Sorbonne Université
UFR des Sciences

Rapport final

Projet M1 ANDROIDE

Développement d'une interface d'évaluation automatique de textes

Étudiants

HIBAOUI Imane
NAJEM Mohamed
VU Hoang Nguyen

Encadrant

M. NEBEL Léo

Année universitaire

2024-2025

Table des matières

1 Introduction

L'application des nouvelles technologies se généralise dans de nombreux secteurs à travers le monde, y compris les services à la clientèle, l'assistance à la recherche scientifique, et en particulier l'éducation.

La correction des copies par les enseignants constitue un problème complexe : le traitement d'un grand nombre de devoirs soumis, en particulier les travaux rédigés, occupe non seulement une grande partie de leur temps libre, mais compromet également leur neutralité et objectivité lors de l'évaluation. Il arrive que certains contenus soient mal notés s'ils ne correspondent pas aux attentes ou aux sensibilités des correcteurs.

Face à la nécessité d'améliorer cette tâche, un outil a été mis au point : la correction et l'évaluation automatiques des travaux textuels.

1.1 Organisation

Ce projet vise à développer une plateforme web interactive de correction orthographique et grammaticale, avec un accent particulier sur l'analyse pédagogique du comportement d'écriture des étudiants.

L'interface permet aux étudiants de saisir un texte librement, d'obtenir une correction instantanée et d'enregistrer automatiquement chaque frappe (ainsi que la position du curseur) dans une base de données. Cela permet aux enseignants d'étudier la progression réelle de l'écriture et de comprendre comment les étudiants formulent et corrigent leurs phrases.

En complément, une interface dédiée aux enseignants a été intégrée : elle leur permet de se connecter de manière sécurisée, de créer des exercices personnalisés (rédigés en XML), et de suivre les résultats des textes soumis par les étudiants. Chaque exercice est lié à son auteur, ce qui facilite la gestion individualisée.

Le système s'appuie sur plusieurs outils et technologies :

- Django (framework web Python) pour la gestion du backend, de la base de données et de l'interface d'administration.
- JavaScript pour la capture en temps réel des frappes clavier et leur envoi automatique.
- spaCy, LanguageTool et pyspellchecker pour la détection des fautes orthographiques, grammaticales et de conjugaison.
- Bootstrap pour un rendu visuel moderne et responsive.
- SQLite pour le stockage des données durant la phase de développement.

Ce rapport décrit en détail l'architecture du projet, ses fonctionnalités principales, les choix techniques effectués et les perspectives d'évolution.

2 Implémentation de notre application

2.1 Architecture générale

```
myproject/
 manage.py
 db.sqlite3
 myproject/        ← notre projet
   settings.py
   urls.py
 templates/        ← template type
   base.html
 accounts/         ← authentification
    apps.py
    urls.py
    views.py
    templates/
       registration/
         login.html
         signup_choice.html
         student_signup.html
         professor_signup.html
       accounts/
          professor_dashboard.html
          student_dashboard.html
text_analysis/      ← analyse synthaxique
 templates/
    text_analysis/
       add_exercise.html
       home.html
```

2.2 Fonctionnalités principales

L'application intègre plusieurs fonctionnalités centrales, destinées à améliorer la qualité rédactionnelle des étudiants tout en offrant aux enseignants des outils de suivi avancés.

2.2.1 Correction orthographique et grammaticale

L'étudiant rédige son texte dans une interface intuitive, conçue pour faciliter la saisie et l'interaction. Lorsqu'il soumet son texte, une analyse automatique est déclenchée afin d'identifier et de corriger les fautes éventuelles. Cette analyse repose sur l'intégration de plusieurs outils de traitement du langage naturel.

Tout d'abord, le module **spaCy** est utilisé pour segmenter le texte en phrases, identifier les entités linguistiques, et préparer le texte pour les traitements ultérieurs. Ensuite, l'outil **LanguageTool**, via son API publique ou locale, détecte les erreurs grammaticales et les fautes de conjugaison, en se basant sur des règles grammaticales précises du français.

Enfin, **PySpellChecker** intervient pour repérer les fautes d'orthographe et suggérer des corrections appropriées à partir d'un dictionnaire de mots en français.

À l'issue de cette analyse, le système retourne une version corrigée du texte, accompagnée d'un *score de qualité linguistique*, d'une liste de suggestions de correction, ainsi que de messages pédagogiques. Ces messages ont pour but de sensibiliser l'étudiant à certaines erreurs fréquentes, telles que l'omission d'une majuscule en début de phrase.

Une fois l'analyse terminée, l'étudiant a la possibilité de sauvegarder son texte corrigé dans la base de données. Chaque sauvegarde conserve non seulement le texte lui-même, mais également le score attribué, ce qui permet un suivi des progrès de l'étudiant au fil du temps.

2.2.2 Suivi des frappes clavier (keylogging éducatif)

L'un des aspects innovants de notre application réside dans la mise en place d'un système de suivi des frappes clavier, conçu dans une optique pédagogique. Cette fonctionnalité permet d'analyser en détail la manière dont l'étudiant construit son texte, en capturant les différentes étapes de son processus d'écriture.

Concrètement, à chaque frappe de clavier ou à chaque déplacement du curseur, une requête *AJAX* est automatiquement envoyée au serveur. Ces interactions sont ensuite enregistrées dans la base de données, accompagnées de plusieurs métadonnées essentielles :

— le contenu textuel à l'instant de l'action,
— la position exacte du curseur dans le texte,
— un identifiant unique de session utilisateur,
— la date et l'heure précises de l'événement.

Grâce à cette collecte d'informations en temps réel, il devient possible de reconstruire l'évolution d'un texte depuis la première saisie jusqu'à sa version finale. Cette traçabilité permet non seulement d'observer les stratégies d'écriture employées par l'étudiant, mais aussi de mettre en évidence ses hésitations, ses corrections spontanées ou ses retours en arrière. Elle constitue ainsi un outil précieux pour les enseignants souhaitant accompagner les apprenants dans leur développement de compétences rédactionnelles.

2.3 Authentification et rôles

Le système d'authentification repose sur le framework Django, avec une personnalisation du modèle utilisateur (`CustomUser`) pour intégrer des rôles distincts : *étudiant* et *enseignant*. Lors de l'inscription, l'utilisateur choisit son rôle, ce qui détermine par la suite son interface d'accès et ses permissions.

Après connexion via l'interface standard fournie par Django, l'utilisateur est redirigé automatiquement vers l'espace qui lui correspond. Cette redirection est gérée dynamiquement en fonction du rôle détecté dans le modèle utilisateur. Ainsi :

— un étudiant est dirigé vers l'espace d'écriture assistée,
— un enseignant accède à son tableau de bord personnel pour gérer les exercices.

Ce mécanisme garantit une séparation claire entre les fonctionnalités accessibles aux différents types d'utilisateurs, tout en restant simple à maintenir et à étendre.

2.3.1 Espace enseignant et gestion des exercices

L'application propose un espace dédié aux enseignants, accessible uniquement après authentification. Ce système repose sur le module d'authentification intégré de Django, permettant de différencier les utilisateurs selon leur rôle (étudiant ou professeur) dès la connexion. Lorsqu'un professeur se connecte, il est automatiquement redirigé vers un tableau de bord personnalisé.

Depuis cet espace sécurisé, l'enseignant peut créer des exercices d'expression écrite en remplissant un formulaire contenant deux champs : un titre et un énoncé. Chaque exercice est associé à son auteur via une clé étrangère dans la base de données, garantissant ainsi un lien clair entre les contenus pédagogiques et leur créateur.

Le tableau de bord permet aussi une gestion complète des exercices :

— affichage de la liste des exercices créés par le professeur connecté,
— possibilité d'ajouter un nouvel exercice via un bouton dédié,
— suppression d'un exercice existant grâce à un bouton intégré avec confirmation,
— consultation directe de l'exercice dans l'espace étudiant (vue pré-remplie).

Cette interface offre ainsi aux enseignants un moyen simple et efficace de mettre à disposition des supports d'entraînement personnalisés, adaptés au niveau de leurs apprenants.

2.4 Dashboard étudiant : exercices et retour

Une fois connecté, l'étudiant accède à une interface simple et conviviale dédiée à la rédaction et à l'analyse de texte. Il peut, s'il le souhaite, sélectionner un exercice proposé par un enseignant via une liste déroulante.

L'espace central est occupé par une zone de saisie où l'étudiant peut rédiger librement. Une fois le texte soumis, une correction automatique est effectuée : fautes d'orthographe, erreurs grammaticales, conjugaison ou oubli de majuscule sont détectés, corrigés, et des suggestions sont affichées.

Les résultats de l'analyse comprennent :

— un texte corrigé,
— un score global de qualité,
— des messages pédagogiques sur les erreurs fréquentes,
— des suggestions de correction mot à mot.

2.4.1 Sauvegarde et administration des textes

L'application permet aux étudiants de sauvegarder les textes qu'ils rédigent après correction. Cette sauvegarde s'effectue via un simple bouton « Sauvegarder » intégré à l'interface, qui envoie les données vers le serveur via une requête POST. Chaque texte est alors enregistré dans la base de données avec son contenu, le score obtenu à l'analyse, ainsi que les métadonnées associées (comme la progression ou la position du curseur).

Une section de l'interface affiche la liste des textes sauvegardés par l'étudiant. Chaque élément de cette liste peut être consulté, ou supprimé à l'aide d'un bouton dédié. L'utilisateur conserve ainsi un historique de ses productions, ce qui favorise un suivi personnel des progrès dans le temps.

Du côté administratif, tous les textes enregistrés sont accessibles via l'interface Django Admin. Les administrateurs peuvent les visualiser, les trier ou les filtrer selon différents critères, notamment par utilisateur, par date ou par score. Cela facilite le contrôle, la modération ou l'analyse à grande échelle des productions écrites.

2.4.2 Modélisation des données

La structure des données repose sur plusieurs modèles relationnels définis avec le framework Django. Chaque modèle représente une entité fonctionnelle majeure de l'application, permettant de stocker, organiser et interroger efficacement les informations nécessaires à la correction, au suivi des utilisateurs et à la gestion pédagogique.

Modèle `SavedText` Le modèle `SavedText` enregistre les textes soumis par les utilisateurs (principalement les étudiants), après une analyse et une éventuelle correction. Il permet de conserver une trace des productions textuelles, avec les informations suivantes :

- le texte complet saisi,
- le score obtenu à l'analyse linguistique (en pourcentage),
- la date de création et la dernière mise à jour,
- les métadonnées facultatives liées à la progression de l'écriture (comme la position du curseur ou l'évolution du texte).

Ce modèle est essentiel pour constituer un historique de l'activité de l'utilisateur et pour favoriser le suivi longitudinal des progrès.

Modèle `TypingEvent` Ce modèle enregistre les événements de frappe clavier, dans une logique de suivi éducatif détaillé. Chaque entrée représente une action de l'utilisateur pendant l'écriture, avec :

- un identifiant de session ($\texttt{session}_id$)$uniquepourdistinguerlesutilisateurssansnécessiteruneauth$
- le contenu intégral du texte à cet instant,
- un horodatage précis (`timestamp`) de l'événement.

Ces données permettent de reconstruire fidèlement l'évolution d'un texte, d'identifier les hésitations, les suppressions et les corrections spontanées de l'élève, ce qui ouvre des perspectives intéressantes pour l'analyse des stratégies d'écriture.

Modèle `UserTyping` Il s'agit d'un modèle complémentaire à `TypingEvent`, utilisé pour des tests ou des traitements simplifiés. Il contient également des informations sur la progression de l'écriture, mais de manière agrégée. Les champs incluent :

- l'utilisateur authentifié (quand disponible),
- la position du curseur,
- la version actuelle du texte,
- la date d'enregistrement.

Modèle `Exercise` Le modèle `Exercise` gère les exercices créés par les enseignants. Il comprend :

- un titre représentatif de l'exercice,
- un contenu libre (souvent en prose) représentant l'énoncé à traiter,

— une relation vers l'auteur de l'exercice (via une clé étrangère vers le modèle utilisateur),
— la date de création.

Chaque enseignant peut créer et gérer ses propres exercices depuis un tableau de bord dédié, et ceux-ci sont ensuite proposés aux étudiants comme sujets de production écrite.

Modèle `CustomUser` Le projet repose sur un modèle utilisateur personnalisé, dérivé de celui de Django, appelé `CustomUser`. Il permet de gérer différents rôles au sein de l'application, notamment :

— les étudiants, qui accèdent à l'espace d'écriture et de correction,
— les enseignants, qui disposent d'un espace de gestion des exercices,
— les administrateurs, qui accèdent à toutes les données via l'interface Django Admin.

Cette distinction des rôles permet d'adapter dynamiquement les interfaces et les permissions à chaque type d'utilisateur.

2.4.3 Relations entre les modèles

Chaque `Exercise` est lié à un utilisateur via une relation `ForeignKey`. Les textes (`SavedText`) ne sont pas encore reliés à un utilisateur spécifique, mais le système peut être facilement étendu pour les lier à un `CustomUser`.

Les événements de frappe sont reliés à une session navigateur via une chaîne `session_id`, ce qui permet de suivre même des utilisateurs non connectés.

2.5 Paramétrage du projet

Le fichier settings.py configure l'environnement du projet. En mode développement, DEBUG est activé et la base de données utilisée est SQLite. Les applications principales (`text_analysis`, `accounts`) sont enregistrées, avec un modèle utilisateur personnalisé (CustomUser).

Les chemins vers les templates et les fichiers statiques sont spécifiés, et l'interface est localisée en français avec le fuseau horaire de Paris.

L'authentification est configurée avec des redirections après connexion et déconnexion. La sécurité CSRF est temporairement allégée pour simplifier les tests en local, mais doit être réactivée en production.

2.6 Sécurité et limites

2.6.1 Sécurité

À ce stade, le projet est principalement en phase de développement local, ce qui implique certaines concessions temporaires en matière de sécurité. Ces ajustements sont nécessaires pour faciliter les tests et l'itération rapide, mais ils ne sont pas adaptés à un déploiement en production. Les principales considérations de sécurité sont les suivantes :

Mode `DEBUG` Le projet utilise encore le mode `DEBUG = True`, qui permet d'afficher les messages d'erreur détaillés en cas de problème serveur (exceptions, erreurs 404, erreurs SQL, etc.). Bien que cela soit utile pour le débogage, cela expose des informations sensibles

sur l'infrastructure de l'application (comme les chemins de fichiers ou les requêtes internes) et doit impérativement être désactivé en production.

Protection CSRF désactivée temporairement Le middleware CSRF de Django, qui protège contre les attaques par requête inter-site (Cross-Site Request Forgery), est partiellement désactivé pour certaines routes POST utilisées par des appels JavaScript (AJAX) non sécurisés, notamment pour le suivi des frappes clavier. Cette désactivation facilite les tests de développement, mais affaiblit la protection contre des actions non autorisées simulées par un site tiers.

De plus, les paramètres `CSRF_COOKIE_SECURE = False` et `CSRF_COOKIE_HTTPONLY = False` permettent respectivement l'envoi de cookies CSRF sur des connexions non sécurisées (HTTP), et leur accès via JavaScript. Ces pratiques sont tolérées localement, mais doivent être strictement interdites en environnement réel.

Accès restreint à l'administration Django. L'interface d'administration native de Django est activée et n'est accessible qu'aux comptes disposant du statut `superuser`. Cela permet aux développeurs de gérer facilement les données pendant le développement, tout en limitant l'accès à cette interface puissante.

Gestion des rôles et permissions Comme expliqué juste avant, nous utilisons un système d'authentification personnalisé (`CustomUser`) intégrant une distinction entre les rôles d'enseignant et d'étudiant. Toutefois, la logique de permissions associées à ces rôles reste simple pour l'instant. Il n'y a pas encore de mécanisme avancé de contrôle d'accès granulaire, par exemple pour empêcher un étudiant d'accéder à certaines vues ou données réservées aux enseignants. Cela constitue une piste d'amélioration importante avant tout passage en production.

Avant un déploiement public, voici les mesures de sécurisation que nous mettrons en œuvre :

- Activer le mode `DEBUG = False` et définir une liste stricte d'hôtes autorisés,
- Réactiver le middleware CSRF sur toutes les requêtes, y compris via AJAX, avec l'envoi sécurisé des jetons,
- Mettre en place un chiffrement SSL/TLS pour toutes les communications,
- Renforcer le système de permissions (via des décorateurs ou des classes d'autorisations),
- Éventuellement, auditer le code avec des outils de sécurité automatisés.

3 Conclusion

Ce projet nous a permis de démontrer la faisabilité d'un outil pédagogique combinant la correction linguistique automatisée et l'analyse comportementale de l'écriture. En tant qu'étudiants en Master 1 Informatique, nous avons eu l'opportunité de concevoir et développer cette application de bout en bout, en mettant en œuvre des outils concrets comme `Django`, `spaCy`, `LanguageTool` et `PySpellChecker`.

Ce travail a été particulièrement enrichissant, car il nous a permis de découvrir un domaine peu abordé dans notre formation initiale : celui de l'évaluation automatique de l'écriture (Automatic Writing Evaluation, AWE). Nous avons également appris à collaborer efficacement en équipe, à structurer un projet en respectant les contraintes d'architecture logicielle, de modélisation des données, et d'expérience utilisateur.

L'outil que nous avons conçu permet aux étudiants de rédiger et corriger leurs productions écrites dans une interface claire et fonctionnelle. En parallèle, il offre aux enseignants un espace dédié pour créer des exercices personnalisés et suivre la progression des apprenants de manière détaillée, y compris par l'enregistrement des frappes clavier.

Ce projet nous a donné l'occasion concrète d'appliquer nos compétences en développement web et en traitement automatique du langage, tout en explorant une application pédagogique à fort potentiel dans le domaine de l'éducation numérique.

3.1 Perspectives d'évolution

Au terme de ce projet, plusieurs pistes concrètes peuvent être envisagées pour enrichir les fonctionnalités existantes et rendre l'application plus robuste, évolutive et pédagogique.

Authentification étendue Actuellement, certaines données liées aux frappes clavier sont collectées de manière anonyme ou via un identifiant de session générique. Une évolution logique serait de relier chaque frappe à un utilisateur authentifié. Cela permettrait de construire un historique détaillé de la progression de chaque étudiant, et d'assurer un meilleur suivi pédagogique individualisé.

Session dynamique La session utilisée pour enregistrer les événements de frappe est actuellement simulée à l'aide d'un identifiant statique. Pour améliorer la fiabilité des données, il serait pertinent de générer dynamiquement un identifiant unique propre à chaque utilisateur ou session navigateur. Cela permettrait de mieux distinguer les différentes sessions de travail et d'éviter les collisions ou les erreurs d'attribution.

Outils de filtrage et d'export Afin de faciliter le travail des enseignants, des outils d'analyse supplémentaires pourraient être intégrés : filtres par étudiant, date ou score, visualisation graphique de la progression, export au format CSV ou PDF. Ces fonctionnalités rendraient l'interface plus complète et offriraient des possibilités concrètes d'évaluation ou de recherche pédagogique.

Analyse comportementale avancée Au-delà du simple score de correction, le système pourrait intégrer de nouveaux indicateurs tels que le temps de frappe moyen, les pauses

prolongées, les réécritures fréquentes ou les retours arrière. Ces données fourniraient un aperçu plus qualitatif du processus d'écriture et des stratégies adoptées par l'étudiant.

Éditeur XML intégré Enfin, pour améliorer l'expérience des enseignants dans la création de contenus, un éditeur visuel XML pourrait être intégré. Il permettrait de structurer les exercices de manière plus souple (par exemple, en ajoutant des consignes, des balises d'analyse ou des métadonnées), sans avoir à manipuler manuellement le code ou les formats bruts.

Ces évolutions représentent autant d'opportunités d'approfondir le projet en M2, dans le cadre d'un stage ou d'un mémoire de recherche appliquée.

Références

[1] Explosion AI, *spaCy : Industrial-Strength Natural Language Processing in Python*, disponible à https://spacy.io/

[2] LanguageTool, *LanguageTool : Open Source Grammar and Style Checker*, disponible à https://languagetool.org/

[3] Barrust, *pyspellchecker : Pure Python Spell Checking*, disponible à https://pyspellchecker.readthedocs.io/

[4] Django Software Foundation, *Django Documentation*, disponible à https://docs.djangoproject.com/en/5.2/

[5] Ranalli, J., Link, S., & Chukharev-Hudilainen, E., *Automated writing evaluation use in second language classrooms*, System, 2024. Disponible à https://www.sciencedirect.com/science/article/pii/S0346251X24001143

[6] Wang, Y., Luo, X., Liu, C., & Tu, J., *An Integrated Automatic Writing Evaluation and SVVR Approach to Improve Students' EFL Writing Performance*, 2022. Disponible à https://www.researchgate.net/publication/363620007

[7] Alharbi, M., *Automated Writing Evaluation (AWE) in Higher Education*, Pegem Journal of Education and Instruction, 2021. Disponible à https://files.eric.ed.gov/fulltext/EJ1320134.pdf

[8] RAND Corporation, *What Do Teachers Want to See in Automated Writing Evaluation Tools ?*, 2020. Disponible à https://www.rand.org/pubs/commentary/2020/07/what-do-teachers-want-to-see-in-automated-writing-evaluation.html

4 Manuel d'utilisateur

4.1 Installation

4.1.1 Pré-requis

Avant d'installer et d'utiliser ce logiciel, assurez-vous que votre environnement respecte les conditions préalables suivantes :

- Le logiciel est compatible avec `Python 3.7` et toutes les versions supérieures de Python.
- Les bibliothèques suivantes doivent être installées dans votre environnement :
  - `spacy` : une bibliothèque de traitement du langage naturel.
  - `transformers` : pour l'intégration de modèles pré-entraînés pour l'analyse de texte.
  - `django` : pour le développement du backend de l'application web.
  - `pyspellchecker` : pour la correction orthographique en français.
- Ces dépendances sont listées dans le fichier `requirements.txt`.

4.1.2 Installation

Une fois les pré-requis installés, vous pouvez installer les bibliothèques nécessaires à l'aide de `pip`. Dans votre terminal, naviguez jusqu'au répertoire du projet et exécutez la commande suivante pour installer toutes les dépendances :

```
pip install -r requirements.txt
```

Cette commande lira le fichier `requirements.txt`, et installera toutes les bibliothèques nécessaires à l'exécution du logiciel.

De plus, pour utiliser `spacy` avec la langue française, vous devez télécharger le modèle de langue français. Exécutez la commande suivante dans votre terminal pour l'obtenir :

```
python3 -m spacy download fr_core_news_sm
```

4.2 Utilisation

4.2.1 Lancer le programme

Une fois que toutes les dépendances sont installées, vous pouvez lancer le programme en exécutant le serveur de développement Django. Ouvrez votre terminal dans le répertoire du projet et exécutez la commande suivante :

```
python3 manage.py runserver
```

Cela démarrera le serveur de développement local sur `http://127.0.0.1:8000/`, où vous pourrez interagir avec l'application. Vous pourrez y accéder via un navigateur web pour tester l'ajout d'exercices, l'analyse de texte, et d'autres fonctionnalités du système.

Assurez-vous que le serveur fonctionne correctement et que l'application est accessible à l'adresse mentionnée.

4.2.2 Accès à l'application

Une fois le serveur démarré, vous pourrez naviguer vers l'interface utilisateur à l'adresse suivante dans votre navigateur :

`http://127.0.0.1:8000/`

À partir de là, les professeurs pourront ajouter des exercices et les étudiants pourront les résoudre. Vous serez redirigé vers les pages appropriées selon votre rôle après vous être connecté.

5 Annexes

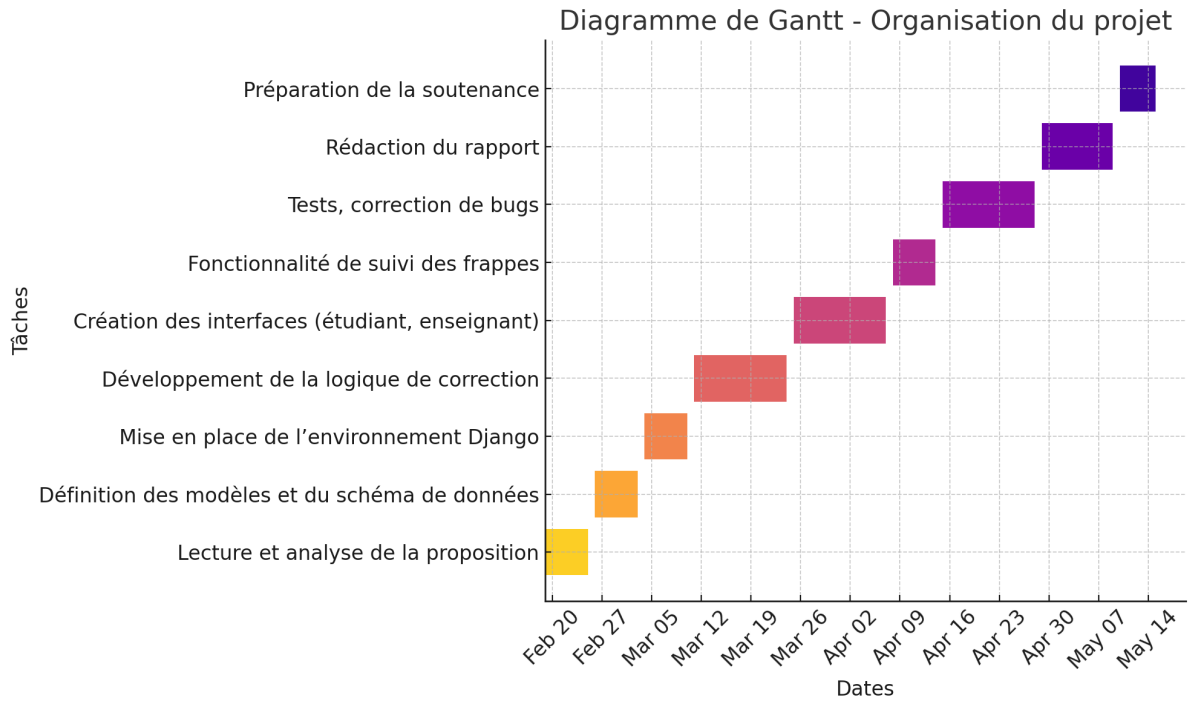

FIGURE 1 – Diagramme de Gantt

Revenir en haut.

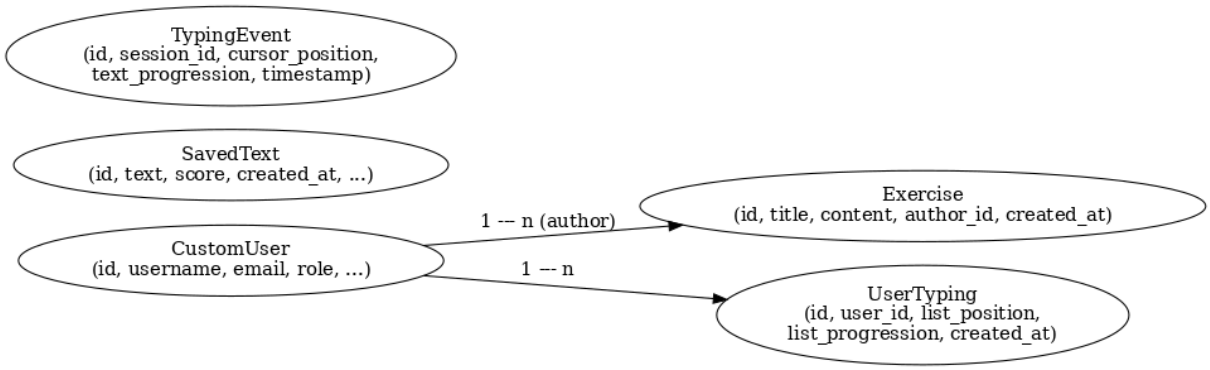

FIGURE 2 – Modèle de nos données sur Django

Revenir en haut.

FIGURE 3 – index

Revenir en haut.

FIGURE 4 – Choix du rôle

Revenir en haut.

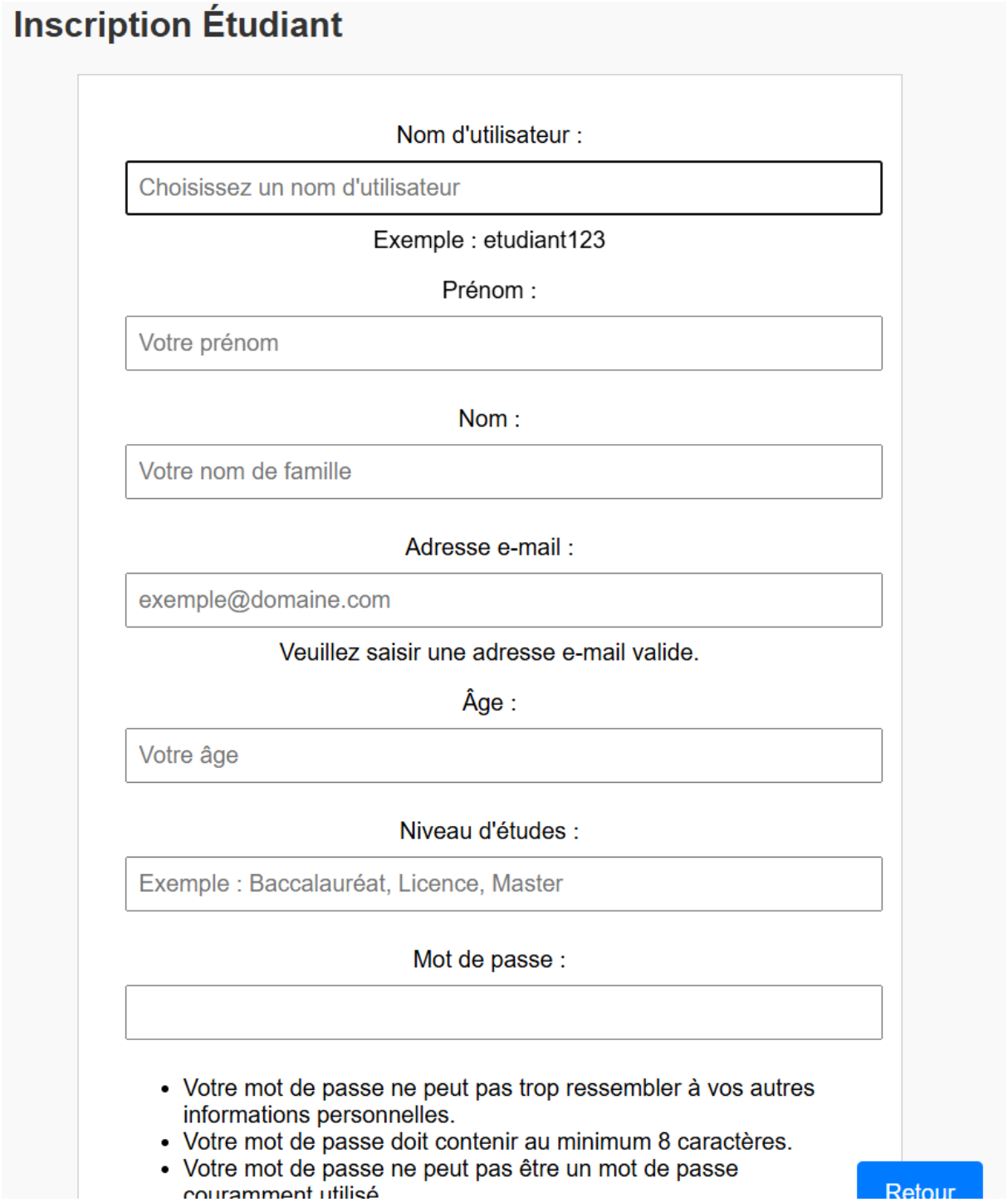

FIGURE 5 – Exemple de formulaire d'inscription pour un étudiant
Revenir en haut.

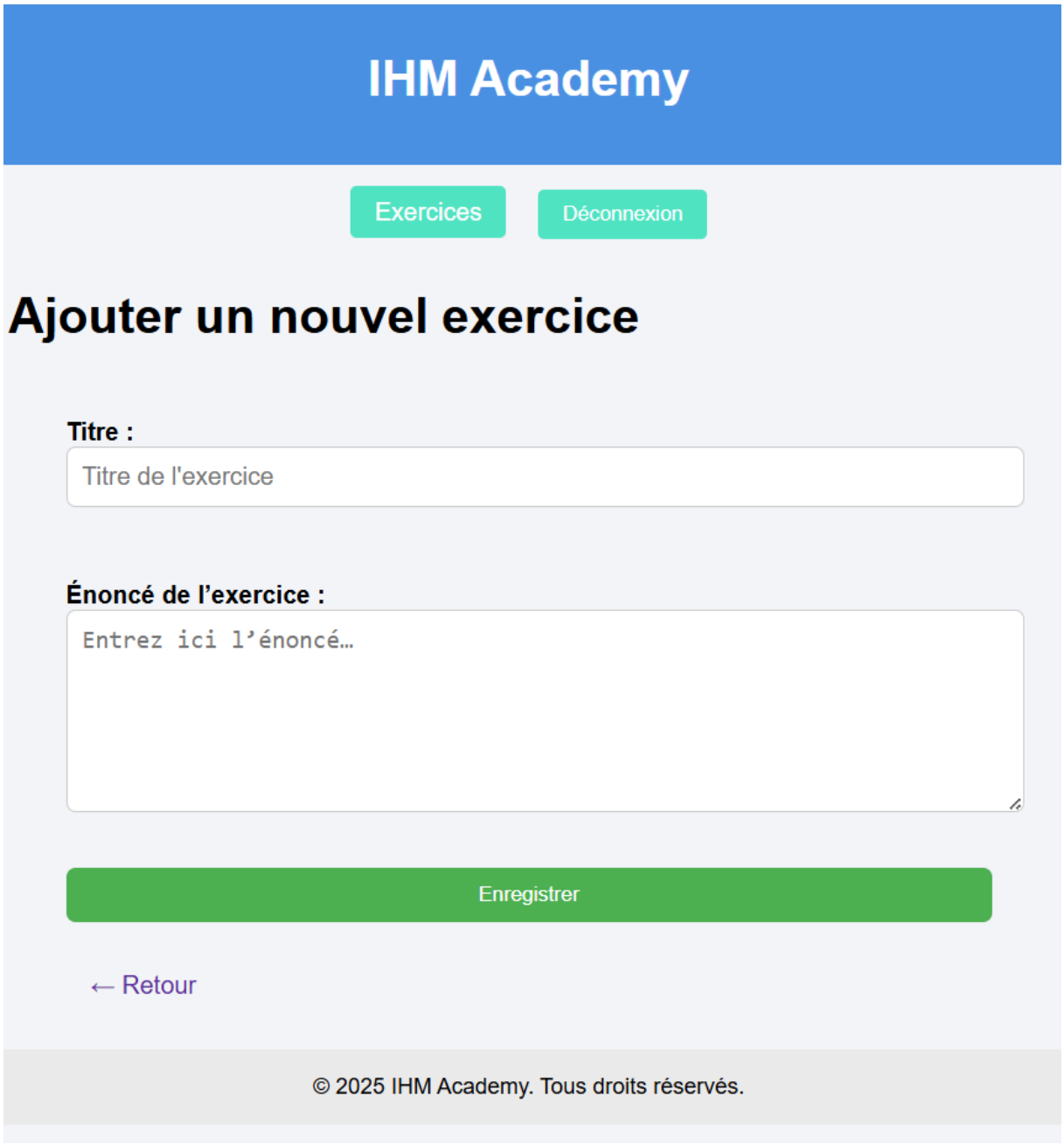

FIGURE 6 – Onglet de création d'un nouvel exercice réservé à chaque professeur
Revenir en haut.

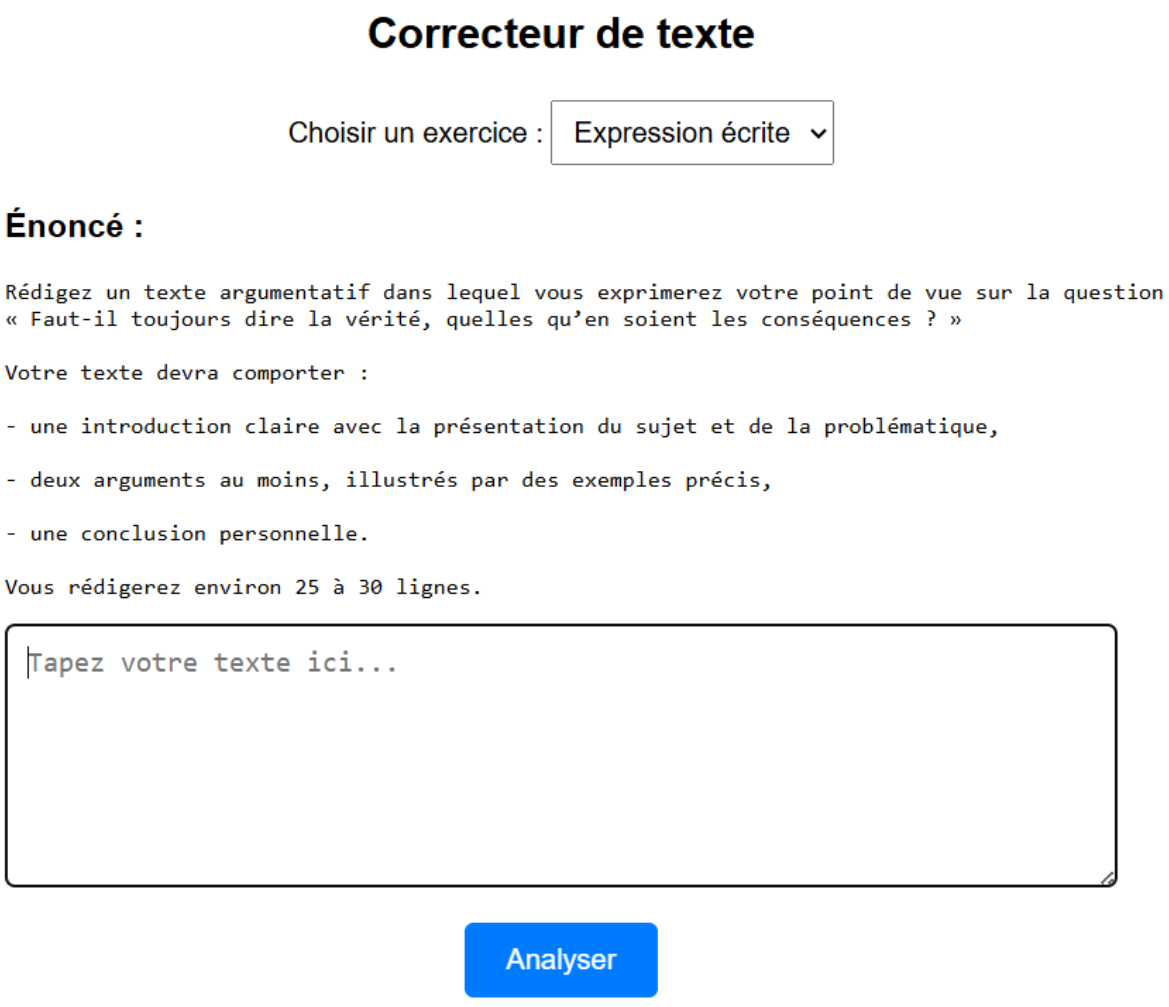

FIGURE 7 – Exemple d'exercice

Revenir en haut.

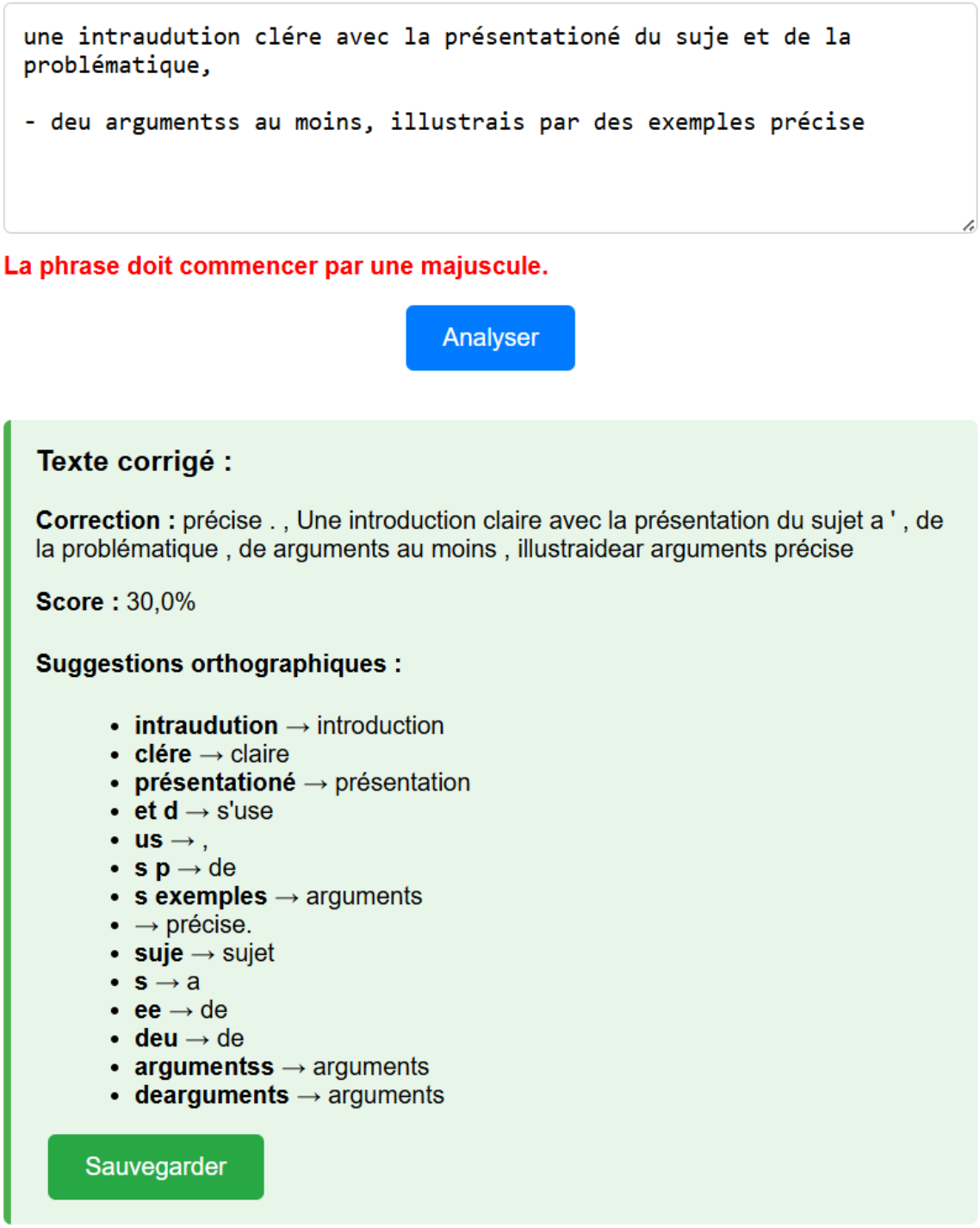

FIGURE 8 – Exemple de feedback et correction de la rédaction d'un étudiant
Revenir en haut.