

Loïc Gressent

RAPPORT DE STAGE

Période du 15 Mars - 30 Juin 2021

Professeur encadrant : Sébastien Linck

Maître de stage : Xavier Facq

Fonction : Développeur Web



R3m Score
54 Rue René Boulanger
75010 Paris



SOMMAIRE

01	Introduction	3
02	Remerciements	4
03	L'entreprise	5
04	Méthode de travail	7
05	Réalisations	15
06	Conclusion	28
07	Annexe	29

INTRODUCTION

Dans le cadre de mon année de licence professionnelle Conception, rédaction et réalisation web, j'ai eu l'opportunité de réaliser un stage au sein de l'entreprise R3m Score en tant que développeur d'une application web. Ce stage d'une durée de 4 mois, s'est déroulé du 15 Mars au 30 Juin 2021 et permet de conclure et de valider le diplôme.

Je présenterais d'abord l'entreprise et ma fonction au sein de celle-ci, avant de parler de mon rôle, mes réalisations et mon expérience au travers de ce stage. Je conclurait sur mon ressenti global, un résumé des 4 mois et de mon expérience globale sur cette période de travail.

REMERCIEMENTS

Je tiens à remercier M. Abiven, le Président de R3m Score, d'avoir accepté de m'intégrer dans son entreprise en tant que stagiaire.

Merci à Virginie pour sa bienveillance et son hospitalité.

Je remercie tout particulièrement Xavier Facq, qui m'a intégré dans l'entreprise et qui a été mon tuteur durant la durée du stage. Son expertise et sa maîtrise m'ont beaucoup aidé.

Enfin, merci à M. Linck et à tous les professeurs de l'EiSINe pour m'avoir appris toutes ces compétences sans lesquelles je n'aurais pas pu réussir mes missions de stage.

L'ENTREPRISE

R3m Score est une startup fondée en 2019 par François ABIVEN, le président de la société mère nommée Repères.

I. REPÈRES

Repères est un institut d'études et de marketing émotionnel fondé en juin 1980. C'est une entreprise de type "Société par actions simplifiée à associé unique" (SASU). Celle-ci est spécialisée dans les études de marchés et de sondages. Entre 20 et 50 salariés y travailles.

M. ABIVEN devient le président de Repères à partir de 2005 et décide de créer R3m Score en 2019.



II. R3M SCORE

Qu'est ce que "R3M" ?

R3M était une méthode utilisé par Repères pour recueillir les émotions d'un panel de personne répondant à plusieurs types de sondages. L'acronyme signifie "Réponse 3 mots". En effet, le principe étant de récolter et de mesurer l'activité émotionnelle du consommateur en 3 mots, indiquant un ressenti spontané selon une expérience vécue. Ceci génère un score d'activation émotionnelle de grande amplitude, exploitable par les marques et les instituts marketings afin qu'ils puissent mieux comprendre et ainsi prendre des décisions selon leurs consommateurs.

L'équipe est composée de Xavier Facq, développeur senior full-stack - François Abiven, président et Virginie, responsable linguistique émotionnelle. Je rejoins celle-ci en tant développeur web.

"L'émotion au cœur des prises de décisions"

III. MON RÔLE

En 2020, R3M Score devient une plateforme SaaS (Software as a Service). Le SaaS est une solution logicielle hébergée sur Internet. Cela signifie que les utilisateurs de la plateforme n'ont pas besoin d'installer localement d'application sur leurs ordinateurs.

R3m Score est hébergé sur le cloud, et est accessible via Internet. Les employés de Repères ayant besoin de la plateforme n'ont qu'une seule connexion à effectuer afin de profiter de l'application.

Le développement de l'application est à la charge M. Facq, qui a été mon maître de stage. Il s'occupe entièrement du développement back-end et front-end de l'application, utilisant de nombreuses technologies et langages, comme par exemple Java, MySQL et ReactJs.

Avec mon arrivée en tant que stagiaire, nous avons donc organisé le travail de la manière suivante : étant plus à l'aise dans le développement Front-end (HTML, CSS, Javascript), je me suis orienté vers les tâches correspondant à l'interface utilisateur. Mon travail sera principalement d'apporter de nouvelles fonctionnalités graphiques et techniques et des améliorations esthétiques. L'amélioration de l'expérience des utilisateurs de l'application devient ma principale responsabilité.

Étant novice, je serais bien évidemment assisté et aidé par Xavier qui s'occupera de toute l'interface back-end et toutes autres tâches inaccessibles de mon côté.

MÉTHODE DE TRAVAIL

Dans cette section, je présenterais les méthodes de travail utilisé durant mon stage, incluant les logiciels utilisés, la gestion des différentes tâches etc.

I. Gestion d'équipe

Chaque membre de l'équipe possède une adresse mail professionnelle au nom de r3mscore.com lié à un compte **Microsoft 365**. Cette suite de Microsoft permet aux membres de gérer et planifier un groupe de travail via différentes applications toutes reliés entre elles.

Nous utilisons le calendrier **Outlook** afin de planifier les réunions. Il y avait deux réunions assez récurrentes. La première est une réunion appelé "Point Equipe" se tient tous les lundi, incluant Xavier, Virginie, François et moi-même. Ensuite, un petit point rapide était fait entre Xavier et moi tous les jours à 9h30. Le but de celle-ci était de faire le point d'avancement des tâches en cours du sprint actif.

Les rendez-vous sont notés sur l'application et sont liés à un lien d'invitation **Teams** où se déroule les réunions. Celles-ci ont pour but de faire le point sur l'avancée de la semaine passée, par exemple le développement technique de la plateforme pour Xavier et moi. D'autres réunions peuvent être organisées, mais n'apparaissent que pour les personnes concernées.

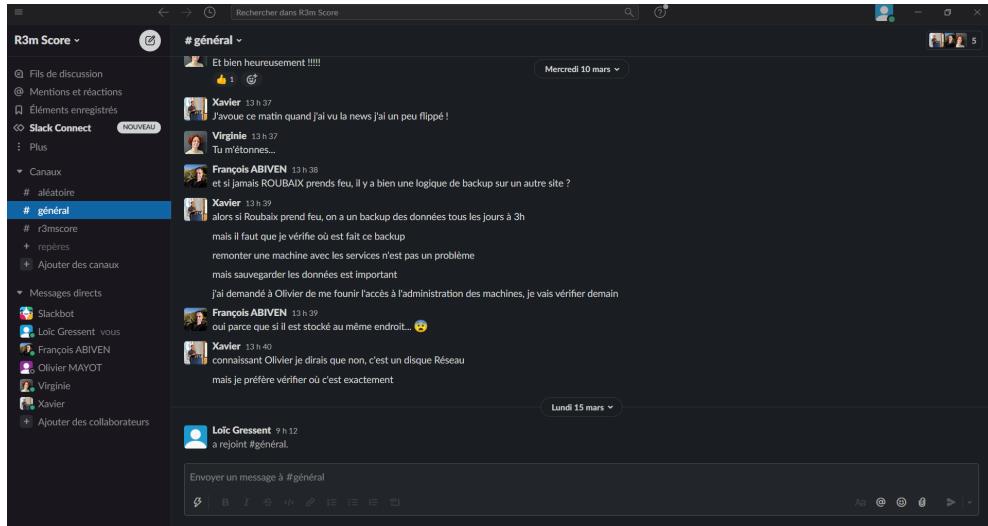
The screenshot shows the Microsoft Outlook web interface for April 2021. The calendar view displays the following events:

- Monday: 14:00 Point Equip (recurring)
- Tuesday: 14:00 Réflexion sur l' (recurring)
- Wednesday: 14:00 Canceled! (one-time event)
- Thursday: 17:30 Point Equip F (one-time event)
- Friday: 14:00 Point Equip (recurring)
- Saturday: 10:00 Réflexion sur l' (recurring)
- Sunday: 14:00 Point Equip (recurring)

The sidebar on the left includes links for "Ajouter un calendrier", "Mes calendriers", "Calendrier", "Jours fériés - France", and "Anniversaires". The bottom right corner features a hot air balloon icon and the message "Aucun événement prévu ce jour. Passez une bonne journée !".

II. Messagerie interne

La communication principale entre les membres de R3M Score se fait via l'application de messagerie Slack. Nous avons la possibilité de créer plusieurs canaux de discussions généraux, ainsi que des messages privés.



III. Gestion de projet

Le début de mon stage m'a donné l'opportunité de découvrir la méthodologie SCRUM que l'on a précédemment étudiée en cours. N'étant pour l'instant que 2 à s'occuper du développement, beaucoup de rôles normalement unique à une personne, sont attribués à Xavier. En effet, il est en quelque sorte le Scrum master, le product owner et l'équipe de développement.

Afin de bien s'organiser, nous utilisons l'application en ligne [Jira](#). C'est un logiciel de gestion de projet, développé par Atlassian. Cette application nous offre un tableau de bord de gestion des tâches, très utile pour les projets de développement comme la nôtre.

Plusieurs fonctionnalités nous sont proposées :

- Gestion de sprint
- Répartition des tâches (A faire, en cours et terminée)
- Classement des tickets selon leur nature (Tâche, sous-tâche, story, bug ou Epic)
- Gestion de la priorité des tickets ([Highest](#), [High](#), [Medium](#), [Low](#), [Lowest](#))
- Assignation du responsable et du rapporteur des tâches
- Interface consacrée au Backlog, les fonctionnalités où il reste du travail à faire qui ne sont pas dans le sprint actif.
- Feuille de route

Beaucoup plus de possibilités s'offrent aux utilisateurs, mais je n'ai pas eu l'occasion de m'en servir. Cet outil sert à structurer une bonne organisation au sein du développement.

La durée des sprints a été fixée à deux semaines, commençant et terminant un vendredi. À la fin de chaque sprint, un bilan est établi : les tâches terminées sont supprimées, les tâches encore en cours ont le choix d'être déplacées dans le Backlog ou d'être migrées vers le prochain Sprint. Idem pour les tâches qui n'ont pas été traitées.

The screenshot shows a Jira Software dashboard for the project 'R3m Score'. The title bar includes 'Jira Software', 'Votre travail', 'Projets', 'Filtres', 'Tableaux de bord', 'Personnes', 'Appli', and a 'Créer' button. The search bar at the top right contains the placeholder 'Rechercher' and various icons. On the left, a sidebar lists project navigation options like 'R3m Score', 'Tableau R3MSCORE', 'Feuille de route', 'Backlog', 'Sprints actifs' (which is selected), 'Rapports', 'Tickets', 'Composants', 'Code', 'Versions', 'Pages de projet', 'Ajouter un élément', and 'Paramètres du projet'. The main content area is titled 'R3m score 5' and shows a 'Tableau R3MSCORE' with three columns: 'A FAIRE', 'EN COURS', and 'TERMINÉ'. Each column contains several tasks, each with a blue checkmark icon (normal task) or a red square icon with a white exclamation mark (bug). The tasks are listed with their descriptions and IDs (e.g., R3SCORE-11, R3SCORE-21, R3SCORE-62, etc.). A 'Quickstart' button is located at the bottom right of the dashboard.

Tableau de bord Jira du 30 Mars. On peut noter les tâches normales indiquées d'une icône bleu, alors que les bugs sont répertoriés avec une icône rouge.

IV. Gestion de développement logiciel

L'éditeur Atlassian à l'origine de Jira Software, propose également un logiciel de gestion du code Git, appelé Bitbucket. Les deux logiciels sont interconnectés et peuvent être synchronisés afin de lier les tâches entre elles.

La méthode GIT est très efficace quand il s'agit de travailler sur le même projet à plusieurs. Bitbucket propose de gérer les différents fichiers, les différentes branches de projets, etc.

Je travaillerais donc dans le repository nommé "r3mscore_frontend_react". Ce dépôt contient plusieurs branches.

The screenshot shows the Bitbucket interface for the repository "r3mscore_frontend_react". The left sidebar has links for Source, Commits, Branches (which is selected), Pull requests, Pipelines, Deployments, Jira issues, Security, and Downloads. The main area is titled "Branches" and shows a table with the following data:

Branch	Behind	Ahead	Updated	Pull request	Builds	Actions
master <small>MAIN / PRODUCTION</small>			3 days ago			...
dev <small>DEVELOPMENT</small>	40	3	4 hours ago	Create		...
feature/R3MSCORE-105_highcharts_for_topics	40	19	12 minutes ago	#36 OPEN		...
feature/R3MSCORE-167_add_password_strength_informations	40	1	23 hours ago	#38 OPEN		...
feature/R3MSCORE-142_add_open_questions_answers_block	69	1	2021-06-01	Create		...

23 Juin 2021

Celui-ci possède actuellement 5 branches. Dans ce cas-ci, les 3 dernières branches représentent des tâches en cours. Chaque tâche de développement doit être codée et travaillée sur une branche à part des deux principales, dev et master. Je reviendrais sur ces deux dernières ultérieurement. (voir page 12)

Le nom donné aux branches tâches suit le schéma suivant :

feature/R3MSCORE-numerodetache_nom_de_la_tache (Le nom de la tâche étant écrit en anglais)

R3MSCORE-167

Donnez votre avis 1 Like 2 ... X

[Password] Ajouter des informations sur la robustesse du password saisi

Joindre Crée une sous-tâche Associer un ticket

Description

Ajouter une description...

Activité

Afficher : Commentaires

Les plus récents d'abord



Ajouter un commentaire...

Conseil de pro : appuyez sur M pour commenter

En cours

Responsable	Loïc Gressent
Rapporteur	Xavier FACQ
Développement	1 branche 1 pull request OPEN
Étiquettes	Aucun
Sprint	R3m score 11
Priorité	Medium

▼ Afficher 5 champs supplémentaires
Estimation originale, Suivi temporel, Epic Link, Composants, Vers...

Création 9 juin 2021, 15:48
Mis à jour il y a 5 jours

Configurer

Tâche

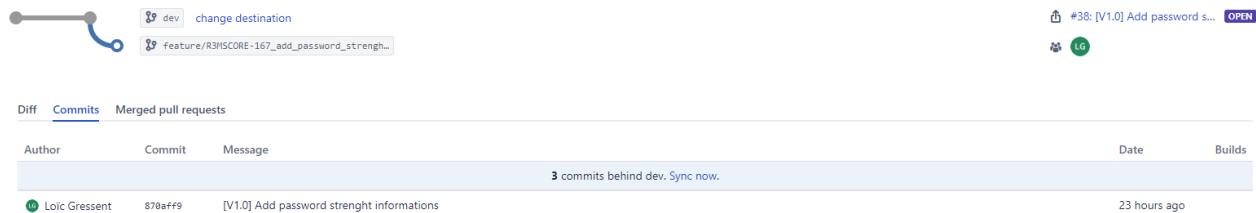


Branche GIT

R3m Score / R3m Score / r3mscore_frontend_react / Branches

feature/R3MSCORE-167_add_password_strenght_informations

Check out View source Merge ...



Ce processus est répété dans pratiquement tous les cas. Certaines fonctionnalités seront développées directement sur la branche "dev" étant donné leur simplicité ou leur maigre place dans le code. On souhaitera éviter la création d'une branche unique pour rajouter seulement quelques lignes de codes.

Une fois que j'estime avoir complété ma tâche et terminé mon développement, mes changements seront "commités" dans ma branche. Je vais ensuite créer une pull-request sur celle-ci. Une pull-request est une fonctionnalité permettant de prévenir les autres, que notre code est terminé. Dans mon cas, Xavier aura accès à une interface montrant les changements que j'aurais apporté aux fichiers modifiés.

```

+   const securityLevels = [
+     {
+       descriptionLabel: <Typography>8 caractères minimum </Typography>,
+
 Xavier FACQ 23 hours ago
Il faut créer 4 traductions pour ces messages.

Réutiliser / modifier ou ajouter de type : react.users.changepassword. ***
Reply · Unlike · ⚡1 like · Create task · Create Jira issue
+
+   validator: /^.{8,}$/,
+
+ },
+
+ {
+   descriptionLabel: <Typography>Aucuns espaces</Typography>,
+   validator: /^[^\s]*$/
+
+ },
+
+ {
+   descriptionLabel: <Typography>Au moins une lettre</Typography>,
+   validator: /[a-zA-Z]/
+
+ },
+
+ {
+   descriptionLabel: <Typography>Au moins un chiffre</Typography>,
+   validator: /.*[0-9].*/
+
+ }
+
+ ];
+

```

Exemple de pull-request avec une remarque sur du code rajouté.

En **vert**, on retrouve les lignes ajoutées qui n'existaient pas auparavant dans le fichier en question et en **rouge** sont affichées les lignes supprimées. Le reviewer a la possibilité d'ajouter des remarques entre les lignes. Dans mon cas, mon code était vérifié et devait être ajusté après chaque review de pull-request, afin d'intégrer un code propre et cohérent. Une fois la pull-request validée, les différents changements seront "merge" sur la branche dev.

Master et dev :

La plateforme R3m possède plusieurs machines, remplissant plusieurs rôles différents. Le serveur de production est la machine principale, la version "officielle" de l'application utilisée par nos clients. La branche master correspond à cette machine. Des livraisons (mises à jour) sont effectuées assez régulièrement, suite aux nombreuses demandes de nos clients, remarques de l'équipe R3M et Repères et les bugs. Xavier s'occupe de livrer les nouveautés quand celles-ci sont importantes, nombreuses ou urgentes pour certaines. En moyenne, une livraison par semaine est effectuée.

Ensuite, il existe le serveur d'intégration. Celui-ci s'apparente sommairement à une machine de test. Le code des différentes tâches effectuées, après tests et vérifications en local, sera envoyé sur l'intégration afin que l'on puisse mettre en pratique les changements. Des tests seront réalisés, afin de s'assurer du bon fonctionnement des nouveautés qui seront ensuite livrées sur le serveur de prod. La branche dev correspond à ce serveur d'intégration.

V. Travail personnel

Ayant eu de bonnes affinités avec Visual Studio Code depuis mon année de licence professionnelle, j'ai naturellement travaillé sur celui-ci durant mon stage. Quelques manipulations ont dû être faites pour que je puisse travailler sur le projet et gérer mes fichiers via Bitbucket.

Visual Studio Code intègre la possibilité de se connecter à un compte Atlassian. Le but étant de synchroniser Bitbucket avec l'éditeur de texte afin de pouvoir travailler dans ma branche GIT et gérer mes fichiers via mon éditeur et sa console.

Après avoir récupéré le repository "r3mscore_frontend_react", j'en ai fait une copie local pour pouvoir travailler dessus. Tout au long de mon stage, je serais amené à utiliser les commandes git pour manipuler mes fichiers. Toutes ces manipulations sont faites dans le terminal intégré à VSCode :

Création d'une branche : git checkout -b nom_de_la_branche

Cette commande sera beaucoup utilisée, à chaque création de branche. Le paramètre -b désigne la création d'une branche non-existante. En étant sur dev, la commande créer une copie de dev.

Choisir et aller sur une branche : git checkout nom_de_la_branche

Cette commande sera principalement utilisée pour switcher entre dev et ma branche personnelle.

Récupérer la dernière version d'une branche : git pull

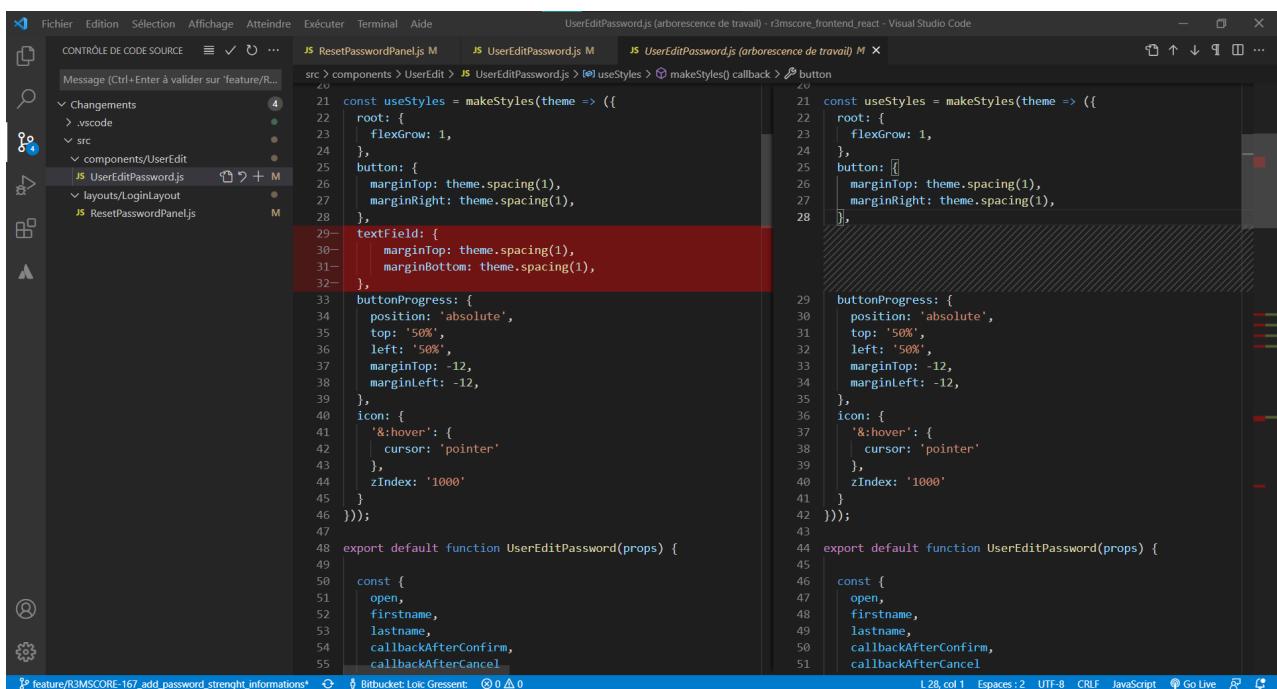
Cette commande nécessite d'être au bon endroit. Ici, je dois être sur dev pour récupérer sa dernière version. Ma version locale de dev peut-être obsolète si un autre développeur publie des changements sur cette branche. Avant de créer une nouvelle branche à partir de dev, je dois récupérer les différences entre mon ordinateur et le dépôt git.

Faire une sauvegarder locale des modifications : git commit

Exceptionnellement, les commit seront faits directement depuis la vue git proposée par Visual Studio Code. (voir image ci-dessous).

Envoyer les modifications sur le serveur : git push

Cette commande me permet d'envoyer les fichiers mis en attente dans le commit, sur le serveur. Bitbucket affichera mon commit, et ma branche sera à jour avec mes modifications. C'est ici que l'on peut ouvrir une pull-request pour valider, et ensuite pousser les modifications sur dev.



The screenshot shows the Visual Studio Code interface with the Git view open. On the left, the file tree shows files like 'ResetPasswordPanel.js', 'UserEditPassword.js', and 'UserEditPassword.js'. A message box at the top says 'Message (Ctrl+Enter à valider sur 'feature/R3MSCORE-167_add_password_strenght_informations')'. The main area displays the code for 'UserEditPassword.js'. Staged changes are highlighted in red, particularly in the 'button' and 'buttonProgress' sections. The status bar at the bottom indicates 'Bitbucket: Loïc Gressent' and other repository details.

UserEditPassword.js en vue git avant commit.

La vue git de Visual Studio Code montre à gauche, le fichier actuellement sur le serveur et à droite le fichier en attente d'être poussé sur la branche. L'éditeur montre les différences entre les deux. Les caractères rouges sont les caractères supprimés, les vertes sont les caractères rajoutés.

Ensuite, l'arborescence à gauche offre la possibilité de "mettre les modifications en attente", ce qui les prépare au commit. Une fois les fichiers voulus mis en attente, un champ nous permet de commit les changements en donnant un nom. Ce nom sera généralement explicite aux tâches effectuées.

Message (Ctrl+Enter à valider sur 'feature/R3MSCORE-167_add_password_strenght_informations')

Un git push sera ensuite effectué dans le terminal pour pousser les modifications sur le serveur.

RÉALISATIONS

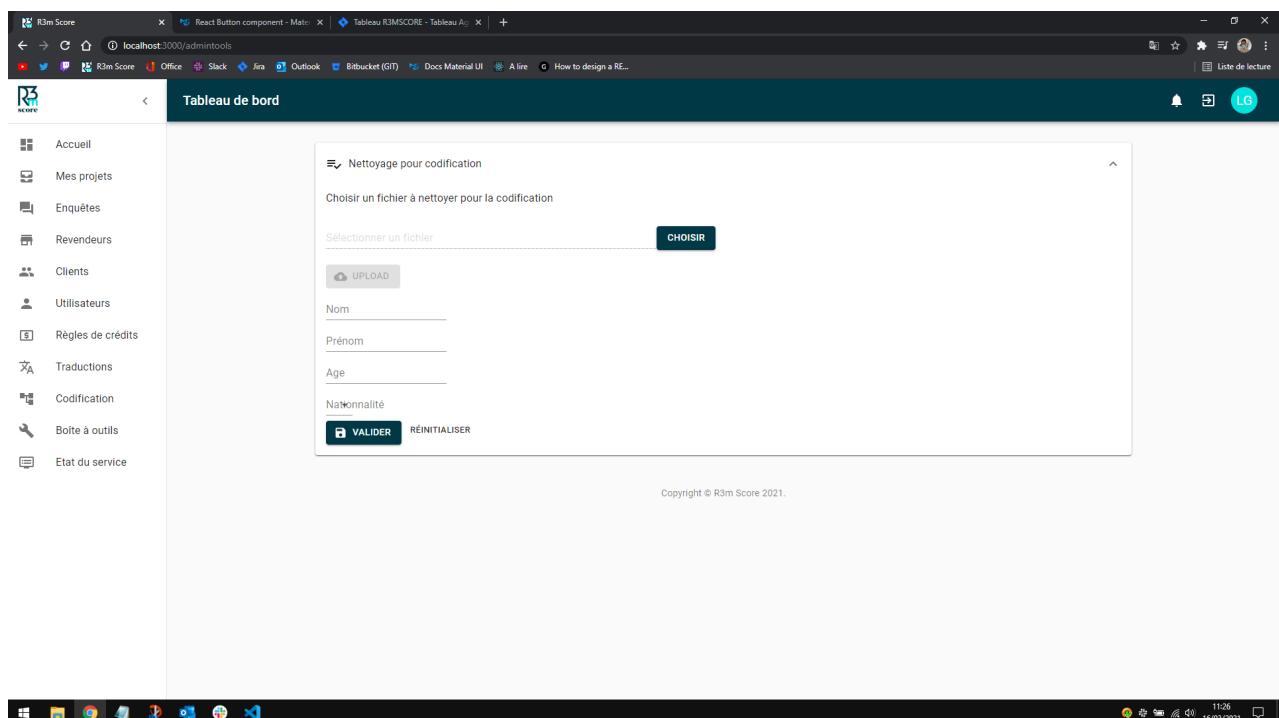
Ce stage m'a permis de découvrir ReactJs. Une bibliothèque déclarative javascript qui est utilisée pour créer des interfaces utilisateur. J'ai aussi utilisé Node JS afin d'exécuter le script. La commande "npm start" permet de lancer la compilation et l'exécution du script.

Dans cette section, je présenterais ce que j'ai réalisé pour l'application. Je parlerais pour chaque réalisation, des problèmes rencontrés, de ma démarche et j'expliquerai mes choix.

I. Introduction à ReactJs

Je n'ai pas commencé directement à développer une fonctionnalité pour l'application dès le début. J'ai pris quelques jours pour assimiler et apprendre à manipuler la bibliothèque ReactJs.

Xavier a créé une tâche fictive, me demandant de créer un formulaire. Ensuite, ce formulaire a été complété avec des champs supplémentaires permettant d'y intégrer une fonction calculant l'indice de masse corporelle (IMC) d'une personne.



Je n'ai pas rencontré de problèmes particuliers à débuter avec React. La difficulté étant de s'approprier une nouvelle manière d'écrire les différents composants et leurs propriétés, appelées "props". Cela m'as entre autre permis de rafraîchir mes capacités en javascript.

II. Formulaire de collecte

L'une des fonctionnalités de l'application est la possibilité de créer un formulaire d'enquête appelé formulaire de collecte. Il était possible jusqu'ici de créer les blocs suivants : introduction, R3M, question et merci.

The screenshot shows a modal window titled 'Question A'. At the top left is a toggle switch labeled 'Activer le bloc' (Activate block) which is turned on. At the top right is a close button 'X'. Below the title, there's a section for 'Nom de la question 1 (8 / 30)' with a placeholder 'Relation'. A text area contains the question: 'Quel est votre degré de relation avec Pauline ? Etes-vous:' followed by three options: 'un(e) collègue', 'un(e) fournisseur/presta', and 'un(e) client(e)', each with a delete icon. Below this is a 'Réponses' section with a plus sign to add more. At the bottom of the modal are four buttons: 'CRÉER UNE INTRODUCTION', 'CRÉER UN BLOC R3M', 'CRÉER UN BLOC QUESTION' (which is highlighted in grey), and 'CRÉER UN BLOC MERCI'. At the very bottom are two buttons: 'ANNULER' and 'ENREGISTRER' (with a save icon).

Suite à la demande de M. Abiven, il m'as été demandé d'y ajouter un bloc question ouverte. Comme son nom l'indique, ce bloc sera utilisé pour poser une question ouverte à la personne répondant a l'enquête. Pour le moment, on ne peux poser que des questions fermées, comme le montre l'image ci-dessus.

Je me suis donc inspiré de la structure déjà présente pour les autres blocs, en adaptant à mes besoins. J'avais besoin d'un champs "Stimulus" et d'un champ pour écrire la question. La plupart de la construction graphique de l'application se faisait via [Material UI](#) qui est une bibliothèque de composant React permettant de construire des interfaces Web.

La particularité de cette tâche était sa division en deux parties. La partie formulaire et la restitution. Le formulaire est utilisé pour créer des enquêtes qui seront ensuite proposés et projetées à des personnes pour qu'ils puissent y répondre. Il fallait donc ensuite développer cette partie, où les utilisateurs doivent écrire leur réponse à la question. J'ai utilisé les éléments "Textarea" de Material UI pour construire les champs nécessaires.

Paramètres du formulaire de collecte

* Nom
Formulaire de test final

Activer la collecte des données

Language
fr

Type de séquence
 Ordre Rotation Aléatoire

Question ouverte A

Activer le bloc

Activer une temporisation



* Nom du stimulus (12 / 128)
Nom stimulus

* Veuillez saisir le texte de la question ouverte
Que pensez vous de cette image ?
32 / 255

Configuration du formulaire



Que pensez vous de cette image ?

Réponse libre*

0

SUIVANT >

Restitution de la question ouverte.

Cette fonctionnalité n'as, au premier abord, pas besoin de beaucoup de développement. En revanche il y a beaucoup de détails et de contraintes auxquels réfléchir. Je vais lister les principales :

- Attribuer des variables uniques aux champs de openQuestion, pour ne pas interférer avec celles des autres blocs.
- Ne pouvoir enregistrer le formulaire uniquement si les champs Stimulus et contenu de la questions sont définis.
- Ne pouvoir passer au bloc suivant seulement si la réponse est définie.
- Vérifier le bon fonctionnement du bloc sur tous les navigateurs.
- S'assurer du fonctionnement de la suppression du bloc ainsi que son enregistrement dans la configuration.
- Limiter le nombre de blocs total à 3. (Similaire au question R3M)
- Créer les labels de traductions. (Voir explications ci-dessous)
- Ne créer le lien de l'enquête uniquement si le bloc est bien actif.
- Trouver une icône adéquate pour la toolbar de création de blocs.
- Désactiver et griser les boutons quand on ne peut plus créer de blocs.

Les traductions :

L'application utilise un service de traduction interne qui permet d'adapter plus facilement le changement de langue selon les profils d'utilisateurs. Cela fonctionne de la manière suivante :

Pour l'affichage d'une phrase ou d'un quelconque mot qui n'est pas variable (Comme les noms de projet etc.), on créer un label sur lequel on va amener plusieurs langues. Par exemple pour écrire "Veuillez saisir le texte de la question ouverte", nous allons écrire une clé de traduction comme ceci :

```
t("react.project.collectforms.openquestioncontent")
```

Ensuite, on ira configurer pour cette clé de traduction, la phrase à écrire dans les différentes langues. Grâce à la fonction de traduction t(), le système changera le label en fonction de la langue de l'application de l'utilisateur.

III. Modification OpenQuestion

Suite à quelques réunions, il y a eu plusieurs modifications à faire concernant l'ajout des questions ouvertes et le formulaire de collecte en général.

- J'ai proposé de déplacer les boutons de création de blocs (Voir capture page 16) en haut à droite de l'écran afin que l'on puisse y avoir accès n'importe où sur la page. En effet si les formulaires créés sont assez longs, il est assez désagréable de devoir descendre tout en bas pour atteindre ces boutons. J'ai donc proposé de créer uniquement des icônes, en positions fixe qui suivent le scroll de la page. La fonction associée à l'icône sera affichée en survol de la souris sur celui-ci. Les icônes deviennent accessible en permanence car ils suivent le défilement de la page.



- Il m'a été demandé de passer le bloc R3M en facultatif. En effet, pour pouvoir enregistrer le formulaire, il fallait obligatoirement la présence d'un bloc de question R3M. Hors, après réflexion avec François, il a décidé de retirer cette contrainte.
- Les formulaires sont parfois utilisés avec des images qui constituent le cœur de la question. Le problème étant que l'affichage de celle-ci était assez petit, et aucunes possibilités d agrandissement n'était possible. Je me suis occupé d'intégrer une librairie pour que les utilisateurs puissent cliquer sur celle-ci et l'a voir en taille réelle sur tout l'écran.

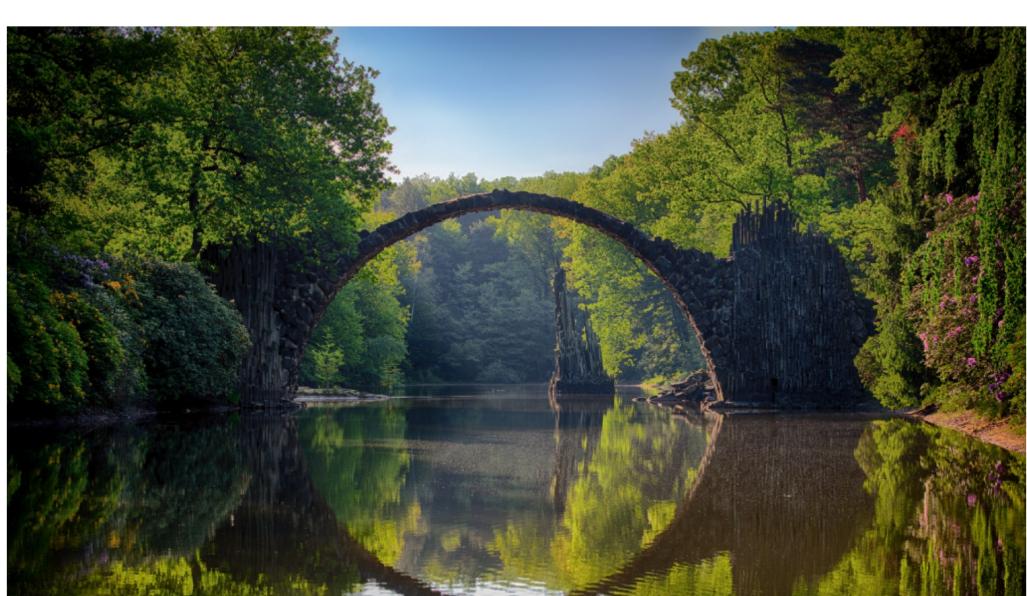
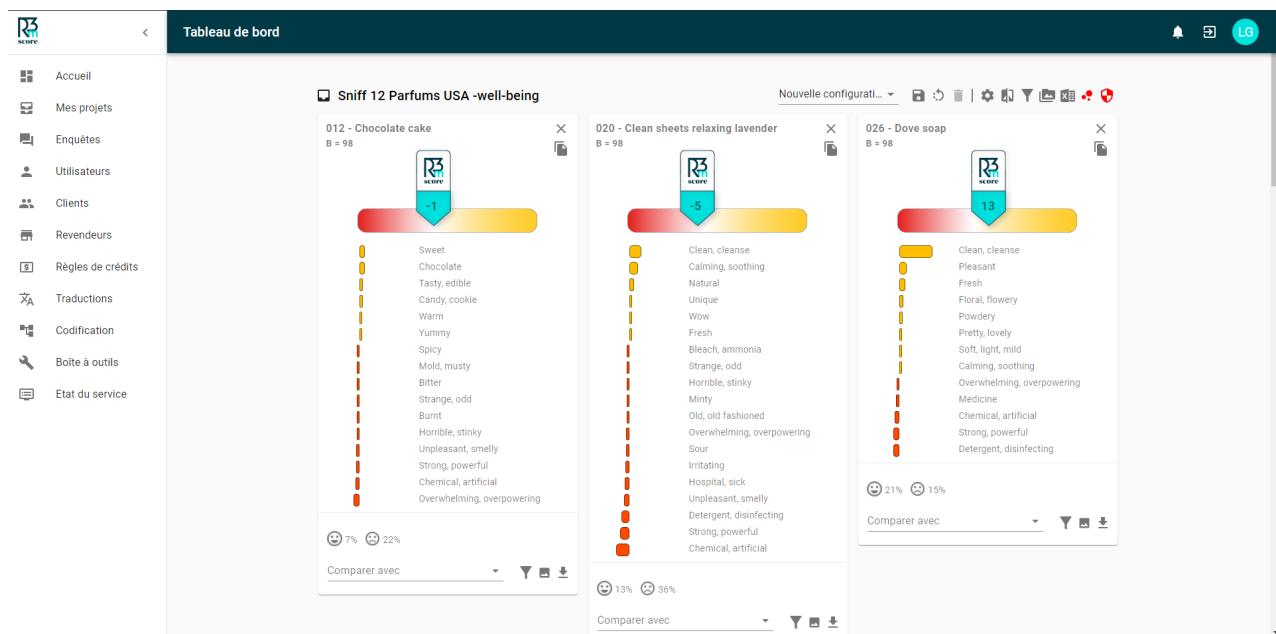


Image cliquée sur la restitution de l'enquête.

III. Dashboard V2

Voici ci-dessous le dashboard classique. Celui-ci affichage des cartes corresponds aux différents stimuli d'un projet. Ici un projet de test sur des parfums, on peut retrouver une carte pour chaque parfum et les résultats obtenus. Un calcul est fait pour déterminé la "score" qui correspond à l'intensité des émotions. Les mots cités et leurs poids et aussi affichés, ainsi que le nombre de participant appelé la base. Il y a la possibilité de comparer les cartes entres elles, appliquer des filtres tel que l'age ou la provenance du panel, copier une carte et télécharger une carte en tant qu'image.



Le projet était de créer une nouvelle page pour afficher une vue plus succincte du dashboard classique, en affichant le thermomètre verticalement pour en afficher plus à l'écran. La demande a été faite par les utilisatrices de l'application, travaillant chez Repères. Cette page représenterait un grand gain de temps. En effet, elles ont l'habitude de créer eux mêmes une vue de synthèse des stimuli afin de les envoyer aux clients, mais cela prends beaucoup de temps.

J'ai donc créer un nouveau fichier qui sera exécuté par ProjectDashboard.js qui est le fichier qui reçoit la liste des stimulus d'un projet, et construit en fonction de cette liste le dashboard en question. Mon fichier DashboardV2.js exécutera StimulusCardVertical.js. Le but étant de construire une vue facile et succincte des thermomètres pour un export rapide d'un projet.



Vue synthèse du projet possédant le plus de stimulus actuellement

Le curseur bleu a été créé et les thermomètres ont été importé et modifier pour l'affichage vertical.

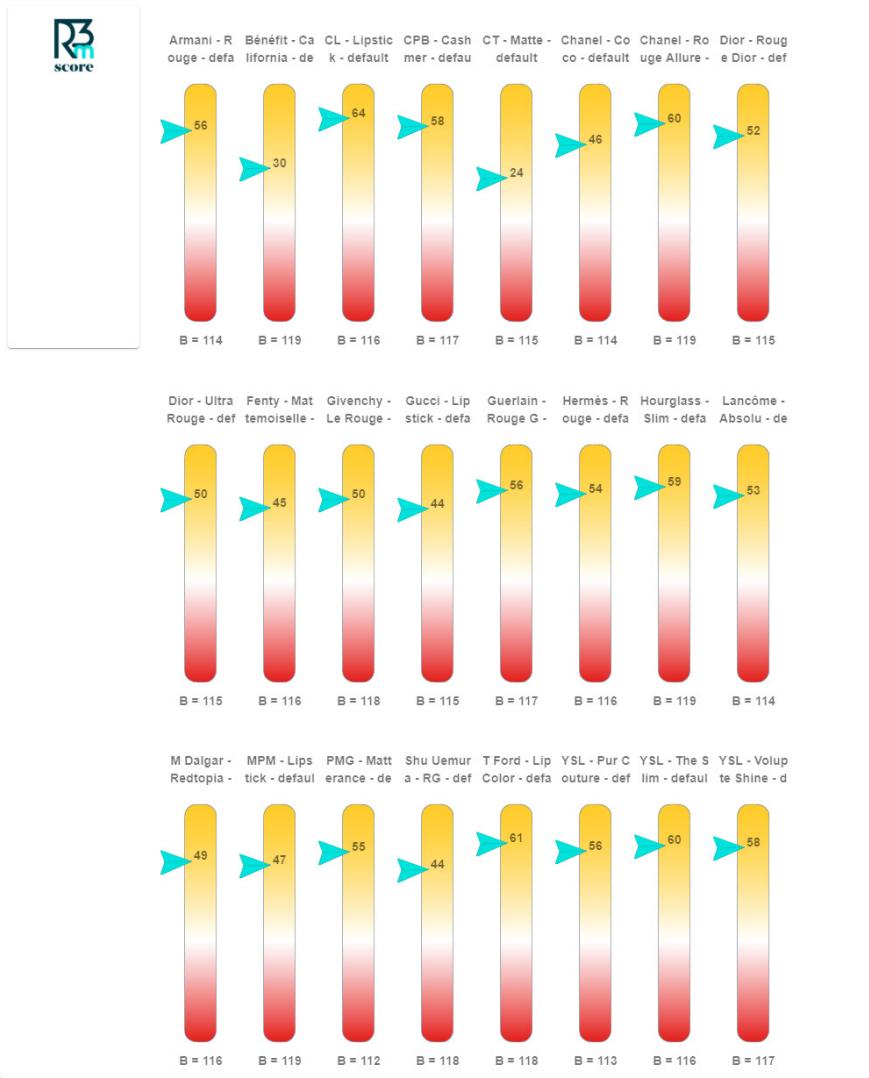
Les principales difficultés de cette fonctionnalité ont été :

- la ré-importation des fonctions de suppression de cartes
- d'ajuster la hauteur du curseur en fonction sa valeur et sa place au niveau du thermomètre.
- Ajouter les flèches de comparaisons déjà présentes dans le dashboard classique.

Optimisation du layout :

Une demande de François a été faite pour que l'affichage des thermomètres s'adapte à leurs nombre. Une ligne affiche 10 cartes maximum. Sur un exemple de projet avec 10 stimulus, il aurait fallu en afficher 5 sur la première ligne et 5 sur la 2nd afin d'harmoniser l'affichage.

J'ai donc développer un algorithme, d'abord dans un fichier de test, puis ensuite dans le fichier voulu. Toutes les fonctions sont regroupées dans utils.js et c'est ce fichier qui est importé dans les autres. Cet algorithme prend le nombre de stimulus en entrée et calcul la meilleure solution d'affichage. Par exemple, pour un nombre de stimulus de 24, l'algorithme sortiras le chiffre 6, qui correspondras au nombre auquel un saut de ligne sera appliquée. L'affichage du dashboard sera donc 4 lignes de 6 stimulus.



Malheureusement, après plusieurs jours de test et de discussions, certains cas ne rentrais pas dans les attentes. De plus, même si l'affichage fonctionnait correctement, le fait de supprimer certaines cartes ne faisait pas s'ajuster les cartes, ce qui aurait perturbé et dérangé les utilisateurs. Nous avons donc garder la fonction dans utils.js mais avons supprimé la fonctionnalité.

Intégration du nouveau dashboard :

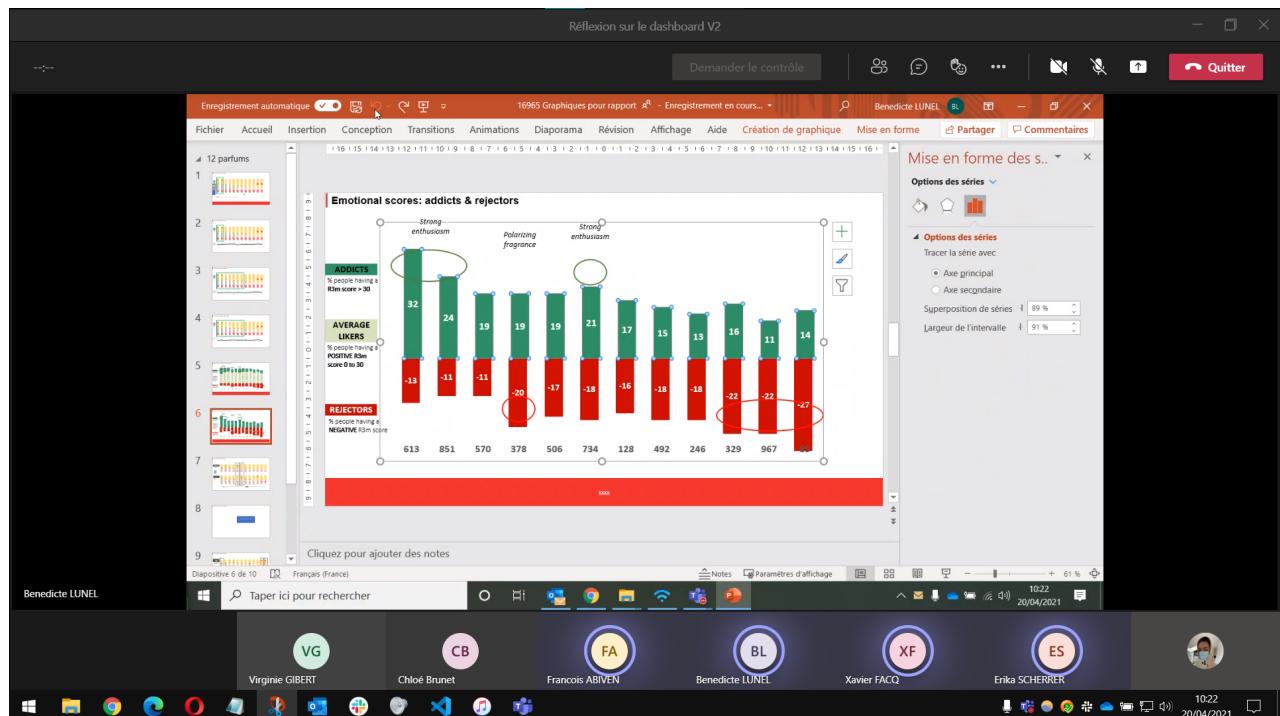
Après plusieurs tests et corrections de code, il a fallu penser à comment intégrer le dashboard. Jusqu'à maintenant, il n'y avait aucun moyen d'y accéder hormis de taper le nom du fichier dans l'URL. Après quelques réflexions, nous avons décidé avec Xavier d'intégrer un switch on/off dans un nouvel onglet des options du dashboard.



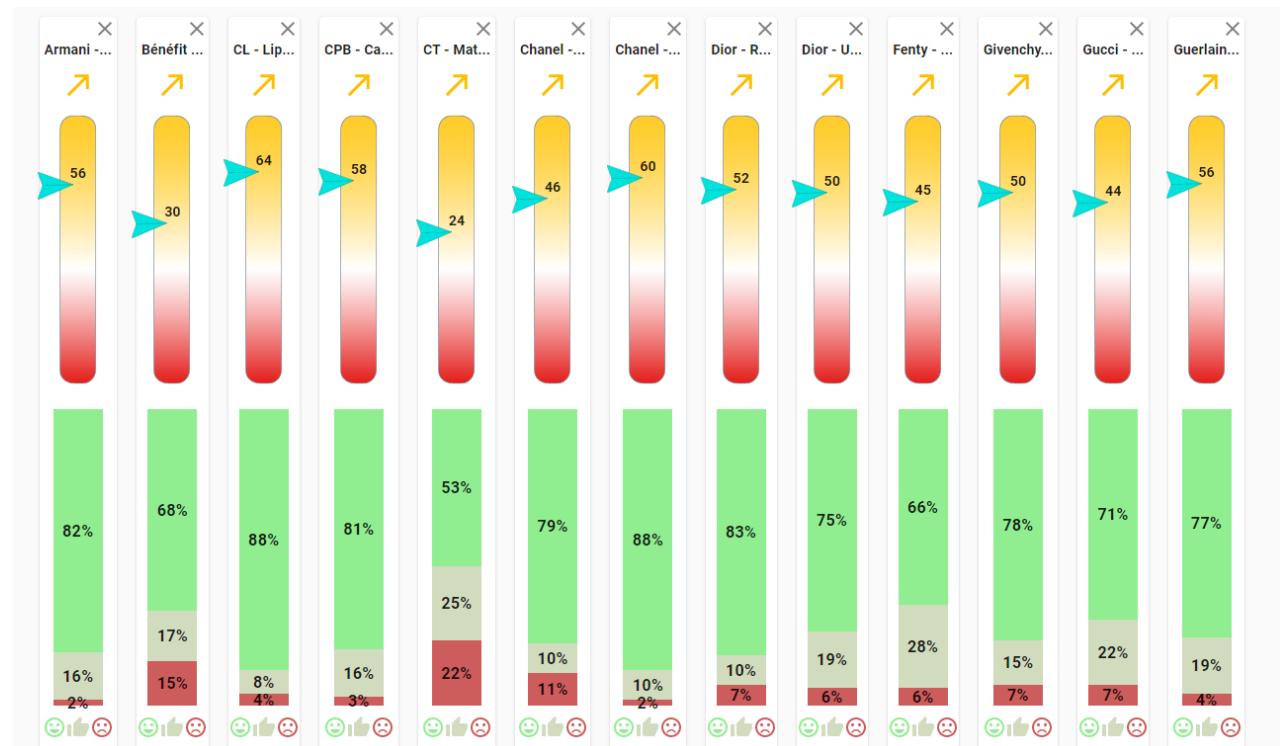
Activer le mode Synthèse

IV. Graphique Addicts/Haters

Suite à une réunion avec l'équipe de Repères concernant le nouveau dashboard, une demande nous a été faite pour intégrer, en dessous des stimulus verticaux, un graphique pour chaque stimulus affichant le nombre de Addicts et de Haters. Ces valeurs représentent le nombre de personnes ayant eu un score positif au stimulus et inversement pour les haters.



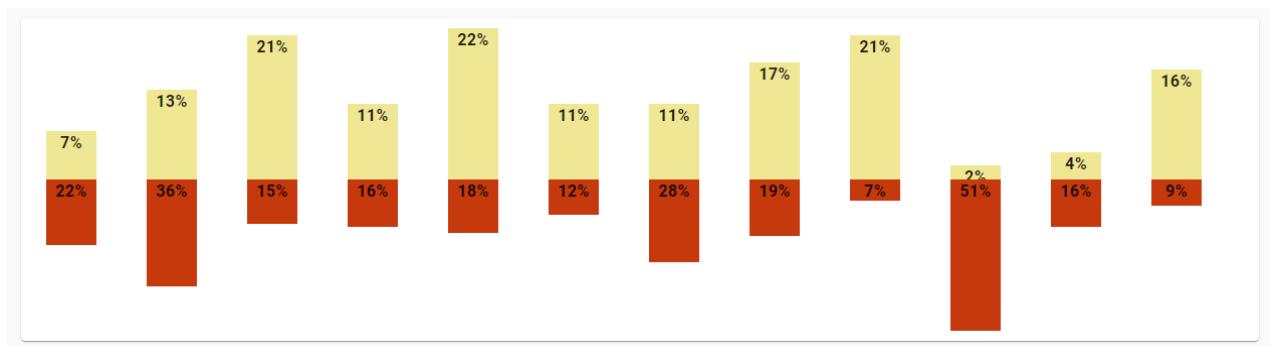
Maquette de graphique présentée lors de la réunion.



Première version des graphique addicts / haters.

N'ayant aucunes bases de développement pour ces graphiques, j'ai alors du créer les graphiques moi même. Chaque stimulus possède une valeur de Addicts et de Haters. J'ai donc créer des blocs de la taille correspondant aux valeurs envoyées.

Rapidement, des soucis de cohérences de taille sont arrivés. Des blocs de plus petites valeurs étaient plus grands que d'autres et inversement. De plus, après une première présentation à l'équipe de Repères, il a été question d'ajuster la présentation des graphique de sorte a ce que les barres commencent toutes au milieu, comme l'image de maquette montrée lors de la première réunion.



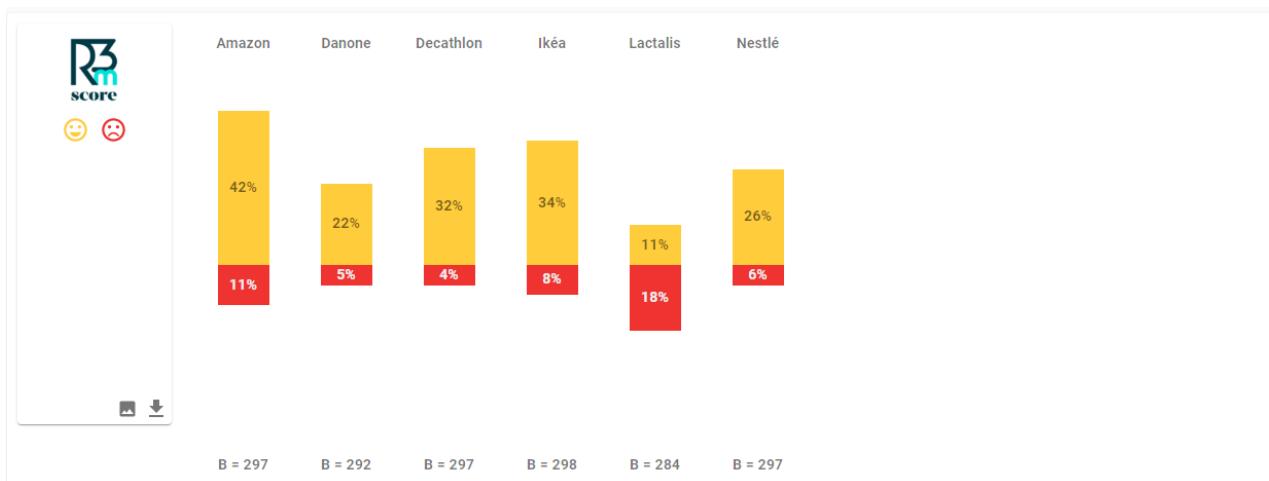
Deuxième version des graphiques.

Beaucoup d'ajustement et de détails ont été pris en compte :

- Ajustement de la position et de la taille des valeurs en pourcentage dans les barres. Elles seront désactivées en dessous d'un certain pourcentage.
- Ajout du nom du stimulus et de la base.
- Changement de couleur pour matcher avec celles des thermomètres, à la demande de François.
- Alignement parfait aux stimulus d'en haut.
- Ajout les filtres utilisés dans l'encadrer de gauche, comme sur le bloc d'en haut.
-

Calcul de la taille :

Après plusieurs essais non concluant, nous avons trouvé une solution efficace pour afficher une taille cohérente aux graphiques. J'ai écrit une fonction permettant de récupérer la valeur maximum positive et négative, afin de définir l'échelle du projet. La barre à 100% de la hauteur de la div sera la valeur la plus grande des Lovers, et la même chose pour la barre du bas. Chaque projet chargé aura une échelle différente, en fonction des résultats obtenus. Le graphique est donc plus harmonieux et facile a comprendre.



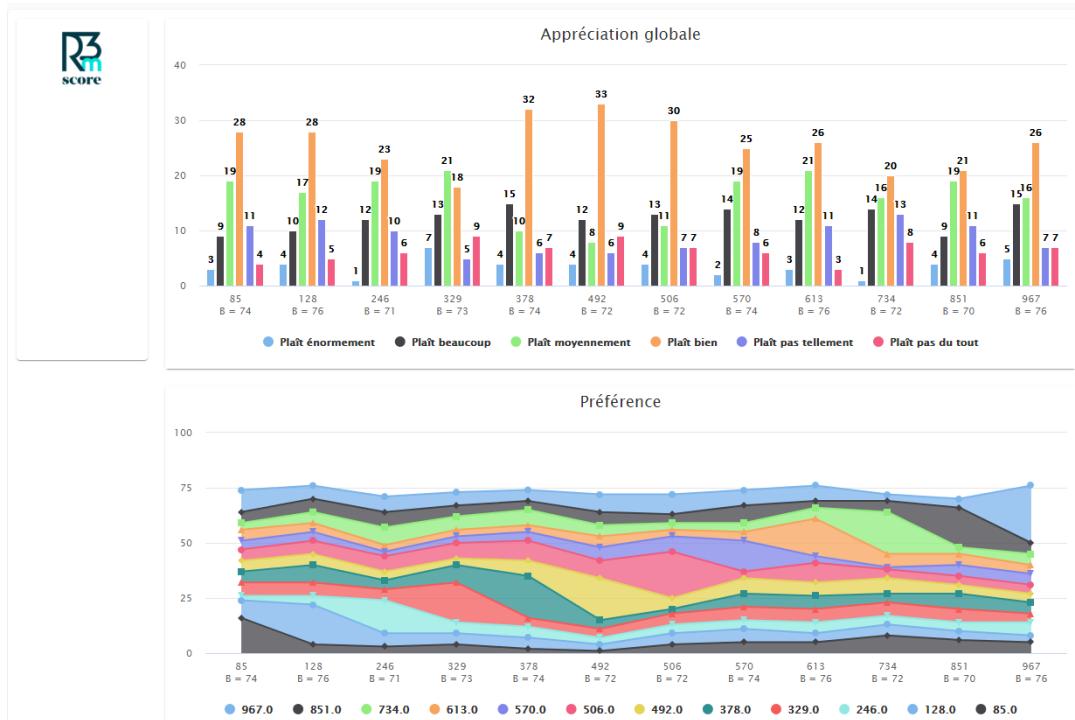
Version actuelle des graphs Addicts / Haters

V. Données neutres (ou Topics)

D'autres données sont calculés sur les projets, ce sont les données neutres aussi appelés topics. Le projet est de rajouter un troisième bloc en dessous, sous forme de graphique plus avancée.

Xavier n'ayant pas encore introduit les topics du côté back-end, j'ai d'abord commencé à tester les graphiques d'une librairie appelé Highcharts, avec des valeurs factices. Ce début m'as permis de me familiariser avec le package Highcharts, d'explorer les différents types de graphiques et les paramètres disponible. La plus partie la plus importante étant de pouvoir importer les données neutres dans les paramètres d'un graphique highchart.

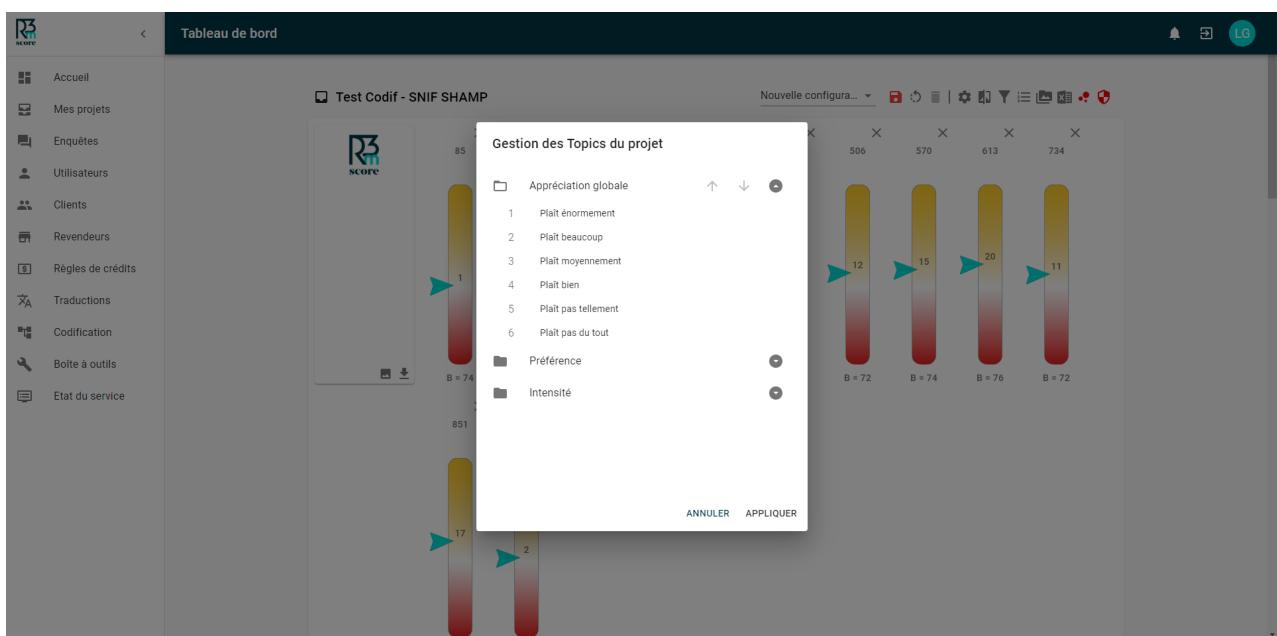
Je vais reprendre le même cheminement que pour les stimulus, à savoir la boucle qui permet de générer un graphique par catégorie de topics. J'ai créé TopicsReportingChart.js, qui sera appellé par PanelReporting.js qui fais la boucle sur les données neutres.



Interface de tri :

Suite à une remarque de François lors d'une réunion, il m'as été demandé de faire une interface qui offre la possibilité de trier l'ordre des valeurs de topics afin qu'ils s'affichent dans l'ordre choisi par les utilisateurs.

J'ai rajouté une icône ouvrant une boite de dialog (Material UI), ouvrant la liste des topics disponible du projet, et ses valeurs à l'intérieur. Avec un système de sélection et de numéro de position appelé "order", il est possible de changer l'ordre d'affichage des valeurs. La restitution des valeurs sera trier par "order" et donc afficher selon l'ordre choisi dans l'interface, après enregistrement.



Ma première intention était de faire un système de drag and drop, mais cela s'est revelé être trop compliqué. Cette fonctionnalité était assez importante, et je n'avais pas le temps de reprendre tout pour le drag and drop. J'ai donc suivi les conseils de Xavier et réalisé avec une sélection en cliquant sur la valeur. La valeur sélectionné activeras les flèches au dessus et l'utilisateur aura la possibilité de descendre ou monter à sa guise.

Difficultés rencontrées :

- Développement de la fonction qui change la valeur "order" sur le clique d'une flèche.

VI. Tâches en cours

Une tâche est encore en cours actuellement, celle de l'amélioration de l'interface de changement de mot de passe. Suite à une remarque d'un client, il a été proposé d'afficher la force du mot de passe en temps réel. Le développement en est à sa fin, les modifications sont en attentes de tests avant d'être déployées sur le serveur d'intégration.

Modifier le mot de passe

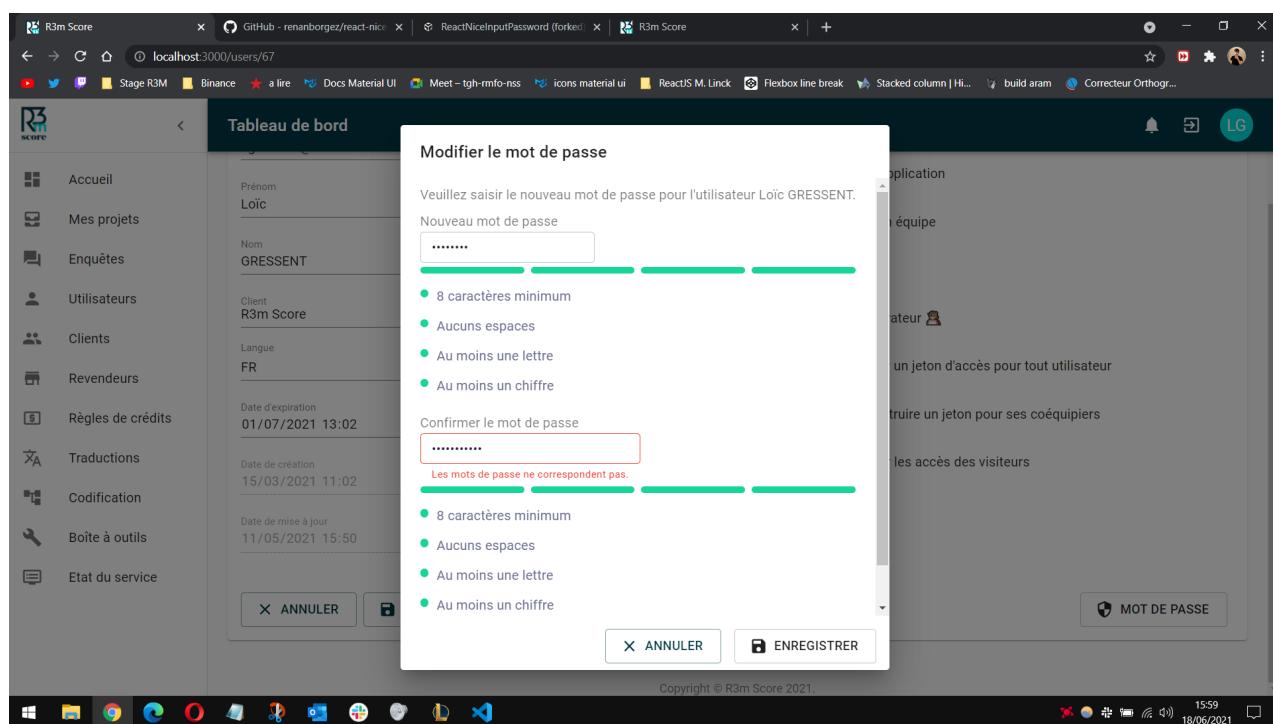
Veuillez saisir le nouveau mot de passe pour l'utilisateur Loïc GRESSENT.

Nouveau mot de passe

Confirmer le mot de passe

Pour que le mot de passe soit valide, il doit contenir : 8 caractères minimum, au moins une lettre, au moins un chiffre et aucun espace

X ANNULER ENREGISTRER



The screenshot shows the R3m Score application interface. On the left is a sidebar with various menu items: Accueil, Mes projets, Enquêtes, Utilisateurs, Clients, Revendeurs, Règles de crédits, Traductions, Codification, Boîte à outils, and Etat du service. The main area is titled "Tableau de bord". In the center, a modal window titled "Modifier le mot de passe" is displayed. It contains two input fields: "Nouveau mot de passe" and "Confirmer le mot de passe". Below these fields is a list of validation rules: "8 caractères minimum", "Aucuns espaces", "Au moins une lettre", and "Au moins un chiffre". The "Nouveau mot de passe" field has a red border and the message "Les mots de passe ne correspondent pas." below it. At the bottom of the modal are "X ANNULER" and "ENREGISTRER" buttons. A "MOT DE PASSE" button is also visible in the bottom right corner of the main dashboard area. The status bar at the bottom shows the date and time: 15/06/2021 15:59.

CONCLUSION

Ce stage m'aura appris énormément dans tous les domaines. J'ai découvert beaucoup d'aspect en gestion de projet ainsi que le travail en petite équipe. Le travail interne avec l'équipe de Repères et les retours des clients était vraiment une expérience unique et j'ai grandement apprécier travailler selon les besoins de certains, et de pouvoir échanger sur les solutions techniques avec eux.

J'ai aussi grandement améliorer mes capacités en algorithmie, aussi bien que ReactJs en lui même. Tout cela m'as permis de voir dans des conditions professionnelles, les notions apprises au cours de ma licence.

ANNEXE

I. Librairies utilisées

Material UI :

<https://material-ui.com/>

Zoom image formulaire de collecte:

<https://www.npmjs.com/package/react-medium-image-zoom>

Highcharts :

<https://github.com/highcharts/highcharts-react>

Informations mot de passe :

<https://www.npmjs.com/package/react-nice-input-password>