ANALYSE DES ÉVOLUTIONS DE CONCEPTION
EQUIPE 1

Contexte

Suite à une étude de santé publique sur les étudiants en cursus informatique, le ministère de la santé nous a demandé de développer une solution permettant la réalisation d'un parcours virtuel via la pratique d'une activité sportive en réel. Elle permet de créer des parcours virtuels via un site web et de se déplacer via une application mobile.

Nous avons réalisé cette application en suivant la méthodologie agile, plus précisément la méthode scrum. Dans cette optique, le projet a été découpé en 4 sprints ce qui nous a permis à chaque itération d'affiner nos développement et nos choix de conception.

Nous avons par ailleurs découpé notre projet en trois parties distinctes :

- L'API qui gère l'accès et les requêtes à la base de données
- Le site web qui permet la gestion des challenges par l'administrateur
- L'application mobile qui est principalement utilisé par le joueur pour se déplacer sur un parcours

Dans le présent document, nous allons vous présenter les choix évolutions qui ont eu lieu entre le début de notre projet et la dernière itération sur ces différentes parties.

Base de données & API

Tout d'abord intéressons-nous à la base de données. Même si globalement elle a peu changé, les modifications apportées ont souvent été le résultat de changements importants dans les choix de conception.

Nous avons ajouté de nouveaux champs afin de mettre en place des nouvelles fonctionnalités ou d'optimiser les interfaces client:

- Ajout de la start_date dans la table des challenges pour pouvoir gérer des challenges à durée limitée
- Ajout de nouveau champs dans dans la table des évènements afin de pouvoir gérer les réponses au obstacle de type question et téléchargement de photo
- Ajout de données pour pouvoir calculer plus facilement les statistiques de l'utilisateur.
- Ajout du photo_url dans la table utilisateur afin de permettre la personnalisation de la page profil
- Création de table event_type afin de pouvoir envoyer aux clients, un référentiel des événements sur les challenges qui soit unique.
- Remplacement des champs position_x et position_y par le champ progress dans la table des obstacle afin de gérer plus facilement l'affichage de l'obstacle sur un segments.

Nous avons modifié la base de données pour supprimer des informations redondantes ou inutiles:

- Suppression de l'identifiant du challenge dans la table des events, car cette information pouvait être aisément récupéré par le biais du segment rattaché à l'événement.
- Fusion des tables obstacle et questions car aucune des fonctionnalités ou des choix faits dans l'application ne nécessitent qu'elles soient distinctes.

Nous avons aussi supprimé des champs créé à l'origine pour des fonctionnalités que nous n'avons pas eu le temps de mettre en place, comme par exemple :

- Le champ alone_only du challenge qui devait servir pour gérer la possibilité de jouer en équipe.
- Le champ nb_point sur les obstacles que nous voulions utiliser pour le système de classement
- Le champ mandatory_move des segments qui aurait permis de rajouter un difficulté supplémentaire

sur certains segments en spécifiant un mode de déplacement obligatoire

D'autres modifications ont été faites pour répondre à des demandes des Products Owner que nous n'avions pas implémenté lors de la conception initiale, comme par exemple l'ajout de la longueur de pas dans les données d'un challenge que nous avions omis.

Du côté des routes de l'API aussi, nous avons aussi fait des modifications. Il y plusieurs raison à cela :

- Tout d'abord la mise en place de nouvelles fonctionnalités :
    - Ajout de de la partie authentification : Création et connexion des utilisateurs, génération d'un token de connexion
    - Ajout de nouvelles fonctions sur les challenges : validation d'un parcours, publication d'un challenge, …
    - Ajout de fonctions spécifiques pour un média, tel que l'envoi d'une image en base64
    - Gestion des évènements sur un parcours
- Nous avons modifié des routes existantes afin notamment de mettre en place des routes atomiques pour les entités d'un challenge au lieu d'envoyer toutes les données via une seule requête.
- Et pour finir, nous avons refactorisé certaines routes afin de rendre les url plus lisible et plus logique pour les utilisateurs, par exemple dans la route de suppression d'un crossing point, nous avons supprimé le challenge.

Bien entendu, toutes ces modifications et ces ajustements ont eu un impact sur le code de la l'API : création de nouveau traitement, changement de requêtes pour les modifications de la base de données et adaptation de l'existant pour répondre aux nouvelles url. En ce qui concerne l'organisation des répertoires et des fichiers de l'API, nous avons fait peu de changements conséquents par rapport au sprint 2, juste l'ajout de nouveau répertoire pour améliorer la lisibilité et l'organisation des fichiers.

Pour résumer, 3 raisons principale ont motivés les changements que nous avons faits sur la base de données et l'API :

- La mise en place de nouvelles fonctionnalités, et aussi l'évolution de celles déjà faites qui été la suite logique de nos développements.
- L'amélioration et la lisibilité du code existant au fur et à mesure que nous progressions dans l'apprentissage du langage choisi, mais aussi suite à l'analyse du code par l'outil SonarQube.
- La suppression de champs et de codes qui avaient été mis en place pour de futures fonctionnalités que nous avons dû abandonner car nous avions été trop ambitieux dans notre conception. Par ailleurs, nous avions laissé l'aspect graphique du site web à la toute fait du projet et avons donc dû consacrer beaucoup de temps à cette partie sur le dernier sprint.

WEB

Du fait que j'ai travaillé seul sur le web pendant tout le projet hormis la dernière semaine, il n'y a pas eu de grands changements d'architecture du projet ou de choix technologiques.

Pour les librairies utilisées, elles ont toutes été choisies au début du projet et n'ont pas changé, hormis la librairie permettant de créer le parcours du challenge sur la carte.

Pour ce cas précis, je me suis tout d'abord concentré sur la possibilité de tracer des segments sur une image. J'ai donc commencé par utiliser l'api canvas "de base" de javascript et cela fonctionnait. Mais suite au premier sprint, j'ai mieux compris les fonctionnalités demandés pour le parcours du challenge et canvas n'était plus adapté (canvas ne permet pas de modifier des segments tracés ou de détecter des clics sur des éléments précis). J'ai donc choisi d'utiliser la librairie konvajs et sa version react car elle me fournit ce qu'il me fallait : des formes toutes prête comme des points pour les points de passage ou des flèches pour les segments, la

possibilité de déplacer ces formes et la gestion des évènements sur les formes comme “onClick”, “onDragEnd” ou “onChange”. Enfin le render de la carte est simple et facile à comprendre avec react-konva. Il ne me fallait plus que gérer les données comme modifier les points ou les segments et dialoguer avec l’API pour les sauvegarder à chaque action.

L’autre librairie majeure du web est Material-ui pour le design. Il n’y a pas eu de changement de librairie pour le design. Cependant durant les 3 premiers sprint j’ai préféré me concentrer sur les fonctionnalités pour être sûr de pouvoir jouer au jeu, de tester que les routes de l’API fonctionnent bien et pour fournir des données à l’application mobile, le design concernant uniquement le web. Cela à permis d’avoir un client web fonctionnel mais il manquait du temps pour faire le design sur toutes les pages du site. Je pense aussi que c’est plus facile d’ajouter le design une fois que les fonctionnalités sont présentes dans le code que l’inverse.

Ensuite du point de vue de l’architecture du projet, il y a eu peu de changements. Les requêtes à l’API dans le dossier api, les composants avec les fonctionnalités d’administration dans le dossier admin, … Cette organisation était suffisante comme je travaillais seul et tant qu’il y a peu de fichiers dans les dossiers. Mais à la fin du projet je me suis rendu compte que mon organisation des fichiers a ses limites quand le nombre de fichiers augmente et que plusieurs personnes travaillent sur le projet. Si le projet devenait plus gros, il aurait fallu définir une architecture des fichiers plus organisée et efficace.

Aussi, je pense que le projet m’a appris à penser aux fonctionnalités en cours et de ne pas trop prévoir les fonctionnalités suivantes en avance. Pendant le développement on prévoit des fonctionnalités mais le manque de temps nous oblige à les abandonner. Par exemple, il était prévu d’afficher la position de l’utilisateur sur la carte, alors que pour un admin on pouvait imaginer d’afficher tous les utilisateurs et d’autres informations. Au final l’affichage de la carte à l’utilisateur ou à l’administrateur est presque la même et j’en ai fait un composant unique réutilisable (alors que c’était initialement 2 fichiers différents presque identiques). Ceci n’a pas été le cas sur l’affichage d’une page challenge à l’utilisateur, on pensait faire des pages un peu différentes selon si l’utilisateur n’est pas inscrit, est inscrit ou a fini le challenge (3 fichiers différents). Au final on aurait pu faire un fichier unique pour cette page sachant que les différences se résument à afficher ou non l’historique et les stats de l’utilisateur selon son statut sur le challenge.

MOBILE

L’architecture du projet mobile à radicalement changé entre le sprint n°3 et le sprint n°4. En effet, la complexité du projet évoluant, nous nous étions retrouvés avec trop de fichiers mal rangés, et nous avions du mal à retrouver notre code.

La principale évolution de ce grand changement fut d’ajouter le store redux. En effet, grâce au store, nous n’avions plus la nécessité d’organiser nos composants de manière hiérarchique, mais nous pouvions les organiser de manière plus linéaires. Les composants appelant les autres, sans se passer de variables. Tout était contenu dans le store.

Au début du projet, nous avions décidé d’utiliser 3 composants de navigations différents : deux Navigations Stack, un menu hamburger et un menu inférieur. A l’usage, c’était une mauvaise idée. Un seul Navigation stack aurait suffit, et cela nous aurait évité d’avoir trop de composants qui s’imbriquent les uns dans les autres, et ainsi de complexifier la lecture de notre code. Le menu inférieur faisait doublon avec notre menu hamburger et nous l’avons assez rapidement supprimé. Pour finir, le menu hamburger qui était censé contenir le menu de déconnection, certaines statistiques globales et la navigation vers des fonctionnalités secondaires à été supprimé, car nous n’avions pas eu le temps de coder ces fonctionnalités. Nous aurions dû commencer par ajouter un seul composant de navigation, puis en ajouter d’autres au fil du développement si cela était nécessaire.

ANALYSE DES ÉVOLUTIONS DE CONCEPTION
EQUIPE 1

En conclusion

Si nous prenons la globalité de notre projet entre le sprint 2 et le sprint 4, nous pouvons voir que :

Le peu d'expérience que nous avions sur les langages utilisés, nous ont parfois amené à des défaut de conception et lorsque nous les avons détecter à remanier parfois de façon drastique les codes que nous avions mise en place.

Le temps dévolu à l'apprentissage a affecté le temps consacré au développement, ce n'était certes pas une perte de temps et cela a été utile pour bien penser nos codes, mais il n'en reste pas moins que cela a eu un impact.

Au début de notre conception, nous avons été trop ambitieux et penser trop en amont certaines fonctionnalités que nous avons dû abandonner par la suite. Il aurait été plus judicieux de segmenter plus notre logique en petit bloc simple et de les complexifier au fur et à mesure.

Nous avons accordé beaucoup de temps à la logique métier, au détriment du design le reléguant en fin de partie. Cela aussi a affecté nos choix sur le dernier sprint car nous avons dû beaucoup nous concentrer sur le graphisme et donc reléguer au second plan certaines fonctionnalités. Mais il est aussi vrai qu'il est plus facile de poser un design sur une logique bien pensé que de poser une logique métier sur un design. Néanmoins, nous aurions pu commencer un peu plus tôt le développement de la partie design, peut être en répartissant un peu différents les tâches de chacun lors du 3éme sprint.

Néanmoins tous ces choix, ces "mauvaises directions" nous ont été profitables et nous ont aidés à grandir dans notre prise en mains des langages choisis et dans la création d'une application avec API, voilà des erreurs que nous ne referons plus. C'est en forgeant qu'on devient forgeron certes, mais dans le développement c'est en tâtonnant et en se trompant qu'on affine ses compétences .