

ENAC-IT2

EPFL | Bonjour Mickael

TPI – Shoot4Stats

« Shoot4Stats » Application WEB de gestion de tir à l’arc

# Contents

[Contents 1](#_Toc480137549)

[1 Analyse préliminaire 3](#_Toc480137550)

[*1.1* Introduction 3](#_Toc480137551)

[*1.2* Objectifs 3](#_Toc480137552)

[1.3 Contexte 3](#_Toc480137553)

[1.3.1 Structure d’un *Shoot* 3](#_Toc480137554)

[1.3.2 Remplissage d’un *Shoot* 3](#_Toc480137555)

[2 Analyse 4](#_Toc480137556)

[*2.1* Cahier des charges détaillé 4](#_Toc480137557)

[2.1.1 Prérequis 4](#_Toc480137558)

[2.1.2 Descriptif du projet 4](#_Toc480137559)

[2.1.3 Logiciels et ressources à disposition 4](#_Toc480137560)

[*2.2* Stratégie de test 4](#_Toc480137561)

[2.2.1 Stratégie globale 4](#_Toc480137562)

[2.2.2 Procédure 5](#_Toc480137563)

[*2.3* Etude de faisabilité 5](#_Toc480137564)

[2.3.1 Risques techniques 5](#_Toc480137565)

[2.3.2 Risques sur le planning 5](#_Toc480137566)

[*2.4* Planification 5](#_Toc480137567)

[3 Dossier de conception 6](#_Toc480137568)

[3.1 Processus pour Archers 6](#_Toc480137569)

[3.2 Base de données 8](#_Toc480137570)

[3.2.1 MLD 8](#_Toc480137571)

[3.2.2 Dictionnaire de données 8](#_Toc480137572)

[3.3 API 10](#_Toc480137573)

[3.4 Outils et ressources 10](#_Toc480137574)

[3.4.1 Vue JS 10](#_Toc480137575)

[3.4.2 VueX 10](#_Toc480137576)

[3.4.3 PassportJS 10](#_Toc480137577)

[*3.5* Historique 10](#_Toc480137578)

[4 Dossier de Réalisation 11](#_Toc480137579)

[4.1 Mise en place du projet 11](#_Toc480137580)

[4.2 Organisation des tâches pour la réalisation 11](#_Toc480137581)

[4.3 Menu et thème 11](#_Toc480137582)

[4.3.1 Menu dynamique 11](#_Toc480137583)

[4.3.2 Barre de Navigation 12](#_Toc480137584)

[4.3.3 Thème 12](#_Toc480137585)

[4.3.4 Framework CSS 13](#_Toc480137586)

[4.4 Partie Archers 13](#_Toc480137587)

[4.4.1 Maquettes 13](#_Toc480137588)

[4.4.2 Vue « Home » 14](#_Toc480137589)

[4.4.3 Vue « Dashboard » 🡪 Home lorsque l’on est authentifié 14](#_Toc480137590)

[4.4.4 Vue « Create Shoot » 19](#_Toc480137591)

[4.4.5 Vue « Edit Shoot » 19](#_Toc480137592)

[4.5 Description des tests effectués 23](#_Toc480137593)

[4.5.1 Test sur editShoot 23](#_Toc480137594)

[4.5.2 Test sur CreateShoot 23](#_Toc480137595)

[*4.6* Erreurs restantes 23](#_Toc480137596)

[*4.7* Dossier d'archivage 23](#_Toc480137597)

[5 Mise en service 24](#_Toc480137598)

[*5.1* Rapport de mise en service 24](#_Toc480137599)

[*5.2* Liste des documents fournis 24](#_Toc480137600)

[6 Conclusions 24](#_Toc480137601)

[7 Annexes 25](#_Toc480137602)

[7.1 Sources – Bibliographie 25](#_Toc480137603)

[7.2 Journal de travail 25](#_Toc480137604)

[7.3 Manuel d'Installation 25](#_Toc480137605)

[7.4 Manuel d'Utilisation 25](#_Toc480137606)

[*7.5* Archives du projet 25](#_Toc480137607)

# Analyse préliminaire

Je vais ici décomposer le cahier des charges dans son ensemble et présenter les buts du projet

## Introduction

Mon travail de fin d’apprentissage a pour but de permettre aux archers, une fois authentifiés à l’aide Facebook, d’enregistre des données sur leurs Shoots (c’est-à-dire des *Ends* et des *Arrows* aussi). Elle permettra de visionner ces données par des informations graphiques ou chiffrées.

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.

## Objectifs

Développer une interface WEB « Frontend » et mobile first puis adapter le « BackEnd » si besoin afin d’effectuer des tâches pour un utilisateur authentifié ou non.

## Contexte

### Structure d’un *Shoot*

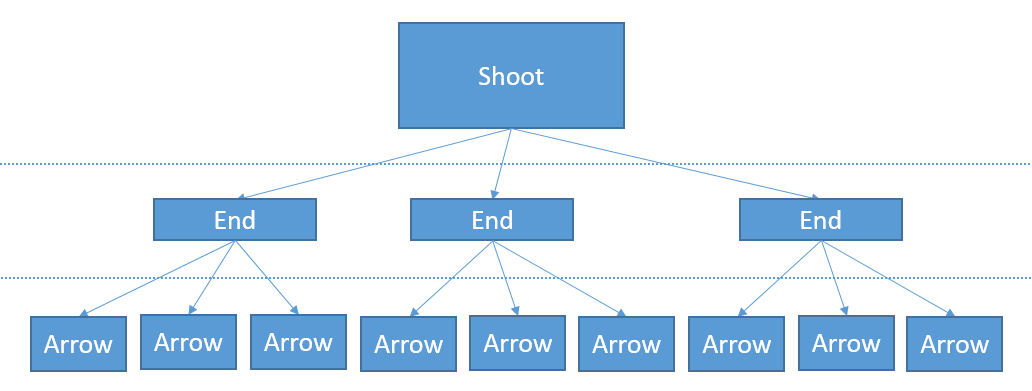
Un *Shoot* est défini comme suit :

Figure 1 Schéma d'un Shoot

Les *Ends* regroupent les flèches entrées par l’Archer

Les *Arrows* sont rentrées par l’utilisateur

Un *Shoot* regroupe toutes les flèches sous forme d’Ends

### Remplissage d’un *Shoot*

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

* L’on tire n flèches (*Arrows*) pour un **Training**. 3 *Arrows* 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.

# Analyse

J’analyse ici la difficulté du projet et explique le projet en globalité

## Cahier des charges détaillé

### 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

### 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 pouvoirs 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.

### 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 / PaspportJS gère l’authentification
* Services :
  + Authentification externe via Facebook Login

## Stratégie de test

### 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 sente concerné et que le produit fini lui convienne au maximum au lieu de se baser uniquement sur un cahier des charges fixe.

### 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.

## Etude de faisabilité

### 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.

### 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 définit 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.

## Planification

Voir en annexe « Planning Initial ».

# Dossier de conception

Je présente ici les moyens mis en œuvre et la conception du projet, cela comprend ce qui a été fait avant en collaboration avec le chef de projet.

## Processus pour Archers

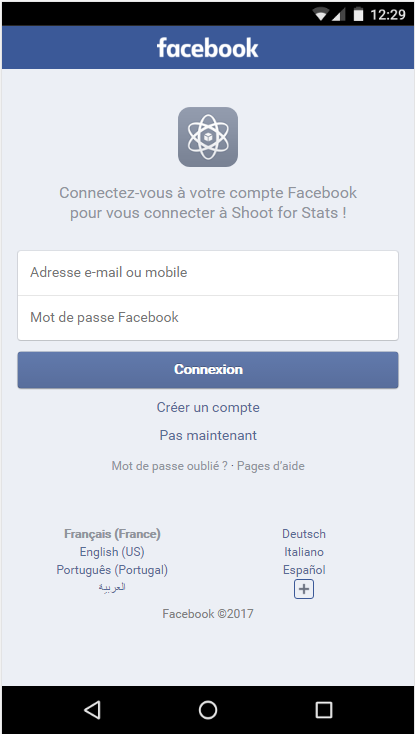
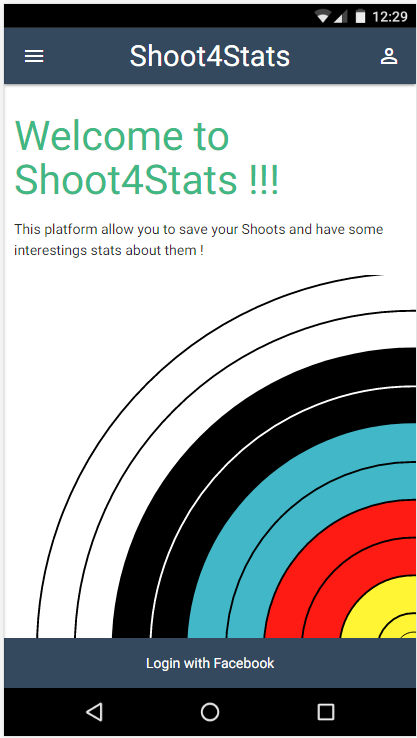
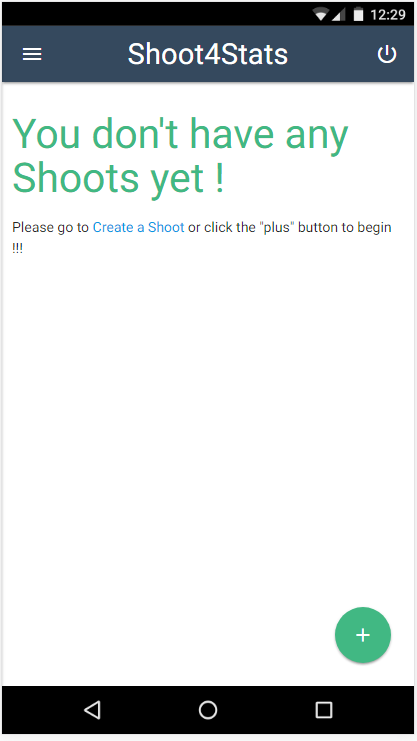
Je démontre par l’exemple le processus de création et de remplissage d’un *Shoot* type, cela s’apparente à un document d’aide à l’utilisation

Figure 2 Login via Facebook

Figure 3 Page d'accueil de l'application

Le client arrive sur une page d’accueil simple et visuellement assez attrayante, il peut alors se connecter de 2 manières ce qui le redirigera (en tout cas lors du premier login) sur une page d’authentification de Facebook.



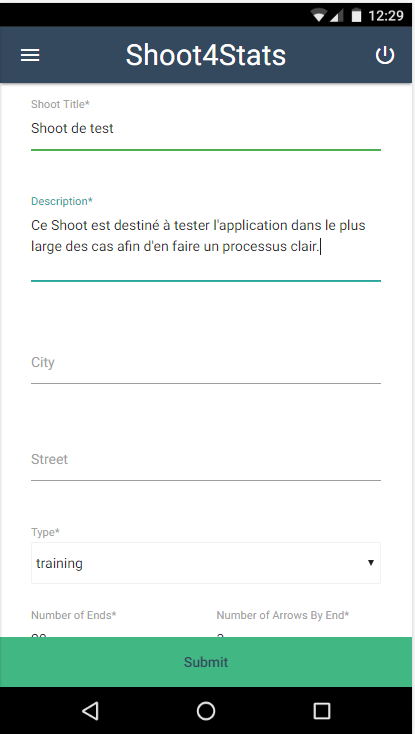


Figure 4 Création d'un Shoot

Figure 5 Dashboard première connexion

Après sa connexion l’utilisateur arrive successivement sur une page « Dashboard » qui lui explique ce qu’il doit faire pour continuer, puis sur la création d’un Shoot où les champs sont faciles à comprendre et le bouton ne s’active que si les champs sont justes.

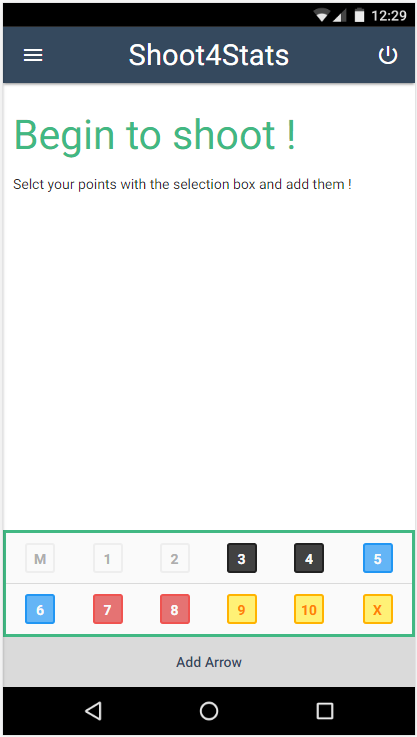
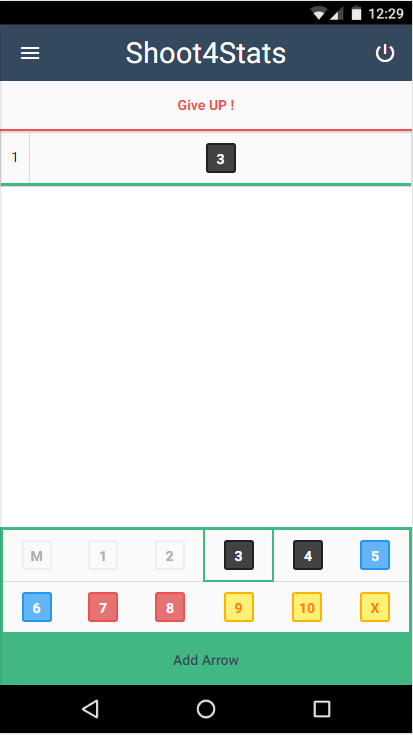


Figure 6 Remplissage d'un Shoot

Figure 7 Vue editShoot vide d'informations

L’utilisateur arrive sur la page de remplissage du *Shoot* qui lui explique qu’il doit entrer ses points grâce à la sélection en bas de l’écran. Puis quand il commence à ajouter ses flèches ses *Ends* vont s’ajouter automatiquement selon ses informations précédentes. Il peut aussi terminer son *Shoot* à n’importe quel moment. Si la connexion est perdue ou qu’il revient sur la page *Home* (*Dashboard*) l’état du Shoot sera visible et il pourra reprendre d’où il a terminé.

## 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.

### MLD

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

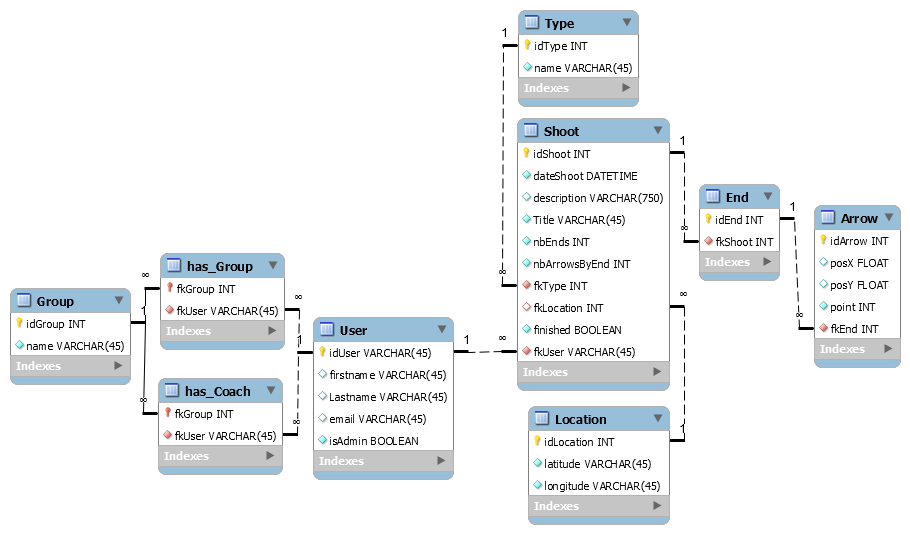


Figure 8 Modèle Logique des données

### Dictionnaire de données

Voici la description de la base de données :

* **Shoot**

Représente un Shoot.

* + **idShoot**

Id du Shoot, auto incrémentiel

* + **dateShoot, description, title**

Propriétés du Shoot, définition du Shoot

DateShoot : valeur par défaut CURRENT\_TIMESTAMP

*description* et *title* set par l’user

* + **nbEnds, nbArrowsByEnd**

Le Frontend choisira automatiquement suivant le type du Shoot, sauf si le type est « Training » auquel cas ce sera à l’utilisateur de choisir ces 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 ces champs pour *Rue* et *Ville* temporairement

* **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 Facebook[[1]](#footnote-1) 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\_*

* + **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

## API

Voir Annexes [*https://github.com/mbonjour/Shoot4Stats/blob/master/Documentation/Annexes/API/apiDocumentation.md*](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>

## Outils et ressources

### Vue JS

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

Expliquer ici les principes de base de VueJS

### VueX

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

Expliquer ici les principes de base de VueX

### PassportJS

<http://passportjs.org/docs>

Expliquer ici les principes de base de PassportJS

## 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.

Après une discussion avec le chef de projet l’on a décidé de faire la partie « Archers » le mieux possible (en termes de design et de gestion des données au sein des composants) les parties Administrateurs et anonymes sont donc à prévoir dans une prochaine version de l’application que les plus petits détails soient régler au sein de la partie Archers et que ce soit le plus facile possible d’ajouter des fonctionnalités. Ce rapport portera donc sur une seule partie du cahier des charges qui était « Partie Archers ».

# Dossier de Réalisation

Je vais ici expliquer au mieux comment le projet s’est déroulé, les problèmes rencontrés, l’aide que j’ai reçue et comment j’ai conçu l’application.

## Mise en place du projet

Explications : Vue init – Vue CLI

Choix du boiler plate, « Webpack » 🡪 Pourquoi Webpack et qu’est-ce que c’est en bref

## 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 (le chef de projet a bien a été clair sur un point : faire de la partie Archers la partie principale et la plus fonctionnelle)
* (Mettre en place la partie des administrateurs) 🡪 Suite aux demandes du client cette partie est optionnelle

## Menu et thème

### Menu dynamique

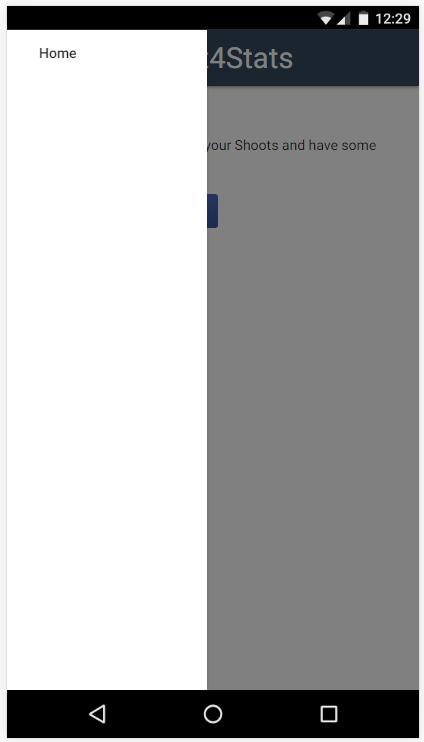
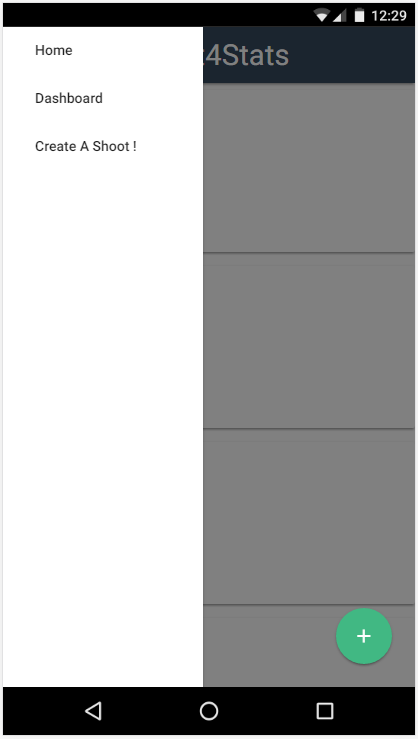
Le menu est dynamique selon qui le regarde afin d’éviter de se retrouver sur des pages sans y avoir accès

Figure 9 Menu en "Archers" v1

Figure 10 Menu en Anonyme

### Barre de Navigation

Figure 11 Menu en "Archers" v2

Figure 12 Menu en Administrateur

Ici le menu a été simplifié, le menu vers Home ramène sur la Dashboard de l’utilisateur. En effet ainsi l’utilisateur n’est pas surchargé d’informations par le menu

La barre de navigation prend 2 formes différentes :



Figure 14 Barre de navigation "Authentifiés mobile"

Figure 13 Barre de navigation lorsque l'on est pas authentifiés

Nous pouvons voir ci-dessus la barre de navigation qui s’affiche lorsque **l’on est authentifié**, à droite avec la possibilité de « log out » et à gauche le menu qui a été présenté plus tôt dans le rapport (cf. 4.3.1 ci-dessus).

Si **l’on n’est pas authentifié** l’icône n’apparaît simplement pas



Figure 15 Barre de navigation en desktop

Le menu en version Desktop

### Thème

#### Explication

Le choix des couleurs s’est fait en collaboration avec le chef de projet et nous avons donc décidé 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)

Voici le logo de VueJS :



Figure 16 Logo de VueJS

#### Exemple

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



Figure 17 Exemple de bouton stylisé

#### V2

Ce que je vais appeler la V2 dans ce rapport, c’est en fait une révision complète du design de l’application pour que ce soit bien plus épuré et plus ergonomique 🡪 Le client a insisté là-dessus c’était donc un objectif crucial de ce travail. C’est pour cela que le contenu est épuré et simplifié afin que l’utilisateur s’y retrouve plus facilement. Cette gestion du frontend m’a pris beaucoup de temps et j’ai reçu beaucoup d’aide du chef de projet sur la gestion de l’espace, mais aussi sur la gestion des données dans l’application.

#### Boutons

Cela paraît anodin mais les boutons sont positionnés de la même manière tout au long du projet, en effet ils sont fixés en bas de l’écran :

* + - Ceci afin de permettre de bien les voir car ils prennent aussi tout la largeur
    - Et de les atteindre très facilement du pouce sur la version mobile de l’application

### Framework CSS

Materialize…

## Partie Archers

### Maquettes

Les maquettes ont été faites avant de débuter les Vues et elles sont en Annexes, elles m’ont bien aidées à la représentation des composants que j’allais devoir mettre en place. Ainsi qu’à la représentation dont les datas sont gérées.

### Vue « Home »

#### Description

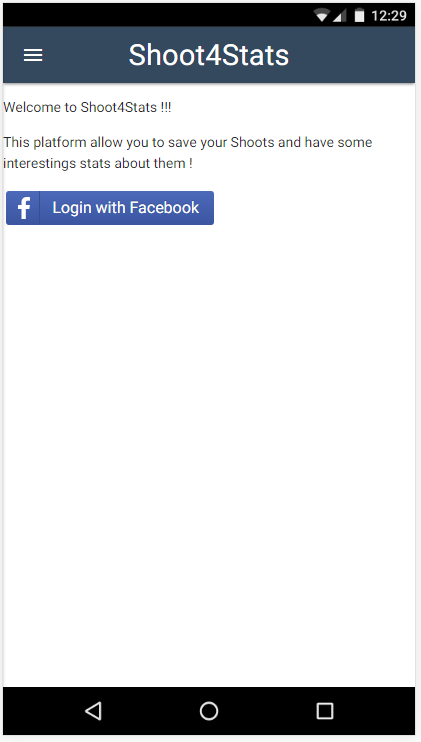
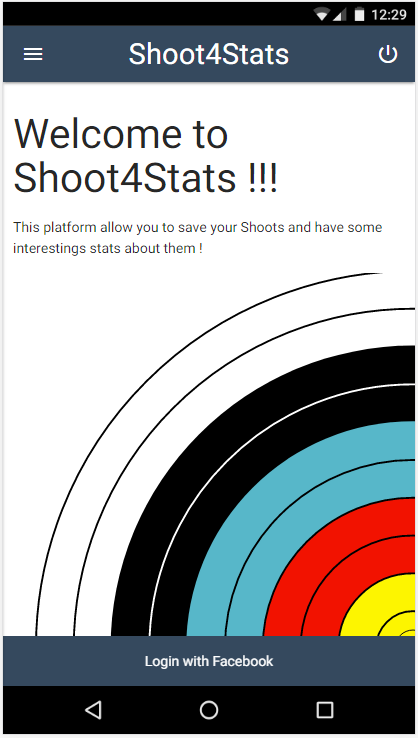
Cette Vue sera le point d’entrée de l’application WEB elle permettra à tout un chacun de s’authentifier (via Facebook, l’on récupère ainsi les emails, le nom et le prénom avec plus de facilité et plus d’ergonomie)

Figure 18 Vue Home v2

Figure 19 Vue Home v1

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 **PassportJS.** Le système d’authentification est expliqué dans la Doc de l’API (cf. 3.3)

#### Logique

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

### Vue « Dashboard » 🡪 Home lorsque l’on est authentifié

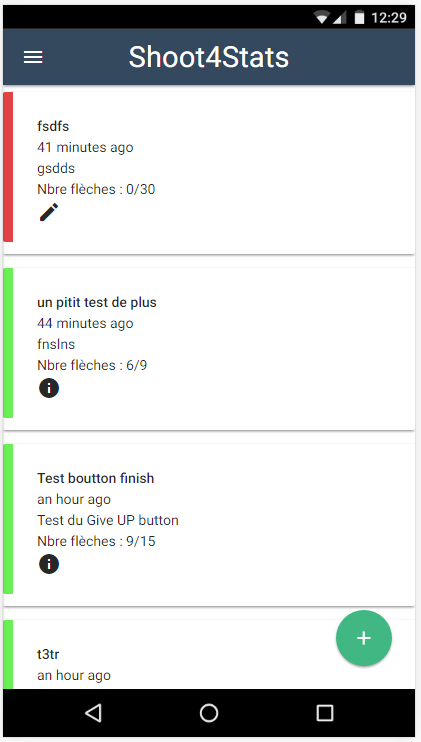
#### Description v1

La vue Dashboard a pu être discutée avec le client.

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 ou d’en reprendre un s’il n’est pas terminé.

#### Composant Dashboard



Composant ShootSummary, il récupère les infos passées par le Dashboard

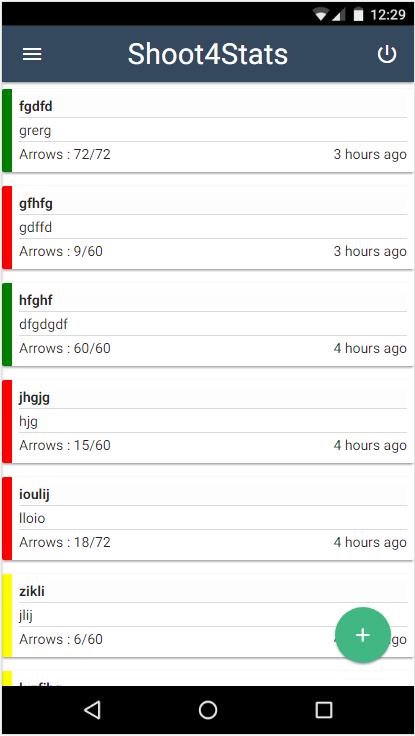
Composant « Dashboard », récupère Shoot et crée pour chacun un composant *ShootSummary*

Bouton d’ajout de *Shoot* il s’étends afin de choisir le type de *Shoot* voulu

Figure 20 Vue Dashboard décomposée v1

#### Description v2

La « V2 » de la vue Dashboard est assez différente, nous avons décidé d’enlever les icônes afin de permettre à l’utilisateur de toucher n’importe où sur la carte afin d’en voir les détails. En soi le fonctionnement du Dashboard n’a pas changé plus que cela, c’est la structure des composants internes qui ont changés comme on le verra au long du chapitre **en continuant avec la terminaison « v1 » et « v2 »**



Composant *ShootItem*, il récupère les infos sur VueX

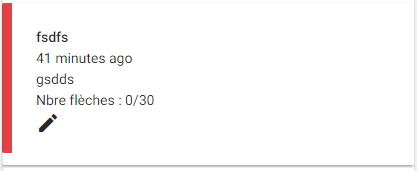
Composant « *Dashboard* », récupère Shoots grâce à VueX et crée pour chacun un composant *ShootItem*

Bouton d’ajout de *Shoot* il s’étends afin de choisir le type de *Shoot* voulu

Dans le cas d’un clic sur un Shoot terminé (vert ou jaune) le composant *ShootDetails* est appelé

Composant *ShootSummary* qui affiche des infos très brèves

#### Composant ShootSummary v1



Bouton permettant de continuer le Shoot ou d’en voir les détails

La bande rouge indique que le Shoot n’est pas terminé, à l’inverse du vert

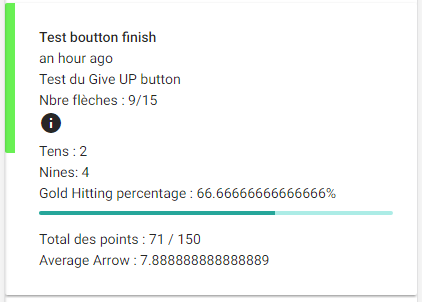
Description

Indication de temps

Titre du *Shoot*

Figure 21 Composant ShootSummary v1

#### Composant ShootDetails v1

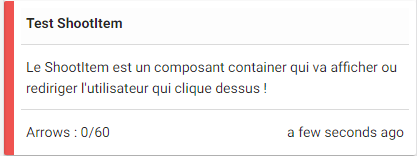


Informations à propos du Shoot plus poussées (p.ex. les points moyens, le pourcentage de flèche dans le Jaune, …)

Lors du clic sur cette icône le composant ShootDetails est affiché

Figure 22 Composant ShootSummary et affichage ShootDetails

#### Composant ShootItem (v2)

Le composant ShootItem contient le titre et la description mais aussi le composant ShootSummary dans son état non actif.

Mais si l’on clique dessus (le composant, soit la carte) on active la version détails du Shoot et l’on voit ainsi le composant ShootDetails prendre la place du composant ShootSummary encadré ci-dessus.

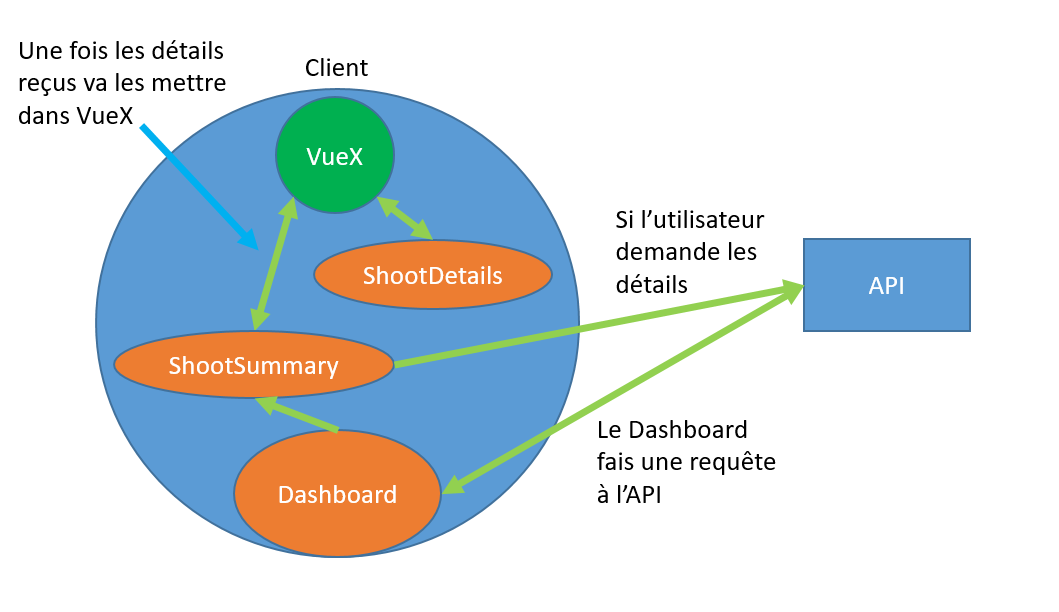
Figure 23 Composant container ShootItem

#### Composant ShootDetails v2

Le composant shootDetails est encore en cours de production mais ne sera pas plus différent que la « v1 » avec des chiffres à 2 décimales (pour plus de lisibilité évidemment) et d’un tableau affichant le nombre de flèches / points (5 flèches dans le 10, 6 flèches dans le 9, …).

#### Logique – Interaction API Schémas v1

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 partagés[[2]](#footnote-2) *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 partagés2 *shootDetails* qui prend les informations du state VueX et va afficher ses informations supplémentaires de manière simple.

#### Logique – Interaction API Schémas v2

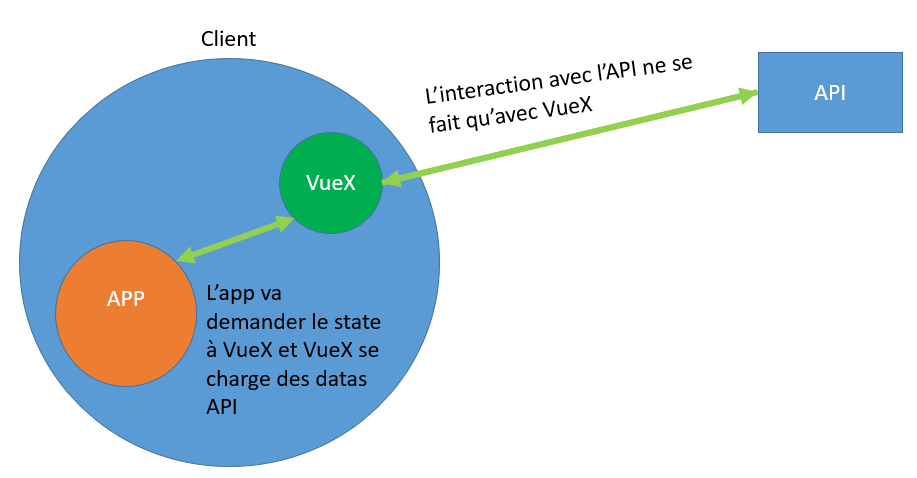


Figure 24 Interaction avec VueX

L’interaction avec l’API ne se fait que par VueX ainsi l’on sait que si l’on a besoin de quelques informations au sein du frontend on doit demander à VueX (de plus c’est réactif[[3]](#footnote-3)).

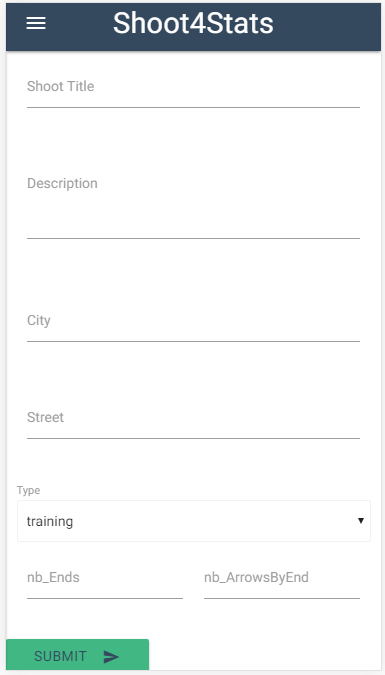
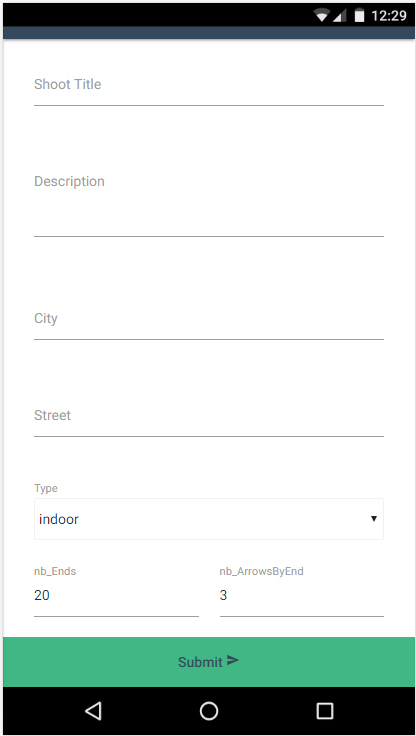
L’on voit si l’on compare les schémas que la v2 est plus simple, plus pratique et plus logique.

### Vue « Create Shoot »

#### Description

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

#### ScreenShots



Le bouton déclenche la validation et envoie la requête avec l’objet voulu pour l’API

Formulaire simple grâce à Materialize

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

Figure 25 Vue CreateShoot v2

Figure 26 Vue CreateShoot v1

#### 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.

#### Interaction API – Schémas

Ici pour la **V2** l’on passe par VueX et v1 direct par l’API. Je mets sur le point sur les 2 versions afin de bien signifier qu’un travail sur les données a été effectué.

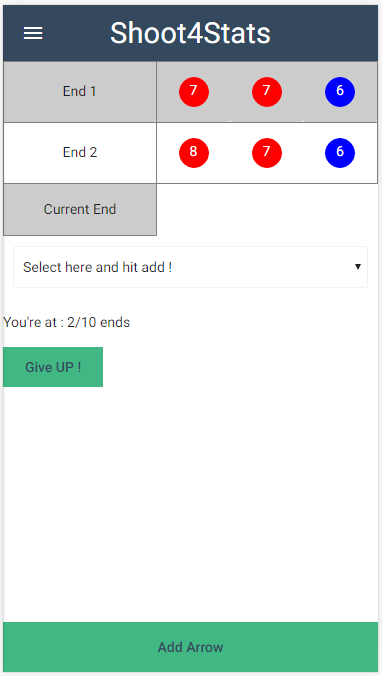
### Vue « Edit Shoot »

#### Description v1

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.

#### ScreenShots – Schémas v1



Les Arrows ajoutées s’affichent ici

Tableau dynamique, il boucle sur les données du currentShoot dans VueX

Sélection de la flèche courante

Bouton permettant de finir le *Shoot*. Envoie */api/shoots/idShoot/finish* à l’API

Permet d’ajouter la flèche sélectionnée actuellement dans la zone de sélection

Figure 27 Vue editShoot v1

#### Composant arrowItem

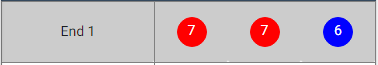


Figure 28 Composant arrowItem

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 :

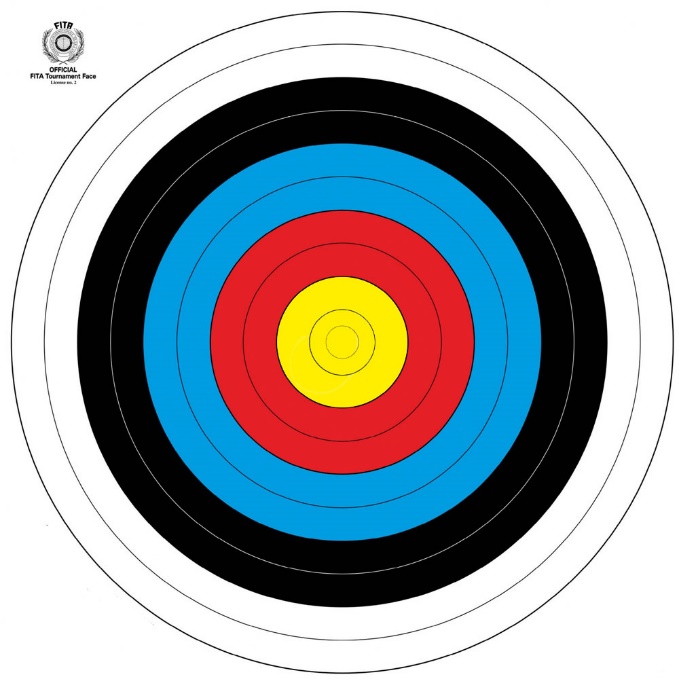


Figure 29 FITA Official Target face

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 interprétés par un « M » et le 11 par un « X » mais compte comme 10 points au sein de l’API

#### 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 modifié l’API pour que lors des calculs il transforme le « X » en 10 pour ne pas fausser les calculs.

#### 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. Ces 2 solutions ont étés mise en place par moi-même ensuite mais il m’a donné des pistes à la complétion de ces tâches. Vous pouvez le voir dans la **V2** ci-dessous.

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.

#### Description V2

La vue a été repensée au long du projet, en effet, comme on le voit dans la v1 le bouton Give Up ressemble bien trop au bouton d’ajout et il est à trop courte portée de la liste de sélection. On a donc décidé de mettre un bouton Give UP plus discret et moins atteignable.

Et les pastilles… Les pastilles dans la v1 attaquent l’œil par leurs couleurs bien trop vives, je me suis donc inspiré des couleurs pastel de Material design pour mettre en forme un tableau plus épuré et plus visible…

Puis quant à faire les choses le mieux possible pour cette partie Archers j’ai fait un composant sélecteur fait maison qui reprend les pastilles faites grâce à un autre composant. C’est dans ces cas-là que l’on voit la puissance de VueJS s’il est bien utilisé, vu que la modularité de mon composant « arrowItem » a été assez bien faite j‘ai pu le réutiliser sans problèmes.

Le sélecteur est un composant fait maison mais on a tout de même la possibilité d’y mettre un v-model (récupération des données réactives) grâce à un événement lors des changements de valeur (ici le clic sur une case du tableau) et en retournant une propriété « data ».

#### ScreenShots

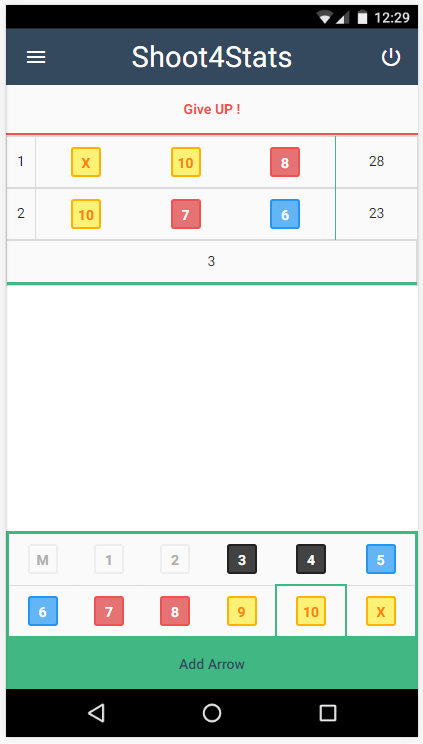


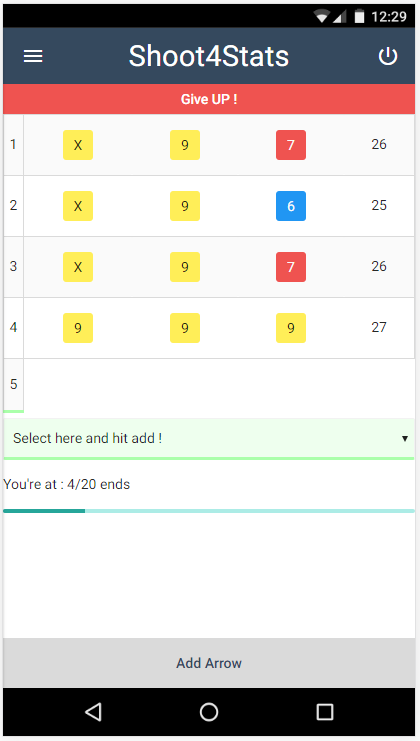
Figure 30 Vue editShoot v2

## Description des tests effectués

### Test sur editShoot

Le test a été effectué dans une version mise en production au cours du projet et hébergée chez moi afin de permettre aux testeurs externes de faire des tests.

Le test a été d’inclure des valeurs nulles et/ou inexactes. Ce test a été effectué par mon chef de projet et m’a donc averti de cette erreur, nous avons donc dû renforcé la faible validation qui était en place du côté de l’API puis éviter au moins de pouvoir mettre des valeurs nulles en désactivant le bouton lorsqu’aucun point n’était sélectionné :



L’on peut voir ici que le bouton est grisé et si l’on essaie de cliquer dessus rien ne se passera !

### Test sur CreateShoot

Là aussi c’est grâce à ma mise en production au cours du projet (de la v1) que j’ai pu avoir un retour de mon collègue Nicolas Crausaz, il m’a fait remarquer que lors de la création d’un Shoot il était possible de mettre un nombre d’Ends négatifs et de même pour les flèches dans les Ends. J’ai donc dû mettre une validation pour cela et toasté les erreurs correctement.

## Erreurs restantes

Le but du projet (et étant donné que je n’ai fait que la partie Archers) était qu’il reste le moins d’erreurs possible et dans l’idéal aucune (on en rêve tous évidemment). Cependant, il reste encore une possibilité d’amélioration :

* Lors de la création d’un *Shoot* quelqu’un qui s’y connaît peut forcer l’entrée de champs vides et comme la vérification du côté de l’API n’est pas active le Shoot est tout de même crée sans *Titre* ou sans *Description* par exemple.

## Dossier d'archivage

Le CD ROM contient à sa racine un dossier nommé « Shoot4Stats » et tout est dedans, en effet voici l’arborescence détaillée du projet (aussi disponible sur [github.com](https://github.com/mbonjour/Shoot4Stats)) :

# Mise en service

Explication de la petite mise en production (fait maison) pour les tests

## 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.*

## 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…

# Conclusions

Les conclusions, les améliorations possibles et le ressenti du projet

*Développez en tous cas les points suivants:*

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

# Annexes

Ici se trouve la description des Annexes et les sources utilisées lors du projet

## Sources – Bibliographie

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

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

<http://passportjs.org/docs>

<http://www.chartjs.org/docs/>

<http://materializecss.com/>

<http://stackoverflow.com/> (je n’ai malheureusement pas gardé tous les liens)

<https://uxplanet.org/optimistic-1000-34d9eefe4c05>

J’ai beaucoup utilisé les sources officielles de VueJS et des Framework / librairies que j’utilisais et peu d’autre liens.

## Journal de travail

Présent en Annexe

## Manuel d'Installation

Présent en Annexe

## Manuel d'Utilisation

Ce référer au processus pour Archers 3.1 ci-dessus

## Archives du projet

*CD, … dans une fourre en plastique*

Annexes

1. Pour plus de détails : https://developers.facebook.com/docs/graph-api/reference/v2.5/user [↑](#footnote-ref-1)
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. [↑](#footnote-ref-2)
3. Réactifs c’est à dire que si une valeur change dans VueX dû à un appel à VueX d’un autre composant la valeur va changer en direct partout dans l’APP [↑](#footnote-ref-3)