



Rapport PFA 2020

Équipe Quickmatch :

Ali ISSAOUI
Antoine DANQUIGNY
Faiz ABDERRAHMANE
Johan CHATAIGNER
Laurent GENT
Maxime ROSAY
Mehdi HALA
Raphaël MELINE

Clients :

Mathieu FAVERGE
Tony DELARUE

Encadrant :

Yves MÉTIVIER

Remerciements

Avant toute chose nous souhaitons remercier les personnes ayant fait office de client dans le cadre de ce projet au fil de l'année, à savoir M. Mathieu FAVERGE et M. Tony DELARUE de nous avoir éclairés, guidés et conseillés durant toute la durée du projet.

Et nous remercions aussi notre encadrant M. Yves METIVIER pour sa disponibilité et ses rapides retours concernant notre travail.

Table des matières

1 Analyse et spécifications des besoins	3
1.1 Présentation de la mission logicielle	3
1.2 Périmètre et limites du projet	5
1.3 Analyse fonctionnelle	5
1.3.1 Bête à cornes	5
1.3.2 Cas d'utilisation	6
1.4 Besoins non fonctionnels	8
1.4.1 Contraintes sur le produit	8
1.4.2 Application mobile	8
2 Méthodologie générale	8
2.1 Processus de développement	8
2.2 Suivi de versions	9
3 Technologies utilisées	9
3.1 Application Web	9
3.1.1 Web : Vue.js et o2Switch	9
3.1.2 Base de données : PostGRESQL	10
3.2 Application mobile : Android - Kotlin	10
3.2.1 Architecture	10
3.2.2 Data Binding	10
3.2.3 Life Cycles	11
3.2.4 Couroutines	11
3.2.5 Communication avec la base de données	11
4 Implémentation réalisée	12
4.1 Application web	12
4.1.1 Gestion de comptes et authentification	12
4.1.2 Sécurisation de Quickmatch	14
4.1.3 Profil joueur	15
4.1.4 Gestion des clubs	15
4.1.5 Calendrier	17
4.1.6 Invitations	19
4.1.7 Suivi de matchs et statistiques	19
4.1.8 Hébergement et gestion de la base de données	20
4.1.9 test.quickmatch.fr : le site test !	22
4.2 Application mobile	22
5 Capitalisation d'expérience sur les outils collaboratifs utilisés	31
5.1 Synchronisation du code via <i>Github</i>	31
5.2 Suivi des tâches grâce à <i>Trello</i>	31
5.3 Plateforme de communication avec le client : <i>Slack</i>	32
5.4 Hébergement de l'application web et de la base de données via <i>O2switch</i>	32
6 Difficultés observées et fonctionnalités non réalisées	32
6.1 Problèmes rencontrés	32
6.2 Ajouts souhaités	32
7 Conclusion	32

Introduction

Dans le cadre de notre formation d'ingénieur en informatique, ce projet nommé *PFA* (*Projet au fil de l'année*) a pour but de nous initier à la réalisation d'un projet de développement logiciel durant une longue période, ici 6 mois, mais aussi de nous introduire à la notion de relation client, d'une méthodologie plus structurée et un retour sur expérience immédiat.

Contexte de la demande client

Les chercheurs du Centre de Recherche Inria Bordeaux - Sud-Ouest aiment disputer des matchs de football en salle entre eux à des dates et horaires plus ou moins répétitifs. Cependant, organiser ces rencontres peut vite devenir fastidieux et c'est pour cela que les clients ont proposé un projet d'application permettant de superviser la gestion de parties de foot afin de faciliter l'organisation de ce genre d'évènements.

Dans un monde où de plus en plus de services sont connectés, ce système permettrait de faire gagner beaucoup de temps d'organisation à beaucoup de monde. Dans un premier temps destiné aux personnes de l'Inria et de l'Enseirb-Matmeca, l'application permettrait de réduire le temps d'organisation concernant des matchs de foot, et de fait, de favoriser l'organisation d'évènements sportifs. L'application possède une utilité plus grande que celle de seulement contenter les chercheurs de l'Inria dans la mesure où elle pourrait profiter à tout le monde par la suite.

L'idée première étant le service proposé par l'application, la portabilité de cette dernière par la suite est aussi un enjeu du projet. Plus de 2,71 milliards de personnes possèdent un smartphone dans le monde, ainsi rendre ce service accessible n'importe où (sous condition d'avoir un accès à internet) faciliterait son usage au quotidien : l'utilisation d'un ordinateur deviendrait donc optionnelle.

Motivation pour le projet

Notre équipe est composée des huit personnes citées en page de couverture de ce dossier. Chaque membre a choisi le sujet pour accroître ses propres connaissances dans le domaine de développement Web dans un premier temps, mais aussi pour adopter en parallèle un esprit rigoureux pour rendre des livrables optimisés et fiables (en terme de maintenabilité).

De plus, ce qui nous a tous intéressés est d'exporter cette application, dans un premier temps Web, vers une application mobile. En effet, pour la plupart d'entre nous cela nous permettra d'acquérir plus d'expérience dans le développement d'applications mobiles de type Android.

Enfin, nous sommes tous friands de sport et ce projet lie bien travail et plaisir, soit une raison de plus afin de le choisir !

1 Analyse et spécifications des besoins

Lorsqu'un projet est soumis, le client a une liste de fonctionnalités qu'il souhaite et de services auxquels il voudrait avoir accès. Dans cette section, nous allons discuter des services offerts aux utilisateurs par notre application, et comment cette application offre ces services en utilisant notamment des diagrammes UML, permettant de représenter ce qu'il se passe derrière les fonctionnalités souhaitées afin d'être sûr que le client et l'équipe projet soient sur la même longueur d'onde.

1.1 Présentation de la mission logicielle

Dans le document spécifiant la demande du client, plusieurs points étaient clairement énoncés. La mission logicielle est de **développer une application Web et mobile** afin de pouvoir organiser la **gestion de matchs de foot en salle**. A l'image du site *Framadate.org*, cette application permettrait d'organiser des matchs récurrents sur une période donnée d'une semaine.

La mission logicielle est donc de fournir une application permettant de gérer des matchs de foot en salle. Mais le but à long terme est de fournir une application permettant de gérer tous types de sport en équipe : le but est de pouvoir organiser des matchs, gérer des clubs, des statistiques et les proposer aux utilisateurs.

Un utilisateur aura la possibilité de choisir des horaires sur une période spécifique, une semaine par exemple, et d'inviter des coéquipiers à ces horaires. Les joueurs recevant les invitations peuvent par la suite les accepter et si il y a assez de joueurs pour un match, alors ce dernier sera créé.

Des groupes (des clubs) entiers pourront être invités et seront gérés par des utilisateurs. L'administrateur (le gérant du club) est tout simplement un utilisateur de l'application mais il a pour but de créer des groupes de jeu et d'inviter ces derniers sur des plages horaires qu'il aura fixées, afin de jouer.

Après que l'administrateur ait invité tous les joueurs d'un groupe, les joueurs devront accepter cette invitation au match. Une fois qu'un match a été accepté par un nombre minimum de personnes, défini par l'administrateur du groupe, alors le match pourra se jouer.

Il est important de faire en sorte que les utilisateurs soient prévenus des organisations futures de matchs, et donc un système de notifications sur le site doit être développé.

Voici donc les grandes demandes du client qui ont été définies dans sa fiche de projet. Tous les axes de développement seront abordés dans ce document. Le client se chargera de valider ou d'infliger les points sur lesquels il n'est pas d'accord.

Voici une liste non exhaustive des demandes principales du client :

- consulter les créneaux de matchs ou ses invitations reçues
- rechercher des matchs et voir le résultat
- visualiser des statistiques
 - par joueur
 - par match
- inviter des joueurs à des matchs
- créer des groupes

Ces services précédents devront être disponibles à travers une interface Web en priorité, mais aussi une application mobile. Ces deux applications ne présenteront pas de différences majeures si ce n'est l'interface et la charte graphique qui n'est pas imposée.

Il a aussi été demandé à ce que l'application soit disponible en français et en anglais, et potentiellement dans d'autres langages.

Une demande du client est aussi de pouvoir implémenter une connexion Google dans la suite du projet mais nous devons déjà assurer une connexion classique.

Enfin les clients souhaiteraient, si possible, une authentification CAS (Bordeaux-INP) et si ce n'est pas possible, au moins un système d'authentification sécurisé. En effet, sachant que l'authentification CAS ne dépend pas de nous, il aurait fallu héberger notre application sur un serveur de l'Enseirb.

Nous nous retrouvons donc avec différents grands axes de spécifications fonctionnelles à satisfaire comme montré sur la Figure 1 :

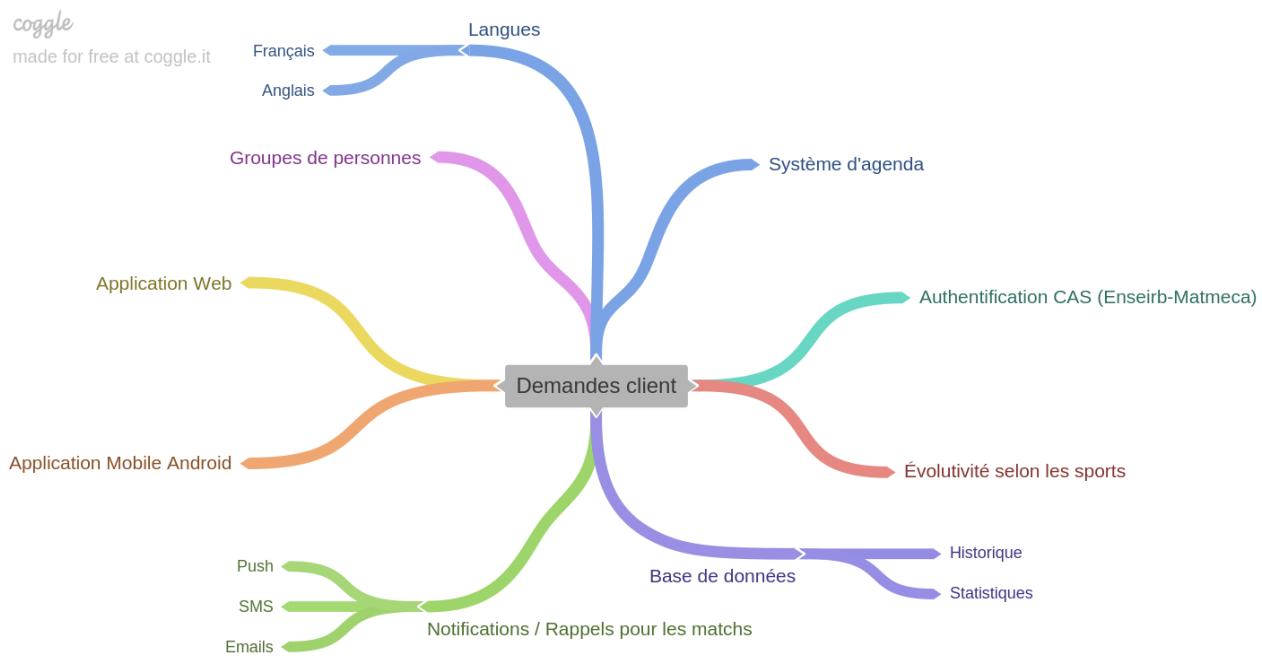


FIGURE 1 – Mind map du projet

1.2 Périmètre et limites du projet

Cette section a pour but de spécifier tout ce qui va être réalisé par les développeurs afin de satisfaire le client.

Ce projet concerne donc la création d'une application Web et mobile permettant à des membres de Bordeaux-INP de pouvoir créer des créneaux de rencontre sportive et faciliter la création de matchs.

Le projet comprend donc plusieurs phases :

- phase de conception
- phase de planification
- phase de réalisation et documentation
- phase de livraison

Cependant, il est important de définir un périmètre. En effet, le périmètre d'un projet concerne tout ce qui doit être fait et seulement ce qui doit être fait. Ainsi, plusieurs tâches n'entrent pas dans le projet et donc il est important de mentionner le fait que les tâches suivantes **ne font pas parties** du périmètre :

- la maintenance après le livrable de fin
- la réservation des terrains après choix final d'une plage horaire
- la création d'équipe : elle se fait directement le jour J

En termes de compatibilité, l'application Web ne prendra pas en charge IE8¹ et les versions antérieures. En effet, plusieurs fonctions fournies par javascript ne sont pas supportées sur ces versions. Cependant, elle fonctionnera sous les navigateurs compatibles avec ECMAScript 5.

1.3 Analyse fonctionnelle

Maintenant que nous avons vu ce que l'application doit fournir comme services aux utilisateurs, nous allons étudier plus en détail les cas d'utilisation de cette application et quelles sont les fonctions de service que l'application doit rendre.

1.3.1 Bête à cornes

Avant d'imposer un "comment" ou une solution il faut se tourner vers l'utilisateur et/ou le demandeur pour aboutir de manière structurée à la solution, un projet n'a de sens que s'il satisfait le client. Il convient donc d'exprimer le besoin et seulement le besoin dès le lancement du projet.

¹Internet Explorer 8

Il s'agit ici d'expliciter l'exigence fondamentale qui justifie la conception, ou la reconception d'un produit. Pour cela, il est essentiel de pouvoir répondre à ces questions :

- A qui le système rend-il service ?
- Sur quoi / qui agit-il ?
- Dans quel but ?

On peut répondre à ces différentes questions de manière synthétique et simple grâce à un diagramme de type APTE² ou plus communément appelé *bête à cornes* permettant de rassembler toutes les informations.

La Figure 2 représente notre diagramme APTE concernant notre solution logicielle :

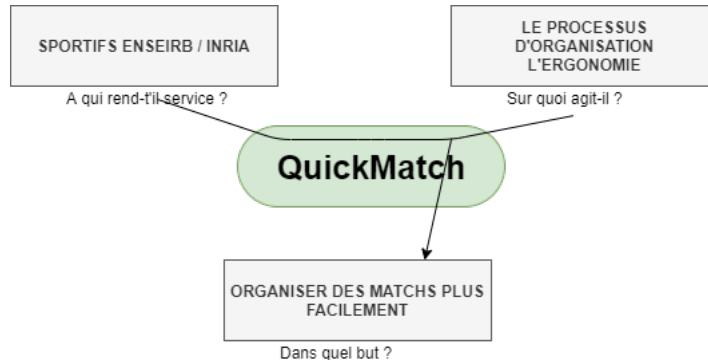


FIGURE 2 – Diagramme APTE

1.3.2 Cas d'utilisation

Afin de bien comprendre comment notre application va s'articuler, et notamment quelles sont les fonctions auxquelles l'utilisateur pourra avoir accès, nous avons réalisé un diagramme de cas d'utilisation³ regroupant toutes les demandes que le client souhaite faire (que cela soit sur son application mobile ou Web).

Sur la Figure 3, sont recensées toutes les actions que les différents acteurs peuvent faire :

²APplication aux Techniques d'Entreprise

³Un cas d'utilisation, ou cas d'usage (« use-case » en anglais), définit en génie logiciel et en ingénierie des systèmes une manière d'utiliser un système qui a une valeur ou une utilité pour les acteurs impliqués. Le cas d'utilisation est une technique pour capturer les exigences d'un système et servir de fil conducteur à l'ensemble des activités nécessaires à sa mise en oeuvre.

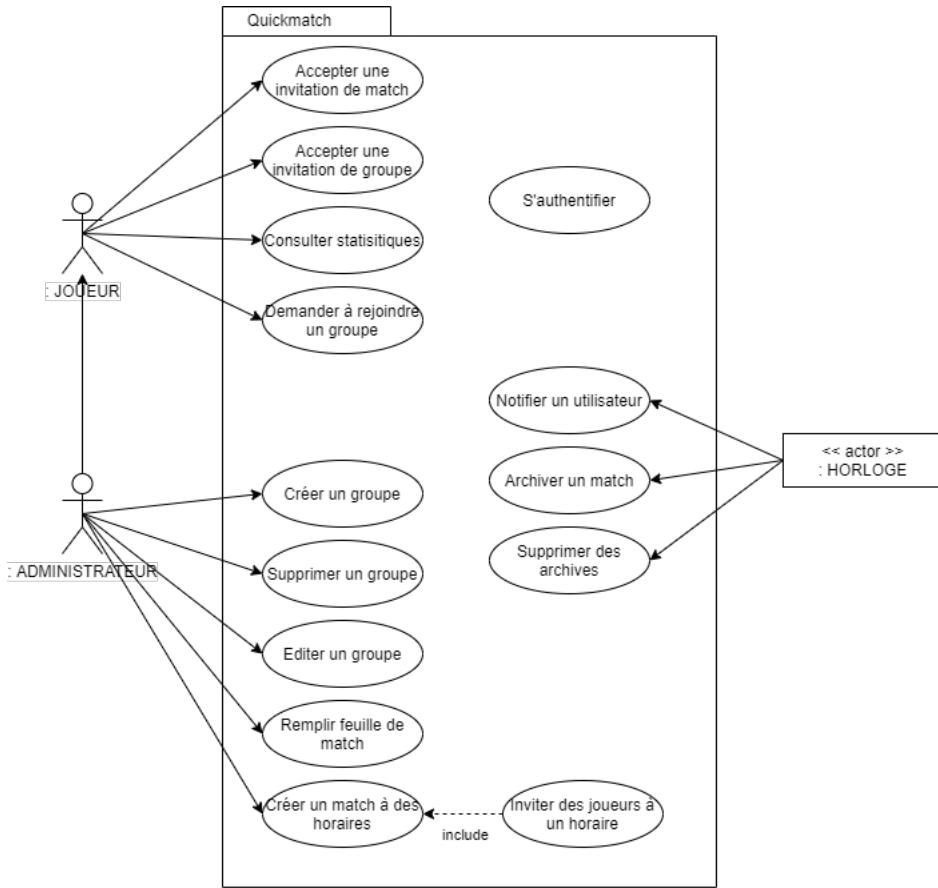


FIGURE 3 – Diagramme de cas d'utilisation de l'application QuickMatch

Il existe plusieurs acteurs humains et non humains :

- le joueur (utilisateur lambda) : acteur humain
- l'administrateur (personne ayant des priviléges sur un groupe) : acteur humain
- l'horloge (système qui va gérer les notifications) : acteur non humain

Ces acteurs peuvent donc effectuer plusieurs actions dans notre système. Nous les avons mentionnés dans le diagramme mais voici la liste exhaustive des cas d'utilisation :

Concernant le joueur lambda :

- consulter les créneaux de matchs : avoir accès à un agenda dans lequel il peut voir les différents créneaux de matchs à la semaine
- supprimer un créneau de match
- confirmer une invitation : confirmer une invitation concernant un match, à une date, à un lieu
- consulter les statistiques concernant un match, un joueur
- s'inscrire dans une équipe publique

L'administrateur, ie. un joueur normal mais administrateur d'un groupe en particulier, a les mêmes actions qu'un joueur (spécifié par la flèche entre les deux acteurs humains) mais en possède également d'autres comme :

- créer un groupe
- supprimer un groupe
- éditer un groupe : rajouter ou supprimer des membres d'un groupe
- remplir feuille de statistiques : il faut bien que quelqu'un remplisse les statistiques pour les joueurs, et nous avons fait le choix de faire en sorte qu'il s'agisse de l'administrateur
- créer un match
- inviter des joueurs à un horaire

À noter que tous les cas d'utilisation précédents *include* le cas d'utilisation de se connecter à l'application via le CAS (ou autre si ce n'est pas possible) à savoir qu'il est nécessaire de s'authentifier avant de pouvoir effectuer une de ses actions.

Il reste un dernier acteur non humain qui est l'horloge qui aura pour but de notifier les utilisateurs :

- notifier un utilisateur : un changement d'horaire, un changement de lieu, un rappel la veille, ...
- archiver un match : après l'horaire du match et l'ajout des informations concernant ce dernier par l'administrateur, alors le match sera archivé pendant un certain temps donné
- supprimer une archive : après un laps de temps fixé par l'administrateur SI

Tous les scénarios énoncés précédemment sont identiques que cela soit sur l'application Web ou l'application mobile.

1.4 Besoins non fonctionnels

Un projet a donc beaucoup de besoins fonctionnels mais possède aussi énormément de besoins dits non fonctionnels à savoir l'usabilité, l'efficacité, ... Dans cette section nous allons les développer et parler plus généralement des contraintes implicites de notre projet et la manière avec laquelle nous allons les gérer.

1.4.1 Contraintes sur le produit

Comme nous l'avons vu précédemment, le produit possède plusieurs contraintes en plus des fonctionnalités qu'il doit remplir. Tout d'abord, le site web doit pouvoir être ergonomique et être le plus facile d'utilisation possible : navigation entre les onglets, les fonctionnalités, avoir accès facilement à l'information... Mais cette contrainte doit aussi être appliquée concernant l'application mobile le tout en ne divergeant pas trop du site web (mêmes onglets, mêmes données aux mêmes endroits).

Le nom de notre produit que nous allons rendre se nomme : *QuickMatch*. Il semble logique d'adopter une application qui permet de **rapidement**, rejoindre un groupe, pour ensuite recevoir des invitations de matchs, les accepter et c'est tout ! Compliquer ce déroulement d'activité reviendrait donc à ne pas remplir un besoin non fonctionnel de l'application.

Un besoin qui semble tout à fait naturel serait de s'assurer que les horaires proposés conviennent bien à tous les membres qui sont invités et non pas seulement à certains du groupe. Le but étant de ne pas spam l'utilisateur de notifications, ces dernières ne concernant que les matchs auxquels il peut participer.

1.4.2 Application mobile

Avec le site internet, nous devons aussi fournir une application mobile sous Android. Bien évidemment, comme nous l'avons précisé auparavant, le premier objectif est de donner un accès seulement aux membres de Bordeaux-INP, mais dans le futur cependant, nous voulions aussi implémenter une interface de connexion pour les autres plateformes telles que Google par exemple.

Le système d'authentification s'il ne s'agit pas du CAS, doit pouvoir assurer la comptabilité entre les comptes créés sur le site et sur l'application (le même système de hashage par exemple).

2 Méthodologie générale

Attardons nous maintenant sur l'aspect méthodologique du projet. Comment nous avons abordé le processus de développement du projet et nous avons séparé les différents modules de développement.

2.1 Processus de développement

Concernant le processus de développement, nous avons choisi d'adopter une méthode agile, la méthode SCRUM⁴. Cette méthode nous permet de rencontrer le client le plus de fois possible afin de lui montrer l'avancée de notre application sous forme de jalons livrables avec pour chaque incrément, des fonctionnalités en plus et un meilleur produit répondant à de plus nombreux besoins.

Nous avons donc convenu durant toute la durée du projet de nous rencontrer **au minimum une fois toutes les deux semaines**, soit la durée d'un sprint, afin de remplir des objectifs que nous avions fixés au début du projet (et adaptés tout au long de notre projet après concertation avec les clients).

Grâce une organisation autour de *Trello*, nous avons listé tous les objectifs et besoins demandés par le client (énoncées précédemment).

Le but ici était donc de faire un vrai premier projet en conditions "réelles" avec un suivi client assez régulier et des demandes pouvant varier suivant les semaines.

A noter que nos rencontres se sont multipliées à la fin de notre projet : nous sommes passés à 1 séance par semaine afin de finaliser nos dernières fonctionnalités principales nécessitant toute notre attention.

⁴Scrum est un schéma d'organisation de développement de produits complexes. Il est défini par ses créateurs comme un « cadre de travail holistique itératif qui se concentre sur les buts communs en livrant de manière productive et créative des produits de la plus grande valeur possible »

2.2 Suivi de versions

Le groupe projet avait fait le choix d'utiliser Github afin de faciliter les phases de développement puisque ce gestionnaire est très complet en proposant :

- un système de gestion de bugs : *issue tracking system*
- des propositions de modification : *pull-request*
- un gestionnaire de tâche
- un wiki
- des graphiques concernant la contribution des membres

3 Technologies utilisées

Revenons ici sur les choix effectués par le groupe concernant le développement de *Quickmatch*. Cette section vise à rappeler quels moyens ont été utilisés pour réaliser au mieux les besoins des clients.

3.1 Application Web

Commençons par l'application web. Elle ne prendra pas en charge IE8⁵ et les versions antérieures mais fonctionnera sous les navigateurs compatibles avec ECMAScript 5. Dans cette section, nous allons détailler le choix du framework utilisé, de notre hébergeur web ainsi que de notre SGBD.

3.1.1 Web : Vue.js et o2Switch

Avant de revenir sur nos choix concernant le framework `Vue.js` ainsi que l'utilisation de `Javascript` en *full-stack*, revenons sur le choix de l'hébergeur `o2Switch`. N'ayant aucune connaissances, ou alors très sommaires, sur les sites web proposant de l'hébergement, nous avions écouté nos clients pour ce choix. Monsieur Faverge ayant de l'expérience dans ce domaine, il nous a recommandé⁶ `o2Switch` qui s'est avéré être un bon service d'hébergement en ligne.

Concernant le site Web, nous avons fait le choix de faire du `Javascript full-stack`, à savoir gérer le *front-end* et le *back-end* en `Javascript` en utilisant un framework spécifique.

Utiliser un framework `Javascript full-stack` nous a permis d'éviter plusieurs problèmes mais aussi de permettre à notre application d'avoir plusieurs avantages. En effet, pour le bien de l'expérience client, nous avons pensé qu'il valait mieux avoir du traitement côté client (sans rafraîchissement de la page) plutôt que de devoir actualiser la page afin d'obtenir un résultat. Ainsi, nous avons réalisé une application de gestion ergonomique et optimisée⁷ en dépit de quelques attentes (nous avions fait le choix de privilégier fluidité à fonctionnalités révolutionnaires).

L'utilisation d'un framework était obligatoire pour nous puisque nous souhaitions avoir une application bien structurée entre modules et gagner du temps concernant le développement. Utiliser un framework nous a permis de nous focaliser sur la partie métier de l'application dans la mesure où la conception et l'élaboration des composants et couches techniques sont fournies par le framework. Cela a parallélisé les tâches et évité les conflits dans la gestion des sources. Le système de gestion de route, de composants et d'états était beaucoup simple à mettre en place en utilisant un framework qu'en le faisant à la main. C'est pourquoi nous avons fait le choix d'en utiliser un.

Concernant le choix du framework, nous avons choisi `Vue.js` bien que nous ayons eu plusieurs choix concernant les frameworks ; puisque la courbe d'apprentissage du framework `Vue.js` s'est révélée clairement plus facile à suivre. De plus `Vue.js` est beaucoup moins verbeux ce qui le rend très efficace et a permis un développement simplifié. Enfin, contrairement aux autres frameworks `Javascript`, nous avons évité la lourdeur des "frameworks tout en un" et nous avons donc eu la possibilité de facilement l'intégrer à d'autres technologies (comme notamment au mobile).

Pour un premier projet d'application Web, l'utilisation du framework `Vue.js` s'est montrée bénéfique en tout point :

- prise en main facile
- bonne courbe de progression
- utilisation d'un DOM virtuel et rapidité du framework
- intégration à d'autres technologies facile (mobile ici notamment)

⁵Internet Explorer 8

⁶La vraie histoire, c'est que cela a été imposé pour faute de "sponsors" du côté de l'équipe Quickmatch
⁷ou presque..

3.1.2 Base de données : PostGRESQL

Nous avons choisi de prendre le langage SQL avec comme SGBD⁸ PostGRESQL.

Plusieurs raisons ont justifié ces choix.

Nous voulions un SGBD qui permettait de faire les opérations de base sans encombre dans la mesure où notre projet ne nécessitait pas d'opérations complexes. Il nous fallait aussi un SGBD facile d'installation, facile de prise en main mais surtout adapté aux petites bases de données. PostGRESQL nous a montré sa force de gestion de clés étrangères (nous avons hésité dans notre choix avec MySQL qui ne prend pas en compte les contraintes sur des clés étrangères), à cela s'est ajouté aussi la gestion de sous-requêtes simplifiées.

En bref, nous avons fait le choix de prendre PostGRESQL qui réunit toutes nos exigences à savoir :

- prise en main facile
- documentation dense (dont nous avons su tirer profit)
- gestion des contraintes de clés étrangères

3.2 Application mobile : Android - Kotlin

Lorsque nous avons commencé à nous intéresser à la partie mobile, nous pensions réaliser un webview (application mobile faite à partir du code de la partie web) dans un premier temps, mais ce n'était pas conseillé, cette méthode générant souvent des comportements non désirés. Nous sommes donc partis de zéro sur un développement en parallèle de l'application mobile début Janvier lorsque le site internet implémentait des fonctionnalités primaires et essentielles.

Actuellement, l'application fonctionne à partir de la version 24 de l'API Android, soit Android 7.0 (Nougat). La version de compilation est Android 10.0 (API 29).

Dans cette section, nous allons brièvement expliquer quelques moyens/méthodes utilisés dans l'implémentation de cette application.

3.2.1 Architecture

Avant de commencer, il faut définir les deux principaux composants de cette architecture : **Activité** et **Fragments** :

- une activité ([documentation](#)) est le point d'entrée principale d'une application qui permet à l'utilisateur d'interagir avec elle et elle peut se composer d'un seul ou de multiples écrans différents.
- un fragment ([documentation](#)) est un morceau d'interface utilisateur ou un comportement.

L'application est construite de façon assez basique en 2 activités. La première est celle de connexion/inscription, appelée activité d'accès à l'application. La seconde est celle dite du contenu, elle contient les fonctionnalités une fois connecté.

Le passage entre les deux se fait après validation du compte qui se connecte, l'activité d'accès transmet à celle de contenu les informations du joueur connecté.

Ensuite, les différentes pages de l'application sont, elles, des fragments. Chaque fragment possède un *layout* associé, il s'agit d'un fichier .xml qui implémente principalement la position des éléments sur l'écran lorsque ce fragment est affiché.

Un fragment possède aussi une classe associée, dans laquelle est implémenté le comportement des éléments du *layout*. C'est également dans cette classe qu'est lancé le chargement des éléments de l'écran à partir du fichier .xml associé.

Enfin, tous ces fragments sont rattachés à l'activité dans laquelle ils servent. Celle-ci, lorsqu'elle en a besoin, crée une instance du fragment voulu ce qui a pour résultat de l'afficher à l'écran. Les fragments sont stockés dans une pile, ce qui permet notamment le fonctionnement du bouton retour de la barre d'outils et de celui du téléphone.

3.2.2 Data Binding

Comme expliqué précédemment, l'affichage des éléments à l'écran dépend de leur disposition dans un fichier .xml. Ce type de fichier contient donc du code *xml*, c'est-à-dire une arborescence de balises correspondant à des éléments d'interface utilisateur. Cela signifie qu'en mémoire, les éléments forment un arbre et donc pour accéder à un élément, il faut parcourir l'arbre.

Dans le code d'un fragment, c'est ce que fait la méthode `findViewById(id)`. Si on veut par exemple changer la couleur d'un élément dynamiquement à l'exécution, il faut tout d'abord récupérer le dit élément et donc appeler la méthode précédente. Or cela peut vite s'avérer très coûteux, car un écran peut aussi bien être simple que complexe avec beaucoup d'éléments. En effet, un écran complexe entraîne un arbre plus important, et donc des parcours plus longs.

⁸Système de gestion de base de données, i.e PostGRESQL

C'est pourquoi nous avons choisi d'utiliser le *Data Binding* dans notre implémentation. Cette solution consiste à créer, à la compilation, un objet de *binding* qui fait le lien entre le code Kotlin et les éléments du *layout*. Ainsi, pour accéder à un élément à l'exécution, il n'y a plus besoin de faire de parcours d'arbre mais seulement d'appeler l'attribut de l'objet obtenu correspondant à notre élément. De plus, cette bibliothèque permet ceci dans les deux sens *i.e.* le code à connaissance du *layout* mais l'inverse est aussi vrai. Ainsi, il est possible d'appeler des méthodes de classes dans le code *xml* par exemple.

En résumé, cette solution permet d'éviter un ralentissement de l'affichage à l'exécution, ce qui rentre dans la philosophie de garder une interface utilisateur fluide en toutes circonstances.

3.2.3 Life Cycles

Toujours dans l'optique d'offrir à l'utilisateur une expérience agréable, il a fallu également s'intéresser aux *lifecycles* des entités (activités et fragments) d'une application Android.

Pour faire simple, les activités et les fragments peuvent se situer dans plusieurs états différents lors de l'utilisation d'une application. Par exemple être à l'écran ou non, être au devant de l'écran ou non.

De plus, ces entités peuvent subir des changements de configuration telle que la rotation de l'écran. Dans ce cas, l'écran actuel est détruit et recréé en format paysage. Cependant qu'en est-il des données affichées ou saisies par l'utilisateur ?

C'est en effet un problème, car par défaut l'application va juste bêtement recharger un nouvel écran tout neuf ne prenant pas en compte les modifications apportées.

Par exemple, tourner un écran qui affiche le résultat d'une longue requête va tout simplement relancer la requête ce qui n'est bien sûr pas optimal.

C'est pourquoi nous avons utilisé dans notre implémentation une classe appelée **ViewModel** ([documentation](#)).

Une telle classe a pour but de gérer toutes les données reliées à une interface utilisateur, tout en les gardant en mémoire même en cas de changement de configuration comme expliqué précédemment.

Les données dans un *ViewModel* sont stockées dans des variables appelées *LiveData*. Il s'agit d'un type de variable observable. En effet, en utilisant un *ViewModel*, on utilise le **design pattern de l'Observer**. La classe d'un fragment donné observe les variables du *ViewModel* et adapte l'affichage en fonction de leur valeur. Pour cela, le *ViewModel* notifie le fragment à chaque changement.

Par exemple, disons que l'on veut afficher à l'écran un minuteur. Il suffit de stocker le temps restant dans une *LiveData* dans le *ViewModel* et l'observer depuis la classe du Fragment qui affiche le temps. Ainsi à chaque changement de temps restant, le temps affiché change. Enfin, une rotation de l'écran ne remet pas le minuteur à sa valeur de départ !

3.2.4 Couroutines

Pour continuer, l'application ne se résume pas à un chronomètre, mais plutôt à l'affichage de données récupérées à partir d'une base de données distante (car commune avec l'application web). Cela implique d'effectuer des requêtes parfois longues. Cependant comme expliqué plus haut, personne n'a envie dans une application de voir son affichage *freeze* ou tout simplement crash. C'est pourquoi, comme il était plus judicieux de séparer données et affichage avec le *ViewModel*, il est aussi bien de séparer l'exécution de l'affichage de celle des requêtes.

C'est pourquoi nous avons utilisé les **Kotlin Couroutines** ([documentation](#)). Cette bibliothèque permet de lancer des traitements longs et asynchrones (tels que des requêtes) sur d'autres threads que le thread principal d'affichage. Ainsi, le main thread ne *freeze* pas en cas de tâches importantes lancées sur les autres threads.

Nous l'avons tout simplement utilisé pour chacune de nos requêtes.

3.2.5 Communication avec la base de données

Pour terminer sur le point technique de l'application mobile, il reste à traiter les requêtes faites sur la base de données distante.

Pour cela nous avons utilisé deux bibliothèques : **Retrofit** ([documentation](#)) et **Moshi** ([documentation](#)).

La première permet de créer en *Kotlin* notre API, c'est-à-dire des fonctions liées à des routes du backend. Celles-ci peuvent prendre des paramètres de n'importe quel type et peuvent aussi renvoyer tout type.

En outre, ces fonctions, précédées par le mot clé *suspend* sont exécutées de façon asynchrone, sans arrêter le fil d'exécution du programme. Ainsi, il nous suffit de les lancer sur un thread à part et d'observer une *LiveData* qui est mise à jour avec le résultat de la requête. Cela sans arrêter l'exécution ni faire *freeze* l'affichage sur le main thread.

Ensuite, la seconde bibliothèque permet de faire le lien entre les données transmises et reçues en *.json* du backend et nos objets *Kotlin*.

En effet, cette bibliothèque est capable de faire l'équivalence en un *.json* et une *data class* si les attributs et les types correspondent.

Par exemple :

```
/* json */  
----- Kotlin -----  
{  
    "id":....,  
    "location":"...","  
    "precise_date":"...","  
    "minimal_team_size":....,  
    "maximal_team_size":....,  
    "deletion_date":....,  
    "played":....  
}  
/* kotlin */  
data class MeetObject(  
    val id : Int,  
    val location : String?,  
    @Json(name = "precise_date") val date : String?,  
    @Json(name = "minimal_team_size") val minimum : Int?,  
    @Json(name = "maximal_team_size") val maximum : Int?,  
    @Json(name = "deletion_date") val date_deletion : String?,  
    val played : Boolean?  
)
```

4 Implémentation réalisée

Cette section vise à présenter tout ce que l'équipe Quickmatch a effectué.

4.1 Application web

Dans cette partie, nous allons aborder ce qui a été implémenté tout au long de ce projet vis à vis de l'interface web de Quickmatch.

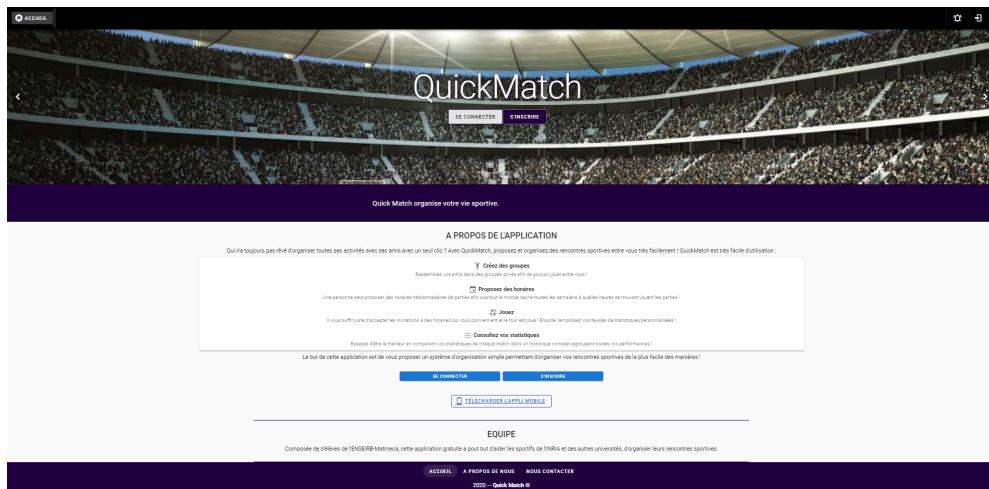


FIGURE 4 – Page d'accueil Quickmatch.fr

4.1.1 Gestion de comptes et authentification

Dans un premier temps, commençons par présenter comment sont gérés les comptes utilisateurs sur Quickmatch. À ce jour il existe deux types de compte : ceux créés directement sur l'application web ou Android ainsi que ceux utilisant Google. Chronologiquement, la connectivité via des comptes Google a été implémentée en première suivi de celle propre à Quickmatch.

Pour s'inscrire, nous avons alors deux options possibles, la seule différence résidant dans le formulaire à remplir pour le faire. En effet, la Figure 5 correspond à la création d'un compte propre Quickmatch où les

champs ne sont pas pré-remplis, alors que sur la Figure 6 seul le pseudo est à déterminer (le nom et prénom étant modifiables mais basés sur le compte Google de l'utilisateur).

FIGURE 5 – Formulaire d'inscription basique

FIGURE 6 – Formulaire d'inscription via Google

Si l'utilisateur utilise Google pour s'inscrire, alors, une fois chose faite, il pourra directement naviguer sur l'application web et rejoindre des clubs par exemple! Tandis qu'un utilisateur passant par une adresse mail⁹ devra effectuer une vérification d'adresse. En effet, à sa création, un mail est automatiquement envoyé depuis notre adresse no.reply.quickmatch@gmail.com¹⁰. Le mail réceptionné est sous la forme présentée par la Figure 7 où le code reçu sera à entrer sur la Figure 8. A noter, que faute de validation, un utilisateur ne pourra utiliser son compte! Il se doit de faire valider son adresse mail pour profiter pleinement de notre application. S'il se déconnecte et se reconnecte, il se retrouvera toujours sur cette page de validation. Un délai d'envoi de deux minutes a aussi été mis en place pour éviter tout abus.



FIGURE 7 – Email de validation de compte

FIGURE 8 – Formulaire de validation

Une fois inscrit, il ne reste plus à l'utilisateur qu'à se connecter quand il le souhaitera via l'interface en Figure 9 ou en utilisant le bouton *Se connecter avec Google* pour renseigner son compte Google en Figure 10.

⁹Logiquement sans terminaison gmail.com

¹⁰Nous avons préféré utiliser le serveur SMTP Google plutôt que Quickmatch pour des raisons de compatibilité

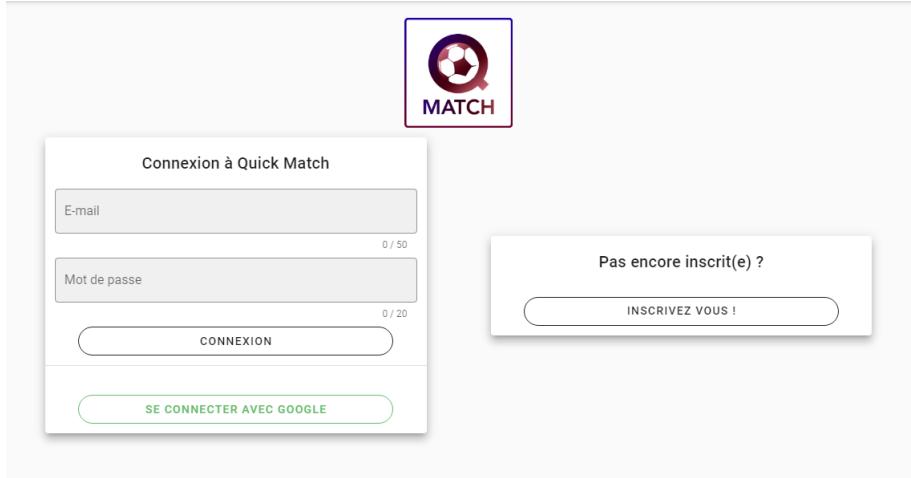


FIGURE 9 – Interface de connexion universelle

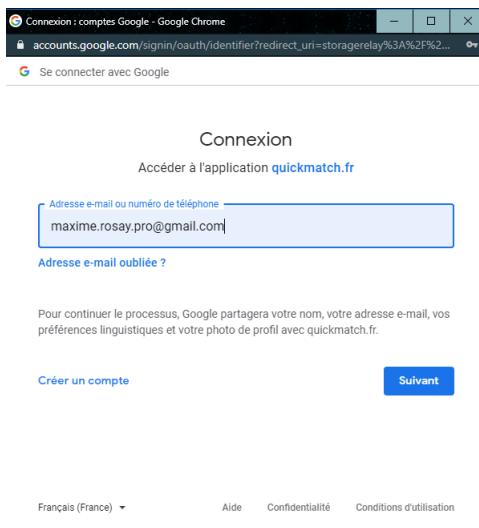


FIGURE 10 – Fenêtre "pop-up" de connexion Google

Afin de faciliter l'usage de **Quickmatch** et pour éviter à l'utilisateur de se reconnecter à chaque fois qu'il quitte la page web, des cookies ont été aussi mis en place afin de préserver sa connectivité.

4.1.2 Sécurisation de Quickmatch

Parlons un peu de sécurité, chose très importante puisque nous stockons des données sensibles comme par exemple des mots de passe.

La première étape a consisté à passer l'entièreté de l'application web en **HTTPS**, par entière nous entendons base de données comprise. Grâce à notre hébergeur **o2Switch**, nous avons pu utiliser un provider de chiffrement très simplement dénommé *Let's Encrypt™ SSL* et recommandé par le support **o2Switch**

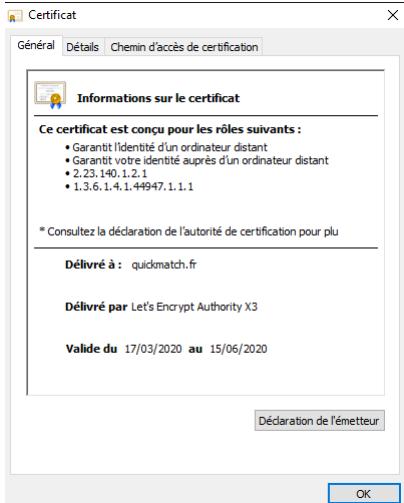


FIGURE 11 – Certification HTTPS web

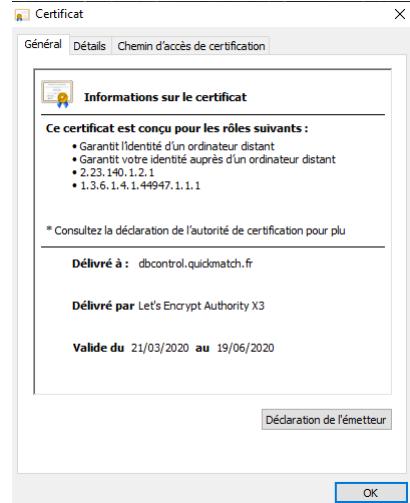


FIGURE 12 – Certification HTTPS base de données

Pour garantir une sécurité au niveau des mots de passe, un chiffrement HTTPS n'est clairement pas suffisant. Nous avons opté alors pour un chiffrement utilisant la méthode SHA-512 pour ces derniers, en y appliquant un *salage* bien spécifique.

4.1.3 Profil joueur

La section *Profil* (Figure 22) affiche à l'utilisateur ses coordonnées personnelles : une première carte qui affiche son pseudo, sa photo de profil et un statut, et une deuxième qui regroupe le nom, prénom, adresse mail et numéro de téléphone. le bouton *EDITER MON PROFIL* donne la possibilité de les modifier à tout temps.

FIGURE 13 – Page profil joueur

4.1.4 Gestion des clubs

La page principale de club nous permet de créer un club en spécifiant son nom et son statut (si un statut est privée, il n'apparaîtra pas dans la liste des clubs qu'on peut rejoindre, il faut qu'un admin du club nous envoie une invitation pour pouvoir y accéder).

Cette page nous permet aussi d'avoir un aperçu de la liste de nos clubs avec quelques informations sur ceux-ci (nombre de joueurs, nombre de matchs joués, ...).

The screenshot shows the main club page. At the top, there's a navigation bar with links for ACCUEIL, AGENDA, STATISTIQUES, MATCHS, PROFIL, CLUBS, and a search icon. Below the navigation is a dark header "Créer un club". A form field "Nom du club" is present with a character limit of 50. There's a checkbox for "Club privé" and a "CRÉER LE CLUB" button. Below this is a section titled "Vos clubs" containing a table of existing clubs:

Nom du Club	Date de création	Nombre de joueur(s)	Nombre de match(s) joué(s)	Club privé	Administrateur	Quitter le club	Gérer le club
ariba	02/04/2020	10	0	non	oui	QUITTER LE CLUB	GÉRER LE CLUB
1A Enseirb	02/04/2020	3	0	non	oui	QUITTER LE CLUB	GÉRER LE CLUB
bevevev	18/03/2020	2	0	non	non	QUITTER LE CLUB	VOIR LE CLUB
Toast	11/03/2020	3	0	non	non	QUITTER LE CLUB	VOIR LE CLUB
2A Enseirb	11/03/2020	9	0	non	non	QUITTER LE CLUB	VOIR LE CLUB

At the bottom of the page are links for ACCUEIL, A PROPOS DE NOUS, and NOUS CONTACTER, along with a footer note: 2020 — Quick Match ©.

FIGURE 14 – Page principale club

En cliquant sur *rejoindre un club*, nous arrivons sur la liste des clubs de profil public dont on ne fait pas encore partie, il suffit d'un clic pour le rejoindre

The screenshot shows the "Rejoindre un club" page. At the top, there's a link "RETOURNER AU MENU PRINCIPALE". Below it is a table listing clubs available to join:

Nom du Club	Date de création	Nombre de joueur(s)	Nombre de match(s) joué(s)	Demander à rejoindre
clubApp	03/04/2020	1	0	DEMANDER À REJOINDRE
Coucou	11/03/2020	1	0	DEMANDER À REJOINDRE
hola	18/03/2020	1	0	DEMANDER À REJOINDRE
Inria	16/11/2019	5	0	DEMANDER À REJOINDRE
New	18/03/2020	1	0	DEMANDER À REJOINDRE

At the bottom of the page are links for ACCUEIL, A PROPOS DE NOUS, and NOUS CONTACTER, along with a footer note: 2020 — Quick Match ©.

FIGURE 15 – Page rejoindre un club

Depuis la page principale de club, nous pouvons voir ou gérer un de nos club (bouton *gérer le club/voir le club*), selon que l'on soit un admin ou pas. Concrètement, un admin a le pouvoir de supprimer un joueur non admin de son club, mais aussi de promouvoir un joueur au rang d'admin ou d'envoyer une invitation à un joueur non membre.

Cette partie nous permet également de voir les statistiques des membres au sein du club (nombre de but, nombre de matchs joués, ...).

The screenshot shows a web application interface for managing a club. At the top, there are navigation links: ACCUEIL, AGENDA, STATISTIQUES, MATCHS, PROFIL, CLUBS, and MODIFIER LE CLUB. Below this is a section titled "Vos statistiques au sein du club ariba" (Your statistics within the club ariba) with a table showing the following data:

Nom	Prénom	Nombre de but(s)	Nombre de but(s) encaissé(s)	Nombre de match joué(s)	Nombre de victoire(s)
Danquigny	Antoine	0	0	0	0

Below this is a section titled "Joueurs du club ariba" (Players of the club ariba) with a table showing the following data:

Nom	Prénom	Pseudo	Nombre de but(s)	Nombre de but(s) encaissé(s)	Nombre de match joué(s)	Nombre de victoire(s)	Nommer admin	Supprimer
Danquigny	Antoine	ad	0	0	0	0	<button>NOMMER ADMIN</button>	<button>SUPPRIMER</button>
test	test	testino	0	0	0	0		
Abderrahmane	Faiz	psfaz	0	0	0	0	<button>NOMMER ADMIN</button>	<button>SUPPRIMER</button>
Faverge	Mathieu	fm	0	0	0	0	<button>NOMMER ADMIN</button>	<button>SUPPRIMER</button>
Hala	Mehdi	BigShaq1208	0	0	0	0	<button>NOMMER ADMIN</button>	<button>SUPPRIMER</button>

At the bottom of the page are navigation buttons (< > 2) and a button labeled "AJOUTER UN JOUEUR".

FIGURE 16 – Page gérer/voir son club

Enfin, depuis la page de gestion de son club nous pouvons envoyer une invitation à un joueur (bouton *ajouter un joueur*). Les statistiques visibles sont leurs statistiques globales.

The screenshot shows a web application interface for adding a player. At the top, there are navigation links: ACCUEIL, AGENDA, STATISTIQUES, MATCHS, PROFIL, CLUBS, RETOURNER AU MENU PRINCIPALE, and RETOURNER AU MENU DE GESTION DU CLUB "ARIBA". Below this is a section titled "Ajouter un joueur" (Add a player) with a table showing the following data:

Nom	Prénom	Pseudo	Nombre de but(s)	Nombre de but(s) encaissé(s)	Nombre de match joué(s)	Nombre de victoire(s)	Ajouter
Faverge	Mathieu	Mateo1170	0	0	0	0	<button>AJOUTER</button>
HALA	MEHDI	BigShaq1	0	0	0	0	<button>AJOUTER</button>
Issaoui	Ali	Alisrey7	2	2	1	0	<button>AJOUTER</button>
ISSAOUI	Ali	AlisRey7	0	0	0	0	<button>AJOUTER</button>
Jean	ZHOO	Abdelkaderyaboualem	0	0	0	0	<button>AJOUTER</button>

At the bottom of the page are navigation buttons (< > 2) and a footer with links: ACCUEIL, A PROPOS DE NOUS, NOUS CONTACTER, and 2020 – Quick Match ©.

FIGURE 17 – Page ajouter un joueur

4.1.5 Calendrier

La calendrier représente, pour l'utilisateur, un rapide résumé du mois en cours affichant :

- Les invitations de match reçues.
- Les matchs futurs (avec un affichage différent selon la réponse de l'utilisateur)
- Quelques matchs passés.

Sur la Figure 24 ci dessous, on peut voir une interface horizontale au dessus de l'agenda qui possède les fonctionnalités suivantes :

- Aujourd'hui : permet de revenir à la date d'aujourd'hui.
- Créer un match : redirige vers la page de création de match
- Menu défilant : qui permet de choisir le mode d'affichage entre :
 - mois
 - semaine
 - 4 jours
 - jour

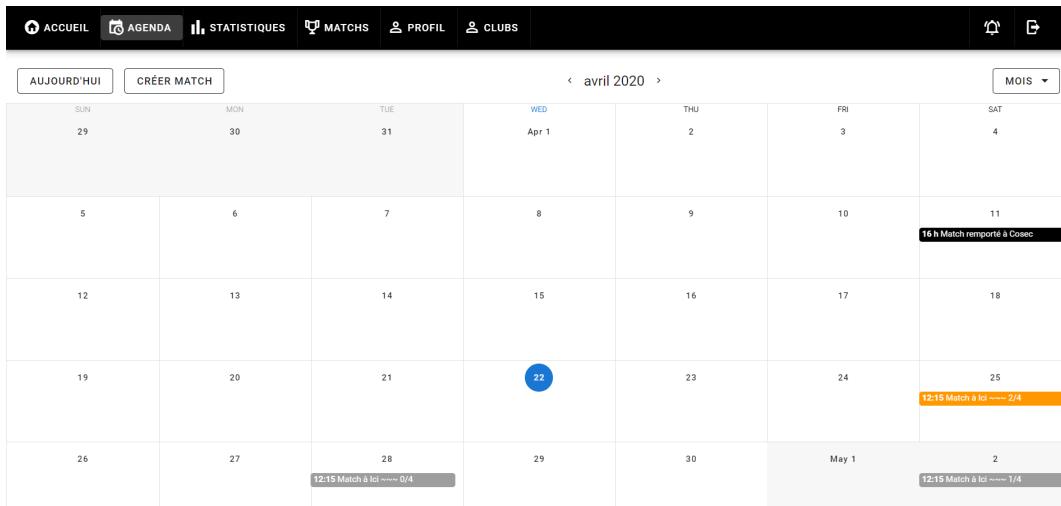


FIGURE 18 – Page de l'agenda

Un code couleur a aussi été mis en place pour faciliter l'accès à un type de données spécifique. Le code couleur est le suivant :

- **Rouge** : Le match n'aura pas lieu puisque le nombre de joueurs invités moins le nombre de joueurs qui ont refusés est inférieur au minimum requis du match.
- **Orange** : Le minimum de joueurs requis n'est toujours pas atteint et il reste suffisamment de joueurs qui n'ont toujours pas répondu à l'invitation pour arriver à celui-ci.
- **Vert** : Le minimum de joueurs requis a été atteint.
- **Gris** : Match refusé.
- **Bleu** : Invitation sans réponse.
- **Noir** : Match déjà joué.

La page relative à l'agenda présente aussi la possibilité de créer un nouveau match, répondre à une invitation ainsi que modifier sa réponse à une invitation.

Création de match :

Création de match

Club

Heure de début

Heure de fin

Localisation

Min et Max de joueurs requis par équipe pour le match

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Jour de répétition:

Lundi Mardi Mercredi Jeudi Vendredi Samedi Dimanche

CRÉER !

FIGURE 19 – Page de création de match

Notre version répète les matchs seulement pour la semaine courante ainsi que la suivante.
Fenêtre de match :

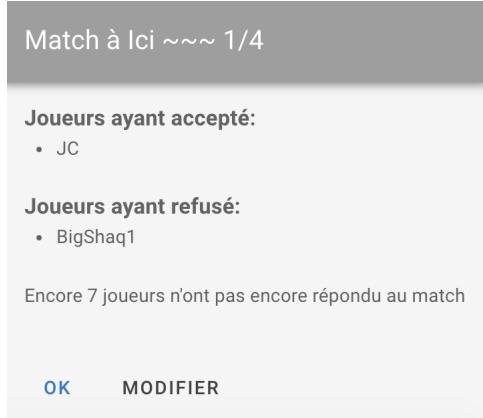


FIGURE 20 – Fenêtre match refusé

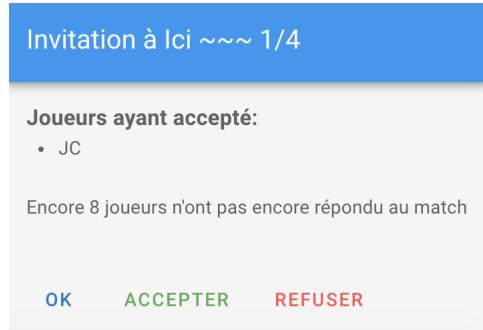


FIGURE 21 – Fenêtre invitation

Il y a deux types de fenêtres :

- Si l'invitation a déjà reçu une réponse (cf Figure 20) : Le bouton modifier permet de remettre le match en invitation si la personne a changé d'avis, i.e que si l'on avait accepté ou refusé un match, cela permet de revenir sur sa décision.
- Sinon (cf Figure 21) : On peut accepter ou refuser le match directement de l'agenda.
- Dans les deux figures, le bouton "OK" se contente de fermer la petite fenêtre.

4.1.6 Invitations

La page invitation recense toutes les invitations, qu'elles soient pour rejoindre un club ou bien un match, en y ajoutant quelques petites informations sur les clubs/matchs dont on a reçu une invitation, afin de nous aider à prendre une décision.

Invitation(s) pour un match					
Localisation	Date	Nombre minimum de joueurs	Nombre maximum de joueurs	Refuser	Accepter
Ici	Le Samedi de 12:15:00 à 05:15:00 2	6		<button>REFUSER</button>	<button>ACCEPTER</button>
Ici	Le Samedi de 12:15:00 à 05:15:00 2	6		<button>REFUSER</button>	<button>ACCEPTER</button>
Ici	Le Samedi de 12:15:00 à 05:15:00 2	6		<button>REFUSER</button>	<button>ACCEPTER</button>
Ma chambre	Le Mardi de 14:09:00 à 16:19:00 1	1		<button>REFUSER</button>	<button>ACCEPTER</button>
Ici	Le Mardi de 12:15:00 à 05:15:00 2	6		<button>REFUSER</button>	<button>ACCEPTER</button>

Invitation(s) dans un club					
Nom du Club	Date de création	Nombre de joueur(s)	Nombre de match(s) joué(s)	Refuser	Accepter
ariba	02/04/2020	0	10	<button>REFUSER</button>	<button>ACCEPTER</button>
zamlacha	07/02/2020	0	2	<button>REFUSER</button>	<button>ACCEPTER</button>

FIGURE 22 – Page Invitations

4.1.7 Suivi de matchs et statistiques

Les deux modules Matchs et Statistiques permettent à l'utilisateur de gérer la liste des matchs qu'il y a chaque semaine, de voir et de modifier sa décision pour les matchs auxquels il a été ajouté, ainsi que comme indiqué dans le titre, de consulter les statistiques des matchs précédents.

The screenshot shows the 'Matchs' (Matches) section of the application. At the top, there are tabs for ACCUEIL, AGENDA, STATISTIQUES, MATCHS (which is highlighted), PROFIL, and CLUBS. On the right side, there are icons for notifications and a search bar.

Vos Matchs de cette semaine

Jour	Heure	Localisation	Joueurs	Votre Décision	Décider
Mardi	12:15:00	Ici	0/4	En Attente	<button>ACCEPTER</button> <button>REFUSER</button>
Samedi	12:15:00	Ici	4/4	Accepté	<button>MODIFIER</button>
Vendredi	16:02:00	là bas	1/12	Accepté	<button>MODIFIER</button>
Vendredi	12:15:00	là bas	1/10	Accepté	<button>MODIFIER</button>
Mardi	12:15:00	Ici	0/4	En Attente	<button>ACCEPTER</button> <button>REFUSER</button>

Below the table are navigation arrows: < 1 2 3 4 >. A button labeled 'CRÉER UN MATCH' is located on the right.

Historique des matchs

Date	Heure	Localisation	Résultat	Buts marqués	Buts encaissés
14/04/2020	14:09:00	Ma chambre	Défaite	0	2
10/04/2020	16:00:00	Ma cuisine	Victoire	1	0

Below the table are navigation arrows: < 1 >. At the bottom of the page, there are links: ACCUEIL, A PROPOS DE NOUS, and NOUS CONTACTER.

FIGURE 23 – La page Matchs

The screenshot shows the 'Statistiques' (Statistics) section of the application. At the top, there are tabs for ACCUEIL, AGENDA, STATISTIQUES (which is highlighted), PROFIL, and CLUBS. On the right side, there are icons for notifications and a search bar.

Statistiques globales

Nom	Prénom	Nombre de but(s)	Nombre de but(s) encaissé(s)	Nombre de match joué(s)	Nombre de victoire(s)
Danquigny	Antoine	5	1	1	1

FIGURE 24 – Les statistiques

La liste des matchs de la semaine actuelle respecte le même code couleur décrit précédemment dans la rubrique 4.1.5, cette page permet aussi de consulter la liste des joueurs ayant acceptés ou refusés un match et le nombre d'invités qui n'ont pas répondu.

Les statistiques sont divisées en deux parties. D'abord les résultats des matchs avec le nombre de buts marqués et encaissés, et ensuite les résultats personnels avec le nombre global de buts marqués et encaissés dans tous les matchs qui ont déjà eu lieu.

4.1.8 Hébergement et gestion de la base de données

Parlons maintenant de o2Switch, l'hébergeur WEB de l'application Quickmatch et de la base de données. L'interface de gestion de o2Switch propose divers outils qui nous ont permis de déployer correctement les fonctionnalités nécessaires au fonctionnement de Quickmatch. Notamment la configuration des domaines et des sous-domaines, la gestion des fichiers sources de l'application, l'installation des applications Node js, le terminal pour lancer des commandes avancées, la gestion de la base de données, etc... Tout ces instruments sont regroupés dans le cpanel (Figure 25).



FIGURE 25 – Cpanel : rubriques fichiers et base de données

Le gestionnaire de fichiers est une interface simple à utiliser pour charger les fichiers sources, les visualiser voire même les modifier. La base de données **postgreSQL** est gérée grâce à l'outil **phpPgAdmin** et **adminer** qui permettent de gérer entièrement notre base de données.

Le *cpanel* offre aussi une vue globale sur l'état de l'hébergement avec des statistiques et des chiffres clés sur le nombre de processus actifs et le pourcentage d'utilisation de la mémoire par exemple (Figure 26).

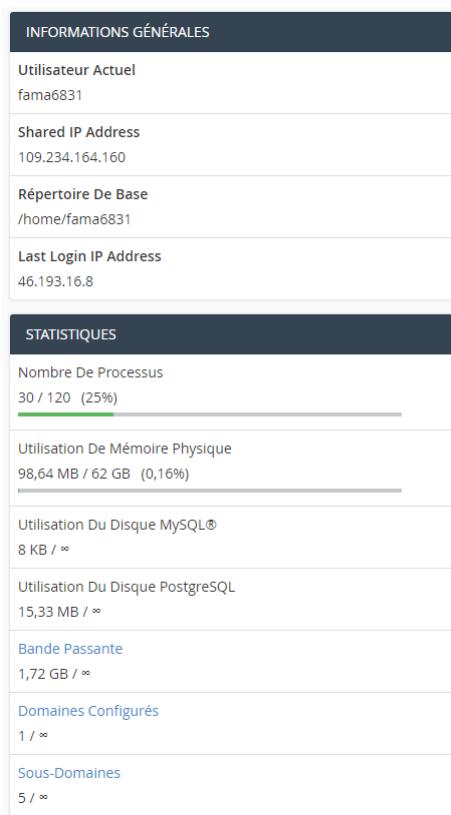


FIGURE 26 – Cpanel : Statistiques et état d'hébergement

Un outil très important présent dans le *cpanel* est le **setup Node.js App** qui permet de mettre en service des applications **Node.js**, en l'occurrence l'API nous permettant de communiquer les informations de la base de données à l'application web et Android, qui est disponible d'ailleurs sur **dbcontrol.quickmatch.fr**.

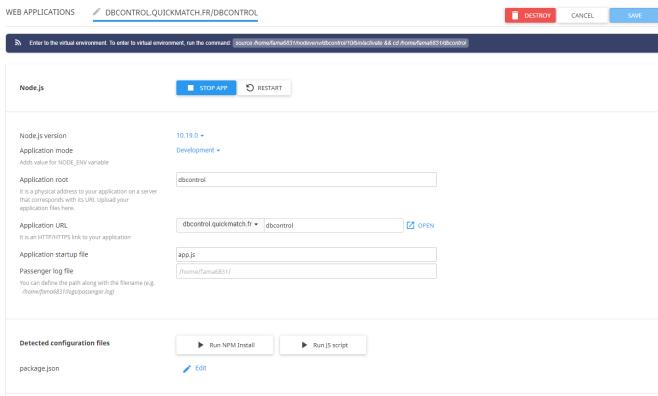


FIGURE 27 – Setup NodeJs App

```

3   {
4     "rows": [
5       (
6         {
7           "id": 265,
8           "surname": "████████",
9           "first_name": "M████████",
10          "pseudo": "████████",
11          "mdy": null,
12          "mail_address": "████████@████████.fr",
13          "phone_number": null,
14          "scored_goals": 0,
15          "conceded_goals": 0,
16          "matches_played": 0,
17          "victories": 0,
18          "avatar": null,
19          "bio": null,
20          "is_valid": true,
21          "private_profil": null
22        },
23        (
24          "id": 1,
25          "surname": "TITI",
26          "first_name": "KOTEO",
27          "pseudo": "TOKITOKI",
28          "mdy": null,
29          "mail_address": "████████@████████.fr",
30          "phone_number": "████████",
31          "scored_goals": 1,
32          "conceded_goals": 0,
33          "matches_played": 1,
34          "victories": 1,
35          "avatar": "kotsq",
36          "bio": null,
37          "is_valid": true,
38          "private_profil": true
39        },
40        (
41          "id": 5,
42          "surname": "Hala"
43        )
44      ]
45    }
46  }

```

FIGURE 28 – API après installation de l’application

Il s’agit concrètement d’une application en Javascript créant des objets JSON pour répondre aux requêtes que nous avons implémentées et qui sont utiles par la suite pour l’interface web et Android de Quickmatch. L’API compte déjà 67 requêtes.

```

// Table invitation
app.post("/dbcontrol/api/v1/Invitations", _invitation2.default.create);
app.get("/dbcontrol/api/v1/Invitations", _invitation2.default.getAll);
app.get("/dbcontrol/api/v1/Invitations/:id", _invitation2.default.getOne);
app.put("/dbcontrol/api/v1/Invitations/:id", _invitation2.default.update);
app.delete("/dbcontrol/api/v1/Invitations/:id", _invitation2.default.delete);
app.get("/dbcontrol/api/v1/InvitationsRows", _invitation2.default.getAllRows);

```

Extrait du code qui affiche quelques routes correspondant à la table invitation

4.1.9 test.quickmatch.fr : le site test !

Afin de tester les fonctionnalités à grande échelle, sur une base de données plus importante, un site de test a été créé, il devait servir à valider les nouvelles fonctionnalités avant de les intégrer au site principal et éventuellement à repérer des problèmes de performance ou d’affichage sur des données plus massives. Malheureusement, faute de temps et de méthode d’organisation rigoureuse, les fonctionnalités étaient souvent directement implémentées sans passer par l’intermédiaire de `test.quickmatch.fr`.

En interne, un script C écrit un fichier `.sql` chargé de peupler la base avec un grand nombre d’entrées, pour ce faire, les données de chaque champ de chaque table sont issues de listes aléatoires récupérées sur cette source. Côté technique, certaines écritures sont modifiées en interne par le script pour se conformer aux contraintes d’intégrité de la base de données, d’autres convenaient sous la forme proposée par www.randomlists.com. Les clés primaires (id) de chaque table sont incrémentées à chaque insertion les concernant, dans le but de garantir leur unicité.

Enfin, un problème de maintenabilité a assez retardé la mise en place d’un site de test fonctionnel : chaque changement dans la définition de la base implique un changement dans le code du script C et la nécessité de générer de nouveau un fichier `.sql` en cohérence avec la nouvelle base.

4.2 Application mobile

Il est maintenant temps de faire le point sur la version mobile de Quickmatch. À ce jour, une application **Android** est disponible. Il est possible de la télécharger via un bouton sur la page d’accueil du site web (disponible ici) ou depuis le dépôt git, où le fichier `.apk` d’installation est *push* régulièrement.

Tout comme pour la version web, nous allons ici faire le tour des fonctionnalités disponibles de l’application dans sa version actuelle et potentiellement voir les différences avec le site web.

Lancement

Sur la Figure 29 on peut observer le logo de l'application.



FIGURE 29 – Icône de lancement



FIGURE 30 – Écran titre de l'application

Au lancement de l'application, la première page proposée aux utilisateurs est celle de la Figure 30.

Après un test de connexion au serveur en ligne, l'utilisateur peut choisir entre "Se connecter" s'il a déjà un compte ou "S'inscrire". Il est évidemment possible de naviguer entre les pages de connexion, d'inscription et cette page d'accueil. À noter que l'application est entièrement au format portrait.

Connexion



FIGURE 31 – Écran de connexion



FIGURE 32 – Détection de compte Google

La page sur la Figure 31 est celle de connexion à l'application. Comme pour la version web, il est possible de se connecter avec un compte Quickmatch mais aussi avec un compte Google :

- Quickmatch : il suffit de saisir son adresse mail et son mot de passe.

- Google : appuyer sur le bouton permet de choisir un compte Google. Si c'est la première fois que ce compte est choisi, un compte Quickmatch associé sera créé automatiquement. Sinon, la connexion est immédiate, avec le compte utilisé habituellement.

L'interface de connexion offre aussi une détection automatique de compte Google (Figure 32). En effet, si l'utilisateur s'est déjà connecté avec un tel compte, il lui sera proposé de se connecter avec ce compte en entrant sur la page de connexion (Figure 31).

Inscription

Sur la Figure 33 est illustrée la page d'inscription à Quickmatch.

FIGURE 33 – Écran de création de match

Sur cette page il est possible de s'inscrire manuellement avec un compte Quickmatch (Google reste conseillé pour plus de sécurité). À l'exception du numéro de téléphone, **tous les champs sont obligatoires**. En effet, nous avons considéré dans la base de données ces champs comme *NOT NULL*, il est donc important de respecter cette contrainte. Chacun d'entre eux doit respecter un certain format, c'est pourquoi lorsque l'on quitte l'édition d'un champ pour un autre, son format est vérifié.

Une fois les champs remplis, un bouton "Vérifier les informations" permet de vérifier l'unicité de certains champs qui identifient un utilisateur telle que l'adresse mail. Ainsi, s'ils sont corrects, on peut procéder à l'inscription avec le bouton "S'inscrire".

A noter que contrairement à la version web, la version mobile ne possède pas de vérification de l'adresse mail. Il est donc conseillé de passer par le site web pour créer un compte Quickmatch.

Accueil

Une fois inscrit ou connecté avec succès, l'utilisateur est redirigé à l'intérieur de l'application et arrive sur la page suivante montrée sur la Figure 35 :

C'est à partir de cette interface qu'il peut accéder à toutes les fonctionnalités disponibles :

- invitations
- matchs
- statistiques
- clubs
- agenda
- profil

Pour cela, il suffit de *swipe* vers la droite à partir du bord à gauche de l'écran ou sinon d'appuyer sur le bouton menu en haut à gauche dans la barre d'outils. Cela ouvre le menu de l'application contenant la liste des différentes pages accessibles, visibles sur la Figure 34.

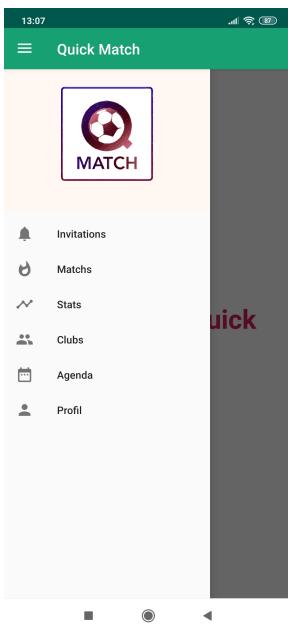


FIGURE 34 – Menu de navigation



FIGURE 35 – Écran d'accueil

Invitations

Le premier choix offert par le menu est donc la page d'invitations. C'est ici que l'utilisateur va pouvoir exprimer son choix pour chaque invitation qu'il reçoit. La page est séparée en deux parties :

- Nouvelles invitations
- Invitations traitées

Il est possible de naviguer entre les deux en cliquant sur les onglets ou en swipant.

La partie "nouvelles invitations" affiche la liste des invitations que l'utilisateur n'a pas encore traitées, c'est-à-dire pour lesquelles il n'a pas fait son choix. Pour chaque invitation affichée, il peut donc choisir de l'accepter ou de la refuser à l'aide des boutons associés comme illustré sur la Figure 36 suivante :



FIGURE 36 – Onglets nouvelles invitations

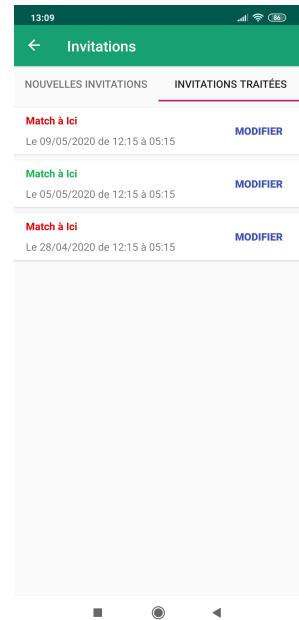


FIGURE 37 – Onglet invitations traitées

"Accepter" ou "Refuser" une invitation redirige vers la partie traitée des invitations pour pouvoir immédiatement modifier son choix et surtout voir que le choix effectué a bien été pris en compte.

La partie invitations traitées de l'interface propose donc à l'utilisateur un affichage des invitations qu'il a déjà acceptées ou refusées. Chacune des invitations affichées possède un bouton "Modifier" pour qu'il puisse

changer ce choix comme montré sur la Figure 37.

Matches

Ce second choix du menu redirige l'utilisateur vers la page des matchs. Tout comme la page d'invitations, cette page est séparée en deux onglets :

- Matchs terminés (Figure 38)
- Matchs à venir (Figure 39)

Les matchs terminés sont ceux qui ont été joués, pour lesquels un score et des statistiques ont été saisis. Pour le moment, seule une liste de matchs est affichée, ceux en vert sont des victoires pour l'utilisateur et ceux en rouge des défaites.

Les matchs à venir sont donc les matchs prévus qui n'ont pas été joués pour la base de données. C'est pourquoi chacun d'entre eux possède un bouton "Jouer".

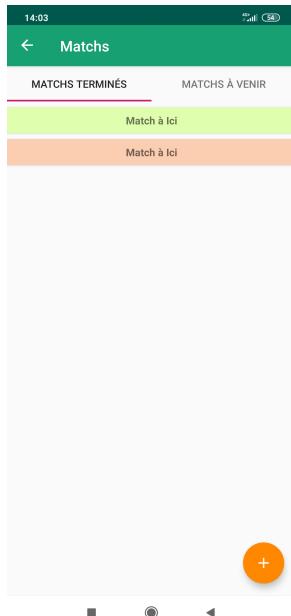


FIGURE 38 – Onglet matchs terminés



FIGURE 39 – Onglet matchs à venir

Celui-ci redirige vers une page permettant à l'utilisateur de jouer le match. La page offre simplement un minuteur et un bouton "Finir le match" (Figure 40). Le plus important est le bouton en question, qui permet de naviguer vers la page de saisie de la feuille de match (Figure 41).

L'utilisateur peut saisir le score du match ainsi que les statistiques personnelles de chaque joueur du match :

- son équipe
- les buts marqués

Une fois ceci rempli de façon cohérente (des vérifications sont effectuées), le bouton "Envoyer la feuille de match" permet de finaliser la saisie et d'enregistrer tout dans la base de données, ceci termine alors le match côté serveur.

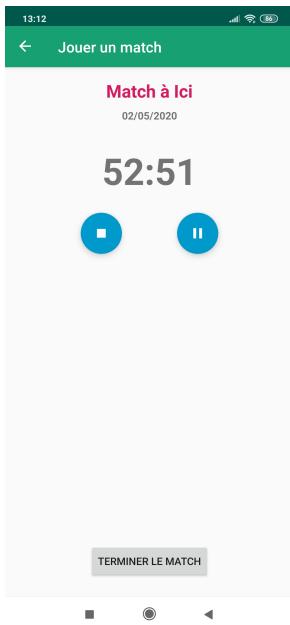


FIGURE 40 – Écran pour jouer un match



FIGURE 41 – Écran pour remplir une feuille de match

Ensuite, il reste une fonctionnalité à cette page des matchs. En effet, sur la page d'accueil des matchs, il y a un bouton en bas à droite.

Celui-ci permet de naviguer vers l'interface de **création d'un match**. Voici le visuel (Figure 42) :

FIGURE 42 – Écran de création de match

Sur cette page, on peut créer plusieurs matchs pour un club donné. Pour cela, il faut tout d'abord choisir le club en question dans la première zone de saisie. La saisie permet d'afficher des suggestions de clubs possibles. Si le nom de club choisi est erroné il sera effacé, c'est pourquoi il est préférable de choisir un nom suggéré. Ensuite, l'utilisateur peut choisir les horaires et la date du match. Les horaires seront les horaires pour tous les matchs créés par cette procédure. Quant à la date, il s'agit simplement de la date de la semaine du premier match.

Par exemple, si l'on veut créer un match le mercredi et le jeudi de la semaine actuelle et les répéter pour plusieurs semaines, il suffit en premier lieu choisir une date quelconque de la semaine actuelle. Il est important de garder en tête que les semaines commencent le dimanche.

Il est ensuite possible pour l'utilisateur d'entrer un nom de lieu pour le match (optionnel) et de choisir le nombre de joueurs minimum et maximum pour les matchs qu'il créé. En effet, dans un but de faire évoluer

l'application (utilisation pour d'autres sports), nous avons fait le choix de laisser de la flexibilité concernant le nombre minimal et maximal de joueurs.

Enfin, dans la partie concernant les répétitions, il est possible de choisir les jours précis où il y aura match et le nombre de semaines de répétition.

Ainsi, si l'utilisateur choisit une date, deux horaires, des jours et un nombre de répétitions il créera un match pour chaque jour souhaité de la semaine choisie, toujours aux horaires sélectionnés. Ces matchs seront répétés selon le nombre de répétitions voulu. Par exemple, si le nombre est 2, les matchs de la semaine choisie seront répétés les deux semaines suivantes. Si le nombre est 0, alors ils ne seront tout simplement pas répétés.

Une fois les bonnes informations saisies, il suffit d'appuyer "Créer le match" pour lancer la création de l'ensemble des matchs ainsi que les invitations et être redirigé vers la page principale des matchs.

Stats

La page "Stats" a pour rôle d'afficher les statistiques du joueur connecté, comme illustré sur la Figure 43.

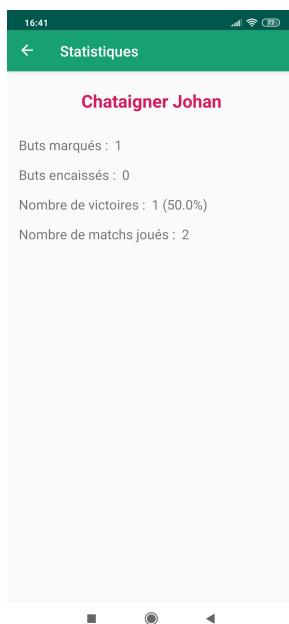


FIGURE 43 – Écran de statistiques du joueur

Il s'agit d'une version minimale des statistiques, qui pourrait évoluer par la suite avec des statistiques par club par exemple.

Clubs

L'interface suivante est donc celle des clubs. La page d'accueil affiche à l'utilisateur la liste des clubs auxquels il appartient. Voici un exemple de visuel présent sur la Figure 44 :

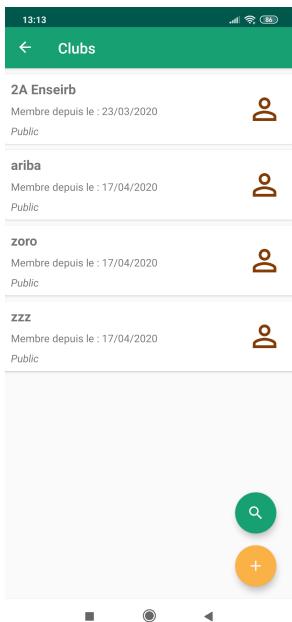


FIGURE 44 – Écran des clubs

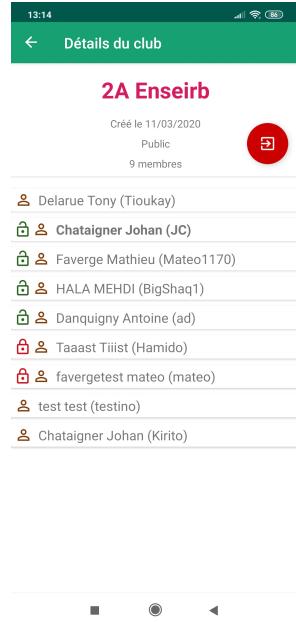


FIGURE 45 – Interface d'un club en particulier

Comme on peut le voir, à droite de chaque club il y a un icône. Celui-ci indique à l'utilisateur s'il est membre ou administrateur du club tout simplement.

En cliquant sur un de ses clubs, il peut ainsi naviguer vers la page de détails du club en question (Figure 45). Celle-ci permet de voir la liste des joueurs du club avec pour chacun le statut de leur profil (privé ou public) ainsi que leur statut dans le club (administrateur ou membre).

Dans la partie supérieure qui contient les informations du club, il y a un bouton permettant de quitter le club.

En retournant sur la page d'accueil, on peut observer en bas à droite deux boutons. Celui avec la loupe redirige vers la page des clubs disponibles (Figure 46). Celle-ci affiche à l'utilisateur la liste des clubs publics qu'il peut rejoindre librement en un seul clique sur l'icône associée.

Le second bouton quant à lui redirige l'utilisateur vers la page de création d'un club (Figure 47). Sur cette page, il suffit de choisir un nom valide de club et le statut privé ou public pour créer simplement un club.



FIGURE 46 – Écran des clubs à rejoindre

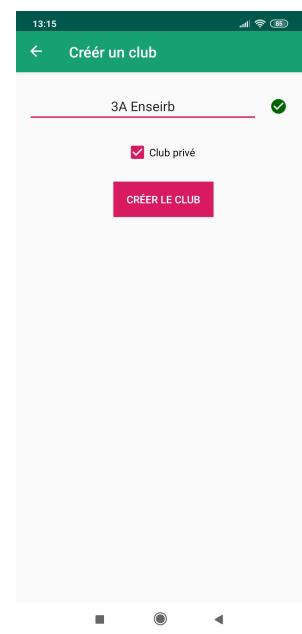


FIGURE 47 – Écran de création de club

Agenda

La page suivante est celle de l'agenda. Contrairement à la version web, cette fonctionnalité n'est pas encore disponible sur la version Android.

Profil

Le dernier onglet du menu fait naviguer l'utilisateur vers son profil (Figure 48).

Il peut y voir son image de profil ainsi que son nom, son prénom et son pseudo. Dans la partie "Infos", il peut visualiser son adresse mail, s'il en a renseigné un, son numéro de téléphone, ainsi que le status privé ou non de son compte. La partie "Préférences" est pour le moment un affichage car les fonctionnalités concernées ne sont pas réalisables à l'heure actuelle.

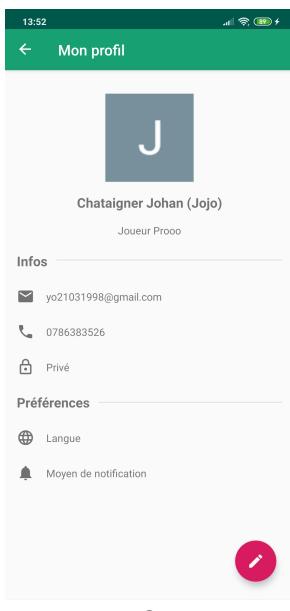


FIGURE 48 – Écran de profil du joueur

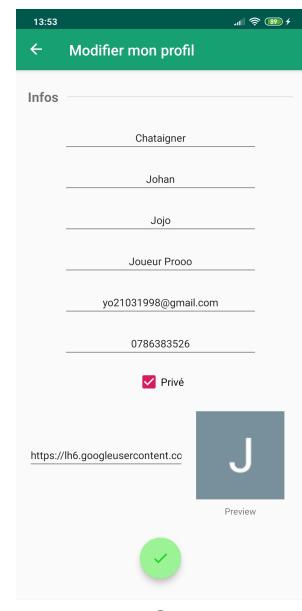


FIGURE 49 – Écran d'édition du profil

Enfin, la page possède en bas à droite un bouton avec un icône d'édition qui redirige vers la page d'édition du profil de l'utilisateur (Figure 49). Dans celle-ci, il peut modifier les informations suivantes :

- Nom
- Prénom
- Pseudo
- Description
- Adresse mail
- Numéro de téléphone
- Status du profil (privé/public)
- Image de profil

Il suffit ensuite de cliquer sur le bouton vert en bas de page pour valider les changements effectués.

5 Capitalisation d'expérience sur les outils collaboratifs utilisés

Dans cette section il est question de revenir et discuter des divers outils que nous avons mis en place pour avancer et communiquer au mieux dans ce projet. Outre le code, il s'agit aussi d'un projet de management, une sorte d'initiation à la vie professionnelle future.

Notre méthodologie de travail était la suivante¹¹ :

1. S'affecter une tâche présente sur le Trello.
2. Développer cette dernière sur une **nouvelle** branche du dépôt `git` s'il s'agissait de code.
3. Tester les changements / ajouts sur le site de test en prévenant obligatoirement l'équipe que le site de test est en *maintenance*.
4. Une fois le test concluant, synchroniser la branche de développement sur la branche principale en gérant les conflits potentiels.
5. Appliquer l'étape 3 mais au site officiel.
6. Rencontrer le client et discuter de la fonctionnalité ajoutée

5.1 Synchronisation du code via *Github*

Concernant notre gestion de l'outil Github, nous avons du bon et du moins bon. En effet, nos modules étaient bien séparés et chacune des personnes travaillaient sur leurs fonctionnalités respectives ce qui nous permettait d'éviter assez souvent des conflits de code.

Cependant, notre méthodologie de push est à revoir. Malgré le fait que la plupart des ajouts fonctionnaient après un push, nous aurions dû mettre en place plus tôt une base de données de tests permettant donc de tester des fonctionnalités. Nous nous retrouvions à tester les fonctionnalités de l'application avec une base de données remplie sur le site web public et donc il aurait fallu implémenter des branches de tests locales et une base de données de tests avant de passer aux push finaux.

5.2 Suivi des tâches grâce à *Trello*

Concernant l'outil de suivi de tâches, nous avons eu une utilisation non régulière. En effet, malgré le fait que lors du commencement du projet nous avons été rigoureux quant à l'ajout, le suivi et la mise à jour des différentes tâches, dès lors que les tâches devenaient plus techniques mais surtout plus spécifiques en particulier, nous avons omis de les recenser. De fait, lorsqu'une personne attaquait une partie plus technique de son module, il était souvent difficile de suivre l'avancée des fonctionnalités pour les autres membres du groupe n'effectuant pas cette partie de l'application.

Nous aurions dû nous attarder plus en détail concernant le suivi des tâches, fondamentalement jouer plus le jeu sur le long terme, et accorder une vraie importance à cet outil : non seulement il aide aussi l'équipe de développement à avancer et comprendre les fonctionnalités développées mais aussi aux clients qui peuvent vraiment savoir sur quelles tâches l'équipe s'est attardée durant le sprint.

De plus, nous n'avons pas exploité le plein potentiel de l'outil en attribuant notamment des poids d'importance pour chacun des tâches qui ont été développées. Cela peut paraître anodin, cependant durant le développement de notre projet, nous n'avons pas mis en lumière les tâches qui nécessitaient le plus d'attention de l'équipe projet et donc nous avons développé des fonctionnalités qui ne sont pas essentielles au lieu de se focaliser sur d'autres points plus importants selon le client.

¹¹À noter que pour presque chaque étape, informer le groupe était très recommandé

5.3 Plateforme de communication avec le client : *Slack*

Concernant l'outil Slack, qui de base servait à un lieu de communication entre les développeurs et les clients, nous l'avons trop vite délaissé. En effet, au bout de quelques semaines, plus rien n'était posté dessus. Nous avons retenu une chose de cette mauvaise gestion d'outil : ne pas mettre en place trop de canaux de communication entre la maîtrise d'oeuvre et la maîtrise d'ouvrage. Les canaux utilisés étaient déjà assez nombreux : Discord, mails, rencontres régulières, conférences... il n'était donc pas nécessaire de surcharger ces canaux pour au final ne pas les utiliser.

5.4 Hébergement de l'application web et de la base de données via *O2switch*

Abordons maintenant le dernier outil utilisé, l'hébergeur *O2switch*.

Il possède une interface très simple permettant de multiples fonctionnalités, que ce soit pour configurer un serveur STMP associé à *Quickmatch* ou bien simplement générer des certificats de chiffrement HTTPS.

Nous avons donc pu simplement créer deux sites internet : un officiel et un pour les tests. Cependant, l'utilisation du site de test a clairement manquée au projet. Malgré son implémentation tardive, nous aurions nous en servir beaucoup plus afin de respecter la méthodologie citée en début de section 5 mais aussi débugger plus facilement sans nuire aux potentiels utilisateurs de *Quickmatch*.

Imaginons que *Quickmatch* ait eu un succès hors norme, nous aurions vite coulé en ne passant pas du tout par `test.quickmatch.fr` !

À cela s'est ajouté un support¹² très actif et nous ayant apporté une aide précieuse notamment pour la mise en place de la base de données. Nous avons eu divers problèmes à la création de l'application Javascript, mais ils ont été très vite résolus.

6 Difficultés observées et fonctionnalités non réalisées

Cette section vise à montrer quels problèmes nous avons pu rencontrer au cours du développement, leur résolution ou non ainsi que des ajouts souhaités ou pensés mais non effectifs.

6.1 Problèmes rencontrés

Voici la liste des difficultés trouvées :

- Authentification via *CAS* avortée pour laisser place à *Google*. Il était très complexe voire impossible d'implémenter ce type d'authentification sans héberger l'application web ainsi que sa base sur un serveur de Bordeaux-INP particulièrement.
- Temps de chargement de la page "Match" devenue long.
- Le comportement du bouton retour du téléphone sur Android n'a pas pu être implémenté comme voulu à cause d'un bug. Il est préférable d'utiliser celui de la barre d'outils en haut à gauche de l'application.

6.2 Ajouts souhaités

Voici la liste des ajouts absents sur *Quickmatch* :

- La réinitialisation de mot de passe des comptes *Quickmatch* en cas d'oubli par l'utilisateur.
- Validation de compte créée via l'application Androïd.
- Invitation à un match par mail, SMS ou même notification *push* (sur mobile).
- La mise en place d'une liste d'attente des joueurs invités dans un match au cas où il y ait déjà suffisamment de joueurs. Sur mobile, le nombre de joueurs d'un match ne limite pas l'acceptation d'invitations.
- La sécurisation de l'API qui est pour le moment accessible par tous, faire en sorte que seules les applications web et mobile aient accès aux données renvoyées par l'API.
- Le calendrier sur l'application Android n'est pas encore implémenté.
- La possibilité de changer de langue (anglais ou français) sur les deux applications n'a pas été implémenté.

7 Conclusion

Après avoir travaillé près de 6 mois sur ce projet, nous en avons retenu beaucoup de choses. Malgré les difficultés rencontrées, nous avons réussi à produire un livrable final fonctionnel et multiplateforme.

Nous avons retenu beaucoup de points positifs : il s'agit de notre premier vrai projet en conditions réelles : demandes du client, rencontres hebdomadaires et organisation minutieuse de l'équipe.

¹²Par mails

Malgré ces nouveautés, ce projet nous a apporté beaucoup d'expérience notamment dans les outils utilisés mais aussi concernant la gestion de projet plus globalement : ne pas être trop ambitieux et mieux gérer l'exécution des tâches au fil du projet.

Ce qui est à retenir serait la bonne communication avec les clients, mais aussi la diversité des domaines abordés (les technologies, les outils...).