



# Joutes CPNV

**Auteurs**

Kevin Jordil

Benjamin Delacombaz



## Table des matières

Introduction.....	3
Application.....	3
Difficultés rencontrées .....	4
Connexion.....	4
Détection de domaine.....	4
Asynchrone.....	5
Améliorations apportées.....	5
Auto-open.....	5
Gestion d'erreurs.....	6
Gestion des pains grillés (toast) .....	7
Gestion des routes .....	8
Gestion des points d'accès .....	9
Améliorations de l'expérience utilisateur .....	10
Refactor des services / providers .....	12
Déploiement de l'application .....	12
Google Play Store .....	12
Apple Store.....	13
Gestion de projet.....	14
Client.....	14
Choix de l'outil.....	14
Git project.....	15
Documentation technique .....	16
Continuation du projet.....	16
Notifications .....	16
Amélioration des providers.....	16
Migration vers Ionic 4.....	17
Correction de la gestion des points d'accès.....	17
Correction de l'authentification SAML sur IOS.....	17
Améliorations de l'expérience utilisateur .....	17
Conclusion .....	17



## 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 travaillé 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 constater 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 survient. Il est impossible d'ouvrir la page d'authentification en mode iframe.

Nous cherchons donc à trouver un autre moyen de se loguer, par exemple d'utiliser le navigateur de l'appareil et non celui intégré dans notre application. Mais là, nous sommes identifiés dans le navigateur et pas dans l'application Joutes.

La seule solution trouvée pour se loguer 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 logué. 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 devons 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 variable 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 ce qui 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 devons 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 devons 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.

Http failure response for (unknown url): 0 Unknown Error CLOSE

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

Le point d'accès désiré n'est pas disponible. CLOSE

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 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 nous devrions accéder à chaque attribut pour pouvoir 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 peut 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 fonctionne correctement. 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 :

```
{
  "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.

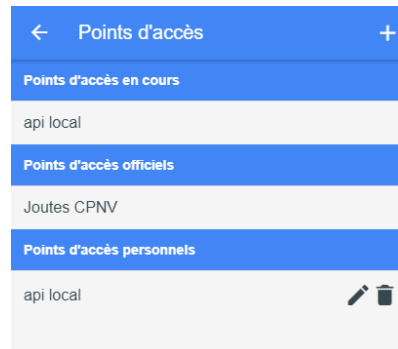
Lors de la méthode de demande d'URL sur le service, on doit fournir en paramètre les différents identifiants ({event}, {notification}) pour avoir une URL utilisable.





## 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 accés 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 :

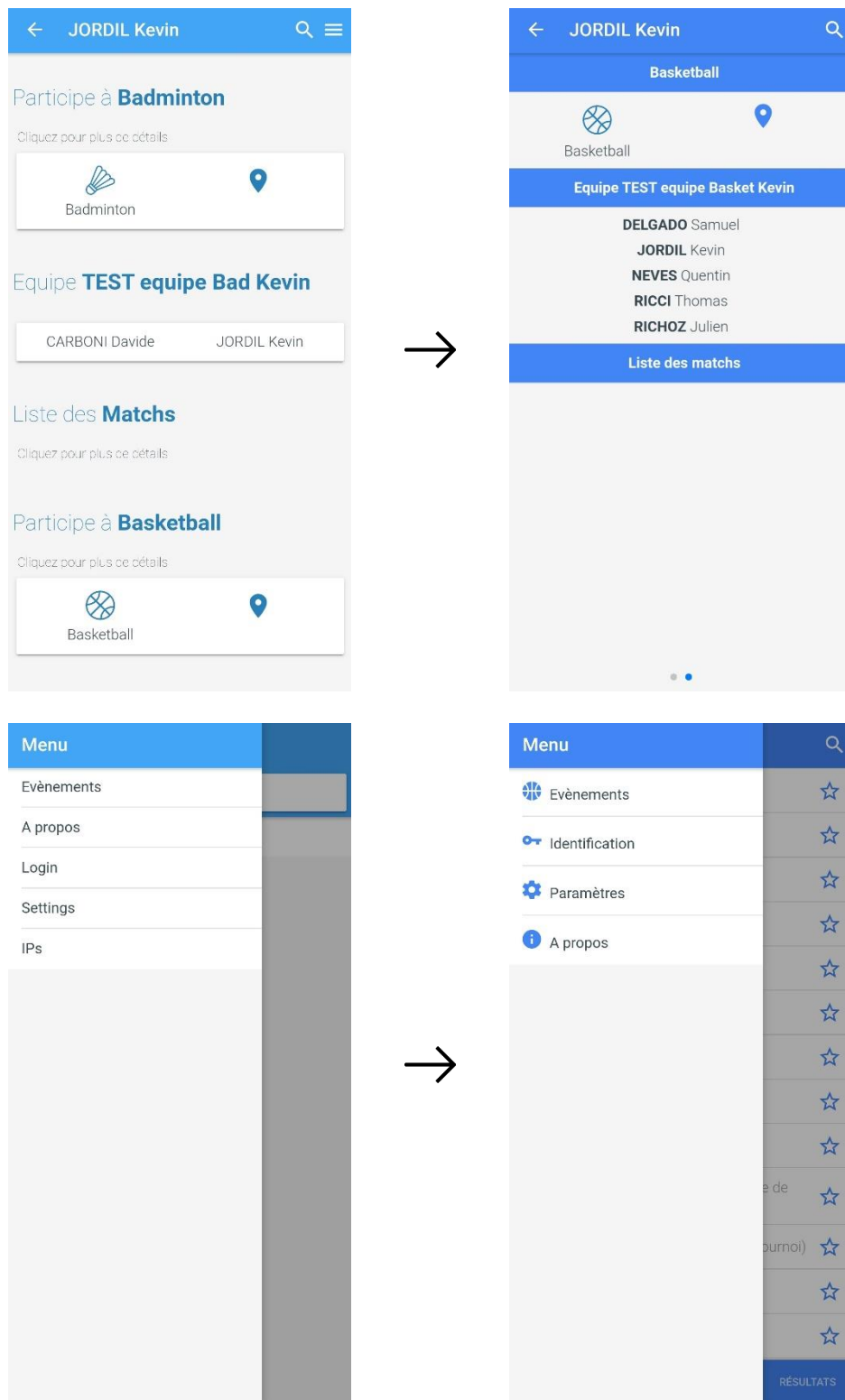
**Avant**

**Après**

→

**Avant**

**Après**



Les informations affichées sont toujours les mêmes excepté pour le menu. Le but second était aussi 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 événements. 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.

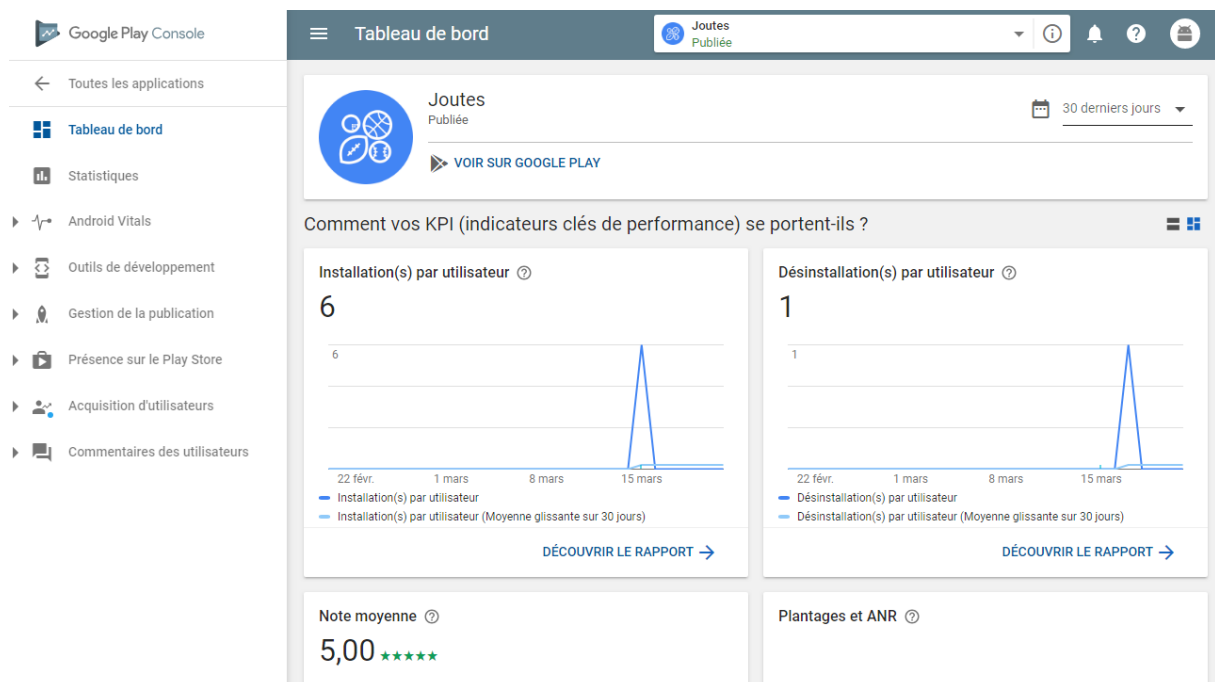
## Déploiement de l'application

### 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 rempli des questionnaires pour définir ce que l'application contenait.

Une fois les contrôles passés l'application est publiée dans les 15 minutes qui suivent sur le Play Store. Sur le dashboard, nous pouvons désormais voir l'état de notre application et différents statistiques sur celle-ci.



## Apple Store

Pour le déploiement sur l'Apple Store, cela fut plus compliqué. Car il faut s'inscrire en payant 99\$ par années. Par la suite 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'a pas passé, car ce n'est pas un nom de personne valide.

Pour faire valider une entreprise en tant que développeur, il faut 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 de nouveau 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 prises d'écrans réalisées pour Android, mais cela n'est pas un format valide ... Il faut 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 faisait 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 à nouveau, 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 la solution.

Sachant que nous sommes dans les derniers jours avant le rendu du projet nous ne pouvons pas trouver une solution qui nécessiterait 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 ajoutons donc ajouter 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](https://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, 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 lui 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 créer, peut-être, qu'il est inexistant ou alors qu'il est sur Trello. Mais avec un nouveau chef de projet qui ne 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 git 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.

## Git 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.

MAW 2.1 2018-2019  
Updated 7 days ago

Filter cards

+ Add cards    Fullscreen    Menu

**5 Product Backlog**

- Notifications Push : Ionic part  
#32 opened by KevinJordil  
Product Backlog task
- As an administrator I want to have access to usage statistics (and errors log)  
#31 opened by jchevillat  
Product Backlog task
- Notifications Push : Node part  
#30 opened by KevinJordil  
Product Backlog task
- Notifications Push : Redis part  
#29 opened by KevinJordil  
Product Backlog task
- Display user info (my teams, my tournaments, ...)  
#14 opened by KevinJordil  
Product Backlog task

Automated as To do    Manage

**2 To do**

- Choose official target  
#11 opened by KevinJordil  
In progress To do pending task
- @BenjaminDelacombaz Asynchronous problems documentation  
Added by BenjaminDelacombaz

**2 In progress**

- Generate Documentation  
#42 opened by KevinJordil  
In progress task
- Publish app on Apple Store  
#41 opened by KevinJordil  
In progress task

**1 Done**

- Publish app on Google Play Store  
#40 opened by KevinJordil  
Done task

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.

MAW 2.1 2018-2019  
Updated 7 days ago

Filter cards

+ Add cards    Fullscreen    Menu

**7 #SPRINT-5 [13.02.2019-13.03.2019]**

- Add download data button in settings  
#39 opened by BenjaminDelacombaz  
Done task
- Refactor UI Details page (team, tournament, participant)  
#37 opened by KevinJordil  
Done task
- Change current routes into dynamic ones with /api  
#23 opened by BenjaminDelacombaz  
Done task
- Refactor ionic navbar  
#34 opened by KevinJordil  
Done task
- Link new endpoint system with the old one  
#35 opened by BenjaminDelacombaz  
Done task
- Change error message when the

**2 #SPRINT-4 [25.01.2019-13.02.2019]**

- Event check internet connection  
#33 opened by KevinJordil  
Done Not in release task
- Notifications Push : Laravel part  
#28 opened by KevinJordil  
Done Not in release task

**7 #SPRINT-3 [11.01.2019 - 25.01.2019]**

- Get user informations  
#18 opened by KevinJordil  
Done task
- Test if user logged  
#17 opened by KevinJordil  
Done task
- Automatically open the current event  
#8 opened by KevinJordil  
Done task
- Slide page between menu (left-right)  
#19 opened by KevinJordil  
Done task
- Mobile store deployment: android and IOS (cost)  
#22 opened by BenjaminDelacombaz  
Done task
- Choose endpoints (UI part)  
#25 opened by KevinJordil  
Done Not in release task

**4 #SPRINT-2 [14.12.2018-11.01.2019]**

- Get current url of InAppBrowse  
#21 opened by KevinJordil  
Done Not in release task
- Unlock access on intranet SAML  
#20 opened by KevinJordil  
Done task
- Check validity of manual endpoint  
#12 opened by KevinJordil  
Done task
- Route /api returns 404  
#5 opened by KevinJordil  
Done bug task

## 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 plus 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 personnalisés. [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 de 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. Le but est de via une [page](#) du site joute, créer une notification pour une équipe pour avertir tous les membres dans le cas où l'un d'eux est en retard. Une première partie a été mise en place sur le site joutes, mais il reste encore à définir comment les notifications doivent apparaître. 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 aussi dire que nous devons aussi regarder du côté de IOS.

C'est donc une grande fonctionnalité qui demande beaucoup de temps et de moyens et qui ne sera que peut 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 la migrer 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'authentification SAML sur IOS

L'authentification SAML 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 page ne se déclenche pas sur IOS. Le navigateur n'est donc pas fermé et reste sur le site des joutes une fois identifié. 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 tenues. 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. Le travail que nous rendons reste assez basique, mais cela nous permet de réaliser comme il est important de bien estimer le temps de développement et de bien connaître 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 maîtrisons pas très bien. Nous espérons réellement que ce projet ne va pas disparaître et qu'il continuera d'être amélioré.