Joutes CPNV

Auteurs

Kevin Jordil

Benjamin Delacombaz

Table des matières

Introduction

Lors de l'attribution des projets pour MAW 2.1, nous avons reçu le projet Joutes avec comme chef de projet Monsieur Chevillat. Cependant, Jérôme Chevillat n'était pas le chef de projet de Joutes pour les autres années.

Nous avons commencé par lire la documentation des anciens élèves ayant travaillés sur le projet, elle explique bien les points exécutés pendant le projet et comment il fonctionne. Cependant il manquait une partie assez importante, l'état du projet, car aucun outil de gestion de projet n'était mentionné. Certes, après quelques configurations standard nous avons réussi lancer le projet et constaté de nous-mêmes l'état.

Application

L'application joutes est une application de visualisation. Elle permet de visionner les différentes équipes inscrites aux tournois et leurs résultats. Il est également possible d'afficher tous les participants. Cette application est également utilisable en étant hors ligne. Par défaut, l'application utilise les informations se trouvant sur le site des joutes du CPNV.

Difficultés rencontrées

Connexion

Une fonctionnalité de connexion aux joutes était en train d'être développée. Or pour se connecter aux joutes, il faut utiliser SAML.

SAML est une technologie qui permet de centraliser l'authentification sur un service au lieu d'avoir une authentification différente sur chaque application.

Dans notre cas, nous ne pouvons donc pas créer notre propre système de login, il a donc fallu utiliser un navigateur intégré à l'application pour ouvrir la page de login SAML de l'intranet. Or à ce moment-là une erreur est survenue. Il est impossible d'ouvrir la page d'authentification en mode iframe.

Nous cherchons donc à trouver un autre moyen de s'identifier, par exemple en utilisant le navigateur de l'appareil et non celui intégré dans notre application. Mais là, nous sommes identifiés dans le navigateur et non dans l'application Joutes.

La seule solution trouvée pour s'authentifier fut d'enlever le header « X-Frame-Options :SAMEORIGIN » dans les fichiers de SimpleSamlPHP de l'intranet.

Sans ce paramètre, il est désormais possible d'ouvrir la page d'authentification directement avec le navigateur intégré de l'application.

Détection de domaine

Pour la partie connexion de l'utilisateur, au site des joutes, il a fallu ouvrir un navigateur « in App », il suffit de rajouter le composant et celui-ci fonctionne.

Cependant pour rendre l'expérience utilisateur meilleure, nous avons pensé qu'il fallait fermer le navigateur une fois que l'utilisateur serait identifié. La meilleure solution était de détecter le domaine de l'authentification, car celle-ci se passe sur intranet.cpnv.ch à cause de SAML et seulement lorsqu'elle est valide, l'utilisateur est redirigé vers joutes.mycpnv.ch.

Après quelques recherches, nous voyons qu'il existe un évènement qui se déclenche lorsque chaque page a fini de charger. Cette fonction peut convenir à nos besoins.

Nous avons donc mis en place ce système d'évènement, mais après plusieurs essais, le simulateur web ne rentre jamais dans la fonction. Plusieurs heures de recherches s'en suivent sans résultat.

Lors d'un autre essai sans trop d'espoir, on se dit que ça ne fonctionne peut-être uniquement avec un appareil réel. Après un test, c'était effectivement ça, cet évènement ne fonctionne pas sur la page web, mais uniquement avec de vrais appareils.

Asynchrone

L'asynchronisme du JavaScript est très puissant, mais peut également poser certains problèmes. En effet, nous avons été confrontés à un souci lorsque nous avons voulu ''lié'' notre gestion des points d'accès avec les providers / services que les groupes précédents avaient mis en place. Le système de récupération des données essayait de récupérer les données avant que notre gestion des points d'accès ait fini de charger, le service n'avait donc pas de point d'accès pour récupérer les données. Nous devions donc faire attendre le constructeur du service principal que notre provider de points d'accès soit prêt. Dommage, il n'est pas possible de faire de l'asynchrone dans les constructeurs (ce qui est normal). Nous avons donc créé une fonction qui nous renseigne si notre provider est complètement chargé ou pas encore. Nous avons également découvert la fonction ''ngOnInit'' fourni par Angular. Cette fonction est utilisable en asynchrone et s'exécute automatique après le constructeur, nous avons donc déplacer le code se trouvant dans le constructeur dans cette fonction. Cette manipulation nous a permis d'attendre que notre provider de points d'accès soit complètement chargé.

```
import { OnInit } from '@angular/core';
// ...
async ngOnInit() {
    await this.endpointProvider.isReady()
}
```

Améliorations apportées

Auto-open

Auparavant lors du lancement de l'application, la page des évènements s'affichait et nous devions à chaque fois sélectionner l'évènement que nous voulions consulter. Le problème est tel que s'il n'y avait qu'un seul évènement nous devions toujours cliquer dessus. C'est pourquoi nous avons fait en sorte que s'il n'y a qu'un seul évènement, il sera automatiquement ouvert.

Gestion d'erreurs

Le message d'erreur récupéré lors de l'échec d'une requête HTTP n'était pas assez explicite.

Nous avons donc décidé de modifier ce message d'erreur par le suivant qui, quant à lui, est bien plus explicite :

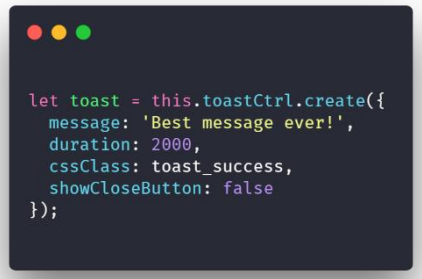
Nous nous sommes dit que si un cas similaire se produisait, il serait utile d'avoir un outil permettant de gérer les messages d'erreur. C'est pourquoi nous avons créé le provider ''ErrorCustomProvider'' cette classe nous permet d'utiliser la fonction statique ''getBetterMessage'' qui, selon le nom de l'erreur, renvoie le message que l'on a défini. Si par hasard le message d'erreur que vous voulez modifier n'est pas traité, le message d'erreur de base sera affiché.

Voici un cas concret d'utilisation de cet outil :

```
catch (error) {
  // Display error in a toast
  this.toastCustom.showToast(ErrorCustomProvider.getBetterMessage(error),10000,this.toastCustom.TYPE_ERR
OR,true)
}
```

Gestion des pains grillés (toast)

La création d'un toast est assez simple, il suffit de déclarer une variable en utilisant le ''toast controller''.



```
let toast = this.toastCtrl.create({
  message: 'Best message ever!',
  duration: 2000,
  cssClass: toast_success,
  showCloseButton: false
});
```

Le problème survient lorsque nous voulons changer les informations de ce toast. Pour pouvoir modifier ce toast nous devrions accéder à chaque attributs et les modifier.

C'est pourquoi nous avons créé le composant ''ToastCustom'' qui nous permet de faire abstraction de la déclaration du toast dans les pages. Il nous suffit d'utiliser la méthode ''showToast'' de notre composant.

```
this.toastCustom.showToast('Best message ever',10000,this.toastCustom.TYPE_ERROR,true)
```

Gestion des routes

L'application a été pensée pour être utilisées par plusieurs écoles, dans cette optique-là, nous avons pensé qu'il faudrait des critères d'API pour être sûr que l'application soit compatible avec l'API. Cependant, après réflexion, nous nous sommes dit qu'il serait plus judicieux de faire un tableau JSON avec le nom des routes et l'école en question renvoie ses routes. Exemple :

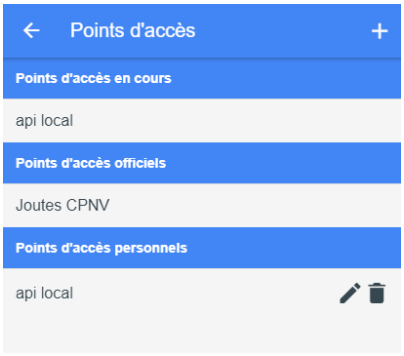

```json
{
    "name": "Joutes CPNV",
    "version": "2.2.0",
    "api_routes": {
        "events.index": "\/api\/events",
        "events.show": "\/api\/events\/{event}",
        "events.tournaments.index": "\/api\/events\/{event}\/tournaments",
        "events.tournaments.show": "\/api\/events\/{event}\/tournaments\/{tournament}",
        "events.tournaments.pools.index": "\/api\/events\/{event}\/tournaments\/{tournament}\/pools",
        "events.teams.index": "\/api\/events\/{event}\/teams",
        "events.teams.show": "\/api\/events\/{event}\/teams\/{team}",
        "events.participants.index": "\/api\/events\/{event}\/participants",
        "events.participants.show": "\/api\/events\/{event}\/participants\/{participant}",
        "participants.notifications.index": "\/api\/participants\/{participant}\/notifications",
        "notifications.update": "\/api\/notifications\/{notification}",
        "tournaments.schedule.index": "\/api\/tournaments\/{tournament}\/schedule",
        "profil.index": "\/api\/profil",
        "login.index": "\/api\/login",
        "login.store": "\/api\/login",
        "login.show": "\/api\/login\/{login}",
        "login.update": "\/api\/login\/{login}",
        "login.destroy": "\/api\/login\/{login}",
        "index": "\/api"
    }
}
```


Ensuite il a fallu créer un service qui permet de renvoyer l'URL de l'application en fonction de la route de l'application courante.

Lors de la méthode de demande d'URL sur le service, on doit fournit en paramètre les différents identifiants ({event}, {notification}) pour avoir une URL utilisable.

Gestion des points d'accès

Lorsque nous avons repris le projet, la gestion des points d'accès se faisait ''en dur'' dans le code. Si nous avions besoin de nous connecter à un autre point d'accès, l'application devait être redéployée. Nous avons donc développé la gestion des points d'accès qui permet d'ajouter des points accès personnels ce qui est assez pratique pour le développement et pour gérer différentes versions d'API.



Cette liste est stockée dans le stockage local du navigateur. Le point d'accès officiel ''Joutes CPNV'' est défini ''en dur'' dans le fichier des variables globales (/src/app/app.const.ts) et est ajouté et sélectionné par défaut au premier lancement de l'application. Pour ajouter ou changer de point d'accès, celui-ci doit être disponible et compatible. Pour qu'un point d'accès soit compatible la page de base de l'api doit retourner un numéro de version qui doit être identique à celle qui se trouve dans le fichier des variables globales et également correspondre à la structure que l'on retrouve également dans ce fichier.

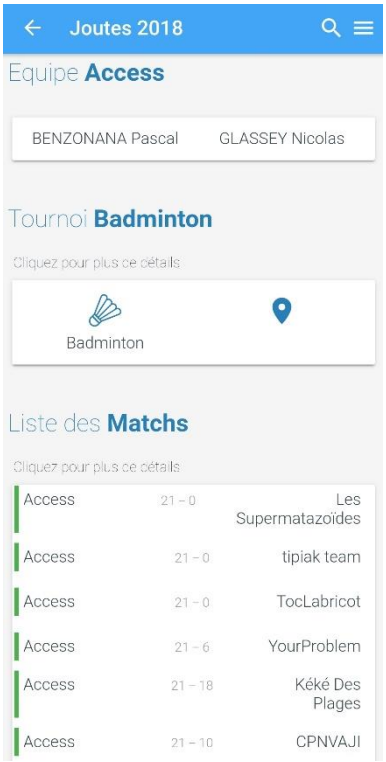
```
export const GLOBAL = {
    apiVersion : '2.2.0',
    apiRequirements: ['name','version','api_routes'],
    apiDefault: new Endpoint('Joutes CPNV', 'http://joutes.mycpnv.ch/api', Endpoint.TYPE_OFFICIAL),
}
```

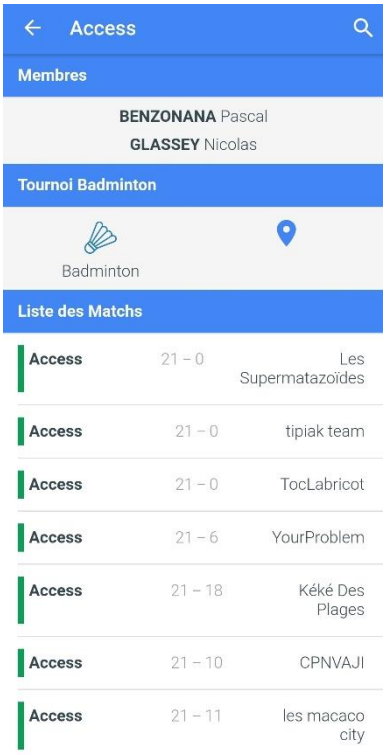
Améliorations de l'expérience utilisateur

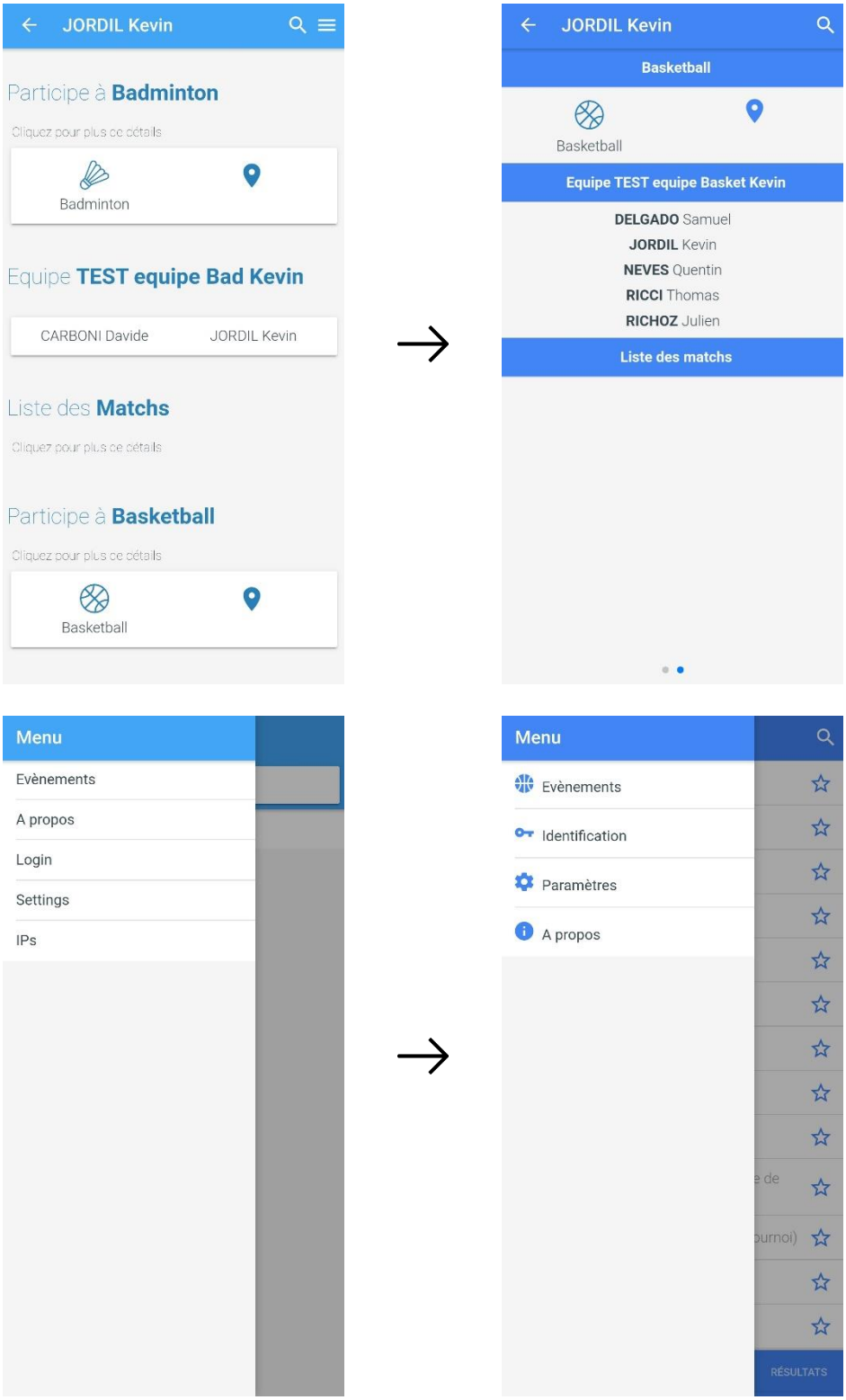
Lorsque nous avons reçu le projet, nous étions un peu perdus dans l'application, c'est pourquoi nous avons décidé d'améliorer l'expérience utilisateur en simplifiant et en rendant plus compréhensibles les diverses informations. Les différences sont visibles sur ces captures d'écrans :

Avant

Après







Les informations affichées sont toujours les mêmes excepté pour le menu. Ces améliorations ont également permis d'uniformiser l'application et donc d'avoir les mêmes couleurs et styles partout.

Refactor des services / providers

Après avoir bien analysé les différents providers / services des anciens groupes ayant travaillé sur le projet, nous avons remarqué qu'il serait bien de revoir tous ces services. Nous avons donc commencé par le service des évènements. Nous n'avons malheureusement pas eu le temps de continuer. Pour permettre le déploiement de l'application, nous avons été contraints de faire cohabiter les nouveaux providers avec les anciens.

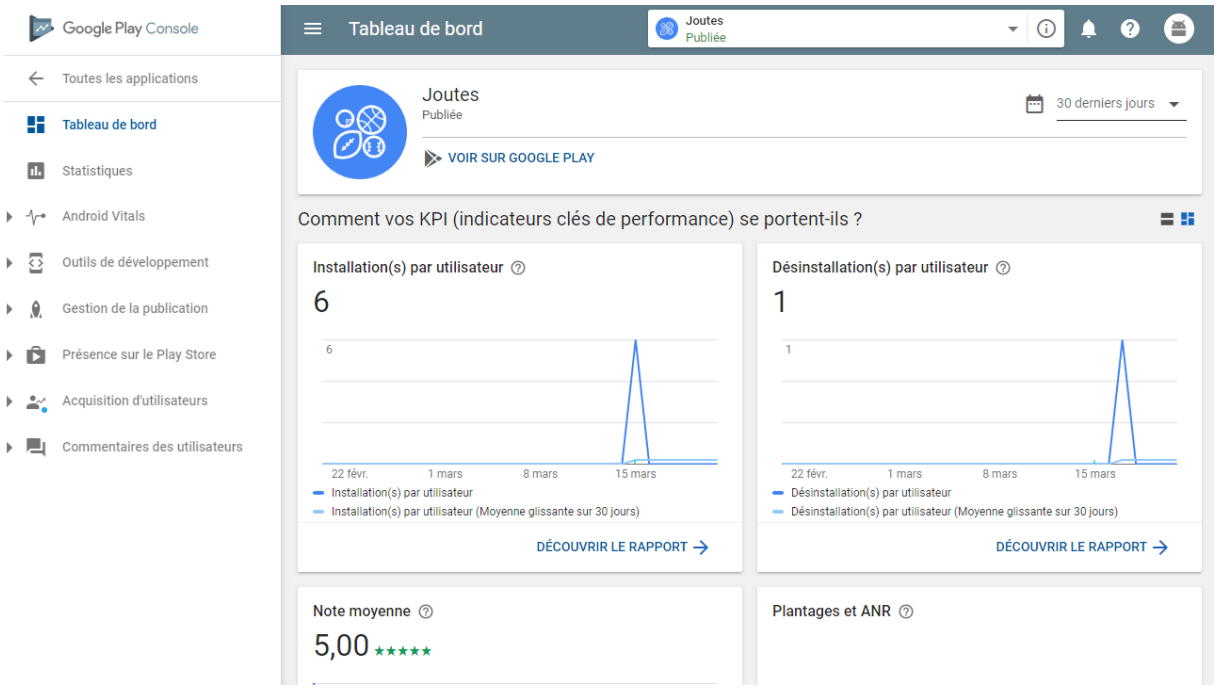
Déploiement de l'application

Google Play Store

Avec un compte Google, nous pouvons nous inscrire en tant que développeurs Google, pour cela, il faut régler un montant de 25 dollars pour finaliser l'inscription.

Nous avons directement accès à la console de déploiement d'application. Par la suite, nous pouvons ajouter le fichier APK signé et remplir des questionnaires pour définir ce que l'application contenait.

Une fois les contrôles passé l'application est publiée sur le Play Store dans les 15 minutes qui suivent. Sur le dashboard, nous pouvons désormais voir l'état de notre application ainsi que différentes statistiques.



Apple Store

Le déploiement sur l'Apple Store fut plus compliqué. Il faut s'inscrire en payant 99$ par années. Ensuite, le compte doit être vérifié avant d'avoir accès à quoi que ce soit. Étant donné que sur Google Play nous étions enregistrés comme ''CPNV École supérieure'', nous avons voulu faire pareil sur Apple, sauf que la validation n'est pas passée, car ce n'est pas un nom de personne valide.

Pour faire valider une entreprise en tant que développeur, il est nécessaire d'envoyer plusieurs documents officiels attestant que nous sommes bien une école. Pour faire plus simple, nous avons décidé, en accord avec de le chef de projet, de mettre le compte développeurs au nom de Monsieur Chevillat.

Après quelques jours d'attente, le compte est validé et nous avons accès au Dashboard développeurs. Nous décidons donc d'essayer de la publier., nous reprenons donc les captures d'écrans réalisées pour Android, mais elles ne sont pas dans un format valide … Il a donc fallu refaire des screenshots avec un appareil Apple. Steven qui se chargeait de faire valider l'application a repris des prises d'écran. C'est lui qui a fait ce travail, car il possédait un Mac.

Une fois l'icône et les écrans de lancement adaptés aux résolutions des appareils Apple, nous avons soumis l'application pour validation. Après quelques jours d'attente, l'application est refusée, car la description est trop courte … Nous ajoutons donc du contenu à la description de l'application Joutes et nous la soumettons à nouveau.

Quelques jours plus tard, l'application est refusée, car l'authentification s'ouvre dans Safari et non dans le navigateur InApp. Effectivement après des tests sur IOS, l'application ouvre safari alors que cela fonctionne sur Android.

Après quelques recherches, nous voyons que le code qui pose problème est celui-ci

```
const browser = this.inAppBrowser.create(url, '_self', options)
```

Le deuxième paramètre indique au navigateur InApp la façon dont il doit s'ouvrir :

- **_self** s'ouvre avec Cordoba WebView si l'URL est dans la liste blanche
- **_blank** ouvre le InAppBrowser
- **_system** ouvre le navigateur du système

Après quelques tests, on voit que si l'on met _blank, cela fonctionne pour IOS, mais plus pour Android. De plus les informations d'authentification ne peuvent plus être reçues sur l'appareil Apple. Ce n'est donc pas la solution.

Sachant que nous sommes dans les derniers jours avant le rendu du projet nous ne pouvons pas trouver une solution qui pourrait nécessité beaucoup de travail.

Après quelques recherches et comme marqué auparavant, self fonctionne uniquement si l'URL est dans la liste blanche alors que sur Android aucune liste blanche n'est actuellement en place. Nous avons donc ajouté une ligne dans le fichier config.xml

```
<!-- Config.xml -->
<navigation href="*" />
```

Après tests, cela fonctionne sur Android et sur IOS, sauf que la détection de domaine ne fonctionne pas sur IOS, le navigateur est donc bloqué sur le site joutes.mycpnv.ch.

A ce moment-là, nous décidons donc d'enlever le menu d'identification sur IOS pour passer la validation. Sachant que celui-ci n'apporte pour le moment aucune fonctionnalité.

A l'heure actuelle (29 mars 2019), l'application est en cours de validation sur le dashboard Apple.

Gestion de projet

Client

Au début du projet, Monsieur Chevillat nous informe que le client est Monsieur Marc Dafflon et qu'il lui a envoyé un mail pour l'informer que nous commençons le projet. Après un certain temps sans réponse de Monsieur Dafflon, notre chef de projet envoya un nouveau mail, mais aucune réponse à nouveau. La décision de Monsieur Chevillat fut de travailler sans et d'essayer de publier une version fonctionnelle sur le store d'ici la fin du projet.

Les améliorations étaient donc soit proposées par le chef de projet soit par nous-mêmes. Ensuite elles étaient validées par le chef de projet.

Choix de l'outil

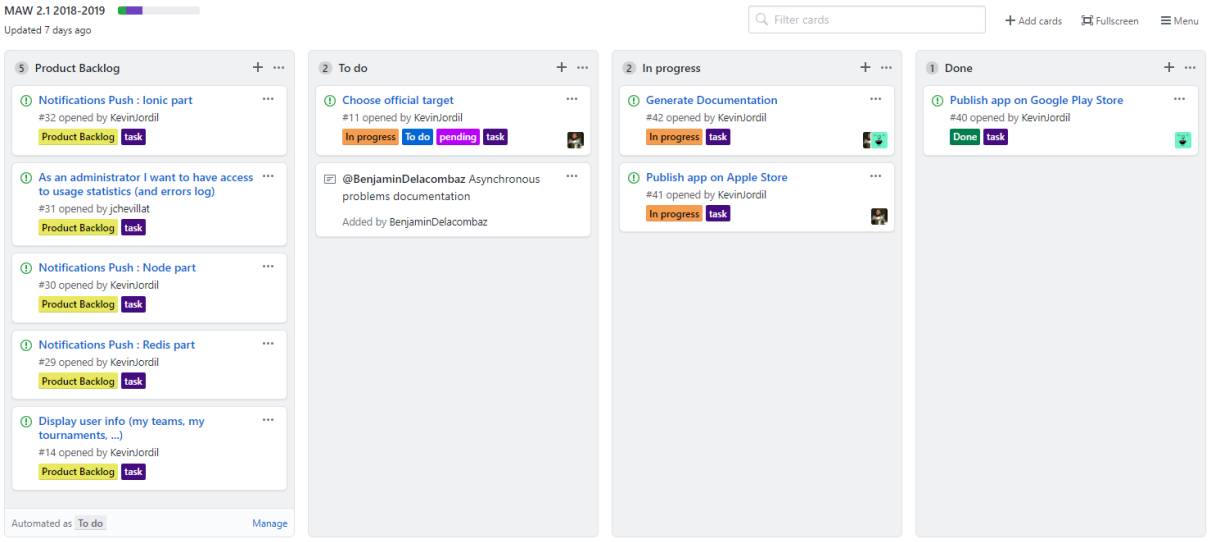
Lors de la réception de notre projet, nous n'avions pas de tableau de bord déjà créé, peut-être, qu'il est inexistant ou alors qu'il est sur Trello. Mais avec un nouveau chef de projet qui n'a pas d'expérience sur ce projet et qui ne savait donc pas où en était le projet. Nous avons donc dû créer notre propre gestion de projet.

Suite aux problèmes rencontrés, nous avons trouvé judicieux de choisir ''Github Project'', car tout est au même endroit, le code source, la gestion de projet et la documentation. De plus les issues créées sont directement intégrées dans la board de projet.

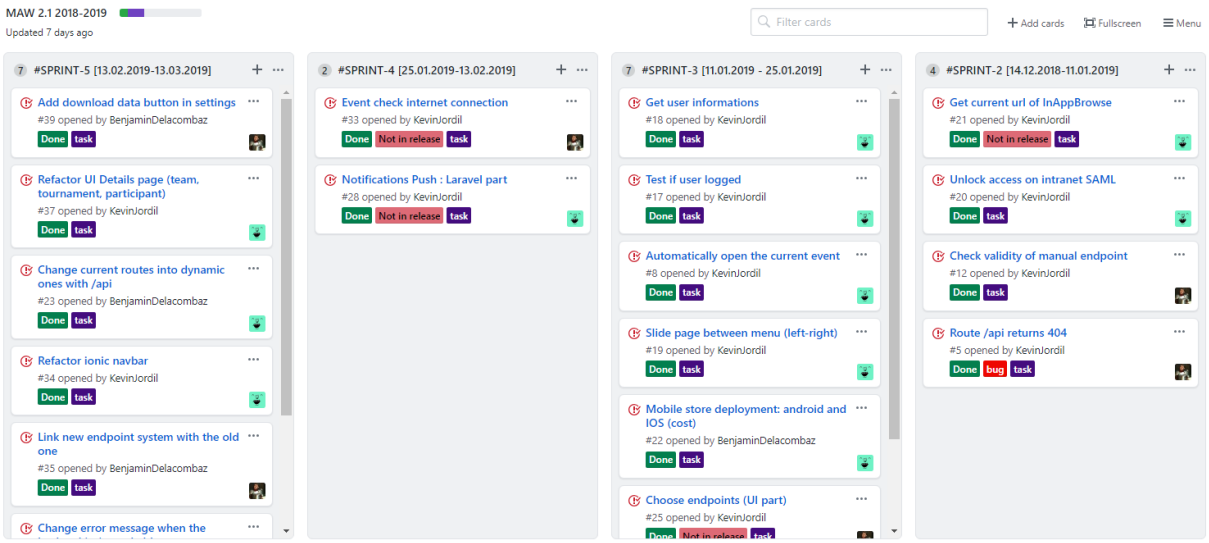
Github Project

En accord avec le chef de projet, nous avons choisi la gestion de projet en mode Kanban simple, mais avec des colonnes spécifiques à chaque sprint avec les tâches liées terminées.

Chaque issue est une tâche qui arrive directement dans la colonne ''Product backlog''.



Lors du début d'un sprint les différentes tâches prévues pour celui-ci sont mises dans la colonne ''To do'' et sont assignées à la personne, si celle-là est déjà définie. En plus de cela un ''milestone'' est créé sur Github pour chaque sprint et donc assigné aux tâches à réaliser pour le sprint. Cela permet aussi d'avoir une vue regroupant uniquement les tâches d'un sprint spécifique.



Documentation technique

Api Requirements

Cette documentation contient les prérequis de l'API pour le bon fonctionnement de l'application. Lien

Mobile Store

Cette documentation contient le prix et une documentation de mise en production sur chaque plateforme : Android et Apple. Lien

Build App

Cette documentation contient la procédure pour générer le fichier apk de Joutes. Lien

Create release

Cette documentation explique les commandes nécessaires à la création d'une release sur Github. Lien

Custom toast

Cette documentation explique le fonctionnement et l'utilisation du composant de notifications personnalisées. Lien

Error Custom

Cette documentation explique le fonctionnement et l'utilisation du gestionnaire d'erreur personnalisé. Lien

Routes Provider

Cette documentation explique le fonctionnement et l'utilisation du service des routes de l'API. Lien

Update dependencies

Cette documentation explique comment mettre à jour les dépendances du projet facilement. Lien

Continuation du projet

Notifications

Une idée de notification a été initiée par les élèves de l'année précédente. Cette fonctionnalité consiste à accéder à une page du site joute et créer une notification pour une équipe pour avertir ses membres dans le cas où l'un d'entre eux serait en retard. Une première partie a été mise en place sur le site joutes, mais il reste encore à définir comment les notifications doivent apparaitre. Car si nous voulons que les notifications s'affichent même si l'application est fermée (comme Whatsapp), nous devons faire ceci avec Google et non juste avec l'application. Cela veut dire que nous devons aussi regarder du côté d'IOS.

C'est donc une grande fonctionnalité qui demande beaucoup de temps et de moyens et qui ne sera que peu utilisée voir jamais. C'est donc une idée de continuation à discuter avec Monsieur Dafflon.

Amélioration des providers

Après une petite analyse des services / providers créés par nos prédécesseurs, nous avons l'impression qu'il est compliqué et lourd pour les mettre à jour si des modifications devaient être effectuées. C'est pourquoi nous avons essayé de simplifier les providers et de les rendre plus facilement modifiables. Nous n'avons malheureusement pas eu le temps de tous les modifier. Actuellement seul le provider des évènements est fonctionnel.

Migration vers Ionic 4

Lorsque nous avons commencé le projet, ionic 4 étaient en beta, nous ne voulions donc pas migrer l'application vers cette nouvelle version. Maintenant que la version 4 de ionic est sortie officiellement, il serait judicieux de la migrer pour pouvoir bénéficier des dernières fonctionnalités.

Correction de la gestion des points d'accès

Lorsque l'utilisateur ajoute, modifie ou change de point d'accès, nous testons sa disponibilité ainsi que sa compatibilité. Le problème à corriger survient au lancement de l'application, car nous ne testons pas si le point d'accès est valide.

Correction de l'évènement du chargement de la page

L'évènement du chargement de la page ne fonctionne pas sur IOS, mais fonctionne très bien sur Android. Il semblerait que le problème provienne du fait que l'évènement de chargement de la page ne se déclenche pas sur IOS. Le navigateur n'est donc pas fermé et reste sur le site des joutes. Nous n'avons malheureusement pas eu le temps de nous pencher dessus, nous avons donc enlevé la possibilité de s'identifier sur IOS.

Améliorations de l'expérience utilisateur

Nous avons développé la partie d'identification avec SAML. Il faudrait maintenant utiliser ses données pour améliorer l'expérience utilisateur en pouvant, par exemple, afficher son équipe, ses tournois, ses résultats …

Conclusion

Ce projet avait pour but de nous familiariser avec le framework mobile Ionic. Nous devions également mettre en pratique une gestion de projet de type Kanban en utilisant ''Github project''. Le projet que nous avons dû reprendre était déjà fonctionnel et nous avons principalement fait de l'amélioration d'UI ou de fonctionnalités. Nous estimons que ce projet a été réussi dans le sens où la plupart des demandes ont été terminées et nous avons su mettre en place une gestion de projet efficace avec une séance toutes les 2 semaines auxquelles nous nous y sommes tenus. Nous avons également confirmé et amélioré nos compétences dans l'utilisation de Ionic 3. Nous avons constaté qu'une bonne gestion de projet est à la fois plaisante à utiliser et d'une puissance redoutable. L'utilisation de Ionic pour développer l'application fut judicieuse, car il s'agit d'un outil très pratique à utiliser et facile à prendre en main. Ce projet a globalement été très instructif et nous avons pu remarquer qu'une bonne cohésion de groupe (chef de projet compris) est plus motivante et permet de travailler bien plus efficacement. Grace à ce projet, cela nous a permis de réaliser comme il est important de bien estimer le temps de développement et de bien connaitre ses technologies. En ce qui concerne les difficultés que nous avons pu rencontrer pendant ce travail, elles sont principalement venues de concept de programmation que nous ne maitrisions pas très bien. Nous espérons réellement que ce projet ne va pas disparaître et qu'il continuera d'être amélioré.