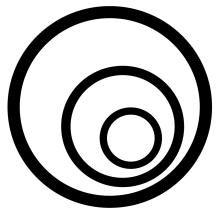


DOSSIER DE PROJET



microscope-web

“A fractal role-playing game of epic histories”

PRÉSENTÉ ET SOUTENU PAR
PABLO PRIETO

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

CENTRE DE FORMATION O'CLOCK

PROMOTION “BLOB”

JANVIER-JUILLET 2022

Référent de formation : Charles Hoffman



SOMMAIRE

I. Remerciements	p.4
II. Introduction	p.5
III. Liste des Compétences couvertes par le projet	p.7
A. Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité	p.7
B. Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité	p.8
IV. Présentation du Projet	p.9
V. Besoins & Objectifs	p.10
VI. Fonctionnalités	p.11
A. MVP (Minimal Viable Product)	p.11
B. Évolutions Potentielles	p.11
C. Arborescence de l'application	p.12
D. Routes Back-End	p.12
E. User Stories du MVP	p.13
VII. Spécifications Techniques	p.14
A. Versioning	p.14
B. Workflow et technologies	p.14
VIII. Conception du Projet	p.17
A. WireFrames	p.17
B. Charte graphique	p.18
IX. Déroulement du Projet	p. 19
A. Présentation de l'équipe	p.19
B. Méthode de Travail	p.20
C. Outils de travail	p. 21

X. Réalisations personnelles	p.23
A. Sprint #0	p.23
B. Sprint #1	p.28
C. Sprint #2	p.30
D. Sprint #3	p.34
E. Sprint #4	p.36
XI. Difficultés rencontrées et solutions apportées	p.37
A. En tant que Product Owner	p.37
B. En tant que Lead Dev Back	p.38
XII. Fonctionnalités représentatives et Jeu d'Essai	p.41
XIII. Vulnérabilités de Sécurité et Veille	p.45
XIV. Extrait et Traduction d'une Ressource Anglophone pour le Projet	p.50
XV. Conclusion	p. 52
XVI. Annexes	p.53



I. REMERCIEMENTS

Mon passage par le centre de formation O'clock m'a véritablement permis de me plonger dans le secteur du développement Web. Enrichi de ces nouvelles compétences, je sais à présent quelle direction suivre pour entreprendre un nouveau parcours professionnel. J'aurais pu imaginer apprendre tout cela seul, mais cela aurait été le fruit de beaucoup de long tâtonnements et d'une accumulation d'incertitudes. J'aurais été également bien plus aveugle à cette industrie, ne sachant pas comment me placer ni me présenter.

Je tiens à remercier **mes proches** qui m'ont toujours fait confiance, ont consolidé tout au long de mon aventure mes aspirations, m'ont permis de mettre de côté mes doutes et a atténué ma fatigue.

Je remercie **l'équipe O'clock**. Grâce à cette formation, j'ai réalisé une montée en compétences extraordinaire en allant droit au but. J'ai maintenant une meilleure idée de mon futur métier et de ces acteurs. Même si le sentiment d'imposteur ne me quittera véritablement jamais, j'avance avec confiance. Et ça, je le dois en partie au centre de formation O'clock.

Enfin, je remercie tous **mes coéquipiers de la promotion Blob** dont leur volonté et leur ténacité nous ont permis d'avancer dans ce mois intensif afin de réaliser au mieux un projet très (trop?) ambitieux.



II. INTRODUCTION

Les confinements successifs qui ont ponctué les années précédant ma formation ont permis à des jeux s'inspirant du format “jeu de plateau” de gagner en popularité. Ainsi, on a pu voir émerger des adaptations de jeux de plateau en format digital tels que, Skribbl.io s'inspirant de Pictionary , Gartic Phone pour le Téléphone arabe mais aussi, Wolfy et Among Us s'inspirant de jeu du Loup-Garou.

Aussi, lorsqu'il nous a été suggéré de proposer des projets de fin de formation, j'ai assez vite pensé à cette tendance. J'avais beaucoup apprécié les applications dont je vous parlais précédemment et j'avais envie de proposer un projet un peu atypique. Le choix de s'atteler à ce type de projet entraînerait l'usage d'algorithmes. Avant la formation, j'avais l'habitude de faire beaucoup d'exercices qui me confrontaient à de l'algorithmie. Je ne les réussissais pas tous et j'y passais beaucoup trop de temps. J'ai persisté et j'ai fini par prendre goût à relever ce type de défis. Étant donné que les occasions de se confronter à des algorithmes se font rares à l'école O'clock, j'avais très envie de renouer avec cette pratique.

En résumé, je souhaitais réaliser :

- une application jeu qui serait l' adaptation d'un *game design* déjà conçu
- un jeu limpide, facile d'accès
- un projet dont le développement permette de se confronter à des problèmes algorithmique

En tant qu'amateur de jeu de rôle indépendant, j'ai joué avec ma compagne à un jeu collaboratif qui correspondait bien à ces critères. Il s'agit d'un livre de Ben Robbins appelé “Microscope”, sous-titré “a fractal role-playing game of epic histories”.

Les règles étaient relativement simples, il était donc facile d'imaginer une adaptation de ce jeu en ligne.

J'ai alors contacté l'auteur de l'édition par l'intermédiaire de sa maison d'édition américaine, “Lame Mage Productions”, afin d'avoir son autorisation. À ma grande surprise, j'ai reçu le lendemain une réponse positive pour réaliser un projet étudiant. L'auteur était très curieux de voir comment son jeu pouvait être adapté en application.

J'ai donc soumis ce projet. Ce projet me paraissait quelque peu ambitieux pour le temps qui nous était imparti. J'ai alors proposé une présentation honnête et transparente sur la

charge de travail induite par la réalisation de ce projet afin que ma future équipe soit préparée au volume de travail auquel elle devra faire face.

Je vous invite à trouver dans ce dossier, la présentation du projet de fin de formation "Microscope-web". Il s'agit de l'adaptation en application web d'un jeu d'écriture collaboratif : un projet avec des problèmes originaux, une réalisation en flux tendu sur un mois et quelques solutions !



III. LISTE DES COMPÉTENCES COUVERTES PAR LE PROJET

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

> Maquetter une application

Pendant le premier sprint de projet, nous nous sommes attelés à la réalisation des *wireframes* de l'application en se basant sur des comptes rendus de partie réalisés lors de parties physiques. Grâce à la dimension écrite du jeu, nous avions beaucoup d'éléments sur lesquels se baser pour réaliser nos *user stories*. Nous nous sommes également aidé d'une partie personnelle afin de mieux saisir l'objet de l'application sur *desktop* comme sur mobile.

> Réaliser une interface web statique et adaptable

L'application n'a pas été conçue en *mobile first*, cependant les *wireframes* conçus pour ce projet prévoient un basculement d'une structure horizontale à une structure verticale pour l'utilisation mobile. La deuxième variation la plus importante était un recours plus systématique à un système de Modales. Pour une utilisation *desktop*, il était simple d'y figurer plus d'informations et une bonne partie du plateau de jeu. Pour maximiser la clarté de lecture pendant l'utilisation mobile, une grande partie des informations et de la structure était accessible au "clic" d'éléments plus synthétiques. Par manque de temps, l'équipe de développement a préféré se concentrer sur l'application *desktop*.

> Développer une interface utilisateur web dynamique

Le projet a été réalisé avec la librairie JavaScript React. Le cœur de l'application étant, comme vous le verrez, un plateau de jeu, chaque élément (que cela soient les paramètres de la partie ou les cartes posées pendant son déroulement) est purement dynamique. Ils sont créés par l'utilisation de modales ou de pages de création situées en amont du commencement de la partie.

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

> Créer une base de données

Pendant le sprint de conception, nous avons fait le choix de créer une base de données relationnelle PostgreSQL. Lors de la création de notre MCD (Modèle Conceptuel de Données), des connexions existaient entre chaque tableau de données. Compte tenu de cela, il était normal de passer par du SQL. Une autre raison plus arbitraire a orienté ce choix : nous avons passé la majorité de notre formation à apprendre les bases de données sur PostgreSQL. Ce projet était donc un parfait prétexte pour exploiter et pratiquer ces connaissances. Nous avons également pris la décision d'utiliser une méthode de *versioning* de notre base de données en utilisant Sqitch.

> Développer les composants d'accès aux données

Nous avons suivi un *design pattern* MVC (*Model View Controller*) qui mettra à disposition de l'application *front-end* une API JSON. Nous n'avons pas utilisé d'ORM car nous préférions construire nous-mêmes les requêtes à la base de données. Notre travail s'est surtout penché sur le développement des *data mappers* et des contrôleurs. Les fonctionnalités de ces derniers ont été ensuite distribuées à travers un routeur API afin de mettre notre base de données à disposition.

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

Notre application Express.js utilise les “JSON Web Token” pour gérer la partie : authentification et autorisation d'accès.

A l'heure où j'écris, nous avons le rôle de visiteur (pas de JWT) qui ne peut pas rejoindre de partie ni en créer. Le rôle de “User” (avec un JWT valide) peut accéder à la création de parties et peut participer à une partie. Le rôle “Admin” peut avoir accès à toute l'API (Application Programming Interface). Pour le moment, la modération administrateur se fait par la “pgAdmin”.

Nous utilisons des variables d'environnement pour gérer les points d'accès à la base de données. Nous en avons un pour gérer le local et un autre sur l'application Heroku qui héberge le serveur.

Les CORS (Cross-Origin Resource Sharing) sont en place mais nous ne les utilisons pas actuellement.



IV. PRÉSENTATION DU PROJET

Ces dernières années, nous avons vu la création de la version web de beaucoup de jeux collaboratifs comme [Gartic Phone](#), [Skribbl.io](#), [Wolfy](#) ou encore [Codenames](#).

Je vous propose de réaliser l'adaptation d'un de ces jeux trouvant son origine dans les jeux de plateau ! Ce jeu, c'est [Microscope](#), ayant remporté le "Most Innovative New Product Gaming Awards" en 2011. Il s'agit d'un jeu collaboratif d'écriture d'histoires où l'on se retrouve à la fois spectateur et auteur au cours d'une même partie. Grâce à un système de cartes simples, les joueurs façonnent avec aisance un monde rythmé par des surprises et des coups de théâtre posés sur la table.

Voici la traduction du texte anglais de la présentation web du jeu :

Qu'est ce que Microscope ? L'humanité se répand à travers les étoiles et fonde une civilisation galactique...

Des nations nouvelles s'érigent dans les ruines de l'Empire...

Une ancienne lignée de roi-dragons disparaît à mesure que la magie faiblit dans le royaume... Ces pitch sont tous des exemples de parties de Microscope. Vous voulez explorer une histoire épique de votre propre cru, s'étalant sur des centaines, voire des milliers d'années, et ça, en une après-midi? C'est Microscope.

Vous ne jouerez pas une partie dans l'ordre chronologique. Vous défieriez les limites de l'espace temps, en bondissant en avant ou en arrière pour explorer les parties de l'histoire qui vous intéressent le plus. Vous voulez sauter d'un milliers d'années dans le futur et voir comment une institution a modelé la société ? Vous voulez revenir sur l'enfance d'un roi qui vient d'être assassiné et comprendre ce qui a fait de lui un suzerain si détesté ? Facile avec Microscope.

Vous avez le vaste pouvoir de créer... et de détruire.

Construisez des civilisations belles et florissantes et consommez les dans un feu nucléaire.

Dézoomez pour voir la vague majestueuse de l'histoire passée à travers les empires, et zoomez pour explorer les vies des personnes l'ayant éprouvé.

Moquez l'ordre chronologique.

Défiez le temps et l'espace. Construisez des mondes et détruisez-les. Un jeu de rôle pour deux à quatre joueurs. Pas de maître du jeu. Pas de préparation. Microscope a été testé pendant deux ans par plus de 150 joueurs talentueux.



V. BESOINS & OBJECTIFS

L'objectif de ce projet est de rendre le jeu "Microscope" plus accessible afin d'augmenter le potentiel de création de parties.

Grâce à cette implémentation web, le jeu devient plus flexible.

On pourra jouer en *remote* afin de se libérer de la contrainte de devoir accommoder les emplois du temps de chacun afin de planifier une partie en présentiel. Les joueurs pourront étaler leurs parties dans le temps au gré de leurs envies et de leurs obligations. Nous pourrons leur proposer un archivage simple et une composition de table de jeu claire. Les parties devront être réservées aux joueurs initialement inscrits à la partie de manière à préserver la cohésion du jeu.

Afin de respecter les délais de réalisation auxquels nous sommes soumis, nous avons pris la décision de nous baser uniquement sur le MVP (Minimal Viable Product).



VI. FONCTIONNALITÉS

A. MVP (Minimal Viable Product)

- Création/Modification de profils
- Consultation des mentions légales
- Modération de contenus (parties et profils utilisateurs) d'une équipe Administrateur via PgAdmin.
- Génération de Lobby permettant au joueur de créer une partie selon les règles (Vue d'ensemble, Début et Fin, Palette). Le créateur de la partie sera le seul à pouvoir opérer pour le MVP.
- Accès à la partie grâce au lien généré permettant d'inviter les autres joueurs inscrits sur la partie.
- Possibilité de rejoindre une partie en cours
- Possibilité de reprendre une partie en pause
- Mise en place de parties simples avec la création de cartes de jeu de type "Période", "Evénement" et "Scène" (de manière chronologique pour le MVP).
- Fonctionnalité de fin de partie
- Consultation possible des parties archivées
- Visuel Identitaire du jeu et logo

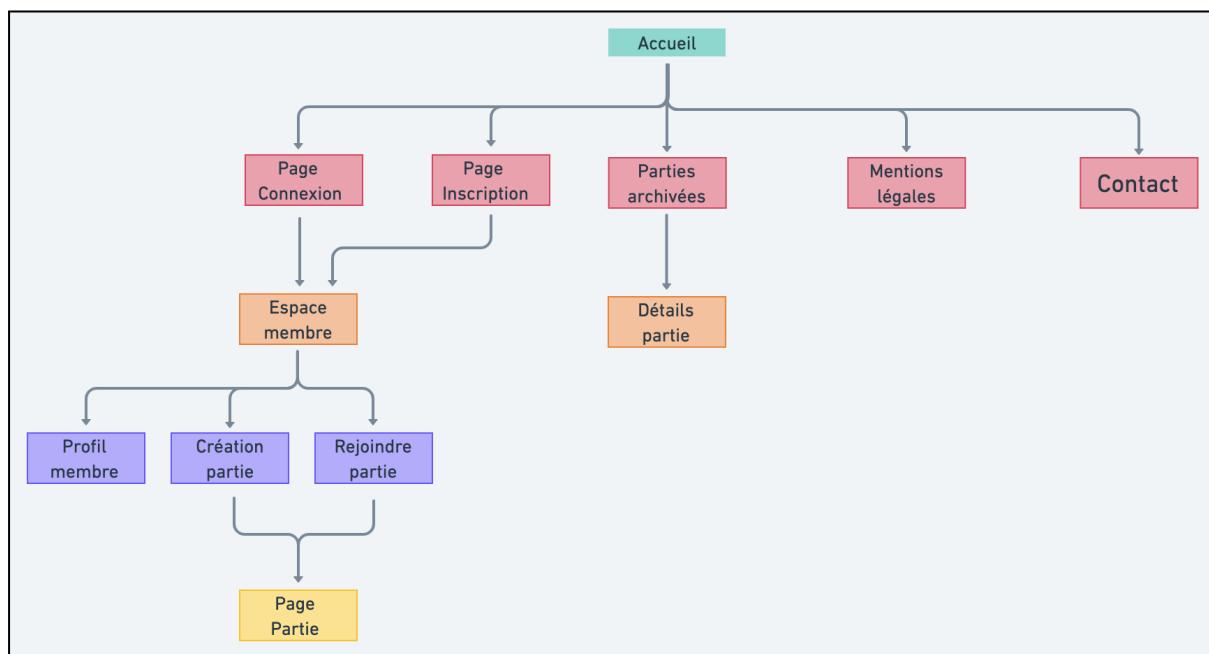
B. Évolutions envisagées

Nous avons dû faire des concessions sur le caractère morcelé de l'application. Nous avons été contraints de réserver à une future évolution du jeu, la possibilité d'offrir aux joueurs la capacité de placer des éléments narratifs à tout moment de l'histoire.

- Ajouter des cartes à n'importe quel endroit de la chronologie
- Implémenter des fonctionnalités de gestion de partie :
 - o Le créateur de la partie peut créer un vote pour *kicker* un joueur.
 - o Les joueurs peuvent acquérir des actions de modération
 - o Les joueurs peuvent, s'ils le souhaitent, reprendre une partie archivée auparavant
- Réaliser des visuels de page d'accueil alternatifs et des évolutions de thèmes s'adaptant à l'ambiance souhaitée par l'utilisateur ou à l'ambiance suggérée par l'histoire écrite

- Réaliser des tests Unitaires avec Jest
- Générer la documentation de la base de données/API
- Ajouter un exemple de partie dans le résumé des règles
- Intégrer des indications de règles lors de la création de la partie et lors de la création de cartes pendant la partie
- Concevoir un contenu éditorial, comme par exemple, mettre à l'honneur des parties archivées
- Intégrer des fonctionnalités de Chat textuel et vocal
- Donner la possibilité au joueur de jumeler son compte à des comptes externes comme : google, apple, discord, meta
- Donner la possibilité aux joueurs d'écrire un résumé de l'histoire en fin de partie

C. Arborescence de l'application



D. Routes Back-end

	DESCRIPTION	ROUTE	MÉTHODE
PAGE D'ACCUEIL	Home	/	GET
ACCÈS VISITEUR	Liste des Parties archivées	/api/archived	GET
	Partie archivées	/api/archived/{id}	GET
	Page de connexion	/api/login	POST
	Page d'inscription	/api/register	POST
	Page des mentions légales	/api/cgu	GET

ACCÈS MEMBRE	Profil	/api/profile/{id}	GET, PATCH & DELETE
	Création de Partie	api/createNewGame	POST
	Vérifier la connexion	/api/verifysignin	POST
	Commencer une Partie	/api/game/{id}/starting	POST
	Salon de Jeu	/api/game/{id}/ongoing	GET, POST & PATCH

E. User Stories du MVP

EN TANT QUE	JE SOUHAITE	AFIN DE
VISITEUR / MEMBRE	voir les parties archivées	lire un résumé de parties
VISITEUR	créer un compte	pouvoir utiliser l'application
VISITEUR	me connecter	pouvoir utiliser l'application
VISITEUR / MEMBRE	accéder aux règles du jeu	mieux comprendre le déroulement d'une partie
VISITEUR / MEMBRE	accéder à la page 404 en cas de problème	permettre de rerouter l'utilisateur en cas d'erreur
VISITEUR / MEMBRE	accéder au mentions légales	avoir accès et pouvoir lire les mentions légales
VISITEUR / MEMBRE	accéder aux contacts	avoir accès et pouvoir contacter un support
MEMBRE	me déconnecter	sécuriser l'accès à mon compte
	accéder a ma page de profil	ajouter et de modifier mes informations
	accéder a ma page de profil	supprimer mon compte
	réinitialiser mon mot de passe	modifier mon mot de passe (oublié, compromis)
	créer une partie	initier une partie
	rejoindre une partie	commencer une partie
	participer au chat	afin de collaborer à l'écriture d'une carte
	ajouter une carte à la partie	faire avancer la partie



VII. SPÉCIFICATIONS TECHNIQUES

A. Versioning

Pour le développement de l'application, nous avons décidé de travailler sur un seul *repository*. La branche "Main" servira à décrire l'application ou à offrir une documentation. L'application est véritablement divisée en deux branches principales qui sont "front" et "back".

> Branches "Back" et "Front"

Chaque fonctionnalité donne lieu à une branche dédiée. Nous avions commencé un développement de branche de manière chronologique pour finalement réorienter notre processus créatif. Sous les conseils de notre *Git Master*, nous avons choisi de réaliser nos branches selon chaque fonctionnalité et non plus chronologiquement.

Dans le cas où nous ferions face à des fonctionnalités complexes ou volumineuses, nous avons offert la possibilité à chaque branche de fonctionnalité de donner lieu à des branches subsidiaires. Une fois la fonctionnalité complètement développée et testée, celle-ci est rapatriée "merged" avec la branche "back".

Le fonctionnement est le même pour la branche "front".

De plus, notre *Git Master* a mis en place une convention d'écriture de message de "commit". Cette convention a trois types de message. Les messages "feat" parlent des fonctionnalités et leurs avancées dans le repository. Les messages "fix" concernent le *commit* de réparation de *bugs*. Les messages "BREAKING CHANGE" parlent d'apport ayant un impact majeur sur les fonctionnalités ou les services de l'API.

B. Workflow et technologies

> FRONT

⇒ React

React est une bibliothèque JavaScript libre développée par Facebook depuis 2013. Le but principal de cette bibliothèque est de faciliter la création d'application web monopage, via la création de composants dépendant d'un état et générant une page (ou portion) HTML à chaque changement d'état.

⇒ React Router

React Router est une bibliothèque complète pour React pour le routage côté client et une bibliothèque JavaScript pour construire des interfaces utilisateur. Il permet une gestion des composants React et des urls associées.

⇒ Redux

Redux est une bibliothèque *open-source* JavaScript de gestion d'état pour applications web. Elle est plus couramment utilisée avec des bibliothèques comme React ou Angular pour la construction d'interfaces utilisateur. Semblable à (et inspirée de) l'architecture Flux, elle a été créée par Dan Abramov et Andrew Clark. Lorsque React devient plus élaboré, le développeur peut rapidement se retrouver face à des problèmes de conception et de mise à l'échelle. Redux permet de combler cette faille de React. Il assiste ce dernier pour fixer les states et rendre leur utilisation beaucoup plus simple.

⇒ Semantic Ui

Semantic Ui est un *framework* CSS *open-source*, qui permet d'éditer les composants React afin de créer des thèmes graphiques rapidement.

⇒ Tailwind

À l'instar de Semantic Ui, Tailwind est un *framework* CSS, qui offre plus de contrôle de l'apparence des composants React grâce à un système de classes. Il permet de se concentrer sur l'écriture HTML et d'utiliser CSS de manière minimale.

> BACK

⇒ Node.js

Node.js est une plateforme *open-source* utilisant JavaScript, donnant ainsi la possibilité de l'utiliser en dehors du navigateur. Il a d'ailleurs été conçu seulement dans cette dernière optique. Il permet de créer des scripts pour les fonctionnalités côté serveur permettant de créer des pages web dynamiques. L'autre qualité de Node.js est l'unification de tous les pans de l'architecture d'une application web, sous un seul langage informatique

⇒ Express.js

Express.js est un *framework* Node.js qui regroupe bon nombre de fonctionnalités contribuant à un développement aisément d'applications web et web mobiles. Il permet l'usage de *middleware* pour répondre aux requêtes HTTP et facilite le *routing* selon les méthodes HTTP.

⇒ PostgreSQL

PostgreSQL aussi appelé Postgres est un système de gestion de base de données gratuit. Il est comparable aux autres systèmes de gestion de base de données relationnelles.

⇒ Bcrypt

bcrypt est une méthode de chiffrage de mot de passe que nous avons utilisé.

⇒ JWT

JWT pour JSON Web Token est un standard internet pour créer signature numérique potentiellement chiffrée qui permet à l'application de réaliser de l'authentification. Cela permet de compartimenter son application selon l'usage prévu et d'en sécuriser les accès.

⇒ PgAdmin 4

PgAdmin est une application libre permettant d'administrer les bases de données PostgreSQL et consort. Il apporte un confort à l'utilisateur en proposant une interface graphique et ainsi faciliter l'administration.



VIII. CONCEPTION DU PROJET

A. WireFrames

La conception a été réalisée avec une partie test réalisée pendant la phase de conception. Cela nous a permis de bien identifier les éléments importants à afficher. Malgré cela, nous avons vite compris qu'il fallait faire des concessions et hiérarchiser les éléments selon leur importance à un instant T de la création ou du déroulement de la partie.

En comparaison à une interface web, un plateau de jeu physique s'établit sur une surface bien supérieure ou de manière plus modulable et flexible. Il est plus facile d'adapter un plateau de jeu sur une application *desktop* que sur un mobile car l'application *desktop* offre une plus large surface de lecture.

Nous avons donc eu recours à une interface de Modales afin de rendre la lecture plus fluide et de renforcer la hiérarchisation des données sur les *wireframes* mobiles. L'utilisation très importante de Modales nous a permis d'offrir aux joueurs une expérience de jeu plus ergonomique.

La conception est plus légère pour l'affichage des pages de connexion de profil ou de visualisation de paramètres de profil.

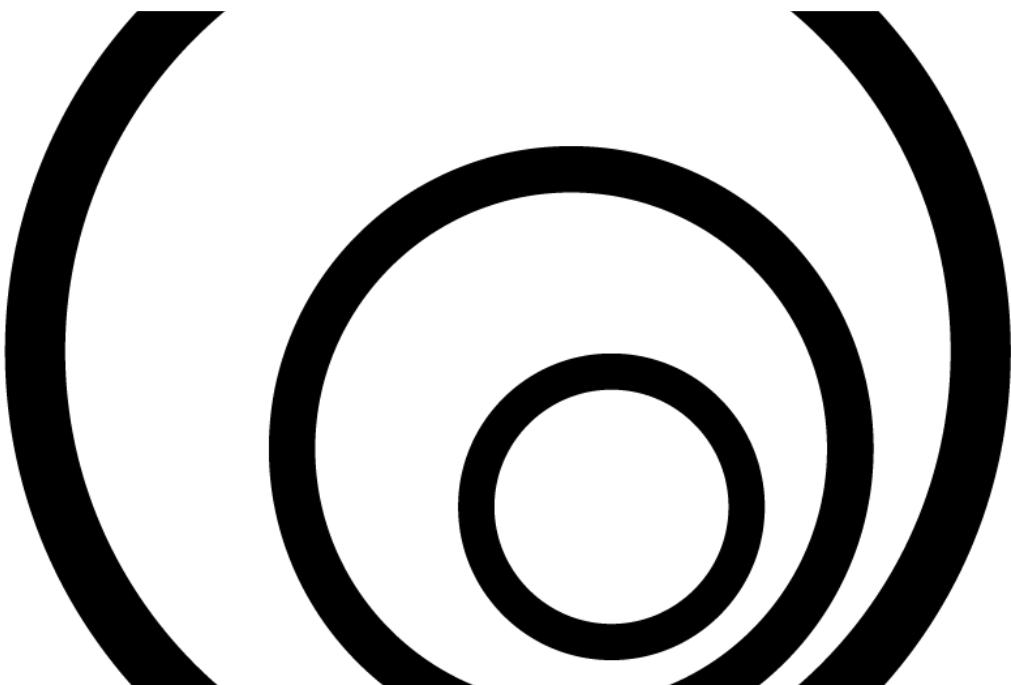
La visualisation des parties archivées n'est pas plus complexe que l'affichage du plateau de jeu. En effet, il s'agit d'une page similaire où les fonctionnalités de création de contenus sont désactivées.

B. Charte graphique

Pendant la phase de conception, j'ai développé une palette de couleurs large. Les couleurs imaginées s'inspirent de la voûte étoilée. Microscope est un jeu narratif pouvant donner lieu à des histoires très différentes les unes des autres. Il était donc judicieux de proposer une gamme de couleurs pouvant épouser n'importe quel thème narratif.

Pour la typographie, nous avons utilisé celle de l'édition : Myriad. J'ai aussi créé le logotype de notre site en reproduisant celui de l'auteur du livre. J'ai produit une version blanche et une version noire pour les besoins du développement *front-end*.

> Logotype de notre site





IX. DÉROULEMENT DU PROJET

A. Présentation de l'équipe

L'équipe est composée de quatre développeurs Full Stack JavaScript ayant tous fait la formation dans la même promotion que moi chez O'clock. Deux d'entre-nous ont suivi la spécialité React et ont travaillé sur la partie *front-end* de l'application. Les deux autres ont suivi la spécialité Data & API et ont développé l'application du côté *back-end*.

> Adlen, Lead Dev Front-end

Son rôle était de faire des choix de conception sur l'application *front-end* et d'échelonner le planning de développement.

> Jérémy ANTONI, Git Master et Développeur Front-end

Son rôle était d'assister les membres du groupe sur les opérations Git, d'assurer une cohérence au niveau des *commits* à l'aide d'une convention de nommage, de résoudre certains conflits lors des *merge* de branches ou encore de valider des *pull-requests*. Il s'est également chargé, avec son binôme, de toute la partie *front-end* du projet, dans le but d'atteindre notre MVP et de répondre aux *user-stories* définies au préalable.

> Victor FERREIRA, Développeur Back-end et Project Manager

Son rôle était de garantir une communication fluide et claire entre le Front et le Back. Il veillait à l'avancée générale du projet. Il soutenait le Product Owner dans les objectifs à tenir pour la réalisation du projet, tout en proposant de nouvelles idées. Concernant le code, il travaillait à la réalisation, en *back*, des fonctionnalités prévues par le Product Owner.

> Pablo PRIETO, Product Owner et Lead Dev Back-end

Je suis l'initiateur de ce projet. J'étais le mieux à même de répondre à toutes les questions concernant le jeu original. J'ai été chargé de l'organisation du développement *back-end*. J'avais la responsabilité de prendre des décisions sur les priorités de missions à effectuer afin de respecter les délais de réalisation. J'avais également en charge d'harmoniser l'écriture du code.

B. Méthode de Travail

> Organisation du travail

L'entièreté du projet a été réalisée en méthode “agile scrum”. Nous avons développé l'application en quatre *sprints* durant approximativement une semaine.

Chaque journée de travail commençait par une réunion à 9h où l'on discutait des objectifs de développement. Nous organisions occasionnellement une petite réunion pendant la journée. Celle-ci avait pour but de répondre verbalement à certaines difficultés de développement et à veiller à l'harmonisation du développement *back*.

En tant que néophytes, nous n'avons pas su communiquer de manière pertinente à certains moments du développement du projet. Nous aurions dû être plus vigilants et ralentir pour offrir plus de cohérence à notre travail.

> Phases de réalisation

Selon les indications de nos référents pédagogiques, nous avons divisé l'avancée de notre projet en trois étapes.

⇒ Étape de Conception / Sprint #0 :

C'est la phase initiatrice du projet.

Les premiers jours de travail ont été dédiés à la mise en place du cahier des charges pour bien cibler les objectifs et besoins de l'application. Ce temps était également dédié à la création d'outils de conception. Nous avons donc produit ensemble les *wireframes*, l'arborescence du site, les *user stories*, le Modèle Conceptuel de Données et d'autres pièces importantes comme le tableau de données ou encore, la définition des outils de gestion.

Cette phase nous a également permis d'attribuer les rôles et les missions confiées à chacun pour le bon déroulement du projet.

Cette étape s'est conclue par l'élaboration du *Minimal Viable Product* et d'une liste des évolutions possibles.

Je tiens à préciser que celui-ci a également pu évoluer pendant l'avancée du développement.

⇒ Étape de Développement / Sprint #1 et #2 :

Nous avons commencé le développement de l'application en s'attelant à rendre les étapes de connexions et de création de profils fonctionnelles pour nos deux applications. L'équipe de la partie *back-end* s'est chargée de l'implémentation des mécaniques de jeu. L'équipe de

la partie front-end s'est occupée de développer les éléments visuels comme le plateau de jeux.

C'est pendant cette phase que le volume de travail a le plus augmenté. L'équipe *front-end* a alors pris beaucoup de retard. Nous n'avons malheureusement pas réussi à compléter toutes les fonctionnalités définies par le MVP du côté front-end. Avec du recul, je pense qu'il aurait été préférable que l'équipe soit formée de cinq membres.

Nous avons déployé l'application *back-end* grâce à un serveur Heroku pendant le sprint #2.

⇒ **Étape de Finalisation / Sprint #3 :**

Cette phase était dédiée à la correction des bugs ainsi qu'au bouclage des fonctionnalités mises en place lors des *sprints* précédents. Pendant cette étape, nous avons permis à l'équipe *front-end* de combler leur retard afin de pouvoir présenter une application fonctionnelle à la fin du projet.

C. Outils de travail

⇒ **Git/Github**

Git/Github est l'outil de *versionning* que nous avons utilisé depuis le début de la formation. C'est un outil primordial et est, à mes yeux, un réel plaisir à utiliser. On s'est vraiment approprié son fonctionnement pendant ce projet. Github est l'extension parfaite pour partager le développement en branche *git* à tous les collaborateurs de l'équipe. Nous avions également la possibilité de déposer des *issues* si jamais nous étions coincés et avions besoin d'aide.

⇒ **Trello**

Nous avons divisé les tâches et les fonctionnalités à implémenter en quatre colonnes. “*Product Backlog*” : où l'on trouvait à l'origine tout notre MVP

“*Sprint Backlog*” : dans lequel figurait les fonctionnalités choisies pour être développer pendant le *sprint* en cours

“*Ongoing*” : dans lequel se trouve les fonctionnalités en cours de développement

“*Finished*” : dans lequel se trouve les fonctionnalités complètement développées

Nous n'avons pas eu une utilisation très régulière de cet outil pendant le début du projet. Ce manque de rigueur a pu poser problème dans notre organisation.

⇒ **Discord**

Toute notre communication, entre membres du projet, écrite et vocale a été réalisée grâce à Discord. Nous avons utilisé Dicord pour organiser nos meetings quotidiens comme nos présentations publiques.

⇒ **Google Drive et sa suite de logiciels**

Nous avons utilisé un dossier privé où l'on y enregistrait toutes les ressources de conception. On y stockait également les journaux de bord de l'équipe. La possibilité d'utiliser toutes les applications de Google en simultanée nous a permis d'être très flexible et a favorisé nos réflexions collectives.

⇒ **Slack**

Nous l'avons utilisé depuis le début de la formation. Slack nous a permis de communiquer exclusivement à l'écrit avec l'équipe pédagogique. Elle nous a également permis de consulter les informations de nos camarades de promotion.

⇒ **VSCode**

Nous utilisons cet IDE depuis le début de la formation et nous avons acquis un certain confort. J'ai beaucoup utilisé la fonction "Live Share" parce que nous faisions du *pair programming*.

⇒ **Heroku**

Nous avons utilisé la version gratuite de Heroku pour héberger l'application *back*. Au moment de la rédaction de ce dossier, nous n'y avons pas hébergé l'application *front*.

⇒ **Insomnia**

C'est le logiciel que l'on a utilisé pour déboguer notre API.



X. RÉALISATIONS PERSONNELLES

> Sprint #0

Étant à l'initiateur de ce projet, l'équipe pédagogique m'a proposé d'endosser le rôle de *Product Owner*. Après les premiers jours de la phase de conception, j'ai vite compris qu'une dissonance existait entre l'idée que je me faisais du projet et la réalité de cette adaptation web. En élaborant le cahier des charges, je me suis rendu compte que les mécaniques de jeu étaient beaucoup plus complexes qu'il n'y paraissait. J'avais le sentiment que le développement prendrait plus d'un mois de travail. Cette contrainte de temps aurait sans doute un effet négatif sur mon équipe et elle pourrait aussi amenuiser le plaisir de travailler. Malgré le fait que nous faisions face à une tâche ardue, il était, de mon point de vue, important de respecter le processus initié et de voir comment allait se dérouler ce travail intensif d'un mois.

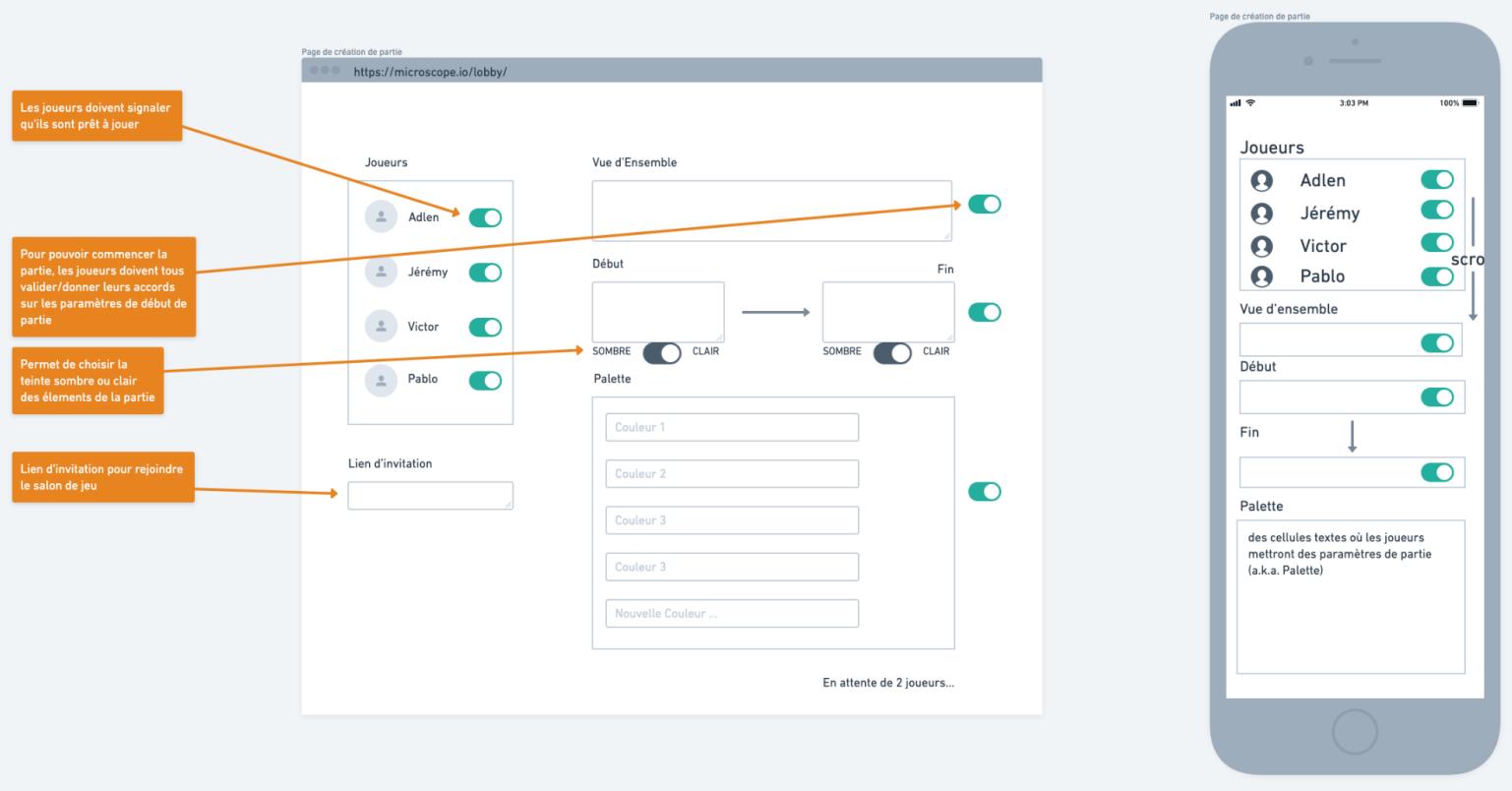
Mon travail s'est d'abord tourné vers la conception de tous les *wireframes* de l'application. Ces derniers reposaient en partie sur ma connaissance des règles du jeu (un résumé est présent en annexe). Avec les développeurs *front-end*, nous avons réalisé une première ébauche pendant laquelle nous avons conçu les morceaux d'interface de gestion de profil et les premiers éléments de l'adaptation des mécaniques de jeu.

Après une mise en situation sous la forme de simulation de partie avec toute l'équipe, nous avons entrepris de faire un deuxième jet plus précis et plus complet.

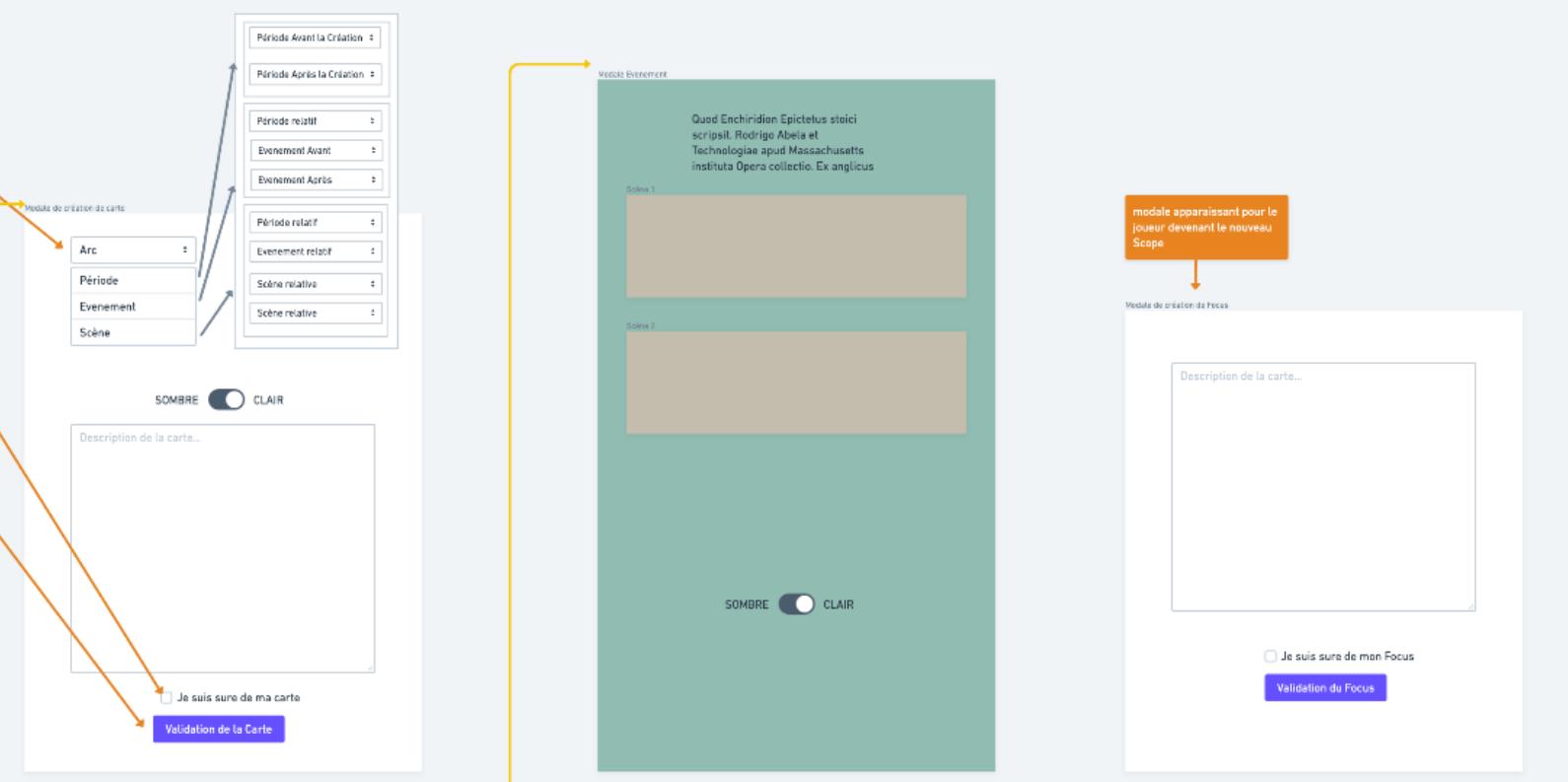
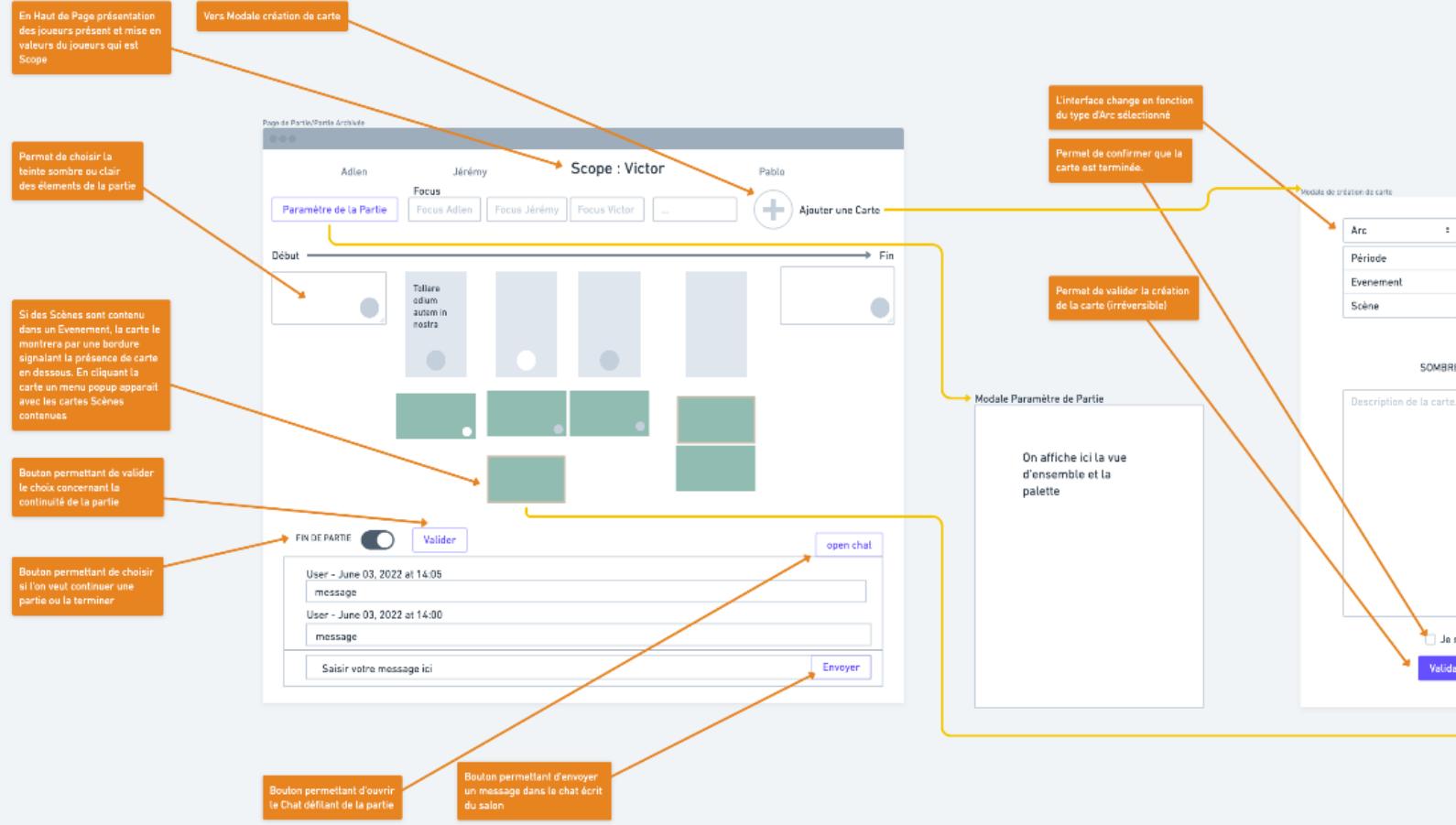
J'ai réalisé les *wireframes* de création de partie et de page de plateau de partie en version *desktop*.

La page de création de partie sert de *lobby* où les joueurs peuvent se rejoindre de manière à poser les bases de leur partie. Chaque élément devant être décidé de manière collégiale, j'ai choisi de mettre en place un système de bouton à bascule que chaque joueur serait obligé de valider afin de démarrer la partie.

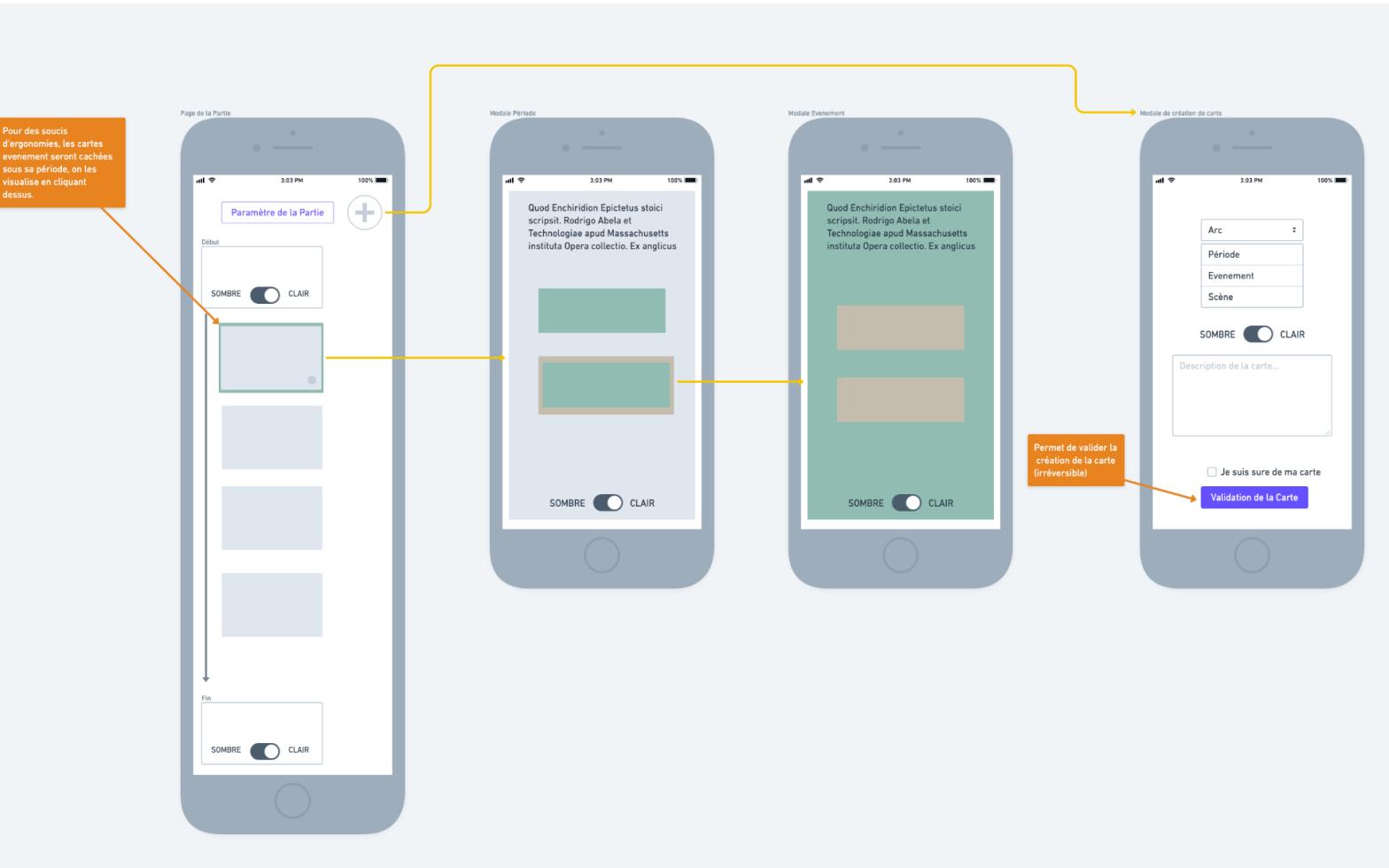
Dans le jeu, certains éléments de jeu ont besoin de renseigner leur caractère négatif ou positif. Par souci de cohérence, j'ai utilisé le même type de bouton pour montrer cette mécanique.



La page de plateau de jeu était difficile à réaliser. J'estimais qu'il fallait réaliser une page sans scroll afin que le joueur puisse avoir une vue d'ensemble de la partie. Aussi, j'ai utilisé un système de boutons "cliquables" qui faisaient apparaître des Modales. Ces dernières étaient utilisées pour fabriquer de nouvelles cartes et pour révéler des paramètres de partie comme la palette et les cartes "scènes" qui prendraient trop de place. Le bas de l'écran était censé montrer le *chat* textuel de la partie.



Toujours dans un souci de lisibilité, j'ai choisi de rendre l'étalement des informations et des fonctionnalités verticales. J'ai également eu un recours plus poussé aux Modales. En plus des cartes "scène" qui étaient déjà cachées dans la version desktop, les cartes "événement" et "période" deviennent accessibles par l'intermédiaire de Modales.



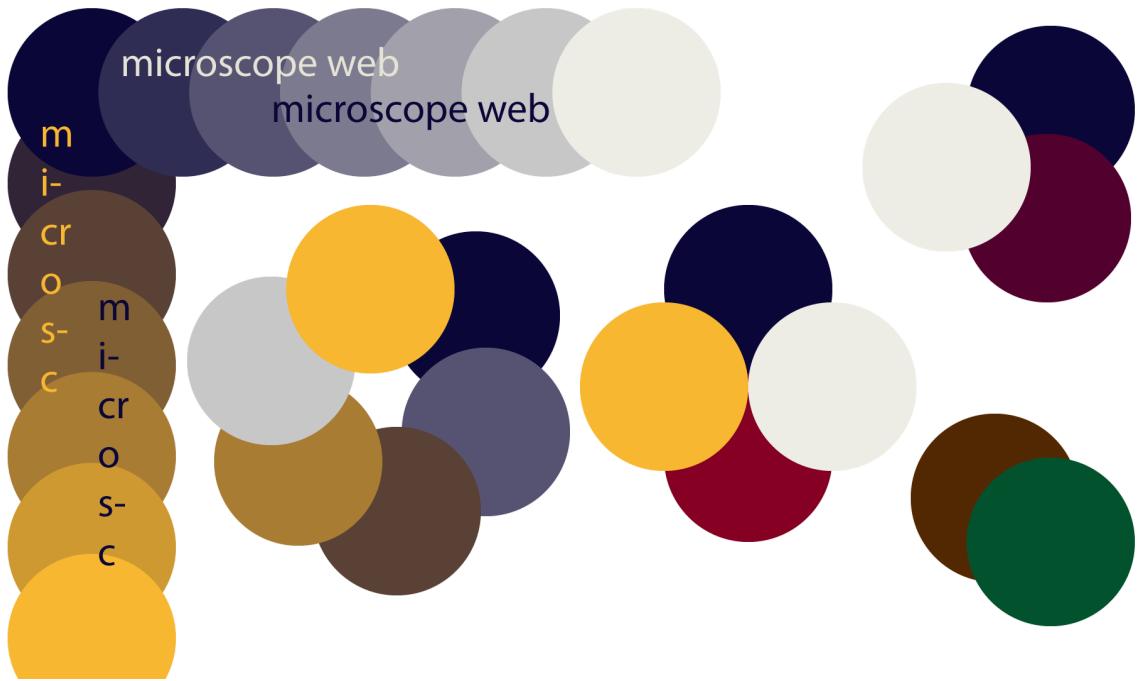
En parallèle de cela, j'ai réalisé avec le Project Manager, les *user stories*. Nous avions prévu d'y faire figurer le rôle d'administrateur. Nous avons finalement décidé de remettre à plus tard cette action afin de développer le rôle d'administrateur une fois qu'une première version du projet soit produite.

Durant cette phase, nous avons également tâché de réaliser la première version du MCD avec Mocodo. À partir de ce dernier, nous avons produit un tableau de données afin de préparer notre première migration Sqitch en déclarant les types de chacune des colonnes et en définissant le placement des clés étrangères.

Même s'il était déconseillé de commencer à coder à ce moment, je regrette de ne pas avoir pu tester immédiatement notre conception dans l'environnement PostgreSQL. En effet, pendant le développement, nous avons dû plusieurs fois revoir l'architecture de la

base de données . Je pense qu'il aurait été donc judicieux de tester notre base de données à ce moment afin d'effectuer certaines modifications.

Cette phase m'a aussi permis de créer des visuels et une palette de couleurs pour le *front*. Comme je le disais, dans la partie dédiée à la charte graphique, cette gamme de couleurs s'inspire de la voûte étoilée.



Enfin, j'ai commencé à établir un cahier des charges du projet. Tout au long de ce *sprint*, nous prenions note de toutes les mécaniques que nécessitaient cette adaptation. À la fin de celui-ci, nous étions face à une liste de fonctionnalités trop importantes. Nous avons donc décidé de nous concentrer sur un cœur de fonctionnalités et de mettre de côté l'utilisation des *web sockets*. Nous savions que cela pourrait nous offrir des possibilités très intéressantes telle qu'une communication en temps réel entre les joueurs se trouvant sur la même page de partie. Ce projet était supposé être dédié à l'utilisation de Socket.io, mais nous ne connaissions pas du tout cette technologie. Sur les recommandations de l'équipe pédagogique nous l'avons mis de côté pour nous concentrer sur une version persistante en base de données. L'autre concession importante que nous avons dû faire à ce moment-là fut l'abandon de l'implémentation des règles dans l'application. Cette implémentation semblait trop gourmande en temps pour être réalisée en un mois. Il aurait fallu implémenter une temporalité dans le fonctionnement qui, certes, rendrait cette adaptation vraiment idéale mais pourrait compromettre la réalisation de notre prototype.

B. Sprint #1

La majorité de mes tâches ont été réalisées en *pair programming*.

Après la présentation de notre sprint de conception et de notre MVP à la promotion Blob O'clock, nous avons développé en priorité la base de données SQL de notre application. Comme je l'ai expliqué au sprint précédent, nous avons utilisé Sqitch pour réaliser le *versionning* des données.

Après la mise en place des fondations avec la base de données, nous avons commencé à développer notre architecture d'application Node.js et Express.js.

La première étape était de mettre en place tout ce qui était en lien avec les profils utilisateurs et s'assurer ainsi que la connexion se fasse correctement avec notre base de données.

C'était vraiment l'objectif de ce *sprint*. Avoir un CRUD (Create Read Update Delete) complet pour notre table "user", nous a vraiment permis de concrétiser des méthodes de base et faire référence à une partie de notre apprentissage sur le fonctionnement de Node.js.

```
const client = require('./client');

const userDataMapper = {

    async findAll() {
        const result = await client.query('SELECT * FROM "user";');

        if (result.rowCount === 0) {
            return null;
        }

        return result.rows;
    },

    async findByPk(userId) {
        const result = await client.query('SELECT * FROM "user" WHERE "id" = $1', [userId]);

        if (result.rowCount === 0) {
            return null;
        }

        return result.rows[0];
    },

    async findUserByUsername(username) {
        const result = await client.query('SELECT * FROM "user" WHERE "username" = $1', [username]);

        if (result.rowCount === 0) {
            return null;
        }

        return result.rows[0];
    },

    async findByGameId(gameId) {
        const result = await client.query('SELECT * FROM "participation" WHERE "game_id" = $1', [gameId]);

        if (result.rowCount === 0) {
            return null;
        }

        return result.rows;
    },

    async findByPlayerId(playerId) {
        const result = await client.query('SELECT * FROM "participation" WHERE "player_id" = $1', [playerId]);
    }
};
```

```

async findByPlayerId(playerId) {
    const result = await client.query('SELECT * FROM "participation" WHERE "player_id" = $1', [playerId]);
    if (result.rowCount === 0) {
        return null;
    }
    return result.rows;
},
async insert(userData, hashedPassword) {
    const preparedQuery = {
        text: `INSERT INTO "user" ("username", "email", "password") VALUES ($1, $2, $3)`,
        values: [userData.username, userData.email, hashedPassword]
    };
    const newUser = await client.query(preparedQuery);
    return newUser.rows[0];
},
async delete(userId) {
    const result = await client.query('DELETE FROM "user" WHERE "id" = $1', [userId]);
    return !!result.rowCount;
},
async update(inputData, userId) {
    const fields = Object.keys(inputData).map((prop, index) => `${prop} = ${index + 1}`);
    const values = Object.values(inputData);
    const date = Date.now();
    const savedUser = await client.query(
        `UPDATE "user"
        SET ${fields}, "updated_at" = ${fields.length + 2}
        WHERE id = ${fields.length + 1}
        RETURNING *`,
        [...values, userId, date],
    );
    return savedUser.rows[0];
},
async isUnique (inputData, userId) {
    const fields = [];
    const values = [];
    // We gather our inputData
    Object.entries(inputData).forEach(([key, value], index) => {
        // We keep the data that are supposed to be unique
        if ([['username', 'email']].includes(key)) {
            // We generate filters
            fields.push(`"${key}" = ${index + 1}`);
            values.push(value);
        }
    });
    const preparedQuery = {
        text: `SELECT * FROM "user" WHERE (${fields.join(' OR ')})`,
        values,
    };
    // If an id is specified, we exclude its registered row
    if (userId) {
        preparedQuery.text += ` AND id <> ${values.length + 1}`;
        preparedQuery.values.push(userId);
    }
    const result = await client.query(preparedQuery);
    if (result.rowCount === 0) {
        return null;
    }
    return result.rows[0];
};
module.exports = userDatamapper;

```

Pour pouvoir profiter de ces premières interactions, il fallait permettre à notre application de détecter la connexion des utilisateurs. Après une veille sur ce sujet, nous avons décidé d'utiliser les JWT (JSON Web Tokens). Vous trouverez le récit de cette veille et de l'utilisation de ces jetons dans la partie “**Vulnérabilités de Sécurité et Veille**”.

À ce moment de notre développement, nous avions créé la méthode de *login* dans le fichier “authcontroller.js” ainsi qu'un premier *middleware* pour contrôler l'accès à la page de profils.

Nous avons fini ce *sprint* en réalisant le CRUD de nos autres tables. J'ai alors pris une décision hâtive, à propos des tables de cartes, en ne réalisant pas une séparation des préoccupations correctes. J'ai pu commencer à rectifier cette erreur dans le *sprint #4* que j'ai réalisé après la présentation Youtube de notre projet le 1er juillet 2022. Vous trouverez une documentation de ce travail dans la partie “**Difficultés rencontrées et solutions apportées -> B. En tant que Lead Dev Back**”.

C. Sprint #2

Le démarrage de ce nouveau *sprint* a été consacré à un premier déploiement de notre application *back*.

Ce déploiement a été principalement réalisé pour que les développeurs *front-end* puissent tester l'avancement de leur application et tester les fonctions de *login* des utilisateurs. J'ai également développé une route et la méthode “*verifyToken()*” dans le “*authController*” pour leur permettre de vérifier ces *login*.

N.B. : Nous avons réellement commencé le déploiement de notre application *back* à la fin du sprint #1. J'ai, cependant, préféré en parler ici car nous avons vraiment commencé à nous en servir lors de ce *sprint*.

```

async verifyToken (request, response) {
  try {
    const token = request.headers.authorization.split(' ')[1];
    console.log("token", token);

    jwt.verify(token, process.env.TOKEN_KEY, function (err, decoded){
      if (err) {
        return response.json({ errorMessage: "Token invalid" });
      }

      return response.json({ userId: decoded.userId, userName: decoded.userName, Message: "Token valid !" });
    });
  }
  catch (err) {
    return response.json({errortype: err.message, errorMessage: "Unable to verify the token"});
  }
}

```

Nous nous sommes servi presque exclusivement de Heroku CLI (Command Line Interface) pour diriger notre application et la dépanner.

J'ai ajouté un *remote* entre mon *repository* local et l'application nouvellement créé sur Heroku avec la commande de commande et *push* sur le *remote* "heroku".

```

heroku git:remote -a microscope-web
git push heroku back:main

```

En plus du déploiement de notre application, j'ai également déployé notre base de données PostgreSQL afin que les développeurs *front-end* aient l'environnement complet pour pouvoir tester leurs fonctionnalités. Pour concrétiser cette transition j'ai changé mon fichier *client* afin qu'il utilise les variables de configuration (*Config Vars*) que nous avons créé pour notre application.

Ainsi, si la variable "NODE_ENV" a pour valeur 'production', le client va se connecter à la base de données avec la variable "DATABASE_URL" située dans le fichier des variables d'environnement de Heroku. Sinon il utilisera la variable d'environnement "PG_URL" présente dans le *git* local de développement.

```

const { Client } = require('pg');
require('dotenv').config();

const config = () => {

  if (process.env.NODE_ENV === 'production') {

    return {
      connectionString: process.env.DATABASE_URL,
      ssl: {
        rejectUnauthorized: false,
      }
    }
  }
  else {
    return { connectionString: process.env.PG_URL };
  }
};

const client = new Client(config());
client.connect();

module.exports = client;

```

À mes yeux, l'autre grande partie du développement ayant eu lieu pendant ce *sprint* fut la création de deux méthodes :

```
async deployGame (request, response) {  
  try {  
    // We need to update data for each starting game relations  
    // Of course the "game" table but also "participation" and "palette"  
    const gameData = request.body.game;  
    const gameId = request.params.id;  
  
    await gameDatamapper.updateGame(gameData, gameId);  
  
    // When a game start insert every user participating with the related "game".id  
    // We need to extract the "user".id who will play for this game  
  
    // As a work around for the absence of socketio we will be presented with the usernames of the users  
    //participating to the game  
    // So the data about user.id will not be in the request.body. It will contain usernames instead  
    //const dataPlayersId = Object.values(request.body.players);  
  
    const dataPlayersUsername = request.body.players;  
    // We need need to increment players position, beginning by creator who will serve as the firs player  
    let position = 1;  
  
    dataPlayersUsername.forEach(async (dataPlayerUsername) => {  
      const player = await playerDatamapper.findByUsername(dataPlayerUsername);  
  
      if (player) {  
        const playerId = player.id;  
  
        playerDatamapper.insert(gameId, playerId, position);  
        position++;  
      }  
    });  
  
    // A similar loop is needed for inserting every palette shade i.e. themes to include  
    //or exclude from the game  
    const paletteArray = request.body.palette;  
  
    paletteArray.forEach(async (paletteCard) => {  
      if (paletteCard) {  
        paletteDatamapper.insert(gameId, paletteCard);  
      }  
    });  
  
    return response.status(201).json({ Message: "game deployed successfully !" });  
  } catch (err) {  
    return response.status(502).json({ errorLog: err.message, errorMessage: "Unable to deploy the game!" });  
  }  
}
```

⇒ Méthode n°1

La méthode utilisée pour insérer les données de début de partie (“deployGame”) “deployGame” demandait une bonne organisation. Le volume de données traitées était très important. J'ai donc tâché de bien découper les différents *insert* dans la base de données.

Le premier morceau de la méthode s'est occupé de réaliser un *update* sur la table “game”. C'était la partie la plus simple de la méthode de contrôleur.

Le deuxième morceau était dédié au joueur intégrant la partie. Par l'intermédiaire d'une boucle “*forEach*”, je viens vérifier l'existence de joueur sujet de l'itération. Ensuite, je l'insert dans la base de données.

Le troisième morceau a aussi utilisé une boucle “forEach” mais il n'y avait pas cette préoccupation d'existence antérieure des données proposées par le client de l'application.

⇒ Méthode n°2

La méthode d'affichage des parties (“getOne”) pour le plateau de jeu prévu par l'application front.

```

async getOne (request, response) {
    // This method will retrieve a game as an all, which every data connected to the a game
    // It will be used to display ongoing games and archived ones
    try {
        // The game object will find all the data contained in the "game" table
        const game = await gameDatamapper.findByPk(request.params.id);

        // The player object will retrieve data in the "participation" table
        const playersFound = await userDatamapper.findByGameId(request.params.id);

        const players= [];

        playersFound.forEach(async (player) => {
            const playerInfo = await userDatamapper.findById(player.player_id);
            players.push({ id: playerInfo.id, username: playerInfo.username, position: player.position });
        });

        const focuses = await focusDatamapper.findByGameId(request.params.id);

        // We also need to retrieve the palette colors
        const palette = await paletteDatamapper.findByGameId(request.params.id);

        // The period object is composed of period of our game and each subsequent event which
        // also reach to related scenes
        const periods = await periodDatamapper.findByGameId(request.params.id);

        if (periods) {
            for (let i = 0; i < periods.length; i++) {
                const eventsFound = await eventDatamapper.findByPeriodId(periods[i].id);

                if (eventsFound) {
                    for (let j = 0; j < eventsFound.length; j++) {
                        const scenesFound = await sceneDatamapper.findById(eventsFound[j].id);

                        if (scenesFound) {
                            eventsFound[j].scenes = scenesFound;
                        }
                    }
                    periods[i].events = eventsFound;
                }
            }
        }
        return response.status(200).json({ game, players, palette, focuses, periods });
    } catch (err) {
        return response.status(404).json({ errType: err.message, errorMessage: "Failed to find game" });
    }
}

```

En plus de demander beaucoup d'organisation, la méthode “getOne” ajoutait une part de complexité. Après m'être occupé du tableau d'objet “players”, des objets “focuses” et “palette”, il fallait bien pensé l'architecture des tables de cartes narratives.

Chaque carte “period” peut posséder des cartes “event” qui elles-mêmes peuvent posséder des cartes “scene”. Pour prendre en compte cette “filiation” et servir à l’application *front-end*, des objets JSON sont créés grâce à des boucles nichées (*nested loop*). Cette structure imbriquée fait figurer les connexions entre les différentes cartes. J’ai préféré utilisé des boucles “for” classiques pour cette partie. Ainsi, l’interaction et les incrémentations de chaque boucle seront plus lisibles.

J’ai mis en place de nouvelles migrations Sqitch de la base de données pendant ce *sprint*. Vous trouverez plus de détails dans “**Difficultés rencontrées et solutions apportées -> B. En tant que Lead Dev Back**”.

D. Sprint #3

Cette partie du projet était dédiée à la correction de bugs et au bouclage des fonctionnalités commencées lors des *sprints* précédents.

Après discussion avec l’équipe de projet, j’ai modifié la structure des fichiers JSON reçus et retournés par notre application *front-end* pour les méthodes mentionnées auparavant : “gameController.deployGame” et “gameController.getOne”.

Nous avons bouclé le CRUD de nos tables de cartes narratives, commencé au *sprint* #1, avec la création des méthodes “updatePosition” (détails dans “**Difficultés rencontrées et solutions apportées -> B. En tant que Lead Dev Back**”).

```

const cardDatamapper = require('../models/cardDatamapper');

const periodDatamapper = require('../models/periodDatamapper');
const eventDatamapper = require('../models/eventDatamapper');
const sceneDatamapper = require('../models/sceneDatamapper');
const focusDatamapper = require('../models/focusDatamapper');

const cardController = {

    async createCard (request, response) {
        try {
            // Focus card have incremental position so no need to change positions
            // Other need to detect positions, because the player can choose to put a card before one already
            // created
            if(request.body.cardType === "focus") {
                const focusFound = await focusDatamapper.findByIdGameId(request.params.id);
                let position;
                if (focusFound) {
                    position = focusFound[focusFound.length - 1].position + 1;
                }
                else {
                    position = 1;
                }
                const focus = await focusDatamapper.insert(request.body, position, request.params.id);

                return response.status(201).json({ Message: "Focus creation succeed !", focus});
            }
            else {
                // SOC distribution
                let cardsToMove = {};

                if(request.body.cardType === "period") {
                    cardsToMove = await periodDatamapper.findAllByPosition(request.body);
                }
                else if (request.body.cardType === "event") {
                    cardsToMove = await eventDatamapper.findAllByPosition(request.body);
                }
                else if (request.body.cardType === "scene") {
                    cardsToMove = await sceneDatamapper.findAllByPosition(request.body);
                }
                console.log("cardsToMove", cardsToMove);
                if (cardsToMove) {
                    // SOC distribution
                    if (request.body.cardType === "period") {
                        for (let i = 0; i < cardsToMove.length; i++) {
                            await periodDatamapper.updatePosition(cardsToMove[i].id,
                            cardsToMove[i].position + 1);
                        }
                    }
                    else if (request.body.cardType === "event") {
                        for (let i = 0; i < cardsToMove.length; i++) {
                            await eventDatamapper.updatePosition(cardsToMove[i].id, cardsToMove[i].position
                            + 1);
                        }
                    }
                    else if (request.body.cardType === "scene") {
                        for (let i = 0; i < cardsToMove.length; i++) {
                            await sceneDatamapper.updatePosition(cardsToMove[i].id, cardsToMove[i].position
                            + 1);
                        }
                    }
                }
                let card = {};
                if (request.body.cardType === "period") {
                    card = await periodDatamapper.insert(request.body);
                }
                if (request.body.cardType === "event") {
                    card = await eventDatamapper.insert(request.body);
                }
                if (request.body.cardType === "scene") {
                    card = await sceneDatamapper.insert(request.body);
                }

                return response.status(201).json({ Message: `${request.body.cardType} creation succeed !`,
                card});
            }
        } catch (err) {
            return response.status(502).json({errorType: err.message, errorMessage: "Card creation failed"});
        }
    },
};

module.exports = cardController;

```

J'ai passé du temps à résoudre les problèmes de CORS et j'ai aidé les développeurs *front-end* à avancer sur leurs CSS. J'aurais pu travailler à l'écriture d'une documentation Swagger mais j'ai préféré me concentrer sur l'application afin de pouvoir faire une démonstration fonctionnelle lors de la présentation du 1er juillet.

J'ai aussi passé en revue notre code afin d'effacer les "console.log()" qui n'étaient plus nécessaires et rendre le code plus propre.

D. Sprint #4

J'ai continué l'application après la fin de ma formation. Au fur et à mesure de cette mise en situation, j'ai dû mettre de côté certains aspects du développement pour me concentrer sur l'avancée du projet.

J'ai commencé par effacer la présence du fichier de configuration qui aurait pu compromettre la sécurité de l'application. Il s'agissait du fichier "sqitch.conf" qui comportait les informations de connexions vers la base de données Heroku. Nous l'avions créé pour apporter les mêmes fonctionnalités de *versionning* que notre version locale. Cela s'est vite avéré délicat. En effet, malgré la suppression de ce fichier des branches concernées, il apparaissait toujours dans l'historique de mes *commit*. J'ai utilisé l'outil BFG Repo-Cleaner (<https://rtyley.github.io/bfg-repo-cleaner/>) pour résoudre ce problème.

J'aurais pu également opérer la commande suivante :

```
git filter-branch --index filter 'git rm --cached --ignore-unmatched sqitch.conf' HEAD
```

Après cela, j'ai commencé à régler les problèmes de SOC de mes *datamappers*. Les détails sont présents dans "**Difficultés rencontrées et solutions apportées -> B. En tant que Lead Dev Back**".

Finalement, j'ai réalisé avec Victor, développeur *back-end*, une veille et rectification des codes HTTP de statut en *pair programming*. Nous avions choisi de les laisser de côté, mais les codes HTTP de statut font partie de la communication avec le client. Il aurait été donc difficile de dépanner l'application sans leur existence



XI. DIFFICULTÉS RENCONTRÉES ET SOLUTIONS APPORTÉES

A. En tant que Product Owner

> Explication des règles du Jeu

Les parties de Microscope sont en réalité assez longues. Après un tour d'horizon du jeu pendant la phase de conception, j'ai décidé de proposer une partie test avec les développeurs de l'équipe. Vous trouverez un rapide résumé des règles, avec des exemples à la clé, dans les annexes.

> Cohésion de l'équipe

En tant que Product Owner, j'ai le sentiment de ne pas avoir rempli correctement ma mission. En effet, j'étais bien trop investi pour garder un point de vue objectif. Je n'ai donc pas réussi à avoir le recul nécessaire pour mener au mieux ce projet périlleux. Nous avons rapidement rencontré des difficultés qui n'ont pas été anticipées car nous manquions certainement d'expériences dans la réalisation de projets de cette envergure. Le noeud de ces dernières était principalement lié à la complexité de l'algorithmie nécessaire pour réaliser les méthodes que je jugeais essentielles pour le fonctionnement de base de cette adaptation. Les problématiques rencontrées ont créé beaucoup de friction et j'aurais dû ré-envisager un découpage des fonctionnalités pour avoir une chance de les implémenter correctement. Cette remise en question aurait pu améliorer la communication et fluidifier le travail entre les collaborateurs. Ce travail n'ayant pas été fait, l'équipe *front-end* a pris beaucoup de retard sur l'implémentation de fonctionnalités de base. En cela, je suis très déçu du résultat. J'aurais bien voulu montrer ce prototype à l'auteur du livre, Ben Robbins, qui nous avait autorisé à utiliser son livre pour ce projet de formation. Je ne pense pas que l'on puisse le faire en l'état. Cela n'est que partie remise car je compte redoubler d'efforts durant la période post formation. Je suis dans l'écriture assidue de ce dossier mais je me replongerai dans ce projet dès que j'aurais passé mon titre professionnel.

B. En tant que Lead Dev Back

> Résolutions de Problèmes

Nous avons rencontré des difficultés de développement similaires sur la partie *back-end* mais nous avons rempli nos objectifs algorithmiques principaux. C'était plus simple pour moi d'imaginer des mécaniques de jeu car j'ai déjà eu l'occasion d'y jouer plusieurs fois. De plus, j'ai une appétence dans la résolution de problèmes donc j'ai apprécié être confronté à des problèmes d'algorithmies pour lesquels je devais être en recherche de solutions.

Le mécanisme qui demandait, à mes yeux, le plus de conception, était lié à la position des cartes relatives aux autres.

En effet, il existe dans le jeu, des cartes représentant différentes longueurs narratives. Par exemple, les cartes "Périodes" représentent l'unité la plus longue. Lors de son tour, le joueur a la possibilité de poser des cartes n'importe où. Il peut poser sa carte "Période" avant celles déjà posées ou bien après.

Afin de concrétiser cette possibilité dans l'application nous avons mis en place une méthode pour :

1° Récupérer les cartes se trouvant après la position désirée.

```
async findAllByPosition (data) {  
  const preparedQuery = {  
    text: `SELECT * FROM "period" WHERE position >= $1 AND $2 = $3`,  
    values: [data.previous_card_position + 1, `${data.parentType}_id`, data.parentId]  
  };  
  
  const result = await client.query(preparedQuery);  
  
  return result.rows;  
}
```

2° Modifier la position des cartes d'un rang

```
async updatePosition (cardId, newPosition) {  
  
  const preparedQuery = {  
    text: `UPDATE "period" SET  
      position = $1  
      WHERE id = $2  
      RETURNING *`,  
    values: [newPosition, cardId],  
  };  
  
  const savedCard = await client.query(preparedQuery);  
  
  return savedCard.rows[0];  
}
```

3° Insérer la nouvelle carte sur la position demandée par le joueur

```
async insert (data) {  
  const preparedQuery = {  
    text: `INSERT INTO "period" ("text", "tone", "position", "game_id") VALUES ($1, $2, $3, $4) RETURNING  
    *`,  
    values: [data.text, data.tone, data.previous_card_position + 1, data.parentId]  
  };  
  
  const result = await client.query(preparedQuery);  
  return result.rows[0];  
}
```

Bien que fonctionnel, ce processus n'a pas été utilisé par l'application *front-end*.

> Sqitch

Le MCD n'a pas vraiment évolué après le sprint de conception. Cependant, les tables ont connu des mutations légères mais primordiales pour que l'on puisse développer une application fonctionnelle. Il avait été décidé de produire de nouvelles parties (table "game") en injectant des données pour toutes colonnes détaillant cette dernière. Afin de respecter cette condition, chaque colonne de la table (sauf "updated_at") possédait la contrainte "NOT NULL". Il a fallu enlever cette contrainte à posteriori sur la quasi-entièreté des colonnes (sauf "creator_id", "current_user_id", "created_at"). Après discussion avec notre équipe, il a été décidé de créer la partie avant que l'utilisateur n'en décide tous les détails. Il était plus simple de rythmer l'utilisation de l'application ainsi. J'ai donc créé une nouvelle migration sur Sqitch afin d'abandonner cette contrainte là où elle n'était plus souhaitée.

Des évolutions similaires ont été réalisées sur d'autres tables de notre base de données. Au départ, nous n'avons pas rencontré de problèmes. C'est en réalisant la fonction "revert" de Sqitch que je me suis rendu compte que la fonction "deploy" n'était pas conçue correctement. Le problème n'a pas été tout de suite rectifié. Il aurait fallu insérer des tables temporaires afin de sauvegarder les valeurs pour les champs retrouvant la contraintes "NOT NULL". Nous aurions pu indiquer la valeur "undefined" pour que la table respecte les nouvelles contraintes et ainsi éviter les conflits. La deuxième partie de la solution aurait été de mettre en place des tables temporaires, des annexes, qui pourraient stocker les données "abandonnées" pendant les différentes migrations Sqitch.

> Séparation des Préoccupations

J'ai fait des choix d'architecture de nos *datamappers* qui ne respectaient pas le SOC (Separation Of Concerns) de l'application. J'étais tellement concentré sur le fait d'obtenir des méthodes factorisées du code que j'en ai oublié le SOC. J'ai donc, par inadvertance, développé un code et un *datamapper* qui générait des requêtes pour plusieurs tables.

```

else {
    const cardsToMove = await cardDatamapper.findAllByPosition(request.body);

    if (cardsToMove) {
        for (let i = 0; i < cardsToMove.length; i++) {
            await cardDatamapper.updatePosition(request.body.cardType, cardsToMove[i].id,
            cardsToMove[i].position + 1);
        }
    }

    const card = await cardDatamapper.insert(request.body);

    // Check status code for error
    return response.json({ Message: `${request.body.cardType} creation succeed !`, card});
}

```

```

async insert (data) {
    console.log("datainsert", data);

    let preparedQuery = {};

    if(data.cardType === "period") {
        preparedQuery = {
            text: `INSERT INTO "period" ("text", "tone", "position", "game_id") VALUES ($1, $2, $3, $4) RETURNING *`,
            values: [data.text, data.tone, data.previous_card_position + 1, data.parentId]
        };
    }
    else if (data.cardType === "event") {
        preparedQuery = {
            text: `INSERT INTO "event" ("text", "tone", "position", "period_id") VALUES ($1, $2, $3, $4) RETURNING *`,
            values: [data.text, data.tone, data.previous_card_position + 1, data.parentId]
        };
    }
    else {
        preparedQuery = {
            text: `INSERT INTO "scene" ("text", "tone", "position", "event_id") VALUES ($1, $2, $3, $4) RETURNING *`,
            values: [data.text, data.tone, data.previous_card_position + 1, data.parentId]
        };
    }
}

```

Pour le Sprint #4, j'ai pris le temps de revoir l'architecture de mes *datamappers* et le code du *cardController*. J'ai obtenu un résultat moins factorisé. Le code réalisé est, ainsi, plus lisible.

```

JS client.js
JS eventDatamapper.js
JS focusDatamapper.js
JS gameDatamapper.js
JS paletteDatamapper.js
JS periodDatamapper.js
JS playerDatamapper.js
JS sceneDatamapper.js
JS userDatamapper.js

```



XII. FONCTIONNALITÉS REPRÉSENTATIVES ET JEU D'ESSAI

Comme expliqué précédemment, les parties du jeu se décomposent en quatre temps :

1. créer une partie avec ses champs encore vides
2. insérer des informations de création de partie par le biais d'une page *lobby* contrôlée par le créateur de la partie
3. rejoindre la partie avec d'autres utilisateurs avec un lien unique
4. poser des cartes thématiques ou narratives avec différents joueurs : cette implémentation reste chronologique du côté de l'application *front-end*

N'ayant pas implémenté de test unitaire pour le moment, j'ai simulé ce workflow avec Insomnia.

J'ai réalisé ce test avec 3 utilisateurs. L'application se comporte exactement comme prévu et les données sont conformes à nos attentes.

Je vous invite à trouver ci-dessous toutes les étapes de ce test avec les codes de statut HTTP prévus

> Je me connecte avec trois utilisateurs et je reçois des JSON Web Token

The screenshot shows the Insomnia API client interface. A POST request is made to `http://localhost:4000/api/login`. The request body is a JSON object:

```
1 {  
2   "username": "Ella",  
3   "password": "THX1138_forever!"  
4 }
```

The response status is 200 OK, with a response time of 189 ms and a response size of 259 B. The response body is a JSON object containing user information and a token:

```
1 {  
2   "username": "Ella",  
3   "email": "ella@gmail.com",  
4   "userId": 2,  
5   "token":  
6     "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOjIsInVzZXJOYWIl  
IjoiRWxsYSIsInVzZXJSb2xlIjoiZWVtYmVjIiwiaWF0ijoxNjU5NjE4ODM2LCJleH  
AiojE2NTk2NDA0Mz9.8F0q1GR2apjBAUc1peMtNnLlmFb4D-LKdr7hZAcwTGM"
```

The screenshot shows the Insomnia API client interface. A POST request is made to `http://localhost:4000/api/login`. The request body is a JSON object:

```
1 {  
2   "username": "Annie",  
3   "password": "THX1138_forever!"  
4 }
```

The response status is 200 OK, with a response time of 92.4 ms and a response size of 263 B. The response body is a JSON object containing user information and a token:

```
1 {  
2   "username": "Annie",  
3   "email": "annie@gmail.com",  
4   "userId": 3,  
5   "token":  
6     "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOjMsInVzZXJOYWIl  
IjoiQW5uawWjLCj1c2VyUm9sZS16Im1lbWlciisImIhdCI6MTY1OTYxODg1NCwiZ  
hwIjoxNjU5NjQwNDU0fQ.X_3Lfq-NvP0JSDVOLB6RuuqvNK8IbjEK6e2UJRg7qA"
```

```

POST http://localhost:4000/api/login
{
  "username": "Lucas",
  "password": "THX1138_forever!"
}

```

200 OK
83.9 ms
263 B
5 Minutes Ago

> Après avoir accéder au *lobby* de création de partie, le premier utilisateur doit remplir les informations demandées. Il est important de ne pas omettre les autres participants. Un *middleware* s'assure que tous les utilisateurs soient bien admis dans la partie.

```

POST http://localhost:4000/api/createNewGame
{
  "creator_id": "2"
}

```

201 Created
5.33 ms
51 B
Just Now

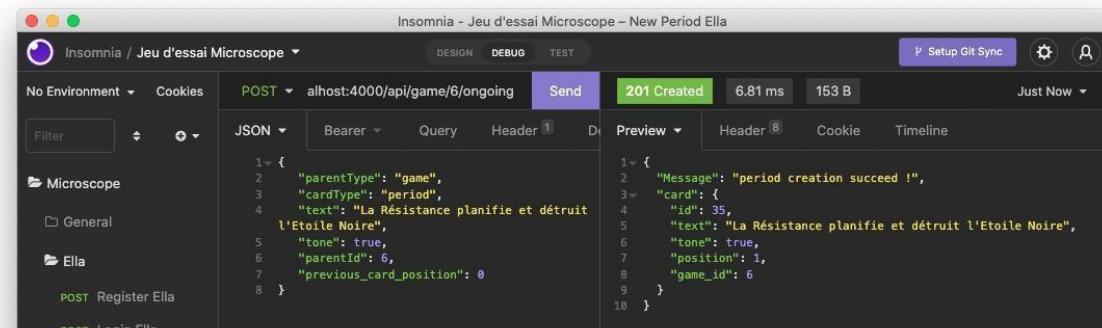
```

POST http://localhost:4000/api/game/6/start
{
  "game": {
    "big_picture": "Un Empire tyrannique chute grâce à un mouvement rébel",
    "state": "ongoing",
    "bookend_start": "L'Empire met au point une station pouvant détruire des planètes entières",
    "bookend_start_tone": "0",
    "bookend_end": "L'Empereur est vaincu et ses soldats fuient à travers la galaxie",
    "bookend_end_tone": "1",
    "creator_id": "2",
    "current_user_id": "2"
  },
  "players": [
    "Ella", "Annie", "Lucas"
  ],
  "palette": [
    {
      "text": "Magie",
      "status": "1"
    },
    {
      "text": "Ordre de chevalier",
      "status": "1"
    },
    {
      "text": "Espace Réaliste",
      "status": "0"
    },
    {
      "text": "Trop de Gentils Qui Meurent",
      "status": "0"
    }
  ]
}

```

201 Created
4.55 ms
42 B
Just Now

> À présent, chaque utilisateur peut accéder au plateau de partie et créer des cartes. Le premier utilisateur pose une carte “Période”.

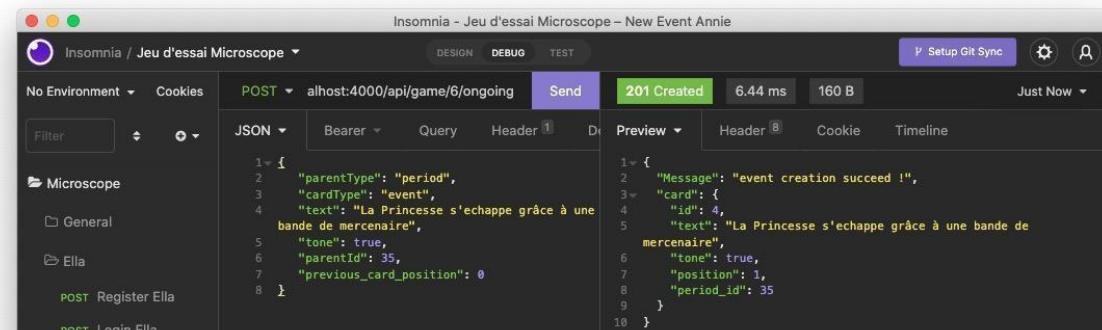


```

POST /api/game/6/ongoing
{
  "parentType": "game",
  "cardType": "period",
  "text": "La Résistance planifie et détruit l'Etoile Noire",
  "tone": true,
  "parentId": 6,
  "previous_card_position": 0
}
  
```

The response is a 201 Created status with a message about period creation succeed and the created card's details.

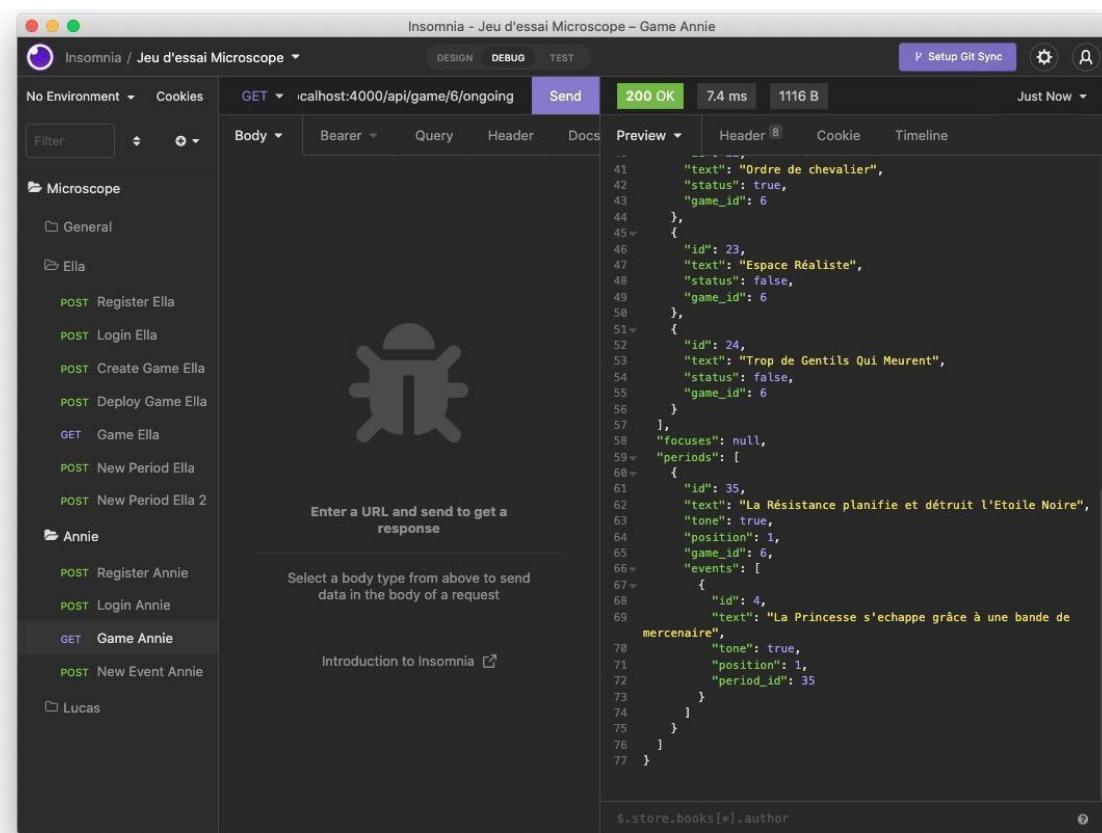
> Le second utilisateur, qui peut voir les cartes déjà posées, propose une carte “Événement”.



```

POST /api/game/6/ongoing
{
  "parentType": "period",
  "cardType": "event",
  "text": "La Princesse s'échappe grâce à une bande de mercenaire",
  "tone": true,
  "parentId": 35,
  "previous_card_position": 0
}
  
```

The response is a 201 Created status with a message about event creation succeed and the created event's details.



```

GET /api/game/6/ongoing
  
```

The response is a 200 OK status with game status data including cards and events.

> Le troisième utilisateur, ayant actualisé sa page de plateau, produit une carte de "Période" après celle posée par le premier utilisateur.

```

1= {
2  "parentType": "game",
3  "cardType": "period",
4  "text": "Les mercenaires et la princesse sont piégés sur une planète gazeuse par le sbire de l'Empereur",
5  "tone": false,
6  "parentid": 6,
7  "previous_card_position": 1
8 }

```

```

1= {
2  "Message": "period creation succeed !",
3  "card": {
4    "id": 36,
5    "text": "Les mercenaires et la princesse sont piégés sur une planète gazeuse par le sbire de l'Empereur",
6    "tone": false,
7    "position": 2,
8    "game_id": 6
9  }
10 }

```

> Le premier utilisateur propose une autre carte "Période" qu'il posera avant celle déjà posée.

```

1= {
2  "parentType": "game",
3  "cardType": "period",
4  "text": "Un jeune fermier rejoint la résistance",
5  "tone": true,
6  "parentid": 6,
7  "previous_card_position": 0
8 }

```

```

1= {
2  "Message": "period creation succeed !",
3  "card": {
4    "id": 37,
5    "text": "Un jeune fermier rejoint la résistance",
6    "tone": true,
7    "position": 1,
8    "game_id": 6
9  }
10 }

```

```

1= {
2  "players": [ .. 3 .. ],
3  "palette": [ .. 4 .. ],
58  "focuses": null,
59  "periods": [
60    {
61      "id": 37,
62      "text": "Un jeune fermier rejoint la résistance",
63      "tone": true,
64      "position": 1,
65      "game_id": 6
66    },
67    {
68      "id": 35,
69      "text": "La Résistance planifie et détruit l'Etoile Noire",
70      "tone": true,
71      "position": 2,
72      "game_id": 6,
73      "events": [
74        {
75          "id": 4,
76          "text": "La Princesse s'échappe grâce à une bande de mercenaire",
77          "tone": true,
78          "position": 1,
79          "period_id": 35
80        }
81      ],
82    },
83    {
84      "id": 36,
85      "text": "Les mercenaires et la princesse sont piégés sur une planète gazeuse par le sbire de l'Empereur",
86      "tone": false,
87      "position": 3,
88      "game_id": 6
89    }
90  ]
91 }

```



XIII. VULNÉRABILITÉS DE SÉCURITÉ ET VEILLE

Les mesures de sécurité ont malheureusement été peu abordées lors de notre formation. En effet, nos cours se sont concentrés sur l'établissement des bonnes pratiques comme la protection contre les injections SQL par exemple. Ce projet était donc l'opportunité de véritablement poser les jalons de ce qu'est l'authentification et de comprendre un aspect primordial des applications web d'aujourd'hui. Avec mon binôme sur le développement *back-end*, nous nous sommes tout de suite intéressés au JSON Web Token.

Notre première question s'est portée sur la manière de stocker les jetons et leur durée de vie, ou plutôt quand s'en débarrasser. J'ai instinctivement pensé à la possibilité de les stocker dans notre base de données. J'ai remarqué que certains utilisateurs sur stack overflow l'utilisaient de cette manière (<https://stackoverflow.com/questions/42763146/does-it-make-sense-to-store-jwt-in-a-database>) donc j'ai tout de suite voulu vérifier cette pratique. J'ai vite compris que c'était une mauvaise idée car cette méthode ne nous permettra pas de profiter de la fonctionnalité d'expiration de ces jetons. D'autre part, cela induirait un trafic superflue sur la base de données qui serait sollicitée à chaque navigation réglementée par notre application.

L'équipe de développement *front-end* a décidé de les stocker dans le *local storage*. Je pense que l'idéal aurait été de les stocker dans un *Cookie* (<https://dev.to/gkoniaris/how-to-securely-store-jwt-tokens-51cf>). Cela nous aurait permis de nous protéger des attaques XSS (Cross-site Scripting).

Nous n'avons pas encore pu changer ce stockage mais il sera réalisé prochainement. Au moment de cette implémentation, nous étions très préoccupés par la manière d'utiliser les JWT afin de cloisonner notre application.

> Voici comment nous avons procédé :

Comme vous avez pu le constater dans les *users stories*, un simple visiteur n'a pas accès au cœur de l'application. Il n'a donc pas la possibilité de créer et de participer à une partie. Pour permettre à l'utilisateur de créer et de participer à une partie, des *middlewares* vérifiant les *tokens* sont placés pour accéder aux routes concernées.

```
//-----Member----->
// Profile
router
  .route('/profile/:id')
  .get(authUser, userController.getOne)
  .patch(authUser, userController.update)
  .delete(authUser, userController.delete);

// Logout occurs in front

//Verify logged in
router.post('/verifsignin', authController.verifyToken);

// New game creation
router.post('/createNewGame', authCreate, gameController.createNewGame);

// Send starting game data
// The following route is meant to update the current with game with all the data needed to
// start a game
router.post('/game/:id/starting', authCreate, gameController.deployGame);

// Access game & refresh
router
  .route('/game/:id/ongoing')
  .get(authGame, gameController.getOne)
  .post(authGame, cardController.createCard)
// Finish game
  .patch(authGame, gameController.endGame);
```

Pour pouvoir passer ces *middlewares*, l'utilisateur se voit distribuer un jeton à sa connexion.

```
async login (request, response) {  
  try {  
    // Validate user input  
    const { username, password } = request.body;  
  
    if (!(username && password)) {  
      return response.status(400).json({ errorMessage: `All input is required` });  
    }  
    // Validate if user exist in our database  
    const user = await userDatamapper.findUserByUsername(request.body.username);  
  
    if (!user) {  
      return response.status(404).json({ errorMessage : "user not found" });  
    }  
    // Validate if password is correct using bcrypt  
    const passwordVerified = await bcrypt.compare(request.body.password,  
      user.password);  
  
    if (user && passwordVerified) {  
      // Create JSON token  
      const token = jwt.sign(  
        { userId: user.id, userName: username, userRole: user.role },  
        process.env.TOKEN_KEY,  
        {  
          expiresIn: "6h",  
        }  
      );  
  
      // We send our user  
      console.log(`user connected as ${user.role}`);  
  
      return response.status(200).json({ username: user.username, email: user.email,  
        userId: user.id, token });  
    }  
    else {  
      return response.status(403).json({ errorMessage : "Invalid Credentials" });  
    }  
  } catch (err) {  
    return response.status(404).json({errorType: err.message, errorMessage: "Unable to  
check credentials"});  
  }  
},
```

Lorsque l'utilisateur rentre le bon nom de profil et le bon de mot de passe, un jeton est généré par la méthode “jwt.sign()”.

Cette méthode prend, en premier argument, un objet “payload” qui va contenir les données importantes nécessaires au bon fonctionnement de l'application (<https://jwt.io/introduction>). C'est en vérifiant ces informations, que l'application sera en mesure de correctement diriger l'utilisateur.

Le deuxième argument pris par cette méthode est la clé permettant de déchiffrer le jeton. Cette variable est stockée dans notre fichier de variables d'environnements qui stocke une partie des données sensibles de notre application.

Cette méthode s'occupe, en troisième argument, d'indiquer au jeton sa durée de vie depuis la création de ce dernier. Dans notre cas, nous avons attribué à chaque jeton une durée de vie de six heures car nous avons estimé qu'il était peu probable qu'une partie se déroule plus de six heures en continue.

> Remarque d'évolution de sécurité

C'est au moment du développement de la méthode de sécurisation et d'authentification, qu'il aurait été intéressant d'intégrer le jeton (généré par la méthode "jwt.sign()") à un *cookie* comportant un paramètre "httpOnly". Cela nous aurait permis de nous protéger des attaques XSS.

Le stockage du jeton dans le *localStorage* du client (et non dans un *cookie*) offre, à l'utilisateur, la possibilité d'accéder aux routes nécessitant une authentification. Le joueur peut s'authentifier grâce à un *middleware* utilisant la méthode "jwt.verify()".

```
const jwt = require("jsonwebtoken");

module.exports = async (request, response, next) => {

    try {
        const token = request.headers.authorization.split(' ')[1];
        console.log("token", token);

        jwt.verify(token, process.env.TOKEN_KEY, function (err, decoded){
            if (err) {
                return response.json({ errorMessage: "Token invalid" });
            }
            else {
                next();
            }
        });

    } catch (err) {
        return response.json({errorType: err.message, errorMessage: "Unable to verify the token"});
    }
}
```

La méthode "jwt.verify()":

- vérifie le jeton mis à disposition (1er argument)
- déchiffre le jeton grâce à la clé trouvée dans les variables d'environnements (2ème argument)
- exécute une fonction qui renvoie une erreur si le jeton n'est pas valide ou un *next* en cas de validité (3ème argument)

Ce jeton Web JSON est utilisé de plusieurs manières :

- Pour la création de partie : il va simplement vérifier la validité du jeton. Une personne connectée, qui est donc inscrite dans la base de données, peut alors créer une partie.
- Pour consulter et modifier sa page de profil : le jeton nous permet de vérifier si le nom d'utilisateur correspond à celui de la page de profil.
- Pour rejoindre une partie : le *middleware* “authGame” va vérifier si l'utilisateur a la possibilité de participer à la partie ou non. Il va vérifier, après avoir déchiffré le jeton du joueur, si son pseudonyme d'utilisateur est associé à la partie qu'il souhaite rejoindre.



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

> Anglais

Why should we use JSON Web Tokens?

Let's talk about the benefits of JSON Web Tokens (JWT) when compared to Simple Web Tokens (SWT) and Security Assertion Markup Language Tokens (SAML).

As JSON is less verbose than XML, when it is encoded its size is also smaller, making JWT more compact than SAML. This makes JWT a good choice to be passed in HTML and HTTP environments.

Security-wise, SWT can only be symmetrically signed by a shared secret using the HMAC algorithm. However, JWT and SAML tokens can use a public/private key pair in the form of a X.509 certificate for signing. Signing XML with XML Digital Signature without introducing obscure security holes is very difficult when compared to the simplicity of signing JSON.

JSON parsers are common in most programming languages because they map directly to objects. Conversely, XML doesn't have a natural document-to-object mapping. This makes it easier to work with JWT than SAML assertions.

Regarding usage, JWT is used at Internet scale. This highlights the ease of client-side processing of the JSON Web token on multiple platforms, especially mobile.

If you want to read more about JSON Web Tokens and even start using them to perform authentication in your own applications, browse to the JSON Web Token landing page at Auth0.

> Français

Pourquoi devrions nous utiliser Jetons Web Json ?

Parlons des bénéfices des Jetons Web JSON (JWT) comparés aux Jetons Web Simple (SWT) et aux Jetons de Langage de Balisage d'Assertion de Sécurité (SAML).

Comme le JSON est moins verbeux que le XML lorsqu'il est encodé, sa taille est aussi plus petite. Le JWT est donc plus compact qu'un SAML. Cela fait du JWT un choix judicieux pour être passé dans les environnements HTML et HTTP.

Sur la question de la sécurité, SWT peut seulement être signé de manière symétrique par un secret partagé qui utilise l'algorithme HMAC. À contrario, les JWT et des jetons SAML peuvent utiliser une paire de clés publiques/privées sous la forme d'un certificat X.509 pour la signature. Signer le XML avec une Signature Digitale XML sans y introduire des failles de sécurité obscure est très difficile quand on le compare à la simplicité de la signature JSON.

Les analyseurs JSON sont communs à la plupart des langages de programmation car ils réfèrent directement à des objets. Inversement, XML ne possède pas cette équivalence naturelle de référencement de document à objet. Cet aspect rend le travail avec les JWT plus facile que celui avec les assertions SAML.

Concernant son utilisation, JWT est utilisé à l'échelle d'Internet. Cela souligne l'aisance du processus côté client du Jeton Web JSON sur une multitude de plateformes, spécialement mobiles.

Si vous désirez en lire plus à propos des Jetons Web JSON et commencer à les utiliser pour réaliser les démarches d'authentification sur vos applications, je vous invite à visiter la page d'accueil du Jeton Web JSON sur le site Auth0.



XV. CONCLUSION

J'ai, à présent, une bien meilleure perception du temps de développement d'une application. Je visualise encore mieux l'importance du travail d'équipe. Au-delà de la force de travail, elle permet de constamment redynamiser le projet et rend plus constant le volume de travail. Il y a une espèce de responsabilité partagée qui a vraiment un effet bénéfique sur l'entièreté du processus. Même si j'ai acquis plusieurs d'expériences dans ce domaine, je pense devoir renforcer mes capacités personnelles à ce niveau là. J'ai réalisé à quel point la réussite du travail d'un développeur est dépendant de l'équipe avec laquelle il collabore.

J'ai vraiment apprécié l'organisation du travail en méthode Agile et j'ai hâte de recommencer. Je pense avoir vraiment gagné en efficacité et rigueur de travail. Le fait d'avoir découpé tous nos objectifs m'a permis de gagner en compréhension. J'étais aussi moins préoccupé par l'état général de l'application, ce qui a eu un effet bénéfique sur ma concentration.

Même si j'ai pu approfondir les compétences apprises pendant ces mois de formations, j'ai le sentiment de n'avoir qu'effleurer les technologies utilisées. J'aimerais vraiment monter en compétences sur les questions de sécurité. D'autre part, j'aurais aimé concevoir des Index PostgreSQL pour améliorer la rapidité d'accès aux données. Je suis tout de même satisfait d'avoir un MVP complet sur l'application *back-end*.

J'ai l'intention de continuer à coder cette application sur mon temps personnel. Je pourrai aussi m'initier aux technologies *front*. J'ai très envie de présenter un prototype à l'auteur du jeu Microscope que nous avons souhaité adapter.

Pour finir, je dirais que cette étape a bien confirmé ma reconversion professionnelle. Je dirais même que c'était assez naturel de passer d'études et de travail créatif au métier de développeur. J'avance aujourd'hui d'un pas serein et plein d'entrain. J'ai hâte de trouver mon premier poste dans le secteur du Web afin de continuer d'apprendre et de mettre à profit toutes les compétences que j'ai acquises jusqu'ici.



XVI. ANNEXES

A. Résumé des règles pour la page d'accueil

Bienvenue sur l'application microscope web. Cette dernière vise à adapter Microscope, un jeu d'écriture collaboratif publié pour la première fois en 2011 aux Etats-Unis.

Ce jeu a pour objectif, grâce à un système de carte, de permettre l'écriture d'un récit de manière non-linéaire.

Ainsi, le premier joueur pourrait s'intéresser à la conclusion de l'histoire et le suivant pourrait se concentrer sur une période antérieure.

Le premier objectif d'une partie est d'incarner un simple spectateur lors du tour des autres joueurs, puis de devenir l'auteur d'une partie de l'histoire quand vient son tour, sans l'influence des autres participants.

Grâce à sa structure morcelée, les parties sont capables de générer des coups de théâtres et autres rebonds transformant une simple anecdotes en instant narratif majeur.

Plus de détails :

Une partie a besoin de quatre élément pour démarrer:

- une Vue d'Ensemble
 - Exemple :
 - *Un Empire tyrannique chute grâce à la un mouvement rebelle*
- le Début et la Fin de l'histoire jouée
 - Exemples :
 - *L'Empire met au point une station pouvant détruire des planètes entières.*
 - *L'Empereur est vaincu et ses soldats fuent à travers la galaxie.*
- un Palette réunissant des thématiques à faire figurées ou à proscrire de la partie
 - Exemples :
 - OUI Ordre de chevalier, Magie
 - NON Espace Réaliste
- une Première Passe de carte “Périodes” et “Événements” défini plus tard
 - Exemples :

- Carte "Période", Le seigneur Vador tend un piège à la Résistance sur la planète Bespin
- Carte "Période", La Résistance planifie et détruit l'Etoile Noire
- Carte "Événement", Le jeune fermier Luke trouve un contrebandier pour échapper au blocus de la planète Tatooine

Il existe trois échelles narratives dans le jeu :

Des Périodes définissant les grands arcs narratifs

- Exemple : un jeune fermier rejoint la résistance et en devient le héros.

Des Événement ponctuant les périodes en cours

- Exemple : un vaisseau transportant des informations critiques à l'existence de la résistance est poursuivi par l'empire.

Et enfin les Scènes (qui ne sont pas permises pour la Première Passe) qui permettent de questionner et de résoudre le déroulement des événements

- Exemple : la jeune diplomate, secrètement membre de la résistance, arrivera-t-elle à fuir son vaisseau ?

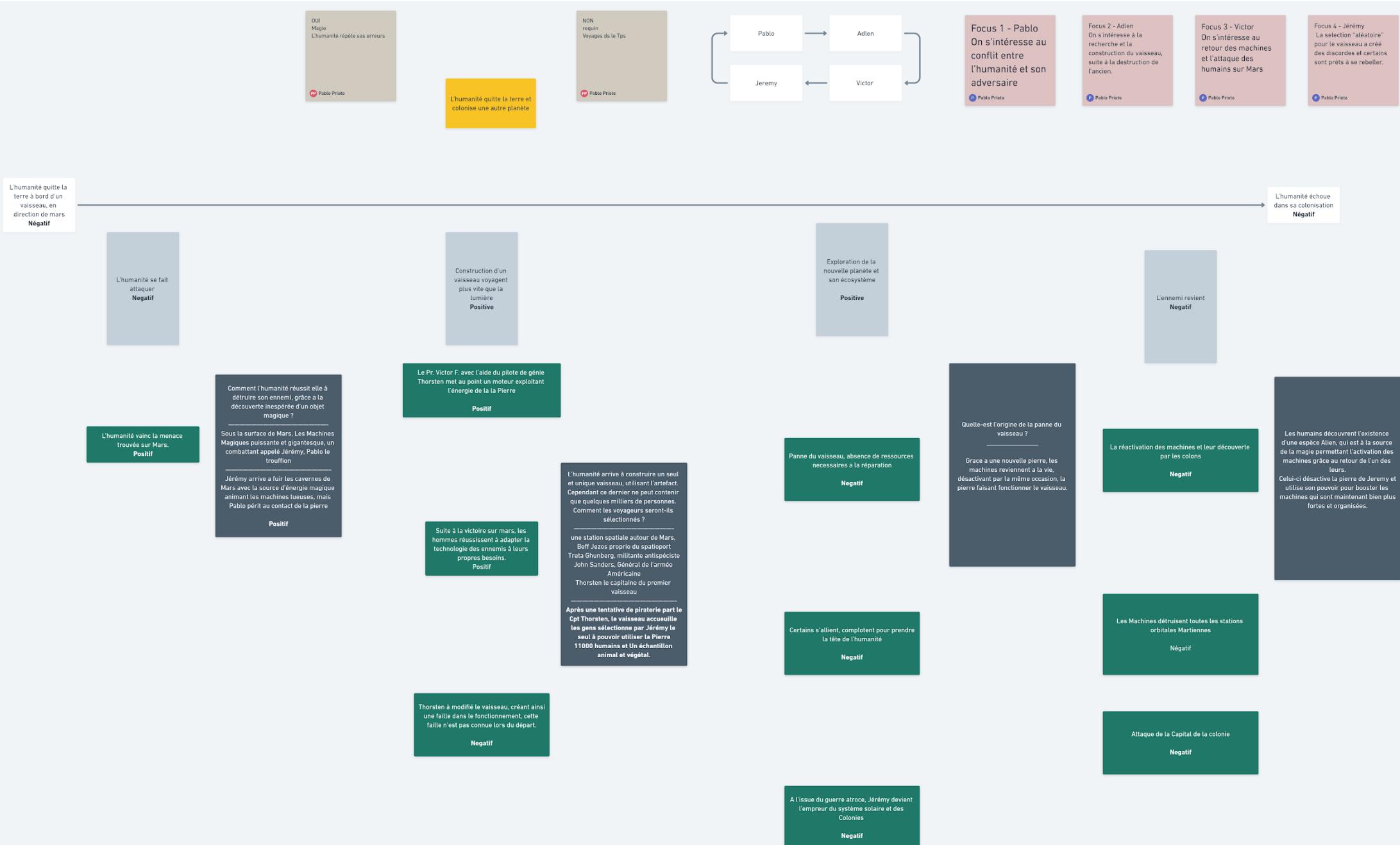
Notre application permet également de produire des cartes Focus !

Elles permettent de définir une thématique ou un point d'intérêt pour le tour de table.

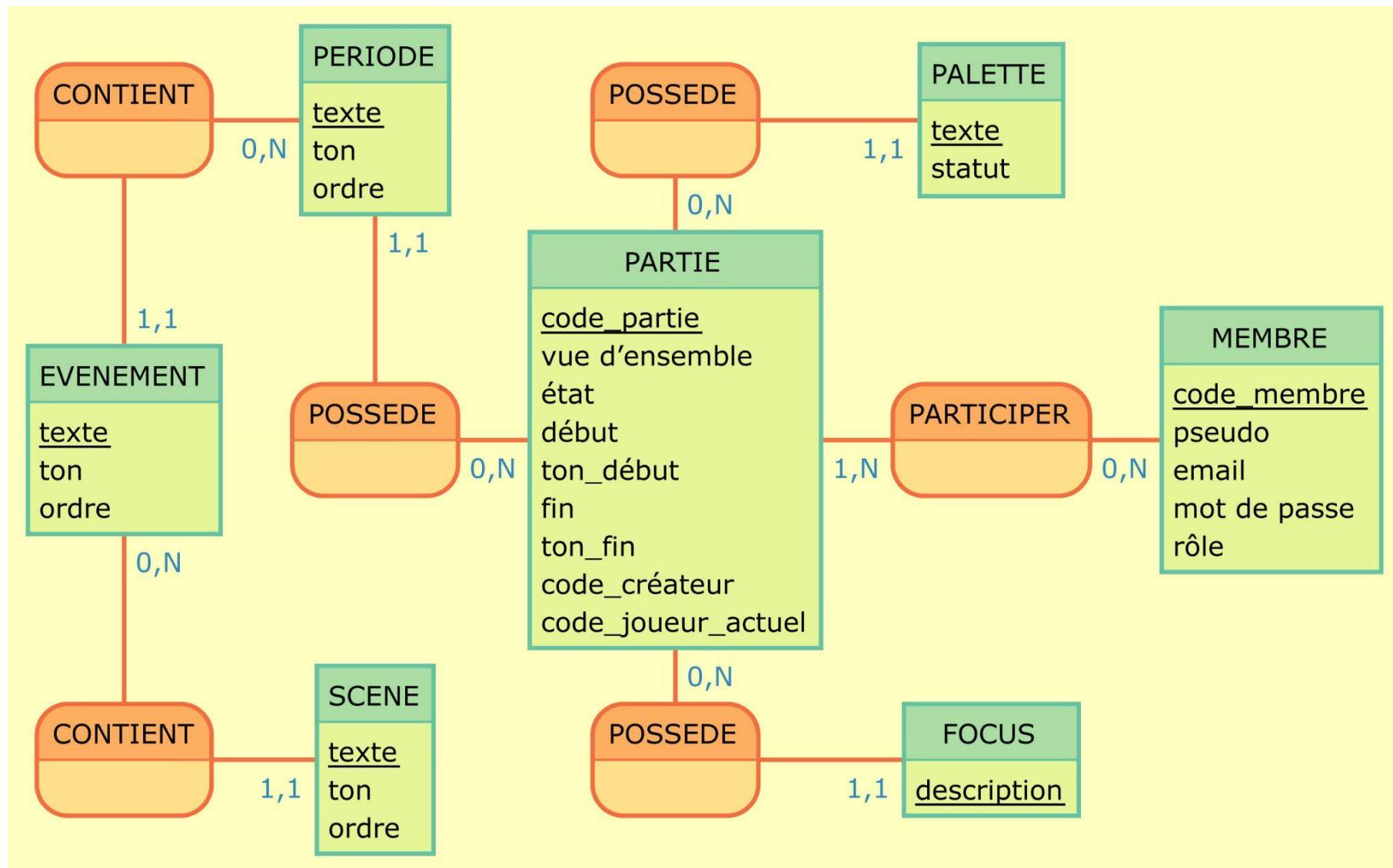
- Exemples: le côté obscur de la force, le personnage du jeune fermier, les voyages à travers les étoiles, etc....

D'autres fonctionnalités seront ajoutées à notre application pour s'adapter à la vision de son auteur Ben Robbins. :)

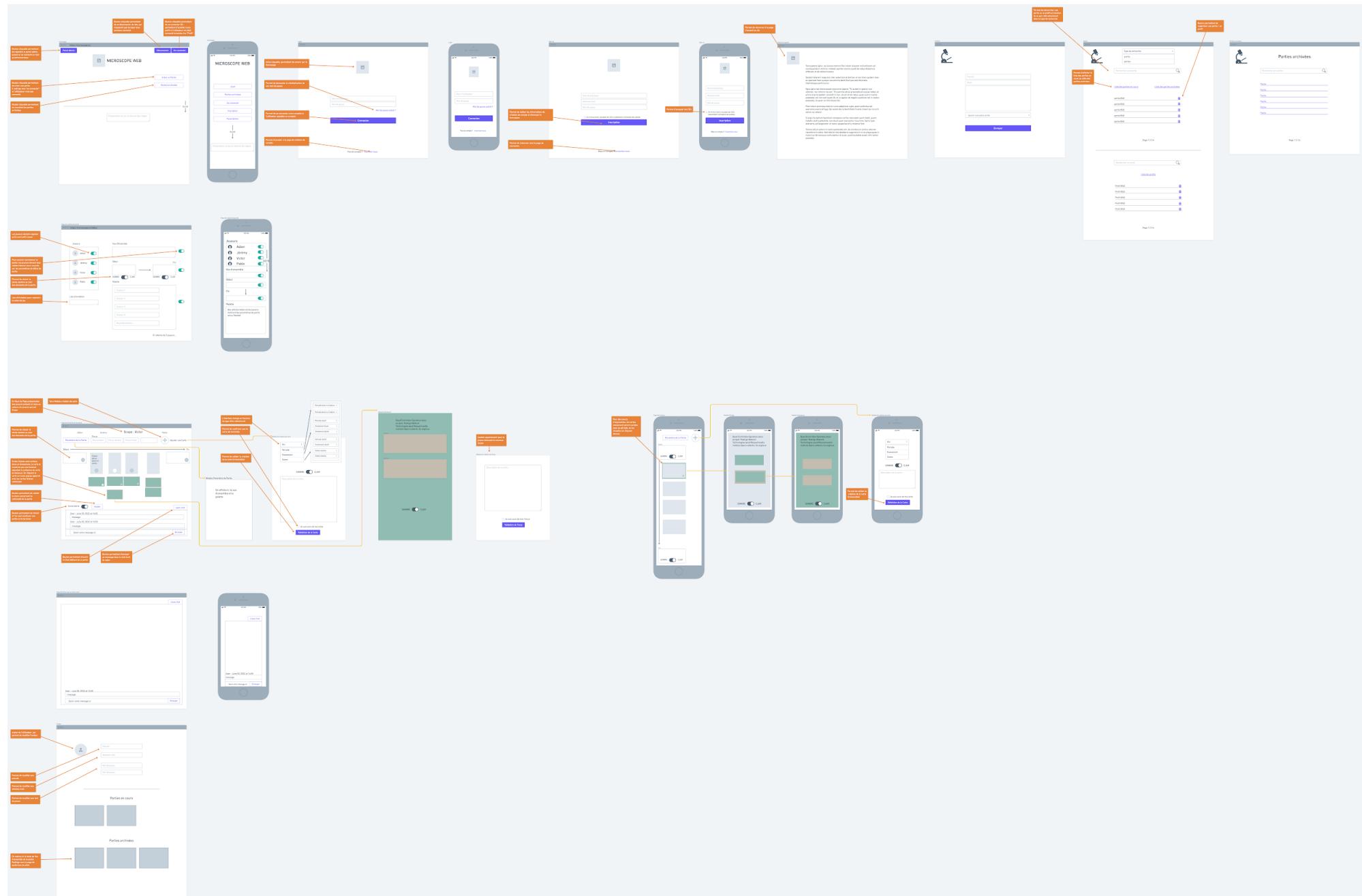
B. Exemple de partie

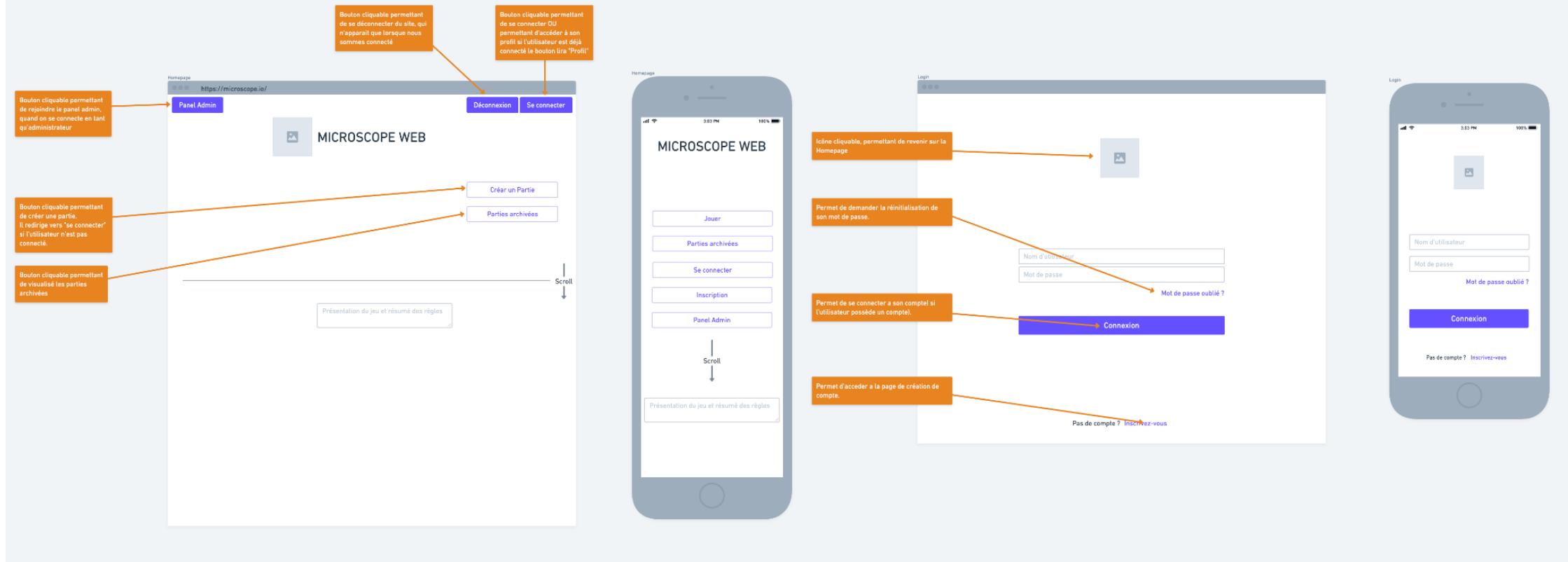


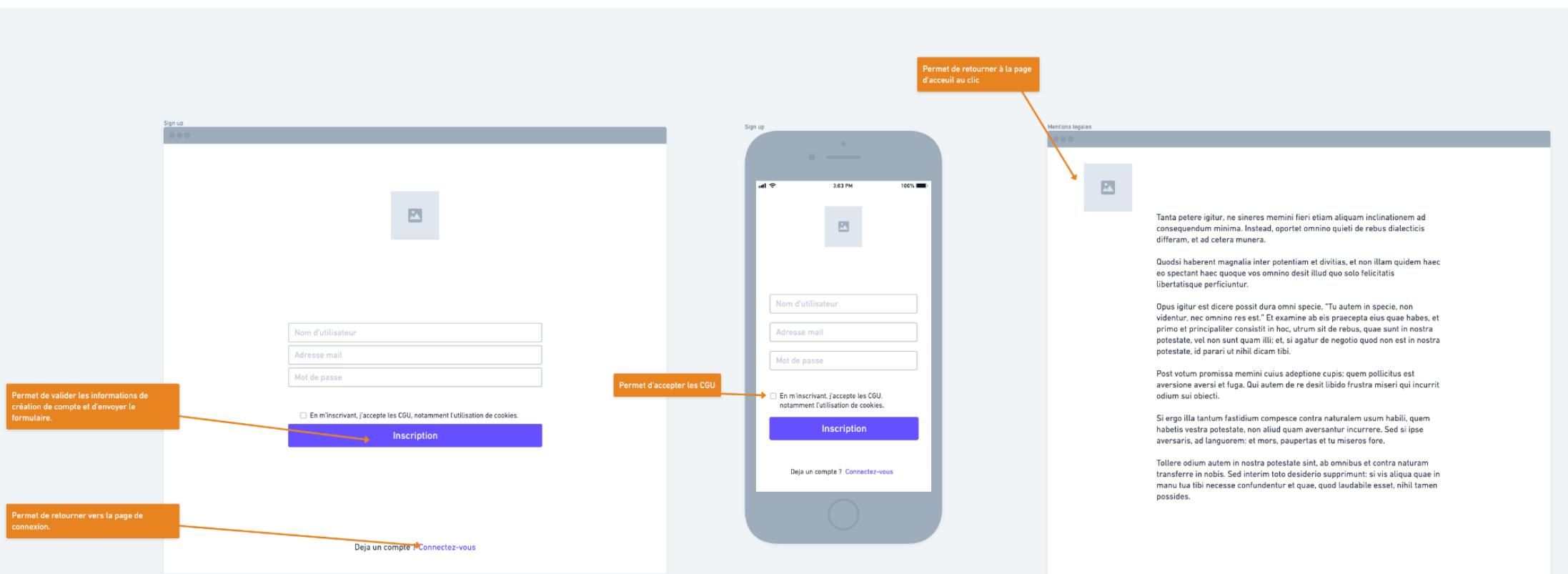
C. MCD



D. Wireframes (Vue Globale & Détails)







Page de création de partie

<https://microscope.io/lobby/>

The screenshot shows the creation room page with the following interface elements:

- Joueurs (Players):** A list of four players: Adlen, Jérémie, Victor, and Pablo, each with a toggle switch.
- Vue d'Ensemble (Overall View):** A large empty rectangular area with a toggle switch to its right.
- Début (Start):** A section with two rectangular boxes labeled "SOMBRE" and "CLAIR" with toggle switches between them.
- Fin (End):** A section with two rectangular boxes labeled "SOMBRE" and "CLAIR" with a toggle switch to its right.
- Palette (Color Palette):** A list of five color swatches labeled "Couleur 1" through "Nouvelle Couleur ...".
- Lien d'invitation (Invitation Link):** A text input field containing a URL.
- Message at the bottom:** "En attente de 2 joueurs..." (Waiting for 2 players...).

Annotations on the left side of the page:

- "Les joueurs doivent signaler qu'ils sont prêt à jouer" (Players must signal they are ready to play)
- "Pour pouvoir commencer la partie, les joueurs doivent tous valider/donner leurs accords sur les paramètres de début de partie" (To start the game, all players must validate/accept the start parameters)
- "Permet de choisir la teinte sombre ou clair des éléments de la partie" (Allows choosing the dark or light tint of the game elements)
- "Lien d'invitation pour rejoindre le salon de jeu" (Invitation link to join the game room)

Page de création de partie

3:03 PM 100%

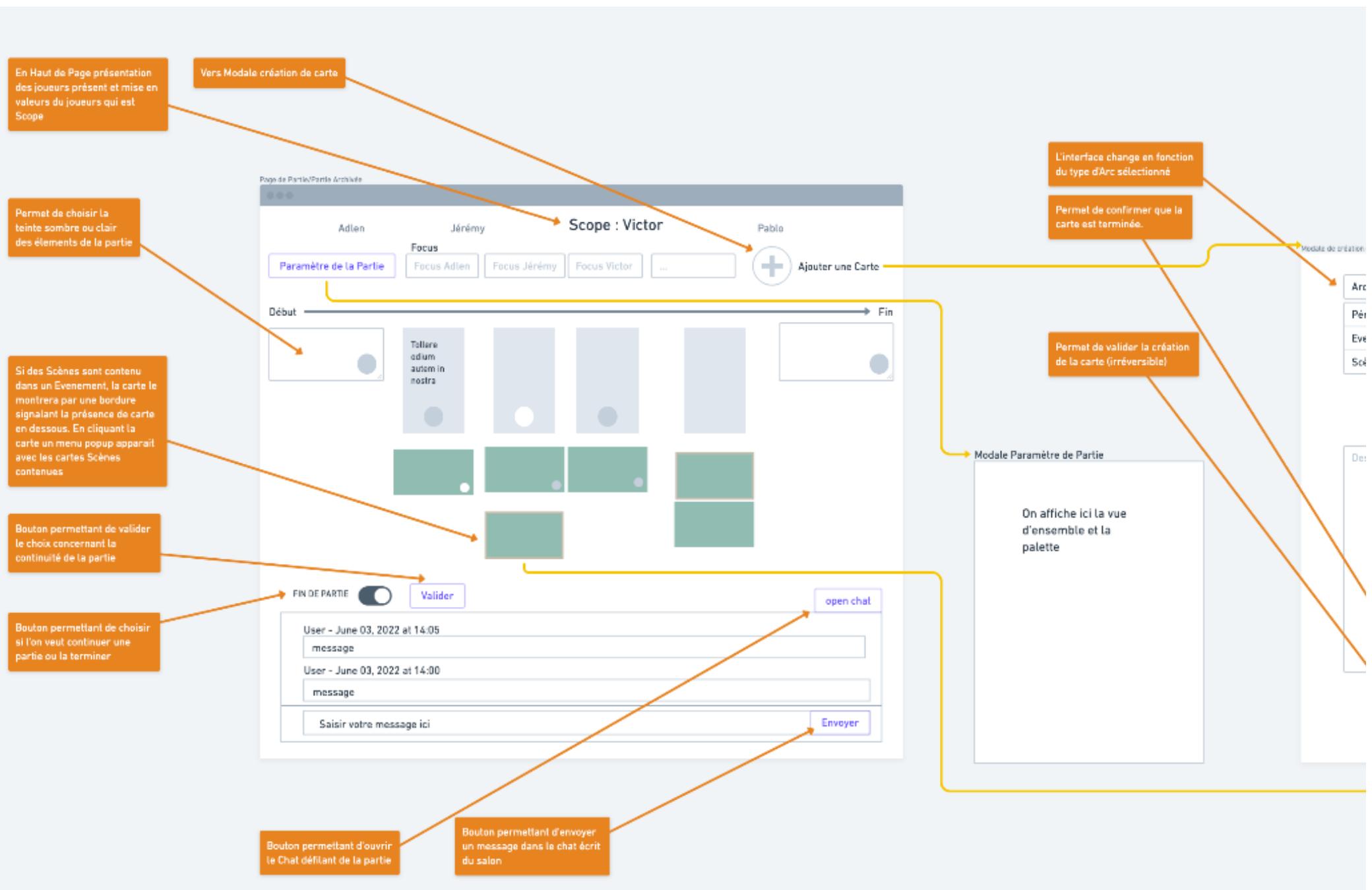
The mobile phone screen shows the same creation room interface as the desktop version, with the following details:

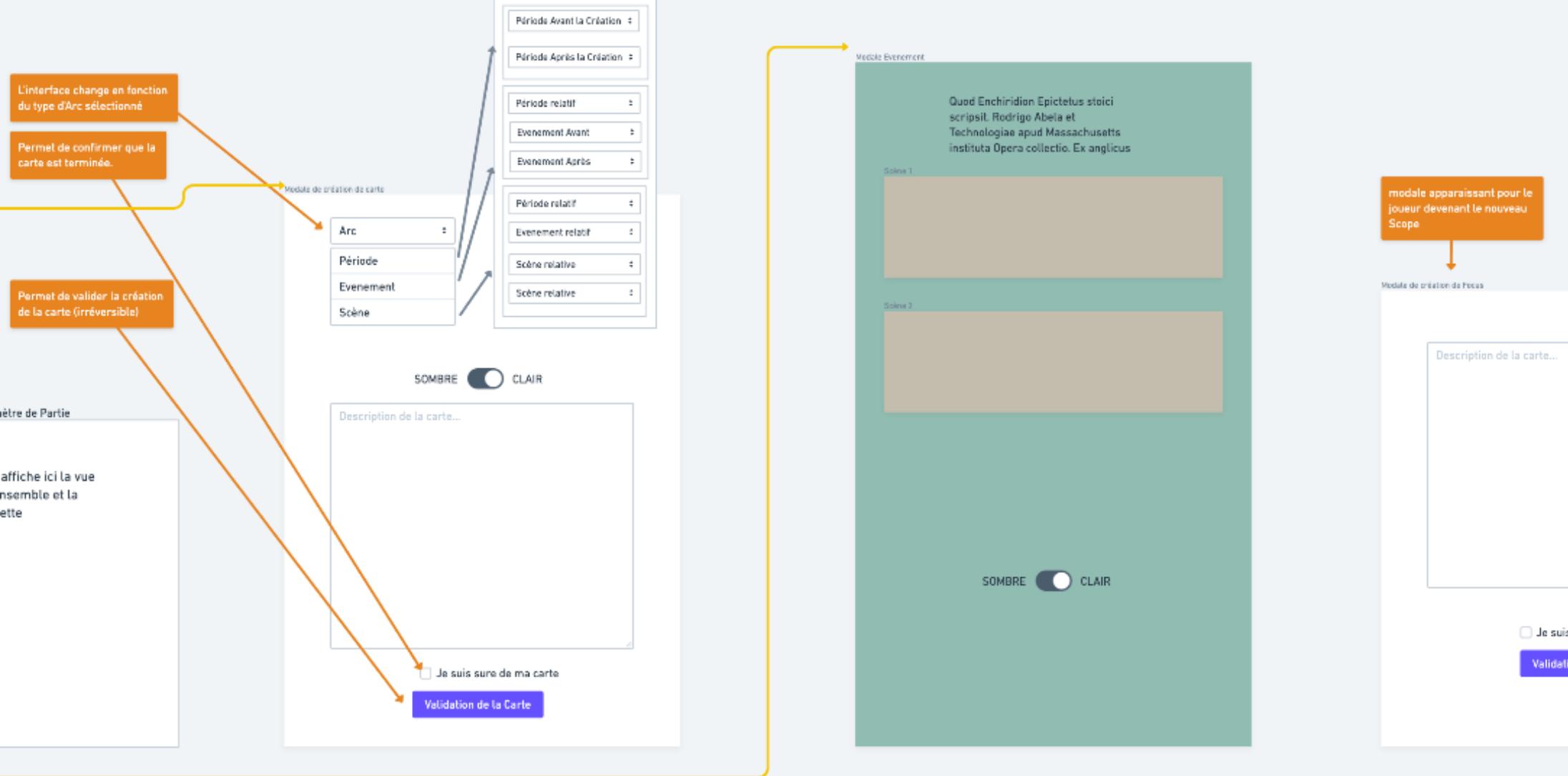
- Joueurs (Players):** Shows the same four players (Adlen, Jérémie, Victor, Pablo) with their status toggles.
- Vue d'ensemble (Overall View):** A large empty rectangular area with a toggle switch to its right.
- Début (Start):** A section with two rectangular boxes labeled "SOMBRE" and "CLAIR" with a toggle switch between them.
- Fin (End):** A section with two rectangular boxes labeled "SOMBRE" and "CLAIR" with a toggle switch to its right.
- Palette (Color Palette):** A list of five color swatches labeled "Couleur 1" through "Nouvelle Couleur ...".

Annotations on the right side of the mobile screen:

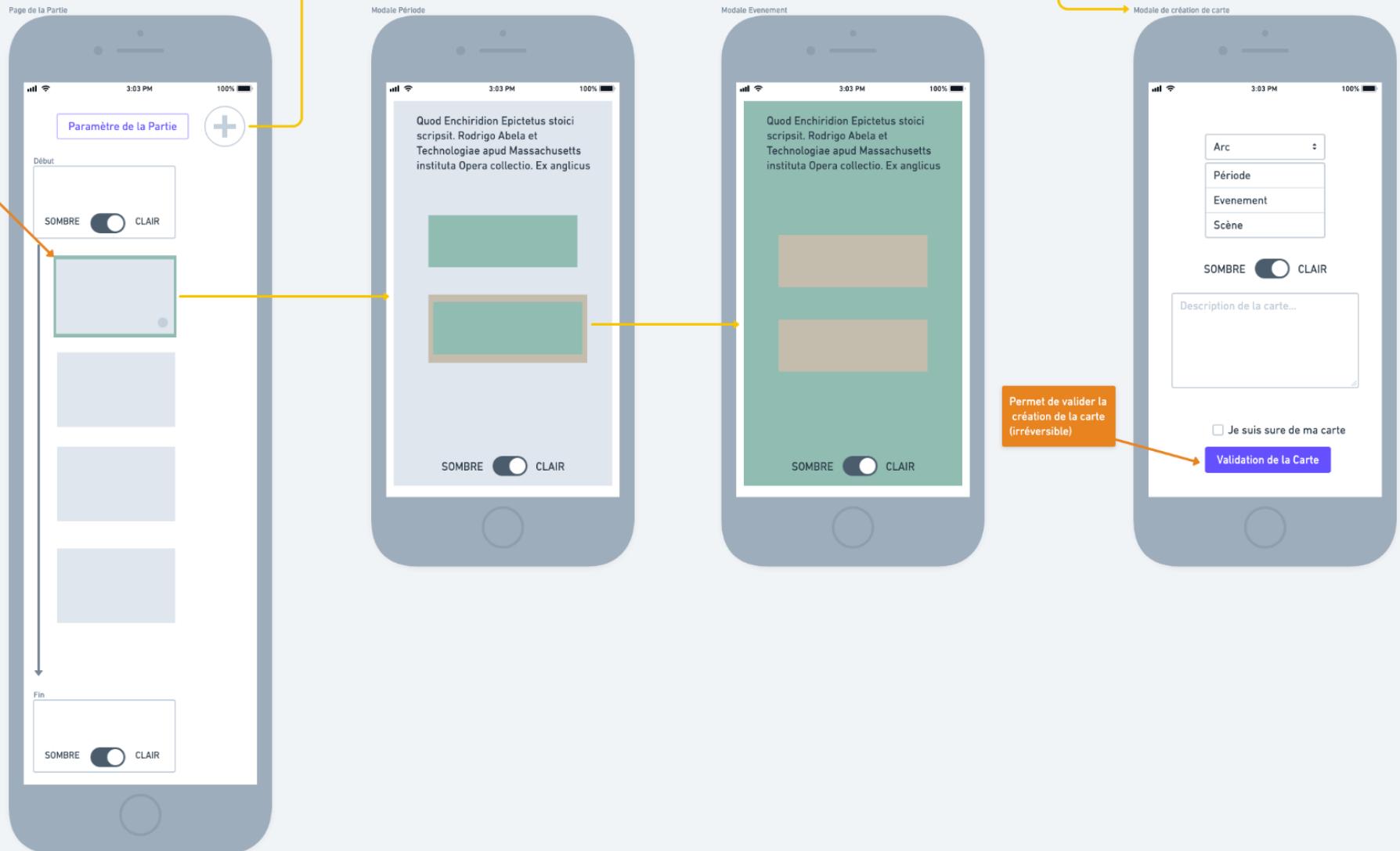
- "scroll" (scrolling indicator)
- "Vue d'ensemble" (Overall View)
- "Début" (Start)
- "Fin" (End)
- "Palette" (Color Palette)

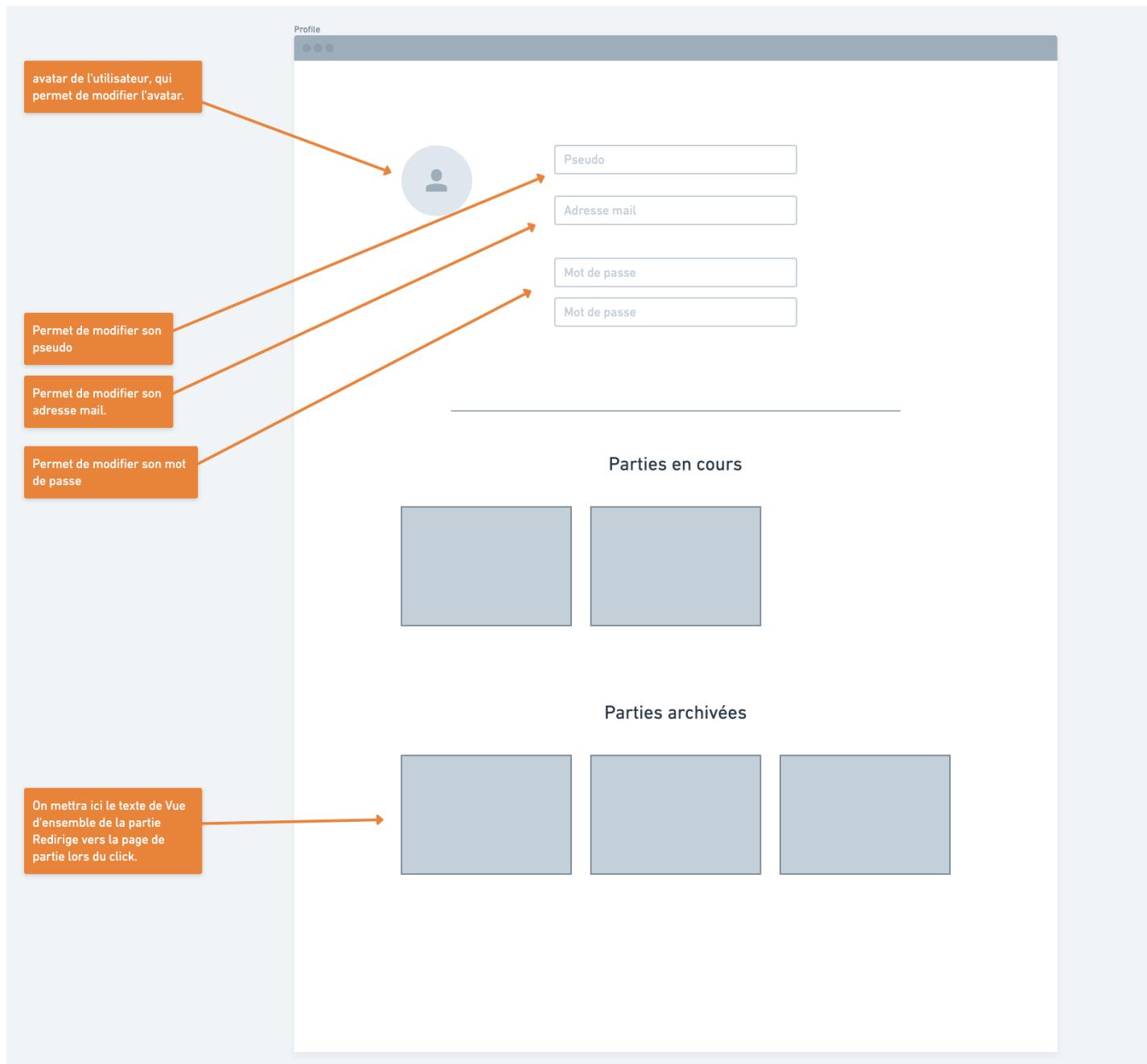
Text in the bottom right of the mobile screen: "des cellules textes où les joueurs mettront des paramètres de partie (a.k.a. Palette)" (text cells where players will set game parameters (a.k.a. Palette))





Pour des soucis d'ergonomie, les cartes evenement seront cachées sous sa période, on les visualise en cliquant dessus.





E. Tableau de Données

Relation	Nom	Description	Type	Contraintes	Expression Régulière
member	id	Identification du Membre	int	GENERATED ALWAYS AS IDENTITY	
	pseudo	Pseudonyme/Nom du Membre	text	NOT NULL, UNIQUE	
	email	Adresse Mail du Membre	text	NOT NULL UNIQUE	<code>^[\w-\.]+@[(\w-]+\.)+[\w-]{2,4}\$</code>
	mot de passe	Mot de passe du Membre	text	NOT NULL UNIQUE	<code>^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[a-zA-Z]).{8,}\$</code>
	rôle	Rôle du Membre	text	NOT NULL, DEFAULT "member"	
game	id	Identification de la Partie	int	GENERATED ALWAYS AS IDENTITY	
	vue d'ensemble	Nom & Thématique de la Partie	text	NOT NULL	
	état	État de la partie	text	NOT NULL	
	début	Début de la partie	text	NOT NULL	
	ton_début	Mode clair ou sombre du début	boolean	NOT NULL,	
	fin	Fin de la partie	text	NOT NULL	
	ton_fin	mode clair ou sombre de la fin	boolean	NOT NULL	
	creator_id	clé étrangère vers la table user	int	REFERENCES user(id)	
focus	current_player_id	clé étrangère vers la table user	int	REFERENCES user(id)	
	description	Description du focus	text	NOT NULL	
	author_id	auteur du focus	int	NOT NULL	
period	position	position à l'ordre de création du focus	int	NOT NULL, UNIQUE	
	text	Description de la période	text	NOT NULL,	
	period_tone	Mode clair ou sombre de la période	boolean	NOT NULL	
event	position	Ordre de placement de la période	int	NOT NULL	
	texte	Description de l'événement	text	NOT NULL, UNIQUE	
	event_tone	Mode clair ou sombre de l'événement	boolean	NOT NULL	
scene	position	Ordre de placement de l'événement	int	NOT NULL	
	texte	Description de la scène	text	NOT NULL, UNIQUE	
	scene_ton	Mode clair ou sombre de la scène	boolean	NOT NULL	
palette	position	Ordre de placement de la scène	int	NOT NULL	
	text	Description de la palette	text	NOT NULL	
participation	status	Statut de la palette	boolean	NOT NULL	
	id_membre	Identification du membre participant	int	NOT NULL, UNIQUE	
	id_partie	Identification de la partie auquel on participe	int	NOT NULL, UNIQUE	
	position	Position du joueur dans la liste des joueurs	int	NOT NULL	