

RAPPORT PROJET DJANGO

"BIBLIOBOX"

APPLICATION DE RÉSERVATION DE BOX SILENCIEUSES



Sommaire

- I. Introduction : Contexte du projet et objectifs à atteindre
- II. Déroulement du projet et organisation
- III. Analyse détaillée des pages de l'application
- IV. Conclusion : Améliorations et perspectives futures

Groupe :ASLIMI Kawther, BALDE DIOULDE Aissata, AFOUCHAL Assia

I- Introduction : Contexte du projet et objectifs à atteindre

A. Contexte du projet

Le projet a pour but de développer une application web qui permet aux étudiants de l'université Paris Nanterre de réserver des boxes silencieux pour passer des appels. Le but est de leur offrir un espace calme sans avoir à sortir de la bibliothèque (ou du bâtiment de Droit), afin qu'ils puissent passer leur appels en toute tranquillité sans perturber les autres ni être dérangés par le bruit environnant.

Notre application est pensée pour être simple et pratique. Chaque étudiant a droit à deux créneaux de 15 minutes par semaine. S'il change d'avis ou a un empêchement il peut annuler sa réservation et en prendre une autre, mais il devra attendre 24 heures avant de pouvoir réserver à nouveau pour la semaine suivante. Cette règle a été mise en place par sécurité et éviter les abus.

Les étudiants pourront garder un œil sur leurs réservations passées et futures via la page historique. L'objectif est de simplifier la vie universitaire des étudiants et de leur permettre de rester concentrés sur leur révisions sans avoir à se soucier de trouver un endroit tranquille pour passer un appel important.

Nous avons développé l'application en veillant à ce qu'elle facile d'utilisation, la plus intuitive possible en mettant un texte décrivant son fonctionnement dans la page d'accueil. Le but est que la réservation d'un box devienne un réflexe simple et rapide.

B. Technologie et outils utilisés

Pour le développement de notre application, nous avons utilisé plusieurs technologies et outils qui nous ont été imposés lors de l'énoncé du projet. La consigne était de coder notre application sur Python Django pour le backend, HTML, CSS, JavaScript pour le frontend, et SQLite pour la base de données.

Nous avons choisi de travailler sur l'éditeur de code Visual Studio Code, simple d'utilisation et possédant de nombreuses fonctionnalités notamment le fait de pouvoir coder plusieurs langages dans le même éditeur de code paraissant comme idéal pour notre cas.

Pour la partie backend, nous avons utilisé Django, un framework web en Python qui permet de créer des applications web facilement. Il nous aide à gérer les données, afficher les pages et ajouter des fonctionnalités comme la connexion des étudiants et la gestion des réservations des boxes.

Concernant la base de données, nous avons décidé d'utiliser SQLite plutôt que MySQL. Son intégration à Django est simple et elle ne nécessite pas de configuration particulière.

Enfin, concernant le frontend, nous avons utilisé HTML pour structurer nos pages, CSS pour apporter du style et du design, et JavaScript pour des interactions dynamiques.

Au cours de ce projet, plusieurs versions ont été réalisées en fonction de l'avancement du projet. Dans un premier temps nous avons utilisé Google Drive pour partager les fichiers car nous étions familière avec cet outil. Toutefois, nous avons rapidement constaté que cet outil n'était pas optimal pour collaborer efficacement. Nous avons donc migré vers GitHub où vous trouverez notre version finale du projet.

GitHub s'est révélé être un outil essentiel, notamment pour le déploiement de notre application sur un serveur, ce qui nous a permis de garantir l'accessibilité du projet et d'éviter de potentiel problème technique en local lors de la soutenance. Pour ce déploiement, nous avons utilisé la plateforme Render, qui nous a permis de mettre notre projet en ligne directement à partir de notre dépôt sur GitHub.

N'étant pas interdites pour ce projet, les IA génératives, nous ont aidées notamment dans la résolution de problèmes rencontrés lors de la conception de notre application.

II. Déroulement du projet et organisation

Notre groupe est composé de trois membres. Initialement nous somme partie sur une répartition simple, une personne s'occupe du backend, une autre du frontend et la dernière de la base de donnée. C'est globalement notre répartition mais au fil du projet nous avons réalisé que cette répartition n'était ni optimale ni enrichissante. Nous avons donc conservé cette répartition de base pour avancer sur nos parties respectives, tout en nous impliquant davantage dans les autres aspects de notre projet afin de mieux collaborer.

A. Détails de la répartition des tâches

Concernant le frontend (HTML, CSS, JavaScript), cette tâche a été déléguée à *Kawther ASLIMI*. Elle s'est également occupée de l'intégration de l'envoi et de la vérification des codes de confirmation dans le fichier `views.py` (`def index`, `def vérification` et `def verify_code`).

Pour le backend (Python Django), cette tâche a été déléguée à *Aïssata DIOULDE BALDE*. Elle s'est concentrée sur l'implémentation des fonctionnalités essentielles au bon fonctionnement de l'application, notamment la gestion des fichiers principaux du framework Django (`views.py`, `settings.py`, `urls.py`,...) et la gestion des réservations via différentes fonctions.

Enfin, pour la base de données (SQLite), cette tâche a été déléguée à *Assia AFOUCHAL*. Elle s'est occupée de la gestion de la base de données en créant les tables nécessaires (StudentProfile, Box, TimeSlot, Reservation, VerificationCode). Elle a également développé un script Python pour générer automatiquement les créneaux de 15 minutes entre 8H30 et 19H45 pour l'année 2025 en excluant les jours fériés et les week-ends, afin de gagner du temps et d'éviter une implémentation manuel qui serait beaucoup trop long et pas optimale.

Cette répartition des tâches ne nous a pas pour autant empêchés de travailler ensemble et d'échanger régulièrement sur les différentes parties du projet que ce soit pour avoir des inspirations ou pour régler un problème survenu dans le code. L'entraide au sein de notre équipe a joué un rôle essentiel dans notre groupe pour surmonter les difficultés plus efficacement.

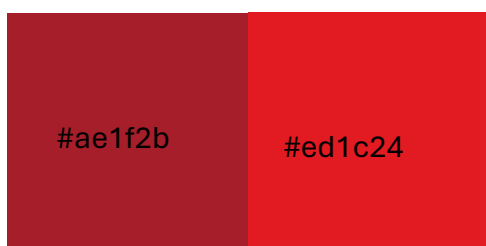
B. Organisation du travail

Le travail a principalement été réalisé à distance, ce qui nous a obligés à adopter une bonne organisation pour avancer efficacement sur le projet. On utilisé plusieurs outils pour collaborer comme *WhatsApp* pour communiquer simplement sur l'avancement du projet. *Google Drive* pour le partage des codes, *Google doc* pour la conception du rapport. Pour plus de professionnalisme et facilité nous avons décidé de partager le code sur un dépôt GitHub, ce qui nous a permis de mieux gérer les différentes versions du projet et de travailler de manière plus structurée.

III. Analyse détaillée des pages de l'application

A. Direction artistique de l'application

Nous avons choisi de conserver l'identité visuelle d l'Université Paris Nanterre afin d'assurer une continuité visuelle et de ne pas dépayser les étudiants. L'objectif était qu'ils puissent retrouver des repères familiers et s'adapter rapidement à l'interface de l'application. Pour cela nous avons repris les couleurs de l'université, notamment deux teintes de rouges distinctes :



Ces couleurs ont été utilisées stratégiquement sur divers éléments de l'interface, tels que les boutons, les titres, les sections, afin de renforcer l'identité visuelle et assurer une reconnaissance immédiate. Le contraste entre ces couleurs dynamiques et des tons plus neutre apporte un équilibre visuel et améliore la lisibilité et rend la navigation plus intuitive et agréable. Concernant la police d'écriture nous avons opté pour la plus part de nos texte pour une police moderne et lisible (Poppins).

Nous nous sommes largement inspirés du design du site de la bibliothèque universitaire de Paris Nanterre, notamment pour la structure du header et du footer, afin de proposer une interface qui s'intègre naturellement à l'ensemble des sites appartenant à l'université.

B. [Page d'accueil \(index.html\)](#)



1. [Fonctionnalités](#)

La page d'accueil est la vitrine de notre application joue un rôle central dans l'expérience utilisateur. Nous avons décidé d'inclure un texte explicatif décrivant le fonctionnement de l'application ainsi que ses principales contraintes. Cette approche vis à offrir aux étudiants une compréhension du principe de notre application et facilite la prise en main et la navigation.


Qu'est-ce qu'un box silencieux ?

📞 Les **boxes silencieux** sont là pour vous ! Besoin de passer un appel en toute tranquillité sans quitter la bibliothèque ? Ces espaces vous permettent de discuter confortablement, sans déranger ni être dérangé.

🔧 Comment ça marche ?

- 📅 2 réservations par semaine maximum pour mieux gérer les disponibilités.
- 🕒 24 heures d'attente avant de pouvoir réserver pour la semaine suivante.
- ❌ Annulation facile si vous changez d'avis.
- 📧 Connexion simplifiée : recevez un code par mail, et c'est parti !

Avec les boxes silencieux, appelez en toute sérénité, restez productif et à l'aise, sans quitter la bibliothèque ! 🍷



L'accès à l'application est simplifié grâce à un système de connexion basé sur l'adresse mail universitaire. L'étudiant doit simplement entrer son adresse, qui est ensuite validée par notre backend via la fonction `def index` dans le fichier `views.py`. Un code de validation unique est

généré de manière aléatoire et envoyé automatiquement à l'étudiant. Une fois reçu, il est invité à le saisir pour accéder à l'application.

Connectez-vous pour réserver un box

Le Pixel

Box Silencieux

Université Paris Nanterre

Voici votre code de validation. Vous avez 3 minutes pour l'entrer, passé ce délai, il e

4 1 4 6

Université Paris Nanterre

Université Paris Nanterre
200 avenue de la République
92001 Nanterre Cedex

Pour enrichir la page d'accueil et lui apporter une dimension plus dynamique, nous avons ajouté une section « Thèse du mois ». Cette rubrique propose un aperçu des dernières recherches académiques en libre accès. Afin de préserver les droits d'auteur nous avons choisi de rediriger directement les étudiants souhaitant en connaître plus sur l'article, vers la page de l'article en question sur le site internet « Le Pixel ».

Blog - Thèse du Mois ! 📄

Tous les mois, la BU met en avant une thèse en Open Access - diffusée sous forme numérique et gratuitement -, ainsi que des ressources en Open Access liées à cette thèse.

OPEN ACCESS

Dépasser l'incertitude environnementale

OPEN ACCESS

Ce que j'ai encore raconté à mon chat

OPEN ACCESS

Les revues en ligne favorisent l'échange d'idées scientifiques

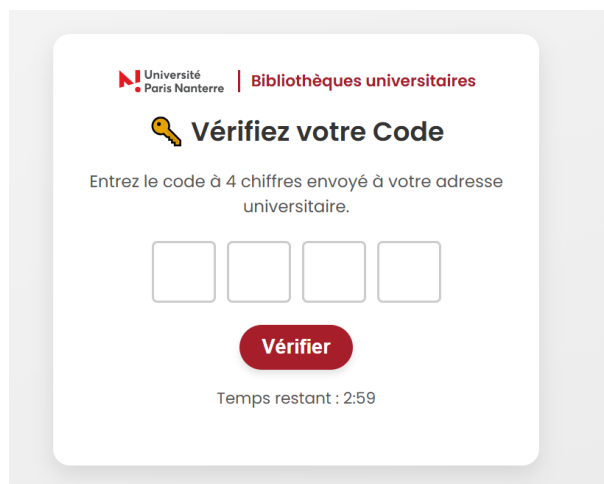
2. Défis rencontrés

La conception et le développement de la page d'accueil ont présenté plusieurs défis, notamment :

- **Hero** : Nous avons voulu rendre l'interface plus attrayante et immersive en utilisant une image de fond représentant la bibliothèque pixel de l'université. Mais également en intégrant le titre de l'application dans l'image tout en superposant une couleur au dessus de l'image pour harmoniser les contrastes visuels et attirer l'attention des utilisateurs.

- **Adresse mail universitaire** : Dans notre script nous avons ajouté une contrainte, on vérifie en temps réel le format de l'adresse saisie, s'assurant que cette adresse respecte le format de l'adresse mail universitaire 12345678@parisnanterre.fr. Cette validation coté frontend permet d'éviter les erreurs avant l'envoi des données au backend.
- **Envoi et gestion des e-mails de confirmation** : L'envoi du code de validation a représenté un défi technique important au cours du projet. En effet, avec le renforcement des mesures de sécurité de Google pour les e-mails automatique, il nous a fallu trouver une solution efficace. Pour répondre à cette contrainte, nous avons mis en place une adresse mail dédiée : boxsilencieusenoreply@gmail.com. L'utilisation d'un mot de passe d'application ainsi que l'activation de l'authentification à deux facteurs nous ont permis d'assurer un envoi sécurisé et fiable des codes aux étudiants.

C. Page Vérification



La page de vérification joue un rôle clé dans le processus d'authentification des étudiants. Elle permet de valider l'identité de l'utilisateur saisissant un code à 4 chiffres reçu par mail après avoir entré son adresse universitaire.

Les principales fonctionnalités de cette page sont la saisie du code de validation, l'étudiant doit entrer un code à 4 chiffres dans des champs prévus à cet effet. Le passage automatique d'un champ à l'autre facilite la saisie sans avoir à utiliser la souris.

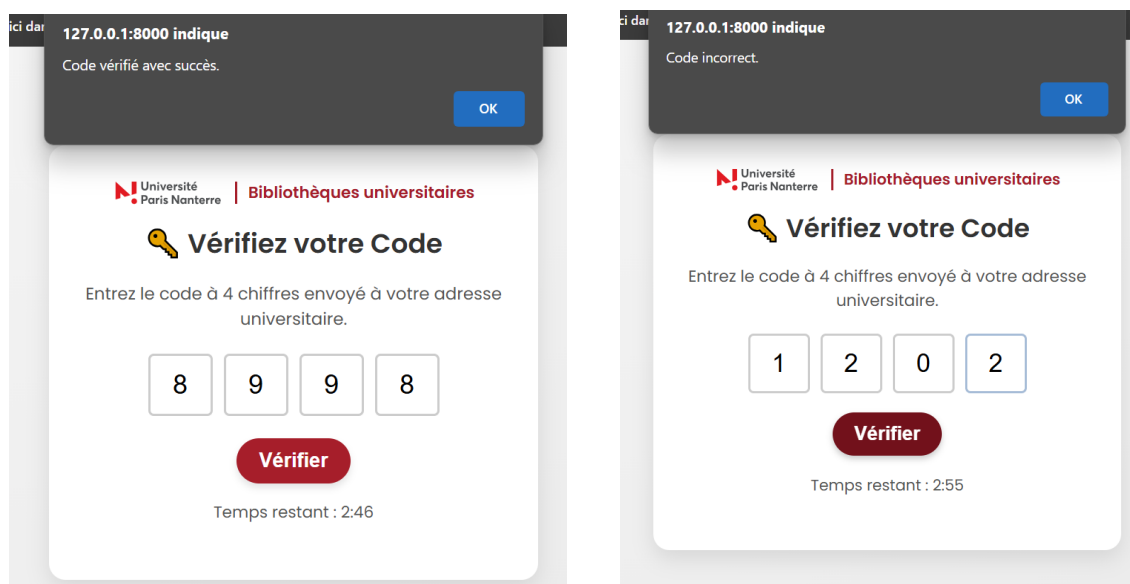
Également, le compte à rebours représenté par un chronomètre de 3 minutes est affiché pour indiquer le temps restant avant l'expiration du code. Une fois ce temps écoulé, le bouton de validation est désactivé, obligeant l'étudiant à demander un nouveau code.

Enfin, la vérification en direct via AJAX du code de validation est faite sans recharger la page, offrant une expérience plus fluide. Si le code est correct, l'étudiant est redirigé vers la page de réservation. En cas d'erreur un message s'affiche immédiatement.

1. Interface utilisateur

La page de vérification a été conçue pour être simple et intuitive, afin que les étudiants puissent l'utiliser sans difficulté. Nous avons tenu à garder l'identité visuelle de l'université en intégrant le logo de l'université. En gardant les couleurs du sites (rouge #ae1f2b, rouge #ed1c24).

L'ergonomie et l'accessibilité ont été des points cruciale, les champs de saisie sont suffisamment espacé pour éviter les erreurs de frappe. Un message d'information guide l'étudiant pour lui expliquer comment utiliser la page. Un bouton de validation est mis en avant pour facilité l'action finale.



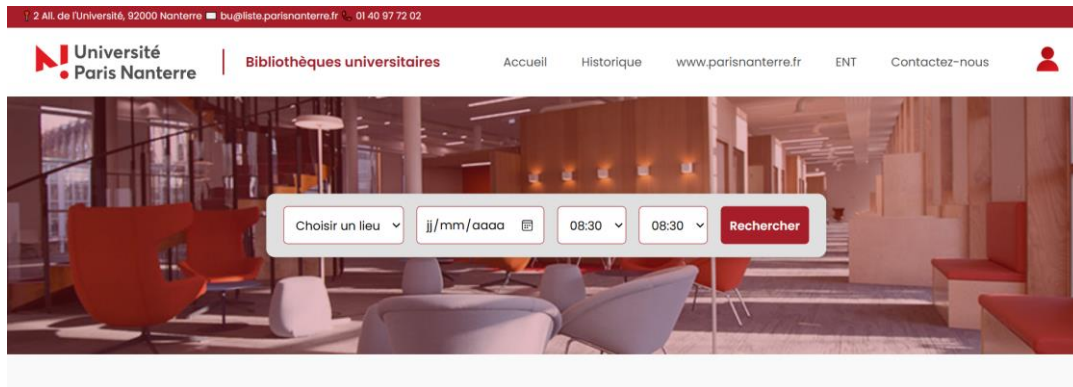
2. Défis rencontrés

Un des principaux défis que nous avons rencontrés concernait la récupération du code de validation généré aléatoirement. Bien que la base de données enregistrerait correctement l'adresse e-mail de l'utilisateur et le code saisi, elle avait du mal à récupérer le code généré, ce qui empêchait de vérifier si le code entré était correct. Ce problème nous a poussés à revoir notre façon de récupérer les données et à tester plusieurs solutions avant de trouver la bonne approche.

Nous avons également rencontré des difficultés avec la saisie du code par l'utilisateur. Chaque chiffre devant être entré dans une case distincte, il a fallu adapter le script JavaScript pour s'assurer que tous les chiffres soient bien regroupés et envoyés correctement au serveur.

Enfin, nous avons initialement envisagé que la page de vérification s'ouvre dans une nouvelle fenêtre afin de mieux séparer les étapes du processus de connexion. Cependant, nous n'avons pas encore trouvé de solution satisfaisante pour intégrer cette fonctionnalité. C'est un point que nous aimerions améliorer dans les prochaines versions de l'application.

D. Page Réservation



1. Fonctionnalités

La page Réservation est l'une des plus importantes de notre application. Elle permet aux étudiants de réserver facilement un box selon leur disponibilités. Nos principales fonctionnalités :

- **La sélection des critères de réservation :** L'étudiant peut choisir la bibliothèque (PIXEL, DROIT), la date et une plage horaire. Une fois tout ces critères entrés, il appuie sur Rechercher et découvre les différents créneaux de 15 minutes disponibles selon ses filtres. Ces créneaux commence dès 8H30 jusqu'à 19h45 et son implémenter dans la table *TimeSlot* ou les différents créneaux sont insérer via un script python *populate_time_slots.py* pour plus de facilité. On a implémenté la table avec tout les créneaux existant de 15 minutes entre 8H30 et 19H45 entre le lundi et vendredi (week-ends et jours fériés non inclus) de l'année 2025. Via un booléen on peut gérer la disponibilité du créneau (1 disponible, 0 indisponible).
- **Affichage des boxes disponibles :** Après avoir rempli les critères l'étudiant observe les différentes salles disponibles. On a utilisé la table *Box* pour énumérer les différents noms des salles.
- **Mail de confirmation :** Pour s'assurer que tout est bien enregistré, un mail de confirmation est envoyé à l'étudiant avec toutes les information de sa réservation. On a implémenté cette fonctionnalité via la fonction *def make_reservation* qui gère la création d'un réservation avec des contraintes données (2 réservations maximum par semaine et un temps d'attente de 24 heures pour les réservation pour les semaines suivantes).

2. Interface utilisateur

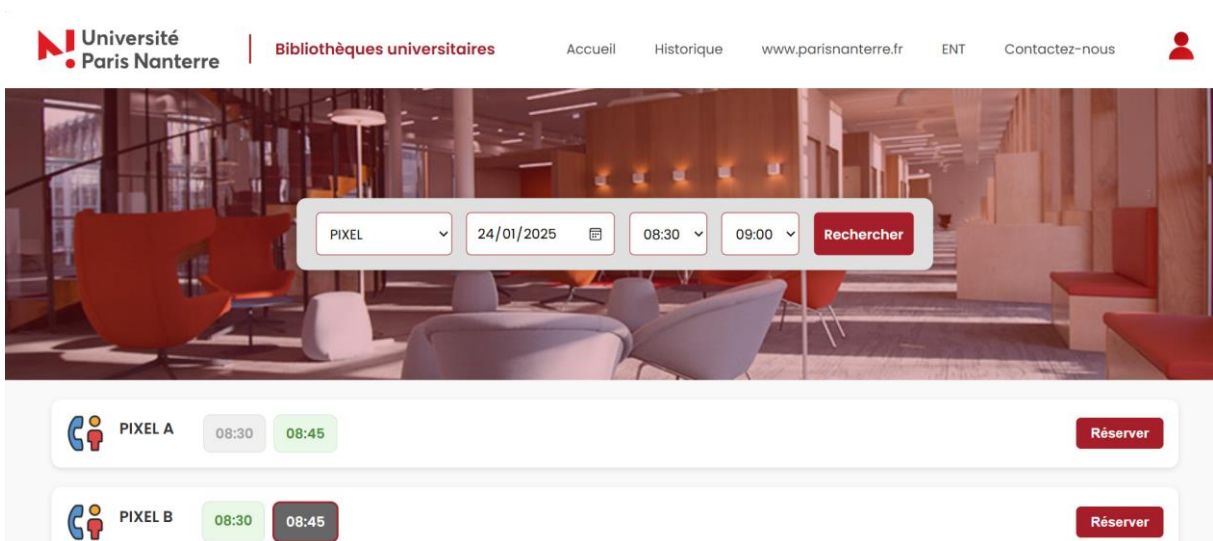
Nous avons voulu que cette page soit la plus intuitive possible. En minimisant l'affichage des éléments de la page avant l'entrer des critères cela permet à l'étudiant de rester maître de son environnement et ne pas se sentir acculé par beaucoup d'informations.

Notre principale inspiration pour la section hero avec la barre de critère incrusté a été le site de réservation de salle de travail de la bibliothèque de l'université Paris Nanterre.



Nous avons trouvé intéressant de garder ce principe de filtrage pour que l'étudiant puisse obtenir le créneau correspondant au maximum à son besoin.

Une fois les critères entrés, les différentes salles disponibles apparaissent.



Découpé par tranche de 15 minutes et avec un code couleur visible (gris clair pour les créneaux indisponible, vert disponible, gris foncé bordure rouge pour les créneaux sélectionnés). Directement relié à la table modélisant les créneaux (TimeSlot). L'affichage des boxes a été implémenté en javascript par praticité. En suggestion d'amélioration il serait peut être plus intéressant de mettre cette partie en python.

L'une de nos inspiration pour nos conteneurs de salles a été le site de réservations de salle de travail de la Bibliothèque nationale de France (BNF) :

Salle de groupe D

6 places

Heure

17h00

17h30

18h00

18h30

19h00

19h30

Durée

30 min

Réserver

Une fois un créneau sélectionné, l'étudiant appuie sur le bouton Réserver et est redirigé vers le bas de la page où un formulaire de confirmation initialement caché apparaît :

CONFIRMATION DE RÉSERVATION


Date : 24/01/2025


Créneau : 08:45

Salle : PIXEL B

Ex: 12345678@parisnanterre.fr

Confirmer

 Université Paris Nanterre
200 avenue de la République
92001 Nanterre Cedex
01 40 97 72 02 (BU)
www.bu.parisnanterre.fr



© 2024 Université Paris Nanterre - Tous droits réservés. Mentions légales

L'étudiant pourra donc entrer son email et confirmer sa réservation. Ce formulaire étant relié à la table Réservation ajoutera la réservation de l'étudiant dans la table et comptabilisera sa réservation pour a terme lui interdire de réserver une fois la limite atteinte (2 réservations par semaine, temps d'attente de 24h pour les réservations des semaines suivantes).

biblioboxes.onrender.com indique

Erreur : Vous avez déjà 2 réservations pour la semaine précédente.
Vous devez attendre 24h avant de réserver pour une nouvelle semaine.

OK

Salle : PIXEL A

40010884@parisnanterre.fr

Confirmer

3. Défis rencontrés

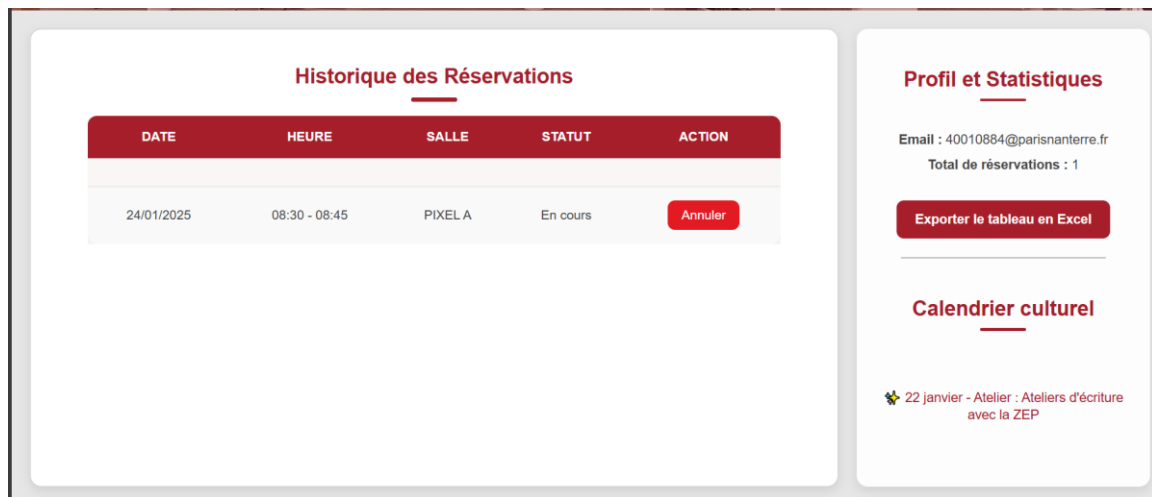
Pendant le développement de la page de réservation, on a rencontré un problème avec l'affichage des créneaux disponibles. Au départ, on avait une seule table Box qui contenait à la fois le nom des salles et leurs disponibilités. Mais en testant l'affichage, on s'est rendu compte que certaines salles, comme "PIXEL A", apparaissaient plusieurs fois. En fait, le script affichait plusieurs fois la même salle parce qu'il y avait plusieurs créneaux pour celle-ci. Cela rendait l'affichage confus et compliquait la gestion des réservations. Pour régler ce souci, on a décidé de revoir la base de données en séparant les informations en deux tables : une table Box qui garde uniquement les noms des salles, et une table TimeSlot pour gérer les créneaux de réservation. Grâce à ce changement, l'affichage est devenu beaucoup plus clair et structuré, ce qui a facilité la réservation et évité les erreurs.

E. Page Historique



1. Fonctionnalités

La page d'historique permet aux étudiants de consulter leurs réservations passées et à venir. Lorsqu'un étudiant accède à cette page, ses réservations sont chargées dynamiquement à partir de la base de données. Chaque réservation est affichée avec la date, l'heure, la salle concernée et son statut (passée ou en cours). Une fonctionnalité d'annulation est également disponible, permettant aux étudiants d'annuler une réservation en un clic, à condition que la date de la réservation ne soit pas encore dépassée. De plus, un bouton d'exportation en format Excel a été ajouté pour permettre aux étudiants de télécharger leur historique.



2. Interface utilisateur

L'interface de la page d'historique a été pensée pour être simple et accessible aux étudiants. Elle affiche un tableau récapitulatif contenant des colonnes pour la date, l'heure, la salle et le statut de chaque réservation. Les réservations passées sont affichées sans possibilité d'action, tandis que celles à venir disposent d'un bouton permettant de les annuler facilement. Sur le côté droit de la page, une section nommée « Profil et Statistiques » ayant comme élément l'email de l'utilisateur comme garanti qu'il est bien sur sa session ainsi que le nombre total de réservations effectuées. Un bouton d'exportation permet aussi de télécharger l'historique des réservations au format Excel.

3. Défis rencontrés

Un des problèmes majeurs que nous avons rencontrés était de réussir à afficher correctement les réservations depuis la base de données. Au début, le tableau ne récupérait pas les bonnes informations dû à la différences des noms des attributs entre la table Reservation et le tableau dans la page Historique. Après plusieurs essais, nous avons réussi à trouver une solution pouvant relié ces deux éléments et offrir une vue d'ensemble des réservations de l'étudiant.

Un autre défi important a été la gestion de l'annulation des réservations. Nous avons utilisé la fonction Python `cancel_reservation_api` pour gérer cette action. Cette fonction vérifie si la réservation appartient bien à l'étudiant connecté et si elle peut encore être annulée. Ensuite, elle remet le créneau correspondant en disponibilité pour qu'un autre étudiant puisse le réserver. Ce processus devait être sécurisé pour éviter les erreurs si plusieurs utilisateurs effectuaient des actions en même temps.

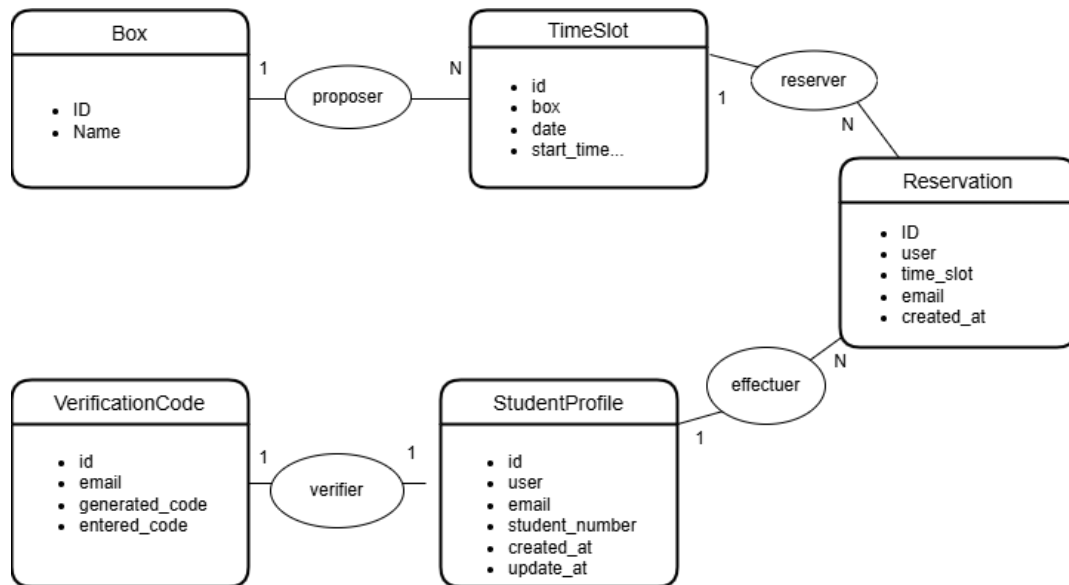


V. Conclusion : Améliorations et perspectives futures

En conclusion, ce projet de réservation de boxes silencieux a été une expérience enrichissante qui nous a permis d'appliquer nos connaissances en développement web tout en répondant à un besoin concret des étudiants. Nous avons rencontré plusieurs défis techniques, notamment la gestion des créneaux, l'annulation des réservations et l'interaction avec la base de données, mais chaque difficulté nous a permis de mieux comprendre les différentes étapes du développement d'une application web. Grâce à nos efforts, nous avons réussi à proposer une solution fonctionnelle et intuitive qui facilite la gestion des réservations tout en garantissant une bonne expérience utilisateur.

Cependant, il reste encore quelques améliorations à apporter. Par exemple, nous devons résoudre le problème de récupération du numéro étudiant afin qu'il s'affiche correctement dans l'onglet utilisateur du menu. De plus, pour améliorer l'expérience utilisateur, il serait préférable que la page de vérification s'ouvre dans une petite fenêtre au lieu d'un nouvel onglet, ce qui rendrait le processus plus fluide. Concernant la page de réservation, une autre amélioration consisterait à récupérer automatiquement l'adresse e-mail de l'étudiant dans le formulaire de confirmation, évitant ainsi une saisie manuelle et réduisant les risques d'erreurs.

Enfin, pour les perspectives futures, une attention particulière devra être portée à l'amélioration du design, en particulier l'affichage sur mobile qui pourrait être optimisé pour offrir une meilleure expérience utilisateur sur les petits écrans. Ces améliorations permettraient de rendre l'application encore plus pratique et accessible à tous les étudiants, quel que soit l'appareil utilisé.



Pour aller plus loin ...

1. Déploiement de l'application sur Render

Pour rendre notre projet accessible en ligne, nous avons choisi de le déployer sur la plateforme Render, qui offre un hébergement cloud simple et gratuit pour les applications django. On a principalement fait ce choix par prévention en cas de problème technique lors de la soutenance.

a. Mis en place de déploiement

Le déploiement sur Render a nécessité plusieurs étapes d'adaptation dans notre projet. Nous avons commencé par modifier certains paramètres dans le fichiers settings.py afin de rendre l'application compatible avec l'environnement en ligne.

Dans un premier temps nous avons utilisé la bibliothèque django-environ pour gérer les variables sensibles comme la clé secrète de l'application, les informations de la base de données et les identifiants d'envoi de mails. Ces informations sont stockées dans un fichier .env, ce qui permet de séparer les configurations locales et de production en toutes sécurité.

Concernant la gestion des fichiers statiques (frontend) on a ajouté Whitenoise, qui permet de les servir efficacement et d'améliorer le temps de chargement des pages. Nous avons également ajouté une base de données PostgreSQL fournie par Render, remplaçant la base locale SQLite. On connecte cette base via la variable d'environnement DATABASE_URL.

Le fichier render.yaml a été créé pour automatiser le processus de déploiement. Il contient différentes étapes comme l'installation des dépendances (pip install -r requirements.txt), la collecte des fichiers statiques pour les préparer à être servis en ligne (python manage.py collectstatic --noinput), l'application des migrations pour préparer la base de donnée (

python manage.py migrate) et enfin le lancement du serveur Gunicorn pour exécuter l'application (gunicorn reservation_system.wsgi:application).

Ces étapes permettent d'assurer que l'application est prête à être utilisée dès que le déploiement est terminé.

Enfin, vous trouverez ci-dessous le lien pour accéder au github contenant l'ensemble de notre projet.

Lien : <https://github.com/KawtherAslimi/bibliobox/invitations>