

A vertical purple band on the left side of the page, featuring a repeating geometric pattern of interlocking hexagons and stars.

Mémoire

DEVELOPPEUR WEB

Formation Adrar

Présenté par :
Jérémy MUSE

SOMMAIRE :

I- Présentation.....	p 3
1)Introduction.....	p 3
2)Abstract.....	p 3
3) Stage N-PY.....	p 4
1.Présentation.....	p 4
2. Cadre de travail.....	p 4
3. Ma mission.....	p 5
4) Compétences couvertes par mon projet.....	p 6
5)Traductions.....	p 6
II- Projet fil rouge.....	p 8
1) Présentation.....	p 8
2) Taches effectuées.....	p 8
III- Spécifications fonctionnelles.....	p 10
1) Diagramme d'utilisation.....	p 10
2) Diagramme de séquence.....	p 11
3) Diagramme d'activité.....	p 12
IV- Conception.....	p 14
1)Arborescence.....	p 14
2) Maquettage Zoning.....	p 15
3) Maquettage Mockup.....	p 19
V- Base de données.....	p 23
1) MCD.....	p 23
2) MLD.....	p 24
VI- Back End : Projet Fil Rouge.....	p 28
1) Fonctionnalité d'inscription.....	p 28
2) Fonctionnalité d'upload et récupération de fichiers.....	p 33
VII- Front End : Calculateur N-PY.....	p 42
1) Structure et fonctionnement.....	p 42
2) Responsive.....	p 54
VIII- Conclusion.....	p 61
1) Les difficultés.....	p 61
2) Ce que j'ai appris.....	p 62
3) Pour la suite.....	p 63
IX- Annexe.....	p 64
X – Remerciements.....	p 70

I/ Présentation

1) Introduction

I'm Jérémy Muse and I am 28. I've been very curious about computers and new technologies since my childhood, but my taste in art and literature made me choose a literary bachelor.

The covid period was the perfect moment for me to start learning web development.

As I started the Adrar curriculum with absolutely no skills in development, I was a little worried but very soon I realised that all I needed to succeed was motivation and curiosity.

So I started to learn programming with Adrar in March 2021 and I discovered a new world, and definitely a new career. My internship in October helped me complete skills that I learned during class, but I worked also at home from my side. I deployed an e-commerce website called roulepopote.fr. This project helped me complete my skills in CMS, especially WordPress.

I'm very proud of my new skills and the improvement I made all along this year, and I hope you'll see all the energy and all the pleasure I had thanks to all my projects.

2) Abstract

During my internship, I worked on the remodeling of the company's current website called N-py. Today, for the front end part, I will present you a module that I made during my internship, and for the back end I will present a website that I made on my own.

The client is fictional, he represents an amusement park called Movie Land. This website is purely fictional, it will not be deployed online.

Specifications have been made in collaboration with the customer, you can find them in the appendice part.

For the front-end I chose HTML, CSS and Javascript languages, and for the back-end I have chosen NodeJS and SQL languages.

I think that the most difficult part for me was all the back end part except the database part because my internship missions were a lot about the front-end, so I think that back-end is my weakness. However, it was difficult but not impossible.

As you can see, the project is not finished, because a lot of functionalities have been provided. All those functionalities will be added in the future, they'll serve me to evaluate my progress.

I'm satisfied about what I have done, especially because before the training, I had literally no skills in programming. This is a great success for me, and I hope you will have the same opinion about my work.

3) Stage N-PY

1. Présentation

La société N-PY est une société ayant pour but de mettre en avant les domaines skiables des Pyrénées. Afin de mener à bien cet objectif, N-PY tache de centraliser les sites des stations de ski pyrénéennes et de certains hébergements autour de leur propre site. Ainsi, l'utilisateur a la possibilité d'organiser son séjour dans n'importe quelle station, à n'importe quelle période et pour n'importe quelle activité directement à partir du site n-py.com. En plus de cela, N-PY s'occupe de la création et de la maintenance des sites des stations, mais aussi d'hébergements pyrénéens ou même des sites de certaines mairies tels que le site de Cauterets ou de l'établissement Skylodge.

Grâce à ce partenariat mis en place avec différentes stations, plusieurs nouveaux systèmes d'organisation de séjour sont apparus dans les Pyrénées, tels que la Carte No'Souci permettant aux skieurs de réserver leurs forfaits à des prix bien plus attractifs qu'à l'accoutumée.

Pour résumer : N-PY permet à ses utilisateurs d'organiser plus facilement et plus économiquement leurs séjours au ski. Les stations de ski les plus importantes des Pyrénées jouant le jeu (Piau Engaly, Luz Ardiden, Peyragudes, La Pierre St-Martin, Cauterets, Grand Tourmalet, Gourette, Pic du Midi) ainsi que les hébergements locaux, et les lieux d'activités hors ski, proposent un grand panel de séjours disponibles, autant en hiver qu'en été. On peut également y trouver toutes les informations concernant les stations et les domaines pyrénéens : webcams, lignes de bus, tarifs, avis, etc.

2. Cadre de travail

L'équipe Digitale d'N-PY comporte six membres. Une personne occupe le poste de trafic manager, le reste de l'équipe est en charge de la conception, du développement front end, de l'intégration (sous Drupal 8) et de l'emailing du site et des sites partenaires. La partie back end est sous traitée à une autre agence. La majeure partie de l'équipe a d'énormes compétences en conception et intégration, le développement est en majeure partie réalisée par le responsable Digital.

Ces larges compétences partagées par les membres de l'équipe m'ont fait énormément progresser car bien qu'ayant un tuteur attiré, j'ai pu travailler et évoluer au contact de chacun des membres, y compris de la trafic manager qui a eu l'opportunité d'approfondir les minces compétences que j'avais acquis en SEO.

La majeure partie de mon stage s'est déroulée en présentiel, dans les bureaux situés à Lourdes. Ce qui m'a permis de découvrir comment se déroule un entretien avec un client concernant l'avancée d'un projet, les entretiens hebdomadaires avec l'équipe complète, etc.

J'ai également eu l'occasion de travailler en télétravail suite aux protocoles COVID mis en place (équivalent à 1 jour pour semaine) ce qui m'a permis de réaliser les contraintes techniques et humaines que peut engendrer cette façon de travailler, notamment en terme de communication.

3. Ma mission

Une grande partie de mon travail a été du développement modulaire. En effet, la plupart des membres de mon équipe étant des intégrateurs et n'ayant pas tous des capacités en développement, les modules que j'ai pu concevoir leur ont permis de gagner un temps précieux et sont réutilisables après mon départ de l'entreprise. J'ai donc manipulé les langages HTML, CSS et Javascript, le tout en me basant sur des maquettes réalisées par mes collègues ou moi-même grâce aux outils Adobe XD et Photoshop. En plus de ces modules réutilisables, j'ai pu réaliser un des projets phares du remodelage du site N-PY, à savoir un calculateur permettant de comparer les tarifs obtenus avec la carte No'Souci qui est leur première source de revenu, et les tarifs tout public. Je vous présenterai cette fonctionnalité lorsque l'on abordera la partie Front End du mémoire.

En plus de ces missions, j'ai également participé à l'intégration de ces modules grâce au CMS Drupal 8, ce qui m'a permis de me familiariser avec cet outil que je ne connaissais pas auparavant. L'approche de ce nouvel outil a été grandement facilitée par ma maîtrise du CMS Wordpress que j'ai pu découvrir durant mes cours et que j'ai put approfondir grâce à la création et le déploiement du site e-commerce roule popote (roulepopote.fr) réalisé sur mon temps libre pendant la formation.

Je peux affirmer sans l'ombre d'un doute que ces missions effectuées durant mon stage sont en majeure partie à l'origine de mes compétences en développement aujourd'hui.

4) Compétences couvertes par mon projet

Ce projet fil rouge (création d'un site pour un parc d'attractions fictif), a été pour moi un grand défi compte tenu du fait que je n'avais aucune compétence en développement avant mon début de formation. Cela m'a donc permis de me tester et également de réaliser sur quels aspects je peux avoir des lacunes.

Ce projet m'a donc permis de travailler sur de nombreuses compétences, à savoir la conception, en passant des diagrammes jusqu'au maquettage, du développement front end et back end, ainsi que de la création et d'échanges avec la base de données. En plus de cela, l'efficacité de mon travail en autonomie a été grandement améliorée grâce à toute la veille technologique que j'ai dû réaliser afin d'approcher certains concepts que je n'avais pas abordé en cours, notamment une fonctionnalité permettant d'upload des fichiers images puis les récupérer avec un autre utilisateur. Une fonctionnalité que je vous présenterai dans la partie Back End du mémoire.

5) Traductions

Comme demandé, je vous présente dans cette partie les traductions des recherches que je pourrais être emmené à faire durant ma carrière professionnelle.

The research I chose is « How to center a div » for a simple reason: I feel that junior and senior programmers, no matter the experience, will have this problem during their whole career. I chose a second research which is the pseudo class: hover because I think it's primordial to be capable of using it if you want to create a website with a modern appearance.

To horizontally center a block element (like <div>), use margin:auto ;

Setting the width of the element will prevent it from stretching out to the edges of its container.

The element will then take up the specified width, and the remaining space will be split equally between the two margins.

Source: https://www.w3schools.com/css/css_align.asp

Pour centrer horizontalement un bloc élément (tel que <div>), utilisez margin: auto;. Paramétrer la largeur de l'élément va l'empêcher de s'étirer jusqu'aux bords de son container.

L'élément va ensuite prendre la largeur spécifiée, et l'espace restant sera séparé équitablement entre les deux marges.

The `:hover` [CSS](#) pseudo-class matches when the user interacts with an element with a pointing device, but does not necessarily activate it. It is generally triggered when the user hovers over an element with the cursor (mouse pointer).

Source : <https://developer.mozilla.org/en-US/docs/Web/CSS/:hover>

La pseudo classe CSS `:hover` s'utilise quand l'utilisateur interagit avec un élément grâce à un dispositif de pointage, mais ne l'active pas forcément. Il est en général déclenché quand l'utilisateur survole un élément à l'aide de son curseur (de sa souris par exemple).

The `:hover` pseudo-class is problematic on touchscreens. Depending on the browser, the `:hover` pseudo-class might never match, match only for a moment after touching an element, or continue to match even after the user has stopped touching and until the user touches another element. Web developers should make sure that content is accessible on devices with limited or non-existent hovering capabilities.

Source : <https://developer.mozilla.org/en-US/docs/Web/CSS/:hover>

La pseudo classe `:hover` est problématique sur les écrans tactiles. En fonction du navigateur, la pseudo classe `:hover` pourrait ne jamais fonctionner, fonctionner uniquement pendant un laps de temps après avoir touché un élément, ou continuer à fonctionner même après que l'utilisateur ait cessé de toucher et jusqu'à ce qu'il touche un autre élément. Les développeurs Web devront s'assurer que cette fonctionnalité soit disponible sur les dispositifs ayant des capacités de « survol » limitées voire inexistantes.

II- Projet Fil Rouge

1) Présentation

En début de formation, il nous a été proposé l'exemple d'un site de parc d'attractions fictif comme projet fil rouge. Il se trouve en effet que ce projet recouvre un panel de fonctionnalités possibles permettant de travailler sur tous les sujets abordés en cours. J'ai donc choisi ce projet en me basant sur un parc d'attractions fictif ayant pour thème le cinéma. Le but de ce site est avant tout d'avoir une utilisation fiable et fluide quelle que soit la fonctionnalité approchée. Il a aussi pour but personnel de m'auto-évaluer sur le long terme car comme vous le verrez, toutes les fonctionnalités prévues ne sont pas encore présentes, et je compte continuer à travailler dessus.

Les utilisateurs ont la possibilité de s'inscrire et de se connecter sur le site, une fonctionnalité que je vous présenterai un peu plus tard. Ils ont également la possibilité de déposer des fichiers images illustrant leur séjour au parc. De plus, il est également possible de s'inscrire et se connecter en tant qu'admin du site. Ceux-ci pourront avoir un visuel des photos envoyées par les utilisateurs et pourront ainsi faire le choix de les utiliser ou non pour alimenter le site. Cette fonctionnalité vous sera également présentée.

Une base de données est également présente afin de stocker les informations concernant entre autres l'inscription des utilisateurs.

2) Taches effectuées

Ce projet a été pour moi un énorme défi compte tenu du fait que je n'avais pratiqué aucun Back End durant mon stage, et m'a permis de vérifier si j'avais bien compris certains concepts, mais aussi de revenir sur d'autres que j'avais peut-être un peu plus de mal à assimiler. Heureusement, ce que j'ai pu apprendre en cours et durant mon stage a été parfaitement complémentaire et m'a permis de bien avancer sur ce projet.

Tout d'abord, la conception des spécifications fonctionnelles a été réalisée grâce à Jmerise et StarUML. Une étape qui peut sembler faire perdre du temps aux premiers abords mais qui m'a permis au final un énorme gain de temps.

Ensuite, les outils utilisés concernant le maquettage sont Adobe XD ainsi que le site Canva.com . Une étape qui m'a paru fastidieuse notamment dû à mon manque de compétences avec les logiciels de design.

Enfin, les technologies utilisées sont l'HTML, le CSS, le Javascript et l'EJS pour la partie front. Côté back vous retrouverez les langages NodeJS, ainsi que le SQL

concernant les requêtes envoyées à la base de données qui est stockée via PHP MyAdmin. Le logiciel PostMan a également été utilisé afin de vérifier certaines requêtes serveur.

En plus de toutes les étapes nécessaires à la création d'un site internet, ce projet m'a poussé à de nombreuses heures de réflexion concernant la structure du projet mais également et tout simplement sur comment aborder certaines fonctionnalités que je ne maîtrisais pas du tout.

III- Spécifications fonctionnelles

Les diagrammes qui vont suivre font partie d'une de ces étapes qui m'ont paru me faire perdre du temps au début, puis m'ont fait réaliser le temps précieux que j'ai gagné au fur et à mesure de l'avancée du projet. Ces diagrammes sont pour moi une étape essentielle au bon déroulement d'un projet, surtout s'il est mené par une équipe de plusieurs personnes, permettant ainsi d'avoir tous la même vision de l'objectif final.

1) Diagramme d'utilisation

Le diagramme d'utilisation a été ma première étape lors de la création de ce projet. J'ai donc d'abord listé les fonctionnalités que je voulais voir apparaître sur le site, puis les ai organisées selon le diagramme d'utilisation, en y ajoutant un code couleur définissant les priorités si le projet était réalisé en équipe. Étant seul sur la création de ce site, les priorités étaient surtout les fonctionnalités que je comptais présenter lors de l'examen.

Le vert concerne les priorités élevées.

Le jaune les priorités moyennes.

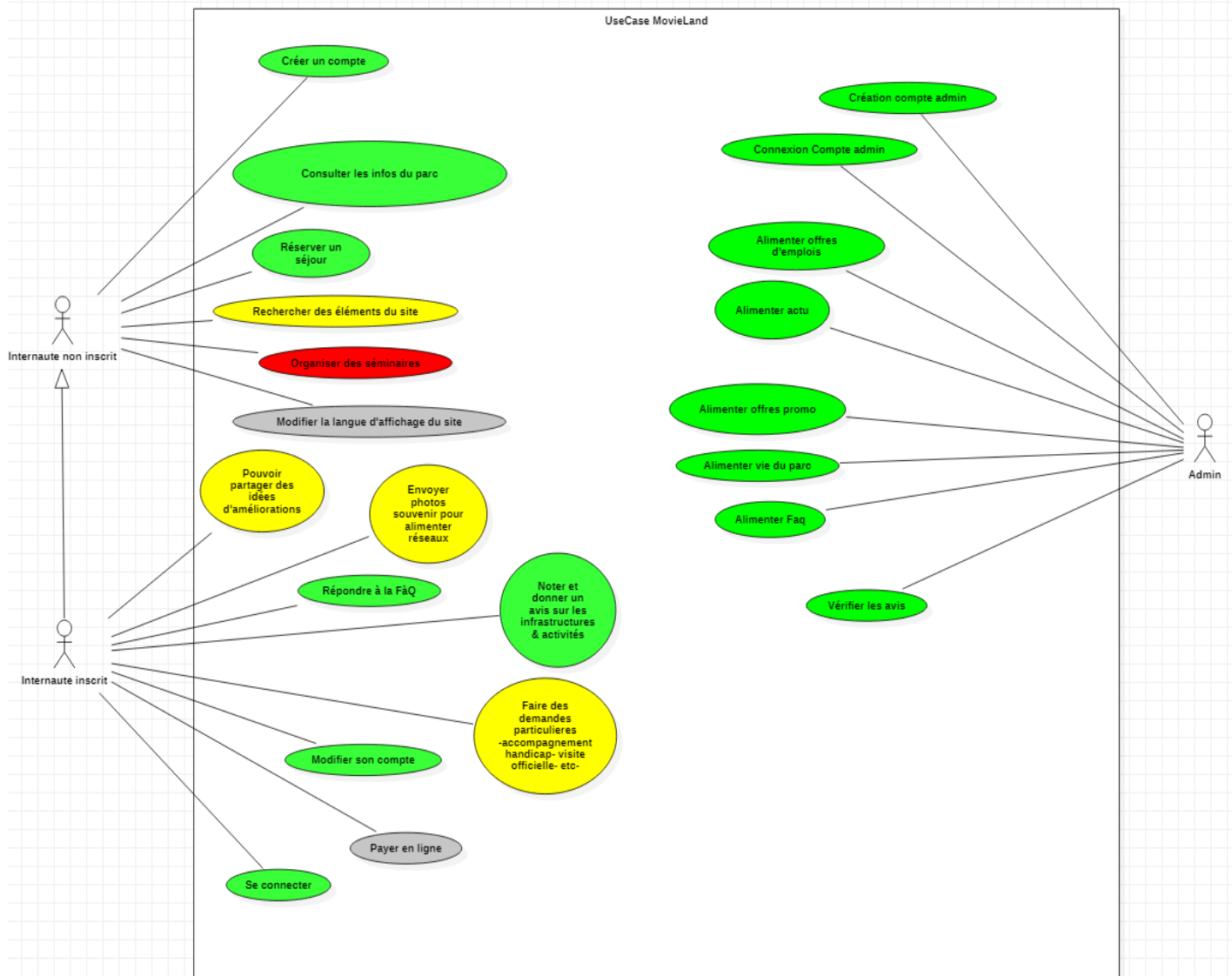
Le rouge et le gris les priorités faibles.

On peut trouver 3 acteurs différents dans ce diagramme.

Tout d'abord nous avons l'utilisateur non inscrit qui peut profiter des fonctionnalités de bases du site, puis en s'inscrivant il peut devenir le deuxième acteur de notre diagramme qui est l'utilisateur inscrit. Il hérite donc des fonctionnalités de notre premier acteur et peut profiter de nombreuses fonctionnalités supplémentaires telles que la possibilité de payer en ligne ou de modifier les informations de son compte.

Et enfin nous avons l'utilisateur admin qui lui peut bénéficier d'un accès back office du site, ce qui lui permettra par exemple de récupérer les fichiers envoyés par les internautes.

Ce diagramme a été la source de beaucoup de réflexion et a été sujet à de nombreuses modifications au cours de l'élaboration du projet. Il n'est pas impossible que par la suite, de nouvelles modifications y soient apportées au cours de l'ajout de nouvelles features.



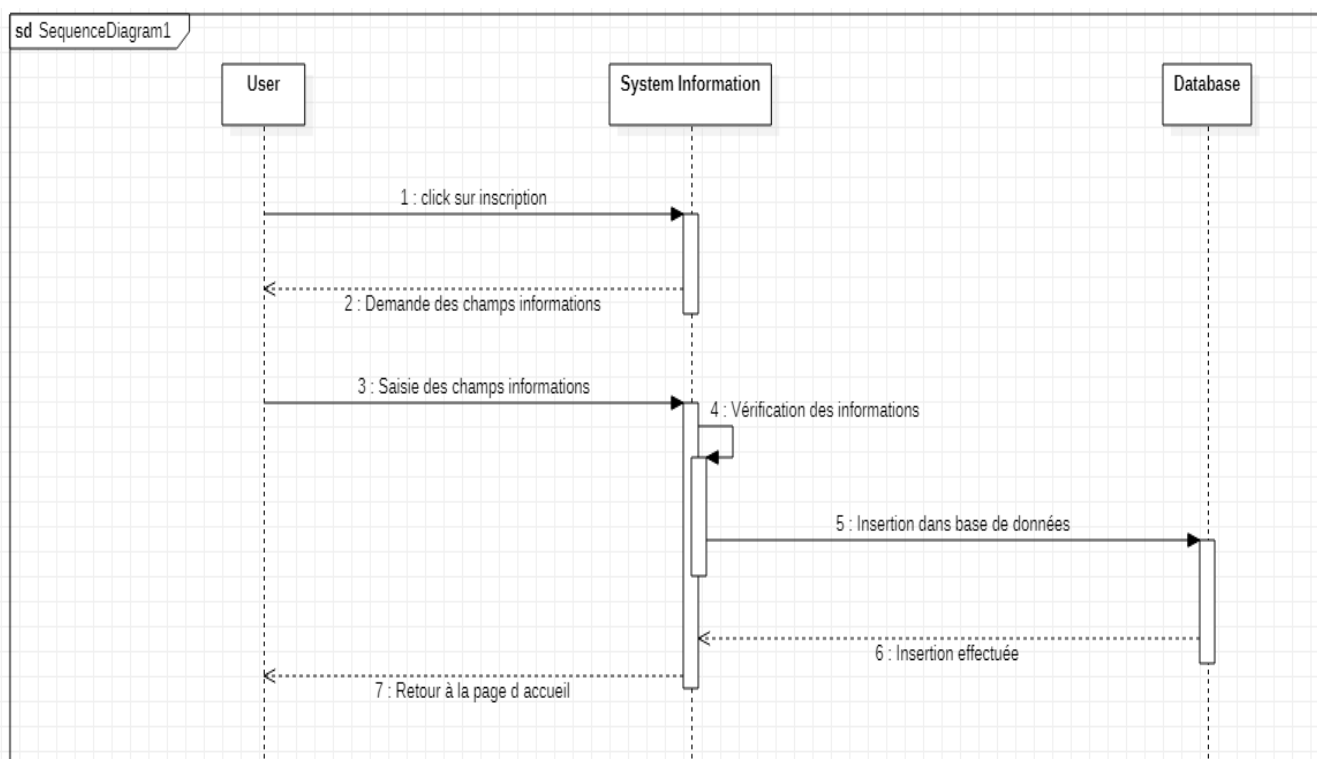
2) Diagramme de séquence

Le diagramme de séquence que je vais vous présenter concerne la fonctionnalité d'inscription du projet. Là également, j'ai pu réaliser que les diagrammes de séquence sont une étape primordiale dans l'élaboration d'un tel projet, surtout lorsque l'on est emmené à côtoyer des concepts compliqués côté serveur. En effet, schématiser de la sorte les échanges qu'il peut y avoir entre le client, le serveur et la base de données permet une approche plus rapide et plus efficace du développement.

On peut voir sur ce diagramme que dès que l'utilisateur demande un processus d'inscription, le serveur lui demande les informations nécessaires (tels que le pseudo,

le mail, le mot de passe, etc.). Le client entre donc les informations nécessaires et les envoie au serveur pour une vérification. Une fois la vérification faite, le serveur envoie une requête d'insertion à la base de données, qui traite la requête. Une fois cette dernière effectuée, la base de données envoie une réponse au serveur, à savoir que l'insertion a bien été effectuée, et le serveur renvoie le client vers la page d'accueil.

On peut voir que plusieurs schémas d'erreurs sont possibles lors de l'étape 4, celle de la vérification des informations. Les possibles schémas d'erreur sont 1 : que le mail entré par le client soit déjà pris, 2 : que le mot de passe admin ne correspondent pas (en partant du principe que c'est un admin qui tente de s'inscrire). Si le serveur rencontre ces schémas d'erreur, alors nous retournons à l'étape 2 qui est la demande des informations du client par le serveur.



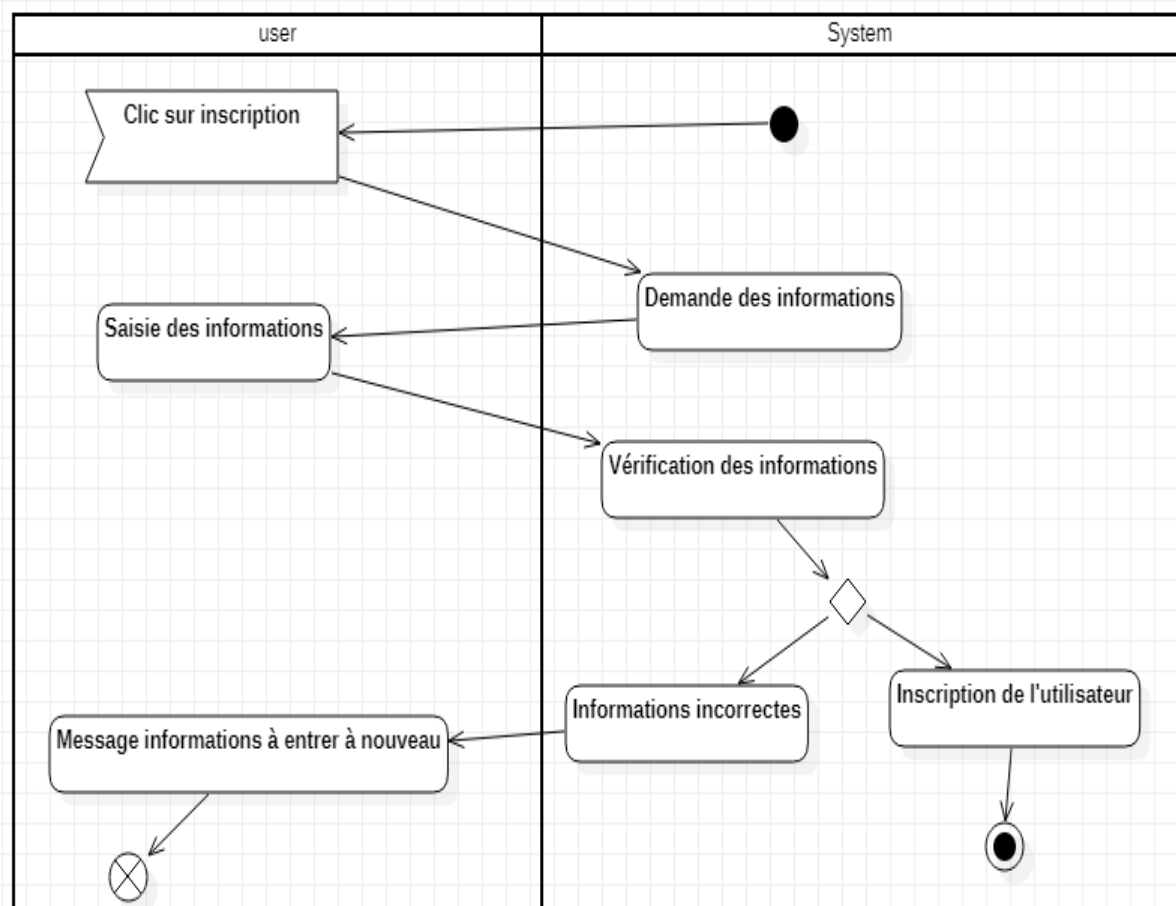
(4) Schémas d'erreur :
 L'inscription n'est pas prise en compte :
 1 : ce mail est déjà pris
 2 : le mdp admin ne correspond pas

3) Diagramme d'activité

Le diagramme d'activité suivant concerne également la fonctionnalité d'inscription.

Là encore, tout comme le diagramme de séquence, on se rend compte rapidement durant le développement que ce diagramme est une étape essentielle précédant le développement des fonctionnalités que l'on pourrait qualifier de complexes et facilite l'organisation au moment du développement.

On peut voir sur le diagramme que le point de départ n'est autre que le lancement du système qui attend une action de l'utilisateur. Ce dernier clique sur le bouton « inscription » qui envoie la demande d'inscription au système. Cependant pour effectuer cette tâche, le système a besoin d'informations, informations qu'il demande à l'utilisateur de renseigner. Une fois fait, le client envoie ces informations qui sont vérifiées par le système. Une fois la vérification effectuée, deux possibilités s'offrent à nous. Soit les informations sont erronées, dans ce cas le système demande au client de renseigner à nouveau les informations et répète l'opération jusqu'à ce que les informations soient bonnes, et si elles sont bonnes, dans ce cas le système procède à l'inscription de l'utilisateur.

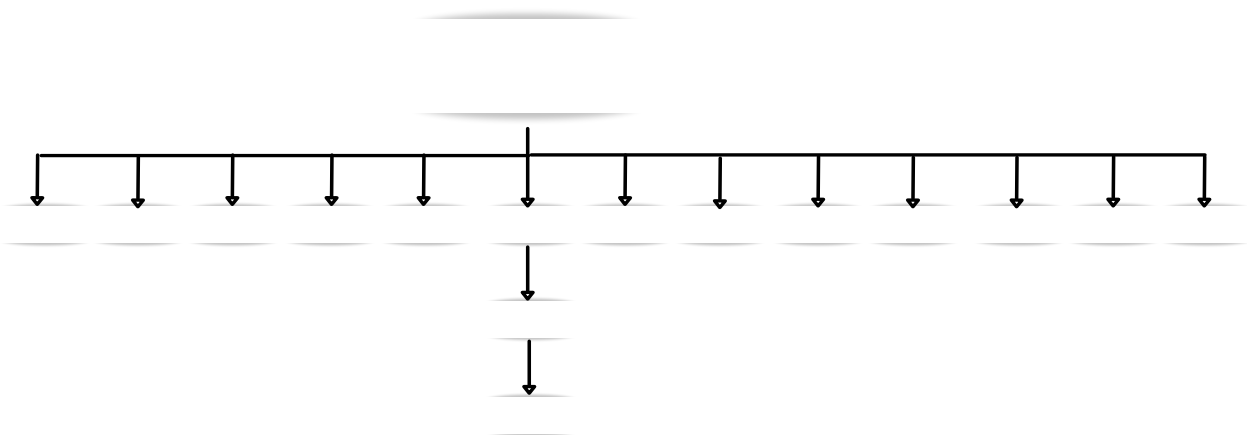


IV- Conception

1- Arborescence

Autant les diagrammes précédents m'ont permis de gagner un temps considérable sur le développement back end du projet, autant l'arborescence et les maquettes qui vont suivre m'ont également facilité le travail mais concernant le développement front end.

Après de longues réflexions et plusieurs modifications apportées, j'ai opté pour une arborescence simple et facile d'utilisation, avec une première page à partir de laquelle on peut accéder à absolument toutes les autres pages. Exception faite pour la page d'inscription. En effet depuis la page d'accueil, vous pouvez accéder à la page « Mon Compte », si vous êtes connecté, cela vous affichera les informations de votre compte, mais si vous n'êtes pas connecté alors cela vous renverra vers le formulaire de connexion. À partir de ce formulaire vous pouvez également accéder au formulaire d'inscription. Je pense que cette structure est une erreur de ma part, en effet, l'utilisateur doit entreprendre deux clics différents pour accéder au formulaire d'inscription, et le chemin n'est pas forcément clair. On peut considérer cette façon de faire comme n'étant pas user friendly, c'est pourquoi j'ai prévu de retravailler sur cet aspect là.



2- Maquettage zoning

Le maquettage en zoning est une étape primordiale de la conception, il donne un premier aperçu du rendu du site, mais ne contient en général aucun détail concernant les images, la police ou autre. Il sert en premier lieu à organiser l'emplacement des éléments prévus sur le site. Cette première maquette permet de gagner du temps à la fois lors de la conception de la maquette finale, et lors du développement front end.

En premier lieu, je vous présente la maquette zoning de la page d'accueil du site. On peut remarquer que le menu de navigation se trouve sur le haut de la page. À sa gauche, le logo du site est mis en avant. On peut également remarquer la présence de deux icônes sur la droite du menu de navigation.

Au centre de la page, on retrouve le slogan du site, parfaitement visible est mis en avant pour l'utilisateur.

Juste en dessous se trouve un menu simplifié, reprenant deux catégories importantes déjà présentes dans le menu de navigation. Trois images sont là pour donner du poids à ce sous menu.



En second lieu, voici la maquette zoning de la page d'inscription. Le haut de page reste le même que la page d'accueil avec son menu de navigation, le logo, et les deux icônes. Juste en dessous se trouve le titre de la page (à savoir « Inscription »). Et enfin on retrouve le formulaire d'inscription avec ses champs, ses labels ainsi qu'une

checkbox qui servira à savoir si l'utilisateur souhaite se connecter en tant qu'admin, puis le bouton permettant de soumettre leur formulaire.

	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	
	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	

Lorem Ipsum

Lorem Ipsum

Lorem Ipsum

Lorem Ipsum

Lorem Ipsum

Lorem Ipsum

Lorem Ipsum

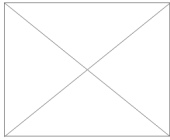
Lorem Ipsum

☐

Lorem Ipsum

Enfin, pour terminer avec le zoning, voici la maquette de la page « vie du parc ». Cette page aura deux rendus différents en fonction de qui est connecté : un utilisateur ou un admin.

Sur le haut de la page on retrouve toujours le même menu de navigation avec son logo et ses deux icônes. Juste en dessous, le titre de la page donne une indication à l'utilisateur d'où il se trouve sur le site (à savoir « vie du parc »).



Lorem Ipsum

Lorem Ipsum

Lorem Ipsum

Offres d'emploi

Lorem Ipsum

Lorem Ipsum



Lorem Ipsum

Lorem Ipsum

Lorem Ipsum

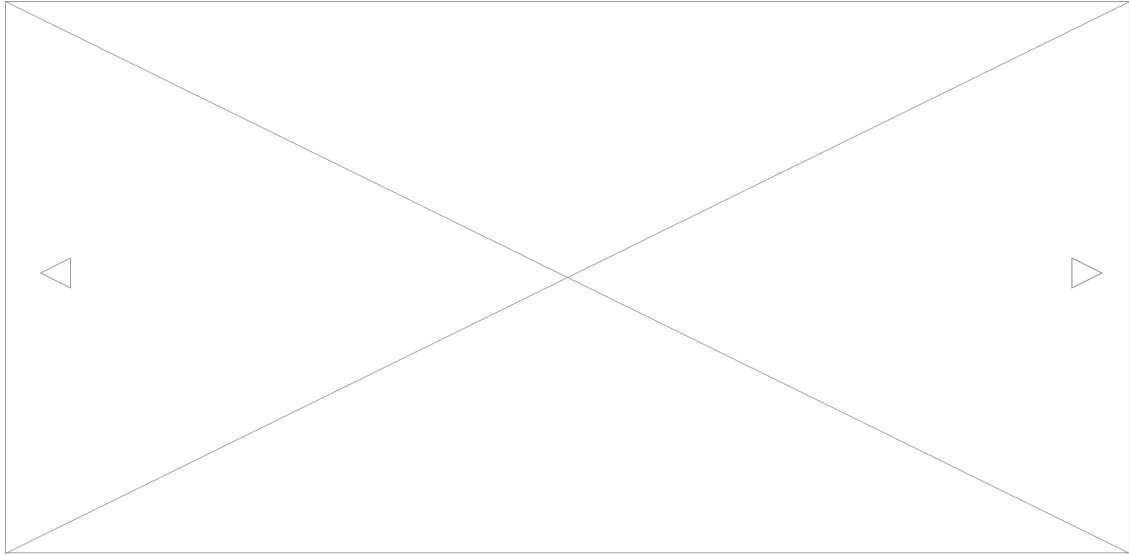
Lorem Ipsum

Lorem Ipsum

Lorem Ipsum



Lorem Ipsum



Lorem ipsum dolor sit amet ?

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Praesent tortor mauris, tristique eu rutrum.

Lorem Ipsum :

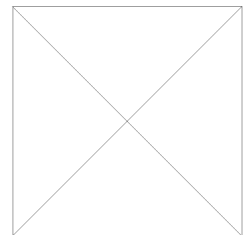
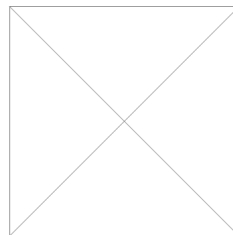
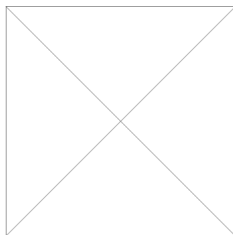
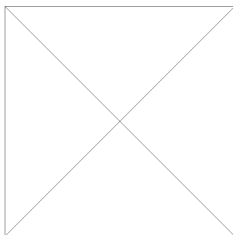
Lorem

Lorem Ipsum

Partie Admin

Lorem !

Lorem ipsum sit amet.



Sous le titre, nous trouvons un slider d'images, avec ses flèches permettant de passer d'une image à l'autre, ainsi que les points permettant de se situer dans le slider. Viens ensuite un court texte s'adressant aux utilisateurs, puis un input et son label (qui permettront d'envoyer un fichier), ainsi qu'un bouton permettant de valider l'envoi du fichier. Tout ceci est visible à la fois par l'utilisateur lambda et par l'admin, mais la section juste en dessous n'est accessible qu'aux personnes connectées en tant qu'admin. Il s'agit d'un aperçu des fichiers images envoyés par les utilisateurs. On peut voir qu'elles sont disposées en trois colonnes.

J'ai choisi de vous présenter ces trois maquettes car elles me paraissent indispensables à la présentation de ce projet. La première car la page d'accueil donne un bon aperçu d'ensemble de ce que sera le site, et les deux suivantes car ces pages concernent les deux fonctionnalités que je vais vous présenter plus tard.

2- Maquettage mockup

Le maquettage mockup vient compléter le maquettage zoning vu précédemment. En effet, là où le zoning ne donne qu'un aperçu simplifié du site en dévoilant seulement l'emplacement des éléments, le mockup, lui, donne le rendu final de ce que sera la page. C'est pour moi la maquette la plus importante de cette étape de conception. Elle permet de gagner un temps précieux lorsque nous nous attaquons au style de notre site, et donc de développer plus fluidement notre CSS sans passer de longues heures de réflexion sur des détails de design.

Je vais vous présenter les maquettes mockup des trois mêmes pages que précédemment pour le zoning.

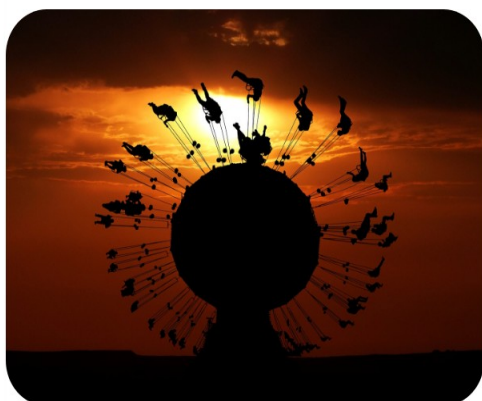
Pour cette maquette qui représente la page d'accueil, on peut remarquer que la structure est exactement la même que pour la maquette zoning. On y retrouve le menu de navigation en haut de page avec pour seules différences, le visuel du logo et des icônes, et les textes voulus sont maintenant présents. C'est également le cas pour le titre qui était auparavant généré en Lorem Ipsum comme pour le reste du texte des maquettes.

Enfin, sur la dernière partie de la page d'accueil, on retrouve le sous-menu avec le texte correspondant, et cette fois le visuel des images présentes dans le sous menu. On peut remarquer qu'un border-radius a été précisé sur ces images, permettant un visuel plus doux pour l'utilisateur.

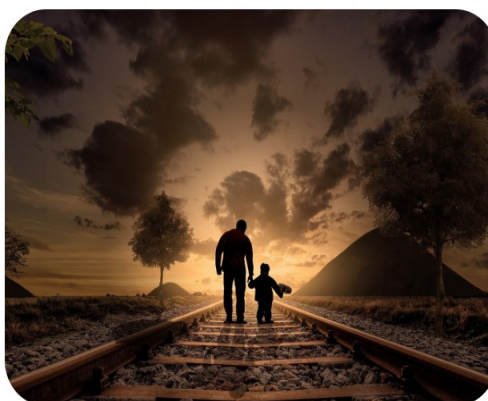
Sur la version prototype de la maquette, on peut également remarquer qu'un changement de couleur des liens a été précisé lorsque les liens sont survolés par le curseur de l'utilisateur.

Soyez acteur de votre propre film chez Movieland, le parc d'attractions qui vous plongera dans vos films préférés !

Nos Attractions



Nos Formules



Vie du Parc



Sur cette deuxième maquette concernant le formulaire d'inscription, là encore, rien n'a changé concernant la structure de la page.

On peut retrouver bien sur le menu de navigation en haut de page, avec là aussi les visuels voulus : logo et icônes. Le véritable texte a là encore remplacé le Lorem Ipsum. C'est également le cas pour le titre, et les labels contenus dans le formulaire d'inscription. Le bouton permettant de soumettre le formulaire contient lui aussi le bon texte.



S'inscrire

Entrez votre prénom

Entrez votre nom

Entrez votre date de naissance

Entrez votre adresse mail

Entrez votre n° de téléphone

Entrez votre mot de passe

Confirmez votre mot de passe

Cochez si vous vous connectez
en tant qu'admin

☐

S'inscrire

Enfin pour la troisième maquette mockup, on retrouve la page « vie du parc ». Là encore pas de surprise, la structure reste la même, et on retrouve le menu de navigation avec son logo et ses icônes. L'ensemble du Lorem Ipsum présent sur la page a été remplacé par le véritable texte.

De plus, une image a été ajoutée dans l'emplacement du slider. Les flèches et les points sont toujours présents pour signifier qu'il ne s'agit pas d'une simple image mais bien d'un slider.

Concernant la partie admin, les images n'ont pas été ajoutées car ce ne sont pas des images prédéfinies, elles varieront en fonction des fichiers envoyés par les utilisateurs.

V- Base de données

1) MCD

Le diagramme MCD suivant (Modèle Conceptuel de Données) est une étape primordiale lors de la conception d'un projet. En effet, il permet d'organiser au mieux la structure de la base de données et les échanges effectués entre les différentes tables.

Le diagramme MCD que je vais vous présenter a été réalisé grâce au logiciel jMerise.

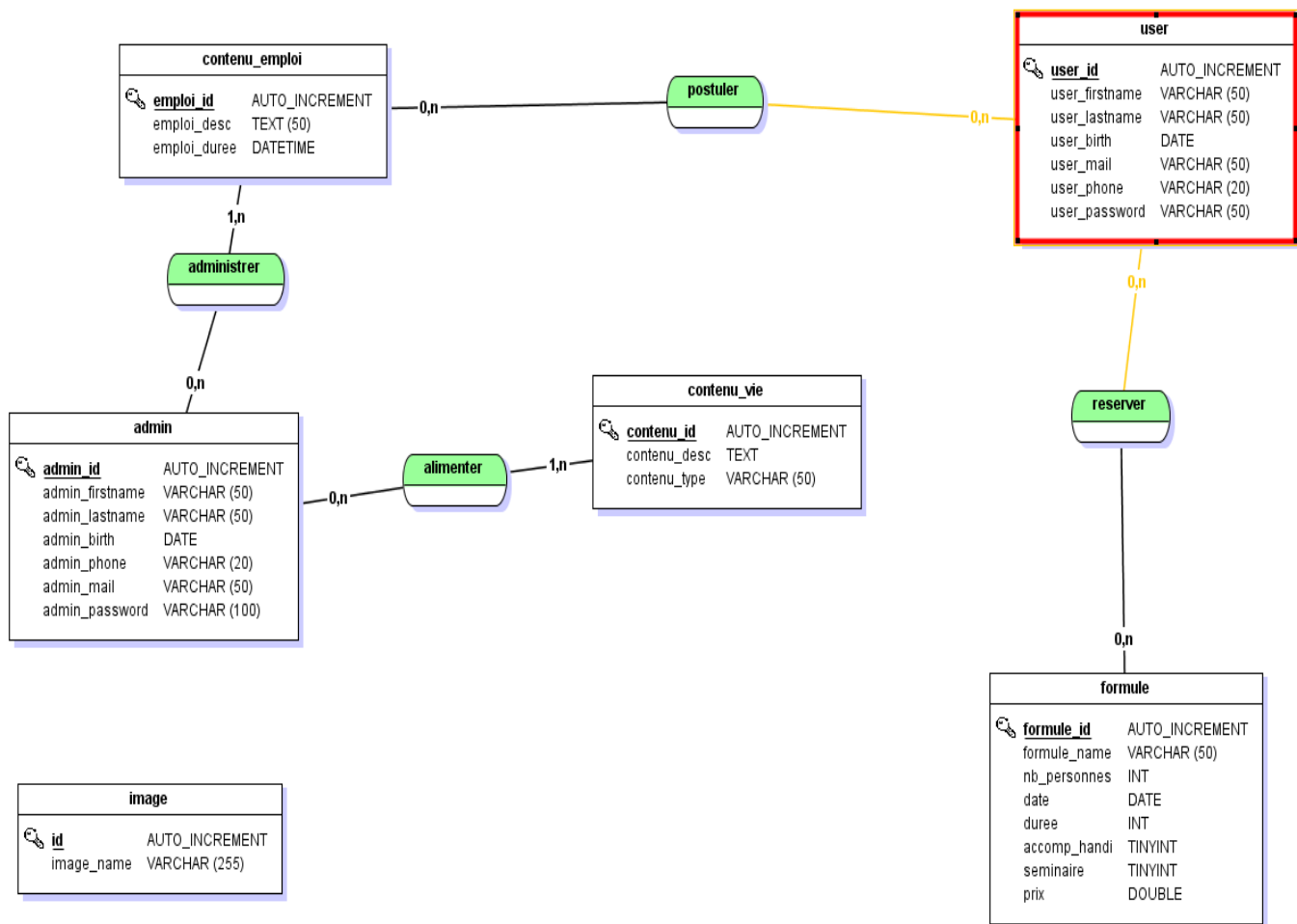
On peut compter au total six tables (user, formule, contenu_emploi, admin, contenu_vie et image) ainsi que quatre tables associatives (réserver, postuler, administrer et alimenter).

On remarque également la présence de cardinalités permettant de préciser les relations entre les deux tables reliées par leur table associative. Par exemple entre la table contenu_vie et la table admin se trouve la table associative alimenter. La cardinalité 0,n signifie qu'un admin peut gérer de 0 à une infinité de contenu_vie et la cardinalité 1,n signifie que le contenu_vie peut être géré par 1 à une infinité d'admin.

On peut apercevoir également sur chaque table la présence d'une ligne « id » avec le symbole d'une clé. C'est l'élément le plus important d'une table car il s'agit là de ce que l'on appelle une Primary Key. Cette clé s'auto-incrémente à chaque ajout d'une nouvelle donnée dans la base de données. Il s'agit donc de sa clé d'identification. Cette clé peut nous servir par exemple à cibler plus rapidement une donnée que l'on recherche.

Cette vision d'ensemble du fonctionnement de la base de données nous permet une meilleure organisation et une facilité d'approche du développement back end. Cela permet également à une équipe d'avoir la même vision des échanges effectués entre les différentes tables de la base de données.

Il s'agit là de la version définitive de ce diagramme MCD. De nombreuses modifications y ont été apportées au cours de l'avancée du projet, certaines tables ont été supprimées ou ajoutées, pareillement en ce qui concerne les tables associatives. De mon point de vue, il est fortement déconseillé de créer la base de données et les tables qui vont avec avant même d'établir ce diagramme.



2- MLD

Le MLD (Modèle Logique de Données) permet de créer et de décrire la structure de la base de données que l'on veut créer. Ce modèle est écrit en langage SQL. Le SQL n'est pas un langage de programmation à proprement parler, mais reste toutefois un langage informatique, il est utilisé dans la création et la gestion d'une base de données.

24

```

1  DROP DATABASE movieland IF EXISTS;
2  CREATE DATABASE movieland;
3
4  CREATE TABLE user(
5      user_id      Int Auto_increment NOT NULL ,
6      user_firstname Varchar (50) NOT NULL ,
7      user_lastname Varchar (50) NOT NULL ,
8      user_birth    Date NOT NULL ,
9      user_mail     Varchar (50) NOT NULL ,
10     user_phone    Varchar (20) ,
11     user_password  Varchar (50) NOT NULL
12     ,CONSTRAINT user_PK PRIMARY KEY (user_id)
13 )ENGINE=InnoDB;
14
15
16
17  CREATE TABLE admin(
18     admin_id      Int Auto_increment NOT NULL ,
19     admin_firstname Varchar (50) NOT NULL ,
20     admin_lastname Varchar (50) NOT NULL ,
21     admin_birth    Date NOT NULL ,
22     admin_phone    Varchar (20) ,
23     admin_mail     Varchar (50) NOT NULL ,
24     admin_password Varchar (100) NOT NULL
25     ,CONSTRAINT admin_PK PRIMARY KEY (admin_id)
26 )ENGINE=InnoDB;
27
28
29  CREATE TABLE formule(
30     formule_id     Int Auto_increment NOT NULL ,
31     formule_name   Varchar (50) NOT NULL ,
32     nb_personnes   Int NOT NULL ,
33     date           Date NOT NULL ,
34     duree          Int NOT NULL ,
35     accomp_handi   TinyINT NOT NULL ,
36     seminaire      TinyINT NOT NULL ,
37     prix           Double NOT NULL
38     ,CONSTRAINT formule_PK PRIMARY KEY (formule_id)
39 )ENGINE=InnoDB;
40

```

```

42 CREATE TABLE contenu_emploi(
43     emploi_id Int Auto_increment NOT NULL ,
44     emploi_desc Text NOT NULL ,
45     emploi_duree Datetime NOT NULL
46     ,CONSTRAINT contenu_emploi_PK PRIMARY KEY (emploi_id)
47 )ENGINE=InnoDB;
48
49
50
51 CREATE TABLE contenu_vie(
52     contenu_id Int Auto_increment NOT NULL ,
53     contenu_desc Text NOT NULL ,
54     contenu_type Varchar (50) NOT NULL
55     ,CONSTRAINT contenu_vie_PK PRIMARY KEY (contenu_id)
56 )ENGINE=InnoDB;
57
58
59
60 CREATE TABLE image(
61     id Int Auto_increment NOT NULL ,
62     image_name Varchar (255)
63     ,CONSTRAINT image_PK PRIMARY KEY (id)
64 )ENGINE=InnoDB;
65
66
67
68 CREATE TABLE reserver(
69     formule_id Int NOT NULL ,
70     user_id Int NOT NULL
71     ,CONSTRAINT reserver_PK PRIMARY KEY (formule_id,user_id)
72
73     ,CONSTRAINT reserver_formule_FK FOREIGN KEY (formule_id) REFERENCES formule(formule_id)
74     ,CONSTRAINT reserver_user0_FK FOREIGN KEY (user_id) REFERENCES user(user_id)
75 )ENGINE=InnoDB;
76
77
78
79 CREATE TABLE administrer(
80     emploi_id Int NOT NULL ,
81     admin_id Int NOT NULL
82     ,CONSTRAINT administrer_PK PRIMARY KEY (emploi_id,admin_id)
83
84     ,CONSTRAINT administrer_contenu_emploi_FK FOREIGN KEY (emploi_id) REFERENCES contenu_emploi(emploi_id)
85     ,CONSTRAINT administrer_admin0_FK FOREIGN KEY (admin_id) REFERENCES admin(admin_id)
86 )ENGINE=InnoDB;
87
88
89
90 CREATE TABLE postuler(
91     user_id Int NOT NULL ,
92     emploi_id Int NOT NULL
93     ,CONSTRAINT postuler_PK PRIMARY KEY (user_id,emploi_id)
94
95     ,CONSTRAINT postuler_user_FK FOREIGN KEY (user_id) REFERENCES user(user_id)
96     ,CONSTRAINT postuler_contenu_emploi0_FK FOREIGN KEY (emploi_id) REFERENCES contenu_emploi(emploi_id)
97 )ENGINE=InnoDB;
98
99
100
101 CREATE TABLE alimenter(
102     contenu_id Int NOT NULL ,
103     admin_id Int NOT NULL
104     ,CONSTRAINT alimenter_PK PRIMARY KEY (contenu_id,admin_id)
105
106     ,CONSTRAINT alimenter_contenu_vie_FK FOREIGN KEY (contenu_id) REFERENCES contenu_vie(contenu_id)
107     ,CONSTRAINT alimenter_admin0_FK FOREIGN KEY (admin_id) REFERENCES admin(admin_id)
108 )ENGINE=InnoDB;
109
110

```

À la première ligne de ce MLD on peut trouver la requête « DROP TABLE ‘...’ IF EXISTS » qui permet d’écraser la base de données s’il en existe une du nom que l’on a précisé, à savoir movieland. Cette manipulation est faite pour une simple raison, juste après, nous procédons à une autre requête qui est « CREATE DATABASE movieland ». Si une base de données existe déjà avec ce nom, et que l’on ne procède pas au DROP DATABASE juste avant, alors la création de la base de données

générera une erreur car elle refuse de créer une base de données avec un nom déjà pris par une autre base de données.

Une fois ces deux requêtes effectuées, notre base de données est créée. Il ne reste plus qu'à créer nos tables. Ce que nous allons faire (exemple à la ligne 4 du MLD) grâce à la requête « CREATE TABLE » à laquelle nous ajoutons le nom de la table voulue. On ouvre ensuite les parenthèses pour ajouter à la table les lignes de données voulues. On commence donc par l'id, qui est la primary key.

La première étape lors de l'insertion de la ligne est de la nommer, pour cet exemple, la ligne se nomme user_id. Vient ensuite le typage des données voulues dans la ligne. On entend par là si les données seront une valeur numérique, une chaîne de caractères, une date, etc. Lorsque la ligne en question concerne une primary key, le typage ciblera obligatoirement des nombres entiers. C'est pourquoi nous ajoutons le typage « int » qui cible le typage en nombre entiers. Nous ajoutons ensuite le paramètre « Auto_increment » qui va permettre à la valeur de l'id de s'auto-incrémenter à chaque ajout d'une nouvelle donnée. On ajoute ensuite la valeur « NOT NULL » qui précise que l'on n'accepte pas de valeur nulle en tant qu'id.

Une fois ceci fait, on peut ajouter les lignes suivantes en gardant le même principe, à quelques différences près selon le typage choisi. Par exemple pour la ligne suivante user_firstname (ligne 6 du MLD), nous avons le typage VARCHAR qui désigne une chaîne de caractères. On peut remarquer en suivant une valeur numérique entre parenthèses : (50). Il s'agit là du nombre de caractères maximum autorisés pour cette ligne. Certains typages ont obligatoirement une valeur précisant la longueur maximum de la donnée.

Nous répétons donc l'opération pour chaque ligne que l'on veut dans sa table, une fois ceci fait, on insère une dernière ligne qui permet de définir laquelle des lignes entrées précédemment sera la primary key. Il s'agit de la requête « ,CONSTRAINT user_PK PRIMARY KEY (user_id) » (ligne 12 du MLD). Le terme « user_pk » définit à quelle table nous faisons référence, tandis que le terme « (user_id) » définit quelle ligne de cette table sera la primary key.

Une fois ceci fait, on peut fermer la parenthèse ouverte juste après la création de la table, puis nous ajoutons « ENGINE=InnoDB; » qui définit le moteur de stockage utilisé. Il peut varier en fonction des systèmes de gestion de base de données.

Nous avons donc créé notre première table. Nous pouvons répéter le procédé pour chaque table principale.

À la ligne 68 du MLD, nous remarquons la création de notre première table associative. On commence par entrer la requête « CREATE TABLE reserver » qui lance la création de la table. On ouvre ensuite les parenthèses, et cette fois ci, au lieu

d'entrer les lignes voulues dans la table, nous insérons les ids des tables que la table associative relie. Dans notre cas il s'agit des tables formule et user. Nous entrons donc en première ligne « formule_id Int NOT NULL ». Il s'agit donc du nom de la primary key de la table formule, son typage, et la valeur NOT NULL précisant que cette ligne n'accepte pas d'id égal à zéro. On répète l'opération pour la deuxième table qui est reliée.

On rajoute ensuite la ligne « ,CONSTRAINT reserver_formule_FK FOREIGN KEY (formule_id) REFERENCES formule(formule_id) » et une deuxième ligne « ,CONSTRAINT reserver_user0_FK FOREIGN KEY (user_id) REFERENCES user(user_id) ». Ces deux lignes permettent de relier les deux ids des tables formule et user grâce à la table associative réserver.

On ferme ensuite la parenthèse puis on rajoute, comme lors des tables précédentes, la ligne « ENGINE=InnoDB; » pour définir le moteur de stockage utilisé.

Pour résumer dans ce MLD nous utilisons peu de requêtes, qui sont le DROP DATABASE, le CREATE DATABASE et le CREATE TABLE. Mais de nombreuses autres requêtes existent, et seront utilisées dans l'élaboration de ce projet durant le développement back end.

On peut rencontrer les requêtes « ALTER TABLE » qui permet de modifier une table, « INSERT INTO » qui permet d'insérer des données dans la table, ou encore la requête « SELECT 'nom_de_la_ligne' FROM 'nom_de_la_table' » qui permet de rechercher une donnée ou plusieurs données en particulier dans une table. Ces requêtes sont les requêtes les plus souvent rencontrées mais il en existe encore bien d'autres.

VI – Back End

1) Fonctionnalité d'inscription

La première fonctionnalité que je vais vous présenter est une fonctionnalité d'inscription. Les utilisateurs du site peuvent s'inscrire via un formulaire. Les admins du site s'inscrivent via le même formulaire à une petite différence près, une checkbox à cocher si l'on est admin permet de faire apparaître un input supplémentaire afin de rentrer le mot de passe réservé aux admins.

Ci-dessous, le formulaire d'inscription html :

```
1 <form class='formInscript' action="/adduser" method="POST">
2   <div>
3     <label for ="firstname">Entrez votre prénom:</label>
4     <input type="text" id="firstnameInsc" name="firstname" placeholder="Entrez votre prénom" autofocus required/>
5   </div>
6   <div>
7     <label for ="lastname">Entrez votre nom:</label>
8     <input type="text" id="lastnameInsc" name="lastname" placeholder="Entrez votre nom" required/>
9   </div>
10  <div>
11    <label for= "birth">Entrez votre date de naissance: </label>
12    <input type="date" id="birthInsc" name="birth" placeholder="Entrez votre date de naissance" required/>
13  </div>
14  <div>
15    <label for ="mail">Entrez votre adresse mail:</label>
16    <input type="email" id="mailInsc" name="mail" placeholder="Entrez votre adresse mail" required/>
17  </div>
18  <div>
19    <label for ="phone">Entrez votre n° de téléphone:</label>
20    <input type="tel" id="phoneInsc" name="phone" placeholder="Entrez votre numéro de téléphone"/>
21  </div>
22  <div>
23    <label for ="password">Entrez votre mot de passe: </label>
24    <input type="password" id="passInsc" name="password" placeholder="Entrez votre mot de passe" required/>
25  </div>
26  <div>
27    <label for="confirmPwd">Confirmation du mot de passe:</label>
28    <input type="password" id="confirmPwd" name="confirmPwd" placeholder="Confirmer votre mot de passe" required />
29  </div>
30  <div class="divAdmin">
31    <label for= "adminCase">Cochez si vous vous inscrivez en tant qu'admin: </label>
32    <input type="checkbox" id="adminCaseInsc" name="adminCase" placeholder="Cochez si vous vous connectez en tant qu'admin" />
33  </div>
34  <div id="divbtn">
35    <input class="btn" type="submit" id="signUp" value="S'inscrire">
36  </div>
37 </form>
```

Les attributs name de chaque input permettront par la suite de récupérer les valeurs entrées par l'utilisateur dans les champs et de les insérer ou les comparer aux données inscrites dans la base de données.

À la première ligne du formulaire d'inscription, on peut remarquer l'attribut « action » avec pour paramètre « /adduser ». Ce procédé permet l'envoi des informations entrées dans le formulaire pour les récupérer lorsque l'url indique la route /adduser.

À la ligne 31 et 32 on remarque un input de type checkbox. Le label indique que cette checkbox doit être cochée par l'utilisateur s'il souhaite s'inscrire en tant qu'admin. Une fois la case cochée, un script Javascript s'enclenche afin de rendre visible un input supplémentaire permettant d'entrer un mot de passe réservé aux admins.

Ci dessous le code Javascript en question :

```
1  let inp = document.createElement('input');
2  let adminInsc = document.querySelector("#adminCaseInsc");
3  let divAdmin = document.querySelector('.divAdmin');
4
5  adminInsc.addEventListener('change', evt => {
6    if (adminInsc.checked == true){
7      divAdmin.append(inp);
8      inp.setAttribute("type", "password");
9      inp.setAttribute("id", "passAdmin");
10     inp.setAttribute("placeholder", "Entrez le mot de passe Admin ");
11     inp.setAttribute("name", "passAdmin");
12
13   }else{
14     let passAdmin = document.querySelector("#passAdmin");
15     passAdmin.remove();
16   }
17 }) ;
```

Une fois ceci fait, nous avons posé les bases pour établir notre fonctionnalité d'inscription. Nous allons donc maintenant devoir nous attaquer au cœur du problème, à savoir, récupérer côté serveur les informations entrées par le client, les comparer à celles présentes dans la base de données puis les insérer dans la base de données si aucun utilisateur n'est déjà inscrit avec ces informations là.

Pour le côté serveur nous allons donc utiliser du NodeJS qui est une plateforme logicielle libre en Javascript.

La première étape est donc de récupérer toutes les informations envoyées par le client. Nous allons pour cela nous servir de la méthode POST entrée dans le formulaire et de l'attribut action='/adduser' lui aussi entré dans le formulaire.

Vous pouvez remarquer à la première ligne du code ci-dessous que nous faisons appel à la méthode POST entrée dans le formulaire HTML précédemment. On y ajoute la route d'url '/adduser' afin de bien réceptionner les informations que le formulaire a envoyé vers cette route. On entre ensuite entre parenthèse en paramètres les termes

« req, res » qui signifient requête et réponse. On ouvre ensuite l'accolade qui ouvre la fonction. Les requêtes entrées précédemment vont nous permettre de récupérer les informations entrées par le client dans le formulaire. On instancie donc une variable pour chaque information comme par exemple à la ligne 2 où il est inscrit : « `const fName = req.body.firstname ;` ». Le `req.body` fait référence au body de notre page d'inscription puis le `.firstname` fait référence à l'input ayant pour name « `firstname` » dans le formulaire d'inscription. Pour résumer, si dans l'input `firstname` j'entre le nom « Morgane » alors `fName` sera égal à Morgane. On répète ensuite l'opération pour chaque information que l'on souhaite récupérer.

À la ligne 10 et 11 on initialise ensuite deux variables qui contiendront ensuite deux requêtes SQL.

À la ligne 13 on vérifie si la checkbox réservée aux admins dans le formulaire d'inscription est cochée. Si elle ne l'est pas, alors on insère dans la variable `query` une requête SQL qui permet de vérifier si un utilisateur est déjà inscrit dans la table `user` avec le mail inscrit dans le formulaire. Le paramètre `LIMIT 1` demande à la base de données de stopper la recherche dès qu'elle trouve un résultat. Cela empêche de parcourir la totalité d'une base de données conséquente inutilement. Si la checkbox est cochée, alors on procède à la même opération mais la requête concernera cette fois la table `admin`.

À la ligne 19, on vérifie également si la checkbox est cochée mais on vérifie aussi si le mot de passe admin est le bon. Si c'est le cas, alors la variable `insert` contiendra une requête SQL qui insérera les informations dans la table `admin`. Si ce n'est pas le cas, alors l'insertion se fera dans la table `user`. Les informations à entrer grâce à cette requête sont indiquées grâce aux constantes que nous avons déclaré un peu plus tôt de la ligne 2 à 9.

À la ligne 27 nous exécutons la requête `query` que nous avons initialisée un peu plus tôt. Cette requête va nous renvoyer soit une erreur, soit un résultat sous forme de tableau. Si la longueur du tableau est égale à 0, alors cela signifie qu'aucun utilisateur n'est déjà inscrit avec ces informations. Dans ce cas, nous exécutons la requête `insert` à la ligne 33. Là encore cette requête nous renvoie soit une erreur, soit un résultat contenu dans la variable `result`. Nous demandons ensuite si la base de données a été affectée par la requête grâce à la méthode `result.affectedRows`. Si c'est le cas, alors l'inscription dans la base de données a bien été effectuée et le serveur nous renvoie sur la page d'accueil grâce au `res.redirect('/')`. Si ce n'est pas le cas, alors le serveur nous redirige vers une page dédiée aux messages d'erreurs.

Si par contre la longueur du tableau est différente de 0, ce que l'on vérifie à la ligne 29, alors le serveur nous redirige vers la page dédiée aux messages d'erreurs mais avec un message différent.

Une fois tout ceci effectué, alors deux résultats s'offrent à nous, soit nous avons effectué l'inscription de l'internaute avec succès, soit l'internaute a été renvoyé vers une page lui affichant un message d'erreur.

```
1 .post("/adduser", (req, res) => {
2   const fName = req.body.firstname;
3   const lName = req.body.lastname;
4   const birth = req.body.birth;
5   const mail = req.body.mail;
6   const phone = req.body.phone;
7   const pwd = req.body.password;
8   const adminC = req.body.adminCase;
9   const passAdmin = req.body.passAdmin;
10  let query;
11  let insert;
12
13  if (adminC !== 'on') {
14    query = `SELECT user_id FROM user WHERE user_mail = "${mail}" LIMIT 1;`;
15  } else {
16    query = `SELECT admin_id FROM admin WHERE admin_mail = "${mail}" LIMIT 1;`;
17  }
18
19  if (adminC === 'on' && passAdmin === adminPassword) {
20    insert = `INSERT INTO admin
21      (admin_firstname, admin_lastname, admin_birth, admin_mail, admin_phone, admin_password) VALUES ("${fName}", "${lName}", "${birth}", "${mail}", "${phone}", "${pwd})";`;
22  } else {
23    insert = `INSERT INTO user
24      (user_firstname, user_lastname, user_birth, user_mail, user_phone, user_password) VALUES ("${fName}", "${lName}", "${birth}", "${mail}", "${phone}", "${pwd})";`;
25  }
26
27  con.query(query, (errorSlt, users) => {
28    if (errorSlt) throw errorSlt;
29    if (users.length !== 0) {
30      console.log(users);
31      res.redirect('/error/1');
32    } else {
33      con.query(insert, (error, result) => {
34        if (error) {
35          console.log("error insc");
36        }
37        if (result.affectedRows) {
38          res.redirect('/');
39        } else {
40          res.redirect('/error/2');
41        }
42      });
43    }
44  });
45 })
```

Maintenant que nous avons vu comment inscrire l'internaute, nous allons voir comment définir le message d'erreur que doit afficher notre page.

Le code ci-dessous nous présente la méthode employée. La méthode `.get('/error/:id')` permet de récupérer l'url de la page, à savoir `/error`. Le terme `:id` correspond au cas d'erreur que nous souhaitons voir affiché. Nous initialisons ensuite les variables dans lesquelles nous stockerons les raisons de l'erreur, le chemin étant à l'origine de l'erreur ainsi que le message d'erreur correspondant. Nous procédons ensuite à la méthode `switch` pour définir le contenu de ces variables. En l'occurrence, case 1 sera le cas d'erreur n°1 et correspondra à l'url `/error/1` et ainsi de suite. Nous paramétrons également un cas d'erreur par défaut au cas où l'utilisateur accède à cette page en tapant directement `/error` dans l'url de son navigateur.

Une fois ceci fait, nous demandons à ce que le fichier error.ejs soit généré en tant que rendu, et on envoie également les informations concernant les cas d'erreurs.

```
1
2 .get('/error/:id', (req, res) => {
3   let error, returnPath, returnMsg;
4   switch (parseInt(req.params.id)) {
5     case 1:
6       error = "Ce mail est déjà pris.";
7       returnPath = "/inscription";
8       returnMsg = "retour au formulaire d'inscription";
9       break;
10    case 2:
11      error = "une erreur est survenue lors de l'inscription. veuillez recommencer.";
12      returnPath = "/inscription";
13      returnMsg = "retour au formulaire d'inscription";
14      break;
15    case 3:
16      error = "ce couple mail / mot de passe n'existe pas dans la base de données.";
17      returnPath = "/connection";
18      returnMsg = "retour au formulaire de connexion";
19      break;
20    default:
21      error = "vous devez vous connecter pour accéder à cette page.";
22      returnPath = "/connection";
23      returnMsg = "retour au formulaire de connexion";
24    }
25    res.render(__dirname + '/public/pages/error.ejs', { error, returnPath, returnMsg });
26  }
```

Notre page d'erreur est maintenant générée, les informations des cas d'erreurs sont envoyées au client, mais comment notre client peut-il afficher dynamiquement l'erreur correspondante ? Pour répondre à cette problématique nous allons manipuler l'EJS pour intégrer du code Javascript directement dans le DOM HTML.

Dans le code ci-dessous les balises ouvrantes < % et fermantes %> permettent d'intégrer le code Javascript dans l'HTML. La variable « error » permettra d'afficher le message d'erreur défini précédemment côté serveur. La variable « returnPath » définit la route vers laquelle va renvoyer le lien. Et enfin la variable « returnMsg » définit l'intitulé du lien.

```
1 <h1>error</h1>
2 <p><%= error %></p>
3 <a href="<%= returnPath %>"><%= returnMsg %></a>
```

2) Fonctionnalité d'upload et récupération de fichiers

Cette fonctionnalité a été source de nombreuses heures de réflexion, notamment dû au fait que ce concept n'avait pas été abordé en cours.

Pour cette fonctionnalité, nous allons utiliser le module `multer` qui est utilisé principalement pour les fonctionnalités d'upload de fichier en NodeJS. Nous installons donc le module grâce à un `npm i multer` dans le terminal.

Une fois ceci fait, il nous faut poser les bases de notre fonctionnalité. Pour cela il va nous falloir un input de type file dans notre document HTML.

Comme pour le formulaire d'inscription il va nous falloir préciser l'attribut `action` pour définir l'url (à savoir `/store-image`) à laquelle seront envoyées les informations. Cette fois ci nous devons en plus préciser l'attribut `enctype` en précisant « `multipart/form-data` » qui va permettre au formulaire de traiter des données sous forme de fichiers. Nous implantons ensuite un input de type file, avec l'attribut « `multiple` » qui permet d'envoyer plusieurs fichiers en même temps. Nous avons bien sur encore le `name` qui nous permettra d'accéder aux informations de l'input.

```
1 <form id="form-file" action="/store-image" method="POST" enctype="multipart/form-data">
2   <div>
3     <label>Sélectionnez votre photo:</label>
4     <input type="file" id="file-upload" name="image" required multiple>
5   </div>
6   <div id="div-file-btn">
7     <input type="submit" class="btn" id="sendFileBtn" value="Envoyer" style="width : 50%">
8   </div>
9 </form>
```

Une fois ceci fait, nous avons créé la base de notre fonctionnalité et nous pouvons nous attaquer au cœur du problème. Nous allons donc maintenant créer un dossier 'models' avec un fichier `image-model.js`.

Nous allons ensuite créer dans ce fichier une fonction `storeImage()` avec en paramètres `inputValues` et `callback`. Le paramètre `inputValues` va récupérer l'image dans un fichier `image-controller.js` que nous allons créer un peu plus tard.

Une fois fait, on initialise une constante qui aura pour valeur une requête SQL. Cette requête (à la ligne 5) nous permettra de vérifier si le fichier que l'on souhaite déposer à son nom déjà présent dans la base de données. Si c'est le cas, alors le serveur nous

renverra un message d'erreur (ligne 14). Si ce n'est pas le cas, alors on redéfinit la constante contenant la requête SQL pour qu'elle exécute une requête différente.

Cette fois ci, il s'agit d'une requête d'insertion concernant le nom du fichier dans la table image de la base de données.

```
1  const db = require('../dbConfig');
2
3  module.exports = {
4    storeImage: function (inputValues, callback) {
5      const sql = 'SELECT * FROM image WHERE image_name =?';
6
7      try {
8        db.con.query(sql, inputValues.image_name, function (err, data, fields) {
9          let msg = "";
10         console.log(err);
11         console.log(data);
12         if (err) throw err;
13         if (data.length > 1) {
14           msg = inputValues.image_name + " is already exist";
15         } else {
16           const sql = 'INSERT INTO image SET ?';
17           db.performQuery(sql, inputValues, function (err, data) {
18             if (err) throw err;
19           });
20           msg = inputValues.image_name + "is uploaded successfully";
21         }
22         return callback(msg)
23       })
24     } catch (error){
25       console.log(error);
26       return callback(error)
27     }
28   }
29 }
```

Nous avons donc vérifié si le nom de l'image envoyé par l'utilisateur est déjà présent dans la base de données, et si ce n'est pas le cas, nous avons inséré le nom du fichier image. Une fois ceci fait, il faut maintenant que nous procédions au stockage du fichier en lui même. Nous allons donc d'abord créer un dossier qui permettra de stocker les fichiers. Nous appellerons ce dossier uploadFiles. Une fois fait, nous créons un autre dossier que nous appelons middlewares avec à l'intérieur un fichier image-middleware.js. Ce fichier contiendra les fonctions permettant de stocker les fichiers dans le dossier uploadFiles.

```

1  const multer = require('multer');
2
3
4  module.exports.image = {
5    storage: function () {
6      const storage = multer.diskStorage({
7        destination: function (req, file, cb) {
8          cb(null, 'public/uploadFiles/')
9        },
10       filename: function (req, file, cb) {
11         cb(null, file.originalname)
12       }
13     })
14
15     console.log('middle 1');
16     return storage;
17   },
18   allowedImage: function (req, file, cb) {
19     // Accept images only
20     if (!file.originalname.match(/\.(jpg|JPG|jpeg|JPEG|png|PNG|gif|GIF)$/)) {
21
22       console.log('middle2');
23       req.fileValidationError = 'Seuls les fichiers images sont autorisés';
24       return cb(new Error('Seuls les fichiers images sont autorisés'), false);
25     }
26
27     cb(null, true);
28   }
29 }

```

À la première ligne, on remarque qu'on précise que le script de ce fichier nécessite l'utilisation du module `multer` que nous avons installé précédemment. On passe ensuite au script que nous allons exporter grâce à la ligne 4. Une fois ceci fait, nous commençons par définir le chemin du dossier dans lequel seront stockés les fichiers images. Nous déclarons donc la constante `storage` à la ligne 6, puis grâce au module `multer` nous lui donnons le chemin de notre dossier à la ligne 8 (à savoir `'public/uploadFiles/'`).

Après avoir défini le chemin du dossier, il faut maintenant préciser comment on nomme le fichier déposer dans le dossier. Pour notre projet, nous garderons le nom original du fichier pour que cela soit cohérent avec l'insertion de son nom dans la base de données. Nous précisons donc cette information supplémentaire dans la constante `storage` également grâce aux lignes 10 et 11.

Nous avons donc créé notre fonction permettant l'upload de fichiers, mais notre fonctionnalité est loin d'être complète car en l'état actuel des choses, l'utilisateur peut déposer des fichiers images, mais aussi des fichiers vidéo, texte ou autre. Or nous ne souhaitons recevoir que des fichiers images. Nous créons donc une seconde fonction que l'on nomme `allowedImage`. Dans cette fonction nous allons préciser une condition (à la ligne 20) qui vérifiera directement dans le nom du fichier si

l'extension de fichier correspond bien à un fichier image. À savoir jpg, jpeg, png, ou gif. Si aucun de ces termes ne se trouve dans le nom du fichier, alors nous créons un message d'erreur informant l'utilisateur que seuls les fichiers images sont autorisés.

Nous avons donc maintenant créé les conditions de stockage, à savoir qu'est-ce que l'on stocke et où on le stocke. Nous allons donc maintenant créer un dossier controllers avec à l'intérieur un fichier image-controller.js. Le script fera appel aux scripts rédigés dans les deux fichiers que nous avons créés précédemment (image-middleware et image-models).

```
1  const multer = require('multer');
2  const imageMiddleware = require('../middlewares/image-middleware');
3  const imageModel = require('../models/image-model');
4
5  module.exports = {
6    imageUploadForm: function (req, res) {
7      console.log('image-control 1');
8    },
9    storeImage: function (req, res) {
10     const upload = multer({
11       storage: imageMiddleware.image.storage(),
12       allowedImage: imageMiddleware.image.allowedImage
13     }).single('image');
14
15     upload(req, res, function (err) {
16       if (err instanceof multer.MulterError) {
17         res.send(err);
18       } else if (err) {
19         res.send(err);
20       } else {
21         // store image in database
22         const imageName = req.file.originalname;
23         const inputValues = {
24           image_name: imageName
25         }
26
27         // call model
28         imageModel.storeImage(inputValues, function (data) {
29           res.redirect('/vie');
30         })
31       }
32     })
33   }
34 }
```

Nous remarquons d'abord de la première ligne à la ligne 3 que nous précisons que notre fichier nécessite l'utilisation du module multer mais également des scripts présents dans les fichiers image-middleware et image-model.

Nous rédigeons ensuite le script qui sera exporté grâce à la ligne 5. Nous initialisons à la ligne 10 une constante qui contient la fonction permettant l'upload du fichier. Dans cette fonction nous appelons les deux fonctions présentes dans le script du

fichier image-middleware. Nous avons donc la fonction storage à la ligne 11 et la fonction allowedImage à la ligne 12. Nous remarquons à la ligne 13 que nous demandons à ce que la fonction traite les fichiers un par un et seulement les fichiers déposés grâce à l'input ayant pour name « image ».

Nous exécutons ensuite la fonction en précisant tout de même trois conditions. La première à la ligne 16 demande à nous retourner un message d'erreur dans le cas où une erreur est survenue via le module multer. La deuxième condition à la ligne 18 demande à nous retourner un autre message d'erreur dans le cas où une erreur est survenue mais n'ayant pas de rapport avec le module multer. Et enfin, la troisième condition sera exécutée si aucune erreur n'a été rencontrée. Dans ce cas, de la ligne 22 à 25 nous lançons l'exécution du stockage du nom du fichier dans la database en stockant dans une constante le nom original du fichier (ligne 22), puis ce nom sera stocké dans la constante qui sert de paramètre à la fonction storeImage appelée depuis le fichier image-model.

Une fois ceci fait, notre méthode storeImage est terminée et elle nous permet de lancer l'exécution des fonctions permettant de stocker les fichiers dans le dossier uploadFiles et de vérifier et stocker le nom des fichiers dans la base de données.

Il nous reste maintenant à établir les routes d'upload et de stockage. Nous créons donc un dossier roadFiles contenant un fichier image-route.js.

```
1  const express = require('express');
2  const router = express.Router();
3  const imageController= require('../controllers/image-controller');
4  router.get('/store-image',imageController.imageUploadForm);
5  router.post('/store-image',imageController.storeImage);
6  module.exports = router;
```

À la première ligne nous précisons que l'exécution de notre fichier nécessite le module express. Mais cela ne suffit pas, nous avons également besoin du middleware Router d'express qui va nous permettre d'établir les routes de notre fonctionnalité (ligne 2). Nous précisons enfin que nous utiliserons les scripts présents dans le fichier image-controller (ligne 3).

Nous avons ensuite une première méthode de routage qui est le get qui nous permettra d'accéder à l'url /store-image, et une seconde méthode qui est le post qui va nous permettre de récupérer les informations envoyées par la fonction storeImage. Enfin nous exportons le tout à la ligne 6.

Enfin il nous faut préciser dans notre fichier app.js les nouvelles routes concernant cette fonctionnalité.

```
1  const imageRouter = require('./roadsFiles/image-route');
2
3  app.use('/', imageRouter);
```

La première ligne précise que l'exécution du fichier nécessite l'utilisation des routes présentes dans le fichier image-route, puis à la ligne 3 nous définissons à quel moment ces routes sont utilisées.

Une fois ceci fait, notre fonctionnalité d'upload de fichiers images est terminée. Mais nous ne voulons pas nous arrêter là. En effet, tout l'intérêt de cette fonctionnalité est le fait que les admins puissent récupérer facilement les images envoyées par les utilisateurs. Nous devons donc intégrer une fonctionnalité de récupération de fichiers pour que notre système soit entièrement fonctionnel.

Pour cela, il va falloir utiliser le module fs (File Search) qui permet de naviguer dans les dossiers et manipuler les fichiers plus facilement.

```
1  const fs = require('fs');
2
3  app.get('/vie', (req, res) => {
4
5      let arrayFiles = [];
6      let uploadFiles = './public/uploadFiles/';
7
8      fs.readdir(uploadFiles, (err, files) => {
9          console.log(files)
10         if (err) {
11             throw err;
12         }
13         files.forEach(file => {
14             console.log(file);
15             uploadFiles = 'http://localhost:8080/uploadFiles/';
16             arrayFiles.push(uploadFiles + file );
17             console.log(arrayFiles);
18         });
19
20         res.render(__dirname + '/public/pages/vie.ejs', { alertMsg: "", arrayFiles });
21     });
22
23
24 })
```

En premier lieu, nous remarquons à la première ligne que nous déclarons la constante fs précisant que l'exécution de ce fichier nécessite l'utilisation du module fs (bien entendu après l'avoir installé grâce à la commande `npm i fs`).

Nous nous rendons ensuite à la ligne concernant la route où nous souhaitons récupérer les fichiers en question. Il s'agit de la page ayant pour url « /vie » (ligne 3). Nous initialisons ensuite un tableau vide ayant pour nom arrayFiles à la ligne 5, ce tableau nous permettra de stocker les chemins des fichiers que nous souhaitons récupérer. Puis nous initialisons une constante qui déterminera le chemin du dossier où se trouve les fichiers images. Enfin nous exécutons une méthode du module fs qui va nous permettre de parcourir chacun des fichiers présents dans le dossier. Pour cela il nous faut rentrer en paramètre de la méthode la constante dans laquelle nous avons stocké le chemin du dossier en question. À l'intérieur de cette méthode, nous créons une boucle forEach qui parcourra un à un chaque fichier présent dans le dossier. Dans

cette boucle, nous redéfinissons le chemin contenu dans la constante uploadFiles (ligne 15) afin que le chemin ne définisse pas le dossier local mais bien le dossier serveur. Enfin, grâce au tableau vide que nous avons initialisé un peu plus tôt, nous stockons le chemin complet dont le client aura besoin pour accéder à l'image (ligne 16), à savoir le chemin du dossier contenu dans la variable uploadFiles en y ajoutant le nom du fichier que nous récupérons grâce au module fs. Une fois ceci fait, nous pouvons fermer notre boucle forEach et nous nous retrouvons avec un tableau contenant le chemin complet de chaque fichier présent dans le dossier uploadFiles. Enfin, lorsque nous demandons à la route de générer le fichier correspondant à la page (vie.ejs), nous lui demandons également d'envoyer comme information le tableau complet (à la ligne 20).

Une fois que nous avons fait cela, nous avons tous les outils nécessaires pour que l'admin puisse récupérer ces fichiers, il ne nous reste qu'à lui donner accès aux informations contenues dans le tableau arrayFiles.

Pour cela nous allons manipuler le langage EJS pour intégrer en Javascript le tableau en question dans le DOM HTML de la page générée par l'url /vie.

```
1 <div id="parc-admin-section" class="">
2   <h3>Fichiers envoyés par les utilisateurs</h3>
3
4   <div id="admin-container">
5     <%
6       for (let i = 0; i < arrayFiles.length ; i++) { %>
7       <img src= <%= arrayFiles[i] %>
8       <%= %>
9     </div>
10
11 </div>
```

Nous créons donc une boucle for dans des balises `< %` et `%>` permettant d'intégrer le Javascript, ayant pour paramètre de nombres de tours maximum la longueur de notre tableau. Enfin, à chaque tour de boucle, nous créons une balise image ayant pour source le chemin complet stocké dans le tableau, chaque tour de boucle ne ciblant qu'un fichier à la fois.

Nos fichiers sont maintenant visibles sur la page, mais il nous reste un dernier détail à régler. En effet, ces fichiers sont visibles par tous les internautes alors que nous souhaitons qu'ils ne soient visibles que par les admins. Pour cela nous devons vérifier que l'internaute est bien un admin. Nous pouvons vérifier cela grâce aux cookies générés et attribués à chaque internaute au moment de sa connexion.

```

1  let adminSection = document.querySelector('#parc-admin-section');
2  const admintkn = getCookie("admintkn");
3  const adminContainer = document.querySelector('#admin-container');
4
5  function getCookie(cname) {
6      let name = cname + "=";
7      let decodedCookie = decodeURIComponent(document.cookie);
8      let ca = decodedCookie.split(';');
9      for (let i = 0; i < ca.length; i++) {
10         let c = ca[i];
11         while (c.charAt(0) == ' ') {
12             c = c.substring(1);
13         }
14         console.log(c);
15         if (c.indexOf(name) == 0) {
16             adminSection.classList.add('active');
17             return c.substring(name.length, c.length);
18         } else {
19             adminSection.classList.remove('active');
20         }
21     }
22 }
23
24 return "";
25 }

```

Lors de la connexion de l'internaute, nous lui avons attribué un token stocké dans un cookie. Ce cookie aura pour nom usertkn s'il s'agit d'un utilisateur qu'il se connecte, et si un admin se connecte le cookie aura pour nom admintkn. Nous devons donc vérifier quel est le cookie actuellement stocké pour savoir s'il s'agit d'un utilisateur ou d'un admin qui est connecté. Nous allons donc mettre en place un script Javascript dans le fichier HTML vie.

Pour cela nous créons une fonction (à la ligne 5) que l'on va nommer getCookie. La valeur cname mise entre parenthèse est une valeur par défaut et sera remplacée par la valeur admintkn (qui est le nom du cookie que l'on vérifie) grâce à la constante initialisée à la ligne 2.

La variable decodeCookie appelle une méthode qui va nous permettre de traduire les cookies présents dans le DOM. Ensuite la variable ca (ligne 8) va effectuer un split de la traduction donnée par la variable précédente. Le résultat retourné sera stocké dans un tableau. Ensuite la boucle for nous permet de transformer toutes les valeurs stockées dans le tableau en chaîne de caractères.

Une fois ceci fait nous demandons si le contenu présent dans la variable c est présente dans la variable name (à savoir 'admintkn ='). La condition sera forcément validée si c'est un admin qui est connecté, mais ne le sera pas si c'est un utilisateur lambda qui

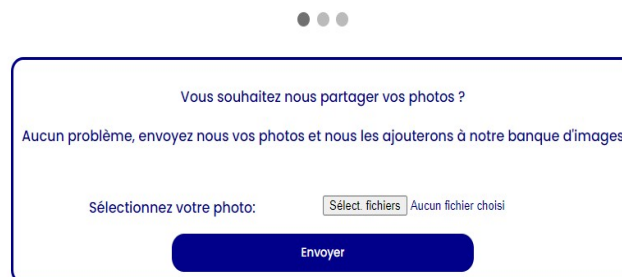
est connecté car on cherchera un caractère dans un cookie qui n'a pas été traduit vu que nous ne traduisons que les cookies nommés admintkn.

Si la condition est vraie, nous n'avons qu'à ajouter la classe « active » (ligne 17) à l'élément HTML que nous avons ciblé grâce `document.querySelector('#parc-admin-section')` à la ligne 1. Si la condition est fausse alors on supprime la classe active (ligne 20). Parallèlement, dans notre fichier CSS, nous précisons que si l'élément `#parc-admin-section` possède la classe active, alors nous le passons en display grid, sinon il sera en display none, ce qui l'affichera aux admins et pas aux utilisateurs.

Nous en avons terminé avec cette fonctionnalité qui a été pour moi un énorme défi que ce soit pour la fonctionnalité d'upload de fichiers en elle-même, ou même la fonction de vérification des cookies. J'ai conscience que ce code peut être simplifié, amélioré, et que de petites erreurs s'y cachent encore, et je compte bien l'améliorer pour tester ma progression.

Ci-dessous, vous trouverez le rendu pour un utilisateur lambda, et le rendu pour un admin.

Rendu utilisateur :



Rendu admin :



Fichiers envoyés par les utilisateurs



VII- Front End : Calculateur N-PY

1) Structure et fonctionnement

Concernant la partie Front End, je vais vous présenter une fonctionnalité que j'ai développée dans le cadre de mon stage chez N-PY. Il faut savoir que l'entreprise propose aux skieurs différentes cartes permettant d'avoir des réductions sur les forfaits de ski. Cependant, les clients sont souvent perdus entre les différentes cartes, les différents tarifs etc.

La fonctionnalité que je vais vous présenter est donc un calculateur qui permet de comparer les tarifs proposés par la carte No'Souci et les tarifs publics. Les tarifs sont calculés en fonction de la durée et date de séjour, de la station de ski, du nombre de skieurs et leurs âges, et de leur souhait de souscrire à une assurance ou pas.

En premier lieu, j'ai dû créer un fichier Javascript qui me servira de base de données. Les données stockées à l'intérieur concernent en majorité les règles tarifaires. Nous y retrouvons donc chaque station, chaque possibilité de formule (nombre d'enfants, nombre d'adultes, étudiants etc.), les durées de séjour, les dates, etc....

```
1  const nsRates = {  
2    horsvacances: {  
3      none: [0, 0, 0],  
4      cauterets: [27.3, 23.1, 27.3],  
5      gourette: [24.85, 21, 24.85],  
6      tourmalet: [32.9, 29.05, 32.9],  
7      lpsm: [23.8, 20.3, 23.8],  
8      luz: [25.2, 22.4, 23.8],  
9      peyragudes: [29.4, 25.9, 29.4],  
10     piau: [26.95, 23.45, 26.95],  
11   },  
12   vacances: {  
13     none: [0, 0, 0],  
14     cauterets: [33.15, 28.05, 27.3],  
15     gourette: [32.3, 27.6, 24.85],  
16     tourmalet: [39.95, 35.3, 32.9],  
17     lpsm: [31, 26.8, 23.8],  
18     luz: [30.6, 27.2, 23.8],  
19     peyragudes: [35.7, 31.45, 29.4],  
20     piau: [32.73, 28.48, 26.95],  
21   },  
22 };
```

Par exemple sur votre gauche vous pouvez voir une partie du fichier Javascript en question. Ces données représentent les différents tarifs possibles en vacances et hors vacances, et selon les stations de ski. Trois valeurs sont disponibles à chaque fois, celles-ci seront triées en fonction de la formule choisie par l'utilisateur.

Une fois notre fichier servant de base de données créé, nous pouvons nous attaquer à la structure du document HTML. Nous souhaitons que notre calculateur soit divisé en

deux parties distinctes, la première étant le conteneur des contrôleurs permettant de choisir la station, la durée, etc. La deuxième partie sera le résultat affiché sur la page. Cette dernière partie affichera à la fois les tarifs publics et les tarifs obtenus avec la carte No'Souci. Le calculateur sera donc divisé en deux colonnes, la dernière colonne étant elle-même divisée en deux colonnes également. Voici ce que le rendu final est censé nous donner.

Cauterets

2 Jours

Adultes

Enfants

Etudiants

Jours consécutifs

A partir du lun. 1 novembre

Pendant les week-ends & vacances scolaires

Tous mes skieurs possèdent une Carte No Souci, je veux juste calculer le prix des journées

78.00€
Prix public
78.00€ Journées de ski
0.00€ Assurance
Support carte Rechargeable offert.
Sans assurance (3€/jour/pers.) [Ajouter +](#)

Commander

Vous économisez 23.40€

54.60€
Tarif No Souci
54.60€ Journées de ski
0.00€ Achat initial des cartes
0 carte(s) offerte(s) pour 0 cartes achetées !
Assurance toujours incluse

Commander

Commençons par la première colonne, celle des contrôleurs. On peut remarquer que les deux premiers contrôleurs sont sous forme de menu déroulant. Pour cela, rien de compliqué, il suffit d'intégrer une balise `<select>`.

43

```

1 <div id="npv-select-controllers">
2   <div class="npv_calc_controllers_selects" id="npv-selecteur-station-nosouci">
3     <div>
4       <select id="npv-select-station" name="station">
5         <option value="" id="station0">Station</option>
6         <option value="cauterets">Cauterets</option>
7         <option value="gourette">Gourette</option>
8         <option value="tourmalet">Grand Tourmalet</option>
9         <option value="lpsm">La Pierre Saint Martin</option>
10        <option value="luz">Luz-Ardiden</option>
11        <option value="peyragudes">Peyragudes</option>
12        <option value="piaou">Piau Engaly</option>
13      </select>
14    </div>
15  </div>
16  <div class="npv_calc_controllers_selects" id="npv-selecteur-duration-nosouci">
17    <div>
18      <select id="npv-select-duration" name="duration">
19        <option value="" id="duration0">Durée</option>
20        <option value="1">1 Jour</option>
21        <option value="2">2 Jours</option>
22        <option value="3">3 Jours</option>
23        <option value="4">4 Jours</option>
24        <option value="5">5 Jours</option>
25        <option value="6">6 Jours</option>
26        <option value="7">7 Jours</option>
27      </select>
28    </div>
29  </div>
30 </div>

```

Prenons l'exemple du premier sélecteur. Nous ouvrons la balise à la ligne 4, et lui attribuons un name qui nous permettra de récupérer plus tard l'information sélectionnée par l'utilisateur. Nous devons également fermer la balise `<select>` car il ne s'agit pas d'une balise orpheline. Entre les deux balises, nous devons intégrer les options sélectionnables par l'utilisateur. Cela se fait grâce à la balise `<option>`. Ce que nous faisons par exemple à la ligne 5. Dans la balise ouvrante, nous devons également renseigner la valeur de l'option ce qui donne `<option value = 'cauterets'>` pour la ligne 6. Cette valeur, associée au name de la balise select va nous permettre de tester les options sélectionnées en Javascript. Juste après la balise ouvrante, nous devons renseigner à l'utilisateur à quoi correspond cette option, à savoir Cauterets. Enfin nous fermons la balise `<option>`. Nous répétons cette opération pour chaque option sélectionnable par l'utilisateur. Tout ceci va nous permettre de récupérer les informations dans un script Javascript présent dans un autre fichier.

```

1 const resortSelect = document.querySelector('#npv-select-station');
2
3 resortSelect.addEventListener('change', (e) => {
4   resort = e.target.value;
5   launchProcess();
6 });

```


Ci-dessus, la constante initialisée à la ligne 1 nous permet de sélectionner la balise `<select>` grâce à l'id 'npy-select-station' que nous lui avons attribué.

Nous créons ensuite une fonction grâce à la méthode `addEventListener` qui permet d'écouter si un événement se passe sur notre balise `<select>`, mais pour cela nous devons préciser quel type d'événement doit écouter la méthode. En l'occurrence, il s'agit d'un événement 'change' qui permet de vérifier s'il y a eu un changement de valeur effectué par l'utilisateur. Dans notre cas, la valeur par défaut est ' ', c'est à dire une valeur que l'on peut considérer comme vide. Si l'utilisateur clique sur l'option Cauterets, alors la valeur deviendra 'Cauterets' ce qui déclenchera l'événement change et lancera la fonction. À la ligne 4 nous pouvons lire « `resort = e.target.value` ». Le terme `resort` a été initialisé comme étant une variable vide plus tôt. Le terme « `e` » signifie événement tandis que le terme « `target` » signifie cible, « `e.target` » désigne donc la cible de l'événement qui n'est autre que notre balise « `select` ». Et enfin le terme `value` désigne la valeur actuellement sélectionnée. Dans notre cas, `resort` est donc égal à 'Cauterets'.

Sous ces sélecteurs, nous remarquons un autre système de contrôleurs. Nous pouvons sélectionner le nombre d'enfants, d'adultes ou d'étudiants en cliquant sur les boutons + ou - qui vont ajouter une valeur numérique dans un champ vide.

```
1 <div>
2   <div>
3     <div class="npy_label_comparateur">Adultes</div>
4     <div id="adult-moins">-</div>
5     <div id="adult-score">1</div>
6     <div id="adult-plus">+</div>
7   </div>
8   <div>
9     <div class="npy_label_comparateur">Enfants</div>
10    <div id="child-moins">-</div>
11    <div id="child-score">0</div>
12    <div id="child-plus">+</div>
13  </div>
14  <div>
15    <div class="npy_label_comparateur">Etudiants</div>
16    <div id="stud-moins">-</div>
17    <div id="stud-score">0</div>
18    <div id="stud-plus">+</div>
19  </div>
20 </div>
```

Ici rien de bien compliqué, il s'agit seulement de balises <div> possédant chacune un id qui lui est propre. Chaque div ayant en texte « + » ou « - » a été défini comme cliquable grâce à notre fichier Javascript.

```
1  const adultPlus = document.getElementById('adult-plus');
2  const adultScore = document.getElementById('adult-score');
3  const adultMoins = document.getElementById('adult-moins');
4
5  let nbAdultes = 1;
6  let nbChild = 0;
7  let nbStud = 0;
8
9
10 adultPlus.addEventListener('click', function () {
11     if (nbAdultes + nbChild + nbStud > 9) {
12         alert('10 personnes max');
13     } else {
14         nbAdultes++;
15         adultScore.innerHTML = nbAdultes;
16         launchProcess();
17     }
18 });
```

Ici nous allons prendre l'exemple du contrôleur permettant de sélectionner le nombre d'adultes. Dans notre fichier Javascript nous commençons par cibler les sélecteurs qui nous intéressent. À savoir le bouton +, le bouton -, et le champ vide permettant de renseigner le nombre d'adultes sélectionné.

Nous initialisons ensuite à la ligne 5 le nombre d'adultes par défaut comme étant à 1 grâce à la variable nbAdultes. Nous faisons de même pour les enfants et les étudiants grâce aux variables nbChild et nbStud, mais en les initialisant à 0.

Une fois ceci fait, nous créons notre fonction qui permettra de mettre à jour le nombre d'adultes sélectionnés à chaque clic sur le bouton + ou -. Dans notre exemple, nous nous occupons uniquement du bouton +. Pour cela, on réutilise la méthode add.eventListener sur le sélecteur désignant le bouton +, à savoir la constante adultPlus. En paramètre nous lui entrons la nature de l'événement à écouter qui est l'événement « click ». Nous souhaitons qu'il y ait au maximum 10 skieurs, nous créons donc une condition qui additionne le nombre d'adultes, d'enfants et d'étudiants et vérifie si avant le click, le total était supérieur à 9. Si c'est le cas, alors nous envoyons un message d'erreur à l'utilisateur l'informant qu'il ne peut sélectionner que 10 personnes maximum. Si le total est inférieur à 9, alors nous incrémentons la valeur de nbAdultes (ligne 14) et l'intégrons dans le champ vide grâce à la méthode innerHTML avec pour cible le sélecteur adultScore qui n'est autre que notre champ vide.

Le « launchProcess() » de la ligne 16 exécute la fonction calculant automatiquement les tarifs. Une fonction que nous ne développerons pas dans ce mémoire mais que nous pourrions aborder durant l’oral si vous le souhaitez.

Enfin, sous ces contrôleurs cliquables, nous pouvons voir un autre système de contrôleurs, qui sont les switches.

```
1 <div class="npy_switch_controllers">
2   <div>
3     <div>
4       <p>Jours consécutifs <br /><span id="changeDate">Modifier</span></p>
5     </div>
6   </div>
7   <div>
8     <label>
9       <input type="checkbox" id="consecutive-switch" checked />
10      <span></span>
11    </label>
12  </div>
13 </div>
14 <div>
15   <div class="div-holiday">
16     <p>Pendant les week-ends & vacances scolaires</p>
17   </div>
18   <div class="div-holiday">
19     <label>
20       <input type="checkbox" id="holiday-switch" />
21       <span></span>
22     </label>
23   </div>
24 </div>
25 <div>
26   <div>
27     <p>Tous mes skieurs possèdent une Carte No Souci, je veux juste calculer le prix des journées</p>
28   </div>
29   <div>
30     <label>
31       <input type="checkbox" id="card-switch" checked />
32       <span></span>
33     </label>
34   </div>
35 </div>
```

Ces switches ne sont rien d’autre que de simples input checkbox auxquels nous avons donné un style un peu plus moderne. L’attribut « checked » dans les input en question précise que la case est cochée par défaut. Chaque input possède un id qui lui est propre pour récupérer plus facilement ses informations dans notre fichier Javascript comme l’on a procédé précédemment.

```

1  let hasCard = true;
2  const hasCardSwitch = document.getElementById('card-switch');
3
4  hasCardSwitch.addEventListener('click', function () {
5      if (hasCard === true) {
6          hasCard = false;
7      } else {
8          hasCard = true;
9      }
10     launchProcess();
11 });


```

Ici nous allons prendre l'exemple du dernier <input> qui est celui définissant si l'utilisateur possède déjà la carte No'Souci. Cet input checkbox possédant l'attribut checked dans le DOM HTML, alors dans notre fichier Javascript nous initialisons la variable hasCard comme étant égale à true. Nous déclarons ensuite la constante hasCardSwitch qui sera notre sélecteur ciblant l'input.

On crée ensuite notre fonction, là encore en utilisant la méthode addEventListener avec en paramètre l'événement 'click'. Lorsqu'un clic est détecté sur cet input, nous vérifions d'abord si la valeur d'hasCard est toujours égale à true, si c'est le cas, alors on passe sa valeur à false, sinon on la passe à true.

Nous en avons terminé avec les contrôleurs. La seconde partie de ce calculateur me donne l'embarras du choix concernant ce que je pourrais vous présenter. J'ai fait le choix de vous présenter la fonctionnalité que je trouve la plus intéressante pour l'expérience de l'utilisateur, mais nous pourrions revenir sur d'autres parties de ce projet durant l'oral si vous le souhaitez.

Je souhaite donc attirer votre attention sur un petit champ informatif de couleur rose affiché dans les résultats des tarifs.



78.00€


Prix public

78.00€ Journées de ski
0.00€ Assurance


Support carte Rechargeable offert.

Sans assurance (3€/jour/pers.) [Ajouter +](#)

[Commander](#)



Vous économisez 23.40€




54.60€

Tarif No Souci

54.60€ Journées de ski
0.00€ Achat initial des cartes

0 carte(s) offerte(s) pour 0 cartes achetées !

Assurance toujours incluse 

[Commander](#)

Ce champ s’affiche soit au-dessus du visuel de la carte No’Souci, soit au dessus de la carte des tarifs publics, en fonction de quelle offre est la plus intéressante pour l’utilisateur. Dans le cas où l’offre No’Souci est plus avantageuse, alors l’information indique les économies que peut réaliser l’utilisateur. Voyons tout d’abord comment nous avons structuré cela dans notre document HTML.

Ci-dessous, vous trouverez la structure HTML de la colonne concernant les tarifs publics.

```

1 <div id="npy-public-price-nosouci">
2 <div>
3 <div>
4 
5 <p>
6 <span id="prix-public">39.00€</span>
7 <span id="prix-public-discount">xx€</span>
8 </p>
9 <p><b>Prix public</b></p>
10 <p id="npy-p-recap-assurance">
11 <span><b id="npy-recap-journees">39.00€</b> Journées de ski</span><br />
12 <span><b id="npy-recap-assurance">0.00€</b> Assurance</span>
13 </p>
14 <p id="npy-p-total-cartes">
15 <span>Support carte Rechargeable offert.</span>
16 </p>
17 <p id="npy-assurance">
18 <span id="npy-span-assurance">
19 Sans assurance (3€/jour/pers.)
20 </span>
21 <span id="npy-assurance-add">
22 Ajouter
23 </span>
24 <i class="fas fa-minus"></i><i class="fas fa-plus"></i>
25 </p>
26
27 <p class="npy_offerLink" id="npy-link-off1"><a href="#">Commander</a></p>
28 <div id="npy-bestchoice" class="active">Meilleur choix !</div>
29 </div>
30 </div>
31 </div>

```

Comme vous pouvez le voir à la ligne 28, le champ informatif qui nous intéresse est une simple balise <div>. Nous lui avons attribué un id qui lui est propre, mais surtout la classe « active ». Nous reviendrons dessus dans quelques instants.

Ci dessous vous trouverez la structure HTML concernant la colonne affichant les tarifs de la carte No’Souci.

```

1 <div id="npy-nosouci-price-nosouci">
2 <div>
3 <div>
4 
5 <p><b id="prix-no-souci">65.30€</b> <span id="npy-tarif-cartes-discounted"></span></p>
6 <p><b>Tarif No Souci</b></p>
7
8 <p id="npy-p-recap-cartes">
9 <span><b id="npy-recap-journees-ns">27.30€</b> Journées de ski</span><br />
10 <span><b id="npy-recap-cartes">38.00€</b> Achat initial des cartes</span>
11 </p>
12 <p id="npy-total-cartes-ns">
13 <span id="npy-cartes-offertes">0 carte(s) offerte(s) pour 1 carte(s) achetée(s).</span>
14 </p>
15 <p id="npy-assurance-ns">Assurance toujours incluse <i class="fas fa-check-circle"></i></p>
16
17 <p class="npy_offerLink" id="npy-link-off2"><a href="https://www.n-py.com/fr/tourisme-pyrenees/carte-no-souci">Commander</a></p>
18 <div id="npy-bestchoice-ns">
19 Vous économisez <span id="npy-eco-prix-bestchoice">10<sup>€</sup></span>
20 </div>
21 </div>
22 </div>
23 </div>

```

Là encore, vous pouvez remarquer de la ligne 18 à 20 que le champ en question est une simple div. Nous lui avons attribué à elle aussi un id qui lui est propre. Une balise `` est présente dans laquelle se trouve le tarif économisé. Ce span nous permettra grâce à une fonction en Javascript de mettre à jour la valeur grâce à une méthode `innerHTML`. Ce qui nous intéresse en particulier, et vous l'aurez remarqué, c'est que cette fois nous n'avons pas défini de classe 'active' à cette div. La raison est simple, il ne peut y avoir qu'une seule de ces div ayant cette classe à la fois car c'est cette classe qui va définir si elle est affichée ou pas. Et bien entendu, il ne peut y avoir qu'un seul meilleur choix.

Comment cette classe active définit l'affichage de cette div ? Tout simplement en lui accordant des propriétés en CSS que n'aura pas l'autre div si elle ne possède pas la classe active.

```

1  .npy_calc_price_results #npy-bestchoice,
2  □.npy_calc_price_results #npy-bestchoice-ns {
3      position: absolute;
4      top: 0;
5      left: 50%;
6      background: var(--pink);
7      padding: 5px;
8      font-size: 0.9em;
9      color: #fff;
10     border-radius: 5px;
11     transform: translateY(-100%) translateX(-50%);
12 }
13
14 .npy_calc_price_results #npy-bestchoice.active,
15 □.npy_calc_price_results #npy-bestchoice-ns.active {
16     display: block;
17     width: fit-content;
18     animation-duration: 0.5s;
19     animation-name: npy-showdiff;
20 }
21
22 □@keyframes npy-showdiff {
23     □ from {
24         opacity: 0;
25     }
26
27     □ to {
28         opacity: 100%;
29     }
30 }
31
32 .npy_calc_price_results #npy-bestchoice-ns,
33 □.npy_calc_price_results #npy-bestchoice {
34     display: none;
35 }

```

Nous pouvons voir à la ligne 1 et 2 que nous attribuons un style aux deux divs qui nous concernent, sans prendre en compte la classe 'active'. Nous commençons par placer nos divs à l'endroit où l'on veut qu'elles s'affichent grâce aux propriétés position : absolute (qui précise que l'élément se positionnera en fonction de l'élément le contenant, en l'occurrence il s'agit d'une div qui comprend tous les éléments de la colonne de résultats des tarifs), top, left et transform (qui sont précisés afin de déplacer notre élément dans son conteneur). Les autres propriétés nous permettent de donner le style général de nos divs. Background : (--var pink) nous permet de lui donner un fond de couleur rose, à partir d'une valeur RGBA que nous avons stockée

dans la variable `--pink`. Le `padding` nous permet d'élargir le fond. Le `font-size` et le `color` nous permet de définir la taille et la couleur de la police d'écriture. Enfin le `border-radius` nous permet d'arrondir les angles afin de lui donner un visuel moins agressif.

Nous pouvons voir aux lignes 32 et 33 que nous refaisons appel à ces sélecteurs pour leur attribuer la propriété `display` : `none` qui fait en sorte que les divs ne s'affichent pas par défaut. En mon sens, j'ai commis là une petite erreur d'organisation car j'aurais pu préciser cette propriété dès l'appel de ces sélecteurs à la ligne 1 et 2, ce qui aurait pu simplifier le code.

Enfin aux lignes 14 et 15 nous refaisons appel à nos divs mais cette fois ci nous précisons que les propriétés que nous allons entrer ne seront prises en compte que si nos divs possèdent la classe `'active'`. Nous précisons donc la propriété `display:block` qui activera l'affichage de ou des divs si elles ont la classe `'active'`. Nous pouvons également remarquer la propriété `width` qui désigne la largeur de l'élément ainsi que deux propriétés d'animation qui permettent un affichage en fondu ce qui rend un visuel moins agressif pour l'utilisateur.

Une fois ceci fait, nous avons nos deux divs, dont une avec une classe `active` par défaut. Nous avons également nos propriétés CSS qui précisent que seule la div avec la classe `'active'` doit s'afficher sur la page. Il nous manque seulement à trouver comment basculer la classe `'active'` d'une div à l'autre. Pour cela nous allons utiliser un script Javascript.

```
1  const bestChoice = document.getElementById('npy-bestchoice');
2  const bestChoiceNs = document.getElementById('npy-bestchoice-ns');
3
4  function getBestChoice() {
5      if (diffTotal > 0) {
6          bestChoice.classList.remove('active');
7          bestChoiceNs.classList.add('active');
8          bestChoiceNs.innerText = `Vous économisez ${diffTotal}€`;
9      } else if (diffTotal == 0) {
10         bestChoice.classList.remove('active');
11         bestChoiceNs.classList.remove('active');
12     } else {
13         bestChoiceNs.classList.remove('active');
14         bestChoice.classList.add('active');
15     }
16 }
```

Tout d'abord, nous initialisons les sélecteurs ciblant nos deux divs aux lignes 1 et 2. Nous créons ensuite notre fonction que l'on va nommer `getBestChoice`. Dans cette fonction, nous allons retrouver trois conditions différentes, chacune en fonction d'une variable `diffTotal` qui est le résultat d'une fonction que je pourrai vous présenter à l'oral si vous le souhaitez. La fonction en question calcule les économies réalisées par

l'utilisateur s'il choisit la formule No'Souci. Dans la première condition (ligne 5), nous vérifions si ces économies sont supérieures à zéro, si c'est le cas, nous faisons appel à la méthode `classList.remove` et `classList.add` afin de supprimer la classe active à la div correspondant aux tarifs publics et d'ajouter la classe active à l'autre div.

Dans la deuxième condition (ligne 9), nous vérifions si les économies sont égales à zéro, si c'est le cas, nous utilisons la méthode `classList.remove` pour supprimer la classe active aux deux divs. Enfin, à la ligne 12, si les deux précédentes conditions ne sont pas vérifiées alors nous ajoutons la classe active à la div correspondant aux tarifs No'Souci et on supprime la classe active à l'autre div.

Ainsi, seule une div à la fois peut posséder la classe 'active', (voire même aucune des deux divs), et cette classe active nous permet de gérer l'affichage ou la disparition de l'élément qui nous intéresse.

Cette fonctionnalité n'est pas forcément très compliquée à développer, mais elle est d'un intérêt certain concernant l'expérience de l'utilisateur et lui permet une meilleure compréhension de l'outil qu'il a sous les yeux.

2) Responsive

Je vais maintenant vous présenter une étape indispensable à la conception d'un site moderne et améliorant l'expérience de l'utilisateur lorsqu'il navigue sur le site. Il s'agit du responsive, un terme qui désigne la capacité d'un site à s'adapter aux différentes taille d'écran. Les aperçus que je vous ai présentés précédemment ont été capturés à partir d'une vision d'ordinateur. Nous allons donc voir maintenant comment adapter cet affichage pour un écran mobile ou tablette.

Ci-dessous, vous trouverez le résultat du rendu attendu.

Ici, le rendu concernant le format tablette :

Cauterets

Durée

Adultes

-

1

+

Enfants

-

0

+

Etudiants

-

0

+

Jours consécutifs

A partir du lun 1 novembre

☒

Pendant les week-ends & vacances scolaires

☐

Tous mes skieurs possèdent une Carte No Souci, je veux juste calculer le prix des journées

☒

Vous économisez 11,70€

39.00€
Prix public
39.00€ Journées de ski
0.00€ Assurance
 Support carte Rechargeable offert.

Sans assurance (3€/jour/pers.) **Ajouter +**

Commander

27.30€
Tarif No Souci
27.30€ Journées de ski
0.00€ Achat initial des cartes
 0 carte(s) offerte(s) pour 0 cartes achetées !

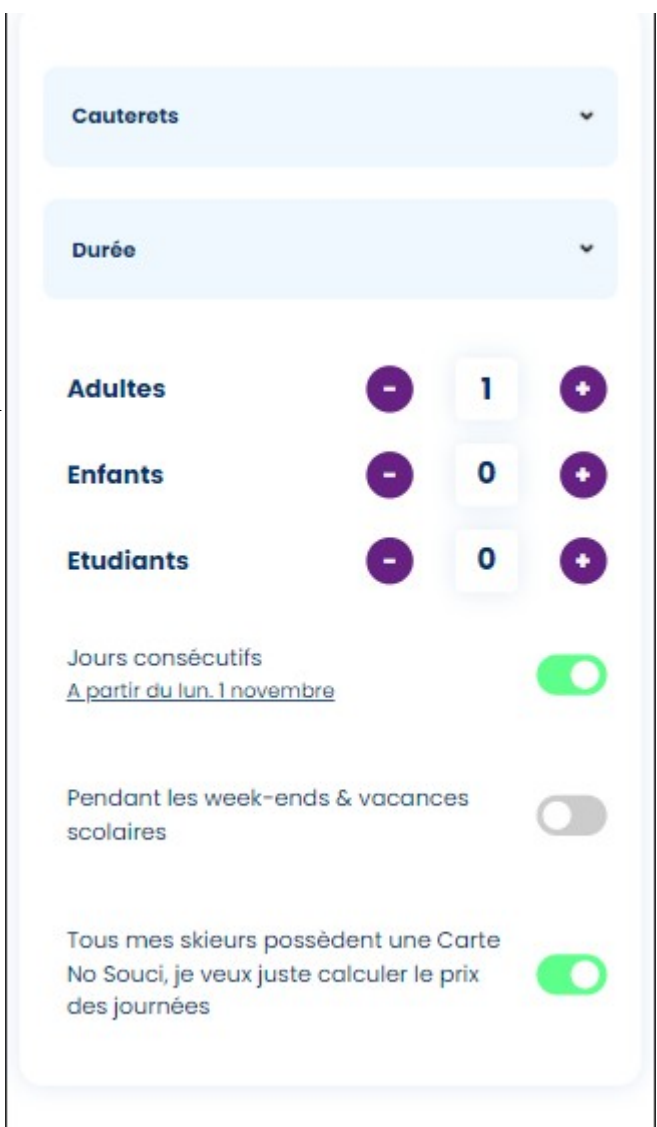
Assurance toujours incluse

Commander

Ci dessous, le rendu concernant le format mobile :

Nous avons donc dans un premier temps le bloc concernant les contrôleurs permettant à l'utilisateur de sélectionner les différents paramètres de la formule qu'il désire.

Pour accéder aux résultats voulus, il doit effectuer un scroll vertical comme c'est la coutume sur les applications et les sites en version mobile.



The image shows a mobile application interface for configuring a ski trip. It features a light blue background with white rounded rectangular buttons and sections. At the top, there are two dropdown menus: 'Cauterets' and 'Durée'. Below these are three rows for selecting the number of participants: 'Adultes' (1), 'Enfants' (0), and 'Etudiants' (0). Each row has a minus button, a central number box, and a plus button. Further down, there are three toggle switches: 'Jours consécutifs' (checked, green), 'Pendant les week-ends & vacances scolaires' (unchecked, grey), and 'Tous mes skieurs possèdent une Carte No Souci, je veux juste calculer le prix des journées' (checked, green). The text 'A partir du lun. 1 novembre' is visible under the first toggle.

Cauterets

Durée

Adultes

Enfants

Etudiants

Jours consécutifs
A partir du lun. 1 novembre

Pendant les week-ends & vacances scolaires

Tous mes skieurs possèdent une Carte No Souci, je veux juste calculer le prix des journées

Une fois effectué le scroll vertical, il peut donc visualiser les résultats affichés par le calculateur. Vous noterez que les images des cartes No'Souci et des tarifs publics ont été supprimées dans un souci de gain de place et de clarté pour l'utilisateur.

39.00€

Prix public

39.00€ Journées de ski
0.00€ Assurance

Support carte Rechargeable offert.

Sans assurance (3€/jour/pers.) **Ajouter +**

Commander

Vous économisez 11.70€

27.30€

Tarif No Souci

27.30€ Journées de ski
0.00€ Achat initial des cartes

0 carte(s) offerte(s) pour 0 cartes achetées !

Assurance toujours incluse 

Commander

Nous allons donc maintenant voir comment nous sommes arrivés à ce rendu.

La première étape est de faire en sorte que notre site puisse détecter la largeur de l'écran utilisé par l'utilisateur. Pour cela, une simple ligne dans notre DOM HTML suffit.

```
1 <meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

Une fois ceci fait, nous devons manipuler notre fichier CSS afin de rendre nos éléments adaptables ou d'en modifier le style pour certaines largeurs d'écran.

Dans un premier temps, nos blocs étant en majeure partie structurés en colonnes ou en lignes symétriques, nous avons favorisé la propriété « display : grid » dans notre fichier CSS, comme vous pouvez le voir dans la capture ci-dessous.

```
1 .npy_calc_price_results {  
2   display: grid;  
3   grid-template-columns: 50% 50%;  
4   width: 90%;  
5   margin: 7%;  
6   margin-top: 5%;  
7 }
```

Dans cet exemple, le sélecteur situé à la première ligne cible le bloc contenant les deux colonnes affichant les résultats du calculateur grâce à une classe que nous lui avons définie dans le DOM HTML. Nous lui renseignons ensuite la propriété « display : grid » qui demande aux éléments contenus par le bloc de se structurer sous forme de grille. Une fois ceci fait, nous précisons la propriété « grid-template-columns : 50% 50% » (ligne 3). Cette propriété permet de disposer les éléments du conteneur en plusieurs colonnes. Les valeurs entrées ensuite précisent les tailles que doivent faire lesdites colonnes. Notre conteneur ne possédant que deux colonnes, la première valeur représente la taille de la première, et la seconde valeur représente la taille de la deuxième colonne.

Cette façon de procéder permet dans un premier temps à nos colonnes de s'adapter aux proportions de son conteneur, qui s'adapte lui-même à la taille de l'écran grâce à sa largeur qui est définie en pourcentage à la ligne 4. Ce pourcentage fait référence à la largeur de l'écran sur lequel est visualisé notre site.

Cependant, ceci n'est pas suffisant, car bien que la largeur de notre conteneur et de nos colonnes s'adaptent simultanément à la largeur de l'écran, il peut arriver qu'à partir d'une certaine dimension, certains éléments contenus dans les colonnes se chevauchent, ce qui rendrait l'expérience utilisateur et le rendu de notre site

déplorable. Il faut donc repenser le placement de nos blocs et éléments à partir de certaines dimensions d'écran.

Grâce à la balise <meta> que nous avons insérée précédemment dans notre DOM HTML, notre site détecte la largeur d'écran utilisée par l'utilisateur. Dans notre fichier CSS, nous devons donc préciser à partir de quelle largeur d'écran nous allons changer le style de nos éléments. Pour cela, nous allons utiliser les requêtes CSS appelées media queries comme dans l'exemple ci-dessous.

```
1 @media screen and (max-width: 768px) {  
2   .npy_calc_price_results {  
3     display: block;  
4   }  
5 }
```

Comme vous pouvez le voir à la première ligne, nous faisons appel à la requête en question. Entre les parenthèses, nous précisons quelles largeurs d'écran notre requête concerne. En l'occurrence, il s'agit de tous les écrans ayant une largeur inférieure à 768 pixels. Ensuite nous ciblons l'élément qui nous intéresse, à savoir le conteneur de nos deux colonnes, et lui attribuons la propriété « display » avec la valeur « block ». Ainsi le « display : grid » que nous avons demandé précédemment sera supprimé et remplacé par notre nouveau style. Les colonnes en question étant de simples divs, alors nos deux colonnes se disposeront maintenant l'une sous l'autre, ce qui prendra moins de largeur, et permettra à l'utilisateur d'y accéder avec un scroll vertical.

Cependant nos images prennent toujours trop de place pour un écran mobile, nous allons donc faire en sorte qu'elles ne soient plus affichées.

```
1 @media screen and (max-width: 768px) {  
2   .npy_calc_price_results {  
3     display: block;  
4   }  
5  
6   .npy_calc_price_results img {  
7     display: none;  
8   }  
9 }
```

Comme vous le voyez à la ligne 6, nous avons ajouté un nouveau sélecteur à l'intérieur de notre requête media queries. Nous faisons toujours appel au sélecteur ciblant notre conteneur, mais cette fois nous y ajoutons le terme « img », ainsi les propriétés que nous allons entrer ne cibleront que les balises images présentes dans notre conteneur. Nous leur attribuons la propriété « display » avec la valeur « none »

qui fera en sorte que nos images ne s'affichent pas pour tous les écrans dont la largeur est inférieure à 768 pixels.

Nous répétons cette méthode pour l'ensemble des éléments de notre site dont nous voulons modifier le style selon les écrans, bien entendu en adaptant les propriétés correspondant au style voulu.

Ainsi, notre site est responsive, ce qui permettra aux utilisateurs d'y naviguer à partir de n'importe quel appareil, ce qui améliorera grandement l'expérience utilisateur.

VIII – Conclusion

1) Les difficultés

Comme dit dans la présentation en début de mémoire, je n'avais littéralement aucune compétence en termes de développement ou même de conception avant la formation. Mes seuls bagages étaient mes connaissances en informatique et l'univers du digital que m'avaient apporté ma curiosité et ma passion pour les jeux vidéos et le monde de l'audio visuel.

Cependant, je reconnais avoir eu quelques facilités concernant certains sujets abordés durant la formation, et avoir eu quelques difficultés concernant des concepts et des outils que nous avons approchés.

Ces difficultés concernent en particulier les logiciels de design, notamment adobe XD (que j'ai heureusement pu manipuler durant les cours) et adobe Photoshop dont j'ai pu avoir une première approche durant mon stage chez N-py. Ces logiciels ayant un très large panel de fonctionnalités, je me suis très vite rendu compte qu'il me faudrait des années de pratique afin de pouvoir les maîtriser complètement. Des années de pratiques nécessaires car ces logiciels sont d'une aide précieuse si l'on souhaite être autonome en tant que développeur Web. En effet, dans la plupart des entreprises ces outils ne sont utilisés que par les Web designers, mais ce n'est pas le cas pour toutes les entreprises, et c'est encore moins le cas pour un développeur freelance. Possédant aujourd'hui des connaissances de base concernant ces logiciels, je me dois d'approfondir le sujet et de découvrir plus en profondeur toutes les possibilités que nous offrent ces outils.

J'ai également eu quelques difficultés concernant certains concepts back end. En effet, les échanges entre serveur et client, ou entre serveur et base de données peuvent parfois paraître obscurs pour un néophyte. C'est pourquoi dans ce projet fil rouge, je me suis attaqué à des fonctionnalités complexes telles que le système d'upload et de récupération de fichiers, ce qui m'a permis de me mettre au défi, et d'améliorer les compétences que j'ai pu acquérir au cours de la formation. Des compétences qui sont là encore nécessaires à un développeur digne de ce nom, surtout pour un développeur qui se dit full-stack qui est censé avoir les compétences nécessaires pour créer un site complet, que ce soit pour le front end ou le back end.

Enfin la dernière difficulté que j'ai put rencontrer sont les lignes de commandes Git. En effet il s'avère qu'au début de la formation, je n'ai pas utilisé les lignes de commandes Git, mais le logiciel Github Desktop qui s'avère avoir une utilisation plus simple pour les novices. Cependant, bien que ce logiciel permette de travailler à plusieurs sur un même projet, il ne possède pas les fonctionnalités de débogage dont

sont équipées les lignes de commandes Git. Je pense donc malheureusement avoir perdu du temps en utilisant Github Desktop, et je pense avoir aujourd'hui quelques lacunes concernant les lignes de commandes Git. Des lacunes qui ont été en partie comblées par mon tuteur de stage qui m'a fait travailler dessus, mais je pense devoir encore les manipuler afin de pouvoir les maîtriser correctement. Surtout sachant que ce système est indispensable pour travailler en entreprise.

2) Ce que j'ai appris

Comme je vous l'ai dit, je n'avais aucune compétence en développement avant la formation. Je possédais bien sur quelques connaissances, par exemple je savais ce qu'était le HTML, le CSS ou encore le Javascript, mais j'étais loin de connaître la moindre syntaxe de ces langages.

Je peux affirmer sans l'ombre d'un doute que la quasi totalité des sujets abordés en cours ou même en stage sont des concepts que j'ai appris et dont je n'avais qu'une vague idée avant tout cela. Que ce soit concernant la conception, le développement, la gestion de base de données, les méthodes de travail, tout était nouveau pour moi et je suis fier de pouvoir dire aujourd'hui que je pense avoir de bonnes connaissances et compétences qui me permettent de pouvoir prétendre à un poste de développeur Web dans la plupart des entreprises.

Bien entendu j'ai conscience qu'il me reste encore énormément de choses à apprendre et que mes compétences actuelles ne demandent qu'à être approfondies. Mais les progrès que j'ai pu réaliser en l'espace de onze mois ne font aucun doute.

Je suis aujourd'hui capable de créer des diagrammes d'utilisation, de séquence, d'activités, des maquettes d'arborescence, de zoning et de mockup, de développer toute la partie front-end d'un site grâce aux langages HTML, CSS et Javascript (vanilla), de développer également la partie back-end grâce au NodeJS et ses modules complémentaires. Je suis également capable de partager mon projet avec une équipe grâce au logiciel Github Desktop et aux lignes de commandes Git. J'ai aussi les capacités pour créer et gérer une base de données en langage SQL, avec ses diagrammes MCD et son MLD. J'ai également des connaissances de base me permettant de rédiger un cahier des charges cohérent, avec les spécificités que peut désirer un client. Enfin, et la plus importante de toutes ces nouvelles compétences, je suis maintenant capable d'apprendre et progresser de façon autonome dans le milieu du digital grâce aux méthodes de travail enseignées en cours, et à la veille technologique que je pratique désormais en permanence.

3) Pour la suite

Cette formation n'est, je l'espère, que le début d'une longue carrière. En effet, je peux affirmer avoir trouvé ma voie grâce à cette formation.

Au niveau personnel, je compte dans un premier temps continuer ce projet fil rouge et le mener à son terme, bien qu'il ne soit pas déployé en ligne. Continuer à travailler dessus me permettra d'aborder de nouveaux concepts et d'auto évaluer ma progression après la formation. Je souhaite également apprendre de nouveaux langages. En particulier le langage PHP qui est un des langages les plus couramment utilisés dans la région et qui m'ouvrirait des portes certaines dans beaucoup d'entreprises. Je souhaite également être capable de maîtriser le framework ReactJS qui là aussi me semble être indispensable pour évoluer en tant que développeur Web. Un framework dont j'ai pu avoir un avant-goût durant la formation.

Cette volonté de vouloir progresser par soi-même est indispensable pour une carrière florissante dans le milieu du digital. Les technologies utilisées dans le développement Web étant en constante évolution, il est nécessaire de se tenir informé en permanence des dernières nouveautés et des dernières méthodes de travail.

Au niveau professionnel, je compte dans un premier temps trouver un emploi en tant que développeur front-end afin d'améliorer au maximum mes compétences dans ce domaine, des compétences que j'estime être mon point fort aujourd'hui. Je compte parallèlement améliorer mes compétences en back-end pour, par la suite, trouver un emploi dans ce domaine. Ainsi, dans quelques années je compte maîtriser à la fois le front-end et le back-end, ce qui me permettra de me lancer en tant que développeur Web en freelance. Ceci me permettra de me mettre au défi au cours de chaque projet abordé, de me pousser à me surpasser et d'en apprendre toujours plus afin d'être toujours compétent, quel que soit le client ou le projet approché.

IX – Annexe

1) Cibles du projet



Identité
Paul, 16 ans, Lycéen
Fils unique, célibataire
Habite avec ses parents

Personnalité
Passionné de jeux vidéos et science fiction.

Timide, curieux.

Ne sait pas quoi faire après le lycée.

Aime découvrir de nouveaux univers créatifs.

Activités
Jeux vidéos, Films / Séries
Lecture
Dessin

Buts personnels
Souhaite se diriger vers une carrière dans la création artistique.

Habitudes de consommation
Visite de parcs d'attraction uniquement pendant les vacances.

Stratégie
Attirer son attention via des partenariats avec des films / jeux vidéos et autres oeuvres.



Identité
Théa, 7 ans, CE1.
Un frère.
Vit avec sa mère et son frère.

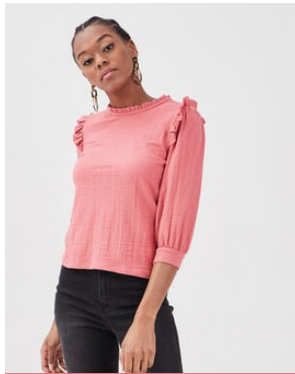
Personnalité
Hyper active.
Extravertie.
Joueuse.
Adore les princesses Disney.

Activités
Dessins animés Disney.
Adore se déguiser.
Joue beaucoup à la princesse et au chevalier.

Buts personnels
S'amuser.

Habitudes de consommation
N'a jamais fait de parc d'attraction.

Stratégie
Partenariats avec des dessins animés.
Princesses dans le parc.
Pubs télévisées.



Identité

33 ans, mère célibataire, aide soignante.
2 enfants.

Personnalité

Calme.
Passionnée de sport.
Aime les films d'amour.

Activités

Jogging, Yoga.
Lecture.
Géo Catching avec les enfants.

Buts personnels

Veut le meilleur pour ses enfants.
Cherche à trouver l'âme soeur.

Habitudes de consommation

Visitait les parcs d'attractions dans sa
jeunesse.
N'a pas les moyens pour y aller avec ses
enfants.

Stratégie

Partenariats avec les CE.
Promotions et places à gagner dans des
jeux concours.

Les trois captures ci-dessus représentent le résultat d'une analyse effectuée avant la création du site MovieLand. Cette analyse a pour but de cibler les différents types de personnes que notre projet doit toucher en priorité. Comme vous pouvez le voir nous nous basons sur leur personnalité, leurs activités, leurs buts personnels et leurs habitudes de consommation pour établir la stratégie marketing qui aura le plus de chance de les toucher et les convaincre de se rendre sur le site MovieLand et d'y réserver un séjour au parc.

Les résultats de cette analyse ont été présentés au client dans le cahier des charges lors de la présentation du projet.

2) Analyse de la concurrence

Enseigne	Disney Land	Niglo Land
Prix	Visible sur le site lors de la réservation	Visible sur l'accueil du site
Activités	Visible sur le site	Visible sur le site
Réservation de séjour	Possible sur le site	Possible sur le site
Charte graphique	Épurée	Enfantine
Présence sur les réseaux sociaux	Importante	Modérée
Popularité	Internationale	Locale
Publicité	Importante	Limitée
Rubrique vie du parc	Présente sur le site	Rubrique actu sur le site
Langue du site	Large panel de langues disponibles	Quatre langues disponibles
Capital	2.035.978.998,82 €	8.000€

La capture ci-dessus représente le résultat d'une analyse effectuée lors de la conception du cahier des charges. Cette analyse concerne les concurrents directs du parc MovieLand.

Dans la partie en rouge vous trouverez le concurrent ayant la plus grande réputation, le capital le plus important ainsi qu'une affluence dont rêve chaque parc d'attractions.

Dans la partie en vert vous trouverez le concurrent le plus proche de notre projet, que ce soit économiquement parlant, ou en terme de visibilité.

Le but de cette analyse est de cibler les différentes stratégies qui s'offrent à nous afin de faire de MovieLand un projet capable de concurrencer ces deux parcs. Cette analyse nous permet donc de savoir sur quel point nous devons nous inspirer d'eux, mais également sur quel point nous avons la possibilité de nous démarquer en innovant sur de nouveaux concepts.

3) Analyse SWOT



La capture ci-dessus représente le résultat de l'analyse SWOT (Strengths, Weaknesses, Opportunities, Threats) effectuée lors de l'élaboration du cahier des charges. Cette analyse a pour but de cibler les forces, les faiblesses, les opportunités et les menaces de notre projet. Ces résultats nous permettront de mettre en place diverses stratégies ayant pour but de nous démarquer de nos concurrents et nous assurer la pérennité de notre projet dans le marché des parcs d'attractions.

4) Devis

	Juil 2021	Aout 2021	Sept 2021	Oct 2021	Nov 2021	Dec 2021	Jan 2022	Fev 2022	Total
Heures travaillées	234h	234h	227h	234h	227h	234h	234h	9h	1633h
Salaire	4387,5	4387,5	4256,25	4387,5	4256,25	4387,5	4387,5	168,75	30618,75
Hébergement & gestion	50	50	50	50	50	50	50	50	400
Montant total	4437,5	4437,5	4306,25	4437,5	4306,25	4437,5	4437,5	218,75	31018,75

La capture ci-dessus représente le devis proposé au client lors de la présentation du projet. Le tarif représente le salaire de l'équipe en charge de la création et de la gestion du site (développeurs, web designers, chef de projet, photographe, etc.), l'hébergement du site et l'achat du nom de domaine.

J'ai énormément de doutes concernant la cohérence des tarifs en question, il s'agit là d'un domaine que je suis loin de maîtriser et que je compte étudier afin de pouvoir me lancer en tant que freelance dans quelques années.



5) Widgets N-PY

Réservez vos forfaits

Durée ▼ Adultes ▼ Enfants ▼ **ACHETER**

Réservation en ligne via n-py.com

Hébergement Forfaits Bainéo Activités

Choisir ▼ jj/mm/aaaa  jj/mm/aaaa  Personnes ▼

RECHERCHER

Réservation en ligne via n-py.com

RÉSERVEZ VOS FORFAITS

 Durée ▼ Adultes ▼ Enfants ▼ **RECHERCHER**

Réservation en ligne via n-py.com

Les captures ci-dessus représentent des widgets que j'ai pu réaliser durant mon stage chez N-PY. Ces widgets permettent un accès plus rapide aux fonctionnalités de réservations présentes sur les sites de Cauterets, Luz-Ardiden et La Pierre Saint-Martin. Pour Cauterets, et La Pierre Saint Martin, il existe plusieurs versions de ces widgets : des widgets fixes, flottants, et d'autres en langue espagnole. Ces widgets ont été réalisés en respectant la charte graphique des sites auxquels ils sont destinés.

X – Remerciements

Ces nouvelles compétences et connaissances que j'ai pu acquérir au cours ces onze derniers mois ne sont pas de mon seul ressort. En effet, de nombreuses personnes m'ont accompagné durant mon parcours et mon permis de progresser à chaque étape de mon apprentissage. C'est pourquoi je tiens à remercier plusieurs personnes pour leur aide précieuse et leur pédagogie exemplaire.

En premier lieu, je tiens à remercier l'ensemble des formateurs de l'institut ADRAR, en particulier Quentin Masse qui a été pour moi le principal acteur des compétences que je possède aujourd'hui. Il ne fait aucun doute que sans lui je n'en serais pas là où j'en suis aujourd'hui et je ne le remercierai jamais assez de toute l'aide qu'il a pu m'apporter au cours de mon apprentissage.

Je souhaite également remercier l'ensemble de l'équipe digitale d'N-PY, en particulier Ivain Bordeneuve (responsable digital) et Claire Maurel (trafic manager) qui ont su me mettre dans d'excellentes conditions dès mon arrivée dans l'entreprise, et ce malgré le stress immense que je ressentais. Grâce à eux, j'ai pu m'intégrer très facilement, appréhender très rapidement les méthodes de travail en vigueur dans leur équipe, et me sentir enfin crédible en tant que développeur web. Cette expérience chez N-PY et la pédagogie et l'accompagnement dont ont fait preuve Ivain et Claire a été pour moi un élément essentiel, je dirais même précieux dans l'acquisition des compétences et des connaissances que j'ai aujourd'hui.

Enfin pour terminer, je tiens à remercier Joel Carneiro, qui a été élève à mes côtés durant cette formation. Il a été un camarade exemplaire, et j'estime que notre duo a été bénéfique à chacun dans notre apprentissage. En effet, ses facilités dans le domaine du back-end ne faisant aucun doute, il s'est toujours montré disponible pour m'expliquer certains concepts que j'avais du mal à appréhender. J'espère lui avoir apporté la même aide que j'ai pu recevoir de lui.