
# PC3R : Rapport de projet

Ewen GLASZIOU (21312544) Julien LE GOFF (21304302)

26 mai 2024

# Table des matières

# 1 Description du Projet

Dans le contexte du cours de PC3R il nous à été demandé de réaliser une application web en GO ou Servlet Java pour la partie backend, et html/javascript pour le frontend, ce projet doit interagir avec au moins une API.

Nous avons choisi de créer une application nommée LearnHub. Cette application a pour objectif de faciliter la recherche de vidéos YouTube et d'articles pertinents pour l'apprentissage de sujets scientifiques dans divers domaines. Sa particularité réside dans le fait que les vidéos sont postées par des utilisateurs inscrits, qui doivent indiquer leur niveau d'étude. Cela permet de s'assurer que les vidéos recommandées proviennent de personnes ayant les compétences requises. L'application est d'ailleurs déployé à l'adresse suivante : pc3r-projet.onrender.com. Cela à été possible grâce au site Render avec un forfait gratuit nous permettant de le déployer avec des capacité réduite.

— Inscription et connexion des utilisateurs.

— Gestion des profils utilisateurs avec des informations personnelles et des photos de profil.

— Partage de documents éducatifs (vidéos, articles) par les utilisateurs.

— Commentaires et évaluations sur les documents partagés.

— Utilisation d'une API externe pour enrichir le contenu de la plateforme.

## 1.1 API Web choisie

Nous avons donc choisi d'utiliser l'API de YouTube pour accéder aux informations des vidéos via les liens fournis par les utilisateurs et générer des recommandations. Grâce à cette API, nous pouvons récupérer des données détaillées sur chaque vidéo, telles que le titre, l'auteur, la date de publication et les miniatures. Cela nous permet non seulement d'afficher des informations précises et à jour sur les vidéos partagées, mais aussi de proposer des recommandations basées sur les thèmes et les préférences des utilisateurs. En utilisant ces données, nous assurons que les vidéos suggérées sont pertinentes et proviennent de sources fiables, alignées avec les niveaux d'étude indiqués par les utilisateurs inscrits sur LearnHub.

## 1.2 Fonctionnalités de l'application

Voici les fonctionnalités que nous avons prévu d'implémenter et leur état actuelle :

— Stockage des ressources ; vidéos ou articles. (fonctionnel)

— Afficher des ressources. (fonctionnel)

— Connexion pour les utilisateurs. (fonctionnel)

— Déconnexion des utilisateurs. (fonctionnel)

— Supprimer un utilisateur inscrit. (fonctionnel)

— Un utilisateur peut rechercher et voir des ressources. (fonctionnel)

— Un utilisateur inscrit peut poster des ressources. (fonctionnel)

— L'application peut faire de la recommandation de ressources en cherchant sur les API de Youtube et DBLP des résultats similaires. (fonctionnel mais pas parfait)

— Un utilisateur peu modifier son profil. (en cours)

— Un utilisateur inscrit peut supprimer un contenue (vidéo ou article) qu'il à poster sur son profile. (en cours)

Ces fonctionnalités peuvent d'être expliqué de manière textuelle de la façon suivante :

— Alice se rend sur le site LearnHub. Elle écrit dans la barre de recherche "Algorithmique avancée". Elle consulte ensuite les vidéos et articles associés au thème qu'elle a recherché, proposés par les utilisateurs de LearnHub.

— Jacques se rend sur le site LearnHub. Il décide de s'inscrire, il entre ses informations personnelles telles que son adresse email et un mot de passe. Il termine son inscription. Il se déconnecte.

— Jacques se rend sur le site LearnHub. Il se connecte. Il décide d'ajouter une vidéo sur la programmation concurrente. Jacques poste la vidéo. Il se déconnecte.

— Alice se rend sur le site LearnHub. Elle se connecte. Elle ajoute un article sur les mathématiques théoriques.

— Alice se rend sur le site LearnHub. Elle se connecte. Elle modifie son profil.

— Jacques se rend sur le site LearnHub. Il se connecte. Il se rend sur son profil. Il supprime son compte.

— Alice se rend sur le site LearnHub. Elle se connecte. Elle consulte les recommandation proposer par le site. Elle se rend sur son profile, supprime la dernière vidéo qu'elle a proposé. Elle change son niveau d'étude. Elle se déconnecte.

## 1.3 Choix de la base de données

Pour le choix de la base de données, nous avons envisagé plusieurs possibilités. Dans un premier temps, PostgreSQL, qui est la base de données disponible sur le site Render.com, nous a été conseillé par M. Demangeon pour le déploiement de l'application. Le problème de ce choix a été notre inexpérience avec cette base de données, ainsi que l'impossibilité de se connecter efficacement avec les identifiants fournis. Nous nous sommes donc tournés vers MySQL, que nous avions tous deux déjà utilisé par le passé. Une première version de la base a été réalisée sur WAMP avant de réaliser un import et de passer sur Aiven.io, un site permettant d'héberger gratuitement une base de données MySQL avec une limite de stockage acceptable de 5 Go et une RAM de 1 Go.

## 1.4 La base de données

Pour la création de la base nous avons réalisé un total de 6 tables :

1. document : Les différentes informations sur les ressources.
2. domaine : Les différents domaines d'études.
3. niveau_etude : Le niveau d'étude des utilisateurs.
4. theme : Le theme des différents documents, appartenent à des domaines.
5. type_document : Le type du doument : vidéo ou article.
6. utilisateur : Nos utilisateurs.

Voici un tableau les résumant avec une table par ligne :

| Table | Colonne | Type | Clé | AI | Description |
|---|---|---|---|---|---|
| document | lien | varchar(255) | PK | | Lien de la ressource |
| | titre | varchar(255) | | | Titre de la ressource |
| | auteur | varchar(255) | | | Auteur de la ressource |
| | id_postant | int | | | ID de l'utilisateur postant |
| | id_theme | int | | | ID du thème associé |
| | id_type_document | int | | | ID du type de document |
| | date | date | | | Date de publication |
| domaine | id | int | PK | AI | ID du domaine |
| | nom | varchar(255) | | | Nom du domaine |
| niveau_etude | id | int | PK | AI | ID du niveau d'étude |
| | intitule | varchar(255) | | | Intitulé du niveau d'étude |
| theme | id | int | PK | AI | ID du thème |
| | nom | varchar(255) | | | Nom du thème |
| | id_domaine | int | | | ID du domaine associé |
| type_document | id | int | PK | AI | ID du type de document |
| | nom | varchar(255) | | | Nom du type de document |
| utilisateur | id | int | PK | AI | ID de l'utilisateur |
| | mail | varchar(255) | | | Adresse email de l'utilisateur |
| | nom | varchar(255) | | | Nom de l'utilisateur |
| | prenom | varchar(255) | | | Prénom de l'utilisateur |
| | mot_de_passe | varchar(255) | | | Mot de passe haché |
| | date_naissance | date | | | Date de naissance de l'utilisateur |
| | id_niveau_etude | int | | | ID du niveau d'étude |
| | lien_linkedin | varchar(255) | | | Lien LinkedIn de l'utilisateur |
| | justificatif | longtext | | | Justificatif du niveau d'étude |

TABLE 1 – Description des tables de la base de données LearnHub

## 1.5 Mise à jour des données et appels à l'API externe

Les données de LearnHub sont mises à jour de manière régulière grâce à des appels à l'API externe de YouTube. Ces appels sont déclenchés principalement dans deux situations :

1. **Ajout de contenu par les utilisateurs** : Lorsqu'un utilisateur inscrit ajoute une nouvelle vidéo sur LearnHub, un appel est fait à l'API YouTube pour vérifier et récupérer les informations détaillées sur cette vidéo. Cela garantit que les informations affichées sont précises et à jour.
2. **Mises à jour périodiques** : Le serveur de LearnHub effectue également des mises à jour périodiques pour s'assurer que les données des vidéos sont à jour. Par exemple, chaque nuit, un processus de mise à jour vérifie les vidéos les plus consultées et actualise leurs informations à partir de l'API YouTube. Cela permet de maintenir la pertinence et l'exactitude des données présentes sur la plateforme.
3. **Recommandation pour les utilisateurs** : Lorsque les utilisateurs effectuent une demande de recommandations de vidéos sur un sujet spécifique, une requête est envoyée à l'API YouTube pour obtenir les vidéos les plus récentes et pertinentes. Les informations récupérées incluent le titre, l'auteur, la date de publication, le lien de la vidéo et une miniature. Ces données sont ensuite affichées sur le site.

Les appels à l'API YouTube sont réalisés de manière asynchrone pour ne pas bloquer l'interaction utilisateur et améliorer les performances de l'application. Les données récupérées sont

encodées en JSON et stockées dans la base de données MySQL, garantissant une gestion efficace et sécurisée des informations.

En résumé, LearnHub utilise l'API YouTube pour enrichir son contenu avec des vidéos éducatives pertinentes et à jour, et pour s'assurer que les informations affichées sur la plateforme sont toujours correctes et actualisées.

# 2 Description du Serveur

## 2.1 Choix de l'approche

Le serveur est construit en Go en utilisant le package `net/http`. L'approche REST est adoptée pour structurer les endpoints du serveur, facilitant ainsi les communications entre le client et le serveur via des requêtes HTTP standardisées.

## 2.2 Composants du Serveur

Le serveur est constitué de plusieurs composants, chacun ayant des responsabilités spécifiques :

### 2.2.1 Inscription et Connexion

- **GET /connexion** : Affiche le formulaire de connexion.
- **POST /connexion** : Traite les données de connexion, authentifie l'utilisateur et crée une session.
- **POST /inscription** : Traite les données d'inscription, crée un nouvel utilisateur et une session.

### 2.2.2 Déconnexion

- **POST /deconnexion** : Supprime la session de l'utilisateur et le cookie de session.

### 2.2.3 Gestion des Profils

- **GET /profil** : Récupère et affiche les informations du profil de l'utilisateur.
- **POST /profil** : Met à jour les informations du profil de l'utilisateur.

### 2.2.4 Gestion des Documents

- **GET /documents** : Récupère et affiche les documents partagés par les utilisateurs.
- **POST /documents** : Ajoute un nouveau document à la base de données.
- **PUT /documents** : Met à jour les informations d'un document existant.

### 2.2.5 Commentaires et Évaluations (à faire)

- **POST /commentaires** : Ajoute un commentaire ou une évaluation à un document.
- **GET /commentaires** : Récupère les commentaires et les évaluations d'un document.

## 2.3 Exemples de Code de Gestion des Sessions

```
func generateSessionID() (string, error) {
    b := make([]byte, 32)
    if _, err := rand.Read(b); err != nil {
        return "", err
    }
    return base64.URLEncoding.EncodeToString(b), nil
}

func loginHandler(w http.ResponseWriter, r *http.Request, user User) {
    sessionID, err := generateSessionID()
    if err != nil {
        http.Error(w, "Erreur interne du serveur", http.StatusInternalServerError)
        return
    }

    // Créer une nouvelle session et la stocker
    session := Session{ID: sessionID, User: user}
    sessionStore.Lock()
    sessionStore.sessions[sessionID] = session
    sessionStore.Unlock()

    // Créer des cookies sécurisés pour la session
    http.SetCookie(w, &http.Cookie{
        Name:     "session_id",
        Value:    sessionID,
        HttpOnly: true,
        Secure:   false, // À mettre à true en passant en HTTPS
        Path:     "/",
        MaxAge:   3600, // 1 heure
    })
    http.Redirect(w, r, "/", http.StatusFound)
}

func logoutHandler(w http.ResponseWriter, r *http.Request) {
    sessionCookie, err := r.Cookie("session_id")
    if err != nil {
        if err == http.ErrNoCookie {
            http.Redirect(w, r, "/login", http.StatusFound)
            return
        }
        http.Error(w, "Erreur interne du serveur", http.StatusInternalServerError)
        return
    }

    // Supprimer la session du magasin de sessions
    sessionStore.Lock()
    delete(sessionStore.sessions, sessionCookie.Value)
    sessionStore.Unlock()
```

```
    // Supprimer le cookie de session
    http.SetCookie(w, &http.Cookie{
        Name:     "session_id",
        Value:    "",
        HttpOnly: true,
        Secure:   false, // À mettre à true en passant en HTTPS
        Path:     "/",
        MaxAge:   -1, // Supprimer le cookie
    })

    http.Redirect(w, r, "/login", http.StatusFound)
}
```

Ces fonctions montrent comment générer des IDs de session sécurisés, créer et gérer les sessions utilisateur, ainsi que gérer la déconnexion des utilisateurs.

# 3 Description du Client

## 3.1 Choix des technologies clients

Pour le développement de l'interface utilisateur de LearnHub, nous avons choisi les technologies suivantes :

— **jQuery** : Utilisé pour effectuer des appels AJAX et manipuler le DOM de manière simplifiée.

## 3.2 Plan de la partie client

L'application LearnHub est conçue comme une application monopage (SPA) gérer pas JavaScript pour une expérience utilisateur fluide. Voici les principaux écrans et leurs fonctionnalités :

— **Écran d'accueil** : Affiche la barre de recherche et ses résultats, ainsi que les options de connexion et d'inscription. Les appels aux composants du serveur incluent la récupération des documents (`GET /documents`). Si l'utilisateur est connecté, affiche un bouton pour l'ajout d'un nouveau contenue.

— **Écran de connexion** : Formulaire pour la connexion des utilisateurs. Les appels au serveur incluent l'envoi des informations de connexion (`POST /login`).

— **Écran de profil** : Affiche les informations de profil de l'utilisateur, ainsi que ses documents partagés. Les appels au serveur incluent la récupération des informations de profil (`GET /profile`) et la mise à jour du profil (`POST /profile`).

## 3.3 Liste des appels AJAX au serveur

Les appels AJAX sont effectués pour communiquer avec le serveur et manipuler les données sans recharger la page. Voici une liste des principaux appels AJAX utilisés dans LearnHub :

— **Inscription** : `POST /register` - Envoie les informations d'inscription au serveur.

— **Connexion** : `POST /login` - Envoie les informations de connexion au serveur.

— **Récupération des profils** : `GET /profile` - Récupère les informations du profil de l'utilisateur.

— **Mise à jour des profils** : `POST /profile` - Envoie les mises à jour du profil de l'utilisateur.

— **Ajout de document** : `POST /documents` - Envoie les informations d'un nouveau document.

— **Récupération des documents** : `GET /documents` - Récupère la liste des documents partagés.

## 3.4 Exemple de gestion des événements et des callbacks

Le fichier `script.js` montre comment nous gérons les événements et les callbacks pour les formulaires d'inscription et de connexion. Voici un extrait :

```
document.addEventListener('DOMContentLoaded', function() {
    let formInscription = document.getElementById('formInscription');
    if (formInscription) {
        formInscription.addEventListener('submit', function(event) {
            event.preventDefault();
            const nom = document.getElementById('nom').value;
            const prenom = document.getElementById('prenom').value;
            const date_naissance = document.getElementById('date-de-naissance').value;
            const niveauEducation = document.getElementById('niveauEducation').value;
            console.log(`Nom : ${nom}, Prénom : ${prenom}, Date de Naissance :
${date_naissance}, Niveau d'éducation : ${niveauEducation}`);
            window.location.href = 'page-daccueil.html';
        });
    }

    let formConnexion = document.getElementById('formConnexion');
    if (formConnexion) {
        formConnexion.addEventListener('submit', function(event) {
            event.preventDefault();
            const email = document.getElementById('email').value;
            const password = document.getElementById('motDePasse').value;
            console.log(`Login Attempt: Email - ${email}, Password - ${password}`);
            window.location.href = 'page-daccueil.html';
        });
    }
});
```

Ce script gère les événements de soumission des formulaires d'inscription et de connexion, empêche le comportement par défaut du formulaire, et redirige l'utilisateur vers la page d'accueil après la soumission.

Le fichier `scriptFormulaire.js` montre comment charger dynamiquement des formulaires via AJAX et manipuler les données JSON. Voici un extrait :

```
$(document).ready(function(){
    attacherEvenement();
});

function chargerFormulaire() {
```

```
    $.ajax({
        url: "/formulaire",
        type: "GET",
        success: function(response) {
            $("#formulaireContainer").html(response);
            attacherEvenement();
            document.getElementById("buttonFormulaire").style.display = "none";
            editSelect();
        },
        error: function(xhr, status, error) {
            console.error("Erreur lors du chargement du formulaire: " + error);
        }
    });
}

function getYoutubeInformations(videoId) {
    let API_KEY = "API_KEY_HERE";
    $.ajax({
        url: `https://www.googleapis.com/youtube/v3/videos?
id=${videoId}&key=${API_KEY}&part=snippet`,
        type: "GET",
        success: function(response) {
            let videoTitle = response.items[0].snippet.title;
            let videoAuthors = response.items[0].snippet.channelTitle;
            let videoDate = response.items[0].snippet.publishedAt.substring(0, 10);
            document.getElementById("documentTitle").value = videoTitle;
            document.getElementById("documentAuthors").value = videoAuthors;
            document.getElementById("documentDate").value = videoDate;
        },
        error: function(xhr, status, error) {
            console.error("Erreur lors de la récupération
des informations de la vidéo YouTube: " + error);
        }
    });
}
```

Ce script montre comment charger des formulaires et récupérer des informations de vidéos YouTube via des appels AJAX.

# 4 Schéma Global du Système

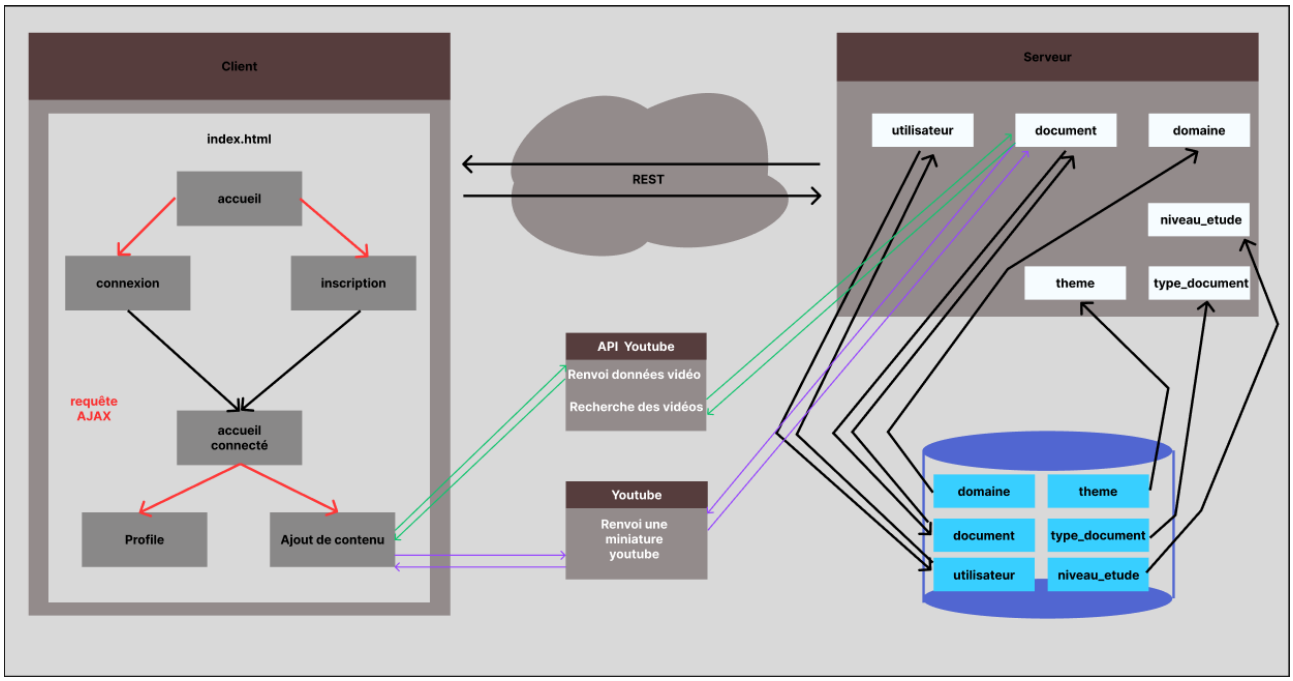


FIGURE 1 – Schéma global du système LearnHub. Le schéma montre les différentes interactions entre le client (navigateur), le serveur (API REST) et la base de données. Les flèches indiquent les flux de données et les requêtes HTTP effectuées entre les composants. Réaliser avec le logiciel Figma.

# 5 Sources

Lors de ce projet, nous nous sommes référés aux documentations des langages utilisé disponible sur le web.