Université de Sherbrooke

IFT 717 - Application Internet & Mobilité

Projet de Session

Stay Awake - Reste éveillé

Thomas Cenci - cent2801

Timothy Le Pallec - lept3101

Laura Scolan - scol2901

Introduction

La procrastination est un phénomène visant à repousser volontairement des tâches prévues, nécessaires et importantes même en sachant que les conséquences potentielles en découlant seront négatives. Actuellement, chez les étudiants, des études ont montré que 80 à 90% des étudiants ont des comportements de procrastination pendant, en moyenne, une heure par jour[1]. Un étude approfondie a même mis en avant l'existence d'une procrastination chronique pour 20% de la population occidentale.[2]

Nous-même étudiants, sommes confrontés à la procrastination. C'est pourquoi nous souhaitons, à travers ce projet, créer une application avec client web et mobile permettant d'organiser efficacement ses tâches journalières. L'objectif est de maintenir un bon niveau de productivité et d'organisation au cours de la journée et de ne pas tomber dans la procrastination.

Nous ne souhaitons pas développer une application trop intrusive au risque de rendre l'utilisateur réticent à utiliser notre service. Ainsi, il nous faut traiter les problèmes qu'imposent la procrastination tout en s'assurant de proposer un service non intrusif et ergonomique visant à favoriser l'utilisation de notre application.

Ce rapport vise à vous faire part des procédés mis en place dans le développement de ladite application et nous vous partagerons notre vision du futur de *Stay Awake*.

[1] Klingsieck, K. (2013). Procrastination in Different Life-Domains: Is Procrastination Domain Specific?. Current Psychology, 32(2), 175-185.

[2] Nguyen, B., Steel, P. and Ferrari, J.R. (2013), Procrastination's Impact. Int J Select Assess, 21: 388-399. https://doi.org/10.1111/ijsa.12048

1. Solution proposée

Pour aider à résoudre la problématique exposée dans l'introduction, à savoir augmenter la productivité et réduire la procrastination, nous avons décidé de développer une application possédant un client web et mobile. Cette dernière permettra de s'organiser dès le début de la journée. On retrouvera, pour chaque client, deux sections primordiales : un agenda journalier et un gestionnaire de tâches. Les tâches pourront être glissées dans l'agenda à l'horaire souhaité pour organiser sa journée. Le gestionnaire de tâches permettra d'ajouter, supprimer ou modifier des tâches. Ces dernières seront également classées par ordre de priorité allant de 1 (priorité faible) à 3 (priorité maximale).

L'application regoupera plus précisément les fonctionnalités suivantes :

- un agenda journalier
- la possibilité d'ajouter une tâche dans la section prévue à cet effet (équivalent d'un *todo list*).
- la possibilité de transférer une tâche depuis les *ToDo lists* à l'agenda en la positionnant à l'horaire voulu.
- Dans l'idéal, selon les tâches, l'utilisateur pourra, s'il le souhaite, entrer l'adresse d'un lieu et l'application fournira le service d'itinéraire associé. Des notifications seront donc ajoutées pour notifier l'utilisateur de son heure de départ.

1.1. Fonctionnalités détaillées

1.1.1. *Aperçu journalier et planification des tâches*

Dès l'ouverture de l'application, l'agenda journalier sera affiché sur toute la hauteur de la page. Par défaut, la tranche d'horaire affichée est de 6h à 23h, bien sûr ce paramètre sera modifiable. L'objectif est d'avoir directement un aperçu rapide de ce que l'utilisateur a à faire de sa journée. Il sera aussi possible d'organiser les prochains jours en faisant glisser de gauche à droite l'écran (client mobile) ou en utilisant des boutons prévus à cet effet (client web). Un bouton sera également disponible pour avoir un aperçu des mois, puis pour revenir à la journée courante.

Depuis cette page, l'utilisateur pourra avoir accès à la liste des tâches à effectuer, il pourra ainsi faire glisser les tâches qu'ils souhaitent planifier depuis la liste vers la journée à l'horaire désirée. Les tâches seront alors facilement accessibles en appuyant dessus depuis l'aperçu pour ainsi en consulter les détails et valider la tâche si elle est terminée. Une tâche validée changera automatiquement de couleur.
Une notification sera envoyée à l'utilisateur à l'heure prévue des tâches.

1.1.2. *Gestion des tâches*

La page de gestion des tâches sera accessible depuis le menu de l'application. Elle permettra d'avoir accès à la liste des tâches. Une tâche est définie par un nom, une description, son niveau d'urgence (définit également la couleur de la tâche) et peut également avoir une date de fin afin de pouvoir envoyer des notifications d'alerte. Une tâche est également divisée en deux catégories :

- Les tâches ponctuelles : tâche que l'on peut glisser dans la journée et qu'une fois validée disparaît de la liste des tâches.
- Les tâches permanentes : tâche que l'on fixe sur un jour de la semaine et qui sera toujours affichée depuis l'aperçu journalier.

1.1.3. *Itinéraire*

Une tâche pourra également avoir une adresse qui lui est liée, ainsi il sera possible depuis le détail de la tâche de lancer l'itinéraire en destination du lieu de celle-ci. L'itinéraire utilisera une API gratuite telle qu'OpenStreetMap. On se basera également sur cette api pour l'autocomplétion des adresses lors de la définition d'une tâche.

2. Architecture

2.1. *Architecture globale*

Globalement, l'architecture de notre application est représentée par la figure 1 et correspond à une architecture REST. Nous utilisons un serveur pour communiquer à la fois avec le client mobile, le client android et la base de données pour permettre le transfert, l'ajout et la suppression des données.

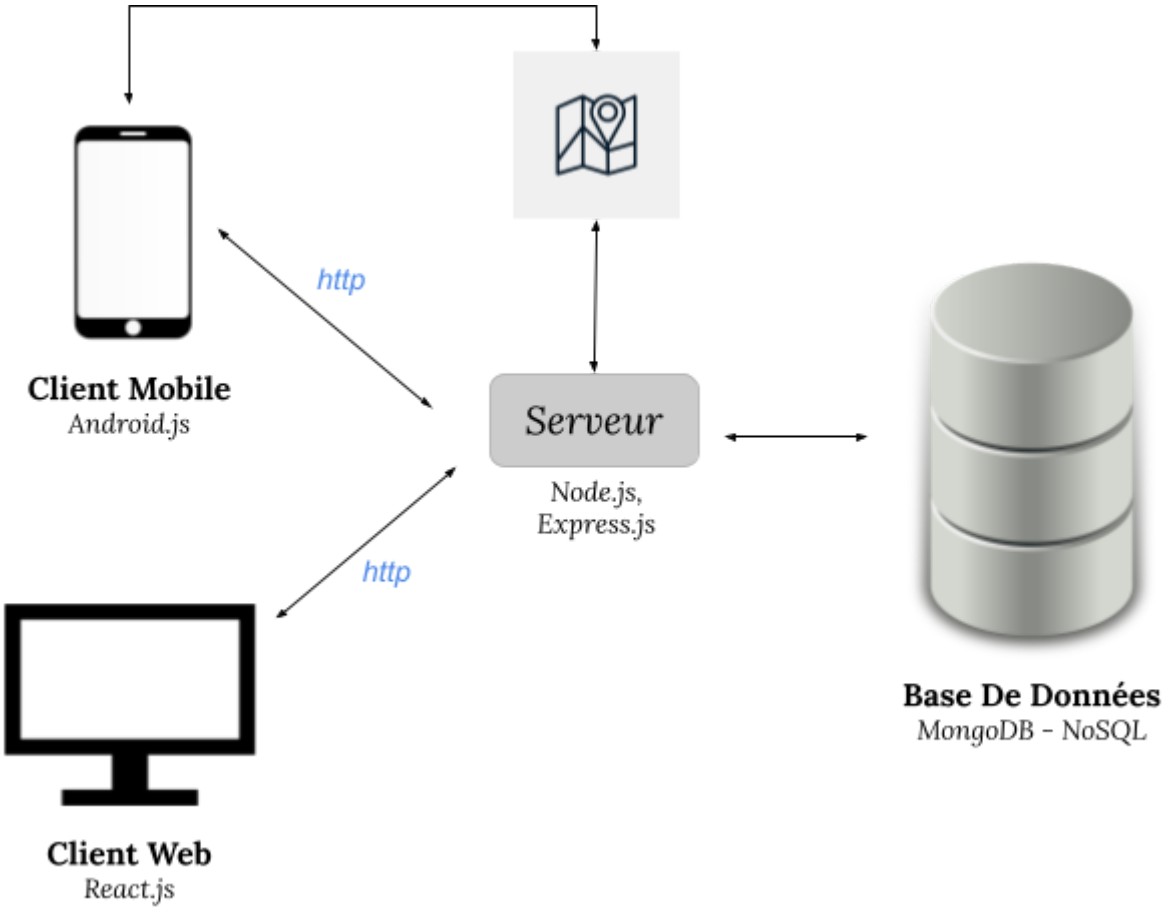

Figure 1 - Architecture globale du système

2.2. *Architecture du client Android*

Le client Android se présente de la manière suivante : une seule Activity qui vient servir de *controller* pour afficher les Fragments désirés. Chaque page de l'application est donc un fragment qui vient s'imbriquer sur l'Activity. Il y a donc deux pages principales : la page du planning avec les tâches à effectuer sur la journée et la page de gestion des tâches qui permet de consulter les tâches, de les modifier, de les supprimer ou d'en ajouter. Ajoutez à cela les modales qui permettent d'afficher les formulaires qui sont également des Fragments, on en dénombre trois : le formulaire d'ajout/modification des tâches, le formulaire de positionnement des tâches et les détails de la tâche avec l'action de valider la tâche.

Les notifications sont gérées par un service qui vient vérifier toutes les heures si une tâche ne doit pas être réalisée dans la demi-heure suivante, si c'est le cas on crée une notification push.

2.3. *Architecture du client Web*

Pour le client web, l'architecture globale est présentée figure 2. La monopage web est organisée autour de quatre composants principaux : l'agenda, la barre de navigation, le panneau de gestion des tâches affichant les tâches à réaliser. Ces quatres composants sont regroupés sur la page App.

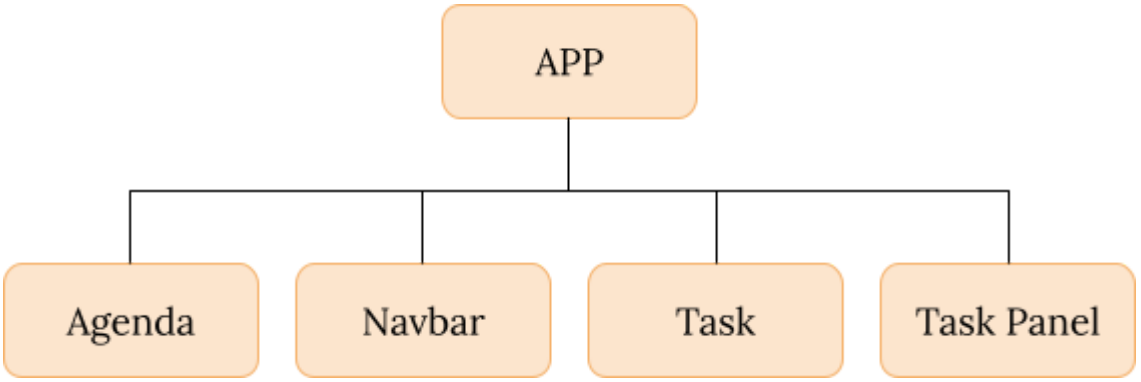

Figure 2 - Architecture globale du client mobile

2.4. *Organisation du serveur*

Implémentée grâce à Node.js, notre serveur est évènementiel donc se compose d'un seul fil d'exécution. Dès lors qu'un événement se produit, une fonction de rappel et de traitement de cet événement se déclenche. Si un nouvel événement apparaît, il sera positionné dans la file d'attente. L'avantage de ce type de serveur est qu'il permet d'éviter la création de nouveau fils d'exécution à chaque requête. Il n'y a pas non plus de réel changement de contexte, ni de problème de concurrence. Enfin, l'usage du CPU est maximisé.

Concernant le fonctionnement du serveur, l'ensemble des routes possibles pour gérer l'ajout, la modification, la suppression et la lecture est définie ci-dessous :

Routes	Méthodes HTTP	Description
api/v1/Tasks	GET	Retourne la liste des tâches
api/v1/Task/:id	GET	Retourne la tâche associée à l'id donné
api/v1/Task/:day	GET	Retourne les tâches associées au jour donné
api/v1/createTask	POST	Ajoute une tâche
api/v1/updateTask/:id	UPDATE	Modifie une tâche
api/v1/deleteTask/:id	DELETE	Supprime une tâche

3. Mise en oeuvre

3.1. Technologies utilisées

Pour implémenter notre système, nous avons utilisé différents outils présentés dans les sections suivantes.

3.1.1. *Serveur*

Pour réaliser le serveur, nous avons utilisé le framework Javascript **Node.js**. Opensource et gratuit, Node.js est asynchrone, mono thread et basé sur les évènements. La gestion des différentes routes est réalisée grâce au framework minimaliste prévu pour Node, **Express.js**. Ce dernier simplifie la gestion des URI et des méthodes http.

3.1.2. *Base de données*

La base de données est NoSQL et est gérée par le gestionnaire **MongoDB** qui utilise des documents de type JSON pour le stockage des données. L'avantage du NoSQL par rapport à une base de données relationnelle concerne sa vitesse, atout important pour de l'utilisation en temps réel notamment au niveau de l'itinérance et des notifications pour notre projet.

La base de données n'est pas locale afin de favoriser une utilisation mobile sans entraver la mémoire du device de l'utilisateur. De plus, l'**Atlas** de **MongoDB** permet une bonne gestion de l'accès aux données ce qui nous permettrait, sur le long terme de mettre en place une certaine garantie de sécurité. Pour ce qui est de ce projet, l'aspect sécuritaire de notre service n'étant pas la priorité, nous n'avons pas déterminé des accès restreints ou réfléchis. Nous sommes restés dans le cadre du développement et rien ne nous empêche de revenir sur cet aspect ultérieurement.

3.1.3. *Client Android*

L'énoncé obligeant la réalisation du client mobile en Android natif, nous n'avions pas d'autre choix que de nous tourner vers Android Studio qui est la seule manière de développer un client Android natif.

Au-delà des consignes, nous aurions aimé implanter une application entièrement sur le web en garantissant un affichage *responsive* et donc adapté à la navigation sur appareil mobile. Cependant, l'utilisation d'application de nos jours n'est pas à négliger de par son gain de confort et l'accès plus facile en déplacement. De plus, il est rare aujourd'hui, pour un service donné, de ne proposer qu'un client web. Cet engouement pour la mobilité n'est donc pas sans raison et nous pensons qu'il est essentiel de pouvoir répondre à ce besoin grandissant.

3.1.4. *Client Web*

Le client a été réalisé en utilisant la bibliothèque Javascript **React.js.** Celle-ci facilite la création d'applications web monopage grâce à l'utilisation de composants et d'états. La page HTML est par la suite générée automatiquement à chaque changement d'état et le rendu dans le navigateur est mis à jour uniquement si besoin.

Aucun framework n'a été utilisé pour le CSS. Nous jugeons peu utile de surcharger de framework une tâche réalisable via le CSS natif. De plus, il nous semble plus gratifiant de réaliser l'ensemble de l'aspect du *front end* de cette façon.

Pour communiquer avec le serveur, les requêtes ont été réalisées via les promesses et un système d'async/await en passant par **fetch.**

3.2. Gestion de projet

Globalement, les tâches ont été réparties selon le tableau suivant. Pour l'ensemble du projet, nous avons utilisé **git**. Des réunions hebdomadaires ont été organisées, à distance ou à l'université pour suivre l'avancement du projet.

	Timothy	**Thomas**	**Laura**
Serveur	X	X	
Client Web		X	X
Client Android	X		
Rédaction du rapport	X	X	X

3.3. Problèmes rencontrés

Au cours du projet, nous avons tout de même rencontré de multiples problèmes concernant le client android, le client web ou le serveur.

Au niveau du client Android, les éléments des *RecyclerView* venaient se cacher sous la barre de navigation, il nous était donc impossible de faire en sorte de compter le haut de la barre de navigation comme bas de page du contenu de la *MainActivity*. Nous avons également rencontré des problèmes avec les formats de date entre le serveur et le client. Nous avons donc préféré se baser sur des chaînes de caractères lors du transfert des données pour garder le même format ce qui nous offre un gain de temps notable en termes de développement.

Enfin, pour développer cette application, nous avons dû revoir à la baisse nos ambitions, fautes de temps et de connaissances poussées sur Android. Nous avons fait le choix d' avoir une application moins user-friendly mais réalisable notamment pour le planning journalier qui n'est finalement qu'une liste des tâches à réaliser sur la journée, avec un placement des tâches via un formulaire. L'ambition de base était de mettre un

place un agenda journalier réel avec les heures positionnées sur le côté et où l'on vient placer, via un *drag & drop*, les tâches dans les bons créneaux.

Au niveau du client web, une accumulation de retard nous a fait retravailler nos attentes. De ce fait, les *gimmicks* visant à améliorer l'expérience utilisateur se sont vu sacrifiés en faveur du bon développement des fonctionnalités principales (*drag & drop* notamment).

Enfin, au niveau du serveur, l'exécution des requêtes devait initialement être réalisée avec *axios* qui proposent des avantages certains en comparaison avec *fetch* (réponses directement en JSON, plus de compatibilité avec les différents *browsers* et leurs versions antérieures). Cependant, nous avons choisi *fetch* pour sa simplicité lors de l'implémentation et pour ne pas avoir à s'imposer l'utilisation d'un package externe.

4. Évolution possible

Actuellement, le projet possède, en partie, les fonctionnalités de base présentées dans ce rapport. Ces dernières sont vouées à être complétées par de nouvelles. Nous aurions voulu :

- Offrir à l'utilisateur la possibilité d'indiquer que la tâche est terminée, en cours ou doit être reportée à une autre date.
- Proposer une section *Dashboard* présentant diverses statistiques comme la tâche la plus fréquemment accomplie, celle la plus souvent reportée, les lieux les plus souvent visités et bien d'autres.
- Mettre en place un système de récompense qui vient motiver l'utilisateur en fonction du nombre de tâches réalisées.
- Mettre en place la fonctionnalité de *drag & drop* pour le positionnement des tâches dans le calendrier. Cette fonctionnalité permettrait d'améliorer l'expérience utilisateur en la rendant plus ergonomique et donc moins lourde dans l'utilisation.
- Ajouter un système de profil et d'authentification pour personnaliser l'expérience et enregistrer les données par utilisateur.
- Ne plus dépendre d'un serveur et pouvoir enregistrer localement les données, et les synchroniser avec le serveur, seulement quand on a de la connexion.

Enfin, l'application s'insère dans un projet à plus long terme. En effet, *Stay Awake* n'a pas pour seule ambition d'être une *ToDo list* améliorée. Elle est vouée à se transformer en plateforme plus complexe regroupant toutes les fonctionnalités nécessaires au maintien d'un mode de vie sain et productif. Les fonctionnalités envisagées à ce jour seraient :

- Amélioration de la nutrition avec la gestion des repas et leurs apports nutritionnels (Macronutriments, calories…), avec des objectifs à atteindre.

- Maintien de la condition physique avec la gestion d'entraînements, suivis des performances et objectifs à réaliser.
- Un système plus global d'objectifs avec des barres de progression parsemé d'étapes à réaliser pour voir la barre monter et ainsi continuer de progresser dans nos domaines de prédilections.

5. Conclusion

Ainsi, nous proposons une application d'organisation du quotidien appelée *Stay Awake*. Notre application permet à l'utilisateur de planifier sa journée depuis un ordinateur ou un téléphone. Pour cela nous avons su mettre en place une solution de stockage de données sur le *cloud* afin de garantir un accès aux données, rapide et synchronisé, sur l'ensemble des appareils utilisés par l'utilisateur.

Ce projet a été pour nous l'occasion de mettre en œuvre les différentes technologies étudiées pendant cette session. Naturellement, il a été important de bien étudier les divers besoins de l'application développée afin de ne pas la surcharger en fonctionnalités.

Notre projet n'est cependant pas parfait, il comporte de nombreuses améliorations possibles. Nous y voyons la possibilité de le développer encore plus, en y apportant, dans un premier temps l'ensemble des fonctionnalités proposées ci-dessus, et plus encore dans le futur. Notre structure, réfléchie et claire, permet une extension des plus faciles.