

Dossier de projet Concepteur Développeur d'application

Delmas Leo

Décembre 2022 – Septembre 2023

Table des matières

Compétences couvertes par le dossier professionnel :.....	4
Abstract :.....	5
Introduction.....	6
A/ Cahier des charges : Conception d'un site vitrine.....	7
I - Descriptif du projet.....	7
II - Equipe du projet.....	8
III - Contexte du projet.....	9
IV - Choix techniques.....	17
V - Planification.....	22
VI - Conception.....	23
VII - Charte graphique et maquettage.....	29
VIII - Analyse des risques.....	33
B/ développement du site de la Monkey Compagnie.....	34
I – Situation de recherche.....	34
II - Création d'un environnement de développement collaboratif.....	34
III - Développement frontend.....	36
IV - Mise en place d'une base de données.....	41
V - Mise en place et utilisation d'inertia.js.....	48
VI - Développement Back-end.....	51
VII - Présentation d'une fonctionnalité.....	57
VIII - Testing.....	67
IX - Mise en production.....	69
Conclusion :.....	71
Annexes :.....	72
1 - Analyse de la concurrence.....	72
2 - Code SQL de la création de la base de données :.....	73
3 – Diagramme de Gantt.....	76
4 – Diagramme de PERT et note de cadrage.....	78
5 – Première version du mockup du site.....	80
6- maquettage des différentes pages du site.....	82
7 - CGU et RGP.....	84
8 - Matrice de risques.....	92

9 – Directives Vue.js.....	94
10 – Layouts et header inertia.....	97
11 – Liste des middlewares définis dans le fichier Kernel.php.....	99
12 – Routes Admin du projet :.....	100
13 – Création d'un formulaire de contact.....	101
14 – ShareErrorsFromSession.....	103
15 – Méthodes update et destroy.....	104
16 – Template du composant Quizz.....	106
17 – Tests de quizz et de connexion.....	107

Compétences couvertes par le dossier professionnel :

Compétences couvertes par ce mémoire	Dossier pro	Mémoire
Maquetter une application		
Développer une interface utilisateur de type desktop		
Développer des composants d'accès aux données		
Développer la partie front-end d'une interface utilisateur web		
Développer la partie back-end d'une interface utilisateur web		
Concevoir une base de données		
Mettre en place une base de données		
Développer des composants dans le langage d'une base de données		
Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement		
Concevoir une application		
Développer des composants métier		
Construire une application organisée en couches		
Développer une application mobile		
Préparer et exécuter les plans de tests d'une application		
Préparer et exécuter le déploiement d'une application		

Toutes les compétences seront couvertes par ce mémoire, mais un dossier professionnel sera ajouté à ce mémoire pour l'agrémenter d'une situation de développement n'étant pas présente dans ce mémoire.

Abstract :

The work presented here is about my internship at an event agency, for which I was mandated to create a complete showcase website. The internship happened between June 19th and September 8th. I was asked to design and develop a website in order to help my client grow on the event market through a better online presence, with a website that could give an insight into the activities of the company and display the upcoming events created and animated by my client.

Through these two months, I designed the different aspects of the website, from graphics and functional design to the creation of the several views on which are now displayed the team of the company, the events of the month, and several albums used to picture the evenings animated by my client.

I was helped in that task by a student in graphic design who wanted to train her skills. She helped me think about and create a design for the website and was leading in all that was about the visual of the project.

I did my best to adapt her choices to produce a functional website, using vue.js as a frontend Framework, paired to Laravel as a backend choice. I chose to make them both work even better together, using the JS library « inertia.js », aiming to make an easy-to-update, maintain and browse website. I still worked on several design aspects, such as the work behind the responsive design.

This essay will be a walk-through of the consecutive steps I went through to create a website for my client.

Introduction

Il sera question dans ce mémoire, de la conception, suivie de la phase de développement de mon projet, réalisé dans le cadre de mon stage de fin de formation au pôle numérique de l'ADRAR.

Ce stage s'est déroulé sur la période du 19 juin au 08 septembre, et avait pour objectif de créer un site vitrine pour le compte de Monsieur Scherrer, gérant de l'entreprise, la "Monkey compagnie", société d'événementiel.

Ce mémoire sera divisé en deux grandes parties, la première reprendra le cahier des charges produit à destination de Monsieur Scherrer, partie qui détaillera la conception du site, ce cahier des charges sera agrémenté de différentes définitions des concepts mobilisés au cours de la création du site.

La deuxième partie suivra le développement du site vitrine, en reprenant les différentes étapes suivies afin de réaliser le souhait du client. Cette partie comprendra de nombreux extraits de code commentés, tout en expliquant les spécificités techniques nécessaires à la compréhension du travail effectué.

A/ Cahier des charges : Conception d'un site vitrine

I - Descriptif du projet

La Monkey Compagnie est une entreprise d'événementiel, basée à Toulouse et spécialisée dans l'organisation d'événements festifs. Le gros de l'activité de l'entreprise consiste à organiser des animations dans des endroits tels que des bars, ou encore lors d'événements festifs tels que des mariages ou des anniversaires.

La Monkey Compagnie fut dans un premier temps créée en mars 2022, et a depuis, pu décrocher des contrats avec des bars toulousains, animant diverses soirées dans différents lieux, en plus de quelques animations de mariages.

Les animations proposées par la Monkey Compagnie sont typiquement des blind-tests, un jeu où les participants doivent deviner le titre d'une musique à partir d'un court extrait, ou encore des questionnaires à choix multiples sur des questions de culture générale.

Au cours d'une période de pause au cours de l'été 2023, Monsieur Scherrer a eu le souhait de développer son activité, décidant de la création d'une page Facebook, ainsi que d'un site vitrine afin de développer son activité auprès de nouveaux clients et d'augmenter sa présence en ligne. La tâche de développer le site vitrine a donc été confiée à ma charge, et j'ai pu profiter de ce besoin de Monsieur Scherrer afin de valider ma formation de neuf mois au sein du pôle numérique de l'ADRAR.

Le site vitrine de la Monkey compagnie devra rendre compte des différentes activités de l'entreprise. Renseignant sur le type d'activités proposées par la Monkey compagnie. Il faudra aussi que les lieux et les dates des événements à venir soient facilement repérables par les visiteurs du site. Le client a aussi manifesté le désir de pouvoir mettre à disposition des visiteurs du site, différentes photographies, prises au cours de différents événements. Afin de montrer à quoi pouvait ressembler de telles soirées. Finalement, le client a manifesté le souhait que le site vitrine de la Monkey Compagnie puisse permettre de le contacter, notamment au cas où le gérant d'un bar souhaiterait demander un devis pour une prestation à Monsieur Scherrer.

II - Equipe du projet

Je me suis alors vu confier la tâche de créer un site vitrine pour le compte de Monsieur Scherrer, seulement, je n'étais pas seul lors de la réalisation du site, une autre personne ayant rejoint le projet dès son commencement.

Il s'agit de Madame Kelly Garcia, connaissance de Monsieur Scherrer, et qui travaille actuellement à l'obtention d'un diplôme au sein d'une formation de web design. Voulant profiter de l'occasion de la création du site vitrine, elle s'est jointe au projet, nous faisant profiter de ses compétences en web design.

Nom – Prénom	Rôle projet	Société	E-mail de contact
Delmas Leo	Responsable développement	Stagiaire	Leodelmas31@gmail.com
Kelly Garcia	Responsable design	Stagiaire	/

L'organisation de la répartition des tâches s'est alors révélée relativement évidente, Madame Garcia ayant le lead pour tout ce qui touchait au design du site, tandis que je prenais en charge tout ce qui concernait de près ou de loin le code. Ce qui ne signifiait pas que les deux axes de travail étaient imperméables l'un à l'autre. En effet, Madame Garcia s'est par exemple proposée de coder, en HTML/CSS, le header et le footer du site, parallèlement, là où Madame Garcia a proposé un design à valider par le client lors de réunions, j'ai aussi participé à ces rencontres, donnant mon avis et prenant en charge l'adaptation du design du site en version mobile et tablette.

Les réunions que j'ai pu avoir avec Madame Garcia, où nous discussions à deux de la mise en place d'un design, avaient pour base des dessins faits par ma collaboratrice, nous reconstruisions alors un figma, dans le but de le présenter au client. Ou, tout en restant lead de la partie design, Madame Garcia restait à l'écoute de mes propositions et suggestions. Une fois le design présenté au client, nous prenions en compte ses suggestions afin de proposer un second, voire un troisième jet, afin d'arriver à un résultat satisfaisant pour toutes les parties.

En tant que lead sur tout ce qui ne touchait pas directement au design, j'avais la main sur toute la partie développement et conception (à l'exception du design). Cependant, je continuais de tenir au courant de l'avance du projet ma collègue, et elle assistait aux différentes réunions avec le client. Nous pouvions alors participer et injecter nos points de vue aux discussions mêmes si elle ne faisait pas complètement partie de nos champs d'expertises respectifs.

Cette manière de faire, basée sur une collaboration, un aller-retour composé de propositions et de contre-propositions entre les prestataires et le client, ainsi que d'une adaptation constante au changement vis-à-vis des demandes du client jusqu'à la cimentation d'un design stable, pourra être défini comme une manière de procéder « agile ».

III - Contexte du projet

I - La Monkey Compagnie

La Monkey Compagnie, fondée donc en mars 2022 par Anthony Scherrer, propose alors diverses animations, telles que des blinds test ou des quizz. Au début du stage, la compagnie ne dispose, en tant que présence en ligne, que d'un compte Instagram.

Le ton sur ce compte y est cordial, informel mais respectueux, le vouvoiement est utilisé, il se prévu de conserver ce ton sur le site afin de garder une uniformité de ton et d'assurer une image de marque uniforme.

Le but de ce site est donc d'étendre cette présence numérique, en proposant aux internautes de s'informer sur les prochains événements, de pouvoir contacter le client, ainsi que de pouvoir avoir accès à diverses photographies des événements passés afin de pouvoir avoir un aperçu de l'ambiance de ces soirées.

Le site répondra donc, dans un premier temps d'un point de vue assez général, à trois besoins :

- Mise à disposition de renseignements
- Création d'un lien supplémentaire entre le client et son public
- Illustration de l'activité du client via des photographies

Je détaillerais dans la partie ci-dessous, plus en détail les demandes du client, ainsi que les propositions faites pour répondre à ses demandes.

I-Expression du besoin

Besoin du client

Comme mentionné ci-dessus, la Monkey compagnie ne dispose actuellement, sur le plan de la communication en ligne, que d'un compte Instagram, sur lequel sont publiés les annonces d'événements à venir.

Cependant, le gérant de la Monkey Compagnie souhaiterait aussi disposer d'un site lui permettant à la fois de présenter le contenu (blind test, quizz...) et le lieu (soirée en bar, mariage...) des animations à venir. Le client a aussi formulé l'idée qu'il voudrait que soit visible sur une des pages de son site, les différents employés de son entreprise, car, même s'il est, pour l'instant le seul salarié de son entreprise, il apprécierait pouvoir présenter les nouveaux employés sur son site, et donner des « têtes à voir » aux gens visitant son site.

Un autre souhait présenté comme primordial pour le client, est de donner à voir le programme des événements à venir. Une, sinon la fonction principale du site serait de tenir informé d'un simple coup

d'œil les personnes recherchant une activité de bar. La liste des événements à venir devra alors être accessible rapidement, et en moins de clics possibles. Et ce, quel que soit le format de l'appareil sur lequel se trouve la personne cherchant l'information.

Il sera aussi possible de poster sur le site des quizz tirés de précédents événements, ou créés pour l'occasion, et de les rendre accessibles au visiteur afin qu'il puisse avoir une idée de la teneur des événements proposés par la compagnie.

Le site comportera aussi une page dédiée aux photographies et aux vidéos prises durant les différents événements, ajoutant un support visuel à la communication de la Monkey compagnie.

Tous ces éléments nécessiteront un panel admin afin que Mr. Scherrer puisse ajouter, supprimer ou modifier des événements, des quizz, des collaborateurs ainsi que des photographies, le besoin d'une interface simple d'utilisation ayant été exprimé, il faudra penser une page admin ergonomique permettant une actualisation fréquente du site. Le tout étant alors sécurisé contre les différents types d'attaques tels que les attaques CSRF, les injections SQL et les failles XSS, qui seront détaillées plus en avant dans le mémoire.

Certains des objectifs du site, considérés comme moins importants ou non-réalisables à l'heure actuelle, ont été considérés comme à réaliser, non pas sur la durée du stage, mais dans un futur plus ou moins proches, ces objectifs ont fait l'objet de recherches, et même de quelques premières lignes de code, mais il n'y avait pas pour objectif de les considérer comme des livrables d'ici la fin du projet.

Monsieur Scherrer souhaiterait disposer d'une newsletter lorsque son activité sera un peu plus développée. Il souhaite aussi, dans le futur, disposer d'une page reprenant, via une API, les différents commentaires laissés sur les pages des réseaux sociaux, lorsqu'ils seront créés. Finalement, le client souhaiterait que son site comporte de petites animations « fun » et interactives, comme un léger changement dans le logo, lorsque la souris passe dessus.

Le site ne proposera cependant pas de système d'inscription, dans la mesure où il n'existera pas de situation nécessitant que les visiteurs s'inscrivent pour bénéficier de telle ou telle fonctionnalité. Cependant, le site conservera, dans le futur, des adresses mail, puisque le souhait qu'une newsletter soit présente sur le site a aussi été exprimé.

Analyse du besoin :

L'existant :

- Outre les différents comptes sur les réseaux sociaux, la compagnie ne dispose pas de site vitrine préexistant. Mr. Scherrer dispose tout de même d'une banque d'image, tirée des précédents événements, prises avec l'accord du public de ces événements, afin de fournir le site en illustrations.

- La Monkey Compagnie dispose à l'heure actuelle d'un logo, désigné par un graphiste, ainsi que d'une palette de couleur discutée et fixée sous le regard de Madame Garcia, quelque temps avant le début du projet, dont il est souhaité qu'elle soit réutilisée pour la palette de couleur du site. Il sera possible de recontacter le graphiste ayant réalisé le logo afin de le modifier si le besoin s'en faisait ressentir, où possiblement s'il s'avérait que le site nécessitait de créer de nouvelles images.
- Madame Garcia s'est aussi proposée dès le début du projet, par une préférence personnelle et par souhait de s'entraîner au code. De coder le header et le footer du site, ils seront alors considérés comme des existants dans la mesure où ils ne feront pas l'objet d'une réflexion au niveau du code. Il est cependant important de noter que Madame Garcia a décidé de coder ces deux composants en utilisant la librairie CSS Bootstrap. Et que, même si son design a été conservé, j'ai proposé avec son accord, de remplacer l'utilisation du framework par du CSS vanilla, le procès et les raisons d'un tel choix seront détaillées plus en amont de ce mémoire.

Livrables :

- Il faudra donc, dans un premier temps et pour respecter les demandes de bases du client, concevoir un site attractif et simple d'accès, afin que les utilisateurs ne soient pas rebutés ni perdus, rendant les autres objectifs du site inatteignables.
- Il faudra aussi apporter un soin particulier à la version mobile, ainsi que tablette, du projet, afin que le site soit consultable sur différents supports, et dans différentes conditions, par exemple des gens sur leur portable dans la rue, à la recherche d'une animation de bar.
- Designer une page d'accueil, qui regroupera une présentation ainsi que certaines informations, telles que le programme des soirées à venir, afin de les proposer directement au visiteur.
- Une page dédiée à la présentation des différentes photographies, idéalement organisées par thématiques, par album, en fonction du type ou du lieu de l'évènement par exemple.
- Une page où sera implémenté un « quizz », avec lequel les utilisateurs pourront interagir, et qui donnera un avant-goût du type de questions posées au cours des jeux proposés par Monsieur Scherrer. Ajoutant une petite dimension interactive et originale au site.
- Une page comportant un formulaire de contact, afin que les utilisateurs puissent envoyer un message au client, soit pour effectuer des feedbacks, des demandes pour les prochaines soirées, des questions ou des demandes de devis pour une prestation.

- Un panel admin permettant à Monsieur Scherrer d'ajouter, de modifier ou de supprimer tout évènement, photographie, collaborateur, quiz du site.
- L'application devra être sécurisée, afin qu'une action tierce malveillante, telle qu'une attaque XSS ou CSRF ne permette par exemple de récupérer les identifiants de connexion de l'administrateur.

Livrables futurs :

- Une newsletter est en discussion et pourra aussi être implémentée au site, permettant de tenir informé les abonnés des évènements dont la date approche.
- Des petites animations réalisées en vue.js, permettant de rendre le site plus dynamique
- Une page, ou un composant qui servira à récupérer les avis déposés sur les plateformes des réseaux sociaux de mon client, une fois qu'ils auront été développés.

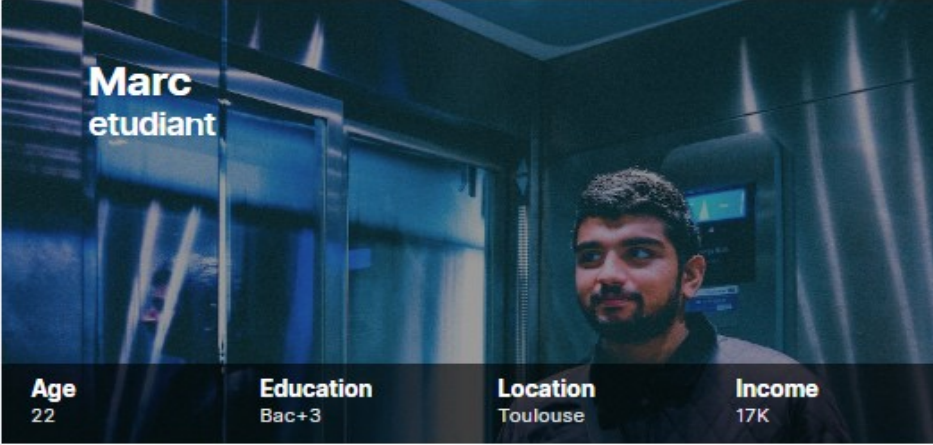
Les cibles :

Les deux cibles typiques du site seront :

- Une personne jeune, à la recherche d'un évènement et de renseignements sur les prochaines activités prévues par la Monkey compagnie. C'est le genre de personnes qui auront tendance à consulter le site en version mobile, possiblement dans la rue alors qu'elle recherche une activité à faire en soirée.
- Un ou une propriétaire de bar, cherchant à organiser un évènement dans son établissement. C'est pour ce genre de public que le formulaire de contact sera créé, afin que ces personnes puissent contacter Monsieur Scherrer, notamment pour demander des devis, en vue de le programmer dans leur établissement.


Ces deux profils seront détaillés plus en avant dans les deux persona ci-dessous.

2-personas




Marc
etudiant

Age 22	Education Bac+3	Location Toulouse	Income 17K
------------------	---------------------------	-----------------------------	----------------------


 **Quote**

On fait quoi ce soir ?


”

 **Goals**

- Trouver où sortir le week-end.
- Se tenir au courant des derniers evenements.

 **Story**

Un jeune étudiant en sortie de License/récemment diplômé, il apprécie sortir dans les bars le week-end, et à apprécié une soirée animée par la Monkey compagnie, et cherche donc a savoir quand et ou se déroulera la prochaine animation.

 **Needs**

- Accéder rapidement aux renseignements sur les soirées disponibles en ville
- Etre renseigné sur la nature et les spécificités de ces événements
- Connaître le feedback des autres utilisateurs



Charles

Patron de bar

Age
50

Education
bac+5

Location
Toulouse

Income
30K

Quote

Qui programmer pour la soirée de vendredi prochain ?

”

Story

Cherche à planifier les animations à venir pour son bar, il sait qu'un autre gérant de bar à déjà embauché la Monkey compagnie pour une animation, et s'est estimé satisfait du service, il cherche donc à accéder à d'avantages d'informations sur cette compagnie d'événementiel.

Goals

- Programmer de nouvelles animations
- Etre parfaitement au fait des services proposées par les prestataires
- Entretenir une communication claire avec les personnes démarchées

Needs

- Augmenter la fréquentation de son bar
- Proposer des activités variées
- Prévoir un budget

3-contraintes

Contraintes techniques :

- Même si le site ne nécessitera pas la mise en place d'un système de connexion et d'inscription pour les utilisateurs, il sera nécessaire de réfléchir à la sécurisation du compte admin de Mr. Scherrer. Dans la mesure où ce sera à travers celui-ci que le site sera actualisé et entretenu. De plus, il faudra penser à conserver et protéger les adresses mail gardées en base de données, et nécessaire au fonctionnement de la newsletter.
- Compatibilité avec des supports desktop, mobile et tablette.
- Une bonne vitesse de chargement des pages et des ressources du site. Ainsi qu'une ergonomie optimale, autant pour l'utilisateur que pour l'administrateur du site.
- Une phase de testing E2E (end to end) sera prévue, via le framework de test cypress.js. Les tests e2e consistent à tester une fonctionnalité du projet, de bout en bout donc, en simulant, grâce à du code javascript, le comportement d'un utilisateur, on testera alors le remplissage d'un formulaire et en comparant le résultat attendu et le résultat réel, les tests seront détaillés plus en avant dans ces pages, dans la partie qui leur est consacrée. Il faudra notamment tester les différentes erreurs potentielles du panel admin, qui constituera le principal regroupement de formulaires du site.
- Afin de s'assurer du bon fonctionnement du site. Il a aussi été convenu que je prendrais en charge la phase de mise en production et de déploiement du site. De plus, je me suis engagé à assurer un suivi du site, afin que Monsieur Scherrer soit assuré d'avoir toujours un support à contacter en cas de problème, en plus du développement des livrables que mon client souhaite voir apparaître sur son site dans le futur, tel que la newsletter

Contraintes légales et réglementaires :

- Le site ne stockera pas d'informations telles que des mots de passe, toujours à l'exception des informations d'admin, cependant il sera utile de se renseigner sur les politiques de droit à l'image, dans la mesure où le site comportera de nombreuses photos d'évènement. Il faudra aussi respecter les lois RGPD concernant la protection et la conservation des données, dans la mesure où des mails, seront, à terme, conservés en base de données afin de faire fonctionner la newsletter. Il faudra aussi rédiger les Conditions Générales d'Utilisations du site afin de les intégrer via un lien dans le footer.

5 – Solutions SEO

Lorsque l'on mentionne la concurrence, c'est, en plus de comprendre les codes du milieu, de savoir aussi comment s'en démarquer et exister au milieu de cet écosystème. Et quelle autre manière d'être visible sur internet que d'étudier et d'offrir un SEO de qualité au client ?

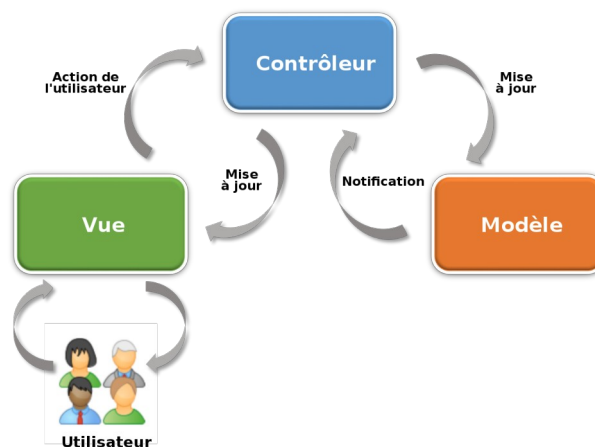
Il existe plusieurs stratégies permettant au site d'acquérir un bon référencement :

- Il faut en premier lieu choisir un ensemble de mots-clés, qui permettront d'identifier l'activité de l'entreprise, ces mots-clefs pourront alors être intégrés au site, à travers les titres et les métadonnées des headers de page. Mais aussi dans les différents en-têtes (h1, h2) du site. Il sera aussi possible de les intégrer dans les URL, et de créer des liens vers ces URL, via des textes d'ancrages faisant eux aussi référence à ces mots-clés. Idéalement, si la Monkey Compagnie gagne en notoriété, il serait aussi intéressant que les textes d'ancrages de liens extérieurs (backlinks), les liens de sites tiers vers le site de la Monkey Compagnie, fassent aussi référence à ces mots-clés. Finalement on pourra aussi retrouver les mots-clés servant de « alt » aux différentes images du site.
- Les mots clés ayant été choisis pour représenter la Monkey Compagnie sur internet sont : animation, bar, fêtes, blind test, soirée, mariage, anniversaire, quizz et beer pong. Le choix de ces mots clés, en plus du fait qu'ils représentent l'activité de la Monkey Compagnie, a été fait en s'aidant de l'outil « Google Keyword Planner », qui permet de mesurer le nombre mensuel de recherches concernant un mot clé, ainsi que la concurrence existant dans ce domaine précis. Les termes mentionnés ci-dessus tournent donc autour de 10k – 100k recherches mensuelles avec une concurrence faible. A l'exception notable de « bar » terme éminemment recherché entre 1M et 10M, toujours avec un indice de concurrence faible. Le terme beer-pong sera aussi ajouté car la Monkey Compagnie, dans un souci de diversifier, a commencé à animer des soirées avec ce type d'activités.
- Il faudra aussi apporter un soin particulier aux types de balises HTML utilisées au sein des différents composants du site, particulièrement vis-à-vis de la sémantique des balises. Une balise <header> devant être utilisée pour coder un en-tête d'un composant, ou d'une page entière. Le choix des différentes balises sera explicité au cours de la deuxième partie de ce mémoire.
- Finalement, un soin apporté à la structure du site, avec des thématiques définies pour chaque page, ainsi qu'un header comportant les principales métadonnées en plus des mots-clés, couplés à des URL simples et explicites, seront nécessaires afin d'offrir un référencement de qualité.

IV - Choix techniques

Vont ici être détaillés les différentes technologies, langages, logiciels, Framework ou librairies ayant été utilisés, il s'agira d'une présentation générale, et les spécificités liées à telles ou telles fonctionnalités seront explicitées dans la deuxième partie du mémoire.

Modèle MVC :



Dans un premier temps, il est nécessaire de noter que le motif de conception du site sera du MVC (Modèle – Vue – Contrôleur). Un modèle couramment utilisé dans le monde du développement afin d'organiser le code d'une application. Il consiste à séparer le code en trois parties :

Le modèle : Contient les données du programme, c'est la partie qui contiendra la BDD, ainsi que la logique ayant trait à ces données, telles que les lectures, enregistrements ou modification de données.

La vue : Consiste en l'interface graphique de l'application, la vue peut servir à afficher des données, mais aussi à les mettre à jour, notamment via des interfaces telles que des boutons ou des formulaires.

Le Controller : Sert d'intermédiaire entre la vue et le modèle, reçoit les actions de l'utilisateur, telles que l'ajout ou la modification de données, notifiant alors le modèle, avant de réafficher une vue avec les données mises à jour. Le Controller peut aussi, dans des cas où l'action de l'utilisateur ne concerne pas une modification directe de données, demander à une vue de se réafficher sans passer par le modèle.

Il faut noter qu'un modèle peut être utilisé par plusieurs vues, qui seront dépendantes, elles, des modèles, les interrogeant pour afficher des données. Le contrôleur dépendra à la fois de la vue, répondant aux actions utilisateurs, et modifiant les modèles en réaction.

Ainsi segmenté, le code gagnera en lisibilité, et les problèmes de fonctionnement seront facilement localisable, augmentant la lisibilité du code, et éventuellement, facilitera l'évolutivité du site, même si je devais passer la main à un autre développeur.

Langages front :



Afin de coder les différentes vues, les langages utilisés au premier niveau seront HTML et CSS

HTML (Hypertext Markup Language) : Un langage de balisage utilisé pour créer et structurer le contenu d'une page web. Des balises, chacune ayant un rôle spécifique, tel que d'afficher une image ou de formater du texte, vont s'imbriquer au sein d'un fichier, afin de créer la structure et le contenu élémentaire d'une page internet.

De plus, comme mentionné au cours de la partie SEO, certaines de ces balises, telles que <header>, <nav> ou <aside>, auront une portée sémantique, et seront conçues pour servir à afficher, ici, un en-tête, une barre de navigation ou une section située sur un côté de la page.

CSS (Cascading Style Sheets) : Langage descriptif servant, via des « feuilles de style » à définir l'apparence et la mise en page du code HTML, en contrôlant sa disposition, sa forme, sa couleur ainsi qu'un ensemble d'autres caractéristiques visuelles, en ciblant les différentes balises du code HTML. C'est aussi via le CSS que sera mis en place un design « réactif », en conditionnant la mise en page du site aux dimensions de l'écran.

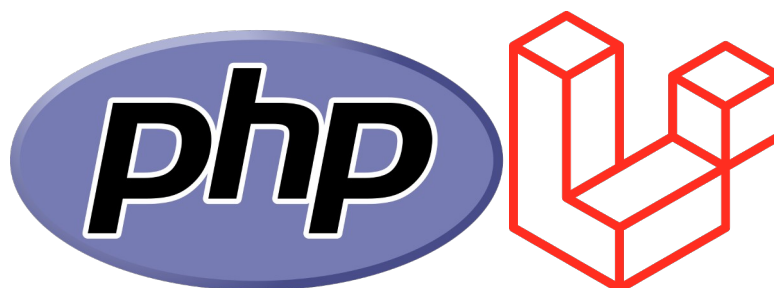
Le côté front du site n'a cependant pas été codé seulement qu'avec ces deux langages, mais avec l'aide d'un Framework front : Vue.js. Un Framework est un ensemble de composants logiciels, ainsi que de conventions de développement, fournissant une infrastructure de base au développeur pour mener son projet à bien, de manière plus rapide, par exemple en automatisant de nombreuses tâches répétitives, lui évitant, selon l'expression consacrée, de « réinventer la roue » à chaque projet. J'ai donc décidé de ne pas me lancer dans un projet codé en « vanilla », version basique d'HTML et CSS, mais de me servir d'un des Framework existant, afin de pouvoir proposer au client, sur les deux mois qui nous étaient offerts, un site « fini », plus performant et plus fonctionnel que si j'avais décidé d'utiliser simplement Html et CSS.



Vue.js : J'ai choisi d'utiliser, parmi les différents Framework front à disposition, vue.js, qui utilise des Template à la syntaxe « html-based », compilés en javascript, afin d'offrir des performances de rendu au-delà de ce que peuvent offrir des pages codées en html/css basiques. Ce gain de performances est notamment dû à l'utilisation par vue.js de la technologie du « virtual DOM », consistant en un DOM « lite » et « intelligent ». Là où l'actualisation d'un DOM classique peut-être couteuse en performance, vue.js appliquera les modifications de la vue d'abord au DOM « virtuel », qui comparera son état avant et après modification, et effectuera les modifications par paquet (batching), rendant moins couteux la mise à jour du DOM réel. Si je ne pense pas que, au stade actuel, le site de monsieur Scherrer nécessite de telles technologies, il faut garder en tête que ce stage ne constitue que la première étape d'un projet plus vaste dans l'esprit de mon client. En utilisant de telles technologies, on s'assure donc d'une durabilité du site, et on garantit par là même la scalabilité du projet délivré. De plus, vue.js fonctionne par « Template » par composants hiérarchisables et réutilisable, et en décomposant ainsi les différentes vues, on s'assure que le projet gagnera en clarté, pour moi-même, mais aussi pour mes potentiels successeurs.

Il faut aussi savoir que vue.js fonctionne en collaboration avec les autres parties de ma pile de technologies, et ce, grâce au choix d'avoir recours à Laravel, couplé à Inertia.js.

Langages back :



Laravel, un Framework PHP : Afin de coder mon back-end, j'ai choisi d'avoir recours au Framework Laravel, afin de coder en PHP mon back-end. Le choix d'utiliser PHP, un langage de script coté serveur a été conditionné par le fait qu'il est le seul langage dont je connais l'utilisation côté serveur, cependant si le choix d'utiliser PHP s'est fait par défaut, le choix d'utiliser Laravel viens des besoins spécifiques du projet.

En effet, si Laravel possède une architecture MVC bien conçue, correspondant à mon choix d'architecture de code, en plus de posséder une documentation particulièrement fournie et une communauté investie et présente sur les forums. Le choix d'avoir recours à Laravel viens surtout de l'existence d'une « glue », tel que la librairie se définit elle-même dans sa documentation : Inertia.js.



Inertia.js : Inertia.js est une librairie javascript, permettant de faire fonctionner de manière plus simple les relations entre le front et le back-end. Habituellement, lorsque l'on construit une SPA (single page application), en utilisant vue.js, il faut alors créer diverses API afin d'assurer la liaison avec le framework back-end. Causant possiblement divers problèmes, tels que des problèmes de compatibilité entre versions, en plus d'un temps de développement accru. Inertia permet alors d'intégrer les pages vue.js

directement au projet Laravel. Le Framework PHP utilise traditionnellement des vues en format .blade, des Template elles aussi html-based, mais l'utilisation d'Inertia permet alors de disposer de pages vues.js, et des avantages décrits ci-dessus, sans avoir à construire d'API. Le choix d'utiliser Laravel devient alors plus clair, dans la mesure où il s'agit du seul Framework PHP fonctionnant avec Inertia, et le fait que je n'avais jamais utilisé Laravel au cours d'un projet, a pu être compensé par l'importante et claire documentation disponible en ligne.

Choix de base de données :



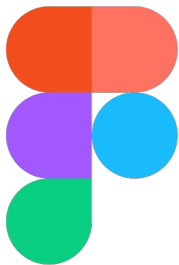
Concernant le choix de la base de données, le choix s'est effectué selon les compétences dont je disposais au moment du projet. Il s'agissait du système de gestion de base de données que j'avais pu utiliser au cours de ma formation, et j'ai pu, au cours du projet, utiliser les query de l'ORM de Laravel « eloquent » afin d'interroger ma base de données, et non pas le langage SQL, traditionnellement utilisé pour les bases de données. Cependant, au cours de ce mémoire, lorsque je montrerais une méthode de l'ORM de laravel me permettant d'effectuer telle ou telle opération, je retranscrirais la méthode en son équivalent SQL, afin de démontrer les équivalences entre ces deux manières de procéder.

Technologies auxiliaires :

En plus de ce stack de trois technologies complémentaires, j'ai pu utiliser au cours du projet les logiciels ou technologies suivantes, afin de m'aider dans la conception et le développement du projet.



GitHub, un service web d'hébergement et de gestion de développement de logiciel. Permettant de versionner son code, et donc d'avoir accès à des états antérieurs du développement, il permet aussi de créer des espaces de travail collaboratifs en mettant en place des branches, chacune dédiée à un participant du projet, tel que moi et Madame Garcia.



Figma, une plateforme collaborative pour éditer des graphiques et faire du maquetage. Particulièrement utile lorsque j'ai pu créer un espace de travail collaboratif avec ma collègue en charge du design, afin que nous puissions travailler à deux sur les maquettes.



Xampp, un ensemble de logiciels permettant de mettre en place un serveur Web local, un serveur FTP et un serveur de messagerie électronique, utilisé avec le système de gestion de base de donnée MariaDB, me permettant de travailler avec mes modèles de données pendant toute la phase de développement.

V - Planification

5- Diagramme de GANTT

[On trouvera en annexe 3 le diagramme de GANTT](#)

6- Diagramme de PERT et note de cadrage

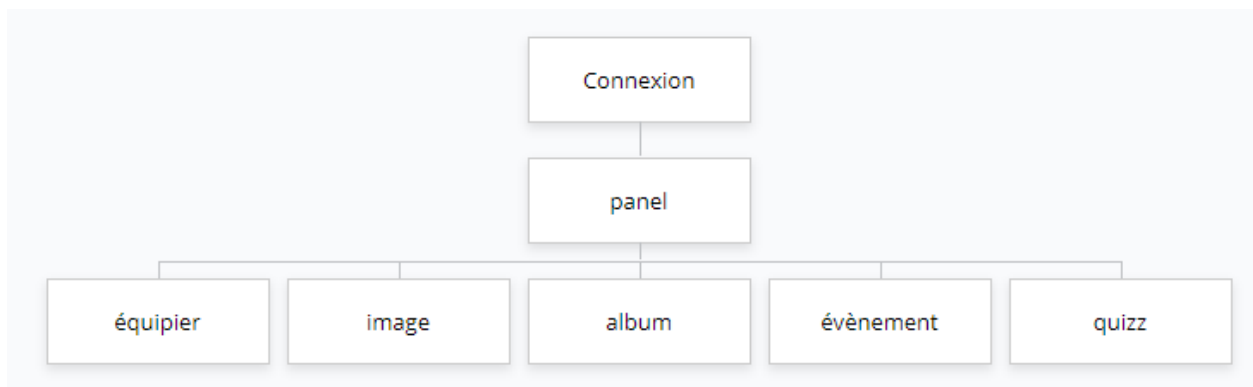
[On trouvera en annexe 4 la partie comportant le diagramme de PERT et la note de cadrage](#)

VI - Conception

I- Arborescence

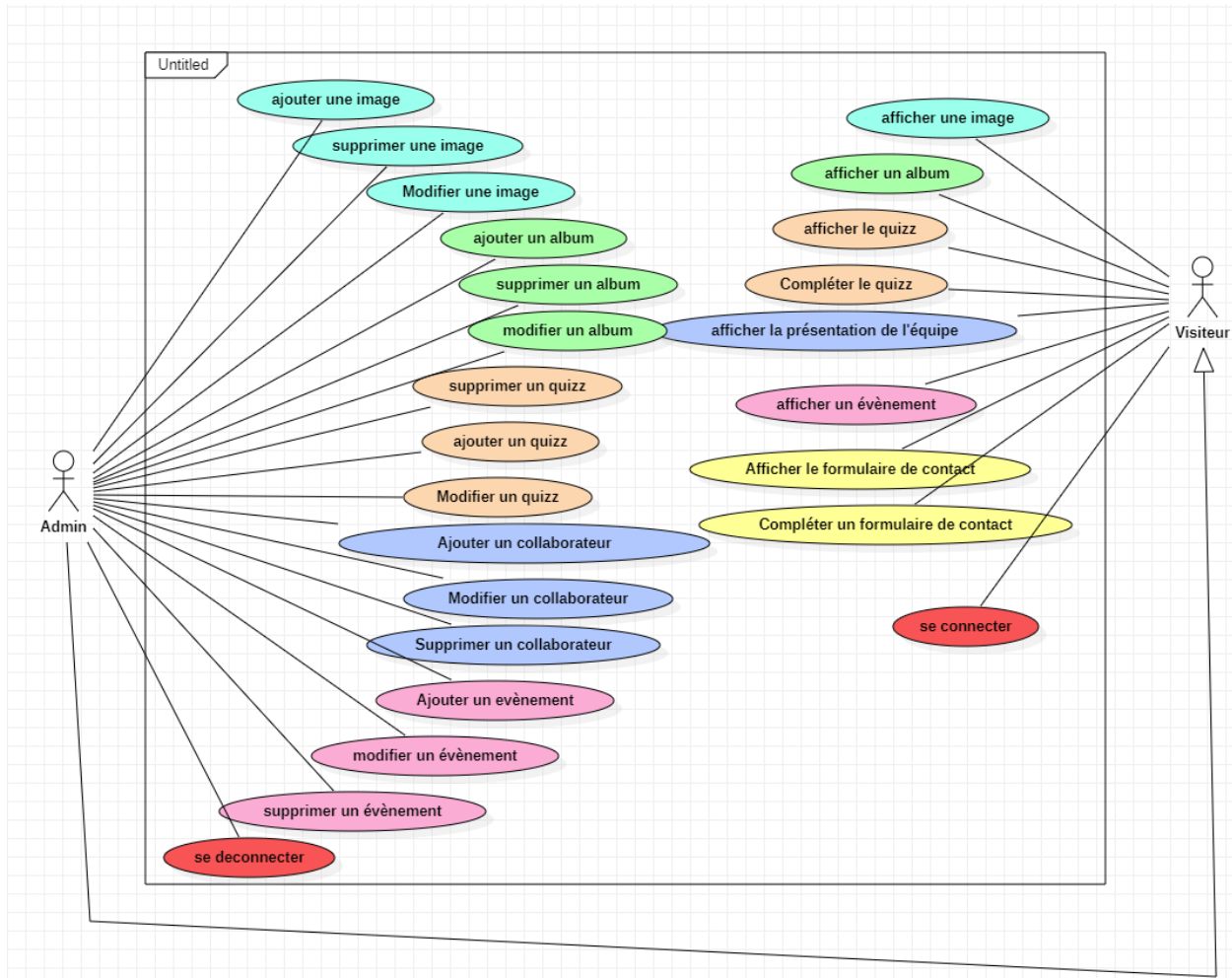


Le site sera donc conçu selon un modèle où la page 'home', sera une page de présentation, comprenant un texte décrivant l'activité de la Monkey Compagnie. Seront aussi présents sur cette page, les différents portraits des employés/animateur de la Monkey Compagnie. Une navbar présente sur les différentes pages du site permettra alors de naviguer entre les 5 différentes pages, chacune portant une ancre de lien, révélatrice de sa fonction, programme, galerie, contact et Quizz.



Ci-dessus, l'arborescence du panel admin, il sera accessible via une adresse spécifique à taper dans l'URL, et redirigera vers un écran de connexion. Une fois connecté, l'utilisateur pourra accéder à ce qui est ici appelé le « panel » c'est-à-dire une simple navbar permettant de naviguer sur les différentes pages permettant d'ajouter, supprimer ou modifier les entités qui seront pour l'instant consultables sur le site.

2-Use Case Diagram

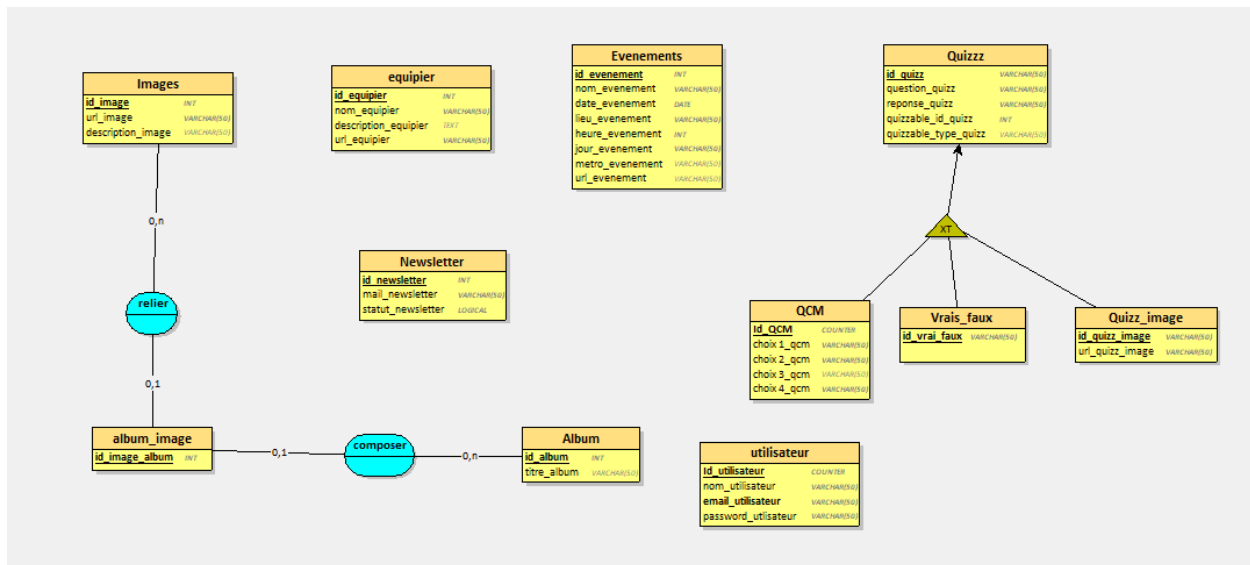


Sont représentés ici les acteurs et les cas d'utilisation inhérents au site de la monkey compagnie. Nous avons deux types d'acteurs, les visiteurs et l'administrateur du site, monsieur Scherrer. Les utilisateurs pourront donc afficher les différentes entités créées pour être affichées sur le site, à savoir : les images, albums, quizz, équipiers, évènements. Le visiteur pourra aussi afficher et remplir le formulaire de contact, afin d'envoyer un mail à Monsieur Scherrer.

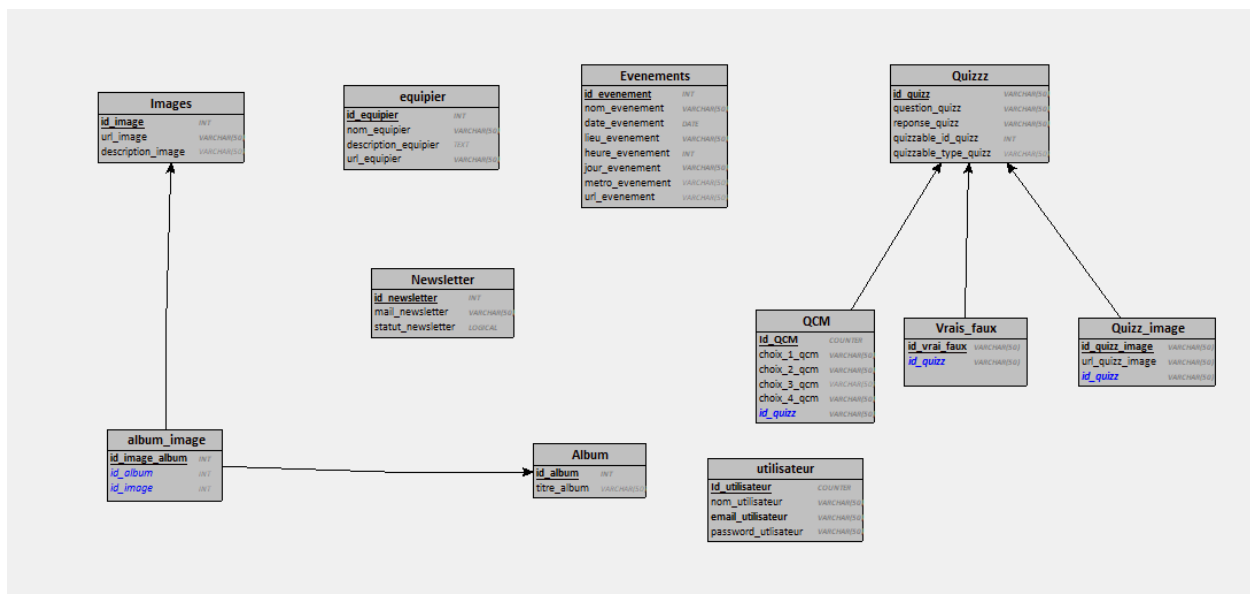
Il sera aussi possible, dans la possibilité où le visiteur est Monsieur Scherrer, qu'il puisse se connecter au panel admin, il pourra alors avoir accès aux cas d'utilisation de l'administrateur, en plus de ceux du visiteur. A savoir, ajouter, modifier ou supprimer une image, un album, un quizz, un équipier ou un évènement. Finalement, Monsieur Scherrer pourra aussi, se déconnecter.

3-MCD/MLD

MCD :



MLD :



Ci-dessus, les représentations MCD/MLD de la base de données nécessaire au site, un MCD (modèle conceptuel de données) est une représentation abstraite de données et de leurs relations, il permet de se représenter les models utilisés, ainsi que les attributs de chaque entité, ainsi que les types de relations qui les contraignent. Le MLD (Modèle Logique de Données) est lui aussi une représentation d'une base de données, basée sur le MCD et qui ajoute des informations telles que les clés étrangères

Nous avons d'abord une table « équipier » regroupant le nom de l'équipier, ainsi qu'une description et une image (url_equipier), qui sera affichée sur la page d'accueil du site, répondant au besoin formulé par le client de pouvoir afficher une présentation de son équipe.

Une table « évènement », correspondra aux évènements à venir plus ou moins proches. La table contiendra des attributs tels que le nom, le jour, le lieu, l'heure et l'arrêt de métro le plus proche. La table contient également un attribut « date », qui servira, lui, au moment de récupérer les données, à pouvoir les classer plus aisément par ordre chronologique. Une image sera aussi présente dans la table afin d'illustrer l'évènement, les soirées animées par mon client étant le plus souvent accompagnées d'une affiche créée par monsieur Scherrer pour son compte instagram.

Un autre des besoins de mon client, le fait de pouvoir afficher diverses images organisées par thématiques, sera rempli en créant une table « images », contenant la destination de l'image (url_image), ainsi qu'une description nullable, facultative. Cette entité sera reliée dans une relation many-to-many à l'entité « album », contenant, elle, simplement un titre.

Une relation many-to-many signifie que les entités images et album sont reliées entre elles, et qu'une image peut faire partie de plusieurs albums différents, et qu'un album peut contenir plusieurs images. En MCD/MLD, la représentation de cette relation nécessite une troisième table, une table pivot, ici, « album_image », qui contiendra, pour chaque occurrence de la relation, l'id de l'image, l'id de l'album, ainsi que l'id de la relation en elle-même.

On peut se demander à ce stade, pourquoi ne pas avoir créé une entité image, reliée à la fois à album, mais aussi à « équipier » ainsi qu'à évènement. Il serait alors possible de relier ces trois entités à « images » dans une relation one-to-many, c'est-à-dire que chaque équipier, par exemple, aurait une seule image, et les images auraient autant de correspondances que possible, et serait mobilisées dans les trois entités nécessitant une image.

Cependant, après discussion avec mon client, il est apparu que chaque image était un type d'illustration différente. Il s'agissait respectivement de portraits, d'affiches et de photographies prises au cours d'évènement, et jamais un de ces types d'images n'aurait été utilisés dans un autre but que celui pour lequel il avait été pensé. Il paraissait alors plus logique de séparer ces trois types d'images et de les stocker dans des entités respectives.

On remarque aussi la table « newsletter », créée en prévision du développement futur d'une newsletter, qui comportera les différents mails de la liste de diffusion, ainsi qu'un statut booléen, indiquant si la personne est inscrite actuellement à la newsletter, dans l'hypothèse où certaines personnes se désabonnerait, mais où les données auraient encore besoin d'être conservées, en accord avec les lois RGPD.

Quant à la table « quizz », elle ne représentera pas tant un quizz, en entier, mais une seule question. Question qui pourra prendre la forme d'un « vrais ou faux », d'un QCM, ou de ce que j'appelle un quizz-image, se composant d'une illustration accompagnée d'une question, comme par exemple « A qui appartiennent ces yeux ? ».

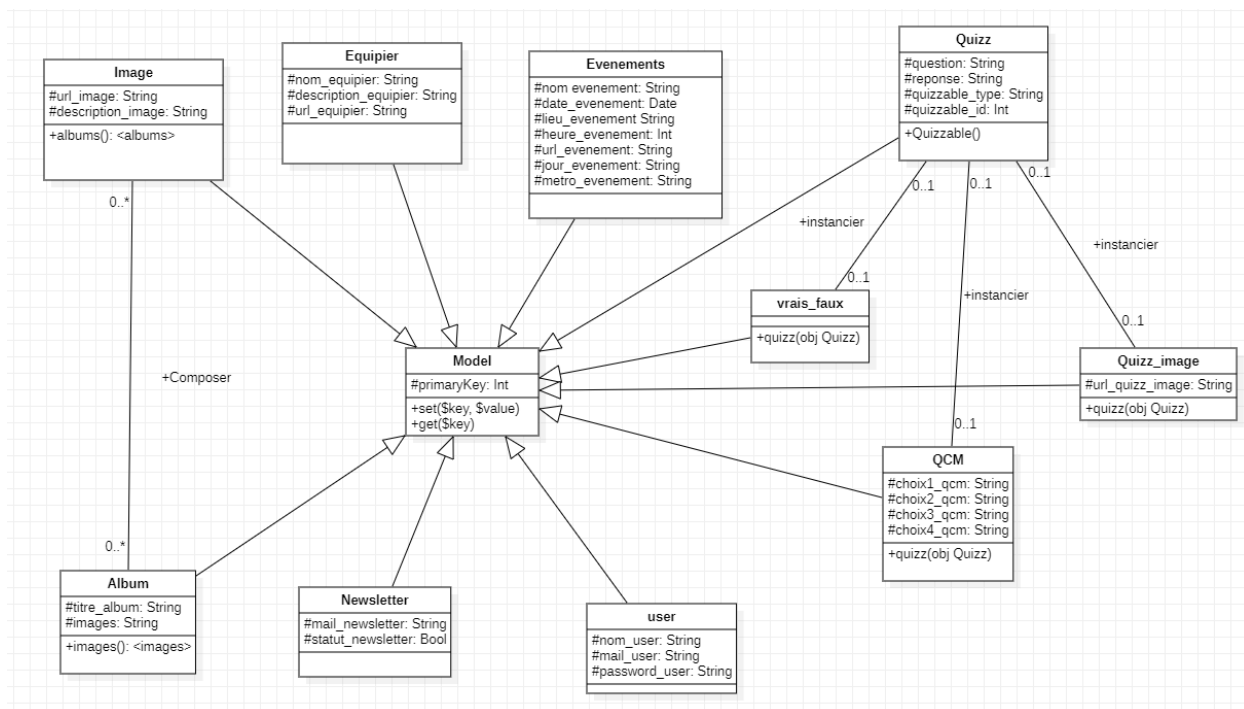
J'ai décidé de traiter cette relation comme une relation d'héritage, ce qui signifie que les trois types de quizz, quel que soit leur type, hériteront des attributs de la classe mère « quizz », à savoir question et réponse, qui sont commun à tous les quizz.

Quant à l'attribut quizzable, il s'agit de la manière qu'à Laravel de gérer les relations d'héritage, en donnant une valeur différente à l'attribut quizzable de l'entité. Cette valeur sera héritée avec les autres attributs et permettra de donner, à chaque occurrence de quizz, un « type ». A noter que, compte tenu de cette manière de procéder, et contrairement à ce que on peut voir dans ce graphique, Larvel ne stocke pas les quizz_id dans les trois types de quizz, mais dans chaque occurrence de la classe supra « quizz », dans quizzable_id, permettant alors de récupérer les différents types de quizz, chacun portant un attribut indiquant son type dans quizzable_type.

Au moment d'ajouter une nouvelle question en base de données, l'administrateur choisira alors le type de question, et pourra ainsi se construire un fond de questions, qui seront ensuite affichées sur la page dédiée au quizz, une par une. Donnant ainsi la possibilité à Mr. Scherrer, de « customiser » à l'envie les quizzes affiché. On notera aussi que chaque question pourra être illustrée d'une image.

En [annexe 2](#), on trouvera le code SQL nécessaire pour créer la base de données, en sachant que, au cours du projet, j'ai donc utilisé la manière de procéder de Laravel, via des fichiers de migration, méthode détaillée dans la partie correspondante du mémoire.

4-Diagramme de classe



Nous avons ici le diagramme de classe du site vitrine de la Monkey compagnie. Contrairement au MCD/MLD qui se construit indépendamment de la technologie utilisée, j'ai ici représenté ici le diagramme de classe tel qu'il est construit en Laravel, sans bien sûr, tomber dans l'exhaustivité, il est intéressant de représenter cette manière de procéder, spécifique à l'ORM Eloquent de Larvel.

Dans un premier temps, on remarque que les attributs de mes différentes classes seront, non pas en *private* comme dans un modèle classique où les accessor et mutator (getters et setters), permettrait d'accéder aux attributs depuis la classe elle-même, mais bien *protected*, ce qui signifie qu'il est possible d'y accéder depuis les classes dérivées. Or, en Laravel, chaque model hérite de la classe Model, située dans `vendor/laravel/framework/src/Illuminate/Database/Eloquent/Model.php`. Cette classe contiendra alors les getters et les setters en public, les générant alors pour chaque attribut de chaque Model créée en héritage de cette classe. Sur le même fonctionnement, on remarque que le model contiendra la clé primaire en *protected*, la rendant disponible dans chaque instantiation du model.

La relation many-to-many entre album et image sera quant à elle gérée par les méthodes `images()` dans la classe album, et `albums()` dans la classe image, permettant de récupérer toutes les entités associées. Il faut noter que si dans la base de données, on trouvera bien une table pivot `albums_images`, il n'existe pas de Model pour cette table pivot, seulement un fichier de migration, héritant de la classe migration.

Quant à la relation que j'avais qualifiée d'héritage entre un quizz et ses instantiations, elle est gérée par Eloquent d'une manière un peu différente, puisqu'il est impossible pour une classe d'hériter de plus d'une seule classe, et tous les models héritant de Model, l'héritage est pensé par Laravel comme une relation *polymorphique*.

La classe « mère » Quizz disposera alors d'une methode `quizzable()`, utilisant la fonction de Laravel `morphTo()`, et les classes pensées comme « filles » disposeront d'une methode `quizz()` utilisant elle la

methode morphOne(). En renvoyant l'objet Quizz à la méthode morphOne, on fera alors comprendre à Eloquent que l'on désire que chaque quizz différent créée, soit relié à un seul Quizz « mère », qui stockera alors la clé du quizz « fille » ainsi que son type, permettant de récupérer pour chaque objet créée les attributs stockés dans la classe supra.

Je présenterais, dans la deuxième partie de ce mémoire, un diagramme plus précis des Models de quizz, prenant en compte les Controllers associés, pour présenter en détail la fonctionnalité d'ajout de quizz.

VII - Charte graphique et maquetage

I-Couleurs et police



Les couleurs ci-dessus ont été choisies pour servir de palette de couleur au site, elles proviennent en grande partie de la palette de couleurs que Monsieur Scherrer utilisait sur son compte Instagram, il s'agit de couleurs qui apparaissaient dans les illustrations de ses publications Instagram.

Les identifiants des couleurs sont, de gauche à droite :

Hexadecimal : # 402927 / RGB 64, 41, 39

Hexadecimal : # 8C6249 / RGB 140, 98, 73

Hexadecimal : # F2D5BB / RGB 242, 213, 187

Hexadecimal : # BB4253 / RGB 187, 66, 83

Concernant la police choisie, elles sont au nombre de deux :

Une des polices utilisées sera « Gobold Upflow » :

La monkey compagnie

Un fond sans-Serif, facilement identifiable et relativement stylisé, qui sera utilisé pour le titre et certains endroits où le texte restera restreint, vu que les longs textes écrits dans des polices trop stylisées peuvent rapidement fatiguer l'utilisateur.

La seconde police utilisée sera « nunito »

la Monkey Compagnie

Une police sans-serif, relativement sobre, utilisée pour les longs textes, typiquement dans le texte de présentation, plus neutre, donc plus agréable et moins fatigante à la lecture.

Il faudra cependant être vigilant, car si nunito est libre de droit, la Gobold Uplow, est payante pour les projets commerciaux. A l'heure actuelle, le site vitrine est un projet personnel et ne nécessite pas de payer la License, mais si cette police était définitivement retenue lors de la mise en ligne officielle, il faudrait alors payer la lifetime License, pour 20 dollars.

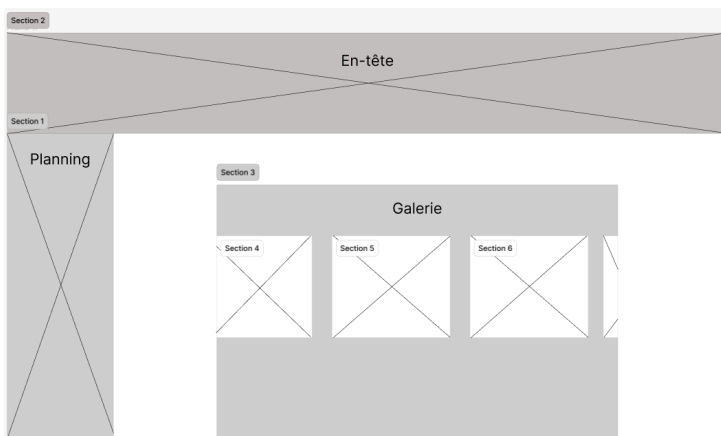
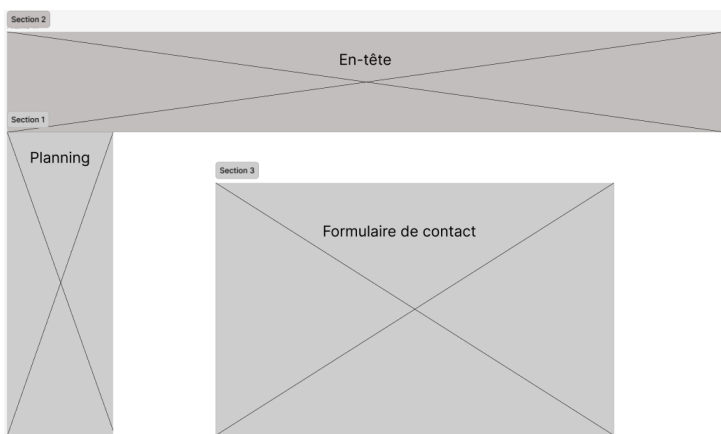
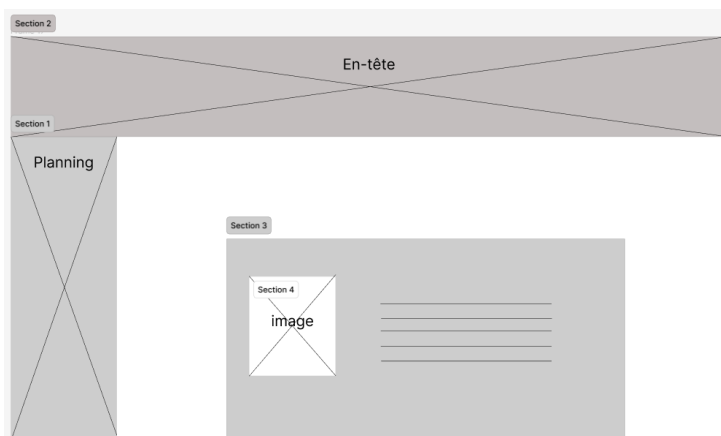
2- Le logo



Monsieur Scherrer a fait appel à un graphiste indépendant afin de désigner un logo utilisable dans l'entête du site. En raison de liens personnels, le graphiste a accepté de créer le logo à titre gracieux, et s'est proposé, dans le futur, de le modifier. Les couleurs utilisées reprennent celles de la palette graphique du site, et une autre version, livrée avec la première, utilise la couleur noire comme couleur dominante, cette version a été créée dans l'éventualité d'un changement de palette de couleur, ayant été jugée plus propice à s'adapter avec plusieurs fonds.

3-Wireframe

Ci-dessous, les images des premiers Wireframe créés un peu en amont du projet suite aux premières discussions avec le client. Ces pages servent à montrer le placement de la div contenant le planning, et sa présence sur les différentes pages de l'application. Le planning ayant été défini comme une des informations principales du site, en raison du fait qu'un des buts du site est d'informer du premier coup d'œil des événements à venir. L'idée originale était, sur les premiers wireframes, de placer une div qui contiendrait alors les différents éléments du site en fonction des pages.



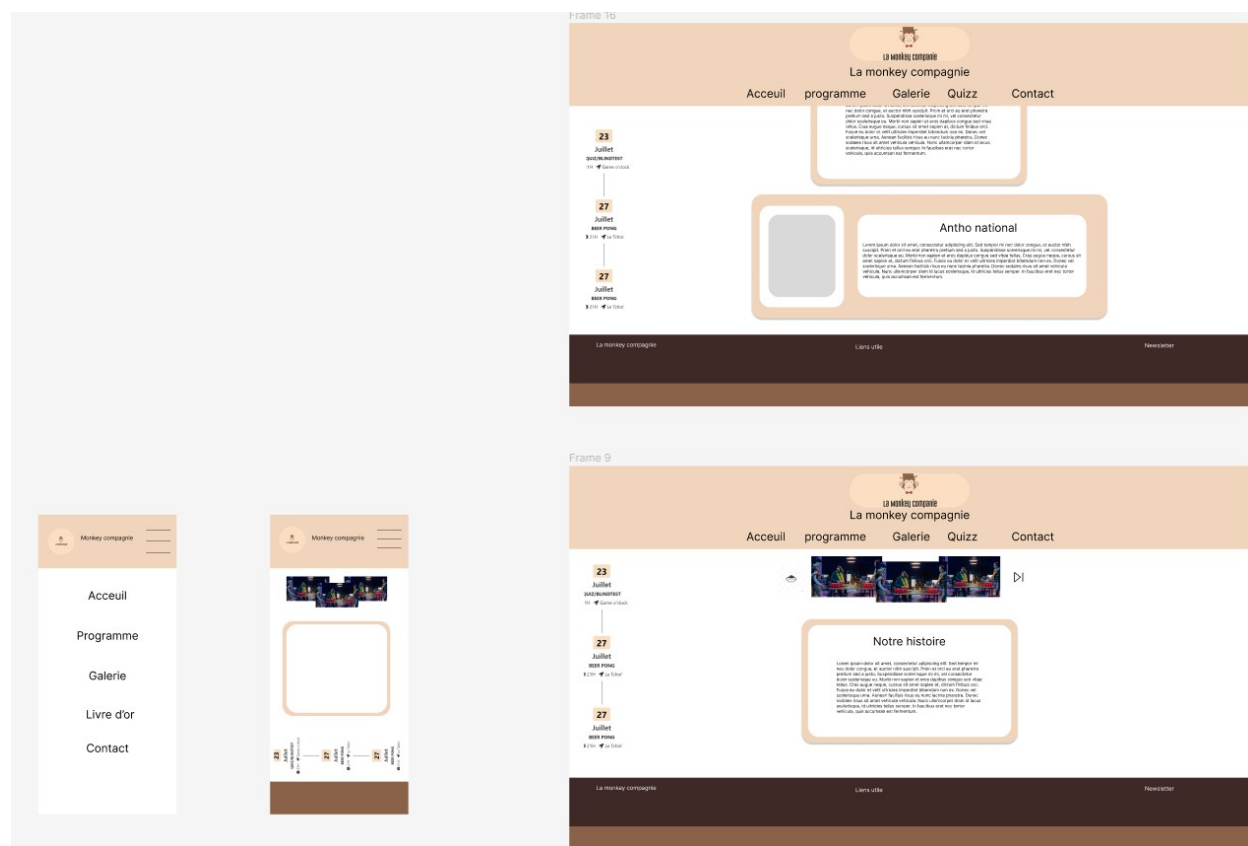
4-mockup

Première version du mockup :

On trouvera en [annexe 5](#) la première version du mockup.

Deuxième version du mockup :

Page d'accueil :



La page d'accueil comportera donc un texte d'introduction décrivant l'activité de l'entreprise, avec, en dessous, un descriptif concernant, pour l'instant, le seul employé de la Monkey Compagnie, à savoir Monsieur Scherrer. Le programme des évènements à venir sera situé à droite de l'écran entre le footer et le header, et sur le même espace horizontal que le corps de page. Un carrousel d'image a aussi été prévu pour être inséré en haut de la page.

Quant à la version mobile, un menu burger avait initialement été prévu, même si Monsieur Scherrer préférera en fin de compte que le menu de navigation soit conservé tel qu'elle et uniquement redimensionné en version mobile. Le programme sera, quant à lui, déplacé en bas de la page, juste au-dessus du footer, et consultable en défilant l'écran de droite à gauche.

On remarque aussi le logo ayant été déplacé de la gauche vers le milieu de l'écran.

On trouvera en [annexe 6](#) les maquettes des différentes pages du site. A l'exception de la page quizz que je montre ici.

Page quizz :



Concernant la page Quizz, Madame Garcia a directement design les différentes cartes, afin que je puisse les coder, ce design s’est fait relativement tard dans l’avancement du projet, et le reste de l’apparence du site était déjà fixée, nous savions alors que il suffisait d’inclure les cards dans le corps du site, et il à été décidé qu’elle se chargerait du design de cette partie tandis que je commençais déjà à coder le frontend, il existera donc trois types de cartes, s’affichant les unes après les autres, une fois que la question était répondue, concernant la version mobile, la carte s’adaptera dans l’espace du corp du site, entre le header, et le programme, toujours affiché juste au-dessus du footer.

VIII - Analyse des risques

I – Matrice des risques identifiés

On trouvera en [Annexe 8](#) la matrice de risques

B/ développement du site de la Monkey Compagnie

I – Situation de recherche

I chose to present in this section my research, made in English, on the documentation of the site of Laravel, the reason is that, for starter, that I tend to usually use English language technical documentation when I find it available. I do it this way because if find it helpful the spot technical and vernacular wording in English through the several documentations I can crawl while trying to solve a specific issue. Moreover, the functions and classes of the several framework used where in English and it made me feel as keeping some kind of coherence through the train of thoughts of the world of coding.

Sometimes you can miss a particular and crucial word if you tend to switch between English and French while going through different sites. And while I'm still quite a beginner in the programming world, I prefer to ensure documentation remains clear without losing in translation the meaning of a concept. Especially when you resort to forums and message bord while looking for an answer, luckily people tend to use key words in English, as to not to lose the meaning of their questions or explanations.

And, the more you code, the more you tend to search for documentation in English, and, by the end of the internship I was exclusively scrolling through English websites, documentations, and videos, learning through youtube tutorial the ability to understand several variations and accents of the English language.

In the end, I Don't know if coding made me a better handler of the English language, but it certainly made me more aware of the particular vocabulary used in the world of computer-science.

II - Création d'un environnement de développement collaboratif

Comme mentionné dans la première partie de ce mémoire, Madame Garcia à créer un header et un footer pour le site, de plus, elle souhaitait pouvoir accéder à mon travail, pour offrir un feedback à ce que je pouvais produire, afin qu'elle puisse réagir, corriger, ou simplement se rendre compte que tel design fonctionnait moins bien une fois codé.

J'ai donc créé un git repository, afin de pouvoir conserver l'avancement du projet, tout en le rendant accessible à ma collaboratrice, d'abord dans un objectif de lui donner accès à mon travail, et possiblement, dans le futur de lui permettre de participer au projet, par exemple lorsqu'elle formule le souhait de coder entièrement le carrousel qui sera inséré dans la page d'accueil. Il serait alors utile de lui créer une branche dans le dépôt Github, et elle pourrait alors collaborer au code à son rythme, et je fusionnerai alors les branches une fois le travail accompli, via la commande 'git merge'.

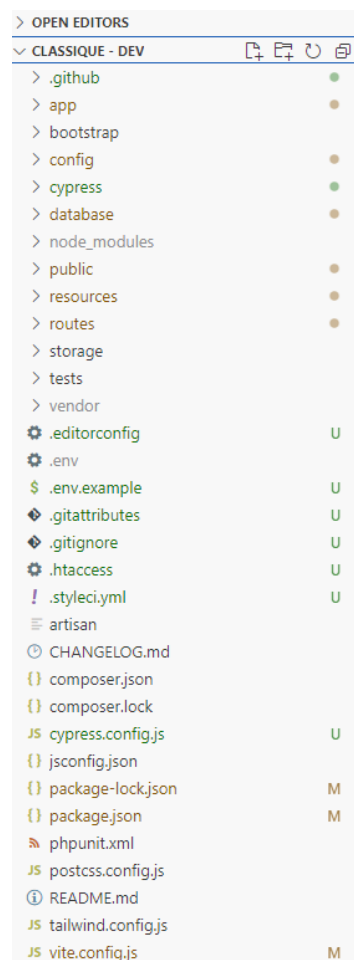
Concernant le header et le footer codés par ma collaboratrice, ils ont été directement incorporés aux composants du projet. Madame Garcia avait décidé de coder ces deux composants en utilisant le framework CSS 'Bootstrap', seulement, le fait d'utiliser un framework de ce volume seulement pour deux

composant me paraissait disproportionné, et, après discussion, nous avons décidé que j'adapterais son code en CSS classique, tout en conservant son design.

Une fois le Github créé, j'ai pu initialiser le projet Laravel, après m'être assuré que je disposais bien des dernières versions de PHP, Composer et node.js, via la commande :

composer create-project laravel/laravel monkeyCompagnie

Ce qui nous donnera le directory ci-dessous :

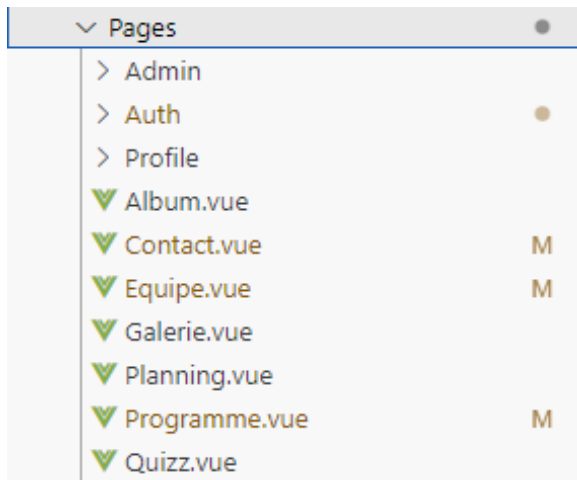


Les views seront situés dans le dossier 'ressources/js', c'est à la mise en place de ces composants vue.js que je vais m'intéresser en premier lieux, avant de détailler la création d'entités au sein de la base de données, dans le dossier 'database' avant de faire un détour par le fonctionnement d'inertia. Finalement, je m'intéresserais à l'aspect plus backend du projet, à travers les controllers dans le dossier 'app' et aux routes dans le dossier du même nom.

III - Développement frontend

I – Mise en place des composants

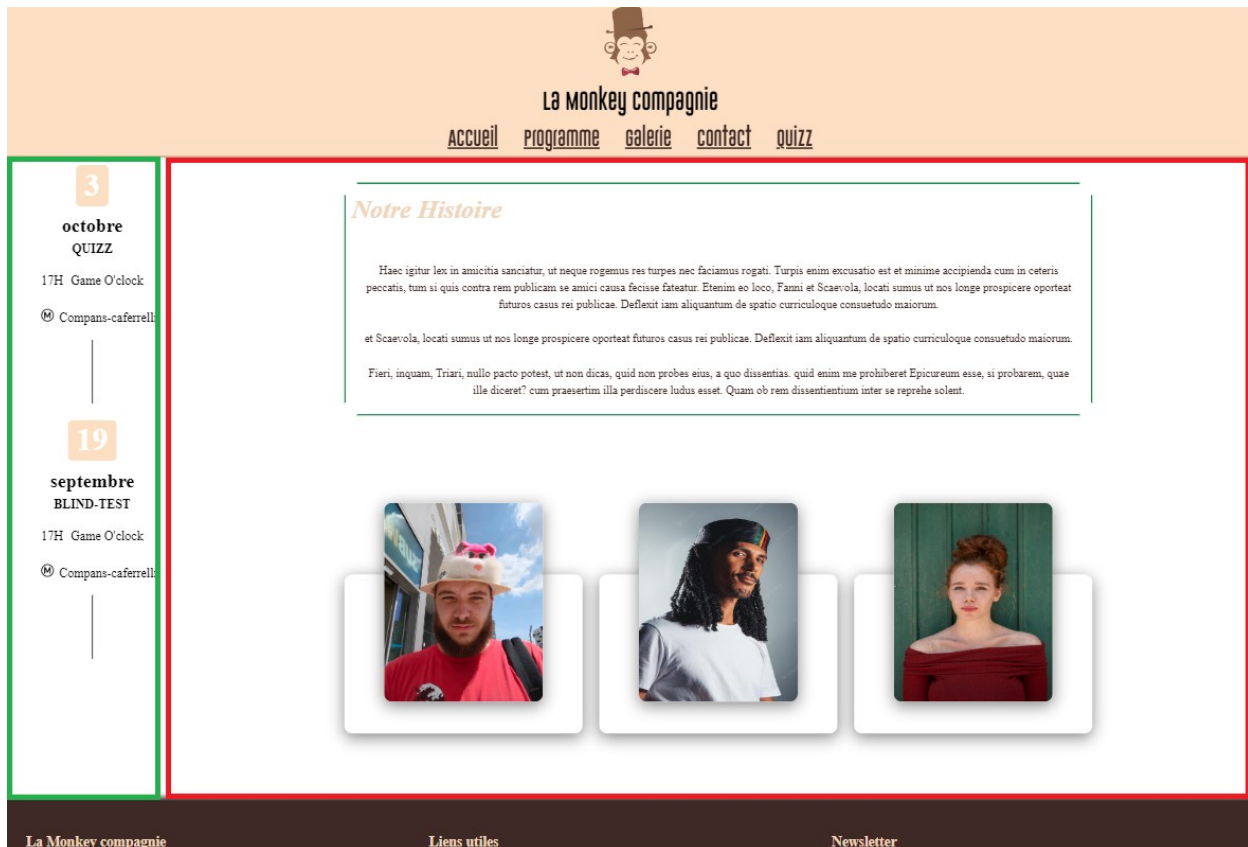
Il faut donc dans un premier lieu, créer les différents composants qui permettront de mettre en place le site, ci-dessous les différents composants du site.



Un composant vue.js peut être défini comme une encapsulation de l'interface utilisateur d'un site. C'est-à-dire une partie précise, autonome, mais encapsulable et réutilisable au sein d'autres composants. Cette manière de procéder permet de délimiter des portions de codes compréhensibles et facilement réutilisables et signifiantes du point de vue de leurs fonctionnalités.

Il a été expliqué dans la première partie les différentes informations qui seront nécessaires à la résolution du souhait du client. On le retrouve toutes ici et ce sont ces composants qui vont servir à les afficher. Ici c'est le composant « équipe » qui servira de page d'accueil, comme mentionné dans l'arborescence du site. On remarque aussi l'existence d'un composant programme et d'un composant planning, tous deux servant à afficher le programme à venir, seulement le composant programme servira à afficher la page où sont présentés uniquement les événements, tandis que le composant planning servira à afficher le planning sous un format réduit au sein des autres composants.

Pour illustrer ce fonctionnement, voici ma page d'accueil 'equipe.vue'.



Nous avons donc sur cette page un composant 'planning.vue' entouré en vert et un composant 'equipe.vue', entouré en rouge. On repère le header et le footer de la page, mais, en raison de la façon particulière d'inertia.js de traiter les headers et les footers, j'aborderais leur cas plus en détail dans la partie dédiée.

D'un point de vue du code en lui-même, l'insertion du composant 'enfant' dans son composant enfant se fera ainsi.

```
<script>
import MonkeyLayout from "@/Layouts/MonkeyLayout.vue";
import planning from "../Planning.vue";
export default{
  layout: MonkeyLayout,
  components:{
    planning,
  },
}
```

Dans un premier temps, on importera le composant en haut de la balise script avant de le déclarer dans l'option 'components', en lui donnant un nom qui servira à appeler le composant dans la page du composant. J'ai décidé de déclarer le composant en local, ce qui signifie que le composant sera uniquement disponible dans les composants où il est déclaré. Il est aussi possible de déclarer les composants de manière globale afin de les rendre disponibles à l'échelle de l'application entière, mais j'ai

décidé de procéder localement afin de maintenir une plus grande clarté et maintenabilité de l'application dans le futur.

```
<section>
<planning :evenements="evenements"></planning>
<div class="mainEquipe">
```

Une fois déclaré, il sera possible d'appeler le composant dans la page, il faudra alors 'bind' les data au composant pour qu'il puisse les utiliser, les données étant, en effet, passées au composant 'parent' il faut utiliser la syntaxe ci-dessus pour transmettre les données au composant 'enfant' afin qu'il puisse les utiliser. :evenements sera alors le nom du prop dans le composant enfant, et 'evenements' le nom du prop correspondant dans le composant parent. On notera que la syntaxe ci-dessus est la version short-hand (raccourcie) de v-bind :evenements= 'evenements'.

C'est grâce à ce système de 'nested' composants que j'ai pu concevoir et créer différentes pages, constituées de différents composants, chacun répondant à une tâche précise, mais il faut maintenant expliquer le fonctionnement d'un template vue.js afin de mieux comprendre le fonctionnement du frontend du projet.

2 – Code et fonctionnalités de vue.js

[Annexe 9 : description des directives vue.js et illustration via un des composants du site.](#)

3 – Mise en place d'un design responsive

Il a ensuite fallu s'atteler à la tâche de concevoir aussi un design mobile, au moment du maquettage, la version mobile avait été conçue plus ou moins en même temps que la version desktop. Cependant, au moment de coder les pages, je me suis efforcé de procéder en 'mobile first', c'est-à-dire en codant d'abord la version mobile, puis en l'adaptant aux différentes tailles d'écran alors que l'on passait sur des versions desktop.

Il faut noter que, normalement, le 'mobile first' commence dès le maquettage, en se concentrant sur des écrans plus petits, on se concentre sur l'essentiel, et on 'augmente' l'expérience utilisateur au fur et à mesure de la place gagnée. Dans le cadre de mon projet, le design mobile et desktop s'est plus ou moins fait en parallèle, mais en raison du fait que les pages ne contenaient qu'un nombre réduit d'information. Je n'ai pas réellement dû faire face à un problème de sélection des éléments à afficher. Même si ce problème surviendra éventuellement dans le futur

Concernant l'adaptation des visuels, je me suis principalement aidé de la propriété 'display' et la valeur 'flex', comme on peut le voir dans l'exemple ci-dessous :

Le fonctionnement de la 'flexbox' permet en CSS, de contrôler l'alignement et la distribution d'un élément ou d'un set d'élément.

Par exemple, dans l'exemple ci-dessous, la 'section' balise sémantique utilisée pour signifier une portion signifiante et que j'utilise pour encapsuler le contenu principal avec le planning en 'aside'. Est en display

flex, ce qui me permet, lorsque la page est en mode desktop, de définir la propriété flex direction en 'row', c'est à dire horizontalement de gauche à droite.

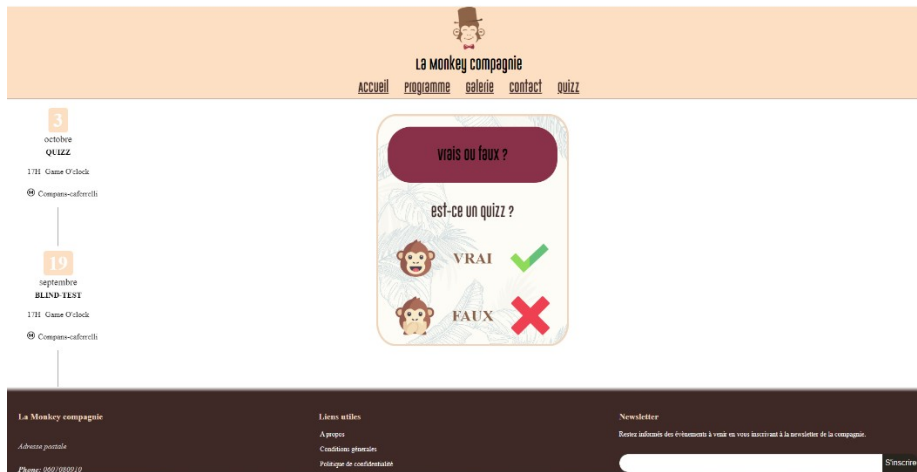


Figure 1 : version desktop

Tandis que pour le mode mobile, cette propriété sera définie en 'column', c'est-à-dire verticalement de haut en bas, de plus, en version mobile, le display même du 'aside' sera placé en flex, et la flex direction passée en 'row' afin que les différents éléments constitutifs du planning s'affichent en ligne horizontale.

Ce système de fonctionnement, couplé à l'utilisation de valeurs relatives comme 'rem', qui calculera la taille d'un élément par rapport à l'élément root, c'est-à-dire généralement 'html'. Ou 'em' qui calculera la taille de l'élément spécifié, par rapport à son élément parent.



Figure 2 : version mobile Figure 3 : version tablette

Les changements de propriétés en fonction de la taille de l'écran seront décidés par des 'media query', qui conditionneront les propriétés à la taille de l'écran. Comme dans l'exemple suivant.

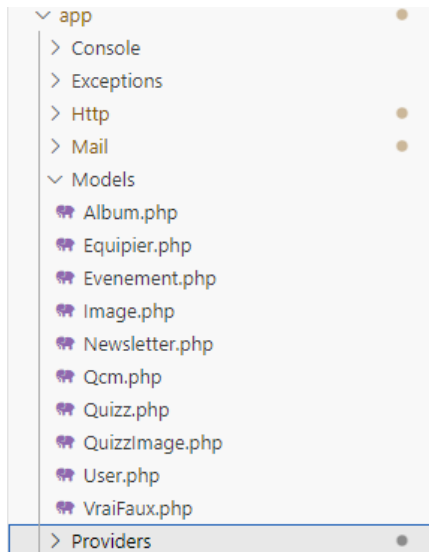
```
@media screen and (min-width: 768px){
  section{
    display: flex;
    position: relative;
    flex-direction: row;
  }
  1 reference
  .mainQuizz{
    width:40%;
    height: auto;
    margin-top: calc(var(--height-header) + 2rem);
    text-align: start;
    margin-left: auto;
    margin-right: auto;
  }
}
```

Figure 4: les règles s'appliquant avec une taille d'écran supérieure à 768px

IV - Mise en place d'une base de données

I – Création des models

Il a donc fallu dans un premier temps créer les models nécessaires pour stocker les données et les afficher sur le site. Ces modèles seront trouvables dans le dossier 'app/Models'.



Laravel, comme j'ai pu l'expliquer dans ma partie diagramme de classe, à une manière particulière de construire les modèles, notamment en déclarant la propriété 'id' dans la classe modèle dont hériteront les différents modèles. Laravel adoptant le même fonctionnement avec les getters et les setters de chaque classe.

```
/**
 * The primary key for the model.
 *
 * @var string
 */
protected $primaryKey = 'id';

/**
 * The "type" of the primary key ID.
 *
 * @var string
 */
protected $keyType = 'int';

/**
 * Indicates if the IDs are auto-incrementing.
 *
 * @var bool
 */
public $incrementing = true;
```

Figure 5 : déclaration de l'id dans la classe Model

```

/**
 * Dynamically retrieve attributes on the model.
 *
 * @param string $key
 * @return mixed
 */
public function __get($key)
{
    return $this->getAttribute($key);
}

/**
 * Dynamically set attributes on the model.
 *
 * @param string $key
 * @param mixed $value
 * @return void
 */
public function __set($key, $value)
{
    $this->setAttribute($key, $value);
}

```

Figure 6 : Déclaration des getters et des setters

Le fichier de création d'une entité, en Laravel, ressemblera alors à ceci :

```

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use App\Models\Event;
use App\Models\Image;

class Equipier extends Model
{
    use HasFactory;

    protected $fillable = ['nom', 'description', 'portrait'];
}

```

Les propriétés protégées 'fillable' vont désigner les propriétés qui pourront être remplies à l'aide de la méthode 'create' ou modifiée via 'update', lors de la création d'un nouvel élément. Définissant quelles propriétés sont remplitables/modifiables, via la méthode 'fill', présente dans la classe 'Model'.

```

/**
 * Fill the model with an array of attributes.
 *
 * @param array $attributes
 * @return $this
 *
 * @throws \Illuminate\Database\Eloquent\MassAssignmentException
 */
public function fill(array $attributes)
{
    $totallyGuarded = $this->totallyGuarded();

    $fillable = $this->fillableFromArray($attributes);

    foreach ($fillable as $key => $value) {
        // The developers may choose to place some attributes in the "fillable" array
        // which means only those attributes may be set through mass assignment to
        // the model, and all others will just get ignored for security reasons.
        if ($this->isFillable($key)) {
            $this->setAttribute($key, $value);
        } elseif ($totallyGuarded || static::preventsSilentlyDiscardingAttributes()) {
            if (isset(static::$discardedAttributeViolationCallback)) {
                call_user_func(static::$discardedAttributeViolationCallback, $this, [$key]);
            } else {
                throw new MassAssignmentException(sprintf(
                    'Add [%s] to fillable property to allow mass assignment on [%s].',
                    $key, get_class($this)
                ));
            }
        }
    }
}

```

Méthode qui sera appelée lors de l'exécution de la méthode magique `__construct`, présente dans la classe Modèle. Une 'méthode magique' signifie une méthode qui sera appelée lors d'une situation spécifique, la méthode 'construct' étant appelée à chaque instantiation d'un nouvel objet. On remarque à ce sujet que `__get` et `__set` sont aussi des méthodes magiques, appelées lorsque l'on cherche à accéder ou modifier une propriété. En php vanilla, il existe aussi des méthodes magiques `__get` et `__set`, elles seront mobilisées lorsque l'on cherche à accéder ou modifier une propriété autrement inaccessible, par exemple car dépourvues d'accessor ou de mutator.

```

/**
 * Create a new Eloquent model instance.
 *
 * @param array $attributes
 * @return void
 */
public function __construct(array $attributes = [])
{
    $this->bootIfNotBooted();

    $this->initializeTraits();

    $this->syncOriginal();

    $this->fill($attributes);
}

```

La façon de procéder de Laravel est considérée comme une bonne pratique, car elle est partagée par de multiples ORM modernes, et facilite l'accès et la manipulation des données, cependant une telle méthode peut avoir des désavantages, comme un niveau plus bas d'encapsulation, et une perte de 'customisation du code', puisque la même méthode standard sera mobilisée pour tous les modèles.

Ci-dessous, un exemple de ce qu'aurait pu être la classe 'équipier', mais en POO classique :

```
<?php

class Equipier
{
    private $id;
    private $nom;
    private $description;
    private $image;

    public function __construct($id, $nom, $description, $image)
    {
        $this->id = $id;
        $this->nom = $nom;
        $this->description = $description;
        $this->image = $image;
    }

    public function getId()
    {
        return $this->id;
    }
    public function setId($id)
    {
        $this->id = $id;
    }

    public function getNom()
    {
        return $this->nom;
    }
    public function setNom($nom)
    {
        $this->nom = $nom;
    }
    public function getDescription()
    {
        return $this->description;
    }
    public function setDescription($description)
    {
        $this->description = $description;
    }
    public function getImage()
    {
        return $this->image;
    }
    public function setImage($image)
    {
        $this->image = $image;
    }
}
```

On remarquera les propriétés de la classe définies en 'private' et non pas en protected.

2 – Création des relations

Il fallait pour rendre la partie ‘modèle’ fonctionnelle, s’atteler à penser et définir deux types de relation, la relation many-to-many entre album et image, et la relation d’héritage simulée via une relation polymorphique one-to-one par Laravel.

Concernant la relation entre album et image, où chaque image peut faire partie de plusieurs albums et inversement. Nous allons procéder un peu comme en POO classique, c’est-à-dire avec une table pivot qui conservera chaque occurrence d’une correspondance entre une image et un album. Cependant, contrairement à de la POO classique, on ne créera pas de classe pour la table pivot, mais on déclarera une fonction dans chacune des classes, comme ceci.

```
public function images(){  
    return $this->belongsToMany(Image::class, 'album_image');  
}
```

Figure 7 : La fonction images dans la classe album

```
public function albums(){  
    return $this->belongsToMany(Album::class, 'album_image');  
}
```

Figure 8 : La fonction albums dans la classe image

La méthode ‘belongsToMany’ de la classe ‘Model’ permettra alors de renvoyer toutes les entités de la classe définie dans la méthode, par exemple ici ‘Album::class’, qui sont en relation avec l’image depuis laquelle la méthode est appelée. Le deuxième argument ici ‘album_image’, permettra de déterminer le nom de la table pivot.

Quant à la relation entre la classe ‘quizz’ et ses trois occurrences, on déclarera la fonction suivante dans la classe ‘quizz’ :

```
class Quizz extends Model  
{  
    use HasFactory;  
  
    protected $fillable = ['question', 'reponse'];  
  
    public function quizzable(): MorphTo  
    {  
        return $this->morphTo();  
    }  
}
```

Tandis que dans chaque classe telle que ‘QCM’, par exemple, on déclarera cette fonction :

```

class Qcm extends Model
{
    use HasFactory;

    protected $fillable = ['choix1','choix2','choix3','choix4'];

    public function quizz(): MorphOne
    {
        return $this->morphOne('App\Models\Quizz', 'quizzable');
    }

    public function getMorphClass()
    {
        return 'QCM';
    }
}

```

Ce que nous faisons ici, c'est relier les deux entités entre elles, et donc, par exemple, de pouvoir accéder aux 'choix' de l'entité QCM via 'quizz', grâce à une syntaxe du type 'quizz.quizzable.choix1'. La différence avec une relation one-to-one classique étant que l'on aura, dans l'entité 'quizz' deux propriétés supplémentaires 'quizzable_id' et 'quizzable_type', générés automatiquement par Laravel. La fonction 'getMorphClass' quant à elle permettra qu'apparaisse dans la colonne 'quizzable_type' le nom de la classe 'QCM' et non 'App/Models/Qcm' rendant plus lisibles les colonnes en base de données.

3 – Création des fichiers de migration

Une fois les modèles définis et les relations codées, il faut créer les différents fichiers de migration afin de créer les tables correspondantes en bas de donnée.

Pour continuer sur le model 'quizz'

```

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('quizzes', function (Blueprint $table) {
            $table->id();
            $table->text('question');
            $table->text('reponse');
            $table->morphs('quizzable');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('quizzes');
    }
};

```

On déclarera dans ce fichier les différentes colonnes nécessaires au stockage des données en BDD, en leur donnant un nom et un type, morphs() permettant de générer automatique les colonnes quizzable_type et quizzable_id, agissant comme une clé étrangère, rendant possible l'accès au qcm, vrai/faux.. Depuis l'entité quizz. C'est donc une manière de procéder alternative plutôt que de stocker

qcm_id, vraiFaux_id.., comme on le ferait si on avait simplement programmé trois relations one-to-one classiques.

Un tel fichier de migration sera généré via la commande suivante dans le CLI de laravel, et mis directement dans 'database/migration', et créant automatiquement les méthodes up and down :

```
php artisan make:migration create_quizzs_table
```

La méthode up() sera appelé via la commande CLI 'php artisan migrate' qui créera alors la table en fonction de la configuration du fichier .env. Et la méthode down() permettra de 'défaire' la migration afin d'accéder à des versions précédentes de la migration via 'php artisan migrate:rollback'.

```
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('album_image', function (Blueprint $table) {
            $table->id();
            $table->foreignId('album_id')->constrained();
            $table->foreignId('image_id')->constrained();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('album_image');
    }
};
```

Figure 9 : Le fichier de migration de la table pivot

Ci-dessus le fichier de migration de la table pivot entre album et image, foreignId()->constrained() servant à définir une clé étrangère, ainsi que la table à laquelle elle correspond via l'argument donné dans la fonction 'foreignId()'

V - Mise en place et utilisation d'inertia.js

Une fois les différentes vues codées, il a fallu les intégrer au projet Laravel, en temps normal, il aurait été nécessaire, afin de combiner les fonctionnements de mes framework frontend et backend, de créer une série d'API RESTfull, et de coder différentes fonctions afin de 'fetch', d'aller chercher les données via le serveur hébergeant le framework backend. Processus nécessaire à l'affichage des pages du site.

La librairie inertia.js va alors nous permettre de traiter les différents composants vue.js comme des templates Laravel. Le framework backend utilise en effet en temps normal un système de vues blade.vue, qui dispose, tout comme vue, d'un système de directives permettant d'afficher de différentes façons, par exemple grâce à une directive @if/@elseif, ainsi que d'un système de boucles grâce à la directive @foreach. Si de nombreuses fonctionnalités se retrouvent alors dans les deux systèmes de vues, le framework vue.js reste préférable en raison de sa gestion de la réactivité et des actualisations du DOM.

Cependant, en temps normal, il aurait été nécessaire de relier les repository back et front via une série d'API, entraînant un temps de développement plus long et l'apparition de nouvelles erreurs potentielles.

J'ai donc opté pour l'utilisation de la librairie inertia.js, le principe étant que grâce à cette 'glue', il va devenir possible de traiter les composants vue.js, comme s'ils étaient des composants blade. On 'injectera' alors les données récupérées depuis la base de données via les controllers, dans les composants et sous forme de props.

Dans un premier temps, l'installation d'inertia.js se fera de la manière suivante.

Via le CLI ci-dessous :

php artisan inertia:middleware

Une fois l'installation lancée, on pourra ajouter les directives, dans le root-template de Laravel :

```
resources > views > app.blade.php
1  <!DOCTYPE html>
2  <html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
3      <head>
4          <meta charset="utf-8">
5          <meta name="viewport" content="width=device-width, initial-scale=1">
6
7          <!-- Fonts -->
8          <link rel="preconnect" href="https://fonts.bunny.net">
9          <link href="https://fonts.bunny.net/css?family=figtree:400,500,600&display=swap" rel="stylesheet" />
10
11         <!-- Scripts -->
12         @routes
13         @vite(['resources/js/app.js'])
14         @inertiaHead
15     </head>
16     <body class="font-sans antialiased">
17         @inertia
18     </body>
19 </html>
20
```


En déclarant ainsi inertia dans le composant racine de Laravel, à savoir la première qui sera chargée par l'application, on indique que c'est ici qu'il faudra charger les assets CSS et Javascript.

On remarque au passage l'utilisation de 'vite', qui est un outil de compilation rapide front end, typiquement utilisé en duo avec vue, et qui, grâce à inertia.js sera rendu utilisable dans notre projet Laravel.

Il faudra ensuite installer les adaptateurs server-side via la commande.

'npm install @inertiajs/vue3'

Puis finalement paramétrer le main fichier JS de l'application, contenu dans 'ressources' de la façon suivante afin de pouvoir boot le Framework client-side (vue.js), au démarrage de l'application.

```
import './bootstrap';
import './css/app.css';
import { createApp, h } from 'vue'
import { createInertiaApp } from '@inertiajs/vue3';
import { resolvePageComponent } from 'laravel-vite-plugin/inertia-helpers';
import { ZiggyVue } from '../../vendor/tightenco/ziggy/dist/vue.m';

const appName = import.meta.env.VITE_APP_NAME || 'bonjour';

createInertiaApp({
  title: (title) => `${title}`,
  resolve: (name) => resolvePageComponent(`./Pages/${name}.vue`, import.meta.glob('./Pages/**/*.vue')),
  setup({ el, App, props, plugin }) {
    return createApp({ render: () => h(App, props) })
      .use(plugin)
      .use(ZiggyVue, Ziggy)
      .mount(el);
  },
  progress: {
    color: '#485563',
  },
});
```

1 – Props inertia

On se souvient que dans un composant vue classique, les données sont passées en tant que props du composant parent, qui les récupère via une API, par exemple, aux composants enfant. Lorsque l'on utilise Inertia, on passera les données aux composants directement dans le composant vue.js intégré au projet Laravel, comme j'ai pu le montrer dans le composant 'galerie'. Props qui seront des tableaux associatifs desquels on pourra afficher les valeurs en appelant les clés.

```
<template>
<aside class="planning">
  <div v-for="evenement in evenements">
    <div class="date">
      <div><span class="badge">{{ evenement.jour }}</span></div>
      <div>{{ evenement.mois }}</div>
    </div>
    <div class="planningText">
      <h4 class="text-uppercase"><strong>{{ evenement.nom }}</strong></h4>
      <ul class="list-inline">
        <li class="list-inline-item"><i class="fa fa-clock-o" aria-hidden="true"></i> {{evenement.heure}}H</li>
        <li class="list-inline-item"><i class="fa fa-location-arrow" aria-hidden="true"></i> {{ evenement.lieu }}</li>
      </ul>

      <ul class="metro-container">
        <li class="metro">{{ evenement.metro }}</li>
      </ul>
    </div>
    <div class="line"></div>
  </div>
</aside>
</template>
```

Comme ici, où il sera possible d'afficher au sein du composant 'planning', qui sera affiché sur le côté de la page dans la majorité des pages, les différents attributs du prop 'événement', en appelant les différentes clés. On remarque l'utilisation de la balise 'aside' utilisée afin d'allier le fonctionnel à la portée sémantique de la balise et de s'assurer un meilleur référencement.

2 – Layouts et header

On trouvera en [annexe 10](#) une explication détaillée du fonctionnement des layouts et des header Inertia

VI - Développement Back-end

I – mise en place du routing

Afin de relier les controllers aux vues et à la base de données, il nous faut coder les routes, elles seront créées dans le fichier 'app/routes/web.php' servant à regrouper les routes http du projet. C'est-à-dire les routes qui utiliseront des méthodes http tels que GET, POST, PUT, DELETE afin d'effectuer des requêtes sur des URI.

```
// routes statiques
Route::get('/', [EquipierController::class, 'indexPublic'])->name('home');
Route::get('/programme', [EvenementController::class, 'indexPublic'])->name('programme');
Route::get('/galerie', [AlbumController::class, 'indexPublic'])->name('galerie');
Route::get('/galerie/{id}', [AlbumController::class, 'show'])->name('album');
Route::get('/quizz', [QuizzController::class, 'indexPublic'])->name('quizz');
Route::get('/contact', function (ContactController $contactController,) {
    $evenements = $contactController->index();
    return Inertia::render('Contact', [
        'evenements'=>$evenements,
    ]);
})->name('contact');
Route::post('/contact', [ContactController::class, 'submitForm'])->name('contact.submit');
Route::resource('newsletter', NewsletterController::class)->only('store');
```

Figure 10 : Les routes statiques

Dans les routes ci-dessus, on trouvera la méthode http utilisé sur l'URI, par exemple 'get' avec '/programme', ainsi que le Controller concerné par la requête, suivie de la méthode appelée au sein de ce Controller. La plupart de ces routes fonctionnent via la méthode GET, méthode utilisée pour récupérer des données associées à une URI, sans les modifier. On remarque l'utilisation de méthodes 'indexPublic' que j'ai utilisé pour renvoyer des données vers mes pages publiques, tandis que les méthodes 'index' seront utilisées dans mon panel admin. La méthode 'show' utilisée dans le controller Album servira à récupérer un album en particulier pour afficher les images qui le composent.

La route POST servira, elle, à envoyer des données au serveur, ici le contenu du formulaire de contact. Quant à la route ressource, elle consiste à créer, via cette seule route, sept routes différentes, une pour chaque méthode http :

Verb	URI	Action	Route Name
GET	/photos	index	photos.index
GET	/photos/create	create	photos.create
POST	/photos	store	photos.store
GET	/photos/{photo}	show	photos.show
GET	/photos/{photo}/edit	edit	photos.edit
PUT/PATCH	/photos/{photo}	update	photos.update
DELETE	/photos/{photo}	destroy	photos.destroy

On remarque que chaque type de route auront la méthode, l'URI, l'action effectuée ainsi qu'un nom. Le fait de nommer une route, que l'on retrouve dans de nombreux Framework actuels, se nomme 'reverse routing' et consiste à donner un nom à une route afin de l'appeler par son nom et non pas par une URL, ce qui peut notamment être pratique au cas où l'URL serait modifiée, nous évitant de changer l'URL dans toutes les occurrences du projet.

La méthode `->only()` quant à elle permet de limiter le nombre de méthodes utilisables au sein des sept routes créées, j'ai utilisé cette manière de procéder car, même si pour l'instant la newsletter n'est pas forcément fonctionnelle, il pourrait être utile de permettre aux utilisateurs de donner leur adresse mail en vue du jour où la fonctionnalité sera prête, il faudra alors permettre d'autres méthodes, mais en attendant seule la méthode 'store' est nécessaire.

2 – Création des Controllers

Il nous faut créer les Controller pour faire le lien entre le model et la vue.

J'ai donc créé les différents Controller, en optant pour la méthode du 'ressource Controller', le principe étant de générer, via le CLI de Laravel, un controller contenant la mise en forme nécessaire à la création d'un CRUD, c'est-à-dire aux opérations de création, lecture, mise à jour et suppression, appelé via les méthodes HTTP `get`, `post`, `put`, `delete` de la route.

La commande suivante : `php artisan make:controller ImageController --resource`

Permettant alors de créer un Controller possédant les méthodes (vides) : `index`, `create`, `store`, `show`, `edit`, `update`, `destroy`. Permettant respectivement d'afficher toutes les occurrences d'un modèle, d'afficher la page de création d'un modèle, d'insérer une occurrence en base de données, d'afficher une occurrence spécifique, d'afficher la page permettant de modifier les données, modifier les données et supprimer une occurrence.

La commande CLI générera aussi les imports basiques, tels que les deux classes ci-dessous :

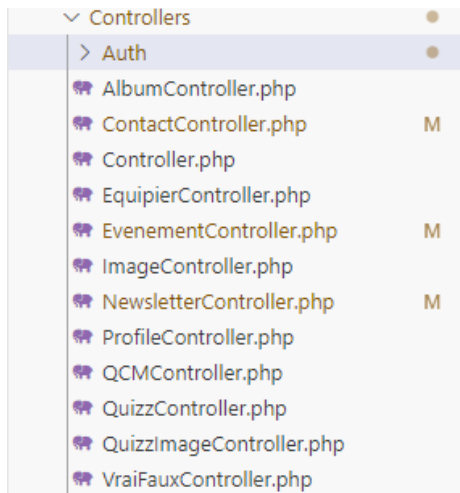
```
namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
```

La classe controller contenant elle-même les classes 'AuthorizeRequest' qui gère les différentes autorisations des users, et 'ValidatesRequests' qui gère l'utilisation des validateurs.

On remarque le namespace 'Controllers', un namespace permet de définir des répertoires abstraits au sein du projet, évitant les conflits dus à l'utilisation de deux fonctions, constantes ou classes possédant le même nom.

Via cette méthode, j'ai créé les Controller suivant :



Voici, par exemple, la méthode 'indexPublic', de mon Controller 'AlbumController', et qui me permet d'afficher la page où sont visibles les albums, ainsi que le planning sur la gauche de la page.

```
public function indexPublic(){

    // recuperation de l'album avec les images associées
    $albums = Album::select('id','titre')
    ->with(['images' => function ($query) {
        $query->select('url')
        ->first();
    }])
    ->get();

    // récupération des évènements du mois à venir
    $now = Carbon::now();
    $thirtyDaysFromNow = $now->copy()->addDays(30);
    $evenements = Evenement::where('date','>',$now)
    ->where('date','<',$thirtyDaysFromNow)
    ->orderByDesc('date')
    ->get();

    return Inertia::render('Galerie',[
        'albums' => $albums,
        'evenements' => $evenements
    ]);
}
```

Dans un premier temps, on va récupérer la totalité des albums présents en base de données, 'select' servant à sélectionner quelles colonnes on récupérera dans la table, pour éviter d'avoir recours au SELECT *. On utilisera ensuite la méthode with() permettant de récupérer pour chaque album les images lui correspondant. On fera alors une query sur 'image' afin de ne récupérer que l'url afin d'illustrer l'album, et on utilisera first pour ne récupérer qu'une seule image.

En SQL, afin de récupérer les enregistrements liés entre albums et images, j'aurais pu procéder de la façon suivante en SQL, en admettant que je parte de la table album_image :

```
SELECT images.id, images.description, images.url, album_image.album_id, album_image.image_id, albums.id, albums.titre FROM album_image
INNER JOIN albums ON album_image.album_id = albums.id
INNER JOIN images ON album_image.image_id = images.id
LIMIT 1
```

On voit alors l'intérêt des query des ORM moderne en ce qu'ils simplifient et accélèrent la masse de travail qui peut être accomplie. De plus ->first() peut être utilisé d'une manière un peu plus complexe, car on peut lui donner un 'test' en argument, et la méthode renverra alors le premier résultat à passer le test. Ce qui serait alors beaucoup plus complexe à rendre en SQL.

Concernant l'évènement, j'ai utilisé Carbon, une bibliothèque de gestion du temps en PHP, grâce à laquelle je récupère la date de l'instant T, et demande à mon Controller de ne renvoyer que les évènements dont la date n'est pas passée, et dont la date n'est pas dans plus de trente jours. Me permettant alors d'afficher suffisamment d'évènements pour montrer un planning rempli, mais éviter qu'il ne soit trop plein et perde l'internaute.

On remarque aussi que les Controller utilise une méthode particulière pour renvoyer les composant vers la vue, le return 'Inertia ::render' est la méthode utilisée pour renvoyer un composant par inertia, et est rendue possible par le middleware 'HandleInertiaRequest.php' qui sera ajouté à la liste des middleware du projet lors du setup d'inertia.

3 – Mise en place du processus d'authentification

Le site ne comporte pas d'interface visiteur/user, cependant il fallait créer un système d'authentification afin de permettre à l'administrateur de se connecter au panel admin.

La connexion se fera non pas via un bouton, mais via une route particulière à taper dans l'URL du site. J'ai choisi d'utiliser Breeze, un package implémentant les processus d'inscription et d'authentification de Laravel. Breeze fournit un ensemble de Controller et de modèles, ainsi que des routes accompagnées de view 'standard', que le développeur peut alors paramétrer à l'envie.

Dans le cadre de mon projet, je n'avais besoin que d'un enregistrement, à usage unique, pour le compte de Monsieur Scherrer, et d'un système de login lui permettant de se connecter/déconnecter à l'envie.

J'ai donc sélectionné les routes nécessaires à l'application, au sein du fichier fourni par Breeze :

```

9 use App\Http\Controllers\Auth\PasswordResetLinkController;
10 use App\Http\Controllers\Auth\RegisteredUserController;
11 use App\Http\Controllers\Auth\VerifyEmailController;
12 use Illuminate\Support\Facades\Route;
13
14 Route::middleware('guest')->group(function () {
15     Route::get('register', [RegisteredUserController::class, 'create'])
16         ->name('register');
17
18     Route::post('register', [RegisteredUserController::class, 'store']);
19
20     Route::get('login', [AuthenticatedSessionController::class, 'create'])
21         ->name('login');
22
23     Route::post('login', [AuthenticatedSessionController::class, 'store']);
24 });
25
26 Route::middleware('auth')->group(function () {
27
28     Route::post('logout', [AuthenticatedSessionController::class, 'destroy'])
29         ->name('logout');
30 });
31

```

J'ai uniquement conservé les routes permettant d'enregistrer un compte et de s'y connecter, on remarque que les routes sont situées derrière deux 'middleware', un middleware est un système de filtrage et de manipulation des requêtes, ils sont paramétrés dans le fichier 'app/http/Kernel' et permettent d'intercepter et d'effectuer un traitement sur les requêtes avant qu'elles n'atteignent la logique interne des Controllers. On trouvera en [annexe 11](#) la liste des différents middleware définis au moment du montage de l'app. Le middleware 'guest' redirigera les requêtes si l'utilisateur essaye d'y accéder lorsqu'il est connecté, tandis que le middleware 'auth' agira à l'inverse et redirigera les requêtes si l'utilisateur a besoin de se connecter pour accéder à ces fonctionnalités.

Je créerais donc un compte au début du site, via le Controller 'RegisteredUserController' fournie par Breeze :

```

/**
 * Handle an incoming registration request.
 *
 * @throws \Illuminate\Validation\ValidationException
 */
public function store(Request $request): RedirectResponse
{
    $request->validate([
        'name' => 'required|string|max:255',
        'email' => 'required|string|email|max:255|unique:'.User::class,
        'password' => ['required', 'confirmed', Rules\Password::defaults()],
    ]);

    $userCount = User::count();
    if ($userCount > 0) {
        return redirect('/');
    } else {
        $user = User::create([
            'name' => $request->name,
            'email' => $request->email,
            'password' => Hash::make($request->password),
        ]);
    }
    event(new Registered($user));

    Auth::login($user);

    return redirect('admin');
}

```

J'ai modifié la méthode afin que l'enregistrement de l'utilisateur ne se fasse pas s'il existe déjà au moins un seul enregistrement dans la table, le site ne nécessitant qu'un seul utilisateur, on remarque le validateur de données, permettant de vérifier que les données envoyées en requête seront bien conformes au format attendu. Le validateur, ici, s'assurera que le mot de passe correspond aux attentes configurées en amont dans la classe 'Rules\Password', comme par exemple le nombre minimum de caractères ou le fait de demander des minuscules et des majuscules dans le mot de passe. L'utilisateur pourra alors être créé, en hashant le mot de passe grâce à l'algorithme de hachage de mot de passe 'bcrypt' tel que mentionné dans 'config/ hashing.php'. Il sera alors possible pour l'utilisateur, accédant à l'écran de connexion, de se connecter au compte admin afin de s'occuper du site.

```
/**
 * Handle an incoming authentication request.
 */
public function store(LoginRequest $request): RedirectResponse
{
    $request->authenticate();

    $request->session()->regenerate();

    return redirect()->intended(RouteServiceProvider::HOMEADMIN);
}
```

Ci-dessus le Contrôleur appelé par la route 'login', qui vérifiera les identifiants de connexion, et s'ils sont exacts, redirigera alors vers le panel admin.

J'ai configuré le fichier 'RouteServiceProvider.php', le service permettant à Laravel de gérer tout le système de routes. Afin d'assigner des routes à des constantes, 'HOMEADMIN' renvoyant par exemple, au panel Admin.

4 – Mise en place du panel admin

Il a donc fallu créer un panel administrateur en créant les vues du panel administrateur, et en les reliant aux différents Contrôleur, les routes étant placées sous le middleware admin afin qu'elles ne soient accessibles que pour l'utilisateur connecté.

On trouvera en [annexe 12](#), les routes admin de mon projet.

Je montrerais un exemple concret d'interface et de fonctionnement du panel admin lorsque je présenterais plus en détail ma fonctionnalité quizz, qui me permettra d'expliquer en profondeur la manière que j'ai eu de gérer, l'administration de mon CRUD par Monsieur Scherrer.

5 – Mise en place d'un formulaire de contact via Laravel

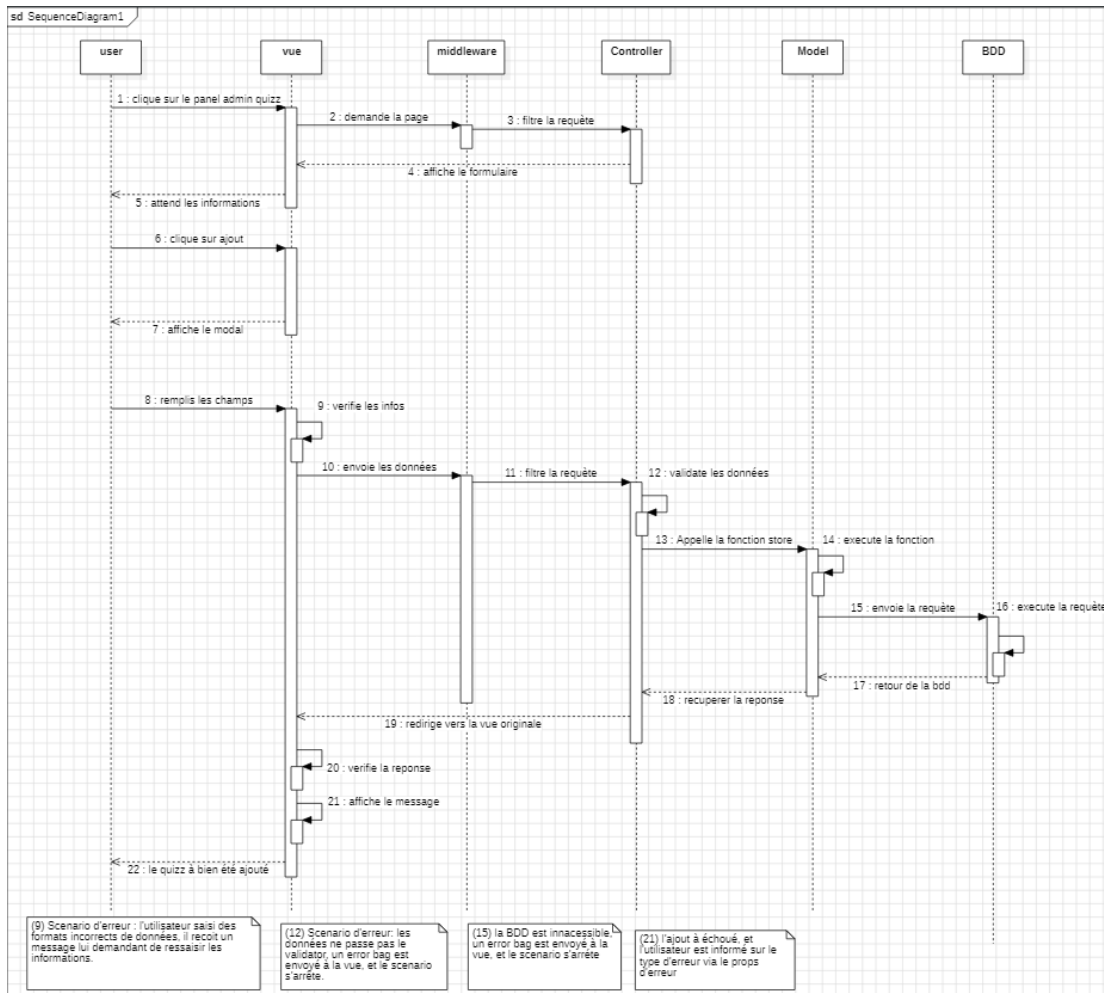
On trouvera en [annexe 13](#) la façon dont j'ai pu coder le formulaire de contact du site vitrine.

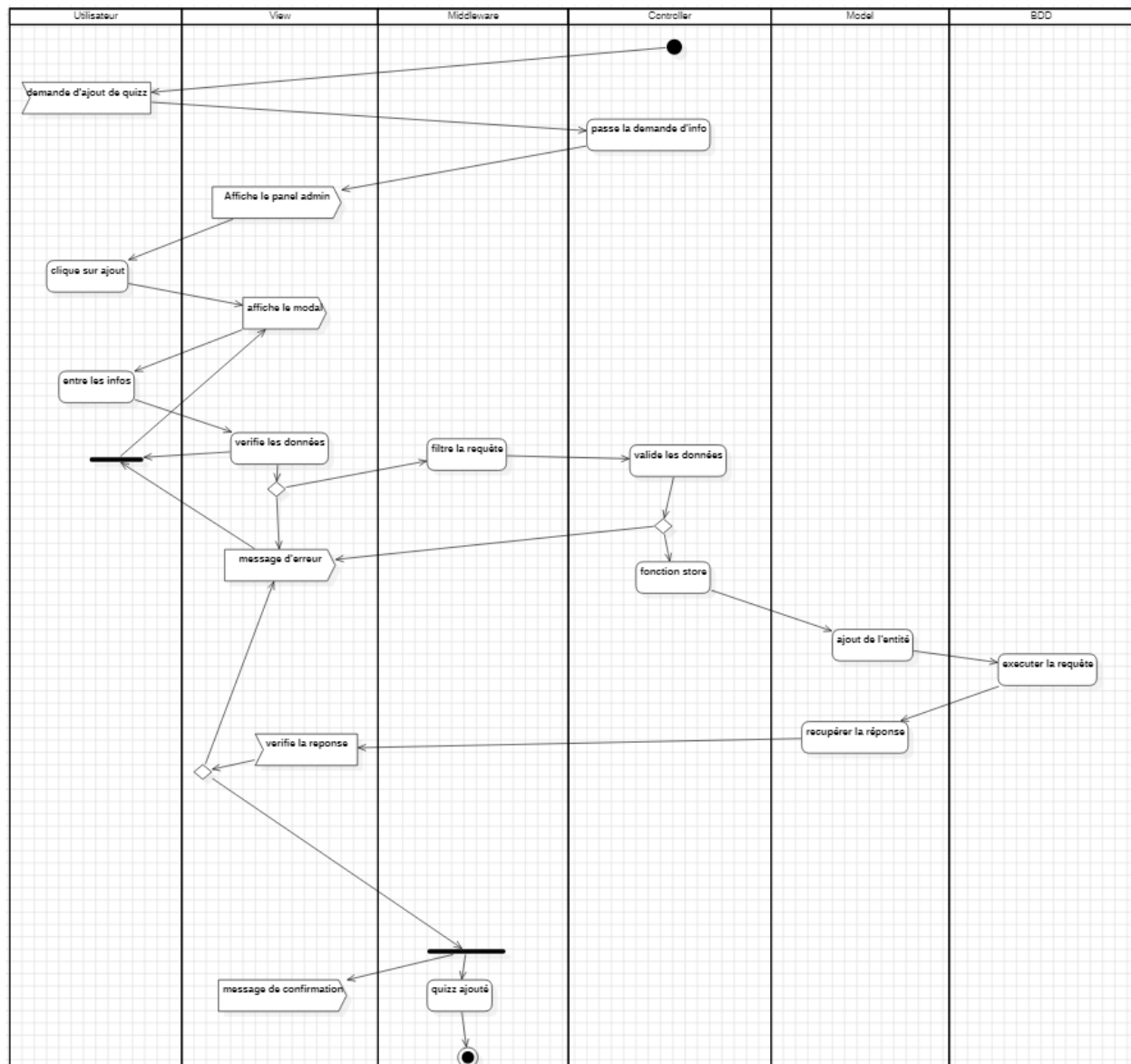
VII - Présentation d'une fonctionnalité

I - Ajouts d'un quizz

Je vais donc présenter la fonctionnalité d'ajout de quizz, qui me permettra d'illustrer le CRUD ainsi que le code du composant front et de l'utilisation de la méthode 'Inertia.post' :

Les routes 'auth' sont donc conçues pour afficher l'écran de connexion, puis pour rediriger l'utilisateur vers le 'panel admin' d'où il pourra opérer les différentes opérations CRUD.





Ci-dessus les diagrammes de séquences et d'activité, qui permettent de retranscrire le déroulement d'une activité en spécifiant les différentes possibilités d'erreur et les scénarios qui découlent de ces erreurs.

Dans le scénario où l'administrateur du site désirerait ajouter un quizz en base de données, il devra donc se connecter via le processus décrit dans la partie précédente. Une fois sur le panel admin il pourra cliquer, dans une navbar sur un lien affichant les différents quizz, lui permettant de visualiser d'un premier coup d'œil les différents quizzs créés. Cette étape est représentée dans le diagramme de séquence par les étapes 1 à 5. Et les différents quizzs sont affichés grâce à l'appel de la fonction 'index' qui retournera la vue 'AdminQuizz' avec le prop 'quizz', récupérant alors l'ensemble des trois types de quizz existant.

L'administrateur pourra alors cliquer sur les boutons 'modifier' et 'supprimer' qui apparaitront à droite de chaque quizz, permettant alors d'agir sur chaque occurrence.

Il existera aussi trois boutons 'ajouter', un pour chaque type de quizz

```
<button class="form-button" @click="openModalImage">Ajouter un Quizz image</button>  
...
```

Habituellement, on utilisera les méthodes 'create' pour la création et 'edit' pour la mise à jour, affichant des vues différentes qui seront renvoyées par ces deux méthodes. J'ai préféré utiliser un système de modal, En cliquant sur le bouton 'ajouter un quizz image' par exemple, dans le but de créer un quizzImage, on appellera la fonction 'openModallImage' avec la directive vue @click, qui permettra d'afficher un modal contenant le formulaire de création d'élément, au niveau d'un node du DOM en particulier grâce au composant 'build-in' de vue <teleport>

```
openModalImage() {  
  this.showModalImage = !this.showModalImage;  
},
```

```
<teleport to="body">  
  <div v-if="showModalImage">  
    <form class="form-container" @submit.prevent="createImage" enctype="multipart/form-data">  
      <label class="form-input" for="question">Question :</label>  
      <textarea id="question" name="question" v-model="question" required></textarea><br>  
      <label class="form-label" for="image">Image:</label>  
      <input class="form-input" type="file" id="image" name="image" accept="image/jpeg, image/png" @change="handleFileChange" required><br>  
      <label class="form-label" for="reponse">Reponse :</label>  
      <textarea class="form-select" id="reponse" name="reponse" v-model="reponse" multiple></textarea><br>  
      <input class="form-input" type="submit" value="Créer">  
    </form>  
  </div>  
</teleport>
```

Le bouton permettant de changer la variable 'showModallImage', déclarée dans les options de l'API option, tel qu'expliqué dans [l'annexe 9](#), et accessible donc via 'this.'. Ce sont les étapes 6 et 7.

Une fois le formulaire affiché, l'administrateur entrera donc une question, tel que « Quel est le pays de ce drapeau ? », une image illustrant la question, et une réponse, on remarque à cette occasion que le premier et le troisième font partie du modèle « quizz », tandis que l'image provient du « QuizzImage ».

Une première vérification s'effectuera au niveau du formulaire notamment au niveau du format du 'file' proposé, n'acceptant que les jpeg et les png. De la même façon que pour le formulaire de contact décrit dans [l'annexe 13](#), c'est inertia qui se chargera de transmettre la requête via la directive @submit.prevent, qui appellera la fonction 'createImage'.

```
createImage() {  
  const formData = new FormData();  
  formData.append('question', this.question);  
  formData.append('image', this.image);  
  formData.append('reponse', this.reponse);  
  
  this.$inertia.post('/admin/quizimage', formData, {  
    onError: (errors) => {  
      console.error(errors);  
    }  
  });  
  this.reset();  
  this.closeModalImage();  
},
```

On déclarera alors dans cette fonction un objet formData, auquel on va rajouter les différentes valeurs du formulaire, grâce à .append. Afin de récupérer les différentes variables du formulaire, on utilisera la directive v-model, qui créera un binding réactif entre un input d'un formulaire, on pensera alors au préalable à déclarer les différentes variables dans l'option API du composant.

```
data(){  
  return{  
    showModalImage:false,  
    showModalQcm:false,  
    showModalVf:false,  
    showEditQcm:false,  
    showEditImage:false,  
    showEditVf:false,  
    question:null,  
    image:null,  
    reponse:null,  
    choix1:null,  
    choix2:null,  
    choix3:null,  
    choix4:null,  
    id:null,  
  }  
},
```

Concernant l'image, on va rajouter une fonction supplémentaire :

```
handleFileChange(event) {  
    this.image = event.target.files[0];  
},
```

Nous permettant alors d'assigner le fichier à une variable 'image'.

Une fois les propriétés de l'objet assignées, on utilisera 'this.\$inertia.post' afin d'envoyer une requête en précisant la méthode du protocole, ici 'post'. On précisera alors l'URI et le formData rempli, la route ressource définie dans la série de routes admin récupèrera alors la requête, la dirigeant vers le contrôleur approprié, ici 'store' de 'QuizzImageController'.

```
public function store(Request $request)  
{  
    $data = $request->validate([  
        'question' => 'string|required',  
        'image' => 'file|required',  
        'reponse' => 'string|required'  
    ]);  
  
    if ($data['image']) {  
        $path = $data['image']->store('images', 'public');  
        $data['image'] = 'storage/' . $path;  
    }  
  
    $quizzImage = QuizzImage::create($data);  
  
    $quizz = new Quizz($data);  
  
    $quizzImage->quizz()->save($quizz);  
  
    return redirect()->back();  
}
```

La requête, ayant au préalable traversée les différents middlewares permettant de sanitize les inputs, sera validée, selon leur type et leur obligation d'avoir été définis. Dans le cas où la validation échoue, un errorBag sera envoyé à la vue, et sera directement accessible dans le composant via \$page.props.error. Ou en console comme dans mon exemple ci-dessus.

Il faudra ensuite traiter l'image, qui sera stockée dans le dossier 'public/assets/image'.

On va alors créer une instance de QuizzImage, en utilisant la query de Laravel 'create' et en lui donnant la variable \$data contenant les données validées. Laravel utilisera alors les champs filables définis dans le modèle pour se servir des données correspondantes dans \$data pour créer une nouvelle instance de QuizzImage. On répète l'opération avec l'entité Quizz et il nous suffira alors d'appeler la fonction quizz() définie dans le modèle de QuizzImage, et de lui fournir le quizz venant d'être créée pour relier les deux entités en utilisant en plus la méthode save() afin de sauvegarder l'opération.

On redirigera alors l'utilisateur vers la page d'où provient la requête, provoquant un rechargement de la page et ajoutant l'entité venant d'être créée.

Les méthodes update et delete fonctionneront de la même façon, à la différence que l'on récupérera au moment de l'affichage du modal l'id de l'élément concerné, pour savoir vers quel élément, et quel paramètre donner à l'URI de la requête update ou delete.

```
deleteImage(quizimage) {  
    if (!confirm('Supprimer le Quizz ?')) return;  
    this.$inertia.delete('/admin/quizz/' + quizimage.id)  
},
```

```
<td><button @click="deleteImage(quiz)">Supprimer</button></td>  
..
```

'Quizz' étant récupéré depuis le v-for servant à générer les différents quizzes, de la même manière que l'exemple dans [l'annexe 9](#).

On trouvera en [annexe 15](#) les méthodes des Controller visant à update et delete les entités héritant de 'Quizz'.

2 – Affichage d'un quizz interactif

Une fois les quizz créés il sera possible de les afficher dans la vue, via la méthode 'indexPublic' du Controller 'Quizz'.

```
public function indexPublic()  
{  
    $quizzes = Quizz::select('id','question','reponse','quizzable_type','quizzable_id')  
        ->with('quizzable')  
        ->get();  
  
    $shuffled = $quizzes->shuffle();  
  
    $now = Carbon::now();  
    $thirtyDaysFromNow = $now->copy()->addDays(30);  
    $evenements = Evenement::where('date','>',$now)  
        ->where('date','<',$thirtyDaysFromNow)  
        ->orderByDesc('date')  
        ->get();  
    dd($ff);  
    return Inertia::render('Quizz',[  
        'quizzes' => $shuffled,  
        'evenements' => $evenements  
    ]);  
}
```

On va donc récupérer les occurrences de la classe 'Quizz', en utilisant with('quizzable') pour récupérer les quizzes, ainsi que tous les types de quizzes associés, voici comment procède Laravel 'sous le capot'.

```

7:01:26 PM 2.12MS MySQL
SELECT `id`, `question`, `reponse`, `quizzable_type`, `quizzable_id` FROM `quizzes`

7:01:26 PM 0.38MS MySQL
SELECT
*
FROM
`quizz_images`
WHERE
`quizz_images`.`id` IN (1, 2)

7:01:26 PM 0.31MS MySQL
SELECT
*
FROM
`qcms`
WHERE
`qcms`.`id` IN (1)

7:01:26 PM 0.29MS MySQL
SELECT
*
FROM
`vrai_fauxes`
WHERE
`vrai_fauxes`.`id` IN (1)

```

On comprend alors que Laravel récupérera les instantiations des quizzes, via son attribut 'quizzable_id', et si on enlève quizzable_id de la liste des éléments sélectionnées, alors les types de quizzes seront impossible à récupérer.

On remarque aussi l'utilisation de la fonction shuffle, qui permettra de mélanger la collection de quizzes, et permettra de ne pas systématiquement afficher les quizz dans le même ordre.

```

<planning :evenements="evenements"></planning>
<div class="mainQuizz">
<div v-for="(quizz, key) in quizzes"
    :key="key"
    v-show="show === key">
<div v-if="quizz.quizzable_type === 'QuizzImage'" class="containerImageQuizz">
    <div class="titreImageQuizz">Question Image</div>
    <div class="questionImage"> {{ quizz.question }}</div>
    
    <input class="input" type="text" v-model="reponse" @keyup.enter="checkReponse">
    
</div>

```

On affichera alors les quizzes de la façon suivante, c'est-à-dire en itérant les props quizzes, et, pour chaque occurrence, en fonction de la propriété 'quizz.quizzable_type' on donnera un formatage différent à l'élément, formatage lui permettant d'afficher les props correspondant à son type.

On trouvera en [annexe](#) le template complet du composant.

Je souhaitais aussi que chaque quizz s’affiche l’un après l’autre, j’ai donc v-bind un attribut spécial de vue :key , utilisé afin de donner un index fixe à la liste rendue par le v-for, le rendant plus facile à repérer, il est conseillé dans un v-for d’attribuer une key à chaque élément. Afin d’afficher chaque élément l’un après l’autre il faudra alors déclarer une variable ‘show’, qui sera incrémentée après chaque réponse, le v-if permettant alors de n’afficher que le quizz dont la key correspond avec la variable ‘show’. On trouvera en [annexe 9](#) une illustration des directives v-if et v-for de vue.js.

```
selectionnerChoix(choix) {
  // Ajoutez le choix à la liste des choix sélectionnés
  if (!this.choixSelectionnes.includes(choix)) {
    this.choixSelectionnes.push(choix);
  } else {
    const index = this.choixSelectionnes.indexOf(choix);
    this.choixSelectionnes.splice(index, 1);
  }
  console.log(this.choixSelectionnes);
},
// A voir si elle peut passer en computed
isSelected(choix) {
  return this.choixSelectionnes.includes(choix);
},
checkReponse(selectedResponse) {
  const currentQuizz = this.quizzes[this.show];
  if (currentQuizz.quizzable_type === 'QuizzImage') {
    if (this.reponse && this.reponse.toLowerCase() === currentQuizz.reponse.toLowerCase()) {
      this.points += 1;
    }
  } else if (this.quizzes[this.show].quizzable_type === 'QCM') {
    const choixCorrect = currentQuizz.reponse.split(' ');
    const allCorrectSelected = choixCorrect.every(val => this.choixSelectionnes.includes(val));
    if (allCorrectSelected && choixCorrect.length === this.choixSelectionnes.length) {
      this.points += 1;
    }
  } else if (this.quizzes[this.show].quizzable_type === 'Vraifaux') {
    if (selectedResponse === (currentQuizz.reponse.toLowerCase() )) {
      this.points += 1;
    }
  }
  this.show++;
  console.log(this.points)
},
```

La fonction ‘checkReponse’ servira à vérifier les réponses en fonction du type de quizz. En accédant à la réponse du quizz concerné en le stockant dans une const, et le récupérant grâce à this.show, qui correspondra à la key en cours d’affichage.

S’il s’agit d’un quizz image, la réponse sera sous la forme d’une string, que le visiteur pourra taper dans un input, réponse récupérée via un v-model, qui sera passée en lowercase et comparée avec la réponse du quizz affichée, elle aussi passée en lowercase. J’ai prévenu mon administrateur que les réponses seront sensibles à la casse dues aux accents par exemple.

Dans le cas d’un qcm, les bonnes réponses seront récupérées et transformées en tableau, si les bonnes réponses sont la réponse a et la réponse b, l’administrateur devra rentrer dans le champ réponse du panel admin, ‘A B’, réponse qui sera transformée en tableau. Les réponses de l’utilisateur seront push dans le tableau ‘choixSelectionnes’ par la fonction ‘selectionnerChoix’. On vérifiera alors si chaque bonne réponse est bien contenue dans les réponses de l’utilisateur, si c’est le cas, la méthode ‘every’ renverra un booléen ‘true’. On vérifiera alors pour finir s’il y a le même nombre de réponses dans la bonne

réponse et dans les réponses sélectionnées (afin d'éviter qu'A B C D soit toujours considéré comme une bonne réponse), et on pourra alors si les deux conditions sont remplies, considérer la réponse comme valide.

Dans le cas où le quizz est un vrai ou faux, la fonction comparera la string 'vrais' ou 'faux' entrée en réponse par l'administrateur, avec la string qui sera renvoyée en fonction de la réponse sur laquelle cliquera l'utilisateur.

A la fin du quizz, il sera montré le nombre de point au joueur, accompagné d'un petit message l'invitant à se joindre aux soirées de la Monkey compagnie, message qui reste à définir avec mon client.

3- mesures de sécurité

Je vais présenter ici les protections mises en place contre les trois types de risques qui menacent le site vitrine.

Attaques XSS : Consiste à injecter du code Javascript malveillant dans l'application, afin de viser le serveur où les visiteurs du site, où de modifier le DOM de la page, notamment via un input accessible à l'utilisateur, ou via l'URL du site. Le risque est particulièrement élevé pour les sites affichant du contenu généré par les utilisateurs.

Le site vitrine n'a pas ce problème dans la mesure où tout le contenu affiché proviendra de l'administrateur. J'ai cependant pris soin de valider chaque donnée entrée dans le site via les validateurs de Laravel et j'ai rajouté un middleware permettant de sanitizez chaque requête transitant vers les Controller.

```
class XssSanitization
{
    /**
     * Handle an incoming request.
     *
     * @param \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response) $next
     */
    public function handle(Request $request, Closure $next)
    {
        $input = $request->all();
        array_walk_recursive($input, function(&$input) {
            $input = strip_tags($input);
        });
        $request->merge($input);
        return $next($request);
    }
}
```

Figure 11: Le middleware XssSanitization

Cette mesure, complétée par la validation systématique des données permettront de sécuriser même les parties du site tel que le formulaire de contact ou la newsletter. Concernant le quizz, toutes les réponses seront vérifiées uniquement dans le frontend, sans envoi vers la base de données.

Injections SQL : Consiste à injecter des requêtes SQL, par exemple dans l'URL, afin de récupérer des informations sensibles ou d'effectuer des opérations sur la base de données. Ici aussi la validation des données passées en paramètre de la requête est primordiale, en particulier de l'id qui peut être oublié au moment de la validation des données. On pourra alors procéder ainsi :

```
public function update(Request $request, string $id)
{
    $validator = Validator::make(['id' => $id], [
        'id' => 'required|numeric'
    ]);
    if ($validator->fails()) {
        abort(404);
    }
}
```

Permettant d'avorter la requête au cas où l'id revient sous un autre format que celui prévu par les algorithmes en front. On notera aussi qu'il faut éviter de rédiger des requêtes en raw, c'est-à-dire où les paramètres des requêtes sont définis en « dur ».

Attaques CSRF : Ce type d'attaque a pour but de faire accomplir une action à son insu à son utilisateur, notamment en cachant une requête dans un mail piégé.

Cependant, Inertia inclus dans chaque requête effectué avec `this.$inertia`, un token XSRF, qui sera généré par le serveur à l'ouverture de la connexion et stocké en tant que cookie sécurisé, puis inclus dans les en-tête de chaque requêtes. Le token étant alors vérifié par le serveur à chaque action effectuée par l'utilisateur connecté. Jeton qui ne pourra pas être généré par l'attaquant malveillant ni deviné. Le middleware `'\App\Http\Middleware\VerifyCsrfToken'` du groupe de routes `'web'` s'occupant de comparer les jetons de chaque requête. J'ai, en mesure supplémentaire, activé la déconnection automatique lorsque l'utilisateur ferme son navigateur.

VIII - Testing

J'ai utilisé durant la phase de testing, le framework de test cypress.js, permettant de coder des tests end-to-end, c'est-à-dire d'un bout de la fonctionnalité à l'autre. Il permet de simuler les actions d'un utilisateur afin de vérifier que le code renvoie bien les données souhaitées à partir d'un comportement et d'une utilisation typique de l'application.

J'ai donc installé cypress via la ligne de commande suivante :

```
'Npx install cypress --save -dev'
```

Et il m'a fallu ensuite configurer le framework via 'npw cypress open' me permettant d'avoir accès à l'interface du framework, qui lancera alors ses tests sur le navigateur chromium de mon choix. Cypress.js permet alors d'observer le comportement de l'application, ainsi que les différents résultats.

On rédigera les tests dans le dossier 'e2e' du dossier 'cypress' que l'on trouvera dans le projet après installation.

J'ai créé un test afin de vérifier la fonctionnalité d'ajout, modification, suppression et d'affichage du quizz, de la connexion à l'affichage de l'élément dans la page.

```
describe("quizzes tests", () =>{
  beforeEach(()=>{
    cy.visit("http://127.0.0.1:8000/admin");
  });

  it("vrai-faux", ()=>{

    cy.get('input[type="email"]').type(Cypress.env('mail'));
    cy.get('input[type="password"]').type(Cypress.env('mdp'));
    cy.get("form").submit();

    cy.url().should("eq", "http://127.0.0.1:8000/admin");
    cy.contains("Quizz").click();
    cy.get(".form-button:contains('Ajouter un Vrai ou faux')").click();
    cy.get("#question").type("Question de test vf");
    cy.get("#reponse").type("reponse de test vf");
    cy.get(".form-container").submit();
    cy.contains("Question de test vf").next().contains("reponse de test vf").should("exist");

    cy.visit("http://127.0.0.1:8000/quizz");
    cy.get('.containerVf').contains("Question de test vf").should('exist');

    cy.visit("http://127.0.0.1:8000/admin/quizz");
    cy.contains("Modifier").click();
    cy.get("#question").clear().type("question de test vf modifié");
    cy.get(".form-container").submit();
    cy.contains("question de test vf modifié").next().contains("reponse de test vf").should("exist");

    cy.visit("http://127.0.0.1:8000/quizz");
    cy.get('.containerVf').contains("question de test vf modifié").should('exist');

    cy.visit("http://127.0.0.1:8000/admin/quizz");
    cy.contains("Supprimer").click();

    cy.on('window:alert', (text) => {
      expect(text).to.equal('Supprimer le Quizz ?');

      return true;
    });

    cy.contains("Question de test vf").should("not.exist");

    cy.visit("http://127.0.0.1:8000/quizz");
    cy.get('.containerVf').should('not.exist');

  });
});
```

Le test consistera à, dans un premier temps, effectuer une visite vers le panel admin, on simulera le click et le remplissage des champs de saisie, remplis avec des variables préalablement stockées dans le fichier cypress.config.js stocké à la racine du projet. Permettant de ne pas rentrer en dur les identifiants de l'admin. Si la connexion est valide, alors l'url sera changée vers celle du panel admin.

Lors de l'ajout de quizz, on rentrera des informations dans chaque champ, et on validera l'insertion du quizz en BDD, avant de vérifier qu'il s'affiche bien sur la page du panel admin. On effectuera alors une visite sur la page '/quizz' afin de voir si le test créé apparaît bien.

On retourne ensuite sur le panel admin afin de modifier un des champs du quizz, et on revérifie si l'affichage s'est bien modifié lui aussi. Enfin, on supprimera le quizz avant de vérifier qu'il est introuvable sur la page.

Il est possible de vérifier le déroulement des tests en temps réel, via l'interface GUI de cypress, ainsi que les différentes étapes des tests, ou alors de lancer la commande 'npx cypress run' qui lancera les tests et renverra les résultats



Figure 12 : Résultats des tests dans l'interface cypress



Figure 13: résultat du test dans la ligne de commande

On trouvera en [annexe 16](#) les tests de quizz complets et les tests de connexion.

IX - Mise en production

1 – choix de l'hébergement

Le type d'hébergement n'ayant pas été fixé avec mon client, je vais présenter ici quelques solutions qui lui seront présentées et qui pourront être discutées avec lui au moment de la mise en ligne.

L'hébergement mutualisé :

Il est possible d'héberger plusieurs sites sur un même serveur, réduisant les couts mais limitant les possibilités en termes de performance, le site reste configurable grâce au cPanel, permettant de gérer les options de son serveur, cependant, le cPanel peut s'avérer limité pour des utilisateurs confirmés.

L'hébergement dédié :

Consiste à héberger un site sur un serveur lui étant dédié entièrement, permettant de le configurer entièrement, et d'éviter les problèmes de limitations de performances. Ideal pour les sites recevant un fort trafic. Mais elle est la solution la plus couteuse et les interventions techniques sont à la responsabilité et à la charge du client.

L'hébergement VPS :

Consiste à créer plusieurs serveurs virtuels sur un seul serveur réel, simulant alors plusieurs hébergements dédiés sur un même serveur. Permettant d'allier un prix raisonnable avec des performances au-delà de l'hébergement mutualisé. Tout en restant hautement paramétrable. Il nécessite néanmoins de bonnes connaissances en administration système.

L'hébergement cloud :

Repose sur une multitude de serveurs et un service à la demande, en payant uniquement ce pour quoi on à besoin, l'utilisateur s'assure du meilleur rapport qualité prix. Les données seront alors hébergées en dehors de l'entreprise, demandant à l'utilisateur de s'assurer de la confiance qu'il place en l'hébergeur.

En l'état actuel du site, il apparait que l'hébergement mutualisé reste la solution la plus simple et la moins couteuse, au regard du fait que le site ne nécessitera que peu de performance en raison de sa cible relativement réduite.

2 – Déroulement de l'hébergement

Quelle que soit la solution choisie, solution qui déterminera aussi la marche à suivre, il faudra penser à réaliser les différentes étapes.

- 'npm run build' : qui permettra de compiler les différents composants de vue.js, et de les minifier, autant d'opérations de regroupement et d'optimisation des différents composants de l'application afin de la rendre prête au déploiement et interprétable pour les différents navigateurs. J'ai notamment utilisé Vite dans le but d'accélérer le processus.

- 'php artisan storage link' : permettant de créer un lien symbolique entre le répertoire de stockage et le répertoire public, permettant de rendre possible d'accès les images stockées dans le projet via des url publiques.
- 'php artisan cache:clear' : permettra de s'assurer de mettre le site en production en ayant vidé le cache, afin de s'assurer de ne pas créer de bugs avec des données obsolètes et de ne pas éventuellement compromettre des données sensibles.

Il faudra aussi mettre le dossier `env` à jour, en précisant les valeurs relatives à la base de données mises à disposition par l'hébergeur, afin que la migration soit rendue possible, ainsi que les données relatives au service mail du serveur, pour gérer les différents services d'envoi de mail.

3 – La solution SSR

Je compte proposer à mon client de configurer le projet afin qu'il soit compatible avec du Server Side Rendering. Dans le cadre d'une application utilisant comme stack, Laravel – inertia – vue, le rendu SSR signifiera de rendre le contenu des balises html directement coté serveur, et de les renvoyer ainsi pré-rendues en front, cet HTML sera alors « hydraté » c'est-à-dire rempli par les props récupérés grâce aux Controller. Contrairement au fonctionnement classique où la page sera rendue client-side. C'est en fait un fonctionnement dit « isomorphe » dans le sens où c'est un fonctionnement reposant à la fois sur le back et le front-end.

L'intérêt étant que, si les robots google savent et peuvent à présent lire le contenu d'une page même lorsqu'elle est rendue client-side, mais les props restent en dehors de leur champ de lecture, or, grâce au SSR, le processus d'hydratation permettra au props d'être lus par le robots, ce qui s'avère utile dans la mesure où ils sont susceptibles, par exemple de contenir des mots clés.

Il faudra pour cela installer le SSR grâce à un fichier `ssr.js` fonctionnant en duo avec `app.js`, et configurer le `package.json` pour qu'il compile les fichiers js en vue du processus.

Le problème étant que le SSR nécessite Node.js installé sur le projet, il faudra donc un type d'hébergement permettant ce type de configuration.

En cherchant on se rend compte que Hostinger permet de telles configurations, mais seulement dans sa forme VPS, pour environ 5 euros par mois, un prix acceptable pour mon client.

Pour le même prix, il sera aussi possible de prendre un plan d'hébergement mutualisé chez PlanetHoster, qui rend compatible ce type d'hébergement avec l'installation de Node.js.

Le choix final se faisant alors entre ces deux possibilités.

Conclusion :

A l'issue de ce mémoire, je me rends à la fois compte des progrès effectués, autant d'un point de vue de la maîtrise des différentes technologies que de la compréhension du métier de développeur concepteur. Cependant je réalise à la fois le chemin qu'il reste à parcourir, que ce soit du point de vue des fonctionnalités et des services qu'il reste à ajouter au projet de Monsieur Scherrer, que des connaissances encore à acquérir pour remplir mes objectifs.

Objectifs qui seront, dans l'idéal, d'acquérir suffisamment d'expérience afin de pouvoir mener une activité de développeur-concepteur en freelance. Il me faudra alors encore acquérir de l'expérience pour mener à bien ce projet, et je compte, après cette formation, rechercher un emploi dans une agence web, ou toute entreprise de taille humaine, afin de pouvoir développer mes compétences. Particulièrement dans le domaine où je me sens le plus à l'aise, le back-end.

En résumé, même si l'acquisition de connaissance, tout comme la réalisation du site de la Monkey compagnie restent des travaux à poursuivre et à parfaire, je suis satisfait de pouvoir continuer ce travail grâce au socle de connaissance que j'ai pu acquérir grâce à cette formation et au cours de mon stage.

Annexes :

1 - Analyse de la concurrence

Une veille documentaire a été effectuée afin de cerner la façon dont les autres entreprises d'évènementiel opéraient sur le marché Toulousain. La plupart de ces entreprises ne partagent pas la taille modeste de l'entreprise de Monsieur Scherrer, il faut donc garder en tête le fait d'adapter le projet dont il est ici question aux capacités de la Monkey compagnie, cependant il est possible de remarquer divers éléments.

Tout d'abord une communication qui passe en grande partie par le visuel, les galeries photo étant le plus possible mises en avant afin de fournir un support visuel à la communication de ces entreprises. Il est aussi possible d'observer des images permettant de voir les visages des équipes travaillant dans ces entreprises.

On remarque aussi sur les sites concurrents une facilitation de la mise en contact avec les organisateurs d'évènements, ainsi que la mise en avant des retours (positifs) de personnes ayant pu bénéficier de ces évènements. Et ce, via l'affichage d'avis Facebook, récupérés et affichés sur les pages du site.

On remarque au final que de nombreux éléments demandés dans un premier par Monsieur Scherrer sont en effet fréquemment présents sur les sites d'évènementiels. Tels que la mise en avant d'images illustrant l'activité ou l'équipe de l'entreprise. On remarquera aussi la présence systématique d'un texte de présentation, insistant sur les qualités des entreprises, le tout porté par un ton convivial mais respectant une certaine distance.

Le but recherché sera donc de reprendre ces différents composants repérés chez la concurrence afin de les implémenter de façon efficace au site, et d'une façon qui corresponde à la taille humaine de la Monkey Compagnie.

2 - Code SQL de la création de la base de données :

```
CREATE DATABASE monkeySQL;

USE monkeySQL;

CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255),
    email VARCHAR(255) UNIQUE,
    password VARCHAR(255)
) ENGINE=InnoDB;

CREATE TABLE newsletters (
    id INT AUTO_INCREMENT PRIMARY KEY,
    mail TEXT,
    statut TINYINT(1)
) ENGINE=InnoDB;

CREATE TABLE images (
    id INT AUTO_INCREMENT PRIMARY KEY,
    url TEXT,
    description TEXT NULL
) ENGINE=InnoDB;

CREATE TABLE equipiers (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom TEXT,
    description TEXT,
    image TEXT
) ENGINE=InnoDB;
```

```

CREATE TABLE evenements (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom TEXT,
    jour INT,
    mois TEXT,
    heure TEXT,
    lieu TEXT,
    metro TEXT,
    date DATE,
    image TEXT
) ENGINE=InnoDB;

CREATE TABLE albums (
    id INT AUTO_INCREMENT PRIMARY KEY,
    titre TEXT
) ENGINE=InnoDB;

CREATE TABLE quizzes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    question TEXT,
    reponse TEXT,
    quizzable_id INT,
    quizzable_type VARCHAR(255)
) ENGINE=InnoDB;

CREATE TABLE qcms (
    id INT AUTO_INCREMENT PRIMARY KEY,
    choix1 TEXT,
    choix2 TEXT,
    choix3 TEXT,
    choix4 TEXT,
    quizz_id INT,
    FOREIGN KEY (quizz_id) REFERENCES quizzes(id)
) ENGINE=InnoDB;

```

```

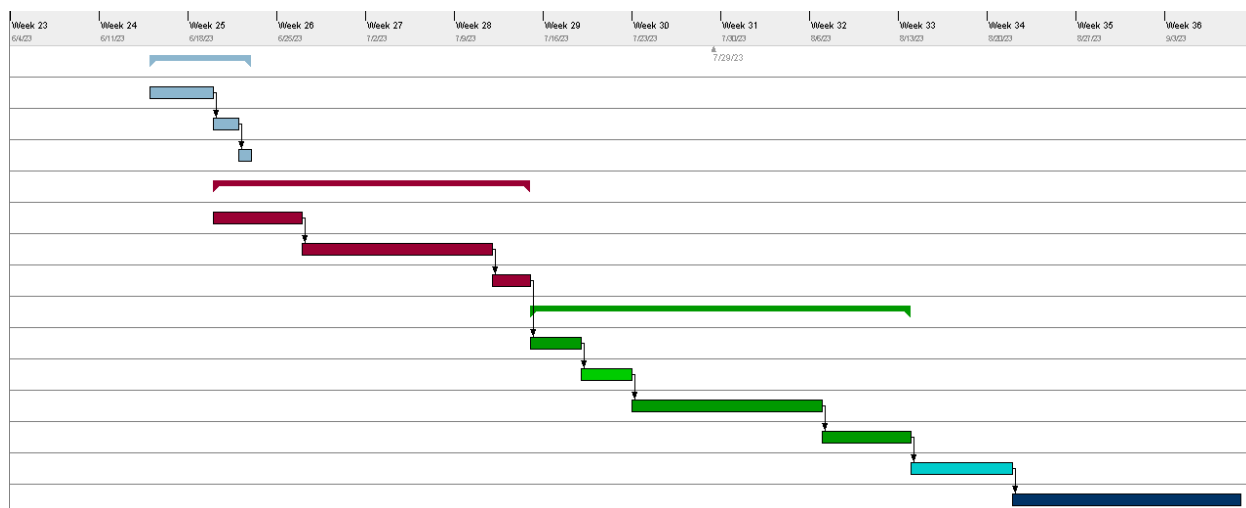
CREATE TABLE vrai_fauxes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    quizz_id INT,
    FOREIGN KEY (quizz_id) REFERENCES quizzes(id)
) ENGINE=InnoDB;

CREATE TABLE quizz_images (
    id INT AUTO_INCREMENT PRIMARY KEY,
    image TEXT,
    quizz_id INT,
    FOREIGN KEY (quizz_id) REFERENCES quizzes(id)
) ENGINE=InnoDB;

|
CREATE TABLE album_image (
    id INT AUTO_INCREMENT PRIMARY KEY,
    album_id INT,
    image_id INT,
    FOREIGN KEY (album_id) REFERENCES albums(id),
    FOREIGN KEY (image_id) REFERENCES images(id)
) ENGINE=InnoDB;

```

3 – Diagramme de Gantt



Il a été dans un premier temps nécessaire de produire un diagramme de GANT afin de visualiser de manière efficace le déroulement du projet. Les dates exactes des différentes phases, ainsi que leur dénomination précise se trouvera dans la page suivante de ce cahier des charges.

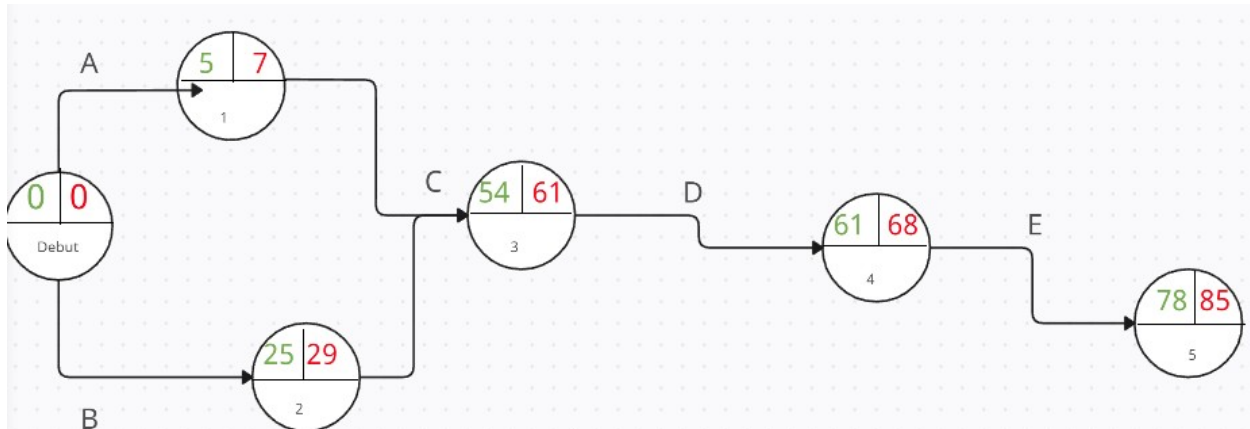
Dans un premier temps, on remarquera, avec la partie bleu ciel, que la réflexion autour du cahier des charges, et sa conception commencera un peu avant le début du stage, commencée au cours de petites réunions informelles avant le début officiel du stage, elle précède le maquetage du site, qui sera donc fait en collaboration avec Madame Garcia. On remarque un chevauchement entre les deux dernières étapes du cahier des charges (rédaction et ajustement), et le maquetage, puisque, comme je l'ai déjà mentionné, plusieurs versions ont été proposées au client, qui les a progressivement modifiées afin qu'il soit satisfait. Il a donc parfois fallu réécrire de courtes parties du cahier des charges pour coller de manière plus précise avec le design proposé. Une fois le maquetage fixé, il a donc été possible de commencer à coder le frontend, étape suivie par l'intégration des composants ainsi codés dans le projet laravel inertia. J'ai alors commencé à coder le backend, découpé ici en quatre grandes étapes, les models, la migration, le code des controllers ainsi que le renvoi des vues grâce aux routes coté serveur et enfin, la construction d'un panel admin ainsi que d'un système d'authentification. Il faut noter que même si ces étapes sont ici séparées et montrées comme successives, une partie du travail des controllers s'est effectué lors du développement du panel admin. De même, certains models ont dû être modifiés au moment de la création du panel admin. Ce qu'il faut retenir, c'est bien que le développement du backend a plus constitué une réflexion globale et semée d'essais parfois infructueux, qu'une suite de tâches méthodiquement effectuées les unes après les autres.

On notera aussi, que, si le développement s'achève ici avec la mise en production le 8 septembre, cette dernière étape continuera en réalité jusqu'à fin octobre, date à laquelle Monsieur Scherrer a exprimé avoir besoin que le site soit mis en ligne, et si le stage en lui-même se termine le 8 septembre, la mise en production n'aura pas atteint son but et sera alors à ce moment-là, encore en cours de travail.

Une fois le site apparemment fonctionnel, j'ai pu coder les différents tests cypress.js, avant de commencer les premiers tests de mise en production, notamment grâce à l'espace d'hébergement hostinger mis à disposition de la promotion par monsieur Mithridate.

▼ cahier des charges	6/15/23	6/22/23
Conception cahier des ch...	6/15/23	6/19/23
redaction	6/20/23	6/21/23
Retour sur CDC/ajustement	6/22/23	6/22/23
▼ Front	6/20/23	7/14/23
Maquettage	6/20/23	6/26/23
code front	6/27/23	7/11/23
Integration des composants	7/12/23	7/14/23
▼ Back	7/15/23	8/13/23
models	7/15/23	7/18/23
migration	7/19/23	7/22/23
controller/routing	7/23/23	8/6/23
panel admin/authenticat...	8/7/23	8/13/23
testing	8/14/23	8/21/23
mise en production	8/22/23	9/8/23

4 – Diagramme de PERT et note de cadrage



Le diagramme de PERT ci-dessus représente peu ou prou le même déroulement des événements que le diagramme de GANT, cependant, sont mentionnés ici les délais idéaux, ainsi que les délais maximums que le projet sera en capacité de supporter.

On remarquera qu'il existe alors une fourchette d'une semaine, entre les deux types de délais, et que ce temps pourra être mobilisé si le besoin de revenir sur des éléments ou des fonctionnalités se faisait sentir.

Je suis parti du principe que les endroits où je pourrais possiblement gagner du temps seront les domaines où j'ai déjà un peu d'expérience, tel que le frontend et backend, tandis que, j'ai préféré ne pas considérer que je pourrais prendre de l'avance sur les sujets que je n'avais pas rencontré en situation, tels que le testing ou l'hébergement.

On remarque aussi, que, tout comme dans le diagramme de GANT, le début du projet sera marqué par un double travail, à la fois de conception, et aussi de design, deux tâches qui seront donc conduite en parallèle au début du projet.

denomination:	phases :	date debut :	date fin :	ressources internes J/H	budget :
A	Conception :	15-Jun	22-Jun		2400
A1	conception cahier des charges	15-Jun	19-Jun	1	
A2	redaction	20-Jun	21-Jun	1	
A3	ajustements	22-Jun	22-Jun	1	
B	Front	20-Jun	14-Jul		7200
B1	maquettage	20-Jun	26-Jun	1	
B2	code Front	27-Jun	11-Jul	1	
B4	integration vue.js	12-Jul	14-Jul	1	
C	Back	15-Jul	13-Aug		8700
C1	models	15-Jul	18-Jul	1	
C2	migration	19-Jul	22-Jul	1	
C3	controllers/routing	23-Jul	06-Aug	1	
C4	panel admin/auth	07-Aug	13-Aug	1	
D	Testing	14-Aug	21-Aug	1	2100
E	Mise en production	22-Aug	08-Sep		5100
	Budget total				25500

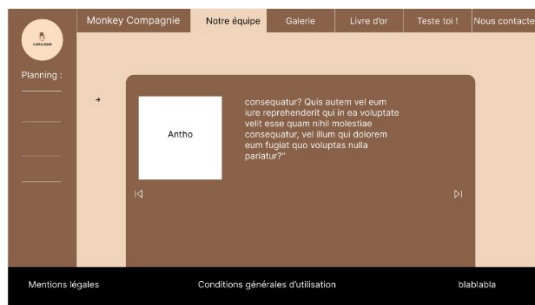
Dans la note de cadrage ci-contre, il a été effectué une simulation de devis, évaluant à 25 500 euros le prix fictif du développement du site. En raison du fait qu'il s'agisse d'un stage non rémunéré, il ne sera pas demandé à Monsieur Scherrer de payer pour le site, mais il s'agit donc d'une estimation, prenant pour cout J/H la somme de 300 euros, standard pour un développeur junior.

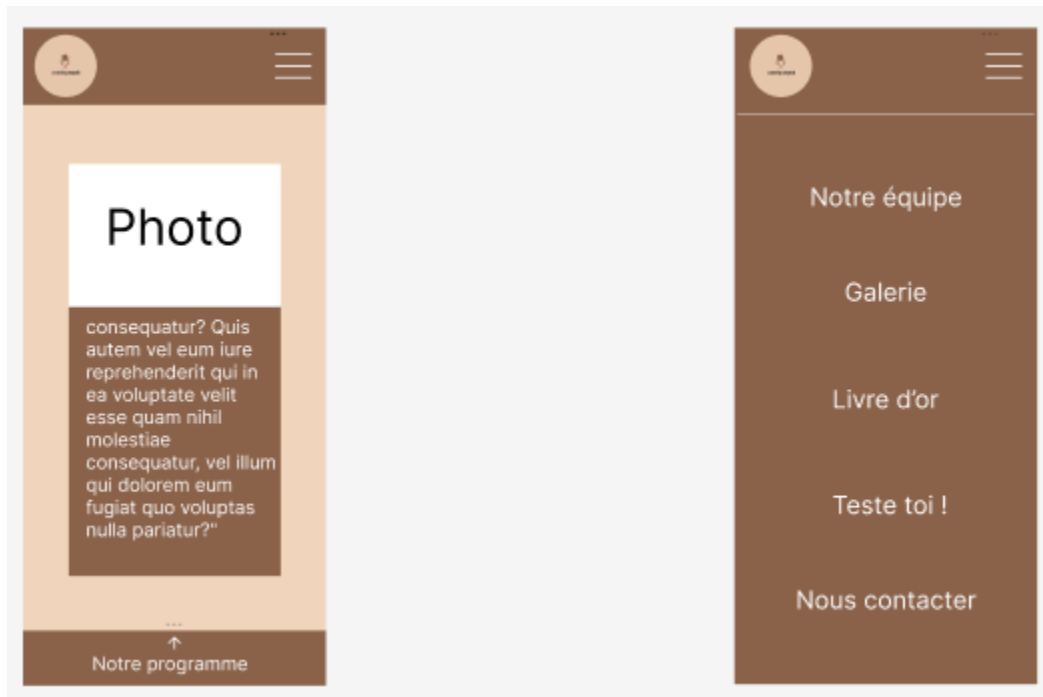
Je n'ai pas non plus inclus les frais que le client devrait théoriquement verser à Madame Garcia durant les 7 jours où elle travaillera sur la maquette, somme que l'on peut évaluer aux environs de 1750 euros, en raison du tarif moyen autour de 250 euros par jours pour un graphiste débutant. On notera aussi que j'ai considéré que je travaillais 7 jours sur 7, en raison du fait que j'accomplissais chaque jour, a minima, un travail de documentation.

5 – Première version du mockup du site

Suite à la conception des premiers wireframes, j'ai commencé à produire des mockups afin de pouvoir montrer plus en détail le design possible de son site à mon client. Tout en intégrant la palette de couleurs prévues précédemment. Ce mockup a été conçu dans un premier temps, avant que je commence à travailler en duo avec Madame Garcia. On y remarque aussi des reliquats de pages qui, à l'heure actuelle, ne sont plus prévues pour apparaître sur le site. Tel que la page « livre d'or » qui devait contenir les commentaires des participants au soirées, fonctionnalité ayant été transformée pour prendre la forme de l'affichage des commentaires via les posts sur la page Facebook de la Monkey Compagnie. La fonctionnalité ayant alors été délayée jusqu'à la création de la page Facebook de la Monkey Compagnie.

Finalement, suite au début de la collaboration avec Madame Garcia, il a été décidé de retravailler ce design, afin de lui offrir une autre identité plus épurée.





6- maquettage des différentes pages du site

Page programme :

La page programme sera la seule où l'on ne retrouvera pas la disposition avec un programme réduit sur la gauche de l'écran, et sera entièrement dédiée à l'affichage de celui-ci. On trouvera, affiché en plus sur cette page, l'affiche associée à l'événement, à droite du composant.

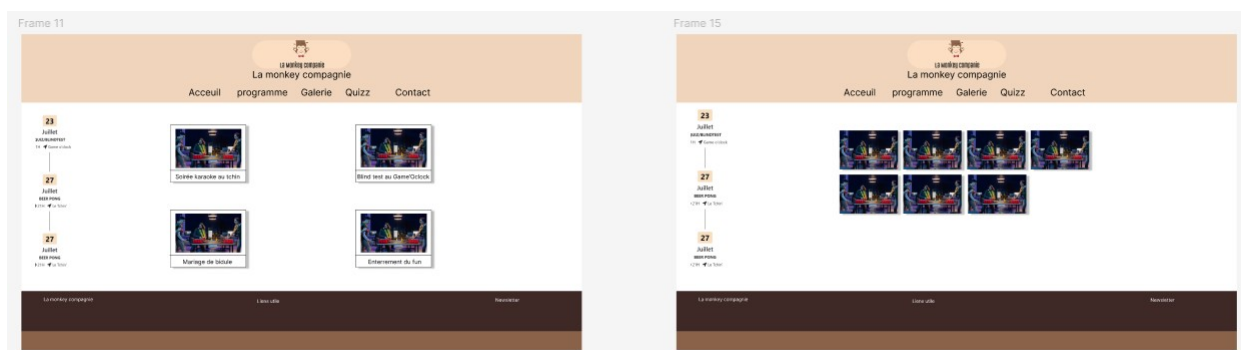
Dans sa version mobile, les événements seront représentés non pas de manière horizontale mais verticale, pour profiter au mieux de la disposition de l'écran.

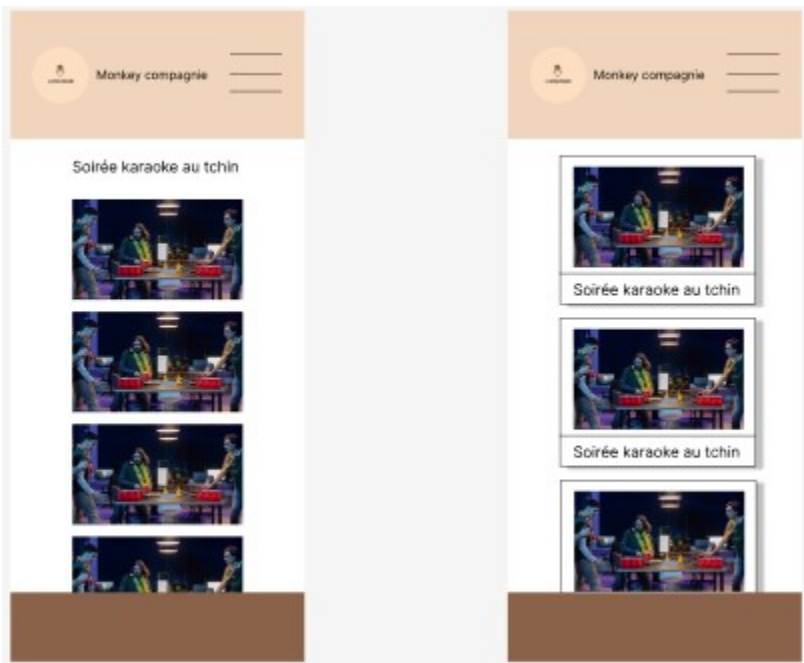


Page galerie :

La galerie proposera d'observer les différentes photographies prises au cours des différents événements de la Monkey Compagnie, on aura sur la première page les albums affichés avec le titre leur étant associé, illustrée par une des images présente dans l'album. La description facultative pouvant être affichée en passant la souris sur les différentes images, affichant alors la description en fade-in sur l'image.

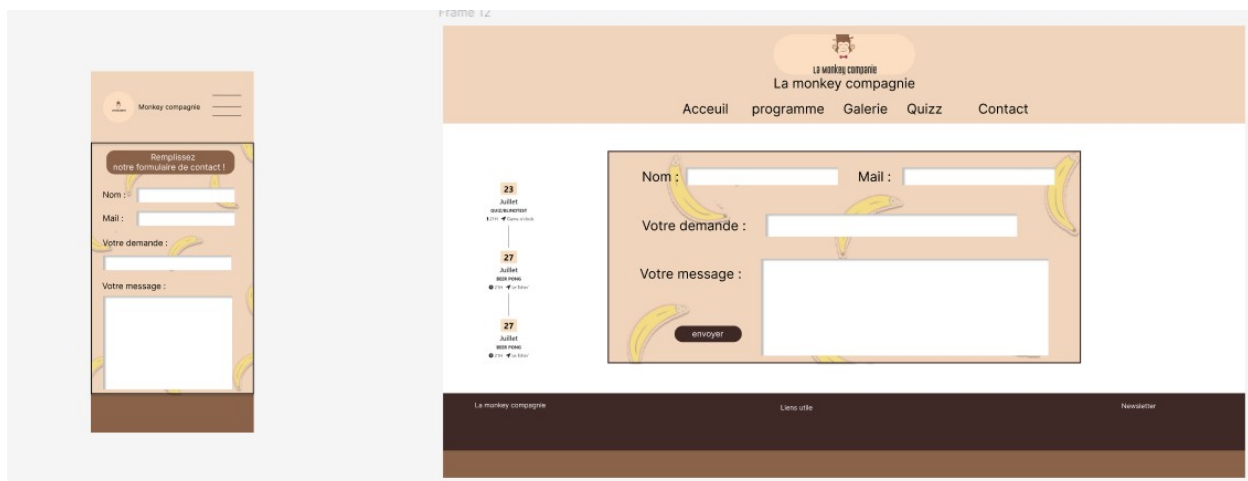
En cliquant sur un album on aura alors accès aux différentes images le composant, et en cliquant sur une image, on aura la possibilité de l'agrandir pour pouvoir avoir une meilleure vue de celle-ci





La vue mobile utilisant elle aussi la verticalité de l'écran mobile afin d'afficher les différents albums/photographies.

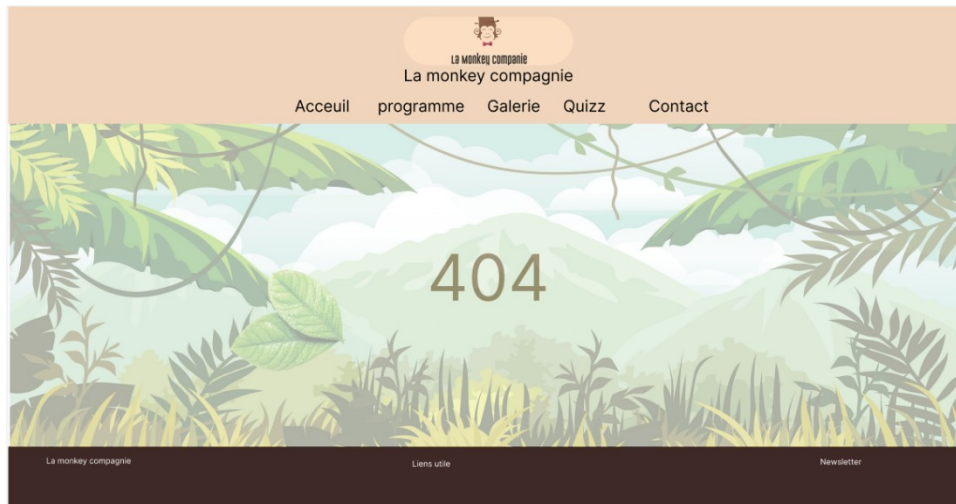
Page formulaire de contact :



La page de formulaire de contact, reprendre, comme la page galerie et accueil, le programme a sa gauche, avec un formulaire à remplir, situé au centre de la page. Le champ demande sera un menu déroulant, permettant, lorsque le mail sera envoyé via le mailer à Monsieur Scherrer, d'identifier directement et facilement le type de demande reçue dans sa boîte mail.

Page 404 :

Finalement, la page 404, désigné par Madame Garcia, et qui permettra de récupérer toutes les routes ne correspondant pas à une des routes définies dans mon projet.



7 - CGU et RGP

1 - CGU

Les CGU auront pour principe de décrire les conditions d'utilisations du site, et de donner les droits et les responsabilités des utilisateurs du site, on trouvera par exemple dans les conditions générales d'utilisation :

- Une définition des termes spécifiques, tels que « cookie »
- Les conditions d'utilisations tels que le fait de ne pas utiliser de langage répréhensible par la loi (insultes, menaces) dans le formulaire de contact.
- La propriété intellectuelle, tel que les droits d'auteurs du le logo utilisé sur le site.
- La conservations des données personnelles de l'utilisateur, telles que les adresses mail collectés pour faire fonctionner la newsletter, renvoyant alors sur la page des RGPD.
- Les responsabilités et exonérations de responsabilités.
- Le fait que les CGU peuvent être modifiée à tout moment.
- Les coordonnées de contact
- Les lois applicables et les juridictions concernées.
- La date d'entrée en vigueur.

Il pourra être utile de contacter un conseiller légal pour s'assurer de la conformité des CGU présentées sur le site, afin de s'assurer de ne pas faire de faux pas. On notera au passage que le site n'utilisera pas de cookies, et ne prévoit pas d'en utiliser, nous dispensant de demander l'accord du visiteur au moment de la première visite.

Le lien vers la page présentant les conditions générales d'utilisation, sera placé dans le footer du site, sur toutes les pages accessibles par le visiteur, afin d'être facilement accessible.

2 - RGPD

Les RGPD, où Règlement Général sur la protection des données, entré en vigueur en mai 2018. Sont un ensemble de règles de protection des données, protégeant l'utilisateur en l'informant sur la collecte, le traitement et le stockage de ses données.

Le principal point d'intérêt du site vitrine de la monkey compagnie, compte tenu de l'absence de cookies, sera la conservation d'adresse mails pour la newsletter. C'est d'ailleurs en raison de la possibilité que les données soient conservées après leur suppression que l'entité « newsletter » contiendra un booléen permettant de conserver l'adresse mail tout en notifiant qu'il ne fait plus partie de la liste de diffusion.

On trouvera à la suite un exemple de conditions générales d'utilisations, reprenant les différentes définitions, droits et devoirs de l'utilisateur, ainsi que les mentions légales RGPD nécessaires. Il s'agit d'un exemple de ce qui peut se faire et se trouver sur internet, permettant de se faire une idée des différentes parties et sous parties présentes sur une telle page.

Définitions

Client : tout professionnel ou personne physique capable au sens des articles 1123 et suivants du Code civil, ou personne morale, qui visite le Site objet des présentes conditions générales.

Prestations et Services : <https://www.monkeycompagnie.com> met à disposition des Clients :

Contenu : Ensemble des éléments constituant l'information présente sur le Site, notamment textes – images – vidéos.

Informations clients : Ci après dénommé « Information (s) » qui correspondent à l'ensemble des données personnelles susceptibles d'être détenues par <https://www.monkeycompagnie.com> pour la gestion de votre compte, de la gestion de la relation client et à des fins d'analyses et de statistiques.

Utilisateur : Internaute se connectant, utilisant le site susnommé.

Informations personnelles : « Les informations qui permettent, sous quelque forme que ce soit, directement ou non, l'identification des personnes physiques auxquelles elles s'appliquent » (article 4 de la loi n° 78-17 du 6 janvier 1978).

Les termes « données à caractère personnel », « personne concernée », « sous traitant » et « données sensibles » ont le sens défini par le Règlement Général sur la Protection des Données (RGPD : n° 2016-679)

1. Présentation du site internet.

En vertu de l'article 6 de la loi n° 2004-575 du 21 juin 2004 pour la confiance dans l'économie numérique, il est précisé aux utilisateurs du site internet <https://www.monkeycompagnie.com> l'identité des différents intervenants dans le cadre de sa réalisation et de son suivi:

Propriétaire : EIRL la monkey compagnie Capital social de /€ Numéro de TVA: / – 3 rue du commandeur cazeneuve 31400 Toulouse

Responsable publication : Scherrer – /

Le responsable publication est une personne physique ou une personne morale.

Webmaster : Delmas – leodelmas31@gmail.com

Hébergeur : Hostinger – 3 rue du commandeur cazeneuve 31400 Toulouse /

Délégué à la protection des données : Delmas – leodelmas31@gmail.com

Ce modèle de mentions légales est proposé par le générateur de mentions légales RGPD d'Orson.io

2. Conditions générales d'utilisation du site et des services proposés.

Le Site constitue une œuvre de l'esprit protégée par les dispositions du Code de la Propriété Intellectuelle et des Réglementations Internationales applicables. Le Client ne peut en aucune manière réutiliser, céder ou exploiter pour son propre compte tout ou partie des éléments ou travaux du Site.

L'utilisation du site <https://www.monkeycompagnie.com> implique l'acceptation pleine et entière des conditions générales d'utilisation ci-après décrites. Ces conditions d'utilisation sont susceptibles d'être modifiées ou complétées à tout moment, les utilisateurs du site <https://www.monkeycompagnie.com> sont donc invités à les consulter de manière régulière.

Ce site internet est normalement accessible à tout moment aux utilisateurs. Une interruption pour raison de maintenance technique peut être toutefois décidée par <https://www.monkeycompagnie.com>, qui s'efforcera alors de communiquer préalablement aux utilisateurs les dates et heures de l'intervention. Le site web <https://www.monkeycompagnie.com> est mis à jour régulièrement par <https://www.monkeycompagnie.com> responsable. De la même façon, les mentions légales peuvent être modifiées à tout moment : elles s'imposent néanmoins à l'utilisateur qui est invité à s'y référer le plus souvent possible afin d'en prendre connaissance.

3. Description des services fournis.

Le site internet <https://www.monkeycompagnie.com> a pour objet de fournir une information concernant l'ensemble des activités de la société. <https://www.monkeycompagnie.com> s'efforce de fournir sur le site <https://www.monkeycompagnie.com> des informations aussi précises que possible. Toutefois, il ne pourra être tenu responsable des oublis, des inexactitudes et des carences dans la mise à jour, qu'elles soient de son fait ou du fait des tiers partenaires qui lui fournissent ces informations.

Toutes les informations indiquées sur le site <https://www.monkeycompagnie.com> sont données à titre indicatif, et sont susceptibles d'évoluer. Par ailleurs, les renseignements figurant sur le site <https://www.monkeycompagnie.com> ne sont pas exhaustifs. Ils sont donnés sous réserve de modifications ayant été apportées depuis leur mise en ligne.

4. Limitations contractuelles sur les données techniques.

Le site utilise la technologie JavaScript. Le site Internet ne pourra être tenu responsable de dommages matériels liés à l'utilisation du site. De plus, l'utilisateur du site s'engage à accéder au site en utilisant un matériel récent, ne contenant pas de virus et avec un navigateur de dernière génération mis-à-jour. Le site <https://www.monkeycompagnie.com> est hébergé chez un prestataire sur le territoire de l'Union Européenne conformément aux dispositions du Règlement Général sur la Protection des Données (RGPD : n° 2016-679).

L'objectif est d'apporter une prestation qui assure le meilleur taux d'accessibilité. L'hébergeur assure la continuité de son service 24 Heures sur 24, tous les jours de l'année. Il se réserve néanmoins la possibilité d'interrompre le service d'hébergement pour les durées les plus courtes possibles notamment à des fins de maintenance, d'amélioration de ses infrastructures, de défaillance de ses infrastructures ou si les Prestations et Services génèrent un trafic réputé anormal.

<https://www.monkeycompagnie.com> et l'hébergeur ne pourront être tenus responsables en cas de dysfonctionnement du réseau Internet, des lignes téléphoniques ou du matériel informatique et de téléphonie lié notamment à l'encombrement du réseau empêchant l'accès au serveur.

5. Propriété intellectuelle et contrefaçons.

<https://www.monkeycompagnie.com> est propriétaire des droits de propriété intellectuelle et détient les droits d'usage sur tous les éléments accessibles sur le site internet, notamment les textes, images, graphismes, logos, vidéos, icônes et sons. Toute reproduction, représentation, modification, publication, adaptation de tout ou partie des éléments du site, quel que soit le moyen ou le procédé utilisé, est interdite, sauf autorisation écrite préalable de : <https://www.monkeycompagnie.com>.

Toute exploitation non autorisée du site ou de l'un quelconque des éléments qu'il contient sera considérée comme constitutive d'une contrefaçon et poursuivie conformément aux dispositions des articles L.335-2 et suivants du Code de Propriété Intellectuelle.

6. Limitations de responsabilité.

<https://www.monkeycompagnie.com> agit en tant qu'éditeur du site. <https://www.monkeycompagnie.com> est responsable de la qualité et de la véracité du Contenu qu'il publie.

<https://www.monkeycompagnie.com> ne pourra être tenu responsable des dommages directs et indirects causés au matériel de l'utilisateur, lors de l'accès au site internet <https://www.monkeycompagnie.com>, et résultant soit de l'utilisation d'un matériel ne répondant pas aux spécifications indiquées au point 4, soit de l'apparition d'un bug ou d'une incompatibilité.

<https://www.monkeycompagnie.com> ne pourra également être tenu responsable des dommages indirects (tels par exemple qu'une perte de marché ou perte d'une chance) consécutifs à l'utilisation du site <https://www.monkeycompagnie.com>. Des espaces interactifs (possibilité de poser des questions dans l'espace contact) sont à la disposition des utilisateurs. <https://www.monkeycompagnie.com> se réserve le droit de supprimer, sans mise en demeure préalable, tout contenu déposé dans cet espace qui contreviendrait à la législation applicable en France, en particulier aux dispositions relatives à la protection des données. Le cas échéant, <https://www.monkeycompagnie.com> se réserve également la possibilité de mettre en cause la responsabilité civile et/ou pénale de l'utilisateur, notamment en cas de message à caractère raciste, injurieux, diffamant ou pornographique, quel que soit le support utilisé (texte, photographie ...).

7. Gestion des données personnelles.

Le Client est informé des réglementations concernant la communication marketing, la loi du 21 Juin 2014 pour la confiance dans l'Economie Numérique, la Loi Informatique et Liberté du 06 Août 2004 ainsi que du Règlement Général sur la Protection des Données (RGPD : n° 2016-679).

7.1 Responsables de la collecte des données personnelles

Pour les Données Personnelles collectées dans le cadre de la création du compte personnel de l'Utilisateur et de sa navigation sur le Site, le responsable du traitement des Données Personnelles est : la monkey compagnie. <https://www.monkeycompagnie.com> est représenté par Scherrer, son représentant légal

En tant que responsable du traitement des données qu'il collecte, <https://www.monkeycompagnie.com> s'engage à respecter le cadre des dispositions légales en vigueur. Il lui appartient notamment au Client d'établir les finalités de ses traitements de données, de fournir à ses prospects et clients, à partir de la collecte de leurs consentements, une information complète sur le traitement de leurs données personnelles et de maintenir un registre des traitements conforme à la réalité. Chaque fois que <https://www.monkeycompagnie.com> traite des Données Personnelles, <https://www.monkeycompagnie.com> prend toutes les mesures raisonnables pour s'assurer de l'exactitude et de la pertinence des Données Personnelles au regard des finalités pour lesquelles <https://www.monkeycompagnie.com> les traite.

7.2 Finalité des données collectées

<https://www.monkeycompagnie.com> est susceptible de traiter tout ou partie des données :

- pour permettre la navigation sur le Site et la gestion et la traçabilité des prestations et services commandés par l'utilisateur : données de connexion et d'utilisation du Site, facturation, historique des commandes, etc.
- pour prévenir et lutter contre la fraude informatique (spamming, hacking...) : matériel informatique utilisé pour la navigation, l'adresse IP, le mot de passe (hashé)
- pour améliorer la navigation sur le Site : données de connexion et d'utilisation
- pour mener des enquêtes de satisfaction facultatives sur <https://www.monkeycompagnie.com> : adresse email
- pour mener des campagnes de communication (sms, mail) : numéro de téléphone, adresse email

<https://www.monkeycompagnie.com> ne commercialise pas vos données personnelles qui sont donc uniquement utilisées par nécessité ou à des fins statistiques et d'analyses.

7.3 Droit d'accès, de rectification et d'opposition

Conformément à la réglementation européenne en vigueur, les Utilisateurs de <https://www.monkeycompagnie.com> disposent des droits suivants :

- droit d'accès (article 15 RGPD) et de rectification (article 16 RGPD), de mise à jour, de complétude des données des Utilisateurs droit de verrouillage ou d'effacement des données des Utilisateurs à caractère personnel (article 17 du RGPD), lorsqu'elles sont inexactes, incomplètes, équivoques, périmées, ou dont la collecte, l'utilisation, la communication ou la conservation est interdite
- droit de retirer à tout moment un consentement (article 13-2c RGPD)
- droit à la limitation du traitement des données des Utilisateurs (article 18 RGPD)
- droit d'opposition au traitement des données des Utilisateurs (article 21 RGPD)
- droit à la portabilité des données que les Utilisateurs auront fournies, lorsque ces données font l'objet de traitements automatisés fondés sur leur consentement ou sur un contrat (article 20 RGPD)
- droit de définir le sort des données des Utilisateurs après leur mort et de choisir à qui <https://www.monkeycompagnie.com> devra communiquer (ou non) ses données à un tiers qu'ils aura préalablement désigné

Dès que <https://www.monkeycompagnie.com> a connaissance du décès d'un Utilisateur et à défaut d'instructions de sa part, <https://www.monkeycompagnie.com> s'engage à détruire ses données, sauf si leur conservation s'avère nécessaire à des fins probatoires ou

7.4 Non-communication des données personnelles

<https://www.monkeycompagnie.com> s'interdit de traiter, héberger ou transférer les Informations collectées sur ses Clients vers un pays situé en dehors de l'Union européenne ou reconnu comme « non adéquat » par la Commission européenne sans en informer préalablement le client. Pour autant, <https://www.monkeycompagnie.com> reste libre du choix de ses sous-traitants techniques et commerciaux à la condition qu'il présentent les garanties suffisantes au regard des exigences du Règlement Général sur la Protection des Données (RGPD : n° 2016-679).

<https://www.monkeycompagnie.com> s'engage à prendre toutes les précautions nécessaires afin de préserver la sécurité des Informations et notamment qu'elles ne soient pas communiquées à des personnes non autorisées. Cependant, si un incident impactant l'intégrité ou la confidentialité des Informations du Client est portée à la connaissance de <https://www.monkeycompagnie.com>, celle-ci devra dans les meilleurs délais informer le Client et lui communiquer les mesures de corrections prises. Par ailleurs <https://www.monkeycompagnie.com> ne collecte aucune « données sensibles ».

Les Données Personnelles de l'Utilisateur peuvent être traitées par des filiales de <https://www.monkeycompagnie.com> et des sous-traitants (prestataires de services), exclusivement afin de réaliser les finalités de la présente politique.

Dans la limite de leurs attributions respectives et pour les finalités rappelées ci-dessus, les principales personnes susceptibles d'avoir accès aux données des Utilisateurs de <https://www.monkeycompagnie.com> sont principalement les agents de notre service client.

8. Notification d'incident

Quels que soient les efforts fournis, aucune méthode de transmission sur Internet et aucune méthode de stockage électronique n'est complètement sûre. Nous ne pouvons en conséquence pas garantir une sécurité absolue. Si nous prenions connaissance d'une brèche de la sécurité, nous avertirions les utilisateurs concernés afin qu'ils puissent prendre les mesures appropriées. Nos procédures de notification d'incident tiennent compte de nos obligations légales, qu'elles se situent au niveau national ou européen. Nous nous engageons à informer pleinement nos clients de toutes les questions relevant de la sécurité de leur compte et à leur fournir toutes les informations nécessaires pour les aider à respecter leurs propres obligations réglementaires en matière de reporting.

Aucune information personnelle de l'utilisateur du site <https://www.monkeycompagnie.com> n'est publiée à l'insu de l'utilisateur, échangée, transférée, cédée ou vendue sur un support quelconque à des tiers. Seule l'hypothèse du rachat de <https://www.monkeycompagnie.com> et de ses droits permettrait la transmission des dites informations à l'éventuel acquéreur qui serait à son tour tenu de la même obligation de conservation et de modification des données vis à vis de l'utilisateur du site <https://www.monkeycompagnie.com>.

Sécurité

Pour assurer la sécurité et la confidentialité des Données Personnelles et des Données Personnelles de Santé, <https://www.monkeycompagnie.com> utilise des réseaux protégés par des dispositifs standards tels que par pare-feu, la pseudonymisation, l'encryption et mot de passe.

Lors du traitement des Données Personnelles, <https://www.monkeycompagnie.com> comprend toutes les mesures raisonnables visant à les protéger contre toute perte, utilisation détournée, accès non autorisé, divulgation, altération ou destruction.

9. Liens hypertextes « cookies » et balises (“tags”) internet

Le site <https://www.monkeycompagnie.com> contient un certain nombre de liens hypertextes vers d'autres sites, mis en place avec l'autorisation de <https://www.monkeycompagnie.com>. Cependant, <https://www.monkeycompagnie.com> n'a pas la possibilité de vérifier le contenu des sites ainsi visités, et n'assumera en conséquence aucune responsabilité de ce fait.

Sauf si vous décidez de désactiver les cookies, vous acceptez que le site puisse les utiliser. Vous pouvez à tout moment désactiver ces cookies et ce gratuitement à partir des possibilités de désactivation qui vous sont offertes et rappelées ci-après, sachant que cela peut réduire ou empêcher l'accessibilité à tout ou partie des Services proposés par le site.

9.1. « COOKIES »

Un « cookie » est un petit fichier d'information envoyé sur le navigateur de l'Utilisateur et enregistré au sein du terminal de l'Utilisateur (ex : ordinateur, smartphone), (ci-après « Cookies »). Ce fichier comprend des informations telles que le nom de domaine de l'Utilisateur, le fournisseur d'accès Internet de l'Utilisateur, le système d'exploitation de l'Utilisateur, ainsi que la date et l'heure d'accès. Les Cookies ne risquent en aucun cas d'endommager le terminal de l'Utilisateur.

<https://www.monkeycompagnie.com> est susceptible de traiter les informations de l'Utilisateur concernant sa visite du Site, telles que les pages consultées, les recherches effectuées. Ces informations permettent à <https://www.monkeycompagnie.com> d'améliorer le contenu du Site, de la navigation de l'Utilisateur.

Les Cookies facilitant la navigation et/ou la fourniture des services proposés par le Site, l'Utilisateur peut configurer son navigateur pour qu'il lui permette de décider s'il souhaite ou non les accepter de manière à ce que des Cookies soient enregistrés dans le terminal ou, au contraire, qu'ils soient rejetés, soit systématiquement, soit selon leur émetteur. L'Utilisateur peut également configurer son logiciel de navigation de manière à ce que l'acceptation ou le refus des Cookies lui soient proposés ponctuellement, avant qu'un Cookie soit susceptible d'être enregistré dans son terminal. <https://www.monkeycompagnie.com> informe l'Utilisateur que, dans ce cas, il se peut que les fonctionnalités de son logiciel de navigation ne soient pas toutes disponibles.

Si l'Utilisateur refuse l'enregistrement de Cookies dans son terminal ou son navigateur, ou si l'Utilisateur supprime ceux qui y sont enregistrés, l'Utilisateur est informé que sa navigation et son expérience sur le Site peuvent être limitées. Cela pourrait également être le cas lorsque <https://www.monkeycompagnie.com> ou l'un de ses prestataires ne peut pas reconnaître, à des fins de compatibilité technique, le type de navigateur utilisé par le terminal, les paramètres de langue et d'affichage ou le pays depuis lequel le terminal semble connecté à Internet.

Le cas échéant, <https://www.monkeycompagnie.com> décline toute responsabilité pour les conséquences liées au fonctionnement dégradé du Site et des services éventuellement proposés par <https://www.monkeycompagnie.com>, résultant (i) du refus de Cookies par l'Utilisateur (ii) de l'impossibilité pour <https://www.monkeycompagnie.com> d'enregistrer ou de consulter les Cookies nécessaires à leur fonctionnement du fait du choix de l'Utilisateur. Pour la gestion des Cookies et des choix de l'Utilisateur, la configuration de chaque navigateur est différente. Elle est décrite dans le menu d'aide du navigateur, qui permettra de savoir de quelle manière l'Utilisateur peut modifier ses souhaits en matière de Cookies.

À tout moment, l'Utilisateur peut faire le choix d'exprimer et de modifier ses souhaits en matière de Cookies.

<https://www.monkeycompagnie.com> pourra en outre faire appel aux services de prestataires externes pour l'aider à recueillir et traiter les informations décrites dans cette section.

Enfin, en cliquant sur les icônes dédiées aux réseaux sociaux Twitter, Facebook, LinkedIn et Google Plus figurant sur le Site de <https://www.monkeycompagnie.com> ou dans son application mobile et si l'Utilisateur a accepté le dépôt de cookies en poursuivant sa navigation sur le Site Internet ou l'application mobile de <https://www.monkeycompagnie.com>, Twitter, Facebook, LinkedIn et Google Plus peuvent également déposer des cookies sur vos terminaux (ordinateur, tablette, téléphone portable).

Ces types de cookies ne sont déposés sur vos terminaux qu'à condition que vous y consentiez, en continuant votre navigation sur le Site Internet ou l'application mobile de <https://www.monkeycompagnie.com>. À tout moment, l'Utilisateur peut néanmoins revenir sur son consentement à ce que <https://www.monkeycompagnie.com> dépose ce type de cookies.

Article 9.2. BALISES (“TAGS”) INTERNET

<https://www.monkeycompagnie.com> peut employer occasionnellement des balises Internet (également appelées « tags », ou balises d'action, GIF à un pixel, GIF transparents, GIF invisibles et GIF un à un) et les déployer par l'intermédiaire d'un partenaire spécialiste d'analyses Web susceptible de se trouver (et donc de stocker les informations correspondantes, y compris l'adresse IP de l'Utilisateur) dans un pays étranger.

Ces balises sont placées à la fois dans les publicités en ligne permettant aux internautes d'accéder au Site, et sur les différentes pages de celui-ci.

Cette technologie permet à <https://www.monkeycompagnie.com> d'évaluer les réponses des visiteurs face au Site et l'efficacité de ses actions (par exemple, le nombre de fois où une page est ouverte et les informations consultées), ainsi que l'utilisation de ce Site par l'Utilisateur.

Le prestataire externe pourra éventuellement recueillir des informations sur les visiteurs du Site et d'autres sites Internet grâce à ces balises, constituer des rapports sur l'activité du Site à l'attention de <https://www.monkeycompagnie.com>, et fournir d'autres services relatifs à l'utilisation de celui-ci et d'Internet.

10. Droit applicable et attribution de juridiction.

Tout litige en relation avec l'utilisation du site <https://www.monkeycompagnie.com> est soumis au droit français. En dehors des cas où la loi ne le permet pas, il est fait attribution exclusive de juridiction aux tribunaux compétents de Toulouse

8 - Matrice de risques

	N°	Description du risque	Détails	Cause	Date	Effet sur l'objectif	Probabilité el/B	Impact/S	Criticité	Mesures Préventives	Mesures Curatives	Coût (en temps ou en ressources financières)
Projet	1	Un manque d'ergonomie du site, poussant les utilisateurs à fermer la page sans avoir obtenu l'information souhaitée et sans s'être redirigés vers les réseaux sociaux de l'entreprise	Un site disposant de trop de pages, de chemins de navigations peu clairs, ou d'une ergonomie mal pensée, pour la version desktop comme pour la version mobile	Un design UX/UI mal pensé, et/ou mal porté pour la version mobile, trop de pages intermédiaires entre les différentes informations	19/06-19/07	Un site fonctionnel, mais qui remplira pas les besoins exprimés par le client.	2	5		Un travail effectué en amont sur les fonctionnalités cruciales du site, et sur leur mise en avant	Un re-design de l'arborescence et des différentes pages du site, long et coûteux en temps et en énergie	entre une semaine et demie et deux semaines de travail supplémentaire
	2	Mauvaise planification des durées nécessaires à la complétude du projet.	De mauvaises estimations concernant le temps nécessaire à accomplir les différentes parties du projet	Mauvaise connaissance des spécificités techniques, et donc du temps nécessaire à leur mise en place	19/06 - 19/08	Des retards imprévus, obligation de replanifier le stage, perte de temps et de qualité du produit fini.	3	3		Deux semaines supplémentaires de prévues au sein du planning, afin de pouvoir encaisser un potentiel retard	Une cadence de travail accrue, afin de pallier au manque de temps	17 J/H
Humain	3	echec à trouver un terrain d'entente concernant la maquette graphique	Dans le cas où les deux prestataires travaillant en collaboration ne parviendraient pas à se mettre d'accord sur une charte graphique, le projet pourrait alors prendre du retard, bloquant l'avancement de	Une mauvaise communication, trop espacée ou trop légèrement approfondie	19/06 - 19/07	Une maquette étant produite en retard, et ne donnant pas satisfaction au destinataire du site internet	2	3		Des réunions hebdomadaires, en présence du destinataire et de la collaboratrice	Un travail de ré-ajustement, permettant de parvenir à une version satisfaisante pour les deux partis	7 J/H pour le prestataire ET la collaboratrice
	4	Modifications trop fréquentes du besoin	Il est possible que des exigences trop fréquemment reformulées nuise au bon développement du projet, obligeant à revenir sur des tâches normalement déjà accomplies	Un manque de clarté, et un besoin insuffisamment défini dans le cahier des charges	19/06-19/08	Des réglages, ajustements, créant à chaque fois des retards à répétition	3	5		Des réunions hebdomadaires, et un suivi précis des différents livrables du projet	Des heures supplémentaire et un allongement de la durée de certaines tâches concernées par les retouches	En fonction de livrable à modifier
Technique	5	securisation insuffisante du panel admin	Une mise en place défectueuse du système de tokens d'authentification pourrait conduire à un détournement des fonctionnalités back-end telles que la modification ou l'envoi de devis	Une mauvaise maîtrise du système d'authentification, et une documentation insuffisante, conduisant à une sécurité défectueuse	12/07-26/07	Un détournement des différentes fonctionnalités du site	3	5		Une documentation extensive du fonctionnement de JivToken	Recommencer toute les fonctions d'authentification du site	7 J/H
	6	Mauvais choix des technologies	Erreur de jugement au moment e choisir les technologies utilisées, notamment au niveau des frameworks qui seront utilisés pour le site	Erreurs de jugement au moment d'identifier les forces et les faiblesses des différents framework	19/06 - 19/08	Un produit fini ne profitant pas pleinement des fonctionnalités qui lui auraient été offerte par le choix d'une autre technologie, perte de qualité du produit final	3	4		Un travail de renseignement en profondeur sur les forces et les faiblesses des différents frameworks et technologies à disposition du prestataire	Un mauvais choix de technologie, pourrait être résolu si il est détecté dans les premiers temps du projet, au delà d'un d'une ou deux semaines, il sera malheureusement plus viable de continuer le projet avec une technologie non optimale	Dépendra du moment ou le choix effectué se révélera être défectueux
Externe	7	Mauvais choix dans le service d'hébergement	Choix d'un service d'hébergement trop coûteux au vue des besoins du site, ou au contraire, choix d'un service peu coûteux mais insuffisant	Mauvais renseignements sur les services proposés par les différentes plateformes d'hébergement	04/08 - 19/08	Un echec de la mise en production/ Une perte d'argent évitable	2	4		Processus de documentation sur les différents services d'hébergement	Changement d'hébergement	4 J/H

Probabilité/gravité	1	2	3	4	5
1					
2			1		3
3			2	6	4 et 5
4		7			
5					

Ci-dessus, la matrice de risque accompagnant le projet de site vitrine de la monkey compagnie. Les différents risques seront : Une mauvaise ergonomie du site, un planning mal pensé, des échecs de communication entre moi et Madame Garcia. Un mauvais choix de technologie ou de service d'hébergement.

Cependant, les deux principaux risques repérés dans cette matrice seront :

1 – Des modifications trop fréquente du besoin de la part du client. En ajoutant trop fréquemment de nouvelles demandes, ou en modifiant de manière trop régulière celles déjà formulées, le client pourra sérieusement compromettre l'avancement du projet, et le retarder à répétition. Il faudra pour contrer cela, s'assurer d'un cahier des charges clair et réfléchi, construit sur la base d'une discussion avec le client.

2 – Une sécurisation insuffisante du panel de connexion, pouvant mener à des attaques CSRF par exemple, attaque qui sera expliquée et détaillée dans la partie dédiée. Ou encore une sécurisation trop faible des champs de saisie du formulaire, ou des injection SQL. Ce panel d'attaques, ainsi que les moyens de s'en prévenir auront leur propre partie dans la seconde partie de ce mémoire.

9 – Directives Vue.js

Un composant vue.js comprend, de base, une balise <template> qui contiendra la syntaxe HTML-like, un composant <style> permettant de coder la mise en forme en CSS, et une balise <script> qui permettra de déclarer et configurer les données, comportements et logique du composant, on pourra trouver dans cette balise :

- Les props, passées par un composant parent ou par un controller au travers d'une route
- Les méthodes (fonctions) permettant de programmer des comportements pour le composant
- Les data, ou variables qui seront utilisées dans les méthodes du composant, les data ainsi déclarées dans l'objet data seront considérés comme réactives, c'est-à-dire mises-à-jour et utilisées pour actualiser le Virtual DOM du composant.

On trouvera des exemples dans mon composant 'Album' qui sert à afficher un album en particulier, on accède à cette page depuis la page 'galerie' qui affiche la totalité de albums créés par l'administrateur.

```
<script>
import MonkeyLayout from '../Layouts/MonkeyLayout.vue';
import planning from "../Planning.vue";
export default{
  layout: MonkeyLayout,
  components:{
    | planning,
  },
  props:{
    images:{
      type:Array,
      default:()=>[],
    },
    album:{
      type:Array,
      default:()=>[],
    },
    evenements:{
      type:Array,
      default:()=>[],
    }
  },
  data(){
    return{
      isHovered:null,
      modal:false,
      selectedImage:null,
    }
  },
  methods:{
    openModal(){
      this.modal = !this.modal;
      console.log(key);
    }
  }
}
</script>
```

On trouve la déclaration du composant enfant, du layout, des props, typés en tant que tableau vide avant la réception des données du controller. On trouve aussi les data, tels que les variables 'isHovered', 'modal' et 'selectedImage' qui serviront à déterminer le comportement du composant.

```

<template>
  <section>
    <planning :evenements="evenements"></planning>
    <div class="containerGalerie">
      <div class="image" v-for="(image,key) in images" :key="key" @mouseover="isHovered = key" @mouseleave="isHovered = null">
        
        <Transition name="fade">
          <p v-show="isHovered===key" class="imagetext" @click="openModal"><b>{{ image.description }}</b></p>
        </Transition>
      </div>
    </div>
  </section>

  <!-- MODAL -->
  <teleport to="section">
    
  </teleport>
</template>

```

Par exemple, au sein du composant 'album' ces variables seront utilisées, pour faire apparaître la description de l'image, si elle a été rentrée au moment de créer l'entité. J'ai donc utilisé ce qui s'appelle des directives vue.js. Par exemple, je vais créer une div de classe 'image' pour chaque itération de l'entité 'image', grâce à la directive v-for dite de 'list-rendering', donnant au passage une clé, un index, à chaque élément. Sur cette div, est placé une directive d'écoute dont la version shorthand est @mouseover, qui permettra, lorsque la souris passe sur une des div de classe image, de donner à la data 'isHovered' la clé correspondante à l'image survolée.

Enfin, dans 'imageText' si la valeur de la variable 'isHovered' est la même que la clé correspondante à la div dont fait partie 'imageText' alors le texte apparaîtra comme ci-dessous.



Cet affichage conditionnel est rendu possible par la directive v-show, qui prend une condition pour afficher, ou non, un élément du Template. Il faut noter que vue.js dispose de deux directives conditionnelles, v-if et v-show, la différence entre les deux étant que v-show sera l'équivalent d'un display:none en css lorsque la condition ne sera pas remplie, tandis que v-if détruira et reconstruira l'élément en fonction du remplissage de la condition. J'ai choisi pour l'affichage de la description de l'image, d'utiliser v-show, puisque le changement de condition risque d'être relativement fréquent, l'utilisation du v-if étant plus adaptée pour des changements de conditions moins sujet au changement.

La variable 'modal' servira à afficher l'image en agrandi lorsque l'utilisateur cliquera dessus, l'affichage se faisant via le composant natif 'téléport' permettant de faire apparaître un élément au niveau d'un nœud du DOM. Ici en cliquant sur une image, on appelle la fonction 'openModal', on récupérera l'image depuis sa key enregistrée dans 'isHovered', et on affichera l'image correspondante, en passant la variable

'modal' à true rendant la div visible, en remplissant la condition du v-if. J'ai ici choisi d'utiliser v-if, mais v-show aurait aussi pu être approprié.

Cette façon de construire la balise script, se nomme 'option API', vue.js peut aussi se faire en 'composition API'. Dans une option API, nous l'avons vu, des objets d'option tels que 'methods' sont utilisées pour déclarer des propriétés, qui deviennent accessibles via 'this.' Pointant vers la variable déclarée dans l'instance, comme dans 'this.modal'. Cette manière de procéder permet un meilleur découpage et est plus pratique à manier pour des gens relativement inexpérimentés, ce qui est en partie la raison pour laquelle j'ai décidé de procéder ainsi, la composition en 'option API' est aussi plus adaptée pour des petits projets simples et m'a donc paru plus adaptée pour le projet de site vitrine.

La 'composition API', quant à elle, permet de déclarer des fonctions, des data et des props, sans les encapsuler dans des objets d'option, et surtout, de les exporter pour les utiliser dans d'autres composants, donnant une multitude de possibilités, comme pouvoir utiliser une fonction en callback sans avoir à la réécrire dans chaque composant. C'est une manière de procéder qui offre de multiples possibilités, tout en demandant une grande rigueur, mais elle ne paraissait pas nécessaire à l'accomplissement du projet, qui se suffisait de l'option API.

10 – Layouts et header inertia

J’ai mentionné dans la partie précédente, la manière qu’avait Inertia.js de traiter les header et les footer, il est en effet possible, avec inertia, de définir un composant comme un ‘layout’, c’est-à-dire un composant présent par défaut sur les pages où il sera défini comme tel

```
<script>
import { reactive } from 'vue'
import MonkeyLayout from "@/Layouts/MonkeyLayout.vue";
import planning from "../Planning.vue";
import { Inertia } from '@inertiajs/inertia';
export default{
  layout:MonkeyLayout,
  components:{
    | | planning,
  },
};
```

La définition du layout se fait un peu de la même façon qu’un composant tel que planning, il faut l’importer et le définir dans l’option API. Il apparaîtra alors dans cette page. Il est aussi possible de définir un layout pour l’ensemble des pages du site, via le fichier app.js.

```
import './bootstrap';
import './css/app.css';
import { createApp, h } from 'vue'
import { createInertiaApp } from '@inertiajs/vue3';
import { resolvePageComponent } from 'laravel-vite-plugin/inertia-helpers';
import { ZiggyVue } from '../../vendor/tightenco/ziggy/dist/vue.m';
import MonkeyLayout from '../Layouts/MonkeyLayout.vue';

const appName = import.meta.env.VITE_APP_NAME || 'bonjour';

createInertiaApp({
  title: (title) => `${title}`,
  resolve: async name =>{
    const page = await resolvePageComponent(`./Pages/${name}.vue`, import.meta.glob('./Pages/**/*.vue'))
    if(!page.default.layout){
      | page.default.layout = MonkeyLayout
    }
    return page
  },
  setup({ el, App, props, plugin }) {
    return createApp({ render: () => h(App, props) })
      | .use(plugin)
      | .use(ZiggyVue, Ziggy)
      | .mount(el);
  },
  progress: {
    | color: '#4B5563',
  },
});
```

On déclarera ainsi le page.default.layout comme le ‘MonkeyLayout’, seulement si un layout préexistant n’existe pas.

Le layout ressemble ainsi à cela :

```

<script setup>
import HeadLink from '@/Components/HeadLinks.vue'
import FooterMonkey from '@/Components/FooterMonkey.vue'
import {Head} from '@inertiajs/vue3';
</script>

<template>
<Head>
  <title>Monkey Compagnie</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta http-equiv="Content-Language" content="fr">
  <meta name="description" content="Site de la monkey Compagnie, animation et fêtes en bar">
  <meta name="author" content="Leo Delmas, Kelly Garcia">
  <meta property="og:title" content="La monkey compagnie">
  <meta property="og:description" content="Site de la monkey Compagnie, animation et fêtes en bar">
  <meta property="og:image" content="/assets/images/logochelou.png">
  <meta name="twitter:card" content="/assets/images/logochelou.png">
  <meta name="twitter:site" content="@nom_du_site">
  <meta name="twitter:title" content="La monkey compagnie">
  <meta name="twitter:description" content="Site de la monkey Compagnie, animation et fêtes en bar">
  <meta name="twitter:image" content="/assets/images/logochelou.png">
  <link rel="icon" href="/assets/images/logochelou.png" type="image/png">
  <link rel="preconnect" href="https://fonts.bunny.net">
  <link href="https://fonts.bunny.net/css?family=figtree:400,500,600&display=swap" rel="stylesheet" />
</head>
<HeadLink></HeadLink>
0 references
<slot/>
<FooterMonkey></FooterMonkey>
</template>

<style scoped>
</style>

```

On trouve un composant, composé du header/barre de navigation 'headLink' et le footer 'FooterMonkey'. La balise <slot /> servira à signaler où le contenu du temple associé au layout sera intégré.

On remarque au passage le composant natif de Laravel <Head>, il est spécifié dans la documentation Inertia.js, que compte tenu du fait que les applications Inertia sont 'render' dans le composant body, le composant <head> du fichier app.blade.php peut être plus difficilement lu par les robots de google, inertia propose alors comme solution un composant head. Et recommande alors de l'intégrer au layout afin de le rendre disponible sur les différentes pages du site.

On trouve alors des méta données, telles que la description contenant certains mots clés, le langage du site, le lien de l'icône dans la barre d'onglet, ainsi que les différentes propriétés nécessaires en cas de partage du site sur twitter ou facebook.

11 – Liste des middlewares définis dans le fichier Kernel.php

```
class Kernel extends HttpKernel
{
    /**
     * The application's global HTTP middleware stack.
     *
     * These middleware are run during every request to your application.
     *
     * @var array<int, class-string|string>
     */
    protected $middleware = [
        // \App\Http\Middleware\TrustHosts::class,
        \App\Http\Middleware\TrustProxies::class,
        \Illuminate\Http\Middleware\HandleCors::class,
        \App\Http\Middleware\PreventRequestsDuringMaintenance::class,
        \Illuminate\Foundation\Http\Middleware\ValidatePostSize::class,
        \App\Http\Middleware\TrimStrings::class,
        \Illuminate\Foundation\Http\Middleware\ConvertEmptyStringsToNull::class,
    ];

    /**
     * The application's route middleware groups.
     *
     * @var array<string, array<int, class-string|string>>
     */
    protected $middlewareGroups = [
        'web' => [
            \App\Http\Middleware\EncryptCookies::class,
            \Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::class,
            \Illuminate\Session\Middleware\StartSession::class,
            \Illuminate\View\Middleware\ShareErrorsFromSession::class,
            \App\Http\Middleware\VerifyCsrfToken::class,
            \Illuminate\Routing\Middleware\SubstituteBindings::class,
            \Illuminate\Http\Middleware\AddLinkHeadersForPreloadedAssets::class,
            \App\Http\Middleware\HandleInertiaRequests::class,
        ],

        'api' => [
            // \Laravel\Sanctum\Http\Middleware\EnsureFrontendRequestsAreStateful::class,
            \Illuminate\Routing\Middleware\ThrottleRequests::class.':api',
            \Illuminate\Routing\Middleware\SubstituteBindings::class,
        ],
    ];
}
```

12 – Routes Admin du projet :

```
// routes admin
Route::middleware('auth')->prefix('admin')->group(function(){

    Route::get('/',function(){
        return Inertia::render('Admin/Panel',[
        ]);
    })->name('admin');

    Route::resource('images', ImageController::class);

    // Routes pour l'entité "album"
    Route::resource('albums', AlbumController::class);

    // Routes pour l'entité "equipier"
    Route::resource('equipe', EquipierController::class);

    // Routes pour l'entité "evenement"
    Route::resource('evenements', EvenementController::class);

    Route::post('evenements/recurring', [EvenementController::class, 'createRecurringEvent'])->name('evenements.recurring');

    // Routes pour l'entité "quizz"
    Route::resource('quizz', QuizzController::class);
    Route::resource('qcm', QCMController::class);
    Route::resource('vraifaux', VraiFauxController::class);
    Route::resource('quizzimage', QuizzImageController::class);
});

require __DIR__.'/auth.php';
```

13 – Création d'un formulaire de contact

Afin de créer le formulaire de contact, il fallait donc créer une vue, un composant vue.js. Ce composant comportera la méthode particulière d'Inertia.js de gérer les formulaires. Plutôt que de subir un 'full page reload', inertia se propose d'intercept les soumissions de formulaire et de les effectuer via inertia. Au lieu de préciser dans le paramètre action de la balise form, la location du fichier php où soumettre le formulaire, on utilisera la directive @submit.prevent = 'submit' afin d'empêcher le rechargement de la page lorsque le formulaire est envoyé, et appeler la fonction 'submit'.

```
methods:{},
  setup(){
    const form = reactive({
      nom:null,
      email:null,
      demande:null,
      mess:null
    })

    const submit = () =>{
      Inertia.post('/contact',form,{
        onSuccess:() => {
          form.value = {
            nom:String,
            email:String,
            demande:String,
            mess:String
          };
          console.log(this.form);
        }
      });
    };
    return{
      form,
      submit,
    }
  }
}
```

La fonction va alors envoyer une requête inertia, à laquelle on passera l'URI ainsi que l'objet form, auquel on aura assigné les valeurs du formulaire.

La requête post sera alors transmise au ContactController

```
public function submitForm(Request $request): RedirectResponse
{
    // Valider la requête
    $data = $request->validate([
        'nom' => 'required',
        'email' => 'required|email',
        'demande' => 'required',
        'mess' => 'required|string',
    ]);
    // Envoi d'un email
    Mail::to('exemple@gmail.com')->send(new ContactMail(
        $data['nom'],
        $data['email'],
        $data['demande'],
        $data['mess'],
    ));

    // Renvoyer une réponse
    return redirect()->back();
}
```

Le Controller se chargera alors de valider la requête grâce au système de validator d'inertia, avant de créer une nouvelle instance de 'contactMail.php', une classe servant à définir un 'Model' pour les mails aux moyens d'un construct contenu dans la classe. Sera alors retourné une vue en Blade.php

```
<?php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Nouveau message de {{ $nom }}</title>
</head>
<body>
<table style="width: 100%; max-width: 600px; margin: 0 auto; font-family: Arial, sans-serif; line-height: 1.4; color: #333333;">
    <tr>
        <td>
            <h1 style="font-size: 24px; color: #333333;">Nouveau message de {{ $nom }}</h1>
            <p style="font-size: 16px; color: #666666; margin-bottom: 16px;"><strong>Email:</strong> {{ $email }}</p>
            <h2 style="font-size: 24px; color: #333333;">Type de demande: {{ $demande }}</h2>
            <p style="font-size: 16px; color: #666666;"><strong>Message:</strong></p>
            <p style="font-size: 16px; color: #666666;">{{ $mess }}</p>
        </td>
    </tr>
</table>
</body>
</html>
```

Le mail sera alors envoyé, à condition d'avoir correctement configuré le fichier .env, soit en le paramétrant avec Gmail à des fins de test ou via STMP, mailGun ou un autre service permettant de gérer l'envoi de courriel.

14 – ShareErrorsFromSession

```
/**
 * Handle an incoming request.
 *
 * @param \Illuminate\Http\Request $request
 * @param \Closure $next
 * @return mixed
 */
public function handle($request, Closure $next)
{
    // If the current session has an "errors" variable bound to it, we will share
    // its value with all view instances so the views can easily access errors
    // without having to bind. An empty bag is set when there aren't errors.
    $this->view->share(
        'errors', $request->session()->get('errors') ?: new ViewErrorBag
    );

    // Putting the errors in the view for every view allows the developer to just
    // assume that some errors are always available, which is convenient since
    // they don't have to continually run checks for the presence of errors.

    return $next($request);
}
```

1.5 – Méthodes update et destroy

```
public function update(Request $request, $id)
{
    $data = $request->validate([
        'question' => 'string|required',
        'image' => 'file|required',
        'reponse' => 'string|required'
    ]);

    if ($data['image']) {
        $path = $data['image']->store('images', 'public');
        $data['image'] = 'storage/' . $path;
    }

    $quizz = Quizz::findOrFail($id);
    $quizzImage = $quizz->quizzable;

    $quizz->update($data);
    $quizzImage->update($data);

    return redirect()->back();
}
```

Figure 14: méthode Update du controller 'QuizzImage'

```
UPDATE quizzes SET reponse = 'reponse modifiée'
WHERE id = 275;

UPDATE `qcms` SET `choix1`='reponse a'
WHERE id = (SELECT quizzes.quizzable_id FROM quizzes WHERE quizzes.id = 275);
```

Figure 15: Exemple de requête SQL équivalente


```

public function destroy(string $id)
{
    $quizz = Quizz::findOrFail($id);

    if ($quizz->quizzable instanceof VraiFaux) {
        $quizz->quizzable->delete();
    }

    if ($quizz->quizzable instanceof QCM) {
        $quizz->quizzable->delete();
    }

    if ($quizz->quizzable instanceof QuizzImage) {
        $quizz->quizzable->delete();
    }

    $quizz->delete();

    return redirect()->back();
}

```

Figure 16: fonction delete du controller 'Quizz'

```

DELETE FROM quizzes
WHERE quizzes.quizzable_id = (SELECT vrai_fauxes.id FROM vrai_fauxes WHERE vrai_fauxes.id = 1);

DELETE FROM vrai_fauxes
WHERE id = 1;

```

Figure 17 : Exemple de requête SQL équivalente

16 – Template du composant Quizz

```
<template>
<section>
<planning :evenements="evenements"></planning>
<div class="mainQuizz">
<div v-for="(quizz, key) in quizzes"
  :key="key"
  v-show="show === key">
<div v-if="quizz.quizzable_type === 'QuizzImage'" class="containerImageQuizz">
  <div class="titreImageQuizz">Question Image</div>
  <div class="questionImage"> {{ quizz.question }}</div>
  
  <input class="input" type="text" v-model="reponse" @keyup.enter="checkReponse">
  
</div>
<div v-if="quizz.quizzable_type === 'QCM'" class="containerQcm">
  <div class="titreQcm">Choix multiples</div>
  <div class="question"> {{ quizz.question }}</div>
  
  <div class="reponseQcm" :class="{ selected: isSelected('A') }" @click="selectionnerChoix('A')"> A. {{ quizz.quizzable.choix1 }}</div>
  <div class="reponseQcm" :class="{ selected: isSelected('B') }" @click="selectionnerChoix('B')"> B. {{ quizz.quizzable.choix2 }}</div>
  <div class="reponseQcm" :class="{ selected: isSelected('C') }" @click="selectionnerChoix('C')"> C. {{ quizz.quizzable.choix3 }}</div>
  <div class="reponseQcm" :class="{ selected: isSelected('D') }" @click="selectionnerChoix('D')"> D. {{ quizz.quizzable.choix4 }}</div>
  <button @click="checkReponse">Vérifier</button>
</div>
<div v-if="quizz.quizzable_type === 'Vraifaux'" class="containerVf">
  <div class="titreVf">Vrais ou faux ?</div>
  <div class="question"> {{ quizz.question }}</div>
  <div class="reponse" @click="checkReponse('vrai')">
    
    <p><b>VRAI</b></p>
    
  </div>
  <div class="reponse" @click="checkReponse('faux')">
    
    <p><b>FAUX</b></p>
    
  </div>
</div>
</div>
<div v-if="show >= quizzes.length" class="resultatQuizz">
  <p>Vous avez fait un score de {{ points }}/{{ quizzes.length }} !</p>
</div>
</section>
</template>
```

leo, last month • 1708 ...

17 – Tests de quizz et de connexion

```
describe("echec de connexion",() =>{  
  beforeEach(()=>{  
    cy.visit("http://127.0.0.1:8000/admin");  
  })  
  
  it("mauvais mail",()=>{  
  
    cy.get('input[type="email"]').type('leondelmas31@gmail.com');  
    cy.get('input[type="password"]').type('12341234');  
    cy.get("form").submit();  
  
    cy.get('#errorEmail p').should('exist').invoke('text').should('include', 'These credentials do not match our records.');  })  
  
  it("mauvais mdp",()=>{  
  
    cy.get('input[type="email"]').type('leodelmas31@gmail.com');  
    cy.get('input[type="password"]').type('5555');  
    cy.get("form").submit();  
  
    cy.get('#errorEmail p').should('exist').invoke('text').should('include', 'These credentials do not match our records.');  })  
  
  it("both",()=>{  
  
    cy.get('input[type="email"]').type('leondelmas31@gmail.com');  
    cy.get('input[type="password"]').type('5555');  
    cy.get("form").submit();  
  
    cy.get('#errorEmail p').should('exist').invoke('text').should('include', 'These credentials do not match our records.');  })  
})
```

Figure 18: test de l'écran de connexion

```

it("QCM", () => {
  cy.get('input[type="email"]').type(Cypress.env('mail'));
  cy.get('input[type="password"]').type(Cypress.env('mdp'));

  cy.get("form").submit();
  cy.url().should("eq", "http://127.0.0.1:8000/admin");
  cy.contains("Quizz").click();
  cy.get(".form-button:contains('Ajouter un QCM')").click();
  cy.get("#question").type("Question de test qcm");
  cy.get("#choix1").type("test A");
  cy.get("#choix2").type("test B");
  cy.get("#choix3").type("test C");
  cy.get("#choix4").type("test D");
  cy.get("#reponse").type("reponse de test qcm");
  cy.get(".form-container").submit();
  cy.contains("Question de test qcm").next().contains("test A").next().contains("test B").next().contains("test C").next().contains("test D").next().contains("reponse de test qcm").should("exist");
  cy.visit("http://127.0.0.1:8000/quizz");
  cy.get('.containerQcm').contains("Question de test qcm").should('exist');
  cy.visit("http://127.0.0.1:8000/admin/quizz");
  cy.contains("Modifier").click();
  cy.get("#question").clear().type("Question de test qcm modifié");
  cy.get(".form-container").submit();
  cy.contains("Question de test qcm").next().contains("test A").next().contains("test B").next().contains("test C").next().contains("test D").next().contains("reponse de test qcm").should("exist");
  cy.visit("http://127.0.0.1:8000/quizz");
  cy.get('.containerQcm').contains("Question de test qcm").should('exist');
  cy.visit("http://127.0.0.1:8000/admin/quizz");

  cy.contains("Supprimer").click();

  cy.on('window:alert', (text) => {
    expect(text).to.equal('Supprimer le Quizz ?');
    return true;
  });

  cy.contains("Question de test qcm").should("not.exist");
});

```

```

it("quizz image", () => {
  cy.get('input[type="email"]').type(Cypress.env('mail'));
  cy.get('input[type="password"]').type(Cypress.env('mdp'));

  cy.get("form").submit();
  cy.url().should("eq", "http://127.0.0.1:8000/admin");
  cy.contains("Quizz").click();
  cy.get(".form-button:contains('Ajouter un Quizz image')").click();
  cy.get("#question").type("Question de test image");
  cy.get("#reponse").type("Reponse de test image");
  cy.get("#image").selectFile("cypress/fixtures/logochelou.png");
  cy.get(".form-container").submit();
  cy.get('img[alt="image"]').parent().next().contains("Question de test image").next().contains("Reponse de test image").should("exist");
  cy.visit("http://127.0.0.1:8000/quizz");
  cy.get('.questionImage').contains("Question de test image").should("exist");
  cy.visit("http://127.0.0.1:8000/admin/quizz");
  cy.contains("Modifier").click();
  cy.get("#question").clear().type("Question de test image modifié");
  cy.get("#image").selectFile("cypress/fixtures/logochelou.png");
  cy.get(".form-container").submit();
  cy.get('img[alt="image"]').parent().next().contains("Question de test image modifié").next().contains("Reponse de test image").should("exist");
  cy.visit("http://127.0.0.1:8000/quizz");
  cy.get('.questionImage').contains("Question de test image modifié").should("exist");
  cy.visit("http://127.0.0.1:8000/admin/quizz");

  cy.contains("Supprimer").click();

  cy.on('window:alert', (text) => {
    expect(text).to.equal('Supprimer le Quizz ?');
    return true;
  });

  cy.contains("Reponse de test image modifié").should("not.exist");

  cy.visit("http://127.0.0.1:8000/quizz");

  cy.get('.containerImageQuizz').should("not.exist");

  cy.visit("http://127.0.0.1:8000/admin/quizz");

});

```