

Projet UNsalib - White paper

I Les origines du projet

1.1 Contextualisation

L'année de L1 commencée, il nous est vite apparu nécessaire de travailler en groupe au sein de "séances de révision". En effet, le travail régulier est indispensable en licence pour réussir.

Pour cela, nous devions disposer d'un espace où les membres de la classe 185 pourraient se retrouver pour étudier ensemble, partager des connaissances et se préparer aux évaluations. Cependant, malgré la mise à disposition de certains locaux par Nantes Université, trouver une salle disponible peut parfois s'avérer compliqué. Pour éviter de perdre du temps à chercher, nous sommes trois étudiants à nous être associés pour créer une solution entièrement nouvelle : UNsalib.

1.2 Présentation des objectifs

UNsalib est tout simplement un site accessible depuis n'importe quel appareil permettant de consulter, en temps réel ou à l'avance, les salles libres sur le campus de la faculté des Sciences et Techniques (campus Lombarderie). Nous avons aussi décidé d'y intégrer la possibilité de consulter l'emploi du temps d'une salle précise.

II La conception

2.1 La réflexion

Pour commencer, nous avons dû évaluer la faisabilité du projet. En effet, inutile de commencer à coder si nous ne pouvions pas avoir accès aux données d'emploi du temps de l'Université. Heureusement, ces données sont en libre accès, sans authentification nécessaire, ce qui nous a grandement facilité la tâche. Nous avons donc ensuite décidé des outils que nous allions utiliser et du fonctionnement global du projet. Il a été décidé que le projet serait codé en JavaScript et reposera sur NodeJS.

Comme beaucoup de services webs, UNsalib est basé sur deux parties essentielles qui fonctionnent de concert : le backend et le frontend.

2.2 Le backend

2.2.1 Récupération des emplois du temps

La première étape a consisté à récupérer les emplois du temps des différentes classes du campus. L'API de Nantes Université fournit les cours et leurs informations au format JSON avec une simple requête incluant en paramètres l'ID du groupe concerné, et des bornes temporelles. Cependant, il nous a fallu d'abord charger virtuellement la page des emplois du temps au format HTML afin de parcourir le DOM à la recherche des IDs de tous les groupes existants.

Puis nous n'avons eu qu'à récupérer en boucle les données JSON pour chacun d'eux et les stocker dans une base de données pour y accéder à tout moment de indépendamment de l'Université.

Pour cela, nous avons utilisé MongoDB, un système de gestion de bases de données performant. Notre base s'articule autour de quatre tables : Bâtiment, Cours, Groupe, et Salle. La table des cours, par

exemple, regroupe les informations sur les horaires, le nom du professeur, le groupe, le module, la classe et un identifiant. La table des groupes contient, elle, un identifiant et un nom. Quant aux salles, nous stockons leurs noms, le bâtiment où elles se situent et des informations supplémentaires, comme le nombre de places assises, pour une future fonctionnalité.

Bien que tout ce processus semble rodé, nous avons dû revoir plusieurs fois notre système de recherche de salles libres, qui s'avère assez complexe.

```
// Recherche de tous les cours qui débordent sur la période demandée selon 4 cas :  
//  
// CAS 1 : Le cours englobe complètement la période  
// Cours      |-----|  
// Demande    |-----|  
//  
// CAS 2 : Le cours est englobé par la période  
// Cours      |-----|  
// Demande    |-----|  
//  
// CAS 3 : Le cours chevauche le début de la période  
// Cours      |-----|  
// Demande    |-----|  
//  
// CAS 4 : Le cours chevauche la fin de la période  
// Cours      |-----|  
// Demande    |-----|  
//
```

Un exemple des cas à gérer pour rechercher une salle libre

Bien sûr, il était hors de question de perturber le fonctionnement de l'Université en surchargeant le site d'emplois du temps. Nous avons donc réparti toutes les requêtes sur un jour de manière à ce qu'elles passent inaperçues et en même temps nous permettent de disposer d'une base de données à jour.

Ces opérations sont totalement automatisées. La récupération de groupes se lance une fois seulement au démarrage du serveur tandis que l'extraction des emplois du temps et leur stockage est quotidienne.

2.2.2 API

Nous avons dû prévoir la recherche des utilisateurs, nécessitant des fonctionnalités pour extraire des informations de la base de données en fonction de la requête. Pour cela, nous avons mis en place plusieurs endpoints, des segments d'URL renvoyant des données spécifiques. Un endpoint global et deux autres plus ciblés sont disponibles :

- /salles, qui affiche toutes les salles et leurs informations
- /salles/disponibles, qui liste les salles libres pour une période donnée (avec date de début et de fin)
- /salles/edit, qui renvoie l'emploi du temps d'une salle pour une semaine précise (par défaut, la semaine en cours)

2.3 Le frontend

2.3.1 Prototype

Une fois le backend terminé, ou en tout cas suffisamment avancé, nous avons développé une Interface Homme Machine (IHM).

Pour commencer, nous avons prototypé l'apparence du site en réalisant une page HTML statique, avec des données exemples, complétée par une feuille de style CSS. Nous avons bien sûr pris en compte les divers formats d'écran et usages (ordinateurs et smartphones).

2.3.2 Réalisation d'une interface dynamique

Puis petit à petit, nous avons rendu fonctionnelle cette page en ajoutant du JavaScript. Grâce à ceci, nous avons pu animer les boutons, la recherche et effectuer des requêtes à notre API pour ensuite afficher dynamiquement les emplois du temps.

Au final, on peut rechercher depuis l'interface l'emploi du temps d'une salle en saisissant son nom, avec une liste de résultats qui se précise en temps réel. L'emploi du temps s'affiche ensuite sous forme de calendrier, en fonction de la semaine sélectionnée (par défaut, celle en cours). On peut aussi chercher une salle libre en précisant une date et un créneau horaire.

2.3.3. Améliorations

Bien sûr, il nous est arrivé de découvrir en codant le frontend que des fonctionnalités manquaient côté backend. Nous avons donc amélioré également celui-ci au fur et à mesure que nous développions le client.

De plus, nous avons dû faire face à des bugs plus compliqués que prévu, par exemple pour l'affichage de l'indicateur d'heure, ou encore sur les onglets pour la recherche de salles.

III L'organisation en équipe

3.1 Répartition des tâches

Pour mener à bien ce projet, le travail a dû être réparti entre nous. Maxence, qui possédait plus de compétences en JavaScript, a aidé le reste du groupe pour démarrer la réalisation du backend. Puis Maël a pris le relai pour cette partie du code. Maxence a donc pu se concentrer sur le développement du frontend. Cette segmentation ne les a pas empêchés d'effectuer des petits changements et corrections sur le code de l'un ou l'autre. Enfin, outre son aide apportée à la réflexion algorithmique et au développement de quelques fonctionnalités, Ethann s'est occupé de la cohérence des données de la base et a également participé à la rédaction de ce document, assisté de Maël.

3.2 Outils de collaboration

Lors de l'élaboration de UNsalib, nous avons utilisé un repository GitHub pour partager nos avancées respectives. Nous avons aussi essayé Github Codespace, mais nous avons rencontré beaucoup de problèmes techniques avec celui-ci, ce qui nous a finalement convaincu de produire notre code localement sur un ordinateur, avant de l'envoyer sur le repository.

Maxence, Ethann et Maël (mis à jour en novembre 2024)

Note postérieure : les détails techniques délivrés ici ne reflètent plus en totalité le fonctionnement réel de l'application