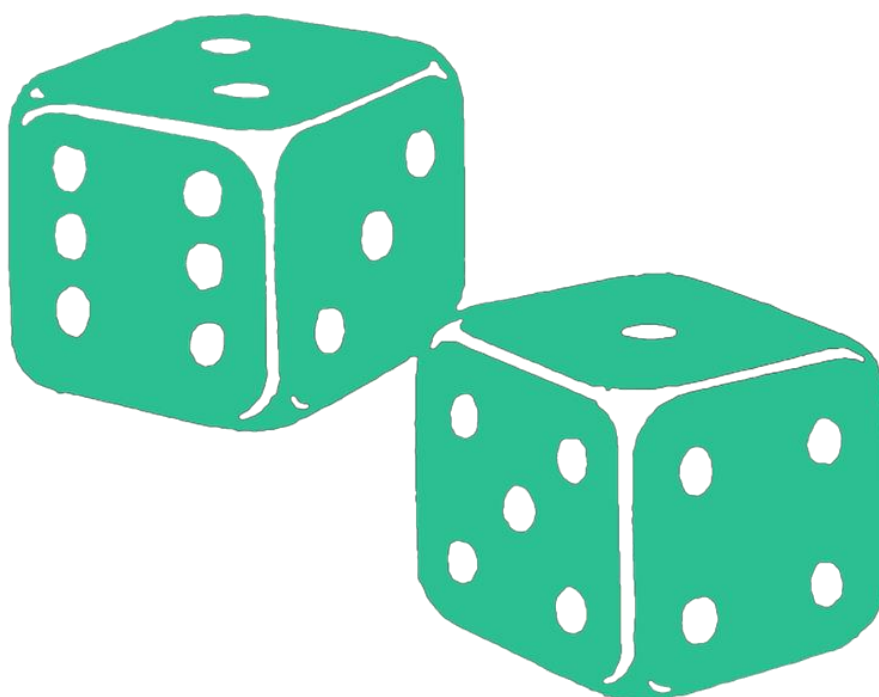


Dossier de projet

MEETPLE



PRÉSENTÉ PAR
ANTHONY PINEAU

EN VUE DE L'OBTENTION DU TITRE PROFESSIONNEL
DÉVELOPPEUR WEB ET WEB MOBILE
CERTIFICATION DE NIVEAU III

Centre de formation O'clock 2019-2020

Sommaire

I. INTRODUCTION	P.4
II. LISTES DES COMPÉTENCES DU RÉFÉRENTIEL	P. 5
A. Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité :	P. 5
1. Maquetter une application	
2. Réaliser une interface utilisateur web statique et adaptable	
3. Développer une interface web dynamique	
B. Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité :	P. 6
1. Créer une base de données	
2. Les composants d'accès aux données	
3. Développer la partie back-end d'une application web ou web mobile	
III.CAHIER DES CHARGES	P. 7
A. Résumé du projet	P. 7
B. Arborescence de l'application	P. 7
C. Wireframes	P. 8
D. Structuration des données	P. 9
E. Produit minimum viable (Minimum Viable Product)	P. 9
F. Version 1,0	
G. Les potentielles évolutions	
H. User stories	P. 11
1. Visiteur	
2. Utilisateur connecté	
3. Organisateur	
4. Administrateur	

IV.SPÉCIFICATIONS TECHNIQUES	P. 13
A. Frontend	P. 13
B. Backend	P. 14
V.RÉALISATION	P. 16
A. Équipe	P. 16
B. Rythme	P. 17
C. Outils de travail utilisés	P. 17
VI.MON RÔLE DANS LE DÉVELOPPEMENT DU PROJET	P. 19
A. Sprint 0	
B. Sprint 1	
C. Sprint 2	
D. Sprint 3	
VII.PRÉSENTATION DU JEU D'ESSAI P. 20	
A. Présentation	P. 20
B. Moyens mis en œuvre	P. 20
VIII.RÉALISATION PERSONNELLE	P. 23
A. Mise en place de l'architecture de l'application	P. 23
B. Mise en place des routeur React	P. 24
C. Intégration des pages	P. 27
D. Système d'authentification	P. 34
IX.VULNÉRABILITÉS DE SÉCURITÉ ET VEILLE	P. 37
A. Persistance de la connexion	P. 37
B. JSON Web Token	P. 37
X.EXTRAIT ET TRADUCTION D'UNE RESSOURCE ANGLOPHONE POUR LE PROJET	P. 38
A. Extrait	P. 38
B. Traduction	P. 41
XI.CONCLUSION	P.43
XII.ANNEXES	P. 44

I. INTRODUCTION

Dans le cadre de ma formation de cinq mois j'ai pu acquérir de nombreuses compétences du métier de développeur web, ça m'a aussi permis de découvrir une méthode de travail et d'apprentissage pour me permettre d'apprendre par moi-même l'univers vaste du développement web et mobile, lors de cette formation nous devions choisir une spécialisation parmi plusieurs choix dont la librairie JavaScript React et le framework PHP Symfony, choisir une spécialisation ne fut pas évidente mais après mûre réflexion j'ai préféré m'orienter sur le front-end avec la librairie JavaScript React.

Après quatre semaines d'apprentissage du framework React nous avons consacré quatre semaines à la réalisation d'un projet d'une application web nommé "Meetple" proposé par William le Product Owner du projet, nous sommes partis sur l'idée d'une application web permettant d'organiser des événements autour du jeu de société.

L'équipe était composée de quatre personnes dont deux développeurs React et deux développeurs Symfony.

Ce dernier mois fut consacré à la réalisation d'un projet qui m'a permis d'approfondir mes connaissances ainsi que de mieux maîtriser les compétences acquises pendant la formation tout en me plongeant dans une situation professionnelle en passant du statut d'apprenant à développeur web.

II. LISTES DES COMPÉTENCES DU RÉFÉRENTIEL

A. Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

1. Maquetter une application

Après avoir défini le projet et ces fonctionnalités, l'une des premières étapes a été de penser l'interface utilisateur, on a pour cela réalisé des schémas appelés wireframe ou maquette fonctionnelle pour définir les zones et composants qu'elle doit contenir, Nous avons chacun réalisé les prototypes sur le site web Whimsical, une première version des wireframes a été réalisée puis améliorée par la suite, nous avons fait en sorte de concevoir une version desktop et une version smartphone de chaque page de l'application. Les wireframes sont disponible en annexe.

2. Réaliser une interface utilisateur web statique et adaptable

A partir des wireframes réalisé et d'une réflexion de groupe sur la charte graphique nous avons intégré les différentes pages de façon responsive c'est à dire adaptable sur toutes les plateformes avec plusieurs possibilités d'affichage en fonction de la taille de l'écran pour le rendre fonctionnel sur ordinateur comme sur smartphone, nous avons fait en sorte de réaliser une interface épurée et bien structurée à partir des prototypes.

L'intégration a été réalisé avec des composants React qui utilise la syntaxe JSX et la librairie Styled-Components pour la mise en forme des pages.

3. Développer une interface web dynamique

L'interface utilisateur de l'application a été basé sur la librairie Javascript React développé par Facebook, cette librairie a été mis à la disposition des développeurs et permet le développement d'application réactive car le code javascript est exécuté côté client.

React gère les données de l'application de façon dynamique, au moindre changement (l'utilisateur clique sur un lien, une donnée est mise à jour), les éléments du DOM virtuel de React sont recalculés et le DOM mis à jour est affiché.

Le DOM (Document Object Model) est une représentation de la structure de l'affichage à l'écran. C'est le DOM qui est interprété par le moteur de rendu des navigateurs internet pour être ensuite affiché.

Les données affichées sont récupérées de façon asynchrones grâce au module Axios, l'api récupéré provient du back-end développé en Symfony qui renvoi les données en Json.

L'interface d'administration a été réalisé en HTML / PHP et avec la librairie Bootstrap pour gagner du temps.

B. Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité

1. Créer une base de données

Pour réaliser la base de données nous avons commencé par réaliser la documentation qui nous a permis d'avoir un visuel de la structure de la base de données et de ses relations, pour cela nous avons fait un MCD (Modèle Conceptuel de Données), il s'agit d'un schéma qui représente les différentes tables et leur association ainsi que la quantité d'occurrences qui peuvent être concernées par ces relations (cardinalités) et ce dans les deux sens, les données ont été recensées dans un dictionnaire de données.

Vous pouvez retrouver le MCD et le dictionnaire de données en annexe.

2. Les composants d'accès aux données

Les données de la base de données sont récupérées côté back-end qui renvoi côté front-end une api au format Json, les données sont ensuite affichées côté front de façon asynchrone grâce à Ajax via le module Axios.

3. Développer la partie back-end d'une application web ou web mobile

Certaines données sont visibles que si la personne est connectée, pour cela nous avons mis en place un système d'authentification avec JWT Token.

III. CAHIER DES CHARGES

A. Résumé du projet

Ce dernier mois "l'apothéose" a été consacré à la réalisation du projet nommé Meetple, le but de l'application est de concevoir une plateforme qui permet de faciliter l'organisation d'événements autour du jeu de plateau s'adressent au grand public ou à des organisateurs professionnels (bars ludiques, associations, magasins, salons, particuliers) souhaitant partager des moments conviviaux, ainsi que permettre de créer des rencontres ludiques. Pour cela l'application permet d'organiser des événements ou de simplement y participer.

Le Minimum Viable Product (MVP) est centré sur les événements, il a été ajusté au fur à mesure face aux problématiques rencontrés. Il est possible de créer des événements ainsi que de participer à des événements, les fonctionnalités ont pour vocation d'être simples à prendre en main. La réalisation du Minimum Viable Product s'est vue répartie sur plusieurs sprints d'une durée chacune d'une semaine.

L'application est développée en deux parties d'un côté le back-end avec le framework Symfony qui renvoie une API récupérée par la partie front-end développée avec la librairie React, l'application est asynchrone pour cela j'ai utilisé le module Axios

Le nom Meetple, c'est un jeu de mot avec Meeple (les "pions"), Meet et People. C'est une rencontre entre gens autour des meeples !

"Vous souhaitez faire une soirée entre amis et jouer à vos jeux de plateau favoris ?
Vous préparez un événement lors d'un festival ou dans un bar ludique ?
Vous voulez vous amuser en famille un dimanche après-midi autour d'une bonne galette ?
Meetple est fait pour vous !"

B. Arborescence de l'application

L'arborescence (Workflow) a été conçu sur le site web Whimsical, il représente les parcours possibles dans l'application en tant que visiteur ou utilisateur connecté. Le workflow est disponible en annexe.

C. Wireframes

Les prototypes ont été conçus sur le site web Whimsical en version ordinateur et version smartphone, je vais vous présenter les pages les plus significatifs permettant de préciser les fonctionnalités.

Page d'accueil :

- On y trouve une barre de navigation qui est présente sur toute l'application et dont le contenu varie en fonction de :
 - Si l'utilisateur est connecté on a accès aux liens "Créer un évènement", "Voir mon profil", "Déconnexion".
 - Si l'utilisateur n'est pas connecté on accès aux liens "Connexion" et "Inscription".
- Une zone qui présente l'application.
- Un filtre par département, tag, catégorie et jeu.
- Une liste des évènements avec différentes informations tel que le titre, l'auteur, les jeux, etc. Pour pouvoir accéder à la page de l'évènement et y participer il faut être connecté.
- Un pied de page qui est lui aussi présent sur tous les pages ou on peut y trouver par exemple les mentions légales.

Page d'un évènement :

- On y retrouve la barre de navigation et le pied de page.
- Toutes les informations concernant l'évènement avec plus de détails telle qu'une carte pour l'adresse.
- Une zone pour laisser un commentaire.
- Un bouton pour participer à l'évènement.
- L'accès à un menu qui apparaît au click avec la liste des évènements auquel l'utilisateur est inscrit avec la possibilité d'accéder à la page de l'évènement ou d'annuler sa participation.

Page profil :

- On y retrouve la barre de navigation et le pied de page.
- Ainsi que l'accès à un menu qui apparaît au click avec la liste des évènements auquel l'utilisateur est inscrit.
- La liste des évènements dont l'utilisateur est l'organisateur avec quelque information sur l'évènement et la possibilité de l'annuler avec un bouton supprimer.

D. Structuration des données

La documentation sur les données a été l'une des premières choses auquel nous avons réfléchi, c'est l'un des éléments centraux de l'application qui permet de définir son contenu, pour cela on a conçu un MCD et un dictionnaire de données, c'est document ont été mis à jour à plusieurs reprises lors de la conception du projet. Le MCD et le dictionnaire de données sont disponible en annexe.

E. Produit minimum viable (Minimum Viable Product - MVP)

1. Version 1,0

- **En tant que visiteur ou utilisateur inscrit**, Dans la page d'accueil il y a la liste des futurs événements sur lesquels seront visibles :
 - Le titre.
 - Le nom de l'organisateur.
 - La date.
 - Le nombre de participants / Le nombre maximums.
 - Le lieu et le département.
 - Le ou les tags.
 - La catégorie.
 - Le ou les jeux.
- **Un visiteur a la possibilité :**
 - De s'inscrire via un formulaire avec les informations suivantes à remplir : pseudo, email, mot de passe.
 - De se connecter via un formulaire de connexion une fois inscrit.
- **Un utilisateur inscrit a la possibilité :**
 - D'accéder à sa page profil privé avec la possibilité :
 - D'éditer le pseudo, L'email et le mot de passe.
 - De supprimer un évènement qu'il organise.
 - De voir les évènements auquel il est inscrit et pouvoir annuler sa présence.
 - D'organiser un événement via un formulaire en remplissant au moins ces informations :
 - Le titre.
 - La description.
 - Le lieu.
 - La date.
 - Le ou les tags (stratégies, jeu de cartes, réflexion ...).
 - La catégorie (Découverte, compétition, familiale ...).
 - Le ou les jeux.

- Nombre de participants possible.
 - D'accéder à la page d'un événement ou il peu :
 - Retrouver toutes les infos (Titre, créateur, date, lieu complet, description, nombre de participants, participants, tags, catégorie, liste des jeux).
 - Pouvoir éditer en tant qu'organisateur les informations (titre, date etc...).
 - Un lien ou bouton permettant de rejoindre ou de quitter l'évènement.
 - Pouvoir laisser un commentaire sur l'évènement.
- **Un administrateur a la possibilité :**
 - De gérer le rôle des utilisateurs.
 - D'éditer ou supprimer un évènement.
 - D'ajouter, supprimer ou modifier les tags, les catégories et les jeux

2. Les potentielles évolutions

- **Un utilisateur inscrit** aurait la possibilité
 - De voir le profil des autres utilisateurs avec :
 - Des informations sur le profil de l'utilisateur visité.
 - Ses événements créés et auxquels ils participent.
 - Une liste d'amis inscrit sur le site pour inviter facilement des amis.
 - Possibilités de noter ou commenter les événements / les organisateurs auxquels on a participé (avec la possibilité d'attacher un commentaire).
 - D'ajouter des photos du lieu de l'évènement.
 - Un système de score sur le résultat de l'évènement.
 - Un système de paiement si l'organisateur souhaite faire payer la participation
- **Un visiteur** pourrait s'inscrire via un compte externe (Google, Facebook etc..).

F. User stories

1. En tant que Visiteur

Je	Afin de
Veux voir la page d'accueil.	Pouvoir voir les futurs évènements.
Veux pouvoir m'identifier.	Pouvoir accéder à ma page de profil, aux évènements et pouvoir participer à un évènement.
Veux pouvoir voir la liste des événements créés, sur la page d'accueil.	Chercher directement les événements qui m'intéresse.
Veux pouvoir m'inscrire.	Pouvoir identifier et pour participer aux événements.
Veux pouvoir voir les informations complémentaires en bas de page.	D'avoir plus d'information sur le site.
Veux pouvoir accéder aux mentions légales	D'avoir plus d'information juridique sur le site.
Veux pouvoir accéder à la page à propos	D'avoir plus d'information sur le site.
Peux appliquer des filtres sur la page d'accueil	Pouvoir rechercher plus en détail les événements qui m'intéresse.

2. Utilisateur connecté

Je	Afin de
Peux me déconnecter à tout moment	De protéger mes informations personnelles.
Peux accéder à la page pour créer un événement	Qu'il soit enregistré et visible pour les autres usagers.
Veux pouvoir m'inscrire à un événement	Pouvoir y participer.
Peux poster un commentaire sur l'événement auquel je participe	De laisser mon avis ou poser des questions.
Veux pouvoir visionner les informations sur une page événement disponible	De savoir plus d'information sur l'évènement ciblé.
Peux accéder à ma page de profil	Voir mes informations personnelles.
Peux accéder à la page profil d'une autre personne	Voir son pseudo et ses événements

Peux éditer les informations sur mon profil	Effectuer des modifications si nécessaires.
Peux voir les événements auquel j'ai participé sur ma page de profil	D'y retrouver des informations et d'y retrouver l'historique des sessions auquel j'ai participé.
Pouvoir récupérer mon mot de passe en cas de perte.	Pouvoir avoir un nouveau accès à page personnelle.

3. Organisateur

Je	Afin de
Veux pouvoir mettre une date à un événement	Prévenir les participants de quand se tiendra l'événement.
Je veux pouvoir renseigner le nombre de participant à un événement.	Prévenir du nombre de participant déjà inscrit et le nombre de place restante.
Je veux pouvoir renseigner le tag à un événement	Définir le type d'événement que je veux créer.
Je veux pouvoir renseigner la catégorie à un événement	Définir le type d'événement que je veux créer.
Je veux pouvoir renseigner le (ou les) jeux à un événement	Prévenir les usagers des activités qui seront proposé.
Veux pouvoir mettre une description à un événement	Détaillé des informations supplémentaires.
Veux pouvoir mettre un lieu à un événement.	Prévenir les participants d'où se tiendra l'événement.

4. Administrateur

Je	Afin de
Veux pouvoir accéder une page personnalisée	D'ajouter les thèmes
Veux pouvoir accéder une page personnalisée	Changer les rôles

IV. SPÉCIFICATIONS TECHNIQUES

A. Front-end

Nous avons opté pour réaliser ce projet voté front-end :

- **Yarn** : un gestionnaire de paquet utilisé dans le développement et le fonctionnement de l'application.
- **React** : Une librairie Javascript qui permet de créer une application asynchrone.
- **Material-UI** : Met à disposition des composants tels que des menus déjà construits réutilisables ou un système de grille pour le responsive.
- **Axios** : Un module permettant de faciliter les appels à l'API (Application Program Interface). Tout comme AJAX en jQuery, elle permet de simplifier le processus des appels API en faisant une fonction asynchrone (en partant d'une requête puis en terminant par une réponse succès ou échec). Ainsi, notre appel est centralisé et permet d'avoir un code plus propre.
- **Prop-Types** : Un module permettant d'effectuer des tests de type et d'existence sur les propriétés des composants. Cela permet de s'assurer que les données s'écoulent de composants en composants sans erreur.
- **React-Router-Dom** : Un module permettant avec la librairie Javascript React conçu pour des applications dites "One page" de concevoir une application multipage avec l'aide de React Router. Il s'agit d'une bibliothèque tierce qui permet le routage dans nos applications React en créant des chemins d'accès aux différentes pages de l'application.
- **React-Redux** : Un module permettant la gestion centralisée des états et des actions des composants React
- **Redux-Phoenix** : Permet de garder le JWT en mémoire même après avoir rafraîchi la page de l'application.
- **Slugify** : Un module permettant de remplacer par exemple des espaces ou des caractères spéciaux dans une phrase.

- **Reactjs-Popup** : Un module permettant de faciliter la mise en place de popup parfois appelée fenêtre surgissant qui apparait au-dessus du contenu.
- **Styled-Components** : Un module permettant de gérer le CSS et de gagner en productivité. La force de Styled-Components c'est de faciliter la création de composants visuels React minimalistes et configurables, en combinant du CSS standard et un zeste de JavaScript. Ces composants deviendront littéralement les pièces de lego qui pourront être utilisées et partagés entre les composants de toutes l'applications.
- **Babel** : Un module permettant de transpiler ce qui va transformer du code sous React (ES6) en un code compréhensible pour la plupart des navigateurs (ES5).
- **Webpack** : Un module permettant de compiler tous les fichiers pour les regrouper en un seul, ce qui est très utile pour améliorer la performance de l'application. Il dispose, en plus, d'un système de "loaders" qui vont permettre d'inclure de nouveaux types de fichiers ou d'appliquer des transformations spécifique (comme une transformation ES2015->ES5).

B. Back-end

Nous avons opté pour réaliser ce projet coté back-end :

- **Composer** : Un gestionnaire de dépendances PHP qui permet la déclaration et l'installation de bibliothèques requis par le projet.
- **Symfony** : Ce framework permet de faciliter la génération de routes, de Controllers, de Models et de bases de données SQL. C'est aussi un framework robuste et sécurisé, muni d'un ORM (Object-Relational Mapping).
- **PhpMyAdmin** : Logiciel écrit en PHP, destiné à gérer l'administration de MySQL sur le Web. Il prend en charge un large éventail d'opérations sur MySQL . Les opérations fréquemment utilisées (gestion des bases de données, des tables, des colonnes, des relations, des index, des utilisateurs, des autorisations, etc.) peuvent être effectuées via l'interface utilisateur, tout en ayant la possibilité d'exécuter directement toute instruction SQL

- **Doctrine** : ORM par défaut du framework Symfony. Cette interface assure la relation entre les objets et les éléments de la base de données. Outil majeur dans le développement sous Symfony, il permet par le biais d'entités (classes d'objets) de
 - Définir complètement les tables et champs de la base de données
 - Les relations entre les tables
 - La persistance des données (mécanisme de sauvegarde et de restitution des données afin qu'elles ne soient pas perdues en fin d'exécution d'une application).
 - La manipulation et la consultation des données via ses Entity et Repository managers à partir des Controllers ou des Vues
- **PSR4** : spécification du standard de chargement automatique des classes à partir de leur chemin.
- **Twig** : Moteur de templates intégré à Symfony et qui offre un pseudo langage qui lui est propre pour la dynamisation des templates. Au travers de sa syntaxe claire et concise, Twig permet entre autres, de créer des boucles complexes, des conditions pour afficher un menu différent pour les utilisateurs authentifiés ou non, afficher des variables, filtrer, etc. Twig assure également la sécurité des données grâce à l'échappement automatique des caractères spéciaux.
- **Fixtures** : Il permet au développeur d'initialiser les données de la base de données à des fins de développement et de test. Ces données sont primordiales car elles permettent de travailler dans un contexte au plus proche de celui de la production, quand l'application sera déployée.
- **Bundle Unirest** : permet la génération de données factices crédibles.
- **LexikJWTAuthenticationBundle** : Il permet de générer le token nécessaire à l'authentification d'un utilisateur. Ce token permet de sécuriser les actions que seul l'utilisateur peut faire, en vérifiant son existence. Il a ainsi permis de créer le lien entre un utilisateur connecté et le back-end.
- **NelmioCorsBundle** : Il permet de faciliter la gestion des CORS, autrement dit des autorisations requises à l'accès à la base de données. Seul le serveur front-end a accès à ces données grâce à ce bundle.

V. RÉALISATION

A. Équipe

En fin de formation on pouvait proposer des projets pour le mois “d’apothéose” de quatre semaines qui permettait de mettre en pratique tous ce qui avait été vu lors de la formation ainsi que de travailler en groupe sur un projet de zéro, William a alors proposé le projet Meetple, un projet qui m'a plu, nous devions faire une liste de 5 choix sur les projets proposés pour constituer les équipes et c’est ainsi que j’ai rejoint l’équipe du projet Meetple.

Je vais vous présenter les membres de l’équipe du projet Meetple qui était composé de deux développeur React et deux développeur Symfony, afin de faciliter le travail nous devions nous attribuer des rôles au sein du groupe :

Product Owner (*William*)

- Responsable du « produit », de la « vision produit »
- C'est un Expert métier (il connaît les tenants & aboutissants, les contraintes tech/business)
- Il représente le client (ses enjeux, intérêts, priorités)
- Il s'assure du ROI (return on investment, retour sur investissement / valeur ajoutée)

Scrum master (*Florent*)

- Facilitateur du projet
- C'est le responsable projet
- Il s'assure du respect de la méthode Scrum (sprint, tâches, responsabilités, ...)
- Il ne peut pas également être le Product owner !

Lead dev front (*Anthony*) / lead dev back (*Fabien*)

- Effectue les choix techniques côté front / back.
- S'assure du bon fonctionnement de la facette du projet.

Référents techniques (Côté frontend *Anthony* et côté backend *William*)

- Git master (gère le versioning du projet).
- Un référent par techno/lib/bundle particulière du projet.

B. Rythme

Nous nous sommes inspirés de la méthode scrum pour la gestion du projet, l'idée est de structurer les actions de l'équipe pour gagner en efficacité en jouant sur les forces de chacun des membres, la méthode scrum est une approche basée sur les principes agiles qui permet de structurer le développement du projet en cycles de travail appelés sprint. Ces cycles durent entre 1 et 4 semaines (une semaine pour ce projet) et s'enchaînent les uns après les autres, un sprint commence et fini à une date précise et n'est jamais prolongé.

Au début de chaque sprint l'équipe se réunissait afin de déterminer les tâches qui sont jugées comme étant à la fois prioritaires et « faisables » durant le sprint. L'idée étant de pouvoir achever & livrer des fonctionnalités bien concrètes, qui en retire une valeur ajoutée immédiate et tangible. Suite à cette réunion, le sprint qui démarre ne recevait plus de nouvelles tâches, l'objectif maximal était fixé. Il arrivait que l'on n'est pas atteint les objectifs fixés et nous devions les décaler dans le sprint suivant.

Un sprint durait une semaine, l'équipe se réunissait chaque matin à 9h00 par une réunion pour mettre en commun ce qui avait été fait la veille, ce sur quoi nous allions travailler ce jour et de relever d'éventuels points bloquants, puis mise en place du travail en individuel ou pairprogramming en fonction du Trello et lancement du développement journalier.

À la fin de chaque sprint, une réunion entre chaque équipe et les consultant(s) de la formation est mise en place pour présenter l'avancement de son projet durant le sprint afin de partager les idées, les problèmes rencontrés et de répondre à toutes les questions rassemblées au cours de la semaine.

C. Outils de travail utilisé

La formation O'clock est délivrée en téléprésentiel et donc nous devons utiliser plusieurs outils nous permettant de travailler à distance, les voici :

Discord :

C'est une application de discussions en ligne, cette application dispose des fonctions de chat vocal et de messagerie texte, avec laquelle nous pouvions partager les photos, les vidéos et les liens. Ça nous a permis de communiquer directement entre nous en vocal et de partager notre écran.

Trello :

C'est un outil en ligne pour la gestion de notre projet inspiré par la méthode Kanban facilitant la gestion et le suivi des tâches dans les sprints grâce à cet outil graphique, collaboratif et ergonomique. Nous définissons les différentes tâches classés dans différentes catégories en fonction de notre avancé sur le projet "À faire", "En cours" et "Terminé".

Slack :

C'est une plate-forme de communication collaborative ainsi qu'un logiciel de gestion de projets, slack fonctionne à la manière d'un chat IRC organisé en canaux. La plateforme permet également de conserver une trace de tous les échanges et permet le partage de fichier au sein des conversations. Nous avons à disposition un Channel spécifique très utile pour le partage de documentations ou autre.

GitHub :

C'est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de versions Git. Il nous a permis de versionner notre code source du projet.

Whimsical :

C'est un outil en ligne qui nous a permis de concevoir les wireframes et le workflow.

Visual Studio Code :

C'est un éditeur de code extensible multiplateforme édité par Microsoft destiné aux développeurs qui intègre plusieurs outils facilitant la saisie de code par les développeurs comme la coloration syntaxique ou encore le système d'auto-complétions IntelliSense.

Nous utilisons plusieurs extensions pour faciliter le travail tel que Live Share qui permet d'éditer et de déboguer en collaboration avec d'autres en temps réel, il permet de partager instantanément un projet.

Google Docs :

C'est un logiciel de traitement de texte basé sur le Web qui permet de créer et de modifier des documents en ligne et de travailler en équipe en temps réel. On l'a utilisé pour réaliser notre cahier des charges et d'autre document telle que les user stories.

VI. MON RÔLE DANS LE DÉVELOPPEMENT DU PROJET

La réalisation du projet a été découpé en Quatre sprints pour réaliser notre MVP :

A. Sprint 0 :

Nous avons commencé par réaliser notre cahier des charges en définissant les grandes lignes du projet ainsi que c'est objectif, par la suite nous nous sommes penchés sur le MVP (Produit Minimum Viable) avec les potentielles évolutions, les fonctionnalités de l'application puis avec quelle technologie réaliser ces fonctionnalités, nous avons définis chacun nos rôles dans la réalisation du projet et nous avons mis en place un trello pour définir les fonctionnalités à réaliser durant chaque sprint, une fois les sprints définis nous avons réalisé les documents de conception de l'application (Workflows, user stories, Wireframes, les routes, dictionnaire des données, MCD (Model Conceptuel des données))

B. Sprint 1

Durant ce sprint nous avons commencé à nous répartir les tâches chacun de notre côté, j'ai mis en place le repository GitHub côté front, mis en place le dossier React avec npm et la structure des dossiers, j'ai aussi réalisé une partie des pages de l'application en statique avec de fausses données.

C. Sprint 2

Durant ce sprint j'ai finalisé l'intégration des pages et je me suis penché sur la récupération des données provenant de l'api, j'ai mis en place l'authentification avec la récupération du JWT qui m'a permis d'afficher les informations de l'utilisateur connecté, de mettre en place le système de participation / annulation à l'évènement et de créer un évènement.

D. Sprint 3

Ce dernier sprint a été consacré à des améliorations et des retouches, j'ai corrigé des erreurs, j'ai affiché des messages d'erreur comme par exemple quand un email n'est pas valide un message apparaît qui l'indique à l'utilisateur, nous avons mis des fausses données dans la liste des événements pour remplir l'application, nous avons aussi déployé l'application sur un serveur AWS (Amazon Web Services).

VII. PRÉSENTATION DU JEU D'ESSAI


A. Présentation

Lors de la réalisation de la page profil d'un utilisateur connecté j'ai affiché les données de l'utilisateur avec la possibilité de modifier ces informations telle que le pseudo.

En cliquant sur le bouton à droite avec l'icône d'un crayon le pseudo se transforme en un input avec le pseudo actuel qui est modifiable, aux cliques sur le bouton validé en forme de "v" le pseudo est modifié.

B. Moyens mis en œuvre

Votre pseudo:

Anthony 

Par défaut la valeur *userModify* est false et affiche le pseudo avec un crayon à droite.

```
{!userModify && (  
  <>  
    <p className="info">{profilePrivate.username}  
      <button className="modify" onClick={handleClickUsername} type="button"> <CreateIcon />  
    </button>  
  </p>  
  </>  
)}
```

Au clic sur le bouton qui correspond au crayon l'action onClick "handleClickUsername" se lance.

```
const handleClickUsername = (event) => {  
  event.preventDefault();  
  InputModifyUsername();  
};
```

Votre pseudo:

Anthony 

La valeur *userModify* devient true elle affiche alors le pseudo dans un input.

```

<h4 className="titleinfo">Votre pseudo:</h4>

{userModify && (
  <form onSubmit={handleSubmitUsername}>
    <input
      name="newUsername"
      className="input"
      type="text"
      placeholder={profilePrivate.username}
      value={newUsername}
      onChange={handleChangeUsername}
    />
    <button className="modify" type="submit"
      <CheckIcon />
    </button>
  </form>
)}

```

Au clic sur le bouton validé l'action onSubmit "handleSubmitUsername" se lance.

```

const handleSubmitUsername = (event) => {
  event.preventDefault();
  modifyUsername();
};

```

La modification du pseudo écrit dans l'input est envoyée au middleware correspondant qui effectue un appel Axios, l'api récupère l'id de l'utilisateur qui a fait cette demande ([url](#)) et lui demande s'il a l'autorisation en vérifiant son *Json Web Token* récupéré lors de la connexion ([headers](#)) ensuite si tout est validé il récupère le nouveau pseudo et l'enregistre puis met à jour la page du profil avec [fetchprofile](#).

```
import axios from 'axios';
import {
  FETCH_PROFILE,
  MODIFY_USERNAME,
} from 'src/actions/profile-private';

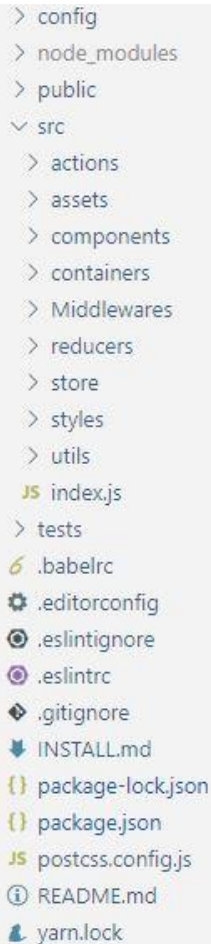
import { fetchMessage } from 'src/actions/event';

const profilePrivateMiddleware = (store) => (next) => (action) => {
  switch (action.type) {
    case MODIFY_USERNAME:
      console.log('c'est le moment de faire l'appel à l'API CURRENT EVENT');
      axios({
        method: 'post',
        url:
          `http://18.207.205.68/profile/${store.getState().profilePrivate.profilePrivate.id}/username/update`,
        headers: {
          Authorization: `Bearer ${store.getState().login.token}`,
        },
        data: {
          username: store.getState().profilePrivate.newUsername,
        },
      })
        .then((response) => {
          console.log('succès : ', response.data);
          store.dispatch(fetchProfile(response.data));
        })
        .catch((error) => {
          console.warn(error.response.data);
        });
      next(action);
      break;
    default:
      next(action);
  }
};

export default profilePrivateMiddleware;
```

VIII. RÉALISATION PERSONNELLE

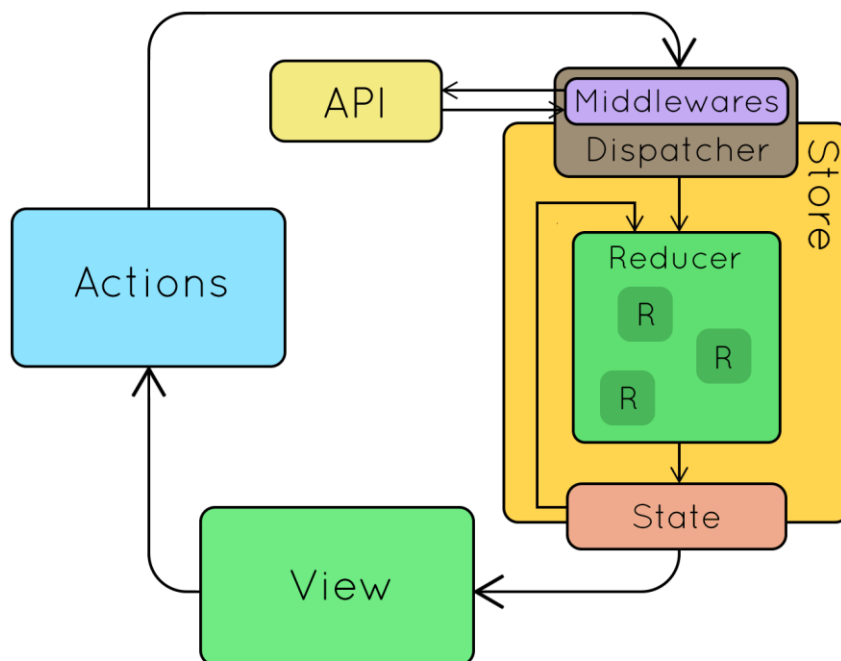
A. Mise en place de l'architecture de l'application



A screenshot of a file explorer showing the project structure. The 'src' directory is expanded, showing subdirectories like 'actions', 'assets', 'components', 'containers', 'Middlewares', 'reducers', 'store', 'styles', and 'utils'. Other files visible include 'index.js', 'tests', '.babelrc', '.editorconfig', '.eslintignore', '.eslintrc', '.gitignore', 'INSTALL.md', 'package-lock.json', 'package.json', 'postcss.config.js', 'README.md', and 'yarn.lock'.

L'application a été conçue côté front-end avec la librairie JavaScript React qui peut s'écrire avec la syntaxe basique de JavaScript mais cela est moins facile à écrire et encore moins à lire, nous avons appris lors de notre apprentissage l'utilisation de la syntaxe JSX qui apporte une souplesse dans l'écriture, dans la lecture et la compréhension qui est par la suite transpilée avec la librairie Babel qui permet de compiler le code JSX en code JavaScript classique compréhensible par les navigateurs. React est combiné avec Redux qui est un système de centralisation des données et des actions. React est particulièrement convaincant lorsqu'on commence mais on se retrouve rapidement confronté à certains problèmes de conceptions lorsque notre application devient un peu plus élaborée qu'une simple liste. Par exemple, nous avons divers composants au sein de notre page mais qu'ils doivent partager certaines propriétés (qui sont local), nous nous retrouvons rapidement dans l'impasse. Avec Redux, on ajoute la notion de *store*. Un *store* est global. Il contient toutes les informations de tous les composants qui y sont rattachés. Lorsqu'un composant **A** lance une action et entraîne une modification du store (dit aussi "state global"), le composant **B** aura également les nouvelles données du composant **A**. Chaque composant peut mettre en place des comportements différents en fonction de l'état des autres. Le store va également contenir les actions disponibles dans les composants.

Ce schéma permet de mieux comprendre le fonctionnement de l'application.



Le Redux Store conserve l'état de l'application. Cet état peut être mis à jour en répartissant les actions. Une action n'est rien de plus qu'un objet JSON avec un type et certains paramètres (par exemple, `{type: 'SET_DATE', date: '2019-05-15T13: 24: 00.000Z'}`). Le magasin dispose d'un réducteur qui, en tenant compte de l'action et de l'état précédent, produira l'état suivant.

React permet de créer des composants qui réagissent aux changements de l'état de l'application. Les composants affectés par un changement d'état sont restitués avec les nouvelles données. Les composants envoient également des actions, par exemple lorsqu'un bouton est cliqué.

B. Mise en place des routeur React

Après la mise en place des dossiers contenant les différentes pages de l'application, j'ai créé des routes permettant d'accéder à c'est différente page avec un chemin d'accès comme par exemple la route "/" était la page d'accueil de l'application ou "/Event" pour accéder à une page d'évènement, j'ai utilisé un package nommé "React-Router-Dom".

La librairie JavaScript React est conçue pour concevoir des applications dites "One page" et donc d'accéder à la page principale de l'application, l'installation du package "React-Router-Dom" permet de pouvoir concevoir une application multipage.

Je vais vous présenter les routes de l'index qui est le point de départ de mon application.

Ici se trouve les imports, je vais chercher dans le module "react-router-dom" le module "Switch" qui englobe les routes, switch cherche la route qui correspond à celle présente dans la barre de recherche et affiche la page qui correspond, le module "Route" permet de construire des routes, j'importe aussi de "material-ui" le module "Grid" que j'utilise pour le responsive dans mon application.

```
// == Import npm
import {
  Switch,
  Route,
} from 'react-router-dom';

import Grid from '@material-ui/core/Grid';
```

Ici se trouve l'import des différentes pages dans l'index de mon application.

```
import Nav from 'src/containers/Nav';
import Home from 'src/containers/Home';
import Event from 'src/containers/Event/event';
import Comment from 'src/components/Event/Comment';
import Games from 'src/containers/Event/eventGames';
import Informations from 'src/containers/Event/eventInformations';
import CreateEvent from 'src/containers/Create-Event';
import ProfilPublic from 'src/components/Profile-public';
import ProfilPrivate from 'src/containers/Profile-private';
import Login from 'src/containers/Login';
import Password from 'src/components/Password';
import Register from 'src/containers/Register';
import Footer from 'src/components/Footer';
import Us from 'src/components/Us';
import Legal from 'src/components/Legal';
import Loading from './loading';
```

Puis les routes de mon application englobée par switch, pour cet exemple on déclare que le chemin d'accès correspondant à la route "/" dans barre de recherche correspond à la page d'accueil importer plus haut.

```
<Route path="/" exact>
  <Home />
</Route>
```

On peut aussi voir "loading" qui est un chargement mis en place qui s'assure que toutes les données ont été récupéré avant d'afficher le contenu de l'application.

Ainsi que "isLoggedIn" qui bloque l'accès aux pages si on n'est pas connecté par défaut la valeur est false, quand on se connecte elle passe à true et permet d'accéder aux pages.

Il y aussi une route "key=404" qui affiche une page d'erreur quand on tape dans la barre de recherche une route qui n'existe pas.

```

import AppStyled from './AppStyled';

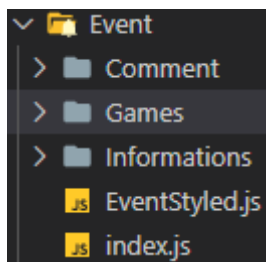
// == Composant
const App = ({}) => {
  return (
    <Grid>
      {loading && <Loading />}
      {!loading && (
        <>
          <Nav />
          <Switch>
            <Route path="/" exact>
              <Home />
            </Route>
            <Route path="/A-propos">
              <Us />
            </Route>
            <AppStyled>
              {isLoggedIn && (
                <>
                  <Route path="/Evenement/:slug">
                    <Event />
                  </Route>
                  <Route path="/Evenement/:slug/Informations/">
                    <Informations />
                  </Route>
                  <Route path="/Evenement/:slug/Jeux/">
                    <Games />
                  </Route>
                  <Route path="/Evenement/:slug/Commentaires/">
                    <Comment />
                  </Route>
                  <Route path="/Créer-un-evenement">
                    <CreateEvent />
                  </Route>
                  <Route path="/Profil-Prive">
                    <ProfilPrivate />
                  </Route>
                  <Route path="/Profil-Public">
                    <ProfilPublic />
                  </Route>
                </>
              )}
            <Route path="/Connexion">
              <Login />
            </Route>
            <Route path="/Mot-de-passe-oublié">
              <Password />
            </Route>
            <Route path="/Inscription">
              <Register />
            </Route>
            <Route path="/Mentions-legales">
              <Legal />
            </Route>
          </AppStyled>
          <Route key="404">
            <div>Area 404: you're not allowed here !</div>
          </Route>
        </Switch>
        <Footer />
      </>
    </Grid>
  );
};

// == Export
export default App;

```

C. Intégration des pages

Les différentes pages de l'application sont présentes dans le dossier "components", chaque dossier est composé au minimum d'un fichier index.js ou se trouve tout le contenu et d'un fichier comportant le nom de la page suivi de Styled.js ou juste Styled.js, ce fichier est le fichier qui permet de définir la forme (CSS) avec le module "Styled-Components" ont défini en premier une balise par exemple une div et on lui applique un style, par la suite le fichier Styled.js est importé dans le component qui correspond et il englobe le contenu du component, le style défini n'impacte que le contenu englobé, ça évite d'avoir des conflits entre les pages qui ont les mêmes "class" qui est d'ailleurs écrit "classname" en syntaxe JSX.



Je vais vous présenter la page d'un événement ou on peut retrouver les différentes informations concernant l'événement telle que le nom, sa catégorie, le nombre de joueur inscrit sur le nombre maximal possible, puis en bas un menu qui affiche l'adresse, les jeux et les commentaires.



+ INFORMATIONS
+ LES JEUX
+ LES COMMENTAIRES

Paris, 18 rue du chou
2020-07-16T00:00:00+00:00

France
Agrandir le plan
Itinéraires

Mentions légales
A Propos de Nous

+ INFORMATIONS
+ LES JEUX
+ LES COMMENTAIRES

tony
hey

Saisissez votre message...

Mentions légales
A Propos de Nous

Dans un composant on retrouve en premier les imports nécessaires au fonctionnement de la page :

```
// == Import npm
import React from 'react';
import PropTypes from 'prop-types';
import {
  NavLink,
  useParams,
} from 'react-router-dom';
import slugify from 'slugify';
import { getEventBySlug } from 'src/utlis/selectors';

import CurrentEvent from 'src/containers/Current-Event';

import PeopleIcon from '@material-ui/icons/People';
import GamepadIcon from '@material-ui/icons/Gamepad';
import AddCircleOutlineIcon from '@material-ui/icons/AddCircleOutline';
import MarkunreadIcon from '@material-ui/icons/Markunread';
import AllInclusiveIcon from '@material-ui/icons/AllInclusive';

import EventStyled from './EventStyled';
```

1. On retrouve toujours l'import de la librairie *React*.
2. "*Prop-types*" qui permet de vérifier que les valeurs reçues sont correctes.
3. "*React-router-dom*" pour les routes comme expliqué précédemment.
4. "*Slugify*" & "*getEventBySlug*" permet de transformer le titre d'un évènement en lien correct en remplaçant les espaces par des barres, exemple : "Un super titre" avec slugify ça devient "Un-super-titre" et apparait ainsi dans la barre d'adresse lorsque l'on accède à un évènement sur la page d'accueil, ce titre modifié est récupéré puis comparé dans l'Api avec chaque titre des évènements pour afficher les données qui corresponde au même titre dans la page de l'évènement.
5. "*Current-Event*" c'est un bouton disposé à droite de la page, au clique une fenêtre arrivant de la droite s'affiche avec la liste des évènements auquel l'utilisateur participe.
6. "*Material-ui*" J'importe des icones de cette librairie.
7. "*EventStyled*" C'est ma page de style.

On retrouve ensuite des variables, puis Slugify qui fait son job expliqué précédemment et enfin *“handleSubmitJoin”* s’active aux cliques sur le bouton “Participer à l’événement” et permet d’ajouter l’évènement cliqué dans liste des évènements auquel l’utilisateur participe.

```
const Event = ({
  events,
  Join,
  errorJoin,
}) => {
  const { slug } = useParams();

  const eventfound = getEventBySlug(slug, events);

  const sluglink = slugify(eventfound.title, {
    lower: true,
    remove: /[*+~.()'"!:@]/g,
  });

  const handleSubmitJoin = (evt) => {
    evt.preventDefault();
    Join(eventfound.id);
  };
};
```

Ensuite le contenu de la page récupéré par l'Api : titre, organisateur, catégories etc. Puis un menu qui affiche différentes informations tout en conservant ce qui se trouve au-dessus.

```
return (
  <EventStyled>
    <CurrentEvent />
    <h1 className="title-event">{eventfound.title}</h1>
    <p className="author"><img className="avatar" src={eventfound.organizeBy.avatar} alt="Avatar" />
      {eventfound.organizeBy.username}</p>
    <p className="icon-category"><AllInclusiveIcon className="icon-event" />
      {eventfound.category.name}</p>
    <p className="icon-nbr"><PeopleIcon className="icon-event" />
      {eventfound.participateBy.length}/
      {eventfound.maxParticipants}
    </p>
    <ul className="listtags">
      {eventfound.tags.map((tag) => (
        <li key={tag.id} className="tag">{tag.name}</li>
      ))}
    </ul>
    <p className="text">
      {eventfound.description}
    </p>
    <form onSubmit={handleSubmitJoin}>
      <button className="action-join" type="submit">Participer à l'événement</button>
    </form>
    <p className="response"> {errorJoin} </p>
    <nav className="nav">
      <NavLink
        className="navigation-item"
        activeClassName="navigation-item-active"
        to={` /Evenement/${sluglink}/Informations`}
      >
        <AddCircleOutlineIcon className="icon" /> Informations
      </NavLink>

      <NavLink
        to={` /Evenement/${sluglink}/Jeux`}
        className="navigation-item"
        activeClassName="navigation-item-active"
      >
        <GamepadIcon className="icon" /> Les jeux
      </NavLink>

      <NavLink
        to={` /Evenement/${sluglink}/Commentaires`}
        className="navigation-item"
        activeClassName="navigation-item-active"
      >
        <MarkunreadIcon className="icon" /> Les commentaires
      </NavLink>
    </nav>
  </EventStyled>
);
};
```

On peut voir que le contenu est englobé par `<EventStyled>` il est importé du fichier `Styled.js`, ce fichier importe la librairie `"styled"` pour son fonctionnement, il crée une `<div>` puis donne du style à ce qui se trouve à l'intérieur sans impacter le reste de l'application, on retrouve aussi l'import `"theme"` qui est un fichier de style utilisé sur toute l'application ou l'on définit des valeurs globales qui sont commune telle que la couleur.

```

import styled from 'styled-components';
import theme from 'src/styles/theme';

export default styled.div`
  text-align: center;
  margin-top: 100px;
  .nav{
    margin-top: 20px;
    border-bottom: 5px solid ${theme.colors.mainColor};
  }
  .navigation-item{
    display: inline-flex;
    padding: 5px;
    text-transform: uppercase;
    margin-right: 10px;
    &-active{
      border-bottom: 5px solid ${theme.colors.bluecolor};
    }
    .icon{
      margin-right: 10px;
    }
  }
  .title-event{
    text-align: left;
    text-transform: uppercase;
    padding: 10px;
    border-left: solid 3px ${theme.colors.mainColor};
    margin: 5px;
  }
  .icon-category{
    border-bottom: 3px solid ${theme.colors.mainColor};
    padding: 5px;
    margin: 5px;
    display: inline-flex;
  }
  .icon-nbr{
    padding: 5px;
    margin: 5px;
    display: inline-flex;
  }
  .icon-event{
    margin-right: 10px;
  }
  .tag{
    display: inline-block;
    padding: 10px;
    margin: 10px;
    border: 3px solid ${theme.colors.bluecolor};
    border-radius: 20px;
    &:hover{
      color: white;
      background-color: ${theme.colors.bluecolor};
    }
  }
}

```

```

  .listtags{
    margin-top: 10px;
  }
  .text{
    margin: 50px 0px;
  }
  .avatar{
    height: 30px;
    width: 30px;
    margin-right: 10px;
    margin-left: 10px;
  }
  .author{
    margin-bottom: 10px;
  }
  .current-event{
    border: 3px solid ${theme.colors.bluecolor};
    border-radius: 20px;
    position: fixed;
    right: 0;
    top: 50%;
    &:hover{
      background-color: white;
    }
  }
  .action-join{
    color: ${theme.colors.bluecolor};
    width: 80%;
    font-size: 1.5em;
    background-color: #F5F4FC;
    padding: 10px;
    border: solid 3px ${theme.colors.bluecolor};
    border-radius: 0.5em;
    text-transform: uppercase;
    &:hover{
      background-color: ${theme.colors.bluecolor};
      color: #F5F4FC;
      transition: background-color 1s;
    }
  }
  .response{
    padding: 10px;
  }
}
;

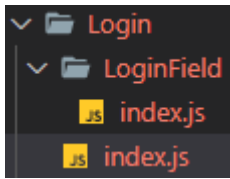
```


En dernier on retrouve les variables vérifiées par le module “*propTypes*”

```
Event.propTypes = {
  events: PropTypes.objectOf(
    PropTypes.shape({
      listevents: PropTypes.objectOf(
        PropTypes.shape({
          id: PropTypes.number.isRequired,
          category: PropTypes.object.isRequired,
          title: PropTypes.string.isRequired,
          organizeBy: PropTypes.objectOf(
            PropTypes.shape({
              username: PropTypes.string.isRequired,
              avatar: PropTypes.string.isRequired,
            }).isRequired,
          ),
          maxParticipants: PropTypes.number.isRequired,
          description: PropTypes.string.isRequired,
          tags: PropTypes.arrayOf(
            PropTypes.shape({
              name: PropTypes.string.isRequired,
              id: PropTypes.number.isRequired,
            }).isRequired,
          ).isRequired,
          participateBy: PropTypes.arrayOf(PropTypes.number).isRequired,
        }).isRequired,
      ).isRequired,
    }).isRequired,
  ).isRequired,
  Join: PropTypes.func.isRequired,
  errorJoin: PropTypes.string.isRequired,
};

// == Export
export default Event;
```

D. Système d'authentification



Une fois inscrit il faut se rendre sur le formulaire de connexion, pour se connecter il faut renseigner son pseudo et son mot de passe, le dossier de la page est nommé "Login" qui est composé d'un fichier index.js et d'un dossier "LoginField" avec un fichier index.js, dans ce fichier se trouve un champ input de formulaire que j'import sur l'index.js principale pour le réutiliser autant que j'ai besoin en modifiant les valeurs, ça me permet de réduire et d'optimiser mon code.

Le fichier "LoginField/index.js" est un input de formulaire avec des variables qui sont récupéré sur l'index.js principale pour réutiliser l'input.

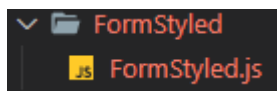
```
const Field = ({
  type,
  description,
  name,
  onChange,
  value,
}) => {
  const handleChange = (evt) => {
    onChange(evt.target.value, name);
  };

  const inputId = `field-${name}`;

  return (
    <div className="input-container">
      <label className="label" htmlFor={inputId}>
        {description}
        <input
          id={inputId}
          placeholder={description}
          className="input"
          type={type}
          name={name}
          autoComplete="off"
          value={value}
          onChange={handleChange}
          required
        />
      </label>
    </div>
  );
};
```

Les balises `<Field/>` c'est le fichier `"LoginField/index.js"` importé avec les valeurs du champ.

```
return (
  <div className="login-form">
    {isLoggedIn && (
      <Redirect to="/" />
    )}
    {!isLoggedIn && (
      <FormStyled onSubmit={handleSubmit}>
        <h1 className="title">Se connecter</h1>
        <Field
          description="Adresse email"
          name="username"
          onChange={changeField}
          value={username}
        />
        <Field
          type="password"
          description="Mot de passe"
          name="password"
          onChange={changeField}
          value={password}
        />
        <p className="response"> {errorlogin} </p>
        <div className="action">
          <button className="action-button" type="submit">Connexion</button>
        </div>
        <div className="links">
          <Link className="links-item" to="/Inscription">Créer un compte</Link> / <Link
            className="links-item" to="/Mot-de-passe-oublié">Mot de passe oublié</Link>
          </div>
        </FormStyled>
      )}
    </div>
  );
};
```



`</FormStyled>` c'est le style de la page qui est récupéré du dossier `"FormStyled"`, ce style est réutilisé pour tous les formulaires : inscription, créer un évènement, connexion etc.

```
const handleSubmit = (evt) => {
  evt.preventDefault();
  handleLogin();
};
```

Au clic sur connexion l'action `onSubmit "handlesubmit"`, l'information est envoyée au middleware correspondant qui effectue un appel Axios, l'api vérifie si le pseudo et mot de passe corresponde, si oui l'utilisateur récupère un JWT avec ses informations et est renvoyé sur la page d'accueil.

```

import axios from 'axios';

import { fetchProfile } from 'src/actions/profile-private';
import { LOG_IN, saveUser, saveErrorLogin } from 'src/actions/login';

const userMiddleware = (store) => (next) => (action) => {
  switch (action.type) {
    case LOG_IN:
      console.log('c'est le moment de faire l'appel à l'API');
      axios.post('http://18.207.205.68/login_check', {
        username: store.getState().login.username,
        password: store.getState().login.password,
      })
        .then((response) => {
          console.log('succès LOGIN : ', response.data);
          store.dispatch(saveUser(response.data.token));
        })
        .catch(() => {
          store.dispatch(saveErrorLogin('Votre connexion n\' a pas pu aboutir, votre mot de passe ou
votre email est invalide'));
        })
        .finally(() => {
          store.dispatch(fetchProfile());
        });
      next(action);
      break;
    default:
      next(action);
  }
};
export default userMiddleware;

```

IX. VULNÉRABILITÉS DE SÉCURITÉ ET VEILLE

A. JSON Web Token

Le JWT avait été abordé rapidement en cours, lors de la mise en pratique j'ai effectué des recherches pour mieux comprendre son fonctionnement et comment sens servir, on peut voir une de mes recherches que j'ai effectuées sur le site jwt.io dans le chapitre "Extrait et traduction d'une ressource anglophone pour le projet".

Un JWT est une session d'un utilisateur qui contient un ensemble d'informations sur un utilisateur authentifié qui est conservé temporairement en mémoire. Pour que le serveur puisse faire confiance à ces informations, comme si elles provenaient de lui-même, il lui suffit de les signer et d'envoyer cette signature avec les informations. Quand elles lui seront retournées, il sera alors en mesure de vérifier l'authenticité de ces informations en comparant la signature reçue à la signature calculée. Les informations du jeton circulent en clair, par conséquent aucune données confidentielles est stocké dans le contenu du jeton JWT. L'essentiel est de pouvoir s'assurer que ce contenu n'a pas changé depuis qu'il a été signé par le serveur. Ce qui veut dire que toutes les applications clientes qui vont exploiter les données du jeton devront s'assurer de son authenticité, en vérifiant sa signature.

B. Persistance de la connexion

Après la mise en place de la connexion et de la récupération du Json Web Token un problème est t'apparut, quand je rafraichissais la page le JWT récupéré lors de la connexion disparaissait comme si je mettais déconnecté, après de multiple recherche j'ai compris que le JWT n'était pas gardé en mémoire par Redux, j'ai dû mettre en place un module nommé "Redux-Phenix" qui m'a permis de conserver le JWT même après rafraichissement.

X. EXTRAIT ET TRADUCTION D'UNE RESSOURCE ANGLOPHONE POUR LE PROJET.

A. Extrait

Lors d'une recherche sur le fonctionnement de JWT je me suis rendu sur le site [web jwt.io/introduction](https://jwt.io/introduction), en voici un extrait :

What is JSON Web Token?

JSON Web Token (JWT) is an open standard ([RFC 7519](https://tools.ietf.org/html/rfc7519)) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with the **HMAC** algorithm) or a public/private key pair using **RSA** or **ECDSA**.

Although JWTs can be encrypted to also provide secrecy between parties, we will focus on *signed* tokens. Signed tokens can verify the *integrity* of the claims contained within it, while encrypted tokens *hide* those claims from other parties. When tokens are signed using public/private key pairs, the signature also certifies that only the party holding the private key is the one that signed it.

When should you use JSON Web Tokens?

Here are some scenarios where JSON Web Tokens are useful:

- **Authorization:** This is the most common scenario for using JWT. Once the user is logged in, each subsequent request will include the JWT, allowing the user to access routes, services, and resources that are permitted with that token. Single Sign On is a feature that widely uses JWT nowadays, because of its small overhead and its ability to be easily used across different domains.
- **Information Exchange:** JSON Web Tokens are a good way of securely transmitting information between parties. Because JWTs can be signed—for example, using public/private key pairs—you can be sure the senders are who they say they are. Additionally, as the signature is calculated using the header and the payload, you can also verify that the content hasn't been tampered with.

How do JSON Web Tokens work?

In authentication, when the user successfully logs in using their credentials, a JSON Web Token will be returned. Since tokens are credentials, great care must be taken to prevent security issues. In general, you should not keep tokens longer than required.

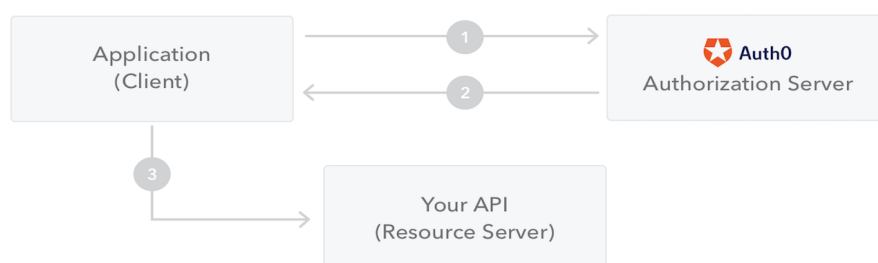
You also [should not store sensitive session data in browser storage due to lack of security](#).

Whenever the user wants to access a protected route or resource, the user agent should send the JWT, typically in the **Authorization** header using the **Bearer** schema. The content of the header should look like the following:

This can be, in certain cases, a stateless authorization mechanism. The server's protected routes will check for a valid JWT in the **Authorization** header, and if it's present, the user will be allowed to access protected resources. If the JWT contains the necessary data, the need to query the database for certain operations may be reduced, though this may not always be the case.

If the token is sent in the **Authorization** header, Cross-Origin Resource Sharing (CORS) won't be an issue as it doesn't use cookies.

The following diagram shows how a JWT is obtained and used to access APIs or resources:



1. The application or client requests authorization to the authorization server. This is performed through one of the different authorization flows. For example, a typical [OpenID Connect](#) compliant web application will go through the `/oauth/authorize` endpoint using the [authorization code flow](#).
2. When the authorization is granted, the authorization server returns an access token to the application.
3. The application uses the access token to access a protected resource (like an API).

Do note that with signed tokens, all the information contained within the token is exposed to users or other parties, even though they are unable to change it. This means you should not put secret information within the token.

B. Traduction

Qu'est-ce que le jeton Web JSON?

JSON Web Token (JWT) est une norme ouverte (RFC 7519) qui définit une manière compacte et autonome de transmettre en toute sécurité des informations entre les parties en tant qu'objet JSON. Ces informations peuvent être vérifiées et fiables car elles sont signées numériquement. Les JWT peuvent être signés à l'aide d'un secret (avec l'algorithme HMAC) ou d'une paire de clés publique / privée à l'aide de RSA ou ECDSA.

Bien que les JWT puissent être chiffrés pour assurer également la confidentialité entre les parties, nous nous concentrerons sur les jetons signés. Les jetons signés peuvent vérifier l'intégrité des revendications qui y sont contenues, tandis que les jetons chiffrés cachent ces revendications aux autres parties. Lorsque les jetons sont signés à l'aide de paires de clés publiques / privées, la signature certifie également que seule la partie détenant la clé privée est celle qui l'a signée.

Quand devez-vous utiliser des jetons Web JSON?

Voici quelques scénarios où les jetons Web JSON sont utiles :

- Autorisation : il s'agit du scénario le plus courant pour utiliser JWT. Une fois l'utilisateur connecté, chaque demande ultérieure inclura le JWT, permettant à l'utilisateur d'accéder aux routes, services et ressources autorisés avec ce jeton. La connexion unique est une fonctionnalité qui utilise largement JWT de nos jours, en raison de sa faible surcharge et de sa capacité à être facilement utilisée dans différents domaines.
- Échange d'informations : les jetons Web JSON sont un bon moyen de transmettre des informations en toute sécurité entre les parties. Étant donné que les JWT peuvent être signés (par exemple, en utilisant des paires de clés publiques / privées), vous pouvez être sûr que les expéditeurs sont bien ceux qu'ils disent être. De plus, comme la signature est calculée à l'aide de l'entête et de la charge utile, vous pouvez également vérifier que le contenu n'a pas été falsifié.

Comment fonctionnent les jetons Web JSON?

Dans l'authentification, lorsque l'utilisateur se connecte avec succès à l'aide de ses informations d'identification, un jeton Web JSON sera retourné. Étant donné que les jetons sont des informations d'identification, un grand soin doit être pris pour éviter les problèmes de sécurité. En général, vous ne devez pas conserver les jetons plus longtemps que nécessaire.

Vous ne devez pas non plus stocker de données de session sensibles dans le stockage du navigateur en raison d'un manque de sécurité .

Chaque fois que l'utilisateur souhaite accéder à une route ou une ressource protégée, l'agent utilisateur doit envoyer le JWT, généralement dans l'entête d'autorisation à l'aide du schéma de support. Le contenu de l'entête doit ressembler à ceci :

Authorization: Bearer <token>

Il peut s'agir, dans certains cas, d'un mécanisme d'autorisation apatride. Les routes protégées du serveur vérifieront un JWT valide dans l'entête, et s'il est présent, l'utilisateur sera autorisé à accéder aux ressources protégées. Si le JWT contient les données nécessaires, la nécessité d'interroger la base de données pour certaines opérations peut être réduite, bien que cela ne soit pas toujours le cas.

Si le jeton est envoyé dans l'entête, le partage de ressources d'origine croisée (CORS) ne sera pas un problème car il n'utilise pas de cookies.

Le diagramme suivant montre comment un JWT est obtenu et utilisé pour accéder aux API ou aux ressources :

[.....]

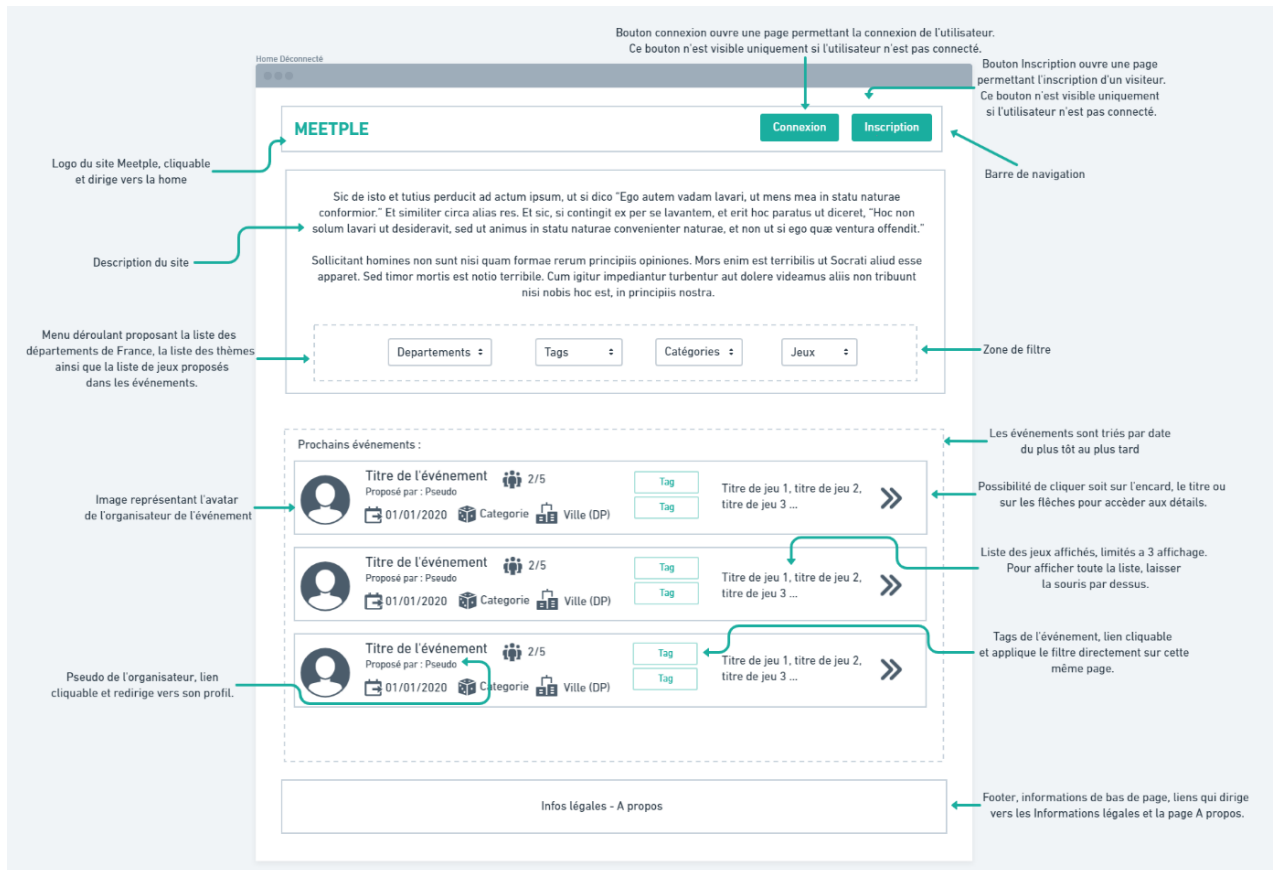
1. L'application ou le client demande l'autorisation au serveur d'autorisation. Cette opération s'effectue via l'un des différents flux d'autorisation. Par exemple, une application Web compatible OpenID Connect Typique passera par le /oauth/authorize de terminaison à l'aide du flux de code d'autorisation .
2. Lorsque l'autorisation est accordée, le serveur d'autorisation renvoie un jeton d'accès à l'application.
3. L'application utilise le jeton d'accès pour accéder à une ressource protégée (comme une API).

Notez qu'avec les jetons signés, toutes les informations contenues dans le jeton sont exposées aux utilisateurs ou à d'autres parties, même si elles ne sont pas en mesure de les modifier. Cela signifie que vous ne devez pas mettre d'informations secrètes dans le jeton.

XII. ANNEXES

A. Wireframes

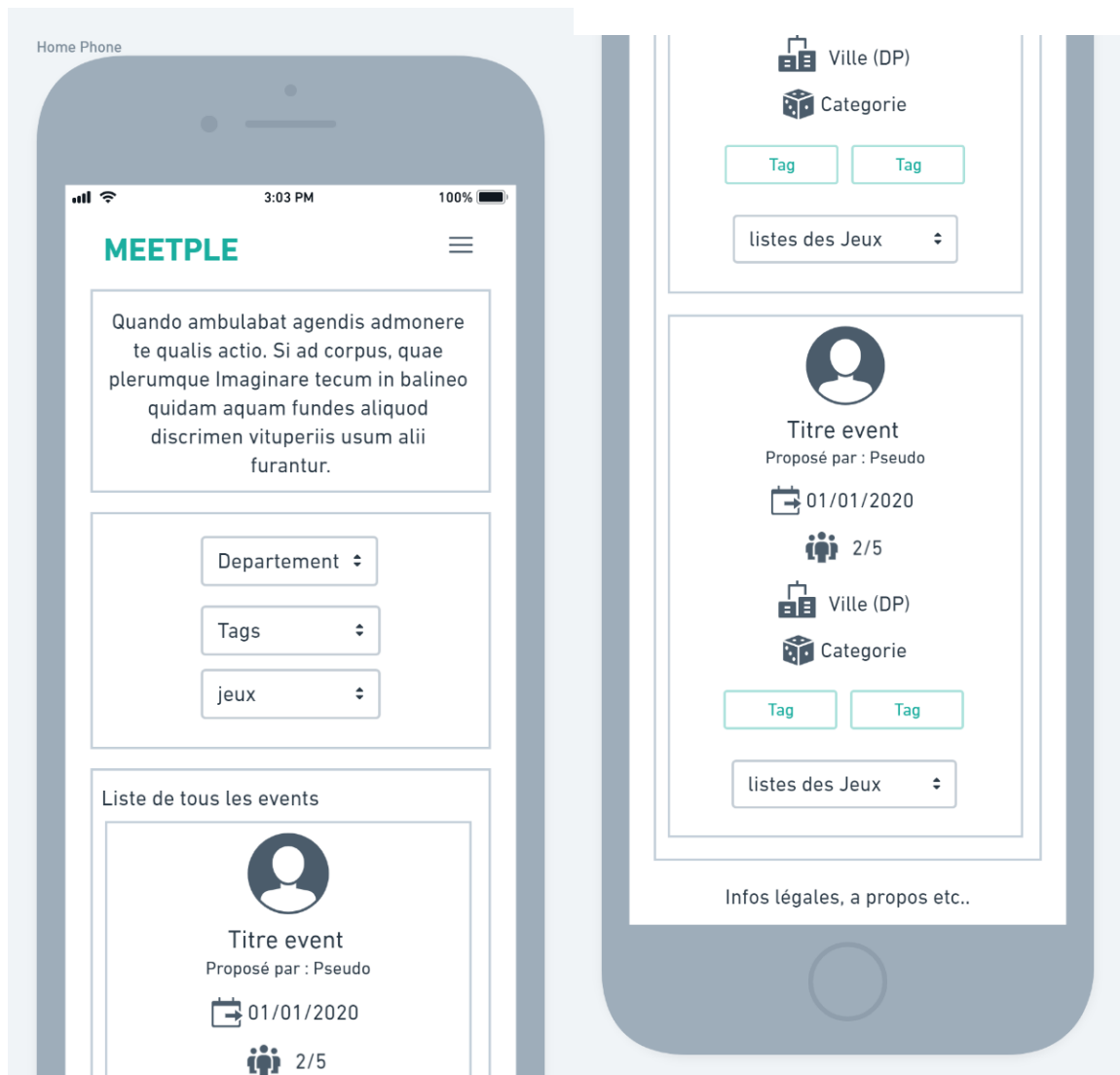
Page d'accueil version desktop.



Menu quand on est connecté.



Page d'accueil version smartphone.



Page profil version desktop.

Le bouton Créer un événement ouvre une fenêtre modal permettant de créer son propre événement

Page profil privé

MEETPLE

Créer un événement Déconnexion

E-mail

Pseudo

Mot de passe

Evenement créés:

Titre de l'événement

01/01/2020

Ville (DP) 2/5

Supprimer l'évènement

Titre de l'événement

01/01/2020

Ville (DP) 2/5

Supprimer l'évènement

Titre de l'événement

01/01/2020

Ville (DP) 2/5

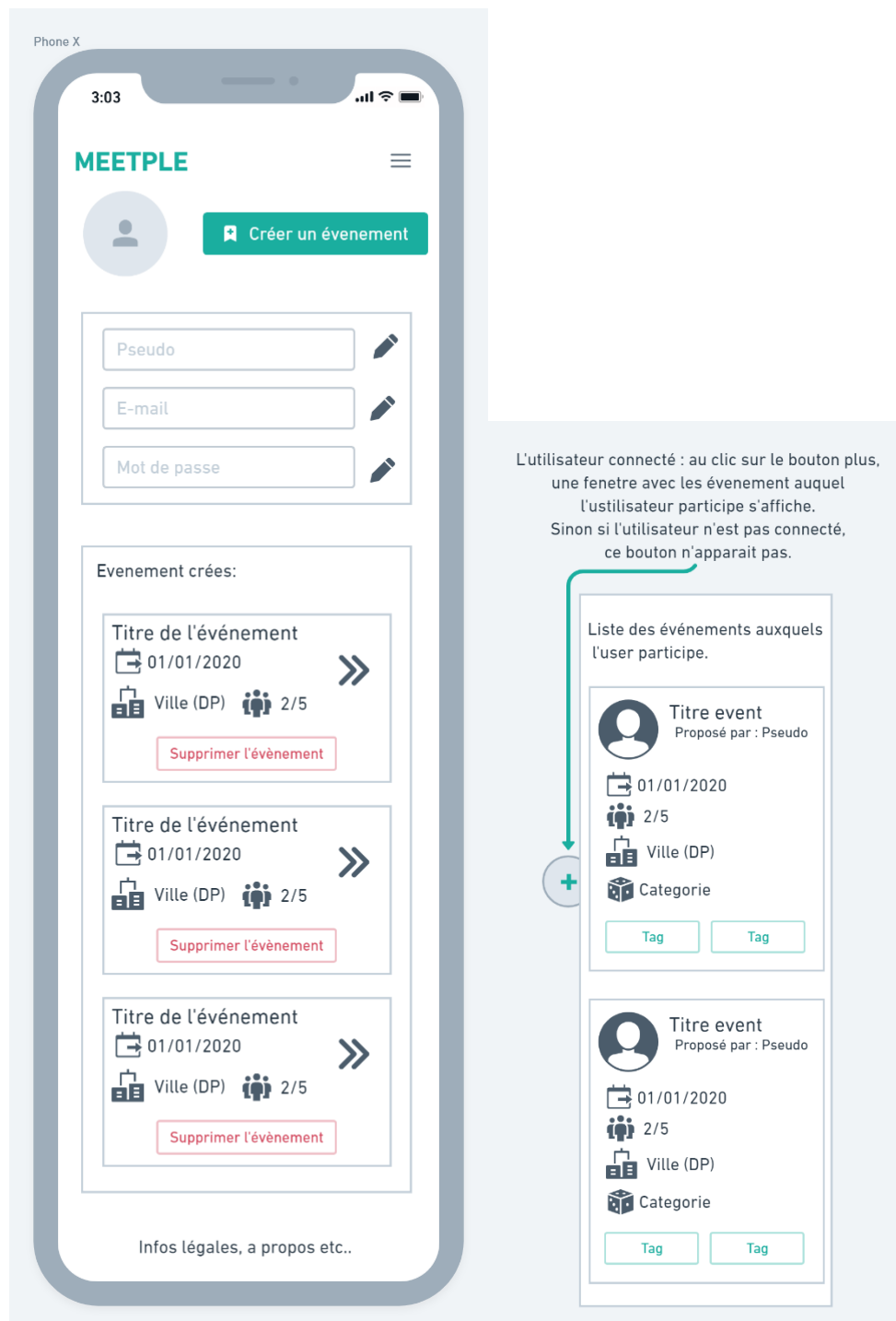
Supprimer l'évènement

Infos légales, a propos etc..

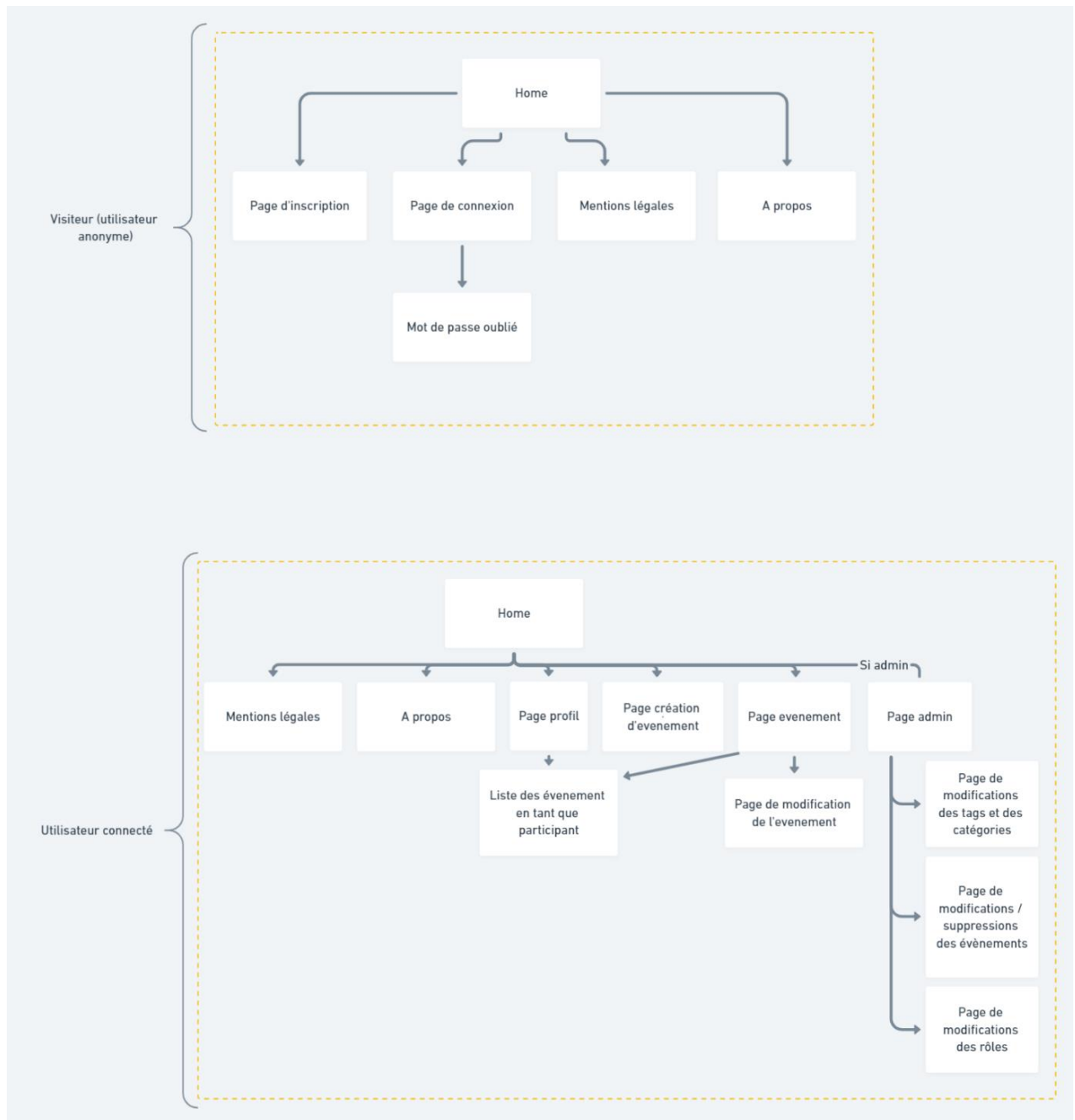
Au clic sur le bouton : possibilité de modifier le texte sélectionné

L'utilisateur connecté : au clic sur le bouton plus, une fenêtre avec les événement auquel l'utilisateur participe s'affiche.
Sinon si l'utilisateur n'est pas connecté, ce bouton n'apparaît pas.

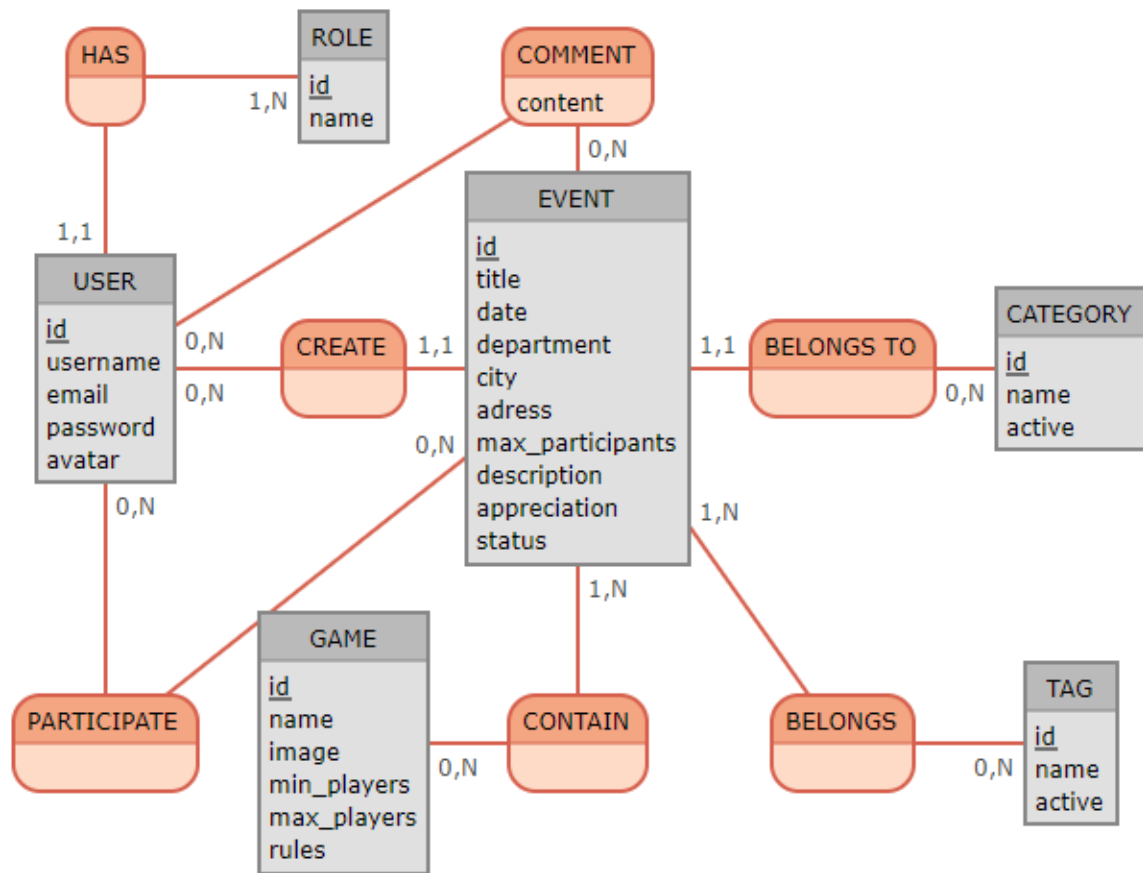
Page profil version smartphone | Fenêtre surgissant de la droite au clique affichant les évènements auquel l'utilisateur est inscrit, ce bouton est présent sur la page profil et la page des évènements.



B. Worflow



C. MCD



D. Dictionnaire de données

m

Utilisateur (user)

Champ	Type	Spécificités	Description
id	INT	PRIMARY KEY, NOT NULL, UNSIGNED, AUTO_INCREMENT	L'identifiant de l'utilisateur
username	VARCHAR(64)	NOT NULL	Le pseudo de l'utilisateur
password	VARCHAR(64)	NOT NULL	Le mot de passe de l'utilisateur
email	VARCHAR(64)	NOT NULL	L'adresse e-mail de l'utilisateur
avatar	VARCHAR(128)	NULL	L'URL de l'avatar de l'utilisateur
created_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	La date de création de l'utilisateur
updated_at	TIMESTAMP	NULL	La date de la dernière mise à jour de l'utilisateur

Theme des events (tag)

Champ	Type	Spécificités	Description
id	INT	PRIMARY KEY, NOT NULL, UNSIGNED, AUTO_INCREMENT	L'identifiant du theme
name	VARCHAR(64)	NOT NULL	Nom du thème
active	BOOL	NOT NULL	Rend l'élément actif ou non
created_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	La date de création du theme
updated_at	TIMESTAMP	NULL	La date de la dernière mise à jour du theme

Jeux des events (games)

Champ	Type	Spécificités	Description
id	INT	PRIMARY KEY, NOT NULL, UNSIGNED, AUTO_INCREMENT	L'identifiant du jeu
name	VARCHAR(64)	NOT NULL	le nom du jeu
description	TEXT	PRIMARY KEY, NOT NULL, UNSIGNED, AUTO_INCREMENT	La description du jeu
image	VARCHAR(128)	NULL	L'url de l'image du jeu
min_players	INT	NULL	Joueur minimum
max_players	INT	NULL	Joueur maximum
created_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	La date de création du jeu
updated_at	TIMESTAMP	NULL	La date de la dernière mise à jour du jeu

Evenements (event)

Champ	Type	Spécificités	Description
id	INT	PRIMARY KEY, NOT NULL, UNSIGNED, AUTO_INCREMENT	L'identifiant de l'utilisateur
title	VARCHAR(64)	NOT NULL	titre du thème
date	TIMESTAMP	NOT NULL	date de l'événement
department	TEXT	NOT NULL	département de l'événement
city	TEXT	NOT NULL	ville de l'événement
adress	TEXT	NOT NULL	adresse de l'événement
max_participants	INT	NOT NULL	nombre maximum de participants
description	TEXT	NOT NULL	description de l'événement
appreciation	TEXT	NOT NULL	appréciation de l'événement
status	TEXT	NOT NULL	status de l'événement
created_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	La date de création de l'événement
updated_at	TIMESTAMP	NULL	La date de la dernière mise à jour de l'événement

Roles des utilisateurs (`role`)

Champ	Type	Spécificités	Description
id	INT	PRIMARY KEY, NOT NULL, UNSIGNED, AUTO_INCREMENT	L'identifiant de l'utilisateur
name	VARCHAR(64)	NOT NULL	Nom du rôle
created_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	La date de création du role
updated_at	TIMESTAMP	NULL	La date de la dernière mise à jour du role

Commentaire de l'event (`message`)

Champ	Type	Spécificités	Description
id	INT	PRIMARY KEY, NOT NULL, UNSIGNED, AUTO_INCREMENT	L'identifiant de l'utilisateur
content	TEXT	NOT NULL	Le commentaire
created_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	La date de création du commentaire
updated_at	TIMESTAMP	NULL	La date de la dernière mise à jour du commentaire

Catégorie de l'event (`category`)

Champ	Type	Spécificités	Description
id	INT	PRIMARY KEY, NOT NULL, UNSIGNED, AUTO_INCREMENT	L'identifiant de l'utilisateur
name	VARCHAR(64)	NOT NULL	Nom de la catégorie
active	BOOL	NOT NULL	Rend l'élément actif ou non
created_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	La date de création du commentaire
updated_at	TIMESTAMP	NULL	La date de la dernière mise à jour du commentaire