

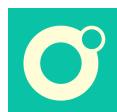
Dossier de Projet



Présenté Et Soutenu Par
Maxime Malandain

En vue de l'obtention du Titre Professionnel
Développeur Web et Web mobile Niveau V

Centre de Formation O'Clock
Promotion : Uranium
2023



Remerciement

Je tiens à exprimer ma sincère gratitude envers toutes les personnes qui ont joué un rôle essentiel dans ma réussite tout au long de ce projet de reconversion. Leur soutien et leur contribution ont été inestimables, et je tiens à les remercier chaleureusement.

Tout d'abord, un immense merci à ma femme pour sa patience et son soutien inébranlable tout au long de ce parcours de reconversion. Sa compréhension, sa patience et son encouragement ont été d'une importance capitale pour moi.

Je tiens également à exprimer ma reconnaissance envers l'école O'Clock pour la qualité exceptionnelle de ses enseignements. Les connaissances et les compétences acquises au sein de cette école ont été essentielles pour ma réussite, et je suis reconnaissant de l'opportunité qui m'a été offerte.

Mes remerciements s'adressent également à tous les professeurs d'O'Clock, dont la disponibilité, les compétences pédagogiques et le dévouement ont grandement contribué à mon parcours d'apprentissage. Leurs conseils et leur soutien ont été inestimables.

Un grand merci à mes collègues apprenants pour la bonne humeur, l'entraide et le partage de connaissances tout au long de notre parcours. Votre présence a rendu cette expérience d'apprentissage encore plus enrichissante.

Enfin, je souhaite exprimer ma reconnaissance envers Google, notre inestimable ami virtuel, pour avoir répondu à des milliers de questions et avoir été une source inépuisable de connaissances. Google a été notre partenaire dans cette aventure d'apprentissage, et je suis reconnaissant pour sa disponibilité constante.

Merci à toutes ces personnes et à ces ressources qui ont contribué à faire de ce projet une réalité, et qui ont rendu cette transition professionnelle possible. Votre soutien et votre contribution ont été un moteur essentiel de ma réussite.

Sommaire

I. Introduction	5
II. Liste des compétences couvertes par le projet	6
III. Résumé du projet.....	8
IV. Cahier des charges.....	9
A) Public cible et définition des besoins.....	9
B) Wireframes.....	9
C) Arborescence des pages.....	12
D) User Stories.....	13
E) MVP.....	14
V. Spécifications techniques du projet.....	15
A) Versionning.....	15
B) Front-end.....	15
C) Back-end.....	16
D) MCD/ MLD/ Dico des données.....	17
E) Sécurité.....	22
F) Circulation de la données.....	23
VI. Gestion de projet.....	25
A) Présentation de l'équipe.....	25
B) Méthodes et outils de travail.....	26
C) Sprints.....	27
D) Difficultés rencontrées.....	29
VII. Réalisations personnelles.....	30
A) authController.....	30
B) verifyAuthMiddleware et une partie du routeur.....	32

C) multerMiddleware.....	34
VIII. Jeu d'essai.....	36
IX. Vulnérabilités de sécurité et veille.....	37
A) Cookie ou localStorage pour Token JWT.....	37
B) Sanitize sur toutes les routes?.....	38
X. Recherches à partir de site anglophone extrait et traduction.....	39
XI. Conclusion.....	41
XII. Annexe.....	42
A) Wireframes.....	42
B) Maquettes.....	50
C) Dico des données.....	55
C) Dico des données.....	55

I. Introduction

Mon parcours professionnel m'a conduit à des années d'engagement au sein des forces de police, une carrière que j'ai abordée avec passion et dévouement.

En tant que policier, j'ai eu l'opportunité de servir les citoyens, de faire preuve de leadership et de prendre des décisions cruciales dans des situations complexes.

Cependant, au fil des années, une deuxième passion a émergé : l'informatique.

Mon intérêt pour le monde de l'informatique et du développement web est devenu de plus en plus évident au fil du temps.

Les aspects fascinants de la programmation, de la création d'applications et de la résolution de problèmes techniques m'ont toujours attiré.

C'est ainsi que j'ai décidé d'explorer cette passion naissante et d'embrasser une nouvelle carrière en tant que développeur web.

C'est là que la formation intensive d'O'Clock est entrée en jeu.

Elle m'a offert la possibilité d'acquérir un ensemble complet de compétences de développement full-stack en seulement trois mois.

Cette formation a posé les bases de ma transition vers le monde du développement web.

Après avoir consolidé mes connaissances de base, j'ai consacré un mois supplémentaire à me spécialiser dans React, un framework JavaScript puissant et polyvalent.

Cette spécialisation a renforcé ma capacité à créer des applications web dynamiques et réactives.

Avec ces compétences en main, j'ai rejoint une équipe de collègues développeurs pour entreprendre un projet ambitieux : Tok-Tok. Notre objectif était de créer un réseau social de proximité qui renforcerait les liens entre voisins et faciliterait l'échange de services et de biens.

Tok-Tok est le fruit de notre engagement à lutter contre l'isolement social, à promouvoir les entreprises locales, à revitaliser les quartiers et à soutenir l'économie circulaire.

II. Liste des compétences couvertes par le projet

A. Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

1) Maqueter une application

Nous avons élaboré un cahier des charges détaillé pour Tok-Tok, décrivant les fonctionnalités attendues et les **besoins des utilisateurs**. Nous avons également créé des **wireframes** avec des commentaires pour les vues **desktop et mobile** avec **Tldraw et Figma**, en nous appuyant sur les **user stories** ainsi de définir l'**aborescence des pages**. Cela a permis de définir la structure de l'application et d'orienter sa conception.

Note: Prototypes disponibles en Annexe

2) Réaliser une interface utilisateur web statique et adaptable

Dans le cadre du projet Tok-Tok, nous avons accordé une attention particulière à la **sémantique** et au **référencement**. Bien que le **responsive design** prévu dans les wireframes n'ait pas pu être mis en œuvre en raison de contraintes de temps, nous avons pu démontrer notre capacité à le réaliser dans d'autres projets inclus dans mon dossier professionnel. De plus, même si le **HTML et le CSS (ainsi que le framework Material-UI)** que nous avons utilisés étaient imbriqués dans les composants de React, cela a montré notre compétence en matière d'intégration web statique à défaut des exemples sont également présents dans le dossier professionnel.

3) Développer une interface utilisateur web dynamique

Dans le cadre de la CP3, nous avons développé des interfaces utilisateur web dynamiques en utilisant des langages de script côté client, notamment **JavaScript**. Cela a impliqué la mise en place de fonctionnalités **interactives pour améliorer l'expérience de l'utilisateur**. Notre projet Tok-Tok, réalisé en **React, Redux, Axios, et TypeScript**, en est un exemple concret. Nous avons mis en place des interactions telles que des requêtes **AJAX** pour récupérer des données en temps réel, la gestion d'états complexes à l'aide de Redux, et la création d'une interface utilisateur réactive. Ces interactions ont permis aux utilisateurs d'interagir avec le contenu de manière fluide, tout en maintenant un site web dynamique.

B. Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité

1) Créer une base de données

Nous avons conçu la base de données de Tok-Tok en créant un Modèle Conceptuel des Données (**MCD**), un Modèle Logique des Données (**MLD**), et un Dictionnaire des Données (**DdD**). Nous n'avons pas créé de Modèle Physique de Données (**MPD**) mais dans la partie V. de la spécification techniques du projets nous montrons les script sql qui servent à faire les imports: créations des tables et insertions de données ainsi que nos data.

2) Développer les composants d'accès aux données

Dans le cadre de la CP6, nous avons établi la connexion de notre application Tok-Tok à une base de données **PostgreSQL** en utilisant **Sequelize** avec **Express**. La structure de notre application assure un accès sécurisé aux données en mettant en place des services et des modèles pour les opérations **CRUD**. Ces modèles facilitent la gestion des données de manière efficace.

Pour garantir la **sécurité**, nous avons pris des mesures pour protéger les informations sensibles, telles que les données de connexion à la base de données. Nous avons utilisé un fichier **.env** pour stocker ces informations en toute sécurité, en veillant à exclure les fichiers sensibles du dépôt Git grâce au fichier **.gitignore**, renforçant ainsi la sécurité de l'application.

3) Développer la partie back-end d'une application web ou web mobile

Nous avons élaboré **l'architecture** de l'ensemble du backend de l'application Tok-Tok. Les **routes** ont été établies pour permettre une interaction fluide entre le frontend et le serveur. Pour renforcer la sécurité, nous avons incorporé des **middlewares** essentiels, tels que la vérification **JWT** pour l'authentification et **Sanitize** pour protéger les données sensibles contre les injections malveillantes.

De plus, les contrôleurs ont été développés pour gérer de manière optimale les données. Pour garantir une sécurité renforcée, nous avons fait usage d'outils tels que **bcrypt pour le hachage des mots de passe**, **cors pour gérer les autorisations d'accès**, **email-validator pour valider les adresses e-mail**, et **uuid pour générer des identifiants uniques**. De surcroît, l'intégration de **Multer** a simplifié la gestion des fichiers, contribuant à une expérience utilisateur plus fluide et sécurisée.

III. Résumé du Projet

Le projet Tok-Tok vise à créer un réseau social de proximité qui rapproche les voisins et renforce la communauté locale. La plateforme en ligne offre un espace interactif permettant aux utilisateurs de partager des informations, de réagir à des publications, d'échanger des services, de publier des petites annonces, et bien plus encore.

Le public cible de Tok-Tok comprend des individus résidant dans le périmètre de l'utilisateur, allant des petits villages aux grandes villes, âgés de 15 ans et plus.

Les fonctionnalités clés du projet comprennent la création et la réaction à des publications, la publication de petites annonces, la recherche de contenus spécifiques, la gestion de profils personnalisés, une fonction de géolocalisation pour visualiser la proximité des membres et des annonces.

Les objectifs du projet incluent la création de liens sociaux pour lutter contre l'isolement, la promotion des commerces locaux, la revitalisation des quartiers, le développement de l'économie locale et circulaire.

Le projet utilise des technologies front-end telles que React, TypeScript, Redux-toolkit, Sass et côté back-end, NodeJS, Express, Postgres, Sequelize et plusieurs autres dépendances pour la sécurité.

L'architecture de l'application est organisée en différentes routes pour gérer les fonctionnalités, y compris l'authentification, la publication de contenu, la messagerie, les annonces, les profils, et bien d'autres.

En résumé, Tok-Tok offre une plateforme conviviale qui vise à renforcer la communauté de proximité en facilitant les interactions entre voisins et en promouvant l'échange de services et de biens, contribuant ainsi au bien-être et à la solidarité au sein de la communauté locale.

IV. Cahier des charges

A) Public cible et définition des besoins

- Voisins : Individu proche d'un rayon donné de l'utilisateur (5km-ville)
- Entre 15 ans et encore en vie
- Petit village à grande ville.

Problèmes		Solutions
Isolement des personnes	→	Création de lien social
Manque de visibilité des commerces locaux	→	Commerces et services de proximité (pro)
Raviver des quartiers	→	Promotion d'événements locaux
Développement de l'économie locale et circulaire	→	Échange de services (particuliers)

B) Wireframes

Dans le cadre de notre projet, nous avons utilisé des wireframes pour concevoir l'interface utilisateur de l'application. Les wireframes sont des maquettes initiales et rudimentaires qui servent à représenter la disposition des éléments sur une page, sans se soucier des détails de conception tels que les couleurs ou les polices. Ils sont essentiels pour la planification et la conception de l'interface utilisateur.

Pour créer ces wireframes, nous avons opté pour des outils de conception tels que **Tldraw** et **Figma**, qui nous ont permis de travailler de manière collaborative en temps réel sur un même fichier partagé. Nous partageons ici trois versions pour chaque page clé de l'application : la page d'accueil (**homepage**), la page d'accueil pour les membres (**member homepage**) et la page de création d'annonce (**create advert**). Ces versions comprenaient une adaptation pour les appareils **mobile** et les **desktop**.

Les wireframes ont été un élément clé de notre processus de conception, car ils nous ont permis de visualiser la structure de chaque page, d'ajuster la disposition des éléments et de garantir une expérience utilisateur fluide. Cela nous a également aidés à partager une vision commune du design de l'application et à faciliter la communication entre les membres de l'équipe. Les wireframes ont servi de base pour le développement de l'interface utilisateur et ont contribué à créer une application conviviale et esthétiquement cohérente.

Note: Autres Wireframes et Maquettes disponibles en Annexe

Overview of the site's pages to whet the appetite
(Member homepage, profile, adverts, advert and PM)

Soft fade transition

Logo
ce site est super cool, inscrivez vous svp

S'inscrire **Se connecter**

Prenom Nom
 Adresse postale
 adresse@mail.com
 Mot de passe
 J'accepte les CGV
S'inscrire

En vous inscrivant, vous acceptez les Conditions d'utilisation et la Politique de confidentialité, notamment l'utilisation des cookies.

A form with a tab to register or log in

Tok-Tok

Venez discutez avec vos voisins

S'inscrire **Se connecter**

Au clic le formulaire prend toute la hauteur de l'écran

Tok-Tok

S'inscrire **Se connecter**

ce site est super cool, inscrivez vous svp

Prenom
 Nom
 Adresse postale
 adresse@mail.com
 Mot de passe
 J'accepte les CGV
S'inscrire

→ "Toggable"

Create an advert

Logo Recherche

Créer un annonce

Catégorie ▾ Distance 1 km ▾ Trier Date : croissant ▾

Ventes Dons

Title

Content

Importer un média

Aperçu de l'image

Category ▾

Price €

Publier Annuler

i

Titre Prix date km Titre Prix date km Titre Prix date km

Créer une annonce

Title

Content

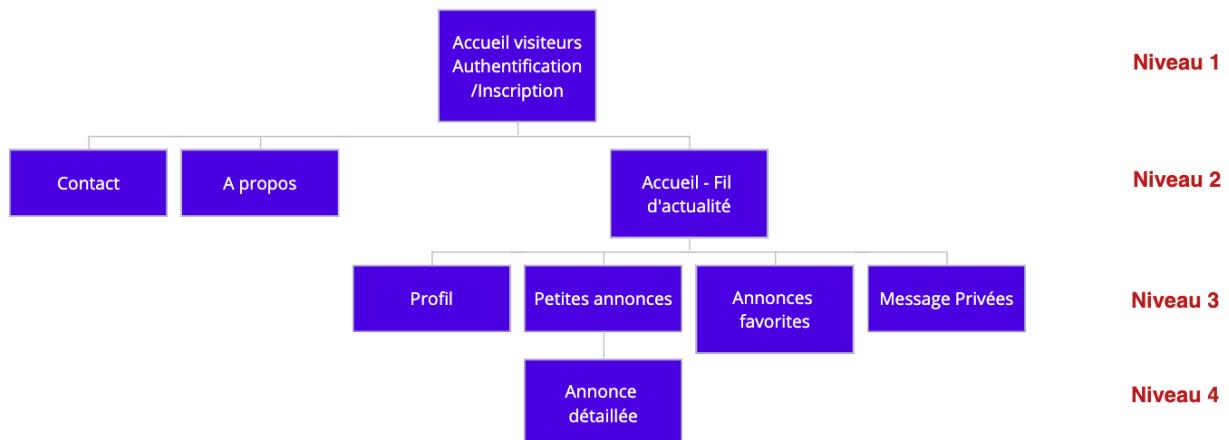
ajouter une photo

Category ▾

Price € Donation Ⓜ

Publier

C) Arborescence des pages



D) User Stories

En tant que...	je veux pouvoir..	dans le but de...	Feature
Visiteur	avoir un aperçu du site	Décider si je m'inscris	Accueil (visiteur)
	m'inscrire	pouvoir m'identifier	
	m'identifier	voir la communauté et ses agissements	
Utilisateur	me connecter	accéder à mon compte	Login
	me déconnecter	fermer l'accès à mon compte	
	modifier mes informations personnelles	mettre à jour mon profil (mdp, adresse, email, nom etc)	Profil
	cliquer sur un profil	afficher ses posts, ses informations publiques et ses annonces (et la suivre ?)	
	voir les posts des personnes dans mon quartier	me tenir au courant	Posts
	écrire et envoyer un post	m'exprimer, informer mes voisins	
	modifier un post	corriger une erreur	
	supprimer un post		
	répondre à un post		
	aimer un post	montrer mon intérêt	
	ne plus aimer un post		
	cacher un post	qu'il n'apparaisse plus sur le fil d'actualités	
	voir les annonces autour de moi	acheter sur ou en dehors de la plateforme	Annonces
	Créer une annonce	proposer un bien ou un service	
	supprimer une annonce	retirer l'annonce	
	Cliquer sur une annonce	la consulter	
	Répondre à une	entrer en contact avec l'annonceur	

En tant que...	je veux pouvoir..	dans le but de...	Feature
	annonces		
	enregistrer des annonces	les retrouver plus tard	Favoris annonces
	supprimer une annonce de mes favoris	plus voir l'annonce dans mes favoris	
	rechercher des posts	voir des posts	Rechercher
	rechercher des annonces		
	cliquer sur ma messagerie	voir mon historique de conversation	MP
	envoyer un message privé	communiquer avec un membre	

E) MVP (Minimal Viable Product)

- Login / Inscription
- Géolocalisation
- Profil
- Écrire/Répondre/Réagir au post (fil d'actualité)
- Petites Annonces (création et favoris)
- Recherche (annonces/posts/membres)

Évolutions potentielles:

- Messages Privées
- Notifications
- Panel de modération
- Système de paiement
- Groupes
- Badges de participation
- Créer des évènements
- Partager un message ou une annonce
- Profils professionnels

V. Spécifications techniques du projet

A) Versionning

Le versioning de notre projet a été effectué grâce à **Git**, un système de contrôle de version populaire qui a grandement facilité la collaboration au sein de notre équipe de développement. En plus de garantir une gestion efficace des modifications, Git a également permis de conserver une trace des versions antérieures du site, ce qui s'est avéré précieux en cas de problèmes.

Pour maintenir l'ordre et l'organisation dans notre workflow de développement, nous avons adopté une approche stratégique en créant **deux dépôts Git distincts : un pour la partie Front-End et un autre pour la partie Back-End** du projet Tok-Tok. Nous avons pris soin de ne pas altérer la branche principale (**main**), conservant ainsi sa stabilité.

Lorsqu'il était nécessaire d'implémenter une nouvelle fonctionnalité ou d'atteindre une étape clé du projet, nous avons suivi une méthodologie rigoureuse. Nous avons créé une nouvelle branche spécifique pour chaque tâche, assurant ainsi un environnement de développement clair et organisé. Une fois le travail sur la nouvelle fonctionnalité achevé, nous avons amorcé une **Pull Request (PR)** pour fusionner cette branche dans la branche de développement. Cette étape cruciale incluait une revue détaillée du code (**code review**), permettant de détecter les erreurs potentielles et d'apporter des ajustements si nécessaire. Si la Pull Request était approuvée et ne présentait aucun conflit avec la branche de destination, nous avons procédé à la fusion (**merge**) avec soin. Cela signifiait que le contenu de la branche de fonctionnalité s'intégrait proprement dans la branche de développement, symbolisant la version la plus avancée du projet.

Cette méthodologie de gestion de version nous a permis de maintenir un code propre, de minimiser les risques de conflits majeurs et d'assurer un développement efficace de Tok-Tok. Elle a grandement contribué à la cohérence et à l'organisation de notre projet, tout en favorisant une collaboration fluide au sein de l'équipe.

B) Technologies côté front-end

Pour la partie Front-End de Tok-Tok, nous avons utilisé une gamme complète de technologies de pointe. Le projet a été développé avec **React**, une bibliothèque JavaScript populaire pour la création d'interfaces utilisateur réactives. React a été étendu avec **Material-UI (MUI)**, qui est devenu l'un des frameworks d'interface les plus utilisés. Material-UI fournit un ensemble complet de composants React pré-conçus, ainsi qu'une conception visuelle cohérente basée sur les principes du Material Design.

Pour gérer l'état global de l'application, nous avons utilisé **Redux et Redux Toolkit**. Ces outils nous ont permis de centraliser et de gérer efficacement l'état de l'application, ce qui est essentiel pour un projet de cette envergure.

React Router Dom a été utilisé pour la gestion de la navigation, garantissant une expérience utilisateur fluide tout en maintenant une structure d'URL cohérente.

L'aspect visuel du projet a été amélioré grâce aux bibliothèques **Iconify React et MUI Icons Material**, permettant d'intégrer des icônes et des graphiques personnalisés.

Pour gérer les requêtes HTTP vers le serveur, **Axios** a été utilisé, assurant une communication efficace entre le Front-End et le Back-End.

La persistance des données a été mise en œuvre grâce à **Redux Persist**, qui a contribué à stocker et à maintenir l'état de l'application entre les sessions.

Pour garantir la qualité du code et une expérience de développement homogène, **TypeScript** a été utilisé pour ajouter un typage statique à JavaScript. De plus, le développement a été facilité grâce à **Vite**, un outil de build rapide pour les applications web modernes. L'intégration de **Sass** a permis de gérer de manière efficace les styles, tandis qu'**ESLint et Prettier** ont été employés pour garantir des normes de code strictes et une qualité de code élevée.

C) Technologies côté back-end

Le Back-End de Tok-Tok est construit autour d'une sélection de technologies performantes qui garantissent la stabilité et l'efficacité de notre serveur:

-Pour commencer, nous avons opté pour **Express**, un framework **Node.js** renommé, pour la création de notre serveur Web. Son extensibilité et ses fonctionnalités en font un choix judicieux pour la construction d'applications Web dynamiques.

-Pour la gestion de notre base de données, nous avons choisi **PostgreSQL**, une solution de base de données relationnelle éprouvée. Elle nous permet de stocker et d'accéder aux données de manière fiable et structurée.

-Pour simplifier les interactions avec la base de données, nous avons mis en place **Sequelize**, un **ORM** (Object-Relational Mapping) qui offre un modèle de données intuitif en JavaScript.

-En ce qui concerne la gestion des fichiers, **Multer** a été notre choix pour une manipulation aisée et sécurisée.

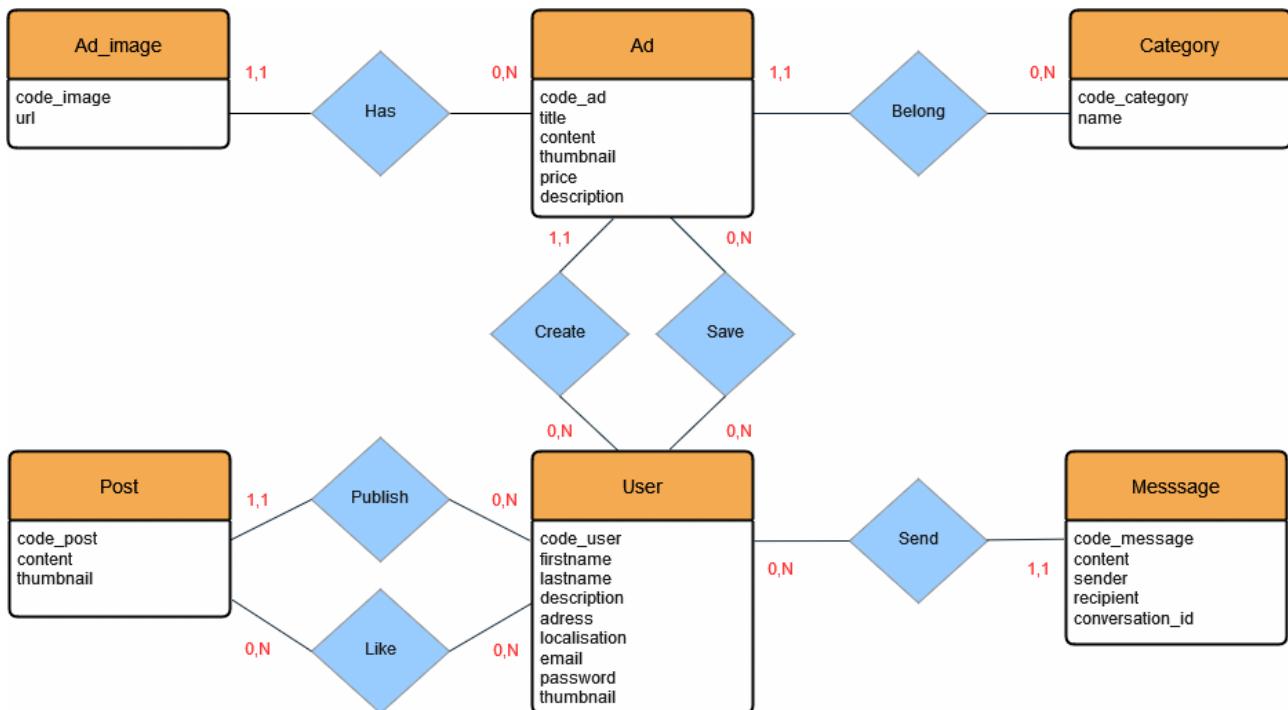
Cette combinaison de technologies garantit un Back-End fiable, performant et sécurisé

Note: voir ci dessus la partie Sécurité

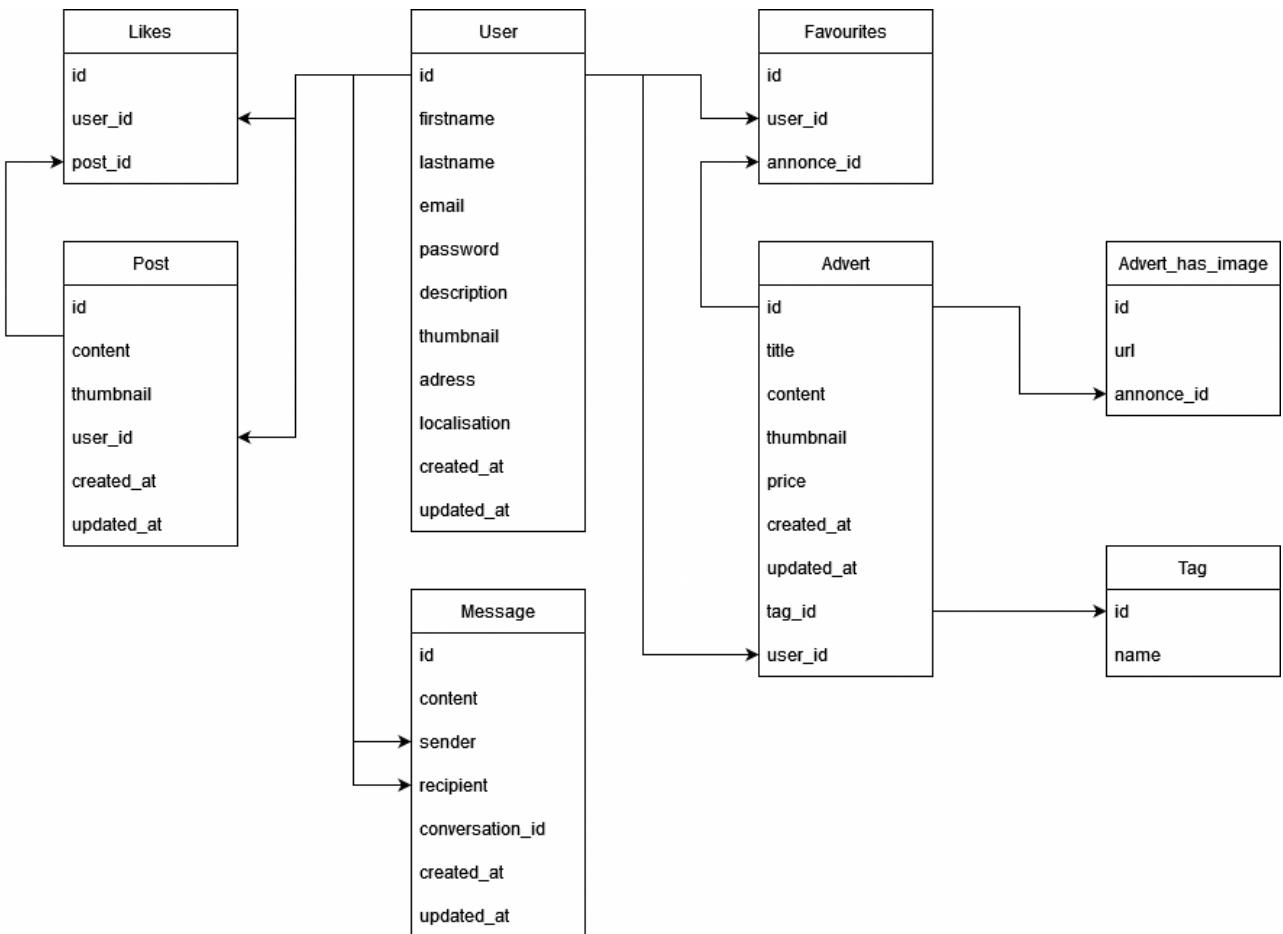
D) MCD/ MLD/ Dico des données

Dans notre projet Tok-Tok, nous avons utilisé l'outil MOCODO puis Draw.io pour la création de notre Modèle Conceptuel de Données (MCD) et de notre Modèle Logique de Données (MLD).

Le MCD représente la structure globale de notre base de données en identifiant les entités, les relations et les attributs clés. Il offre une vue conceptuelle de la manière dont les données sont organisées dans notre application.



Le Modèle Logique de Données (MLD) est une étape cruciale dans la conception de bases de données. Il permet de passer du Modèle Conceptuel de Données (MCD), qui est principalement une représentation conceptuelle des données et de leurs relations, à une représentation plus précise et technique des données dans la base de données.



Ces modèles ont été essentiels dans le développement de notre application. Ils nous ont permis de planifier et de concevoir la base de données en fonction des besoins de l'application. En utilisant Draw.io, nous avons pu créer visuellement ces modèles, ce qui a simplifié la communication au sein de l'équipe de développement et nous a aidé à avoir une vue claire de l'architecture de la base de données de Tok-Tok. Ils servent de référence précieuse pour la création, la maintenance et l'évolution de notre base de données tout au long du développement de l'application.

Le Dico des données

Le Dictionnaire des Données, basé sur le Modèle Logique de Données (MLD), est une composante cruciale de l'architecture de l'application Tok-Tok. Il offre une vue systématique et organisée de la structure de la base de données, ce qui facilite la gestion des informations essentielles de notre système. Voici un aperçu du contenu de notre Dictionnaire des Données :

Note: L'intégralité du Dictionnaire des Données est en annexe

Users (Utilisateurs)

Champ	Type	Spécificités	Description
id	INT	PRIMARY KEY, UNSIGNED, NOT NULL, AUTO_INCREMENT	l'identifiant de la personne
firstname	VARCHAR(64)	NOT NULL	Le prénom de la personne
lastname	VARCHAR(64)	NOT NULL	Le nom de la personne
description	VARCHAR(255)		La description de la personne sur son profil
adress	TEXT	NOT NULL	L'adresse de personne
localisation	TEXT	NOT NULL	La position exact de l'utilisateur à son inscription
email	VARCHAR(64)	NOT NULL	L'email de la personne
password	VARCHAR(64)	NOT NULL	Le mot de passe de la personne
thumbnail	TEXT		La photo de profil de la personne
created_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	La date de création du profil
updated_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	La date de dernière modification du profil

Adverts (Annonces)

Champ	Type	Spécificités	Description
id	INT	PRIMARY KEY, UNSIGNED, NOT NULL, AUTO_INCREMENT	l'identifiant de l'annonce
title	VARCHAR(64)	NOT NULL	Titre de l'Annonce
content	TEXT	NOT NULL	Contenu de l'annonce
price	SMALLINT	NOT NULL	Prix de l'objet
user_id	INT	PRIMARY KEY, UNSIGNED, NOT NULL, AUTO_INCREMENT	L'identifiant de la personne qui a créée l'annonce
tag_id	INT	PRIMARY KEY, UNSIGNED, NOT NULL, AUTO_INCREMENT	L'identifiant de la catégorie
created_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Date de la création de l'annonce
updated_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Date de la dernière modification

Relations :

- **Users (Utilisateurs) - Adverts (Annonces)**
 - Chaque annonce est liée à un utilisateur.
 - La table "Adverts" contient une clé étrangère "UserID" permettant de relier chaque annonce à un utilisateur.

Chaque champ est caractérisé par diverses propriétés, telles que son type (numérique, texte, booléen, valeur calculée), ses spécificités (clé primaire, auto-incrémentée, valeur unique, non nulle), une description et des commentaires pertinents.

Voici la séquence SQL qui définit la structure de la base de données, il définit plusieurs tables, y compris les utilisateurs, les annonces, les messages, les tags, les conversations, et d'autres entités, avec des colonnes spécifiques et des contraintes de clé primaire et étrangère.

```

1 BEGIN;
2 DROP TABLE IF EXISTS "user",
3   "like",
4   "post",
5   "message",
6   "favourite",
7   "advert",
8   "advert_has_image",
9   "tag",
10  "conversation";
11
12 CREATE TABLE "user" (
13   id INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
14   firstname VARCHAR(64) NOT NULL,
15   lastname VARCHAR(64) NOT NULL,
16   description VARCHAR(255),
17   address TEXT NOT NULL,
18   city TEXT NOT NULL,
19   longitude TEXT NOT NULL,
20   latitude TEXT NOT NULL,
21   email VARCHAR(64) NOT NULL,
22   password VARCHAR(64) NOT NULL,
23   thumbnail TEXT,
24   banner TEXT DEFAULT 'http://localhost:3000/images/default-banner-picture.png',
25   slug TEXT,
26   created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
27   updated_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
28 );
29 CREATE TABLE "tag" (
30   id INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
31   name VARCHAR(64) NOT NULL,
32   created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
33   updated_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
34 );
35 CREATE TABLE "advert" (
36   id INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
37   title VARCHAR(64) NOT NULL,
38   content TEXT NOT NULL,
39   price SMALLINT NOT NULL,
40   user_id INTEGER NOT NULL,
41   tag_id INTEGER NOT NULL,
42   slug TEXT,
43   created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
44   updated_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
45   FOREIGN KEY (user_id) REFERENCES "user"(id) ON DELETE CASCADE,
46   FOREIGN KEY (tag_id) REFERENCES "tag"(id)
47 );
48 CREATE TABLE "advert_has_image" (
49   id INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
50   advert_id INTEGER NOT NULL,
51   thumbnail TEXT NOT NULL,
52   created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
53   updated_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
54   FOREIGN KEY (advert_id) REFERENCES "advert"(id) ON DELETE CASCADE
55 );
56 CREATE TABLE "post" (
57   id INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
58   content TEXT NOT NULL,
59   thumbnail TEXT,
60   reply_to INTEGER REFERENCES "post"(id) ON DELETE CASCADE,
61   user_id INTEGER NOT NULL,
62   created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
63   updated_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
64   FOREIGN KEY (user_id) REFERENCES "user"(id) ON DELETE CASCADE
65 );
66
67 CREATE TABLE "conversation" (
68   id INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
69   user1 INTEGER NOT NULL,
70   user2 INTEGER NOT NULL,
71   created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
72   updated_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
73 );
74 CREATE TABLE "message" (
75   id INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
76   content TEXT NOT NULL,
77   sender INTEGER NOT NULL,
78   conversation_id INTEGER NOT NULL,
79   created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
80   updated_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
81   FOREIGN KEY (sender) REFERENCES "user"(id) ON DELETE CASCADE,
82   FOREIGN KEY (conversation_id) REFERENCES "conversation"(id) ON DELETE CASCADE
83 );
84 CREATE TABLE "favourite" (
85   id INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
86   advert_id INTEGER NOT NULL,
87   user_id INTEGER NOT NULL,
88   created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
89   updated_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
90   FOREIGN KEY (advert_id) REFERENCES "advert"(id) ON DELETE CASCADE,
91   FOREIGN KEY (user_id) REFERENCES "user"(id) ON DELETE CASCADE
92 );
93 CREATE TABLE "like" (
94   id INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
95   post_id INTEGER NOT NULL,
96   user_id INTEGER NOT NULL,
97   created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
98   updated_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
99   FOREIGN KEY (post_id) REFERENCES "post"(id) ON DELETE CASCADE,
100  FOREIGN KEY (user_id) REFERENCES "user"(id) ON DELETE CASCADE
101 );
102
103 COMMIT ;

```

Egalement dans le package.json nous avons fait ces scripts pour plus facilement manipuler la base de données:

```

1 "scripts": {
2   "test": "echo \"Error: no test specified\" && exit 1",
3   "start": "node index.js",
4   "dev": "nodemon index.js",
5   "db:create": "psql -U tok -d tok -f ./data/create_db.sql",
6   "db:populate": "psql -U tok -d tok -f ./data/data.sql",
7   "db:reset": "npm run db:create && npm run db:populate",
8   "db:connect": "psql -U tok -d tok"

```

E) Sécurité

La sécurité de l'application Tok-Tok est renforcée grâce à plusieurs mesures:

-Tout d'abord, elle utilise le module **bcrypt** pour hacher et sécuriser les mots de passe des utilisateurs.

-De plus, l'application vérifie la validité des adresses e-mail des utilisateurs grâce à la dépendance **email-validator**.

-L'application intègre une couche de sécurité en utilisant des middleware, notamment le middleware **verifyJWT** qui vérifie les jetons JWT (JSON Web Tokens) pour l'authentification des utilisateurs sur les routes nécessitant une authentification. Il assure que seuls les utilisateurs authentifiés ont accès à certaines fonctionnalités de l'application.

-Les données entrées par les utilisateurs sont également protégées grâce au middleware **sanitize**, qui filtre et nettoie les données en utilisant la bibliothèque sanitize-html. Cette mesure aide à prévenir les attaques de type **XSS** (Cross-Site Scripting) en garantissant que les données de l'utilisateur ne contiennent pas de code malveillant.

-L'application utilise également des dépendances telles que **cors** pour gérer les politiques de sécurité des requêtes HTTP.

-En ce qui concerne la génération d'identifiants uniques et de noms de fichiers sécurisés, l'application utilise **UUID**, contribuant ainsi à renforcer la sécurité en évitant les collisions et en garantissant l'unicité des identifiants et des fichiers.

-Le fichier authController montre la mise en place d'une gestion des erreurs pour des cas spécifiques, renvoyant des réponses appropriées pour informer les utilisateurs en cas d'erreurs.

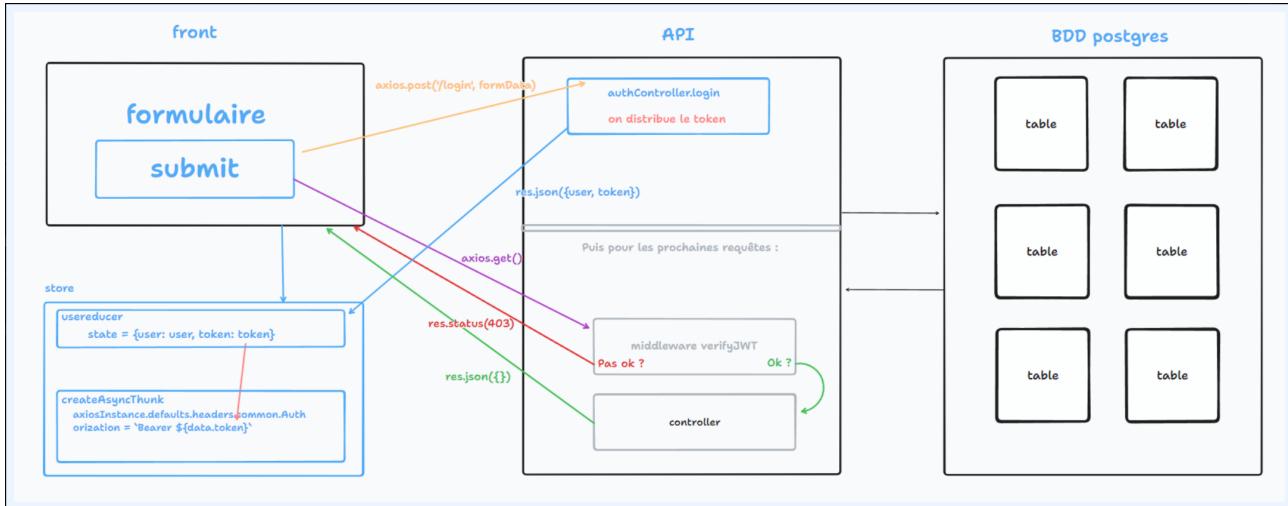
-Enfin comme dit plus haut, le dossier **.env** a été mis dans le fichier **.gitignore** pour protéger les données sensibles, personnaliser la configuration de l'application et simplifier la gestion des paramètres de configuration, tout en facilitant la collaboration.



```
● ● ●
1 PORT = 3000
2 PG_URL = 'postgres://tok:tok@localhost:5432/tok'
3 SALT_ROUNDS=10
4 SECRETTOKEN='thisIsASecretToPutInDotEnv'
5 EXPIREDATETOKEN= '24h'
```

Ces efforts combinés contribuent à la robustesse de la sécurité de l'application, garantissant ainsi la protection des données et des utilisateurs.

F) Circulation de la données



La partie front-end interagit avec une **API back-end** pour gérer l'authentification des utilisateurs et accéder aux données stockées dans **la base de donnée PostgreSQL**.

Pour expliquer la circulation des données au sein de notre application nous allons prendre l'exemple de l'authentification de l'utilisateur et ainsi lui permettre oui ou non d'accéder et de naviguer sur le site:

1) Front-end: L'utilisateur interact avec votre application front-end. Lorsqu'un utilisateur soumet un formulaire de connexion (login), les données du formulaire (par exemple, nom d'utilisateur et mot de passe) sont collectées.

2) Requête vers l'API: L'application front-end envoie une requête POST à l'API en utilisant Axios. Cette requête est destinée à la route /login, qui est géré par le contrôleur authController.login.

3) Authentication: Le contrôleur authController.login reçoit les données du formulaire de connexion. Il vérifie l'authenticité de l'utilisateur en vérifiant le nom d'utilisateur et le mot de passe. Si l'authentification réussit, le contrôleur génère un jeton JWT (JSON Web Token) qui contient des informations d'authentification, telles qu'un identifiant d'utilisateur (par exemple, `userId`), puis renvoie ce jeton sous forme de réponse JSON (par exemple, { user, token }).

4) Réponse au Front-end: Le front-end reçoit cette réponse JSON et peut extraire le jeton et les données utilisateur (par exemple, l'identifiant de l'utilisateur) de la réponse.

5) Stockage dans le Store: Les données extraites de la réponse (comme le jeton) sont stockées dans le Store de l'application, dans un Redux Store. Ces données sont accessibles à d'autres parties de l'application.

6) Requêtes ultérieures: Lorsque l'utilisateur effectue d'autres actions, telles que demander des informations depuis l'API (par exemple, axios.get()), le jeton JWT est ajouté aux en-têtes de ces requêtes. Cela permet à l'API de vérifier l'authenticité de l'utilisateur sur chaque requête.

7) Middleware de vérification JWT: Avant que ces requêtes n'atteignent les contrôleurs spécifiques, elles passent par un middleware appelé verifyJWT. Ce middleware vérifie la validité du jeton JWT. Si le jeton est valide, la demande est autorisée à accéder à la ressource demandée. Sinon, une réponse avec un statut 403 (interdit) est renvoyée.

8) Contrôleur de l'API: Si la vérification JWT est réussie, la demande atteint le contrôleur approprié qui gère la ressource demandée. Le contrôleur peut accéder à la base de données PostgreSQL pour obtenir les données requises. Une réponse appropriée est renvoyée au front-end.

En résumé, notre application front-end envoie des demandes d'authentification et d'accès aux données à l'API back-end.

L'API gère l'authentification, vérifie l'authenticité de l'utilisateur en utilisant un jeton JWT, puis renvoie les données demandées. Le middleware verifyJWT assure que seules les demandes authentifiées sont autorisées à accéder aux ressources protégées.

Cela permet de sécuriser notre application et de contrôler l'accès aux données stockées dans la base de données PostgreSQL.

VI. Gestion de Projet

A) Présentation de l'équipe

L'attribution des rôles au sein de notre équipe a été le résultat d'une discussion collaborative. Nous avons pris en compte nos préférences personnelles tout en évaluant nos compétences et points forts afin de garantir une répartition équilibrée des rôles. Les membres de l'équipe ont été assignés à des postes qui correspondent le mieux à leurs compétences et à l'orientation du projet. Voici la répartition des rôles et les détails associés :

John :

- **Product Owner**, donne son avis sur la priorité des tâches lors des sprints plannings et s'assure de la cohérence du projet.
- **Lead Dev Front**, supervise la partie front du développement, y compris les choix techniques si nécessaire.

Chloé BATILLET :

- **Scrum Master**, supervise l'avancement global du projet et des tâches effectuées, gère le Trello avec les stories.
- **Développeur Full Stack**, capable d'intervenir sur l'ensemble du projet, tant en front-end qu'en back-end.

Maxime MALANDAIN :

- **Lead Dev Back**, supervise la partie back du développement, y compris les choix techniques si nécessaire.
- **Back Dev**, impliqué dans le développement back-end du projet.

Tom ESTEBAN :

- **Référent Git**, garant du bon fonctionnement du versionnage avec Git vérifie les PR et merge, gère les conflits ect...
- **Dev Front**, participe au développement front-end du projet.

B) Méthodes et outils de travail

1) Méthode de travail

La méthode de travail adoptée pour ce projet repose sur les principes de **Scrum**, une approche **agile** de gestion de projet qui favorise la flexibilité et la collaboration au sein de l'équipe. Notre équipe a initié ce processus en organisant des réunions régulières pour définir les objectifs et les priorités du projet.

Nous avons choisi de travailler ensemble de 9h30 à 17h en début de projet, et, selon les disponibilités de chacun, nous avons également consacré du temps en soirée pour maintenir notre progression. **Chaque matin, nous avons organisé des réunions de revue pour évaluer les progrès, discuter des défis rencontrés et fixer les objectifs pour la journée.**

Au fur et à mesure de l'avancement, une fois le cahier des charges finalisé et la phase de conception entamée, notre équipe a adopté une approche collaborative. Les trois développeurs front-end ont travaillé en étroite coordination sur leurs parties respectives, communiquant à travers des discussions vocales, tandis que le développeur back-end se consacrait au développement du back end, dans un dépôt distinct et en travaillant en autonomie.

Néanmoins, il restait toujours rapidement accessible en cas de besoin.

Au milieu du projet, un membre de l'équipe a démontré sa polyvalence en intervenant à la fois en front-end et en back-end. Vers cette période, il est venu en aide au développeur en charge de la partie back-end, offrant une assistance précieuse pour équilibrer la répartition de la charge de travail au sein de l'équipe.

Enfin à la fin du projet, notre collaboration a été renforcée par l'adoption du **pair programming**. Cette approche nous a permis de détecter et résoudre les dernières erreurs, d'optimiser notre code et d'améliorer notre compréhension collective du projet. Le pair programming a encouragé des échanges en temps réel, accélérant ainsi la résolution des problèmes et renforçant notre cohésion d'équipe.

2) Outils de travail

Les outils que nous avons utilisés tout au long de ce projet ont grandement contribué à notre efficacité et à la gestion collaborative de notre travail. L'élément central de notre environnement de développement a été **Visual Studio Code (VSCode)**, un éditeur de code très polyvalent et extensible. Il nous a permis de travailler sur nos fichiers de code avec facilité, en intégrant des extensions utiles pour différentes technologies, facilitant ainsi le développement.

Pour la documentation de notre projet, nous avons utilisé **Notion**, une plateforme de prise de notes et de gestion de projets en ligne. Notion a été essentiel pour créer le **cahier des charges** et le **carnet de bord du projet**. Nous y avons centralisé toutes nos informations, nos idées, et nos avancées. Cela nous a permis de garder une trace claire de nos objectifs, des tâches à accomplir, et de rester organisés tout au long du projet.

Pour la communication au sein de l'équipe, **Discord** et **Slack** ont été nos principales plateformes. Ces outils de messagerie ont facilité les échanges entre les membres de l'équipe, nous permettant de discuter des problèmes rencontrés, de poser des questions, et de partager des informations rapidement. De plus, les canaux de discussion spécifiques dans Discord nous ont permis de séparer les discussions sur des sujets particuliers, améliorant ainsi la clarté des échanges.

Lors du développement, l'outil **Insomnia** a été notre choix pour tester les API que nous avons créées. Cet outil nous a permis de simuler les requêtes HTTP et d'inspecter les réponses, garantissant ainsi le bon fonctionnement de nos API.

Enfin, **GitHub** a été notre plateforme pour l'hébergement du code source du projet. Nous avons utilisé GitHub pour gérer les **issues** et les **projects** liés au projet. Les issues ont été essentielles pour suivre les bogues, les améliorations, et les tâches à réaliser. Les projects nous ont permis de planifier, organiser, et suivre les fonctionnalités et les objectifs à atteindre.

fix DB errors
chloebatillet committed on Jul 25
added rows for localization in DB, user obj in res.json, categories r...
chloebatillet committed on Jul 25
merge-API to Dev
Maxime-Malandain committed on Jul 25
✓ add token Creation on signup and ✓ add examples in images/videos d...
Maxime-Malandain committed on Jul 25
🏗 directory public, 🗃 the code, Changing the language for French er...
Maxime-Malandain committed on Jul 25
✓ finish config multer and — date-fns
Maxime-Malandain committed on Jul 25

Commits on Jul 24, 2023

▶ refactor in progress Multer (Date-UUID)
Maxime-Malandain committed on Jul 24
Merge pull request #31 from O-clock-Uranium/Delete-On-cascade
Maxime-Malandain committed on Jul 24
Ajout ON DELETE CASCADE,
Maxime-Malandain committed on Jul 24
Merge pull request #30 from O-clock-Uranium/Multer_V2
chloebatillet committed on Jul 24

Todo (5)	In Progress (4)	Done (21)
This item hasn't been started	This is actively being worked on	This has been completed
<p>Draft Post Controller/ GEt All Post récupérer les users info sur l'association replies</p> <p>Draft Notification</p> <p>Draft Messages</p> <p>Draft ... Ajout fonctionnalité supprimé une image</p> <p>Draft Améliorer Multer pour bien contrôlés les sources et éviter une "attaque"</p>	<p>projet-tok-tok-back #29 Voir pour préparer la FUSION!!!!</p> <p>projet-tok-tok-back #22 Socket.io</p> <p>Draft intégrer l'api adresse en front</p> <p>projet-tok-tok-back #28 API geolocalisation</p>	<p>Draft adapter les requêtes sequelized op.between pour la latitude et getAll posts et adverts</p> <p>Draft ajouter les champs nécessaires localisation</p> <p>✓ projet-tok-tok-back #35 slug sur les annonces et profil</p> <p>✓ projet-tok-tok-back #3 models sequelize</p> <p>✓ projet-tok-tok-back #4 mail validator</p>

C) Sprints

Sprint 0:

Au cours du Sprint 0, notre principal objectif était de poser les bases solides du projet. Cela signifiait que nous ne nous sommes pas encore lancés dans la phase de codage, mais plutôt dans la préparation et la planification du travail à venir. Les documents clés que nous avons produits comprennent :

1. Le cahier des charges (CDC) :
 - Présentation du projet
 - Définition des besoins et objectifs du projet
 - Spécifications fonctionnelles, y compris le MVP (Minimum Viable Product)
 - Listes des fonctionnalités, des évolutions potentielles, des technologies utilisées, de la cible du projet, des navigateurs compatibles, et de l'arborescence de l'application.
 - Listes des routes prévues et des User stories.

Note: Les routes prévues en front et back sont disponibles dans l'index.

2. Rôles individuels :
 - Attribution des rôles aux membres de l'équipe, notamment le Product Owner, le Scrum Master, les Lead Dev Back et Front et le Référent Git.
3. Documents relatifs à la base de données :
 - Modèle Conceptuel de Données (MCD) et le Modèle Logique de Données (MLD).
 - Dictionnaire de données.
4. Wireframes :
 - Représentation des éléments des pages en version mobile et desktop.

5. Documents de veille :

- Chaque membre a noté ses recherches, mots-clés, liens vers les ressources, bugs rencontrés et solutions apportées.

6. Carnets de bord :

- Deux types de carnets de bord, un journal d'équipe tenu par le Scrum Master et des journaux personnels pour chaque membre. Ils servent à suivre la progression quotidienne du projet.

7. Outil de gestion et suivi de projet :

- Nous avons utilisé Notion en outil de gestion de projet mais étant limité en block la partie Back est passé sur Github Project.

Sprint 1 & 2:

Les Sprints 1 et 2 ont marqué le début de la phase de codage. Nous avons commencé à mettre en œuvre les fonctionnalités de notre projet tout en restant flexibles pour ajuster nos objectifs au fur et à mesure de l'avancée du projet.

Pendant ces sprints, nous avons principalement travaillé sur le développement et la mise en place des fonctionnalités prévues.

Nous avons également continué à réévaluer nos priorités, ajoutant ou supprimant des fonctionnalités en fonction de l'évolution de nos besoins. Les documents préparés lors du Sprint 0 ont été mis à jour pour refléter les changements dans le projet.

Sprint 3 :

Au cours du Sprint 3, notre objectif était de finaliser le projet et de nous assurer de sa qualité. Nous avons cessé de développer de nouvelles fonctionnalités et nous nous sommes concentrés sur les tâches suivantes :

1. Terminer les fonctionnalités en cours :

- Nous avons travaillé sur les fonctionnalités qui n'étaient pas encore terminées pour les amener à maturité.

2. Tests et correction de bugs :

- Nous avons effectué des tests approfondis pour identifier et corriger les derniers bugs et erreurs.

3. Refactoring :

- Nous avons révisé et nettoyé notre code existant, en cherchant à le rendre plus homogène et optimisé.

4. Préparation pour la démo :

- Nous nous sommes entraînés et avons préparé la démo finale du projet, en veillant à ce qu'elle se déroule sans accroc.

D) Difficultés rencontrées

Pendant le développement de notre projet, nous avons rencontré plusieurs difficultés et défis auxquels nous avons dû faire face pour mener à bien notre mission. Voici quelques-unes des principales difficultés que nous avons rencontrées :

1. **Envisager un MVP trop ambitieux** : L'une de nos premières erreurs a été de prévoir un MVP (Minimum Viable Product) qui incluait un grand nombre de fonctionnalités complexes, telles qu'un système de paiement, des messages privés et des notifications. En rétrospective, nous avons réalisé que cela a contribué à des attentes démesurées en matière de fonctionnalités et de délais, ce qui a rendu la planification initiale difficile à suivre.
2. **Transition vers le développement Back-End** : Au départ, notre équipe était composée de membres spécialisés en développement Front-End. Lorsque la nécessité de développer une partie Back-End s'est présentée, j'ai volontairement accepté le défi de m'occuper de cette tâche. Bien que ce fût une opportunité intéressante, cela a entraîné une période d'adaptation et d'apprentissage, car je devais me remettre rapidement dans l'univers du Back-End, notamment en ce qui concerne la gestion de la base de données et la mise en place des API.
3. **Configuration du système de chat instantané avec Socket.IO** : Nous avions initialement prévu d'intégrer un système de chat instantané dans notre application en utilisant Socket.IO, une technologie populaire pour les applications temps réel. Malheureusement, malgré nos nombreuses tentatives pour configurer correctement Socket.IO, nous avons rencontré des problèmes techniques qui ont rendu cette fonctionnalité difficile à mettre en place. Après de nombreux essais, nous avons finalement pris la décision de la mettre de côté pour nous concentrer sur d'autres priorités.

VII. Réalisations personnelles

A) authController

Ce code est conçu pour gérer l'authentification d'un utilisateur dans l'application, en vérifiant leur identité lors de l'inscription et de la connexion, et en générant un token d'authentification pour les sessions ultérieures.

Inscription (signup) :

Lorsqu'un utilisateur tente de s'inscrire, le code vérifie d'abord que toutes les informations nécessaires (prénom, nom, adresse, ville, longitude, latitude, email, mot de passe, confirmation) ont été fournies.

Si certaines informations manquent, une réponse d'erreur est renvoyée. Le code vérifie également que les mots de passe correspondent et que le **format de l'email est valide**. Ensuite, il vérifie si l'email fourni est déjà associé à un compte existant. Si l'email est déjà utilisé, une autre réponse d'erreur est renvoyée. Enfin, le mot de passe est **haché (sécurisé) à l'aide de bcrypt**, et un utilisateur est créé dans la base de données avec toutes les informations nécessaires.

Un token d'authentification est généré à l'aide de jsonwebtoken, et le nouvel utilisateur est renvoyé en réponse, y compris le token.

Connexion (login) :

Lorsqu'un utilisateur tente de se connecter, le code vérifie d'abord si l'email fourni existe dans la base de données. Si l'email n'est pas trouvé ou si le mot de passe ne correspond pas à l'email, une réponse d'erreur est renvoyée. Si l'email est trouvé et que le mot de passe correspond, un nouveau token d'authentification est généré à l'aide de **jsonwebtoken**, et l'utilisateur est renvoyé en réponse, y compris le token.

Autres informations :

- Les fonctions signup et login sont **asynchrones (async)** pour gérer des opérations asynchrones telles que les requêtes à la base de données.
- Les erreurs sont gérées avec des blocs **try...catch** pour renvoyer des réponses d'erreur appropriées en cas de problème.
- Les tokens d'authentification sont générés avec une clé secrète (stockée dans la variable **process.env.SECRET TOKEN**) et une durée de validité définie dans **process.env.EXPIREDATETOKEN**.
- Des informations sur l'utilisateur (prénom, nom, description, adresse, ville, etc.) sont renvoyées dans l'objet de réponse, ce qui permet au front-end de savoir quel utilisateur est actuellement connecté.

```

1 const bcrypt = require("bcrypt");
2 const validator = require("email-validator");
3 const { v4: uuidv4 } = require("uuid");
4 //const zxcvbn = require("zxcvbn");
5
6 const jwt = require("jsonwebtoken");
7
8 const { User } = require("../models/index");
9
10 const authController = {
11   signup: async (req, res) => {
12     try {
13       const {
14         firstname,
15         lastname,
16         address,
17         city,
18         longitude,
19         latitude,
20         email,
21         password,
22         confirmation,
23       } = req.body;
24
25     //! NOUVEAU
26     if (
27       !firstname ||
28       !lastname ||
29       !address ||
30       !city ||
31       !longitude ||
32       !latitude ||
33       !email ||
34       !password ||
35       !confirmation
36     ) {
37       res
38         .status(400)
39         .json({ error: "Tous les champs doivent être renseignés" });
40       return;
41     }
42
43     if (password !== confirmation) {
44       res
45         .status(400)
46         .json({ error: "Les deux mots de passe ne correspondent pas" });
47       return;
48     }
49
50     if (!validator.validate(email)) {
51       res.status(400).json({ error: "Le format de l'email est invalide" });
52       return;
53     }
54
55     const existingEmail = await User.findOne({
56       where: {
57         email: email,
58       },
59     });
60
61     if (existingEmail) {
62       res
63         .status(400)
64         .json({ error: "Cet email est déjà associé à un compte" });
65       return;
66     }
67
68    /* on le retire tant qu'on ne sait pas à quoi correspond le score
69    // (pas cool pour l'ux si pas d'indications), voire on fait notre propre fonction
70    // const passwordStrength = zxcvbn(password);
71    // if (passwordStrength.score < 1) {
72    //   res
73    //     .status(400)
74    //     .json({ errorMessage: "Le mot de passe est trop faible." });
75    //   return;
76    // }
77
78    const saltRounds = parseInt(process.env.SALT_ROUNDS);
79    const hashedPassword = await bcrypt.hash(password, saltRounds);
80
81    const user = await User.create({
82      firstname,
83      lastname,
84      description: "",
85      address,
86      city,
87      longitude,
88      latitude,
89      email: email.toLowerCase(),
90      password: hashedPassword,
91      slug: `${firstname.toLowerCase()}-${lastname.toLowerCase()}-${uuidv4()}`,
92      thumbnail: `${req.protocol}://${req.get(
93        "host"
94      )}/images/default-profile-picture.png`,
95    });
96    //await user.save();
97    console.log(user);
98  }

```

```

1 const token = jwt.sign({ userId: user.id }, process.env.SECRETKEY, {
2   expiresIn: process.env.EXPIREDATETOKEN,
3 });
4 console.log(token);
5
6 const userObj = {
7   id: user.id,
8   firstname: user.firstname,
9   lastname: user.lastname,
10  description: user.description,
11  address: user.address,
12  city: user.city,
13  longitude: user.longitude,
14  latitude: user.latitude,
15  thumbnail: user.thumbnail,
16  slug: user.slug,
17  email: user.email,
18  banner: user.banner,
19 };
20
21 res.status(201).json({
22   message: "Compte créé",
23   auth: true,
24   token: token,
25   user: userObj,
26 });
27 } catch (error) {
28   console.log(error);
29   res.status(500).json({error: "Erreur Serveur !"});
30 }
31
32 },
33
34 login: async (req, res) => {
35   try {
36     const { email, password } = req.body;
37
38     const user = await User.findOne({
39       where: { email: email.toLowerCase() },
40     });
41
42     const isMatching = await bcrypt.compare(password, user.password);
43
44     if (!user || !isMatching) {
45       return res
46         .status(400)
47         .json({ error: "Mauvais couple email/password" });
48     }
49
50    // A chaque connexion, le user reçoit un token que l'on mettra
51    // en tête des requêtes http sur les routes où il faut être loggué/authentifié
52    const token = jwt.sign({ userId: user.id }, process.env.SECRETKEY, {
53      expiresIn: process.env.EXPIREDATETOKEN,
54    });
55
56    const userObj = {
57      id: user.id,
58      firstname: user.firstname,
59      lastname: user.lastname,
60      description: user.description,
61      address: user.address,
62      city: user.city,
63      longitude: user.longitude,
64      latitude: user.latitude,
65      thumbnail: user.thumbnail,
66      slug: user.slug,
67      email: user.email,
68      banner: user.banner,
69    };
70
71    console.log(token);
72    res.json({ auth: true, token: token, user: userObj });
73  } catch (error) {
74    console.log(error);
75    res.status(500).json({error: "Erreur Serveur !"});
76  }
77 },
78 }
79
80 module.exports = authController;
81

```

B) verifyAuthMiddleware et une partie du routeur

Ce middleware est utilisé pour garantir que seuls les utilisateurs authentifiés, munis de **jetons JWT valides, ont accès aux routes protégées**.

- Si le jeton n'est pas valide ou si l'utilisateur correspondant n'est pas trouvé dans la base de données, une réponse d'erreur est renvoyée.

- Si le jeton est valide et correspond à un utilisateur existant, l'objet user est attaché à la requête pour une utilisation ultérieure dans le traitement de la route.

1. Importations :

- Le middleware utilise le module **jsonwebtoken** pour vérifier la validité des jetons d'authentification JWT.

- Il importe également le modèle User pour effectuer des recherches sur les utilisateurs dans la base de données.

2. Middleware verifyJWT:

- Le middleware est défini comme une fonction asynchrone qui prend les paramètres req, res, et next, représentant respectivement la requête, la réponse et la fonction middleware suivante.

- Il commence par extraire le jeton JWT de l'en-tête de la requête. Le jeton est généralement transmis dans l'en-tête d'autorisation au format "Bearer <token>".

La ligne const jwtToken = token.split(" ")[1]; permet d'extraire la partie du jeton après "Bearer".

- Ensuite, le middleware utilise la fonction jwt.verify pour vérifier la validité du jeton. Il vérifie si le jeton est valide en utilisant la **clé secrète process.env.SECRETOKEN**.

- Si le jeton est invalide, le middleware renvoie une réponse avec le **code d'état 403** (Interdit) et un message d'erreur indiquant que le jeton n'est pas valide.

- Si le jeton est valide, le middleware utilise l'identifiant de l'utilisateur (userId) extrait du jeton pour rechercher l'utilisateur correspondant dans la base de données à l'aide de User.findOne.

- Si aucun utilisateur correspondant n'est trouvé, le middleware renvoie à nouveau une réponse avec le code d'état 403 et un message d'erreur indiquant que le jeton n'est pas valide.

- Si un utilisateur correspondant est trouvé, le middleware ajoute l'objet user à la requête req, ce qui permet aux routes ultérieures d'accéder aux informations de l'utilisateur authentifié.

- Enfin, le middleware appelle la fonction next(), ce qui signifie qu'il a terminé sa vérification et que le traitement de la requête peut se poursuivre normalement par d'autres middlewares ou les routes finales.

3. Exportation du middleware :

- À la fin du fichier, le middleware verifyJWT est exporté pour être utilisé dans le router où l'authentification est requise.

```

● ○ ●
1 const jwt = require("jsonwebtoken");
2 const { User } = require("../models");
3
4 const verifyJWT = async (req, res, next) => {
5
6   const token = req.headers.authorization;
7
8   if (!token)
9     return res
10    .status(401)
11    .json({ auth: false, error: "You are not authorized to see this!" });
12
13 const jwtToken = token.split(" ")[1]; /* -> Extraction du JWT du format "Bearer <token>" */
14
15 jwt.verify(jwtToken, process.env.SECRETOKEN, async (err, decoded) => {
16   if (err)
17     return res.status(403).json({ auth: false, error: "Your token is not valid" });
18
19   const user = await User.findOne({ where: { id: decoded.userId } });
20
21   if (!user)
22     return res.status(403).json({ auth: false, error: "Your token is not valid" });
23
24   req.user = user;
25
26   next();
27 });
28 };
29
30 module.exports = verifyJWT;
31

```

```

● ○ ●
1 const { Router } = require("express");
2 const sanitize = require("./middlewares/sanitize");
3 const verifyAuthMiddleware = require("./middlewares/verifyAuthMiddleware");
4 const multer = require("./middlewares/multerMiddleware");
5 const { advertController, authController, categoriesController, favouriteController,
6   likeController, messageController, postController, UserController } = require("./controllers/index");
7
8 const router = Router();
9
10 router.post("*", sanitize);
11 router.patch("*", sanitize);
12
13 /* login/signup -----*/
14 router.post("/login", authController.login);
15 router.post("/signup", authController.signup);
16
17 /* Posts -----*/
18 router.get("/posts", verifyAuthMiddleware, postController.getAll);
19 router.get("/post/:id", verifyAuthMiddleware, postController.getOne);
20 router.post(
21   "/posts",
22   verifyAuthMiddleware,
23   multer.single("thumbnail"),
24   postController.createPost
25 );
26 router.patch(
27   "/posts/:id",
28   verifyAuthMiddleware,
29   multer.single("thumbnail"),
30   postController.update
31 );
32 router.delete("/posts/:id", verifyAuthMiddleware, postController.remove);
33

```

C) multerMiddleware

Ce fichier est une configuration de middleware utilisée pour gérer les fichiers téléchargés dans l'application par les utilisateurs authentifiés. Il vérifie les types de fichiers autorisés, détermine le dossier de stockage approprié et génère des noms de fichiers uniques, garantissant ainsi un téléchargement de fichiers sécurisé et organisé.

1. Importations :

Le code commence par importer deux modules essentiels :

- **multer**: C'est une bibliothèque qui facilite la gestion des fichiers téléchargés dans une application Express.

- **uuidv4**: Il s'agit d'un module qui génère des identifiants universels uniques.

2. Types de fichiers autorisés :

- Deux objets sont définis pour spécifier les types de fichiers autorisés :

- **MAGE_MIME_TYPES**: Cet objet associe les types MIME des images aux extensions de fichier correspondantes (par exemple, "image/jpeg" est associé à "jpg").

- **VIDEO_MIME_TYPES**: Cet objet fait la même chose pour les fichiers vidéo.

3. Configuration du stockage des fichiers :

- La variable **storage** est définie comme une configuration pour le stockage des fichiers téléchargés. Cette configuration utilise **multer.diskStorage**, ce qui signifie que les fichiers sont stockés sur le disque du serveur.

- La fonction **destination** est utilisée pour déterminer où les fichiers téléchargés seront enregistrés. Elle prend en compte le type de fichier en fonction de son type MIME. Si le fichier est une image, il sera stocké dans le dossier "public/images". Si c'est une vidéo, elle sera stockée dans "public/videos". Si le type de fichier n'est ni une image ni une vidéo, une erreur est renvoyée pour indiquer un format de fichier invalide.

- La fonction **filename** génère un nom de fichier unique en utilisant l'identifiant UUID et l'extension de fichier en fonction de son type MIME. Cela garantit que les noms de fichiers sont uniques et évite les conflits.

4. Exportation du middleware :

- À la fin du code, le middleware est exporté. Il est configuré pour utiliser le stockage (storage) défini précédemment. Ce middleware est utilisé sur les routes où des fichiers sont téléchargés.



```
1 const multer = require("multer");
2 const { v4: uuidv4 } = require("uuid");
3
4 // Bibliothèque des types de fichiers qu'un utilisateur peut nous envoyer
5 // Pour les images
6 const IMAGE_MIME_TYPES = {
7   "image/jpeg": "jpg",
8   "image/png": "png",
9 };
10
11 // Pour les vidéos
12 const VIDEO_MIME_TYPES = {
13   "video/mp4": "mp4",
14   "video/quicktime": "mov",
15 };
16
17 const storage = multer.diskStorage({
18   destination: (_, file, callback) => {
19     if (IMAGE_MIME_TYPES[file.mimetype]) {
20       callback(null, "public/images"); // Destination pour les images
21     } else if (VIDEO_MIME_TYPES[file.mimetype]) {
22       callback(null, "public/videos"); // Destination pour les vidéos
23     } else {
24       callback(new Error("Format de fichier invalide"));
25     }
26   },
27   filename: (_, file, callback) => {
28     const extension =
29       IMAGE_MIME_TYPES[file.mimetype] || VIDEO_MIME_TYPES[file.mimetype];
30     const uniqueFilename = `${uuidv4()}.${extension}`;
31     callback(null, uniqueFilename);
32   },
33 });
34
35 // Dans l'exports, le premier storage est pour définir les settings et l'autre pour configurer l'instance du middleware multer.
36 module.exports = multer({ storage: storage });
37 /* Sur la route : multer +
38 /* .single("thumbnail") quand on reçoit un fichier -> retourne obj
39 /* .array() quand on reçoit plusieurs fichiers -> retourne tableau d'obj
40
```

VIII. Jeu d'essai

La fonctionnalité de création de compte a été soumise à un jeu d'essai approfondi pour évaluer son comportement. Les tests se sont concentrés sur le **formulaire d'inscription**, composé de **6 champs obligatoires** à savoir:

Prénom / Nom / Adresse / Mail / Mot de passe / Confirmation

Un test a été effectué pour vérifier que les **champs étaient correctement remplis**, avec l'attente d'un message d'erreur "**Tous les champs doivent être renseignés**" en cas de champ vide.

De plus, **une validation croisée entre les champs Mot de passe et Confirmez votre mot de passe** a été effectuée pour détecter l'erreur "**Les deux mots de passe ne correspondent pas**" en cas de non-correspondance.

Enfin, un test a été réalisé pour s'assurer que le **champ d'e-mail suivait le format attendu**, signalant une erreur "**Le format de l'e-mail est invalide**" en cas de non-conformité.

La **gestion d'un e-mail déjà enregistré** a également été testée, générant un message d'erreur "**Impossible de créer un compte avec ces informations**".

Ces tests garantissent une expérience utilisateur robuste lors de la création de compte.

Champ testé	Action utilisateur	Action attendue	Action réelle	Conclusion
N'importe quel champ	Oublie de remplir un ou plusieurs champ(s)	Message d'erreur: "Tous les champs doivent être renseignés"	Message d'erreur: "Tous les champs doivent être renseignés"	Concluant
Mot de passe / confirmation	Ne sait pas le même mot de passe	Message d'erreur: "Les deux mots de passe ne correspondent pas"	Message d'erreur: "Les deux mots de passe ne correspondent pas"	Concluant
Email	Saisit un email qui n'a pas au minimum cette structure: xxx@xxx.xx	Message d'erreur: "Le format de l'email est invalide"	Message d'erreur: "Le format de l'email est invalide"	Concluant
Email	Saisit un email déjà enregistré	Message d'erreur: "Impossible de créer un compte avec ces informations"	Message d'erreur: "Impossible de créer un compte avec ces informations"	Concluant

IX. Vulnérabilités de sécurité et veille

A) Cookie ou localStorage pour Token JWT

Lors de la définition du cahier des charges, nous avons dû examiner les options disponibles pour l'authentification des utilisateurs, notamment deux solutions pour le stockage du token JWT : dans un cookie ou dans le navigateur (localStorage).

Chacune de ces solutions présente des avantages et des inconvénients, et nous avons dû les évaluer pour décider de l'implémentation la plus adaptée à notre application Tok-Tok.

Avantages et Inconvénients :

- Le stockage du token JWT dans le navigateur, que ce soit via localStorage, est une option idéale pour une application principalement orientée front-end avec des requêtes AJAX. Dans ce scénario, c'est à l'application elle-même d'envoyer systématiquement le token aux serveurs via l'en-tête Authorization au format "Bearer <token>". Cette approche nous protège nativement contre les attaques de type CSRF (Cross-Site Request Forgery) car le jeton n'est pas accessible à l'attaquant, à condition de suivre ce processus pour toutes les requêtes. Cependant, il est important de noter que le jeton étant stocké côté front, il est exposé aux failles XSS (Cross-Site Scripting), car un attaquant pourrait tenter de le récupérer.
- D'un autre côté, le stockage d'un jeton JWT dans un cookie du navigateur peut être plus sécurisé en appliquant des attributs tels que 'httpOnly', 'secure' et 'sameSite=strict' pour le protéger contre les failles XSS, car il n'est pas accessible par JavaScript. Cependant, le cookie étant automatiquement inclus dans chaque requête, il est récupérable à ce moment-là, ce qui le rend vulnérable aux attaques de type CSRF. Malgré cela, il présente l'avantage de réduire le temps d'exploitation disponible pour l'attaquant et nécessite moins de code côté front, car il repose sur le comportement du navigateur.

Notre Choix : Après une évaluation minutieuse, nous avons décidé d'implémenter le stockage du token JWT dans le localStorage du navigateur. Cette décision a été motivée par la sécurité adéquate qu'elle offre pour notre application et sa simplicité de mise en œuvre. Nous avons pris des précautions pour gérer les failles XSS potentielles (via sanitize-html) et considéré que cette solution répondait à nos besoins en matière de sécurité au vu de notre application restant simple.

B) Sanitize sur toutes les routes?

La question s'est posé quand à la sanitization des données utilisateur dans notre application. En pesant les avantages et inconvénients d'une approche globale par rapport à une approche sélective sur certaines routes.

Avantages:

1. Sécurité Améliorée: La principale raison d'appliquer une sanitization globale est d'améliorer la sécurité de votre application. Cela aide à protéger contre les attaques de type XSS (Cross-Site Scripting) en nettoyant les entrées utilisateur potentiellement dangereuses.

2. Prévention des Attaques: En nettoyant toutes les données provenant des utilisateurs, nous évitons la possibilité d'enregistrer ou d'exécuter un code malveillant que les utilisateurs pourraient nous envoyer, que ce soit côté front-end ou côté API.

3. Moins de Risques: Les entrées utilisateur, quelle que soit leur provenance, peuvent potentiellement être exploitées pour des attaques. En nettoyant systématiquement ces données, on réduit considérablement les risques de failles de sécurité.

4. Simplicité: En appliquant la sanitization de manière globale, on simplifie également le processus de développement, car nous n'avons pas besoin de nous soucier de choisir quelles routes nécessitent une sanitization et quelles routes ne le font pas.

Inconvénients:

1. Performance: L'application de la sanitization à toutes les routes peut potentiellement entraîner une légère surcharge en termes de performances, car des opérations de nettoyage sont effectuées même sur des données qui n'ont pas besoin d'être nettoyées.

2. Complexité Inutile: Si la plupart des routes de l'application ne traitent pas de données potentiellement dangereuses, l'application d'une sanitization globale peut sembler excessive et inutile.

Notre Choix: Après une discussion approfondie, la sécurité prime sur les préoccupations de performances.

L'approche recommandée est d'appliquer une sanitization globale pour toutes les routes donc dans notre application nous l'avons passés sur les routes Post et Path.

Cela offre une protection efficace contre les failles de sécurité potentielles, en garantissant que toutes les données utilisateur sont nettoyées. .

X. Recherches à partir de site anglophone extrait et traduction

1) Recherche

Nous nous sommes posé la question, comme mentionné précédemment, de savoir où stocker nos jetons JWT. Pour obtenir des informations sur ce sujet, nous avons effectué une recherche sur Google en utilisant la question suivante : "**local storage or cookie for jwt?**".

Nous avons évalué la pertinence des résultats en fonction de la source, en privilégiant les sites bien établis et fiables, tels que Stack Overflow ou Mozilla. De plus, nous avons pris en compte la date de l'article ou de la question/réponse pour nous assurer qu'elle reste d'actualité et n'est pas deprecated.

Ici nous avons pris comme référence un article de **Dev.to** datant de 2020 qui explique les intérêts de chacune des méthodes.

2) Extrait (*copié coller*)

“Local Storages

Pros: It's convenient.

- It's pure JavaScript and it's convenient. If you don't have a back-end and you're relying on a third-party API, you can't always ask them to set a specific cookie for your site.
- Works with APIs that require you to put your access token in the header like this: `Authorization Bearer ${access_token}`.

Cons: It's vulnerable to XSS attacks.

An XSS attack happens when an attacker can run JavaScript on your website. This means that the attacker can just take the access token that you stored in your `localStorage`.

An XSS attack can happen from a third-party JavaScript code included in your website, like React, Vue, jQuery, Google Analytics, etc. It's almost impossible not to include any third-party libraries in your site.

Cookies

Pros: The cookie is not accessible via JavaScript; hence, it is not as vulnerable to XSS attacks as `localStorage`.

- If you're using `httpOnly` and `secure` cookies, that means your cookies cannot be accessed using JavaScript. This means, even if an attacker can run JS on your site, they can't read your access token from the cookie.
- It's automatically sent in every HTTP request to your server.

Cons: Depending on the use case, you might not be able to store your tokens in the cookies.

- Cookies have a size limit of 4KB. Therefore, if you're using a big JWT Token, storing in the cookie is not an option.
- There are scenarios where you can't share cookies with your API server or the API requires you to put the access token in the Authorization header. In this case, you won't be able to use cookies to store your tokens.”

3) Traduction:

-Pour le Local Storage c'est du JavaScript pur et donc pratique. Si vous n'avez pas de back end et que vous êtes dépendant d'une API tierce, vous ne pouvez pas avoir un cookie spécifique pour votre site.

Cela fonctionne avec les API qui ont besoin que nous mettions le jetons d'accès dans l'en tête comme ceci: `Authorization Bearer ${access_token}`.

Mais c'est vulnérable aux attaques XSS.

Une attaque XSS peut arriver quand un attaquant peut exécuter du code Javascript sur notre site. Cela veut dire que l'attaquant peut juste accéder aux jetons d'accès stocké dans notre localStorage.

Une attaque XSS peut venir d'un code JavaScript d'un tiers inclus dans votre site web, comme React, Vue, jQuery, Google Analytics, etc. C'est presque impossible de ne pas inclure de bibliothèques tierces dans votre site.

-Le cookie n'est pas accessible via JavaScript. Il n'est pas autant vulnérable aux attaques XSS que le localStorage. Si on utilise httpOnly et des cookies sécurisés, cela veut dire que nos cookies ne sont pas accessibles en utilisant JavaScript. Cela signifie, même si un attaquant peut utiliser Javascript sur le site, ils ne peuvent pas lire le jeton d'accès dans le cookie.

Il est automatiquement envoyé à toutes les requêtes HTPP sur notre serveur.

Dépendant de votre cas d'utilisation, il est possible que vous ne puissiez pas stocker vos jetons dans les cookies.

La taille des cookies est limitée à 4 Ko. Donc, si on utilise un gros jeton JWT , le stocker dans le cookie n'est pas envisageable.

Dans différents cas on ne peut pas partager les cookies avec le serveur d'API ou l'API exige de mettre le jeton d'accès dans l'en-tête d'autorisation. Dans ce cas, vous ne pourrez pas utiliser de cookies pour stocker vos jetons".

XI. Conclusion

En conclusion, le projet Tok-Tok a été une expérience à la fois enrichissante et exigeante. Pendant cette période intense d'un mois, nous avons dû sortir de notre zone de confort pour surmonter divers défis. Cependant, malgré les difficultés rencontrées, ce projet a été extrêmement gratifiant.

L'apprentissage quotidien de nouvelles compétences et la recherche de solutions pour des problèmes complexes ont été des aspects passionnants de cette aventure. Travailler en équipe a également été une expérience précieuse, renforçant notre capacité à collaborer efficacement et à gérer toutes les phases du développement d'une application web complète.

Sur le plan professionnel, ce projet a renforcé nos compétences essentielles en conception, développement d'interfaces, création de bases de données, développement de composants d'accès aux données et mise en place de la partie back-end. Ces compétences couvrent un large éventail d'aspects du développement web, ce qui nous donne une solide base pour aborder de futurs projets avec confiance. Le projet Tok-Tok a été une étape importante dans notre cheminement professionnel, nous préparant à relever de nouveaux défis passionnants dans le domaine du développement web.

XII. Annexes

A) Wireframes

The wireframe illustrates the Tok-Tok user profile interface and a modal for profile modification.

User Profile Page (Tok-Tok):

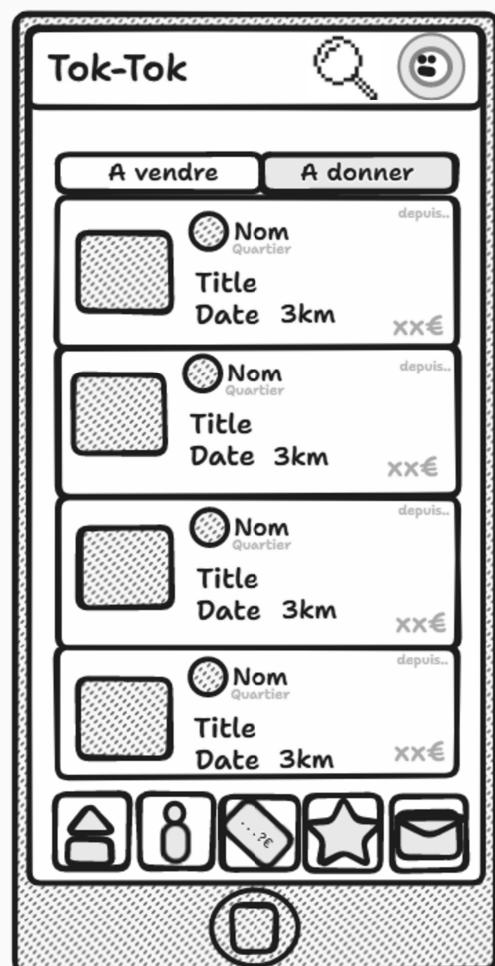
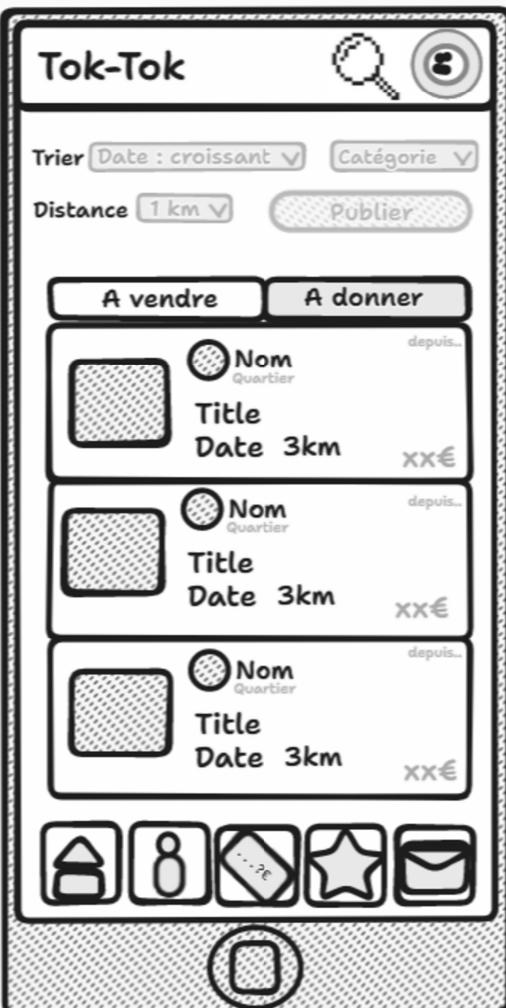
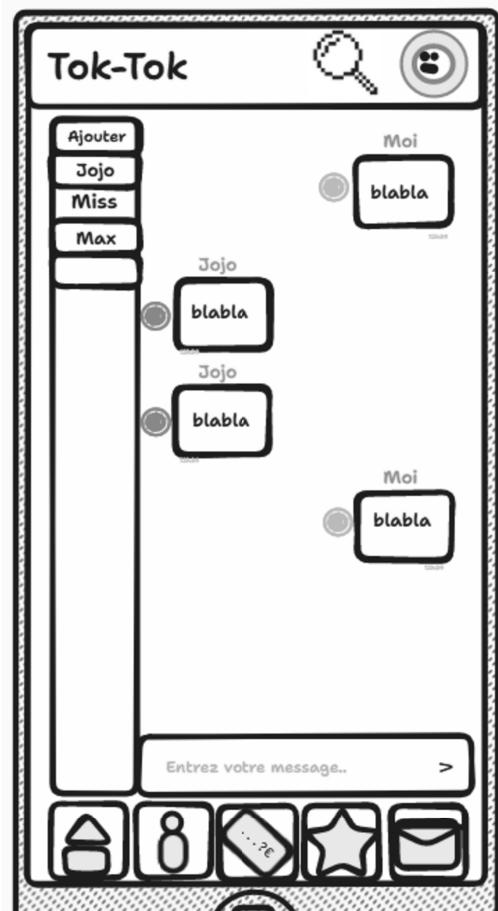
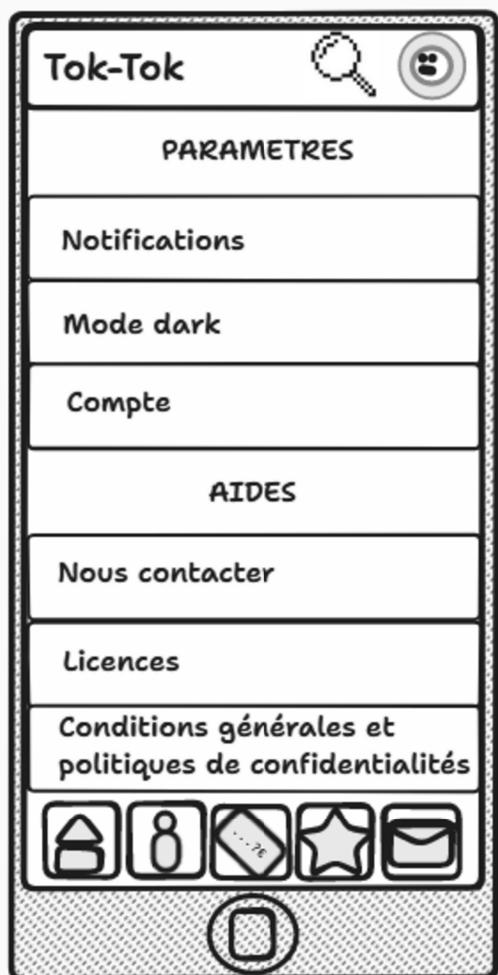
- Header: Tok-Tok, search icon, user icon.
- Profile picture placeholder.
- User Information:
 - Nom Prénom: Jean Jean
 - Localisation: date
 - Texte présentation du Profil: lorem ipsum dolor sit ame, cosecteur adipiscing elit, sed do
- Publications and Annonces sections (each showing a post by Jean Jean).
- Bottom navigation icons: house, person, settings, star, envelope.

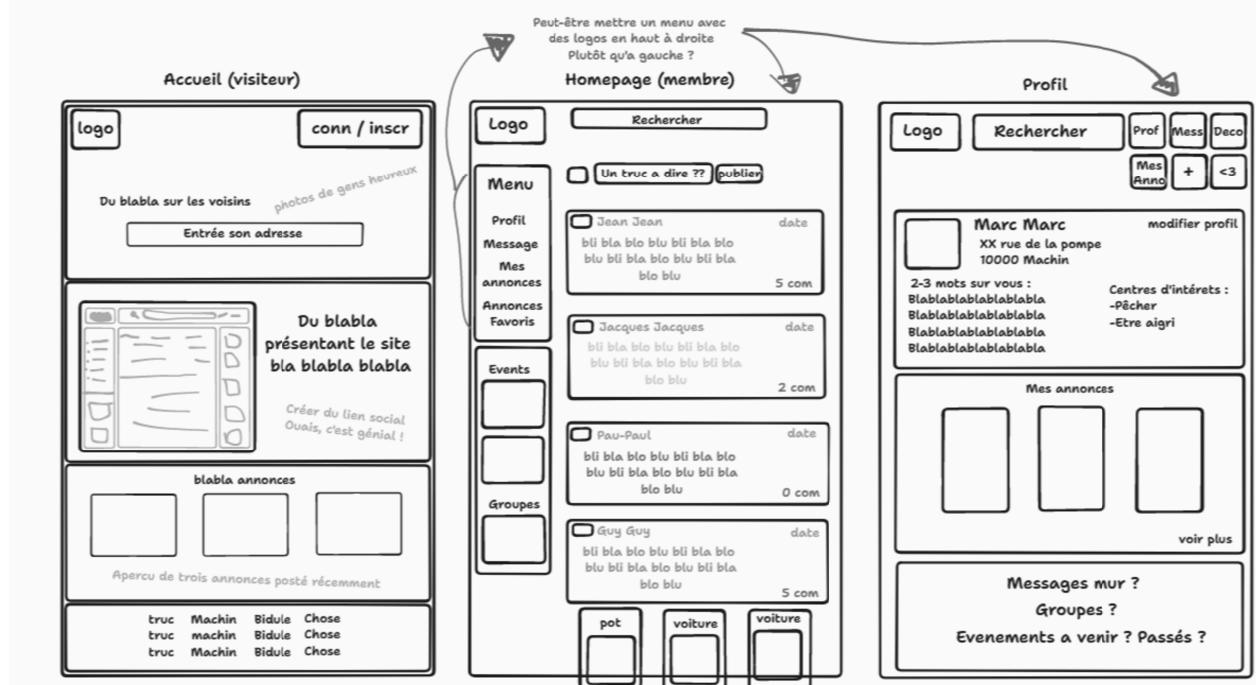
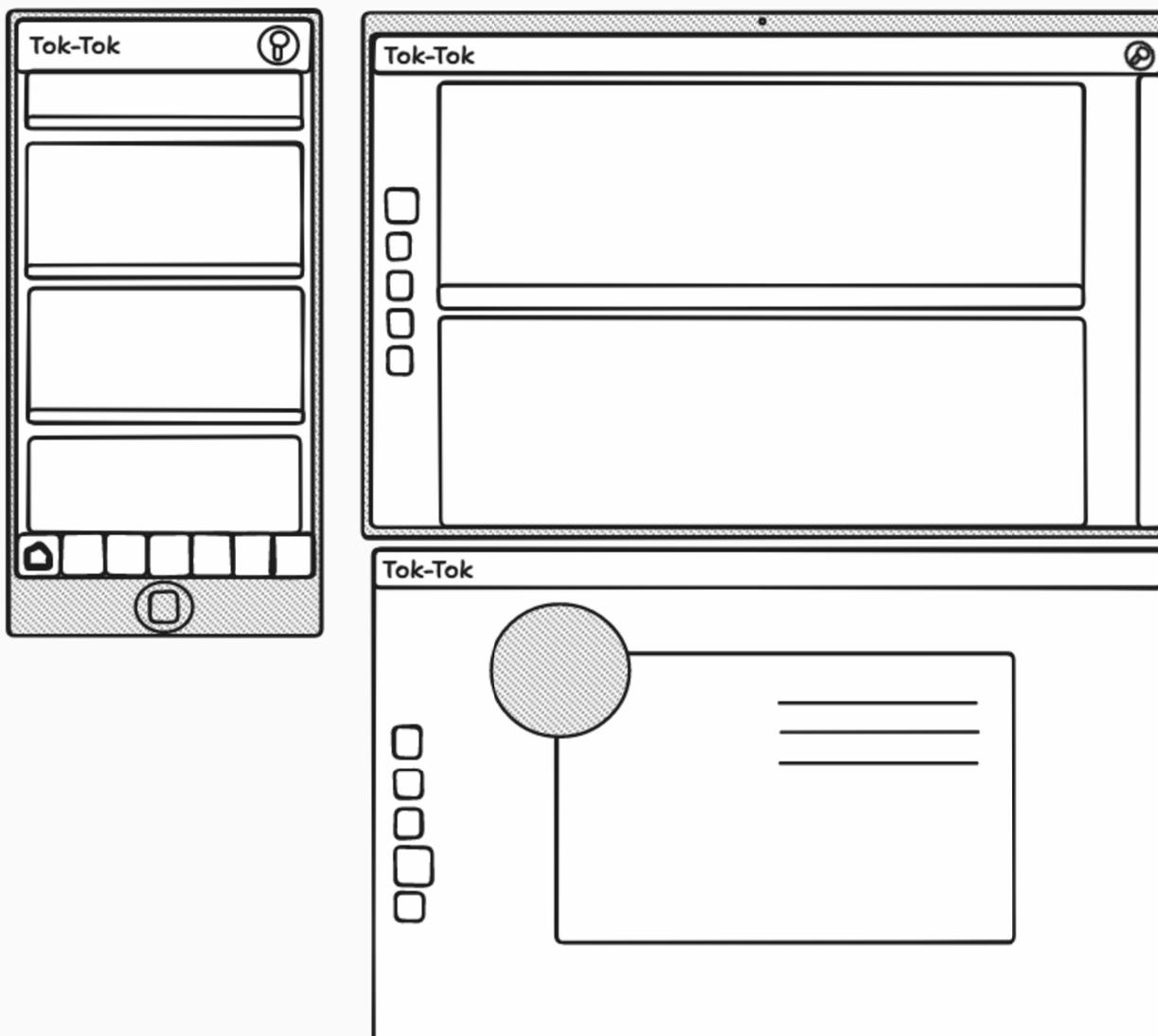
Profile Modification Modal:

- Header: Modifier le profil, close button.
- Profile picture placeholder with a plus sign (+).
- Input field: Nom Prénom (Jean Jean).
- Text area: Texte présentation du Profil (lorem ipsum dolor sit ame, cosecteur adipiscing elit, sed do).
- Save button: Sauvegarder.

Annotations:

- Si page profil du user loggué logo modification (If the user is logged in, logo modification)
- Sinon logo message (Otherwise logo message)
- Ouvre une modale pour modifier ses informations (Opens a modal to modify its information)





 Accueil Petites annonces Mon profil Messages (2) Enregistrés

Ce qui se passe autour de chez vous

Ecrire un message à vos voisins...

[Publier](#)

 John Doe - quartier Le Truc

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

 26 Voir les réponses

 John Doe - quartier Le Truc

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

 26 Voir les réponses

 John Doe - quartier Le Truc

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

 26 Voir les réponses

 John Doe - quartier Le Truc

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

 26 Voir les réponses

 John Doe - quartier Le Truc

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

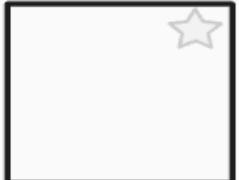
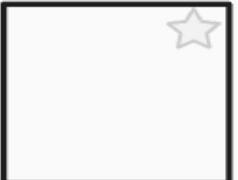
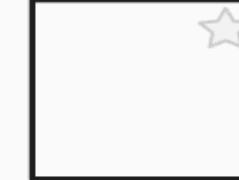
 26 Voir les réponses

[Click to return to the ads page](#)

Favoris

Logo Recherche

Mes Favoris

  Titre Prix date km	  Titre Prix date km	  Titre Prix date km	
  	  Titre Prix date km	  Titre Prix date km	  Titre Prix date km
 	  Titre Prix date km	  Titre Prix date km	  Titre Prix date km
  Titre Prix date km	  Titre Prix date km	  Titre Prix date km	

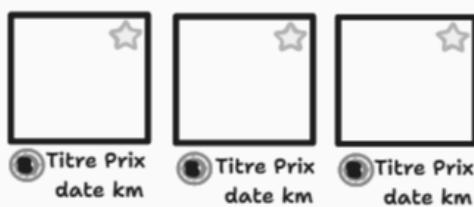
 i

Apparition d'un bouton retour



Au scroll on affiche
en dessous les
annonces du même auteur

[Autres annonces de ce vendeur](#)



Slider of photos with the possibility
to skip direction by clicking on the preview

Advert

Logo

Recherche

Add to favorites

Titre de l'annonce

XX €

Jojo date - 2 km

bli bla bla bla bli bla bla
bli bla bla bla bli bla
bli bla bla bla bli bla bla
bli bla bla bla bli bla
bli bla bla bla bli bla bla
bli bla bla bla bli bla
blo blo

Contacter le vendeur

Autres annonces de ce vendeur

i

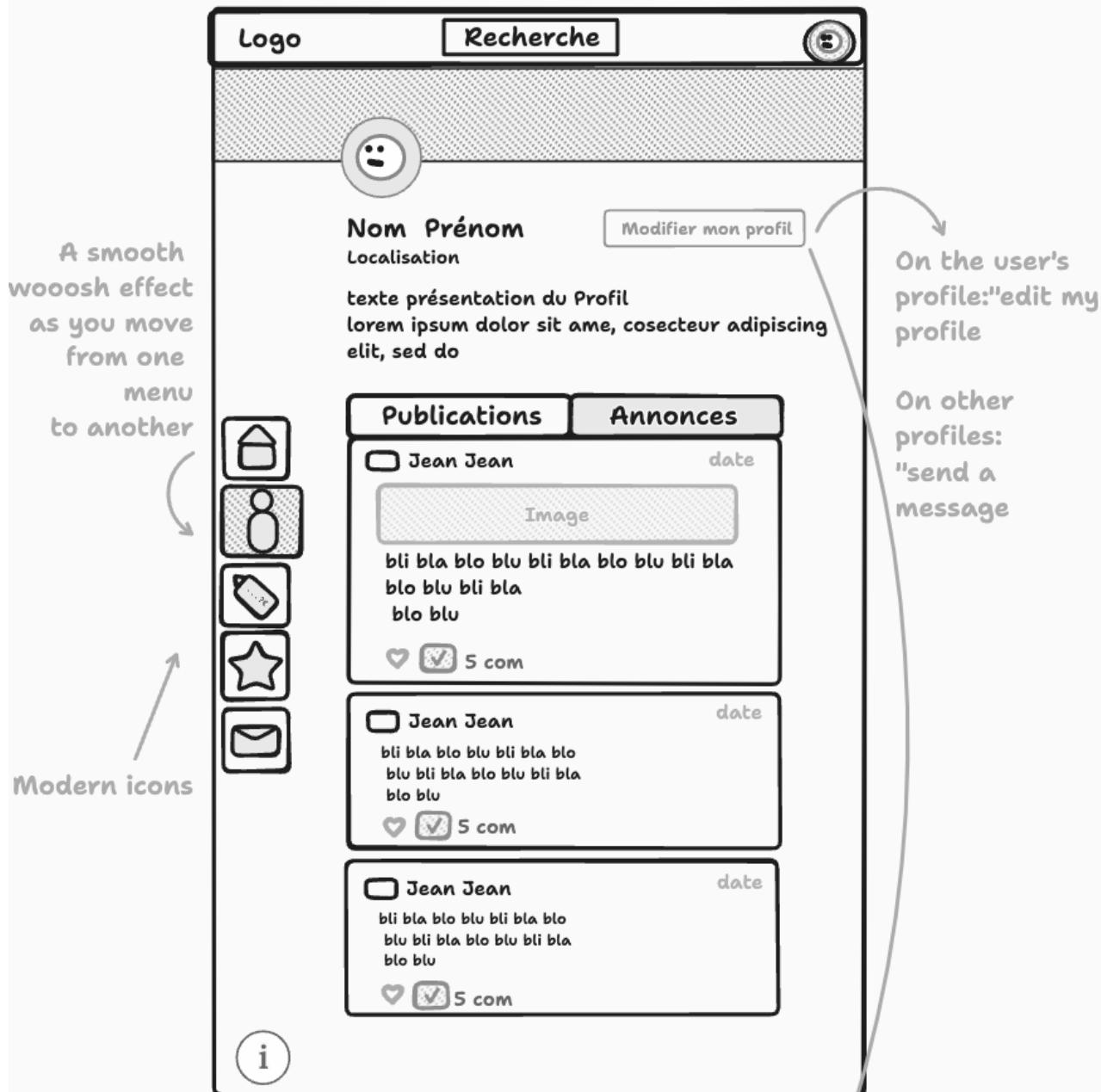
Titre Prix date km

Titre Prix date km

Titre Prix date km

Other announcements if any
Otherwise nothing

Profil



Modifier le profil

Nom Prénom

texte présentation du Profil
lorem ipsum dolor sit ame, cosecteur adipiscing elit, sed do

Sauvegarder

B) Maquettes

The image displays two side-by-side mobile application wireframes for a platform named 'TOK TOK'.

Mobile - Profile Nav (Left):

- Header: Shows the time (9:41), signal strength, and battery level.
- Top Bar: Includes the 'TOK TOK' logo, a search bar with the placeholder "Fallecida Copp", and a user profile picture.
- Notifications Panel:
 - Notifications icon with a blue badge showing '5'.
 - Settings icon.
 - Logout icon.
- Post by Justin Nicolas (55 minutes ago):

There is a heavy traffic jam on Alberta between St Jhon's and St Paul's. The jam is currently about 5 miles long and has been going on for about 6 hours. The cause of the jam is not yet known.
If you are planning to travel on this highway, please allow extra time for your commute. You may also want to consider taking an alternate route.

6.2k likes, 156 comments.
- Bottom Navigation Bar:
 - Home icon (highlighted in green).
 - Notifications icon with a blue badge showing '5'.
 - Profile icon with a blue badge showing '2'.
 - Search icon.
 - Bookmark icon.

Mobile - Home (Right):

- Header: Shows the time (9:41), signal strength, and battery level.
- Top Bar: Includes the 'TOK TOK' logo, a search bar with the placeholder "Fallecida Copp", and a user profile picture.
- Post by Justin Nicolas (55 minutes ago):

What's on your mind?

Image and Video options.

6.2k likes, 156 comments.
- Post by Sinthia Jonathon (25 minutes ago):

There is a heavy traffic jam on Alberta between St Jhon's and St Paul's. The jam is currently about 5 miles long and has been going on for about 6 hours. The cause of the jam is not yet known.
If you are planning to travel on this highway, please allow extra time for your commute. You may also want to consider taking an alternate route.

6.2k likes, 156 comments.
- Bottom Navigation Bar:
 - Home icon (highlighted in green).
 - Notifications icon with a blue badge showing '5'.
 - Profile icon with a blue badge showing '2'.
 - Search icon.
 - Bookmark icon.

Homepage

TOK TOK

Fallecida Copp

Smith Dhin Local Guide

What's on your mind?

Publish

Image Video

Justin Nicolas 55 minutes ago

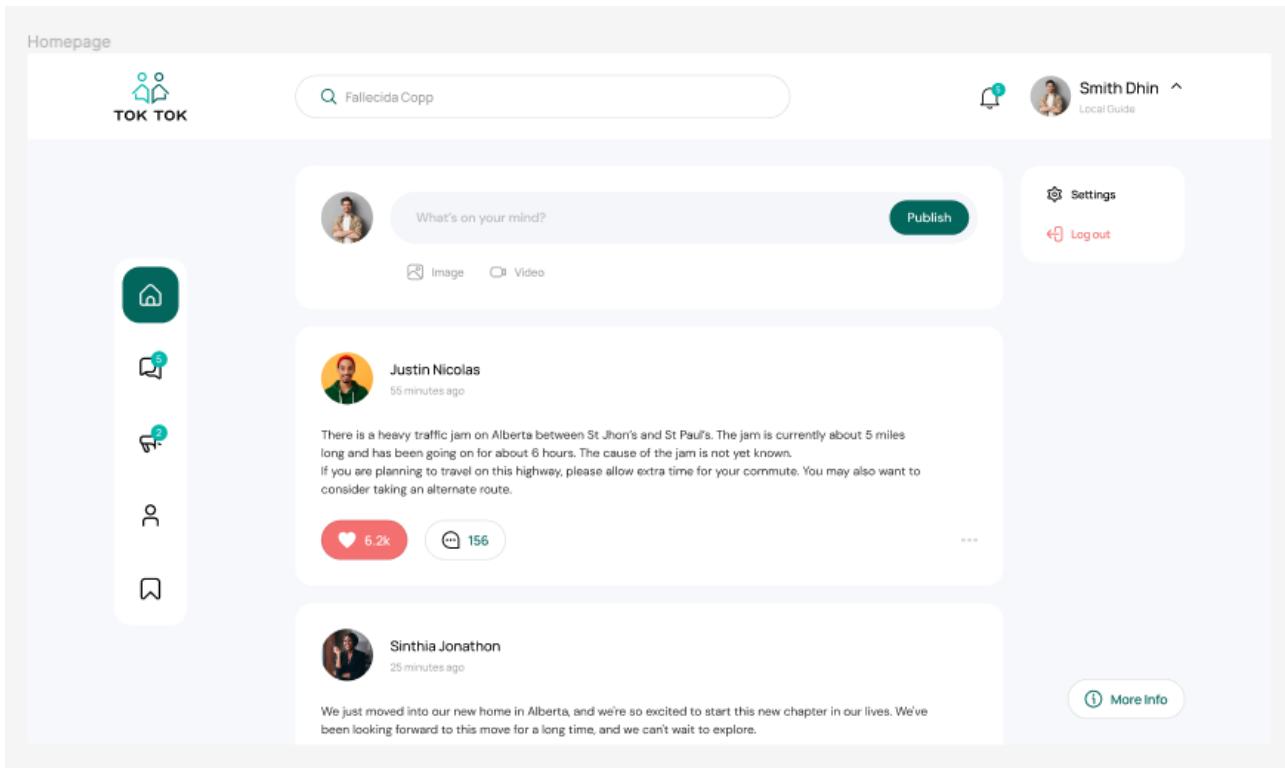
There is a heavy traffic jam on Alberta between St Jhoni's and St Paul's. The jam is currently about 5 miles long and has been going on for about 6 hours. The cause of the jam is not yet known. If you are planning to travel on this highway, please allow extra time for your commute. You may also want to consider taking an alternate route.

6.2k 156

Sinthia Jonathon 25 minutes ago

We just moved into our new home in Alberta, and we're so excited to start this new chapter in our lives. We've been looking forward to this move for a long time, and we can't wait to explore.

More Info



Profile - Edit

TOK TOK

Fallecida Copp

Smith Dhin Local Guide

Edit cover

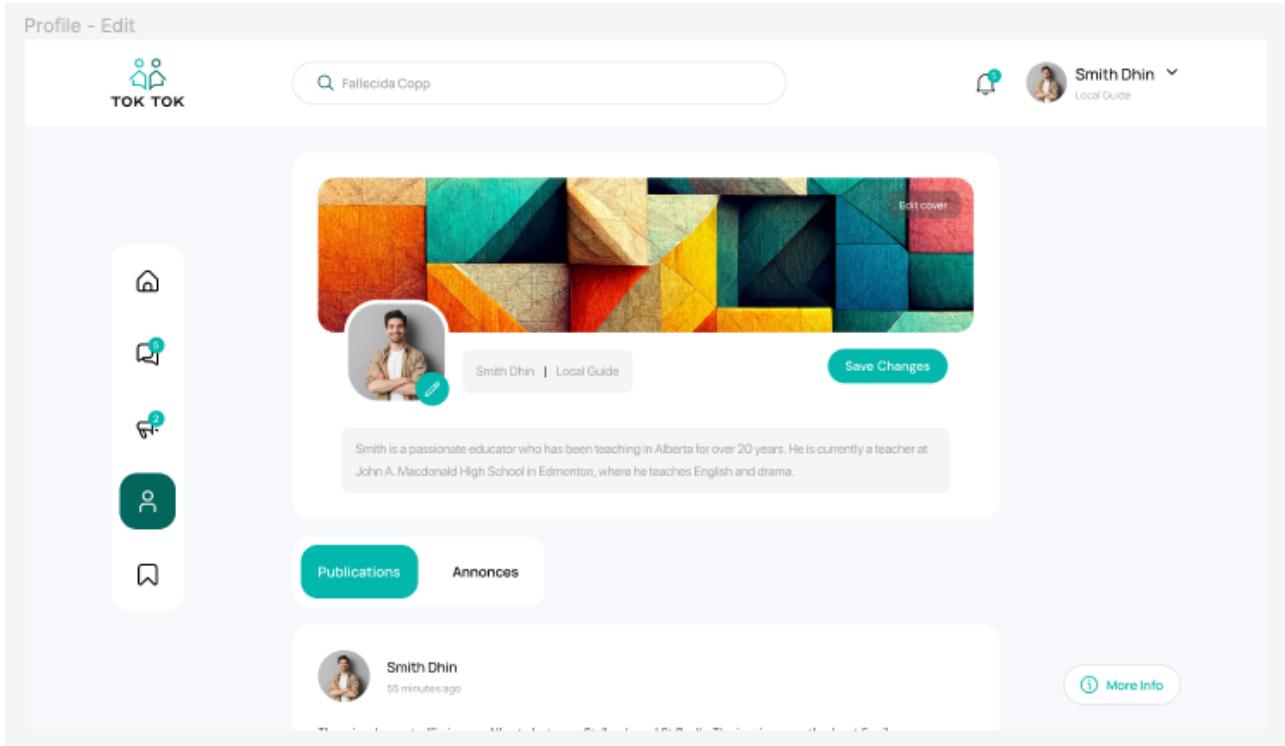
Save Changes

Smith is a passionate educator who has been teaching in Alberta for over 20 years. He is currently a teacher at John A. Macdonald High School in Edmonton, where he teaches English and drama.

Publications Announcements

Smith Dhin 55 minutes ago

More Info



9:41



Fallecida Copp



Smith Dhin

Local Guide



Smith is a passionate educator who has been teaching in Alberta for over 20 years. He is currently a teacher at John A. Macdonald High School in Edmonton, where he teaches English and drama.

Publications

Annonces



9:41



Fallecida Copp



Smith Dhin | Local Guide

Smith is a passionate educator who has been teaching in Alberta for over 20 years. He is currently a teacher at John A. Macdonald High School in Edmonton, where he teaches English and drama.

Save Changes




 Falliceda Copp


Smith Dhin Local Guide



What's on your mind?

Publish

 Image VideoJustin Nicolas
55 minutes ago

There is a heavy traffic jam on Alberta between St Jhon's and St Paul's. The jam is currently about 5 miles long and has been going on for about 6 hours. The cause of the jam is not yet known. If you are planning to travel on this highway, please allow extra time for your commute. You may also want to consider taking an alternate route.

 6.2k 156

...

Sinthia Jonathon
26 minutes ago More Info

We just moved into our new home in Alberta, and we're so excited to start this new chapter in our lives. We've been looking forward to this move for a long time, and we can't wait to explore.

 450k 1.2k

...

Maeve Ka
2 hours ago

I'm always on the lookout for new cycling routes to explore. If you have any suggestions, please let me know! Cycling is a great way to get exercise, explore your surroundings, and meet new people.

 616k 112k

...

Samantha Love
6 hours ago

There is a heavy traffic jam on Alberta between St Jhon's and St Paul's. The jam is currently about 5 miles long and has been going on for about 6 hours. The cause of the jam is not yet known. If you are planning to travel on this highway, please allow extra time for your commute. You may also want to consider taking an alternate route.

 450k 1.2k

...

Samantha Love
6 hours ago

There is a heavy traffic jam on Alberta between St Jhon's and St Paul's. The jam is currently about 5 miles long and has been going on for about 6 hours. The cause of the jam is not yet known. If you are planning to travel on this highway, please allow extra time for your commute. You may also want to consider taking an alternate route.

 450k 1.2k

...

 Load more

Profile

 Fallecida Copp

 Smith Dhin Local Guide



 Smith Dhin Local Guide [Edit Profile](#)

Smith is a passionate educator who has been teaching in Alberta for over 20 years. He is currently a teacher at John A. Macdonald High School in Edmonton, where he teaches English and drama.

[Publications](#) [Announcements](#)

 Smith Dhin 55 minutes ago [More Info](#)

There is a heavy traffic jam on Alberta between St Jhon's and St Paul's. The jam is currently about 5 miles long and has been going on for about 6 hours. The cause of the jam is not yet known. If you are planning to travel on this highway, please allow extra time for your commute. You may also want to consider taking an alternate route.

 6.2k  156 ...

 Smith Dhin 25 minutes ago

We just moved into our new home in Alberta, and we're so excited to start this new chapter in our lives. We've been looking forward to this move for a long time, and we can't wait to explore.

 450k  1.2k ...

 Smith Dhin 6 hours ago

There is a heavy traffic jam on Alberta between St Jhon's and St Paul's. The jam is currently about 5 miles long and has been going on for about 6 hours. The cause of the jam is not yet known. If you are planning to travel on this highway, please allow extra time for your commute. You may also want to consider taking an alternate route.

 450k  1.2k ...

 Samantha Love 6 hours ago

There is a heavy traffic jam on Alberta between St Jhon's and St Paul's. The jam is currently about 5 miles long and has been going on for about 6 hours. The cause of the jam is not yet known. If you are planning to travel on this highway, please allow extra time for your commute. You may also want to consider taking an alternate route.

 450k  1.2k ...



C) Dico de données

Table user

Champ	Type	Spécificités	Description
id	INT	PRIMARY KEY, UNSIGNED, NOT NULL, AUTO_INCREMENT	L'identifiant de la personne
firstname	VARCHAR(64)	NOT NULL	Le prénom de la personne
lastname	VARCHAR(64)	NOT NULL	Le nom de la personne
description	VARCHAR(255)		La description de la personne sur son profil
adress	TEXT	NOT NULL	L'adresse de personne
localisation	TEXT	NOT NULL	La position exact de l'utilisateur à son inscription
email	VARCHAR(64)	NOT NULL	L'email de la personne
password	VARCHAR(64)	NOT NULL	Le mot de passe de la personne
thumb nail	TEXT		La photo de profil de la personne
created_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	La date de création du profil
updated_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	La date de dernière modification du profil

Table annonce

Champ	Type	Spécificités	Description
id	INT	PRIMARY KEY, UNSIGNED, NOT NULL, AUTO_INCREMENT	L'identifiant de l'annonce
title	VARCHAR (64)	NOT NULL	Titre de l'Annonce
content	TEXT	NOT NULL	Contenu de l'annonce
price	SMALLINT	NOT NULL	Prix de l'objet
user_id	INT	PRIMARY KEY, UNSIGNED, NOT NULL, AUTO_INCREMENT	L'identifiant de la personne qui à crée l'annonce
tag_id	INT	PRIMARY KEY, UNSIGNED, NOT NULL, AUTO_INCREMENT	L'identifiant de la catégorie
created_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Date de la création de l'annonce
updated_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Date de la dernière modification

Table annonce_image

Champ	Type	Spécificités	Description
id	INT	PRIMARY KEY, UNSIGNED, NOT NULL, AUTO_INCREMENT	L'identifiant de la relation
annonce_id	INT	PRIMARY KEY, UNSIGNED, NOT NULL, AUTO_INCREMENT	I'identifiant de l'annonce
thumbnail	TEXT	NOT NULL	Photo de l'objet de l'annonce
created_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Date de la création de l'annonce
updated_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Date de la dernière modification

Table post

Champ	Type	Spécificités	Description
id	INT	PRIMARY KEY, UNSIGNED, NOT NULL, AUTO_INCREMENT	L'identifiant du message
content	TEXT	NOT NULL	Le texte du message
thumbnail	TEXT		L'image du post
reply_to	INT		L'id du post auquel ce post répond
user_id	ENTITY	PRIMARY KEY, NOT NULL	Le nom de la personne qui a envoyé le message
created_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	La date de création/envoi du post
updated_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	La date de modification du post

Table message

Champ	Type	Spécificités	Description
id	INT	PRIMARY KEY, UNSIGNED, NOT NULL, AUTO_INCREMENT	L'identifiant du message
content	TEXT	NOT NULL	Le texte du message
expéditeur	ENTITY	PRIMARY KEY, NOT NULL	L'id de la personne qui a envoyé le message
destinataire	ENTITY	PRIMARY KEY, NOT NULL	L'id de la personne à qui le message est envoyé
conversation_id	SEQUENCE	NOT NULL	L'identifiant de la conversation
created_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	La date de création/envoi du message

Table tag

Champ	Type	Spécificités	Description
id	INT	PRIMARY KEY, UNSIGNED, NOT NULL, AUTO_INCREMENT	L'identifiant de la catégorie
name	VARCHAR (64)	NOT NULL	Nom de la catégorie

Table favoris

Champ	Type	Spécificités	Description
id	INT	PRIMARY KEY, UNSIGNED, NOT NULL, AUTO_INCREMENT	L'identifiant de la liaison annonce-user
annonce_id	ENTI TY	PRIMARY KEY, NOT NULL	L'identifiant de l'annonce
user_id	ENTI TY	PRIMARY KEY, NOT NULL	L'identifiant de l'utilisateur

Table likes

Champ	Type	Spécificités	Description
id	INT	PRIMARY KEY, UNSIGNED, NOT NULL, AUTO_INCREMENT	L'identifiant de la liaison post-user
post_id	ENTI TY	PRIMARY KEY, NOT NULL	L'identifiant du post
user_id	ENTI TY	PRIMARY KEY, NOT NULL	L'identifiant de l'utilisateur

D) Liste des routes

Routes front

/ ⇒ Route accueil non-loggué

/home ⇒ Route accueil loggué

/profil/:id ⇒ Route profil paramétré

/annonces ⇒ Route toutes les annonces

/annonces/:id ⇒ Route une annonce paramétré

/favoris ⇒ Route les favoris d'annonces

/messagerie ⇒ Route messagerie

/about ⇒ Route à propos

/contact ⇒ Route contact

/search ⇒ Route de résultat des recherches

Routes Back

Méthode	nom de la route	paramètres	description	retour
POST	/login		Authentifier un utilisateur	objet
POST	/register		Enregistrer un nouvel utilisateur	objet
GET	/posts		Obtenir tous les posts dans notre rayon, tri décroissant par date	tableau d'objets
POST	/posts		Créer un nouveau post	objet
PATCH	/posts/:id	L'id du post à modifier	Modifier une publication	objet
DELETE	/posts/:id	L'id du post à supprimer	Supprimer une publication	objet
GET	/profil/:id	L'identifiant du profil	Afficher le profil d'un utilisateur	objet
PATCH	/profil/:id	L'identifiant du profil	Modifier les informations d'un profil	objet
DELETE	/profil/:id	L'identifiant du profil	Supprimer un profil	objet
GET	/annonces		Afficher toutes les annonces dans notre rayon, tri décroissant par date (par défaut)	tableau d'objets
POST	/annonces		Créer une nouvelle annonce	objet
GET	/annonces/:id	L'id de l'annonce cliquée	Afficher les informations de l'annonce cliquée et ses photos	objet
PATCH	/annonces/:id	L'id de l'annonce cliquée	Modifier l'annonce cliquée et ses photos	objet
DELETE	/annonces/:id	L'id de l'annonce cliquée	Supprimer l'annonce cliquée et ses photos	objet
GET	/annonces?category=category	category : La catégorie sélectionnée dans les filtres de recherche	Afficher les annonces dans notre rayon dont la catégorie est category	tableau d'objets
GET	/annonces?	distance : La distance	Afficher les annonces	tableau

Méthode	nom de la route	paramètres	description	retour
	distance=distance	sélectionnée dans les filtres de recherche	dans le rayon sélectionné	d'objets
GET	/annonces?orderby=date	distance : La distance sélectionnée dans les filtres de recherche	Afficher les annonces dans le rayon sélectionné	tableau d'objets
GET	/favourites		Afficher les annonces favorites du user loggué	tableau d'objets
POST	/favourites/:id/add	L'id de l'annonce à ajouter aux favoris	Ajouter une annonce aux favoris	objet
DELETE	/favourites/:id/remove	L'id de l'annonce à ajouter aux favoris	Supprimer une annonce des favoris	objet
GET	/search			tableau d'objets
GET	/messages		Afficher tous les messages de l'utilisateur	tableau d'objets
GET	/messages/:id	L'id de la conversation	Afficher tous les messages de la conversation	tableau d'objets
POST	/messages/:id	L'id de la conversation	Envoyer un message dans la conversation	objet