

Titre : Développement de l'interface client (front-end) pour la visualisation et l'interaction avec les données collectées.

Cours	Technologies de l'inforoute (INF37407)
Session	Automne 2025
Contact	yacine_yaddaden@uqar.ca

Table des matières

1.	Objectif du travail	1
2.	Technologies à utiliser	1
3.	Description détaillée	2
3.1.	Définition des différentes requêtes à utiliser	2
3.2.	Mise en place de l'environnement <i>front-end</i>	2
3.3.	Gestion du profil utilisateur	2
3.4.	Récupération des données	2
3.5.	Développement de l'interface utilisateur	2
3.6.	Mise en place des filtres dynamiques	2
3.7.	Affichage des données – graphiques	2
3.8.	Exporter des rapports en PDF	2
3.9.	Déploiement de l'application	3
4.	Structure du projet	3
5.	Modalité d'évaluation	3
6.	Date de remise	3
7.	Assistance	3
8.	Points importants	3
9.	Documentation	3

1. Objectif du travail

L'objectif de ce deuxième travail pratique est de développer une interface client, ou une application Web, permettant d'interagir avec l'API REST et l'API GraphQL réalisées durant le premier travail pratique. Ce projet vise à mettre en pratique les notions apprises en lien avec React et à faciliter l'accès aux données au moyen d'une interface conviviale. Il s'agit également de produire des graphiques interactifs en utilisant TypeScript combiné avec les différentes bibliothèques disponibles.

2. Technologies à utiliser

Dans le cadre du projet, il sera nécessaire d'utiliser les technologies suivantes :

- Environnement de développement :
 - Microsoft Visual Studio Code
- Langages de programmation :
 - TypeScript
- Bibliothèques JavaScript :
 - React (*bibliothèque front-end*)
 - Next.js (*optionnel*)
 - React-router (*routage et navigation*)
 - Redux Toolkit (*gestion de l'état et des données*)
 - Axios (*requêtes HTTP*)
- Interface graphique :
 - ShadCN UI (*Framework CSS*)
 - Lucide React (*icônes*)
 - Recharts (*graphique et visualisation de données*)
 - jsPDF + html2canvas (*exportation en PDF*)
- Logiciels :
 - Postman

Il est recommandé de faire de la *gestion de version* avec l'outil [Git](#). Afin de faciliter le travail d'équipe, il est recommandé de créer un dépôt **GitHub**.

INFORMATION : il est possible d'utiliser d'autres paquets si c'est jugé pertinent pour la réalisation du projet.

3. Description détaillée

Afin de réaliser le projet, il est nécessaire de suivre les étapes suivantes :

3.1. Définition des différentes requêtes à utiliser

Cette étape consiste à analyser les API REST et GraphQL réalisées lors du premier travail pratique afin d'identifier précisément les requêtes nécessaires au bon fonctionnement de l'application *front-end*. Les étudiants doivent répertorier les *endpoints* et requêtes GraphQL à utiliser pour : l'authentification, la gestion du profil, la récupération des jeux de données, l'application des filtres, et l'obtention des statistiques. Chaque requête devra être documentée (méthode, URL, paramètres, corps et structure de la réponse) afin d'assurer une intégration claire et cohérente dans l'interface cliente.

3.2. Mise en place de l'environnement *front-end*

Les étudiants doivent configurer un environnement de développement moderne en créant une application React en TypeScript. L'installation des bibliothèques nécessaires (React Router, Redux Toolkit, Axios, ShadCN UI, Lucide React, Recharts, jsPDF et html2canvas) doit être effectuée dès le départ. L'architecture du projet doit être organisée en modules avec des sous-dossiers dédiés pour les composants, pages, services API, slices Redux et types TypeScript. Cette mise en place garantit une structure claire, scalable et alignée avec les bonnes pratiques actuelles du développement *front-end*. Il est également possible d'utiliser Next.js comme alternative

3.3. Gestion du profil utilisateur

Cette section vise à implémenter les fonctionnalités permettant à un utilisateur de consulter et modifier ses informations personnelles. Les étudiants doivent créer une page dédiée au profil, connectée à l'API REST, permettant d'afficher les informations de l'utilisateur authentifié (nom, courriel, rôle, etc.) et d'effectuer des mises à jour (modification du courriel, du mot de passe ou de l'avatar, si disponible). L'interface doit être sécurisée, accessible uniquement après authentification, et synchronisée avec le serveur via Redux Toolkit.

3.4. Récupération des données

Les données exposées par les API REST et GraphQL du TP1 doivent être récupérées via Axios et intégrées dans Redux Toolkit. Les étudiants doivent configurer les requêtes HTTP, gérer l'authentification par *token* et définir des

slices Redux permettant de stocker l'état de la récupération des données : chargement, succès, erreur. Les informations obtenues doivent ensuite être exploitées par les composants de l'interface pour alimenter les listes, les tableaux, les statistiques et les graphiques.

3.5. Développement de l'interface utilisateur

L'interface doit être construite de manière ergonomique et moderne en utilisant ShadCN UI et Lucide React. Les étudiants doivent développer les différentes pages de l'application, notamment : la page d'accueil, la liste des données, la page de détails, la page du profil utilisateur, la page de statistiques, et les interfaces d'inscription/connexion. Chaque page doit intégrer les composants graphiques nécessaires, assurer une navigation fluide via React Router et respecter une cohérence visuelle entre les différentes sections.

3.6. Mise en place des filtres dynamiques

Les filtres doivent permettre aux utilisateurs d'affiner l'affichage des données selon plusieurs critères (catalogue, mots-clés, thématique, date, producteur, etc.). Les étudiants doivent créer des composants interactifs tels que des menus déroulants, champs de recherche, commutateurs et sélecteurs multiples. Les filtres doivent déclencher des requêtes vers l'API et mettre à jour les données affichées ainsi que les graphiques associés. L'objectif est d'offrir une expérience utilisateur fluide et réactive.

3.7. Affichage des données – graphiques

Les étudiants doivent développer une page dédiée à la visualisation statistique des données. À l'aide de Recharts, ils devront produire au minimum trois graphiques interactifs illustrant des tendances, répartitions ou comparaisons pertinentes (ex. répartition par catalogue, histogramme des dates, évolution temporelle). Les graphiques doivent se mettre à jour automatiquement lorsque les filtres définis précédemment sont appliqués.

3.8. Exporter des rapports en PDF

L'application doit offrir la possibilité d'exporter les données ou les graphiques affichés au format PDF. Les étudiants doivent utiliser *jsPDF* combiné avec *html2canvas* pour capturer les éléments visuels affichés dans l'interface et générer un document PDF contenant un en-tête, la date d'exportation et les éléments

sélectionnés. Cette fonction doit être accessible via un bouton bien identifié et fournir un fichier PDF exploitable.

3.9. Déploiement de l'application

La dernière étape consiste à déployer l'application sur une plateforme d'hébergement telle que **Vercel**. Les étudiants doivent configurer les paramètres de *build*, assurer le fonctionnement correct des routes et vérifier que les appels aux API restent fonctionnels dans l'environnement déployé. Le lien public de l'application devra être inclus dans le rapport final et dans la démonstration vidéo.

4. Structure du projet

Il est recommandé d'utiliser l'outil *nano-react-app* pour créer le projet en utilisant le modèle TypeScript. Cela permettra de mettre en place une structure minimale et propre dès le départ. Par la suite, il sera nécessaire de créer des sous-dossiers afin de structurer et d'organiser les différents composants, pages, services et autres éléments du projet.

5. Modalité d'évaluation

Le travail pratique comptera pour **30%** de la note du cours. Ces points sont répartis de la manière suivante :

Partie	Points
Définition des requêtes et analyse des API	1,0%
Mise en place de l'environnement <i>front-end</i>	1,0%
Authentification et gestion du profil utilisateur	4,0%
Récupération et gestion des données	4,0%
Développement de l'interface utilisateur	4,0%
Mise en place des filtres dynamiques	4,0%
Visualisation des données – graphiques	4,0%
Fonctionnalité d'exportation PDF	2,0%
Déploiement de l'application sur Vercel	2,0%
Qualité du code source	1,0%
Rapport rédigé en Français	2,0%
Démonstration vidéo de l'application	1,0%
Total	30%

6. Date de remise

- La date limite de remise est le **18 décembre 2025 avant 23h00**
- Fichiers à remettre :
 - ✓ Un fichier compressé (.zip) contenant le code source,
 - ✓ Le rapport (un modèle **LaTeX** est fourni),
 - ✓ Une courte vidéo de démonstration faite avec [OBS Studio](#).

Tous les fichiers doivent être remis sur Moodle avant la date limite.

Si la vidéo est trop volumineuse, il est possible de la téléverser sur YouTube et d'inclure le lien dans le rapport et en commentaire de la remise.

7. Assistance

Si vous avez besoin d'assistance, il est possible de contacter le professeur directement par courriel afin d'organiser une rencontre ZOOM. Des questions peuvent également être posées à la fin des séances de cours. Si un auxiliaire d'enseignement est disponible, il pourra également vous assister.

8. Points importants

Note I :

- Le travail est en équipe de deux,
- Le professeur peut poser des questions liées au travail pratique,
- Le non-respect de l'énoncé entraînera une perte de points,
- Le plagiat sera sanctionné selon la politique de l'université,
- Le retard dans la remise du projet entraînera des pénalités.

Note II : *Dans le cas où il y a des aspects qui ne sont pas clairs, n'hésitez pas à m'en faire part afin que je puisse apporter des éclaircissements et éventuellement mettre à jour l'énoncé du travail pratique.*

9. Documentation

La documentation à utiliser est disponible sur les sites officiels (*voir plus haut*). Vous pouvez également vous référer à celle fournie dans le plan de cours.