

TPI – Shoot4Stats

« SHOOT4STATS » APPLICATION WEB DE GESTION
DE TIR À L'ARC

Contents

CONTENTS	1
1 ANALYSE PRÉLIMINAIRE	3
1.1 INTRODUCTION.....	3
1.2 OBJECTIFS.....	3
1.3 CONTEXTE	3
1.3.1 STRUCTURE D'UN <i>SHOOT</i>	3
1.3.2 REMPLISSAGE D'UN <i>SHOOT</i>	3
2 ANALYSE	4
2.1 CAHIER DES CHARGES DÉTAILLÉ	4
2.1.1 PRÉREQUIS	4
2.1.2 DESCRIPTIF DU PROJET	4
2.1.3 LOGICIELS ET RESSOURCES À DISPOSITION	4
2.2 STRATÉGIE DE TEST.....	4
2.2.1 STRATÉGIE GLOBALE	4
2.2.2 PROCÉDURE	5
2.3 ETUDE DE FAISABILITÉ	5
2.3.1 RISQUES TECHNIQUES	5
2.3.2 RISQUES SUR LE PLANNING	5
2.4 PLANIFICATION	5
3 DOSSIER DE CONCEPTION	6
3.1 PROCESSUS POUR ARCHERS	6
3.2 BASE DE DONNÉES.....	6
3.2.1 MLD	6
3.2.2 DICTIONNAIRE DE DONNÉES	6
3.3 API	8
3.4 OUTILS ET RESSOURCES	8
3.4.1 VUE JS	8
3.4.2 VUEX	8
3.4.3 PASSPORTJS	8
3.5 HISTORIQUE.....	8
4 DOSSIER DE RÉALISATION	9
4.1 MISE EN PLACE DU PROJET.....	9
4.2 ORGANISATION DES TÂCHES POUR LA RÉALISATION	9
4.3 MENU ET THÈME	9
4.3.1 MENU DYNAMIQUE	9
4.3.2 THÈME	9
4.4 PARTIE ARCHERS.....	10
4.4.1 MAQUETTES	10
4.4.2 VUE « HOME »	10
4.4.3 VUE « DASHBOARD »	11
4.4.4 VUE « CREATE SHOOT »	12
4.4.5 VUE « EDIT SHOOT »	13
4.5 PARTIE ADMINISTRATEURS	15
4.6 PARTIE ANONYMES	15
4.7 DESCRIPTION DES TESTS EFFECTUÉS.....	15
4.8 ERREURS RESTANTES	16
4.9 DOSSIER D'ARCHIVAGE.....	16

5	MISE EN SERVICE	17
5.1	RAPPORT DE MISE EN SERVICE	17
5.2	LISTE DES DOCUMENTS FOURNIS	17
6	CONCLUSIONS	17
7	ANNEXES	18
7.1	SOURCES – BIBLIOGRAPHIE	18
7.2	JOURNAL DE TRAVAIL	18
7.3	MANUEL D'INSTALLATION	18
7.4	MANUEL D'UTILISATION	18
7.5	ARCHIVES DU PROJET	18

1 Analyse préliminaire

1.1 Introduction

Mon travail de fin d'apprentissage a pour but de rentrer un Shoot (tir à l'arc) dans la DB afin d'avoir des statistiques rapidement sur le Shoot.

Ce travail permet de fortifier mes connaissances en développement WEB et d'apprendre et voir comment se déroule le développement d'une application WEB de manière professionnelle.

Le choix de cette application a été fait en corrélation avec le chef de projet à ma demande, pour répondre à des besoins personnels et peut-être même aux besoins des archers en règle générale.

1.2 Objectifs

Développer une interface WEB « Frontend » et mobile first puis adapter le « BackEnd » si besoin afin d'effectuer des tâches pour un utilisateur avec un rôle propre au sein de l'application.

1.3 Contexte

1.3.1 Structure d'un Shoot

Un Shoot est défini comme suit :

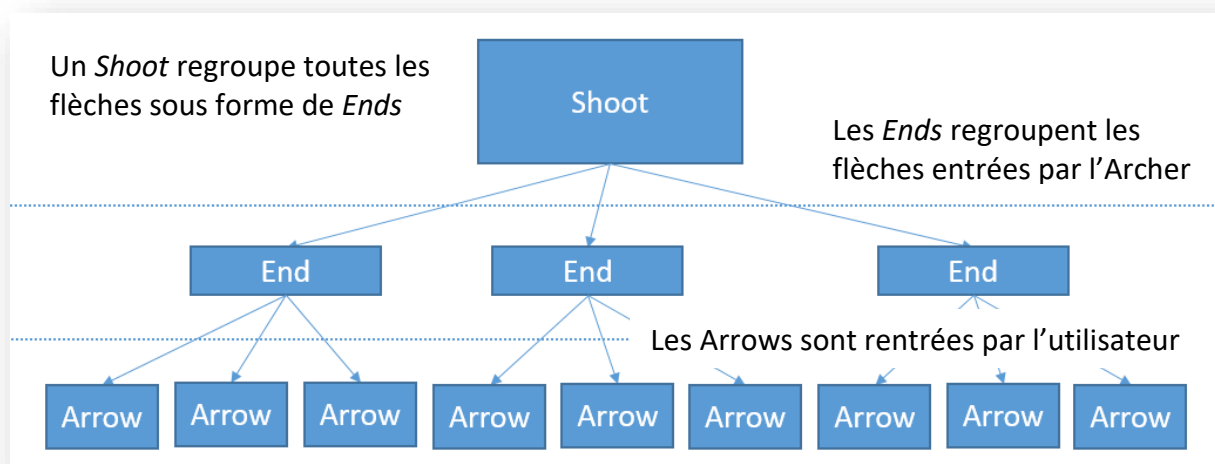


Figure 1 Schéma d'un Shoot

1.3.2 Remplissage d'un Shoot

La séquence de tir au tir à l'arc se passe comme suit ;

- L'on tire n flèches (*Arrows*) 3 pour les compétitions **Indoor** et 6 pour les compétitions **Outdoor**
- Ces n flèches se « regroupent » et forment une volée « *End* »
- Et après n volées (20 pour **Indoor**, 12 pour **Outdoor**) Le Shoot se termine, sachant que généralement l'on marque un point d'arrêt à la moitié du Shoot afin de définir deux séries.

2 Analyse

2.1 Cahier des charges détaillé

2.1.1 Prérequis

Une partie du travail a déjà été réalisé en collaboration avec le chef de projet, soit :

- Modéliser, implémenter et documenter la base de données
- Mettre en place un environnement de développement
 - IDE (en l'occurrence Visual Studio Code)
 - Gestion de source (GIT)
- Développer la partie « BackEnd » sous la forme d'une API
 - Requêtes principales préalablement développée
 - Authentification externe via Facebook à l'aide de PassportJS

2.1.2 Descriptif du projet

- Les Archers « utilisateurs authentifiés » doivent pouvoir ajouter des Shoots complets et voir les détails de ceux-ci. Reprendre les Shoots s'ils n'ont pas été terminés.
- Aux « administrateurs » (définis au sein de la DB directement) doivent pouvoir lister les Archers et avoir accès à leurs « Shoots ».
- Aux utilisateurs anonymes (non authentifiés) de voir les statistiques du dernier Shoot d'un utilisateur en ayant l'URL personnelle.

2.1.3 Logiciels et ressources à disposition

- Un PC :
 - OS : Windows 10 Etudiant
 - IDE : Visual Studio Code
 - Langage : JavaScript
 - Framework : NodeJS **v.7.4.0** (API)
 - Librairies : VueJS (Frontend) / Axios / JQuery / Framework CSS Materialize / ExpressJS (Backend) / ORM Sequelize / PassportJS gère l'authentification
- Services :
 - Authentification externe via Facebook Login

2.2 Stratégie de test

2.2.1 Stratégie globale

Comme le cahier des charges le spécifie la méthode de développement utilisée doit être itérative (AGILE), voici donc les 4 valeurs fondamentales de la méthode AGILE :

- **L'équipe** ("Personnes et interaction plutôt que processus et outils")
- **L'application** ("Logiciel fonctionnel plutôt que documentation complète")
- **La collaboration** ("Collaboration avec le client plutôt que négociation de contrat")
- **L'acceptation du changement** ("Réagir au changement plutôt que suivre un plan")

L'on remarque que l'accent est mis sur le client, le faire participer un maximum au développement de l'application afin qu'il se sentes concerné et que le produit fini lui convienne au maximum au lieu de se baser uniquement sur un cahier des charges fixe.

2.2.2 Procédure

Lors des gros ajouts de fonctionnalités je ferai tester à des personnes externes (archers, coach de tir) dans la limite du possible afin d'avoir un retour externe sur les tests (interface utilisateur et simplicité d'utilisation)

Puis de mon côté lors du développement j'effectuerai des tests tels que :

1. Explication du test
2. Résultats attendus
3. Résultats observés
4. Discussion des résultats / résolution des problèmes

Mes tests s'effectueront sur la validation des formulaires, la restriction relative aux pages de l'application, la récupération des informations sur l'API.

2.3 Etude de faisabilité

2.3.1 Risques techniques

La complexité technique réside dans la modularisation du code afin de permettre une bonne évolutivité du code pour permettre les améliorations / ajouts de fonctionnalités futures de fonctionner rapidement grâce aux modules présents.

Ayant déjà fait le BackEnd en collaboration avec le chef de projet, ce concept de modularité a été appris à ce moment-là, je vais donc faire en sorte de bien séparer mes Vues et que les données soient bien partagées par composants afin de toujours savoir où l'on en est.

2.3.2 Risques sur le planning

Je pense qu'au niveau du planning il ne va pas y avoir des gros problèmes tant que je ne m'éloigne pas de l'objectif principal (c'est-à-dire d'avoir une application fonctionnelle avant tout) et que je me concentre sur les points principaux.

Pour cela j'ai défini des priorités sur les tâches comme on peut le voir dans le Planning Initial, je ferai d'abord la partie « Archers », puis Administrateurs et enfin les utilisateurs anonymes.

2.4 Planification

Voir en annexe « Planning Initial ».

3 Dossier de conception

3.1 Processus pour Archers

Processus de création de Shoot et remplissage de celui-ci

3.2 Base de données

La base de données fait partie des prérequis du projet, c'est-à-dire qu'elle a été faite en collaboration avec le chef de projet soit Stefano Nepa.

3.2.1 MLD

Le MLD a été généré par MySQL WorkBench d'après le diagramme que j'ai fait.

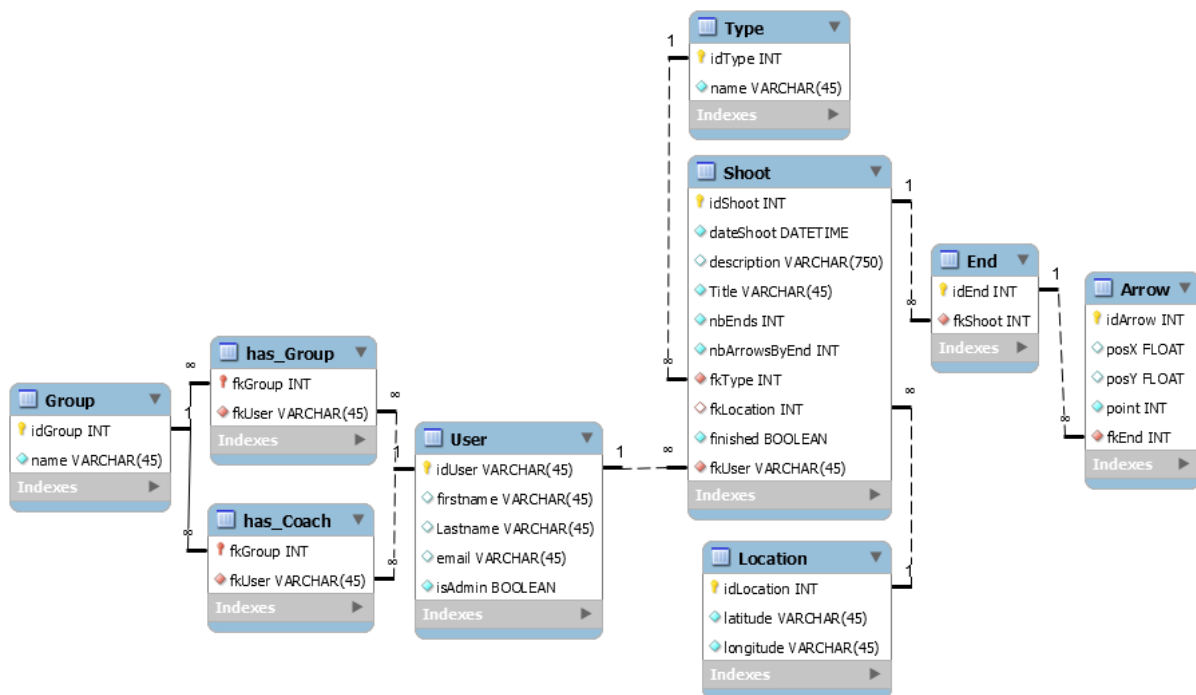


Figure 2 Modèle Logique des données

3.2.2 Dictionnaire de données

Voici la description de la base de données :

- **Shoot**
Représente un Shoot.
 - **idShoot**
Id du Shoot, auto incrémentiel, permet de pouvoir créer des Shoots sans contraintes
 - **dateShoot, description, title**
Propriétés du Shoot, définition du Shoot
DateShoot : valeur par défaut CURRENT_TIMESTAMP
Description et title set par l'utilisateur
 - **nbEnds, nbArrowsByEnd**
L'API chargera automatiquement suivant le type du Shoot, sauf si le type est « Training » auquel cas ce sera à l'utilisateur de choisir ses chiffres
 - **fkType**
Clé étrangère pour la définition du Type du Shoot

- **fkLocation**
Clé étrangère qui peut être nulle, définit la *location* du Shoot si l'utilisateur le veut
- **fkUser**
Clé étrangère définissant l'Archer qui a tiré le Shoot
- **finished**
Propriété qui est définie à « true » par l'API quand le nombre de *Ends* associée avec le Shoot est le même que le nombre de *Ends* mise dans **nbEnds**
- **Type**
Représente le Type du Shoot
 - **idType**
Identifiant du type
 - **name**
Le nom du Type, dans ce cas il y en a 3
- **Location**
Permet d'ajouter une localisation au *Shoot*
 - **idLocation**
Identifiant unique de la localisation (peut-être redondance des données si des utilisateurs tirent au même endroit)
 - **latitude, longitude**
Description de la localisation, dans le cas de mon application je pensais utiliser les services de Google Maps, si le temps me manque la ville et la rue Seront dans ces 2 champs en clair → à corriger si cela se produit
- **End**
Contient les flèches tirées, appartient au *Shoot*
 - **idEnd**
Permet l'identification au sein des Arrows, agit uniquement comme une table intermédiaire « Conteneur » Ajoute de la logique à la DB
 - **fkShoot**
Décrit l'appartenance à un *Shoot*
- **Arrow**
Décrit la flèche et sa valeur tirée par l'archer
 - **idArrow**
 - **posX, posY, point**
Décrit la position de la flèche sur la cible (hors cadre du TPI mais possibilités d'amélioration), et le point qui est attribué à la flèche (attribut le plus important de la DB)
 - **fkEnd**
Détermine sur quel End la flèche va venir se greffer afin de respecter ce principe de séries et de chronologie évoqué plus haut.
- **User**
Décrit un utilisateur qu'il soit admin ou archer
 - **idUser**
Id unique attribué par Facebook¹ pour l'user utilisant cette application, je m'en sers afin d'identifier les utilisateurs, lors de l'entrée dans la DB l'ID est « salté » avec *facebook_*

¹ Pour plus de détails : <https://developers.facebook.com/docs/graph-api/reference/v2.5/user>

- **firstname, lastname, email**
Informations relatives à l'utilisateur, récupérées via Facebook
- **isAdmin**
Propriété booléenne afin de mettre un utilisateur en « Admin »
- **has_Group, has_Coach, Group**
Ces tables ne seront pas utilisées dans le cadre de ce TPI mais seront utilisées ensuite afin de définir des rôles bien plus précis aux utilisateurs

3.3 API

Voir Annexes

<https://github.com/mbonjour/Shoot4Stats/blob/master/Documentation/Annexes/API/apiDocumentation.md>

et

<https://github.com/mbonjour/Shoot4Stats/blob/master/Documentation/Annexes/API/authDocumentation.md>

3.4 Outils et ressources

3.4.1 Vue JS

<https://vuejs.org/v2/guide/>

Expliquer ici les principes de base de VueJS

3.4.2 Vuex

<https://vuex.vuejs.org/en/>

Expliquer ici les principes de base de Vuex

3.4.3 PassportJS

<http://passportjs.org/docs>

Expliquer ici les principes de base de PassportJS

3.5 Historique

Je listerai au mieux les changements effectués, sachant que ces sera une programmation suivant la norme « AGILE » qui consiste à discuter en permanence avec le client et à développer par étapes l'application. Je vais donc répertorier ici les discussions avec le chef de projet / client.

Si la conception du projet a dû être modifiée plusieurs fois, ou de manière significative, expliquez ces changements et leurs causes.

***Attention:** Pour faciliter la maintenance, à la fin du projet, le dossier de conception doit correspondre à ce qui a été effectivement réalisé !*

4 Dossier de Réalisation

4.1 Mise en place du projet

Explications : Vue init – Vue CLI

Choix du boilerplate, « webpack » → Pourquoi Webpack et qu'est-ce que c'est en bref

4.2 Organisation des tâches pour la réalisation

J'ai organisé en accord avec le client les tâches de la manière suivante :

- Mettre en place les Vues pour les archers authentifiés
- FeedBack avec les testeurs et le client principal
- Mettre en place la partie des administrateurs
- Effectuer les tests de sécurité
- Mettre en place la partie pour les utilisateurs anonymes

4.3 Menu et thème

4.3.1 Menu dynamique

Le menu est dynamique selon qui le regarde afin d'éviter de se retrouver sur des pages sans y avoir accès

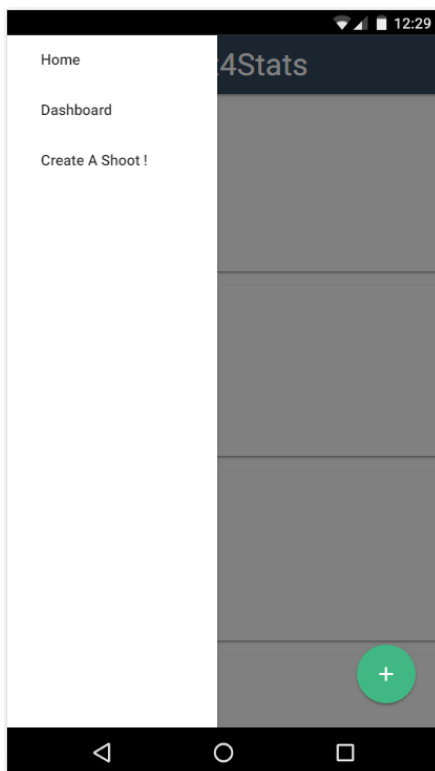


Figure 4 Menu en "Archers"

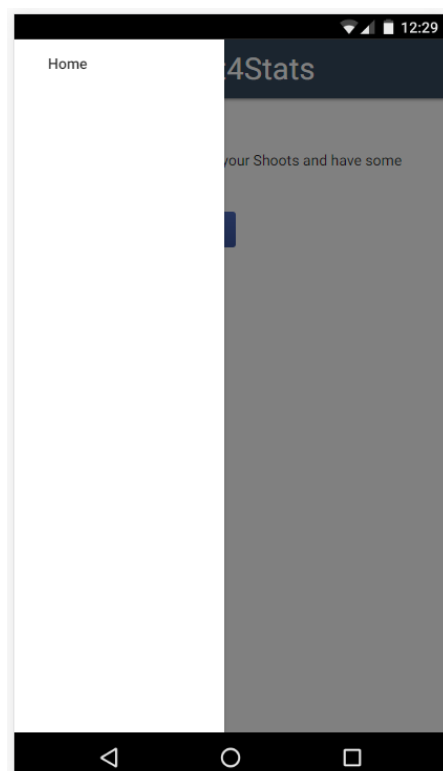


Figure 3 Menu en Anonyme

4.3.2 Thème

4.3.2.1 Explication

Le choix des couleurs s'est fait en collaboration avec le chef de projet et nous avons donc décider de partir sur une palette de couleur qui est celle du symbole de VueJS.

Ce qui donne un vert clair (#41B883) et un bleu marine foncé (#35495E)

4.3.2.2 Exemple

Voici les couleurs au sein d'un bouton p.ex. :



Figure 5 Exemple de bouton stylisé

4.4 Partie Archers

4.4.1 Maquettes

Les maquettes ont été faites avant de débiter les Vues et elles sont en Annexes, elles m'ont bien aidées à la représentation des composants que j'allais devoir mettre en place.

4.4.2 Vue « Home »

4.4.2.1 Description

Cette Vue sera le point d'entrée de l'application WEB elle permettra à tout un chacun de s'authentifier (via Facebook, l'on récupère ainsi les emails, le nom et le prénom avec plus de facilité et plus d'ergonomie)

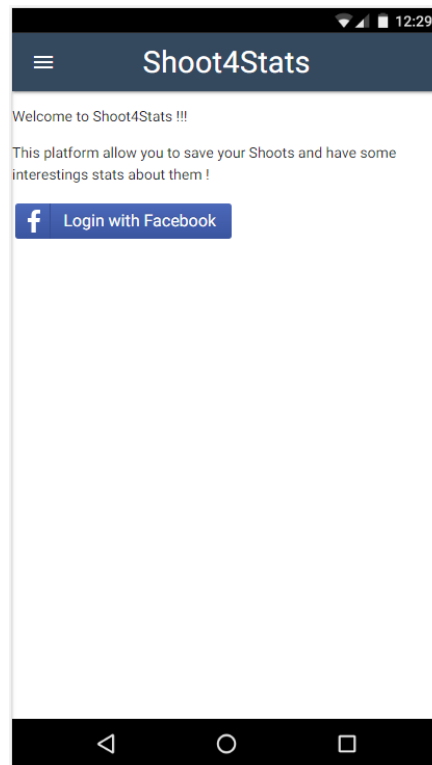


Figure 6 Vue Home

L'authentification est gérée en BackEnd et a été développée en collaboration avec le chef de projet, elle se base sur un middleware pour Node.JS qui se nomme **PaspportJS**. Le système d'authentification est expliqué dans la Doc de l'API (cf. 3.3)

4.4.2.2 Logique

La vue « Home » n'a pas de logique, c'est une page HTML conventionnelle.

4.4.3 Vue « Dashboard »

4.4.3.1 Description

La vue Dashboard a pu être discutée avec le client et le sera encore à l'heure où j'écris ceci. Le client m'a fait parvenir ses préférences (qui ne sont pas si différentes du CDC) du fait d'avoir un historique de Shoots et d'en avoir un bref aperçu rapidement et simplement. Puis avoir à l'aide d'un clic des détails précis concernant le Shoot sélectionné.

4.4.3.2 Composant Dashboard

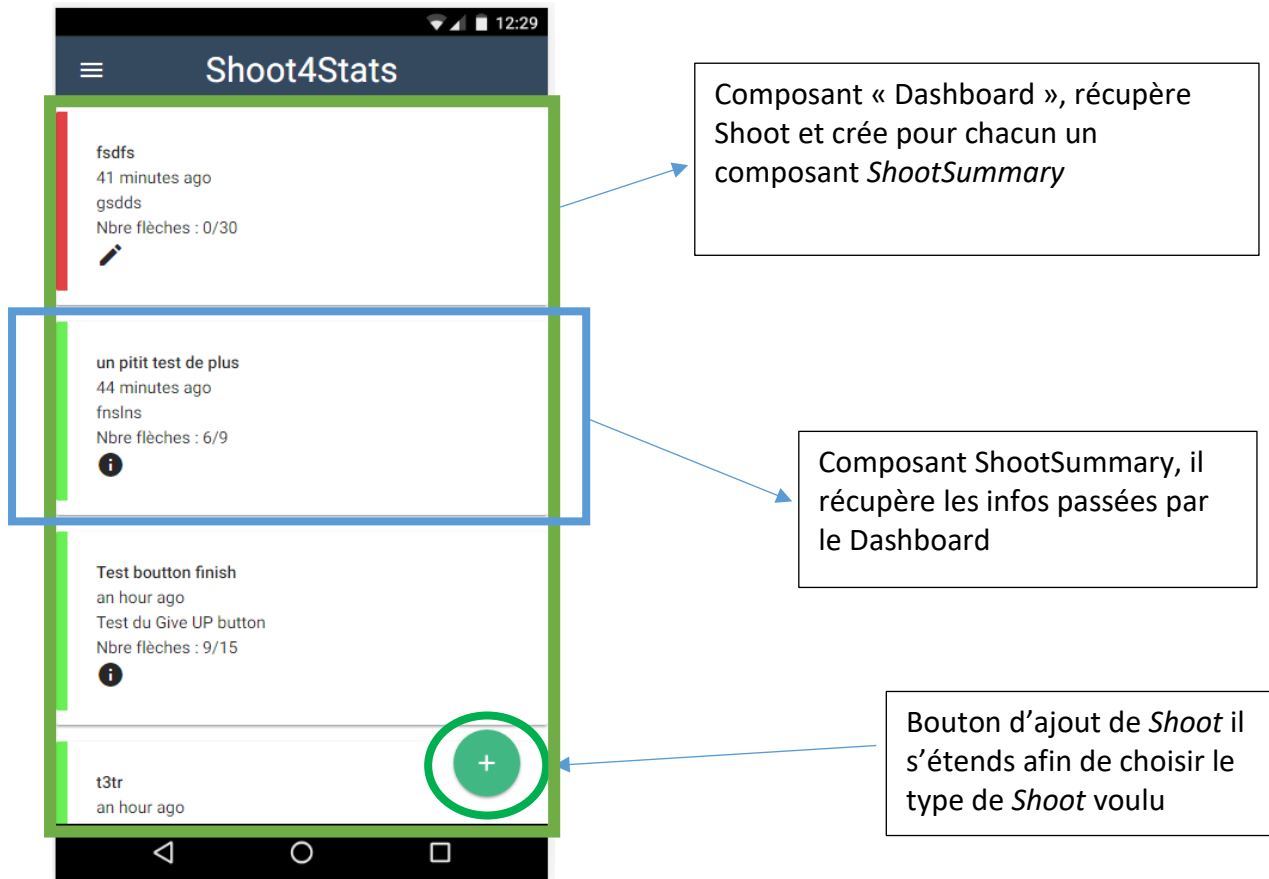


Figure 7 Vue Dashboard décomposée

4.4.3.3 Composant ShootSummary

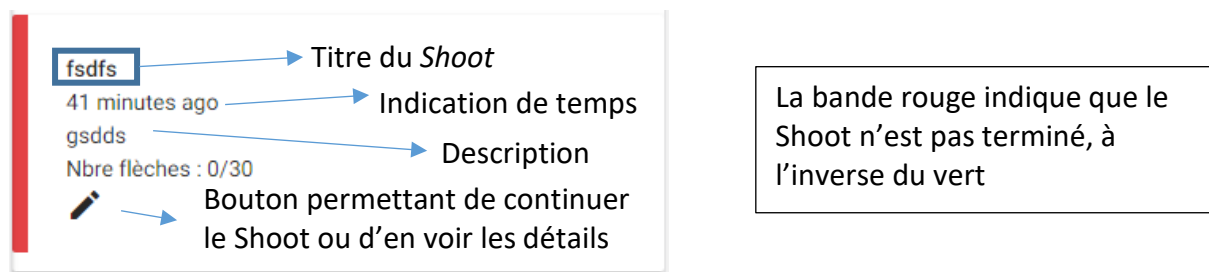


Figure 8 Composant ShootSummary

4.4.3.4 Composant ShootDetails

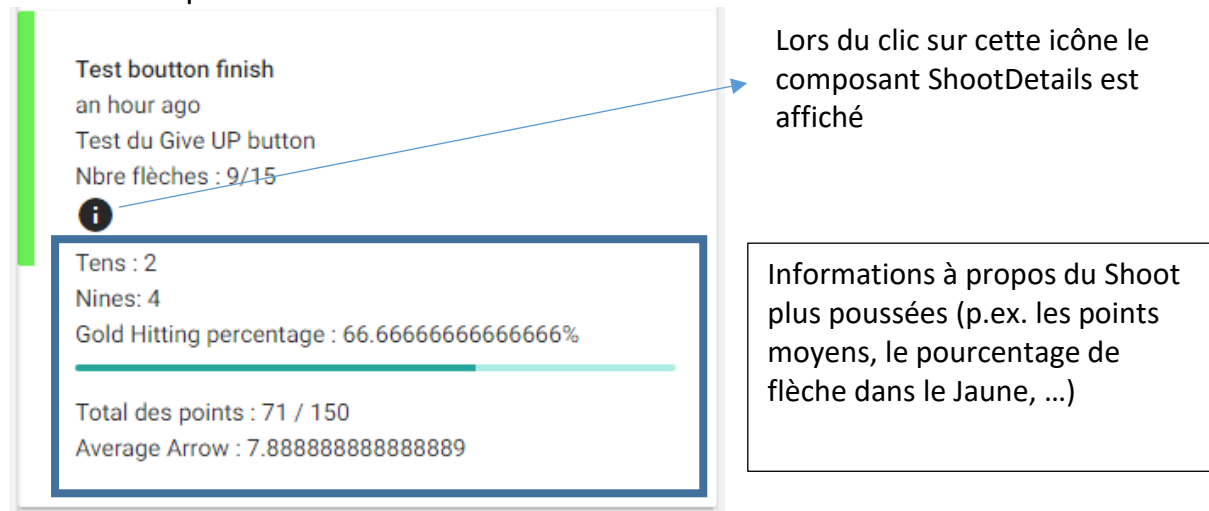


Figure 9 Composant ShootSummary et affichage ShootDetails

4.4.3.5 Logique – Interaction API Schémas

Ce que fait la Vue Dashboard c'est qu'elle va aller récupérer les shoots de l'utilisateur grâce à l'API (/api/shoots) puis elle va boucler en passant en propriétés un shoot à un composant partagé² *shootSummary*.

Le composant *shootSummary* appelle à son tour l'API si l'utilisateur désire voir les détails de son Shoot. Il va mettre ces détails dans un state de Vuex *currentShoot* il va afficher un composant partagé² *shootDetails* qui prend les informations du state Vuex et va afficher ses informations supplémentaires de manière simple.

4.4.4 Vue « Create Shoot »

4.4.4.1 Description

Cette vue va permettre à un utilisateur de débiter un Shoot. Sa route (routage des pages en frontend) prend un paramètre, car le bouton « + » du dashboard peut permettre à l'utilisateur de définir quel type de Shoot il veut débiter, ceci afin de permettre une facilité d'utilisation de l'application.

² Les composants partagés sont des composants qui seront utilisés à plusieurs reprises, ils seront donc mis au même endroit afin d'en faciliter l'accès.

4.4.4.2 ScreenShots

The screenshot shows a mobile application interface for 'Shoot4Stats'. The form contains the following fields: 'Shoot Title', 'Description', 'City', 'Street', 'Type' (a dropdown menu currently showing 'training'), 'nb_Ends', and 'nb_ArrowsByEnd'. At the bottom is a green 'SUBMIT' button with a right-pointing arrow. Three blue arrows point from text boxes to specific parts of the form: one to the main form area, one to the 'Type' dropdown, and one to the 'SUBMIT' button.

Shoot4Stats

Shoot Title

Description

City

Street

Type
training

nb_Ends nb_ArrowsByEnd

SUBMIT ➤

Formulaire simple grâce à Materialize

Partie du formulaire dynamique, si l'on change le « type » dans la sélection mets à jour les deux champs en dessous

Le bouton déclenche la validation et envoie la requête avec l'objet formaté comme voulu pour l'API

Figure 10 Vue CreateShoot

4.4.4.3 Logique

Cette Vue est un simple formulaire, je désactive l'envoi des données et « toast » (fenêtre modale) si la validation n'est pas ok, donc tant que tous les champs ne sont pas remplis ou que le format n'est pas respecté les données ne sont pas envoyées. Il ne fait par contre pas de POST comme un formulaire HTML, je récupère les données dans les balises « data » de mon composant et fait un POST uniquement lorsque j'ai vérifié les données.

4.4.4.4 Interaction API - Schémas

4.4.5 Vue « Edit Shoot »

4.4.5.1 Description

Cette Vue n'est pas accessible par le menu car elle a besoin d'avoir l'ID d'un Shoot pour fonctionner, de plus cette vue ne sert qu'à continuer un Shoot depuis le Dashboard ou commencer un Shoot lorsqu'on vient de le créer.

Cette Vue permet d'ajouter des *Arrows* et *Ends* au *Shoot* courant.

4.4.5.2 ScreenShots - Schémas

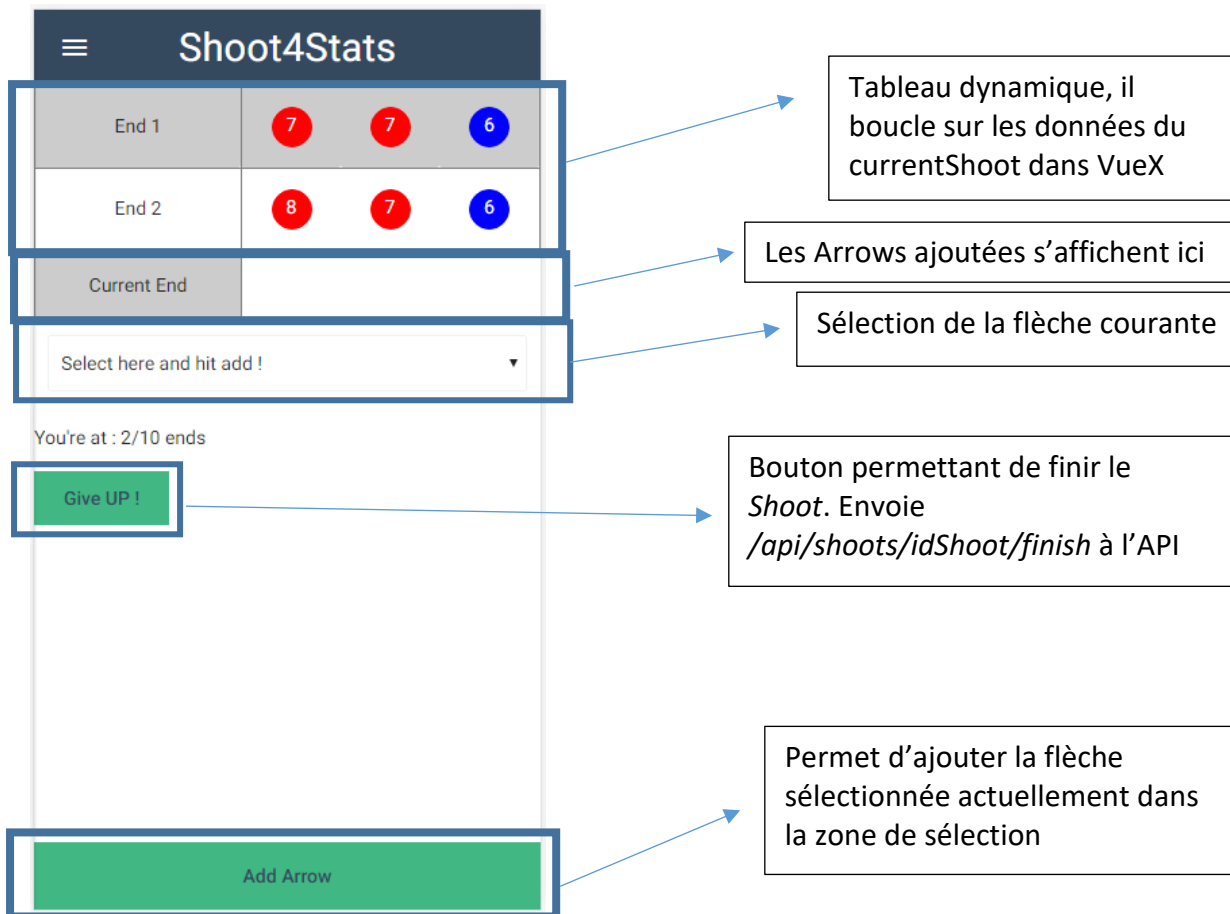


Figure 11 Vue editShoot

4.4.5.3 Composant arrowItem

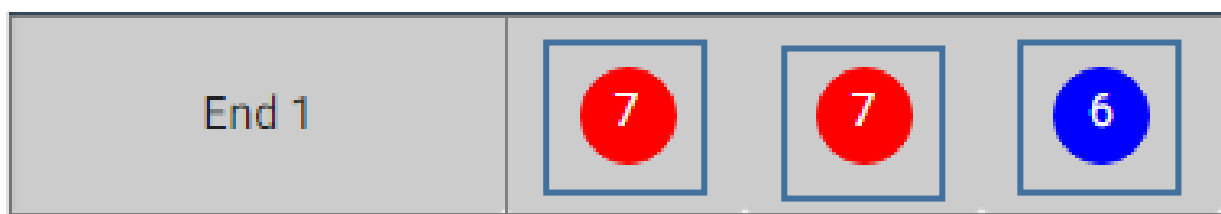


Figure 12 Composant arrowItem

Le composant `arrowItem` prend comme propriétés (paramètre) un nombre qui représente le point de la flèche (0 – 11).

Voici l'image d'un blason de tir à l'arc officiel :

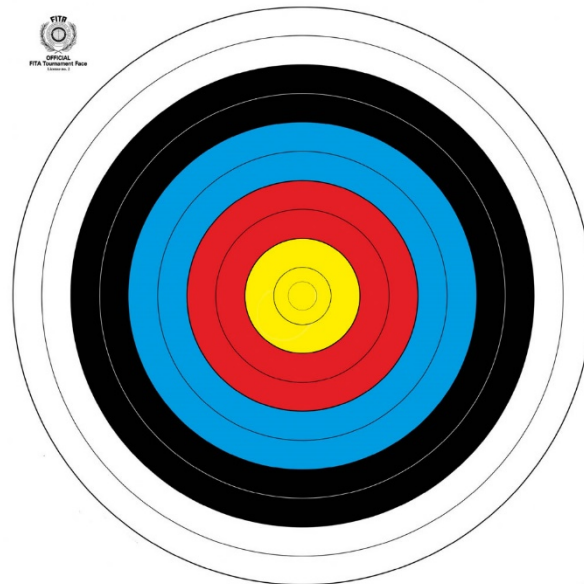


Figure 13 FITA Official Target face

Comme vous pouvez le voir, il y a une paire de cercle pour chaque couleur, et chaque cercle représente un point. Le dernier (le plus au centre) cependant représente un « Avantage » on appelle ça un « X » mais il vaut tout de même 10 points.

Lorsqu’une flèche ne va pas sur la cible l’on appelle ça un « Missed » (ou manqué) et cela vaut 0 point.

Alors pour en revenir à notre composant, il prend la valeur du point et affiche une pastille de couleur afin que l’archer puisse facilement voir dans la globalité ses points. Le 0 est interprété par un « M » et le 11 par un « X » mais compte comme 10 points au sein de l’API

4.4.5.4 Problèmes rencontrés

En faisant le composant je me suis rapidement aperçu que lors de la création de l’API je n’avais pas pris en compte les « M » et les « X » car le champ point de la DB n’accepte que des entiers.

J’ai alors modifier l’API pour que lors des calculs il transforme le « X » en 10 pour ne pas fausser les calculs.

4.4.5.5 Aide reçue

J’ai reçu de l’aide et discuter avec mon chef de projet pour la mise en place des pastilles pour plus d’ergonomie. Il m’a aussi demandé d’implémenter une fonction de « toaster » afin d’afficher les erreurs à l’utilisateur rapidement et simplement.

Il m’a aussi aidé à m’imaginer la structure des données pour ces composants, c’est grâce à lui que je passe par Vuex pour faire mes requêtes vers l’API qui permet alors d’avoir une meilleure modularisation du code par la suite.

4.5 Partie Administrateurs

4.6 Partie Anonymes

4.7 Description des tests effectués

Pour chaque partie testée de votre projet, il faut décrire:

- *les conditions exactes de chaque test*
- *les preuves de test (papier ou fichier)*
- *tests sans preuve: fournir au moins une description*

4.8 Erreurs restantes

S'il reste encore des erreurs:

- *Description détaillée*
- *Conséquences sur l'utilisation du produit*
- *Actions envisagées ou possibles*

4.9 Dossier d'archivage

Décrire de manière détaillée les 2 archives du projet (CD-ROM, disque zip ou jazz, bandes magnétiques, ...)

Attention: les documents de réalisation doivent permettre à une autre personne de maintenir et modifier votre projet sans votre aide !

5 Mise en service

5.1 Rapport de mise en service

Fournir une description:

- *de l'installation du projet chez le client (pour un site web: publication chez un provider)*
- *des test officiels effectués chez le client et/ou par le client.*
- *des erreurs répertoriées*
 - *description détaillée*
 - *conséquences pour le client*
 - *actions envisagées.*

5.2 Liste des documents fournis

Lister les documents fournis au client avec votre produit, en indiquant les numéros de versions

- *le rapport de projet*
- *le manuel d'Installation (en annexe)*
- *le manuel d'Utilisation avec des exemples graphiques (en annexe)*
- *autres...*

6 Conclusions

Développez en tous cas les points suivants:

- *Objectifs atteints / non-atteints*
- *Points positifs / négatifs*
- *Difficultés particulières*
- *Suites possibles pour le projet (évolutions & améliorations)*

7 Annexes

7.1 Sources – Bibliographie

Liste des livres utilisés (Titre, auteur, date), des sites Internet (URL) consultés, des articles (Revue, date, titre, auteur)... Et de toutes les aides externes (noms)

7.2 Journal de travail

7.3 Manuel d'Installation

7.4 Manuel d'Utilisation

7.5 Archives du projet


CD, ... dans une fourre en plastique

Annexes

Planning TPI - Mickael Bonjour

EPFL, le 27.03.2017

Temps total : 101 heures

 = 1 heure

[illegible]

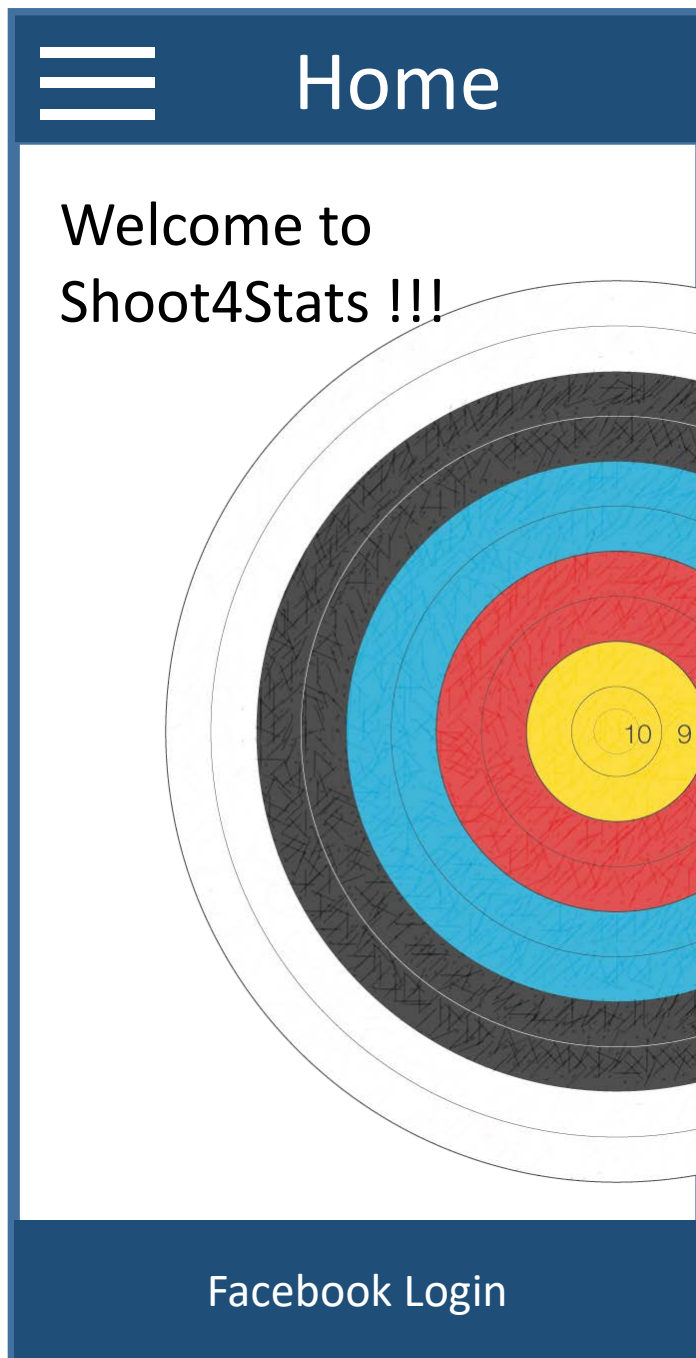
Maquettes Archers

Attention ce sont juste des squelettes pour voir plus ou moins le fonctionnement de mon appli côté Archers, Il faut imaginer que les composants du Framework CSS choisi vont remplacés mes éléments

Maquettes Archers

Home (point d'entrée de l'appli) ici en anonyme

localhost:3000/#/



Le bouton “Facebook Login” va taper dans l’API sur `/api/login/facebook`




Maquettes Archers

localhost:3000/#/dashboard

Mettre
firstname du
User ici pour
montrer qu'il
est authentifié

Du plus récent au plus vieux (charge tous les
Shoots de l'utilisateur)

 Dashboard Mickael

Titre_Shoot4

Type of training

At Lausanne, VD

Flèches : 60/60

A day ago

Titre_Shoot3

Type of training

At Lausanne, VD

Flèches : 57/60

2 days ago

Titre_Shoot2

Type of training

At Lausanne, VD

Flèches : 60/60

3 days ago

Titre_Shoot1

Type of training

At Lausanne, VD

Flèches : 60/60

4 days ago

Go Shooting !

Si l'on clique
sur un Shoot
non terminé,
nous renvoie
sur la page

Ce bouton amène sur la creation d'un
nouveau Shoot

Maquettes Archers

localhost:3000/#/createShoot

Title_NewShoot

Description

Add a Description here

Location

City

Street

Type

Training

Indoor

Outdoor

Nb_Ends

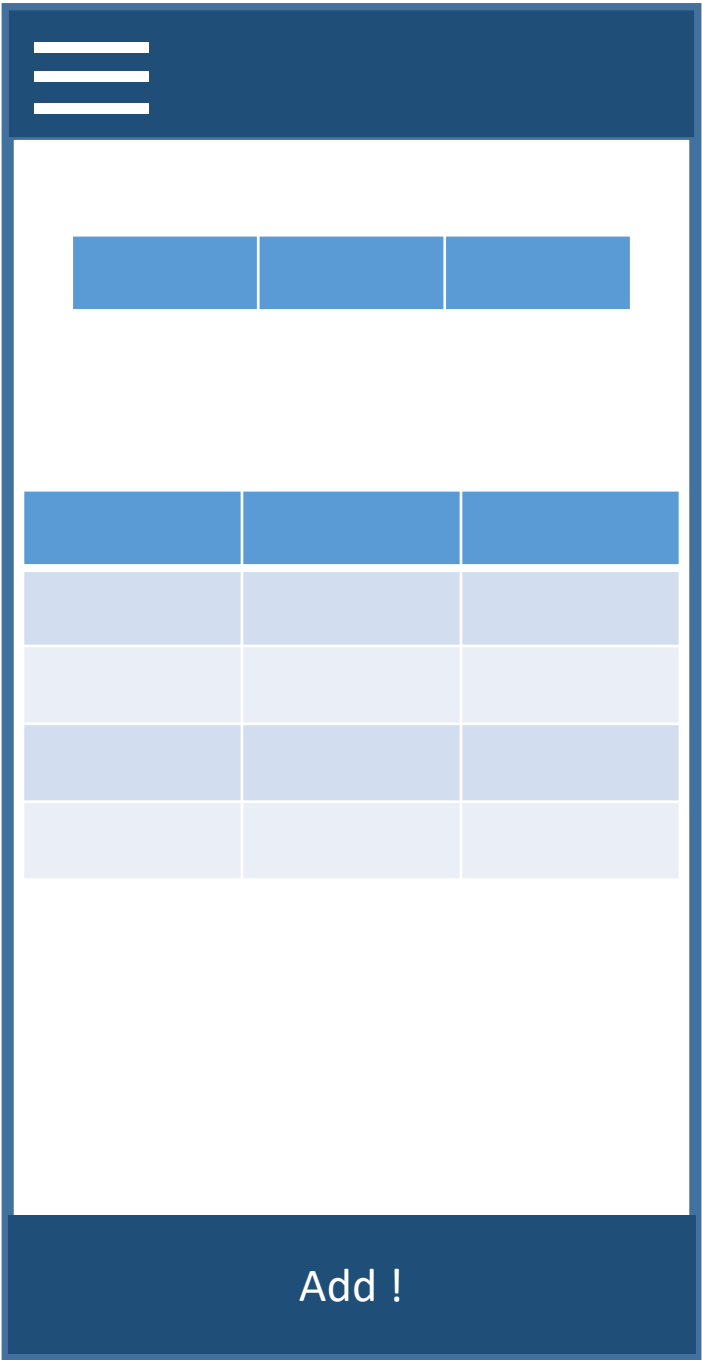
Nb_ArrowsByEnd

Begin !

Les cases Nb_Ends et Nb_ArrowsByEnd seront préremplies

Maquettes Archers

localhost:3000/#/editShoot



Ajout des flèches dans les carrés en haut puis après pression bouton Add ! → POST /api/ends

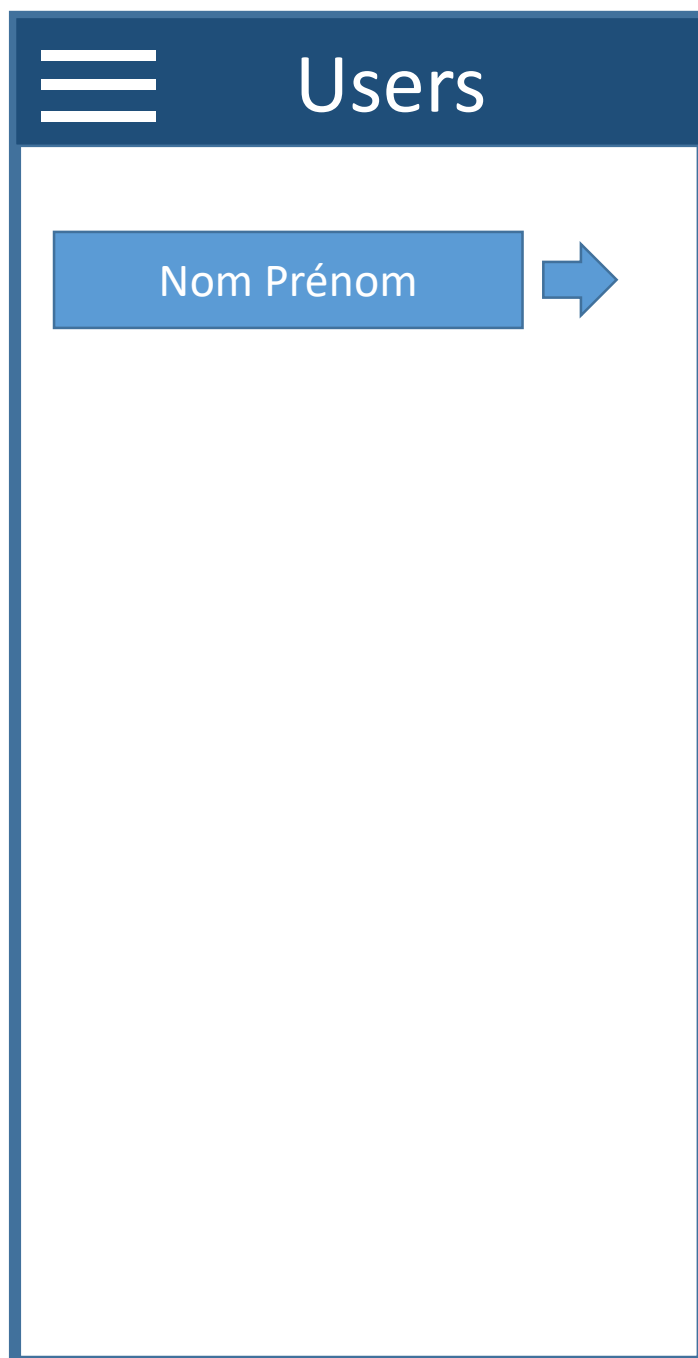
Mets la volée dans le tableau

Fait un GET /api/shoots/id pour avoir les volées déjà faite si le Shoot est déjà entamé

Attention ce sont juste des squelettes pour voir plus ou moins le fonctionnement de mon appli côté Archers, Il faut imaginer que les composants du Framework CSS choisi vont remplacer mes éléments

Page spéciale pour les Admins (ajoutée dans le menu si admin)

localhost:3000/#/admins



La flèche permettra de voir le Dashboard de l'utilisateur en question et ainsi administrer ses *Shoots*