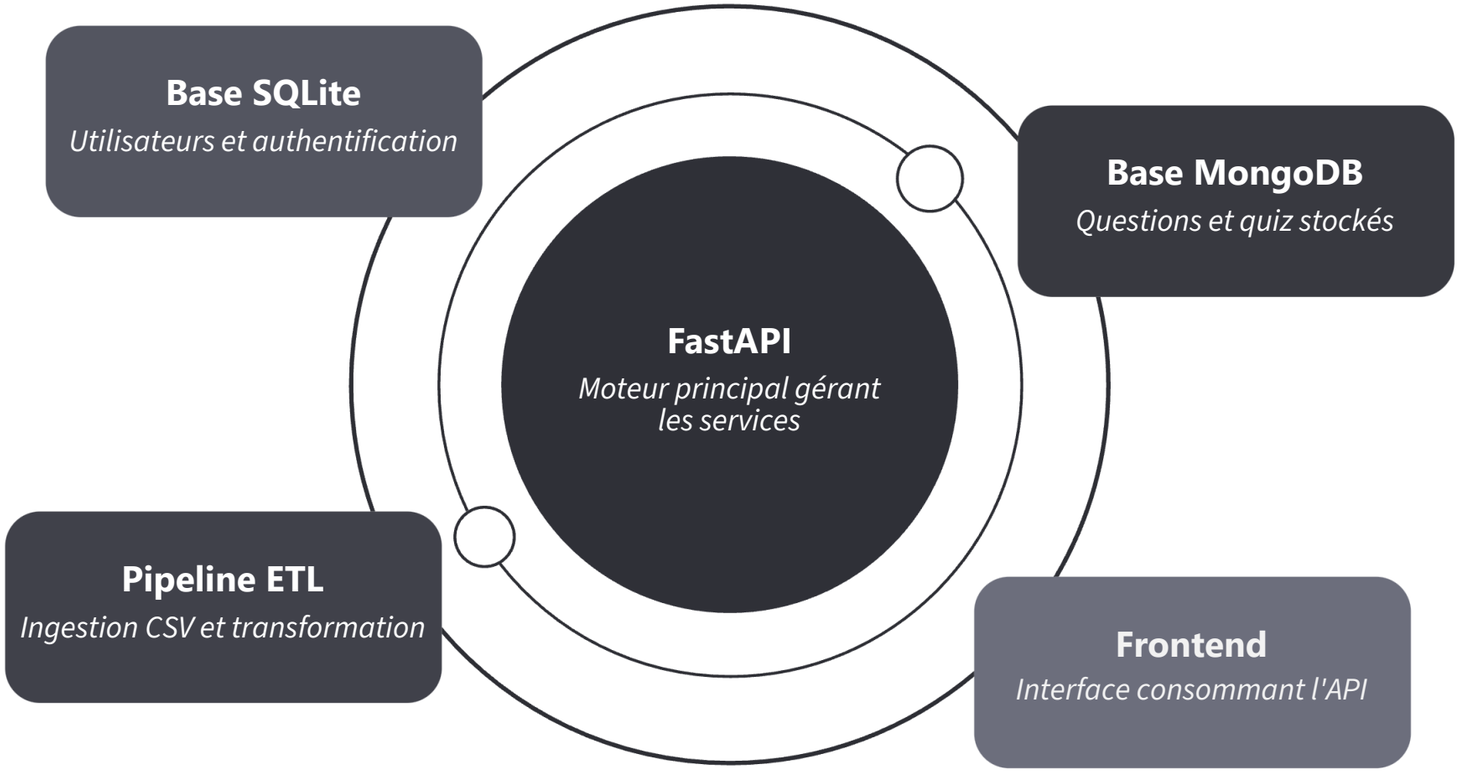
Capacité à générer des quiz uniques basés sur des critères spécifiques tels que la matière, le niveau et l'usage pédagogique.

Interface Web

Conception d'une interface utilisateur intuitive et réactive, basée sur FastAPI et Jinja2, pour interagir avec la plateforme.

FONDATIONS DU PROJET

Architecture Technique Modulaire



L'architecture est construite autour de micro-services et de bases de données spécialisées, garantissant une bonne séparation des préoccupations :

API & Front-end

- ***Fast API*** *: Moteur principal gérant l'API REST et le rendu des pages HTML.*
- ***Templates Jinja2*** *: Utilisés pour générer les interface utilisateurs (login, menu, import ETL, gestion des quiz).*

Stockage

- ***SQLite*** *: Base de données relationnelle légère pour la gestion des utilisateurs, des rôles et des journaux d'authentification.*
- ***MongoDB*** *: Base NoSQL optimisée pour le stockage flexible du contenu pédagogique (questions et quiz générés).*

La sécurité est intégrée via la gestion des sessions et l'application stricte des dépendances `require_roles` *sur les endpoints critiques.*

ACCÈS SÉCURISÉ

Le Flux d'Authentification Utilisateur

L'authentification est le premier rempart de sécurité de la plateforme.
Le processus est linéaire et assure que seuls les utilisateurs autorisés accèdent aux fonctionnalités sensibles.

Soumission

L'utilisateur soumet ses identifiants via le formulaire `HTML /login`.

Vérification

Le système vérifie la correspondance dans la base SQLite et hache le mot de passe via `bcrypt`.

Création de Session

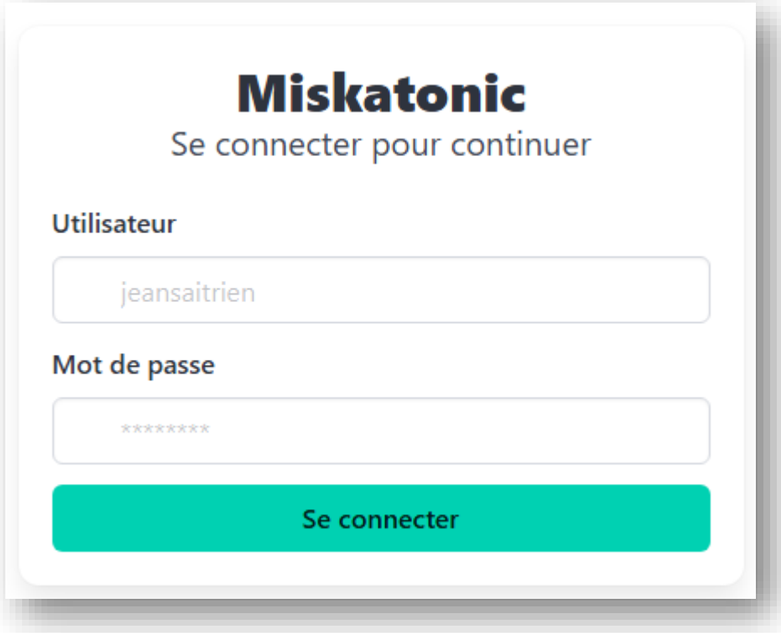
En cas de succès, une session utilisateur sécurisée est créée et maintenue.

Redirection

L'utilisateur est redirigé vers la page d'accueil `/menu`.

Accès aux Modules

La session autorise désormais l'accès aux routes protégées, telles que `/etl` *(importation) et* `/quizzes` *(génération).*



> Mécanisme de session minimal pour ce projet, signé avec une clé.

QUALITÉ DES DONNÉES

Le Pipeline ETL : De CSV à MongoDB

Le processus d'Extraction, Transformation et Chargement (ETL) est crucial pour ingérer de nouvelles questions dans la base de données de manière fiable.

1. Upload Fichier

L'enseignant ou l'administrateur télécharge le fichier de questions au format CSV via l'interface web dédiée.

2. Lecture et Normalisation

Le pipeline lit le CSV et standardise les noms de colonnes et les formats de données pour correspondre au modèle interne.

3. Correction Floue (Fuzzy Matching)

Application de `rapidfuzz` *pour corriger les variations légères ou les fautes de frappe dans les champs* `matières` *et* `usage`*, assurant l'uniformité des catégories.*

4. Validation Structurale

Vérification que chaque question respecte les règles minimales : au moins deux propositions de réponse et au moins une réponse correcte marquée comme telle.

5. Insertion MongoDB

Les questions validées sont enregistrées dans la collection MongoDB via le service `ServiceQuestion.create`*.*

6. Rapport de Log

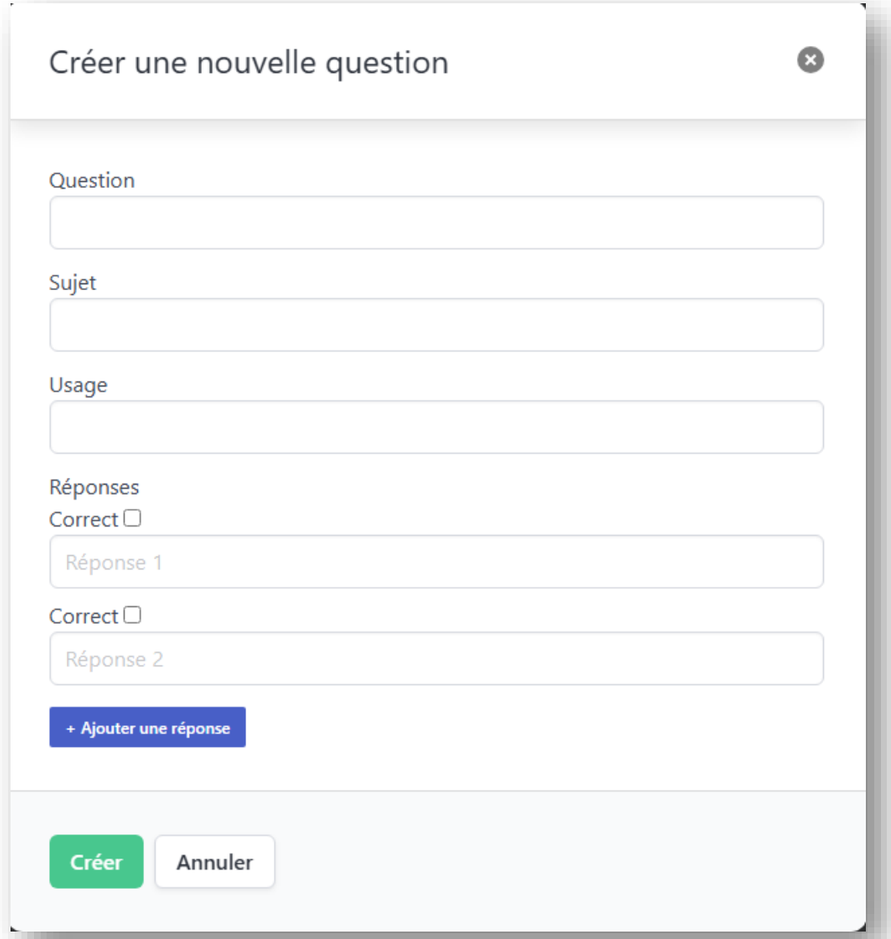
Génération d'un rapport CSV de log détaillé stocké dans `data/log`*, traçant l'état (succès/échec) de chaque ligne importée.*

BASE DE CONNAISSANCES

Structure du Modèle de Question

Chaque question est stockée comme un document JSON dans MongoDB, suivant un schéma strict défini par Pydantic pour garantir la cohérence des données.

```
{
  "question": "Que signifie le sigle No-SQL ?",
  "subject": "BDD",
  "use": "Test de positionnement",
  "responses": [
  {"answer":"Pas seulement SQL","isCorrect":true},
  {"answer":"Pas de SQL"},
  {"answer":"Pas tout SQL"}
  ],
  "remark": null,
  "metadata": { "author": "enseignant_X"},
  "date_creation": { "$date": "2025-09-17" },
  "date_modification": { "$date": "2025-09-17" }
}
```



Validation Pydantic

Le modèle exige un minimum de 2 réponses par question et qu'au moins 1 des réponses soit désignée comme correcte (`isCorrect: true`). Cette validation est faite en amont lors de l'ETL.

Indexation MongoDB

Un index combiné sur `{subject, use, question_key}` (clé unique générée à partir de la question) assure des recherches rapides lors de la sélection des questions pour la génération de quiz.

Statut d'Activité

Le champ `active: true` permet de désactiver temporairement des questions sans les supprimer, facilitant la maintenance du corpus.

LE CŒUR DE MISKATONIC

Algorithme de Génération de Quiz

La fonction de génération permet aux enseignants de créer rapidement des quiz uniques en sélectionnant des critères spécifiques (matière, usage) à partir de la base de questions.

Appel de l'Endpoint

L'utilisateur initie la création du quiz via l'endpoint sécurisé `/quizs/generate`*, en spécifiant les paramètres désirés (nombre de questions, matières, usage).*

Sélection Aléatoire

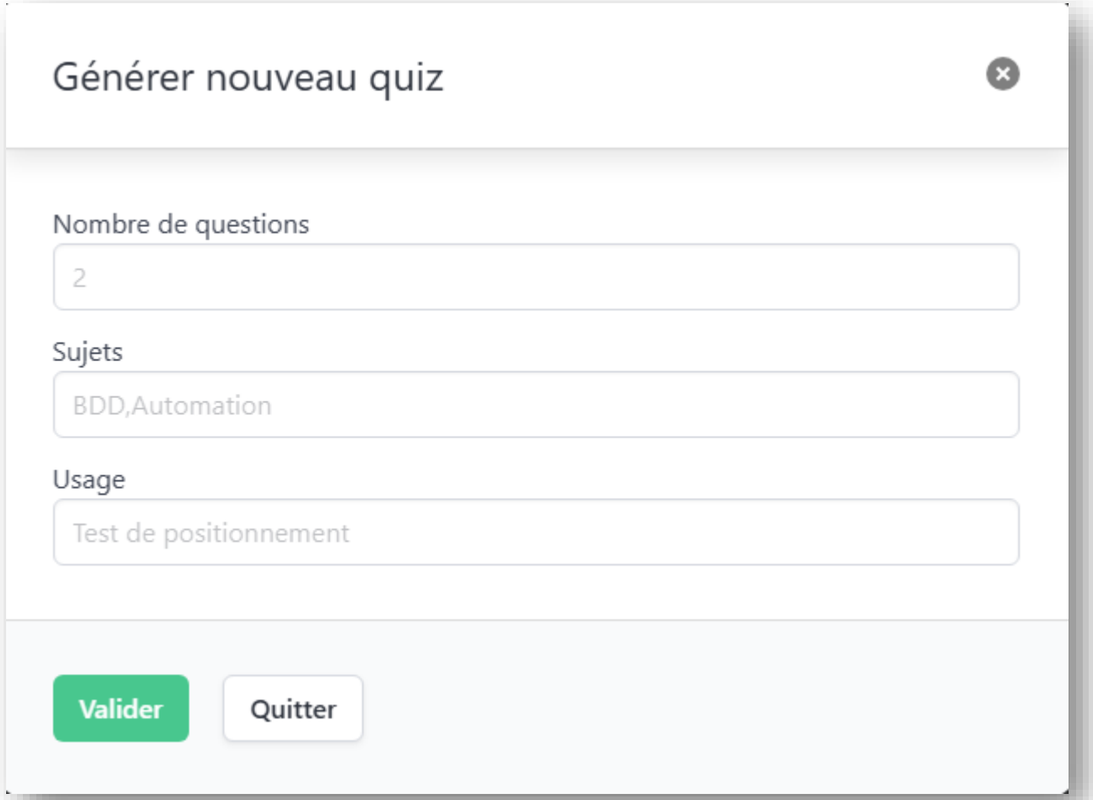
La logique interne utilise `random.sample` *pour tirer aléatoirement un nombre n de questions pertinentes, assurant l'unicité des quiz.*

Construction du Modèle

Un `QuizModel` *est créé, encapsulant la liste des questions sélectionnées ainsi que toutes les métadonnées associées (matières, usage, auteur, date, etc.).*

Persistance et ID

Le quiz complet est inséré dans MongoDB via `ServiceQuiz.create`*. La réponse JSON renvoie une confirmation et l'*`quiz_id` *unique.*



LE PRODUIT FINAL

Structure du Modèle de Quiz

Le modèle Quiz sert de conteneur pour un ensemble de questions spécifiques, ajoutant une couche de métadonnées essentielles à la traçabilité et à l'organisation.

```
{
questions: list[QuestionModel]
    subjects: ["BDD", "Python"],
    use: "Examen",
    metadata: { "author": "enseignant_X"},
    date_modification: "2025-10-10T10:00:00Z",
    date_creation: "2025-10-10T10:00:00Z",
"active": true
}
```

Composants Clés

- ***subjects*** *: Liste des matières couvertes par le quiz.*
- ***use*** *: Le contexte pédagogique (ex: "Examen", "Test rapide", "Révision").*
- ***questions*** *: Un tableau ordonné des documents* `QuestionModel` *sélectionnés.*
- ***Métadonnées*** *: Incluent l'auteur de la génération*

Accessibilité

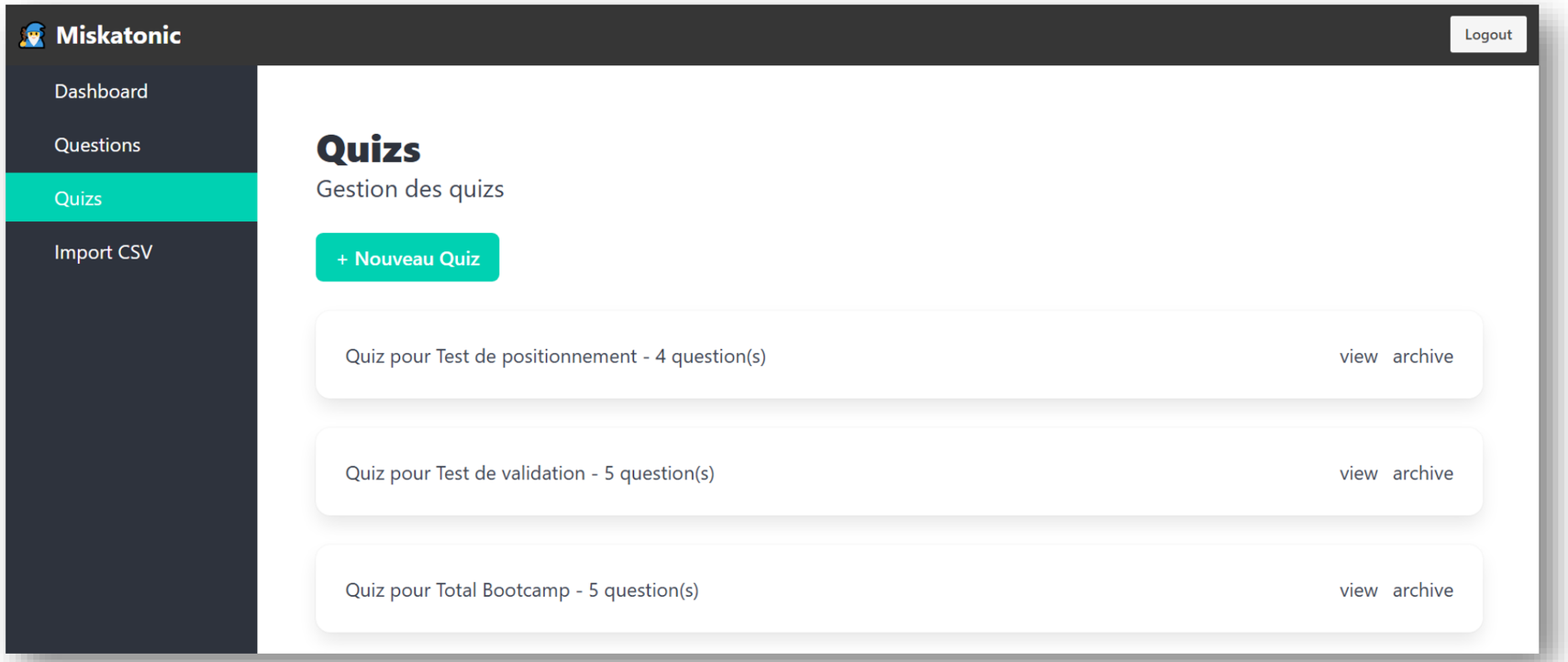
Les quiz générés peuvent être consultés de deux manières principales :

Interface HTML

Visualisation via l'interface web (/quizzes).

API JSON

Accès aux données brutes pour d'éventuelles intégrations externes.



CONCLUSION

Forces et Axes d’Amélioration

Forces du Projet

- ***Architecture*** *: Séparation nette entre la présentation (Jinja2), la logique métier (FastAPI services) et la persistance (Mongo/SQLite).*
- ***Sécurité*** *: Mise en œuvre de l'authentification et de l'autorisation simples mais efficaces.*
- ***ETL Traçable*** *: Un pipeline d'importation automatisé avec un rapport de log détaillé pour chaque exécution.*
- ***Double BDD Cohérente*** *: Chaque base de données sert son propre objectif.*

Pistes d’Amélioration Futures

- *Optimiser la performance du fuzzy-matching pour les questions.*
- *Ajouter des menus déroulants dans l'interface web.*
- *Intégrer des statistiques d'utilisation et de performance des quiz.*
- *Implémenter un système de versionnage des quiz et des questions.*
- *Explorer l'intégration d'IA pour la suggestion de questions.*