Centre psychothérapique DE L'AIN

Avenue de Marboz – CS 20503

01012 Bourg-en-Bresse Cedex

Pablo Guinard

Promotion 2021 – 2022

Diplôme Universitaire Technologique Informatique

RAPPORT DE STAGE

"Développement de l'App Rehab et de son tableau de bord"

19 avril - 24 juin

Soutenance : 24 juin

Maître de stage

MICHON Alexis

Responsable pédagogique

TELLEZ Bruno

IUT Lyon 1 l'excellence technologique

Département Informatique

71 rue Peter Fink – 01000 BOURG-EN-BRESSE

Téléphone : 04 74 45 50 65 – Télécopie : 04 74 45 50 51

Courriel : iutbourg.info@univ-lyon1.fr

URL : https://iut.univ-lyon1.fr

Lyon 1

Remerciements

Tout d'abord, j'aimerai remercier Monsieur Alexis MICHON d'avoir été mon tuteur durant ce stage, ainsi que Monsieur Bruno TELLEZ, mon tuteur pédagogique, qui m'a aidé à obtenir ce stage. Je tiens également à remercier Monsieur Jean-Noël Perrimbert de m'avoir initié à l'architecture du service informatique et pour son aide technique lors de la mise en place du serveur web et de la sécurisation de l'application.

Enfin, je remercie Monsieur Louis CHARNAY, avec qui j'avais déjà travaillé sur le projet avant le début du stage, pour ses explications techniques de certains aspects de l'application qu'il avait réalisés.

Table des matières

Introduction

Le DUT (Diplôme Universitaire Technologique) Informatique offre l'opportunité de faire un stage à la fin de la formation afin de valider le diplôme. Ce stage a pour but de mettre en œuvre les connaissances et compétences acquises lors de la formation. J'ai donc eu l'occasion de faire un stage du 19 avril au 24 juin 2022 (10 semaines) au sein du service informatique du Centre Psychothérapique de l'Ain (CPA).

Le sujet de mon stage s'inscrit dans la continuité de mon projet tutoré (projet en groupe de 3 mené au cours de la deuxième année de la formation). En effet, le sujet du stage était le même que celui du projet tutoré, à savoir une application mobile permettant de proposer une sorte de blog aux patients à distance. Un tableau de bord web était aussi nécessaire, permettant d'ajouter régulièrement du contenu dans l'application. Une version web du blog existait déjà, le but de l'application était donc, en plus d'offrir une solution mobile, de trier le blog tout en le modernisant.

Ayant déjà réalisé une première version du projet lors de l'année, mon stage s'est concentré sur l'amélioration de certains aspects du projet, l'ajout de fonctionnalités demandées par les rédacteurs du blog actuel et la publication de l'application ainsi que du tableau de bord. En parallèle de ça, j'ai participé au développement de la nouvelle version web du blog.

Ces semaines de stage m'amènent à la rédaction de ce rapport de stage, dans lequel je présenterai dans un premier temps l'environnement dans lequel j'ai travaillé, pour ensuite décrire l'état du projet au début du stage. Je détaillerai dans une deuxième partie l'ensemble des améliorations que j'ai apporté au projet. Je conclurai sur un bilan sur les plans personnels et professionnels ainsi qu'une conclusion générale sur ce stage.

Environnement de stage

1. Présentation de l'entreprise

Le Centre psychothérapique de l'Ain (CPA) est un établissement de santé privé d'intérêt collectif créé le 2 décembre 1971 par la fusion des hôpitaux psychiatriques Saint-Georges et Sainte-Madeleine, assurant pour les enfants, adolescents et adultes le service public de lutte contre les maladies mentales pour tout le département de l'Ain. Il est géré par l'association Orsac (Organisation pour la Santé et l'Accueil), à but non lucratif et situé Avenue de Marboz à Bourg en Bresse (01000). Le CPA régit les CMP (Centre Médico-Psychologique), CATTP (Centre d'Accueil Thérapeutique à Temps Partiel) et les équipes mobiles de tout le département.

J'ai pu travailler avec deux de ces antennes :

Le Centre de Jour de la Rehab : ce centre est composé d'infirmiers et, comme son nom l'indique, a pour but la réhabilitation sociale des patients du CPA.

L'autre antenne avec laquelle j'ai collaboré est le CATTP le Par'Chemin. C'est une imprimerie, mais aussi un atelier thérapeutique de réadaptation au travail, ce qui signifie que le Par'Chemin emploie des patients du CPA afin de les aider à se réinsérer dans le monde du travail. Au moment du stage, une partie des employés travaillait en parallèle sur le développement d'un nouveau blog web pour la Rehab.

2. Environnement de travail et outils utilisés

a. Environnement de travail

Le stage s'est déroulé sur trois sites du CPA : le mercredi à la Rehab, afin de pouvoir échanger avec les rédacteurs du blog actuel, les futurs administrateurs de l'App Rehab, le vendredi matin au Par'Chemin, où j'ai pu travailler sur le développement de la nouvelle version du blog de la Rehab et le reste de la semaine à la DSI[1] du CPA. La DSI[1] ne possédant pas de pôle de développement, j'ai eu droit à beaucoup d'autonomie en ce qui concerne le développement du projet. J'ai eu un PC portable à disposition pour travailler tout le long du stage.

b. Logiciels utilisés

Voici une liste des logiciels utilisés lors de la réalisation du projet :

	Visual Studio Code pour le développement
	Laragon[48] pour lancer le serveur web de développement
	Apache pour lancer le serveur web de production
	L'AVD Manager d'Android Studio pour gérer les émulateurs[2] Android[3]
	Git[49] pour la gestion du projet
	PuTTY[37] pour administrer le serveur de production
	WinSCP[46] pour envoyer des fichiers sur le serveur de production via SFTP[4]
	Expo[44] pour aider au développement en React Native[43]
	HeidiSQL[45] pour administrer la base de données

c. Technologies utilisées

Le projet étant divisé en plusieurs logiciels, j'ai utilisé plusieurs technologies :

	HTML / CSS[13] pour le développement du frontend[5] du tableau de bord
	JavaScript[14] pour la logique du frontend[5] du tableau de bord
	MySQL[10] pour la base de données du tableau de bord
	PHP[9] pour le développement du backend[6] du tableau de bord
	React Native[43] pour le développement de l'application mobile
	Joomla[47] pour le développement du frontend[5] du blog web

3. Présentation du projet

a. Origine du projet

La Rehab est une antenne du CPA ayant pour mission de s'occuper de la réhabilitation des patients. Avec le confinement de 2020, un groupe de rédacteurs s'est formé en son sein pour tenir un blog ayant pour but de maintenir le contact avec les patients malgré l'impossibilité de les voir. Le blog a été un succès, il a donc été maintenu et les rédacteurs ont continué d'y ajouter du contenu régulièrement. A ce moment-là, ils utilisaient Blogger[7], qui leur imposait certaines limites en termes de charte graphique et d'accessibilité. C'est alors qu'est venue l'idée de faire un site personnalisé dédié au blog ainsi qu'une application mobile, permettant une plus grande personnalisation du contenu ainsi qu'une large amélioration de l'accessibilité du blog.

Le projet du site a été confié au Par'Chemin[8] tandis que l'IUT a utilisé l'application mobile comme sujet de projet tutoré, qui a été confié au groupe auquel j'appartenais. C'est donc à ce moment que le cahier des charges du projet a été mis en place, voici les besoins de l'application qui en sont ressortis : l'application devrait pouvoir afficher des articles triés par catégorie ainsi qu'un challenge (sous forme de poster) et contenir un lexique, avec la possibilité de mettre tout le contenu à jour régulièrement.

b. Version déjà existante

Comme dit plus tôt, il existait déjà une version prototype du projet, développée durant l'année. Cette version était composée de 2 logiciels : d'un côté l'application mobile, codée en React Native[43] et de l'autre le tableau de bord, en PHP[9] et MySQL[10] permettant à un administrateur de mettre à jour le contenu de l'application facilement.

Concernant l'application, elle a été développée en React Native[43] dans l'optique de la déployer sur l'App Store[11] ainsi que sur le Google Play Store[12] Elle était articulée de la façon suivante :

- Un Menu principal duquel on pouvait accéder à 6 Catégories d'articles (Sport, Créativité, Cognition, Culture & Infos, Psycho-éducation et Relaxation), à l'intérieur desquelles on trouvait des Thèmes dans lesquels étaient rangés les Articles. Pour chaque article, il était possible de le marquer comme terminé et d'envoyer un commentaire et une note aux rédacteurs.
- Une page Challenge, sur laquelle était affiché un poster lançant un défi aux utilisateurs
- Une page Profil, duquel l'utilisateur pouvait accéder à diverses fonctionnalités telles que l'historique des articles consultés, le Lexique, le Qui sommes-nous (une présentation de la Rehab et du blog) ainsi que des statistiques utilisateur (Catégorie la plus ouverte, nombre d'articles ouverts pendant le mois, nombre d'articles terminés pendant le mois).

L'application a été développée autour de deux principes fondamentaux : la rendre la plus facile d'accès possible et ne collecter aucune donnée personnelle.

Le tableau de bord, quant à lui, a été développé en PHP[9] HTML[13] CSS[13] et JavaScript[14] et était relié à une base de données MySQL[10] Il permettait de visualiser l'entièreté du contenu disponible sur l'application, à savoir Catégories, Thèmes, Articles, Challenge, Lexique, Qui sommes-nous et dispose des fonctionnalités suivantes :

- Ajouter / modifier / supprimer un Thème, Article ou mot du Lexique
- Changer le poster du Challenge
- Editer le Qui sommes-nous
- Visualiser les commentaires utilisateurs

Le tableau de bord possédait aussi un système d'authentification rudimentaire (1 seul identifiant dont le nom d'utilisateur et le hash[29] du mot de passe étaient écrits dans le code).

c. Architecture du projet

Voici un schéma résumant les interactions entre les différents logiciels du projet :

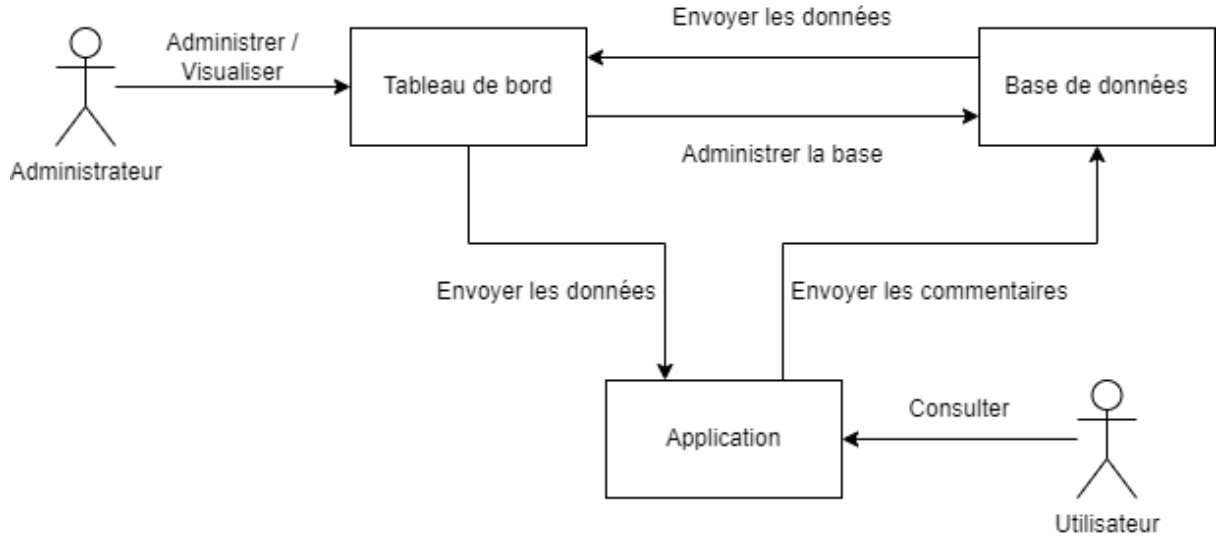

Figure 1 / Diagramme d'environnement

Pour récupérer les données du tableau de bord, l'application se connectait à un fichier JSON[15] disponible sur le tableau de bord contenant toutes les données de la base de données qu'elle parcourait pour en récupérer toutes les données. Ce fichier JSON[15] n'était pas mis à jour automatiquement à jour à chaque modification de la base de données afin d'éviter les conflits avec l'application (risque de récupérer des articles non entiers et impossibilité de vérifier un article avant de le publier). La mise à jour de ce fichier se faisait en appuyant sur un bouton depuis le tableau de bord.

Concernant les données, elles ont été ordonnées de la façon suivante dans une base de données MySQL[10] :

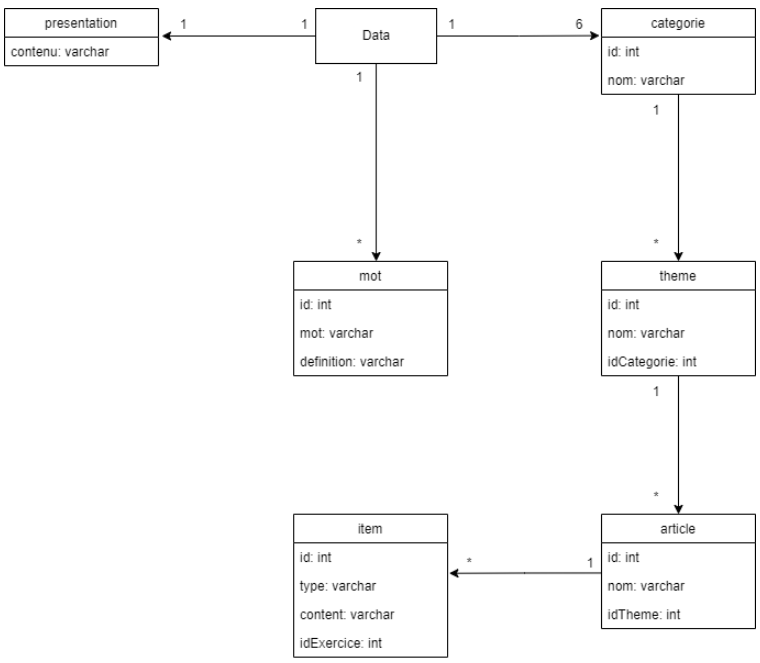

Figure 2 / Modèle de données

Comme dit plus haut, les articles étaient rangés par Thèmes, eux-mêmes triés dans les six Catégories. On peut voir qu'une ligne de la table contenait seulement un nom, pas de contenu. C'est parce qu'un Article contenait plusieurs items, possédant chacun un type (Texte, Lien, Image) et un contenu (URL de l'image pour les images).

La table présentation contenait une seule ligne, dont la colonne "contenu" contenait le Qui sommes-nous.

d. Missions du stage

Voici les tâches qui m'ont été confiées au cours de ce stage :

- Développer une API[16] faisant la communication entre l'application et le tableau de bord
- Ajouter certaines fonctionnalités aux deux logiciels
- Améliorer l'aspect visuel de l'application / résoudre des bugs
- Collecter les retours des rédacteurs du blog
- Déployer l'application et le tableau de bord
- Aider à développer la nouvelle version du blog web (au Par'Chemin[8])

Le sujet du stage a donc été de terminer le projet et le déployer, ce qui a impliqué plusieurs missions. La plus lourde a été de remplacer le fichier JSON[15] servant à envoyer les données à l'application. En effet, même s'il fonctionnait, ce fichier posait problème : il envoyait tout le contenu à l'application, qui le téléchargeait entièrement à chaque lancement. Une API[16] personnalisée a donc été développée afin de remédier à ce problème. L'autre problème majeur du projet se trouvait au niveau de l'authentification du tableau de bord : une simple identification dont le nom d'utilisateur et le mot de passe étaient stockées dans le code. Après cela, en écoutant les retours des rédacteurs du blog, il a fallu ajouter et modifier certaines fonctionnalités des deux logiciels et retravailler plusieurs aspects graphiques de l'application. Une fois la version finale de l'application obtenue, il a fallu la déployer.

Ces tâches ont été menées à bien tout en échangeant avec les rédacteurs du blog le mercredi et en développant le blog web avec le Par'Chemin[8] le vendredi matin.

Réalisation du projet

1. l'API[16]

Comme dit plus tôt, dans la première version du projet, l'application récupérait les nouvelles données en téléchargeant un fichier JSON[15] contenant toutes les données et mis à jour par un administrateur. L'inconvénient majeur de ce système est le fait que beaucoup de données dupliquées transitaient, étant donné que l'application écrasait puis téléchargeait tout le contenu à chaque lancement.

La première chose à faire pour faire fonctionner cette API[16] a été de modifier la base de données pour ajouter les champs suivants dans chaque table :

- createdAt : Timestamp[17] de la création de la ligne
- modifiedAt : Timestamp[17] de la dernière modification de la ligne
- isDeleted : Booléen[18] indiquant si la ligne est supprimée ou pas

Il a ensuite fallu modifier les fonctions PHP[9] servant à l'insertion pour qu'elles ajoutent automatiquement les Timestamp[17] pour les champs createdAt et modifiedAt et qu'à chaque fois qu'une modification soit faite et changer le système de suppression : au lieu de supprimer la ligne de la base de données, son champ isDeleted est assigné à 1 et son champ modifiedAt mis à jour.

Sur demande des rédacteurs du blog, le bouton "Mettre à jour l'application" (bouton permettant de publier les modifications au moment voulu au lieu qu'elles ne soient publiées automatiquement au moment de leur création) a été gardé, ce qui a impliqué l'insertion du champ isReady dans toutes les tables de la base de données. C'est un booléen[18] indiquant si une ligne est prête à être publiée, il vaut donc 0 par défaut, devient 1 pour toute la base de données à l'appui sur le bouton "Mettre à jour l'application" et redevient 0 si la ligne est modifiée.

L'API[16] développée fonctionne de la façon suivante : lorsque l'application envoie une requête contenant le Timestamp[17] de sa dernière connexion (pour ne recevoir que les nouvelles données), l'API[16] lui renvoie au format JSON[15] les données suivantes :

- Les nouvelles données (isReady = 1, createdAt > Timestamp[17]
- Les données modifiées (isReady = 1, modifiedAt > Timestamp[17] isDeleted = 0)
- Les données supprimées (isReady = 1, modifiedAt > Timestamp[17] isDeleted = 1)
- Le "Qui sommes-nous" (isReady = 1, modifiedAt > Timestamp[17])

Une fois toutes ces données récupérées, elles sont triées (notamment afin d'éviter les doublons dans le cas où un élément aurait été ajouté puis supprimé entre 2 connexions), puis organisées de la façon suivante :

```
{
    "news":{
        "categories":[],
```

```
        "themes":[],
        "articles":[],
        "items":[],
        "mots":[]
    },
    "modified":{
        "categories":[],
        "themes":[],
        "articles":[],
        "items":[],
        "mots":[]
    },
    "deleted":{
        "categories":[],
        "themes":[],
        "articles":[],
        "items":[],
        "mots":[]
    },
    "presentation": ""
}
```

L'arborescence est plus compliquée que dans le fichier JSON[15] d'origine :

```
{
"categories":[],
"themes":[],
"articles":[],
"items":[],
"mots":[],
"presentation": ""
}
```

La supériorité de la nouvelle version se voit au niveau du contenu des champs : là où l'ancienne version contenait tous les éléments ajoutés depuis la création du tableau de bord, la nouvelle version ne contient que les modifications faites entre deux connexions de l'application.

Un autre avantage de cette API[16] est qu'elle ne supprime plus les éléments : elle se contente de les marquer comme supprimés. Cela permet à l'administrateur de retrouver un élément supprimé par erreur.

Une fois que l'application a récupéré toutes ces données, elle les stocke localement via AsyncStorage[19] ce qui lui permet de fonctionner même hors ligne. Les données sont stockées sous forme de paires clé-valeurs, chaque clé étant un équivalent d'une table de la base de données (Catégories, Thèmes, Articles, Items, Mots, Présentation) et chaque valeur un tableau JSON[15]

- Le stockage s'effectue donc de la façon suivante :
- L'application récupère le fichier JSON[15] fourni par l'API[16]
- Le Timestamp[17] de connexion est mis à jour à la date actuelle
- Le contenu du fichier JSON[15] est parsé[20] et divisé en 3 catégories :

- Les nouveautés : pour chaque type (Catégorie, Thème, Article…), les nouveautés sont concaténées au tableau JSON[15] correspondant déjà existant
- Les modifications : pour chaque type, les éléments à modifier du tableau JSON[15] correspondant sont remplacés par ceux venant de l'API[16]
- Les suppressions : pour chaque type, les éléments à supprimer sont retrouvés dans le tableau correspondant puis supprimés

2. Les nouveautés

L'utilité principale de l'application est de servir de blog, donc d'afficher des articles. Comme vous pouvez le voir, il n'existait pas de mise en page possible pour ces derniers, rendant l'affichage brouillon et la lecture globalement désagréable, surtout pour les articles les plus longs.

Figure 3 / Article dans la première version de l'application

Un autre gros chantier du projet a donc été la mise en place d'un éditeur de texte personnalisé. Pour décider quelles fonctionnalités allaient être ajoutées, il a d'abord été décidé de parcourir le blog actuel de la Rehab afin d'observer leurs habitudes de mise en page. Suite à cela, voici les fonctionnalités à implanter dans l'éditeur de texte :

- Mettre en gras
- Souligner
- Mettre en italique
- Effacer la mise en page
- Modifier la taille de la police

- Changer la couleur du texte

Les deux difficultés majeures de cette fonctionnalité ont été la prise en charge du chevauchement des différents styles et la conversion entre deux langages : sur le tableau de bord, le texte est affiché en HTML[13] tandis que sur l'application, les articles doivent être au format JSX[21].

Une première version de la fonctionnalité a donc été mise en place de la façon suivante :

Côté tableau bord, l'éditeur de texte fonctionnait de la façon suivante : un bouton par fonction, sélectionner du texte puis appuyer sur un bouton faisait apparaître une balise personnalisée :

- <g> pour gras
- pour italique
- <s> pour souligné
- <p10> pour la taille de police (10 correspondant à la taille)
- <#FFFFF> pour la couleur du texte (code hexadécimal)

Chaque balise possédait sa version fermante (</g>,).

Côté application, j'ai d'abord voulu simplement remplacer les balises personnalisées par leur équivalent JSX[21] en leur ajoutant des styles, sans savoir que la conversion de String[22] vers JSX[21] n'est pas prise en charge. Il a donc fallu mettre en place un algorithme fonctionnant de la façon suivante :

- Stocker toutes les positions des balises dans 2 tableaux (un pour les balises ouvrantes, l'autre pour les versions fermantes)
- Faire récursivement[23] :
    - Prendre la dernière balise fermante du tableau
    - Trouver sa version ouvrante
    - Supprimer les deux balises des tableaux
    - Parcourir le texte entre les deux balises
    - Tant que ce texte contient des balises rappeler la fonction en lui envoyant les deux tableaux de balises et le texte situé entre les deux balises
    - Renvoyer un objet JSX[21] contenant le texte situé entre les deux balises et le style associé ainsi que les deux tableaux modifiés
- Afficher le résultat

Cette mise en place fonctionnait mais possédait plusieurs faiblesses :

- Les balises n'étaient pas interprétées dans l'éditeur de texte, donc visibles. Pour compenser cela, un bouton montrant une prévisualisation dans une autre fenêtre a été ajouté, mais ne fonctionnait pas en temps réel.
- La suppression des balises se faisait à la main dans l'éditeur en supprimant les balises (système peu pratique, surtout sachant que les utilisateurs du tableau de bord ne seraient

pas des informaticiens, donc pas habitués aux balises type HTML[13])

- L'impossibilité de faire se chevaucher les styles (exemple : chev**auchement**)
- Le manque de visibilité du texte dans l'éditeur, dû aux balises
- Les erreurs n'étaient pas traitées (un résidu de balise aurait provoqué une erreur)

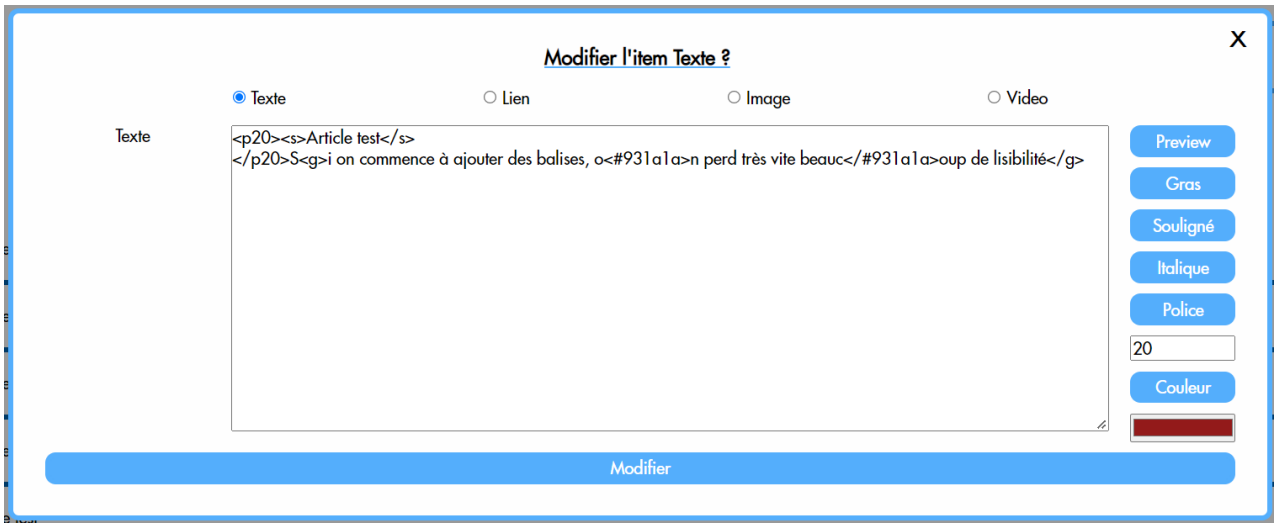

Figure 4 / Exemple de balises dans l'éditeur de texte

Il a donc été décidé de remanier le système de 0 avec pour objectif de faire en sorte que les balises ne soient pas visibles dans l'éditeur de texte. Un début de solution basée sur Editorjs.io (bibliothèque de JavaScript[14] permettant de stocker du texte stylisé au format JSON[15] a été développé et fonctionnait côté tableau de bord, mais la conversion du JSON[15] vers du JSX[21] s'est avérée encore plus complexe que pour la version précédente, ce qui m'a poussé à chercher une troisième solution.

En me penchant sur les bases de Editorjs.io, j'ai trouvé la fonction native JavaScript[14] `document.execCommand(aCommandName, aShowDefaultIU, aValueArgument )`, permettant d'insérer des balises HTML[13] dynamiquement dans une page HTML[13] Il a ensuite fallu remplacer la TextArea[24] utilisée (cet élément n'interprétant pas les balises HTML[13] par une div[27] rendue éditable par l'utilisateur pour avoir un éditeur de texte dynamique côté tableau de bord. En ce qui concerne le stockage dans la base de données et l'API[16] les problèmes de sécurité ont tout simplement été résolus en ajoutant des guillemets autour de chaque article.

L'étape suivante a été d'afficher ces articles dans l'application. Une tentative de modification de l'algorithme décrit plus tôt a été faite mais n'a pas abouti pour des problèmes de complexité : les balises HTML[13] utilisées étaient plus compliquées à analyser que les anciennes balises personnalisées et contenaient parfois plusieurs styles, rendant les possibilités à couvrir trop grandes. Je me suis donc tourné vers une bibliothèque React Native[43] développée par la communauté : RenderHTML[25]. Comme son nom l'indique, ce composant est capable, à partir d'un texte composé de balises HTML[25], d'afficher un objet JSX[21] semblable.

La bibliothèque RenderHTML[25] et la fonction `document.execCommand()` étant basées sur des

versions de HTML[13] différentes, il a tout de même fallu développer l'algorithme suivant afin que l'application puisse faire la conversion:

- Remplacer chaque "" par "</span">
- Parcourir chaque caractère du texte et à chaque balise :
    - Si c'est une balise à modifier :
        - Chercher tous les styles contenus dans la balise
        - Remplacer la balise par sa version interprétable par RenderHTML[25]
- Envelopper le texte dans une balise pour lui appliquer le style par défaut d'un article
- Afficher le texte via RenderHTML[25]

Cette méthode a permis d'obtenir le même rendu dans l'application et le tableau de bord, malgré la différence de langage.

En lisant la documentation de RenderHTML[25], j'ai trouvé plusieurs paramètres intéressants, ce qui a permis d'implémenter les fonctionnalités suivantes :

- Liens hypertextes : permet d'ajouter des liens hypertexte, remplaçant l'item Lien et ajoutant plus de lisibilité lors de l'écriture d'un article.
- Nettoyer le texte : enlève toutes les balises du texte, notamment utile lors de copié-collés
- Rétablir la police par défaut : rétablit la police Arial sur le texte sélectionné, une alternative au bouton du dessus lors de copié-collés où seule la police est à changer

Voici donc le résultat final de l'éditeur de texte :

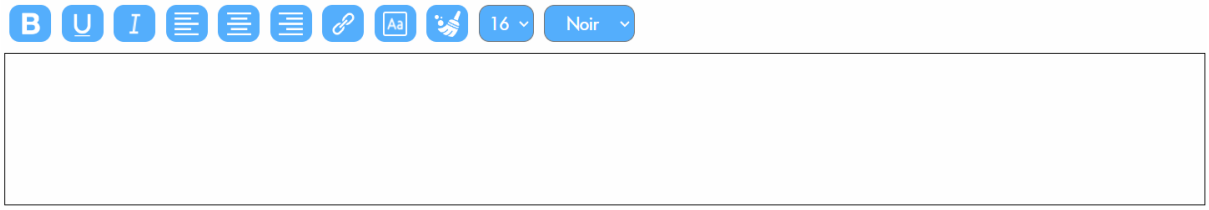

Figure 5 / Version finale de l'éditeur de texte personnalisé

Durant l'analyse du blog, je me suis rendu compte qu'il contenait un nombre important de vidéos, toutes hébergées sur Youtube. Dans la première version, la seule façon de mettre une vidéo était d'insérer un lien vers celle-ci pour rediriger l'utilisateur. Pour améliorer l'accessibilité, il a donc été décidé d'insérer directement les vidéos dans les articles.

Toujours en cherchant du côté des bibliothèques développées par la communauté React Native[43] la bibliothèque YoutubePlayer[26] a été trouvée. Comme son nom l'indique, elle permet d'afficher des vidéos Youtube dans une application React Native[43] ce qui nous convenait parfaitement, étant donné que toutes les vidéos du blog sont hébergées sur cette plateforme.

La principale difficulté de cette implémentation est le fait que pour fonctionner, YoutubePlayer[26]

requiert l'identifiant d'une vidéo et non un lien menant vers celle-ci (c'est à dire seulement cette partie de l'URL : https://www.youtube.com/watch?v=aY1QuPR1ibY). Ce détail couplé au fait qu'il existe plusieurs sortes d'URL menant vers la même vidéo (https://youtu.be/aY1QuPR1ibY , https://www.youtube.com/watch?v=aY1QuPR1ibY&start_radio=1 …) a ajouté une difficulté au développement. Il a d'abord été décidé que les rédacteurs devraient eux-mêmes trouver et ajouter l'identifiant dans le tableau de bord, mais il s'est trouvé que le risque d'erreur était trop grand.

La mise en place d'une expression régulière a donc été nécessaire :

```
function getIdFromUrl(url){
  let regex =
/^.*((youtu.be\/)|(v\/)|(\/u\/\w\/)|(embed\/)|(watch\?))\??v?=?([^#&?]*).
*/
  let match = url.match(regex)
  return (match && match[7].length == 11) ? match[7] : false
}
```

Cette expression régulière a permis de retourner l'identifiant d'une vidéo Youtube à partir d'une URL en couvrant avec les types d'URL suivants :

http://www.youtube.com/watch?v=aY1QuPR1ibY&feature=feedrec_grec_index
http://www.youtube.com/user/IngridMichaelsonVEVO#p/a/u/1/aY1QuPR1ibY
http://www.youtube.com/v/aY1QuPR1ibY?fs=1&hl=en_US&rel=0
http://www.youtube.com/watch?v=aY1QuPR1ibY#t=0m10s
http://www.youtube.com/embed/aY1QuPR1ibY?rel=0
http://www.youtube.com/watch?v=aY1QuPR1ibY
http://youtu.be/aY1QuPR1ibY

Cette fonction est appelée juste avant le rendu d'une vidéo dans l'application et son retour est envoyé à YoutubePlayer[26], qui se charge d'aller chercher la vidéo et de l'afficher.

Figure 6 / Rendu d'une vidéo dans un article grâce à YoutubePlayer[26]

Du côté du tableau de bord, un bouton "Video" faisant apparaître un champ texte pour renseigner l'URL de la vidéo a été ajouté dans la boîte de dialogue permettant d'ajouter un Item.

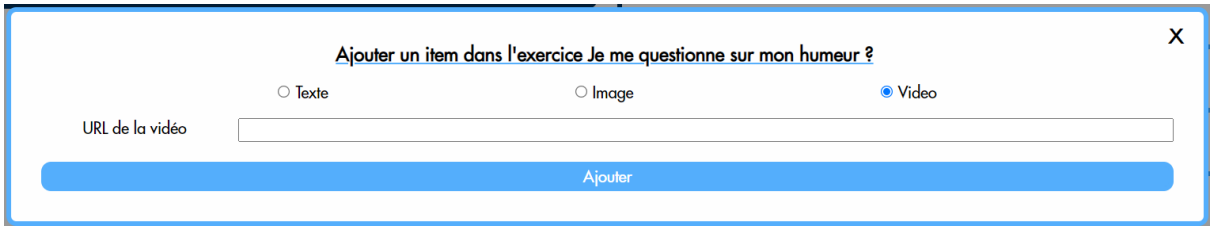

Figure 7 / Boîte de dialogue d'ajout d'Item lorsque le bouton Video est coché

a. L'application

En utilisant la première version de l'application, un problème s'est très vite posé au niveau de la visibilité des nouveaux articles. En effet, pour trouver un nouvel article, il fallait se rendre dans la Catégorie puis le Thème correspondant. Sachant qu'il y avait 6 Catégories et un nombre grandissant de Thèmes, trouver un nouvel article sans aucune indication était long et fastidieux.

Il a donc fallu remédier à ce problème et pour cela, deux solutions complémentaires ont été trouvées.

La première a été de faire en sorte que l'application affiche une boîte de dialogue à son lancement s'il y avait du nouveau contenu. Cette implémentation a été simple étant donné qu'il suffit à l'application d'afficher ou non la boîte de dialogue en fonction de si l'objet `news` venant de l'API[16] est vide ou non.

Figure 8 / Boîte de dialogue indiquant la présence de nouveau contenu

La deuxième solution a consisté en un système de pastilles rouge indiquant le contenu non lu (à la différence de la boîte de dialogue, avertissant l'ajout de contenu dans l'application depuis la dernière ouverture).

La première étape du développement de cette fonctionnalité a été de reconnaître les nouveaux articles de ceux déjà lus. Pour cela, au moment de l'analyse de l'objet `news.articles` de l'API[16] il a fallu ajouter à chaque élément un champ booléen[18] `isNew`, ayant pour valeur par défaut 1. Au moment de l'ouverture d'un article, son champ `isNew` était assigné à 0. La dernière étape a été d'ajouter une div[27] rouge et ronde, de la positionner dans le coin haut gauche d'un élément de la liste d'articles et de faire en sorte que son opacité soit équivalente au champ `isNew` de l'article auquel elle correspond.

Figure 9 / Liste d'articles dont le deuxième a déjà été ouvert

Pour rendre cette fonctionnalité vraiment utile, il a ensuite fallu la propager aux Thèmes, mais avec un fonctionnement différent : au lieu d'indiquer un Thème non ouvert, la pastille devrait indiquer qu'il y a des articles non ouverts dans un Thème. On retrouve donc le champ `isNew`, mais celui-ci est déclaré à 0 par défaut. Au moment de l'affichage de la liste de Thèmes, le tableau JSON[15] contenant les articles est récupéré, puis parcouru : pour chaque Thème, si au moins un article est son enfant (champ `parentId` de l'article égal au champ `id` du Thème) et que le champ `isNew` de l'article vaut

1, alors le champ `isNew` du Thème est assigné à 1. Comme pour les articles, on retrouve le système de div[27] ronde et rouge dont l'opacité est équivalente au champ `isNew` de l'élément auquel elle est rattachée. La même logique a été utilisée lors de l'appui sur le bouton retour depuis un article.

Est ensuite venu le moment de propager des pastilles aux Catégories, présentes sur l'écran principal. Cette fonctionnalité n'ayant pas été pensée dans le modèle de données (car ajoutée à la fin du projet) a posé un problème au niveau de l'architecture : il n'existait aucun lien direct entre un article et la Catégorie à laquelle il appartenait. Le seul lien se trouvait en passant par les Thèmes. Il a donc fallu développer un algorithme suivant cette logique :

- Récupérer le tableau JSON[15] contenant toutes les Catégories
- Récupérer le tableau JSON[15] contenant tous les Thèmes
- Récupérer le tableau JSON[15] contenant tous les Articles
- Pour chaque Catégorie :
    - Filtrer tous les Thèmes appartenant à la Catégorie (grâce au champ `parentId`)
    - Pour chaque Thème filtré :
        - Filtrer tous les Articles appartenant au Thème (grâce au champ `parentId`)
        - Pour chaque Article filtré :
            - Si son champ `isNew` vaut 1 : retourner 1
    - Retourner 0
    - Assigner au champ `isNew` de la Catégorie la valeur retournée précédemment

Comme pour les deux systèmes précédents, on retrouve ensuite la pastille rouge dont l'opacité équivaut au champ `isNew` de la Catégorie à laquelle elle est rattachée.

Le même algorithme est utilisé à l'appui du bouton retour depuis un Thème, ne prenant cette fois en compte que la Catégorie parent, pour des soucis d'optimisation.

L'algorithme de mise en place des pastilles étant long, ajouté au temps de requête de l'API[16] puis du traitement des données, l'apparition de certains composants était visible lors du lancement de l'application, en plus de faire crash l'application en cas d'appui sur un bouton avant le chargement entier des données. La mise en place d'un splashscreen[28] artificiel en plus de celui déjà existant a donc été nécessaire. Le nouveau fonctionnement de l'application à son lancement suit maintenant l'algorithme suivant :

- Le splashscreen[28] natif est affiché
- L'application charge ses fonctions primaires
- Le splashscreen[28] artificiel est affiché
- L'application envoie la requête à l'API[16]
- La réponse de l'API[16] est traitée
- Le calcul des champs "isNew" de tous les éléments est calculé
- Les éléments de la page principale sont rendus sous le splashscreen[28] artificiel

- Le splashscreen[28] artificiel disparaît

Lors de la phase finale, il a été observé quelques rares problèmes de synchronisation entre le tableau de bord et l'API[16], résultant en problèmes de contenu dans l'application. N'ayant pas de solution miracle pour parer ces aléas (mauvaise connexion, fermeture prématurée de l'application), un bouton "Un problème ?" a été mis en place sur la page Profil.

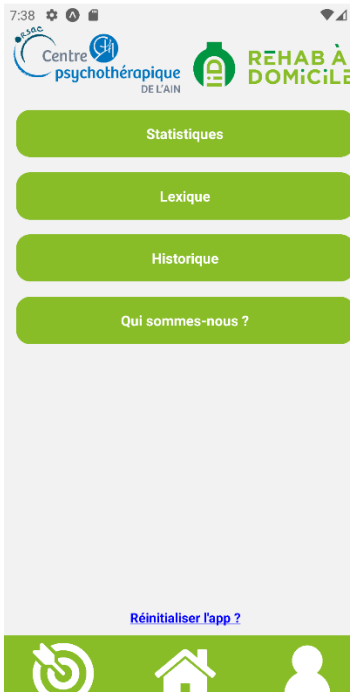

Figure 10 / Page Profil de l'application

L'appui sur ce bouton fait apparaître une boîte de dialogue proposant à l'utilisateur de réinitialiser les données de l'application, puis de la redémarrer. Confirmer fait appel à la fonction de suppression de tout le contenu d'AsyncStorage[19], supprimant tout le contenu de l'application, à savoir les articles (Catégories, Thèmes, Articles, Items), le Lexique, le Challenge, le Qui sommes-nous, l'historique, les statistiques utilisateurs et le Timestamp[17] indiquant la date de la dernière connexion à l'API[16] L'impossibilité de programmer le redémarrage d'une application React Native[43] automatiquement a fait qu'il a fallu demander à l'utilisateur de le faire manuellement afin de recharger les données comme au premier démarrage de l'application.

b. Le tableau de bord

Comme dit lors de l'introduction, pour des raisons de temps, la première version du tableau de bord était sécurisée par un système d'authentification rudimentaire : il n'existait d'un seul utilisateur possible, dont l'identifiant et le hash[29] du mot de passe étaient écrits dans le code PHP[9].

La mise en place du tableau de bord sur les serveurs externes du CPA a donc nécessité une refonte

totale de ce système incomplet. Un premier système d'authentification a donc été conceptualisé comme tel :

- Un administrateur (identifiant / mot de passe)
- Des utilisateurs (identifiant / mot de passe)
- La possibilité pour l'administrateur d'ajouter / supprimer un utilisateur
- Une fonction "Mot de passe oublié", demandant l'accord à l'administrateur pour modifier le mot de passe d'un utilisateur

Ce système a été abandonné au profit d'un autre déjà existant, plus performant et plus sécurisé : CAS[30](Central Authentication Service). Ce service est déjà utilisé par le CPA et permet de centraliser la connexion à tous les services utilisés par le personnel. Cela permet à chaque utilisateur de posséder un seul couple identifiant / mot de passe pour tous les services, mais surtout de réunir toutes les authentifications à un seul endroit, simplifiant énormément la sécurisation de l'ensemble des services du CPA.

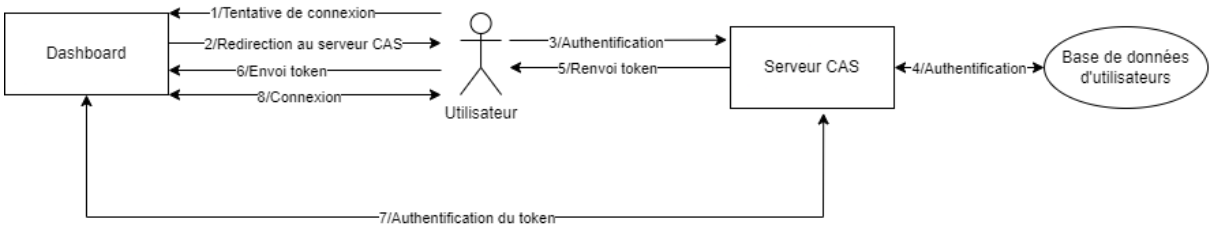

Figure 11 / Schéma du fonctionnement de CAS[30] lors d'une connexion

Explication du schéma :

1/ : L'utilisateur tente d'accéder au tableau de bord

2/ : Le tableau de bord redirige l'utilisateur sur le serveur CAS[30] du CPA

3/ : L'utilisateur s'authentifie au niveau du serveur CAS[30]

4/ : Le serveur CAS[30] vérifie l'authentification de l'utilisateur dans la base de données Active Directory[31] contenant tous les utilisateurs des services du CPA ainsi que les autorisations de chacun

5/ : Le serveur CAS[30] renvoie un jeton unique et périssable à l'utilisateur

6/ : Le jeton est automatiquement renvoyé au tableau de bord

7/ : Le tableau de bord authentifie le jeton auprès du serveur CAS[30]

8/ : Le tableau de bord autorise la connexion de l'utilisateur

En plus de tous les avantages cités précédemment, l'utilisation du service CAS[30] permet une authentification personnelle (chaque utilisateur possède son instance). Ceci est représenté par le bouton de déconnexion (à droite), indiquant quel utilisateur est connecté :

Figure 12 / Boutons de navigation du tableau de bord

Cela a aussi permis d'identifier l'auteur de chaque modification dans la base de données, via un champ `modifiedBy`, récupérant automatiquement l'adresse email de l'utilisateur ayant modifié une ligne pour la dernière fois.

id	nom	parentId	isReady	modifiedAt	createdAt	isDeleted	modifiedBy
8	Cuisine	1	1	2022-05-02 13:52:40	2022-04-22 09:17:32	0	pablo.guinard@orsac-cpa01.fr
9	On poursuit les présentations	1	1	2022-04-25 15:28:38	2022-04-22 09:17:32	0	pablo.guinard@orsac-cpa01.fr
10	J'ai le smile !	1	1	(NULL)	2022-04-22 09:17:32	0	pablo.guinard@orsac-cpa01.fr
11	C'est la rentrée !	1	1	2022-04-26 14:05:41	2022-04-22 09:17:32	0	pablo.guinard@orsac-cpa01.fr

Figure 13 / Extrait de la table "themes" de la base données

3. Les améliorations du projet

a. L'application

La première modification concernant l'application a été l'ajout d'une description pour le Challenge. En effet, cet ajout n'était pas prévu dans le cahier des charges initial, la description de défi lancé aux utilisateurs se trouvant sur le poster, mais les rédacteurs du blog se sont rendus compte que pouvoir ajouter une description supplémentaire pouvait être une bonne chose.

La mise en place de cette fonctionnalité a majoritairement été de la récupération et de l'adaptation de fonctionnalités déjà existantes : l'éditeur de texte des articles a été récupéré pour le rajouter au niveau du Challenge et la bibliothèque React Native[43] RenderHTML[25] a été insérée dans la page de l'application contenant le Challenge, qui a été réadaptée afin de la rendre défilable vers le bas. Il a aussi fallu ajouter une table challenge dans la base de données, ne contenant qu'une seule ligne et les champ `contenu` (servant à stocker la description), `modifiedAt`, `modifiedBy`, `isReady` et `createdAt`, ainsi que modifier le comportement de l'API[16] afin qu'il prenne en compte la nouvelle table.

Figure 14 / page Challenge de l'application avant / après

Étant donné la simplicité technique du procédé et l'amélioration visuelle qu'il proposait, il a été décidé d'ajouter le même éditeur de texte pour le Qui sommes-nous. Il a donc fallu ajouter le même éditeur au niveau du tableau de bord, puis paramétrer la bibliothèque RenderHTML[25] pour l'adapter à la page. Une discussion a aussi été engagée avec la chargée de communication du CPA afin de savoir quels logos il fallait afficher sur la page.

Figure 15 / page Qui sommes-nous de l'application avant / après

Encore sur demande de la chargée de communication du CPA, le header[32] de l'application a été modifié afin d'apparaître plus petit et de contenir le logo du CPA en plus de celui de la Rehab :

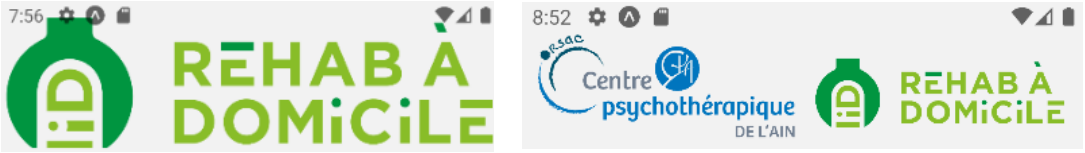

Figure 16 / header[32] de l'application avant / après

Il a ensuite fallu retravailler les visuels qui donnaient à l'application l'impression d'être bâclés. La plus grosse modification concerne la boîte de dialogue permettant à l'utilisateur d'envoyer un commentaire après avoir lu un article.

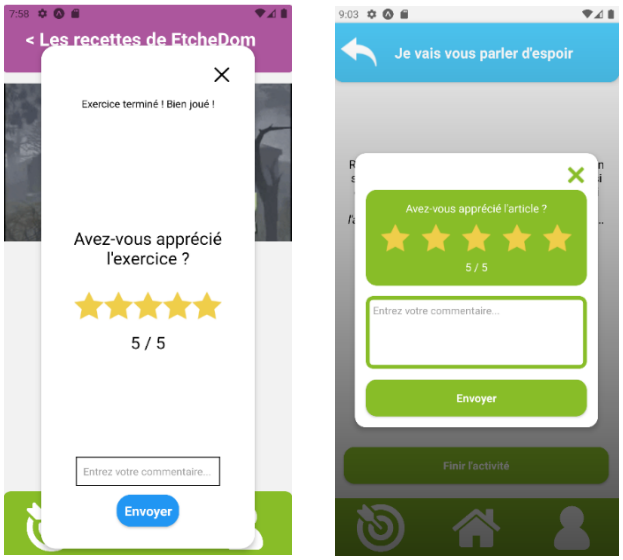

Figure 17 / boîte de dialogue d'envoi de commentaire avant / après

Pour terminer sur l'aspect graphique de l'application, il a fallu la rendre entièrement responsive[33] sur mobile et tablette, une étape négligée dans la première version de l'application.

Figure 18 / Exemples de visuels de l'application non responsives[33]

Concernant le backend[6] de l'application, il y avait un gros problème de fluidité lors du chargement des pages contenant des listes (Thèmes, Articles, Profil, Historique et Lexique). Après des tests, il s'est avéré que ces problèmes venaient du système de navigation : il avait été développé au tout début de l'application, puis des ajouts y avaient été faits au cours du temps, sans se soucier de l'optimisation, résultant en des boucles trop longues, des accès au stockage inutiles et même un dédoublement de données dans le stockage de l'appareil. Une reprise à 0 de la partie navigation de l'application a donc été entreprise, permettant de faire disparaître tous les ralentissements au

chargement de pages.

b. Le tableau de bord

Les principaux changements concernant le tableau de bord ont déjà été évoqués (à savoir l'ajout de l'éditeur de texte pour différentes données, la suppression de l'item Lien, l'ajout de l'item Vidéo et la mise en place de l'authentification grâce à CAS[30]). Il en reste néanmoins deux qu'il est intéressant de mentionner.

Premièrement, tout comme l'application, le tableau de bord n'était pas responsive[33], il a donc fallu remédier à ce problème :

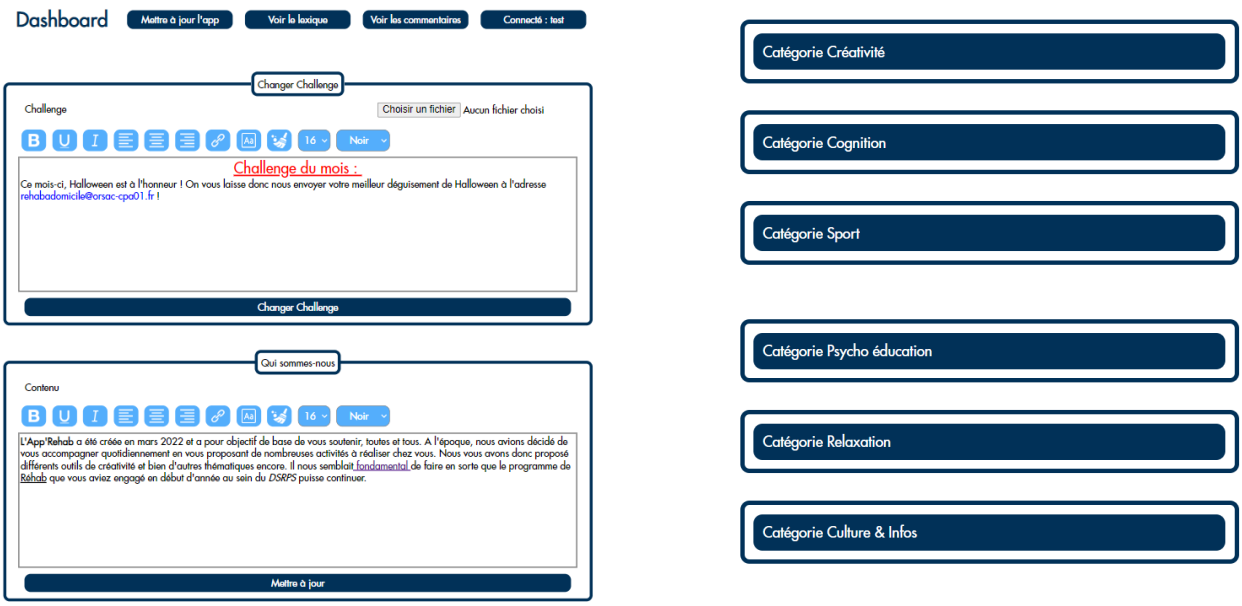

Figure 19 / Version mobile du tableau de bord

L'autre amélioration majeure du tableau de bord a été la mise en place d'un système de mémoire pour l'arborescence. En effet, les articles sont présentés sous forme d'arborescence, ne faisant apparaître que les Catégories par défaut. Cliquer sur une Catégorie fait apparaître les Thèmes associés, pareil pour les Articles et les Items, apparaissant en cliquant sur leur parent.

Figure 20 / Affichage par défaut d'un Catégorie

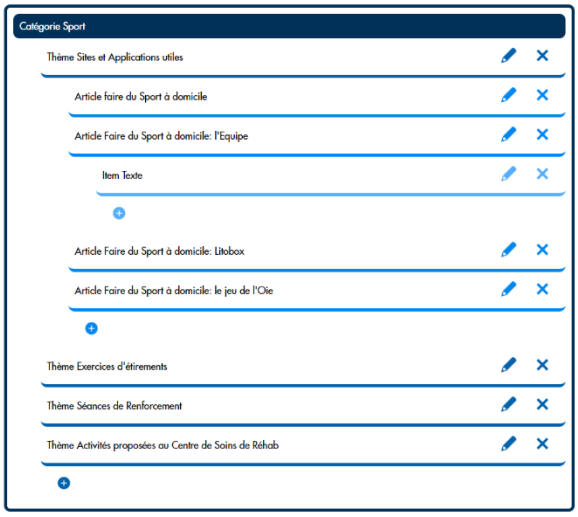

Figure 21 / Catégorie dont l'arborescence est développée

Le problème était qu'à chaque rechargement de la page (donc à chaque ajout / modification / suppression d'un élément) l'arborescence se repliait, rendant fastidieuse l'utilisation de cette partie du tableau de bord.

La solution suivante a donc été mise en place en JavaScript[14] : trois tableaux de booléens[18] (correspondant chacun aux Catégories, Thèmes et Articles) sont stockés dans localStorage[34] du navigateur. Chaque valeur des tableaux correspond à l'état d'un élément (1 pour ouvert, 0 pour fermé) et est modifiée dynamiquement à l'ouverture / fermeture de ce dernier. A chaque chargement de la page, ces tableaux sont récupérés et parcourus, permettant de rétablir l'arborescence telle qu'elle était à la fermeture de la page. L'avantage d'utiliser localStorage[34] est que l'arborescence est sauvegardée même après la fermeture du navigateur et instanciée (deux utilisateurs sur des navigateurs différents ne partageront pas leur arborescence).

4. Le déploiement

(Note : Au moment de la rédaction de ce rapport, le stage n'est pas encore terminé, ce qui implique que la phase de déploiement est encore en cours)

Le but du stage étant de livrer une application fonctionnelle, le déploiement est une étape essentielle et se déroule en plusieurs étapes.

La première a été de rendre le tableau de bord public. Pour cela, une machine virtuelle[35] Ubuntu 20.04.4 a été installée sur un des serveurs du CPA, accessible via PuTTY[37]. Les fichiers du tableau de bord ont été envoyés en SCP (Secure Copy, un protocole permettant le transfert sécurisé de fichiers entre deux ordinateurs via SSH[36]), puis la base de données MySQL[10] a été initialisée grâce à un script MySQL[10]. Il a ensuite fallu adapter le fonctionnement de CAS[30] puis régler quelques problèmes de compatibilité (accès au pare feu[38], différence d'écriture des chemins entre Windows et Linux…) pour enfin rendre le site accessible depuis l'extérieur. J'ai pu observer cette étape

réalisée par le DSI[1] adjoint, voici la procédure qu'il a suivi :

- Déclarer le DNS[40] du serveur (interne puis externe)
- Ajouter les règles d'accès nécessaires au pare feu[38]
- Sur le portail interne (CAS[30]) :
    - Déclaration du DNS[40] (pour les redirections lors de l'authentification)
    - Création des fichiers logs d'erreurs / accès
    - Configuration d'Apache pour prendre en compte le nouveau DNS[40] et les fichiers logs
    - Ajout des règles d'accès (restreindre l'accès aux membres autorisés, sauf sur la partie API[16] du site)
    - Création d'un groupe de membres autorisés à accéder au tableau de bord

Grâce à cette procédure, le tableau de bord est maintenant accessible depuis n'importe quel réseau, sous réserve d'être authentifié et d'appartenir au groupe de membres autorisés. Ceci a permis aux rédacteurs du blog de commencer à se familiariser avec l'utilisation du tableau de bord ainsi que de faire remonter les bugs.

En parallèle de ces tests, une version finale de l'application a été terminée, puis générée en temps qu'APK[41], lui aussi envoyé aux rédacteurs du blog afin de leur permettre de continuer la phase de tests (*Cette étape est encore en cours au moment de la rédaction du rapport*). Cette phase de tests a pour le moment permis de de résoudre certains bugs non vus lors de la phase de développement (les images non stockées dans la base de données du site à cause d'un problème de droits, des problèmes de mise en page dus à des copié-collés…) et est encore en cours.

L'étape suivante du déploiement du projet sera la validation des deux logiciels, puis la publication de l'application sur le Google Play Store[12].

5. Le blog web

Une partie du stage s'est déroulée au Par'Chemin[8], aux côtés de patients en voie de réinsertion professionnelle, aux moments où l'activité de l'imprimerie s'arrêtait pour laisser place au développement du nouveau blog de la Rehab (le vendredi matin). La maquette du site ayant déjà été réalisée, le travail a été divisé en trois parties chronologiques.

Tout d'abord a eu lieu une étape de formation : Nous avons utilisé le framework[42] Joomla[47], que seul le responsable de production savait utiliser au début du développement. Il nous a donc enseigné les bases du framework[42], à savoir la mise en page, l'édition d'articles et la navigation à l'intérieur d'un site web.

A suivi une étape de développement où le but a été de reproduire de façon fonctionnelle et le plus fidèlement possible les maquettes grâce à Joomla[47]. Ma partie du travail a été la réalisation du bas de page ainsi que des bandeaux présents au niveau des pages listant chaque catégorie d'articles.

Figure 22 / Bas de page du nouveau blog de la Rehab

Enfin, comme ce site est une sorte de mise à jour du blog déjà existant, il a fallu récupérer les articles déjà existants pour les ajouter dans le nouveau site. Les rédacteurs de la Rehab ont donc d'abord fait un travail de tri afin de ne garder que les articles jugés encore intéressants et de pouvoir les ranger dans les six Catégories, le système de Thèmes n'ayant pas été retenu pour le site. La procédure utilisée pour intégrer les anciens articles dans le nouveau site a été la suivante :

- Pour chaque catégorie :
    - Télécharger toutes les images / GIF contenus dans les articles de la catégorie
    - Pour chaque article :
        - Copier-coller le contenu de l'article dans l'éditeur de Joomla[47]
        - Supprimer tout le style (pour éviter les résidus HTML/CSS[13])
        - Ajouter les images / GIF
        - Vérifier les liens hypertextes (changer les liens morts / menant vers l'ancien blog)
        - Refaire la mise en page telle qu'elle était

Ce travail n'est pas encore fini au moment de la rédaction de ce rapport, sa fin est prévue fin juin 2022 et la mise en ligne du blog se fera en septembre 2022.

Bilan du stage

1. Bilan professionnel

Professionnellement, j'ai trouvé ce stage très enrichissant sur plusieurs plans. Premièrement, il m'a permis de découvrir le monde de l'entreprise au travers de la DSI[1] du CPA, avec toutes les nouveautés que cela implique, à savoir les réunions de travail, l'ambiance d'une entreprise mais aussi le fait de travailler dans un service où chaque membre possède sa spécialisation. Deuxièmement, j'ai pu mettre utiliser les connaissances que j'ai acquises durant mes deux ans de formation à l'IUT, notamment en JavaScript[14] et en web, mais surtout la façon de travailler : la recherche de solutions, l'apprentissage constant et le versionnage[42] grâce à Git m'ont beaucoup aidé à avancer dans le projet.

Ce stage m'a aussi fait passer par des étapes du travail de développeur jamais vues à l'IUT telles que la méthode agilela communication avec le client au travers des rédacteurs du blog, la phase de tests approfondie d'un logiciel et sa publication.

Enfin, la grande autonomie dont j'ai pu bénéficier m'a permis d'approfondir mes connaissances sur React Native[43], de revoir les bases concernant l'administration d'un serveur web Linux et d'apprendre à me servir du framework[42] Joomla[47].

2. Bilan personnel

Ce stage a également été une expérience intéressante du point de vue humain. En effet, comme dit plus haut, j'ai travaillé avec beaucoup d'autonomie, ce qui m'a appris à mieux connaître mes forces et mes faiblesses, mais aussi mes limites.

Le fait d'avoir travaillé au sein d'un établissement de santé, mais plus particulièrement dans trois antennes différentes du CPA m'a montré trois ambiances de travail et fait rencontrer des profils différents : la DSI[1] m'a montré à quoi pouvait ressembler un secteur informatique dans une entreprise, le Centre de la Rehab m'a permis de discuter avec des psychologues, d'avoir un échange prestataire / client et de participer à leurs réunions, me permettant de mieux comprendre le secteur de la psychologie. Enfin, le temps passé au Par'Chemin[8], à côtoyer des patients du CPA m'a fait mieux me rendre compte de ce que pouvait être une maladie mentale (même si les patients étaient en fin de traitement) et à quel point l'accompagnement de patients psychologiques peut être important.

Conclusion

Pour conclure sur ce stage, même s'il n'est pas encore terminé au moment où je rédige ce rapport, je tiens tout d'abord à remercier l'ensemble des gens avec qui j'ai pu travailler et qui m'ont aidé durant ces semaines. Cette première expérience dans le domaine de l'informatique a été très enrichissante et m'a permis de découvrir plusieurs aspects du travail de développeur, mais aussi de m'initier au monde du soin psychologique, un milieu que je trouve maintenant très intéressant et complexe. De plus, les compétences sociales et professionnelles que j'ai acquises au cours de ce stage me seront particulièrement utiles dans ma future vie professionnelle.

En ce qui concerne le projet, il me tenait particulièrement à cœur, ayant travaillé dessus depuis sa création, donc durant toute l'année scolaire. J'ai réussi à implémenter toutes les fonctionnalités attendues dans le temps imparti, je suis donc satisfait du travail accompli et d'autant plus fier en sachant que je l'ai fait dans une quasi totale autonomie (en ce qui concerne le développement).

Pour finir, j'espère que la fin du stage se passera aussi bien que le reste, que la phase de tests sera validée, aboutissant le projet et me permettant de publier l'application sur le Google Play Store[12] d'où elle pourra être utilisée par les patients.

Glossaire

1. DSI : (Direction / Directeur des Services Informatiques) Pôle Informatique du CPA
2. Emulateur : logiciel permettant de simuler un système d'exploitation
3. Android : Système d'exploitation mobile développé par Google, présent sur la plupart des téléphones / tablettes
4. SFTP : (Secure File Transfer Protocol) Protocole sécurisé permettant le transfert de fichiers entre deux machines via SSH
5. Frontend : partie d'un logiciel visible par l'utilisateur (interface graphique)
6. Backend : partie d'un logiciel non visible par l'utilisateur (partie logique)
7. Blogger : Application web développée par Google permettant de tenir un blog en ligne sans avoir de prérequis en informatique
8. Par'Chemin : Antenne du CPA (voir la partie Présentation de l'entreprise)
9. PHP : Langage permettant de gérer la partie serveur d'un site web
10. MySQL :Langage permettant d'administrer une base de données
11. App store : Magasin d'applications mobiles d'Apple
12. Google Play Store : Magasin d'applications mobiles de Google
13. HTML / CSS : Langages permettant de coder la partie visible d'un site web
14. JavaScript : Langage permettant d'interéagir avec les éléments d'une page web
15. JSON : (JavaScript Object Notation) Format utilisé pour représenter des données structurées à la façon d'objets Javascript
16. API : (Application Programation Interface) Protocole permettant le transfert de données d'un logiciel à un autre
17. Timestamp : nombre de secondes écoulés depuis le 1er janvier 1970
18. Booléen : varaible dont la valeur peut être soit 1 (true) soit 0 (false)
19. AsyncStorage : Bibliothèque de React Native permettant de stocker localement des données
20. Parser : Parcourir un tableau pour en récupérer les informations
21. JSX : Protocole permettant à Javascript d'afficher des composants
22. String : Format de données texte
23. Fonction récursive : Fonction s'appelant elle-même un nombre donné de fois
24. TextArea : Balise HTML affichant une zone de texte éditable par l'utilisateur
25. renderHTML : Bibliothèque de React Native permettant d'afficher des balises HTML en JSX
26. YoutubePlayer : Bibliothèque de React Native permettant d'afficher des vidéos Youtube
27. Div : balise HTML permettant d'affihcer du contenu stylisable
28. Splashscreen : Ecran affiché au démarrage d'une application
29. Hash : Protocole servant à chiffrer un mot de passe
30. CAS : (Central Authentication Service) Service installé sur un serveur permettant de centraliser l'authentification de plusieurs applications

31. Active Directory (AD) : Annuaire de Windows
32. Header : Element placé en haut d'une application
33. Responsive : Application pouvant s'adapter à n'importe quelle taille d'écran
34. LocalStorage : Stockage interne d'un navigateur
35. Machine virtuelle : Simulation d'un ordianteur sur un autre
36. SSH : (Secure SHell) permet d'accéder à un ordinateur via un autre
37. PuTTY : Logiciel permettant d'utiliser le SSH sur Windows
38. Pare feu : dispositif de protection d'un système informatique
39. DNS : (Domain Name Server) Identifiant d'un serveur
40. APK : Fichier servant à installer une application sur Android
41. Framework : ensemble de bibliothèques
42. Versionnage : fait d'avoir plusieurs versions d'un logiciel
43. React Native : framework de Javascript permettant de développer en web et en mobile avec le même code
44. Expo : Framework basé sur Node.js permettant de simplifier le développement en React Native
45. HeidiSQL : interface permettant d'administrer une base de données MySQL
46. WinSCP : Logiciel permettant de transférer des fichiers depuis Windows sur Linux
47. Joomla : framework de PHP / MySQL servant à développer des sites web
48. Laragon : Serveur de développement web pour Windows
49. Git : Logiciel de gestion de versions décentralisé

Sitographie

Editorjs.io : https://editorjs.io/getting-started

RenderHTML : https://www.npmjs.com/package/react-native-render-html

YoutubePlayer : https://www.npmjs.com/package/react-native-youtube

exeCommand : https://developer.mozilla.org/fr/docs/Web/API/Document/execCommand

Forums : https://stackoverflow.com / https://www.geeksforgeeks.org/

Documentation officielle PHP : https://www.php.net/docs.php

Documentation officielle React Native : https://reactnative.dev/docs/getting-started

Documentation CAS : https://www.apereo.org/projects/cas

Annexes

Tableau de bord

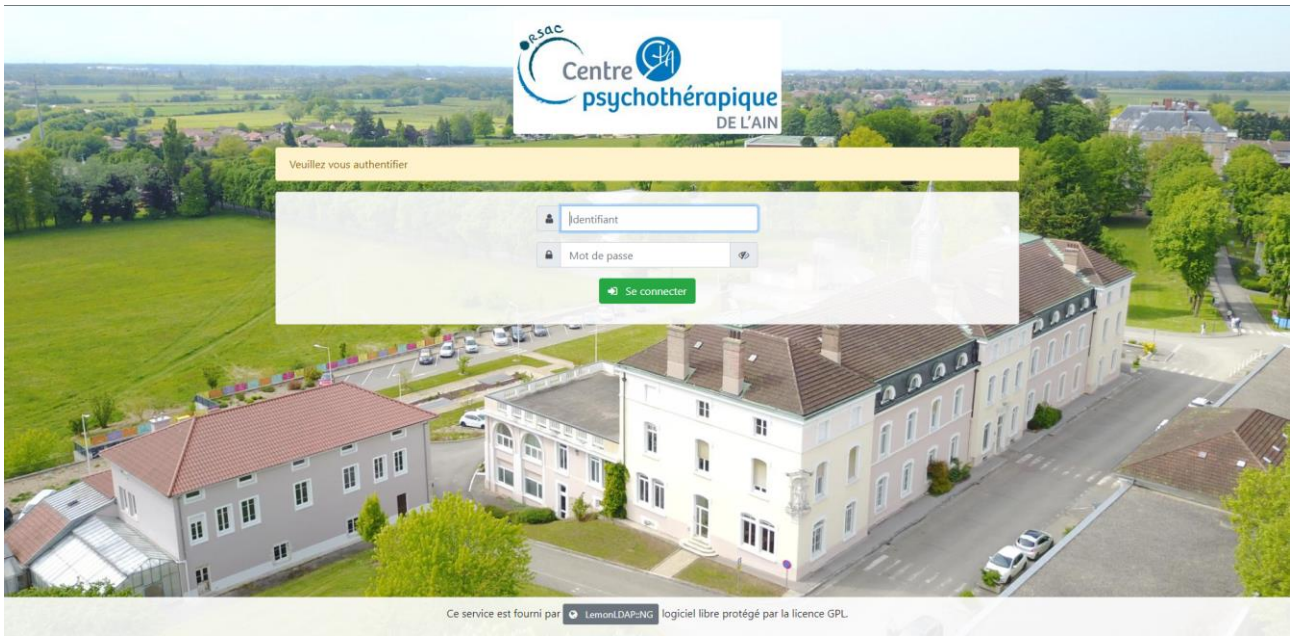

Figure 23 / Page d'authentification CAS[30]

Figure 24 / Page principale du tableau de bord

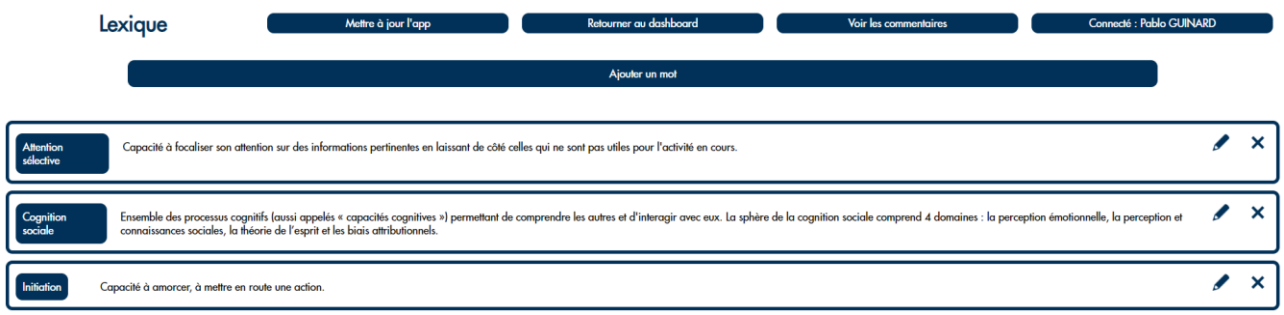

Figure 25 / Page Lexique

Figure 26 / Page Commentaires

Site développé au Par'Chemin

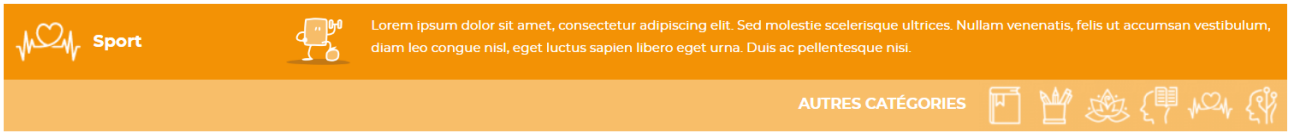

Figure 27 / Template de bandeau

Figure 28 / Bas de page

Report de bugs

Description du bug	Résolu
L'éditeur de texte reste ouvert quand on change le type d'un Item (Vidéo ou Image)	oui
Erreur 403 quand on essaie d'ajouter un Item contenant des balises HTML incompatibles	Interdiction par le serveur
Items images non importées dans la base de données	oui
mise en page non respectée lors de copié collés	solution : copier coller dans un bloc notes
vidéos impossibles à lire	bloquées par le pare feu CPA (donc pas un bug)
impossible de changer la description du challenge sans changer l'image	

Figure 29 / Feuille reportant les bus (extrait)

Application

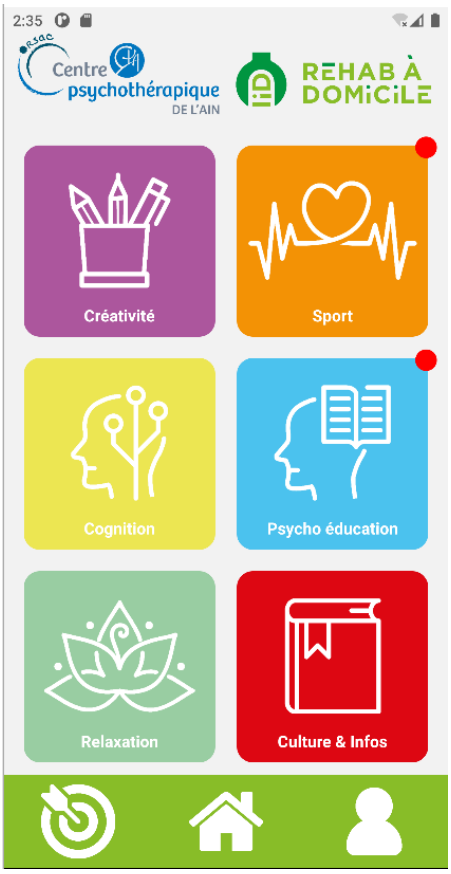

Figure 30 / Menu principal

Figure 31 / Page Challenge

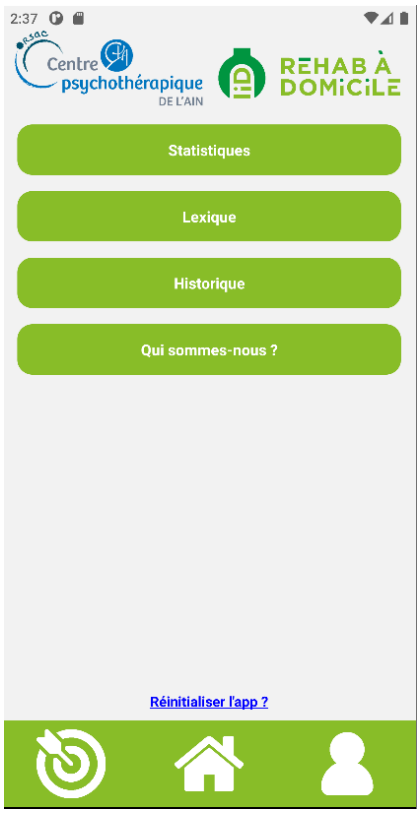

Figure 32 / Page Profil

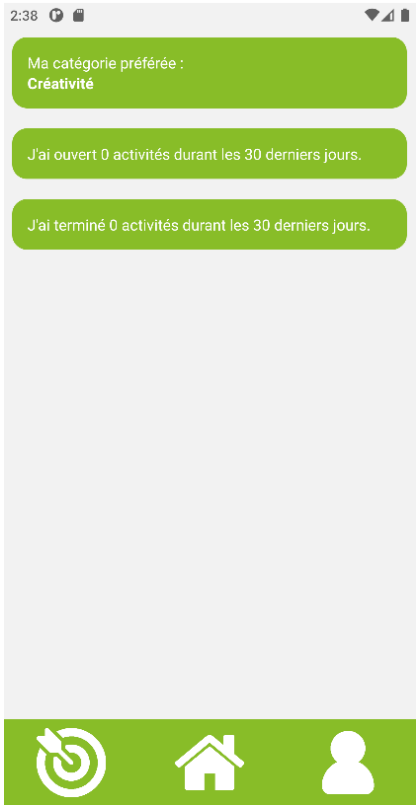

Figure 303 / Page Statistiques

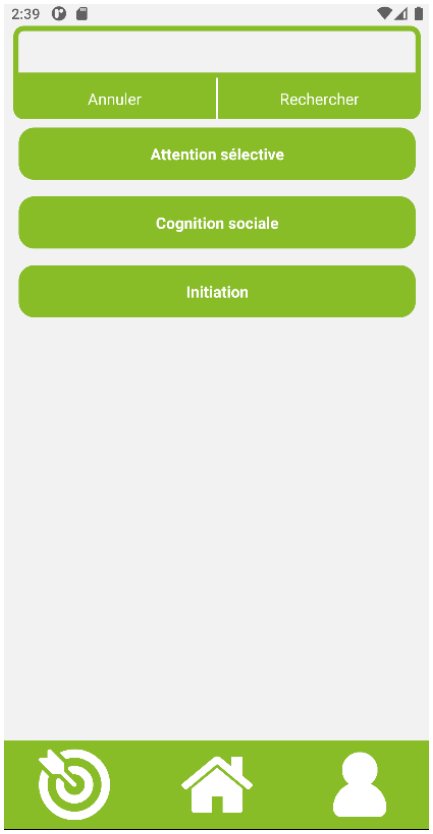

Figure 34 / Page Lexique

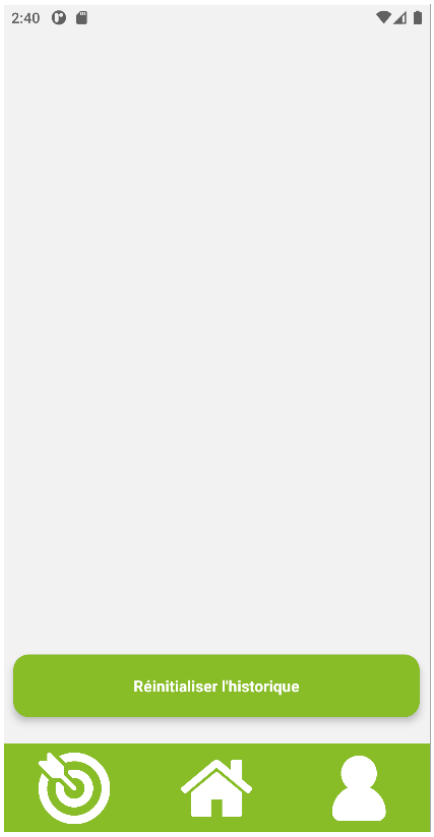

Figure 35 / Page Historique

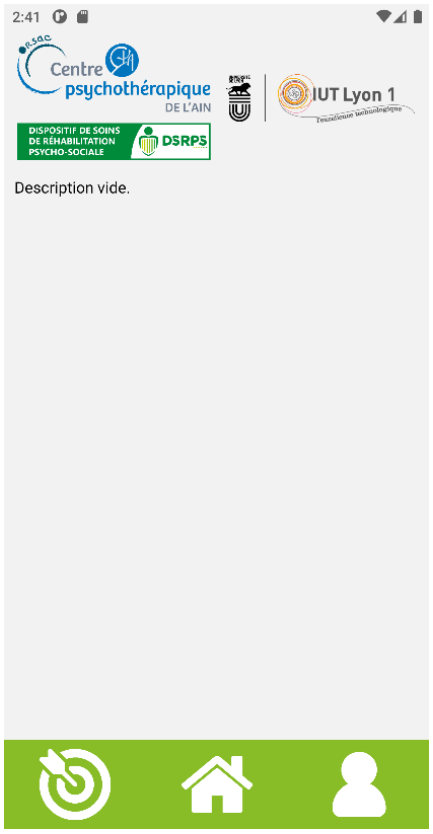

Figure 36 / Page Qui sommes-nous

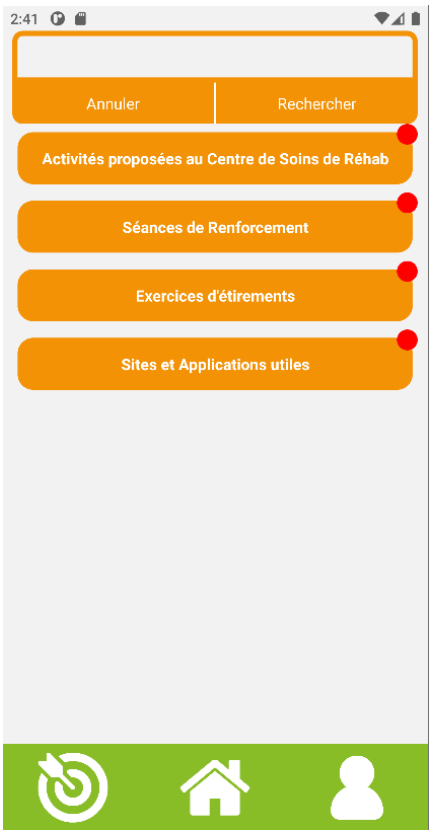

Figure 37 / Page Thèmes

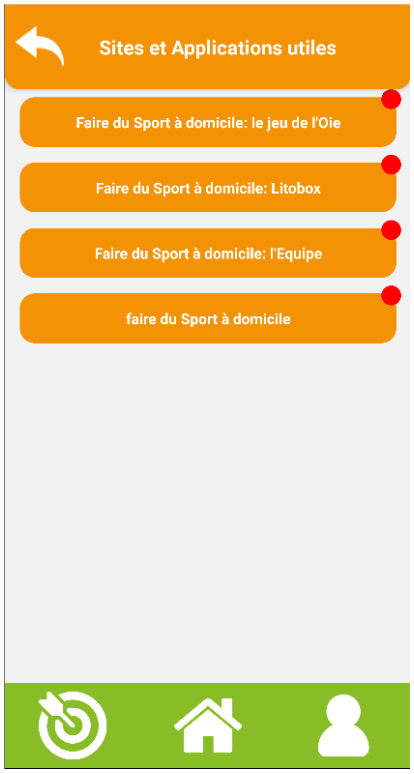

Figure 38 / Pages Articles

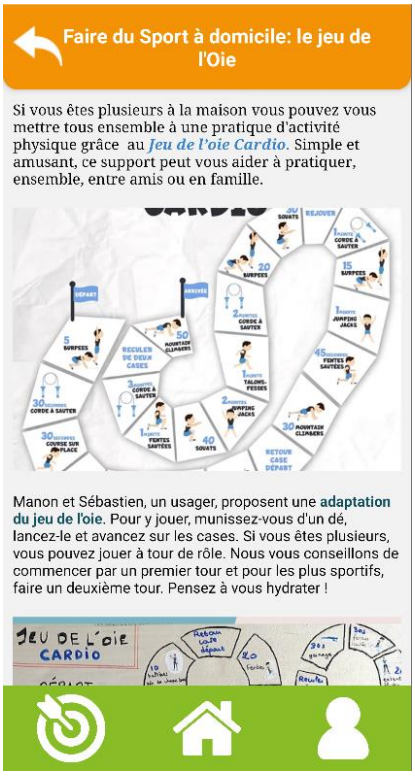

Figure 39 / Exemple d'article

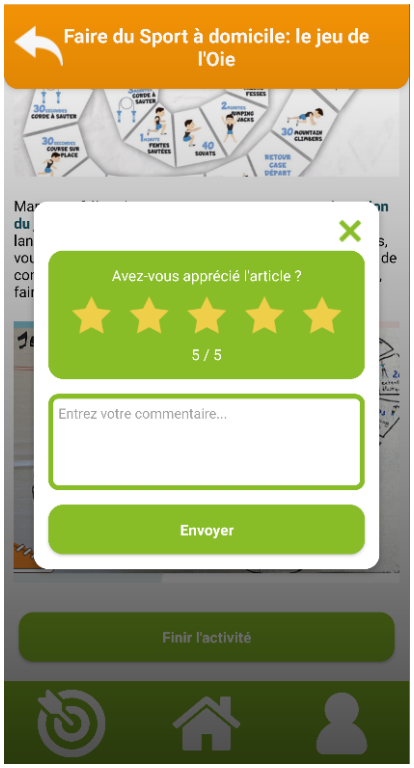

Figure 40 / Boîte de dialogue d'envoi de commentaire

Diagrammes

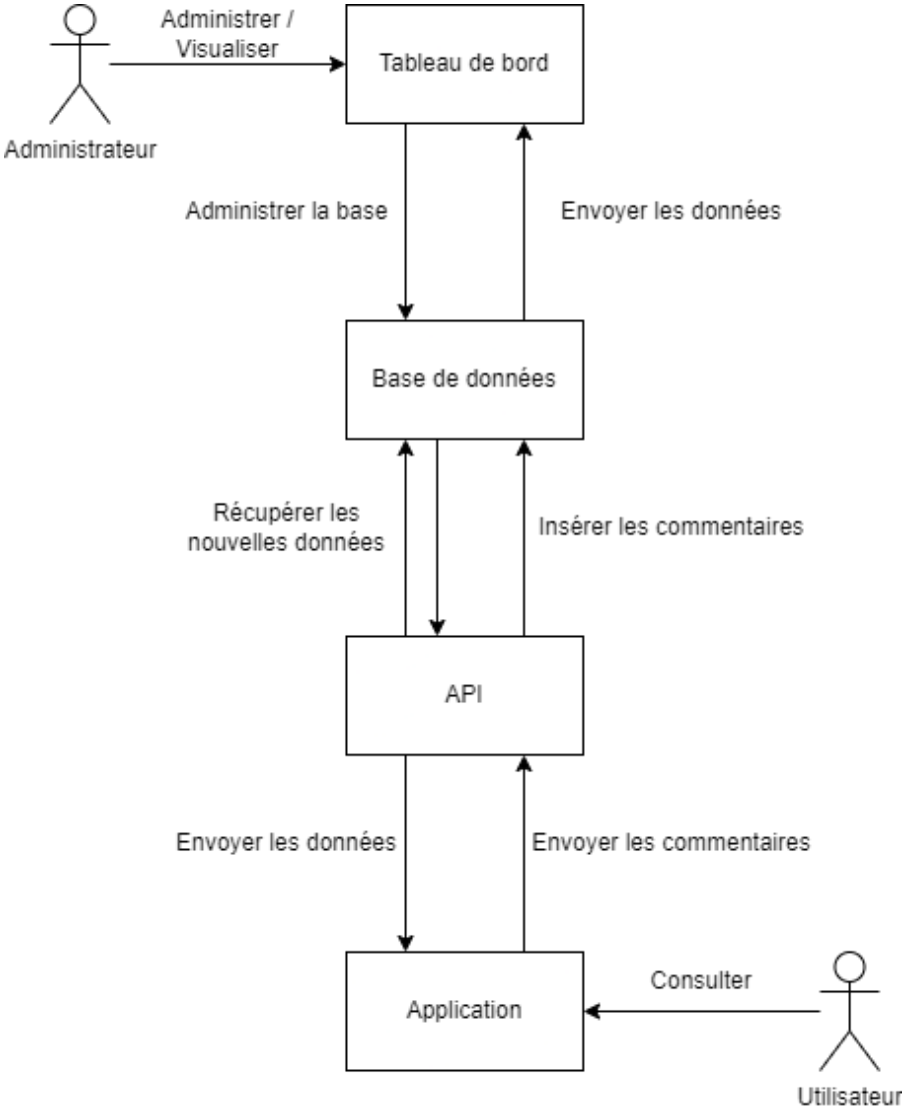

Figure 41 / Diagramme d'environnement

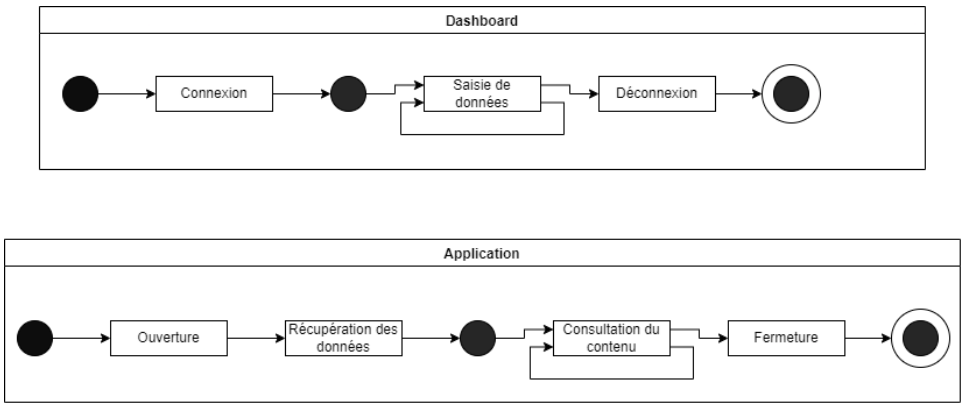

Figure 42 / Diagramme d'états transitions

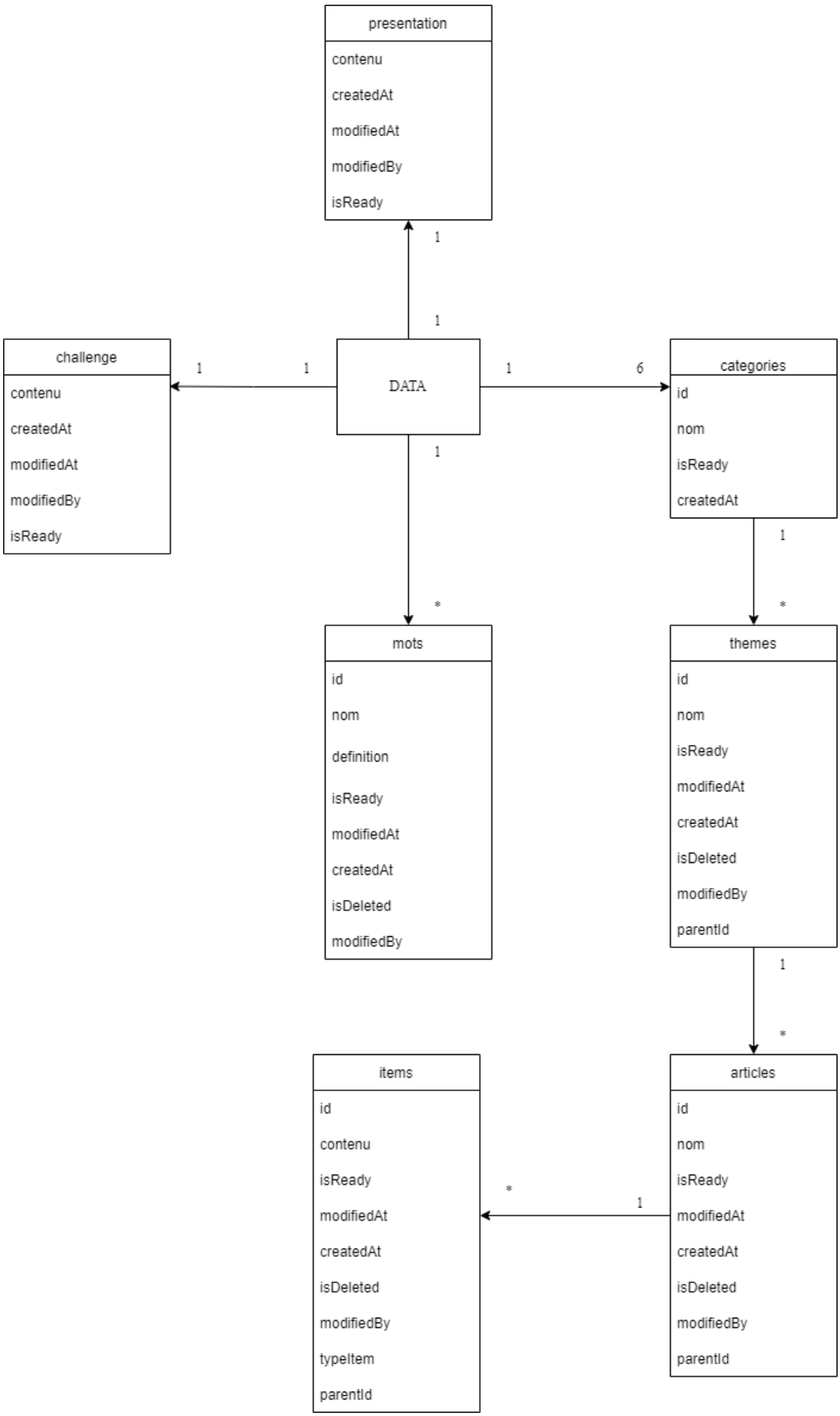

Figure 43 / Modèle de données final

Résumés

Développement d'une application mobile et d'un tableau de bord web

Ce dossier est le rapport de stage de fin de DUT de Pablo Guinard. Ce stage s'est déroulé au CPA (Centre Psychothérapique de l'Ain) du 19 avril au 24 juin 2022. Le sujet a été de reprendre un projet que j'avais déjà commencé afin de le terminer.

Le projet était décomposé en deux logiciels complémentaires. D'un côté, une application mobile développée en React Native (framework de JavaScript) ayant pour principale fonctionnalité d'afficher un blog dont les articles étaient triés sur trois niveaux. L'application embarquais différentes fonctionnalités telles que l'envoi de commentaire aux rédacteurs des articles, un lexique personnalisé et des statistiques utilisateurs. De l'autre côté, un tableau de bord web, développé en PHP / MySQL, permettant de mettre à jour facilement le contenu de l'application (articles, lexique, voir les commentaires). Le tableau de bord intégrait une API permettant de communiquer avec l'application.

Le stage a majoritairement porté sur le développement de l'API, l'amélioration de la lecture des articles, a continué avec une phase de tests et de résolution de bugs pour se terminer sur le déploiement des deux logiciels.

Development of a mobile app and a customized web dashboard

This document is Pablo Guinard's DUT-end-internship report. This internship has taken place in CPA (Centre Psychothérapique de l'Ain) between the 19th of April and the 24th of June 2022. It was about take a project in progress and end it.

This project was split in two software. In the first hand, a mobile app developed with React Native (a JavaScript framework) with the goal of displaying a blog which articles were sorted in different categories. The app had different feature like sending comments, a custom dictionary or user stats. In the other hand, a web dashboard, developed with PHP / MySQL, allowing to update dynamically the app's content (articles, dictionary, view comments…). The dashboard included an API which allowed the app to communicate with the dashboard.

The internship has mainly focused on developing the API, improving the frontend of articles in the blog. It has continued with a testing phase and bug tracking and has ended by publishing both software.

Mots-clés du stage

React Native – PHP – MySQL – JavaScript – Application mobile - web