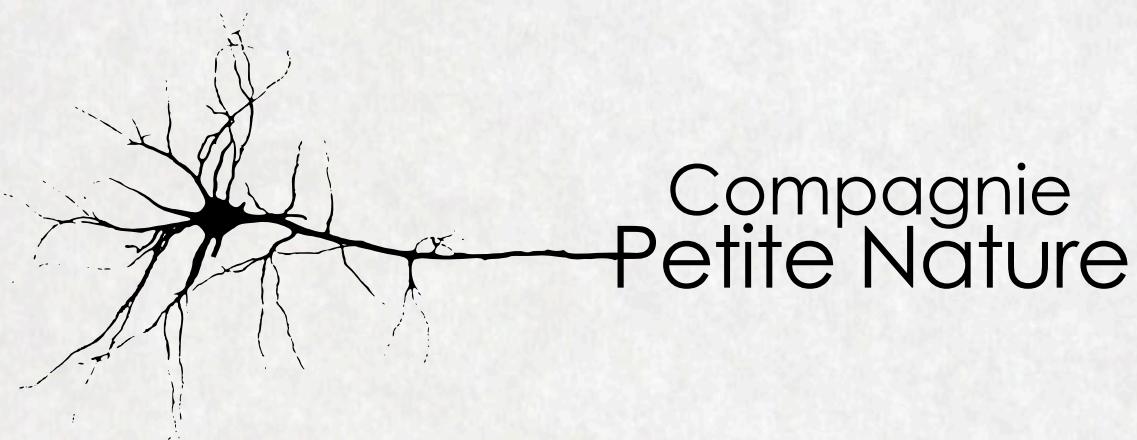


RAPPORT DE STAGE

DU 11/03/2024 AU 21/06/2024



IUT DE JEAN JAURES BLOIS

BUT MMI PARCOURS DEVELOPPEMENT WEB
ET DISPOSITIFS INTERACTIFS

TABLE DES MATIÈRES

I. REMERCIEMENT	2
II. INTRODUCTION	3
1.D'où venons-nous ? Que sommes-nous ?	3
A. Qui suis-je ?	3
B. Contexte du stage	3
2. La compagnie petite nature	4
A. L'association compagnie petite nature	4
B. Quelle est ma mission dans la compagnie ?	6
3. Haïkus Numériques	6
A. Micro Haïkus numériques	6
B. Atelier, Exposition et résidences	7
III. EPHÉMERIDE	8
1. Le projet 12 Poèmes Numériques	8
A. Le spectacle	8
B. Le rôle d'Éphéméride au sein de 12 PMN	9
C. Organisation numérique	10
2. Le développement de l'application	11
A. Architecture de l'application	11
1. L'expérience utilisateur dans Ephéméride	11
2. Le système de navigation	13
B. Composant interactifs	15
1. Mois de MAI: Berlin	15
2. Mois de juillet : Papillon	17
3. Mois de septembre : Rentrée	19
C. Contrôle distant	21
3. Le Serveur webSocket	23
A. Le WebSocket, le moteur de l'application	23
1. L'utilisation du serveur WebSocket	23
2. Gestion des événements	24
B. Système d'utilisateur et d'état	25
1. Gestion des données des utilisateurs	25
2. Gestion de l'état de l'application en cours	26
4. La mise en production de l'application	28
A. Automatisation	28
B. Serveur Apache et sécurité	29
IV. STRAPI & MAX MSP, LA GESTION ADMIN	30
1. STRAPI, LE CHOIX DU BACK OFFICE	30
A. Une base de donnée NO CODE	30
B. Utilisation de l'API et la media library	30
2. MAX MSP	31
V. SOLUTIONS ENVISAGEES	32
1. LE CHOIX DU BACK OFFICE	32
A. Back office EBO	32
B. Back office directus	32
2. LES VERSIONS BÊTA DES FONCTIONNALITES	33
A. Les fonctionnalités implémentées	33
B. Les solutions en développement	34
VI. DIFFICULTES RENCONTREES	35
1. Le couple React/Socket	35
2. L'humain, dans les ateliers et l'application	36
3. Manque de pratique & la gestion de serveur	36
VII. BILAN DU STAGE	37
Où allons nous ?	37

I. REMERCIEMENT

Avant d'entamer le développement de ce rapport, je souhaite exprimer ma gratitude envers toutes les personnes qui ont rendu possible cette immersion en entreprise. Je remercie celles et ceux qui m'ont accompagné tout au long de ces 15 semaines de stage, qui ont partagé leur temps et leurs connaissances, ou qui ont contribué, de près ou de loin, à la réussite de cette expérience. Je tiens donc à remercier :

- L'équipe du **Café Charbon** ainsi que celle de **La Maison de la Culture**, qui nous ont chaleureusement accueillis durant la tournée à Nevers du 14/05 au 21/05 et ont créé des conditions optimales pour un travail efficace.
- L'équipe de **l'Espace Mendès France de Poitiers**, qui nous a accueillis au sein de leur espace culturel pour une résidence du 03/06 au 07/06, a prêté leur salle de spectacle et fait de leur mieux pour que nous puissions travailler dans les meilleures conditions.
- **M. Jonathan Ochej**, la personne que j'ai remplacée pendant le stage et qui, malgré son départ de la compagnie, n'a pas hésité à consacrer du temps pour me guider et me donner de précieux conseils.
- **M. Simon Couratier** et **Mme. Charlotte Dutilleul**, pour avoir été des collègues de travail les plus agréables, créant une ambiance propice à la bonne humeur et à la motivation.
- Enfin, mon tuteur et maître de stage **M. Alessandro Vuillermin** pour son incroyable bienveillance à mon égard et à l'égard de toutes les personnes avec qui je l'ai vu travailler. Il a tout au long du stage, fait en sorte que je sois dans les meilleures conditions de travail. Grâce à lui, j'ai eu le sentiment d'être un membre à part entière de l'association, et pas seulement un stagiaire.

II. INTRODUCTION

1.D'OU VENONS-NOUS ? QUE SOMMES-NOUS ?

A. QUI SUIS-JE ?

Âgé de 26 ans, je suis actuellement en 3e année de BUT Métier du multimédia et d'Internet parcours développement web et dispositifs interactifs à l'IUT Jean Jaurès à Blois. J'ai suivi un cursus scolaire relativement atypique ; ayant arrêté l'école à l'âge de 15 ans après une année de seconde dans un lycée professionnel de commerce. J'ai décidé à l'âge de 20 ans, de reprendre les études. J'ai donc passé 3 ans au CNED afin d'obtenir mon baccalauréat. J'ai ensuite effectué 2 années de BTS Système Numérique au sein du lycée Antoine Bourdelle à Montauban. Pour finalement réussir à intégrer une formation qui me permettrait de développer mes compétences dans un domaine qui m'attirait depuis ma reprise d'étude, le développement web.

B. CONTEXTE DU STAGE

Dans le cadre de ma formation en BUT Métiers du Multimédia et de l'Internet, parcours Développement Web et Dispositifs Interactifs, un stage en entreprise de 15 semaines est obligatoire lors de la troisième année. Cette période de stage vise à nous immerger dans le monde professionnel, et c'est également une opportunité de mettre en pratique les diverses connaissances acquises durant l'année scolaire, de les perfectionner et d'en développer de nouvelles.

II. INTRODUCTION

Désireux de mettre en œuvre les compétences acquises au cours de l'année, mon objectif en intégrant la compagnie Petite Nature durant cette période de stage était de me constituer une première expérience professionnelle dans le domaine du développement web. Je souhaitais ainsi identifier les compétences manquantes dans un contexte professionnel et déterminer de quelle manière je pouvais les améliorer.

J'ai ainsi eu l'opportunité d'assister à une conférence de **M. Alessandro Vuillermin** nous présentant son travail, portant sur des actions culturelles intégrant des objets connectés et des servomoteurs. Ayant également travaillé sur des objets connectés ainsi que dans un projet en robotique, nous avons donc décidé de collaborer.

2. LA COMPAGNIE PETITE NATURE

A. L'ASSOCIATION COMPAGNIE PETITE NATURE

La Compagnie Petite Nature a été créée le 16 janvier 2007 sous la forme juridique d'une association déclarée par **Elise Truchard** et **Alessandro Vuillermin**. En tant qu'organisme à but non lucratif, l'objectif de la compagnie Petite Nature n'est pas de générer des bénéfices pour les partager entre ses membres. Tous les revenus de la compagnie sont réinvestis dans les activités et les projets de l'association afin de soutenir et de promouvoir ses objectifs. La compagnie est scindée en deux pôles bien distincts qui correspondent aux centres d'intérêts des créateurs.

II. INTRODUCTION

La compagnie est en majorité financée par des subventions du pouvoir public en période de création et principalement la vente de spectacles et d'ateliers hors période de création. Elle est gérée par deux membres du bureau bénévoles que je n'ai pas eu l'occasion de rencontrer. Tous les autres salariés sont des intermittents du spectacle, salariés à la mission.

M. Vuillermin développe des spectacles et des ateliers numériques depuis 2015, année de création du premier spectacle numérique de la compagnie, **La théorie du nuage**. Depuis, il a créé le spectacle **Haïkus numériques**, les ateliers **Micro Haïkus numériques** et **Ornithologie de la pensée**.

B. QUELLE EST MA MISSION DANS LA COMPAGNIE ?

La Compagnie Petite Nature effectue des actions culturelles dans le numérique, qu'elles soient des actions de médiations culturelles, des ateliers mêlant artistique et numérique ainsi que des spectacles. Le numérique est présent partout, dans chacune de leurs actions culturelles. Naturellement, ils ont donc besoin de travailler en collaboration avec un développeur web afin d'améliorer les solutions déjà existantes et d'en élaborer de nouvelles.

Ma mission fut donc de remplacer l'ancien développeur le temps du stage, d'épauler **M. Vuillermin** dans ses projets numériques et de commencer le développement d'une application web nommée Éphéméride pour le prochain spectacle en cours de création et dont la première représentation se fera en 7 février d'année 2025 à Vendôme.

L'application est disponible en utilisant ce lien :

<https://app.petitenature.fr/EPH/>

Ou en scannant le QR CODE ci-contre :



II. INTRODUCTION

3. HAÏKUS NUMÉRIQUES

A. MICRO HAÏKUS NUMÉRIQUES

Micro Haïkus numériques, c'est une version miniature du spectacle **Haïkus numériques** qui propose aux participants de créer leur propre spectacle sur une micro scène. Chaque spectacle dure 45 secondes et se déroule sur la micro scène ainsi que sur une tablette.

Par petit groupe, les participants sont dans un premier temps amenés à écrire un haïkus, c'est un petit poème japonais en 3 lignes. Puis dans un second temps, ils créent une représentation de cet Haïkus à travers la programmation sonore et de petits moteurs reliés à la micro scène.



Figure 1: Atelier Micro HKN

Haïkus numériques et **Micro Haïkus numériques** ont une place très importante dans la compagnie, c'est actuellement le seul spectacle en tournée de la compagnie et l'atelier qui est le plus demandé.

II. INTRODUCTION

B. ATELIER, EXPOSITION ET RÉSIDENCES

A côté du travail de réflexion et de développement de l'application, la compagnie petite nature m'a également permis de les accompagner lors de leurs tournées et lors de leurs résidences. J'ai également pu les suivre lors d'ateliers qu'ils exercent dans des écoles ou en médiathèque.

Le premier atelier dans lequel j'ai pu intervenir a été au **Lycée Paul Gauguin** situé à Orléans, j'ai pu participer à l'encadrement durant 3 jours d'adolescents. En les guidant dans leur démarche de création, les conseillant dans la rédaction de leur haïkus ainsi que dans l'utilisation de l'application **Micro Haïkus Numériques**.

On m'a également confié la gestion d'une exposition d'élèves ayant réalisé leur propre **Micro Haïkus Numériques** qui s'est tenue à Nevers du 14/04 au 21/04. Durant cette période, j'ai pu accompagner les visiteurs tout au long de l'exposition, veiller à la sécurité du matériel, au bon fonctionnement de celui-ci ainsi qu'à l'installation de cette exposition.

Ces actions représentent une part non négligeable de ma contribution au sein de la compagnie durant cette période de stage.

III. EPHEMERIDE

1. LE PROJET 12 POÈMES NUMÉRIQUES

A. LE SPECTACLE

Pour comprendre le fonctionnement de l'application Ephéméride, il est nécessaire de saisir le concept du spectacle pour lequel elle a été conçue.

12 Poèmes numériques, c'est un spectacle immersif qui regroupe sur scène 2 musiciens et 1 artiste numérique. Mettant en lumière douze représentations musicales et numériques mêlant la poésie à l'art numérique.

Le public se retrouve à partager le même espace que les artistes, entourés par deux écrans géants et par 21 enceintes permettant de spatialiser les musiques et les effets sonores. Chaque spectateur tient un smartphone équipé d'une application qui leur permet d'interagir et de participer activement au spectacle.

Le but de ce spectacle est de mêler la sensibilité de la poésie et les petits actes du quotidien avec des outils numériques.

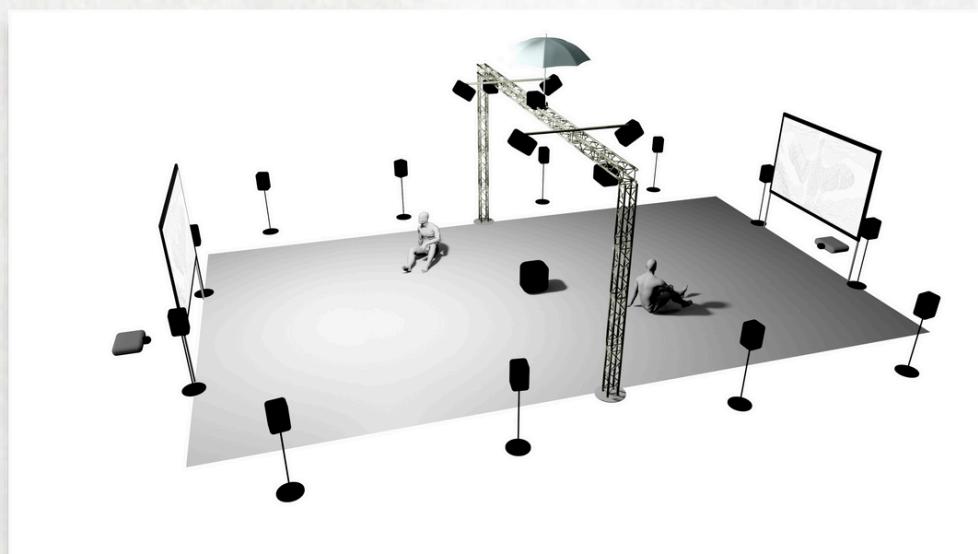


Figure 2 : Scène de 12 Poèmes Numériques

III. EPHEMERIDE

B. LE RÔLE D'ÉPHÉMERIDE AU SEIN DE 12 POÈMES NUMÉRIQUES

Le spectacle **12 Poèmes numériques** est pensé pour pouvoir utiliser une application pendant son déroulé et pouvoir interagir de près ou de loin avec ce qui se passe sur les vidéoprojecteurs, les panneaux d'affichage, sur scène et également de jouer des sons spécifiques. Ainsi, l'application permet aux spectateurs de devenir acteurs du spectacle.

L'application s'intègre au spectacle sans en devenir obligatoire pour autant.



Figure 3 : Les spectateurs participent activement, téléphone en main.

III. EPHEMERIDE

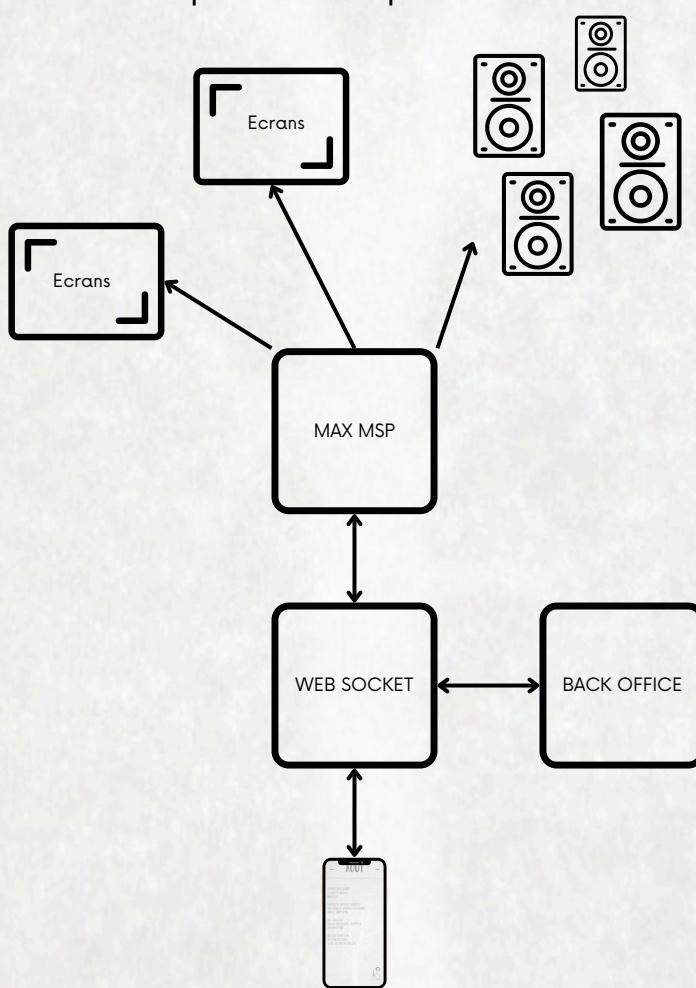
C. ORGANISATION NUMÉRIQUE

Du téléphone des spectateurs aux écrans et enceintes, plusieurs systèmes intermédiaires jouent un rôle important.

Les deux écrans et les 21 enceintes sont connectés à un logiciel, MAX MSP, qui permet de gérer l'image et de synchroniser le son avec les musiciens. MAX MSP est lui-même connecté à un serveur webSocket, lequel gère les interactions entre les différents acteurs : le serveur de base de données, la partie administrateur ainsi que les spectateurs utilisant l'application.

Pour interagir avec les écrans, les spectateurs utilisent l'application qui envoie des événements au serveur webSocket. Ce serveur les enregistre, les traite en conséquence, puis les transmet à MAX MSP, qui s'occupe d'afficher les interactions sur les écrans.

Figure 4 : Organisation du numérique durant le spectacle



III. EPHÉMERIDE

2. LE DÉVELOPPEMENT DE L'APPLICATION

A. ARCHITECTURE DE L'APPLICATION À TRAVERS LE SPECTACLE

1. L'EXPÉRIENCE UTILISATEUR À TRAVERS EPHÉMÉRIDE

Tout comme le spectacle, qui dépeint une année en 12 mois, l'application reflète cette découpe temporelle en présentant une interface faisant référence au calendrier éphéméride. Chaque mois est représenté comme une page détachable d'un calendrier virtuel, offrant ainsi une progression linéaire à travers l'année. Chaque mois contient un poème découpé en plusieurs "pages" selon sa longueur, accompagné d'une page dédiée à l'interaction avec le spectacle.

L'application est entièrement tactile et est pensée pour être utilisée uniquement sur téléphone ou tablette offrant une navigation intuitive. Les utilisateurs peuvent faire défiler horizontalement pour avancer dans l'année ou revenir en arrière, et verticalement pour passer d'un jour à l'autre et ainsi explorer les poèmes. Le défilement vertical se fait de haut en bas pour aller à l'opposé de ce qui se fait sur les réseaux sociaux où le défilement se fait de bas en haut.

Un nombre de pages indique à l'utilisateur à quelle page il se trouve et combien de pages il existe dans le mois en cours. Également, les flèches dans la barre de navigation indiquent à l'utilisateur qu'il peut changer de mois. De même pour les jours, lorsque l'icône de défilement des pages est visible, cela indique à l'utilisateur qu'il peut naviguer à travers les pages.

Le glissement horizontal permet le changement de mois. Le glissement vertical de haut en bas permet le changement de jour.

III. EPHEMERIDE

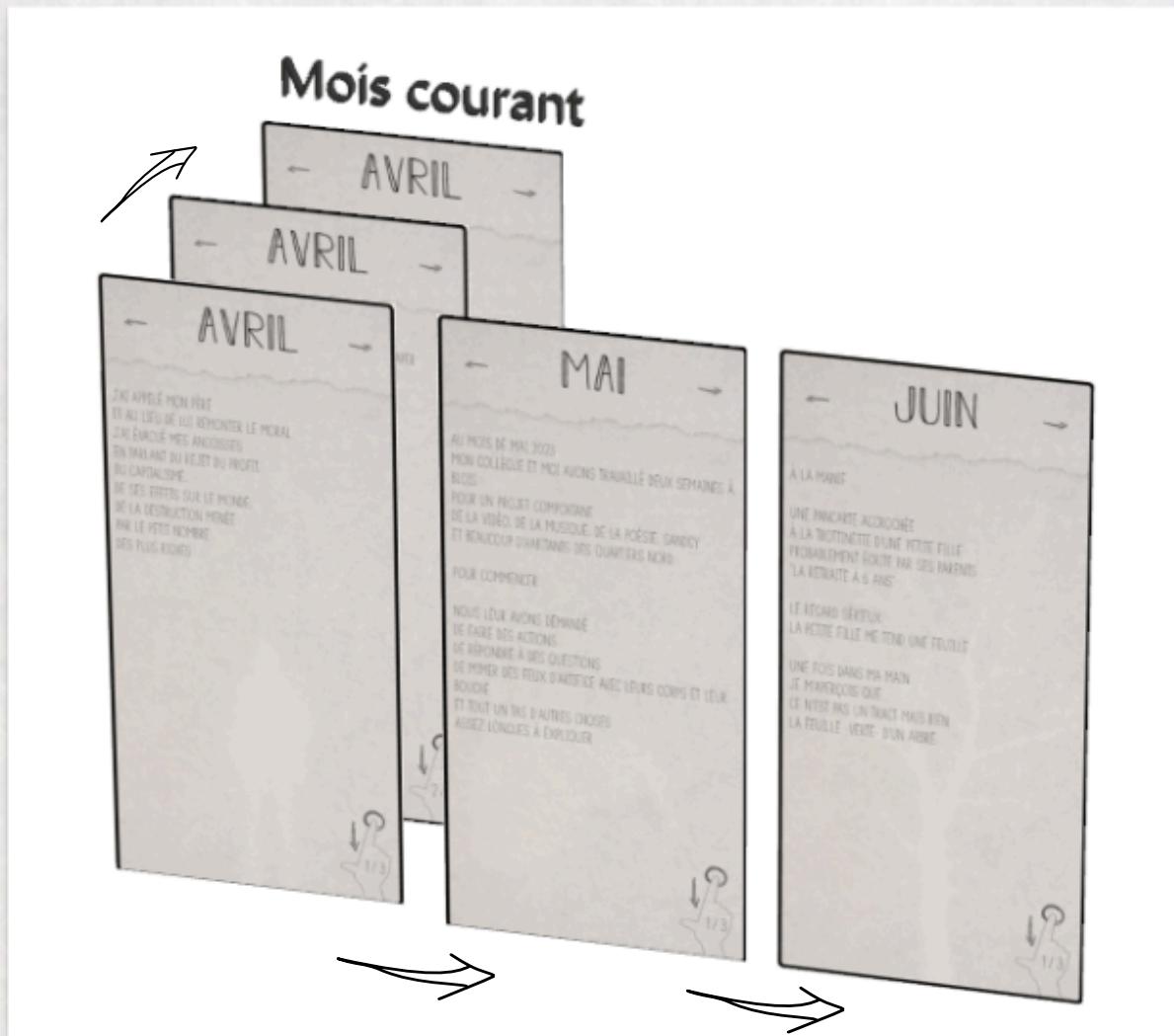


Figure 5 : Le système de navigation de l'application

Avec son design minimaliste et son interface intuitive, Ephéméride est conçu pour être accessible à un public de tout âge. Afin de créer un compagnon autonome pour le spectacle, il a été décidé de permettre à un administrateur de contrôler l'application à distance depuis un back-office. Cette fonctionnalité facilite la guidance des utilisateurs pendant le spectacle et assure une cohérence entre ce qui se passe sur scène, sur les écrans, et au sein de l'application.

III. EPHEMERIDE

2. LE SYSTÈME DE NAVIGATION

L'application est conçue avec un système de composants "mois", au nombre de 13, 12 mois et 1 mois de démo, chacun contenant un nombre de pages "jours" qui dépendent de la taille du poème. Chaque mois est un composant React qui intègre autant de composants "jour" que nécessaires.

Le système de navigation d'Éphéméride utilise une pile de pages superposées, qui défilent en fonction des actions des utilisateurs pour laisser place aux suivantes. Lorsqu'une page descend, elle reçoit une classe spécifique qui déclenche l'animation de descente. Une fois la page suivante visible, l'index de la page précédente est réinitialisé à la valeur minimale, la plaçant ainsi en dernière position dans la file. Ensuite, son nom de classe change, ce qui la fait remonter, évitant ainsi que l'utilisateur ne voie l'animation de remontée. Cela crée un effet d'Éphéméride infini lorsque les jours changent.

Le changement de mois fonctionne de manière similaire. Parmi une liste de mois, le mois courant obtient le Z-index le plus élevé. Lorsqu'un changement de mois se produit, c'est le bloc de pages entier qui descend.

Pour économiser des ressources, seuls les mois visibles par l'utilisateur contiennent toutes leurs pages. Les autres mois ne possèdent que leur première page, et les pages supplémentaires ainsi que la page interactive ne sont montées qu'au moment où l'utilisateur s'arrête sur un mois spécifique.

Sur l'exemple ci-contre, on peut voir que tous les mois sauf le mois courant (le troisième mois) sont à l'index 10.

```
<div class="MonthContainer">
  <div class="Month" style="z-index: 10;">...</div>
  <div class="Month" style="z-index: 10;">...</div>
  <div class="Month" style="z-index: 20;">...</div>
  <div class="Month" style="z-index: 10;">...</div>
  <div class="Month" style="z-index: 10;">...</div>
</div>
```

Figure 6 : Indexation des mois

III. EPHEMERIDE

Ici on peut voir un mois et ses jours qui viennent d'être montés. La première page possède l'index le plus haut et s'affiche alors en premier.

```
▼ <div class="Month" style="z-index: 20;"> == $0
  ▶ <div class="Day Current" style="z-index: 4;"> ...
  ▶ <div class="Day Current" style="z-index: 3;"> ...
  ▶ <div class="Day Current" style="z-index: 2;"> ...
  ▶ <div class="Day Current" style="z-index: 1;"> ...
</div>
```

Figure 7 : Indexation des jours

La première page récupère la classe slide-down qui déclenche l'animation de descente.

```
▼ <div class="Month" style="z-index: 20;"> == $0
  ▶ <div class="Day Current slide-down" style="z-index: 4;">
    ...
  ▶ <div class="Day Current" style="z-index: 3;"> ...
  ▶ <div class="Day Current" style="z-index: 2;"> ...
  ▶ <div class="Day Current" style="z-index: 1;"> ...
</div>
```

Figure 8 : Animation en cours

La première page initiale a fini son animation, elle obtient un Z-index inférieur à la précédente dernière page et passe donc en arrière-plan. Elle perd sa classe slide-down et se remet ainsi en position initiale.

```
▼ <div class="Month" style="z-index: 20;"> == $0
  ▶ <div class="Day Current" style="z-index: 0;"> ...
  ▶ <div class="Day Current" style="z-index: 3;"> ...
  ▶ <div class="Day Current" style="z-index: 2;"> ...
  ▶ <div class="Day Current" style="z-index: 1;"> ...
</div>
```

Figure 9 : Animation terminée

III. EPHEMERIDE

B. COMPOSANT INTERACTIFS

Comme nous l'avons vu précédemment, l'application est découpée en "mois" et en pages qui la constituent. À chaque mois, est associée une page spécifique qui suit le poème, c'est la dernière page de chaque mois. Nous appellerons ces pages des composants.

Les composants sont tous associés d'une certaine façon au poème de leur mois. Ces derniers ont tous un rôle à jouer à travers du spectacle. Et ils ont pour but de faire participer le spectateur en direct de différentes façons. Certains composants vont utiliser une API pour récupérer des informations depuis la base de données ou pour en ajouter de nouvelles. D'autres vont envoyer des données à un serveur websocket sous différentes formes qui s'occupera de les transmettre à MAX MSP pour la gestion de l'affichage.

Pour davantage de clarté, nous allons nous intéresser à 3 composants différents, comprendre la démarche qu'il y a derrière la création de ces derniers et expliquer comment ils ont été développés.

1. MOIS DE MAI: BERLIN

Le composant Berlin fait partie de la performance correspondante au mois de mai. Le poème de ce mois relate une expérience menée en 2023 par la compagnie avec la Maison de Bégon , structure culturelle blésoise, et les habitants des quartiers nord de Blois. Le texte insiste sur l'impossibilité de rendre une expérience complexe, nourrie de multiples rencontres. Au lieu de faire une sélection dans l'imposante base de données de vidéos et d'interview des habitants, la compagnie a décidé de confier aux spectateurs le choix des personnages qui s'animent à l'écran au rythme de la musique.

III. EPHEMERIDE

Le développement de Berlin

La base de données contient une table "character" où sont stockées les photos des personnes ayant participé à cette action culturelle. Cette table est liée à une table "action", qui définit les noms des actions et une représentation dessinée de ces actions. Chaque personne est ainsi associée à plusieurs actions par une relation de type One-to-Many entre ces tables.

L'application effectue un appel API pour récupérer les photos des personnes stockées dans la base de données, permettant aux utilisateurs de choisir le personnage qu'ils souhaitent voir animé à l'écran.



En fonction du personnage sélectionné, une liste d'actions disponibles s'affiche. Lorsqu'une touche d'action est pressée, un message est envoyé au serveur socket, qui transmet cette information à MAX MSP avec le nom de la personne et l'action à réaliser. Une position aléatoire est ensuite déterminée sur l'écran, et le personnage exécute l'action tant que l'utilisateur maintient son doigt sur le bouton.

Figure 10 : Mois de mai, choix du personnage

III. EPHEMERIDE

2. MOIS DE JUILLET : PAPILLON

Le composant Papillon fait partie du mois de juillet. Le but de ce mois est de flirter avec le politiquement correct, de se plonger dans quelque chose d'absurde, de se démarquer du ton du spectacle et de se demander quel est le rapport entre le beau et le laid, qui en détient la définition ?

Ce composant permet aux spectateurs de prendre le contrôle d'un papillon sur les écrans du spectacle, en le faisant bouger sur les axes X et Y. L'application se transforme ainsi en contrôleur et permet à l'utilisateur d'identifier le papillon qu'il peut contrôler en affichant sur l'application le design de ce dernier.



Figure 11 : L'application se transforme en contrôleur

III. EPHEMERIDE

Le développement de Papillon :

Un joueur appuie sur "jouer" et est alors enregistré sur le serveur. Un tableau enregistre chaque identifiant unique des joueurs pour savoir s'ils sont en jeu. Le serveur envoie ensuite un identifiant de papillon au joueur pour lui permettre de visualiser le papillon qu'il contrôle.

En parallèle, le serveur envoie à Max MSP, (logiciel qui permet l'affichage des papillons), le tableau des utilisateurs mis à jour et lui indique de créer un papillon pour ce nouvel utilisateur. Max MSP récupère l'identifiant du papillon et l'identifiant du joueur.

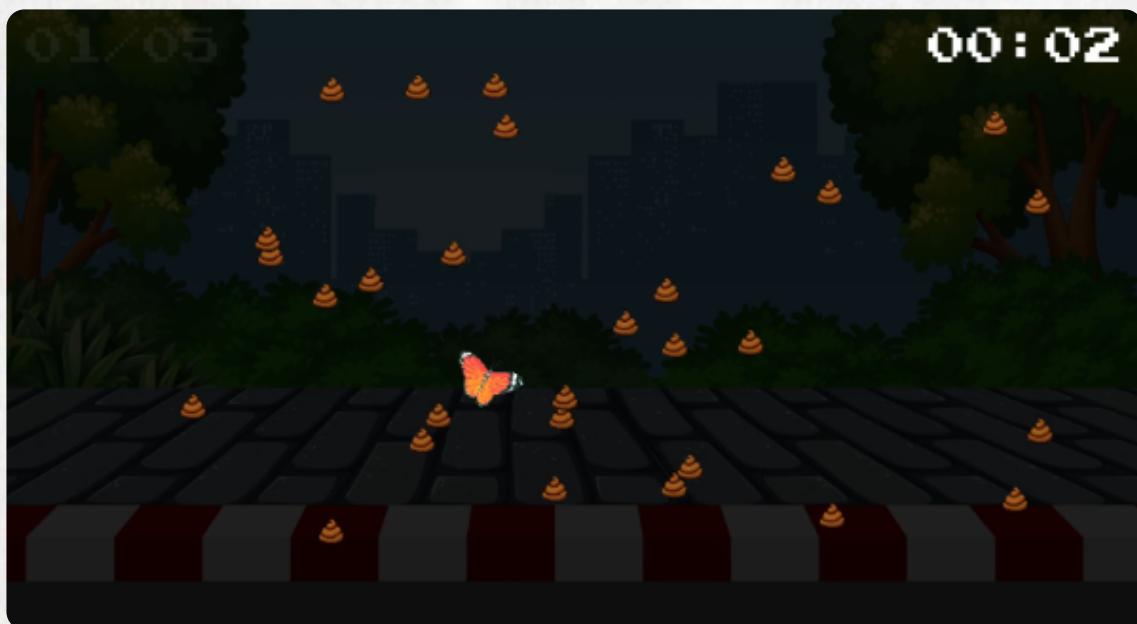


Figure 12 : Le jeu du papillon sur les écrans



III. EPHEMERIDE



Une fois cela fait, l'utilisateur peut faire bouger son papillon. Il indique au serveur la direction qu'il prend et son identifiant. Le serveur vérifie si le joueur est bien dans le tableau ; s'il y est, il envoie à Max MSP l'identifiant et la direction. Max MSP traite ensuite ces informations pour afficher et animer le papillon.

Lors de la déconnexion de l'utilisateur, son papillon est retiré de l'écran et du tableau des joueurs dans le serveur webSocket. L'utilisateur peut alors se reconnecter avec un nouveau papillon. Si un joueur subit une déconnexion inattendue, il est possible que son papillon ne soit pas retiré. À sa prochaine connexion, le serveur vérifiera l'existence de ce joueur dans le tableau, retire l'ancien papillon, et permettra au joueur de jouer à nouveau comme un tout nouveau joueur.

3. MOIS DE SEPTEMBRE : RENTRÉE

Le composant Rentrée fait partie du mois de septembre, représentant la période de la rentrée, où les gens sont souvent submergés par les activités liées à la reprise des cours ou à un nouvel emploi. L'objectif est d'immerger les utilisateurs dans cette atmosphère dynamique et intense, évoquant la frénésie de la rentrée et donc du mois de septembre.

Ainsi après un moment de calme, le but est de faire sonner et vibrer les téléphones des utilisateurs simulant ainsi des messages reçus ou un appel en cours. Dans un premier temps, de faire sonner une personne du public puis un groupe plus important jusqu'à ce que chaque téléphone du public sonne dans un brouhaha numérique.

III. EPHEMERIDE

Le développement de Rentrée

Des fichiers son .mp3 sont stockés dans la base de données, et leurs URL sont accessibles via l'API du back-office. Lors du montage du composant Rentrée, des sons sont tirés aléatoirement de cette base de données par l'application, de sorte que chaque utilisateur reçoit des sons différents. L'administrateur dispose de boutons permettant de faire sonner le téléphone d'une personne, de plusieurs personnes selon un pourcentage, ou de tous les utilisateurs. Les identifiants des utilisateurs sont tirés aléatoirement par le serveur, et l'événement déclenchant le son sur leur téléphone leur est transmis.

Pour des raisons de sécurité et d'expérience utilisateur, les navigateurs n'autorisent pas les sites web à produire du son tant que l'utilisateur n'a pas interagi avec eux (par exemple, en cliquant sur quelque chose). Aucun son ne sera produit tant que l'utilisateur n'aura pas cliqué, tapé ou déclenché une action.

C'est grâce au hook React useSound que les sons sont joués, mis en pause ou arrêtés. Il simplifie l'intégration et le contrôle des effets sonores dans les applications, permettant ainsi de gérer efficacement la lecture des sons.

```
const audioUrl1 = `${URL}/strapi${notifications[0].url}`;
const [play1, { stop : stop1 }] = useSound(audioUrl1);
```

Figure 13 : Déclaration du hook useSound

III. EPHEMERIDE

C. CONTRÔLE DISTANT

Un des principaux défis de l'application a été de permettre à chaque utilisateur de suivre le spectacle en direct sans se perdre dans l'interface. Pour ce faire, il était essentiel de guider les spectateurs de manière efficace. L'objectif est donc de pouvoir contrôler leur application à distance tout au long du spectacle.

Pour répondre à ce besoin, un panneau d'administration a été créé, permettant de gérer l'état de l'application en temps réel. Ce panel permet la gestion de l'application de façon générale. Il met à disposition les outils suivants pour l'administrateur :

- Choix du mois courant : Permet de diriger chaque utilisateur vers le mois sélectionné.
- Blocage des mois et des jours : Empêche les utilisateurs de naviguer vers un autre mois et/ou jour.
- Affichage unique : Permet d'afficher uniquement les poèmes, uniquement les composants, ou les deux.
- Blocage complet de l'application : Désactive toutes les interactions utilisateur.
- Contrôle de la luminosité : Ajuste la luminosité de l'application.
- Texte sur les panneaux lumineux : Affiche un texte personnalisé sur les panneaux lumineux présent dans la salle de spectacle.
- Retour visuel : Fournit un retour visuel en temps réel pour suivre l'état de l'application.

III. EPHEMERIDE

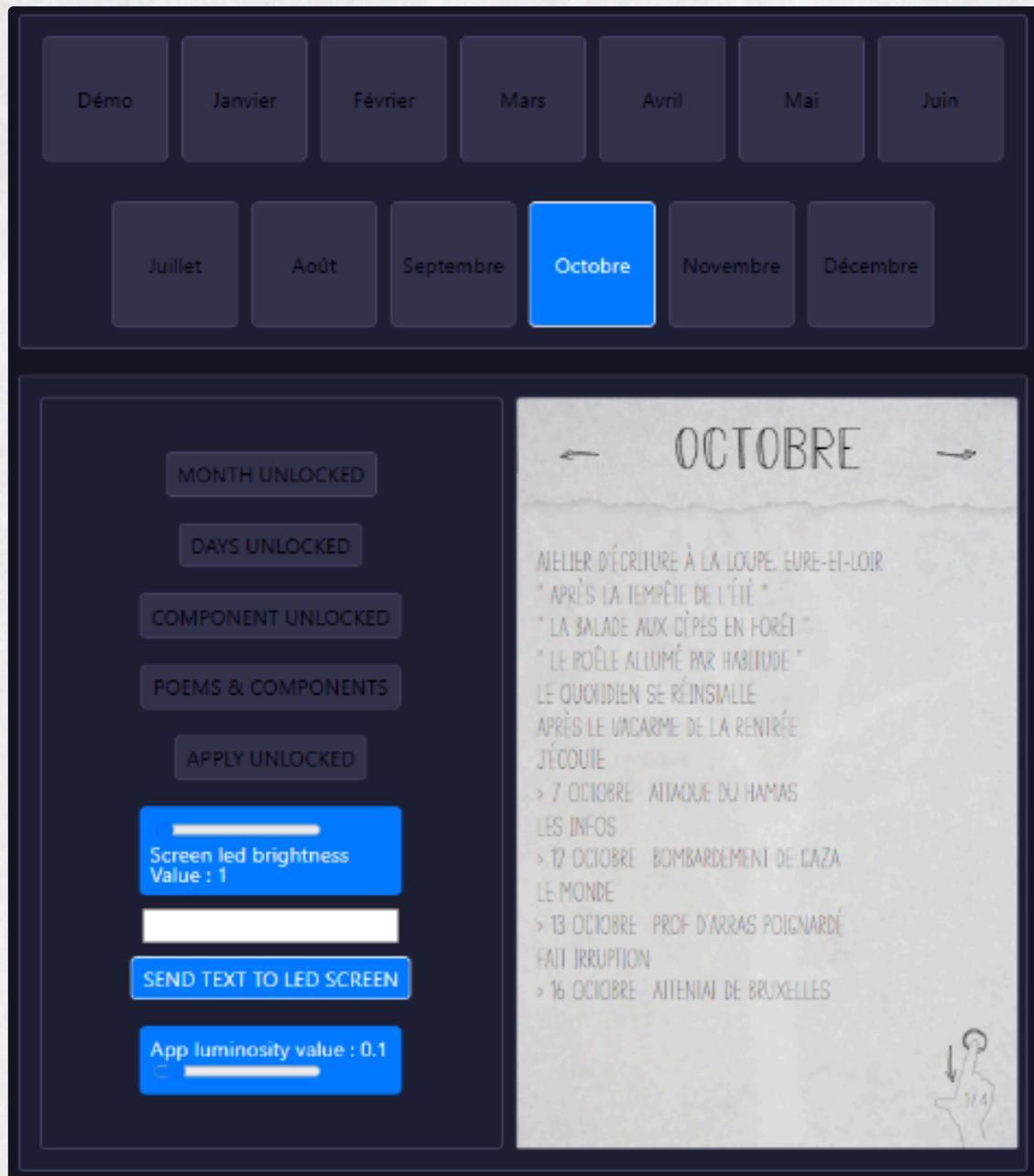


Figure 14 : Panel d'administration général

Pour chacun des mois, un autre panel existe également permettant à l'administrateur d'avoir à sa disposition la possibilité de créer des événements sur le serveur socket de façon spécifique pour chacun des mois.

III. EPHEMERIDE

3. LE SERVEUR WEBSOCKET

A. LE WEBSOCKET, LE MOTEUR DE L'APPLICATION

1. L'UTILISATION DU SERVEUR WEBSOCKET

Le serveur webSocket est la pièce maîtresse de l'application. C'est à travers ce serveur que presque toutes les informations vont transiter. C'est la pierre angulaire de l'application Ephéméride. L'administrateur, à travers le back office, MAX MSP et les utilisateurs, tous se connectent par défaut au serveur websocket et communiquent grâce à lui.

Le serveur WebSocket a été configuré pour accepter les connexions entrantes depuis l'extérieur, permettant ainsi la communication des utilisateurs sans qu'ils n'aient besoin d'être sur le même réseau.

Pour remédier à de potentiels problèmes de port bloqué et d'accès dans les salles de spectacle, il existe aussi une version de l'application et du serveur socket complètement local.

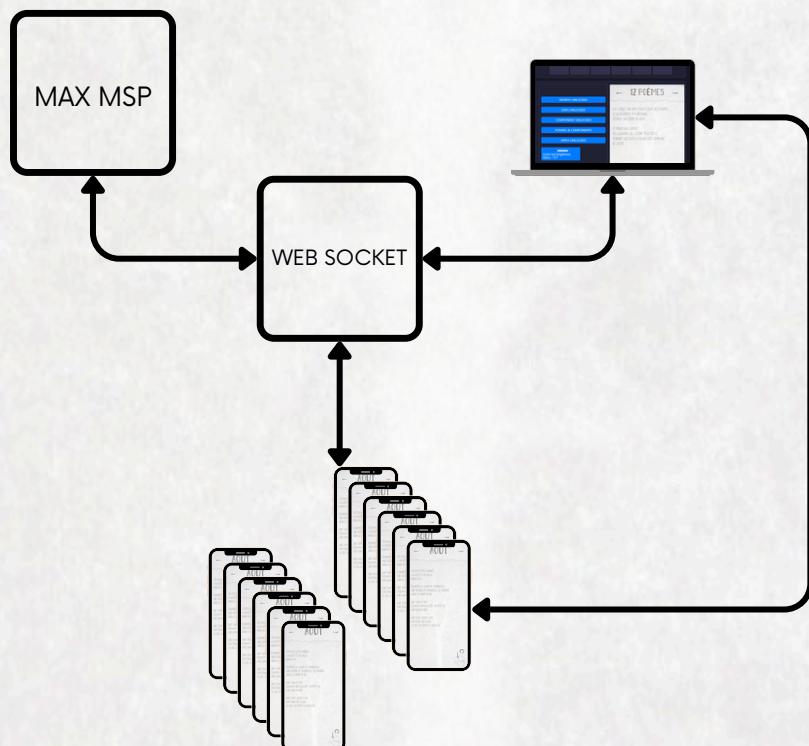


Figure 15 : Les interactions avec le serveur web socket

III. EPHEMERIDE

2. GESTION DES ÉVÈNEMENTS

Tous les événements passent naturellement par le serveur socket. Qu'il s'agisse de la connexion ou de la déconnexion d'un utilisateur, chaque interaction avec le spectacle est gérée par le serveur socket.

Par exemple, lorsqu'un utilisateur quitte l'application, l'événement "disconnect" se produit par défaut. Il est alors nécessaire d'identifier quel utilisateur s'est déconnecté et de le retirer des tableaux correspondants.

```
this.socket.on('disconnect', () =>{
    console.log("disconnect : ", this.socketId);
    this.state.RemoveAdmin(this.socketId);
    this.state.RemoveMaxMsp(this.socketId);
    this.state.RemoveLedScreen(this.socketId);
    this.state.RemoveUser(this.socketId);
    this.state.RemovePlayers(this.identifier);
    this.state.RemoveIdentifier(this.identifier); //
    this.socket.to(this.state.GetMaxMspId()).emit('playerLeft',{
        msg: 'playerLeft',
        month: 'Juillet',
        userIdentifier: this.identifier});
    this.socket.disconnect(true);
    this.socket.to(this.state.GetAdminId()).emit('state',this.state);
});
```

Figure 16: Événement "disconnect"

La transmission des événements se fait par diffusion générale (broadcast) lorsqu'il est nécessaire de transmettre une information à tous, comme le changement général de mois ou le blocage de l'application. Pour des communications plus ciblées, les identifiants socket des différents acteurs sont utilisés.

```
this.socket.broadcast.emit('lockDay', value);

this.socket.to(this.state.GetAdminId()).emit('state', this.state);
```

Figure 17: Emission en broadcast ou unicast

III. EPHEMERIDE

B. SYSTÈME D'UTILISATEUR ET D'ÉTAT

1. GESTION DES DONNÉES DES UTILISATEURS

Pour garantir le bon fonctionnement de l'application, il est impératif d'identifier chaque utilisateur connecté en temps réel et de traiter les événements en conséquence. Ce besoin se traduit par plusieurs défis. La gestion des connexions en temps réel pour identifier chaque utilisateur qui se connecte via WebSocket, le stockage et la mise à jour de l'état des utilisateurs pour conserver et gérer les informations de chaque utilisateur connecté, et la reconnaissance des utilisateurs récurrents pour identifier si un appareil a déjà été connecté auparavant grâce à un identifiant unique stocké en local storage.

Pour résoudre ce problème, nous avons adopté la stratégie suivante : lorsqu'un utilisateur se connecte au WebSocket, un objet User est instancié avec trois paramètres, à savoir l'objet socket qui contient un identifiant spécifique à l'utilisateur, son adresse IP, et un objet state qui permet de suivre l'état du spectacle pour tous les utilisateurs.

```
constructor(socket,ip,state){  
    this.socket = socket;  
    this.socketId = socket.id;  
    this.ip = ip;  
    this.role = "USER";  
    this.state = state;  
    this.identifier;  
  
    this.state.AddUser(this.socketId);  
  
    socket.emit('hello', {  
        hello: 'Bienvenue sur le WEBSOCKET PROD',  
        time: Date.now() / 1000,  
        socketId: this.socketId  
    });  
  
    console.log("User connected : ", this.socketId);  
  
    this.socket.emit('state',state);  
    if(this.state.GetAdminId()){  
        this.socket.to(this.state.GetAdminId()).emit('state',state)  
    }  
}
```

25/38

Figure 18: Constructeur de la classe User

III. EPHEMERIDE

L'objet User nouvellement instancié est ensuite ajouté à la liste des utilisateurs connectés dans l'objet state. À chaque nouvel événement, les informations de l'utilisateur, telles que le rôle, l'adresse IP, l'identifiant, et l'identifiant socket, sont conservées en mémoire pour une utilisation ultérieure.

Lors de l'arrivée d'un utilisateur sur l'application, celui-ci reçoit un identifiant unique, une chaîne de caractères stockée en local storage, permettant à l'application de reconnaître si l'appareil a déjà été connecté ou s'il s'agit d'une première visite.

```
io.on('connection', (socket) => {
  let ip = socket.handshake.headers['x-forwarded-for'] || socket.handshake.address;

  const user = new User(socket, ip, state);
  user.addListener();
});
```

Figure 19: Instanciation de la classe User

2. GESTION DE L'ÉTAT DE L'APPLICATION EN COURS

Comme expliqué précédemment, le serveur websocket a besoin de connaître l'état du spectacle en cours afin de le transmettre aux utilisateurs qui se connecte et qui arrive en cours de route. Le défi principal est de s'assurer que ces informations ne soient pas stockées sur les appareils des utilisateurs ni sur le poste d'administration, mais sur un serveur centralisé qui reste actif pendant toute la durée du spectacle.

III. EPHEMERIDE

À son lancement, le serveur WebSocket instancie un objet “State” qui permet de stocker toutes les données relatives à l'état actuel du spectacle, telles que le nombre d'utilisateurs connectés, le mois courant, l'affichage ou non des composants ou des poèmes, la luminosité, etc... Ces informations sont mises à jour et stockées dans la classe State à chaque nouvel événement via des setters. De cette façon, les informations essentielles sont centralisées et gérées en temps réel par le serveur.

De plus, lorsque un utilisateur se connecte à l'application en cours de spectacle, il peut reprendre le fil là où en sont les autres spectateurs. L'application de l'utilisateur se met automatiquement à jour à chaque nouvelle connexion, en récupérant l'état actuel du spectacle depuis l'objet State sur le serveur WebSocket, assurant ainsi une expérience continue et synchronisée pour tous les participants.

```
constructor(){
    this.monthLocked = false;
    this.dayLocked = false;
    this.componentLocked = false;
    this.currentMonth = 0;
    this.luminosity = 0;
    this.users = [];
    this.identifiers = [];
    this.players = [];
    this.userId = [];
    this.sounds = [];
    this.adminId;
    this.maxmspId;
    this.ledScreenId;
    this.maisonId = ["","",""];
    this.butterflies = [0,1,2,3,4,5,6,7];
    this.trainStep = 0;
    this.ledText="";
    this.ledBrightness=127;
    this.maisonLed=[0,0,0];
    this.maisonMotor=[0,0,0];
    this.applyBlocked = false;
    this.displayMode = 0;
}
```

Figure 20: Constructeur de la classe State

III. EPHEMERIDE

4. LA MISE EN PRODUCTION DE L'APPLICATION

A. AUTOMATISATION

La mise en production de l'application est une compétence que j'ai acquise lors de ce stage. À l'approche de la fin du stage, il était nécessaire de simplifier ce processus pour éviter de devoir copier manuellement les différentes versions des builds entre les environnements de développement et de production.

Il faut savoir qu'il existe une version de développement et une version de production, que ce soit pour le back office, la partie destinée aux utilisateurs ou encore le serveur WebSocket. Cette coexistence permet de développer de nouvelles fonctionnalités sans risquer de compromettre le fonctionnement de l'application en production.

Grâce à `window.location.href`, on est en mesure de connaître l'URL de l'application et donc de permettre à la build de savoir si elle est supposée être en production ou en développement. Selon l'URL de la build, c'est-à-dire l'endroit où les fichiers sont situés, l'application utilise le serveur WebSocket et le back-office du côté développement ou production. Ainsi, il n'est plus nécessaire d'effectuer une build distincte pour chaque environnement.

```
if(window.location.href.includes("app.petitenature.fr")) URL='https://sio.petitenature.fr:443';
```

Figure 21 : `window.location.href`

III. EPHEMERIDE

B. SERVEUR APACHE ET SÉCURITÉ

La mise en production implique également la configuration du serveur Apache et la définition des règles pour le nom de domaine. Dans ce cas, "app.petitenature.fr" est le nom de domaine utilisé pour les applications mises en production.

- Définir le chemin des fichiers de l'application :
 - Le chemin relatif vers les fichiers des différentes applications doit être défini dans le serveur Apache, par exemple, "~/sync-fr.petitenature.app/www/".
- Configuration du reverse proxy :
 - Les requêtes vers "app.petitenature.fr/strapi" doivent être redirigées vers "localhost:1338". Cela implique la configuration d'un reverse proxy pour gérer cette redirection.
- Certificats SSL :
 - Les certificats générés par Certbot sont nécessaires pour sécuriser les communications en HTTPS, ce qui est particulièrement important pour le serveur websocket.

```
<VirtualHost *:443>
    ServerName app.petitenature.fr
    DocumentRoot "/home/ale/sync-fr.petitenature.app/www"

    <Directory /home/ale/sync-fr.petitenature.app/www>
        Options -Indexes +FollowSymLinks +MultiViews
        AllowOverride All
        Require all granted
    </Directory>

    # EPH & STRAPI RULES
    RewriteEngine On
    RewriteRule ^/EPH(/.*)?$ /app/EPH$1 [L]

    Redirect permanent "/strapi" "/strapi/"
    ProxyPass /strapi !
    <Location /strapi/>
        ProxyPass http://127.0.0.1:1338/
        ProxyPassReverse http://127.0.0.1:1338/
        Require all granted
    </Location>
    #EPH & STRAPI RULES

    Include /etc/letsencrypt/options-ssl-apache.conf
    SSLCertificateFile /etc/letsencrypt/live/app.petitenature.fr/fullchain.pem
    SSLCertificateKeyFile /etc/letsencrypt/live/app.petitenature.fr/privkey.pem
</VirtualHost>
```

29/38

Figure 22 : Virtualhost du domaine de production

IV. STRAPI & MAXMSP, LA GESTION ADMINISTRATEUR

1. STRAPI, LE CHOIX DU BACK OFFICE

A. UNE BASE DE DONNÉE NO CODE

Strapi est un système de gestion de contenu open-source, basé sur React, qui permet de créer facilement des API REST ou GraphQL pour gérer les données d'une application. Avec Strapi, il est possible de définir des types de contenu (collection type) pour structurer les données, et de gérer la base de données sans écrire de code grâce à une interface relativement intuitive. Strapi utilise une base de données MySQL.

B. UTILISATION DE L'API ET DE LA MEDIA LIBRAIRIE

L'API créée avec Strapi permet de fournir les données nécessaires à l'application Ephéméride. Cette API est utilisée pour récupérer les poèmes, gérer les interactions avec le spectacle, et fournir d'autres contenus dynamiques pour alimenter l'application côté client. Elle offre une interface standardisée pour interagir efficacement avec la base de données.

L'accès à l'API est sécurisé par des tokens, garantissant ainsi la confidentialité et la sécurité des données.

L'utilisation de Postman pour effectuer les premiers tests de requêtes API a représenté un gain de temps considérable, permettant une validation rapide des fonctionnalités.

Le back-office Strapi met également à disposition une "media library" permettant de stocker des fichiers sous différentes formes et de les récupérer via un endpoint spécifique à cette bibliothèque de fichiers. Grâce à ce endpoint, on peut obtenir les URL de ces fichiers, facilitant ainsi leur utilisation au sein de l'application. C'est de cette façon que sont récupérés les sons et les images utilisés sur Ephéméride.

IV. STRAPI & MAXMSP, LA GESTION ADMINISTRATEUR

2. MAX MSP

Max MSP est un environnement de programmation visuelle, principalement utilisé pour la création de musique et de médias interactifs. Il permet notamment la manipulation et la synthèse sonore et le traitement audio en temps réel.

Pour ce qui est de l'image, Il est utilisé pour créer des effets visuels, des installations interactives et des performances audiovisuelles. Les artistes peuvent créer des animations et des graphiques qui réagissent à la musique, aux données des capteurs ou à d'autres types d'entrées comme des données provenant du serveur socket.

En plus du traitement du son et de l'image, il est possible d'écrire nos propres lignes de code à travers Max MSP.

Max MSP est une plateforme extrêmement versatile qui permet de créer des œuvres multimédias complexes en combinant l'audio, vidéo, images et code.

C'est donc à travers cette plateforme que la plupart des données de l'application sont transmises puis retranscrites sur les écrans. Je n'ai pas eu à utiliser Max MSP durant le stage, mais il a fallu que je comprenne comment il fonctionnait pour savoir comment communiquer avec cet environnement.

C'est également un relais du panel d'administration servant à gérer l'application à distance. Tous les outils de gestion sur le back-office ont leur équivalent sur MAX MSP.

V. SOLUTIONS ENVISAGEES

1. LE CHOIX DU BACK OFFICE

A. BACK OFFICE EBO

Une première solution envisagée dans la création d'un back-office dédié à l'application était de réutiliser celui déjà présent au sein de la compagnie. L'ancien développeur **M. Jonathan Ocheg** ayant utilisé son propre back-office pour toutes les solutions de la compagnie.

La problématique fut la difficulté de prendre en main l'outil et la complexité du code. Un back-office qui a été créé il y a 20 ans et qui était à l'origine un CMS mis à disposition pour des étudiants. Le choix a été de laisser tomber cette solution qui ne présentait pas de documentation, et qui s'avérait très complexe à étudier.

Il a donc été choisi d'utiliser une solution existante, facile à prendre en main et en indépendante de la personne qui l'utilise, afin qu'à mon départ, une personne puisse reprendre en main aisément.

B. BACK OFFICE DIRECTUS

La première solution NO CODE envisagée a été Directus DB, une plateforme open source de gestion de bases de données et de contenu, souvent utilisée comme un CMS. Elle permet aux développeurs et aux utilisateurs de gérer des contenus de manière flexible et structurée grâce à une interface web et également de créer des plugins spécifiques à nos besoins.

Directus offre des outils pour utiliser son propre serveur WebSocket. Cependant, nous souhaitons utiliser notre back-office comme client socket plutôt que comme serveur. Cela nous a conduits à un problème de sécurité : la communication via WebSocket n'était pas autorisée sur un serveur distant. C'est principalement pour cette raison que Directus n'a pas été retenu pour faire office du futur back-office de la compagnie.

V. SOLUTIONS ENVISAGEES

2. LES VERSIONS BÊTA DES FONCTIONNALITÉS

A. LES FONCTIONNALITÉS IMPLÉMENTÉES

Avant le début du développement de l'application, il m'a été demandé de découvrir et de prendre en main le fonctionnement d'un serveur socket et de mettre en place des tests sur différents éléments afin d'évaluer la faisabilité de certaines fonctionnalités que nous voulions implémenter.

Dans un premier temps, il s'agissait de réussir à faire communiquer des chaînes de caractères entre des téléphones distants, de jouer des sons d'un téléphone à l'autre, et de comprendre comment les appareils communiquent entre eux ainsi que les barrières potentielles.

Ces tests ont notamment mis en évidence la nécessité d'utiliser le protocole HTTPS pour les fonctionnalités requérant l'accès à la sonnerie du téléphone ou à la caméra. La fonction GetUserMedia(), qui gère ces opérations, ne peut pas être utilisée en dehors d'une connexion sécurisée.

J'ai également pu développer une première version du jeu des papillons, avec une commande distante depuis un téléphone. En utilisant une balise canvas et des carrés faisant office de papillons, j'ai pu démontrer le concept de contrôle à distance via WebSocket.

V. SOLUTIONS ENVISAGEES

B. LES SOLUTIONS EN DEVELOPPEMENT

Un point important restant à développer pour l'application est la localisation des téléphones dans l'espace. Il s'agit de réussir à déterminer la position des utilisateurs dans la salle de spectacle. Plusieurs pistes ont été évoquées et certaines étudiées, mais aucune solution n'a été concluante pour le moment.

Le mois d'avril inclut des fonctionnalités encore en développement. L'objectif est de permettre aux utilisateurs d'enregistrer des messages vocaux, qui seront ensuite stockés sur notre serveur pour être réutilisés plus tard dans le spectacle. Actuellement, l'enregistrement, l'écoute et l'envoi des messages vocaux sont fonctionnels, mais le stockage dans la base de données et la réutilisation de ces messages n'ont pas encore été finalisés.

VI. DIFFICULTES RENCONTREES

1. LE COUPLE REACT/SOCKET

L'une des principales difficultés tout au long du stage a été de faire corréler React et le serveur web socket. Les différentes problématiques ont été de faire face au comportement asynchrone de certains événements au sein de l'application tels que la récupération des poèmes depuis leur fichier et le remontage constant de certains composants react dû à leur hook useState.

Le fait de mettre des fonctions qui traitent les événements sockets au sein de composant react peut parfois amener à des boucles d'événements et à des rendus à répétition

De ce fait, à de nombreuses reprises, certaines variables se retrouvaient à posséder leur valeur initiale le temps du montage du composant, faussant complètement la façon dont a été pensé au sein de l'application.

La solution à ce problème a été d'utiliser des hooks UseRef à la place de UseState. Le principal avantage des hooks UseRef est le fait qu'ils soient persistants entre les différents rendus, ne voyant ainsi pas leur valeur modifiée.

De ce fait, j'ai appris certains comportements que je n'aurais pas appris en cours, des subtilités de fonctionnement qu'on ne rencontre que lorsque nous avons un besoin très précis.

VI. DIFFICULTES RENCONTREES

2. L'HUMAIN, À TRAVERS LES ATELIERS ET L'APPLICATION

La gestion des foules n'est pas mon point fort, et devoir encadrer des personnes, y compris des élèves parfois dissipés, a souvent été éprouvant sur le plan émotionnel. Pour autant, je suis très satisfait de mon travail à ce niveau, que ce soit avec les classes que j'ai assisté durant l'exposition ainsi que durant les ateliers.

Egalement, un temps conséquent a été consacré au cloisonnement de l'application pour éviter les comportements non désirés des utilisateurs en se mettant le plus possible à la place des utilisateurs

3. MANQUE DE PRATIQUE SUR LA GESTION DE SERVEUR

Une des lacunes que j'ai rapidement identifiées concernait la gestion du serveur Apache, en particulier les redirections d'URL et les configurations nécessaires pour permettre la cohabitation de plusieurs applications et domaines sur un unique serveur. N'ayant jamais utilisé Apache pour de tels besoins, notamment en ce qui concerne les configurations de redirection et de virtualhost, j'ai pu éprouver des difficultés dans un premier temps.

Ces lacunes se sont tout de même résorbées au fil du temps, notamment lorsque je me suis rendu compte de l'importance que cela avait dans le développement et dans la mise en service d'une application web. J'ai donc eu de nombreux moments où j'ai pu expérimenté et faire des essais qui m'ont permis d'appréhender la chose avec plus de confiance.

VII. BILAN DU STAGE

OÙ ALLONS NOUS ?

La confiance qui m'a été accordée est ce qui m'a le plus marqué durant ces 15 semaines de stage. J'ai pu être force de proposition sur de nombreux aspects, et ce, à toutes les phases de développement. On m'a traité comme un employé parmi d'autres, et non comme un simple stagiaire. Cette confiance m'a donné le sentiment d'être important, voire très important, dans la création de ce projet. Ressentir ces responsabilités m'a été extrêmement gratifiant.

Cette première réelle expérience dans le milieu professionnel du développement web m'a permis de développer ma vision globale quant au développement d'une application. J'ai pu comprendre les tenants et aboutissants nécessaires à la création de système complexe et je suis heureux d'avoir pu travailler sur une application qui diffère largement de ce que j'ai pu développer précédemment.

Je suis également très reconnaissant d'avoir pu suivre la compagnie dans ces tournées, ces ateliers et ses résidences. Le fait d'avoir pu aider la compagnie dans ses actions et dans différents domaines m'a donné confiance en moi et dans ma capacité à sortir de ma zone de confort, particulièrement dans la gestion des relations humaines que j'ai pu avoir avec les différentes classes que j'ai aiguillé.

VII. BILAN DU STAGE

Ce stage, c'est pour moi la confirmation de deux choses :

C'est la volonté de continuer à m'améliorer, à apprendre dans le domaine du développement web et l'envie de devenir expert sur des technologies précises. C'est donc de poursuivre mes études vers un master qui me permettra de me spécialiser encore davantage.

Mais c'est aussi la confirmation que j'ai besoin de travailler sur des projets concrets, que je peux développer avec passion et que je vais pouvoir créer des applications ou des sites web qui serviront réellement après mon passage. J'ai donc cette envie de travailler rapidement.

Ainsi, pour la suite, je souhaite intégrer une alternance. Cela me permettrait de continuer à apprendre à l'école tout en mettant mes compétences au service d'une entreprise, avec l'envie de voir naître des projets ambitieux.