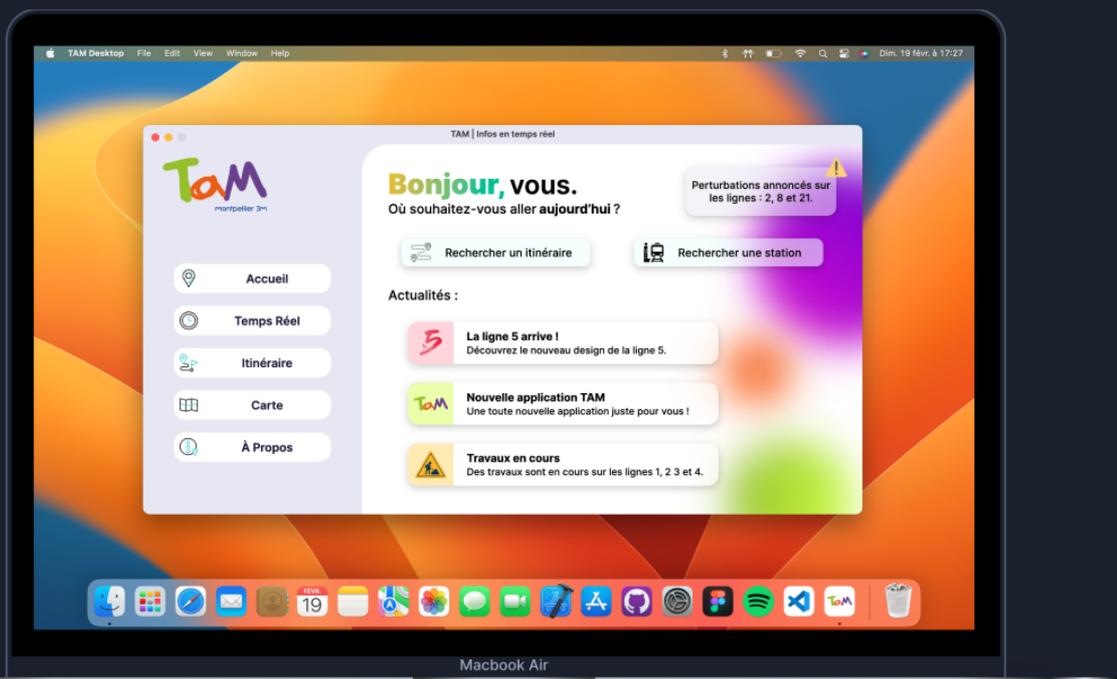


Découvrez, TaM Desktop.



Ergonomique, simple et rapide.



> PRÉSENTATION GÉNÉRALE :

L'idée générale de ce projet est d'offrir aux utilisateurs une application simple, efficace, gratuite et intuitive pour le réseau de transport montpelliérain.

Elle permet à l'utilisateur de pouvoir se renseigner en un clic sur les horaires d'une ligne ou d'une station de son choix, le tout en temps réel.

Nous avons aussi programmé un algorithme de recherche qui donne une liste d'itinéraires en fonction des stations de départ et d'arrivée entrées par l'utilisateur.

L'application est construite sur Electron JS, un « *framework* » permettant de développer des applications multi-plateformes de bureau. Elle est entièrement codée en HTML, CSS et JavaScript. Nous avons utilisé ces langages afin de proposer une bonne interface graphique tout en restant dans les langages appris dans le programme de NSI.

Le but de cette application n'était pas dans un premier lieu de proposer plus de fonctionnalités déjà présentes mais plutôt d'améliorer une application pour la rendre plus fonctionnelle et intuitive que l'application déjà existante. (voir captures d'écran à la fin)

Elle est aujourd'hui disponible sur Windows, Linux et MacOS. Dans un futur proche elle sera portée sur les plateformes iOS et Android. Elle disposera aussi de sa propre API en temps réel afin d'accroître la rapidité des temps de requête et d'améliorer la qualité de l'API.

> ORGANISATION DU TRAVAIL :

L'équipe :

- Louis RAVERDY : Chargé de développement, développeur et UI/UX designer.
- Lucas REVERBEL : Développeur back-end.

Lucas s'est chargé de toute la partie back-end, l'algorithme de recherche et de destination. Louis s'est occupé de l'affichage et le design des pages, le lien vers l'API et le lien entre le back-end et le front-end.

Depuis septembre maintenant, nous travaillons d'arrachepied pour proposer le meilleur résultat possible pour cela, nous collaborons grâce à GitHub pour le partage du code, nous supervisons nos tâches via Trello et communiquons plus que possible en présentiel ou sur Discord si ce n'est pas possible. Louis a aussi fait ses maquettes sur Figma.

LES ÉTAPES DU PROJET :

Rentrée septembre 2022 : émergence de l'idée à la suite d'un réel besoin d'une application simple et esthétique.

Mi-septembre 2022 : Louis met en place le projet, les tâches à faire et commence à faire des maquettes pour chaque page.

Octobre 2022 : Lucas rejoint le projet et s'attaque à l'algorithme de recherche des stations et des lignes. Louis code le système de fenêtres et commence à coder les pages en HTML et CSS.

Fin novembre 2022 : Notre MVP est officiellement terminé malgré un grand nombre de bugs.

Fin décembre 2022 : L'application est enfin debout ! Une grande partie des bugs ont maintenant été résolus. Toute la partie temps réel est désormais fonctionnelle mais quelques bugs sont encore persistants.

Mi-février 2023 : Refonte du back-end et des liens vers l'API, améliorations de l'algorithme de recherche pour des résultats plus pertinents. Nous avons aussi commencé à coder toute la partie sur l'itinéraire et mis en place une carte interactive ainsi qu'un système de mise à jour qui permettra de garder à jour le logiciel une fois publié.

Avancement :

- Accueil : (100%)
- Temps Réel : (100% mais susceptible de contenir des bugs)
- Itinéraire : (backend 70% et front-end 100%)
- Carte : (100%)

> FONCTIONNEMENT ET OPÉRATIONNALITÉ :

Nous avons mis en place des séances de « bug Bounty » afin de prendre connaissance d'un maximum de bugs. Nous avons aussi fait tester l'application à des amis et proches pour tester l'accessibilité et l'ergonomie de l'app.

Nous avons rencontré un bon nombre de difficultés, la plupart venant de la qualité de l'API vraiment médiocre. Pour pallier cela, nous avons mis un grand nombre de sécurités permettant d'éviter tout bug ou résultat non voulu de l'application notamment en refaisant une partie de l'API et en conditionnant les résultats reçus.

La plupart des autres bugs venaient de notre connaissance en JavaScript très faible au début de ce projet. Cela nous a donc permis d'approfondir nos connaissances en JavaScript notamment en revisitant les parties au programmes tel que les algorithmes sur les graphes, le traitement des données, la modularité et la complexité déjà vus en Python.

> OUVERTURE :

L'application que nous avons développée avait pour but initial d'être disponible sur iOS et Android. Cependant, pour des raisons de simplicité, nous avons décidé de commencer par la version desktop. Dans les mois à venir, notre objectif est de porter l'application sur mobile.

Nous avons également constaté que l'API était de très mauvaise qualité, avec des informations inutiles et une organisation peu claire. Louis travaille actuellement à une refonte complète de cette API.

Nous aimerions proposer cette application à la TAM afin de devenir l'application officielle pour un grand nombre d'utilisateurs. Nous sommes en mesure de le faire car toutes les bibliothèques utilisées sont libres de droit et les données collectées respectent les réglementations RGPD.

Si nous devions refaire ce projet, nous commencerions peut-être par recréer l'API afin d'économiser du temps et d'éviter des découragements. Cependant, malgré les difficultés rencontrées, nous sommes parvenus à obtenir un résultat satisfaisant.

DOCUMENTATION

Spécifications techniques :

OS : Windows, Linux et MacOS (prochainement iOS et Android)

Langages : HTML, CSS et JavaScript

Bibliothèques : ElectronJS, ElectronUpdater (système de MàJ), ElectronBuilder (package)

Serveurs : Firebase (prochainement Google Analytics)

Configuration minimale : connexion internet, OS compatible.

Guide d'utilisation :

Etape 1 : Lancez le *TAM-Desktop-Setup-x.x.x.exe*.

Etape 2 : Suivez le programme d'installation.

Etape 3 : Lancez TAM Desktop.exe

Etape 4 : Enjoy !

Cinq onglets vous sont disponibles :

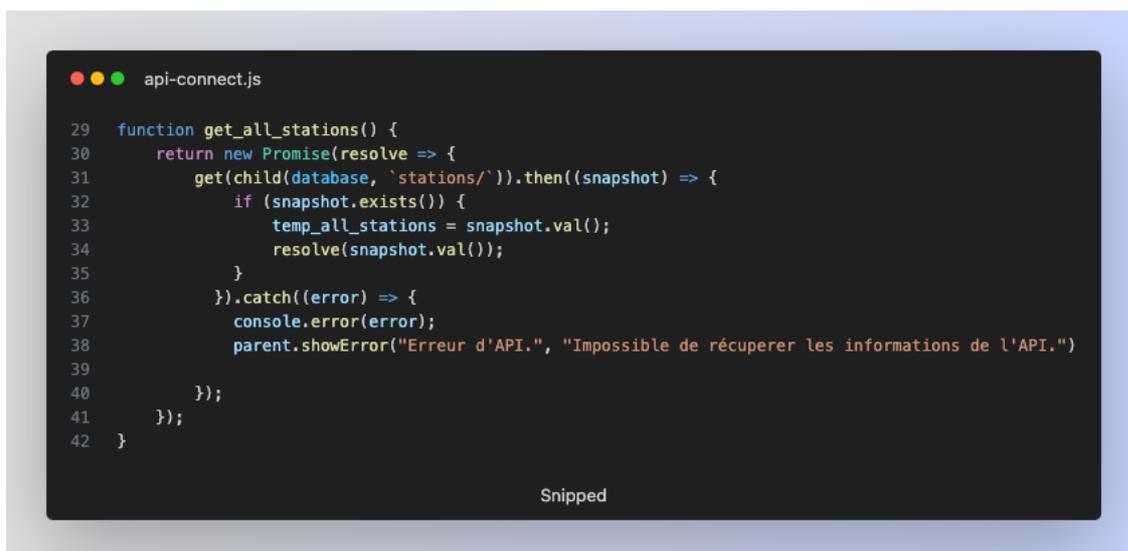
L'accueil : vous permet d'avoir des accès rapides aux fonctionnalités importantes de l'app ainsi que des actualités.

Le temps réel : vous permet de vous renseigner en un clic sur la station ou la ligne de votre choix.

L'itinéraire : vous permet de vous guider pour aller d'un point A à un point B le plus rapidement.

La carte : découvrez l'étendu du réseau de transport de la métropole. Elle vous permet aussi de vous renseigner sur un arrêt ou une ligne.

Le à-propos : vous donne accès aux crédits de l'app ainsi qu'aux mentions légales.



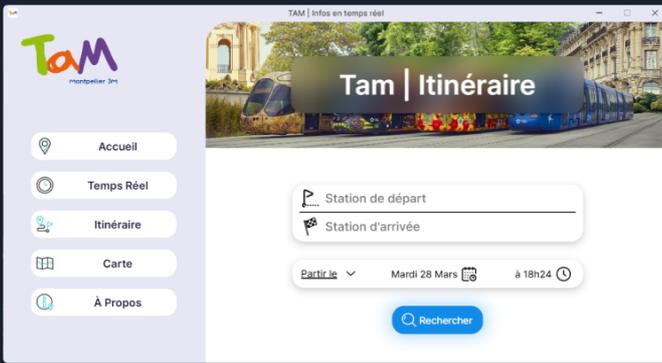
```
api-connect.js

29 function get_all_stations() {
30   return new Promise(resolve => {
31     get(child(database, `stations/`)).then((snapshot) => {
32       if (snapshot.exists()) {
33         temp_all_stations = snapshot.val();
34         resolve(snapshot.val());
35       }
36     }).catch((error) => {
37       console.error(error);
38       parent.showError("Erreur d'API.", "Impossible de récupérer les informations de l'API.")
39     });
40   });
41 });
42 }
```

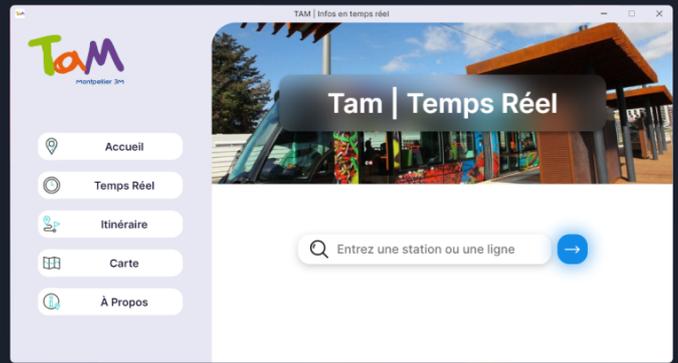
Snipped

Exemple de code pour récupérer des informations de l'API.

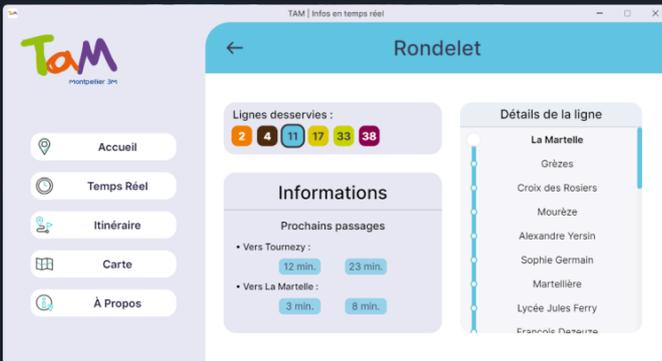
Captures d'écran Desktop



Page itinéraire



Page temps reel

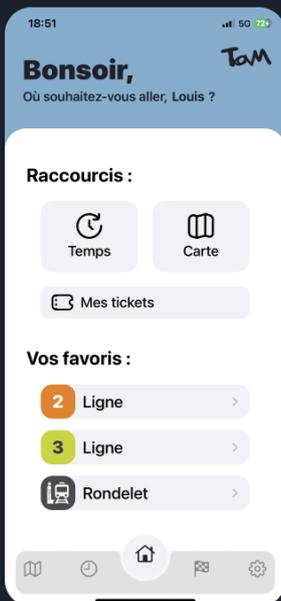


Page station rondellet

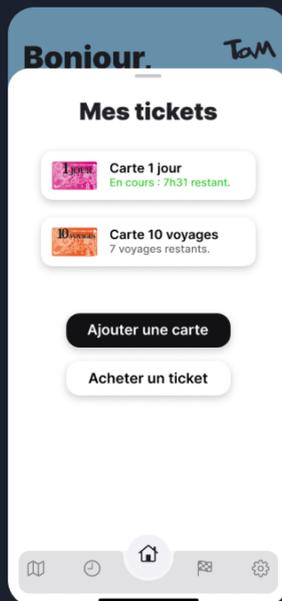


Page de la ligne 3

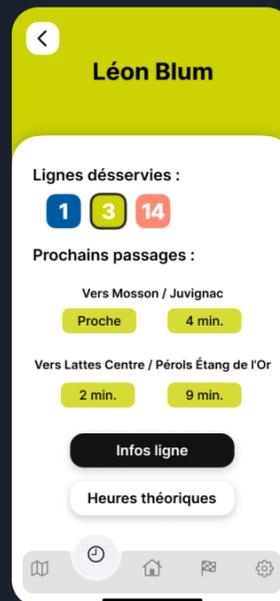
Maquette iOS



Page d'accueil



Page des tickets

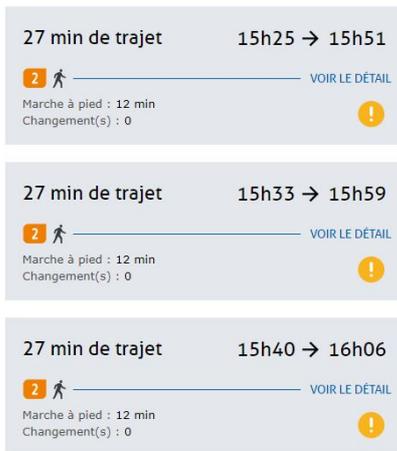


Page temps reel



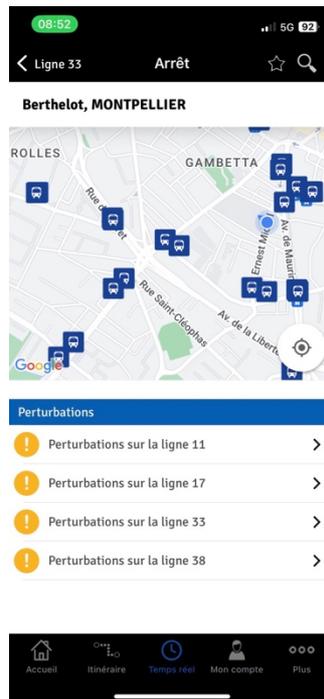
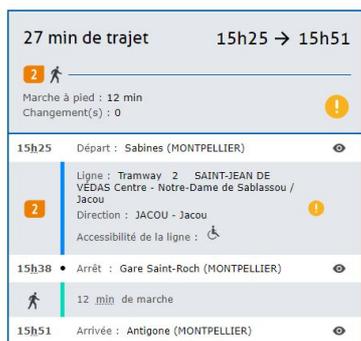
Page temps reel

Report de comportements non voulu sur l'application/site officielle :



Votre type de trajet :

- le plus rapide
- le moins de marche à pied
- le moins de changements
- le plus économique



L'app nous affiche les perturbations mais aucun temps.

Le dark mode n'est pas supporté sur cette page + impossible de zoomer sur la carte, le zoom opère sur la page entière enlevant les éléments de navigation.

L'itinéraire nous propose 3 fois le même trajet ou il est demandé le moins de marche à pied. Il propose donc 3 fois la même solution très peu optimale avec 12 min de marche alors qu'il existe un arrêt à 2 min.