Projet Développement Web

Lien GIT : https://github.com/souleim95/HubUniversity

Nom du projet : HubUniversité

Enseignante : Mme Guidini Goncalves

Etablissement: Cy Tech

Date de rendu : 04/05/2025

Paul PITIOT - Louaye SAGHIR - Florian DELSUC - Souleim GHOUDI

Introduction

Dans le cadre du module de développement web de première année à CY Tech, notre équipe a travaillé sur la conception d'un site d'université intelligente respectant les enjeux scolaires autour des étudiants, enseignants et du personnels de HubUniversité. Notre projet, intitulé HubUniversité, s'inscrit dans le thème d'une établissement intelligent, intégrant des objets connectés, une interface utilisateur responsive, et une architecture modulaire. Cette plateforme vise à centraliser différents services liés à la gestion, l'administration et la visualisation des données de capteurs ou objets intelligents.

Conformément au cahier des charges, le site se décompose en plusieurs modules fonctionnels : Information, Visualisation, Gestion, et Administration, chacun étant accessible selon le type d'utilisateur (visiteur, étudiant, enseignant, directeur). Les technologies utilisées incluent React.js pour le frontend, Node.js/Express pour le backend, PostgreSQL pour la base de données, et Docker pour l'orchestration des services. Le dépôt GitHub a été utilisé pour le versionnement et la collaboration continue.

Arborescence du projet

```
- HubUniversité/
  ├── backend/ – API, routes, serveur Node
  ├── database/ – Scripts PostgreSQL
  ├── docs/ – Documentation projet
  ├── frontend/
  |   ├── public/
  |   └── src/
  |       ├── assets/
  |       ├── components/
  |       ├── context/
  |       ├── data/
  |       ├── hooks/
  |       ├── styles/
  |       └── utils/
  ├── .gitignore
  ├── docker-compose.yml
  ├── package.json
  └── package-lock.json
```

Pour optimiser la maintenabilité et réduire la complexité des composants, toute la logique métier (useState, useEffect, constantes, etc.) a été extraite dans des hooks personnalisés situés dans le dossier `/hooks`. On y trouve notamment :

- `useAdmin.js` - Gestion des fonctionnalités administrateur
- `useCampusMap.js` - Logique de la carte du campus
- `useClickOutside.js` - Détection des clics hors composant
- `useDashboard.js & useDashboard_renderControls.js` - États et effets du tableau de bord
- `useGestion.js` - Logique de gestion des ressources
- `useHeader.js` - Utilitaires pour composants sans UI
- `useSearch.js` - Fonctionnalités de recherche

Cette architecture modulaire permet d'éviter les composants surchargés et facilite la réutilisation du code à travers l'application.

Répartition des tâches au sein de l'équipe

Dans le cadre du développement de la plateforme **HubUniversité**, chaque membre du groupe s'est vu attribuer des responsabilités spécifiques, réparties entre le frontend et le backend de l'application. Voici le détail de cette répartition :

Backend

Paul PITIOT – Infrastructure, API externe, orchestration Docker

Paul s'est chargé de la mise en place de l'environnement backend complet, permettant à tout le projet de fonctionner automatiquement via Docker. Ses contributions majeures incluent :

- **Configuration Docker (fichier docker-compose.yml) :**
  - Orchestration des conteneurs Node.js, PostgreSQL et React ;
  - Construction d'un environnement prêt à l'emploi : la commande docker-compose up --build -d permet de lancer l'ensemble du projet ;
- **Initialisation de la base de données (init.sql) :**
  - Création de la structure de base, avec scripts d'insertion de données ;

- Préparation des tables essentielles pour les utilisateurs et les objets ;

- **Serveur Node.js (server.js) :**
  - Mise en place du serveur Express.js ;
  - Déploiement d'un point d'entrée unique pour les routes backend ;
- **Intégration de l'API SNCF (RER A - Cergy) :**
  - Récupération et traitement en temps réel des données de transport pour enrichir les services proposés aux utilisateurs.

Grâce à cette architecture, le projet peut être lancé sans configuration manuelle supplémentaire.

Florian DELSUC – Sécurité, gestion des utilisateurs, logique métier

Florian a travaillé sur la gestion des utilisateurs, les droits d'accès, et la sécurisation des échanges avec le backend. Ses réalisations comprennent :

- **Intégration de bcrypt pour le hachage des mots de passe :**
  - Sécurisation des connexions et inscriptions utilisateurs ;
  - Prévention des accès non autorisés et stockage sécurisé dans la base de données ;
- **Création de la table user dans init.sql :**
  - Définition des champs nécessaires à la gestion des profils (identité, niveau, rôle, mot de passe) ;
  - Intégration avec la logique de connexion via l'auth backend ;
- **Définition des types d'utilisateurs :**
  - Implémentation des rôles (visiteur, simple, complexe, administrateur) côté backend ;
  - Coordination avec le frontend pour garantir des permissions adaptées selon le type de compte.

Florian a aussi contribué à la structure des **routes Express** en vue de connecter les modules frontend aux ressources backend (utilisateurs, objets, données).

Frontend

Souleim GHOUDI – Architecture des pages et responsivité

Souleim s'est concentré sur la construction fonctionnelle des pages clés du projet et leur adaptation responsive. Ses contributions incluent :

- **Responsive Design :** adaptation de toutes les pages aux formats mobile, tablette et desktop, selon le principe *Mobile First* ;
- **Page Profil utilisateur** :
    - Modification et affichage des informations personnelles (nom, rôle, avatar, etc.) ;
    - Accès aux données privées selon le type d'utilisateur ;
- **Page Gestion (module Gestion)** :
    - Interface permettant aux utilisateurs expérimentés de modifier les paramètres des objets connectés (température, statut, consommation, etc.) ;
    - Visualisation de statistiques et historique des interactions ;
    - Configuration/création/modification/demande de suppression des objets
- **Page Connexion/ Inscription et Campus (module Visualisation)** :

  - Interface de connexion sécurisée avec vérification de l'identité (type "membre du campus" via la BDD) ;
  - Accès aux outils liés à l'université pour les utilisateurs authentifiés.
  - Campus connecté via cartes interactives (« Vue Liste »)
  - Formulaire d'inscription et de connexion d'un utilisateur.
- **Intégration de l'API Météo (Cergy)** :
  - Récupération et traitement en temps réel des données météorologiques pour Cergy, permettant d'afficher la météo en direct pour enrichir l'expérience utilisateur.

Souleim a également participé à la structuration du code dans les dossiers components/ et context/, en créant des composants réutilisables et les contextes utilisateur et l'esthétique du site en général.

Louaye SAGHIR – Design, pages d'accueil et administration

Louaye a pris en charge l'ensemble de la conception graphique du site ainsi que le développement de plusieurs modules vitaux :

- **Charte graphique du projet :**
  - Choix des couleurs, typographies, marges, espacement ;
  - Conception des composants UI (boutons, menus, barres de navigation, cartes) ;
- **Page Visiteur (module Information) :**
  - Affichage des actualités, filtres de recherche, contenu informatif librement accessible ;
  - Accès sans authentification, avec "free tour" de la plateforme ;
- **Campus (module Visualisation) :**
  - Campus connecté via plan interactive (« Vue Carte »)
- **Dashboard (module Visualisation) :**
  - Page d'accueil personnalisée pour chaque utilisateur connecté ;
  - Affichage de ses objets connectés, services, points d'expérience ;

- **Page Administrateur (module Administration) :**
    - Gestion des utilisateurs : création, suppression, attribution de niveaux ;
    - Statistiques globales (fréquence de connexion, consommation des objets) ;
    - Configuration des objets, catégories, alertes de maintenance.

Louaye a aussi travaillé sur la **navigation générale**, l'**esthétique cohérente** entre les modules, et a optimisé le CSS dans styles/.

Module Information

Découverte libre de la plateforme

Le module Information peut-être interpréter comme la porte d'entrée de la plateforme de notre université HubUniversité pour les utilisateurs non authentifiés, appelés « visiteurs ». Il s'agit d'un accès libre offrant un aperçu de la plateforme et des services accessibles sans avoir besoin de créer un compte. Cette fonctionnalité joue un rôle clé pour attirer de nouveaux étudiants en leur présentant les formations que notre université propose**.**

Objectifs universitaires fonctionnels

- Offrir une première expérience utilisateur fluide et responsive.
- Montrer les actualités universitaires, les formations universitaires, et quelques données ouvertes (Horaire RER A à Cergy-Préfecture, Donnée Météo Cergy).
- Susciter l'intérêt pour inciter à l'inscription.

Fonctionnalités principales

- Accès à une page d'accueil publique contenant les actualités récentes (ex : événements, annonces du campus).
- Navigation intuitive dans les différentes sections accessibles :

  - Infos pratiques de l'établissement,
  - Services disponibles (wifi, bibliothèque, transports, etc.),
  - Localisation des bâtiments et départements.
- Interface adaptée aux différents écrans (responsive design).
- Affichage conditionnel des fonctions : les zones réservées (connexion, objets connectés, etc.) sont grisées ou inactives.

Implémentation technique

- **Frontend :**
  - Composant PageAccueil.jsx (anciennement PublicPage.jsx) (dossier frontend/src/components/), conçu avec React.
  - Données préchargées ou issues d'API publiques (transport, événements).
  - CSS adapté pour l'expérience mobile-first (frontend/src/styles).

- **Backend :**
  - Route REST publique exposée via Express dans backend/routes/public.js.
  - Données extraites de la base via des requêtes SQL (PostgreSQL).

Exemple d'utilisation

Un visiteur accède à la page d'accueil, voit les prochains événements universitaires (conférences, salons, portes ouvertes), consulte les horaires du RER A via l'API SNCF, puis décide de créer un compte

Recherche d'informations locales

Par ailleurs, le module Information intègre un système de recherche qui permet aux visiteurs de consulter des informations utiles sur la vie universitaire. Ce moteur de recherche permet d'effectuer des recherches multicritères grâce à des filtres pertinents.

Objectifs universitaires fonctionnels

- Permettre aux visiteurs de trouver rapidement une information utile sans être connecté.
- Favoriser l'engagement et la curiosité.

Fonctionnalités principales

- **Filtres multiples disponibles** :

    - Type d'information (Formation, FAQ, RER A, Météo).
- Affichage des résultats sous forme de redirections vers la réponse la plus cohérente associé.
- Système de pagination ou scroll infini pour l'exploration.

Implémentation technique

- **Frontend** :
    - Composants React : SearchBox.
    - Utilisation de context pour passer les critères de recherche entre composants.
- **Backend** :
    - Route GET /public/search filtrant selon les paramètres passés en URL.
    - Requêtes SQL avec WHERE dynamiques sur les tables events, news, services.

Exemple d'utilisation

« Exemple de visiteur qui va sur la page des formations »

Invitation à l'inscription

Le module Information propose aussi une transition fluide vers l'inscription pour les visiteurs souhaitant accéder à plus de services. Cette fonctionnalité permet de transformer un simple visiteur en utilisateur authentifié.

Objectifs universitaires fonctionnels

- Rendre visible et attractive l'inscription depuis toutes les pages publiques.
- Guider l'utilisateur dans le processus d'enregistrement.

Fonctionnalités principales

- Boutons d'appel à l'action : "Connexion".
- Redirection vers le formulaire d'inscription avec vérification du rôle.
- Une fois l'inscription faites, l'étudiant ou l'enseignant a accès aux objets connectés, au tableau de bord, etc.

Implémentation technique

- **Frontend** :
    - Profile.jsx, HeroSection.jsx, Formation.jsx, CampusMap.jsx, RerSchedule.jsx.

    - Popup ou section fixe en bas de page avec bouton d'inscription.

- **Backend** :
    - Validation du rôle via la base PostgreSQL dans la table users.
    - Envoi de mail de confirmation via nodemailer configuré dans auth.controller.js.

Exemple d'utilisation

Après avoir consulté les formations proposées, le campus connecté, les infos sur les transports, la FAQ, le visiteur clique sur "Connexion" (ou s'il tente d'accéder à l'objet, il sera directement rediriger sur la page "Connexion"), remplit le formulaire et accède désormais au module Visualisation.

Module Visualisation

Inscription sécurisée et gestion du compte

Le module Visualisation est le premier point d'entrée pour les utilisateurs authentifiés (type "simple"). Il permet l'inscription sécurisée, la connexion via mot de passe et la gestion des informations personnelles. Ce module conditionne l'accès aux objets connectés et services universitaires internes.

Objectifs universitaires fonctionnels

- Garantir un accès facile aux services internes.
- Gérer les informations personnelles de façon claire et modulable.
- Créer un système d'authentification robuste.

Fonctionnalités principales

- Formulaire d'inscription avec vérification du rôle (enseignant, étudiant...).
- Connexion par identifiants + gestion de mot de passe (modification sécurisée).
- Formulaire de profil avec deux sections :
    - **Partie publique** : pseudo, genre, rôle, avatar, type de membre.
    - **Partie privée** : nom, prénom, mot de passe.

Implémentation technique

- **Frontend** :
    - Profile.jsx, ProtectedRoute.jsx dans frontend/src/components/ ; App.jsx (<ProtectedRoute/>)
    - Utilisation du UserContext pour gérer la session utilisateur.
- **Backend** :
    - Authentification via bcrypt et JWT (routes auth.controller.js).
    - Vérification du rôle dans la base users (colonne role).
    - Stockage sécurisé dans PostgreSQL, gestion des tokens JWT.

Exemple d'utilisation

Un étudiant s'inscrit, se connecte, puis accède à son espace personnel en cliquant sur la première lettre de son prénom en haut à droite de la page pour consulter son profil ou modifier son mot de passe.

Profil utilisateur et niveaux d'expérience

Description générale

Chaque utilisateur possède un profil associé à un niveau d'expérience (débutant, intermédiaire, avancé, expert). Cette stratégie gamifiée encourage l'utilisation de la plateforme et permet le déblocage progressif des modules.

Objectifs universitaires fonctionnels

- Valoriser l'activité sur la plateforme par des points.
- Offrir des fonctionnalités évolutives selon le niveau atteint.

Fonctionnalités principales

- Barre de progression avec points gagnés par actions :
    - 0.25 points par connexion,
    - 0.50 points par consultation d'un objet/service.
- Possibilité de changement de niveau selon le cumul de points.
- Interface de visualisation du niveau actuel, de la progression et des avantages.

Implémentation technique

- **Frontend** :
    - useDashboard.js (hooks/) appeler dans Dashboard.jsx.
    - Utilisation de graphiques ou jauges de progression (librairie comme Chart.js).
- **Backend** :
    - Points mis à jour dans PostgreSQL via middleware lors des actions utilisateurs.
    - Table user_points liée à user_id.

Exemple d'utilisation

Un enseignant se connecte régulièrement, consulte les objets connectés, atteint le niveau "avancé" et débloque l'accès au module "Gestion".

Accès aux objets et services connectés

Le module permet dans un autre temps de consulter les objets intelligents du campus et les services internes. L'accès est restreint aux utilisateurs authentifiés.

Objectifs universitaires fonctionnels

- Offrir un accès ciblé aux objets/systèmes connectés liés au rôle utilisateur.
- Permettre la consultation, sans modification, des données de fonctionnement.

Fonctionnalités principales

- Recherche d'objets avec **au moins 2 filtres** (ex : type, salle, statut, disponible).
- Consultation des données temps réel :
    - Température, état, connectivité, projecteur, etc.
- Liste des services accessibles : consommations, historiques, horaires de salle...

Implémentation technique

- **Frontend** :
    - Composants Header.jsx (ligne 295), SearchBox.jsx.
    - Affichage dynamique via React Hooks et Context.
- **Backend** :

- Requêtes SQL avancées avec jointures dans objects.controller.js.
- Filtres dynamiques à travers les routes Express : /api/objects?type=...&status=...

Exemple d'utilisation

Un utilisateur utilise la barre de recherche pour rechercher un objet en particulier rapidement.

Module Gestion

Administration des objets connectés
Configuration intelligente des services
Suivi de l'efficacité et rapports

Module Administration

Supervision des utilisateurs
Gestion globale des objets et services
Sécurité, maintenance et personnalisation

Conclusion et perspectives

Conclusion générale

Le projet HubUniversité a permis à notre équipe de développer un site web complexe intégrant des fonctionnalités avancées (objets connectés, gestion de rôles, visualisation de données).
Nous avons su collaborer efficacement en appliquant une méthodologie rigoureuse, et en utilisant des technologies modernes adaptées aux enjeux du développement web d'aujourd'hui.

Nous avons atteint les objectifs fixés, en livrant une solution opérationnelle, sécurisée et ergonomique, démontrant ainsi la capacité de notre groupe à concevoir et réaliser un projet web d'envergure dans un temps imparti.

Perspectives d'amélioration

Afin d'améliorer encore la plateforme, plusieurs pistes d'évolution sont envisageables :

- **Ajout de notifications en temps réel** (WebSockets) pour les alertes sur les objets connectés.
- **Optimisation du backend**.
- **Amélioration de l'interface utilisateur** avec des animations (ex : transitions page, loaders dynamiques).
- **Extension du système de points d'expérience** pour inclure des badges, challenges et classements.

Nous recommandons également de continuer à enrichir la base de données d'objets connectés pour refléter les évolutions du campus.