



**POLYTECHNIQUE  
MONTREAL**

UNIVERSITÉ  
D'INGÉNIERIE

# LOG2440 – Méthod. de dévelop. et conc. d'applic. Web

## Travail pratique 1

Chargés de laboratoire:

Charles De Lafontaine

Antoine Poellhuber

Justin Lachapelle

Hiver 2023

Département de génie informatique et génie logiciel

# 1 Objectifs

Le but de ce travail pratique est de vous familiariser avec les langages HTML et CSS.

Plus particulièrement, vous allez élaborer la structure de pages web tout en respectant la structure sémantique apportée par HTML. De plus, vous allez mettre en place l'interaction entre les différentes pages à l'aide des hyperliens. Finalement, vous aurez à reproduire les maquettes des différentes pages du site web qui vous sont fournies en annexe de ce document.

Vous allez vous familiariser avec tests de bout en bout avec la librairie Nightwatch.

## 2 Introduction

Tout au long de la session, vous aurez à réaliser un site web similaire à une plateforme de musique que vous devez bien connaître : [Spotify](#). Pour y arriver, les différents travaux pratiques à réaliser dans le cadre du cours intégreront plusieurs notions pour obtenir un site web complètement fonctionnel à la fin de la session.

Tout comme un site web, les sites de musique respectent une sémantique particulière. En effet, les utilisateurs recherchent des chansons qui sont contenues dans des *playlists*. Toutefois, il n'est pas rare de voir la même chanson dans plusieurs playlists différentes. On intégrera dans notre site une partie administrative permettant d'ajouter et de supprimer des playlists.

## 3 Tests automatisés

Vous remarquerez que plusieurs tests pour chacune de vos pages, ainsi que l'entête/pied de page vous sont fournis dans le répertoire `tests/e2e`. Ces tests vous permettront de valider votre travail et le contenu de vos pages HTML. Il est fortement recommandé de lire les tests et leur description avant d'écrire votre code pour vous aider avec le résultat final attendu.

Pour lancer la suite de tests, vous n'avez qu'à aller à la racine de votre répertoire `site-web` avec un terminal et de lancer la commande « `npm ci` » pour installer les librairies et outils nécessaires. Par la suite, entrez la commande « `npm run e2e` » afin de lancer les tests fournis. Évidemment, vous devrez au préalable avoir installé l'environnement d'exécution [NodeJS](#).



## Configuration des tests

---

Afin de pouvoir interagir avec le navigateur, vous utiliserez un [WebDriver](#), plus spécifiquement, [ChromeDriver](#) qui permet d'interagir avec le navigateur Chrome. Vous aurez à modifier le chemin du WebDriver dans la configuration du fichier `nightwatch.json`, notamment le paramètre `server_path` dépendamment de votre système d'exploitation :

MacOS : `"node_modules/chromedriver/lib/chromedriver/chromedriver"`

Linux : `"node_modules/.bin/chromedriver"`

Windows : `"node_modules/chromedriver/lib/chromedriver/chromedriver.exe"`

---

Lors du premier lancement de la commande «`npm run e2e`», seulement les tests pour la page *À propos* seront lancés et certains tests pour l'entête et pied de page échoueront puisque l'HTML pour ces deux parties n'est pas encore fait. Une fois que vous avez écrit le code HTML pour les sections 4.2 et 4.3, relancez les tests pour valider votre travail. Lisez bien le code dans `shared.js` pour mieux comprendre le résultat attendu par les tests.

Vous pouvez utiliser la même stratégie pour les autres pages de ce travail pratique. Pour lancer les tests d'une page spécifique, modifiez la valeur de la propriété `filter` dans `nightwatch.json`. **Assurez-vous d'enlever le filtre pour votre remise finale.**

Note : les tests de l'entête et le pied de page seront exécutés pour chaque page du site.

## 4 Travail à réaliser

Pour ce travail pratique, vous devez d'abord élaborer la structure des pages HTML qui composeront votre site web. La structure attendue pour une page est d'ailleurs illustrée à la figure 1. Cette structure devra être définie à l'aide des balises sémantiques de HTML telles que les balises «`header`», «`nav`», «`main`», et «`footer`». Les sous-sections qui suivent décrivent les différentes pages à réaliser.

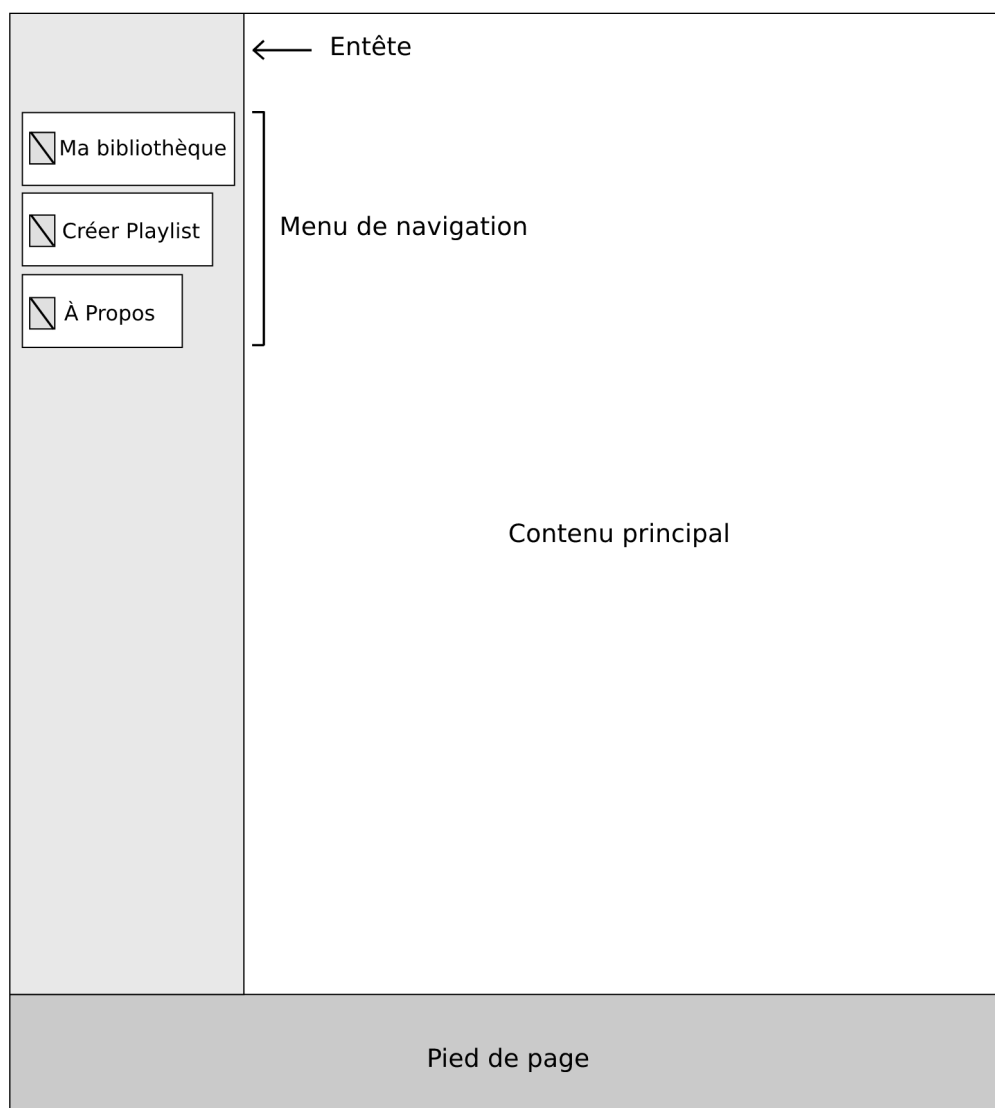


FIGURE 1 – Maquette d’une page web

Ensuite, à partir de votre code HTML, vous aurez à mettre en forme les différentes pages du site web à l’aide du langage CSS.

### **Avertissement**

L’utilisation de bibliothèques ou de frameworks CSS est interdite (p. ex. [Bootstrap](#)), sauf en ce qui concerne la bibliothèque [Font Awesome](#). Puisque l’objectif de ce travail pratique est de vous familiariser avec le langage CSS, il est important que vous maîtrisiez l’ensemble des éléments du langage. Font Awesome est nécessaire, puisque le site web utilise plusieurs icônes fournies par cette bibliothèque.

Le fichier `styles.css` contiendra l'ensemble de votre code CSS de vos pages web et se trouve dans le répertoire `/assets/css/`. Également, afin de vous assurer que la bibliothèque Font Awesome et que votre style soient pris en compte dans vos pages, vous devez inclure les lignes de code suivantes entre les balises `head` de vos fichiers HTML :

```
<link rel="stylesheet" type="text/css"
href="https://use.fontawesome.com/releases/v6.1.2/css/all.css">
<link rel="stylesheet" type="text/css" href="./assets/css/styles.css">
```

Avant de débiter, assurez-vous d'avoir accès à votre dépôt GitHub associé à ce travail pratique. Le dépôt contient les fichiers HTML et CSS à compléter ainsi que des images à inclure dans les pages.

Il est à noter que lorsque des dimensions ou des éléments ne sont pas spécifiés dans le présent document, ceux-ci sont à votre discrétion. Vous devez seulement vous assurer que votre affichage **soit similaire** aux maquettes des pages en annexe. La police à utiliser pour les pages est *Roboto*. La couleur par défaut du texte est noire. Si vous le souhaitez, il vous est possible de changer les couleurs définies comme variables globales CSS. Les sous-sections suivantes décrivent les éléments à réaliser pour ce travail pratique.

Certaines règles CSS vous sont déjà données pour vous aider à partir du bon pied. Notez que des **TODOs** sont placés dans certains sélecteurs pour indiquer que vous devez les compléter avec les bonnes règles CSS pour obtenir le visuel voulu.

## 4.1 Affichage en grille

Les sections de l'Entête, le Pied de page et le Contenu principal de chaque page sont affichés sous la forme d'une grille CSS tel que présenté dans la figure 1. Cette grille utilise des zones nommées qui vous sont déjà fournies dans `styles.css` : `nav-bar`, `main-area` et `playing-bar`. Vous devez lier les bons éléments à chacune de ces zones.

## 4.2 Entête

Chacune des pages du site web doit contenir la même entête. Les éléments contenus dans l'entête doivent se retrouver dans une balise `<header>`. L'entête est composé d'un élément de navigation (`<nav>`) qui inclut les liens suivants dans une liste sans ordre particulier (`<ul>`) :

- Bibliothèque (`index.html`)
- Créer une playlist (`create_playlist.html`)
- À Propos (`about.html`)

La largeur de l'entête correspond à la largeur maximale des items qui se trouve à l'intérieur de celui-ci. Comme il est possible à la figure 19, les liens se trouvant dans la barre de navigation doivent avoir une icône et être suivi par leur description. Les icones utilisées proviennent de la librairie **Font Awesome** et sont ajoutées à travers l'attribut `class`. Par exemple, le lien vers la Bibliothèque possède l'icone suivante : `<i class="fa fa-music"></i>`. Les liens suivants ont les icones `fa-plus` (Créer Playlist) et `fa-info-circle` (À Propos).

Les éléments contenus dans la barre de navigation doivent aussi avoir un `padding`. La couleur du texte doit être celle de la variable `--text-color`. Les coins des items doivent être arrondis. Par défaut, les items ne sont pas soulignés. La couleur d'arrière plan de ces items doit être blanc. Lorsqu'un item est sélectionné, sa couleur d'arrière plan doit être celle de la variable `--selected-page`. Il est aussi nécessaire que l'élément devienne souligné. Lorsqu'un item de la barre de navigation est survolé par la souris, il faut que cet item ait le même comportement que lorsqu'un item est sélectionné. La figure 2 présente l'item "Ma Bibliothèque" sélectionné. Lorsqu'un item est survolé, il doit avoir les mêmes propriétés.

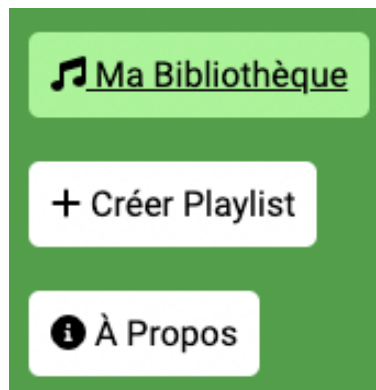


FIGURE 2 – Liens de navigation présent dans la bare de navigation

### 4.3 Pied de page

Tout comme pour l'entête, chaque page doit contenir un pied de page. Ce dernier doit contenir les informations de votre équipe, soit le nom et la matricule de chaque membre. Le pied de page sera aussi le conteneur pour les contrôles de musique pour la page `playlist.html`.

Deux éléments doivent être présents à l'intérieur du pied de page. Un premier élément situé à gauche doit contenir un court paragraphe contenant un lien vers la page `index.html`. Le second élément est présent à droite et contient les informations (nom et matricule) des membres de votre équipe. Cet élément doit avoir l'attribut `id="creators"`.

Lorsque la page change de proportions, les éléments doivent rester aux extrémités.

## 4.4 Page "À Propos" (`about.html`)

L'implémentation du contenu principal de cette page vous est déjà fournie. Vous n'aurez qu'à respecter la forme de la page et son style. Un exemple de la page est présent en annexe à la figure 23. Comme il est possible de voir, le titre se distingue par la couleur de son fond d'écran utilisant la variable `--highlight`. Il est aussi possible de voir qu'un `padding` est présent et que le titre possède des coins arrondis. Le poids du titre doit être similaire à celui présent. Un exemple de titre est montré dans la figure 3.



FIGURE 3 – Titre présent sur la page "À propos"

Une fois les sections *header* et *footer* implémentées correctement et que le contenu de cette page est respecté, tous les tests dans les fichiers `shared.js` et `about.js` devraient passer.

## 4.5 Page "Ma Bibliothèque" (`index.html`)

Un exemple du résultat désiré est présenté en annexe à la figure 21. Comme le *header* et le *footer* sont déjà implémenté, il ne suffit que de s'attarder au contenu de la page. **Les tests vous fourniront beaucoup d'indications quant aux éléments à ajouter aux pages.**

### 4.5.1 Outil de recherche

Le premier élément à implémenter consiste à un outil de recherche représenté par un formulaire (`<form>`). Un exemple de cet outil de recherche vous est présenté à la figure 4. La barre de recherche possède une largeur fixe. Une marge et un `padding` sont aussi présents pour créer un espace à l'intérieur de la boîte de texte. Le champ de saisie de texte et le bouton de recherche ne doivent pas être collés au sommet à gauche de la page comme ils le seraient par défaut. Un espace doit être présent à gauche de l'outil et au-dessus de celui-ci. L'icône du bouton utilise la classe suivante : `fa-search`.



FIGURE 4 – Outil de recherche à implémenter

### 4.5.2 Titres

Comme il est possible de voir à la figure 21 de l'annexe, il sera nécessaire d'utiliser deux titres. Le style utilisé est similaire à celui décrit pour le titre présent dans la page "À propos". La seule différence est que les titres de cette page devront être de la largeur de l'écran. Un exemple est présenté aux figures 5 et 6.



FIGURE 5 – Titre "Mes Playlists"



FIGURE 6 – Titre "Mes Chansons"

### 4.5.3 Playlist individuelle

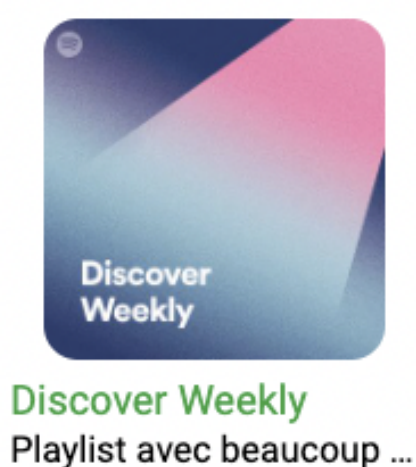
**Référez-vous au fichier `playlists.js` pour les informations des playlists.**

En réalité, chaque playlist est une balise `<a>` qui permet de rediriger l'utilisateur à la page `playlist.html`. Cette balise contient l'ensemble des informations de la playlist. Une image de 150px \* 150px avec des coins arrondis est présente dans un `<div class="playlist-preview">`. Par la suite, un premier paragraphe de description doit être d'une taille large et utiliser la couleur de la variable `--main-color`. Le second paragraphe est la description de la playlist et doit avoir la couleur de la variable `--text-color`. La description peut être trop longue pour être affichée au complet. Pour résoudre ce problème, une largeur doit être définie. Pour éviter que le reste de la description s'affiche, l'attribut `overflow` permet de cacher le texte excédentaire. Pour afficher les trois petits points utilisés qui remplacent la description excédentaire, l'attribut `text-overflow` pourrait permettre d'obtenir ce résultat. Un exemple de playlist est présenté à la figure 7a.

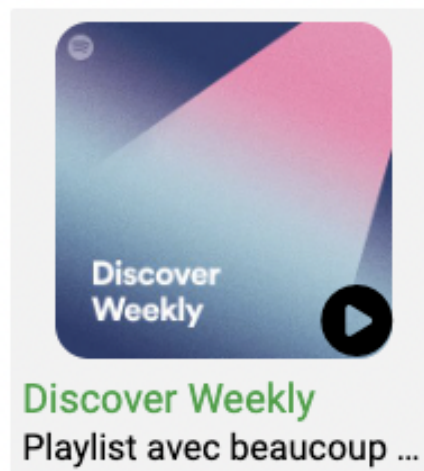
Lorsque la souris survole la playlist, la couleur de son arrière-plan devient celle de la variable `--hover-playlist`. Ceci fait également apparaître une icône sur l'image de la playlist. L'icône est placée dans le coin inférieur droit de l'image et possède les classes suivantes : `class="fa fa-2x fa-play-circle hidden playlist-play-icon"`.

Un exemple de playlist survolée par une souris est présent à la figure 7b. Une comparaison entre la playlist survolée et la playlist à son état initiale est présentée en annexe à la figure 7.





(a) Playlist non-survolée



(b) Playlist survolée par la souris

FIGURE 7 – Effet du survol de la souris sur une playlist

#### 4.5.4 Affichage des playlists

L’affichage des playlists dépend de la largeur de l’écran. Lorsque la largeur de l’écran est moins de 800px, les playlists sont affichées sous la forme d’une grille où seulement deux éléments sont présents par ligne. Un exemple de ce comportement est présenté à la figure 8.

Lorsque la taille de l’écran est supérieure à 800px, les items sont placés un à la suite de l’autre avec un affichage `flex`. Lorsque la largeur de l’écran devient trop petite pour l’ensemble des éléments d’une ligne, un élément retournera à la ligne inférieure à l’aide de l’attribut `flex-wrap: wrap`. Un exemple de ce comportement est présenté à la figure 9.

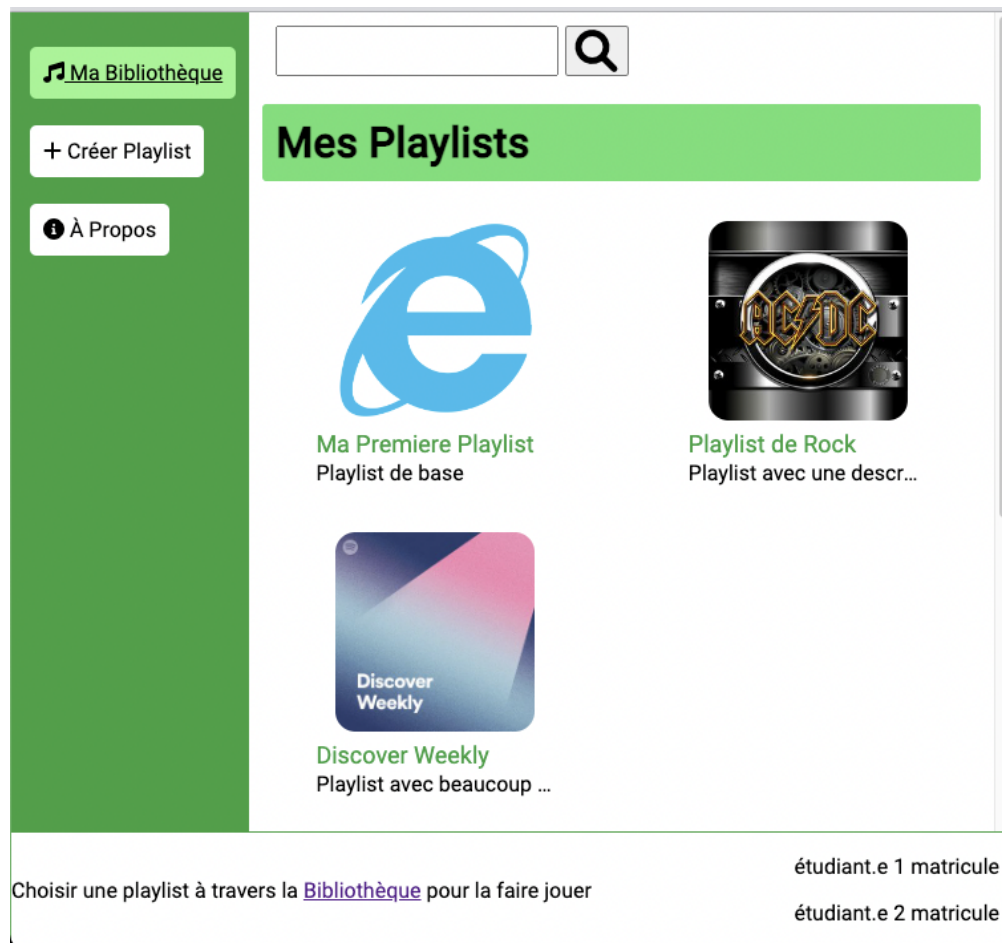
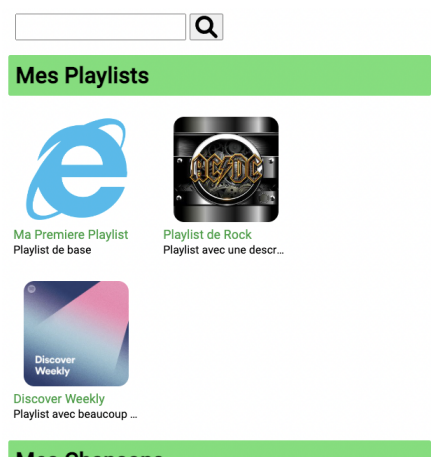
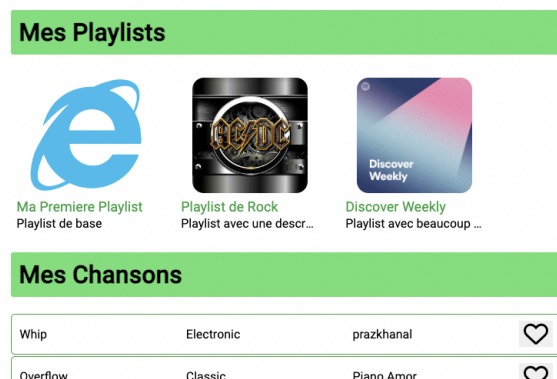


FIGURE 8 – Impact de l'utilisation de @media pour faire l'affichage des playlists



(a) Retour à la ligne d'une playlist en raison du manque d'espace



(b) Ensemble de playlists avec espace suffisant

FIGURE 9 – Comportement attendu par l'utilisation de flex pour afficher les playlists

#### 4.5.5 Affichage des chansons

Référez-vous au fichier `songs.js` pour les informations de chaque chanson.

Les chansons sont placées dans une colonne flexible. Le contour de la boîte de chanson (`class="song-item"`) est arrondi et de couleur verte. Le contenu de la boîte doit être espacé équitablement et s'adapter à la largeur de la page. Un exemple est présenté à la figure 10. Lorsque la souris survole une chanson, le contour doit devenir plus épais. Un exemple est présenté à la figure 11. Certaines chansons doivent avoir un bouton avec une icône de coeur plein (`class="fa fa-2x fa-heart"`) et d'autres non (`class="fa-regular fa-2x fa-heart"`) en fonction de l'attribut `liked`.






Whip	Electronic	prazkhanal	
Overflow	Classic	Piano Amor	
Intrigue Fun	Jazz	Music Town	
Bounce	Electronic	Coma-Media	
Summer Pranks	Rock	Lemon Music Studio	

FIGURE 10 – Chanson non-survolée






Whip	Electronic	prazkhanal	
Overflow	Classic	Piano Amor	
Intrigue Fun	Jazz	Music Town	
Bounce	Electronic	Coma-Media	
Summer Pranks	Rock	Lemon Music Studio	

FIGURE 11 – Chanson survolée par la souris

## 4.6 Page "Créer Playlist" (`create_playlist.html`)

Un exemple de cette page est présenté dans l'annexe à la figure 24. La page est composée d'un formulaire (`<form>`) qui contient 2 parties (`<fieldset>`) qui ont une `<legend>` chaque.

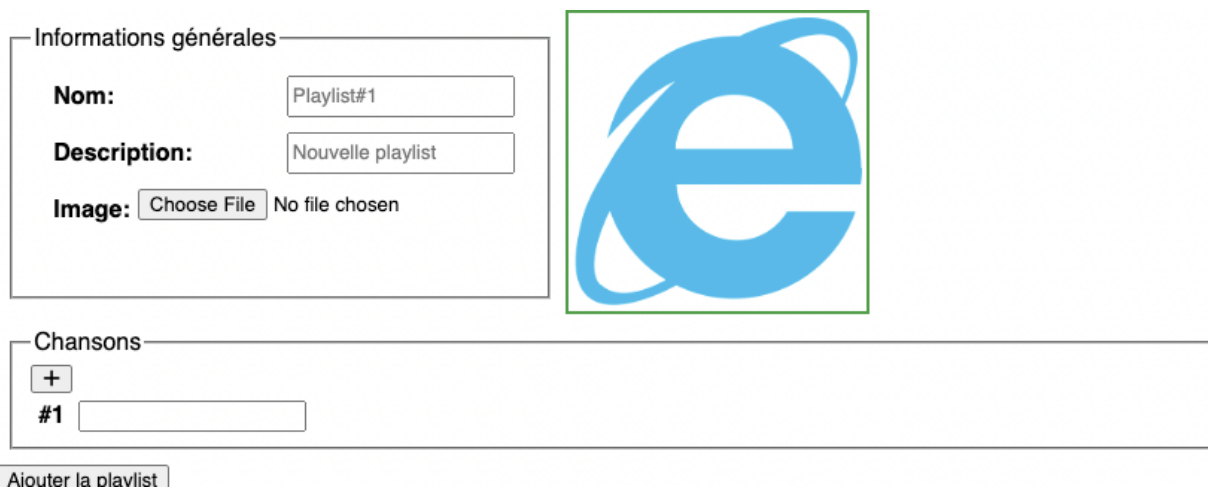
La première partie "Informations générales" doit contenir des champs de saisie (`<input>`) pour le nom, description et image de la nouvelle playlist. Ces éléments devront être justifiés de la même manière que cette présentée dans la figure 12. Les éléments présents à l'intérieur du formulaire font l'usage de `padding` et de `margin`. L'image utilisée doit avoir une largeur de 200px par 200px. L'image doit avoir une bordure verte. Les valeurs "Playlist #1" et "Nouvelle playlist" doivent être utilisées dans les champs de texte par défaut.

Informations générales

**Nom:**

**Description:**

**Image:**  No file chosen



Chansons

FIGURE 12 – Formulaire utilisé dans la création d'une playlist

Dans la deuxième partie, "Chansons", lorsqu'un utilisateur clique dans le champ de sélection (`<input>` de type `select`) de chanson, une liste déroulante de suggestions de chansons devrait apparaître comme montré à la figure 13. Un élément `<datalist>` avec l'id `song-dataList` doit contenir des éléments `<option>` qui sont les suggestions de chansons. Référez-vous à la documentation de l'attribut `list` de l'élément `<input>`. pour lier le champ de sélection à la liste d'options. Un bouton ayant les classes `fa fa-plus` et l'id `add-song-btn` est présent avant la liste des chansons choisies.

Finalement, un `<input>` de type `submit` est présent à la fin du formulaire pour sa soumission. Le texte affiché est *Ajouter la playlist*.

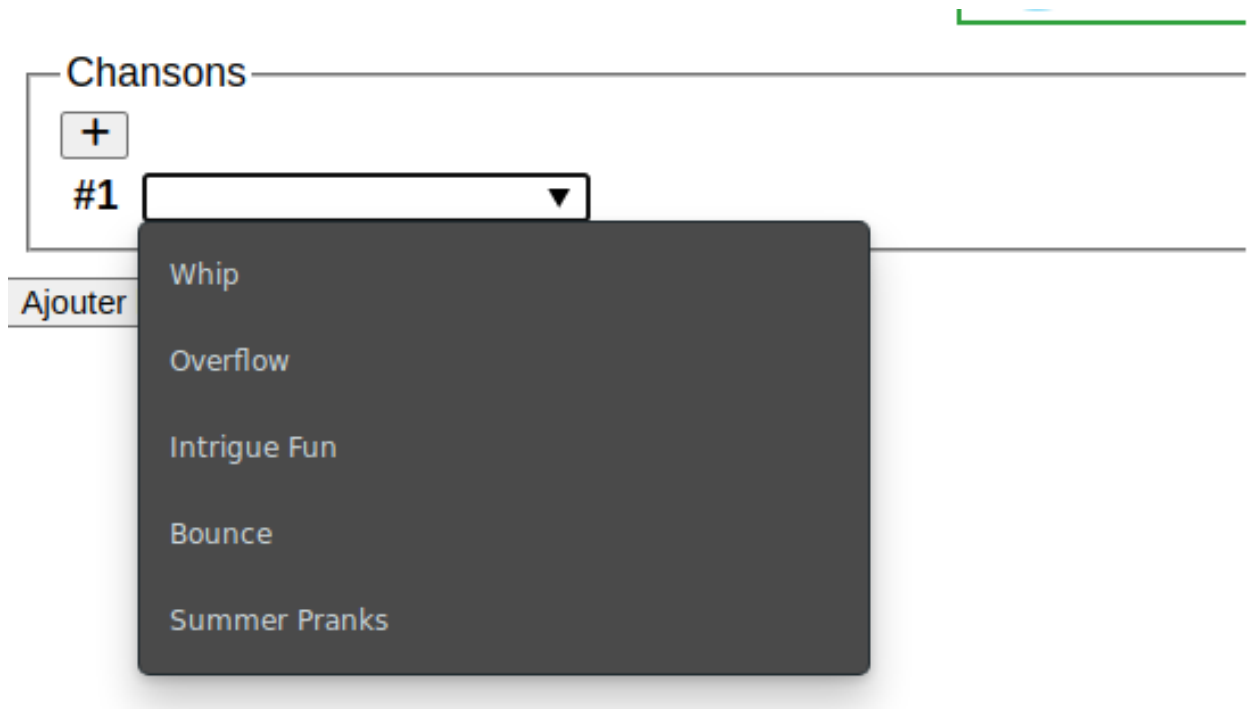


FIGURE 13 – Suggestion de chansons à ajouter à une playlist

## 4.7 Page "Playlist de Rock" (`playlist.html`)

Pour avoir accès à cette playlist, il doit être possible de cliquer sur l'une des playlists présentes dans la page `index.html` se trouvant dans la section "Mes Playlists". Un exemple de cette page est présenté dans l'annexe à la figure 22.

Dans le cadre de ce travail, seulement la page pour la playlist "Playlist de Rock" sera implémentée. Référez-vous au fichier `playlist.js` pour les informations de la playlist.

### 4.7.1 Section au sommet de la page

Le sommet de la page devrait avoir la même apparence que celle montrée à la figure 14.



FIGURE 14 – Entête présent dans la page playlist

Au sommet de la page doit se trouver l'image de la playlist. Cette image doit avoir une taille de 80px par 80px. Un titre doit être présent. Il s'agit du même format que celui utilisé

dans la page "À Propos". Finalement, à droite complètement devrait se trouver l'icône d'un crayon(`class="fa fa-2x fa-pencil"`). Lorsque le crayon est cliqué, sa couleur devrait passer de mauve à rouge et l'utilisateur doit être redirigé vers la page `create_playlist.html`. Un exemple est présenté à la figure 15.

Ces trois éléments sont dans un header avec l'id `playlist-header` et la classe `flex-row`.



(a) Icône de crayon dans son état initial



(b) Icône de crayon lorsque cliqué

FIGURE 15 – Effet de cliquer sur l'icône de crayon

#### 4.7.2 Liste des chansons

La liste des chansons possède le même comportement que la liste de chansons présentée à la page `index.html`. La seule différence est l'ajout d'un index avant chaque chanson dans un élément `<span>`. Un exemple des chansons attendues est présenté à la figure 16

1	Whip	Electronic	prazkhanal	♡
2	Bounce	Electronic	Coma-Media	♡
3	Summer Pranks	Rock	Lemon Music Studio	♥

FIGURE 16 – Exemple des chansons attendues dans la playlist

#### 4.7.3 Barre de lecture de la musique

Finalement, le dernier élément à implémenter est la barre de lecture de la musique. Celle-ci devrait avoir une apparence similaire à celle présente dans la figure 17 et se trouve dans l'élément `<footer>` de la page dans un élément `<div id="controls" class="flex-column">`.



FIGURE 17 – Paramètres présents dans la barre de lecture

La barre de lecture est composée de 2 `<section>` : la première, ayant l'id `buttons-container` contient les boutons de contrôle et la deuxième, ayant l'id `timeline-container` contient la barre de progrès et le temps. Les 2 sections sont des boîtes flexibles en rangée.

Chaque bouton possède les classes communes suivantes : `class="control-btn fa fa-2x"`. Chaque bouton possède en plus une classe différente pour l'affichage de l'icône et un id. Ces éléments sont, dans l'ordre d'apparition, les suivants :

```
class="fa-arrow-left" id="previous"
class="fa-play" id="play"
class="fa-arrow-right" id="next"
class="fa-shuffle" id="shuffle"
class="fa-volume-high" id="mute".
```

La barre de progrès contient un élément `<span>` avec le temps de début (0 :00), un élément `<input id="timeline">` de type `range` et un élément `<span>` avec le temps de fin (5 :00).

Il est possible de survoler les boutons présents dans la barre de lecture et avoir une réaction visuelle. Lorsque les boutons sont survolés, ils doivent s'agrandir et devenir mauves et revenir à leur état normal par la suite. Un exemple de cela est montré à la figure 18.

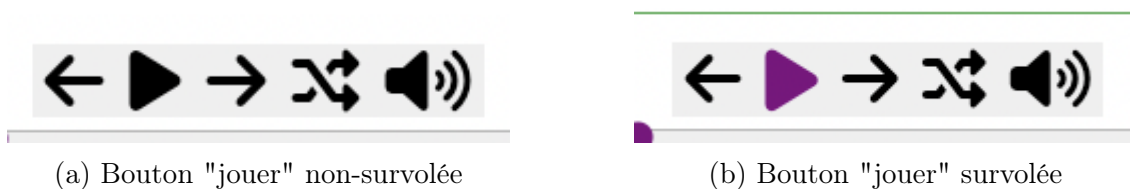



FIGURE 18 – Effet du survol de la souris sur les boutons de la barre de son

## Conseils pour la réalisation du travail pratique

---

1. Lisez les tests fournis avant d'écrire votre code.
  2. Implémentez les éléments demandés un à la fois dans l'ordre indiqué dans l'énoncé.
  3. Jetez un coup d'œil aux différentes [balises sémantiques](#) de HTML5 afin de vous assurer que les éléments que vous utilisez soient bien adaptés.
  4. Effectuez une validation de vos formulaires grâce aux attributs de HTML5 (voir la section « *Input Restrictions* » de la page [HTML Input Types](#)).
  5. Identifiez les éléments semblables dans l'énoncé afin de définir des classes CSS génériques et réutilisables.
  6. Utilisez les [boîtes flexibles](#) (*flexbox*) pour mettre en forme des lignes/colonnes. Vous avez les classes `flex-row` et `flex-column` fournies pour vous aider.
  7. Utilisez les outils de développement de votre navigateur web pour vous aider à déboguer votre code CSS (raccourci ).
  8. Soyez consistants dans vos conventions de codage (voir [ce guide de codage](#)).
  9. Exécutez les tests fournis souvent afin de valider votre code.
- 

## 5 Remise

Voici les consignes à suivre pour la remise de ce travail pratique :

1. Le nom de votre entrepôt Git doit avoir le nom suivant : **tp1\_\_matricule1\_\_matricule2** avec les matricules des 2 membres de l'équipe.
2. Vous devez remettre votre code (*push*) sur la branche **master** de votre dépôt git. (pénalité de 5% si non respecté)
3. Le travail pratique doit être remis avant **23h55**, mardi le **7 février**.

**Aucun retard** ne sera accepté pour la remise. En cas de retard, la note sera de **0**.

Le navigateur web **Google Chrome** sera utilisé pour tester votre site web.



## 6 Évaluation

Vous serez évalués sur la qualité de votre structure HTML, ainsi que sur la qualité de votre mise en forme en CSS. Plus précisément, le barème de correction est le suivant :

Exigences	Points
Aspect global des pages	
Respect des exigences fonctionnelles de l'énoncé	6
Formulaires	
Utilisation adéquate des éléments d'un formulaire	1
Validation des données entrées	1
Structure sémantique du code HTML	
Utilisation adéquate des balises sémantiques	1
Validité de la sémantique de la page	1
Respect des exigences pour la mise en forme CSS	
Entête et Pied de page	1
Page <code>about.html</code>	1
Page <code>create_playlist.html</code>	2
Page <code>index.html</code>	2
Page <code>playlist.html</code>	2
Qualité et clarté du code HTML et CSS	2
Total	20

L'évaluation se fera à partir de la page HTML initiale, soit `index.html` (qui correspond à la page d'accueil). À partir de cette page, le correcteur devrait être capable de consulter toutes les autres pages de votre site web. Tous les tests fournis doivent obligatoirement passer.

Ce travail pratique a une pondération de 5% sur la note du cours.

# Annexe



FIGURE 19 – Exemple de barre de navigation



FIGURE 20 – Maquette du pied de page

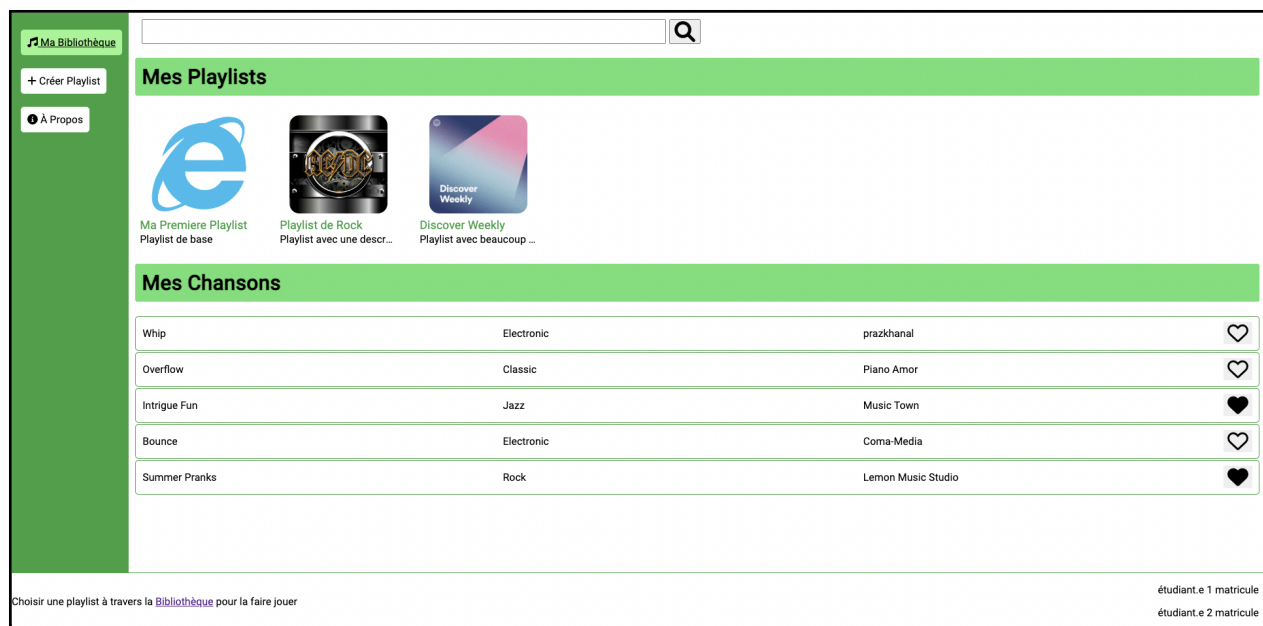


FIGURE 21 – Maquette de la page "Ma Bibliothèque"



FIGURE 22 – Maquette de la page "Playlist"

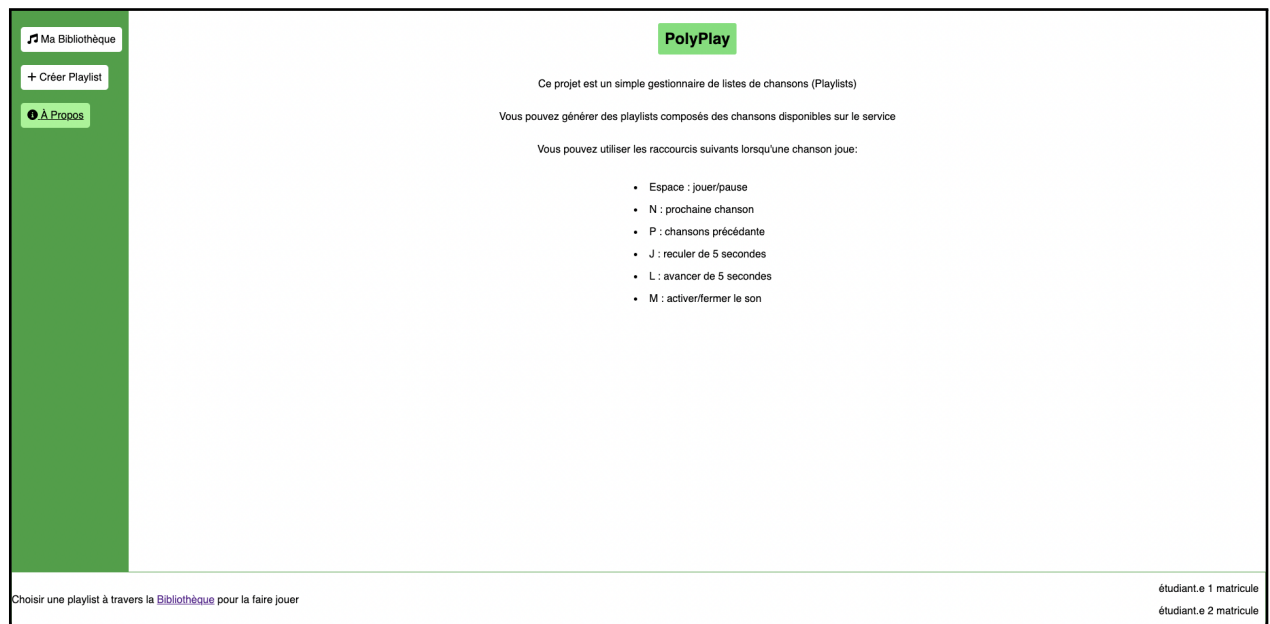


FIGURE 23 – Maquette de la page "À propos"

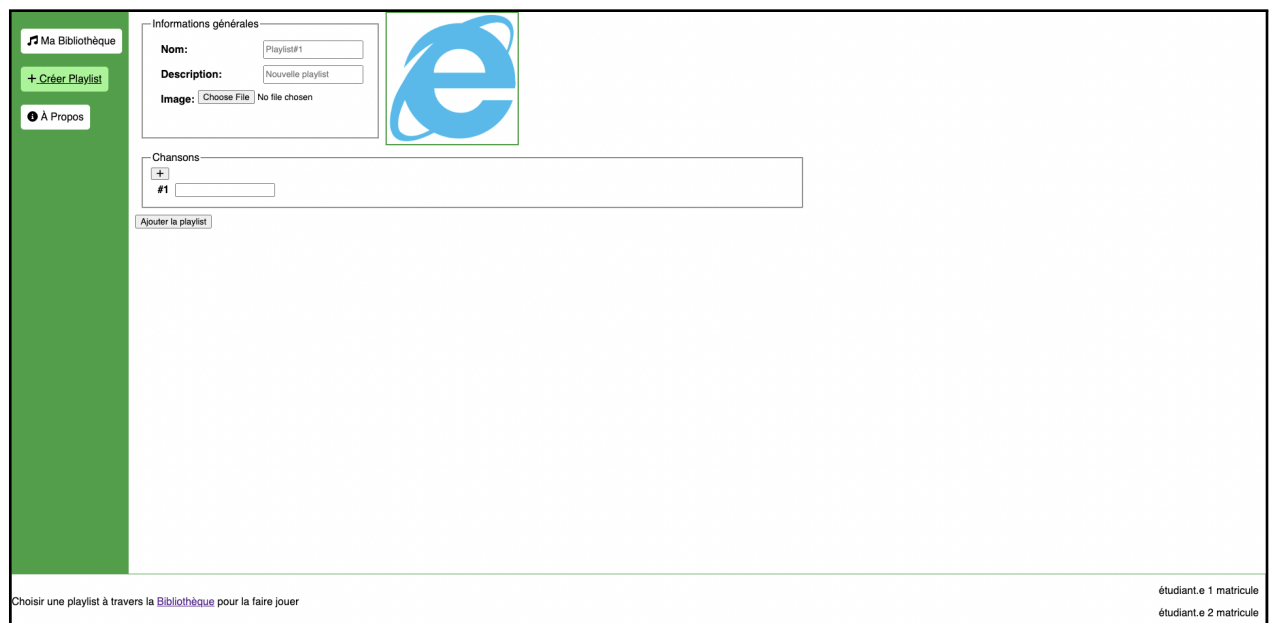


FIGURE 24 – Maquette de la page "Créer Playlist"