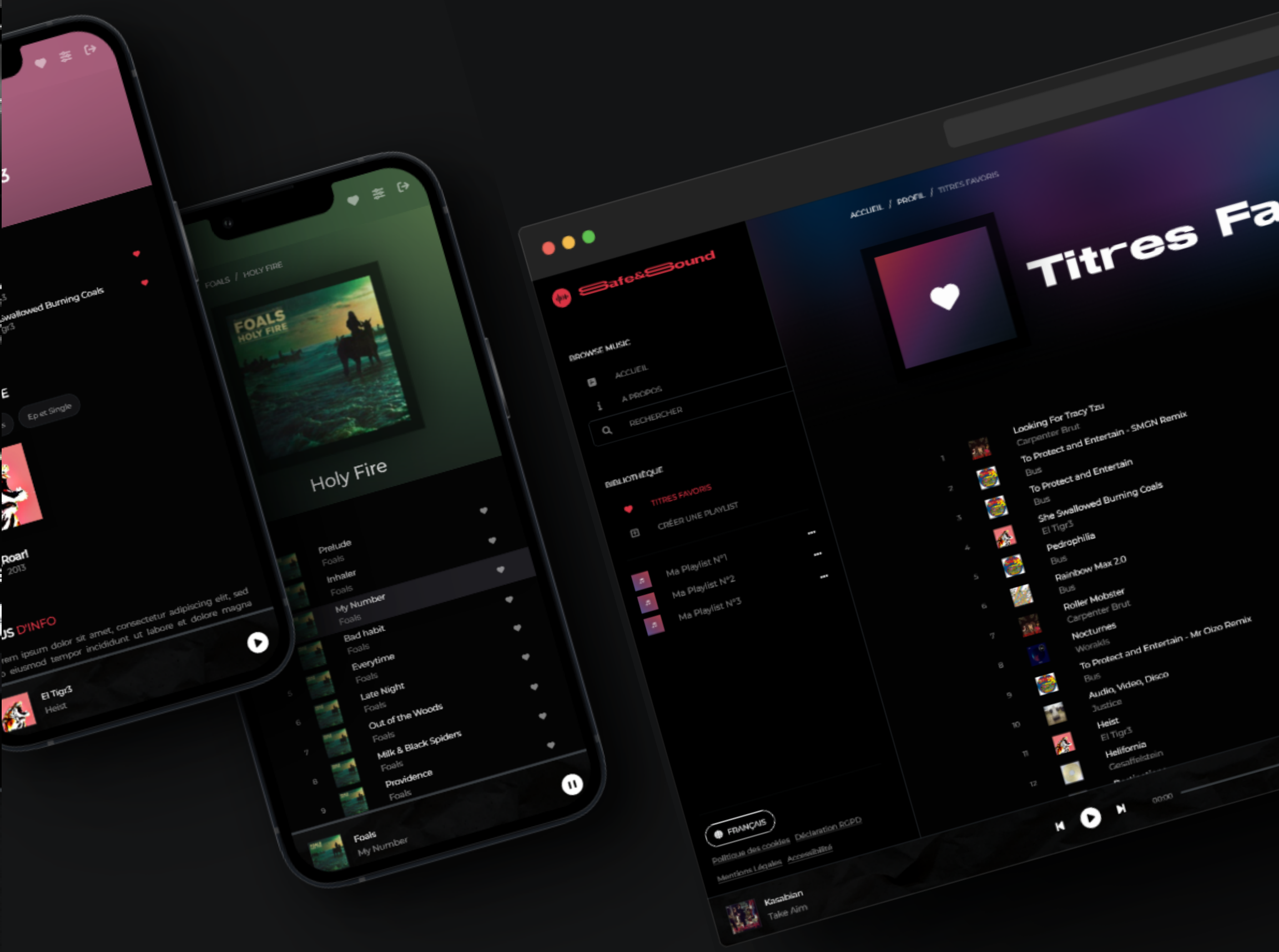
CCI MOSELLE
MÉTROPOLE METZ

# DÉVELOPPEUR WEB
# WEB MOBILE

## RÉSUMÉ DE PROJET

Réalisation d'un site de streaming musical

par Rougeux Max
Développeur Web en formation
2022-2023




Safe&Sound

# PRÉSENTATION PERSONNELLE

Bonjour,

Je m'appelle Max ROUGEUX, né à Créteil le 26 février 1994. J'ai obtenu un bac Scientifique en 2013 et étant passionné par le monde de la musique, j'ai d'abord tenté un DUT Mesures Physiques dans le but de devenir Ingénieur du son. Suite à plusieurs difficultés dans certaines matières, j'ai pris la décision de me réorienter vers une licence SPM (science des particules et de la matière). N'ayant pas vraiment de projet professionnel défini, mis à part un intérêt pour la chimie et l'astronomie, j'ai pris la décision d'arrêter mes études en 2016 après avoir redoublé ma première année de licence.

À partir de là, j'ai principalement alterné entre plusieurs postes en intérim (principalement dans la restauration) et des contrats saisonniers dans quelques centres de vacances pour enfants.

En 2018, n'ayant toujours aucune idée précise d'un métier qui me plairait, Pôle Emploi m'a suggéré de faire une POP (Projet d'Orientation Professionnelle) afin de m'aider dans cet objectif. Un intérêt pour le métier d'ambulancier est né durant cette formation mais a été vite abandonné suite à l'impossibilité de trouver un stage. En perte de motivation, j'ai enchaîné quelques stages dans divers domaines sans éveiller la moindre curiosité de ma part, avant de retourner à des contrats dans la restauration.

En 2021, en discutant avec trois amis qui étaient alors en formation à l'ENI à Rennes, ils m'ont conseillé de me renseigner sur le métier de Développeur Web, sachant mon intérêt pour l'informatique. Ayant eu quelques cours de programmation (C++) à la fac, je me suis dit pourquoi ne pas essayer. Après avoir essuyé un refus de l'école où mes amis étaient en formation, j'ai commencé à chercher ailleurs une formation. Ayant des attaches à Metz, j'ai pu trouver une remise à niveau proposée par le GRETA qui m'a permis de découvrir les métiers de développeur web.

De mars à juin 2023, j'ai donc suivi cette remise à niveau, et à la fin de cette formation, ma décision était prise : je voulais travailler dans ce domaine. J'ai découvert au cours de cette période d'apprentissage un métier dans lequel j'aimerais évoluer, car il est enrichissant en termes de connaissances, non routinier et évolutif. Ayant encore des doutes entre le développement et le design, j'ai cherché la première formation disponible dans ces deux domaines. La formation de Développeur Web et Web Mobile proposée par la CCI de Metz a été la première à répondre positivement à ma candidature.

J'ai toujours quelques doutes quant à la suite, hésitant entre entrer directement dans le monde du travail ou continuer à me former (par exemple, en tant que concepteur d'applications en alternance ou à l'université).

Je vous présente mon projet d'un site d'écoute musicale, Safe&Sound, dans le but de valider cette formation et d'obtenir le titre de Développeur Web et Web Mobile.

Bonne lecture,
Cordialement ROUGEUX Max.

# RÉSUMÉ DU PROJET

Le projet s'articulera autour de sa principale fonctionnalité : l'écoute musicale. L'objectif de ce site web est de mettre à disposition une interface claire et agréable à l'utilisateur pour lui permettre d'écouter, d'ajouter en favoris et de créer des playlist contenant les musiques mise à disposition par le site.

Un utilisateur devra forcement se connecter afin d'accéder à toutes les fonctionnalités, une tentative d'écoute pour un utilisateur déconnecté ou non inscrit se verra interrompue par le formulaire de connexion et un lien vers la page d'inscription. Une fois connecté, le catalogue complet sera disponible pour l'utilisateur, ainsi qu'un suivi de ses écoutes récentes présent sur l'accueil. La mise en place d'un menu latéral présent sur la plupart des pages du site en version pc aura pour fonction de facilité l'accès au contenu créé par ce dernier(favoris et playlist) ainsi qu'aux informations générales du site web.

La page Favoris regroupera toutes les musiques ajoutées aux favoris de l'utilisateur, indiquées par un icône “cœur” coloré en rouge présentent sur cette page.
Un menu présent également sur chaque piste permettra l'ajout aux playlist créé par l'utilisateur ou directement, via un fonction de recherche, sur la page de la playlist en question.
L'utilisateur pourra également modifier son interface (image et bannière) sur les pages contenant les musiques favorites ou ses playlists.
Côté administrateur, le propriétaire du site aura la possibilité d'importer, de modifier ou supprimer le contenu et une gestion des utilisateurs inscrits via une interface dédiée. Un graphique de statistique sera également disponible pour le suivi de son site.

Avant le début du stage, mon tuteur et moi avons eu une réunion d’information afin d’en savoir un peu plus sur ses attentes et sur nos apports mutuels.
Aux premiers jours de la période de stage nous avons définis la structure architecturale du site, les fonctionnalités, code couleur, nombre de pages, différents formulaires, et nom du site et plus tard l’espace administrateur et son contenu. Il y a eu lieu évidemment de faire quelques mises au point tout au long du stage et de ma mission.

# COMPÉTENCES COUVERTE PAR LE PROJET

## COMPÉTENCES LIÉES AU FRONT-END CCP1

- **Développer la partie Front-end d'une application web ou web mobile en intégrant les recommandations de sécurité :**
  - Maquetter une application.
  - Réaliser une interface utilisateur web statique et adaptable.
  - Développer une interface utilisateur web dynamique.
  - Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce.

## COMPÉTENCES LIÉES AU BACK-END CCP2

- **Développer la partie Back-end d'une application web ou web mobile en intégrant les recommandations de sécurité :**
  - Créer une base de Données.
  - Développer les composants d'accès aux données.
  - Développer la partie back-end d'une application web ou web mobile.
  - Elaborer et mettre en oeuvre des composants dans une applications de gestion de contenu ou e-commerce.

# LE PROJET

## CONTEXTE

Ayant garder contact avec une personne voulant créer sa propre plateforme de streaming musical, je lui ai toute de suite proposé mes services pour le stage de ma formation.
Il avait pour projet de faire un site d'écoute musical permettant à l'utilisateur de créer ses propres playlists et d'ajouter des titres à ses favoris tout en proposant un service gratuit. J'ai donc pris contact avec lui pour connaître ses goût en matière de design pour son site et le projet a été lancé.
J'ai donc créé le site web de zéro en passant par les étapes de base d'une création de site web. La maquette fut validée et le code couleur défini. Une fois le développement commencé, quelques échanges furent nécessaires afin de régler certains détails, tels que les tailles des polices et certaines optimisations pour l'utilisateur..

## ORGANISATION DE TRAVAIL

Le commanditaire du site ne se trouvant pas sur Metz, j'ai dû travailler en distanciel. Nous nous sommes régulièrement contacté par téléphone et visio en appliquant la méthode agile, pour faire un point sur l'avancement du site et des potentiels modifications et améliorations à faire.

## COMPOSITION DU SITE

- Menu latéral
- barre de navigation
- page d'accueil
- page artiste
- page album
- lecteur audio
- page de favoris
- page de playlist
- modal de création de playlist
- page compte utilisateur
- page d'inscription
- modal de connexion

# CAHIER DES CHARGES

## OBJECTIF DU SITE :

- Création d'une plateforme d'écoute de musique.
- Créer une interface pratique, intuitive et fonctionnelle.
- Système d'historique d'écoute utilisateur.
- Gestion des favoris et playlist par l'utilisateur.
- Création d'un espace administrateur permettant la gestion du contenu du site.

## CHARTE GRAPHIQUE :

LOGO

### COULEURS UTILISÉES

#DC3545 #f8f9fa

#040404 #212529 #FFFFFF

### POLICES

**Strech Pro**

aAbBcCeEfF

**Montserrat**

aAbBcCeEfF

# DÉTAILS TECHNIQUES

## LOGICIEL UTILISÉ :

Visual Studio Code

Adobe Xd

Xampp

Photoshop

Looping

Canva

FileZilla

## LANGAGE ET LIBRAIRIE UTILISÉS :

### PHP / SQL

PHP (Hypertext Preprocessor) est un langage de programmation open-source principalement utilisé pour créer des applications Web dynamiques et des sites Web. Il est souvent utilisé en conjonction avec des bases de données telles que MySQL, PostgreSQL ou Oracle pour stocker et récupérer des informations. PHP est facile à apprendre, à utiliser et à déployer, et est compatible avec la plupart des serveurs Web. SQL (Structured Query Language) est un langage de requête utilisé pour accéder à une base de données et effectuer des opérations de gestion de données telles que la création, la lecture, la mise à jour et la suppression de données. PHP utilise souvent SQL pour interagir avec des bases de données.

### HTML 5 / CSS 3

HTML (HyperText Markup Language) est un langage de balisage utilisé pour structurer et afficher le contenu d'une page Web. Il fournit la structure de base de la page, telle que les en-têtes, les paragraphes, les listes, les images, etc. CSS (Cascading Style Sheets) est un langage de feuille de style utilisé pour définir la présentation visuelle d'une page Web, telle que les couleurs, les polices, les marges, les bordures, etc. HTML et CSS sont tous deux des langages clés pour la conception et le développement de sites Web.

### Java Script / Jquery

JavaScript est un langage de programmation utilisé pour créer des pages Web interactives et dynamiques. Il est principalement utilisé pour ajouter des fonctionnalités telles que des animations, des effets visuels, des formulaires interactifs, des pop-ups, des menus déroulants, etc. JQuery est une bibliothèque JavaScript qui facilite l'écriture de scripts JavaScript en fournissant une écriture simplifiée pour accéder et manipuler les éléments de la page Web, tels que les balises HTML et les styles CSS. JavaScript et JQuery sont deux langages clés pour le développement front-end de sites Web.

### Bootstrap 5.3 / Fontawesome

Bootstrap est une bibliothèque front-end open-source qui facilite la création de sites Web réactifs et mobiles en fournissant une collection de composants HTML, CSS et JavaScript pré construits, tels que des modèles de grille, des boutons, des formulaires, des icônes, des typographies, des navbars, des carrousels, etc. Fontawesome est une bibliothèque d'icônes open-source qui fournit une collection d'icônes vectorielles utilisables dans les projets Web et les applications. Bootstrap et Fontawesome sont des bibliothèques clés pour le développement de sites Web réactifs et mobiles.

**MySql**
MySQL est un système de gestion de base de données relationnelle open-source largement utilisé pour stocker et récupérer des informations à partir d'une base de données. Il utilise le langage SQL pour effectuer des opérations de gestion de données telles que la création, la lecture, la mise à jour et la suppression de données. MySQL est souvent utilisé avec PHP pour créer des applications Web dynamiques et des sites Web.

# FONCTIONNALITÉS DE SAFE&SOUND :

Il s'agit ici de la logique et des différentes fonctionnalités du site.

- **Système de lecture**
  Safe&Sound propose un service de lecteur audio disponible uniquement pour un utilisateur connecté. Via une condition par rapport à une variable seulement définie lors de la connexion, ce dernier se verra proposer de se connecter ou s'inscrire s'il clique sur un piste en étant déconnecter (if(isset($_SESSION['id_user']))).
  Une fois la lecture lancé, l'utilisateur pourra retrouver son historique de lecture sur la page d'accueil grâce à une fonction AJAX exécutée à chaque lecture. Cette dernière récupère l'id de l'utlisateur et celui la musique lancée ainsi que la date d'écoute (heure, minute et secondes comprises) dans une table nommée "historique").

- **Favoris et playlist**
  Chaque utilisateurs connectés aura la possibilité de mettre en favoris les musiques voulues via une icône présente sur chaque piste. Après un clic sur ce dernier, l'icône devient rouge et la musique pourra être retrouvée sur la page favoris propre à l'utilisateur ainsi que la date d'ajout. Une fonction AJAX s'occupe de cette fonctionnalité en ajoutant l'id de l'utilisateur et celle de la musique ainsi que la date d'ajout dans la table "favoris".

  Même principe pour les playlists, après avoir créée une playlist via le menu latéral, l'utilisateur pourra ajouter des musiques à chacune de ses playlist via un menu “dropdown” présent sur chaque piste ou directement sur la page de la playlist en question via un formulaire de recherche. Cette fonction recherche s'effectue via une fonction AJAX récupérant le texte saisi dans l'input et affiche les musiques commençant par le texte saisi et les musiques contenant le texte saisi non présente dans la première condition. Toutes les musiques affichées par cette fonction doivent respecter la condition de ne pas être déjà présente dans la playlist.

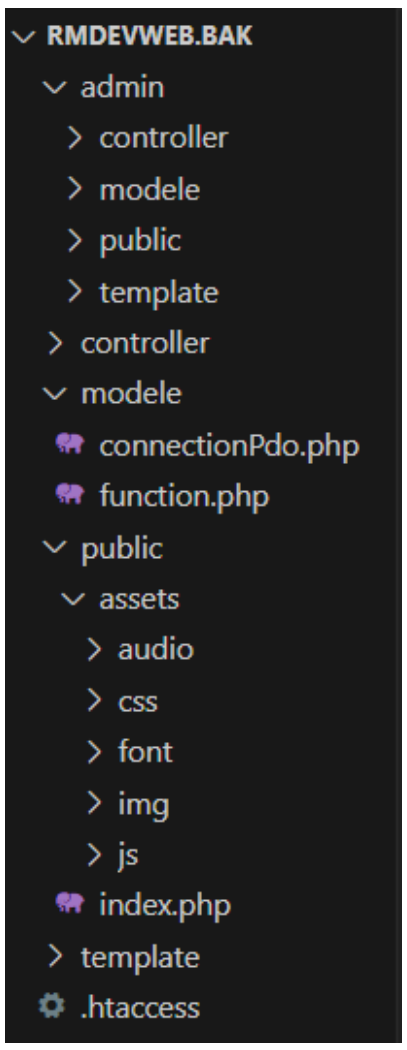
- **Recherche**
  Une barre de recherche est mise à disposition (sur le menu latéral en version pc, en haut de la page d'accueil en version mobile) pour l'utilisateur afin de chercher directement le contenu qui l'intéresse. Une fois valider le formulaire de recherche, le texte saisi est récupéré via méthode POST et la fonction lié à cette page se lance et elle récupère :
  - tous les artistes commençant par le texte saisi et les artistes contenant le texte saisi non présents dans la première condition
  - tous les albums commençant par le texte saisi et les artistes contenant le texte saisi non présents dans la première condition
  - toutes les musiques commençant par le texte saisi et les musiques contenant le texte saisi non présents dans la première condition

  Les résultats s'affichent sous forme de lien de la manière suivantes :
  - Le premier artistes récupéré par la fonction est considéré comme meilleur résultat et sera mise en valeur.
  - une liste de musique
  - une liste d'artiste sans l'artiste mise en valeur
  - une liste d'album

- **Système de suivi artiste**
  Sur chaque page d'artiste, un bouton suivre est disponible pour permettre à l'utilisateur d'accéder plus facilement à ces artistes préférés. Une fois cliqué, le bouton change d'aspect (bouton noir "s'abonner" devient un bouton blanc "suivi") et une fonction AJAX rentre l'id utilisateur et l'id de l'artiste dans une table abonnement. Dans le menu latéral, une fonction récupérant les données de la table abonnement propre à l'utilisateur connecté se chargera de construire via une boucle tous les liens des artistes suivi.

- **Personnalisation du profil utilisateur**
  Sur la page compte de l'utilisateur,  une gestion de sa photo de profil et photo de bannière (présente sur sa page "favoris" et "playlist") lui est proposé. Chaque onglet de l'accordion Bootstrap lui offre la possibilité de changer la photo par défaut fourni lors de l'inscription ou de remettre une ancienne image encore stocké dans la base de données. Un booléen étant présent pour chaque image des deux table "img_user" et "banner_user" permet de savoir quelle image est considérée comme active pour l'utilisateur. Changer l'image actuelle par une nouvelle passera l'ancienne en hors ligne via la valeur du booléen mis a 0 et ajoutera la nouvelle image dans la table de l'utilisateur avec le booléen à 1.
  Même principe pour la gestions des anciennes images, une modification du booléen de 0 à 1 réactivera les anciennes images.

# STRUCTURE ET AGENCEMENT DES FICHIERS POUR LA CONCEPTION DU SITE WEB :




Le modèle MVC (Modèle-Vue-Contrôleur) est une architecture de conception de logiciels largement utilisée pour développer des applications web. La raison principale pour choisir le modèle MVC lors de la conception d'un site web est qu'il offre une structure bien organisée et facile à maintenir.
Voici quelques avantages clés de l'utilisation du modèle MVC :

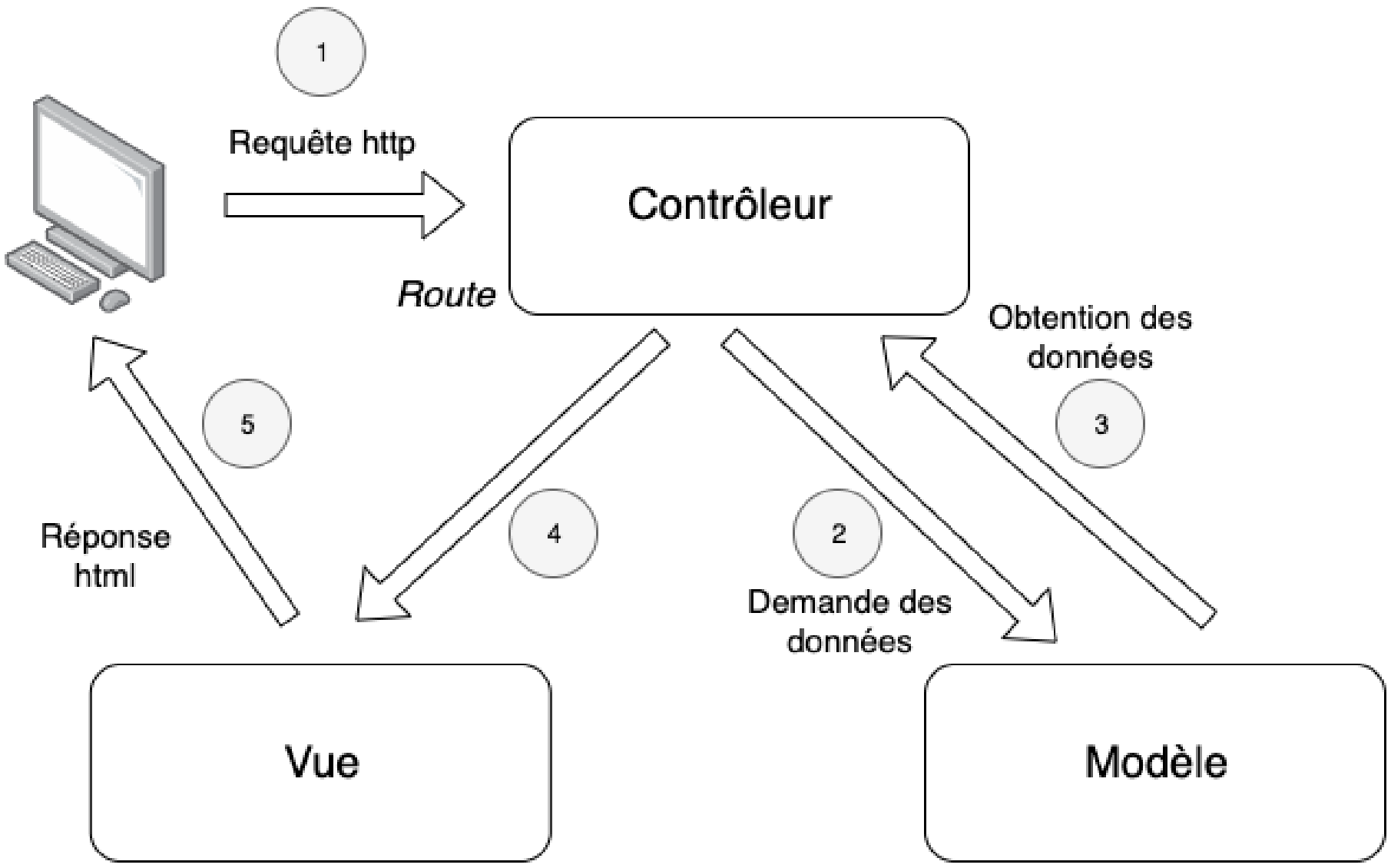
- Séparation des préoccupations : Le modèle MVC permet de séparer les préoccupations liées aux données, à la logique de présentation et à la gestion des demandes. Cela permet une meilleure organisation et une maintenance plus facile du code.

- Facilité de maintenance : Étant donné que chaque composant du modèle MVC est séparé, il est plus facile de mettre à jour ou de remplacer un composant sans affecter le reste de l'application.

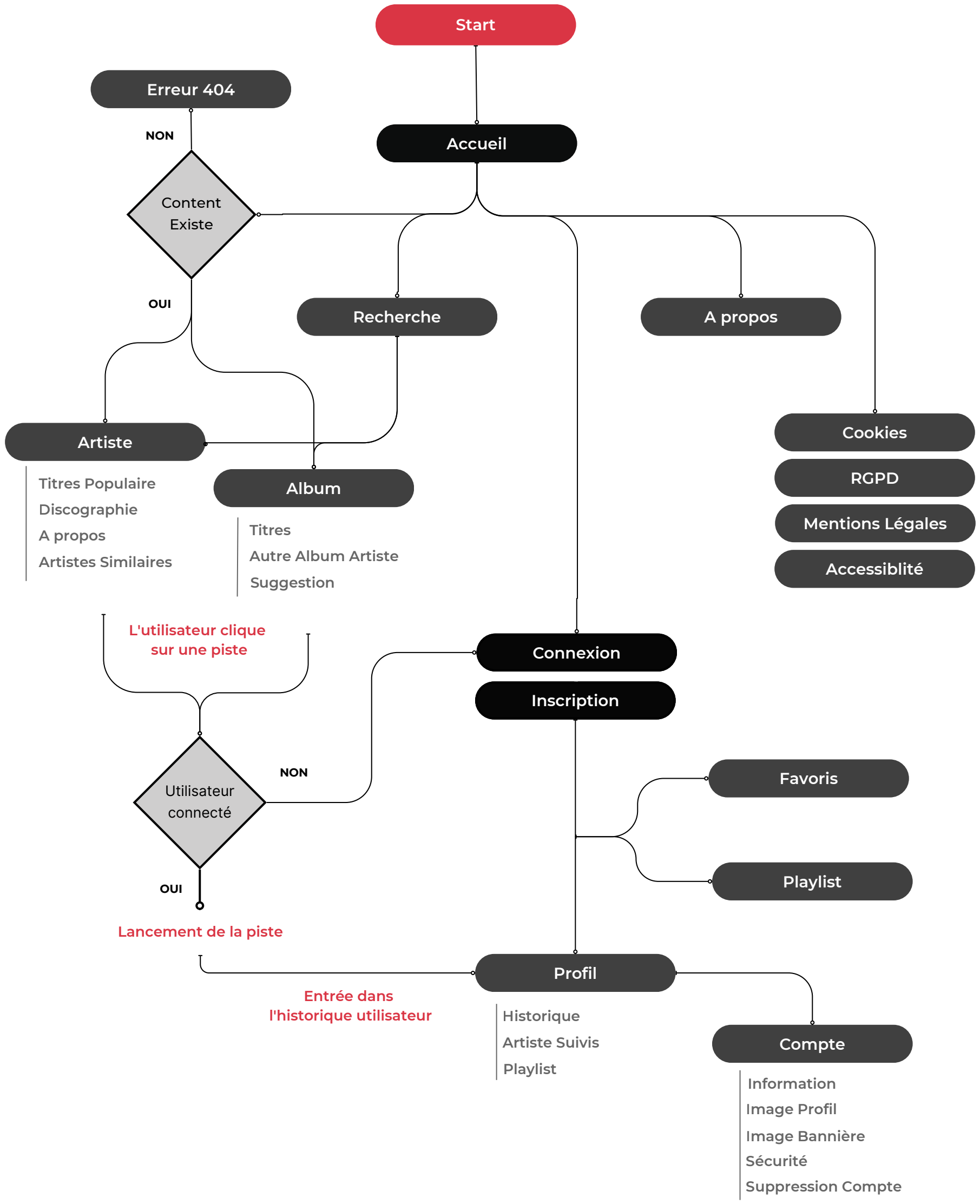
- Amélioration de la lisibilité du code : La structure claire du modèle MVC permet une meilleure lisibilité du code et facilite la compréhension des développeurs travaillant sur le projet.

- Support de multiples vues : Le modèle MVC permet de créer plusieurs vues pour la même logique de présentation, ce qui facilite l'adaptation de l'application aux différents types de périphériques et d'utilisateurs.

En somme, le modèle MVC est une architecture de conception efficace pour les sites web car elle offre une structure claire et facile à maintenir, permet une meilleure séparation des préoccupations.



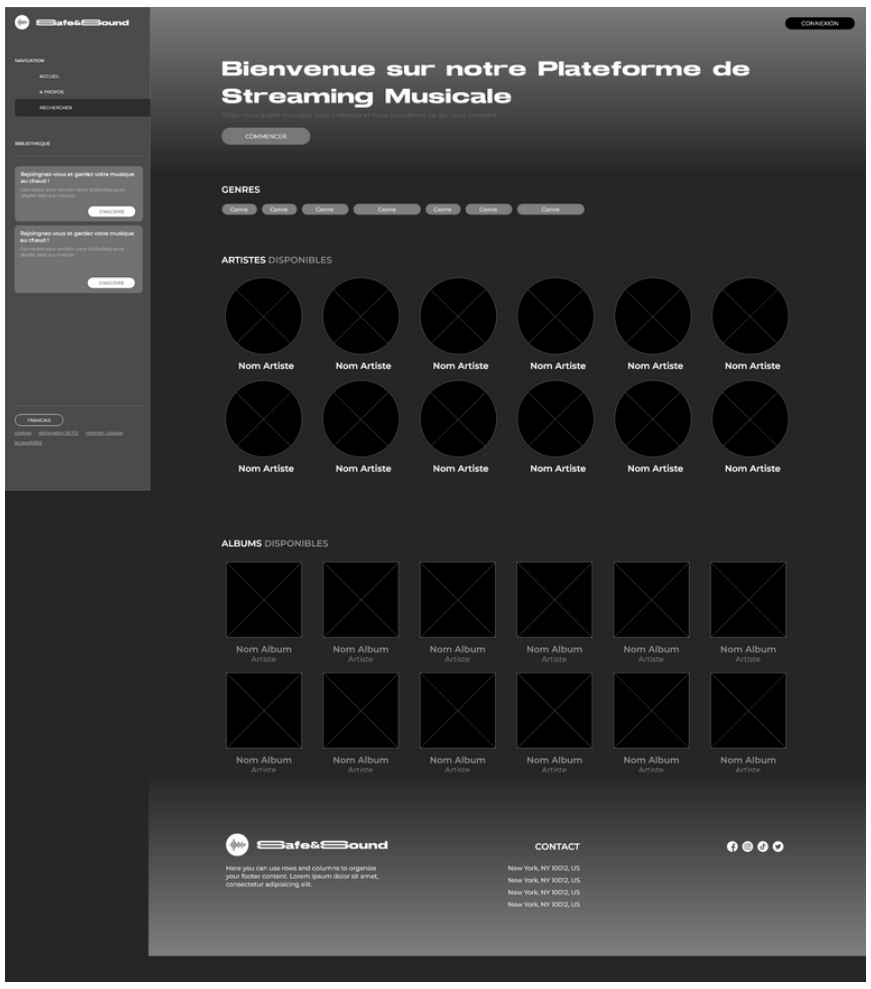
# ARBORESCENCE DU PROJET



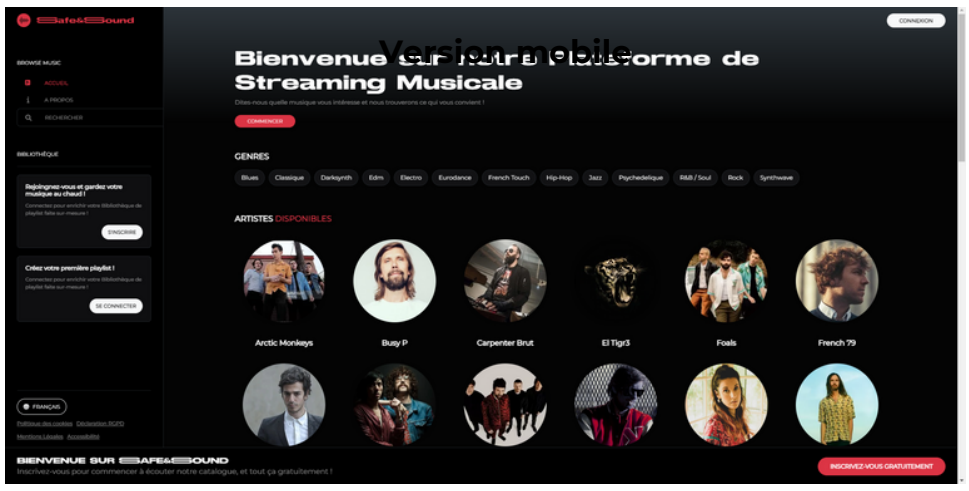
# RÉALISATION DU PROJET

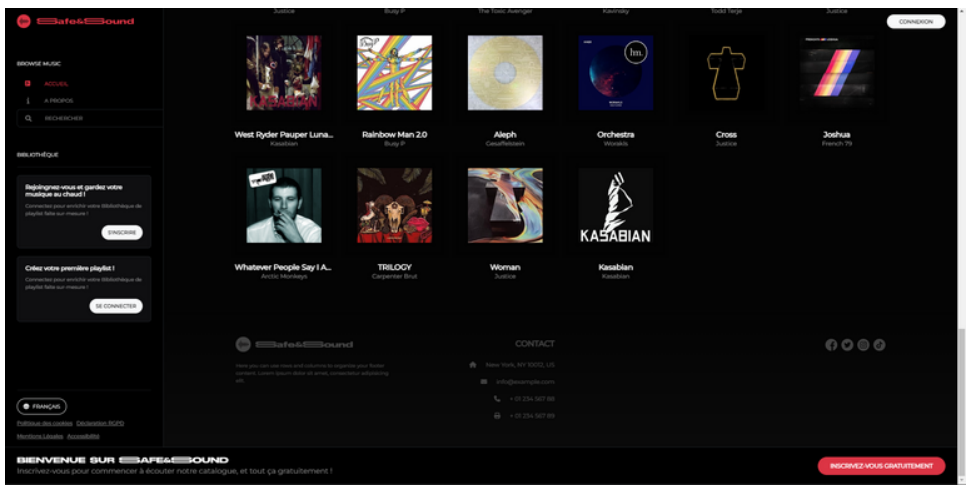
## MAQUETTAGE DU SITE - PC

### PAGE D'ACCUEIL

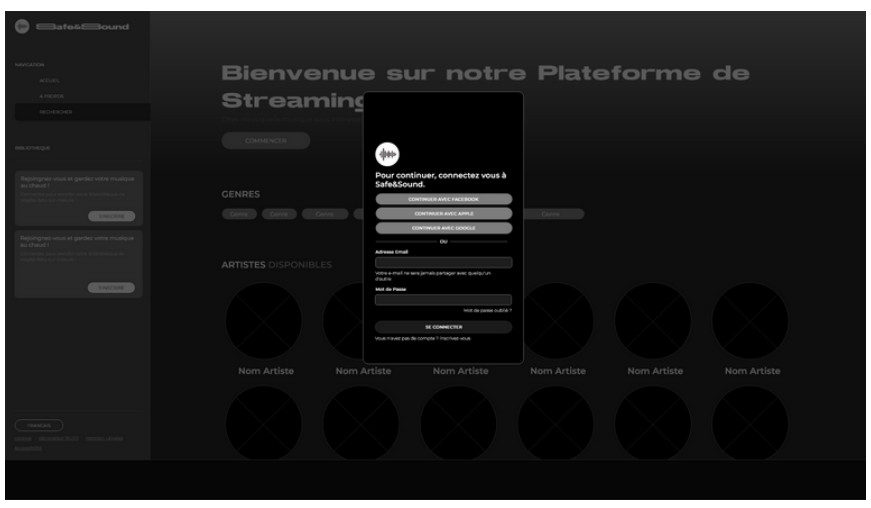


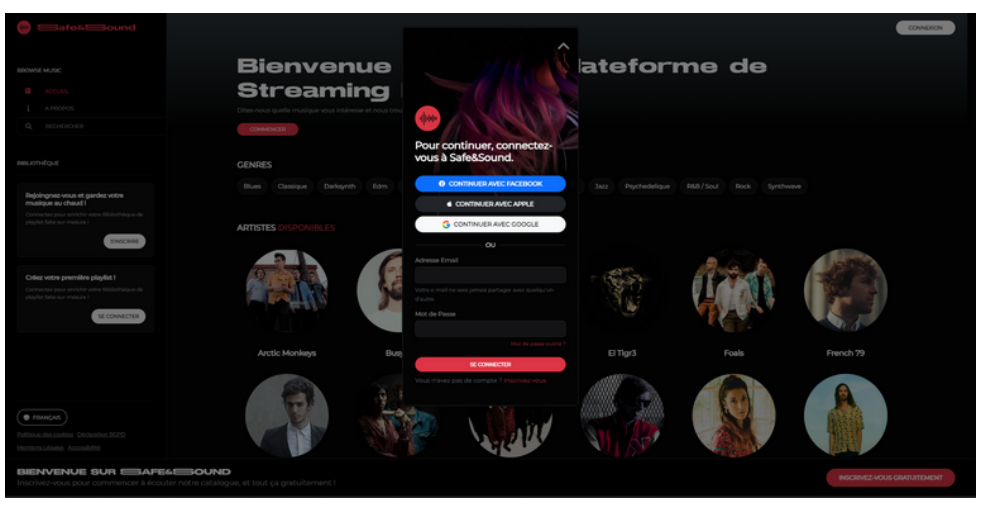

Version mobile



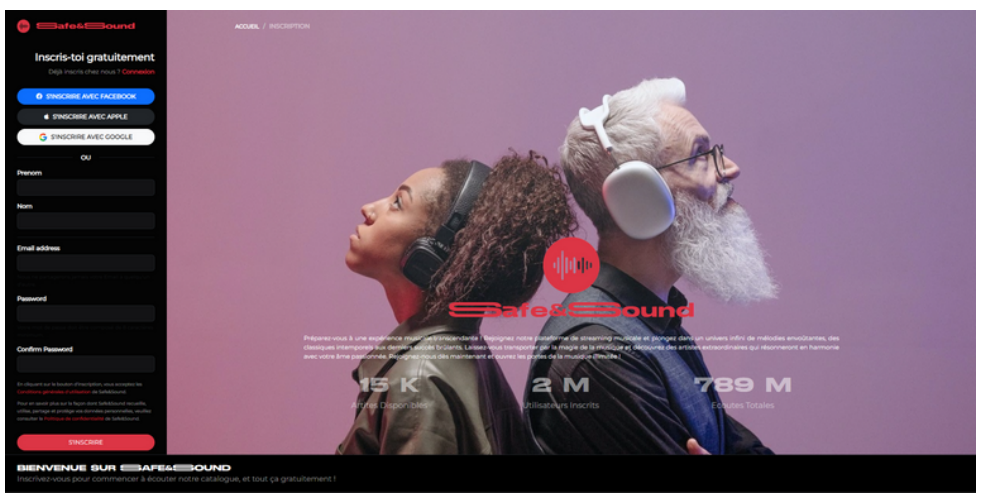




### MODAL CONNEXION

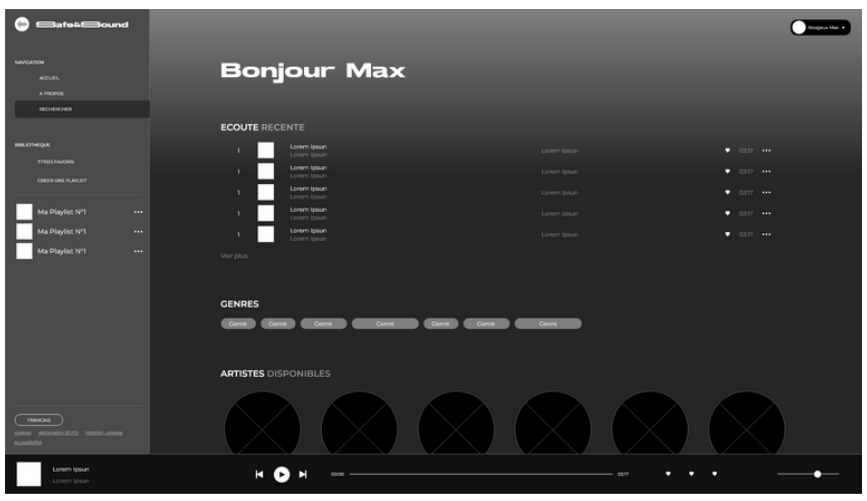


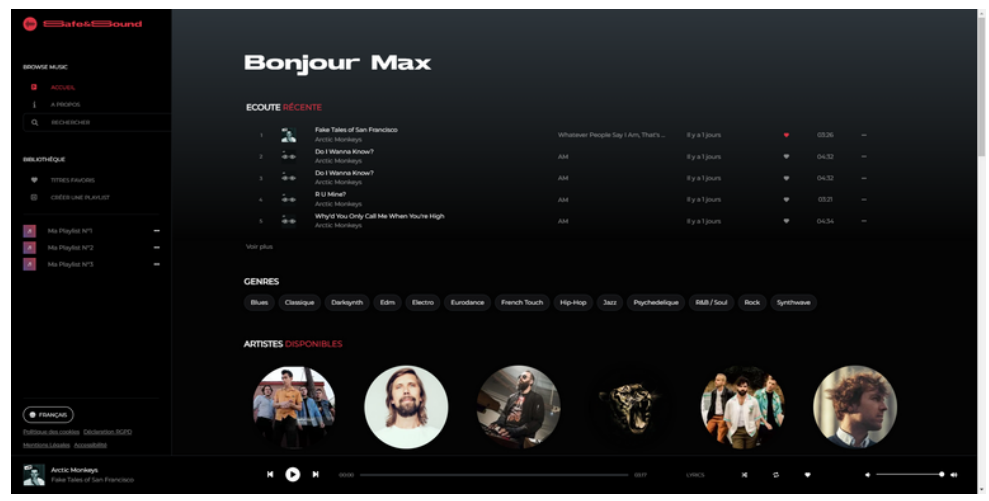




### PAGE D'INSCRIPTION

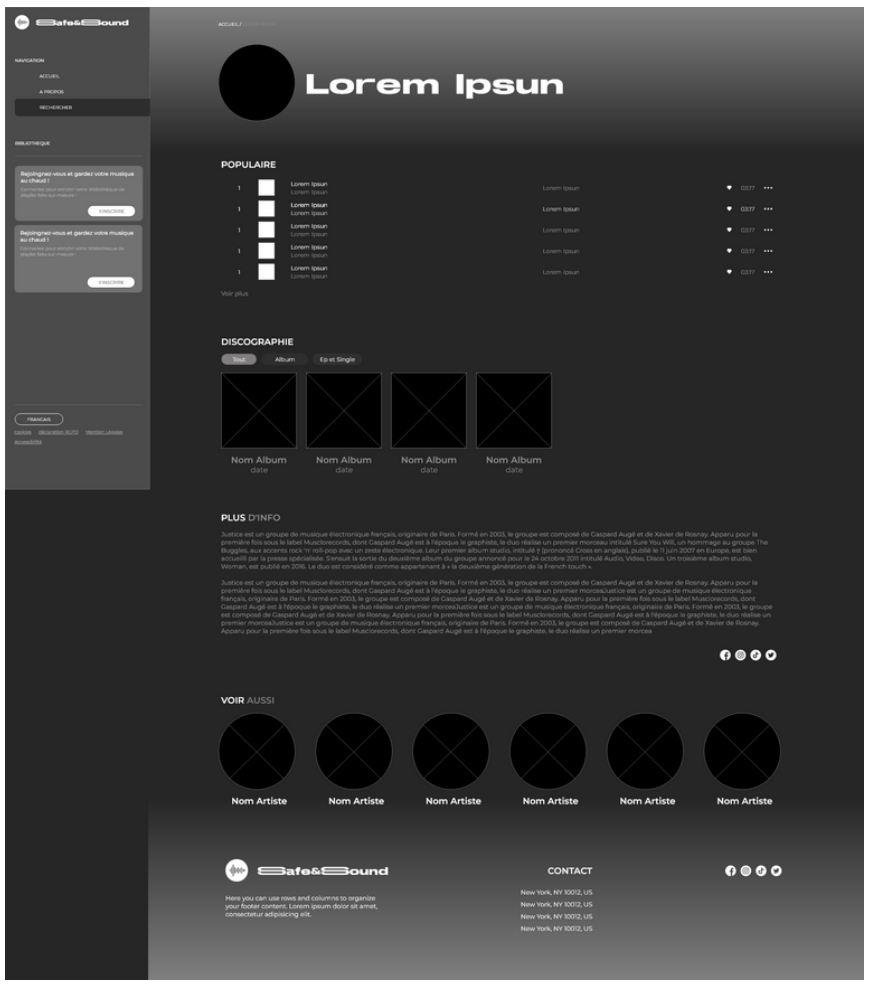



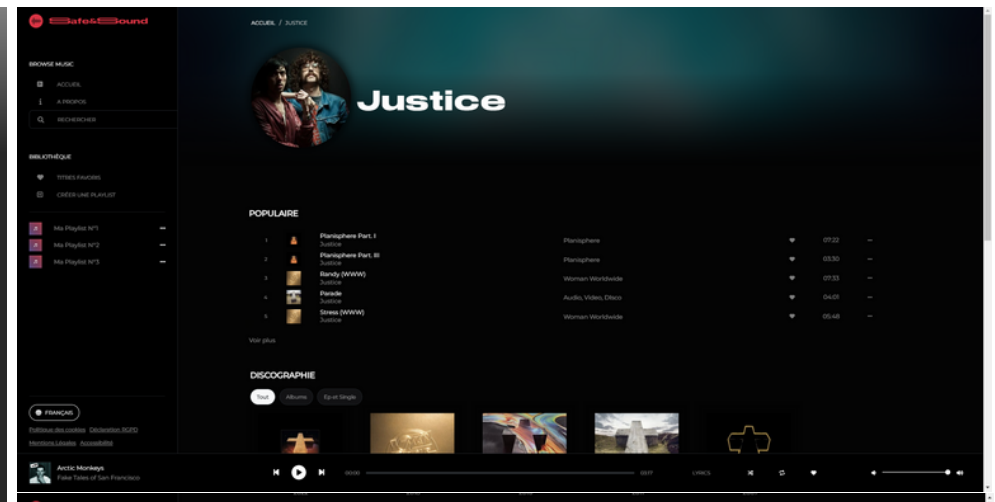
# ACCUEIL - UTILISATEUR CONNECTÉ

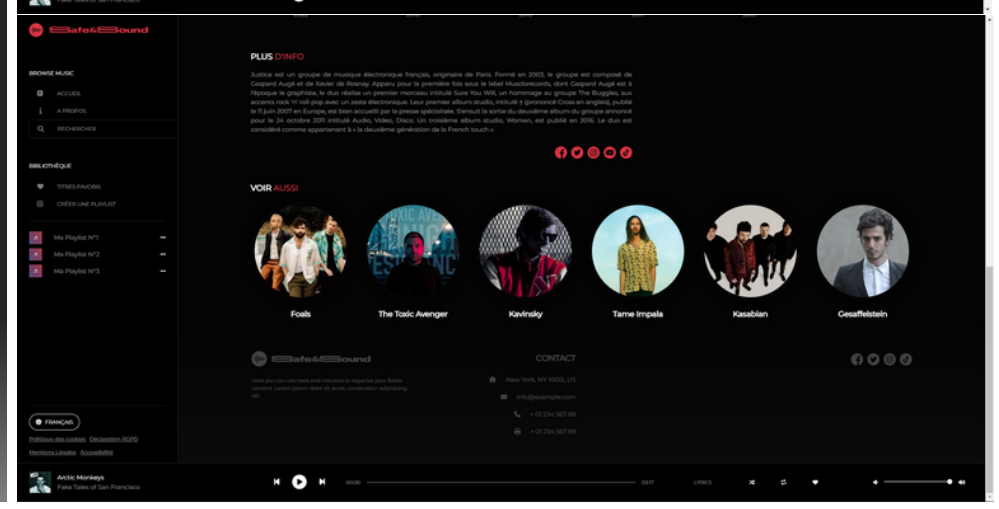


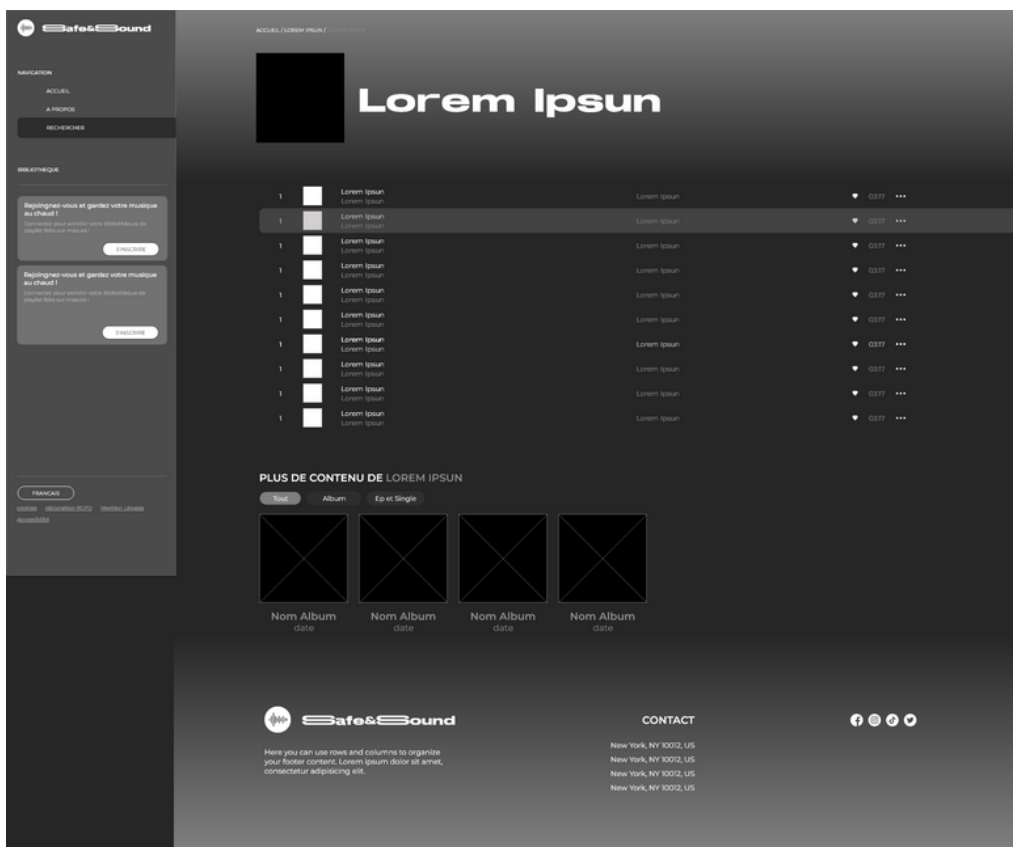

# PAGE ARTISTE



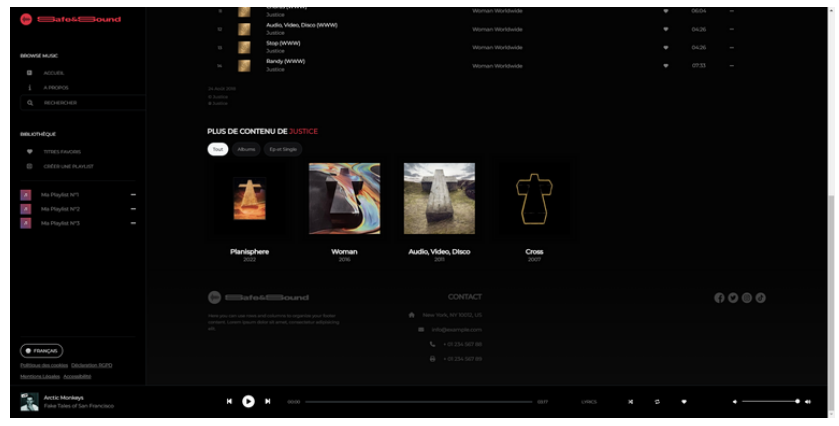




# PAGE ALBUM

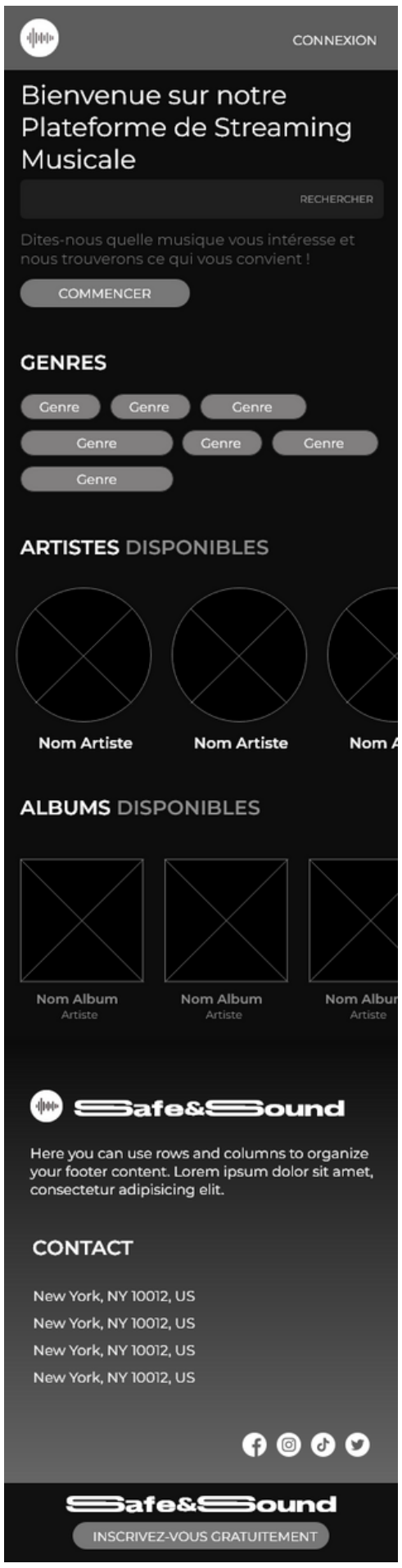


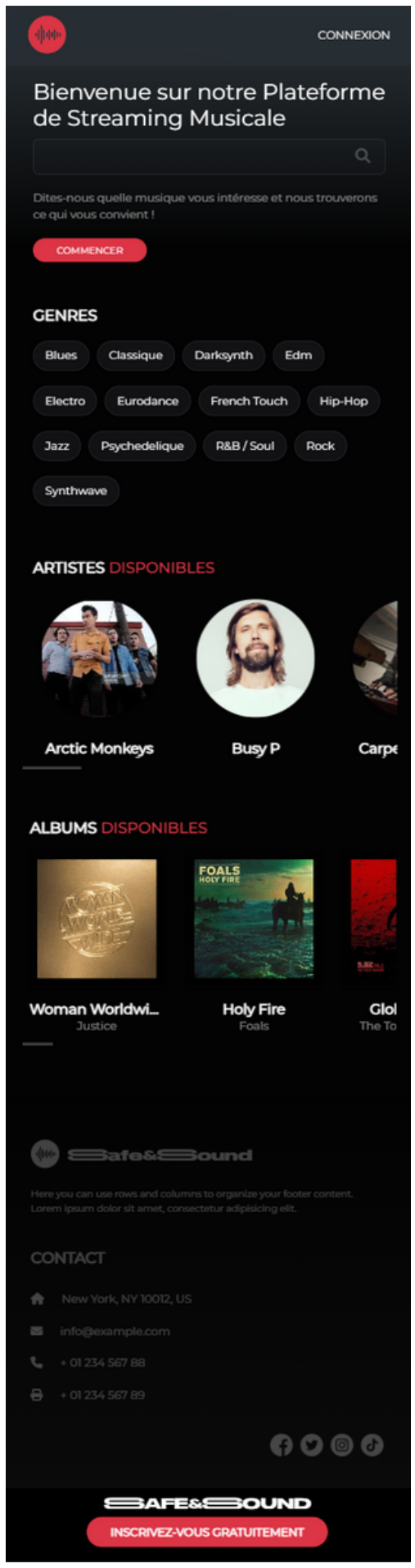




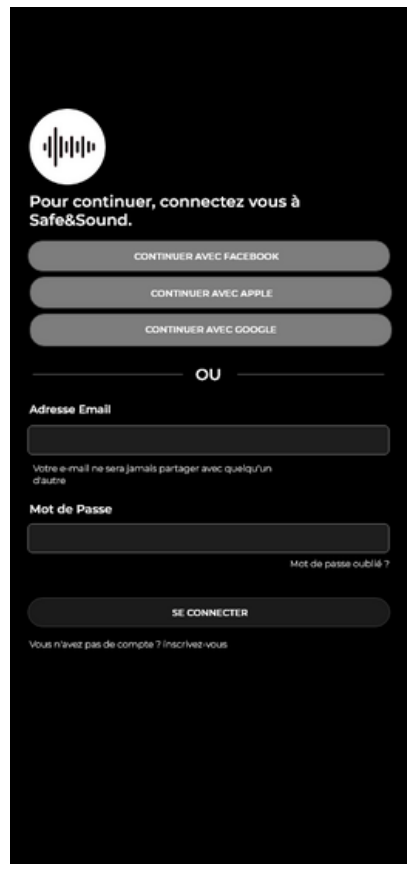
# MAQUETTAGE DU SITE - MOBILE

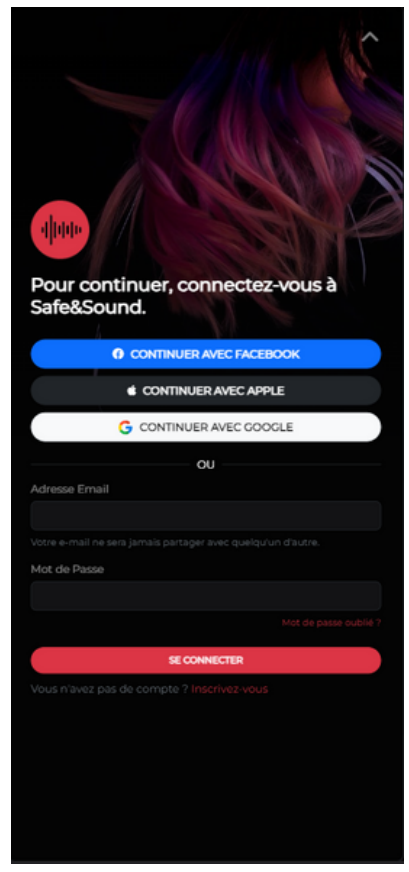
## PAGE D'ACCUEIL



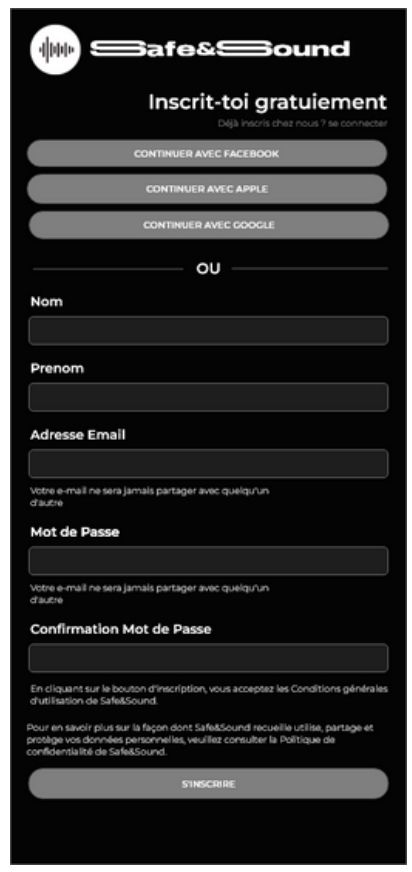



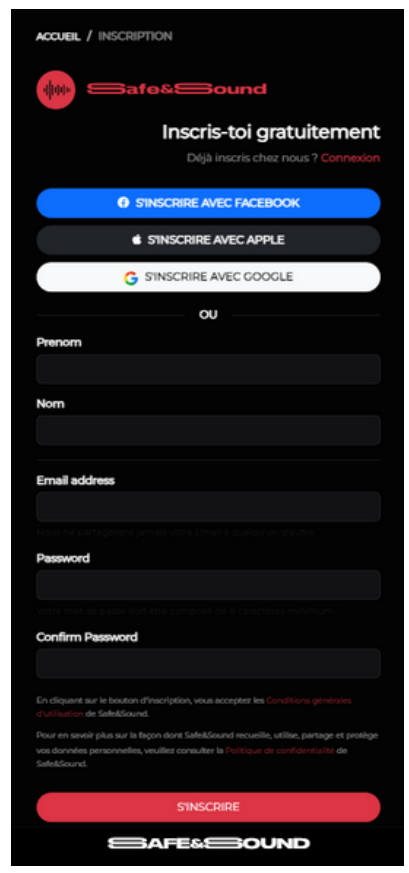

## MODAL CONNEXION



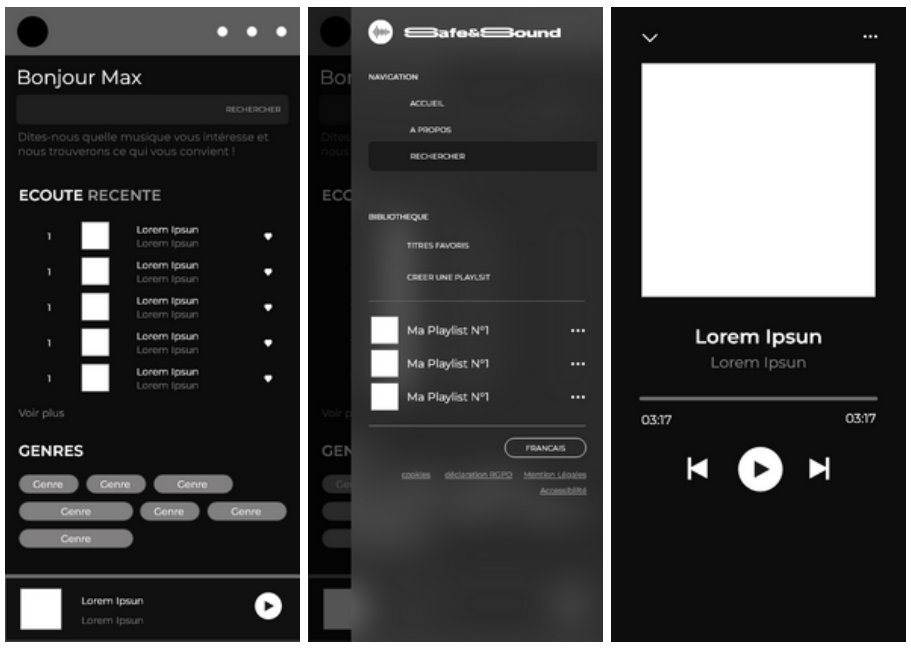



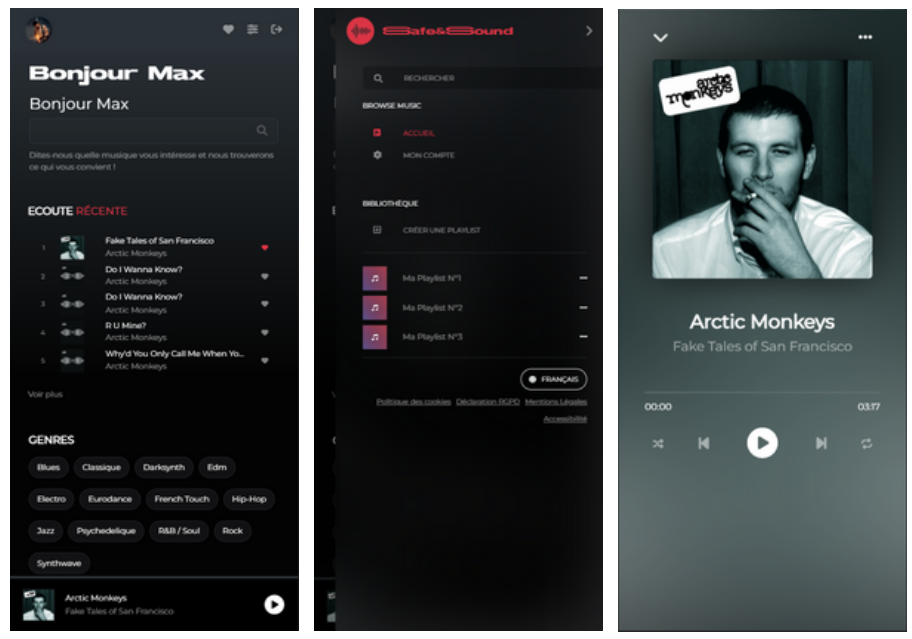

## PAGE D'INSCRIIPTION



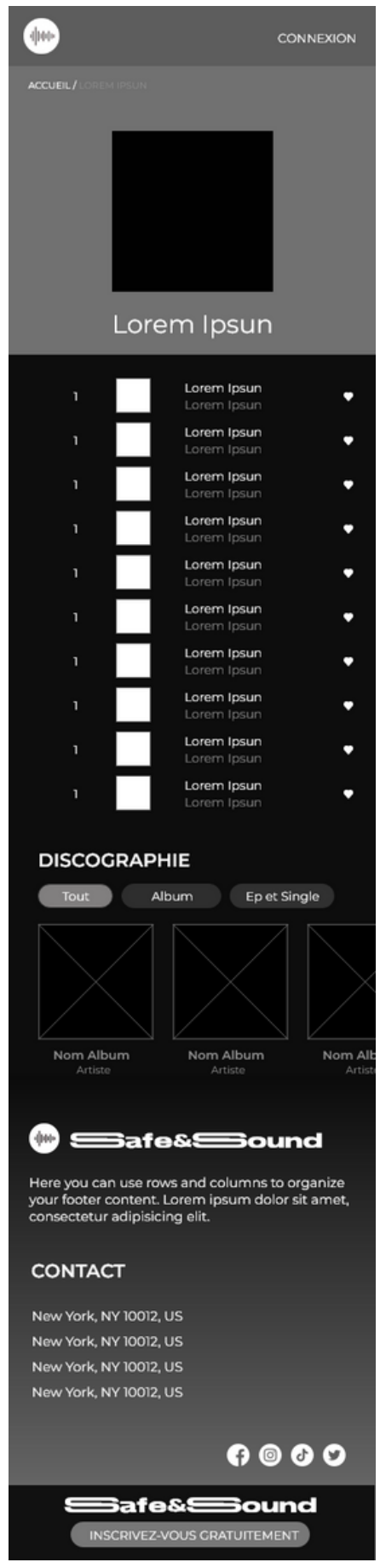
# UTILISATEUR CONNECTÉ - MENU / LECTEUR EN PLEIN ECRAN



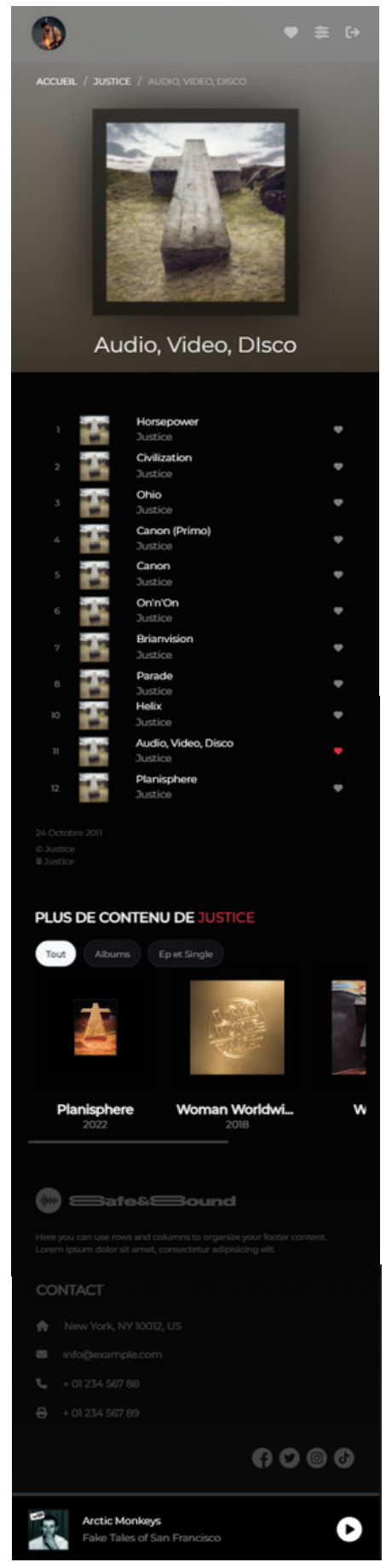




## PAGE ARTISTE






## PAGE ALBUM

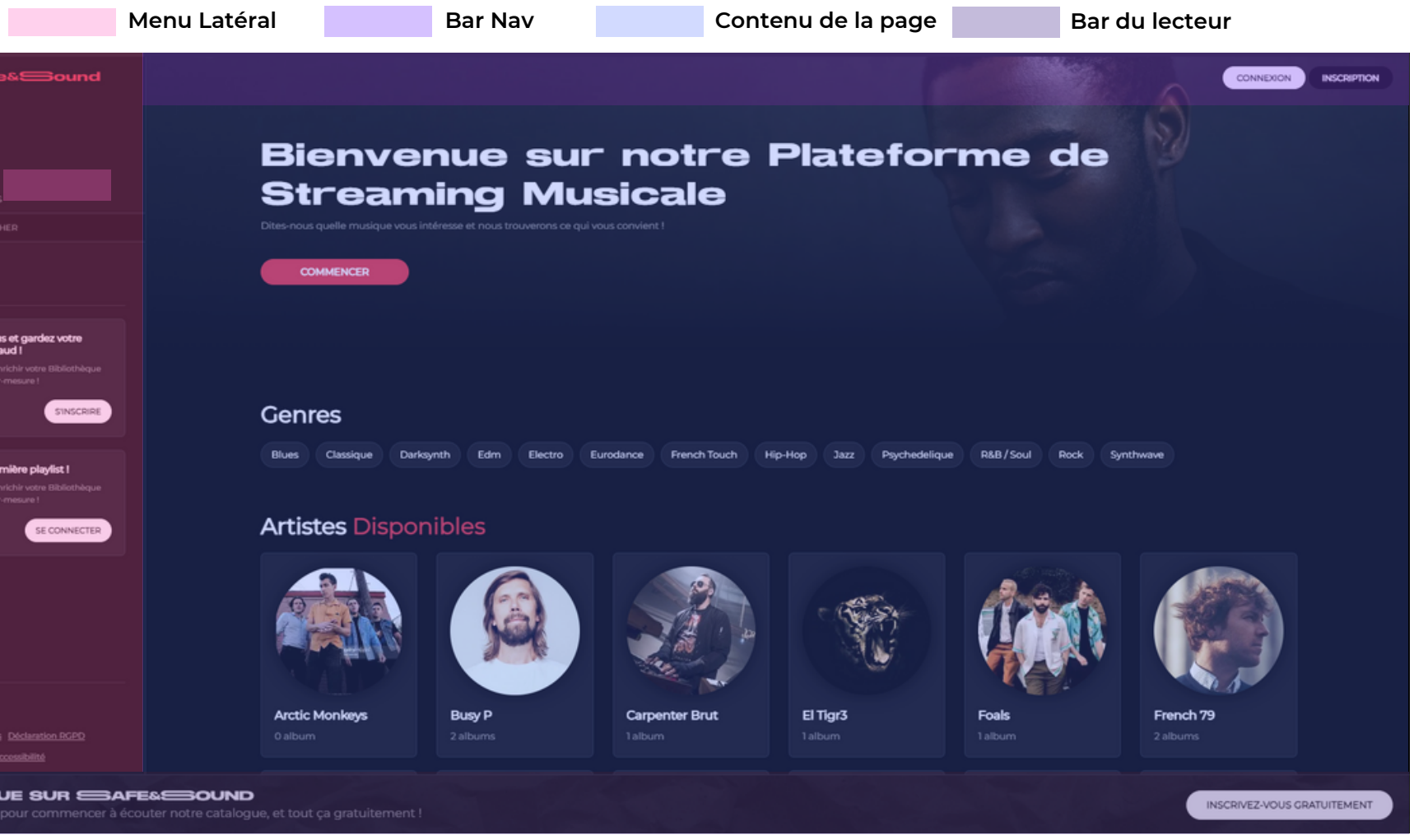


# RÉALISATION DE LA PARTIE FRONTEND

## PAGE D'ACCUEIL - VERSION PC

Suite au maquettage, j'ai commencé à développer le front end du site en commençant par les parties principale (la structure générale du site).

Grâce au système de colonne proposé par Bootstrap, j'ai commencé par préparé la futur disposition de mon site en plaçant un **élément DIV avec la classe "container-fluid"** suivi d'une **DIV avec la classe "row"** me permettant de travailler sur une largeur égale à l'écran de l'utilisateur :




- Le menu latéral sera placé à gauche avec une largeur définie par la classe "col-2". J'ai rajouté la classe "fixed-top" afin que les éléments situé dans ce menu soient toujours accessibles si l'utilisateur défile la page. Il l est composé d'une div.row avec la classe flex-column me permettant de le travailler sur l'axe verticale. La section Bibliothèque affiche des messages invitant l'utilisateur a se connecter ou s'inscrire s'il est déconnecter et affiche un lien vers les favoris et les playlist de l'utilisateur s'il est connecté.

- Le contenu principale ,où seront chargées les données, occupera une largeur définie par la classe "col-10" pour compléter la vue du site avec le menu latéral. L'ajout de la classe "offset-2" est nécessaire due à la position "fixed" du menu pour éviter que le contenu se retrouve sous ce dernier. Chaque section est sous la forme d'une div.col-10.offset-1 afin d'avoir un contenu bien centré sans prendre toute la largeur du conteneur.

- la barre de navigation et celle du lecteur auront respectivement la classe "fixed-top" et "fixed-bottom" afin d'être positionner en permanence en haut et en bas du site. Le lecteur, quand l'utilisateur est déconnecté, affiche un bandeau invitant l'utilisateur à s'inscrire. La barre de navigation affichera un menu dropdown avec les onglets : "compte et déconnexion" quand l'utilisateur est connecté et un bouton connexion et inscription quand il est déconnecté.

# PAGE D'ACCUEIL - VERSION MOBILE

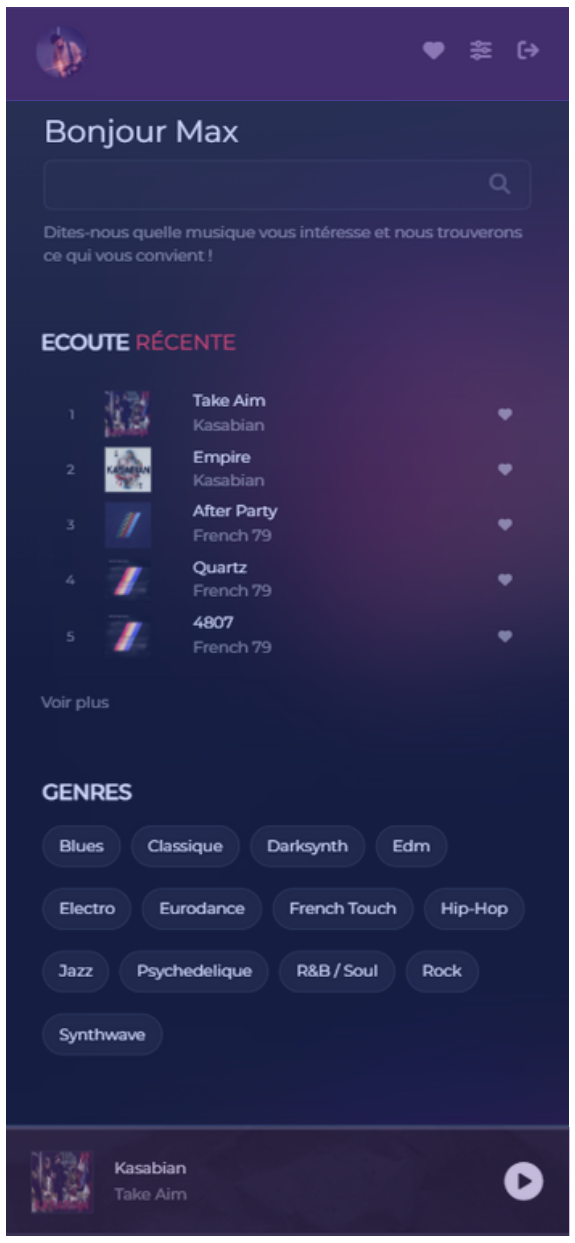
Pour la version mobile, tous les div principales repassent en " col-12 " via les breakpoint de Bootstrap "lg".

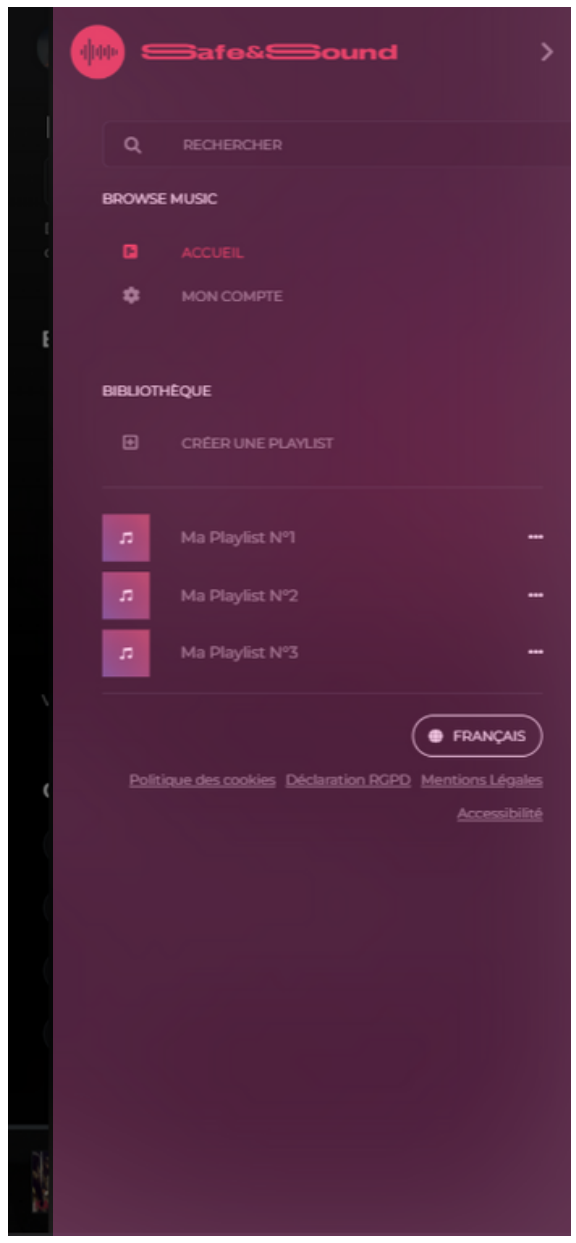
- Le menu latéral est changé en "offcanvas" Bootstrap, étant accessible via le deuxieme bouton de la nav.

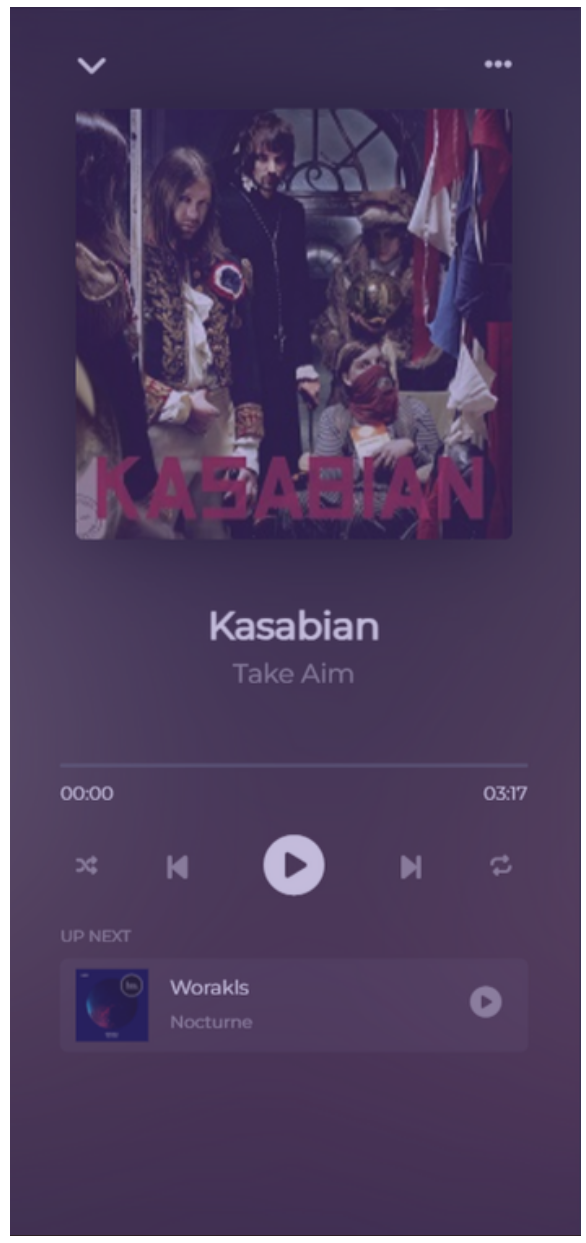
- Le lecteur se voit caché la plupart de son contenu :
  Etant donné la possibilité de modifier directement le volume via les boutons de son téléphone, le volume est caché.
  Un offcanvas venant du bas est mis à disposition en cliquant sur la piste en cours pour faire apparaitre un lecteur en plein écran offrant tout le contrôle disponible sur la version pc.

- Le menu dropdown de la barre de navigation est caché pour afficher directement un icône de déconnexion et un accès direct aux favoris.

**Menu Latéral** **Bar Nav** **Contenu de la page** **Bar du lecteur**



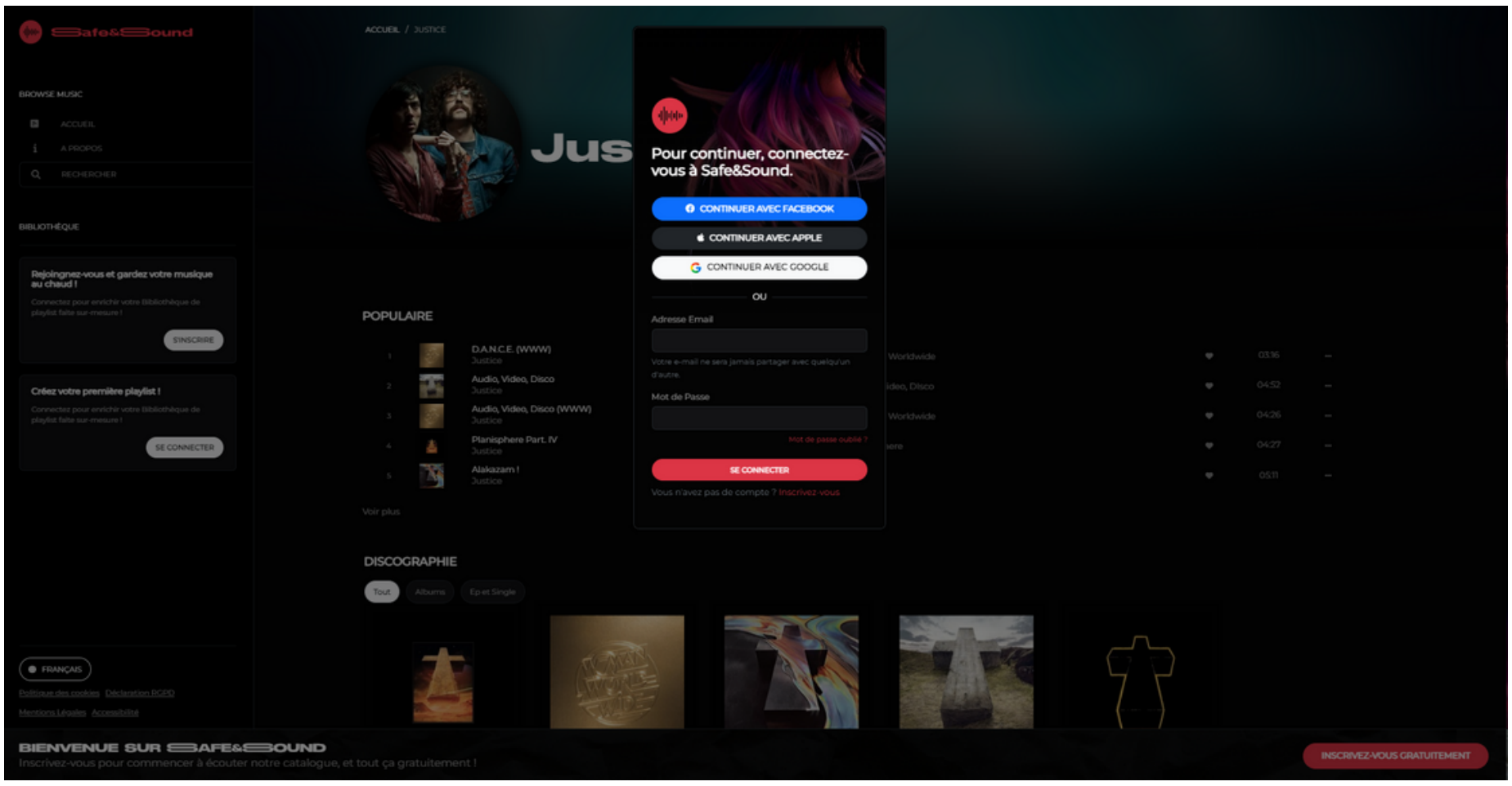



# FORMULAIRE DE CONNEXION / INSCRIPTION

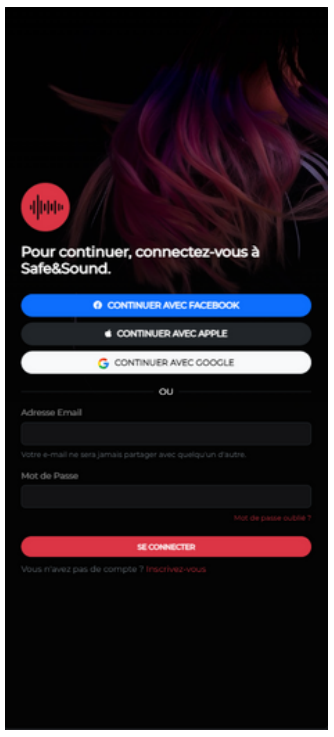
## FORMULAIRE DE CONNEXION

En version pc, le formulaire sera accessible sur tous le site via une modal bootstrap.

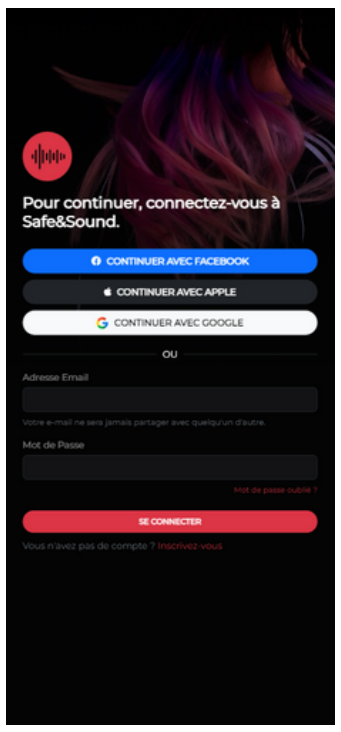



Pour un meilleur visuel en version mobile, la classe "modal-fullscreen-sm-down" permet à cette modal de couvrir l'intégralité de l'écran du téléphone. Afin d'avoir aucun soucis de dimension sur les écran de téléphone plus petit, la partie supérieur de la modal, contenant les trois bouton a été transformer en flex-column afin de remplacer le margin-top en version large par un margin-top: auto

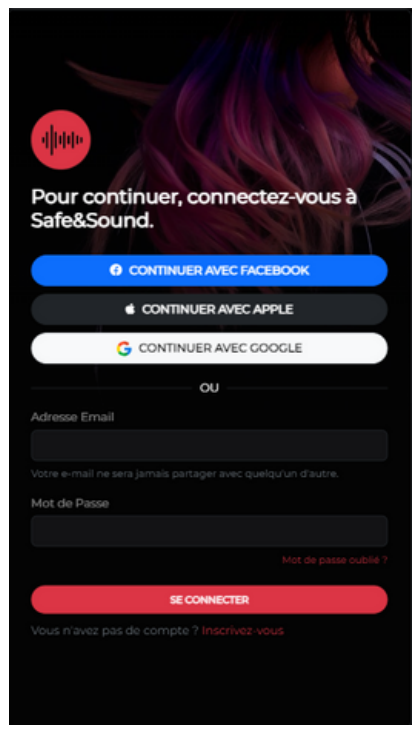
**Galaxy S20 Ultra**




**Iphone 12 pro**



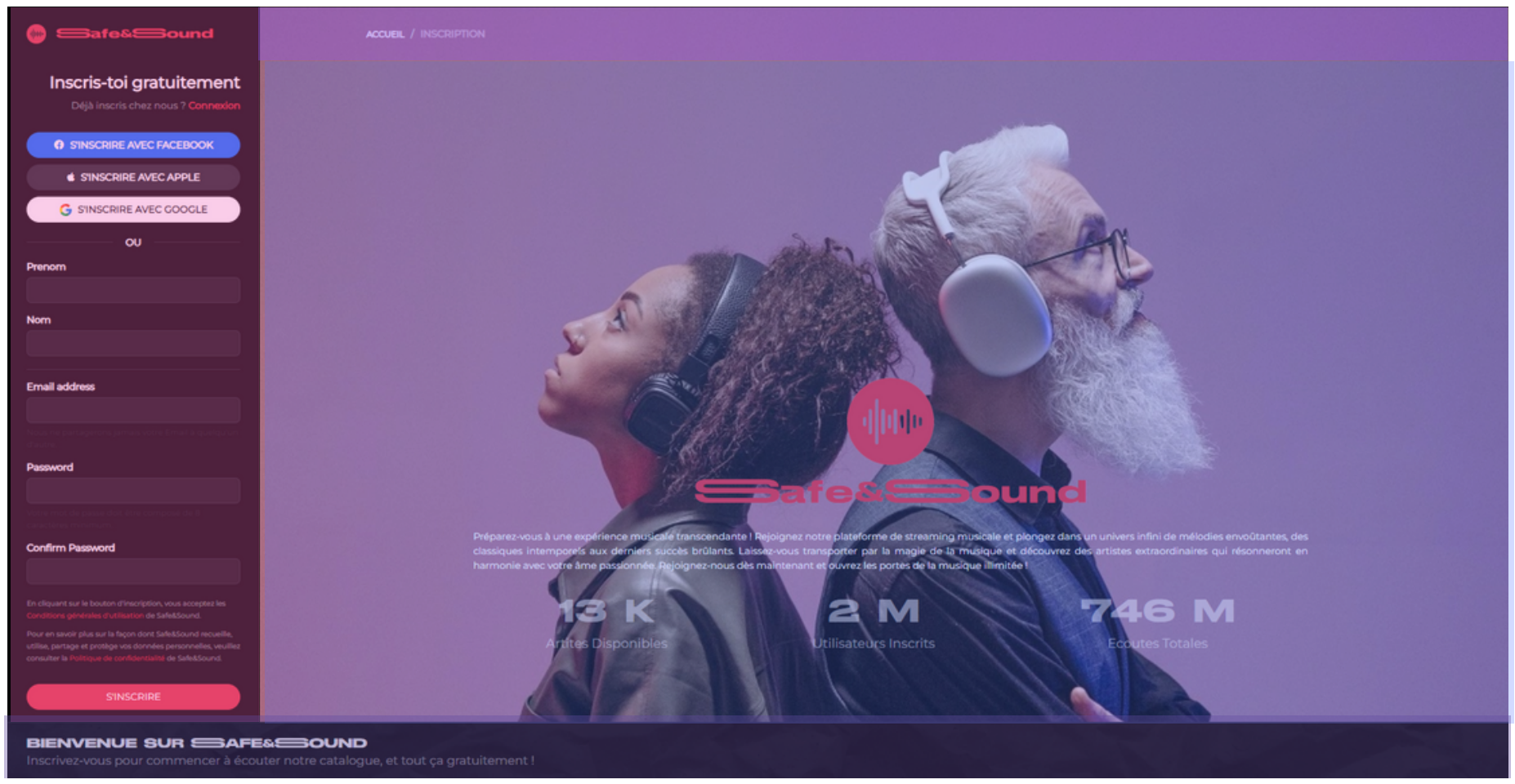

**Iphone SE 2016**




## FORMULAIRE D'INSCRIPTION

La page d'inscription reprend la structure générale du site, un menu latéral en col-2 contenant le formulaire. Cependant, cette fois, cest la div.col-10 contenant un carousel Bootstrap qui est en fixed-top. Cela permet de faire défiler le menu latéral sans bouger le carrousel.
Le carousel est un carousel crossfade, qui permet d'avoir des transitions en foudu entre les differentes images. Le même texte est conservé sur les trois différents diapos.

Formulaire Bar Nav Carousel Bar du lecteur



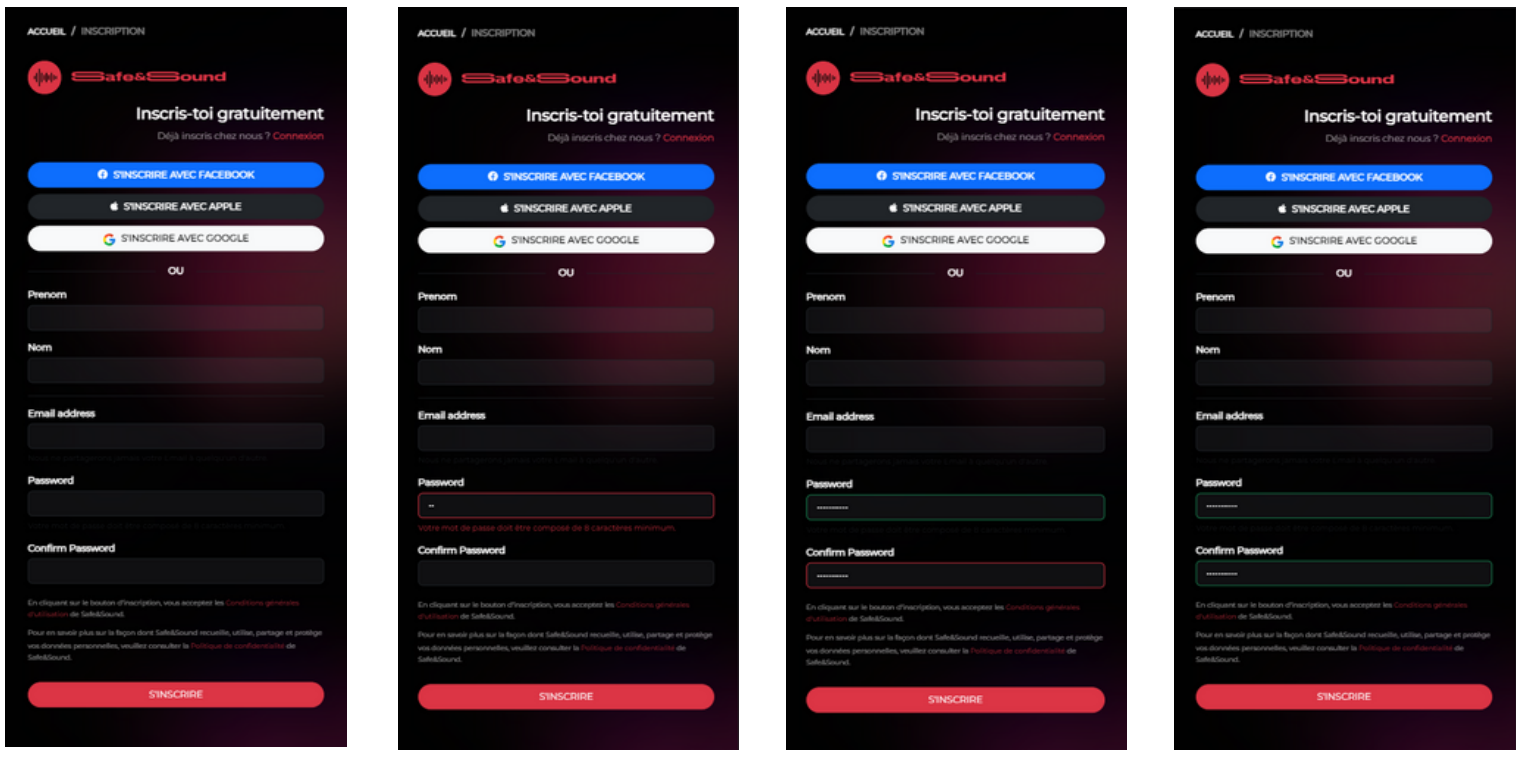

En version mobile, le carousel est caché et le menu, initialement en col-2, passe en col-12

Une fonction JS aide l'utilisateur pour le mot de passe :

```
$("#form-mdp").on("input", (() => {
    $("#form-mdp").val().length >= 8 ? ($("#form-mdpHelp").removeClass("text-danger"),
    $("#form-mdp").removeClass("border-danger").addClass("border-success")) : ($("#form-mdpHelp").addClass("text-danger"),
    $("#form-mdp").removeClass("border-dark").addClass("border-danger"))
})),
$("#form-mdpConfirm").on("input", (() => {
    $("#form-mdpConfirm").val() !== $("#form-mdp").val() ? $("#form-mdpConfirm").removeClass("border-dark").addClass("border-danger") : ($("#form-mdpConfirm").removeClass("border-danger")
    $("#form-mdp").val().length >= 8 && $(".form-submit").removeClass("disable"))
})),
$(".form-submit").on("click", (e => {
    $(".form-submit").hasClass("disable") && e.preventDefault()
}))
```

A chaque saisie dans l'input:text correspondant au mot de passe, le JS va ajouté des bordures rouges tant que le la longueur du mot de passe en inférieur à 8 (tant que value.length <= 8).
Pour la confirmation du mot de passe, à chaque saisie, le JS va comparer la valeur de l'input#mdpConfirm avec celle de l'input#mdp. Tant qu'elles sont différentes, une bordure rouge sera ajouté à l'input de confirmation et l'utlisateur ne pourra pas valider le formulaire d'inscription.
Si toute les conditions sont remplies, les inputs de mot de passe se verront attribuer une bordure verte et l'utilisateur pourra s'inscrire.

**Version mobile** | **mot de passe trop petit** | **confirmation invalide** | **mots de passes valides**



# ANIMATION JS DES ICÔNES

Pour une meilleure lisibilité, certaines interactions avec le site ont nécessitées une animations sur les icônes correspondant. Les icônes pour la mise en favoris d'un titre et les modes aléatoires / répété du lecteur sont liés à la même fonction js s'occupant de l'animation lors du clique sur ces derniers.

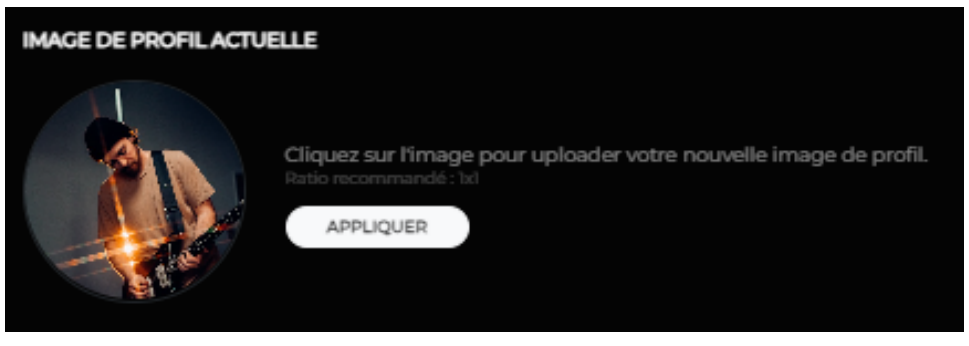
```
function animIconOn(e) {
    e.css("transform", "scale(0)"), setTimeout((() => {
        e.addClass("text-danger"), e.css("transform", "scale(1.3)")
    }), "200"), setTimeout((() => {
        e.css("transform", "scale(1)")
    }), "300"), e.addClass("active")
}
function animIconOff(e) {
    e.css("transform", "scale(0)"), setTimeout((() => {
        e.removeClass("text-danger"), e.css("transform", "scale(1.3)")
    }), "200"), setTimeout((() => {
        e.css("transform", "scale(1)")
    }), "300"), e.removeClass("active")
}
```

Chaque fonction s'execute de la même façon en plusieurs étapes :

- sur les première 0.2 secondes, l'icône voit sa taille diminuer jusqu'à disparaitre.
- En étant invisible, sa couleur change.
- Enfin sur les 0.3 secondes suivantes, l'icone réapparait en grossissant un peu plus que ça taille d'origine et rétréci.


# MISE EN FORME DU FORMULAIRE DE CHANGEMENT D'IMAGE

Sur la page de gestion du compte utilisateur, un formulaire de changement d'image de profil est disponible sous forme d'une image cliquable.

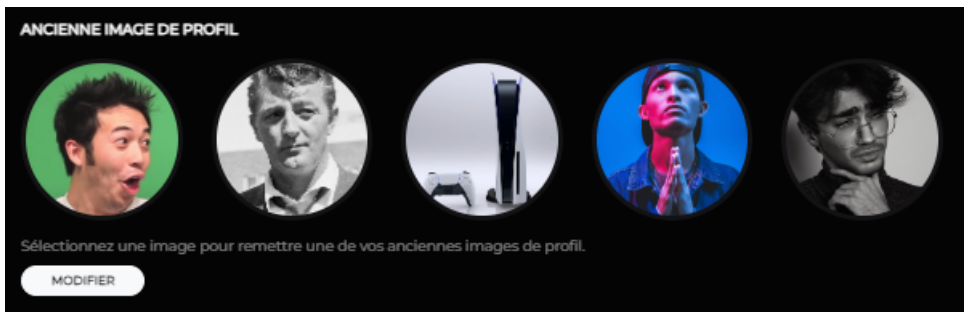



L'input:file en question est invisible avec un display:none et nommé par une id.
Le label, contenant l'image, est lié à l'input:name via l'id de ce dernier et l'attribut "for" du label. Un clique sur l'image, sur le label par extension, déclenchera l'ouverture de la fenêtre d'upload de fichier de l'input:file.

Un événement JS sur la modification de la valeur de l'input:file permet à l'utilisateur de visualisé l'image uploadée avant de valider la modification.

```
$("#user-newBanner").change((function () {
    let e = $("#user-newBanner").get(0).files[0];
    e && $("#previewUserBanner").attr("src", URL.createObjectURL(e))
}));
```

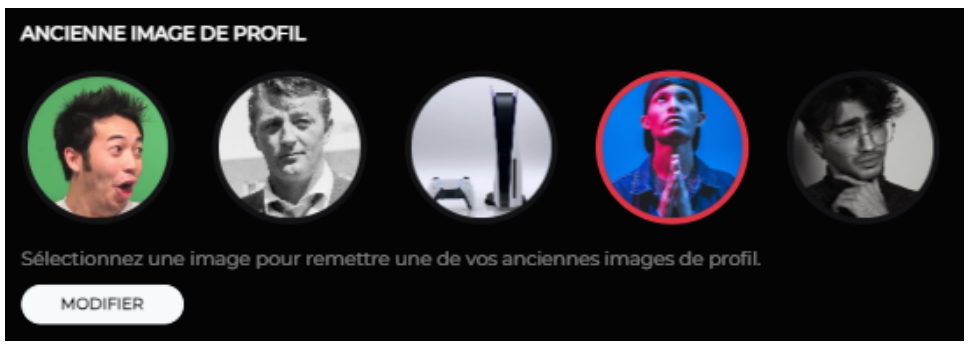
Pour la gestion des anciennes photos utilisées par l'utilisateur, même principe utilisé précédemment, mise à part que les inputs utilisés sont des checkbox.




Chaque image correspond à un label lié à un input:checkbox.

Pour mettre en évidence l'image sélectionnée, étant donné que les input:checkbox sont cachés, un événement onclick s'occupe d'ajouter des bordures rouges sur l'image liée au checkbox sélectionné. Chaque div contenant le label / image et l'input est doté d'un onclick avec la fonction ci-dessous et d'un nombre unique en paramètre.

```
function imgRadio(e) {
    $(".img-check label").each(((e, t) => {
        $(t).removeClass("bg-danger")
    })), $(".img-check label[for=radioImg" + e + "]").addClass("bg-danger")
}
```




une fois la fonction éxecutée, le js parcours une fois toute les images et enlève les bordures rouges si une ancienne image été sélectionnée et viens les ajouter sur l'image cliquée.

# LECTEUR AUDIO

## PRÉPARATION DE LA PLAYLIST

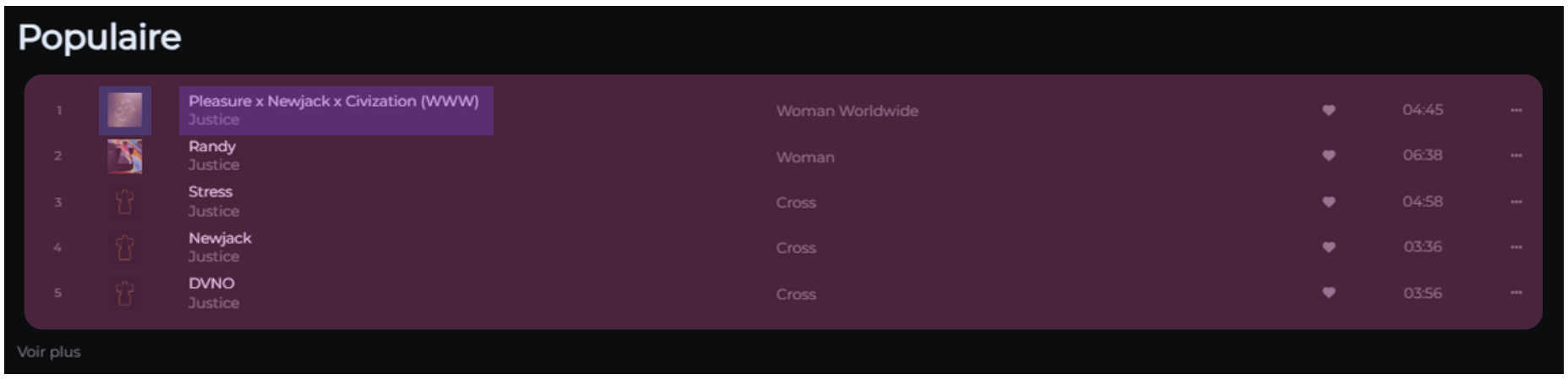
Afin d'avoir un lancement de musique sans problème sur chaque page du site, les pistes audio et les playlists sont construites de manière uniforme.

Toutes pistes audio **div.track** est contenues dans une **div.album** qui permet au navigateur (via JS) de récupérer tous ce que le lecteur à besoin pour fonctionner via des classes pour chaques informations de la musique :

- **div.track-src = source audio** (contenu caché = display : none)
- **div.track-name = nom de la musique**
- **div.track-artist = artiste de la musique**
- **div.track-cover= pochette de l'album correspondant**

**exemple :**

div.album  div.track-cover  div.track-name/track-artist




Lors du chargement de la page, le code suivant s'execute :

- **Avec une classe "Song" créée en amont**

```
class Song{ // Creation de la classe Song
    constructor(nom, artist, album, source) {
        this.nom = nom;
        this.artist = artist;
        this.album = album
        this.source = source;
    }
};
```

- **Une double boucle se lance parcourant la page pour créer la playlist :**

```
const track = [];

$('.album').each((i, x) => {
    const album = [];

    $(x).children('.track').each((j, y) => {
        let song = new Song(
            $(y).find('.track-name').html(),
            $(y).find('.track-artist').html(),
            $(y).find('.track-cover').attr('src'),
            $(y).find('.track-src').html());
        album.push(song);
    });

    track.push(album);
})
```

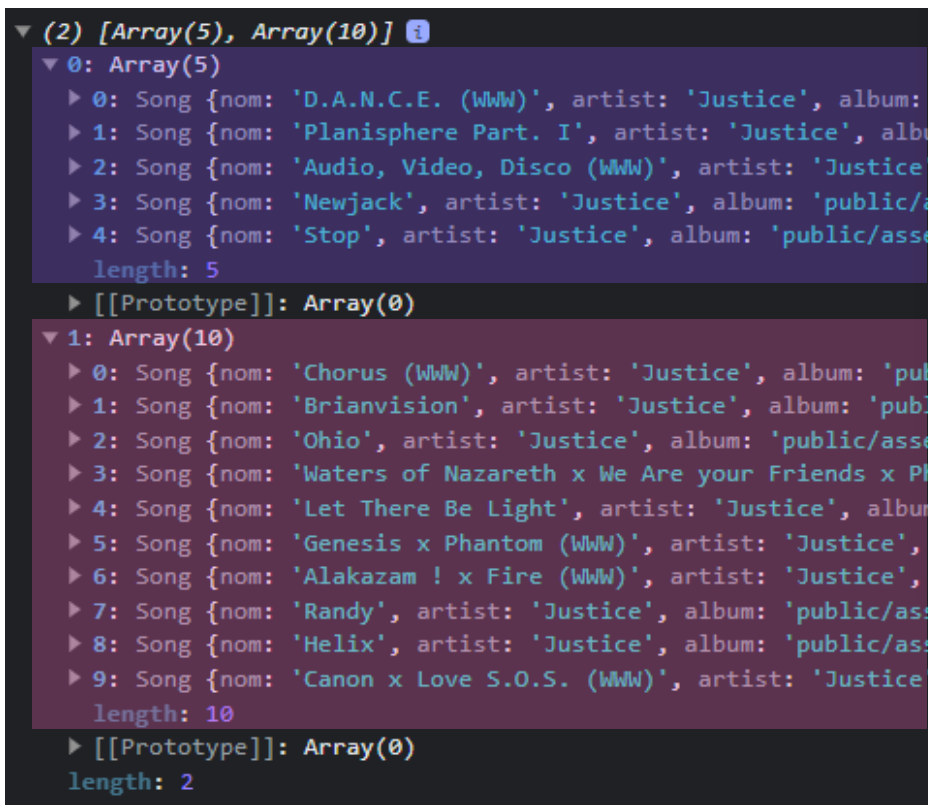
Je prepare un **tableau vide "track"**.

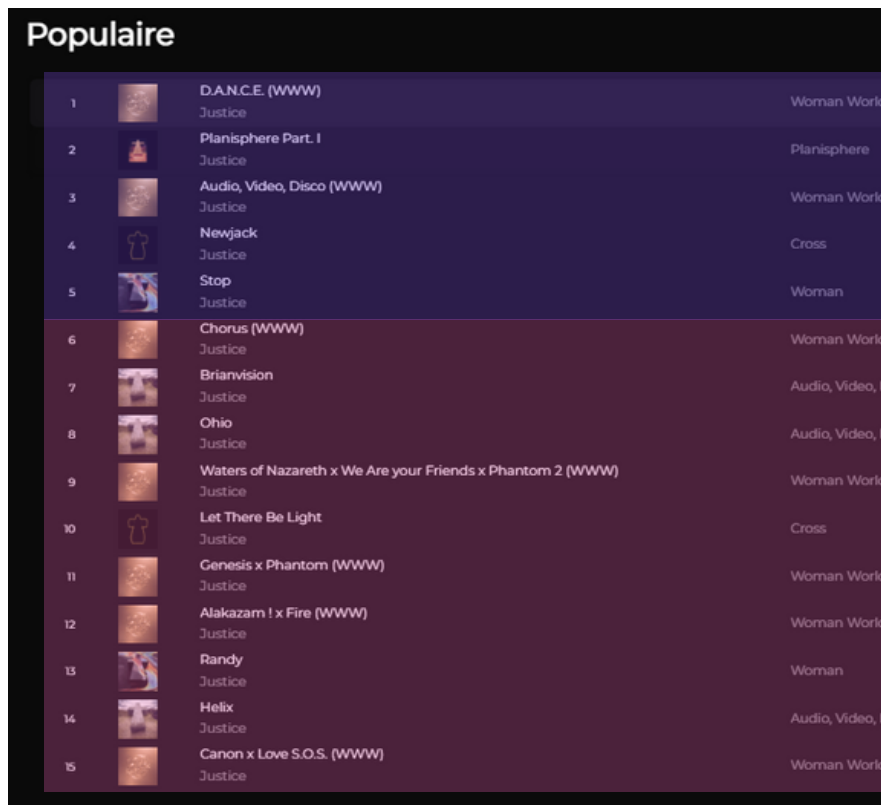
La première boucle se lance pour chaque **div.album**.

- A chaque **itération i**, je crée un **tableau "album"**.
- je lance une deuxième boucle  qui va parcourir chaque **div.track** présent dans l'élément **div.album** correspondant à l'**itération x** de la première boucle :
  - A chaque **itération j**, une **variable "song"**. **associé à la classe objet "Song"** récupère les informations nécessaire au lancement de la musique correspondant à l'élément **div.track y**.
  - Une fois les informations stockées dans la variable "song", j'associe cette dernière au tableau "album" créé précédemment.
- Enfin, j'associe de la tableau "album" avec le tableau "track" créé avant les boucles.

.

**J'obtient alors un tableau de tableaux correspondant à la structure de mon html :**




dans cet exemple, le deuxième élément **div.album** correspond à une **collapse de Bootstrap**.
A l'arrivé sur la page d'un artiste, l'utilisateur se voit proposé **5 titres** de l'artiste. **Un lien "voir plus"** lui permet d'afficher **10 titres supplémentaire qui apparaissent après le click sur ce dernier**.

Avec ce résultat, chaque piste audio  a **2 coordonnées** correspondant aux **index du tableau associatif** :

- La piste "D.A.N.C.E (WWW)" de Justice correspond aux coordonnées 0 ; 0
- La piste "Genesis x Phantom (WWW)" de Justice correspond aux coordonnées 1 ; 5

## LANCEMENT D'UNE PISTE / CONTRÔLE DU LECTEUR

Tout d'abord, pour permettre au navigateur d'émettre du son, j'utilise une instance de l'objet Audio (interface intégrée au navigateur qui permet de créer et de contrôler des éléments audio dans une page web).

```
const audio = new Audio();
```

### Volume

Afin de gérer le volume du site, je lie la propriété audio.volume à un input:range dont je défini les valeurs entre 0 et 0.5 (le volume natif de l'object audio étant compris entre 0 et 1).

```
<input type="range" class="w-100 " min="0" max="0.5" step="0.005" value="0.25" id="volume">
```

A l'arrivée sur le site je force l'audio à prendre la valeur (0.25) définit en amont pour ne pas avoir un volume trop fort pour l'utilisateur au lancement de sa première musique.

```
$(window).on('load', () => {
    audio.volume = $('#volume').val();
});
```

Enfin, sur l'action de l'utilisateur via l'input:range du lecteur, une modification de la valeur de ce dernier affectera instanément le volume du navigateur.

```
$('#volume').on('input', () => { // Volume du player
    audio.volume = $('#volume').val();
})
```

### Fonction

Je prépare ensuite plusieurs fonctions JS permettant le chargement d'une piste audio, la lecture etc..., tout ce que l'utilisateur pourrait avoir besoin lors de son écoute sur le site.

```
function play() {
    audio.play();
    isPlaying = true;

    $('#play-btn').addClass('fa-circle-pause').removeClass('fa-circle-play');
}
function preview(e) { // affichage des infos de la piste en cours sur le lecteur
    $('#CurrentSongArtist').html(e.artist);
    $('#CurrentSongName').html(e.nom);
    $('#CurrentSongImg').attr(
        'src', e.album
    )
}
function loadSong(e) { // chargement de la piste sélectionnée
    currentSong = e.source;
    preview(e);
    audio.src = currentSong;
    audio.load();
}
function active(e) { // affichage en surbrillance de la piste en cours
    $('.album').children().removeClass('active');

    $(e).addClass('active');
}
```

**La fonction play()** permet de lancer l'audio de navigateur. Cette fonction sera exécutée apres la fonction loadSong().
Elle permet également de changer l'icone du lecteur au moment de la lecture.

**La fonction preview()** permettra de charger les informations stockés dans l'objet Song créer au plus tôt sur lecteur pour permettre à l'utilisateur de savoir quelle piste est actuellement en lecture.

**La fonction loadSong()** permet de charger la source stockés dans la propriété "source" de l'objet Song.

**La fonction active()** permet d'enlever la classe active de tout éléments situé dans div.album et de mettre en surbrillance (par un changement de couleur de fond) le div.row de la piste actuellement joué par le lecteur.
la ligne du removeClass('active') permet d'éviter d'avoir deux pistes en surbrillance.

### Event onClick sur une piste

Pour le bon fonctionnement du lecteur sur chaque page du site, deux DIV en display:none sont mise à disposition pour aider le navigateur à charger la bonne musique au fonction des actions de l'utilisateur

```
<div class="row pb-5 mb-lg-5 mb-3 bg-home">
    <!-- POPULAIRE -->
    <section>
        <div class="d-none" id="res"></div>
        <div class="d-none" id="res2"></div>
        <div class="col-lg-11 offset-lg-1 ps-lg-0 mt-4 mt-lg-0">
            <h3 class="text-white fw-semibold mb-4">Populaire</h3>
            <div class="col-12 album ps-3">
```

Si l'utilisateur veut lancé une musique, un clic sur la piste voulu lancera un event onclic() qui permettra d'écouter la musique voulu :

```
$(y).click(function() {
    loadSong(track[i][j]);
    active($(y));
    play();

    $('#res').html(i);
    $('#res2').html(j);
});
```

Cette événement click se situe dans la deuxième boucle vue précédemment dans la préparation de la playlist.

Les variables i, j correspondent aux itérations des deux boucles et font également référence aux index du tableaux "track" :

- i pour les albums
- j pour les musiques

Etant donné que la structure de l'HTML correspond à la structure du tableau, un clique sur la musique voulu chargera les propriété de l'objet Song correspondant à ces coordonnées.

Les deux div citées au dessus se verront attribuer les coordonnées i et j de la musique lancée pour la suite.

## EVOLUTION DE LA LECTURE, MODE ALÉATOIRE / RÉPÉTÉ

### Pause

Une variable Booléen est définie afin de permettre de savoir quand un audio est en lecture ou non. Un event onclick sur le bouton #play-btn permet, à l'aide d'une ternaire, de modifier l'icône du bouton et de mettre l'audio en lecture ou pause en fonction de la valeur de la variable "Isplaying"

```
let isPlaying = !1;
$("#play-btn").click((() => {
    isPlaying ? ($("#play-btn").addClass("fa-circle-play").removeClass("fa-circle-pause"), audio.pause(), isPlaying = !1) : ($("#play-btn").addClass("fa-circle-pause").removeClass("fa-circle-play"), audio.play(), isPlaying = !0)
}));
```

## Mode Aléatoire / Répéter

Même principe que la lecture, variable Booléenne est définie pour chaque mode afin de permettre de savoir quand un mode est actif. Un event onclick sur le bouton du mode permet, à l'aide d'une ternaire, d'activer le mode ou non en fonction de la valeur de la variable Booléenne respective. Cet event est accompagné de la fonction d'animation d'icône vue plus haut.

```
let repeatMode = !1;
$("#repeat-btn").click((function () {
    repeatMode ? (repeatMode = !1, animIconOff($("#repeat-btn"))) : (repeatMode = !0, animIconOn($("#repeat-btn")))
}));
let shuffleMode = !1;
$("#shuffle-btn").click((function () {
    shuffleMode ? (shuffleMode = !1, animIconOff($("#shuffle-btn"))) : (shuffleMode = !0, animIconOn($("#shuffle-btn")))
})),
```

## Suivant / Précédent

Une fonction de charge du changement de musique lorsque l'utilisateur cliquera sur le bouton suivant ou précédent. Chaque bouton est associé avec un onclick faisant appelle à cette fonction. "t" est la valeur définie lors du onclick des boutons et transmise à la fonction:

- 1 pour le bouton suivant
- -1 pour le bouton précédent

```
function nextPrevious(t) {
    let e = parseInt($("#res").html()), a = parseInt($("#res2").html());
    let i = $(".album").eq(e);

    if (!repeatMode) {
        if (shuffleMode) {
            e = Math.floor(Math.random() * $(".album").length);
            a = Math.floor(Math.random() * ($(".album").eq(e).find(".track").length));
        } else {
            a += t;
            if (a < 0) {
                e = (e === 0) ? 0 : e - 1;
                i = $(".album").eq(e);
                a = $(".track", i).length - 1;
            } else if (a >= $(".track", i).length) {
                e = (e + 1) % $(".album").length;
                a = 0;
                i = $(".album").eq(e);
            }
        }
    }
    let n = $(i).find(".track:nth-child(" + (a + 1) + ")"), r = $(n).attr("id").split("-");
    loadSong(track[e][a]), active($(n)), play(), historiqueUser(r[1]), $("#res").html(e), $("#res2").html(a)
}
```

La fonction récupère ainsi les coordonnées stocker dans les deux div#res et div#res2 vue précédemment :

- la variable e récupère la coordonnée faisant référence à l'album.
- la variable a récupère la coordonnée faisant référence à la piste.

Plusieurs conditions vont alors rentrer en jeu :

- Si le mode répéter est actif ou non. Si oui, a et e prendront les valeurs déjà stocker dans div#res et div#res2.
- Si le mode aléatoire est actif ou non. Si oui, e prendra une valeur aléatoire comprise entre 0 et le nombre d'album présent sur la page et a une valeur comprise entre 0 et le nombre de piste présente dans l'album e.
- Si la somme de a et t donne un résultat inférieur à 0 : e prendra la valeur e - 1 et a prendra une valeur égale à la longueur de l'album e - 1 afin de lancer la dernière piste de l'album précédent. Dans le cas ou e est égal à 0, aucun changement ne sera effectués et la première piste du premier album de la page se relancera.
- Si la somme de a et t donne un résultat supérieur à la longueur de l'album e : e prendra la valeur e + 1 et a sera réinitialisé à 0 afin de lancer la première piste de l'album suivant.. Dans le cas ou e + 1 donne un résultat supérieur au nombre d'album présent sur la page, e sera également réinitialisé à 0 afin de revenir à la toute première piste de la page.
- Si aucune des conditions précédentes n'est remplie, a prendra juste la valeur de la somme a + t afin de lancer la musique suivante ou précédente.

## Fin de la piste

Lors d'une fin de piste audio, un event JS lié à l'objet JS Audio se lance reprenant la fonction vue juste au-dessus ainsi que la même logique.

```
$(audio).on("ended", (() => {
    nextPrevious(1)
})),
```

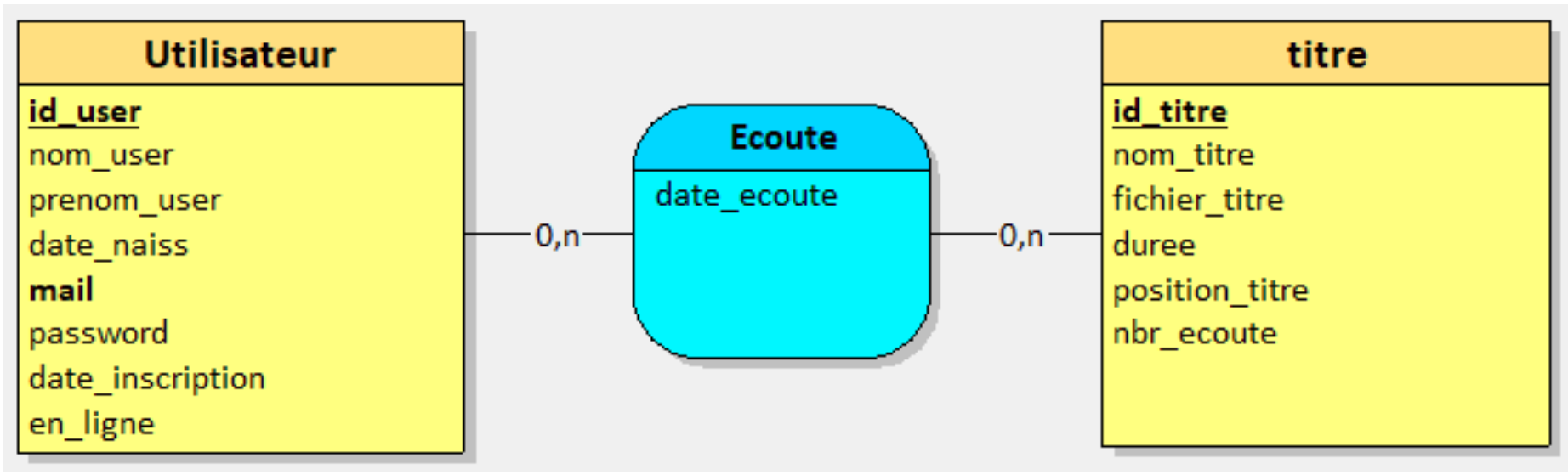
# CONCEPTION DE LA BASE DE DONNÉES

## MCD ET MLD

### RÉALISATION DU MCD - MODÈLE CONCEPTUEL DES DONNÉES

Avant de m'attaquer à la création de la base de données, j'ai créé le MCD en utilisant le logiciel looping.
J'ai dû réfléchir aux cardinalités et aux associations à effectuer entre les tables et les créer en fonction des données que le site devait afficher et celles récupérées. Je devais donc créer les attributs correspondants à ces informations tout en évitant au maximum les redondances.

**Exemple 1 :**

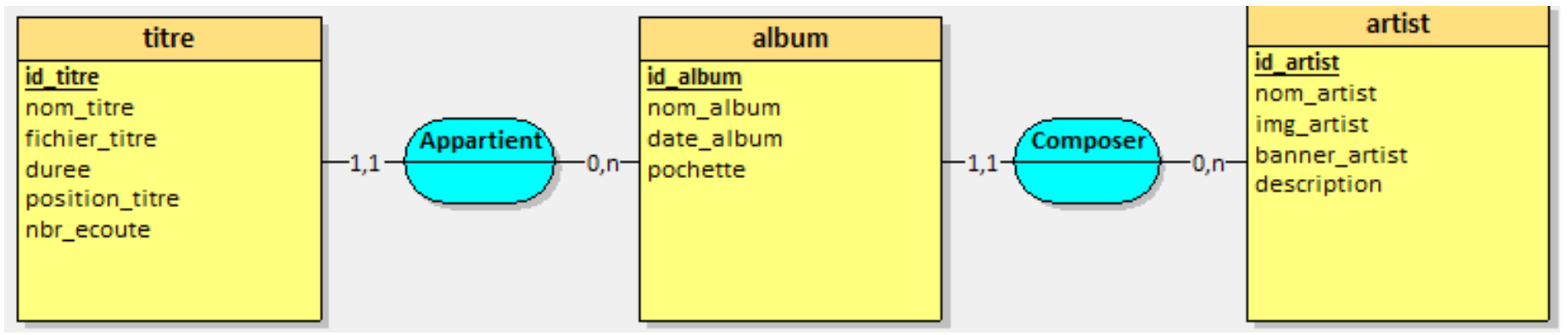



Dans notre exemple, je prends en situation la relation entre l'entité "utilisateur" et "titre" pour la génération d'un historique de lecture. Les cardinalités sont définies suivant la logique suivante :

- Un utilisateur peut très bien écouter aucun titre ou un nombre n de titre.
- Un titre peut être écouté par aucun utilisateur ou un nombre n d'utilisateur.

Un relation many to many se crée entre les deux entités.

**Exemple 2 :**




Dans ce deuxième exemple les cardinalités suivent la logique suivante :

- Un titre appartient forcément à un et un seul album et un album peut avoir aucun ou un nombre n de titre.
- Un album est forcément composé par un et un seul artiste et un artiste peut composer 0 ou un nombre n d'album.

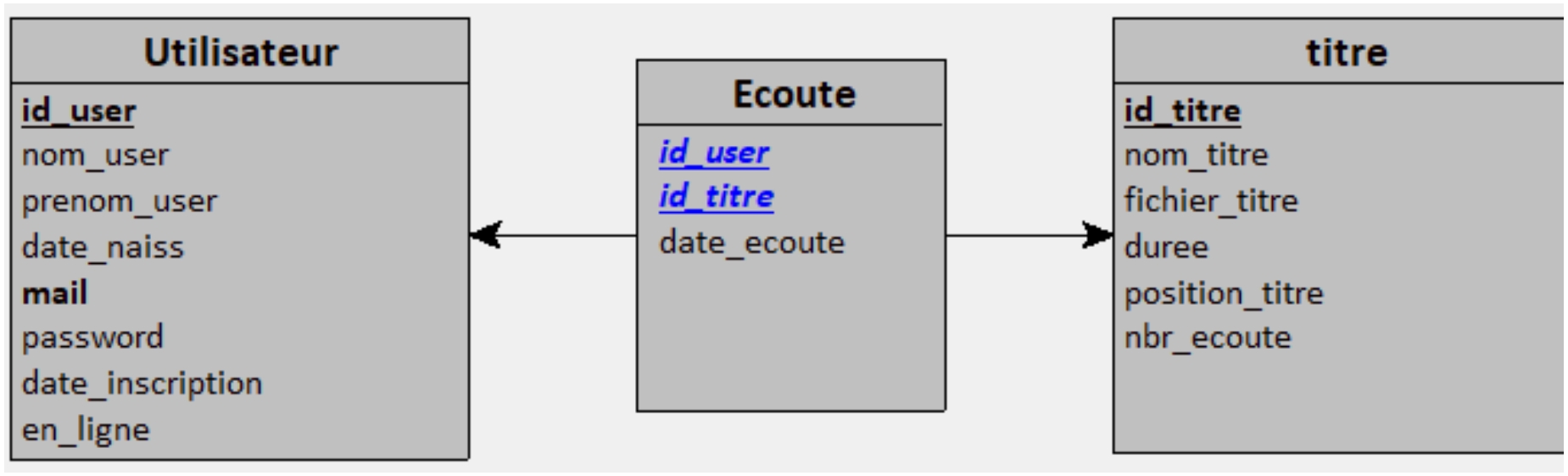
Les relations définies sont alors considérées comme many to one.

# RÉALISATION DU MLD - MODÈLE LOGIQUE DES DONNÉES

Le MLD permet de connaître le nombre de tables ainsi que leurs contraintes (liaisons entre tables) à mettre en œuvre dans une base de données relationnelle, ce qui permet de vérifier si les clés primaires et étrangères sont définies correctement et si mes tables intermédiaires correspondent bien (avec les bonnes clés étrangères). Lors du passage du mcd au mld :

- Les entités deviennent des tables
- Si une relation est de type many to one, une clé étrangère est ajoutée du côté du one  faisant référence à la clé primaire de l'autre table.
- Si une relation est de type many to many, une table intermédiaire apparaît avec notamment deux clés étrangères correspondant aux clés primaires des 2 tables concernées.

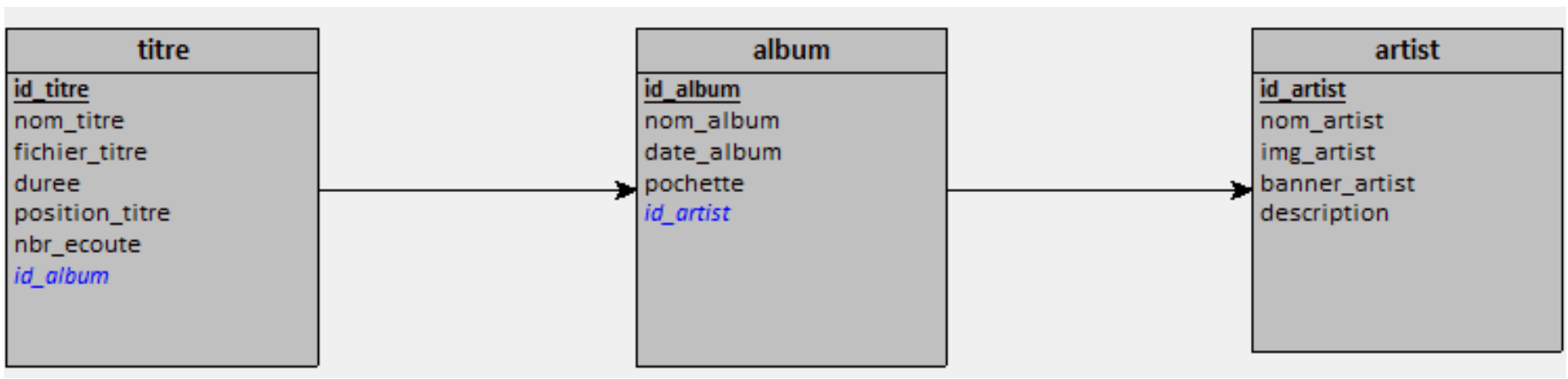
**Exemple 1 :**




Dans notre exemple, l'association "ecoute" devient un table intermédiaire due à la relation many to many entre les deux tables 'utilisateur' et 'titre', elle sera renommé 'historique_user' pour plus de clarté.  Afin de permettre de savoir si un titre a été relu plusieurs fois, la clé primaire de cette table sera composé de ses trois attributs :

**HISTORIQUE_USER{#id_user, #id_titre, date_ecoute}**

**Exemple 2 :**




Dans le deuxième exemple, due aux relations many to one, la table album se voit recevoir en clé étrangère #id_artist faisant référance à la clé primaire de la table artiste et la table titre celle de la table album. Grâce à ça, l'organisation de ses données facilitera l'affichage dynamique de ce contenu.

**ARTIST{id_artist, nom_artist, img_artist, banner_artist, description}**

**ALBUM{id_album, nom_album, date_album, pochette, #id_artist}**

**TITRE{id_titre, nom_titre, fichier_titre, duree, position_titre, nbr_ecoute, #id_album}**

# LA BASE DE DONNÉES

| Table | Action | Lignes | Type | Interclassement | Taille | Perte |
|---|---|---|---|---|---|---|
| abonnement | Parcourir Structure Rechercher Insérer Vider Supprimer | 4 | InnoDB | utf8_unicode_ci | 32,0 kio | - |
| admin | Parcourir Structure Rechercher Insérer Vider Supprimer | 1 | InnoDB | latin1_swedish_ci | 16,0 kio | - |
| album | Parcourir Structure Rechercher Insérer Vider Supprimer | 23 | InnoDB | utf8_unicode_ci | 32,0 kio | - |
| artist | Parcourir Structure Rechercher Insérer Vider Supprimer | 13 | InnoDB | utf8_unicode_ci | 16,0 kio | - |
| calendrier | Parcourir Structure Rechercher Insérer Vider Supprimer | 144 | InnoDB | latin1_swedish_ci | 16,0 kio | - |
| genre | Parcourir Structure Rechercher Insérer Vider Supprimer | 13 | InnoDB | utf8_unicode_ci | 16,0 kio | - |
| genre_album | Parcourir Structure Rechercher Insérer Vider Supprimer | 10 | InnoDB | utf8_unicode_ci | 32,0 kio | - |
| genre_artist | Parcourir Structure Rechercher Insérer Vider Supprimer | 19 | InnoDB | utf8_unicode_ci | 32,0 kio | - |
| historique_user | Parcourir Structure Rechercher Insérer Vider Supprimer | 748 | InnoDB | utf8_unicode_ci | 80,0 kio | - |
| img_banner_user | Parcourir Structure Rechercher Insérer Vider Supprimer | 5 | InnoDB | latin1_swedish_ci | 32,0 kio | - |
| img_user | Parcourir Structure Rechercher Insérer Vider Supprimer | 7 | InnoDB | latin1_swedish_ci | 32,0 kio | - |
| playlist | Parcourir Structure Rechercher Insérer Vider Supprimer | 3 | InnoDB | utf8_unicode_ci | 16,0 kio | - |
| reseaux_artist | Parcourir Structure Rechercher Insérer Vider Supprimer | 8 | InnoDB | latin1_swedish_ci | 32,0 kio | - |
| reseaux_sociaux | Parcourir Structure Rechercher Insérer Vider Supprimer | 4 | InnoDB | latin1_swedish_ci | 16,0 kio | - |
| titre | Parcourir Structure Rechercher Insérer Vider Supprimer | 213 | InnoDB | utf8_unicode_ci | 32,0 kio | - |
| titre_like | Parcourir Structure Rechercher Insérer Vider Supprimer | 18 | InnoDB | utf8_unicode_ci | 32,0 kio | - |
| titre_playlist | Parcourir Structure Rechercher Insérer Vider Supprimer | 46 | InnoDB | utf8_unicode_ci | 32,0 kio | - |
| user | Parcourir Structure Rechercher Insérer Vider Supprimer | 2 | InnoDB | utf8_unicode_ci | 16,0 kio | - |
| user_abonne | Parcourir Structure Rechercher Insérer Vider Supprimer | 0 | InnoDB | utf8_unicode_ci | 32,0 kio | - |

J'ai récupéré les requêtes SQL générées par looping qui m'ont permis de créer la base de données et les tables dans l'interface web phpMyAdmin.
Pour commencer la création de la base de données en local à l'aide de Xampp. Et le SGBD (system de gestion de base de données) est MySQL. La base de données est donc une base de données relationnelle MySQL dont le moteur de stockage est innoDB, l'utf-8 a été utilisé car cela permet la prise en charge de l'Unicode qui permet entre autre de prendre en compte tous les caractères peu importe la langue et est insensible à la casse.

# CONNEXION PDO

Afin d'établir la connexion entre la base de données et le site, un fichier nommé "connection_pdo.php" sera placé dans la dossier "modele" de l'architecture MVC et contiendra le code suivant :

```php
<?php
    try{
        $bdd = new PDO('mysql:host=localhost;dbname=rmdevwd12;charset=UTF8','root','');
    } catch(PDOException $e) {
        echo $e->getMessage();
        die();
    }
?>
```

Stocké dans une variable $bdd, l'extension php PDO (PHP DATA OBJECT) établi la connexion à la base de données en lui fournissant les informations nécessaire en paramètre :

- host : adresse serveur où la base de données est située, "localhost" si la base de données se trouve sur le même serveur que le site.
- dbname : nom de la base de données.
- charset : type d'encodage correspondant à la base de données
- user/mdp : identifiants nécéssaire pour accéder à la base de données

Dans le cas présent,

# RÉALISATION DE LA PARTIE BACKEND

## AFFICHAGE DYNAMIQUE

Afin d'afficher le contenu enregistré sur la base de données sur le site, des fonctions récupérant les informations souhaitées stockent ces données dans des tableaux associatifs (avec les clef correspondant au nom des attributs des tables de la base de données).

```php
function getAllArtist($bdd) {
    try{
        $req = $bdd->prepare('SELECT * FROM artist a WHERE en_ligne = 1 ORDER BY a.nom_artist ASC');
        $req->execute();
        $artists = $req->fetchAll();

        return $artists;
    } catch(PDOException $e) {
        echo $e->getMessage();
    }
}
```

**try** et **catch** sont des blocs de code utilisés en programmation pour gérer les erreurs et les exceptions de manière contrôlée. Ils sont couramment utilisés pour anticiper et gérer les situations où une erreur pourrait survenir pendant l'exécution d'un programme, afin d'éviter des arrêts brusques ou des comportements inattendus.

- $bdd correspond à l'objet PDO vu précédemment et faisant le lien avec la base de données.
- SELECT * correspond à une requête SQL permettant de récupérer tous les attributs présent dans la table "artiste'
- la condition WHERE en_ligne = 1 correspond au booléen permettant de savoir si le contenu doit etre affiché ou non sur le site
- fetchAll() est utilisée après avoir exécuté une requête SELECT avec PDO pour récupérer toutes les lignes de résultats de la requête.

La fonction ci dessus est appelé pour récupéré toutes les informations liées aux artistes présent dans la base de données. Une fois stocké dans une varaiable, un foreach permet d'afficher le contenu HTML de chaque artistes.

```php
if($artists) {
    foreach($artists as $i =>$artist) {
        ?>
            <div class="col-lg-2 col-4 mb-3">
                <a href="artist/<?php echo $artist['id_artist'] ?>">
                    <div class="text-center">
                        <div class="ratio ratio-1x1">
                            <img src="public/assets/img/artist/<?php echo $artist['img_artist'] ?>" alt="<?php echo $artist['nom_artist'] ?>"
                        </div>
                        <div class="mt-3">
                            <h5 class="card-title text-white fs-15 text-truncate fw-semibold"> <?php echo $artist['nom_artist'] ?></h5>
                        </div>
                    </div>
                </a>
            </div>
        <?php
    }
}
```

Le principe sera le même pour les albums présents sur la page d'accueil.

## PAGE ARTISTE / ALBUM

L'id du contenu étant présent dans le lien menant à la page dédiée, en arrivant sur cette derniere plusieurs fonction avec cette id en paramètre seront éxecutées.

```php
$idArtist = htmlspecialchars($_GET['artist']);
$artist = getArtistInfo($bdd,$idArtist);
```

C'est deux ligne de code sont présente au haut de la page include permettant d'actualisé le contenu en fonction de l'id fourni par le lien.

# TRATIEMENT INSCRIPTION

Un nouvel utilisateur voulant accéder au contenu du site devrait passer par la validation du formulaire d'inscription.

```php
<!-- SAFE&SOUND FORM -->
<form action="traitement_inscription" method="POST">
    <div class="mb-2">
        <label for="form-prenom" class="form-label text-white fs-12 fw-semibold mb-1" required>Prenom</label>
        <input type="text" name="prenom" id="form-prenom" class="form-control fs-12 bg-light-primary border-dark text-white" required>
    </div>
    <div>
        <label for="form-nom" class="form-label text-white fs-12 fw-semibold mb-1" required>Nom</label>
        <input type="text" name="nom" id="form-nom" class="form-control fs-12 bg-light-primary border-dark text-white" required>
    </div>
    <div class="mb-2">
        <label for="form-mdp"  class="form-label text-white  fs-12 fw-semibold mb-1" required>Password</label>
        <input type="password" name="mdp" id="form-mdp" class="form-control bg-light-primary border-dark fs-12 text-white" required>
        <div id="form-mdpHelp" class="form-text fs-10">Votre mot de passe doit être composé de 8 caractères minimum.</div>
    </div>
    <div class="mb-3">
        <label for="form-mdpConfirm"  class="form-label text-white fs-12 fw-semibold mb-1" required>Confirm Password</label>
        <input type="password" name="mdpConfirm" id="form-mdpConfirm" class="form-control bg-light-primary border-dark fs-12 text-white" required>
    </div>
    <button type="submit" class="form-submit btn btn-danger w-100 text-uppercase fs-12 rounded-pill disable">S'Inscrire</button>
</form>
```

Une fois validé via le bouton:submit, les données saisies dans les champs sont envoyé vers un fichier de traitement défini dans l'attribut "action" de la balise form. La method=POST correspond à la méthode d'envois des données :

- POST enverra les données dans le coeur de la requête HTTP
- GET les enverra dans l'url

La method POST permet un envois plus sécurisé.

En arrivant sur le fichier de traitement, chaque informations saisies par l'utilisateur est stocké dans la superglobale $_POST à la clef correspondant au name de l'input :

- $_POST['nom'] contiendra le texte saisi par l'utilisateur dans l'input:name="nom".
- $_POST['mdp'] contiendra le texte saisi par l'utilisateur dans l'input:name="mdp".

```php
$nom = htmlspecialchars($_POST['nom']);
$prenom = htmlspecialchars($_POST['prenom']);
$mail = htmlspecialchars($_POST['mail']);
$mdp = htmlspecialchars($_POST['mdp']);
$mdpConfirm = htmlspecialchars($_POST['mdpConfirm']);
$date = (new DateTime())->format('Y-m-d');

$userExist = userExist($bdd, $mail);

if($userExist) {
    header('location:inscription' . '?error=signIn');

} else {
    $mdpHash = password_hash($mdp, PASSWORD_DEFAULT);

    $user = newUser($bdd, $prenom, $nom, $mail, $mdpHash, $date);

    if($user) {
        $_SESSION['id_user'] = $user;
        setOnlineUser($bdd, $user);

        header('location:accueil');
    } else {
        header('location:inscription' . '?error=signIn');
    }
}
```

Chaque informations est récupérées dans une variable et un fonction contenant le mail saisi s'éxecute afin de savoir si ce dernier est déjà utilisé par un autre compte (le mail étant défini comme unique dans la table user).
Si cette fonction retourne quelque chose, la condition if(userexist()) sera considéré comme VRAI et l'utilisateur se verra redirigé vers la page d'inscription avec un message l'invitant à changer de mail ou se connecter.
Si la fonction ne retourne rien,  le code contenant dans le ELSE s'éxecute et une fonction contenant toute les informations saisie va envoyer ces données dans la table user présent dans la base de données.

L'envois des données dans la base de données s'effectue via une requête SQL INSERT sous la forme suivante :

```php
// INSCRIPTION
function newUser($bdd, $prenom, $nom, $mail, $mdpHash, $date) { // CREATION NOUVEL USER
    $req = $bdd->prepare('INSERT INTO user(prenom_user, nom_user, mail_user, `password`, date_inscription) VALUES(  :prenom_user, :nom_user, :mail_user, :mdp, :date_signin)');
    $req->execute([':nom_user'=>$nom, ':prenom_user'=>$prenom, ':mail_user'=>$mail, ':mdp'=>$mdpHash, ':date_signin'=>$date]);
    $user = $bdd->lastInsertId();

    return $user;
}
```

La requête préparé se voit attribué des paramètres représentant chaque données qui vont lui être attribué lors son éxécution.
Une fois cette étape réalisée, la méthode lastInsertId() permet de récupéré directement l'id crée de cette insertion afin de connecté directement l'utilisateur fraichement inscrit en la stockant dans une variable $_SESSION['id_user'] nécéssaire pour l'affichage du contenu disponible qu'en étant connecté.

# TRATIEMENT CONNEXION

Même principe que pour l'inscription, une fois le formulaire rempli, les données envoyé via method POST sont récupéré dans le fichier de traitement et utilisées par un fonction récupérant les informations stockées dans la base de données.

```php
$mail = htmlspecialchars($_POST['mail']);
$mdp = htmlspecialchars($_POST['mdp']);

$userExist = userExist($bdd, $mail);

if($userExist) {
    if(password_verify($mdp, $userExist['password'])) {

        $_SESSION['id_user'] = $userExist['id_user'];
        setOnlineUser($bdd, $userExist['id_user']);

        header('location:accueil');
    } else {
        header('location:accueil' . '?error=login');
    }
} else {
    header('location:accueil' . '?error=login');
}
```

Un SELECT * est utilisé suivi d'une condition disant à la base de données de sortir seulement les données associé au mail saisi dans la table "user". Ne voulant récupérer qu'une seule rangée de cette table, un fetch() sera utilisé pour récupéré les données dans une variable.
Si cette requête retourne bien quelque chose, une condition comparant le mot de passe saisi et celui stocké et encodé dans la base de données sera exécutée. Si et seulement si cette dernière condition est vraie, l'utilisateur sera alors connecté et renvoyé à la page d'accueil.

Si une des conditions présente dans ce traitement est fausse, l'utilisateur sera renvoyé sur la page d'accueil avec la modal de connexion affichant un message d'erreur et l'invitant à retenter en vérifiant ses informations.

# AJAX - ASYNCHRONOUS JAVASCRIPT AND XML

C'est une technique de programmation utilisée pour créer des applications Web interactives en permettant aux navigateurs de communiquer avec le serveur en arrière-plan, sans avoir à recharger la page entière. L'utilisation d'AJAX permet de mettre à jour dynamiquement une partie spécifique d'une page sans interrompre l'expérience de l'utilisateur

Etant donné que le site propose l'écoute de musique, l'utilisation de fonction AJAX permettant la modification des données de la base de données sans redirections interrompant la lecture me semblait essentiel.

## HISTORIQUE UTILISATEUR ET AJOUT DE FAVORIS

```
function historiqueUser(t) {
    $.ajax({
        url: "https://localhost/rmdevwd/traitement_user",
        type: "POST",
        data: {
            source: "historiqueUser",
            titre: t
        }
    })
}

function like(t, e) {
    $.ajax({
        url: "https://localhost/rmdevwd/traitement_user",
        type: "POST",
        data: {
            source: "likeTitre",
            user: t,
            titre: e
        }
    })
}

function unlike(t, e) {
    $.ajax({
        url: "https://localhost/rmdevwd/traitement_user",
        type: "POST",
        data: {
            source: "unlikeTitre",
            user: t,
            titre: e
        }
    })
}
```

Utilisant JQUERY, les fonctions ajax seront écrites de cette manière :

- url correspondant à l'url de destination de la requête
- type, de la même façon que la method d'un formulaire, les données sont envoyé en POST au coeur de la requête HTTP
- Les données à envoyer sont définies sous forme d'objet JavaScript dans la propriété data de la requête AJAX. Cet objet contient les paires clé-valeur pour les paramètres à envoyer.

La fonction historiqueUser() se voit passer en paramètre l'id correspondant à la musique lancée et sera exécutée dans les événements vu précédemment liés au lancement d'une musique et aux contrôle du lecteur Audio.

Les fonctions like() et unlike() seront appelées via un onclick sur l'icône favoris présent sur chaque pistes avec l'id fourni en paramètre.

Afin d'éviter que tout le code prévu pour ces fonctions s'exécute, la variable "source" servira à compartimenter le fichier de traitement et sa valeur définira le code à éxécuter.

**exemple :**

```
} elseif($_POST['source'] == 'historiqueUser') {

    $user = htmlspecialchars($_SESSION['id_user']);
    $titre = htmlspecialchars($_POST['titre']);
    $date = new DateTime();
    $date = $date->format('Y/m/d H:i:s');

    addTitreToHist($bdd, $user, $titre, $date);
    updateNbrEcoute($bdd, $titre);
}
```

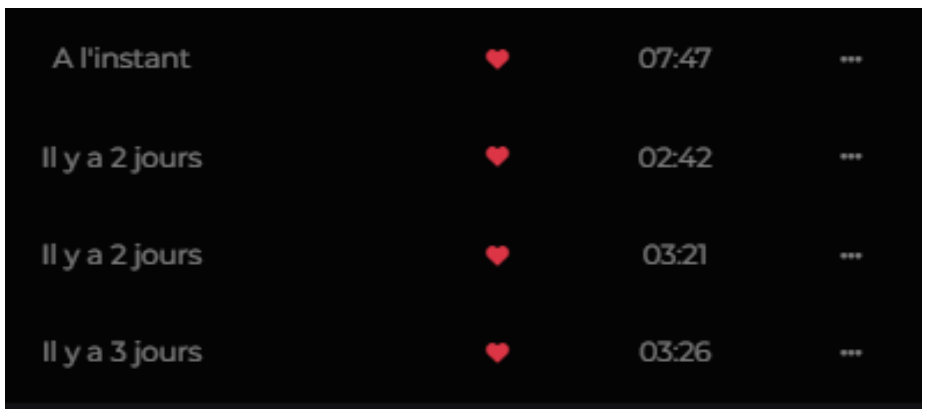
lors du la lecture d'une musique, la fonction AJAX historiqueUser() s'exécute et envois l'id de la piste lancée au fichier de traitement.
La valeur de source définie dans la fonction AJAX lancera le code lié à cette fonction et permettra le lancement de deux fonction :

- Une ajoutant dans la table historique_user l'id de l'utilisateur initialement stocké en session et l'id du titre ainsi que la date actuelle.
- L'autre faisant un UPDATE de la valeur nbr_ecoute de la piste en cours en remplacer la valeur actuelle par nbr_ecoute + 1.

Même principe pour like() et unlike(), la variable source définira le compartiment de code à exécuter et ajoutera ou supprimera la ligne contenant l'id de l'utilisateur, de la piste et de la date du moment de l'exécution du code.

# MISE EN FORME DE LA DATE AJOUT FAVORIS / HISTORIQUE

Afin d'avoir un affichage plus clair pour l'utilisateur, une modification de la date est nécessaire afin d'obtenir ce type d'affichage :




Pour ce qui est de l'ajout en favoris, de l'écoute d'un titre, la date correspondant à cet événement est stocké dans la base de données sous la forme d'une dateTime ( Année/Mois/Jour Heure/munite/secondes).

La date récupérée se voit changé en Timestamp via un fonction native PHP strtime(). Un "timestamp" est un moyen de représenter une date et une heure précises sous forme d'une valeur numérique. Il s'agit généralement du nombre de secondes (ou de millisecondes) écoulées depuis un point de référence fixe.

```php
$date = new DateTime($titre['date_ajout']);
$date = strtotime($date->format('Y-m-d H:i:s'));
$dateNow = time();
```

Une autre variable nommée $dateNow récupére le Timestamp du moment où l'utilisateur arrive sur la page et une soutraction entre $dateNow et $date s'oppère afin de déterminer l'écart en seconde entre ce moment et celui où le titre a été écouté ou ajouté en favoris.

```php
$diff = $dateNow - $date;
// SECONDES
if($diff < 60) {
    if($diff >= 2) {
        echo 'Il y a ' . $diff . ' secondes';
    } else {
        echo "A l'instant";
    }
// MINUTES
} elseif(($diff / 60) < 60) {
    if($diff / 60 >= 2) {
        echo 'Il y a ' . number_format($diff / 60) . ' minutes';
    } else {
        echo 'Il y a ' . number_format($diff / 60) . ' minute';
    }
// HEURES
} elseif(($diff / 3600) < 24) {
    if(($diff / 3600) > 1) {
        echo 'Il y a ' . number_format($diff / 3600) . ' heures';
    } else {
        echo 'Il y a ' . number_format($diff / 3600) . ' heure';
    }
// JOURS
} elseif(($diff / 86400) < 31) {
    if(($diff / 86400) > 1) {
        echo 'Il y a ' . number_format($diff / 86400) . ' jours';
    } else {
        echo 'Il y a ' . number_format($diff / 86400) . ' jour';
    }
// MOIS
} elseif(($diff / 2678400) < 12) {
    echo 'Il y a ' . number_format($diff / 2678400) . ' mois';
// ANNEES
} else {
    echo 'Il y a ' . number_format($diff / 2678400 * 12) . ' ans';
}
```

Enfin, une suite de condition et d'opérations va déterminer le texte à affiché :

- Si la différence est inférieur à 60 secondes, "à l'instant ". sera affiché si $diff < 2 et "il y a $diff secondes" inversement.
- Si la différence est supérieur à 60, une conversion en minute via un division par 60 affichera "il y a n minute"

etc...

# BACK OFFICE

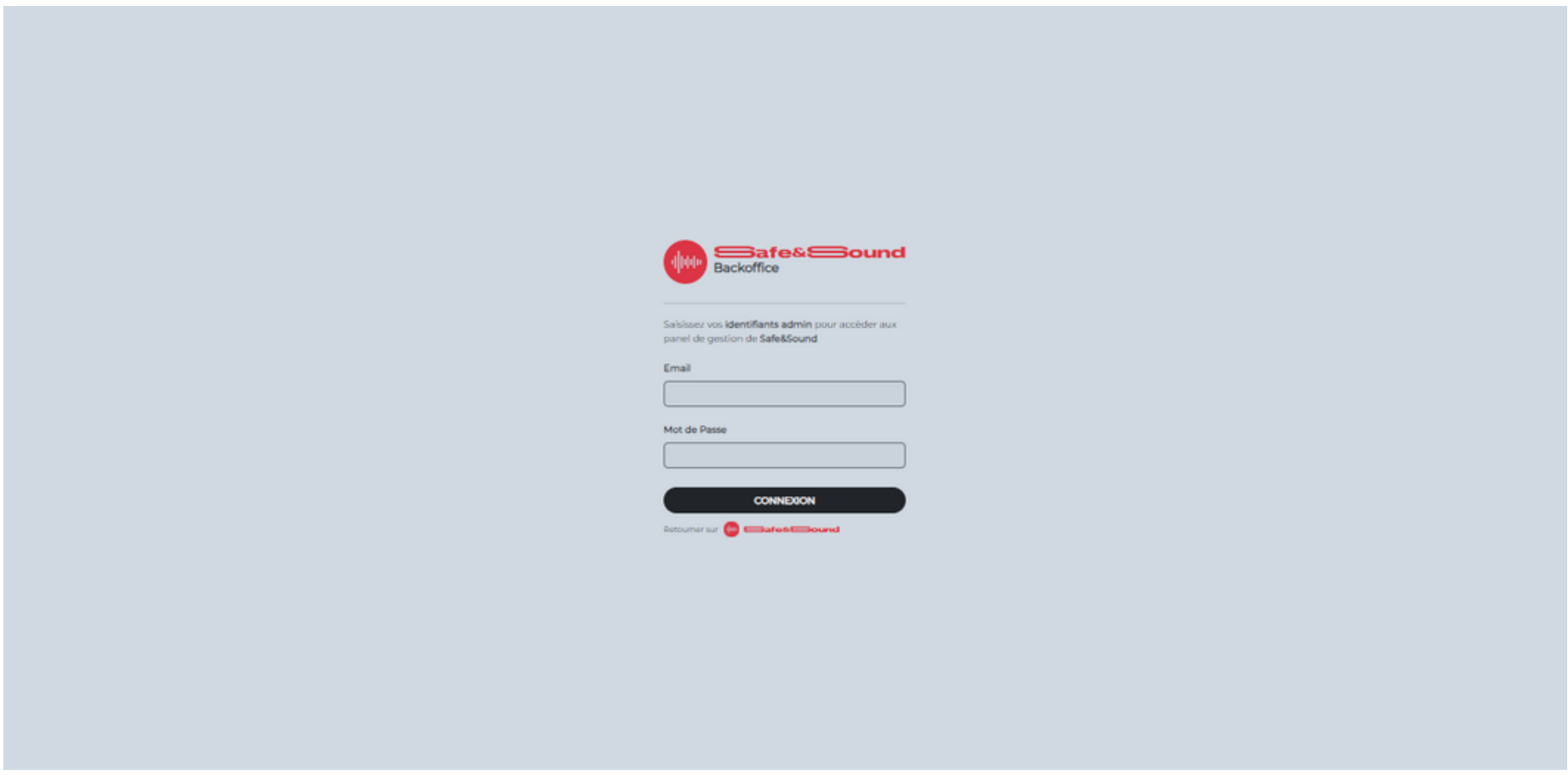
## CONNEXION BACK OFFICE

Même principe que pour la connexion d'un utilisateur sur le site, l'administrateur devra saisir les informations stocké dans un table "admin" de la base de données.

```php
$nom = htmlspecialchars($_POST['email']);
$password = htmlspecialchars($_POST['password']);

$admin = checkAdmin($bdd);
```

Une fois connecter, une variable SESSION permettra de vérifier si l'administrateur doit ou non passer par le formulaire de connexion pour accéder au back-office.

```php
if($admin) {
    if(password_verify($password, $admin['password'])) {
        $_SESSION['admin'] = $admin['id_admin'];

        $lastDateEntry = recupLastDateUpdate($bdd);

        getBetweenDates($lastDateEntry['jour'], date('Y-m-d'), $bdd);
        header('location:accueil');

    } else {
        header('location:accueil?erreur=1');
    }
} else {
    header('location:accueil?erreur=1');
}
```

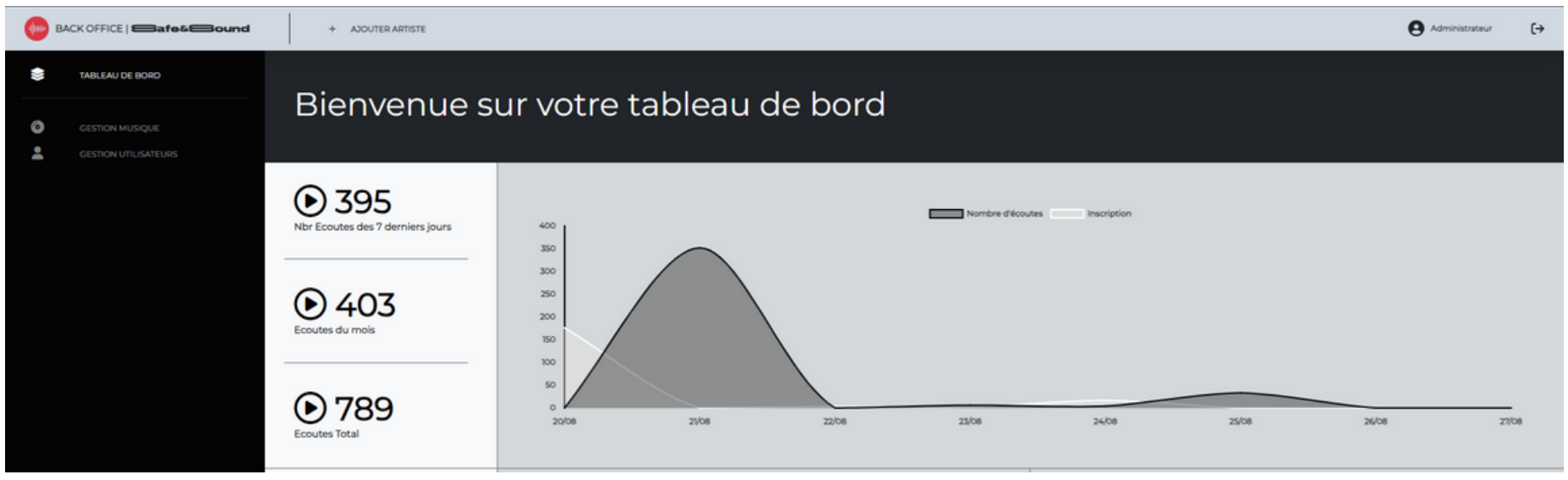


# BACK OFFICE

## CHART.JS ET STATISTIQUE D'ÉCOUTE

A l'arrivée sur le backoffice, l'administrateur aura accès aux statistiques d'écoute du site via un graphique et trois statistiques différentes.

L'accès à ces stats est possible via la récupération totale des données de la table historique_user




Le premier problème rencontré en travaillant en local était le faite de n'avoir pas d'entrée pour tous les jours d'une semaine, d'un mois etc... N'ayant pas assez d'utilisateur actif, la table historique utilisateur n'avait d'entrée pour un calendrier complet. Afin de ce régler ce problème, j'ai mis en place une table calendrier avec une seul entrée défini par la date de création du site(définie le 1er mars 2023). A chaque connexion de l'administrateur, un fonction se chargera de comparer la date du jour à la date la plus récente présente dans la table calendrier et d'insérer dans cette table tous les jours compris entre ces deux dates, date du jour compris.

```php
function getBetweenDates($startDate, $endDate, $bdd) {
    $startDate = strtotime($startDate);
    $endDate = strtotime($endDate);

    for ($currentDate = $startDate += (86400); $currentDate  <= $endDate; $currentDate += (86400)) {
        $date = date('Y-m-d', $currentDate);
        insertDate($bdd, $date);
    }
}
```

Par exemple, lors de la première connexion de l'administrateur le 27 août 2023, la date de tous les jours compris entre cette date et celle de création le 1er mars 2023 seront ajoutés dans la table calendrier.

Une fois cette étape effectuée, une fonction ajax sur un événement onload(une fois que la page a fini d'être chargée) va se chargée de récupérer les statistiques nécessaire à la création du graphique :

```php
function getStatXDays($bdd, $test) {
    $req = $bdd->prepare('SELECT c.jour, COUNT(u.id_user) AS total_inscris, COUNT(h.id_titre) AS total_listens
                          FROM calendrier c
                          LEFT JOIN user u ON DATE(u.date_inscription) = c.jour
                          LEFT JOIN historique_user h ON DATE(h.date_ecoute) = c.jour
                          WHERE c.jour >= CURDATE() - INTERVAL :test DAY
                          GROUP BY c.jour
                          ORDER BY c.jour ASC');

    $req->execute([':test'=>$test]);
    $data = $req->fetchAll();

    return $data;
}
```

Via un COUNT() permettant d'obtenir un somme d'un même élément et d'un LEFT JOIN de la table historique_user permettant l'affichage de jour avec aucune écoute avec un count() égale à 0 et une condition basé sur la date du jour sourtrait par un interval défini en paramètre lors de l'appel de la fonction.

```javascript
success: function(response) {
    data = JSON.parse(response);
    let labels = [];
    let stats = [];
    let stats2 = [];
    for(let i = 0; i < data.length; i++) {

        labels[i] = formatDate((data[i].jour));
        stats[i] = data[i].total_listens;

    }
    listenChart(ctx, labels, stats, stats2);
}
```

Une fois les données récupérées et encodées en JSON sur le fichier de traitement, la fonction ajax décodera le json obtenu pour permettre de préparé les statistiques pour le graphique

Au moment de l'arrivé sur la page d'accueil, la fonction AJAX s'exécutera avec le paramètre 7 correspondant à l'affichage des statistiques d'écoute de la semaine passée.

# NOUVEAU CONTENU

Afin de permettre à l'administrateur d'importer du contenu sur le site sans passer par l'ajout de l'artiste puis de l'album et des titres un par un, j'ai mis en place une manière d'importer plusieurs albums d'un coup via un fichier excel.

Grâce au composer et à l'extension import de PHP, une redirection sur l'index du dossier d'import permet l'exécution du code suivant :

**fichier Excel :**

| | nom_titre | fichier_titre | duree_titre | position_titre | nom_album | date_album | nom_artist |
|---|---|---|---|---|---|---|---|
| 2 | Do I Wanna Know? | NULL | 04:32 | 1 | AM | 2013-09-09 | Arctic Monkeys |
| 3 | R U Mine? | NULL | 03:21 | 2 | AM | 2013-09-09 | Arctic Monkeys |
| 4 | One For The Road | NULL | 03:26 | 3 | AM | 2013-09-09 | Arctic Monkeys |
| 5 | Arabella | NULL | 03:27 | 4 | AM | 2013-09-09 | Arctic Monkeys |
| 6 | I Want it All | NULL | 04:33 | 5 | AM | 2013-09-09 | Arctic Monkeys |
| 7 | No. 1 Party Anthem | NULL | 03:21 | 6 | AM | 2013-09-09 | Arctic Monkeys |
| 8 | Mad Sounds | NULL | 03:26 | 7 | AM | 2013-09-09 | Arctic Monkeys |

```php
if ($xlsx = SimpleXLSX::parse('import.xlsx')) {
    $imports=$xlsx->rows();

    $i=-1;
    $nbArticles=0;
    foreach ($imports as $import){
        $i++;
        if ($i>0){
            $nom_titre = $import[0];
            $fichier_titre = $import[1];
            $duree_titre = strval($import[2]);
            $position_titre = $import[3];
            $nom_album = $import[4];
            $date_album = $import[5];
            $nom_artist = $import[6];
```

Le fichier excel étant préparé en sorte d'avoir des titres de colonne, le code récupère, via un boucle en fonction du nombre de colonnes présentent sur le fichier.

Une fois les données sur le fichier récupérées dans des variables, je commence par faire une condition pour savoir si l'artiste présent sur le fichier existe déjà dans la base de données :

- SI oui, je récupère l'id correspondant
- Si non, je l'insert dans la base de données et récupère l'id grâce au lastInsertId()

Même principe pour les album.

```php
$reqArtistExist = $bdd->prepare('SELECT * FROM artist WHERE nom_artist = :artist');
$reqArtistExist->execute([':artist'=>$nom_artist]);
$reqArtistExist = $reqArtistExist->fetch();

if ($reqArtistExist){
    $id_artist = $reqArtistExist['id_artist'];
}else{
    // requete prepare INSERT INTO artist...
    $reqInsertArtist=$bdd->prepare('INSERT INTO artist(nom_artist) VALUES (:artist)');
    $reqInsertArtist->execute([':artist'=>$nom_artist]);

    $id_artist = $bdd->lastInsertId();
}
```

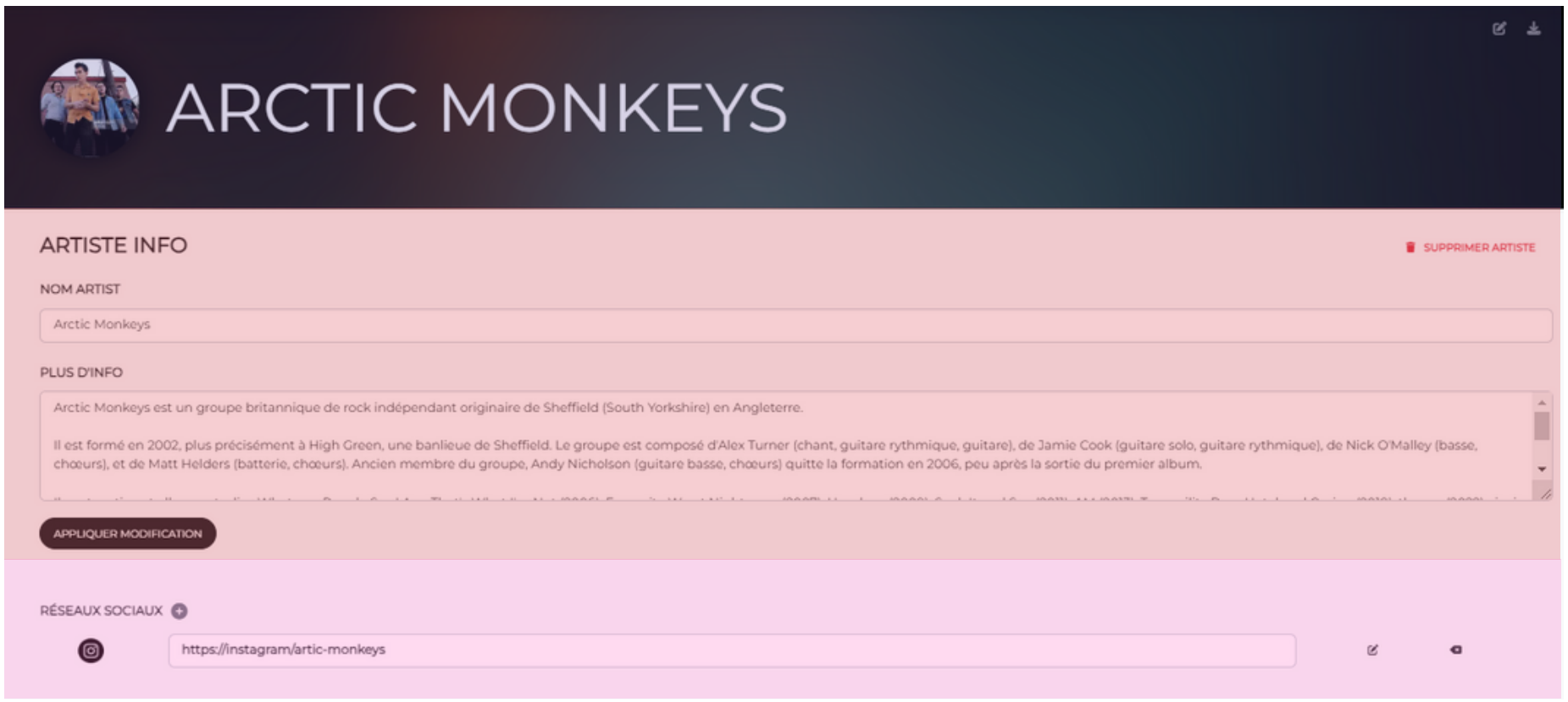
Une fois l'id de l'album récupéré, les titres seront ajouté dans la base de données avec toutes les informations présentent sur le fichier excel.

Une fois l'importation terminé, l'administrateur se verra redirigé vers la page de gestion d'artiste et pourra modifier les images de l'artiste récemment ajouté et des nouveaux albums.

# MISE A JOUR DU CONTENU

Sur la page de gestion de contenu l'administrateur aura accès à une page par artiste contenant plusieurs formulaire pour chaque catégorie de contenu

formulaire image artiste  formulaire Information  formulaire Réseaux Sociaux



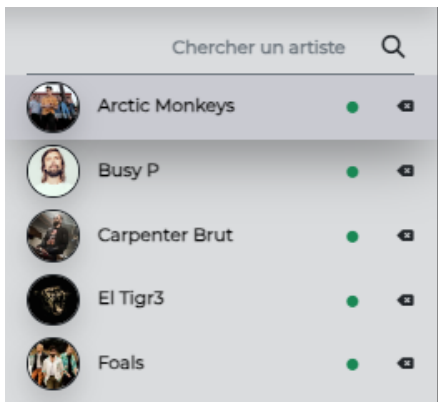

formulaire album  formulaire titre




Au chargement de la page, les formulaires sont préremplis par les informations stockées dans la base de données.
Chaque formulaires est associé au fichier de traitement_artist fourni avec une variable GET servant à compartimenter le code présent sur ce fichier.
A chaque compartiment une fonction avec une requête d'update est mise à disposition avec en paramètre les informations saisie dans les inputs du formulaire validé.

# FONCTION DE RECHERCHE AJAX

Pour permettre à l'administrateur de modifier le contenu d'un artiste précis plus rapidement, j'ai mis en place une fonction de recherche d'artiste :




Le menu latéral de la page de gestion d'artiste est composé d'un input avec une id lié à un événement "input" permettant d'exécuter une fonction à chaque saisie de l'administrateur :

- à la saisie de l'utilisateur, une fonction AJAX se lance et envois le texte issu de cette saisie à un fichier de traitement
- afin de préparé la requête correspondante, je concatène le texte récupéré avec un % permettant la condition LIKE en SQL. par exemple, si l'administrateur tape "t", la condition LIKE "t%" permettra de récupérer tous ce qui commence par la lettre t dans la table voulue.

```
$('#search-artist').on('input',() => {
    $.ajax({
        url: 'traitement_artist',
        type:'POST',
        data: {source: 'search', input: $('#search-artist').val()},
        success: function(response) {
            let artists = JSON.parse(response)
```

```
} elseif($_POST['source'] == 'search') {
    $input = htmlspecialchars($_POST['input']) . '%';

    $artists = searchArtist($bdd, $input);
    echo json_encode($artists);
}
```

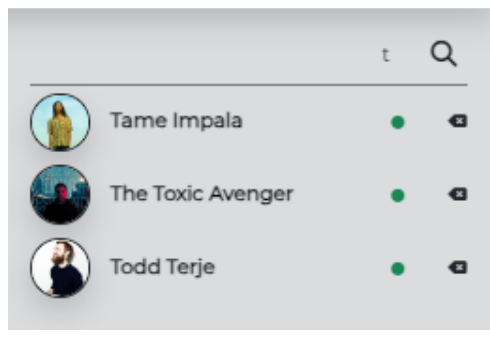
Dans le cas présent, la requête récupère les artistes commençant par la saisie effectuée par l'administrateur :
**SELECT * FROM artist WHERE nom_artist LIKE :input%**

Une fois la requête exécutée, le résultat est stocké dans une variable puis encodée en JSON pour le fichier JS.

```
success: function(response) {
    let artists = JSON.parse(response)
    $('#res').html('');

    $(artists).each((_, x) => {
        $('#res').append(
            $('<a>', { 'href': 'artist-' + x.id_artist, 'class': 'side-link text-white
                $('<div>', { 'class': 'row align-items-center p-1 track' }).append(
                    $('<div>', { 'class': 'col-2 pe-0' }).append(
                        $('<div>', { 'class': 'ratio ratio-1x1' }).append(
                            $('<img>', { 'class': 'img-fluid object-fit-cover rounded-
                        ),
                    ),
                    $('<div>', { 'class': 'col-7 fs-12 text-black fw-semibold', 'text'
                    $('<div>', { 'class': 'col-3 fs-10' }).append(
                        $('<div>', { 'class': 'row' }).append(
                            $('<div>', { 'class': 'col-6 text-center' }).append(
                                $('<small>').append(
                                    $('<small>').append(
                                        $('<i>', { 'class': 'fa-solid fa-circle text-s
                                    )
                                )
                            ),
                            $('<div>', { 'class': 'col-6 text-center text-dark' }).app
                                $('<i>', { 'class': 'fa-solid fa-delete-left' })
                            )
                        )
                    )
                )
            )
        );
    });
    let delay = 0

    $('.fade').each((i, x) => {
        setTimeout(() => {
            $(x).addClass('show');
        }, delay)

        delay = delay + 30
    })
}
```

Le js va donc vider le contenu du menu latéral, une boucle va s'effectuée et construite l'HTML similaire pour chaque artiste récupérés.

Pour un meilleur rendu, la classe fade est ajouté à chaque artiste. Une fois la première boucle finie, une deuxième se lance avec un délai qui verra sa valeur augmenter à chaque itérations permettant l'affichage en fondu des résultats décalé par le délai incrémenter.



# SÉCURITÉ ET VEILLE INFORMATIQUE

## INJECTION SQL

PDO (PHP Data Objects) est une extension de PHP qui permet de communiquer avec une base de données. Pour protéger le site des injections sql, j'ai utilisé des requêtes préparées.

**exemple :**

```php
function updateInfoArtist($bdd, $nom, $desc, $artist) {
    $req = $bdd->prepare('UPDATE artist SET nom_artist = :nom, desc_artist = :descript WHERE id_artist = :artist');
    $req->execute([':nom'=>$nom,':descript'=>$desc,':artist'=>$artist]);
}
```

## HASHAGE DU MOT DE PASSE

Pour enregistrer les mots de passes en base de données, j'ai utilisé La fonction password_hash() qui prend deux arguments : le mot de passe en clair que je souhaite hasher, et un algorithme de hachage à utiliser. PASSWORD_DEFAULT est le deuxième paramètre que j'ai passé à cette fonction, il permet d'utiliser l'algorithme de hashage de php qui est le plus à jour.

```php
$mdpHash = password_hash($mdp, PASSWORD_DEFAULT);
```

→ `$2y$10$hhFLj9Rr3FikaLYsifDnPOokyB4JWz5x.PX6TIVMWyx...`

Puis pour vérifier le mot de passe lors de la connexion, j'utilise la fonction password_verify() qui vérifie si le mot de passe fourni correspond au hash stocké dans ma base de données. Si les deux correspondent, cela signifie que le mot de passe fourni est correct.

```php
if(password_verify($mdp, $userExist['password'])) {
```

## LES FAILLES XSS

La faille XSS (Cross-Site Scripting) est une vulnérabilité de sécurité courante qui peut permettre à un attaquant d'exécuter du code malveillant sur un site web ou d'obtenir des informations confidentielles de l'utilisateur. Ce type d'attaque permet d'injecter du code qui va modifier l'apparence ou le comportement du site du côté client.

Pour protéger le site contre ces attaques, j'ai traité les données provenant des formulaires avec la fonction de php htmlspecialchars().

```php
$mail = htmlspecialchars($_POST['mail']);
$mdp = htmlspecialchars($_POST['mdp']);
```

## INTRUSION VIA URL

pour protéger au mieux les URL j'ai créé des variables de SESSION d'identifiant et de rôle. L'admin se distingue donc des utilisateurs, et les utilisateurs non connectés n'auront pas accès à certaines pages ; sinon à des pages avec un message d'erreur.

```php
$_SESSION['admin'] = $admin['id_admin'];
$_SESSION['id_user'] = $userExist['id_user'];
```

Redirection Si l'utilisateur n'est pas connecté

```php
if(!isset($_SESSION['id_user']) && $_GET['page'] == 4) { // COMPTE
    header('location:accueil');
```

# VEILLE INFORMATIQUE

En tant que développeur web, je m'efforce de rester à jour avec les dernières tendances et les nouvelles avancées dans le domaine. Pour cela, je parcours régulièrement des sites spécialisés, des forums et des communautés en ligne pour découvrir de nouveaux outils et des Framework. J'essaie également de tester de nouvelles technologies pour voir comment je peux les utiliser. Et bien sûr, je ne néglige pas la documentation technique qui me fournit des instructions claires sur l'utilisation des outils, la résolution de problèmes et la personnalisation des fonctionnalités.

# JEU D'ESSAI

Ce jeu d'essai permet de vérifier la fonctionnalité d'écoute de musique proposé par le site, modification du volume du navigateur, clique sur les contrôle du lecteur ainsi que de l'ajout du titre lancé dans les favoris de l'utilisateur. Ce test a été réalisé en local.

## CAS N°1, L'UTILISATEUR EST DÉCONNECTÉ :

Afin de forcer la connexion d'un utilisateur pour écouter de la musique, chaque piste doit déclencher l'affichage de la modal de connexion si l'utilisateur est déconnecté

| DONNEES | ETAPE | ENTREE | ATTENDUS | REALISER | CHECK |
|---|---|---|---|---|---|
| CAS N°1 : utilisateur déconnecté | ETAPE N°1 | clic sur la piste | affichage de la modal de connexion | affichage de la modal de connexion | ✅ |

## CAS N°2, L'UTILISATEUR EST CONNECTÉ :

Une fois l'utilisateur connecté, il a accès à plusieurs listes de musiques présente sur différentes pages du site. Chaque piste est composé de la même structure HTML vue précédemment dans la section FRONTEND.

La première étape consiste à ce que l'utilisateur clique sur la piste voulu

| DONNEES | ETAPE | ENTREE | ATTENDUS | REALISER | CHECK |
|---|---|---|---|---|---|
| CAS N°2: utilisateur connecté | ETAPE N°1 | clic sur la piste | lancement de la piste | lancement de la piste | ✅ |

L'étape suivante correspond à l'affichage de la piste en cours dans le lecteur avec la mise a jour de la barre de progression et de la durée de la piste

| DONNEES | ETAPE | ENTREE | ATTENDUS | REALISER | CHECK |
|---|---|---|---|---|---|
| CAS N°2: utilisateur connecté | ETAPE N°2 | mise à jour du lecteur | Affichage du nom de la piste, de l'artiste et de la pochette de l'album de la piste en cours | Affichage du nom de la piste, de l'artiste et de la pochette de l'album de la piste en cours | ✅ |
| | | | Mise a jour de la barre de progression et de la durée de la piste | Mise a jour de la barre de progression et de la durée de la piste | ✅ |

La troisième étape correspond à l'entrée de la piste lancée dans la table historique_user afin que permettre à l'utilisateur de suivre ses écoutes récentes

| DONNEES | ETAPE | ENTREE | ATTENDUS | REALISER | CHECK |
|---|---|---|---|---|---|
| CAS N°2: utilisateur connecté | ETAPE N°3 | ajout du titre dans les écoutes récente | Ajout de l'id utilisateur et de l'id titre dans la table historique | Ajout de l'id utilisateur et de l'id titre dans la table historique | ✅ |
| | | | Présence du titre dans la playlist des écoutes récentes | Présence du titre dans la playlist des écoutes récentes | ✅ |

# RÉSOLUTION D'UN PROBLÈME

## CONVERT TIME AND UPDATE PROGRESS

Like Spotify and other music streaming plateforms, I wanted to make the progress bar interactive for the user.

First of all, I needed to prepare some functions and properties to set up the progress bar and the audio time.
My first problem encountered was to convert the audio time format (second to MM:SS). After some search on Google with the key words **"convert second to MM:SS js"** , I've found on **https://stackoverflow.com** a function for that :

```
function convertTime(t) {
    var e = Math.floor(t / 60);
    e < 10 && (e = "0" + String(e));
    var a = Math.floor(t % 60);
    return a < 10 && (a = "0" + String(a)), e + ":" + a
}
```

**e** is for minute, **a** for second

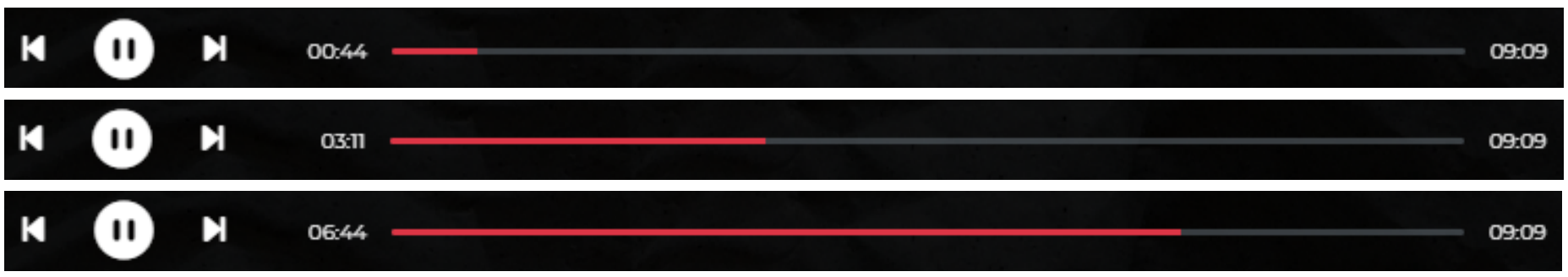
For each, if their value is below 10, it make a **concatenation with 0** to get the result of 01 : 01

With that, I was able to prepare two events linked to the Audio JS object, 'timeupdate'  and 'loadeddata':

```
$(audio).on("loadeddata", (() => {
    $("#duration").text(convertTime(audio.duration)), $("#currentTime").text(convertTime(0))
})),
$(audio).on("timeupdate", (() => {
    $("#currentTime").text(convertTime(audio.currentTime));
    const t = 100 * audio.currentTime / audio.duration;
    $("#playerProgress").css("width", t + "%")
})),
```

- **'loaddata'** is when an track source is loaded in the audio.source of  the **Audio JS object**. Audio.duration is set to the format MM:SS with the **function convertTime()** and  is applied to the text of **div#duration**. For the current time, the text of **div#curruenTime** is set to **00:00 with convertTIme(0).**

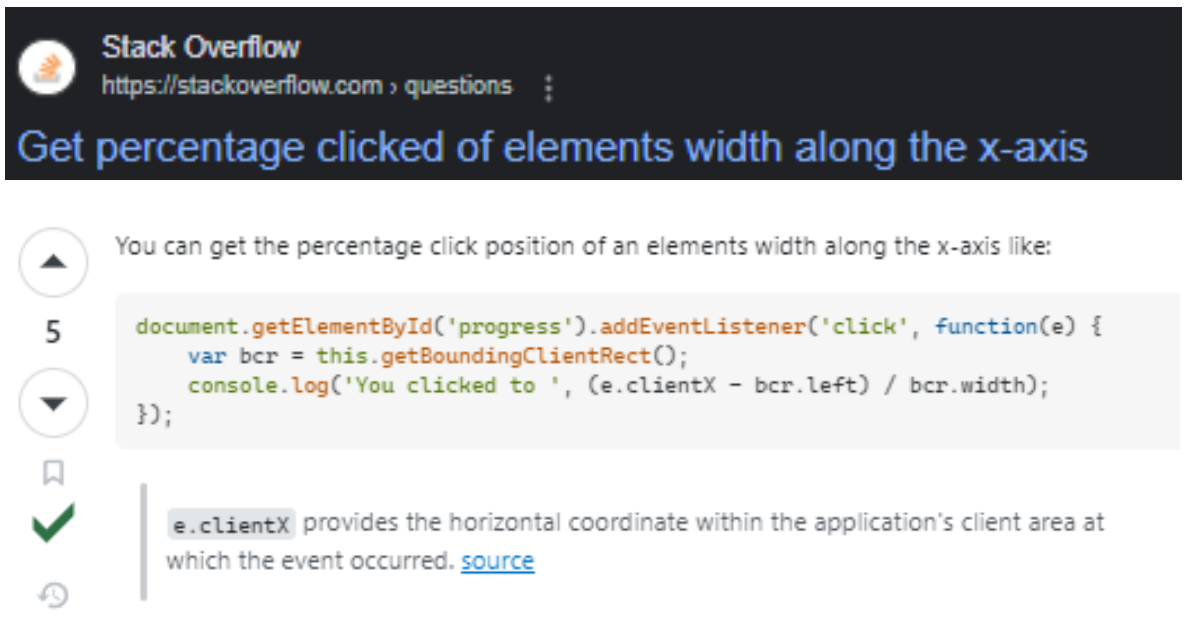

- **'timeupdate'** refreshes the text of the div#currentTime **while the Audio is playing the track with convertTime(audio.currentTime)**. For more visibility, I did a calculation to convert the progress of the track to percent : **(current time * 100) /  total duration**. Finally, the result of this is applied to the width of the **div#playerProgress**.




## UPDATE PROGRESS ON CLICK

Next, I wanted to give the possiblity to skip some part of the track by clicking on the progress bar and set the current time at the value of the width percent in relation to the position of the cursor on the bar. After many tries, I decided to look for help on the internet.

I seached **"get width percent on click"** on Google and I found an answer on **https://stackoverflow.com** :




The **Element.getBoundingClientRect()** method returns information about the size of an element and its position relative to **the window of the Web browser**.

After fiew search on **https://developer.mozilla.org/fr/** to understand this function, I adapted the code found on stackoverflow to my own :

```
$("#progress").on("click", (function (e) {
    var bcr = this.getBoundingClientRect();
    audio.currentTime = (e.clientX - bcr.left) / bcr.width * audio.duration
}));
```

- **e.clientX** refer to the position on the **x-axys of the cursor relative to the window**.
- **bcr.left** refet to the **position of the div#progress element**.
- By multiplying the ratio **between the click position and the total width by the total duration** of the track, you obtain the **new value of the current time**

Finally, with **the event 'timeupdate'** seen previously, this "onclick" event will also modify **the div#progress's width** for better clarity for the user.

# REMERCIEMENT

Je voudrais prendre un moment pour remercier ma responsable de formation Mme Godfroy et  la CCI de Metz de m'avoir fais confiance en acceptant ma candidature à cette formation et de m'avoir donné la chance d'apprendre le métier de développeur web et web mobile.
Je souhaite également remercier mes formateurs, Laurent Lagondet et Jugurta Akli pour leur enseignement et leurs conseils tout au long de la formation.

Je tiens également à exprimer ma gratitude envers Thibault Wingerter , mon formateur en anglais, et Tom Nauer, mon formateur en web design, pour leurs enseignements précieux. Grâce à eux, j'ai pu acquérir des compétences techniques et des connaissances approfondies en anglais et en web design.

Enfin, je voudrais exprimer ma reconnaissance envers le jury pour avoir pris le temps d'évaluer mon dossier professionnel. Merci de votre considération et de votre attention.