ILEVIA simulator

Conduisez l'innovation, simulez l'avenir.

Sommaire :

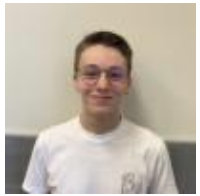
Problématique :

Après un appel d'offres, Ilevia proposait une refonte de leur système de monitoring. Notre équipe a répondu à cette offre, et nous allons vous présenter le projet.

Le projet

ILEVIA Simulator est une solution qui permet de gérer votre flotte de métro et les incidents aléatoires pouvant se produire dans les différentes stations. Son interface simple offre une meilleure lisibilité sur vos lignes de métro, ce qui facilite leur gestion. Elle intègre un système de notification en cas de problème sur l'une de vos stations et permet également d'assigner un agent en quelques clics.

Présentation de l’ équipe



Baptiste Longuepee : Développeur Full-stack

Vianney Basquin : Développeur Full-stack

Thibaut GABET : Développeur Full-Stack

Théotime Pagies : Designer, Développeur Full-Stack

Les réalisations de notre équipe sur ce projet

Baptiste LONGUEPEE

Sur ce projet, j'ai travaillé sur la conception de la page d'accueil ainsi que sur les fonctionnalités associées. Cela inclut l'implémentation de la base de données nécessaire à l'affichage des stations. J'ai également développé un système de changement de couleurs destiné à signaler les problèmes rencontrés sur les différentes stations. Enfin, j'ai amélioré le design pour une meilleure lisibilité.

Théotime Pagies

Travail réalisé :

- Maquette de l'application : UX/UI, animations.
- Partie Back-end : création de la base de données, requêtes SQL et envoi des données en front.
- Refactoring : modifications des données pour les utiliser côté front à partir du back.
- Création de la partie agent : SlideBar avec sélection des agents, assignation des agents à une station et indication de leur disponibilité.
- Popup : alerte en cas de problème aléatoire.
- Récupération des notifications et affichage.
- Animation sur la page d'accueil du métro.
- Animation des icônes de métro sur la page d'accueil et mise en place du système d'images.
- Design global.

Thibaut Gabet

Je me suis occupé de l'initialisation du projet en proposant une technologie moderne et facile à prendre en main, qui convenait à l'équipe et au type de projet que nous souhaitions développer.

J'ai développé la page d'affichage d'une station comportant les bornes qui lui sont attribuées, affichant également le niveau d'encre de chaque borne. Cette interface offre la possibilité de visualiser et de recharger le niveau d'encre d'une borne lorsque cela est nécessaire.

J'ai également pris en charge le développement des fonctionnalités et la gestion des données, depuis la base de données jusqu'à l'affichage des bornes pour une station spécifique, avec la possibilité de remettre son niveau d'encre à 100%.

Basquin Vianney

Sur ce projet, j'ai réalisé la création du MCD, MLD et des différentes données pour la base de données, telles que l'ajout de chaque station de métro en fonction de leurs noms et de leurs positions sur chaque ligne, ainsi que des bornes.

J'ai également implémenté une fonctionnalité sur les bornes permettant la réinitialisation du niveau de papier afin de simuler l'action du changement de papier, comme cela est nécessaire en situation réelle.

Technologies utilisées :

Utilisation de Tauri : Tauri est un framework pour développer des applications de bureau avec n'importe quel framework frontend et un backend en Rust.
Langages : Rust et Angular, SQL, HTML, CSS, Typescript.

Pourquoi utiliser Tauri ?

- Technologie Web.
- Taille légère donc facilite les installations.
- Sécurité : Utilisation du langage Rust qui est connu pour éviter les erreurs de mémoires et les vulnérabilités courantes.
- Multiplateforme.
- Application en couche : séparation du front et du back.

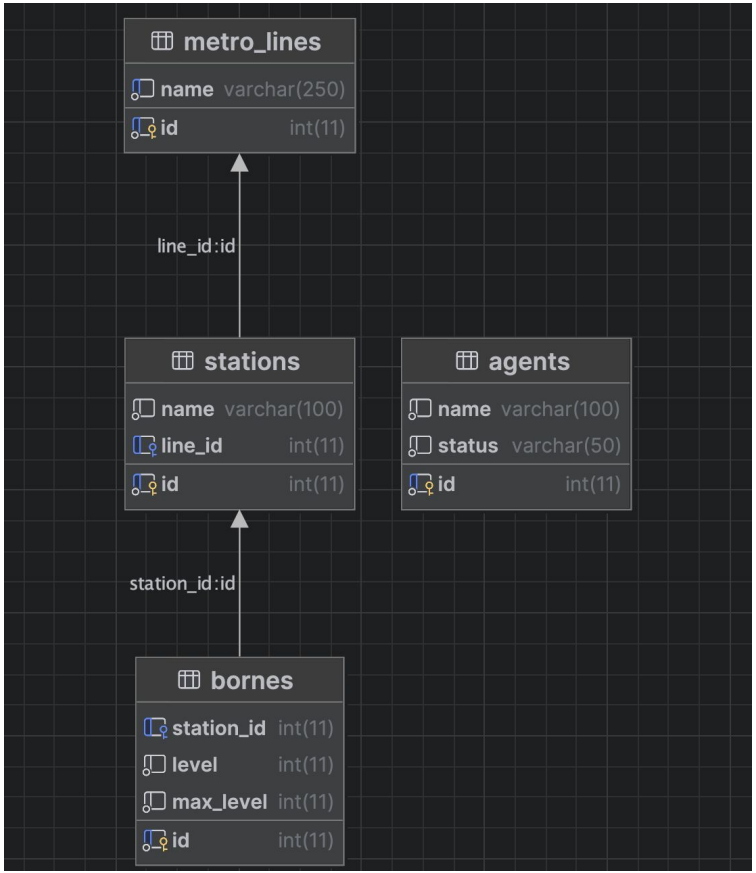


Table metro_lines :

id : Clé primaire (int(11))
name : Nom de la ligne (varchar(250))
Table stations :

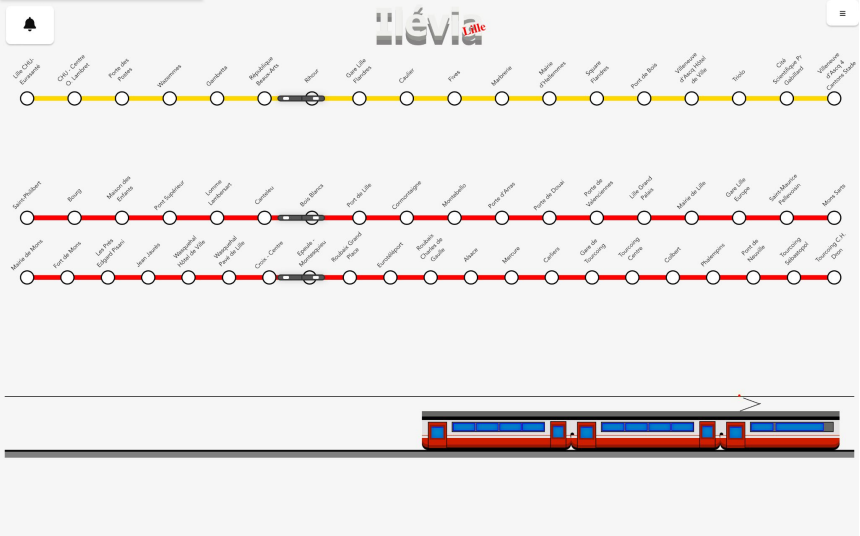
id : Clé primaire (int(11))
name : Nom de la station (varchar(100))
line_id : Clé étrangère référençant id de la table metro_lines (int(11))
Table bornes :

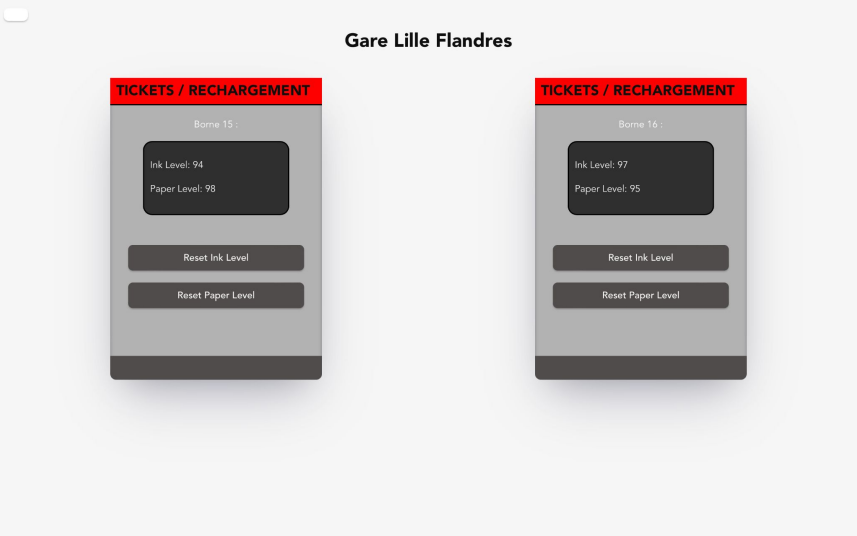
id : Clé primaire (int(11))
station_id : Clé étrangère référençant id de la table stations (int(11))
level : Niveau actuel (int(11))
max_level : Niveau maximal (int(11))
Table agents :

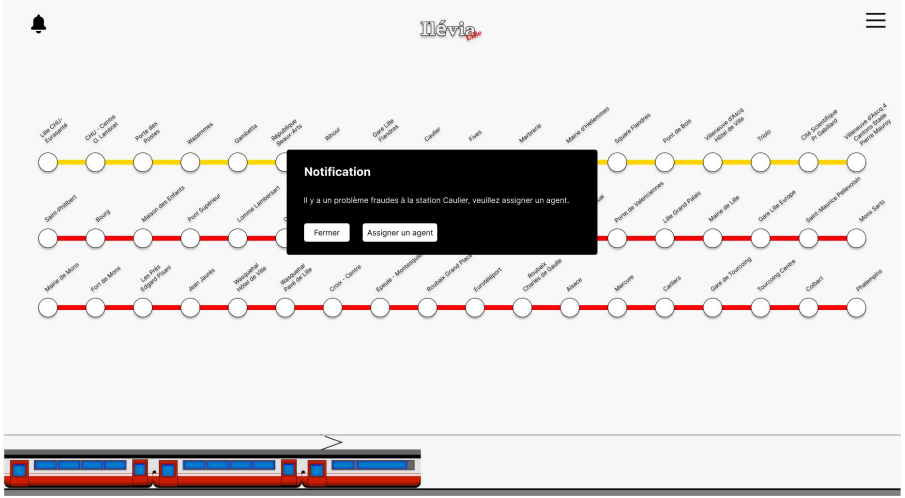
id : Clé primaire (int(11))
name : Nom de l'agent (varchar(100))
status : Statut de l'agent (varchar(50))
Relations :

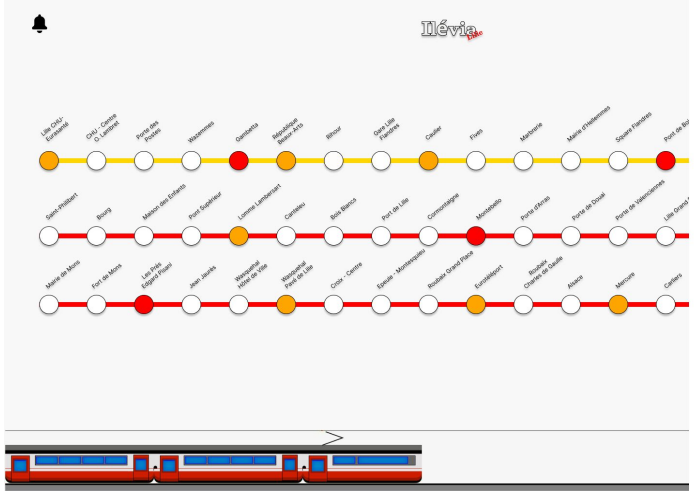
metro_lines à stations : Une ligne de métro peut avoir plusieurs stations (relation 1-n).
stations à bornes : Une station peut avoir plusieurs bornes (relation 1-n).

Aperçu du projet :






Lévia
Notification
Il y a un problème fraudes à la station Caulier, veuillez assigner un agent.
Fermer
Assigner un agent


Lévia


Sélectionner un agent :
Voir la disponibilité
Voir la disponibilité
Voir la disponibilité
Voir la disponibilité
Sélectionner une station
Valider
Statut de l'agent :

Liens utiles :

Le repository du projet : https://github.com/B3DevSubwaySurfer/subwaysurfer

La maquette : https://www.figma.com/file/boMAmcYyRdGsGGOidhRKWF/SubwaySurfer?type=design&node-id=0-1&mode=design&t=V1GeJCAOHejd6yxc-0