Documentation technique Node.js

Remindr

Année universitaire : 2023-2024

Auteurs :

IMBAUD Florian - Étudiant en seconde année BUT Informatique Le Puy-en-Velay (Groupe B2)
LABROSSE Olivier - Étudiant en seconde année BUT Informatique Le Puy-en-Velay (Groupe B2)

A jour en date du :

22/12/2023

Sommaire de la documentation :

© IMBAUD Florian - LABROSSE Olivier

Choix technologiques

Node.js

Nous avons utilisé, pour l'entièreté du projet, l'environnement Node.js, permettant l'exécution de code JavaScript côté serveur, mais également côté client. Cela facilite la programmation en évitant d'utiliser 2 langages différents entre le côté client et le côté serveur.

Express.js

Nous avons également utilisé le framework Express.js lors de ce projet. Cela nous a permis d'avoir accès à une documentation et une communauté étendue afin de résoudre d'éventuels bugs. Le framework Express est également conçu avec des middlewares permettant de créer l'application correctement par rapport à ce que nous voulions faire. Il existe également un grand nombre d'extensions avec NPM qui nous a permis de faire évoluer continuellement le projet.

Son système de routes et de routeurs nous a permis de pouvoir gérer convenablement les différentes URL saisies ou sur lesquelles nous avons réorienté l'utilisateur.

Prisma.js

Nous avons utilisé Prisma.js pour la création, la gestion et la modification de la base de données. Cela a permis d'avoir une base optimisée, et de pouvoir gérer directement la base de données en JavaScript de la meilleure des manières possible dans notre cas, sans avoir à gérer des contraintes ou choses plus complexes, qui directement gérées par Prisma.js lors de la création de la table par l'envoi du fichier schema.prisma.

Supabase

Supabase est le site hébergeant la base de données créée par Prisma.js. Ce dernier était recommandé par Prisma, et nous a permis d'éviter les problèmes de connexions que nous avions avec des systèmes comme AlwaysData. Le fait d'avoir une base directement hébergée en ligne nous a également facilité le travail d'un point de vue concordance de données, en nous permettant de tester notre code en temps réel directement avec plusieurs ordinateurs et utilisateurs créés.

© IMBAUD Florian - LABROSSE Olivier

Handlebars

Handlebars est le système utilisé afin d'avoir des pages web dynamiques dans lesquelles nous pouvons, via Prisma.js et le JavaScript, choisir d'afficher telle ou telle information en fonction de la page, de l'utilisateur actuellement connecté, et ainsi d'avoir des informations cohérentes affichées.

© IMBAUD Florian - LABROSSE Olivier

Structure du projet

Controllers

- authController.js : il contient l'ensemble des fonctions utilisées par le Routeur de gestion d'authentification.
- groupControlleur.js : il contient l'ensemble des fonctions utilisées par le Routeur de gestion des groupes.
- mainController.js : il contient l'ensemble des fonctions utilisées par le Routeur de gestion de pages principales (Racine du site et dashboard).
- reminderController.js : il contient toutes les fonctions utilisées par le Routeur de gestion des rappels.

Middlewares

- addreminderingroupe.js : il s'occupe de l'ajout des rappels au sein des groupes créés.
- adduseringroup.js : il s'occupe d'ajouter un utilisateur dans le groupe, en vérifiant que ce dernier existe, et qu'il n'est pas déjà présent dans le groupe.
- bodyparser.js : il s'occupe des entrées utilisateurs dans les formulaires POST et les rend utilisables dans les fonctions nécessaires.
- creategroupe.js : il s'occupe de créer le groupe avec un nom unique, s'assurant que ce dernier n'est pas déjà utilisé. Il ajoute également le créateur dans le groupe.
- deletereminder.js : il supprime le rappel dont-il est question quand le bouton de cette action est utilisé par le créateur du rappel.
- login.js : il s'occupe de l'authentification de l'utilisateur, en vérifiant l'identifiant (l'adresse mail) et le mot de passe haché.
- register.js : il s'occupe de la création de l'utilisateur dans la base de données, en utilisant son nom, son prénom, son adresse mail et son mot de passe que l'on hache pour l'insérer dans ladite base.
- updatereminder.js : il s'occupe de modifier le rappel créé quand le bouton de cette action est utilisé par le créateur du rappel.

Prisma

- schema.prisma : ce fichier contient la gestion de la base de données via Prisma.js.

Fichiers de styles (CSS)

- groupe.css : il contient le style de la page de groupe.
- styles.css : il contient le style commun à plusieurs pages du site.

© IMBAUD Florian - LABROSSE Olivier

Routeurs

- authRouteur.js : il s'occupe de la gestion des routes liées à l'authentification de l'utilisateur, la création de compte, ou encore la déconnexion.
- groupRouteur.js : il s'occupe de la gestion des routes liées à l'accès aux groupes, à l'ajout d'utilisateurs dans un groupe, et à la création d'un groupe.
- mainRouter.js : il s'occupe de la gestion des routes principales telles que la racine du site ainsi que le dashboard.
- reminderRouter.js : il s'occupe de la gestion des routes liées aux rappels, à leur ajout, leur modification par le créateur du rappel, ainsi que par la suppression du rappel par le créateur du rappel.

Templates Handlebars et pages fixes HTML

- dashboard.hbs : il contient le dashboard de chaque utilisateur, affichant ses groupes, et lui permettant d'en créer de nouveaux. Il voit également ses rappels arrivant à échéance prochainement.
- groupe.hbs : il contient la page de chaque groupe, affichant les rappels du groupe, ainsi que les boutons permettant l'ajout d'un utilisateur dans le groupe, et bien évidemment la création d'un rappel. Chaque rappel peut être modifié ou supprimé par son auteur. Il y a également un affichage des couleurs choisies pour chaque rappel, et un changement des bordures du rappel en fonction de la proximité dans le temps de l'échéance du rappel.
- login.html : elle contient la page de login de l'utilisateur, lui permettant de se connecter avec son adresse mail en tant qu'identifiant et son mot de passe qui est bien évidemment haché dans la base de données pour plus de sécurité. Il a également la possibilité d'aller sur la page de création de compte s' il n'en a pas encore un.
- register.html : elle contient la page de création de compte, en demandant le nom de famille, le prénom, l'adresse mail de l'utilisateur ainsi que son futur mot de passe, qui sera haché avant d'être inséré dans la base de données. Il faut savoir que la même adresse mail ne peut pas être utilisée deux fois sur le site, car elle sert d'identifiant unique pour l'application.

Sécurité

Dans l'entièreté du projet, des vérifications sont faites pour vérifier si l'utilisateur est toujours connecté (si sa session est toujours valide) lorsqu'il fait des actions. De plus, en ce qui concerne les groupes et l'affichage de la page associée, nous vérifions dans la base de données si l'utilisateur est dans ce groupe demandé, afin de prévenir de toute tentative d'intrusion par saisie d'URL sans être présent dans ce dernier.

© IMBAUD Florian - LABROSSE Olivier

Détails de mise en œuvre

Mise en œuvre pratique côté serveur

Nous avons utilisé Gitpod tout au long du projet, afin d'avoir une réelle interface serveur simple d'utilisation sur plusieurs machines directement.
Les dépendances et extensions utilisées peuvent être directement installées par le biais de la console de l'espace de travail créé par Gitpod. Le code JavaScript côté serveur.

Mise en œuvre pratique côté client

Nous avons donc utilisé l'hébergement local de l'espace de travail Gitpod lors de nos tests et de notre avancement global afin d'avoir une simplicité d'accès aux pages web.
Il suffit donc au poste client de posséder un navigateur, et d'entrer l'URL de localhost donné par Gitpod, sans avoir besoin d'installer d'autre chose.

Mise en œuvre communicationnelle

Nous avons pu utiliser GitHub lors l'entièreté du projet pour la communication du code, avec des livraisons de différentes branches créées sur la branche principale. Le maître mot a été de garder cette branche principale fonctionnelle tout au long du projet, afin d'éviter de travailler sur cette branche principale et de corrompre le projet déjà avancé jusqu'à l'étape définie. Chaque branche avait donc un but d'ajout de fonctionnalité à la branche principale pour enfin arriver où nous en sommes aujourd'hui. Le versionnage permis par GitHub nous a également servi dans le cas de livraisons de code sur la branche principale qui entre en conflit avec cette dernière, pour revenir à une version antérieure et résoudre les problèmes avant de retenter la livraison finale.

En ce qui concerne la communication directe, nous avons pu nous voir en présentiel au sein de l'IUT, ou nous avons également pu communiquer textuellement par le biais de Discord.

Cela nous a donc permis, en globalité, d'avancer convenablement et de concert pour ne jamais faire deux fois la même chose en même temps sans le savoir, et ainsi garder un avancement constant dans le projet, du début jusqu'à la fin.

Nous avons également créé un fichier permettant de répartir les différentes tâches à effectuer et leurs états (en cours ou terminé)

© IMBAUD Florian - LABROSSE Olivier

Mise en œuvre humaine

Le projet effectué nous paraissait globalement immense dès le début, mais nous a permis d'apprendre de nombreuses choses, que ce soit sur Node.js, Prisma.js ou encore Express.js et toutes ses extensions possibles. Cela a également permis d'avoir un projet liant pratique de programmation avec communication globale, afin d'avoir un très léger aperçu d'un projet qui pourrait être donné au sein d'une équipe d'une entreprise.

© IMBAUD Florian - LABROSSE Olivier