Université de Montréal

DIF

CALQUE

Camille Divisia, Emanuel Rollin et Xia Benoit

Supervisé par Louis-Edouard Lafontant

IFT3150 - Projet en Informatique

Plan de la présentation

- Introduction
- Analyse
- Conception et Implémentation
- Démonstration
- Tests unitaires
- Roadmap
- Conclusion

Introduction - Problématique

- L'orientation dans des espaces inconnus pose des défis quotidiens
- L’aspect souvent statique des cartes limite l’accessibilité et la flexibilité de la navigation
- Les cartes interactives offrent une solution pour améliorer l'accessibilité et l'efficacité de la navigation aux utilisateurs.

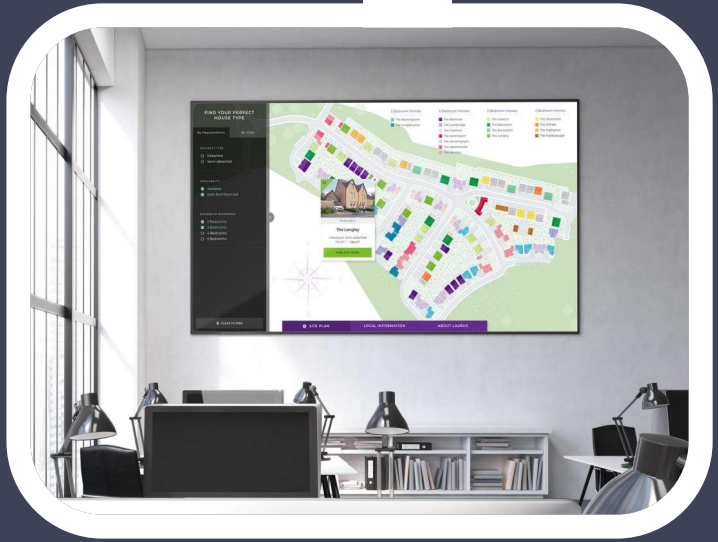
Introduction - But du Projet

- Faciliter leur création et leur maintenance par les designers.
- Faciliter la navigation et l’utilisation de cartes

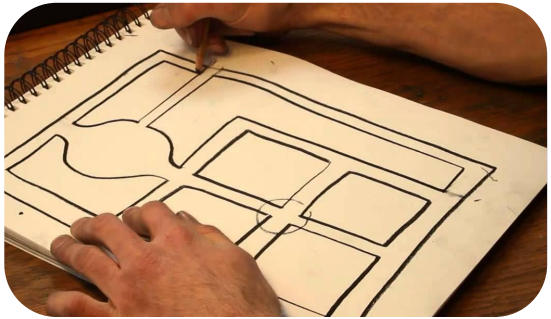
Fonctionnalités: l'ajout de points d'intérêt, la simulation de trajets, et la génération d'itinéraires basés sur des requêtes.

But:

Créer un outil de génération de cartes interactives, avec une plateforme adéquate pour pouvoir les utiliser.



Analyse - Acteurs principaux



Designer

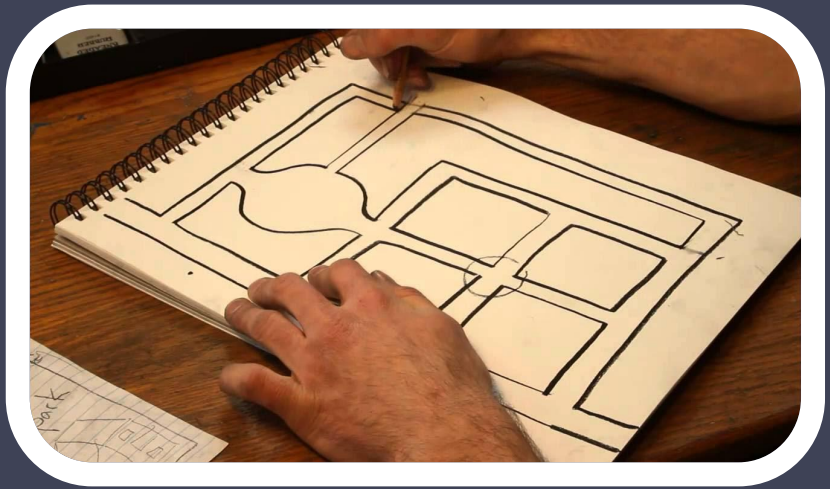
Crée des cartes et ajoute des interactions

Utilisateur Final

Utilise la carte créée par le designer.

Analyse - Besoin fonctionnels - Designer

- Ajouter les entités sur le canvas.
- Manipuler les entités et leurs attributs
- Exporter la carte en un fichier .calque, permettant à un utilisateur final de s’en servir sur l’application.
- Sauvegarder l'état de la carte



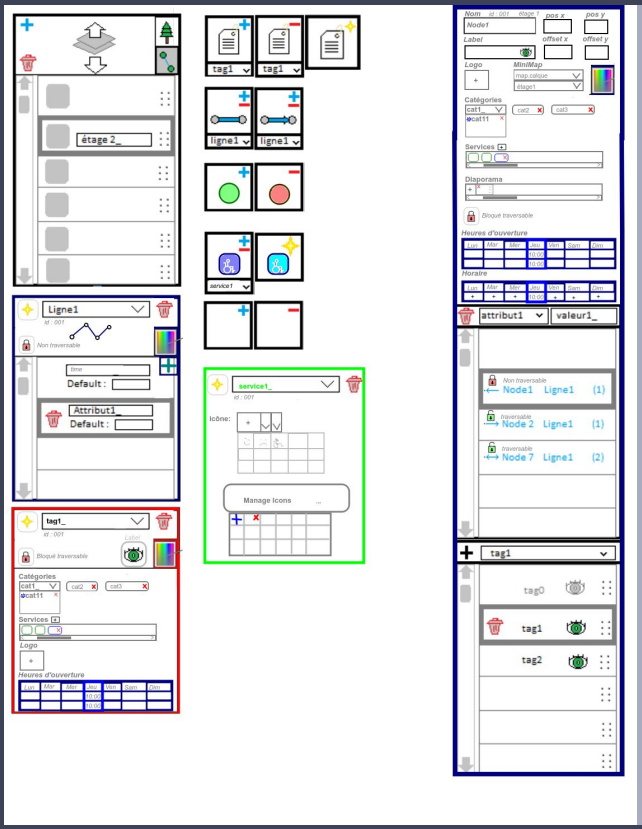
Analyse - Besoin fonctionnels - Utilisateur Final

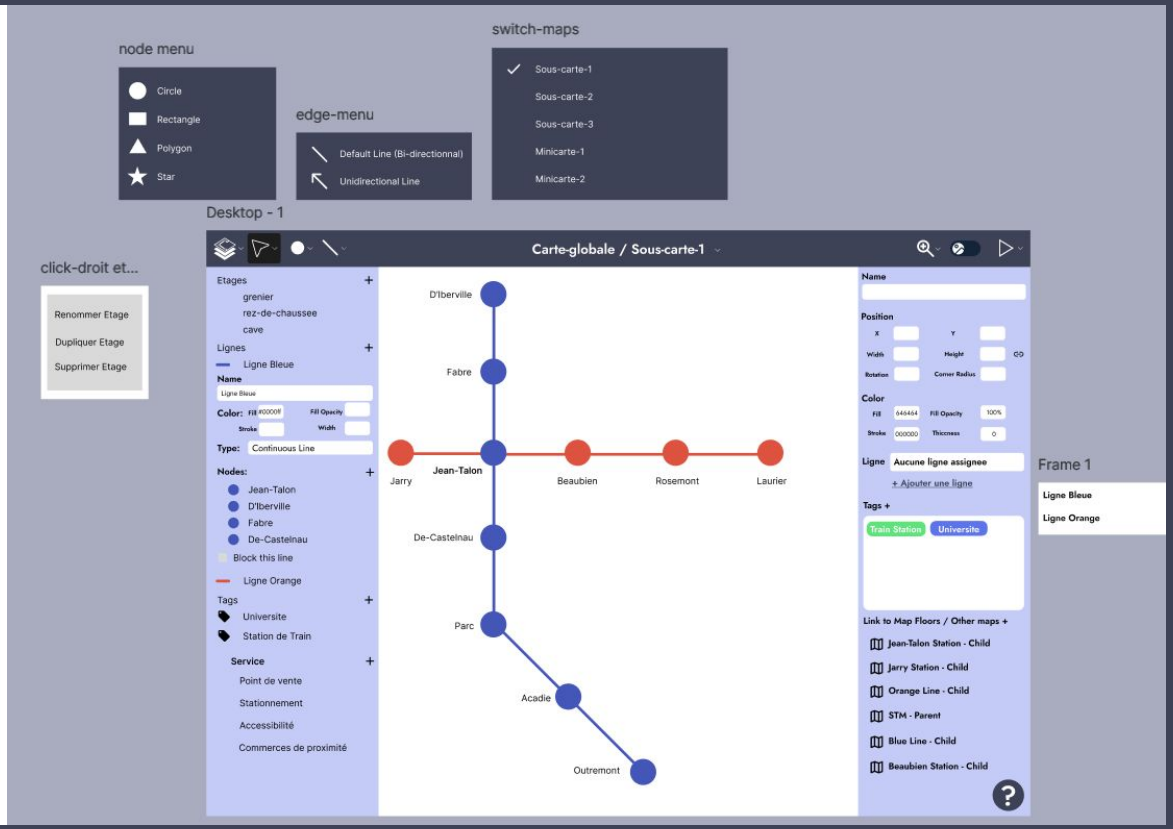
- Tracer un itinéraire (Ici vers B, A vers B, A vers B vers C)
- Afficher les blocs d'information du nœud sélectionné
- Afficher / masquer certains éléments de la carte.
- Pouvoir réinitialiser la carte à son état de départ

Analyse - Besoin non fonctionnels

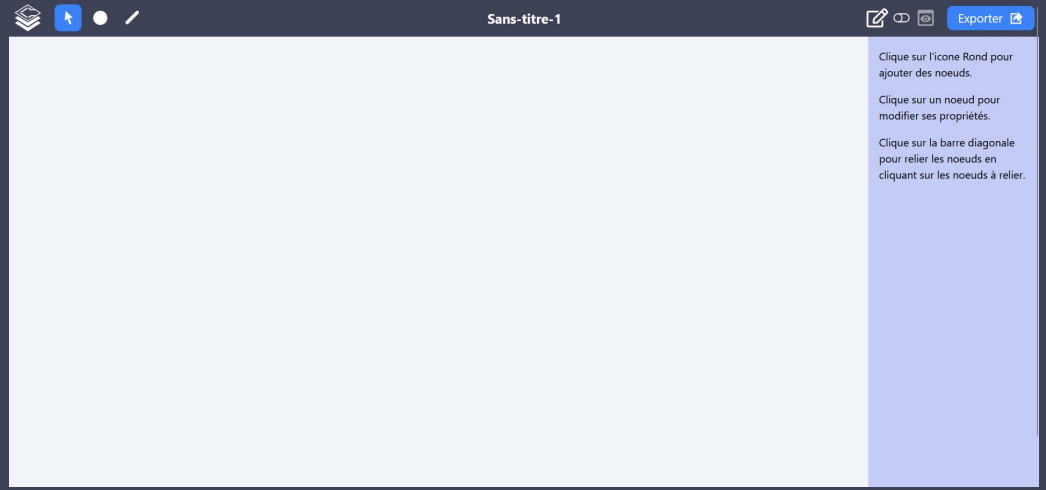
- **Flexibilité**: Accommoder le designer dans les choix de design
- **Documentation**: Aide disponible aux utilisateurs
- **Portabilité**: Compatibilité maximale sur tous les OS / systèmes
- **Évolutivité**: Amélioration du projet de la façon la moins destructive possible
- **Accessibilité / Utilisabilité**: Interface intuitive et facile à utiliser pour tous
- **Maintenabilité:** Modularité du code
- **Mode hors-ligne:** Cartes utilisables sans connexion internet.

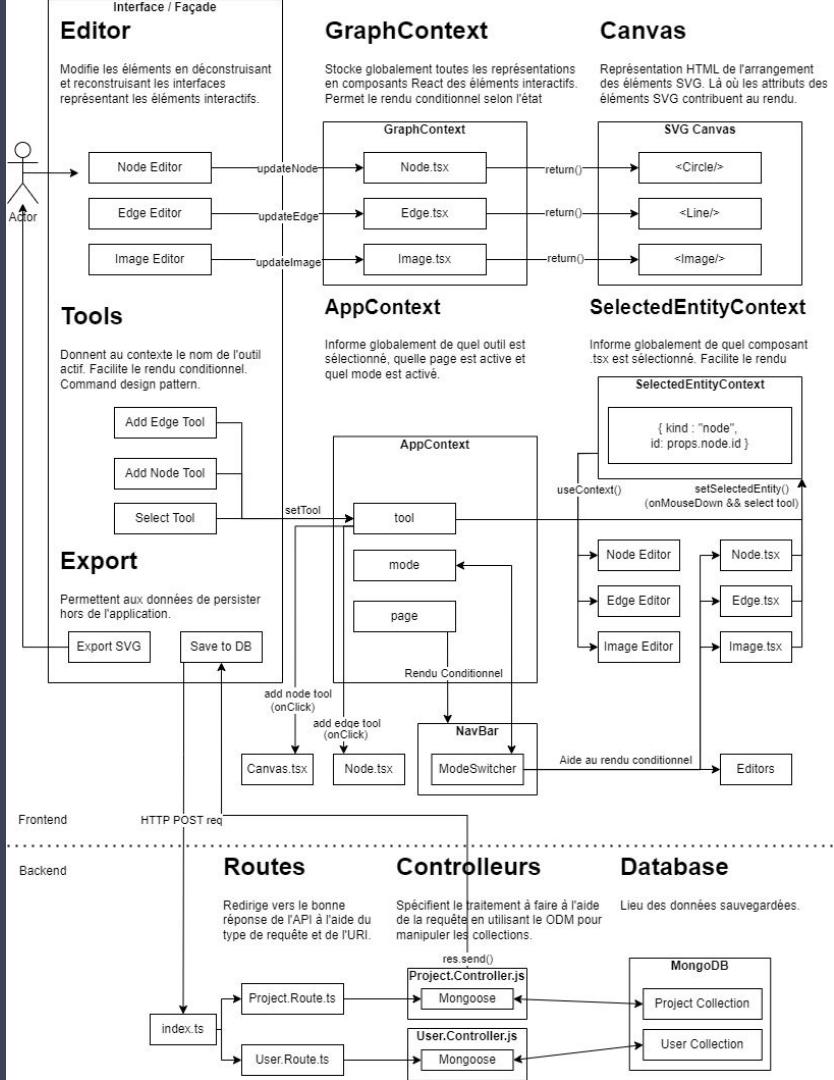
Conception et Implémentation - Prototypes



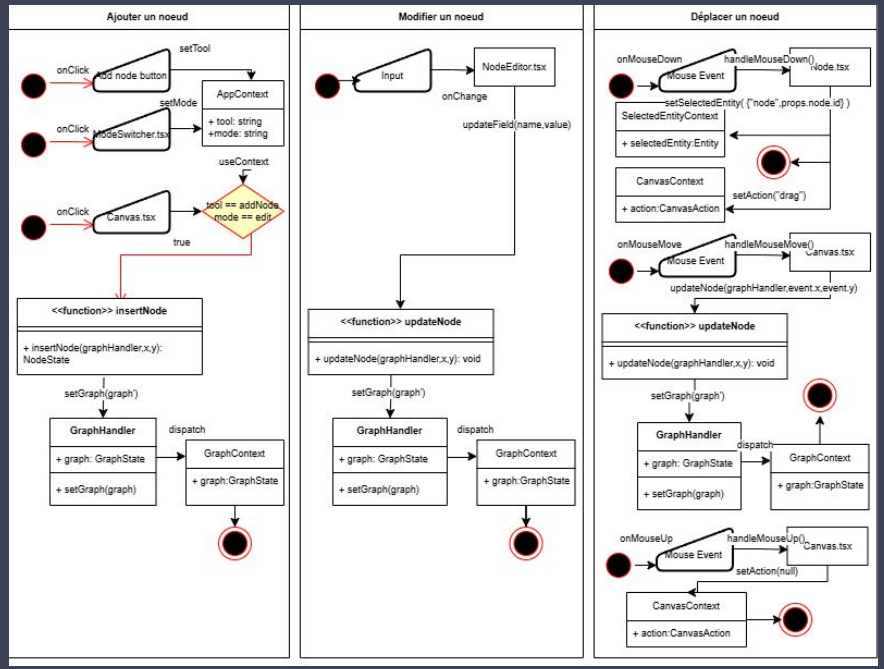


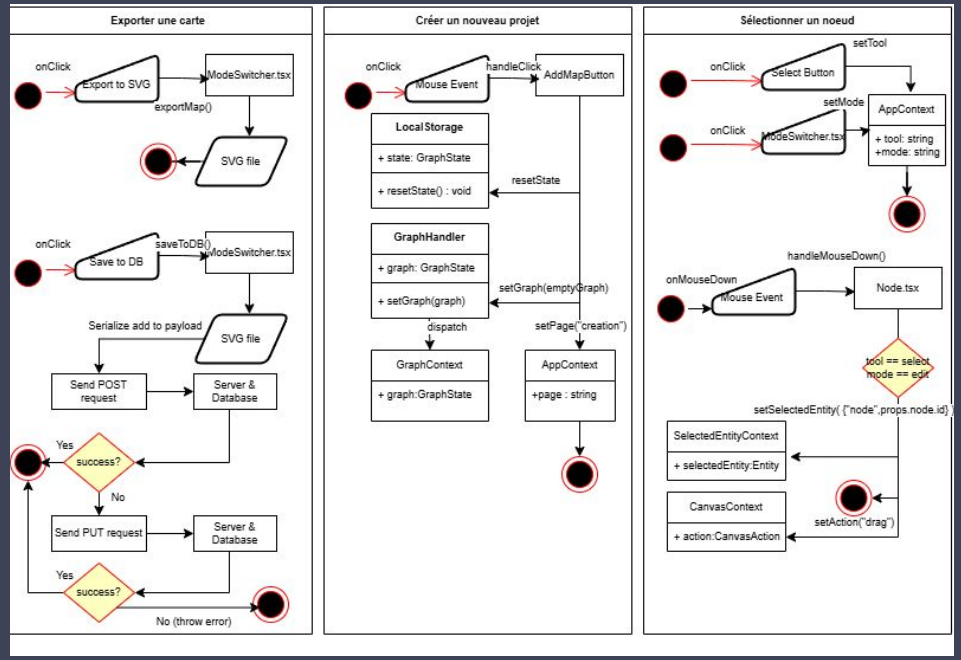
Conception et Implémentation - Frontend (Haut Niveau)





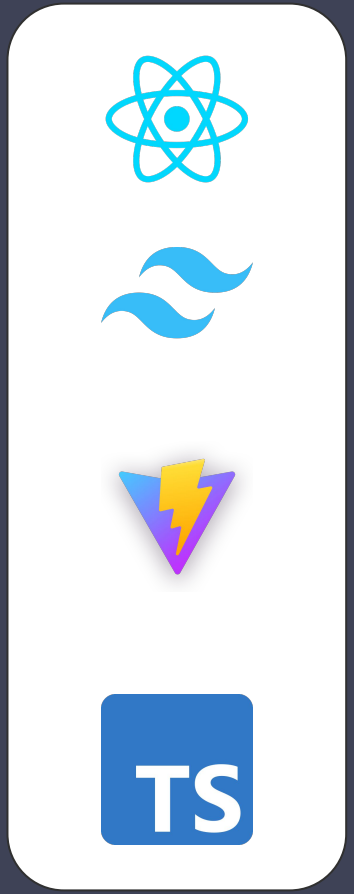
Conception et Implémentation - Frontend (Bas niveau)





Conception et Implémentation - Frontend

- React : Bibliothèque JavaScript pour construire des interfaces utilisateur interactives, en utilisant des composants réutilisables.
- Vite : Hot-reloading et détection d’erreurs
- React Hooks : useState, useContext
- Tailwindcss : Framework pour CSS avec classes prédéfinies, donne du style de manière rapide et flexible.
- TypeScript: JavaScript avec typage statique



Conception et Implémentation - Technologies du Backend

MongoDB

Base de données NoSQL orientée document, pour une gestion efficace des données

ex

Express.js

Framework minimaliste pour Node.js qui facilite la création d'applications web et d'API.

Typescript

Typage statique à JavaScript. Détection d'erreurs de type à la compilation. Application plus robuste et fiable

Node.js

Environnement d'exécution JavaScript côté serveur pour construire des applications web rapides et évolutives

➡ Architecture REST

Serveur - Architecture REST

app.ts

```ts
// Route setup
app.use('/api/user', userRoute);
app.use('/api/project', projectRoute);
```

user.route.ts

```ts
const router: Router = express.Router();


// Route: GET all users
router.get('/', UserController.getAllUsers);

// Route: GET user by ID
router.get('/:id', UserController.getUserById);

// Route: POST create a new user
router.post('/', UserController.createUser);

// Route: PUT update user by ID
router.put('/:id', UserController.updateUser);

// Route: DELETE user by ID
router.delete('/:id', UserController.deleteUser);

export default router;
```

user.controller.ts

```ts
// GET user by ID
public async getUserById(req: Request, res: Response): Promise<void> {
    try {
        //console.log(req.params.id)
        const user: string | null = await User.findOne({username:req.params.id});
        if (!user) {
            res.status(404).json({ message: 'User not found' });
            return;
        }
        res.json(user);
    } catch (err:any) {
        res.status(500).json({ message: err.message });
    }
}
```

Mongoose - MongoDB

```
await User.findOne({username:req.params.id});
```

```
// Define the Mongoose schema for User
const userSchema: Schema = new Schema({
    username: {
        type: String,
        required: true,
        unique: true
    },
    email: {
        type: String,
        required: true,
        unique: true
    },
    password: {
        type: String,
        required: true
    },
    fullName: {
        type: String,
        default: ''
    },
    bio: {
        type: String,
        default: ''
    },
```

test.users

Filter Type a query: { field: 'value' }

```
_id: ObjectId('667cd95c44fc68edc1629fa0')
username : "john_doe"
email : "john.doe@example.com"
password : "password123"
fullName : "John Doe Updated"
bio : "Software Engineer interested in AI and machine learning
age : 30
gender : "Male"
location : "New York, USA"
▸ interests : Array (3)
role : "user"
createdAt : 2024-06-27T03:15:40.359+00:00
avatarUrl : "https://example.com/avatar/john_doe.jpg"
isActive : true
lastLogin : null
__v : 0
```

Backend - Structure générale

Tests unitaires

- Utilise JEST
- Utilise supertest

```
PASS src/tests/userRoutesSuccess.test.ts (5.503 s)
  GET /api/user
    √ should return all users (1227 ms)
  GET /api/user/:id
    √ should return a user with the id "john_doe" (135 ms)
  PUT /api/user/:id
    √ should update the user with the id "john_doe" (111 ms)
  POST /api/user/:id
    √ should create the user with the id "john_doe2" (263 ms)
  DELETE /api/user/:id
    √ should delete the user with the id "john_doe2" (124 ms)

PASS src/tests/projectRoutesSuccess.test.ts (5.498 s)
  GET /api/project
    √ should return all projects (1243 ms)
  GET /api/project/:id
    √ should return a project with the id "test" (113 ms)
  PUT /api/project/:id
    √ should update the project with the id "test" (145 ms)
  POST /api/project/:id
    √ should create the project with the id "test2" (291 ms)
  DELETE /api/project/:id
    √ should delete the project with the id "test2" (124 ms)
```

tests
- app.test.ts
- projectRoutesFail.test.ts
- projectRoutesSuccess.test.ts
- userRoutesFail.test.ts
- userRoutesSuccess.test.ts

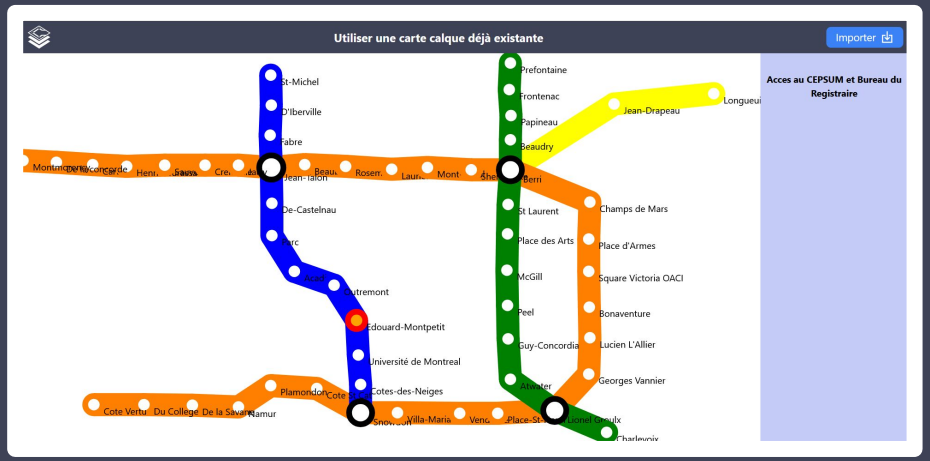
File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	89.1	62.16	81.25	89.1	
src	76.13	60	33.33	76.13	
app.ts	100	100	100	100	
database.ts	100	100	100	100	
index.ts	0	0	0	0	1-5
server.ts	0	0	0	0	1-16
src/controllers	83.63	60	90.9	83.63	
project.controller.ts	85.86	60	100	85.86	16-17,30-31,40-42,59-60,73-74,87-88
user.controller.ts	82.03	60	83.33	82.03	9-18,29-30,44-45,57-59,93-94,108-109,122-123
src/middleware	100	100	100	100	
res_logger.middleware.ts	100	100	100	100	
src/models	100	100	100	100	
project.model.ts	100	100	100	100	
user.model.ts	100	100	100	100	
src/routes	100	100	100	100	
project.route.ts	100	100	100	100	
user.route.ts	100	100	100	100	

```
Test Suites: 5 passed, 5 total
Tests:       19 passed, 19 total
Snapshots:   0 total
Time:        6.188 s
Ran all test suites.
PS C:\Users\lipob\Documents\GitHub\tp2245 os\calque\calque-app-tsx\backend>
```

Démonstration

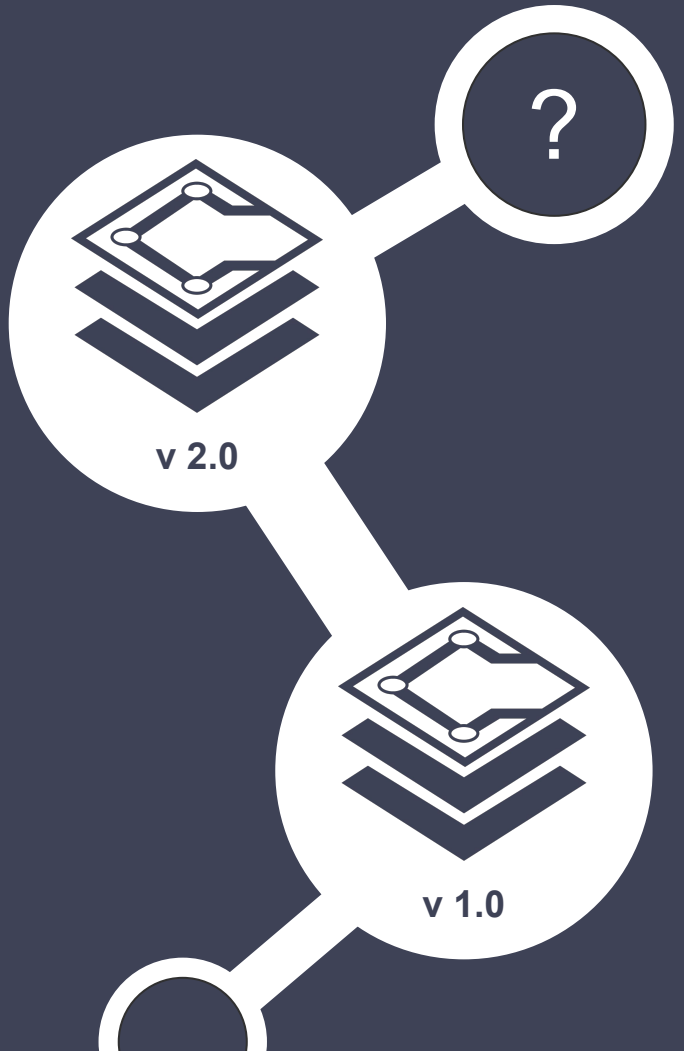
- Créer une carte
- Revenir en arrière
- Continuer la carte
- Exporter la carte
- Utiliser la carte exportée

Déploiement du Frontend sur https://calque.netlify.app/



Roadmap

- Étages / layers
- Étiquettes (Tags)
- Descriptions plus détaillées
- Comptes d’utilisateurs
- Zoom / Pan, Canvas Infini
- Documentation / Page d’Aide
- Responsivité



Conclusion

- Projet facilite la création et l'interaction avec des cartes variées.
- Constitue une solide base mais très ambitieux pour une session
- Développement du Frontend en majorité
- Ajout de nouvelles fonctionnalités dans des itérations futures

➢ Ce projet a permis d'explorer toutes les étapes de création d'une application, de l'élaboration des exigences à la programmation, en passant par le prototypage et le travail d'équipe, des compétences essentielles pour le marché du travail.

Merci pour votre écoute!

Avez-vous des Questions?