**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Mémoire**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Raphaël Sigaud**

**Introduction :**

***Scherzando*** is a music sharing website and an online audio distribution platform that involves creativity and building communities around a musical proposal.

The creation of this project will involve all the differents elements of a web developpement classical project, which means writing a specification (involving an IT company, a definition of the project, different diagrams, comparison with other projects, the necessary services and finally a quotation), building mock-up (tree diagram, zoning and wireframe), the conception of the website (UML, MCD, MLD, etc.), working on the front-end which means using HTML, CSS and maybe some dynamic language like Javascript. And finally : working on the back-end by using PHP and SQL to build a database and be able to work with a server to provide a totally functional website.

**CAHIER DES CHARGES :**

**Ce document a pour but d’établir le cahier des charges du projet *Scherzando*. Il s’agit d’un projet de plateforme d’hébergement, de diffusion et de partage musical à visée communautaire. Ce site devra être disponible sur pc, tablette et smartphone afin de gagner une plus large audience.**

**Le présent cahier des charges décrira les objectifs du produit, ses forces et faiblesses, établira une comparaison avec d’autres sites équivalents. Enfin, il abordera les fonctionnalités et les technologies qui seront utilisées au cours de la création dudit projet et présentera un diagramme prévisionnel - de Gantt - pour spécifier les différentes périodes de création par catégories (maquettage, front-end, etc.) sur un temps global donné.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**L’ENTREPRISE :**

**Orca.DX est une entreprise spécialisée en conception et développement web basée en Aveyron. Ladite entreprise se chargera de créer le site *Scherzando* from scratch de sa conception au maquettage dans un premier temps, puis dans un second temps au développement des parties front et back-end afin que le site soit pleinement fonctionnel.**

**Orca.DX**

**Principal intervenant :**

**Raphaël Sigaud**

Maquette

Conception

Développement front-end & back-end

**Partie fonctionnelle :**

**Strengths Weakness Opportunities Threats**

|  | | |  | | | | |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Positif** | |  | **Négatif** | |  |
| **Forces** | | **Faiblesses** | |
|
|  | **Origine interne** |  | **Gratuité** | | **Indépendant** | |
| **Simplicité d'utilisation** | | **Petite plateforme** | |
|  | | **Gratuité** | |
|  | **Opportunités** | | **Menaces** | |
|
| **Origine externe** | **Modèle économique (gratuit)** | | **Concurrence** | |
| **Diffusion via les Réseaux Sociaux** | | **Piratage** | |
| **Format court** | | **Coût de l'hébergement** | |
|  |  | | | | | |

**Résumé :**

Nous avons constaté au niveau des atouts bien installés : la gratuité, qui offre aussi une opportunité d'attrait et de diffusion, mais peut aussi apparaître comme une faiblesse par l'aspect indépendant qui se répercute sur la dimension économique du site, notamment du fait du coût de l'hébergement. On note par ailleurs, sur la question de la concurrence, deux sites majeurs assez semblables : soundcloud et mixcloud.

**Tableau comparatif des existants :**

|  | **SoundCloud** | **MixCloud** | **Scherzando** |
| --- | --- | --- | --- |
| **page d’accueil** | subscribe/  trending songs/  infos/  offers/  logs/barre de recherche | subscribe/  creators/  infos/  offers/logs/barre de recherche | subscribe/  trending songs/trending creator/logs/barre de recherche |
| **connexion** | mail/mdp/(confirm) | mail/mdp/(confirm) | mail/mdp/(confirm) |
| **page morceau** | barre de recherche/artiste/titre/  photo/morceau (visualiser)/  commentaires/#/  description | barre de recherche/titre/  artiste/  photo/morceau (visualiser)/#/  description/  commentaires | barre de recherche/artiste/  titre/  photo/morceau (visualiser)/  commentaires/#/  description |
| **page artiste** | barre de recherche /bannière/pp/nom/  localisation/  mention pro ou non/  station/follow/  share/followers/  following/nbr tracks/All/Pop Tracks/Tracks/  Albums/Playlists/  Reposts/  les morceaux | barre de recherche /bannière/pp  (étoile pro)/ nom/follow/share/.../  followers/  stream/shows/fav/  history/ /les morceaux | barre de recherche /bannière/pp/nom/  morceaux/albums/  playlists/  système de rt/like à revoir ?  ⇒ épurer |
| **page perso** | barre de recherche /équivalent page artiste avec possibilité de modifier/ulpoad | barre de recherche /équivalent page artiste avec possibilité de modifier/ulpoad | barre de recherche /équivalent page artiste avec possibilité de modifier/ulpoad |
| **couleurs** | orange/blanc | bleu/bleu gris | ??? |
| **version mobile responsive** | oui | oui | oui |
| **version payante** | oui | oui | non |
| **Diffusion en direct/**  **Streaming** | non | non | non |

**Conclusion :**

Après comparaison, nous pouvons définir le projet Scherzando comme une version lite de ce qui se fait dans la concurrence, tout en soulignant l’aspect totalement gratuit du projet et potentiellement une simplification de l’interface par comparaison.

**- Enoncé du besoin :**

* Site d’hébergement et de publication de morceaux de musique à des fins de partage et de promotion pour les artistes et leurs publics.
* Une BDD pour répertorier les informations de profils, notamment les informations relatives aux morceaux.
* Personnalisation du profil importante.
* L’aspect “partage” est une partie importante du projet.

**- Les fonctions du produit :**

* **Upload/Affichage/Modification/Suppression (CRUD prioritaire)**
* Inscription/Connexion
* Gestion de profil
* Fonctions liées à l’écoute => création d’un lecteur via JS
* Commenter
* Like/Repost/Abonnement

**- Contraintes :**

* Gestion de serveur web/hébergement
* conception d’un lecteur et d’une UI épurée
* fonctionnalités PHP et JS

**Gestion du projet:**

* **Livrables**
  + **Conception :**
    - Maquettage du site
    - Diagrammes (Use case, activité, séquence, classes, MCD, MLD)
    - Design / Graphisme
  + **Front end :** 
    - html css js
  + **Back end :**
    - PHP, SQL
    - BDD PHPMyAdmin

**Diagramme de Gantt - prévisionnel :**

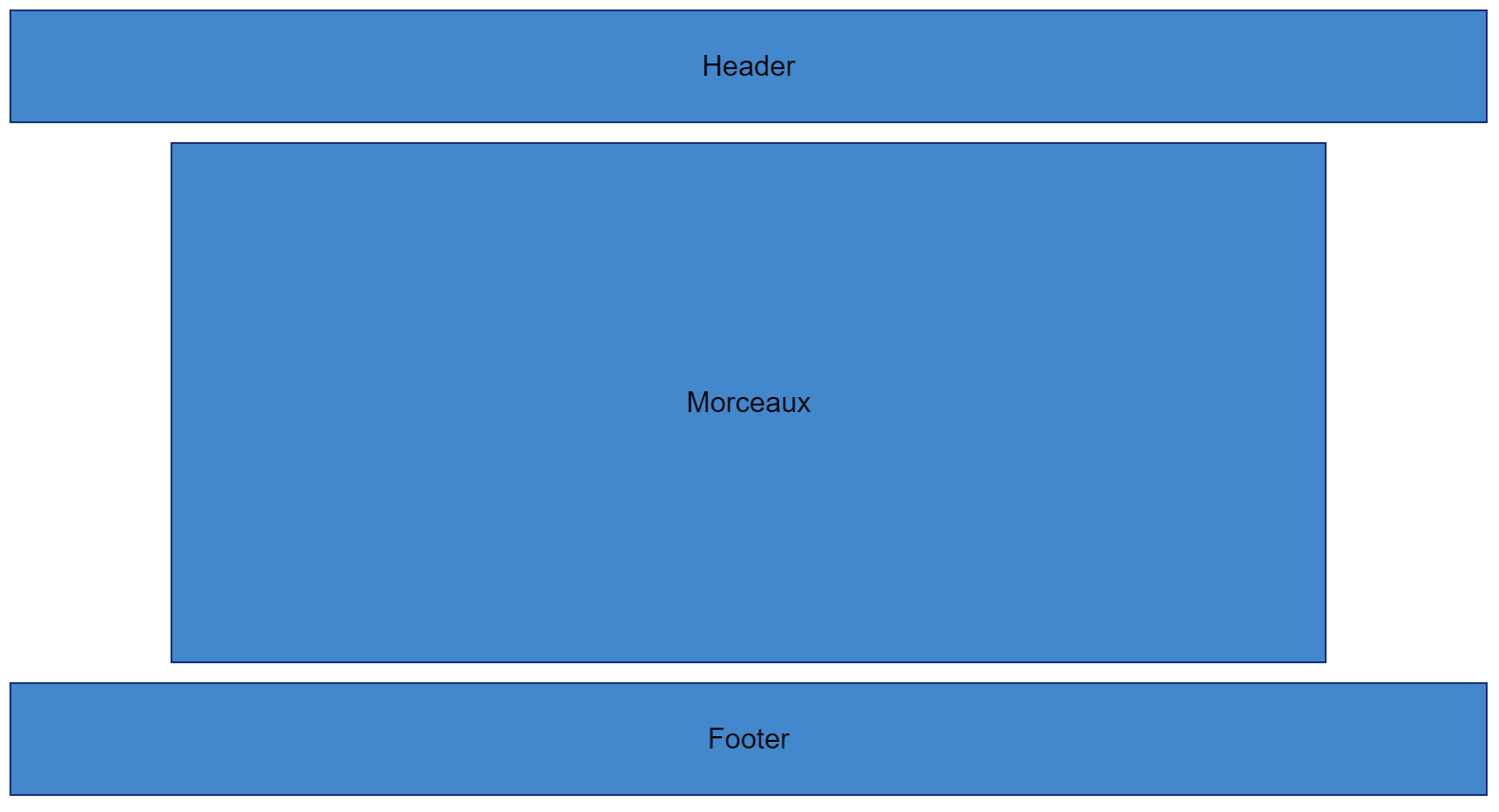
| **Tâches/**  **Livrables** | **Sept** | **Oct** | **Nov** | **Déc** | **Janv** | **Févr** | **Mars** | **Avr** | **Mai** | **Juin** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CDC/Maquettage/**  **Design/**  **Graphisme** | 13  sept |  |  | 15  déc |  |  |  |  |  |  |
| **Front-end**  **(html-css- javascript)** |  |  |  | 15 déc |  |  | 15 mars |  |  |  |
| **Back-end**  **(hébergement**  **-bdd)** |  |  |  |  |  |  | 15 mars |  |  | 14 **juin** |

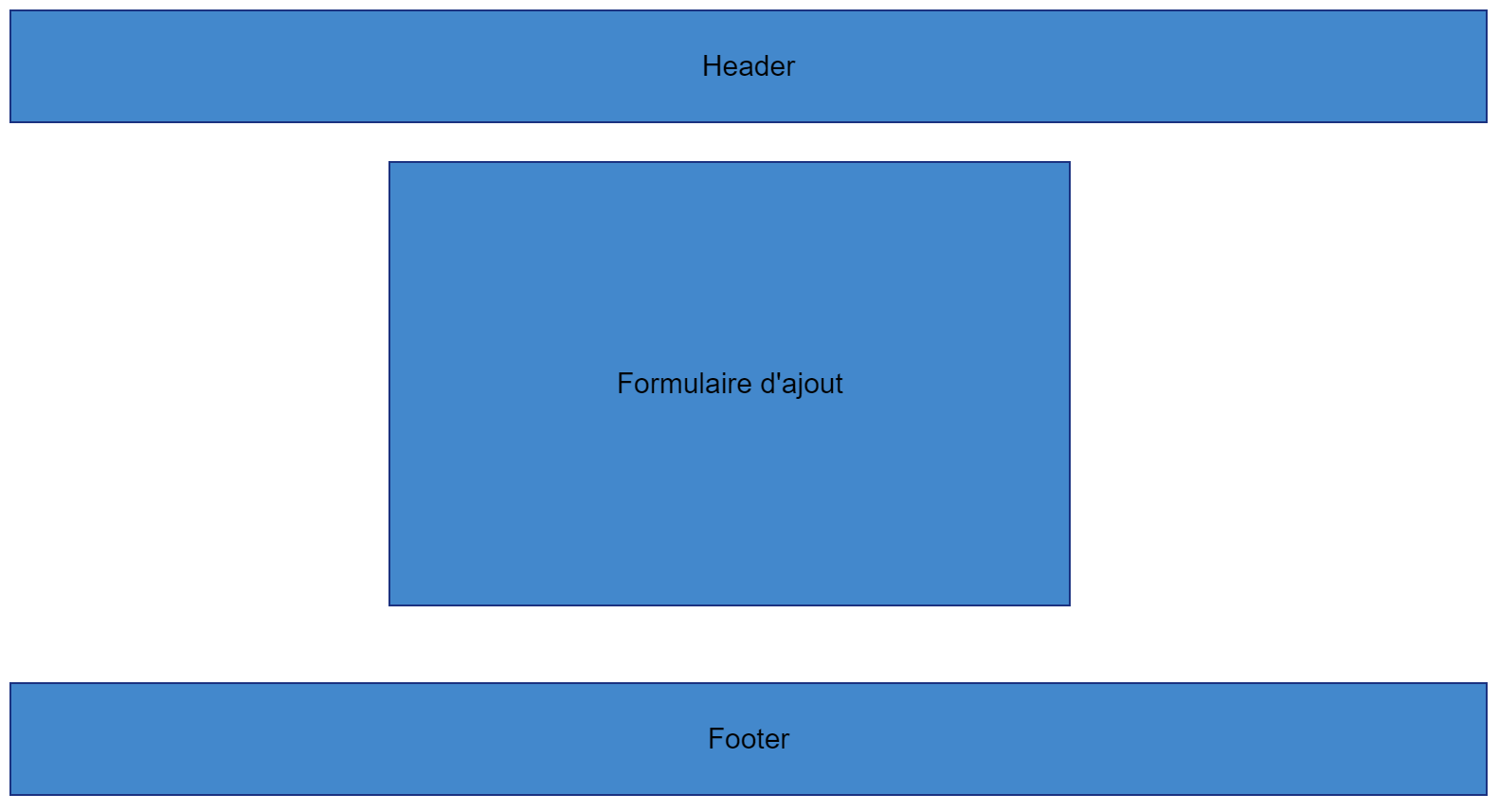
**Devis :**

| **Référence** | **Désignation** | **Tps** | **Prix HT**  **(15€/h)** | **TVA**  **20%** | **Total HT**  **(15€/h)** |
| --- | --- | --- | --- | --- | --- |
| Site-01 | Cahier des charges | 12 heures | 180 € | 36€ | **180€** |
| Site-02 | Maquette du site   * Pencil Project * Logo /Charte graphique * Templates | *≈ 3 mois*  35 h  35 h  35 h | 525 €  525 €  525 € | 20% (315 €)  105 €  105 €  105 € | **1575 €** |
| Site-03 | Développement Front-End   * HTML * CSS * JS | *≈ 3 mois*  105 h  105 h  105 h | 1575 €  1575 €  1575 € | 20% (1396 €)  315 €  315 €  315 € | **4725 €** |
| Site-04 | Développement Back-End   * Hébergement du site * Base de données | *≈ 3 mois*  140 h  140 h | 2100 €  2100 € | 20% (840 €)  420 €  420 € | **4200 €** |
| Site-05 | Mise en ligne et optimisation   * SEO * Google Analytics | 60 h  20 h | 900 €  300 € | 20% (240 €)  180 €  60 € | **1200 €** |
| Site-06 | Maintenance   * Suite Back-End (gestion de l’hébergement) * Mise à jour du site | *≈ 105 h* | *≈ 1575 €* | 20% *≈* 315 € | ***≈ 1575 €*** |
| **Total** |  | ***≈ 897 h***  *(réparties sur 3 mois)* |  | **3 142 €** | ***13 455 €*** |

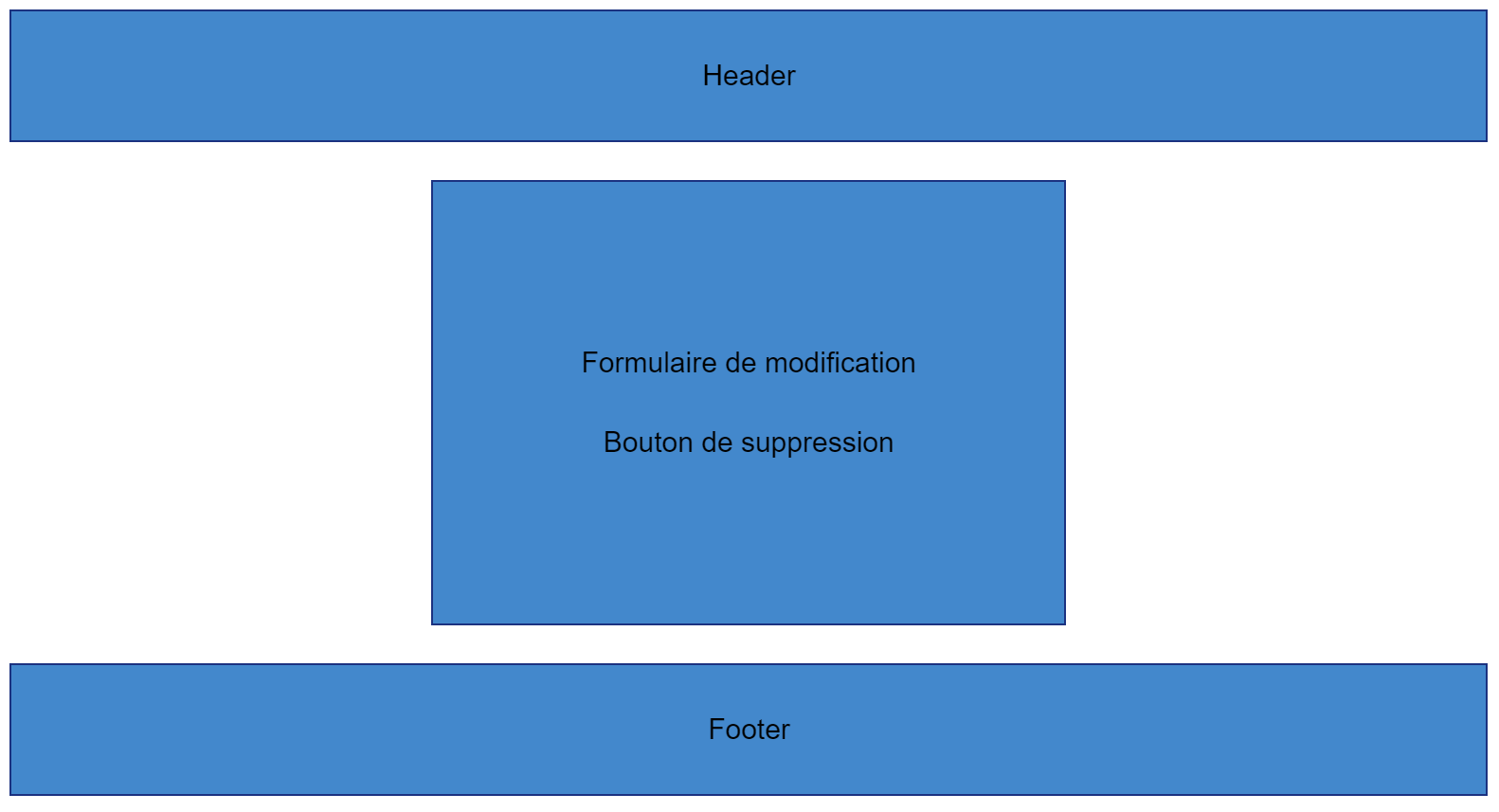
**Maquettage (Zoning-Arborescence) :**

Dans un premier temps, le zoning et l’arborescence sont réalisés dans l’optique d’établir un premier jet de ce qui deviendra l’apparence du site. Chacun des trois zoning est constitué des trois éléments classiques d’une page web, à savoir un Header, un Footer et un corps de page. Ce dernier sera le seul élément à changer entre les différentes pages.

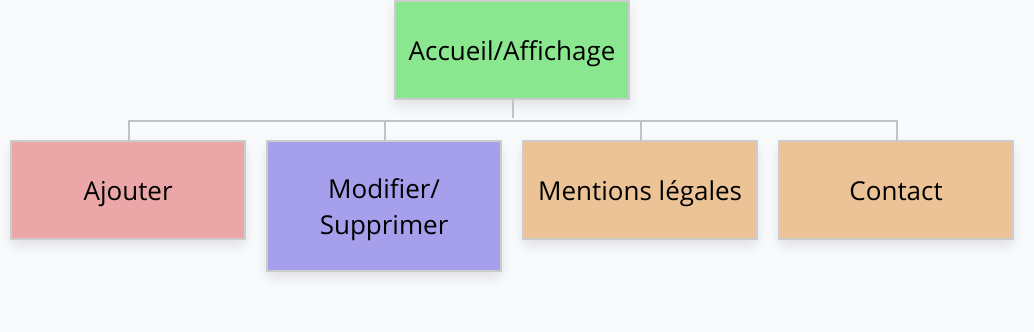
**Page d’accueil et d’affichage des morceaux (Read) :**

**Page d’ajout des morceaux (Create) :**

**Page de modification et de suppression des morceaux (Update, Delete) :**



**Arborescence générale du site :**



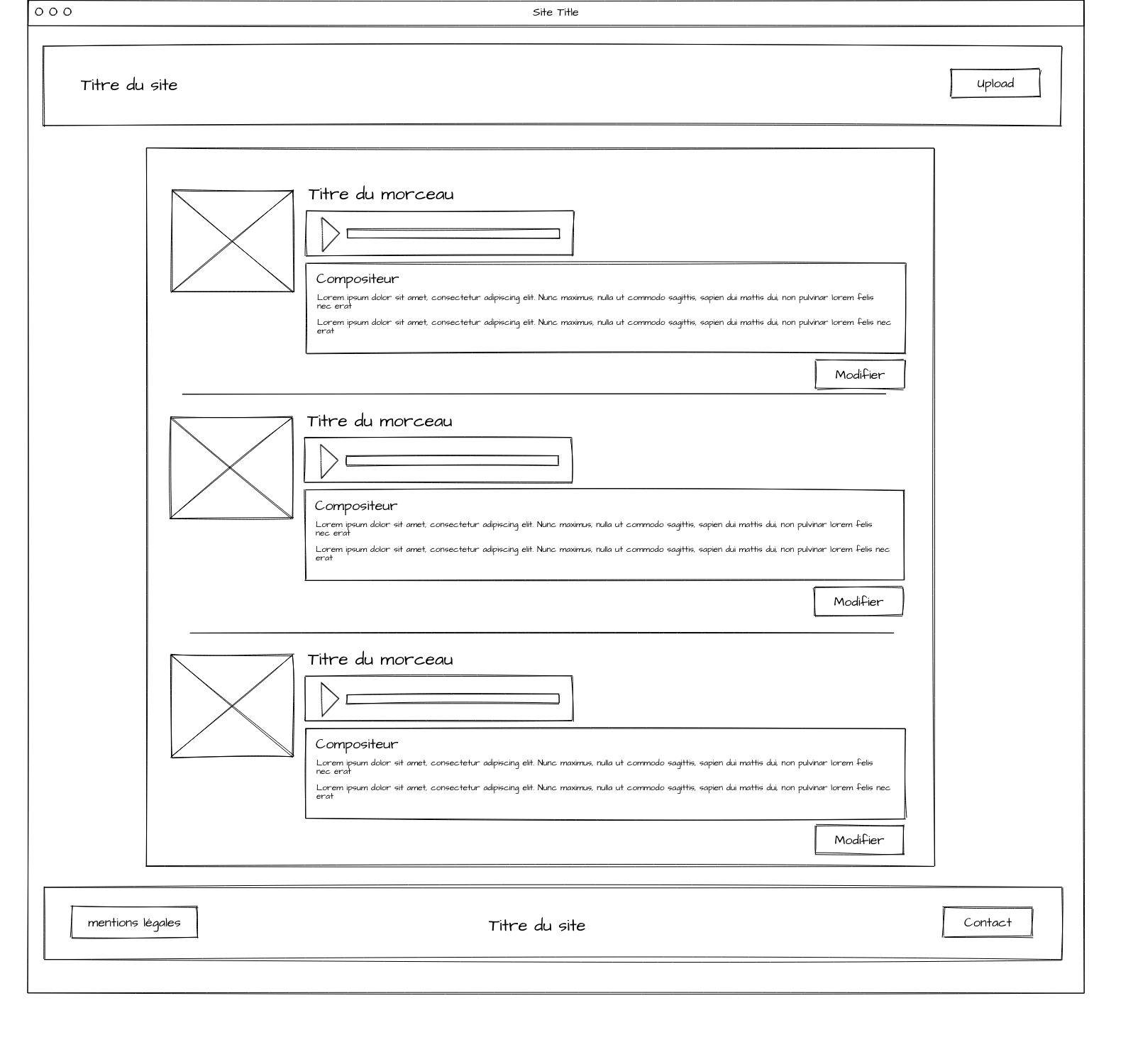
Ici, l’arborescence est gardée au plus simple et est censée représenter les différents boutons présents sur toutes les pages. L’accueil se retrouvera sur les header et footer tandis que les mentions légales et contact seront sur le footer seul. L’ajout sera sur le header et le couple modification/suppression seront des fonctionnalités propres au contenu des pages.

**Présentation des Header et Footer :**

Nous allons donc procéder à une rapide présentation des éléments fixes que sont les Header et Footer.

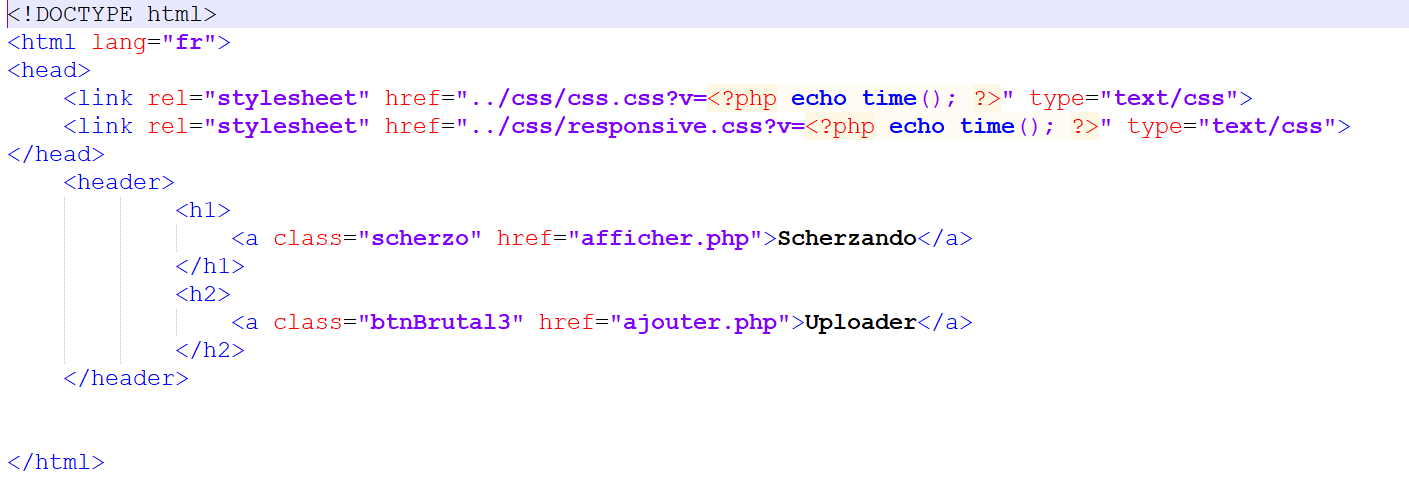
**Header :**

**Wireframe du Header :**

****

Le wireframe présente donc un bandeau avec deux liens/boutons, le premier reprenant le logo du site permet de retourner à l’accueil, le second permet d’accéder à la page d’upload.

**Code HTML du Header :**

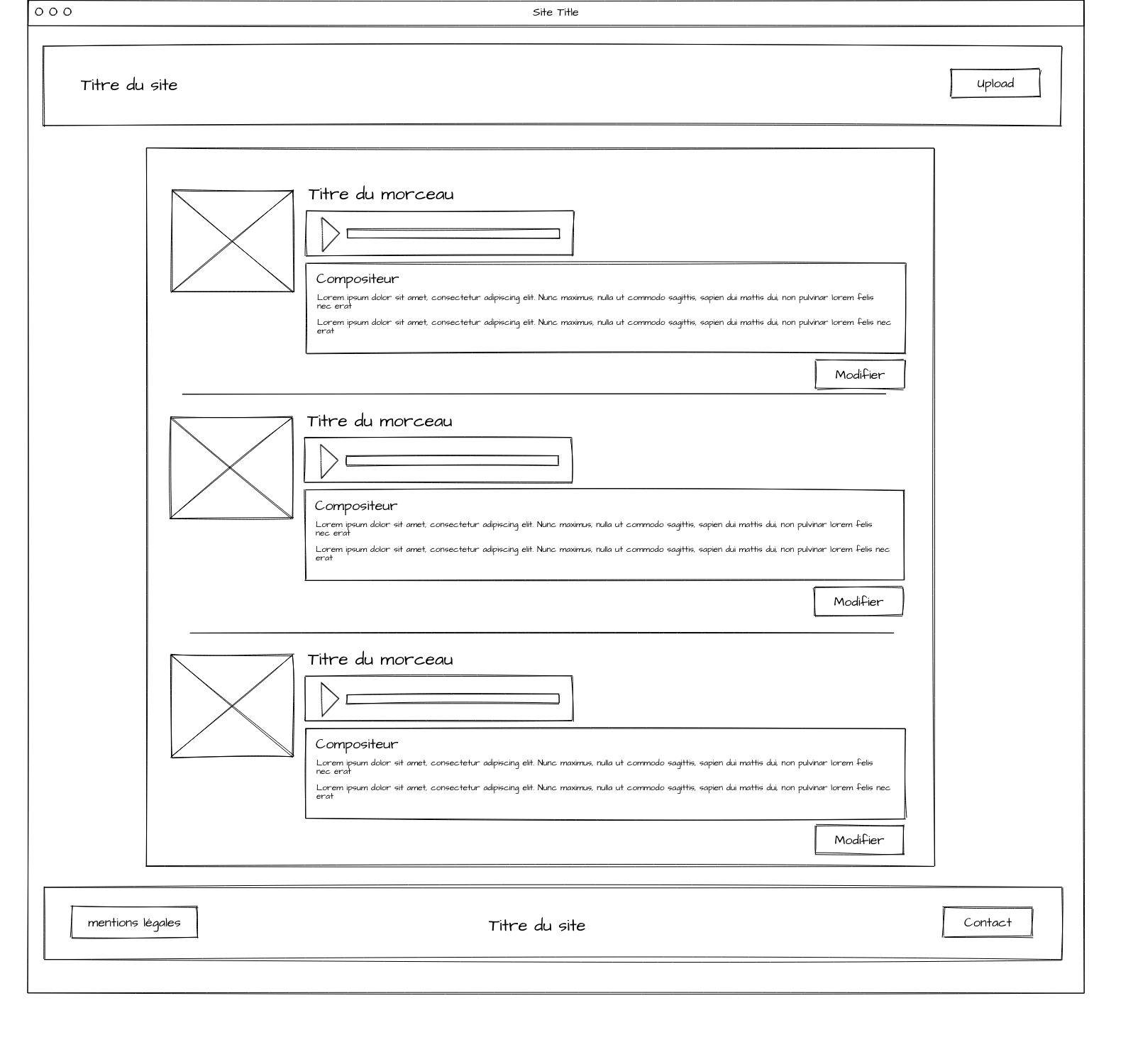
****

**Capture d’écran du Header final :**

****

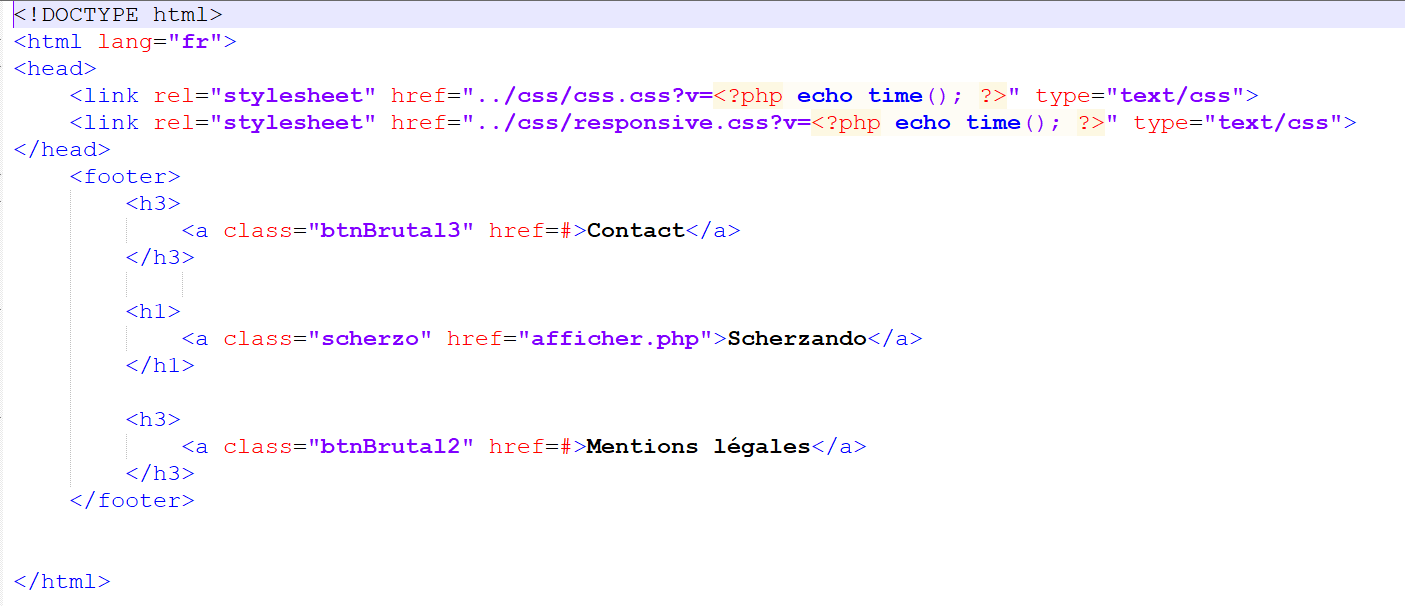
**Footer :**

**Wireframe du Footer :**

****

Ici le footer présente trois liens/boutons, le logo du site menant à l’accueil est au centre, les mentions légales et le bouton menant au contact sont de chaque côté.

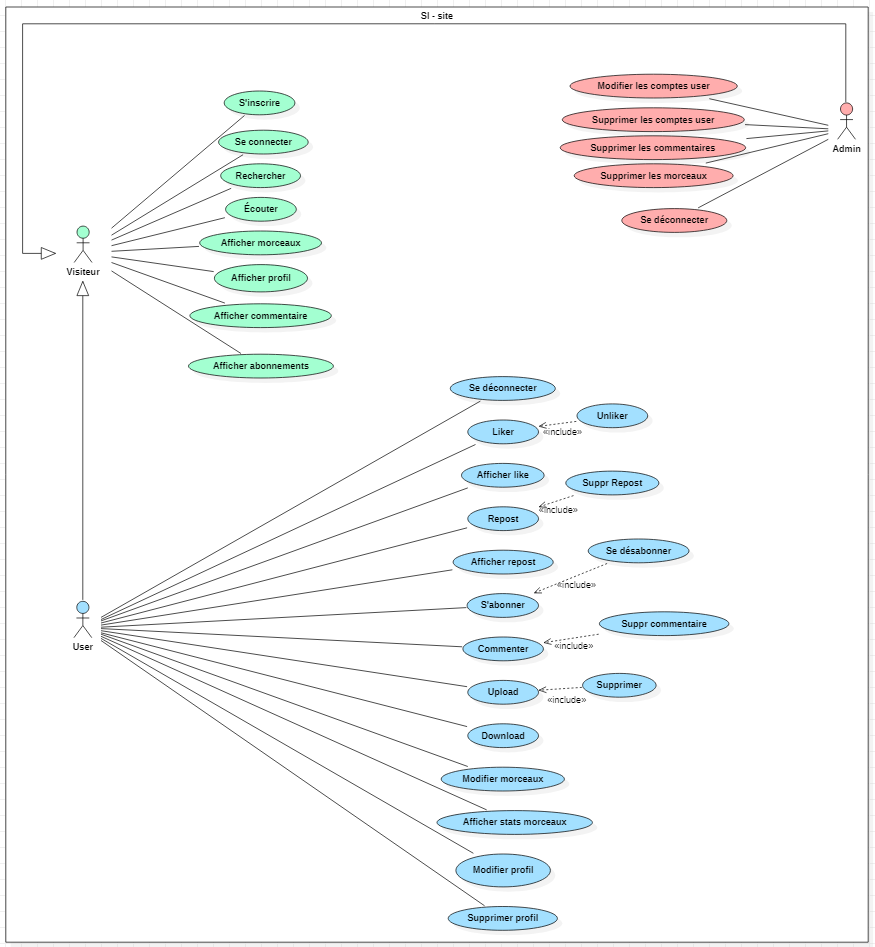
**Code HTML du Footer :**

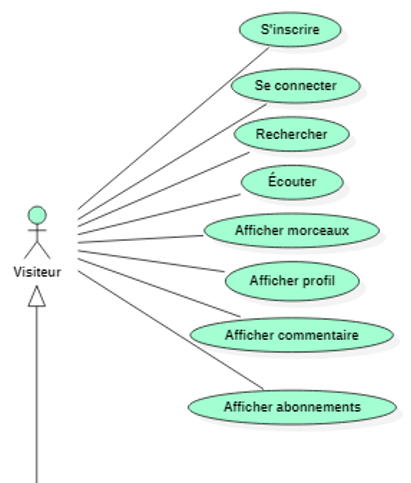
****

**Capture d’écran du Footer :**

****

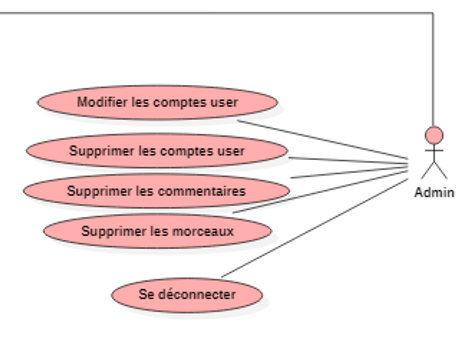
**Use case :**



**Le visiteur :**

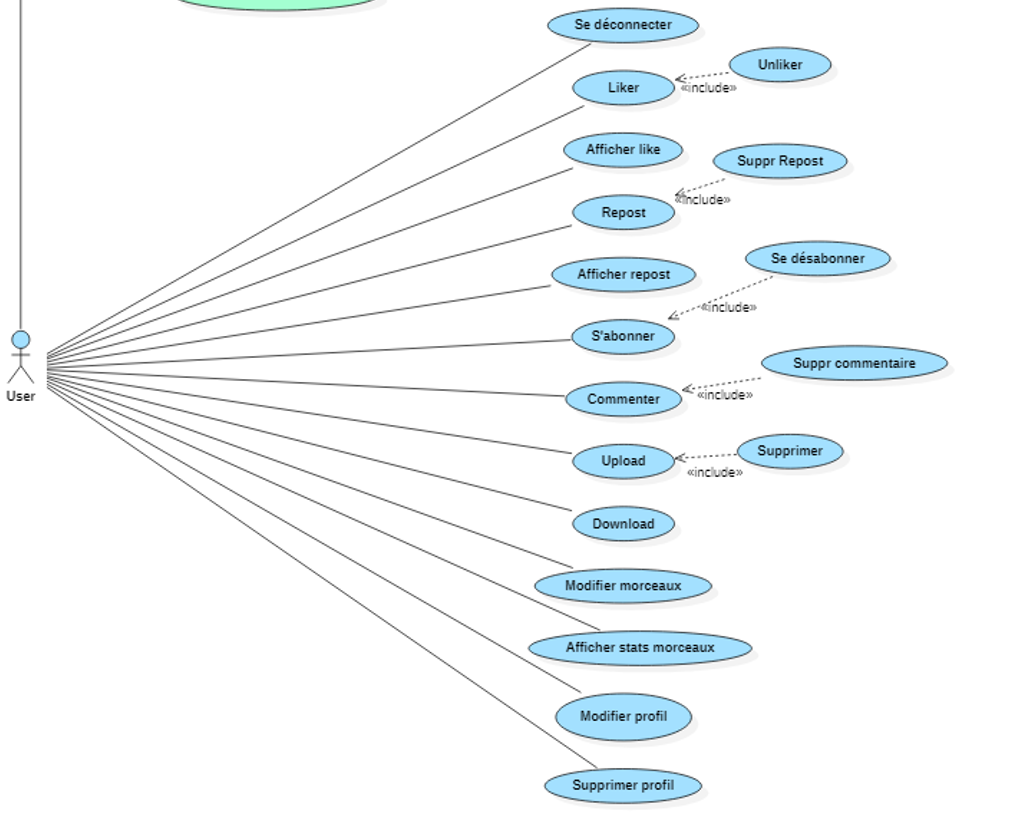
En premier lieu, il convient de présenter le visiteur, il est l’acteur dont les autres héritent pleinement. Celui-ci peut s'inscrire et donc se connecter au site (puisqu’on accède au site sans être connecté). Dans l’optique d’un site complètement terminé, le visiteur a accès à une fonctionnalité de recherche de morceaux. Morceaux qu’il peut donc afficher et écouter. Il peut aussi afficher les profils et les commentaires et les différents abonnements d’un profil.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**L’admin :**

L'administrateur, lui, hérite du visiteur. Il a donc accès en plus à la partie modération du site. Il s’agit là de la possibilité de supprimer du contenu contraire au règlement du site. Par exemple des commentaires ou des morceaux et en dernier recours un compte utilisateur. Il peut aussi modifier les comptes utilisateur, il s’agit là plus de gestion des demandes d’un utilisateur (éventuellement une modification de pseudo ou autre).

**L’user :**



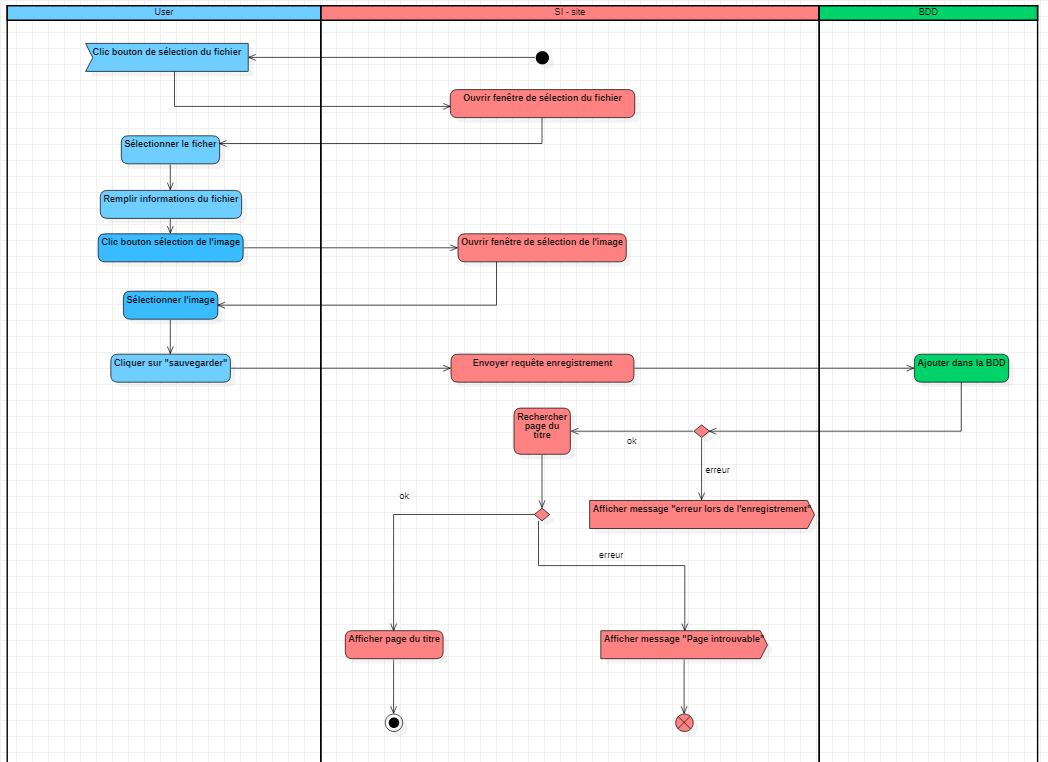
Tout d’abord, l’utilisateur hérite lui aussi du visiteur, en conséquence il peut, tout comme l’administrateur, se déconnecter. Différentes fonctions communautaires lui sont accessibles, ainsi que leur contraire : il peut liker, afficher les like et unliker, il peut reposter un morceau, afficher la liste des repost et supprimer les repost qu’il souhaite enlever. Il peut s’abonner et se désabonner, de la même façon il peut commenter et supprimer ses commentaires.

Viennent ensuite les quatre fonctionnalités qui seront présentées plus bas, celles du CRUD (create, read, update, delete). L’utilisateur peut en effet téléverser un morceau, le créer donc. De toute évidence il peut ensuite l’afficher (fonctionnalité héritée du visiteur), le modifier si une information lui semble erronée et enfin, si cela lui semble nécessaire, l’utilisateur peut supprimer le morceau.

Toujours en rapport aux morceaux, il peut afficher les statistiques d’écoute de ses morceaux afin de connaître le trafic de sa communauté et savoir quels sont morceaux les plus populaires de son profil. Enfin, s’il peut s’inscrire et se connecter à son profil, il peut aussi le modifier voire le supprimer.

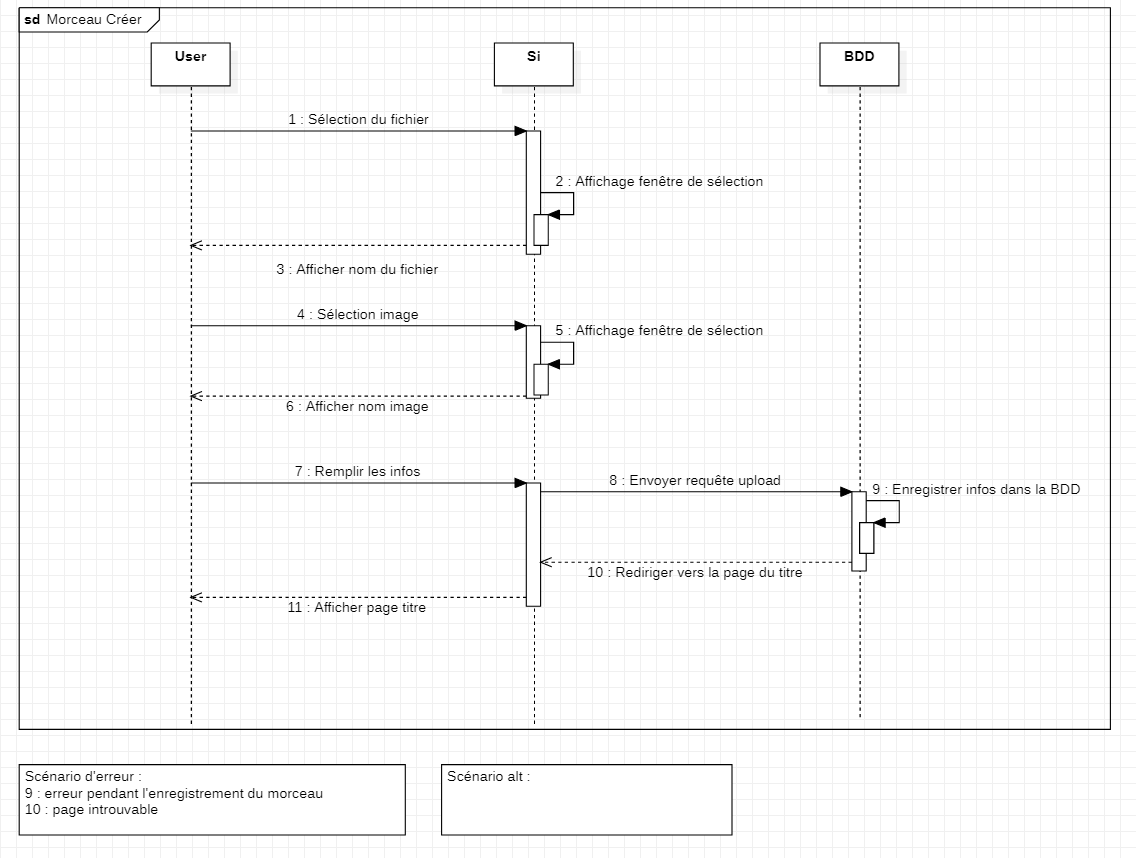
**Fonctionnalité Create :**

Pour la première fonction du CRUD, celle qui consiste à créer - ici à uploader un fichier et les informations le concernant - nous allons voir le déroulé du processus tel qu’il a été conçu en UML.

**Diagramme d’activité :** 

Ici, la fonctionnalité démarre sur la page d’ajout d’un morceau, face à un formulaire l’utilisateur va devoir en remplir chaque partie à commencer par l’input permettant de sélectionner le fichier à uploader via une fenêtre pop up, ensuite l’utilisateur devra remplir les différentes informations demandées : titre, compositeur, description, genre et enfin sélectionner une image de la même manière que le fichier de départ. Enfin l’upload peut démarrer lorsqu’il clique sur le bouton “Ajouter”. Le système envoie donc une requête à la base de donnée, qui va ajouter les différentes informations et notamment le chemin de stockage des deux fichiers (audio et image). S’il y a une erreur lors de l’ajout, le système en informe l’utilisateur, sinon la page d’affichage est recherchée et affichée (sauf erreur là aussi).

**Diagramme de Séquence :**



Pour le diagramme de séquence, nous préciserons tout d’abord qu’il suit le même déroulé que le diagramme d’activité en ne gardant que le scénario premier, celui d’une réussite de la fonctionnalité. Les scénarios d’erreur et alternatif sont présentés en dessous. Ici ce sont les mêmes erreurs qu'au-dessus à savoir une erreur lors de l’exécution de la requête au niveau de la base de donnée et une erreur lors de la recherche de la page à afficher, ici introuvable, en sortie de fonctionnalité.

**Code HTML :**



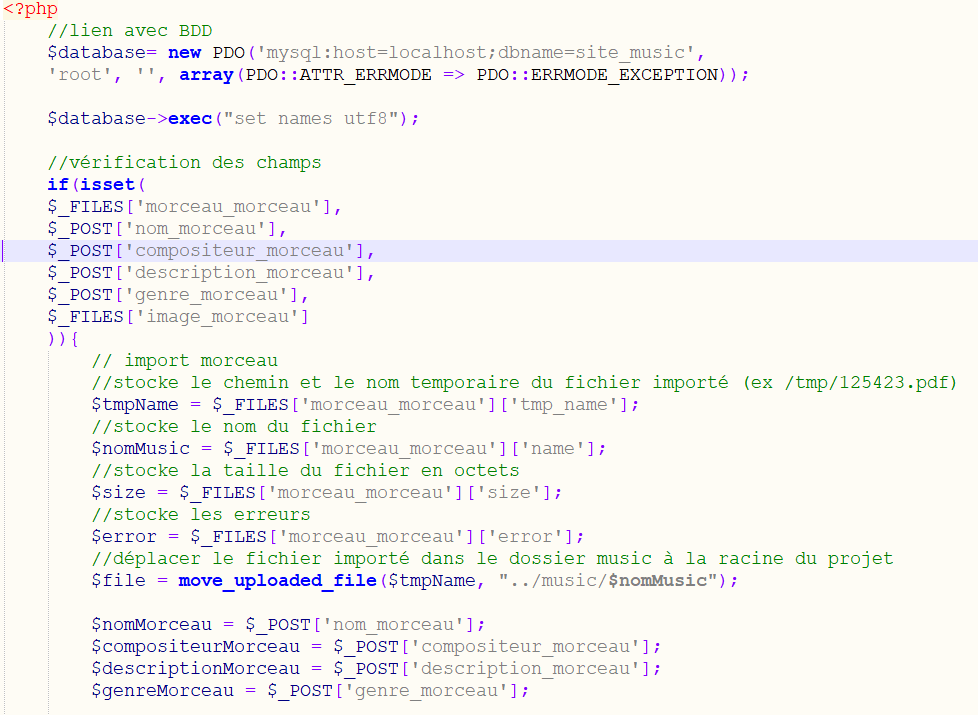
Le code HTML est constitué d’un formulaire, via la balise **<form>**, contenant des **<fieldset>** permettant de bien séparer les différents éléments du formulaire.

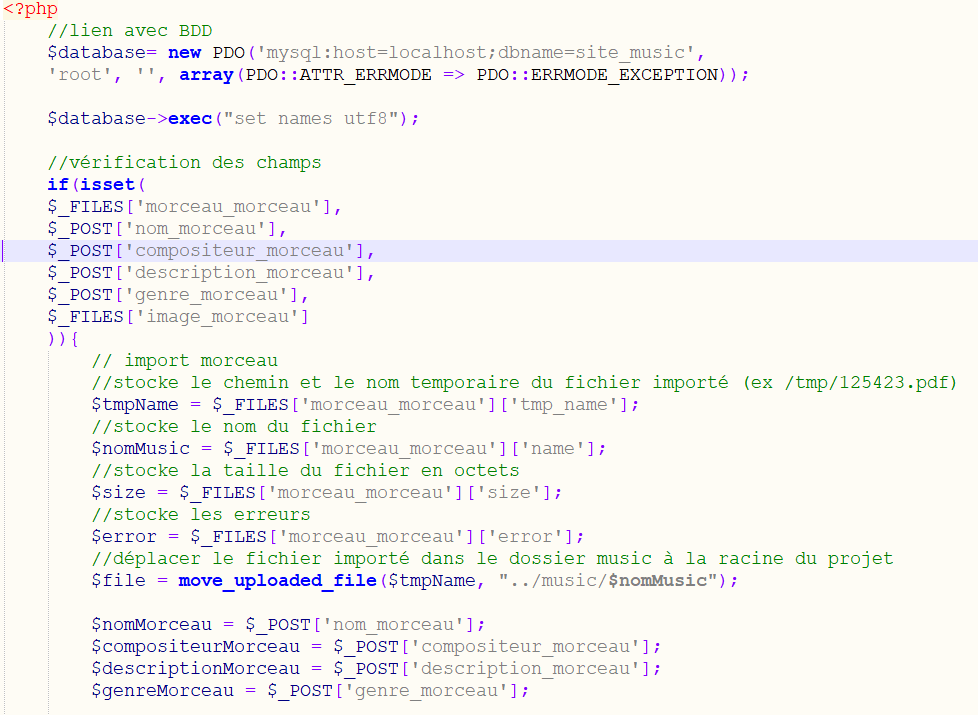
Au sein de ces fieldset nous retrouvons de façon générale une balise <label>

permettant de nommer la section et une balise **<input>** (ou **<textarea>**) pour que l’utilisateur puisse insérer les informations requises.

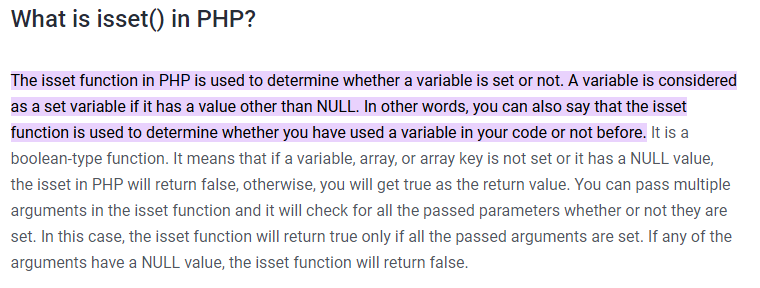
Les différentes sections sont donc : l’insertion du fichier audio (avec une restriction sur les formats acceptés), un titre de morceau, un compositeur, une description en **<textarea>** (plus adapté aux textes de taille moyenne et plus) et enfin un nom de genre à insérer.

Le formulaire se conclut sur un **<input>** de type submit, un bouton d’ajout donc qui permettra de déclencher le traitement via php puis sql des informations ajoutées.

**Code PHP :**

Il est tout d’abord nécessaire de relier notre fonctionnalité à la base de donnée, en insérant au sein du code les différents identifiant de la bdd et ensuite en vérifiant que le format utf8 corresponde bien.

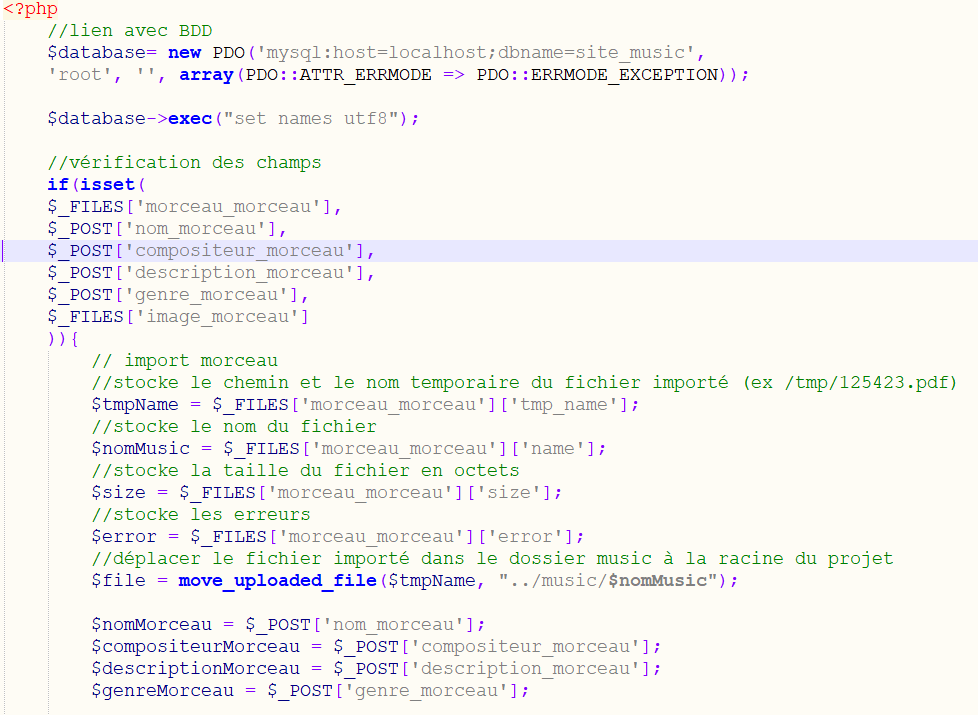
Ensuite il nous faut vérifier que les différents champs soient remplis via **la fonction isset**.

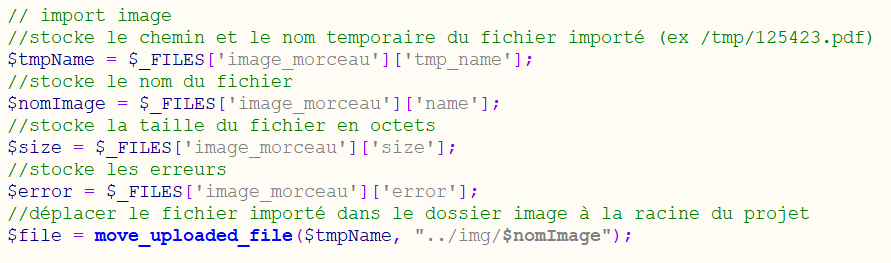
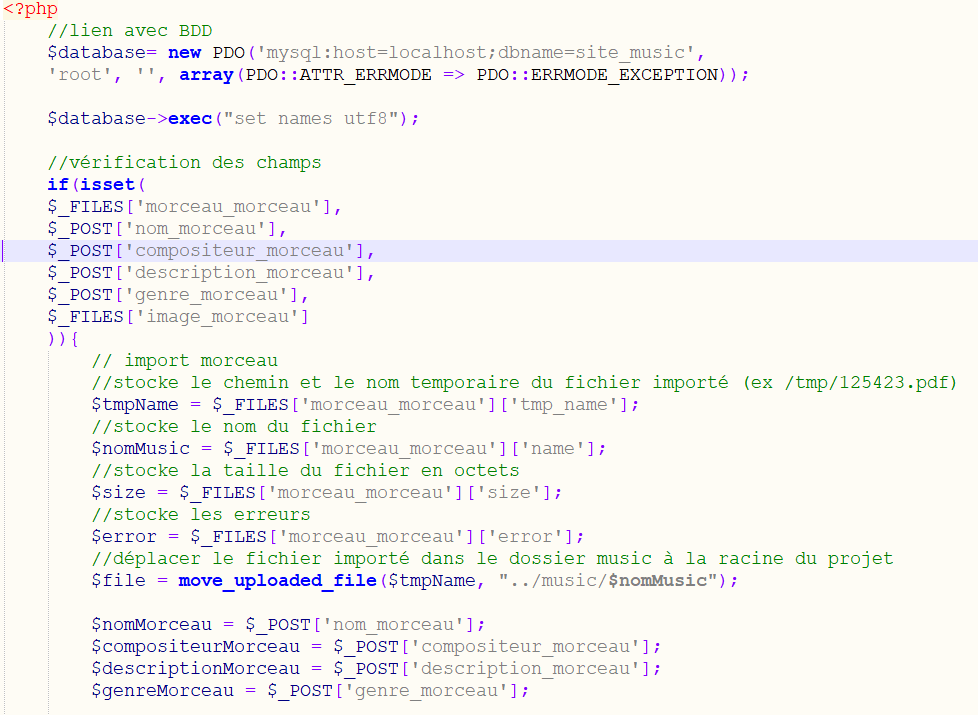


(https://www.simplilearn.com/tutorials/php-tutorial/isset-in-php)

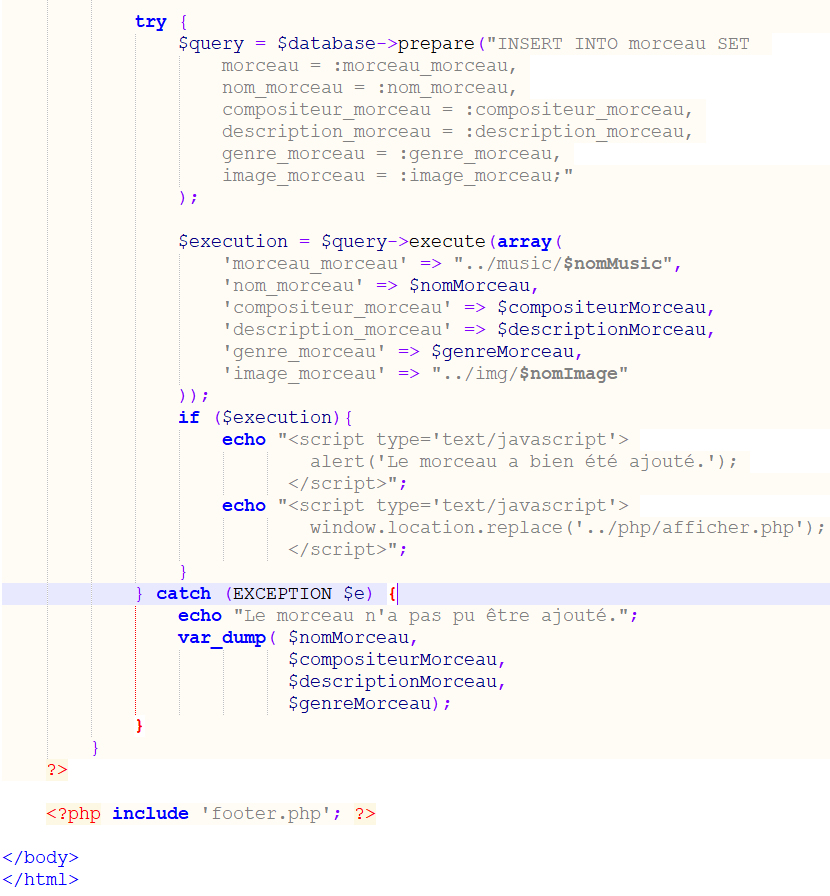
**Traduction :**

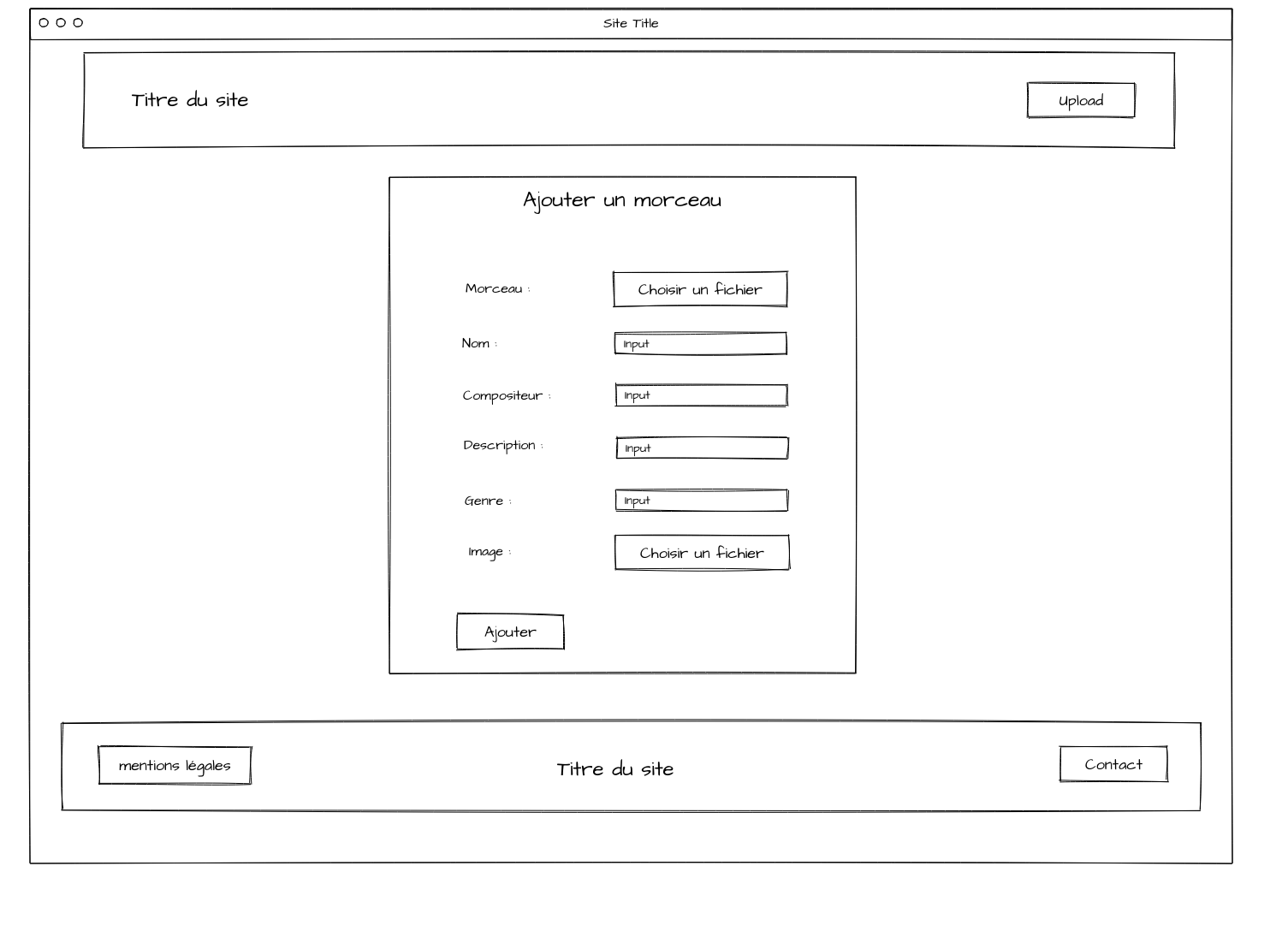
Qu’est-ce qu’isset() en php ?

La fonction isset en PHP est utilisée pour déterminer si une variable est définie ou non. Une variable est considérée comme une variable définie si sa valeur est différente de NULL. En d’autres termes, nous pourrions dire que la fonction isset est utilisée pour déterminer si, oui ou non, nous avons utilisé une variable dans notre code précédemment. Il s’agit d’une fonction de type booléen. Ce qui signifie que si une variable, un tableau ou une clé de tableau ne sont pas définis, ou ont une valeur NULL, isset retournera “false” dans PHP, autrement, la valeur retournée sera “true”. Nous pouvons passer plusieurs arguments dans la fonction isset, celle-ci vérifiera pour tous les paramètres s’ils sont ou ne sont pas définis. Dans ce cas précis, la fonction isset ne retournera “true” que si tous les paramètres sont définis. Si un seul argument à une valeur de type NULL, la fonction isset retournera “false”. 

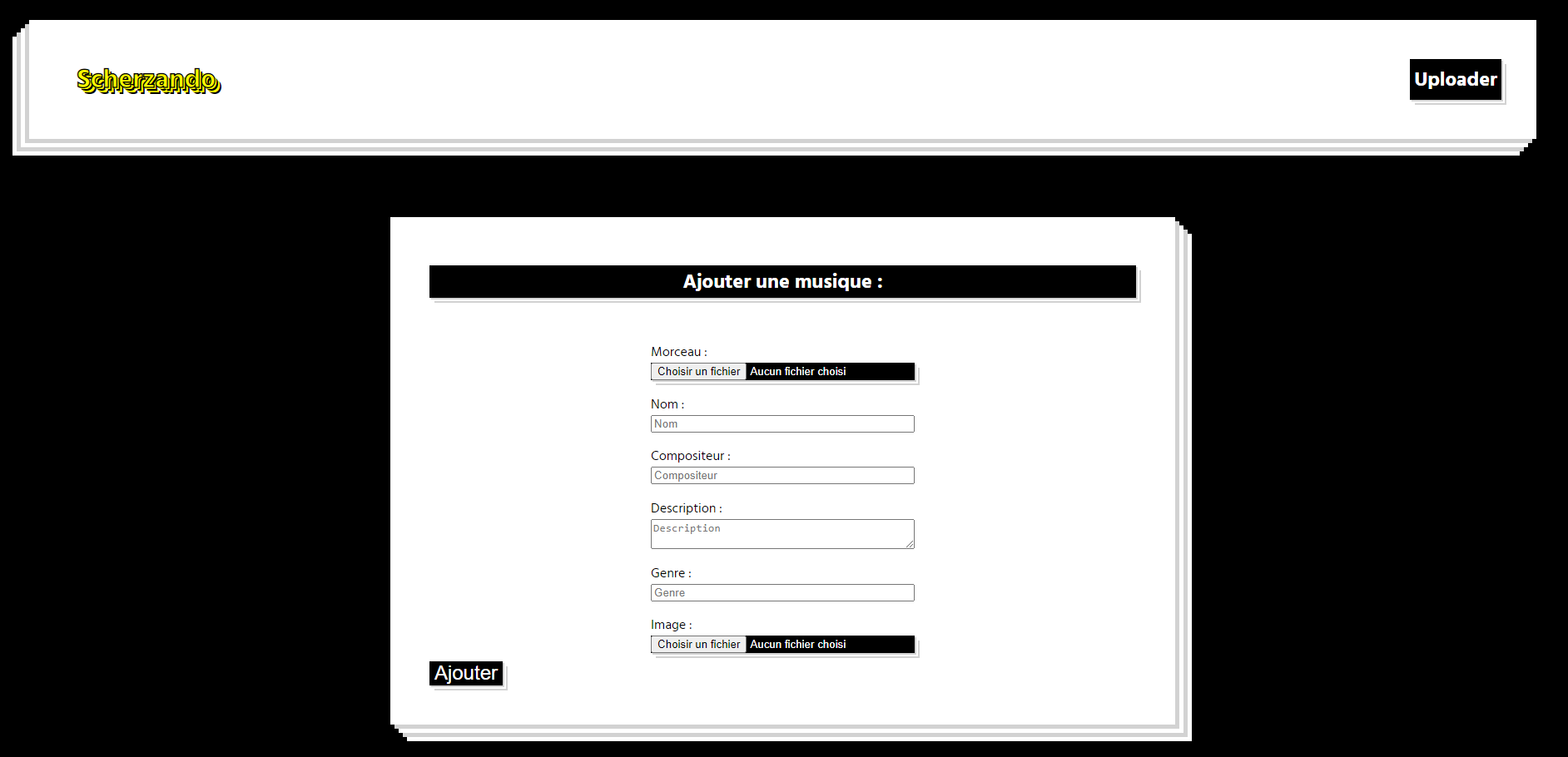


**Code PHP (try & catch) :**



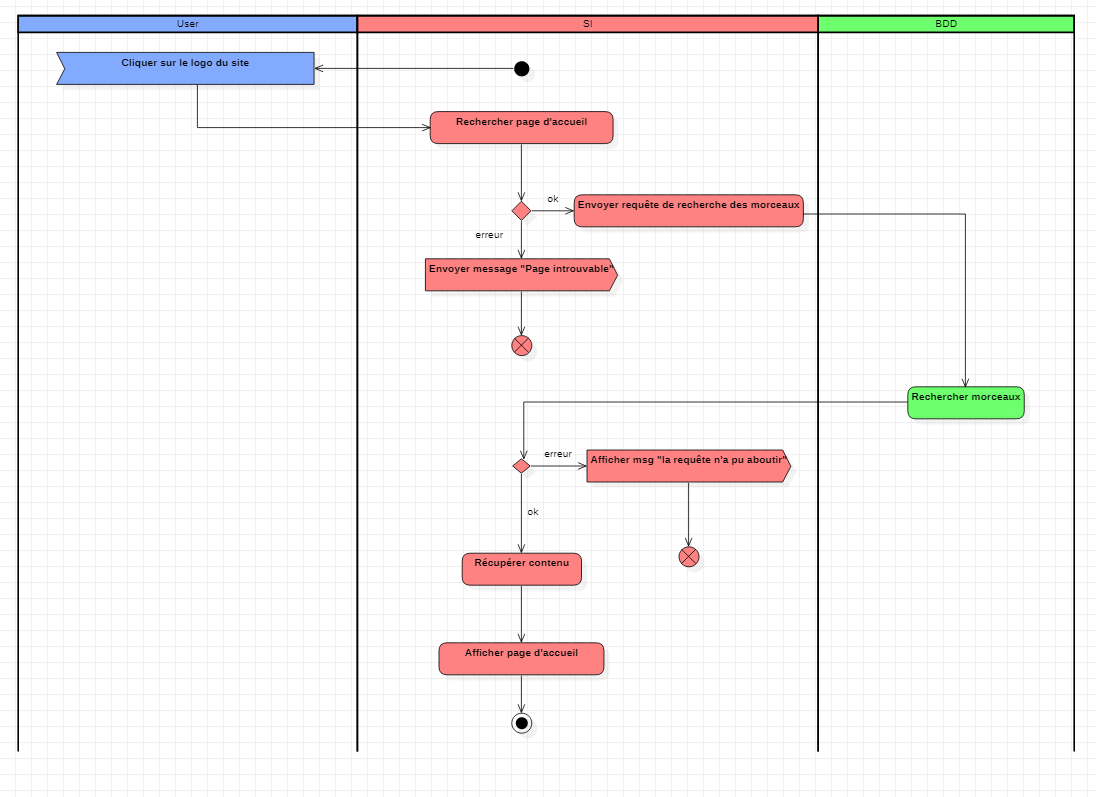
**Wireframe :**

**Capture d’écran :**

****

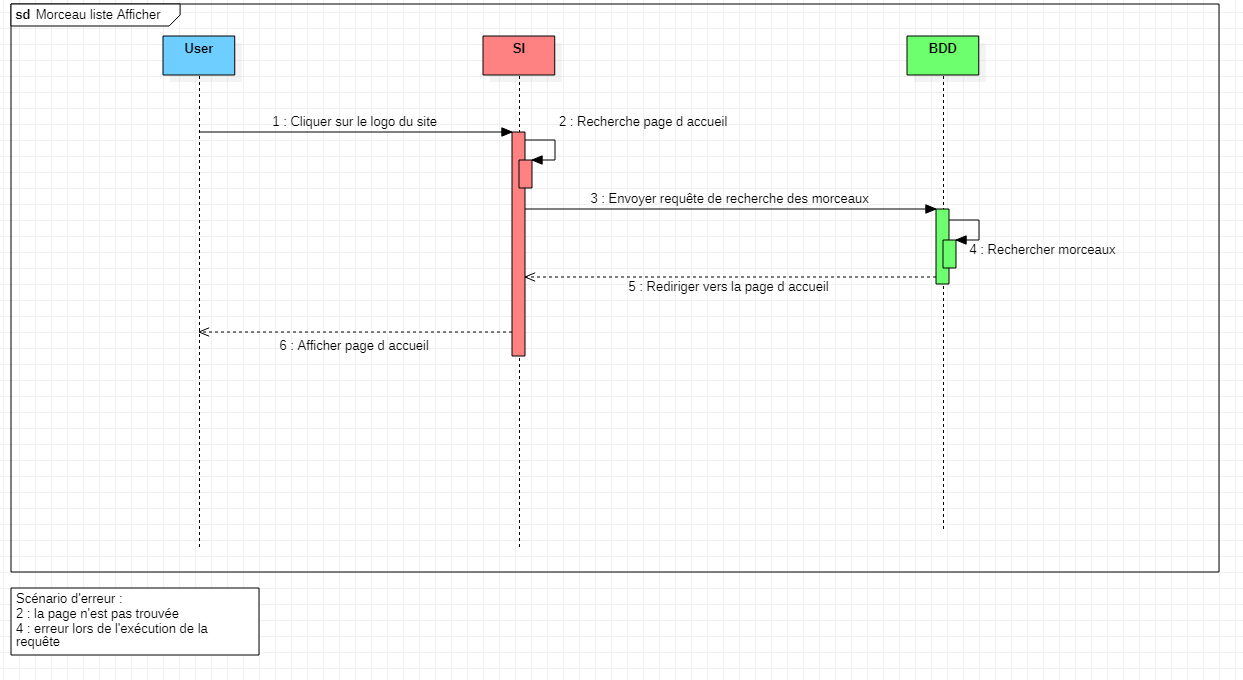
**Fonctionnalité Read :**

Pour la seconde fonction du CRUD, qui ici nous permettra d’afficher un morceau, nous allons voir les différents diagrammes de conception.

**Diagramme d’activité :**

L’affichage se fait lorsque l’on charge la page d’accueil (les morceaux existant dans la BDD et le serveur s’afficheront) ou lorsque l’on vient d’ajouter, de supprimer ou de modifier un morceau, chaque fonction renvoyant vers la page d’accueil et d’affichage. La page est donc recherchée par le système, qui envoie une requête pour récupérer les informations des morceaux à afficher à la BDD, qui renverra le contenu désiré au système qui affichera la page avec le contenu.

**Diagramme de séquence :**



Le déroulé du diagramme de séquence sera le même que celui du diagramme d’activité au niveau du scénario principal. Il n’y a pas ici de scénario alternatif, donc nous nous concentrerons sur les scénarios d’erreur. La première erreur se trouve au niveau de la recherche de la page à afficher, si le système ne trouve pas la page désirée, il renverra un message d’erreur. Il en va de même pour l’aboutissement de la requête en 4, si la base de donnée ne parvient pas à exécuter la requête un message d’erreur sera retourné.

**Code HTML & PHP :**

Ci-dessus, le code PHP s’ouvre par la connexion avec la base de donnée, en spécifiant dans le **PDO** (qui permet cette connexion) les différentes informations permettant de la reconnaître. On définit ensuite la norme de codage universelle en **utf8**.

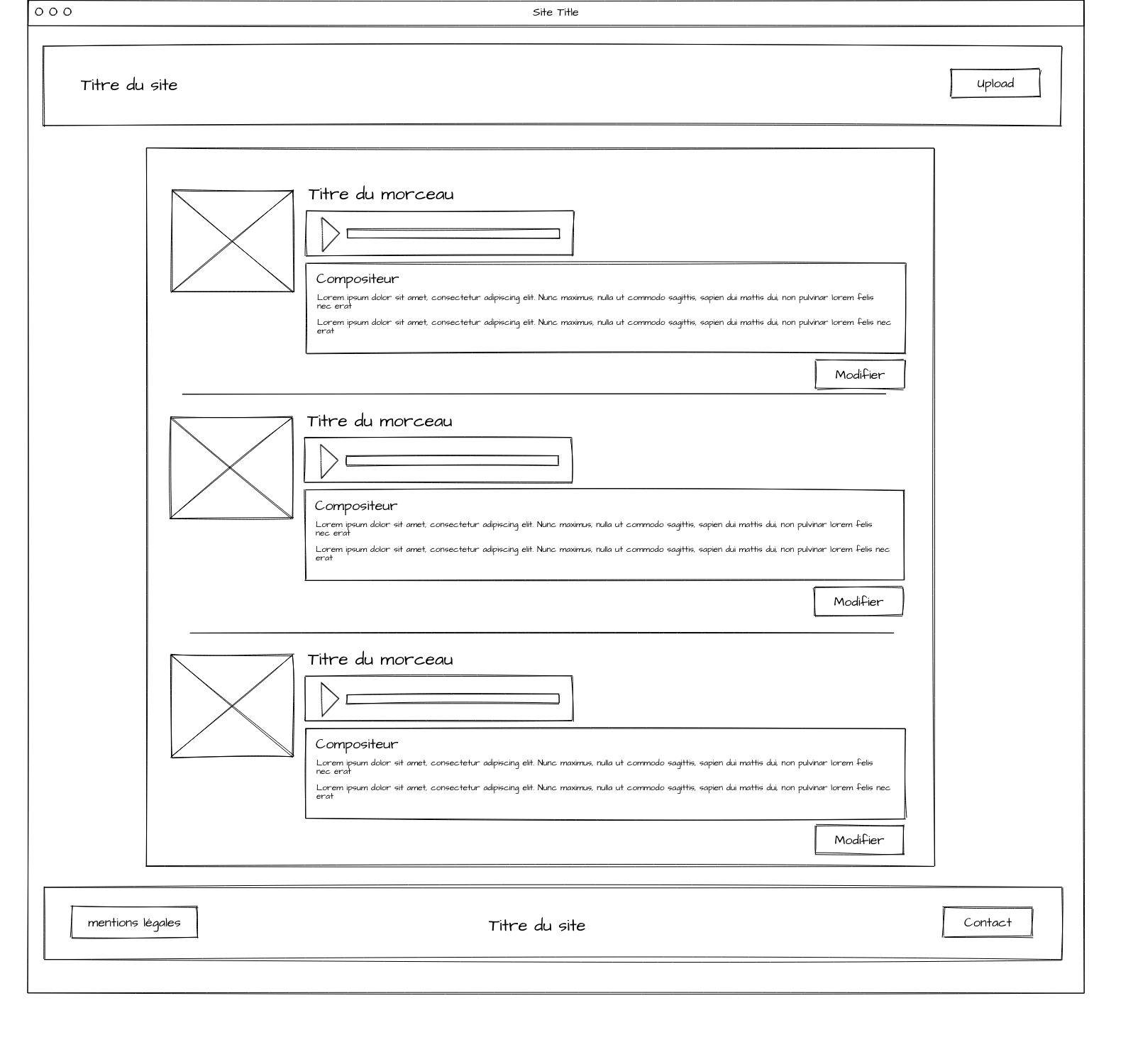
Ensuite on ouvre le bloc **try** - qui nous permettra de lancer notre requête et de gérer les exceptions via le **catch** - et on va ensuite sélectionner dans la base de données les informations qui nous intéressent pour les insérer dans un ensemble de balises HTML créées dynamiquement.

On sélectionne donc la totalité des données dans la **table** “morceau”. Le résultat de la requête sera récupéré et stocké dans un tableau grâce à **fetch()**, tableau contenu dans **$data**.

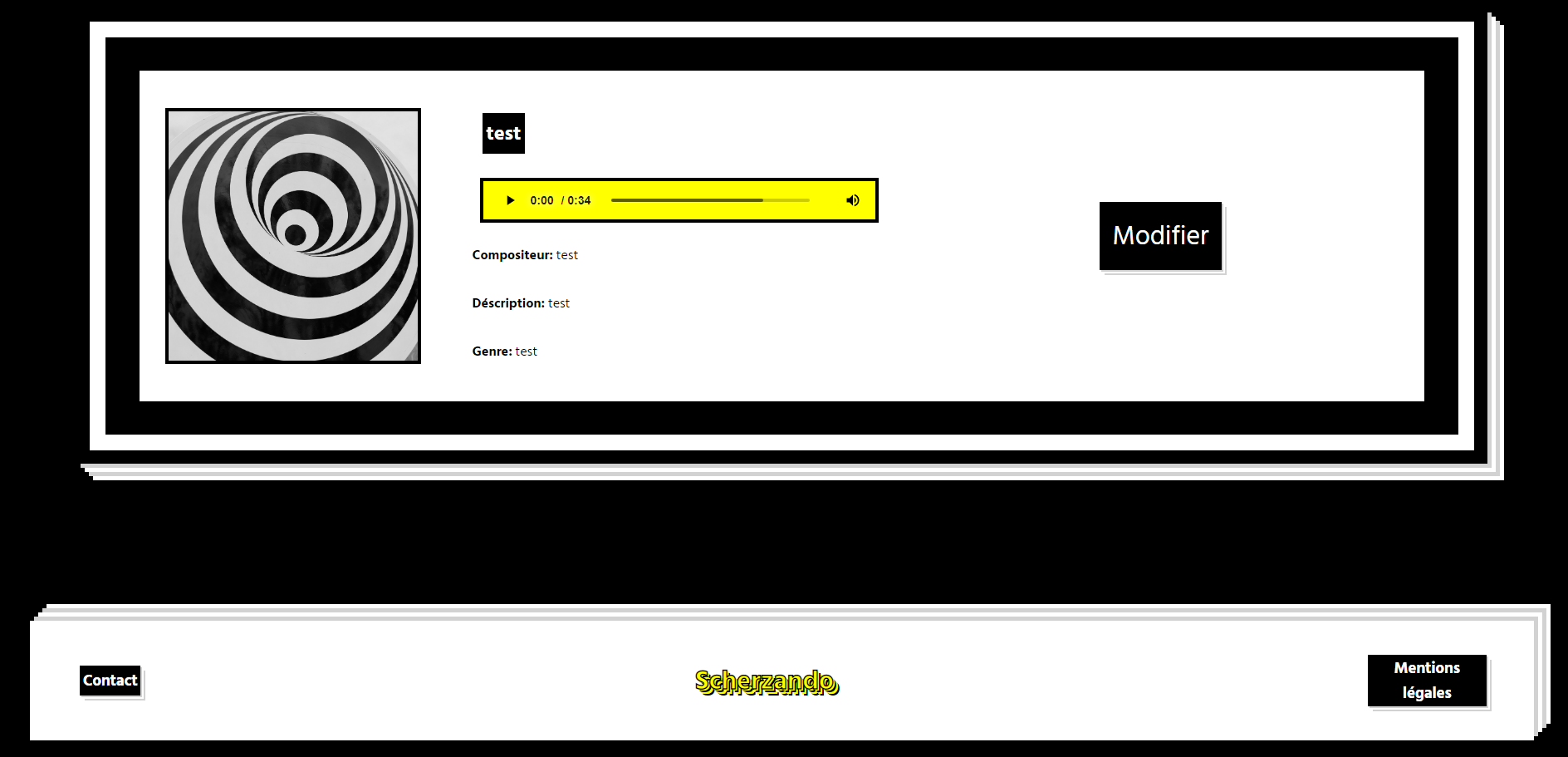
Le **while** implique que tant qu’il y aura des données dans **$data**, on affichera l’ensemble de balises HTML indiquées par echo. On utilise la concaténation via la syntaxe PHP pour insérer des variables (**$data[‘’]**) dans les balises HTML.

Enfin le **catch** nous permet d’afficher un message d’erreur **$e** tout en mettant fin au script via **die()**.

**Wireframe :**

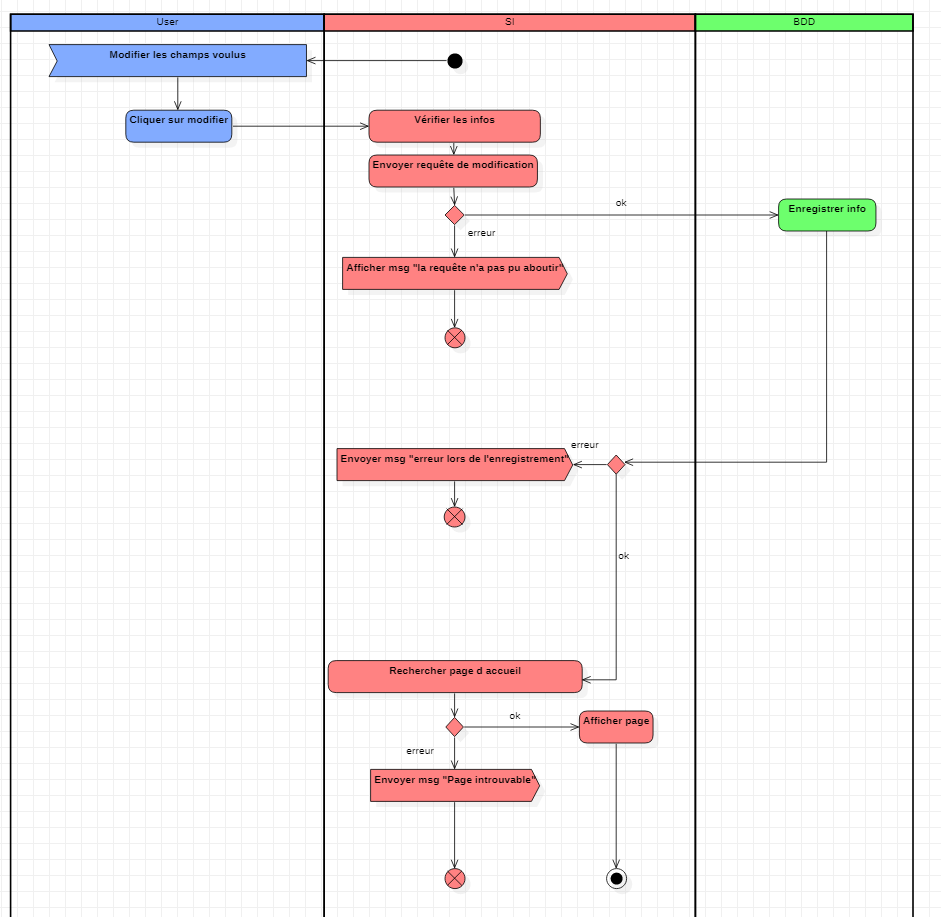


**Capture d’écran :**

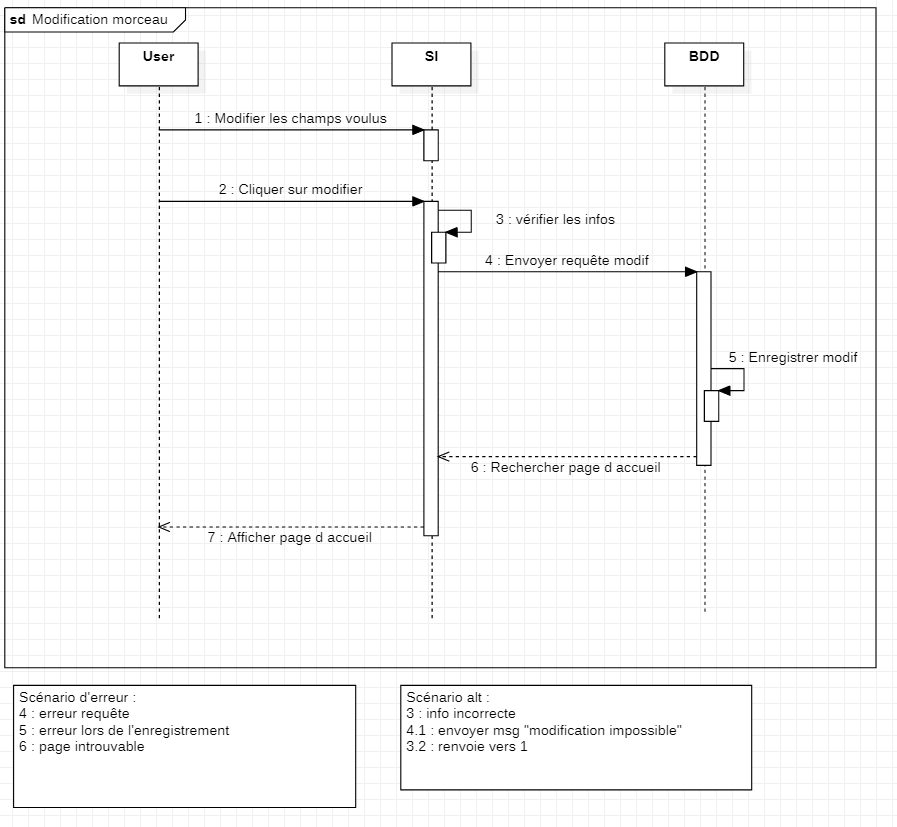
****

**Fonctionnalités Update et Delete :**

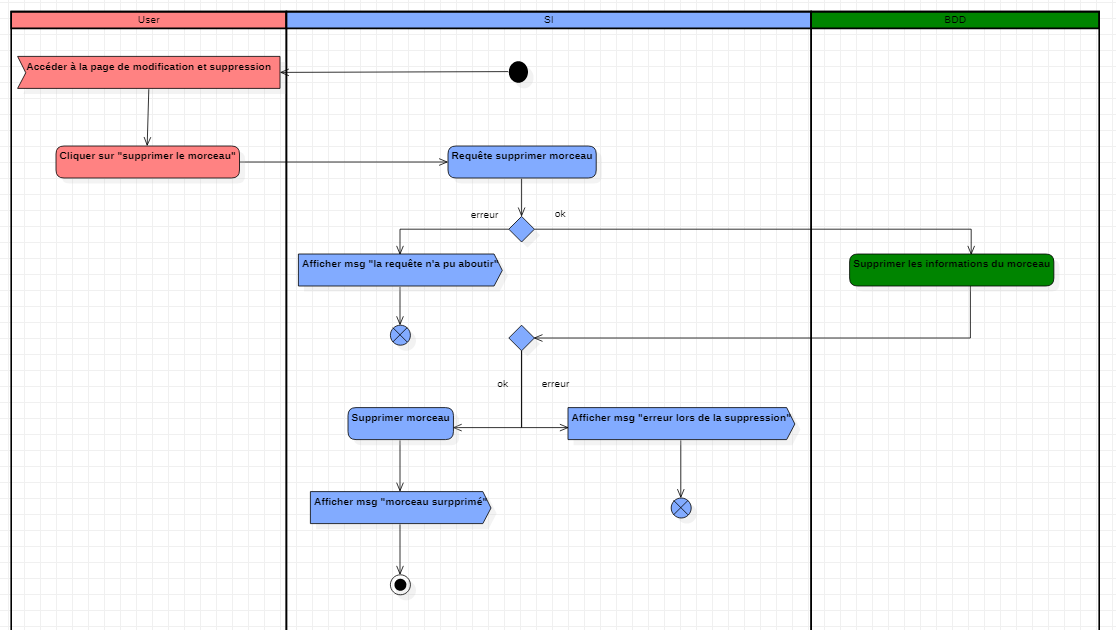
**Diagramme d’activité de l’Update :**

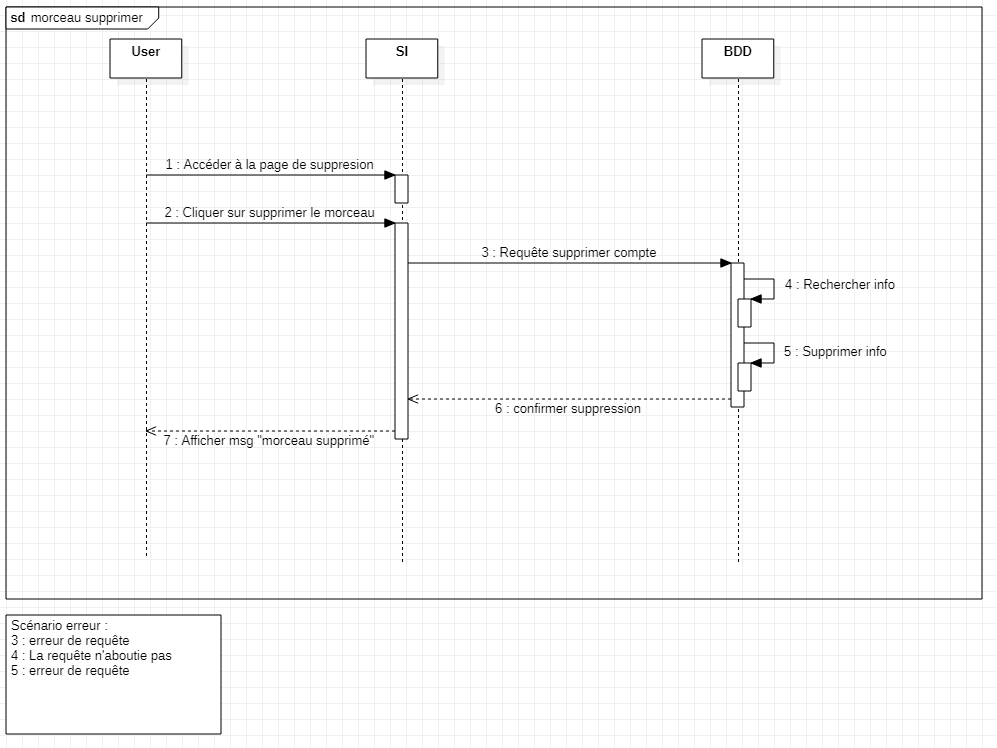
****

**Diagramme de séquence de l’Update :**

****

**Diagrammes d’activité de la Suppression :**

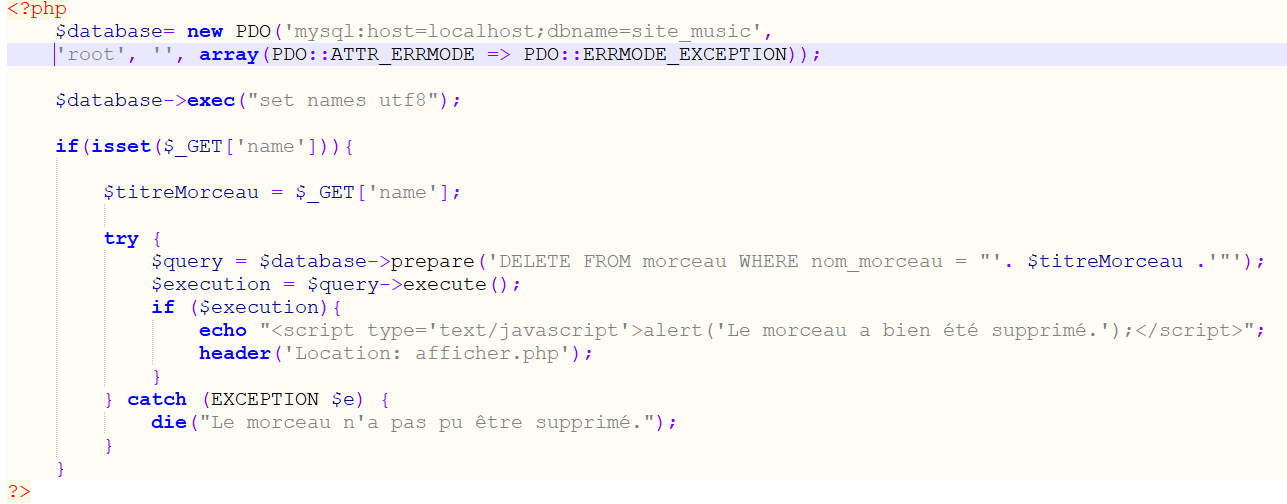
****

**Diagrammes de séquence de la Suppression :**

**Code PHP et HTML regroupant les deux fonctionnalités :**

****

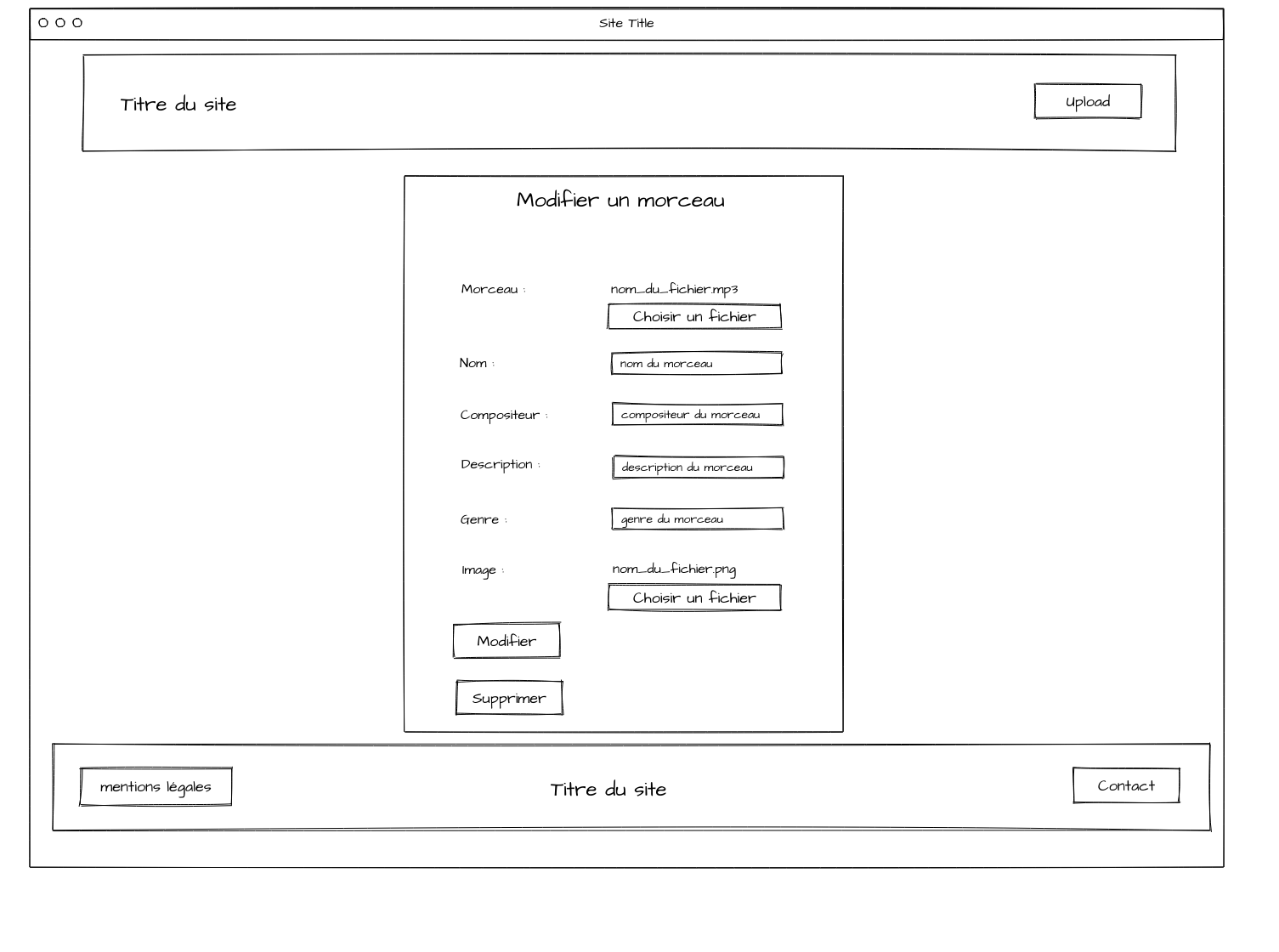
**Script de suppression :**

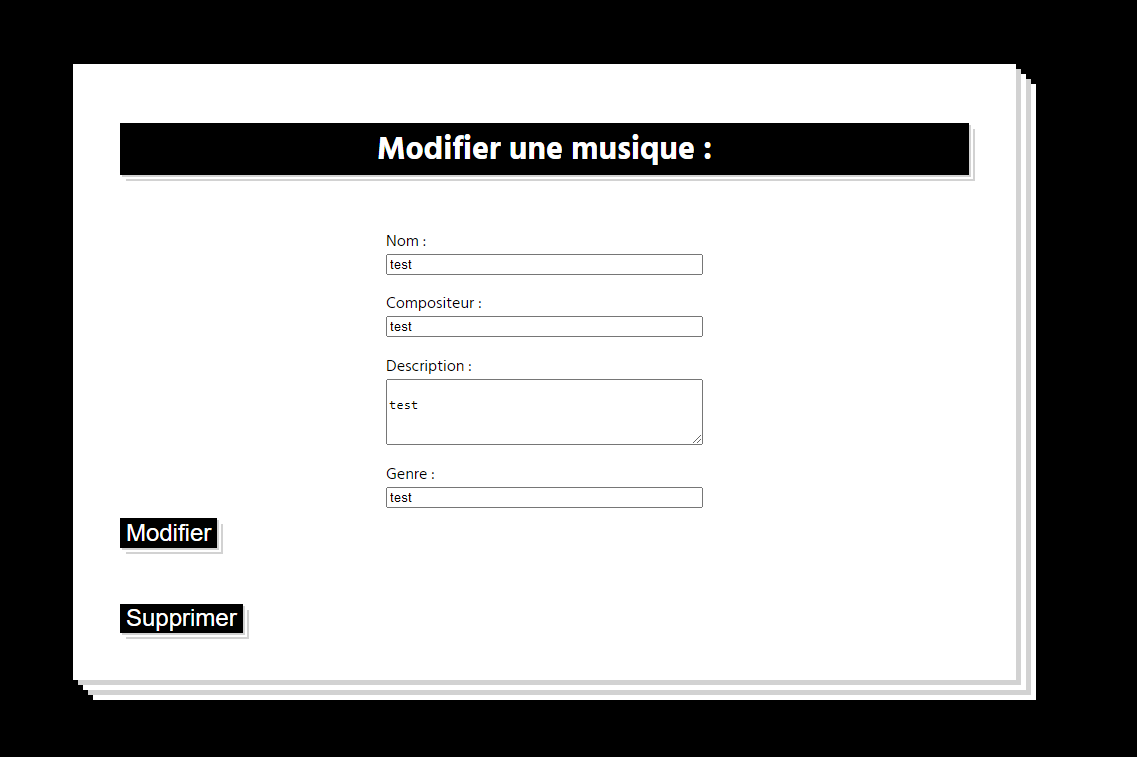
****

**Script de modification :**

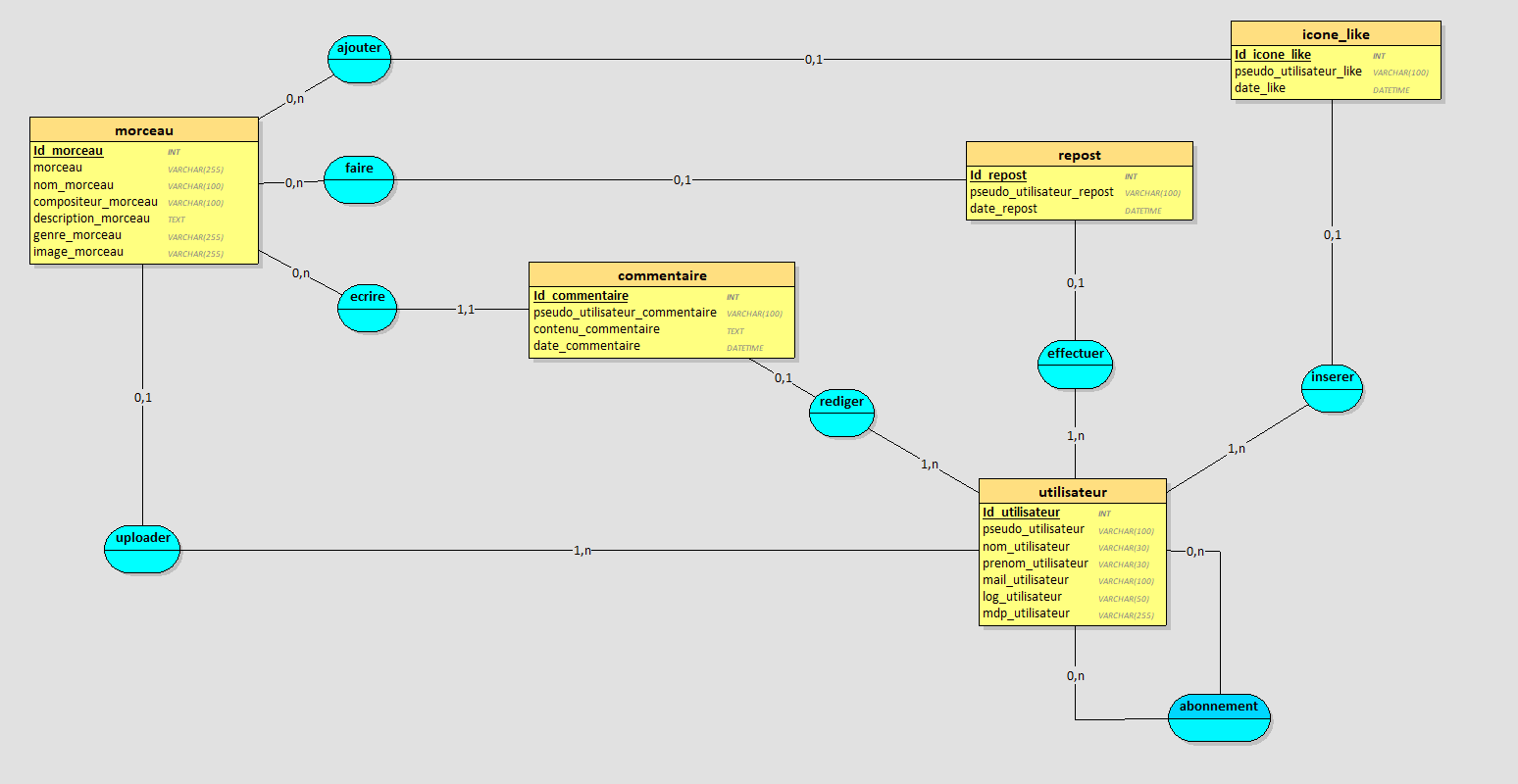
****

**Wireframe de la modification et de la suppression :**

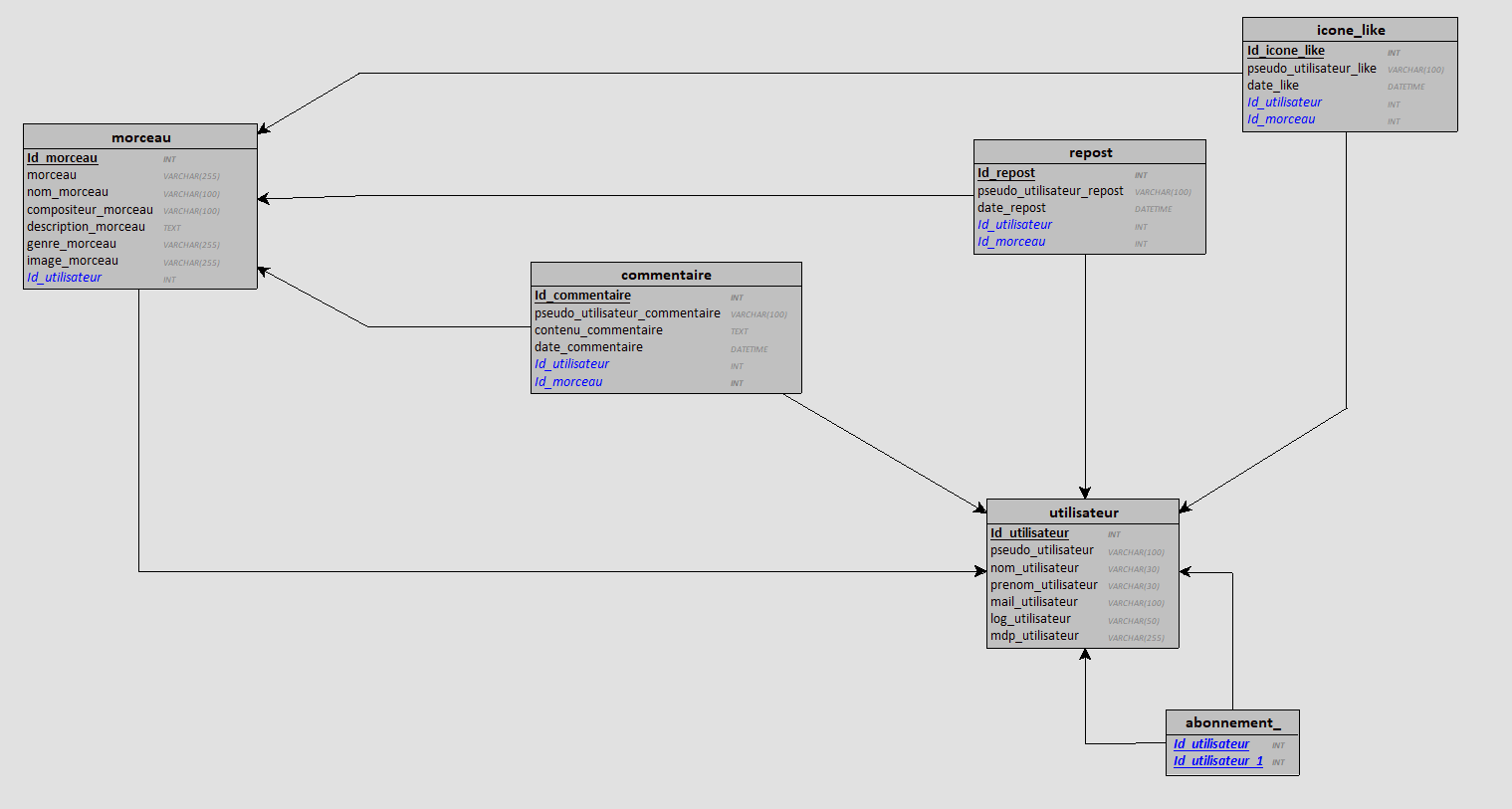
**Capture d’écran de la page de modification et de suppression :**

****

**Modèle Conceptuel des Données :**

****

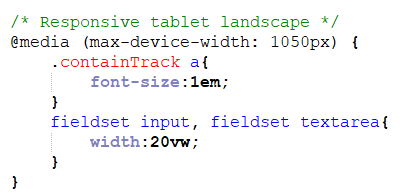
**Modèle Logique des Données :**

****

**Responsivité :**

**À partir de 1050px :**

**Partie CSS :**

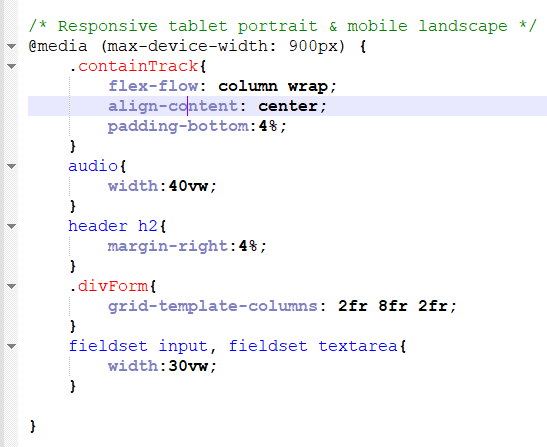
****

****

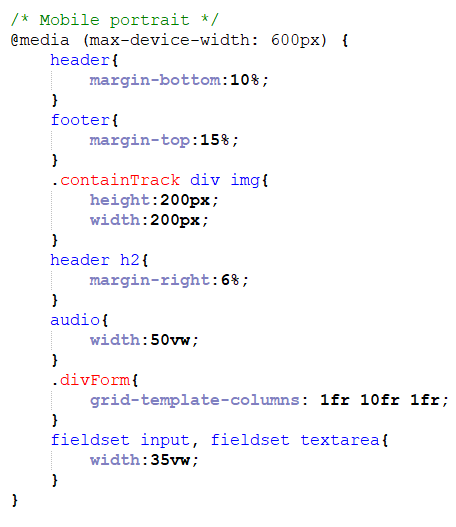
**À partir de 900px :**

****

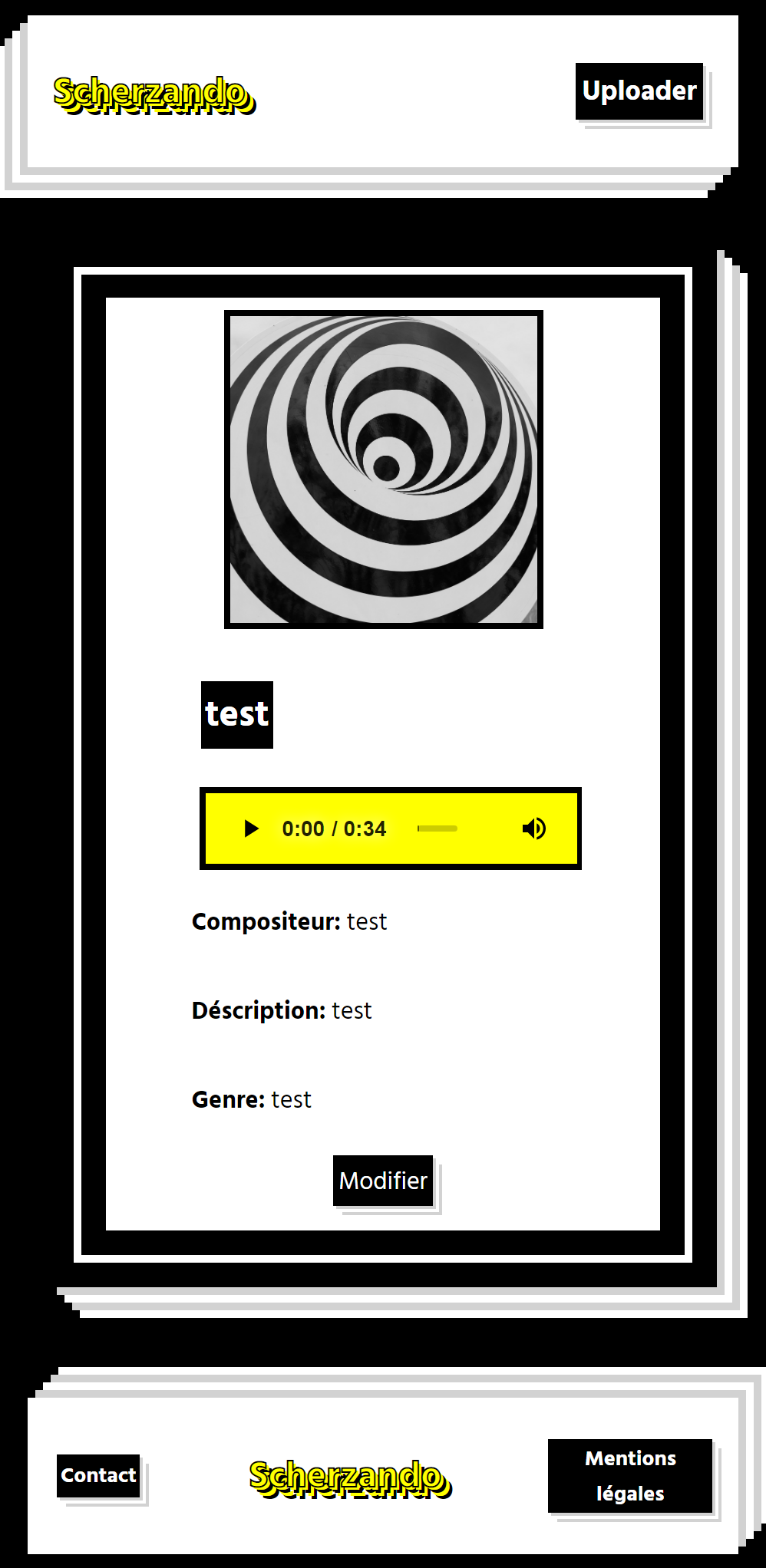
**Partie CSS :**

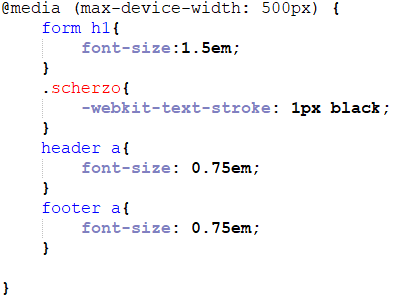
****

**À partir de 600px: Partie CSS :**

****

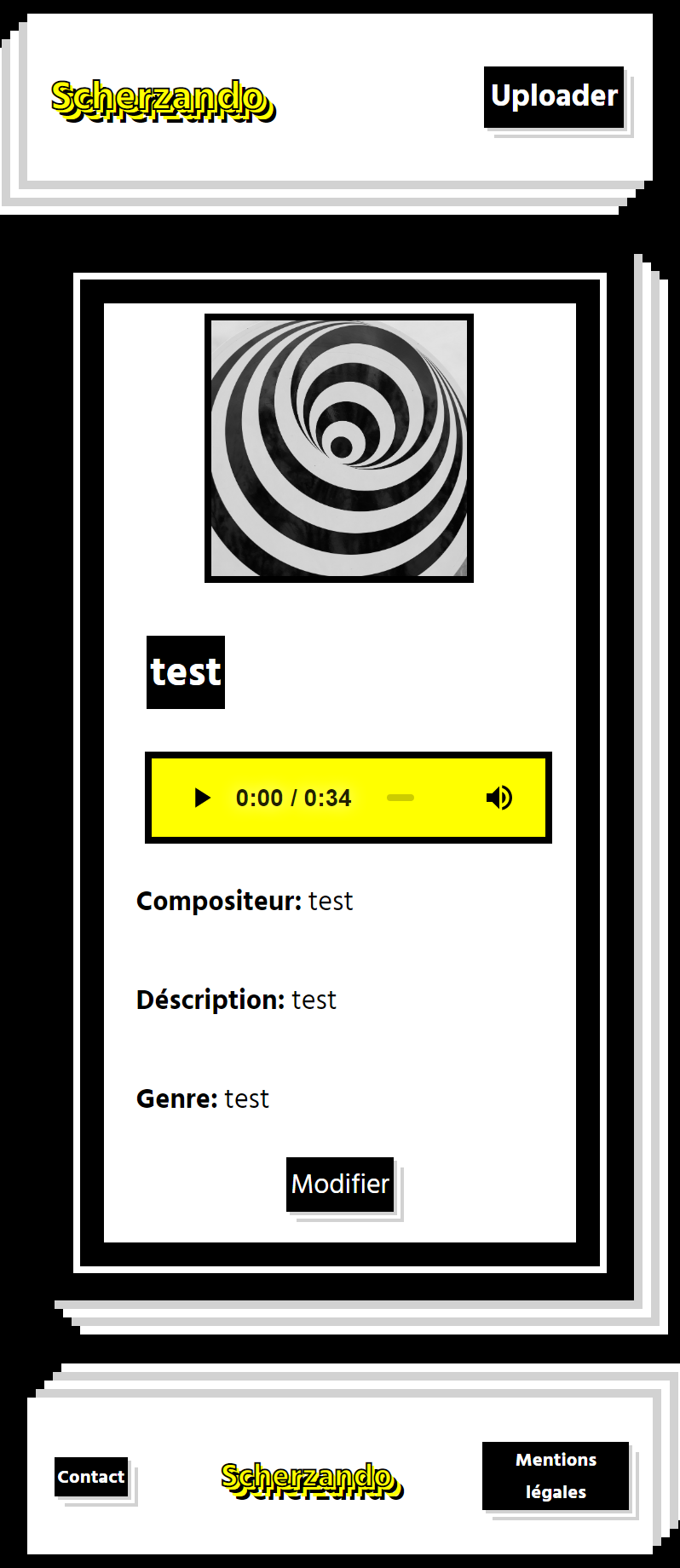
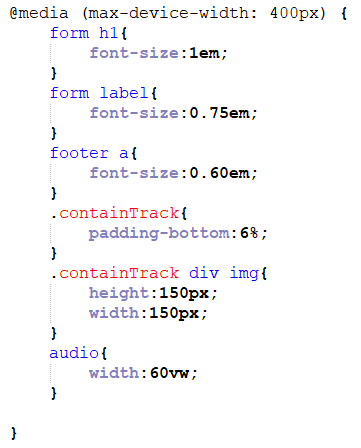
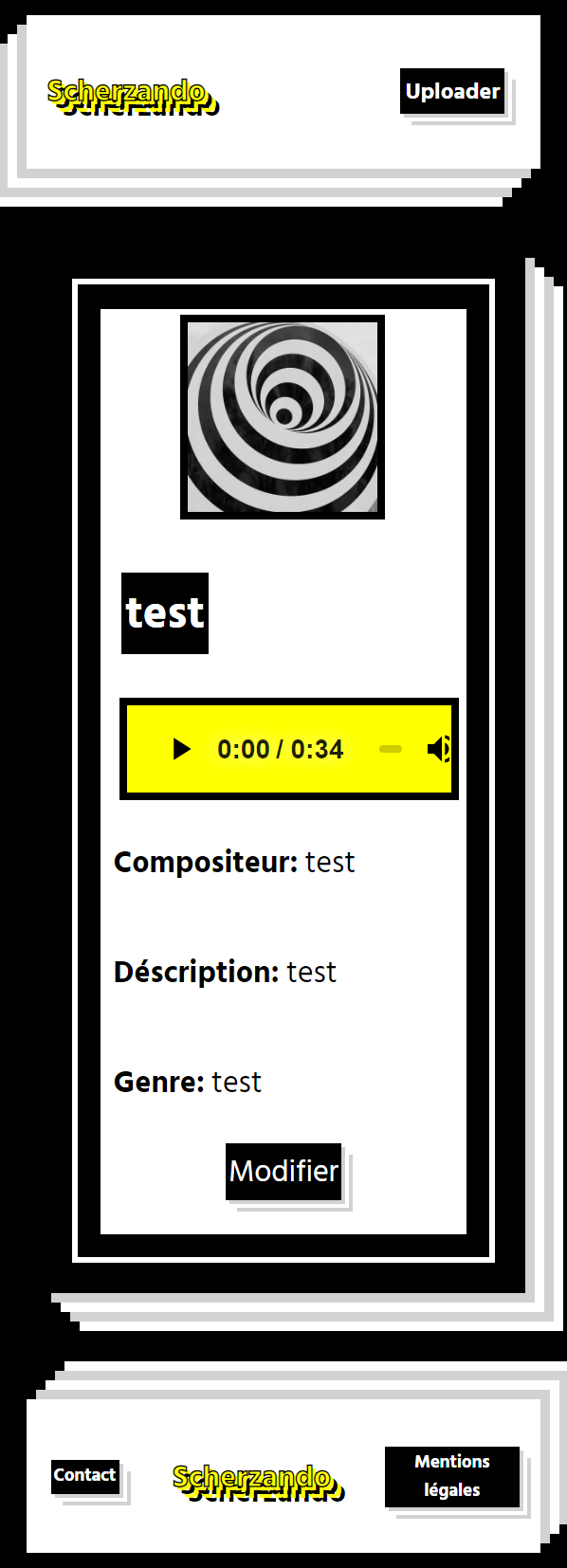
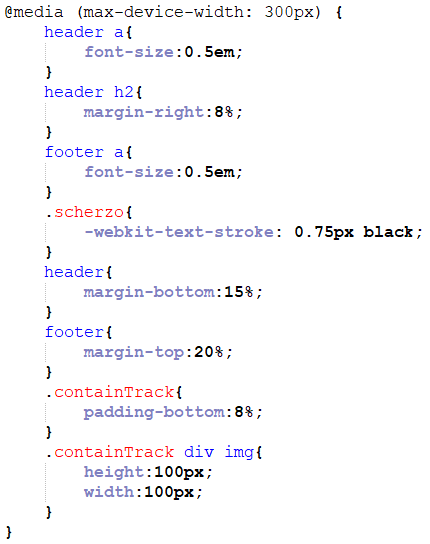
**À partir de 500px : Partie CSS :**

****

****

**À partir de 400px : Partie CSS :**

**À partir de 300px : Partie CSS :**

****