

# SocialShuffle

---



Candidat : **DOS SANTOS Samuel**, FIN2 2024

Chef de Projet : **CHARMIER Grégory**

Expert 1 : **VENRIES Luc**

Expert 2 : **OBERSON Bernard**

Durée : **88h**, du 13.05.2024 au 03.06.2024



# Table des matières

---

<b>1 SPÉCIFICATIONS .....</b>	<b>5</b>
1.1 TITRE .....	5
1.2 INTRODUCTION.....	5
1.3 OBJECTIFS.....	5
1.4 POINTS TECHNIQUES .....	5
1.5 MATÉRIEL ET LOGICIELS À DISPOSITION .....	6
1.6 PRÉREQUIS.....	6
1.7 LIVRABLES .....	6
<b>2 PLANIFICATION INITIALE .....</b>	<b>7</b>
<b>3 ANALYSE.....</b>	<b>8</b>
3.1 MÉTHODOLOGIE DE TRAVAIL .....	8
3.2 ORGANISATION DES RÉSULTATS DU TRAVAIL .....	8
3.3 ENVIRONNEMENT .....	8
3.3.1 <i>Laravel</i> .....	8
3.3.2 <i>MVC</i> .....	8
3.3.3 <i>L'ORM Eloquent</i> .....	9
3.4 CONCEPTION ET ANALYSE.....	10
3.4.1 <i>MCD / MLD</i> .....	10
3.4.2 <i>Diagramme de flux</i> .....	11
3.4.3 <i>Maquettes graphiques</i> .....	11
3.5 STRATÉGIE DE TESTS .....	15
3.5.1 <i>Tests de la base de données avec les Factories</i> .....	15
3.5.2 <i>Laravel Dusk</i> .....	16
3.5.3 <i>Tests automatisés avec GitHub Action</i> .....	16
3.5.4 <i>Scenarios des tests End-To-End</i> .....	16
3.6 RISQUES TECHNIQUES .....	17
<b>4 RÉALISATION .....</b>	<b>17</b>
4.1 MISE EN PLACE DE LARAVEL 11 .....	17
4.1.1 <i>Le fichier .env</i> .....	17
4.1.2 <i>Tester le fonctionnement de l'installation.</i> .....	18
4.1.3 <i>Installation de Tailwind CSS</i> .....	19
4.2 MIGRATIONS .....	20
4.2.1 <i>Table t_user (users)</i> .....	20
4.2.2 <i>Table t_team (teams)</i> .....	20
4.2.3 <i>Table t_member (members)</i> .....	21
4.2.4 <i>Table t_group (groups)</i> .....	21
4.2.5 <i>Table t_dispatch (group_member)</i> .....	22
4.3 MISE EN PLACE DES CONTRÔLEURS .....	22
4.3.1 <i>Contrôleurs de ressources</i> .....	22
4.4 MISE EN PLACE DES ROUTES .....	23
4.4.1 <i>La route « / » (Racine)</i> .....	25
4.4.2 <i>Ressources</i> .....	25
4.4.3 <i>Authentification</i> .....	25
4.4.4 <i>À propos</i> .....	25
4.5 MISE EN PLACE DES VUES .....	25
4.5.1 <i>Layout</i> .....	25
4.5.2 <i>Composant réutilisable pour l'affichage des membres</i> .....	27
4.5.3 <i>Utilisation du même formulaire pour la création et la modification</i> .....	27

4.5.4	<i>Affichage des groupes</i>	28
4.5.5	<i>Difficultés rencontrées</i>	29
4.6	FORMULAIRES DE CRÉATION D'UNE ÉQUIPE AVEC LES MEMBRES	29
4.7	AUTHENTIFICATION	30
4.7.1	<i>Formulaire d'authentification</i>	30
4.7.2	<i>Traitement de l'authentification</i>	30
4.7.3	<i>Déconnexion</i>	31
4.8	DROITS DES UTILISATEURS	31
4.8.1	<i>Création d'une Policy</i>	32
4.8.2	<i>Utilisation des Policies dans les vues</i>	32
4.8.3	<i>Utilisation des Policies dans les contrôleurs</i>	33
4.9	ALGORITHME DE RÉPARTITION DES MEMBRES	33
4.9.1	<i>Pistes d'améliorations</i>	33
4.10	AFFICHAGE DU QR CODE	33
4.11	PROCÉDURE UTILISÉE POUR LE DÉPLOIEMENT	33
4.11.1	<i>Contraintes</i>	33
4.11.2	<i>Procédure</i>	34
4.11.3	<i>Exécuter des commandes Artisan depuis une URL</i>	36
4.12	TROIS MESURES DE SÉCURITÉ	36
4.12.1	<i>Token CSRF</i>	36
4.12.2	<i>Validation des formulaires</i>	37
4.12.3	<i>Hachage des mots de passe</i>	38
4.13	TROIS MESURES POUR LE « RESPONSIVE DESIGN »	39
4.13.1	<i>Affichage des cartes en fonction de la largeur de l'écran</i>	39
4.13.2	<i>Navigation</i>	40
4.13.3	<i>Boutons de gestion des membres</i>	41
4.14	MISE EN PLACE DES TESTS LARAVEL DUSK DANS GITHUB ACTIONS	41
4.14.1	<i>Installation de Laravel Dusk</i>	41
4.14.2	<i>Mise en place de GitHub Action</i>	42
4.14.3	<i>Création d'un test Laravel Dusk</i>	43
4.14.4	<i>Risque de sécurité</i>	45
4.14.5	<i>Difficultés rencontrées</i>	45
<b>5</b>	<b>TESTS</b>	<b>47</b>
5.1	DOSSIER DES TESTS	47
<b>6</b>	<b>CONCLUSION</b>	<b>48</b>
6.1	BILAN DES FONCTIONNALITÉS DEMANDÉES	48
6.2	BILAN DE LA PLANIFICATION	48
6.3	BILAN PERSONNEL	48
<b>GLOSSAIRE</b>		<b>49</b>
<b>7</b>	<b>ANNEXES</b>	<b>51</b>
7.1	RÉSUMÉ DU RAPPORT DE TPI	51
7.2	ACCÈS AU REPOSITORY GITHUB	51
7.3	BIBLIOGRAPHIE	51
7.4	PLANNING	52
7.6	JOURNAL DE TRAVAIL	61

# 1 SPÉCIFICATIONS

---

## 1.1 Titre

Réalisation de l'application web SocialShuffle qui a pour but de mélanger des participants pour favoriser les interactions sociales.

## 1.2 Introduction

Ce projet a pour but de réaliser une application Web ayant pour but de mélanger des membres d'une équipe dans des groupes hétérogènes dans le but de favoriser les interactions sociales.

Un des cas d'utilisation possible est lors du début d'une année scolaire, de proposer des activités de team building en faisant en sorte que tous les participants interagissent entre eux. Cette application permettra donc de retirer la difficulté de devoir créer les groupes manuellement.

L'enjeu principal est la mise en pratique du Framework Laravel<sup>\*1</sup> dans un projet concret.

## 1.3 Objectifs

Les objectifs évalués pour ce projet sont les suivants :

- Les formulaires doivent afficher des erreurs tout en remplaçant à nouveau les formulaires lors de ces occurrences. Cela implique une validation minutieuse des champs.
- Mise en place de mesures de sécurité
- Un manuel d'installation sous la forme d'un fichier README.md sur le repository GitHub.
- Mettre en place et expliquer l'algorithme permettant la répartition des membres dans les différents groupes.
- Le site doit être responsive\*.
- Justification des choix faits dans le MCD / MLD / MPD

En plus de ces objectifs, des éléments supplémentaires devront être intégrés à l'application si le temps le permet:

- Importation des membres depuis un fichier CSV en plus de l'entrée manuelle sur le site.
- Affichage d'un code QR permettant de rediriger vers la page montrant les différents groupes.

## 1.4 Points techniques

L'application devra respecter les contraintes techniques suivantes :

- Utilisation du Framework **Laravel**.
- Une base de données **MySQL** reliée à l'application.
- L'interface de l'application devra être responsive.

---

<sup>1</sup> Les termes marqués du symbole \* sont détaillés dans le glossaire

## 1.5 Matériel et logiciels à disposition

- Un PC de l'ETML
- Visual Studio avec des extensions facilitant le développement en Laravel
- PHP 8.3 pour l'environnement local
- Composer
- uWamp pour le serveur de base de données de développement

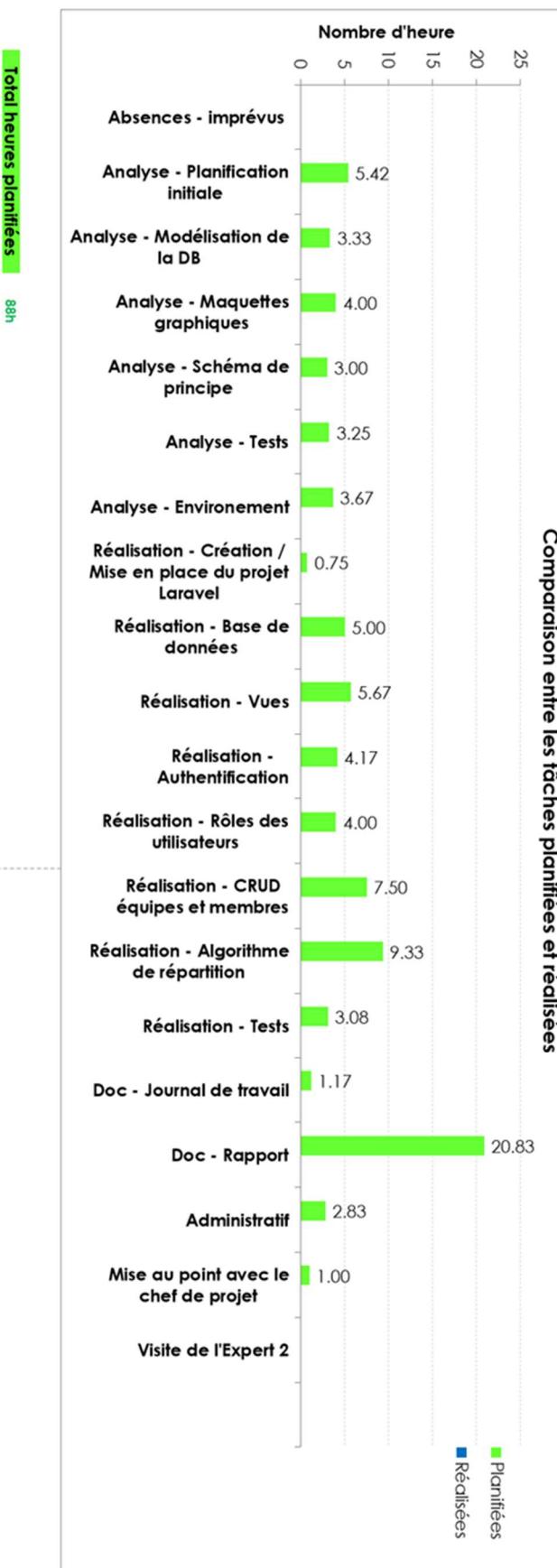
## 1.6 Prérequis

- Connaître les notions de la Programmation Orienté Objet
- Savoir coder en PHP
- Connaître et comprendre les mécanismes principaux de Laravel

## 1.7 Livrables

- Planification initiale
- Rapport du projet
- Journal de travail

## 2 PLANIFICATION INITIALE



## 3 ANALYSE

---

### 3.1 Méthodologie de travail

La méthode des **six pas** sera utilisée pour la réalisation de ce projet. Celle-ci s'inscrit parfaitement dans le cadre du TPI où une limite de temps est imposée.

### 3.2 Organisation des résultats du travail

Afin d'assurer la bonne organisation du travail tout au long du projet, la procédure suivante est appliquée :

Chaque mise à jour aux experts (mardi et jeudi soir) correspond à une nouvelle version des documents.

- Les versions sont citées au début du nom du document
- Chaque jour, au moins un commit et push sont effectués afin d'enregistrer le travail
- Un répertoire est créé, dedans se trouve l'entièreté du projet. Il s'agit du repository.

### 3.3 Environnement

#### 3.3.1 Laravel

Laravel<sup>2</sup> est un Framework WEB basé sur le modèle MVC\*. Il possède une syntaxe expressive qui permet de faire appel à des fonctions de manière naturelle. L'interface en ligne de commandes (CLI) Artisan<sup>3</sup> permet de pouvoir interagir rapidement avec l'application (créer un contrôleur, effectuer des migrations vers la base de données ou même optimiser l'application pour le déploiement).

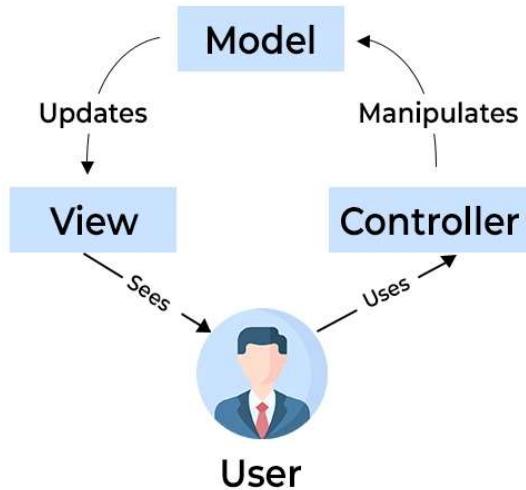
#### 3.3.2 MVC

Le modèle MVC (Modèle\*, Vue\*, Contrôleur\*) vise à diviser la charge de travail d'une application. Le modèle a pour fonction d'interagir avec la base de données en créant ou en récupérant des enregistrements, la vue quant à elle sert à proposer un affichage à l'application et le contrôleur permet de transporter les requêtes entre le modèle et la vue.

---

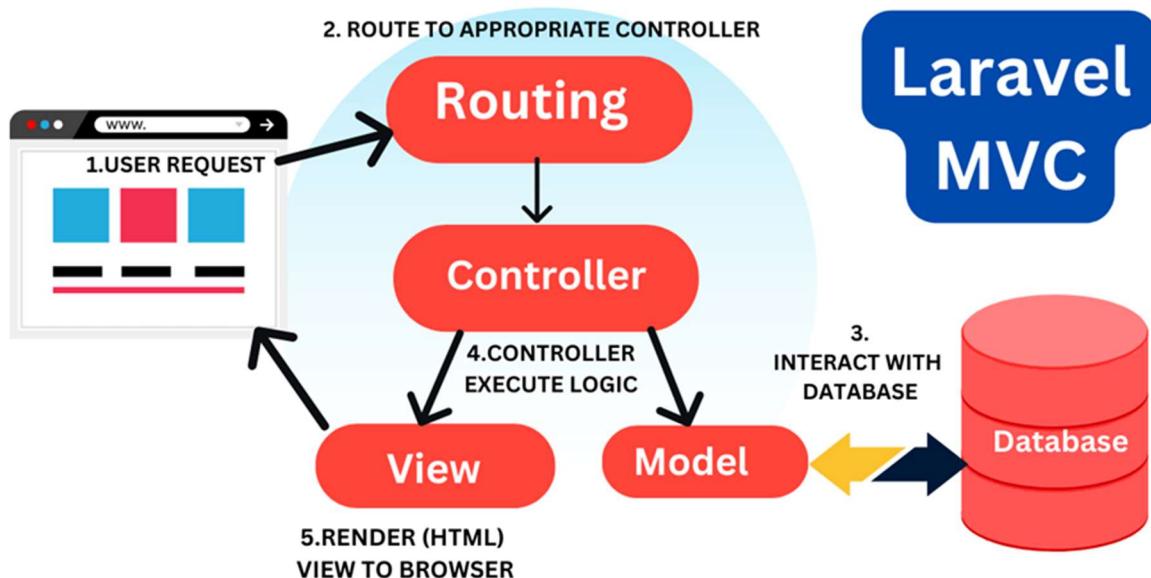
<sup>2</sup> Documentation officielle de Laravel <https://laravel.com/docs/11.x>

<sup>3</sup> Documentation sur Artisan : <https://laravel.com/docs/11.x/artisan#introduction>



1. Représentation schématique du modèle MVC

Laravel respecte ce principe fondamental en ajoutant le concept de routes\*. Celles-ci permettent à partir de l'Url de rediriger vers une fonction spécifique d'un contrôleur, ou plus globalement, d'exécuter du code.



2. Représentation schématique du modèle MVC appliquée à Laravel.

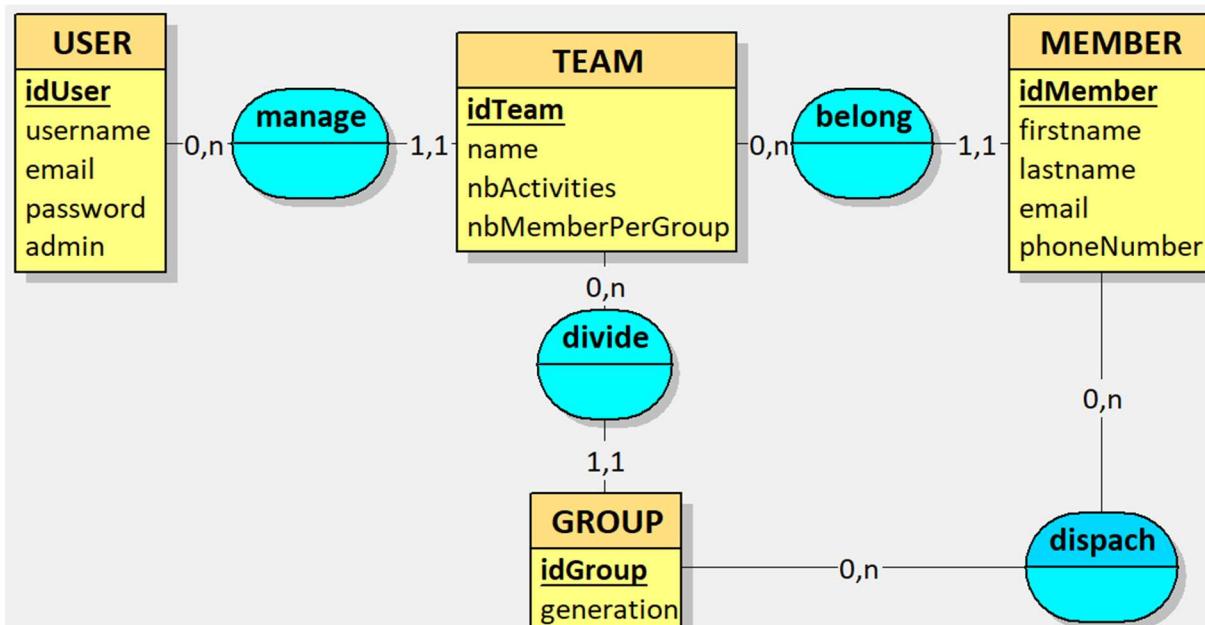
### 3.3.3 L'ORM Eloquent

Laravel, met à disposition un ORM\* qui permet de décrire des requêtes à la base de données avec du code au lieu de taper des commandes directement des commandes SQL. L'utilisation de cet outil permet de rendre les interactions avec la base de données plus naturelles, ce qui est censé rendre le développement plus rapide et agréable.

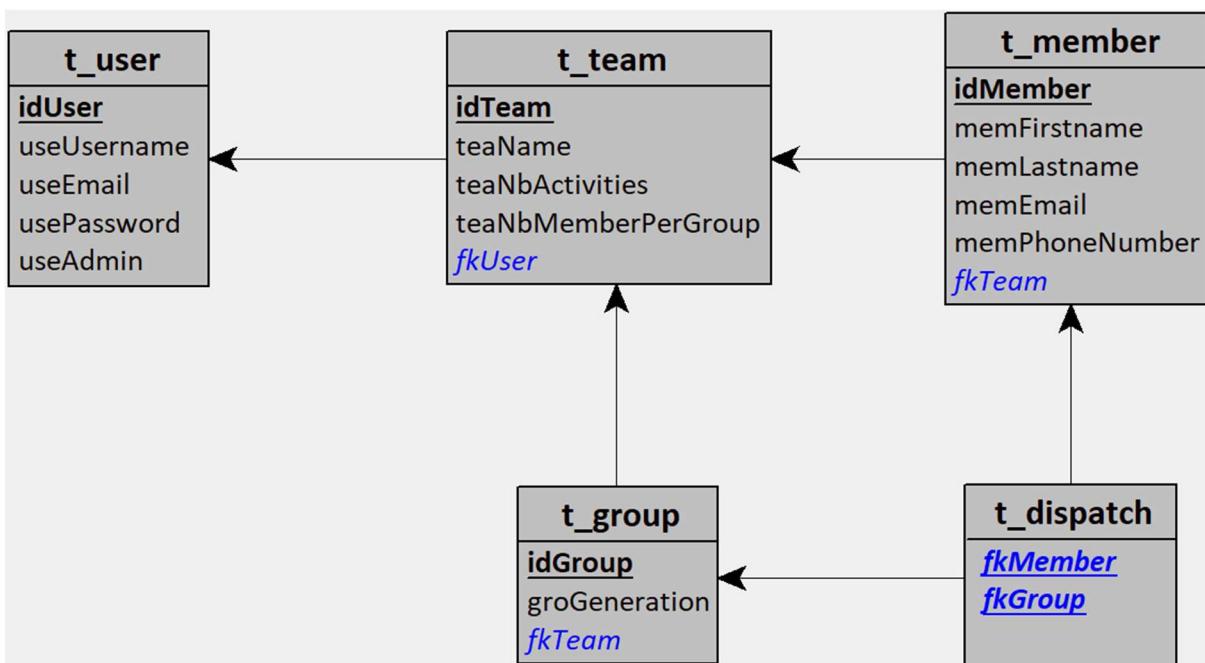
## 3.4 Conception et analyse

### 3.4.1 MCD / MLD

Le MCD et le MLD respectent ici les normes de codage de l'ETML. Afin de respecter les bonnes pratiques de développement en Laravel, ce seront les normes de Laravel qui seront utilisées durant la réalisation. Ces normes permettent d'automatiser<sup>4</sup> certains éléments de l'application Ceci est la raison pour laquelle les noms diffèrent entre les modèles et le code.



3. Modèle Conceptuel de Données (MCD) pour l'application SocialShuffle

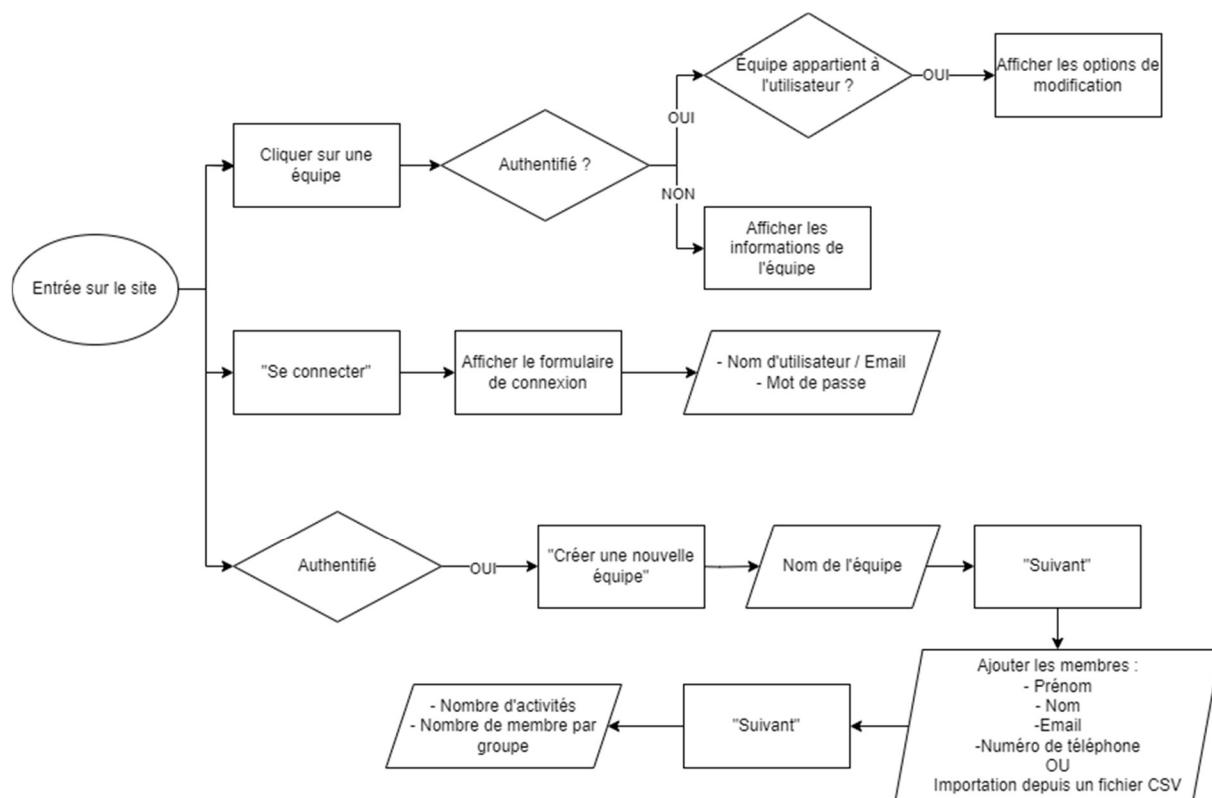


4. Modèle Logique de Données (MLD) pour l'application SocialShuffle

<sup>4</sup> Exemple d'automatisation avec les normes de Laravel :

<https://laravel.com/docs/11.x/migrations#:~:text=Since%20this%20syntax,rewritten%20like%20so%3A>

### 3.4.2 Diagramme de flux

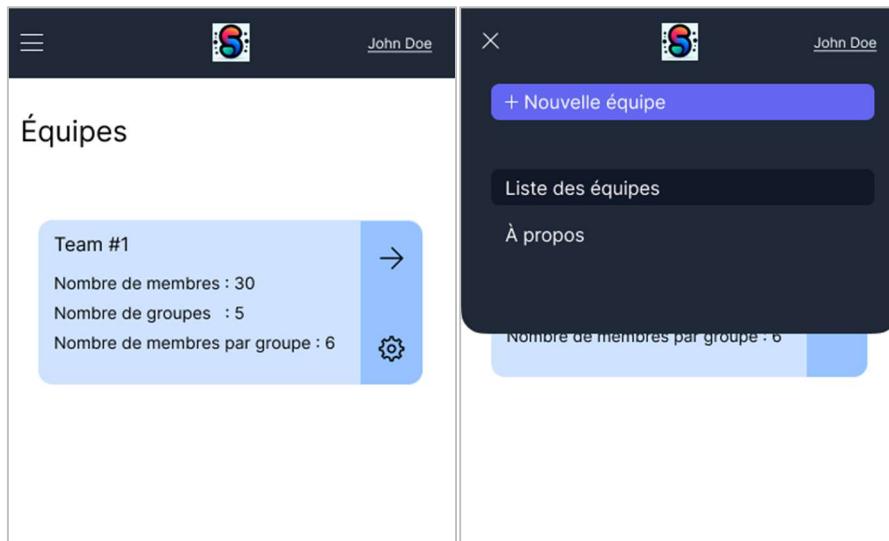


5. Représentation des interactions principales avec le site.

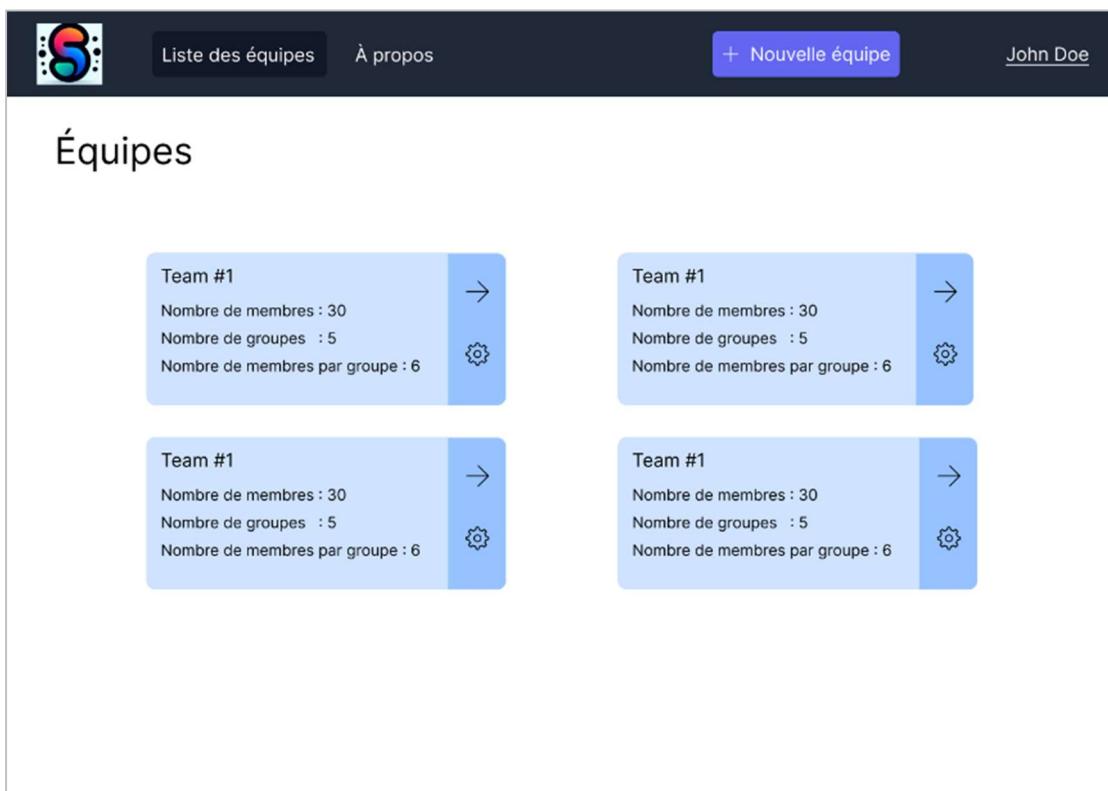
### 3.4.3 Maquettes graphiques

#### 3.4.3.1 Page principale

Les maquettes graphiques qui suivront, permettront de donner une perspective de ce à quoi le site devrait ressembler. Le but est notamment de définir les principales différences entre la version mobile et desktop (responsive design). Elles permettent également d'avoir un aperçu des messages d'erreurs ainsi que les messages d'avertissemens. Un des aspects majeurs est l'affichage des informations sous forme de cartes. Cela permet de distinguer intuitivement chaque équipe par exemple.



6. Maquette de la page principale en version mobile. Avec le menu ouvert et fermé.



7. Maquette de la page principale en version desktop

Dans la version mobile, contrairement à la version desktop, pour pouvoir accéder à la navigation, il faut ouvrir un menu. Cela évite que trop d'éléments se trouvent en même temps sur l'écran ce qui réduirait l'expérience utilisateur.

### 3.4.3.2 Exemple de formulaire

The screenshot shows a mobile application interface for adding members. At the top, there is a dark header bar with a menu icon (three horizontal lines), a logo consisting of three colored dots (red, green, blue), and the text "John Doe". Below the header, the main title "Ajouter les membres" is displayed. The form consists of several input fields: "Prénom" (First Name) with the value "John", "Nom" (Last Name) with the value "Doe", and "Email" with the value "john.doe@examplech". A red error message below the email field states "Email invalide. Veuillez respecter le format : exemple@example.com". The "Email" field is highlighted with a red border. There is also a field for "Numéro de téléphone" (Phone Number) with the value "+41790000000". Below the form are two buttons: "Ajouter un membre" (Add member) and "Importer depuis un CSV" (Import from CSV). A note above the member list says "Nombre de membres : 25". A single member card is visible, showing "John Doe" and "john.doe@example.ch".

8. Exemple de formulaire avec la maquette d'ajout des membres en version mobile.

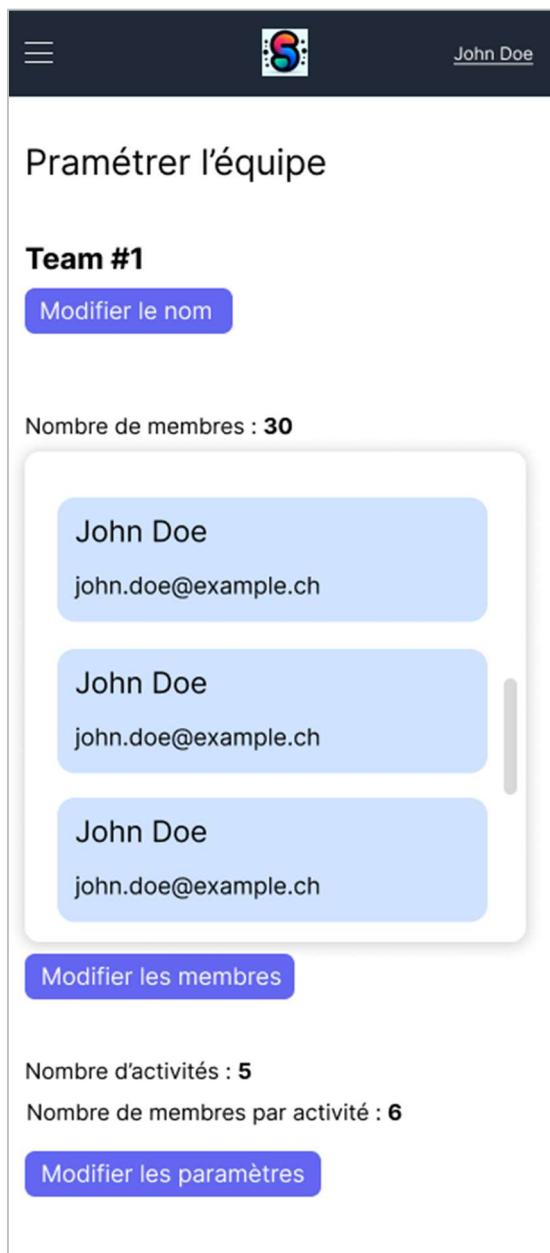
Cette maquette montre un exemple de formulaire. Celui-ci correspond à l'ajout des membres lors de la création d'une équipe. On peut y voir une erreur dans le champ « email » où l'utilisateur fictif a oublié un point, ce qui a déclenché l'affichage d'un message d'erreur. Une fois les membres ajoutés, ils apparaissent sous forme de cartes en dessous du formulaire.

Voici également l'équivalent de ce formulaire dans la version desktop :

The screenshot shows a mobile application interface for adding a member. At the top, there is a navigation bar with a logo, a "Liste des équipes" button, an "À propos" button, a purple "+ Nouvelle équipe" button, and a user profile "John Doe". Below the navigation bar, the main content area has a title "Ajouter un membre" and a "Suivant →" button. The form fields include "Prénom" (John), "Nom" (Doe), "Email" (john.doe@examplech) which is marked as invalid with the message "Email invalide. Veuillez respecter le format : exemple@example.com", and "Numéro de téléphone" (+41792101885). There are two buttons at the bottom: "Importer depuis un CSV" and "Ajouter un membre". At the bottom of the screen, it says "Nombre de membres : 25" and shows a grid of four member cards, each with the name "John Doe" and email "john.doe@example.ch".

9. Exemple de formulaire avec la maquette d'ajout des membres en version mobile.

### 3.4.3.3 Paramétrage de l'équipe



10. Page de paramétrage d'une équipe

Cette maquette montre la page où sont affiché toutes les options de paramétrage qu'il est possible de faire une fois qu'une équipe a été créée. Chaque bouton sera ensuite censé rediriger vers un formulaire de modification.

## 3.5 Stratégie de tests

### 3.5.1 Tests de la base de données avec les Factories

Les factories<sup>5</sup> sont un outil qui permet de tester une base de données et les modèles associés en définissant un type de données fictives dans des tables et colonnes données. Cela permet habituellement de populer facilement et rapidement une base de données afin de vérifier que l'application fonctionne correctement avec ces dernières. Les factories sont utilisées en parallèle des seeders\* qui eux exécutent la création des données et les enregistrent.

<sup>5</sup> Factories: <https://laravel.com/docs/11.x/database-testing>

### 3.5.2 Laravel Dusk

Laravel Dusk<sup>\*6</sup> est un outil qui s'intègre à Laravel et qui s'inscrit dans la catégorie des tests de navigateurs. Il permet notamment de simuler des interactions qu'effectuerai un utilisateur avec un site

Ce type de tests et très intéressant car ils sont effectués en interagissant directement avec le produit. Cela permet donc par exemple de tester le bon fonctionnement des différents formulaires. Il sera donc possible de tester la création des équipes avec leurs membres et la génération automatique des groupes

L'utilisation de ChromeDriver<sup>\*</sup> permet à Laravel Dusk de pouvoir lancer le site sur un navigateur.

### 3.5.3 Tests automatisés avec GitHub Action

Github Actions<sup>\*78</sup> est un outil d'intégration continue avec lequel il est possible d'automatiser des tests lors de chaque push. Cela permettra lors de chaque modification de s'assurer que l'application fonctionne toujours comme prévu. Si une erreur apparait pendant un test, une notification sera affichée sur GitHub. Dans le cadre de ce projet, il sera utilisé pour automatiser les tests Dusk.

### 3.5.4 Scenarios des tests End-To-End

La planification des scénarios de tests End-to-End<sup>\*</sup> permet de définir tous les aspects fonctionnels de l'application. Ainsi si tous les tests sont validés, on pourra considérer que l'application est fonctionnelle et donc utilisable.

Nom du test	Description	Critère de validation
LoginTest	Test de l'authentification. L'utilisateur entre ses authentifiant dans les champs requis.	L'utilisateur est connecté.
WrongLoginTest	Test de l'authentification avec des informations erronées.	L'utilisateur n'est pas authentifié, il reste sur la page de connexion.
CreateTeamTest	Test de la création d'une équipe complète avec ses membres et ses groupes.	L'équipe est créée sans erreurs. Tous les membres sont présents dans leurs groupes respectifs et selon les générations.
CreateTeamWrongValuesTest	Test de la création d'une équipe en insérant des données erronées.	Des messages d'erreur sont retournés avec les données erronées.
DeleteTeamTest	Tester la suppression d'une équipe complète, qui contient des membres et des groupes.	L'équipe est supprimée sans erreurs retournées.

<sup>6</sup> Laravel Dusk : <https://laravel.com/docs/11.x/dusk>

<sup>7</sup> GitHub Actions : <https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions>

<sup>8</sup>Laravel Dusk sur Github Actions : <https://laravel.com/docs/11.x/dusk#running-tests-on-github-actions>

### 3.6 Risques techniques

Plusieurs éléments techniques de ce projet risquent de présenter une difficulté plus marquée. L'élément le plus évident est la mise en place de l'algorithme permettant de distribuer les membres dans leurs différents groupes.

Un autre point technique est celui de l'implémentation d'un code QR qui redirige sur une page spécifique du site. Pour ce point particulier, des librairies devraient déjà exister pour Laravel. Une librairie qui pourrait être utilisée est « simple-qrcode<sup>9</sup> ».

## 4 RÉALISATION

---

### 4.1 Mise en place de Laravel 11

L'installation de Laravel<sup>10</sup> se fait à l'aide de l'outil en ligne de commande Composer\* qui permet de créer le projet et d'installer toutes les dépendances nécessaires au fonctionnement du Framework.

La création du projet se fait donc avec la commande suivante :

```
$ composer create-project Laravel/laravel SocialShuffle
```

#### 4.1.1 Le fichier .env

Le fichier .env contient les valeurs de configuration de l'application. Ces valeurs définissent notamment la connexion à la base de données ou bien encore le statut de l'application, c'est-à-dire si le site est lancé localement dans un environnement de développement ou sur un serveur de production.

Par défaut, ce fichier se trouve dans la liste des fichiers ignorés par Git car il contient des informations de configuration sensibles comme le mot de passe de connexion à la base de données. Ces informations ne doivent donc en aucun cas être rendues publique.

Afin d'adapter ces valeurs d'environnement à ce projet, il faut modifier certains champs comme le nom de l'application, sa langue et entrer les informations de connexion à la base de données.

---

<sup>9</sup> Tutoriel d'installation et utilisation de « simple-qrcode » : <https://www.akilischool.com/cours/laravel-generer-un-qr-code-avec-simple-qrcode>

<sup>10</sup> Installation de Laravel : <https://laravel.com/docs/11.x>

```
1 APP_NAME=SocialShuffle
2 APP_ENV=local
3 APP_KEY=base64:vnuMnfZnTYIJxuLPxJvf3xzaUaxyY3Ydd/xRoENT/Fk=
4 APP_DEBUG=true
5 APP_TIMEZONE=UTC
6 APP_URL=http://localhost
7
8 APP_LOCALE=fr
9 APP_FALLBACK_LOCALE=en
10 APP_FAKE_LOCALE=en_US
11
12 APP_MAINTENANCE_DRIVER=file
13 APP_MAINTENANCE_STORE=database
14
15 BCRYPT_ROUNDS=12
16
17 LOG_CHANNEL=stack
18 LOG_STACK=single
19 LOG_DEPRECATED_CHANNEL=null
20 LOG_LEVEL=debug
21
22 DB_CONNECTION=mysql
23 DB_HOST=127.0.0.1
24 DB_PORT=3306
25 DB_DATABASE=SocialShuffle
26 DB_USERNAME=root
27 DB_PASSWORD=root
```

11. Configuration initiale du fichier .env pour un environnement de développement. (En rouge, les valeurs qui ont été modifiées).

#### 4.1.2 Tester le fonctionnement de l'installation

Après avoir lancé le serveur de base de données locale, il faut effectuer une première migration afin de créer les tables natives au Framework dans la base de données. Pour cela, il faut lancer la commande suivante :

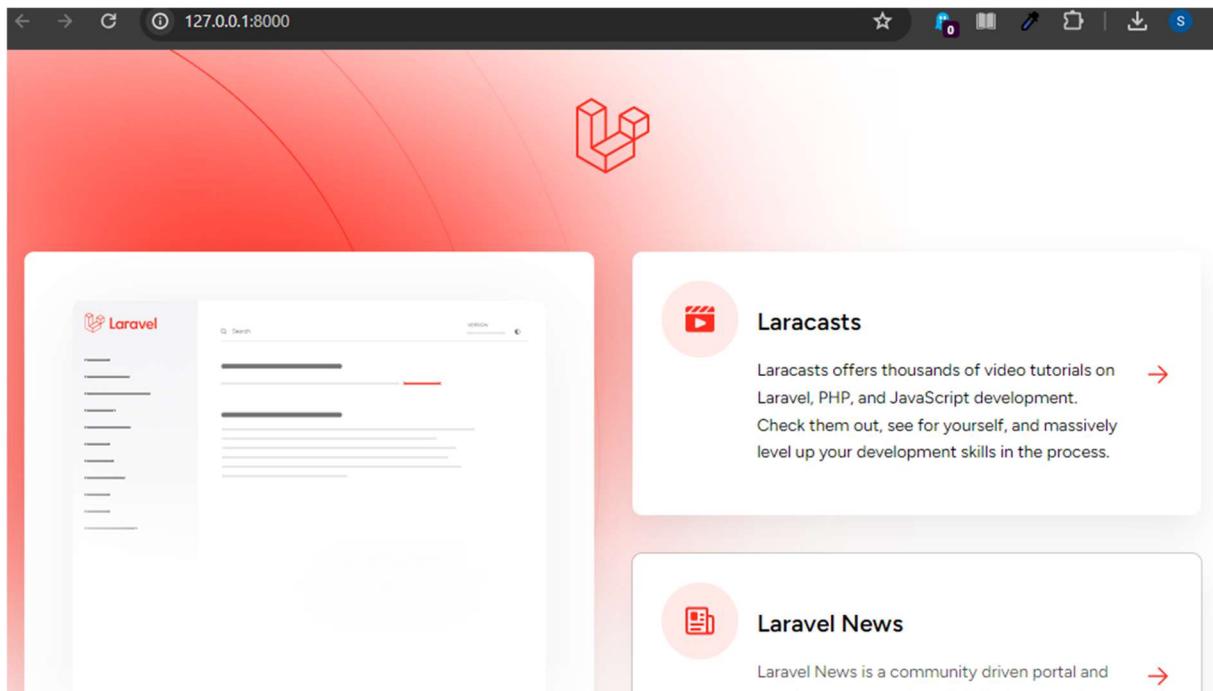
```
$ php artisan migrate
```

Il nous sera ensuite demandé si l'on souhaite créer la base de données (si celle-ci n'existe pas encore).

Une fois cela fait, on peut tester que Laravel ait été correctement installé en lançant le serveur de développement fourni avec Artisan, l'interface en ligne de commande fourni avec Laravel. Pour lancer le serveur, il faut lancer la commande suivante :

```
$ php artisan serve
```

On peut ensuite accéder au site depuis le navigateur et constater que Laravel fonctionne :



12. Page par défaut de laravel.

#### 4.1.3 Installation de Tailwind CSS

L'installation de Tailwind<sup>\*11</sup> se fait en installant les dépendances avec npm :

```
$ npm install -D tailwindcss postcss autoprefixer
```

Il faut ensuite générer les fichiers « tailwind.config.js » et « postcss.config.js » :

```
$ npx tailwindcss init -p
```

Une fois que le fichier « tailwind.config.js » a été créé, il faut ajouter les chemins pour tous les fichiers Template. Dans ce cas-ci, les fichier « .js » et « .vue » sont laissés afin de prévoir le cas où l'application sera maintenue et mise à jour après la réalisation de ce projet :

```
export default {  
    content: [  
        "./resources/**/*.blade.php",  
        "./resources/**/*.js",  
        "./resources/**/*.vue",  
    ],
```

Et pour terminer, il faut ajouter les directives suivantes dans le fichier app.css qui est présent par défaut à la création d'un projet Laravel. Ces directives permettent d'importer les éléments CSS de Tailwind :

```
@tailwind base;  
@tailwind components;  
@tailwind utilities;
```

<sup>11</sup> Installation de Tailwind : <https://tailwindcss.com/docs/guides/laravel#vite>

## 4.2 Migrations

Les tableaux qui suivent représentent les tables qui ont été créées après avoir lancé les migrations\*. Pour des raisons de contraintes techniques ainsi que pour respecter les bonnes pratiques, ce sont les conventions de nommage de Laravel qui sont utilisées<sup>12</sup>.

### 4.2.1 Table *t\_user* (*users*)<sup>13</sup>

Colonne	Type	Longueur	NULL	Autre	Valeur par défaut
id	BIGINT	-	Non	UNSIGNED AUTO_INCREMENT	-
username	String	-	Non	-	-
email	String	320	Non	-	-
password	String	-	Non	-	-
admin	Bool	1	Oui	-	NULL
created_at	DATE	-	Oui	-	NULL
updated_at	DATE	-	Oui	-	NULL

Cette table contient toutes les informations d'authentification des utilisateurs. La colonne **email** est limitée à 320 caractères afin de respecter la norme RFC 3696<sup>14</sup>.

### 4.2.2 Table *t\_team* (*teams*)

Colonne	Type	Longueur	NULL	Autre	Valeur par défaut
id	BIGINT	-	Non	UNSIGNED AUTO_INCREMENT	-
name	String	-	Non	-	-
nb_activities	INTEGER	-	Oui	-	NULL
group_size	INTEGER	-	Oui	-	NULL
user_id	BIGINT	-	Non	UNSIGNED AUTO_INCREMENT CASCADE ON DELETE	-

<sup>12</sup> Conventions de nommage de Laravel : <https://webdevetc.com/blog/laravel-naming-conventions/>

<sup>13</sup> Le nom de table entre parenthèses représente le nommage appliqué dans l'application qui respecte les noms de Laravel.

<sup>14</sup> Norme RFC 3696 : <https://www.rfc-editor.org/rfc/rfc3696#page-7>

created_at	DATE	-	Oui	-	NULL
updated_at	DATE	-	Oui	-	NULL

Cette table contient les données relatives aux équipes. La colonne **nbActivities** contient le nombre d'activités pour une équipe. La colonne **group\_size** contient le nombre de membre qu'il devrait y avoir dans une équipe. Dans la modélisation, cette colonne est appelée **nbMemberPerGroup** mais le nom a été changé afin de le rendre plus simple et explicite. Ces deux informations permettent de générer les groupes dans lesquels se trouveront les membres.

#### 4.2.3 Table *t\_member* (members)

Colonne	Type	Longueur	NULL	Autre	Valeur par défaut
id	BIGINT	-	Non	UNSIGNED AUTO_INCREMENT	-
firstname	String	-	Non	-	-
lastname	String	-	Non	-	-
email	String	320	Non	-	-
phone_number	String	-	Non	-	-
team_id	BIGINT	-	Non	UNSIGNED AUTO_INCREMENT CASCADE_ON_DELETE	-
created_at	DATE	-	Oui	-	NULL
updated_at	DATE	-	Oui	-	NULL

Cette table contient les données relatives aux membres d'une équipe. La colonne **team\_id** correspond à la clé étrangère qui lie le membre à une équipe.

#### 4.2.4 Table *t\_group* (groups)

Colonne	Type	Longueur	NULL	Autre	Valeur par défaut
id	BIGINT	-	Non	UNSIGNED AUTO_INCREMENT	-
generation	INTEGER	-	Non	-	-
team_id	BIGINT	-	Non	UNSIGNED AUTO_INCREMENT CASCADE_ON_DELETE	-

created_at	DATE	-	Oui	-	NULL
updated_at	DATE	-	Oui	-	NULL

Cette table correspond aux groupes auquel appartiendront les membres d'une équipe. La colonne **generation** permet de spécifier la génération groupe.

#### 4.2.5 Table *t\_dispatch* (*group\_member*)

Colonne	Type	Longueur	NULL	Autre	Valeur par défaut
member_id	BIGINT	-	Non	UNSIGNED AUTO_INCREMENT CASCADE_ON_DELETE	-
group_id	BIGINT	-	Non	UNSIGNED AUTO_INCREMENT CASCADE_ON_DELETE	-

Cette table est la table de pivot\* qui lie entre elles les tables **groups** et **members** dans leur relation « many-to-many ».

### 4.3 Mise en place des contrôleurs

Les contrôleurs, à l'exception du contrôleur d'authentification, seront des contrôleurs de ressources.

#### 4.3.1 Contrôleurs de ressources

Les contrôleurs de ressources offrent une structure de code prédéfinie une fois créés. La création d'un contrôleur de ressource se fait avec la commande suivante :

```
$ php artisan make:controller TeamController --resource
```

Une fois exécuté, un nouveau fichier est créé sous **app/http/Controllers**. Ce fichier contient les fonctions présentes par défaut qui permettent de réaliser les opérations CRUD sur la ressource. Les fonctions par défaut sont les suivantes :

- **index()** : Affiche un listing de la ressource.
- **create()** : Affiche le formulaire pour créer une nouvelle ressource.
- **store()** : Stock la nouvelle ressource dans la base de données.
- **show()** : Affiche une ressource spécifique.
- **edit()** : Affiche le formulaire pour modifier une ressource spécifique.
- **update()** : Modifie une ressource spécifique dans la base de données.
- **destroy()** : Supprime une ressource spécifique de la base de données.

Laravel met à disposition des méthodes pour créer des enregistrements en base de données. La création et la modification se fera par l'assignement de masse qui permet de créer ou modifier plusieurs attributs d'un modèle avec une seul ligne de code.

Cette méthode présente un risque de sécurité car il est possible, si il n'y avait pas de protections, d'insérer des attributs à modifier dans une requête HTML. Pour contrer cela, il faut définir les attributs qui sont autorisés pour l'assignement de masse<sup>15</sup> dans le modèle correspondant en créant une propriété nommée **\$fillable** :

```
protected $fillable = [  
    'name',  
    'nbActivities',  
    'nbMemberPerGroup',  
    'user_id',  
];
```

Avec cela, il est désormais possible de créer modèles avec plusieurs attributs en une ligne de code. Voici un exemple avec la création d'une équipe :

```
public function store(TeamRequest $request)  
{  
    $validatedName = $request->validated();  
    $team = Team::create($validatedName);  
    return ...;  
}
```

## 4.4 Mise en place des routes

Laravel met à disposition un système de routes permettant d'exécuter des instructions en fonction d'une URL spécifique et d'un type de requêtes. Les routes sont définies dans le répertoire « routes » dans le fichier web.php. Laravel offre la possibilité de nommer les routes afin de les catégoriser et de les grouper. Les routes de ressources n'ont pas besoin d'être nommées car cela est fait automatiquement.

Le lien entre les routes et les contrôleurs des ressources se fait en spécifiant la classe correspondant à la route. Pour ce qui est des autres contrôleurs, il faut également préciser la classe du contrôleur, mais également le nom de la méthode à exécuter.

<sup>15</sup> Assignement de masse : <https://laravel.com/docs/11.x/eloquent#mass-assignment>

Voici la version finale des routes qui ont été mises en place pour le fonctionnement de l'application :

```
// root
Route::get('/', function () {
    return redirect()->route('team.index');
});

// Team algorithm (Additional routes to a resource controller must be defined before calling the resource route)

Route::post('import-csv/{team}', [TeamController::class, 'importCSV'])
    ->name('team.importCSV');

Route::get('groupForm/{team}', [TeamController::class, 'groupForm'])
    ->name('team.groupForm');

Route::post('createGroups/{team}', [TeamController::class,
    'generateGroups'])
    ->name('team.createGroups');

Route::get('Activity/{team}', [TeamController::class, 'showActivity'])
    ->name('team.showActivity');

// resources
Route::resource('team', TeamController::class);
Route::resource('team.members', MemberController::class);

// Authentication
Route::get('login', [AuthController::class, 'login'])
    ->name('auth.login');

Route::post('login', [AuthController::class, 'applyLogin'])
    ->name('auth.applyLogin');

Route::get('logout', [AuthController::class, 'logout'])
    ->name('auth.logout');

// About page
Route::get('about', function(){
    return view('about');
})->name('about');
```

#### 4.4.1 La route « / » (Racine)

Cette route a pour seul fonction de rediriger la requête vers la page d'index correspondant aux équipes. C'est là où sera affichée la liste de toutes les équipes ou les équipes appartenant à l'utilisateur authentifié.

#### 4.4.2 Ressources

Des routes de type « resource » ont été mises en place. Les ressources en Laravel permettent de créer automatiquement les routes nécessaires à la mise en place des opérations CRUD sur un modèle Eloquent. Cela permet d'éviter d'avoir à créer un nombre important de routes qui se répètent pour chaque modèle. Les routes de ressources s'appliquent pour les **équipes**, les **membres** et les **groupes**.

La table des **membres** s'appelle **team.member** en raison de la contrainte qui lie un membre à son équipe. On parle de ressource imbriquée.<sup>16</sup>

#### 4.4.3 Authentification

La partie authentification contient trois routes : la première pour appeler la méthode qui affiche le formulaire de connexion, la deuxième qui redirige la requête vers la méthode de validation une fois le formulaire validé et la troisième qui redirige sur la méthode qui permet de déconnecter l'utilisateur en regénérant une nouvelle session.

#### 4.4.4 À propos

Une simple route exécutant une fonction qui appelle la vue contenant des informations à propos de l'application.

### 4.5 Mise en place des vues

#### 4.5.1 Layout

Une bonne pratique mentionnée par Piotr Jura<sup>17</sup>, formateur sur la plateforme Udemy, est l'utilisation d'un fichier Blade contenant la structure HTML de base avec des éléments visuels redondants, comme une barre de navigation ou un footer. Afin de pouvoir ajouter du contenu à ce fichier, il faut définir les sections dans lesquelles il est possible d'ajouter du code. Cela est possible avec la directive Blade **@yield('nom\_de\_la\_section')**.

Ce layout peut ensuite être appelé depuis n'importe quel fichier à l'aide de la directive **@extends('chemin\_du\_fichier')**. Puis, pour ajouter du contenu aux sections définies dans le layout, il faudra utiliser la directive **@section('nom\_de\_la\_section')**.

L'utilisation de cette méthode de création des vues rend les fichiers légers, lisibles et peu redondants, ce qui facilitera la maintenance de l'application dans le futur. Par exemple, si un

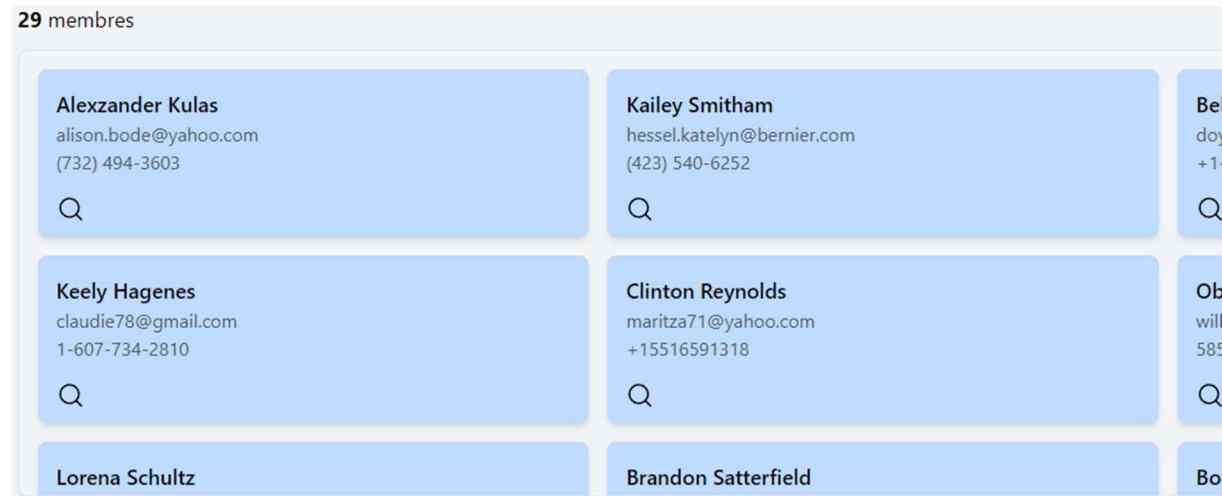
<sup>16</sup> Ressources imbriquées : <https://laravel.com/docs/11.x/controllers#restful-nested-resources>

<sup>17</sup> Formation Udemy sur Laravel 10, Piotr Jura : <https://www.udemy.com/course/laravel-beginner-fundamentals/>

élément de la barre de navigation venait à changer, il suffirait d'effectuer la modification sur un seul fichier.

#### 4.5.2 Composant réutilisable pour l'affichage des membres

L'affichage des membres lors de l'ajout des membres dans l'équipe et sur la page où sont affichés les détails de l'équipe se fait sous la forme d'une collection de cartes que voici :



13. Affichage de la collection des membres.

Ce composant devant être présent à deux endroits, il a été décidé de déplacer le code associé à l'affichage de la collection des membres dans un seul fichier appelé **membersComponent.blade.php** qui se situe dans le répertoire **layouts** et qui peut désormais être appelé depuis n'importe où grâce à la ligne de code suivante utilisant une directive Blade :

```
@include('layouts.membersComponent')
```

Si l'affichage de la liste des membres venait à devoir être changée, seul le fichier de ce composant devra être modifié.

#### 4.5.3 Utilisation du même formulaire pour la création et la modification

L'application propose trois formulaires pour la gestion des équipes. Pour optimiser le code, les fonctionnalités de création et de modification ont été fusionnées en un seul formulaire par action.

Le mécanisme qui détermine si les fonctionnalités de création ou de modification doivent être affichées repose sur la présence ou l'absence de données existantes.

Ce mécanisme est réparti sur trois principaux éléments du formulaire :

1. L'action du formulaire, c'est-à-dire la route à emprunter une fois le formulaire soumis.

```
<form action="{{ isset($team) ? route('team.update', ['team' => $team]) : route('team.store') }}" method="POST">
```

Dans cet exemple, selon si la variable **\$team** existe ou non, ce sera la route liée à la modification ou à un nouvel enregistrement dans la base de données qui sera utilisée.

2. La méthode de la requête à utiliser lorsque le formulaire soumis (POST, PUT).

```
@isset($team)
    @method('PUT')
@endisset
```

La méthode par défaut du formulaire est POST mais avec Laravel il est possible d'utiliser d'autres types de méthodes. Dans cet exemple, si la variable **\$team** existe, ce sera alors la méthode PUT qui sera utilisée au moment de la soumission du formulaire.

3. L'affichage ou non des données déjà existantes.

```
<input type="text" ... value="{{ $team->name ?? old('name') }}" class="...">
```

L'existence de la propriété **\$team->name** est vérifiée. Si elle existe, on utilise la fonction **old()** qui permet de récupérer automatiquement l'ancienne valeur présente dans la requête précédente ayant été automatiquement enregistrée dans la session<sup>18</sup>.

#### 4.5.4 Affichage des groupes

Lorsque l'utilisateur accède à la page d'une équipe, il peut voir les groupes pour autant qu'ils aient été générés.

Voici comment les groupes sont affichés :

The screenshot shows a web browser displaying the 'Détails de l'équipe' (Team Details) page at the URL 127.0.0.1:8000/team/2. The page has a dark header with the SocialShuffle logo, a navigation bar with 'Liste des équipes', 'À propos', and '+ Nouvelle équipe', and a user icon 'jdof'. The main content area has a light gray background. It displays two sections of hierarchical groups:

- Generation 1:** Five blue boxes labeled 1 through 5, each containing a list of names.
  - Box 1: Steuber Noemy, Smith Selmer, Von Carolyne, Stiedemann Karli, Batz Zackery, Walsh Dovie
  - Box 2: Swift Dianna, Rath Lelia, Upton Maxwell, Torp Darrel, Dooley Wilma, Ortiz Theodora
  - Box 3: Brakus Oscar, Von Lila, Walker Winston, Runolfsdottir Mae, Beatty Rodrigo, Goodwin Dylan
  - Box 4: Tremblay Ralph, Fay Payton, Heidenreich Madisen, Parisian Valerie, Hahn Wiley, Mayer London
  - Box 5: Jast Dora, Beier Susanna, Mraz Joanny
- Generation 2:** Five blue boxes labeled 1 through 5, each containing a list of names.
  - Box 1: Upton Maxwell, Beier Susanna, Mayer London
  - Box 2: Batz Zackery, Parisian Valerie, Heidenreich
  - Box 3: Walsh Dovie, Stiedemann Karli, Fay Payton
  - Box 4: Tremblay Ralph, Swift Dianna, Steuber Noemy
  - Box 5: Ortiz Theodora, Runolfsdottir Mae, Von Lila

<sup>18</sup> Récupération d'anciennes valeurs : <https://laravel.com/docs/11.x/requests#retrieving-old-input>

La répartition des cartes se fait dans une grille dont le nombre de colonnes change en fonction de la largeur de l'écran afin de respecter les contraintes de responsivité de l'application.

#### 4.5.5 Difficultés rencontrées

L'affichage des groupes a présenté une difficulté particulière en raison du fait que l'affichage se fait dynamiquement en fonction de plusieurs éléments que sont les membres, les groupes et la génération à laquelle appartient un groupe. Cela a entraîné la mise en place relativement complexe dans l'utilisation de boucles et de conditions avec les directives Blade.

Il a fallu notamment suivre la valeur actuelle de la génération pour afficher les bons groupes ce qui permet de dynamiquement fermer des **div** en fonction de la valeur en cours de la génération.

Lorsque les données des groupes sont envoyées à la vue, elles sont triées en fonction de leur numéro de génération.

## 4.6 Formulaires de création d'une équipe avec les membres

La création des équipes a été séparée en trois formulaires :

1. Entrée du nom de l'équipe qui permet d'enregistrer une nouvelle équipe dans la base de données
2. Ajout des membres.
3. Entrée des informations concernant la répartition des membres dans les différents groupes.

La raison de ce choix vient du fait qu'un groupe ne peut pas être créé sans membres et les membres ne peuvent être créés sans équipe existante. De plus, ce choix permet une meilleure modularité car l'utilisateur peut choisir de créer une équipe sans en terminer la configuration.

Voici les formulaires par lesquels passera l'utilisateur s'il souhaite créer une équipe complète :

The screenshot shows a mobile application interface. At the top, there is a dark header bar with three horizontal lines on the left, a user icon in the center, and the text 'jdo' on the right. Below this is a light-colored card with the title 'Créer une nouvelle équipe' in bold. Inside the card, there is a text input field containing 'Équipe 1'. Below the input field, a red error message reads 'Ce champ est obligatoire'. At the bottom right of the card is a blue rectangular button with the white text 'Suivant'.

14. Formulaire d'ajout du nom lors de la création d'une équipe en version mobile.

19

<sup>19</sup> Ici les messages d'erreurs ont été volontairement affiché à des fins de démonstration.

Ajouter les membres

Prénom  
Ce champ est obligatoire

Nom  
Ce champ est obligatoire

Email  
Ce champ est obligatoire

Numéro de téléphone  
Ce champ est obligatoire

+ Ajouter un membre      Suivant →

3 membres

John Doe  
jdoe@gmail.com  
0790000000

Jane Doe  
Jane@yahoo.com  
0760000000

Jeff Doe  
jdof@gmail.com  
0790000000

15. Démonstration du formulaire d'ajout des membres.

## 4.7 Authentification

### 4.7.1 Formulaire d'authentification

Le formulaire d'authentification est très simple. Il comporte un champ pour l'entrée du nom d'utilisateur ou de son email et un second champ qui est destiné à l'entrée du mot de passe.

Se connecter

Email ou Nom d'utilisateur

Mot de passe

Se connecter

16. Formulaire d'authentification

### 4.7.2 Traitement de l'authentification

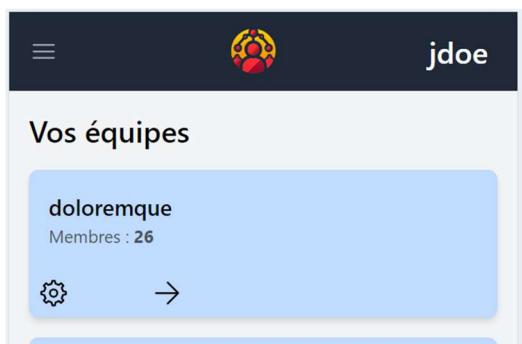
L'authentification des utilisateurs est gérée depuis le contrôleur **AuthController** et doit pouvoir accepter à la fois les noms d'utilisateurs et les emails. Pour cela, il faut trouver dans la base de données une correspondance entre ce que l'utilisateur a entré et la base de données.

```
$user = User::where('email', $login)
    ->orWhere('username', $login)->first();
```

Une fois l'utilisateur trouvé, il faut tenter de l'authentifier avec l'email puis le nom d'utilisateur séparément ce qui donne cela :

```
// Attempt login with the email, then with the username
if(Auth::attempt(['email' => $user->email,
                  'password' => $request->input('password')]) ||
   Auth::attempt(['username' => $user->username,
                  'password' => $request->input('password')]))
```

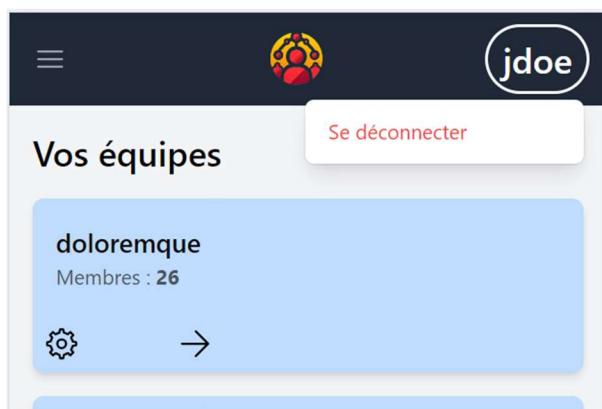
Une fois l'utilisateur connecté, son nom d'utilisateur est affiché en haut à droite de son écran :



17. Affichage du nom d'utilisateur.

#### 4.7.3 Déconnexion

Pour se déconnecter, l'utilisateur doit cliquer sur son nom d'utilisateur. De là s'affichera un menu ou l'option « Se déconnecter » apparaîtra :



18. Affichage du bouton "Se déconnecter".

### 4.8 Droits des utilisateurs

Laravel possède plusieurs outils permettant la mise en place de règles permettant d'autoriser des actions<sup>20</sup>. Ces autorisations permettent de définir si un utilisateur authentifié a le droit

<sup>20</sup> Autorisations avec Laravel : <https://laravel.com/docs/11.x/authorization#creating-policies>

d'effectuer une action, comme modifier un modèle Eloquent spécifique. Une des manière de mettre en place ces autorisation est l'utilisation des Policies<sup>21</sup>.

#### 4.8.1 Crédation d'une Policy

Afin de créer une Policy<sup>21</sup> sur un modèle en particulier, voici la commande Artisan à utiliser (ici il s'agit d'une Policy s'appliquant aux équipes) :

```
$ php artisan make:policy TeamPolicy --model=Team
```

Après cela, un nouveau fichier sera créé sous **app/Policies** contenant déjà des fonctions prédéfinies. Ces fonctions servent à retourner un booléen après avoir vérifié si l'utilisateur spécifié a le droit d'effectuer une action sur la ressource.

Pour les équipes, il faut vérifier les droits sur la modification et la suppression. Voici comment cela a été mis en place :

```
/**  
 * Determine whether the user can update the model.  
 */  
public function update(User $user, Team $team): bool  
{  
    return $user->id == $team->user_id || $user->admin;  
}  
  
/**  
 * Determine whether the user can delete the model.  
 */  
public function delete(User $user, Team $team): bool  
{  
    return $this->update($user, $team);  
}
```

Dans le cas de **delete**, on fait appel à la fonction **update** car la règle de vérification est identique.

En ce qui concerne les autres fonctions qui ont été créées automatiquement, elles ont été retirées car elles n'ont pas d'utilité dans l'état actuel de l'application.

#### 4.8.2 Utilisation des Policies dans les vues

Afin de déterminer s'il faut afficher certains éléments ou non il est possible d'accéder aux autorisations avec la directive Blade **can()**. Voici comment cela a été mis en place pour déterminer l'affichage du bouton permettant de supprimer un utilisateur :

<sup>21</sup> Créer des Policies : <https://laravel.com/docs/11.x/authorization#creating-policies>

```
@can('delete', $team)
    <form action="{{ route('team.destroy', ['team' => $team]) }}" method="POST" class="...">
        @csrf
        @method('DELETE')
        <label>
            <input type="submit" class="hidden">
        </label>
        <svg ...>
    </form>
@endcan
```

#### 4.8.3 Utilisation des Policies dans les contrôleurs

Dans le cas où un utilisateur non-autorisé parvenait à envoyer une requête afin de modifier ou supprimer une ressource, une vérification supplémentaire a été ajoutée dans les fonctions permettant la modification et la suppression. Voici une démonstration avec le cas de la modification<sup>22</sup> :

```
public function update(TeamRequest $request, Team $team)
{
    if($request->user()->cannot('update', $team)){
        return abort(403);
    }
    ...
```

Le fonctionnement ressemble à celui utilisé dans la directive Blad, il faut récupérer l'utilisateur qui a émis la requête et vérifier s'il ne possède pas le droit de réaliser une certaine action. Dans le cas où il ne possède pas le droit, il faut renvoyer une erreur 403 qui correspond à une interdiction d'accès.

## 4.9 Algorithme de répartition des membres

### 4.9.1 Pistes d'améliorations

Refactoriser le code en séparant les éléments de logique pour une meilleure facilité de lecture et de maintenance.

## 4.10 Affichage du QR Code

## 4.11 Procédure utilisée pour le déploiement

### 4.11.1 Contraintes

<sup>22</sup> Provient de la documentation officielle : <https://laravel.com/docs/11.x/authorization#via-the-user-model>

N'ayant pas d'accès SSH au serveur, il va être impossible d'exécuter des commandes artisan liées à l'optimisation, notamment la mise en cache de certains éléments comme les routes.

Il existe cependant une solution qui est la mise en place de routes permettant d'exécuter des commandes Artisan en accédant à une certaine URL. Ces routes ne devront donc pas rester indéfiniment car elles présentent un risque de sécurité évident, elles seront présentes sur la version de production uniquement durant le temps du déploiement et à des fins de tests.

#### 4.11.2 Procédure

Afin de ne pas endommager la version de développement du projet, une copie du répertoire est effectuée afin de réaliser les manipulations nécessaires à la mise en production.

##### 4.11.2.1 Modification du fichier d'environnement (.env)

L'affichage ou non des erreurs et les informations de connexion à la base de données sont définis par les valeurs définies dans les variables présentes dans le fichier .env.

Afin de pourvoir mettre l'application en production, il faut donc modifier plusieurs valeurs, en voici la liste :

- **APP\_ENV=production**
- **APP\_DEBUG=false** (ou true temporairement afin de pouvoir facilement débuger en cas de besoin)
- **APP\_URL=[URL réel du site]**
- **DB\_CONNECTION=[Type de connexion pour la base de données, pour ce projet, ce sera mysql]**
- **DB\_HOST=[Adresse IP du serveur de base de données]**
- **DB\_PORT=[Port permettant d'accéder à la base de données, généralement 3306]**
- **DB\_DATABASE=[nom de la base de données]**
- **DB\_USERNAME & DB\_PASSWORD=[valeurs d'authentification à la base de données].**

#### 4.11.2.2 Suppression des dépendances liées au développement

Afin d'éviter d'importer des dépendances inutiles en production, il faut veiller à les supprimer correctement. Pour cela, il faut lancer la commande suivante :

```
$ composer install --no-dev
```

Cette commande permet d'installer uniquement les dépendances nécessaires à la production<sup>23</sup>. On ne retrouvera donc pas les outils comme les Factories.

#### 4.11.2.3 Compiler le CSS

Afin de compiler le CSS utilisé avec tailwind dans un seul fichier, il faut utiliser la commande suivante :

```
$ npm run build
```

Cela permet de créer dans le dossier **public**, un répertoire nommé **build** dans lequel se trouve un fichier .css et .js.

#### 4.11.2.4 Préparation de la base de données

Maintenant que le fichier d'environnement est configuré avec les informations de connexion à la base de données de production, il est possible d'exécuter une migration sur la base de données avec artisan depuis l'environnement local. Il suffit pour cela d'exécuter la commande suivante :

```
$ php artisan migrate
```

#### 4.11.2.5 Éléments à vérifier avant de transférer les fichiers sur le serveur.

Il est important de vérifier deux points importants du serveur afin de s'assurer du bon fonctionnement de l'application :

1. La bonne version de PHP est installée (pour la version 11 de Laravel, il faut au minimum la version 8.2 de PHP).
2. Le serveur effectue une redirection sur le dossier **public** de Laravel car le fichier **index.php** du site s'y trouve. C'est le point d'entrée de l'application.

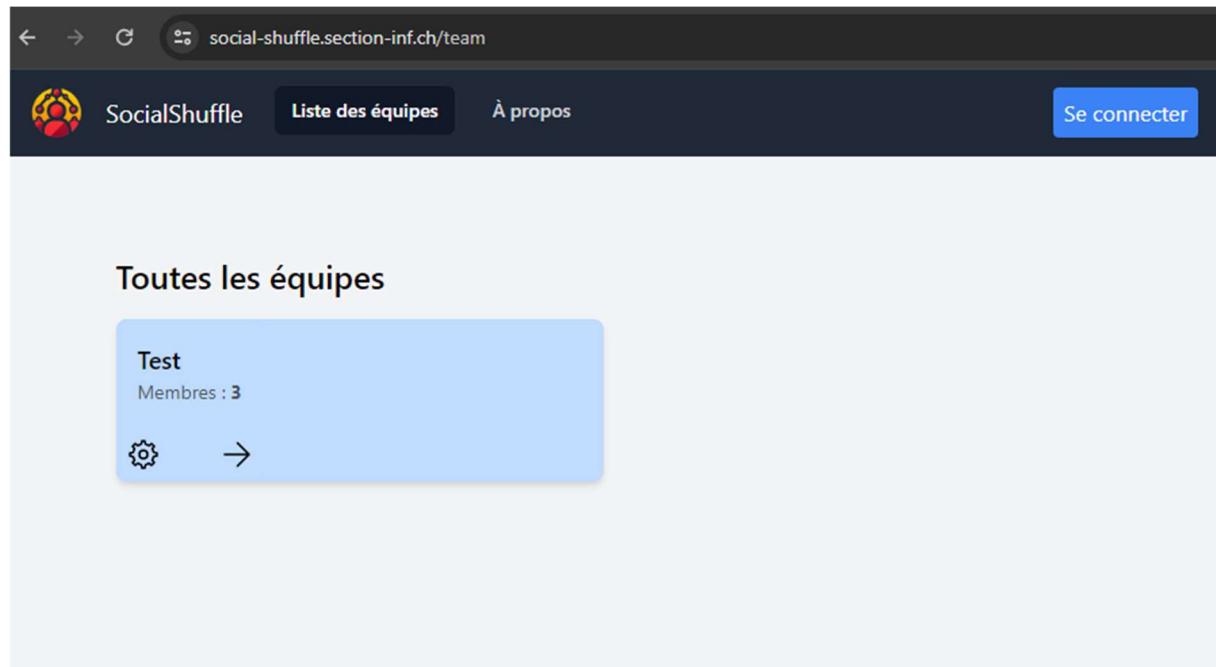
#### 4.11.2.6 Transfer des fichiers

Avec un accès FTP au serveur il va être possible de transférer le projet une fois prêt pour la production. Pour cela il suffit d'effectuer un copier-coller de tous les fichiers de l'application à la racine du site. Il faut faire attention au temp de transfert qui peut parfois être long. Pour cette

<sup>23</sup> Mise en production d'un projet Laravel sans accès SSH :

<https://laracasts.com/discuss/channels/laravel/installing-laravel-on-server-without-ssh>

application, le transfert prend environ 20 minutes. Si le transfert s'est fait sans erreurs, le site devrait être accessible :



#### 4.11.3 Exécuter des commandes Artisan depuis une URL

Comme mentionné précédemment, il est possible d'exécuter des commandes Artisan en accédant à une URL spécifique. Voici comment cela a été mis en place pour cette application :

```
Route::get('cache', function(){
    Artisan::call('optimize:clear');
    Artisan::call('optimize');
});

Route::get('routes', function(){
    Artisan::call('route:clear');
    Artisan::call('route:cache');
});
```

Pour des raisons de sécurité, ces routes devront être supprimées une fois qu'elles ne sont plus utilisées.

## 4.12 Trois mesures de sécurité

### 4.12.1 Token CSRF

Les attaques CSRF permettent l'exécution de requêtes au travers d'utilisateurs enregistrés ayant souvent des droits élevés sur les sites. Pour contrer cela, Laravel met à disposition une manière simple d'implémenter la protection CSRF. Pour chaque session d'utilisateurs, un jeton (token) CSRF est automatiquement généré. Ce jeton permet de s'assurer qu'il s'agit de l'utilisateur authentifié est bien la personne à l'origine de la requête.

La mise en place de ce jeton se fait à chaque formulaire HTML de type POST, PUT, PATCH ou DELETE. Il est ensuite nécessaire d'ajouter la directive Blade `@csrf` au tout début du formulaire sous peine d'avoir une erreur. Le Framework oblige donc les développeurs à protéger les formulaires de leurs applications contre ce type d'attaques.

```
<form action="{{ ... }}" method="POST">
    @csrf
    ...
</form>
```

#### 4.12.2 Validation des formulaires

La validation des formulaires se fait par la validation de règles définies pour une requête particulière. Une fois le formulaire soumis, les entrées de l'utilisateur seront vérifiées pour s'assurer qu'elles respectent les règles définies.

Voici un exemple avec les règles liées à la création des membres :

```
class MemberRequest extends FormRequest
{
    ...
    public function rules(): array
    {
        return [
            'firstname' => 'required|string',
            'lastname'   => 'required|string',
            'email'      => 'required|email',
            'phoneNumber' => 'required|min:9|numeric',
        ];
    }
}
```

Le déclenchement de cette vérification se fait dans la méthode du contrôleur correspondant à la validation. Dans ce cas il s'agit de la création et la modification d'un membre. La validation est appelée de la manière suivante :

```
/**
 * Store a newly created resource in storage.
 */
public function store(MemberRequest $request, Team $team)
{
    // Validate the request containing the new member data
    $data = $request->validated();
    ...
    return ...;
}
```

Si l'utilisateur a entré une information erronée, cela déclenchera une erreur. Selon la règle qui a été enfreinte, le message d'erreur sera différent. Ces messages se trouvent dans le répertoire **/lang** qui est accessible après avoir entré la commande suivante :

```
$ php artisan lang:publish24
```

Il est ensuite possible de définir plusieurs langues pour le site en copiant le dossier en anglais par défaut et en traduisant les messages.

```
/lang  
/en  
    messages.php  
/fr  
    messages.php
```

Pour afficher les messages d'erreur dans les vues, il faut ajouter la directive Blade **@error** qui est automatiquement déclenchée quand une erreur est retournée à la vue. Il est ensuite possible d'afficher le message d'erreur avec la variable **\$message**. En ce qui concerne la récupération de la valeur précédente, elle se fait avec l'utilisation de la fonction **old()** après avoir vérifié que la ressource a été envoyée à la vue:

```
<input type="text" name="name" id="name" placeholder="Nom de l'équipe"  
      value="{{ $team->name ?? old('name') }}>  
@error('name')  
    <p class="text-red-500">{{ $message }}</p>  
@enderror
```

#### 4.12.3 Hachage des mots de passe

Bien que l'application ne permette pas aux utilisateurs de créer leurs comptes, des comptes par défaut ont été créés grâce aux Factories. Afin de garantir la sécurité de ces comptes, les mots de passe ont été hachés.

Laravel met à disposition la classe **Hash** permettant de gérer tout ce qui est lié au hachage. Cette classe utilise la librairie Bcrypt bien qu'il soit possible de la configurer afin d'utiliser d'autres drivers tels que **argon** ou **argon2id**.<sup>25</sup>

La mise en place du hachage lors de la création des comptes a été faite de la manière suivante :

```
'password' => Hash::make('exempleDeMotDePasse'),
```

<sup>24</sup> Mise en place de plusieurs langues dans un projet Laravel :

<https://laravel.com/docs/11.x/localization>

<sup>25</sup> Hachage avec Laravel : <https://laravel.com/docs/11.x/hashing>

## 4.13 Trois mesures pour le « Responsive design »

Afin de répondre aux exigences de responsivité de l'application, le Framework Tailwind a été utilisé pour la mise en page de l'application. Ce Framework permet de mettre en place un design responsive très rapidement avec peu de code.

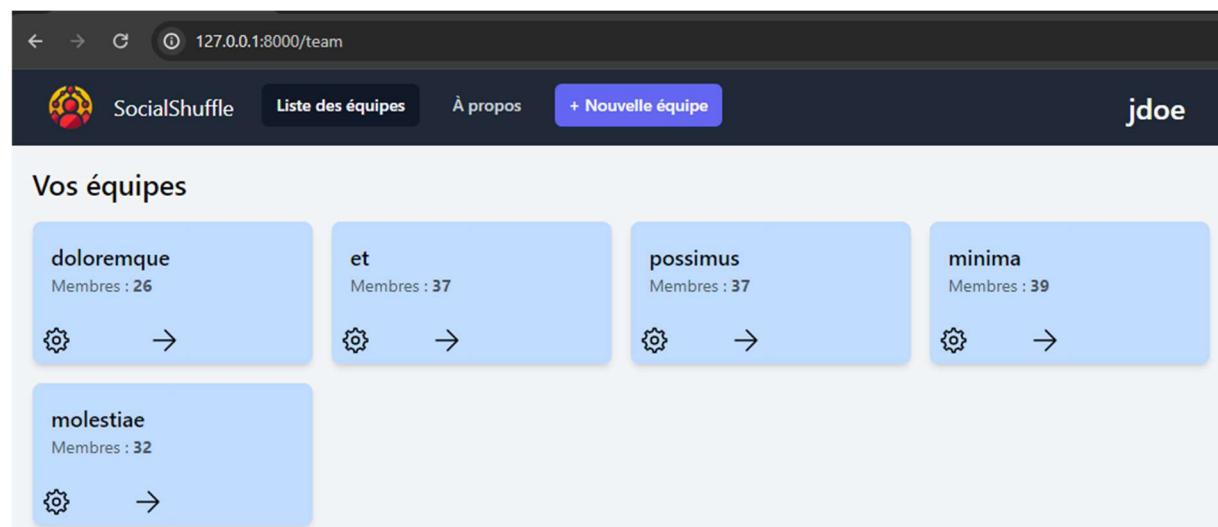
### 4.13.1 Affichage des cartes en fonction de la largeur de l'écran

Un bon exemple pour démontrer la mise en place d'un design responsive avec Tailwind est la manière dont les cartes contenant les équipes enregistrées sont disposées dans une grille dont le nombre de colonnes change en fonction de la largeur de la page :

```
<div class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-4">
```

La classe par défaut **grid-cols-1** est destinée aux écrans mobiles, elle permet d'afficher une seule carte par ligne. Les deux classes restantes sont destinées aux écrans plus larges.

Voici la mise en pratique de ce principe sur la version Desktop et mobile :



19. Affichage de l'écran principal en version Desktop.

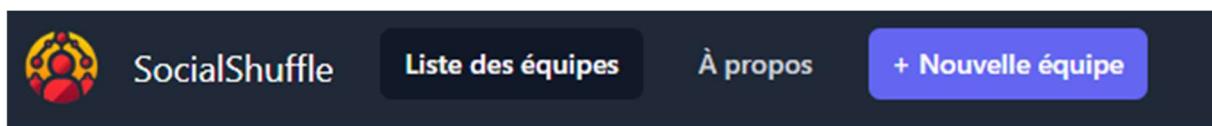


20. Version mobile de l'écran principal

#### 4.13.2 Barre de navigation

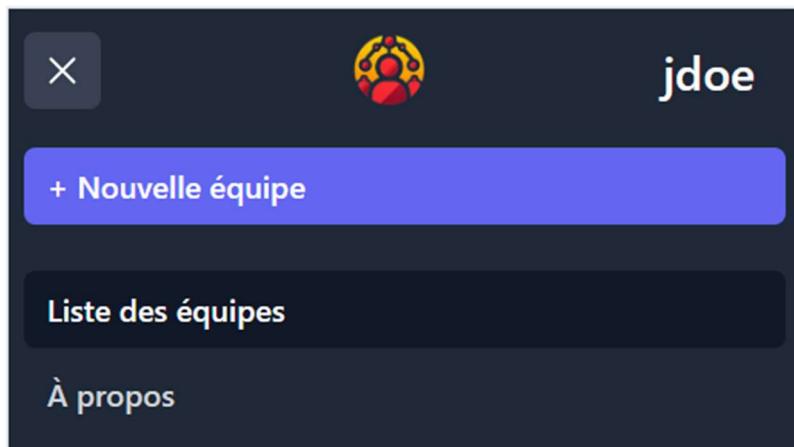
En raison de la largeur réduite de l'écran d'un smartphone, il était difficile de placer tous les éléments de la barre de navigation sur l'écran. Pour remédier à cela, un Template de barre de navigation responsif fourni sur une plateforme de Tailwind<sup>26</sup> a été utilisé. Ce Template permet d'afficher une barre de navigation au-dessus de l'écran pour la version Desktop et dans un menu séparé dans la version mobile.

Voici une comparaison entre les deux versions de la barre de navigation :



21. Barre de navigation Desktop

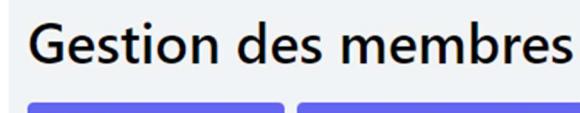
<sup>26</sup> Template de barre de navigation : <https://tailwindui.com/components/application-ui/navigation/navbars>



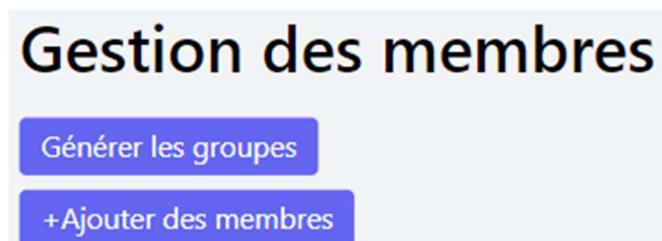
22. Barre de navigation mobile

#### 4.13.3 Boutons de gestion des membres

Sur la page permettant d'afficher les groupes et les membres, deux boutons permettent respectivement de Générer des groupes et d'ajouter des membres supplémentaires à l'équipe. Sur un écran large, ces boutons sont disposés sur la même ligne et sur les écrans mobiles, ils sont placés l'un au-dessus de l'autre :



23. Boutons de gestion des membres en version Desktop.



24. Boutons de gestion des membres au format mobile.

### 4.14 Mise en place des tests Laravel Dusk dans GitHub Actions

#### 4.14.1 Installation de Laravel Dusk<sup>27</sup>

L'installation de Laravel Dusk se fait uniquement dans un environnement de développement. La commande suivante permet d'installer Laravel Dusk dans le projet.

```
$ composer require laravel/dusk --dev
```

<sup>27</sup> Installation de Laravel Dusk : <https://laravel.com/docs/11.x/dusk#installation>

Laravel Dusk étant un outil de test End-To-End, il doit pouvoir accéder à un navigateur Web. Pour cela, il faut installer le pilote Chrome (Chrome Driver) avec la commande suivante :

```
$ php artisan dusk:install
```

Désormais, un nouveau répertoire nommé **Browser** a été créé et contient un exemple de test simple qui permet de visiter la racine du site et vérifier qu'un certain texte est présent sur la page.

Voici le code du test légèrement modifié :

```
public function testBasicExample(): void
{
    $this->browse(function (Browser $browser) {
        $browser->visit('/')
            ->assertSee('SocialShuffle');
    });
}
```

Ce test est lancé afin de s'assurer de la bonne installation de Laravel Dusk :

```
$ php artisan dusk
```

On peut ensuite constater sur le terminal que le test s'est exécuté sans problèmes :

Avant de lancer le test, il faut s'assurer que l'adresse du site (APP\_URL) dans le fichier d'environnement, soit sous forme de IP (<http://127.0.0.1:8000>), sinon une erreur sera retournée lors de l'exécution du test.

Une fois la commande exécutée, on peut constater sur le terminal que le test s'est exécuté sans problèmes :

```
PASS Tests\Browser\ExampleTest
  ✓ basic example
```

#### 4.14.2 Mise en place de GitHub Action

La configuration d'un Workflow GitHub Actions se fait dans sur fichier **.yml** contenant la procédure à suivre pour la mise en place d'un environnement de test virtuel. Laravel fournit un exemple de ce fichier qui peut servir de base pour la configuration de tests<sup>28</sup>.

Dans le cas de SocialShuffle, les tests seront exécutés à chaque push et se feront dans un Container Ubuntu sur laquelle sera installé un nouvel environnement propre pour l'exécution de l'application.

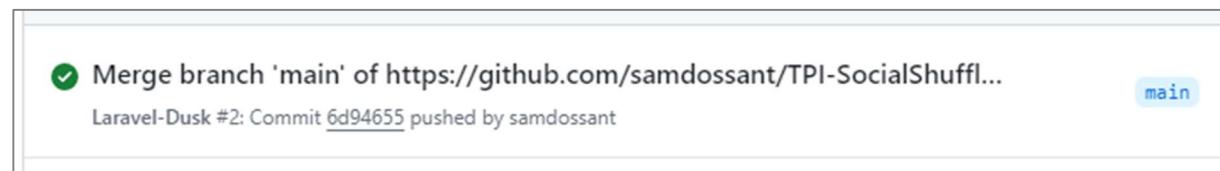
Le fichier **.yml** utilisé pour ce projet provient d'un autre projet réalisé avant le TPI. Il a fallu cependant adapter certaines valeurs pour qu'elles correspondent avec le projet actuel.

---

<sup>28</sup> Exemple de fichier **.yml** : <https://laravel.com/docs/11.x/dusk#running-tests-on-github-actions>

Pour mettre en place le workflow sur GitHub, il faut se rendre dans l'onglet **Actions** depuis lequel on peut créer le fichier .yml.

Une fois le fichier ajouté au projet, on peut essayer la mise en place des tests en faisant un push sur le repository, ce qui déclenchera le test configuré précédemment.



25. Test automatisé réussi sur GitHub Actions.

#### 4.14.3 Création d'un test Laravel Dusk

Dans ce cette partie, est détaillée la mise en place du test **CreateTeamTest** qui permet de tester la création d'une équipe au complet.

Pour créer le fichier contenant le test, il faut exécuter la commande suivante :

```
$ php artisan dusk:make testCreateTeam
```

Une fois terminé, un nouveau fichier est créé dans le dossier **Browser**.

Voici le code en détails comment ce test a été mis en place :

```
class CreateTeamTest extends DuskTestCase
{
    use DatabaseMigrations;
    ...
```

L'utilisation de **use DatabaseMigrations** permet de lancer une migration à chaque nouveau test. Cela permet de s'assurer que chaque test est effectué avec des données propres.

```
public function testCreateTeam(): void
{
    // Arrange
    $user = User::factory()->create([
        'username' => 'jDoe',
        'email' => 'johndoe@gmail.com',
        'password' => bcrypt('0000'),
        'admin' => false,
    ]);
}
```

Avant de lancer les différentes étapes du test, il faut créer un nouvel utilisateur. Afin de simuler au mieux le comportement d'un utilisateur lambda, celui-ci ne priviliege pas des droits admins.

```
// Act
```

```
$this->browse(function (Browser $browser) use ($user) {  
    $browser->visit('/'  
  
    // Login  
    ->clickLink('Se connecter')  
    ->type('email', $user->username)  
    ->type('password', '0000')  
    ->click('@login-button')
```

Dans cette partie, Dusk va accéder au navigateur à l'aide du pilote Chrome discuté précédemment. Une fois le site chargé, le lien contenant le texte « Se connecter » sera activé, ce qui va permettre d'être redirigé sur le formulaire de connexion. Sur ce formulaire, les informations de connexion seront entrées et le bouton de connexion sera enclenché.

Ce bouton est marqué l'attribut **dusk** sur la balise HTML responsable de la soumission du formulaire comme montré ci-après :

```
<input type="submit" value="Se connecter" dusk="login-button">
```

Cet attribut permet de cibler précisément lors des interactions simulées par les tests.

Le reste des étapes de ce test ayant un fonctionnement similaire à ce qui vient d'être vu, elles ne seront pas discutées ici.

```
->screenshot('finish')  
  
// Assert  
->assertPathIs('/team/1');
```

Une fois que toutes les étapes du test ont été réalisées, il faut faire une assertion afin de s'assurer que le résultat du test est celui qui est attendu. Dans le cas de ce test, on s'assure que le lien est celui qui est attendu avec l'**id** correcte de l'équipe, ce qui démontre également l'importance de réaliser une nouvelle migration de la base de données avant chaque nouveau test, car sinon celui-ci serait invalidé.

De plus ici, une capture d'écran est réalisée afin d'obtenir un visuel de la page que voici :

The screenshot shows the 'Détails de l'équipe' (Team Details) page for 'duskTeam'. At the top, there is a QR code and two blue buttons: 'Générer les groupes' (Generate groups) and '+ Ajouter des membres' (Add members). Below these are two light blue boxes labeled '1 - m1 m1' and '2 - m2 m2'. Under the heading 'Gestion des membres' (Member Management), it says '2 membres' (2 members) and lists 'm1 m1' (m1@a.a) and 'm2 m2' (m2@a.a).

26. Capture d'écran provenant d'un test Dusk

La mise en place du reste des tests étant très similaire et répétitif par rapport à ce qui vient d'être démontré, ChatGPT-4 a été utilisé dans un but de gain de temps.

#### 4.14.4 Risque de sécurité

L'utilisation de l'attribut Dusk dans les fichiers Blade peut présenter des risques, surtout si l'on ne veut pas exposer des informations concernant les tests internes au public.

Une manière potentielle de résoudre ce problème qui n'a pas été mis en place dans ce projet est l'utilisation d'une condition dans le fichier Blade qui vérifier l'état de l'environnement de l'application (donc s'il est en local ou en production).

#### 4.14.5 Difficultés rencontrées

Lors de la mise en place des tests, il est crucial de placer le préfixe « test » dans le nom des tests. Sinon, ils ne seront pas détectés et exécutés par Laravel Dusk. Une erreur avait été commise lors de la mise en place du test présenté précédemment en nommant le test « createTeamTest », ce qui a entraîné des erreurs lors de l'exécution.

Après avoir diagnostiqué le problème, la solution a été trouvée en comparant le nom avec celui du test créé par défaut lors de l'installation de Dusk qui commence par « test ». Une fois que le nom a été corrigé, le test a fonctionné correctement.

- Cette partie permet de reproduire ou reprendre le projet par un tiers.
- Pour chaque étape, il faut décrire sa mise en œuvre. Typiquement :

- Versions des outils logiciels utilisés (OS, applications, pilotes, librairies, etc.)
  - Configurations spéciales des outils (Equipements, PC, machines, outillage, etc.)
  - Code source commenté des éléments logiciels développés.
  - Modèle physique d'une base de données.
  - Arborescences des documents produits.
- Il faut décrire le parcours de réalisation et justifier les choix.

## 5 TESTS

---

### 5.1 Dossier des tests

- On dresse le bilan des tests effectués (qui, quand, avec quelles données...) sous forme de procédure. Lorsque cela est possible, fournir un tableau des tests effectués avec les résultats obtenus et les actions à entreprendre en conséquence (et une estimation de leur durée).
- Si des tests prévus dans la stratégie n'ont pas pu être effectués :
  - raison, décisions, etc.
- Liste des bugs répertoriés avec la date de découverte et leur état:
  - Corrigé, date de correction, corrigé par, etc.

## 6 CONCLUSION

---

### 6.1 Bilan des fonctionnalités demandées

- Il s'agit de reprendre point par point les fonctionnalités décrites dans les spécifications de départ et de définir si elles sont atteintes ou pas, et pourquoi.
- Si ce n'est pas le cas, estimer en « % » ou en « temps supplémentaire » le travail qu'il reste à accomplir pour terminer le tout.

### 6.2 Bilan de la planification

- Distinguer et expliquer les tâches qui ont généré des retards ou de l'avance dans la gestion du projet. Indiquer les différences entre les planifications initiales et détaillées avec le journal de travail.

### 6.3 Bilan personnel

- Si c'était à refaire:
  - Qu'est-ce qu'il faudrait garder ? Les plus et les moins ?
  - Qu'est-ce qu'il faudrait gérer, réaliser ou traiter différemment ?
- Qu'est-ce que ce projet m'a appris ?
- Suite à donner, améliorations souhaitables, ...
- Remerciements, signature, etc.

## GLOSSAIRE

---

### A

Artisan

Interface en ligne de commande fourni avec Laravel permettant d'interagir avec différents aspects de l'application.

7

---

### C

ChromeDriver

Pilote qui permet à Laravel Dusk de d'accéder au navigateur chrome.

15

Composer

Gestionnaire de dépendances écrit en PHP.

16

Contrôleur

S'inscrit dans l'architecuture MVC. Permet de faire le lien entre le modèle et la vue. S'occupe aussi de la pluspart des opérations de logique.

7

---

### D

Dusk

Outil de tests fonctionnant avec Laravel qui permet de simuler des interactions utilisateurs.

15

---

### F

factories

Outil de Laravel permettant de décrire des données fictives à insérer dans les champs d'une base de données.

14

---

### G

Github Actions

Outil d'intégration continue (CI/CD) de GitHub.

15

---

### L

**Laravel**

Framework PHP basé sur une architechture MVC.

4

---

### M

Migrations

Outil intégré à Laravel permettant la gestion des bases de données avec du code PHP.

18

Modèle

S'inscrit dans l'architechture MVC. Il s'agit de la partie du code qui interragit avec la base de données

7

MVC

Architechture Model - View - Controller

7

**O**

## ORM

(Object Relational Mapping) Interface permettant d'exécuter des requêtes sur la base de données avec du code logique. 8

**R**

## Responsive Design

Adaptation automatique de l'interface selon le type d'appareils utilisés (Smartphone, Desktop). 4

## Routes

Permet d'exécuter des instructions en fonction d'une URL et d'un type de requête. 8

**S**

## Seeders

Outil de Laravel permettant de définir comment générer des données fictives, comme le nombre d'occurrences d'un modèle. 15

**T**

## Table de pivot

Table dans une base de données dont l'identifiant est la concaténation de deux clés étrangères. Ce genre de tables permet la mise en pratique d'une relation n-n (many-to-many). 20

## Tailwind

Framework CSS. 17

## Tests End-to-End

Méthodologie de test permettant de tester l'entièreté d'une fonctionnalité dans des conditions réelles. 15

**V**

## Vue

S'inscrit dans l'architecture MVC. Il s'agit de la partie contenant généralement une interface graphique. 7

---

## 7 ANNEXES

---

### 7.1 Résumé du rapport de TPI

### 7.2 Accès au repository GitHub

Lien : <https://github.com/samdossant/TPI-SocialShuffle-Dos-Santos>

Le code source de l'application se trouve dans le répertoire « SocialShuffle »

- Listing du code source (partiel ou, plus rarement complet)
- Guide(s) d'utilisation et/ou guide de l'administrateur
- Etat ou « dump » de la configuration des équipements (routeur, switch, robot, etc.).
- Extraits de catalogue, documentation de fabricant, etc.

### 7.3 Bibliographie

## 7.4 Planning

Séquence 27			Date: lundi, 3 juin 2024	Après-Midi
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	
Doc - Rapport	28	2h20min	Impression si elle n'est pas encore faite, dernière vérification et envoi de la version papier et électronique.	
Total tranche	28	2h20min		

Séquence 26			Date: lundi, 3 juin 2024	Matin
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	
Doc - Rapport	36	3h	Finaliser, créer le document PDF annexe	
Administratif	12	1h	Impression de la version papier (1h pour prendre en compte le potentiel bouchon à l'imprimante)	
Total tranche	48	4h		

Séquence 25			Date: vendredi, 31 mai 2024	Après-Midi
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	
Réalisation - Tests	37	3h5min		
Doc - Journal de travail	1	0h5min		
Total tranche	38	3h10min		

Séquence 24			Date: vendredi, 31 mai 2024	Matin
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
Réalisation - Rôles des utilisateurs	48	4h		
Total tranche	48	4h		

Séquence 23			Date: jeudi, 30 mai 2024	Après-Midi
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
Réalisation - Authentification	7	0h35min	Gestion de la connexion	
Doc - Rapport	28	2h20min		
Administratif	2	0h10min	Envoi du rapport et du journal de travail mis à jour.	
Doc - Journal de travail	1	0h5min		
Total tranche	38	3h10min		

Séquence 22			Date: jeudi, 30 mai 2024	Matin
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
Réalisation - Algorithme de répartition	5	0h25min	Implémentation de l'algorythme dans le but de répartir les membres de l'équipe dans les groupes	
Réalisation - Authentification	43	3h35min	Gestion de la connexion	
Total tranche	48	4h		

Séquence	21		Date: mercredi, 29 mai 2024	Matin
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
Réalisation - Algorithme de répartition	27	2h15min	Implémentation de l'algorythme dans le but de répartir les membres de l'équipe dans les groupes	
Doc - Journal de travail	1	0h5min		
Total tranche	28	2h20min		

Séquence	20		Date: mardi, 28 mai 2024	Après-Midi
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
Doc - Rapport	25	2h5min		
Administratif	2	0h10min	Envoi du rapport et du journal de travail mis à jour.	
Doc - Journal de travail	1	0h5min		
Total tranche	28	2h20min		

Séquence	19		Date: mardi, 28 mai 2024	Matin
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
Mise au point avec le chef de projet	4	0h20min		
Réalisation - Algorithme de répartition	44	3h40min	Implémentation de l'algorythme dans le but de répartir les membres de l'équipe dans les groupes	
Total tranche	48	4h		

Séquence			18	Date: lundi, 27 mai 2024	Après-Midi
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...	
Doc - Rapport	27	2h15min			
Doc - Journal de travail	1	0h5min			
Total tranche	28	2h20min			

Séquence			17	Date: lundi, 27 mai 2024	Matin
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...	
Réalisation - CRUD équipes et membres	48	4h	Contrôleur permettant de réaliser les opérations CRUD sur les membres		
Total tranche	48	4h			

Séquence			16	Date: vendredi, 24 mai 2024	Après-Midi
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...	
Réalisation - CRUD équipes et membres	30	2h30min	Contrôleur permettant l'ajout / modification / lecture / suppression des équipes		
Doc - Rapport	7	0h35min			
Doc - Journal de travail	1	0h5min			
Total tranche	38	3h10min			

Séquence	15			Date: vendredi, 24 mai 2024	Matin
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...	
Réalisation - Algorithme de répartition	36	3h	Rechercher et tester des algorithmes permettant la mise en place de la fonctionnalité voulue.		
Réalisation - CRUD équipes et membres	12	1h	Contrôleur permettant l'ajout / modification / lecture / suppression des équipes		
Total tranche	48	4h			

Séquence	14			Date: jeudi, 23 mai 2024	Après-Midi
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...	
Réalisation - Vues	12	1h	Formulaire de création des membres		
Réalisation - Vues	8	0h40min	Page d'affichage de l'équipe		
Doc - Rapport	15	1h15min			
Administratif	2	0h10min	Envoi du rapport et du journal de travail mis à jour.		
Doc - Journal de travail	1	0h5min			
Total tranche	38	3h10min			

Séquence	13			Date: jeudi, 23 mai 2024	Matin
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...	
Réalisation - Vues	12	1h	Layout		
Réalisation - Vues	12	1h	Formulaire de connexion		
Réalisation - Vues	12	1h	Page principale où s'affichent les équipes		
Réalisation - Vues	12	1h	Formulaire de création d'équipes		
Total tranche	48	4h			

Séquence 12			Date: mercredi, 22 mai 2024	Matin
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
Réalisation - Base de données	25	2h5min		
Doc - Rapport	2	0h10min		
Doc - Journal de travail	1	0h5min		
Total tranche	28	2h20min		

Séquence 11			Date: mardi, 21 mai 2024	Après-Midi
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
Doc - Rapport	25	2h5min		
Administratif	2	0h10min	Envoyer du rapport et du journal de travail mis à jour.	
Doc - Journal de travail	1	0h5min		
Total tranche	28	2h20min		

Séquence 10			Date: mardi, 21 mai 2024	Matin
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
Mise au point avec le chef de projet	4	0h20min		
Réalisation - Crédit / Mise en place du projet Laravel	9	0h45min		
Réalisation - Base de données	35	2h55min	Définir les migrations et les lancer afin de créer la base de données	
Total tranche	48	4h		

Séquence	9		Date: vendredi, 17 mai 2024	Après-Midi
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
Analyse - Tests	27	2h15min	Décrire la stratégie de tests à employer	
Doc - Rapport	10	0h50min		
Doc - Journal de travail	1	0h5min		
Total tranche	38	3h10min		

Séquence	8		Date: vendredi, 17 mai 2024	Matin
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
Analyse - Schéma de principe	36	3h	Réalisation d'un schéma démontrant les actions possibles pour l'utilisateur.	
Analyse - Tests	12	1h		
Total tranche	48	4h		

Séquence	7		Date: jeudi, 16 mai 2024	Après-Midi
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
Doc - Rapport	35	2h55min		
Administratif	2	0h10min	Envoyer du rapport et du journal de travail mis à jour.	
Doc - Journal de travail	1	0h5min		
Total tranche	38	3h10min		

Séquence			6	Date: jeudi, 16 mai 2024	Matin
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...	
Analyse - Maquettes graphiques	48	4h			
Total tranche	48	4h			
Séquence			5	Date: mercredi, 15 mai 2024	Matin
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...	
Analyse - Modélisation de la DB	27	2h15min			
Doc - Journal de travail	1	0h5min			
Total tranche	28	2h20min			
Séquence			4	Date: mardi, 14 mai 2024	Après-Midi
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...	
Doc - Rapport	12	1h	Création du document et mise en page initiale		
Analyse - Modélisation de la DB	13	1h5min			
Administratif	2	0h10min	Envoi de la version mise à jour du rapport et du journal de travail		
Doc - Journal de travail	1	0h5min			
Total tranche	28	2h20min			

Séquence 3			Date: mardi, 14 mai 2024	Matin
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
Mise au point avec le chef de projet	4	0h20min		
Analyse - Environement	44	3h40min	Analyse de l'environnement du projejt (Public cible, méthodologie de projet, Laravel, MVC)	
Total tranche	48	4h		

Séquence 2			Date: lundi, 13 mai 2024	Après-Midi
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
Analyse - Planification initiale	25	2h5min		
Administratif	2	0h10min	Envol de la planif initiale aux experts et au chef de projet	
Doc - Journal de travail	1	0h5min		
Total tranche	28	2h20min		

Séquence 1			Date: lundi, 13 mai 2024	Matin
Tâche	Tranche [5min]		Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
Administratif	8	0h40min		
Analyse - Planification initiale	40	3h20min		
Total tranche	48	4h		

## 7.6 Journal de travail