

Documentation Projet S2

par Léopold Lapendery et William Fontaine

Sommaire :

1. IHM : Interface Homme-Machine (XAML, WPF)	3-37
a. Contexte	3-5
b. Persona / user stories	6-11
i. Thomas Dupont	6-7
ii. Camille Rose	8-9
iii. Huguette Martin	10-11
c. Sketchs	12-16
i. Accueil	12
ii. Recherche	13
iii. Morceaux favoris	13
iv. Artistes	14
v. Page artiste détaillée	14
vi. Albums	15
vii. Page albums détaillée	15
viii. Page playlist détaillée	16
d. Storyboards	17-27
i. Se connecter	17-18
ii. Recherche de titre/album/artiste/genre	19-20
iii. Accès aux titres favoris	20-21
iv. Jouer une musique	21-22
v. Boutons utilitaires	22
vi. Accès aux artistes/albums favoris	23
vii. Accès aux playlists	24-25
viii. Création d'une playlist	25-26
ix. Ajout d'une musique aux favoris/playlist	27
e. Diagramme de cas d'utilisation	28-35
f. Considération ergonomique	36
g. Prise en compte de l'accessibilité	37
2. Conception et Programmation Orientées Objets (C#, .NET)	38-45
a. Diagramme de paquetage	38
b. Diagramme de classes	39-45
i. Diagramme général	39-40
ii. Diagramme de gestion des musiques	41-43
iii. Diagramme du player	43
iv. Diagramme de recherche	44

v.	Diagramme du tri.....	44-45
c.	Diagramme de séquence.....	46
3.	Projet Tutoré S2	47-50
a.	diagramme de paquetage mettant en avant la partie persistance.....	47
b.	diagramme de classes mettant en avant la partie persistance.....	48-49
c.	diagramme de classes sur votre (vos) partie(s) ajoutée(s).....	49-50

IHM : Interface Homme-Machine (XAML, WPF)

Contexte

Introduction

Vous avez déjà eu l'air d'une musique dans la tête sans arriver à vous souvenir de son titre ? Aujourd'hui, de multiples plateformes vous permettent de retrouver ce que vous cherchez, en cherchant par exemple un artiste ou un album. Des plateformes comme YouTube, Deezer, Spotify, ... Avec chacune leur particularité. Par exemple, YouTube permet d'avoir les commentaires des utilisateurs ou encore un système de notation. Spotify et Deezer sont des plateformes limitées à la musique et récemment étendues aux podcasts.

Elles permettent d'écouter vos musiques où vous le voulez sans connexion grâce à un système d'abonnement. Ces plateformes prennent plus l'apparence de plateformes de streaming de musique qu'autre chose ...

Elles ne présentent en général ni leurs artistes, ni leurs œuvres. Désaccordé est là pour que vous puissiez connaître les moindres détails sur ces derniers et leur travail.

Désaccordé est une sorte de plateforme musicale complète, fusionnant les fonctionnalités de base d'une plateforme de streaming de musique et des informations clés sur les artistes, pour ne pas à avoir à faire de multiples recherches lorsque vous écoutez une musique.

Ces applications de musiques vous permettent d'écouter les musiques de vos choix où vous le voulez grâce à une simple recherche. Il suffit de taper le titre de votre musique, son artiste, ou encore son album.

Dans les applications spécialisées dans la musique, il est aussi possible de filtrer les musiques par genre musical et d'écouter des musiques jusque-là inconnues pour vous les unes après les autres. Ce système vous permet donc d'écouter des musiques de vos genres préférés et d'en découvrir par la même occasion.

Ces applications vous permettent aussi de créer des playlists et de les personnaliser entièrement à tout moment pour qu'elles vous ressemblent le plus et pour répondre le mieux possible à vos envies du moment.

Si vous cherchez des informations au sujet des artistes ou albums de vos choix, il faut donc bien souvent aller chercher sur des sites tiers comme Wikipédia ou autre.

C'est pour cela que l'on vous propose aujourd'hui Désaccordé. C'est un registre musical regroupant toutes sortes de morceaux de musique. Lorsque l'on fait une recherche de titre, si le morceau recherché n'appartient à aucun album, il apparaît alors seul dans les suggestions. Si, au contraire, il appartient à un album, la recherche affichera d'abord le titre recherché puis les autres morceaux appartenant à cet album.

En plus de tout cela, lorsque l'on fait une recherche d'artistes, les pages de ces derniers répertorient leurs titres phares puis leurs albums et potentiellement leurs titres parus seuls. On pourra obtenir l'ensemble des musiques d'un artiste sur sa page ainsi que tous les albums qu'il aura réalisés. Chaque utilisateur pourra ajouter un morceau, un album ou encore un artiste à ses favoris.

Résumé des fonctionnalités

Pour commencer, la recherche se fait selon le nom du morceau, celui de l'album, l'artiste ou le genre de la chanson. Ainsi, l'application sera simple à comprendre pour toucher le plus grand nombre de personnes. On peut également faire des recherches selon le genre musical que l'on veut écouter.

Le résultat des recherches se fait par rubrique. On trouve différentes rubriques, par exemple des rubriques titres, albums ou encore artistes.

Chaque album aura une page dédiée détaillant tous les morceaux le composant ainsi que les prix qu'il a reçus et une petite description de l'album, ce qui la différencie des autres applications du genre.

Parlons des artistes. Chaque artiste aura également sa page détaillée. On y trouvera ses morceaux les plus écoutés et/ou mieux notés, une liste de tous ses albums ainsi que sa biographie détaillée.

Ensuite, chaque album ainsi que chaque morceau pourront recevoir des notations laissées par les utilisateurs. Ainsi, cela permettra de savoir quel morceau est le plus apprécié par la communauté.

De plus, si un utilisateur apprécie beaucoup une musique, un album ou un artiste en particulier, il pourra les ajouter à ses favoris. Ainsi, retrouver la page de son album préféré sera un jeu d'enfant pour l'utilisateur en passant par ses favoris. Chaque titre pourra être ajouté, modifié ou supprimé des favoris.

Enfin, l'utilisateur aura à sa disposition une fonctionnalité pour créer sa propre playlist. Il peut dans un premier temps créer des playlists personnalisées. Pour ajouter des musiques dans ces dernières, il aura simplement à cliquer sur Ajouter à une playlist, choisir la playlist voulue et ajouter le titre en question.

Les playlists et favoris pourront avoir différents modes de tri des titres. Le premier sera un tri chronologique par date d'ajout du titre dans la playlist ou les favoris, le second un tri par ordre alphabétique, le troisième un tri par ordre alphabétique des artistes et enfin un tri par ordre alphabétique des albums. Quand la playlist est constituée, les titres sont à la suite les uns des autres, on pourra alors réorganiser l'ordre des titres dans la playlist en fonction des titres que l'on veut écouter en premier, puis en second, etc.

Sans oublier que les playlists seront modulables. Ainsi les erreurs d'ajout pourront être vite corrigées et la suppression sera possible après confirmation.

Pour finir, chaque morceau pourra être écouté en partie à l'aide d'un bouton play mais ce ne sera pas le but principal de l'application.

La principale différence de l'application réside dans le fait que cette dernière restera gratuite. Vous pourrez tout de même faire des donations, mais rien ne l'oblige. La principale source de revenus de l'application réside dans la publicité. Au début de vos écoutes, des pubs seront insérées, mais elles ne vous dérangeront plus par la suite.

Persona / user stories

Thomas Dupont

Persona



Prénom : Thomas
Nom : Dupont
Âge : 33 ans
Profession : Entrepreneur
Situation: En couple, 1 enfant
Revenus: 34 000 € / an

Thomas Dupont est entrepreneur, il a lancé sa start-up il y a 2 ans et demi, il emploie actuellement 2 personnes, un mécanicien et une vendeuse pour sa boutique de vélos.

Il est fan de sport et de musique mais ne se retrouve jamais dans tous ses albums virtuels qu'il a téléchargés. Depuis le confinement, il cherche de plus en plus à découvrir de nouveaux artistes qu'il ne connaît pas mais retombe toujours sur les mêmes. Quand il écoute de la musique, il aime connaître la vie des artistes et leur vécu. Il utilise des applications de musique à la maison mais aussi au travail pour des musiques d'ambiance dans sa boutique ou dans son atelier.

Utilisation d'internet: en moyenne 8h/jour, avec l'utilisation à la maison et au travail, environ 30% du temps à la maison.

Apps préférées: Strava, YouTube
Sites préférés : lapierrebikes.com

User story

1^{er} cas:

Thomas Dupont veut mettre de la musique d'ambiance dans sa boutique. Le problème étant qu'il ne connaît pas beaucoup de musiques d'ambiance, il peut donc, grâce à l'application Désaccordé, faire une recherche de musique par thème et écouter des extraits de ces dernières. Il peut ensuite, s'il apprécie la musique et la juge appropriée au contexte, ajouter cette dernière dans la playlist ambiance qu'il aura créée pour diffuser dans sa boutique. Il peut alors jouer les musiques en mode aléatoire ou en fonction de la date d'ajout dans la playlist ou encore par titre.

2^{eme} cas:

Thomas Dupont n'a pas vraiment de style de musique préféré, mais ce qu'il aime par-dessus tout, c'est connaître la vie et le passé des artistes. Pour cela, lorsqu'il écoute une musique qui lui plaît, en radio par exemple, il prend en note le nom de l'artiste, et cherche sur son temps libre cet artiste sur l'application Désaccordé. Il peut alors se documenter grâce à la section biographie de l'artiste et peut même se documenter sur un album en particulier de cet artiste.

Camille Rose

Persona



Prénom : Camille
Nom : Rose
Âge : 17 ans
Profession : Etudiante
Situation : Célibataire
Revenus : Aucun

Camille Rose est en première dans le lycée de sa ville. Elle fait de la gymnastique en compétition et adore s'entraîner en musique. Elle est fan de musique pop mais ces derniers temps elle cherche à découvrir de nouveaux styles et aussi à enrichir ses connaissances. Le problème, c'est qu'elle n'est pas du tout familière avec ces autres styles et elle n'arrive donc jamais à se rappeler si elle a déjà écouté ce morceau ou non.

Utilisation d'internet : 5h/jour, Principalement réseaux sociaux et messages

App préférées : Instagram, Spotify

User story

1^{er} cas:

Camille veut organiser une petite soirée avec ses copines. Elle est très organisée et cherche donc des musiques qu'elle pourra diffuser lors de sa soirée. Le problème étant qu'elle ne connaît pas vraiment de musique d'ambiance pour une soirée, elle demande à ses copines mais elles sont dans le même cas. Désaccordé lui permettra de trouver des musiques appropriées. Grâce à son système de notation, elle pourra consulter cette rubrique pour l'aider dans ses choix. Elle peut aussi consulter la page d'accueil de l'application pour voir les titres les plus aimés par la communauté en ce moment.

2^{ème} cas:

Camille écoute souvent les nouvelles sorties de ses artistes préférés. Bien souvent, ces sorties sont sous forme d'album et elle aime beaucoup écouter ces albums dans l'ordre car, quelquefois, les albums racontent une sorte d'histoire. La fonctionnalité de recherche par album lui permettra donc de répondre à ce besoin. Si les albums qu'elle recherche lui plaisent, elle pourra alors les ajouter à sa liste d'albums préférés.

Huguette Martin

Persona



Prénom : Huguette

Nom : Martin

Âge : 71 ans

Profession : Retraitée

Situation : Mariée, 3 enfants indépendants

Revenus : 16 000€ / an

Huguette Martin est retraitée et cherche de nouvelles activités pour occuper ses longues journées quand son mari part faire du vélo dans l'association du village. Elle aime écouter de la musique pendant ses activités. Elle écoute principalement des musiques anciennes, ne se souvient pas des noms de ces dernières mais plutôt des artistes. Elle aime aussi découvrir de nouveaux styles, notamment grâce à ses petits-enfants.

Elle utilise peu internet mais se sert quand même d'un ordinateur pour gérer les comptes de l'association de vélo ou pour lire ses mails.

Utilisation d'internet : 1 ou 2h/jour maximum

Site préféré : Mail Orange

Appli préférée : Google Duo

User story

1^{er} cas:

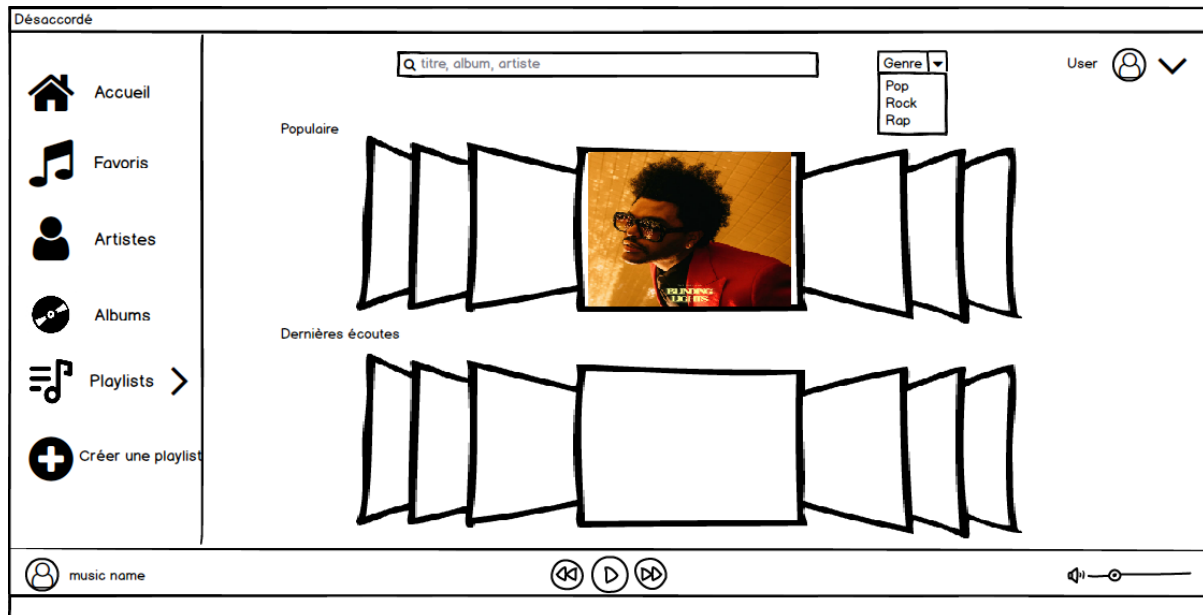
Huguette écoutait beaucoup de musique étant jeune, elle a donc une culture musicale conséquente. Le problème est qu'elle connaît beaucoup d'artistes mais ne se souvient pas forcément au moment voulu d'un titre spécifique. La fonctionnalité de recherche par artiste lui permettra donc de retrouver ses titres préférés et de les écouter à nouveau. Elle pourra retrouver ses artistes préférés en les ajoutant à ses artistes préférés.

2^{eme} cas:

Parfois, les petits-enfants de Huguette lui font écouter de la musique comme le rap. Huguette étant curieuse, elle cherche à comprendre ce qui se dit dans ces musiques mais ne comprend pas grand-chose. Elle cherche alors à comprendre dans quel contexte et comment ces artistes arrivent à créer leurs albums. Pour cela, elle consulte la description des albums.

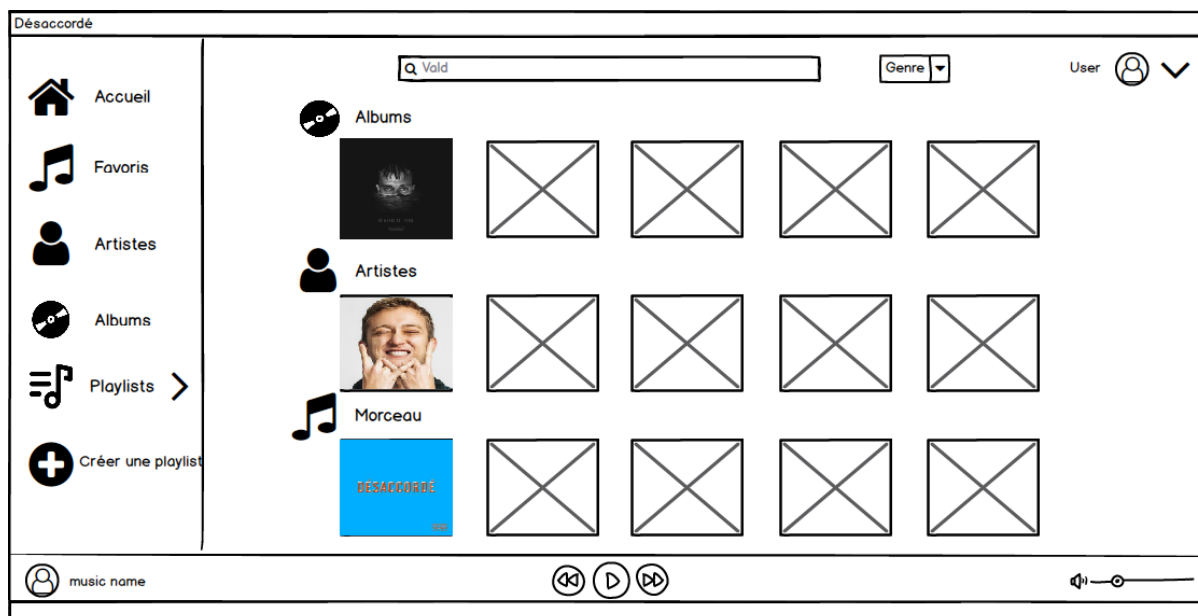
Sketchs

Accueil



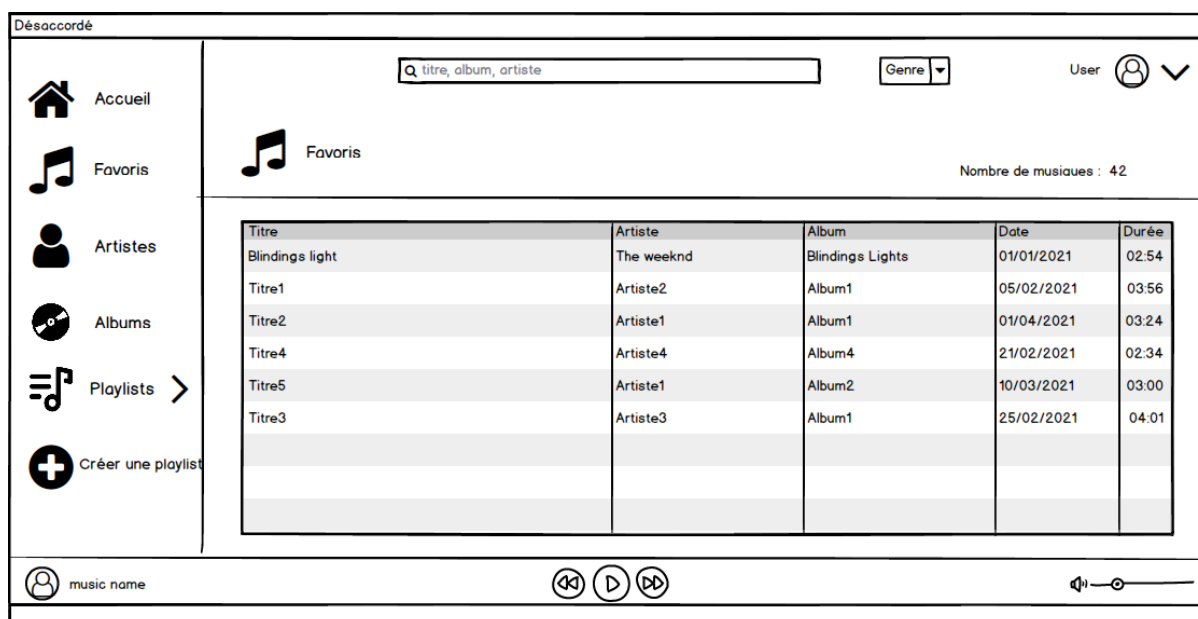
Ce sketch est le menu principal de l'application. C'est la page d'accueil. On pourra y voir les morceaux les plus populaires du moment ainsi que les derniers morceaux écoutés. On pourra rechercher les artistes, albums ou morceaux. Ou choisir la recherche par genre grâce au menu déroulant. Le menu de gauche permettra de naviguer dans les différentes pages de l'application et sera statique. On aperçoit sur le bandeau en bas de la page des informations sur la musique en lecture, des commandes pour passer à la musique suivante ou précédente, mettre en pause la musique et changer le volume de cette dernière.

Recherche



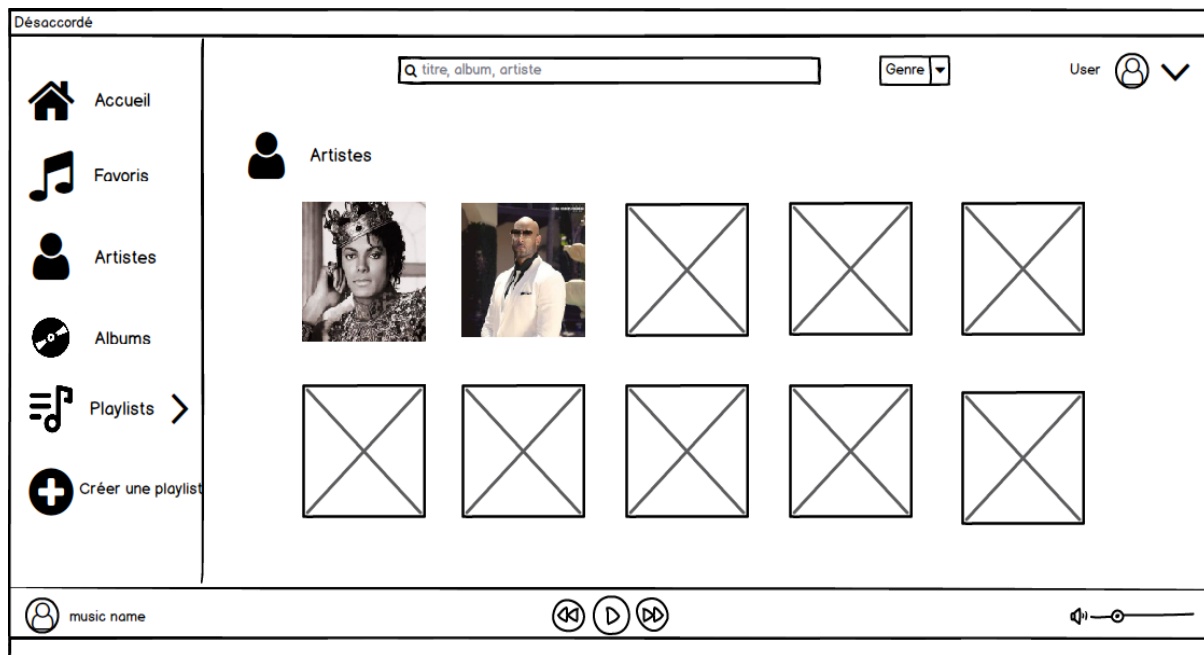
Lorsque l'utilisateur rentre un mot dans la barre de recherche, plusieurs albums, artistes et morceaux apparaîtront en fonction du mot tapé. Ensuite, l'utilisateur pourra choisir ce qu'il voudra pour poursuivre sa recherche et écouter le morceau voulu.

Morceaux Favoris



La page des favoris sera une liste de morceaux placés en favoris avec des détails comme l'artiste, l'album, la date d'ajout et la durée du morceau.

Artistes



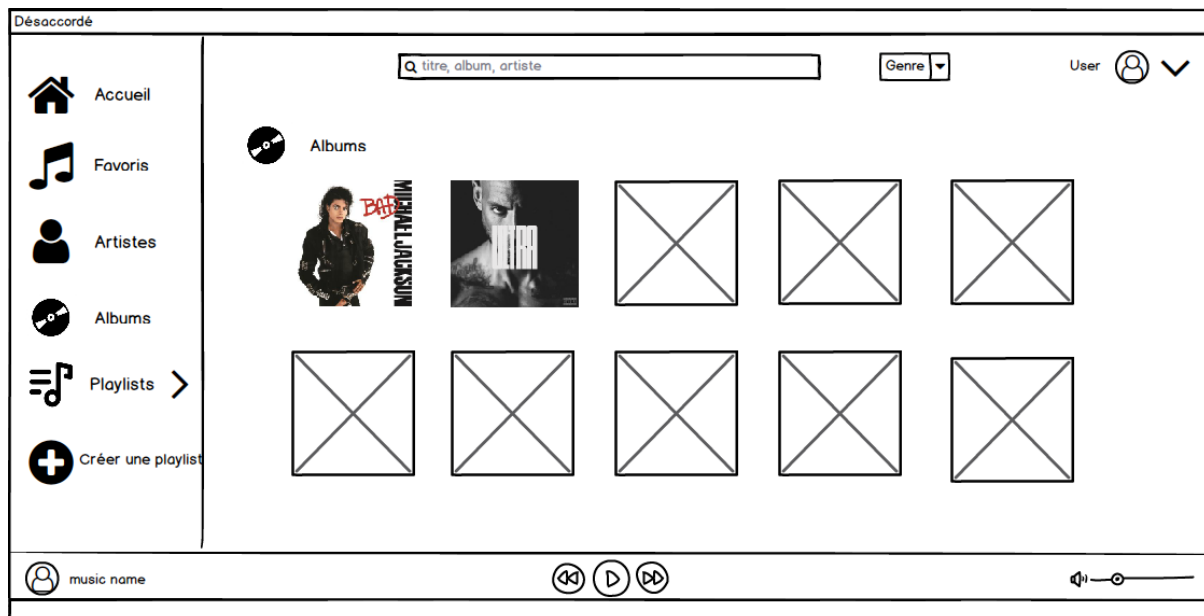
La page des artistes présente tous les artistes mis en favori sur l'application. On pourra ainsi cliquer sur chaque icône d'artistes pour avoir un descriptif détaillé de l'artiste en question.

Page artiste détaillée



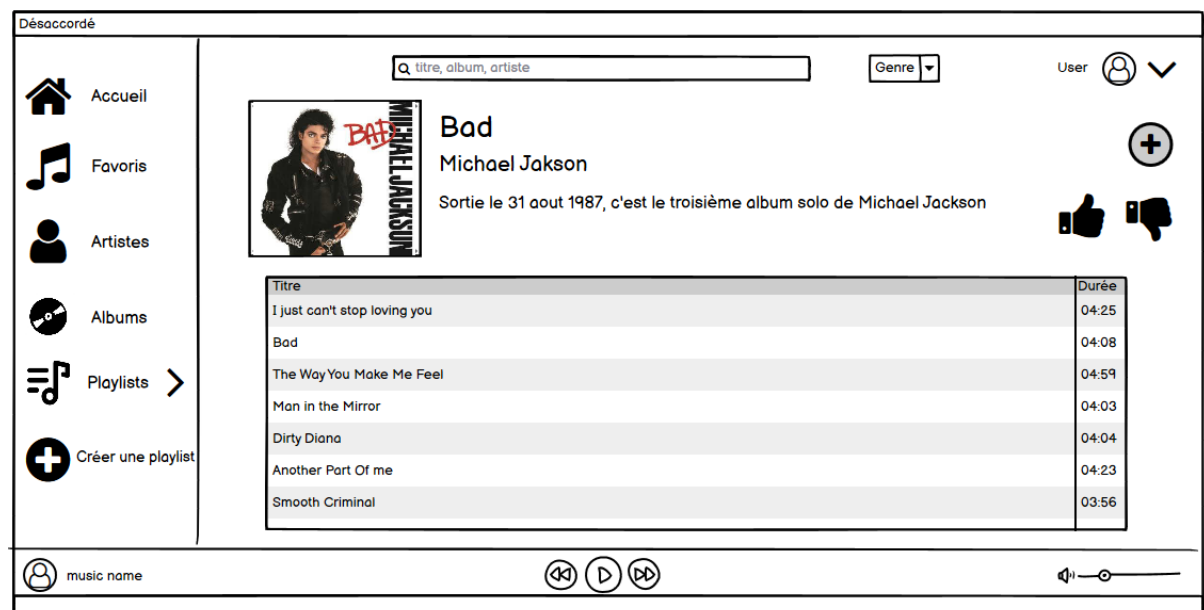
Ainsi, chaque artiste aura sa propre page avec ses morceaux populaires et une description sur l'artiste. On pourra dire si on aime ou pas l'artiste grâce aux pouces et l'ajouter aux favoris s'il n'y est pas déjà.

Albums



Ensuite, la page des albums énumère chaque album ajouté en favoris dans l'application. On pourra comme pour les artistes cliquer sur l'icône de l'album pour avoir sa description :


Page album détaillée





Chaque album aura sa page avec le nom de l'artiste ainsi que sa description suivie de la liste des morceaux présents dans l'album. On pourra également donner son avis avec les pouces comme pour les artistes ou l'ajouter aux favoris.


Page playlist détaillée


Désaccordé


 Accueil

 Favoris


 Artistes

 Albums

 Playlists >

 Créer une playlist

Genre ▾


User  ▾




Ma playlist



Description

01/01/2021

Titre	Artiste	Album	Date d'ajout	Durée
Allumer le feu	Johnny Hallyday	Ce que je sais	01/01/2021	04:19
Titre1	Artiste2	Album1	05/02/2021	03:56
Titre2	Artiste1	Album1	01/04/2021	03:24
Titre4	Artiste4	Album4	21/02/2021	02:34
Titre5	Artiste1	Album2	10/03/2021	03:00
Titre3	Artiste3	Album1	25/02/2021	04:01

 music name

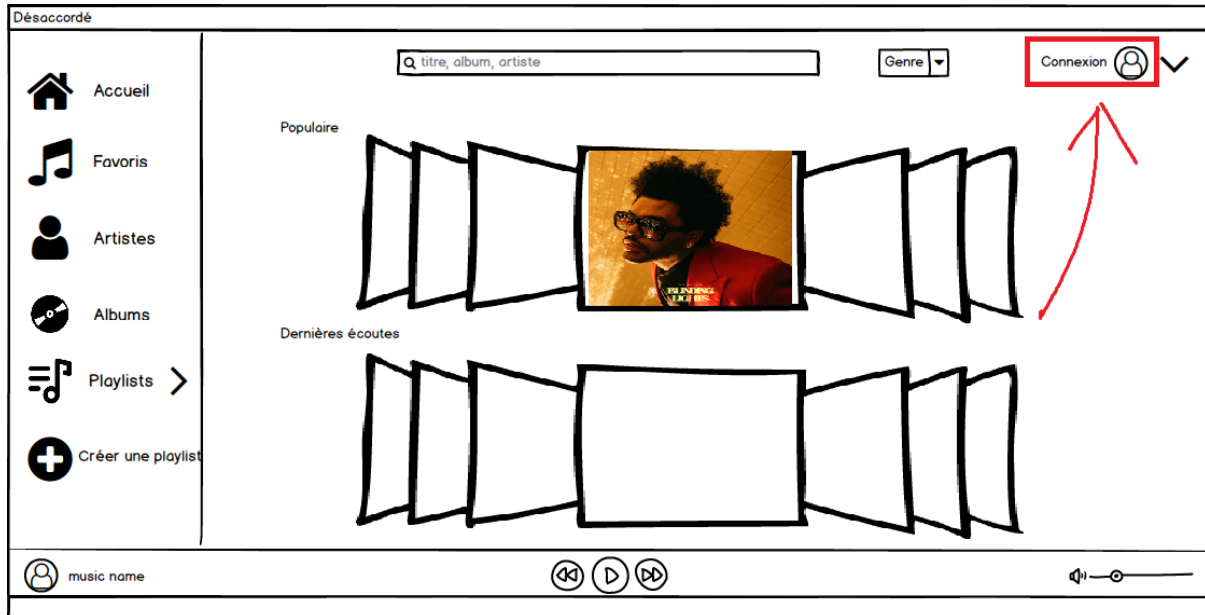
 

L'utilisateur pourra créer des playlists, les nommer, leur donner une description et ajouter des morceaux à celles-ci. Une liste de musique apparaîtra avec le titre.

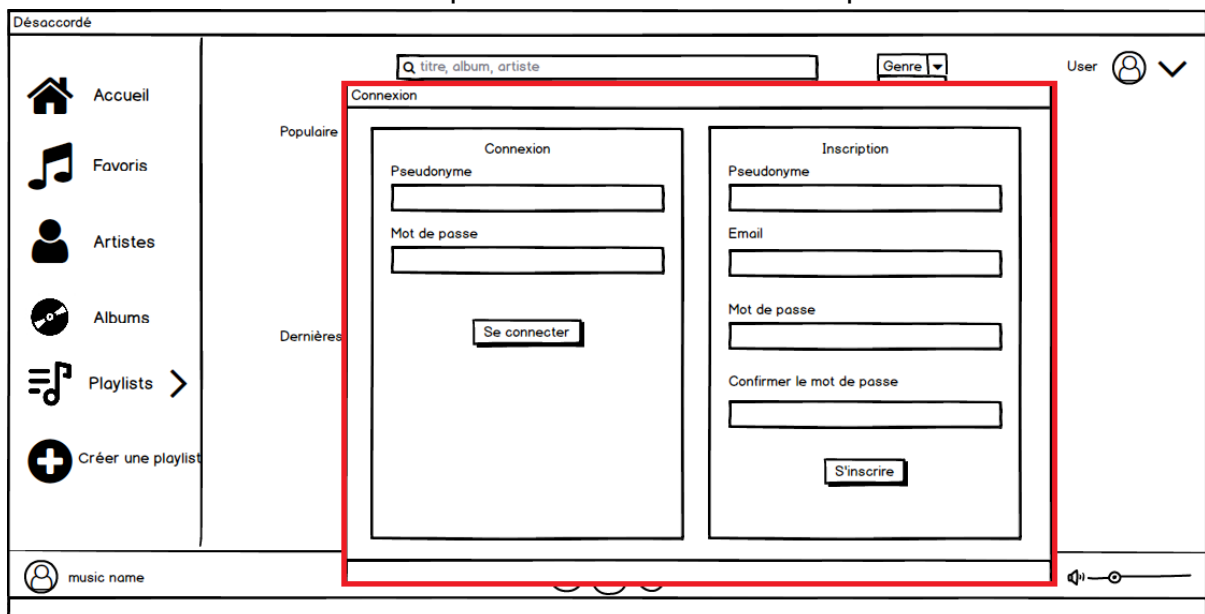
Storyboards

Se connecter

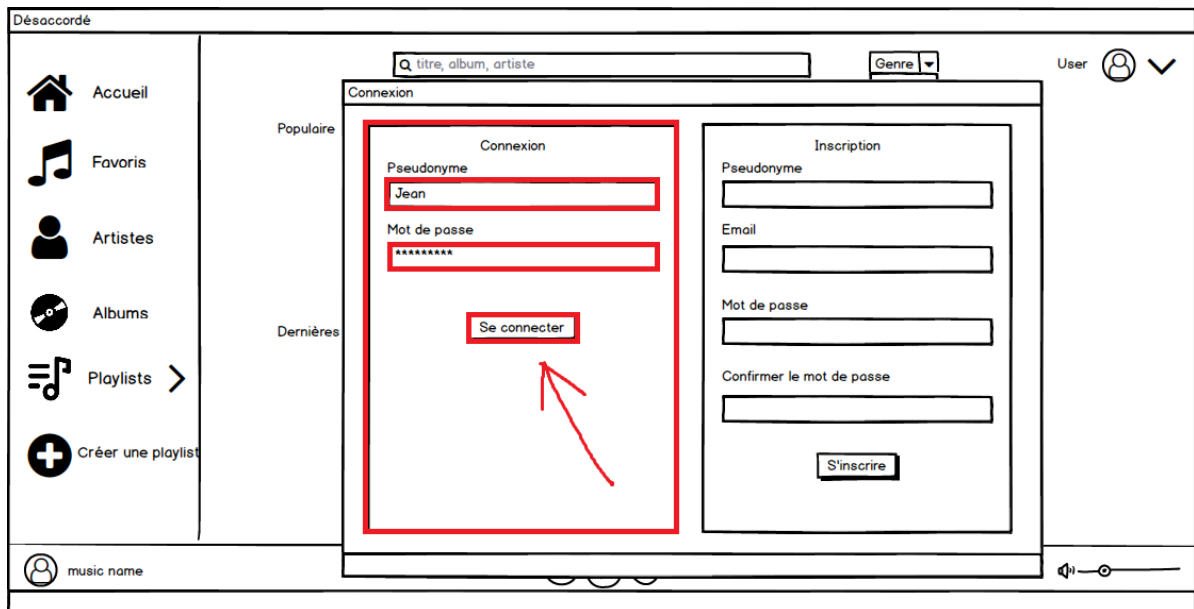
L'utilisateur se trouve dans l'accueil, il clique sur le bouton connexion en haut à gauche.



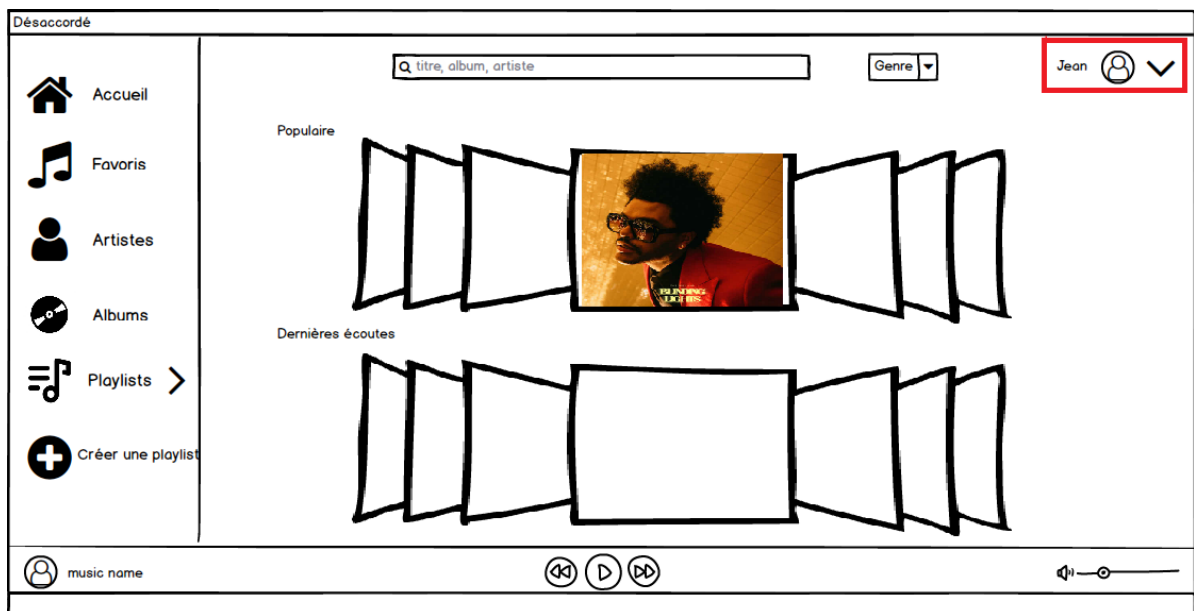
Une fenêtre s'ouvre avec deux parties : Connexion et Inscription :



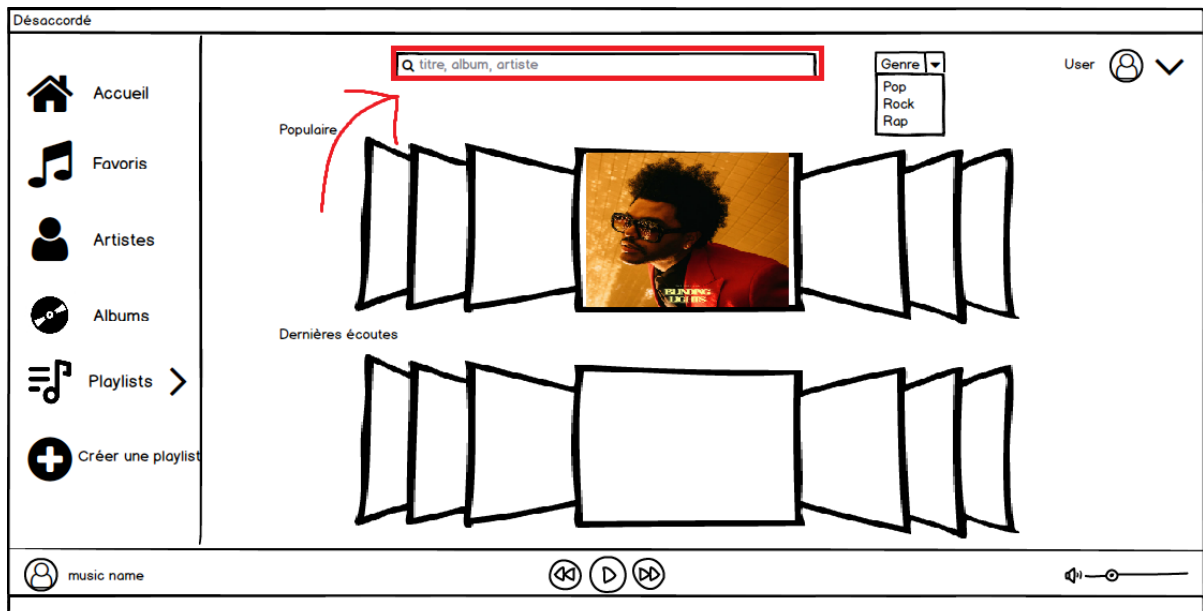
L'utilisateur se connecte ou s'inscrit en rentrant ses informations puis clique sur le bouton "Connexion" (ou "S'inscrire") pour terminer :



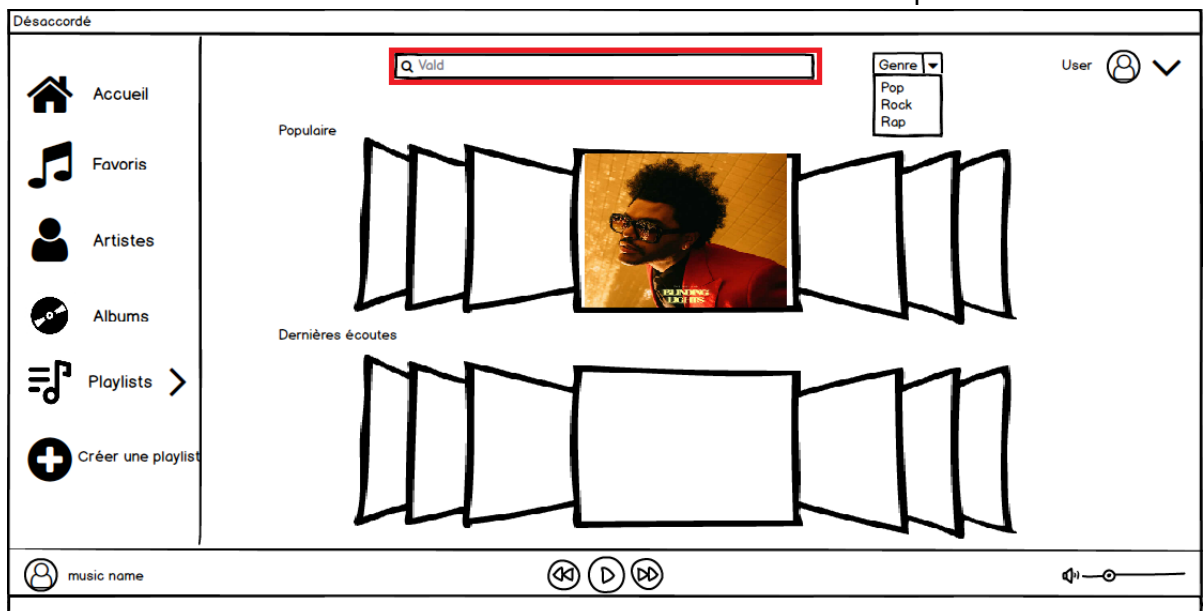
Enfin, son pseudonyme apparaît à la place du bouton “connexion”. Il est donc bien identifié.



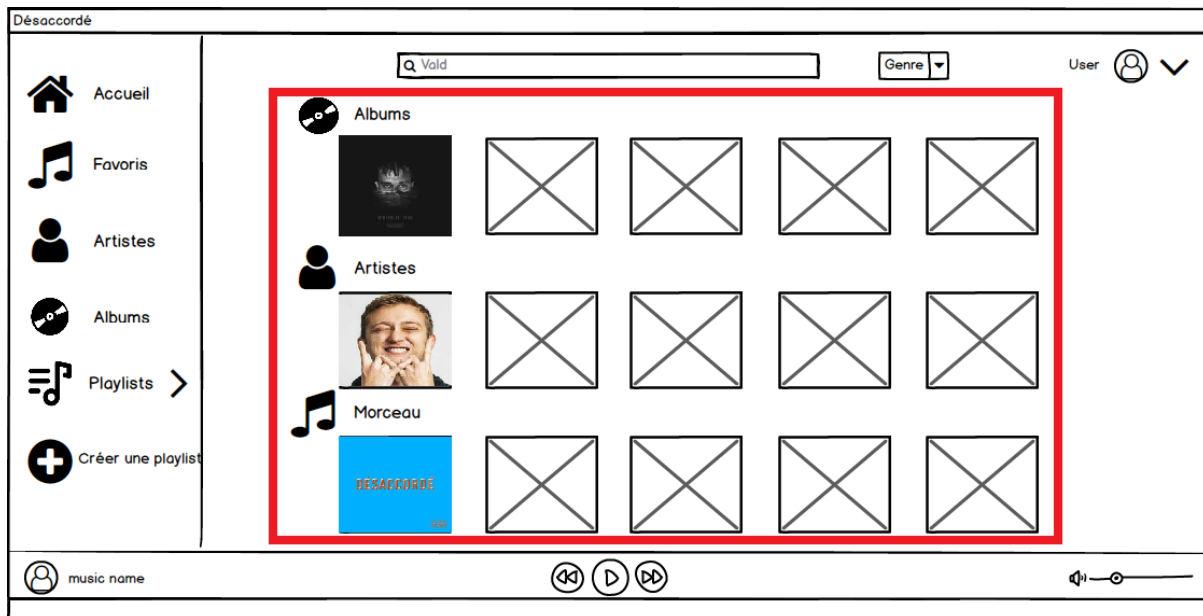
Recherche de titre/album/artiste/genre



L'utilisateur va devoir utiliser la barre de recherche. Il rentre les mots qu'il veut :

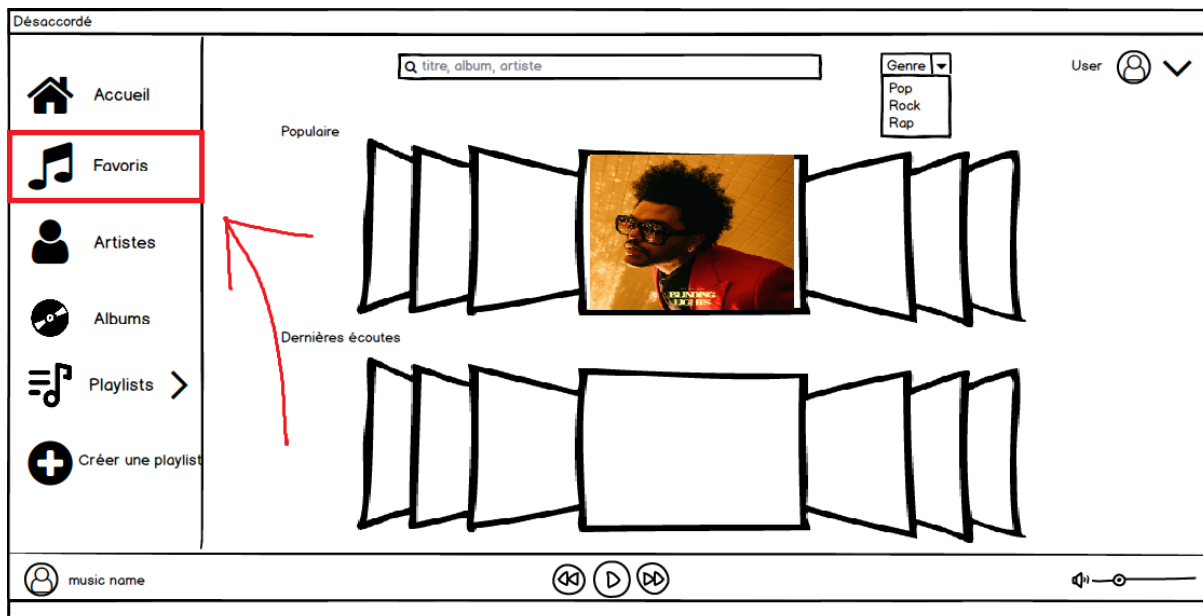


Puis il appuie sur Entrée :

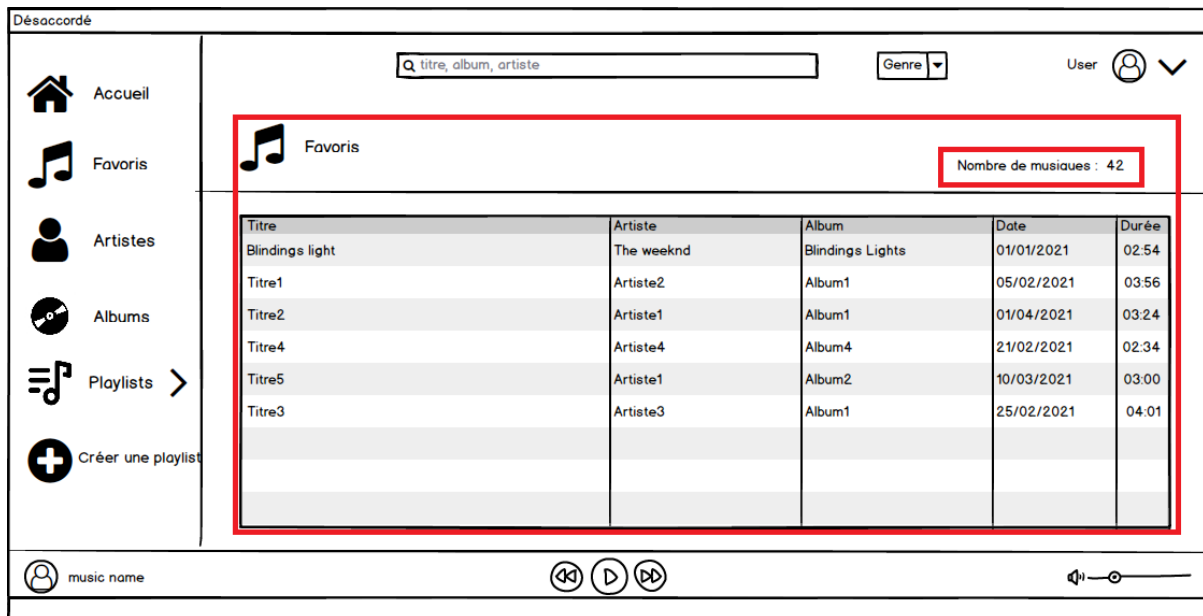


Les différents albums / artistes / morceaux s'affichent en fonction des mots qu'il a tapés.

Accès aux titres favoris

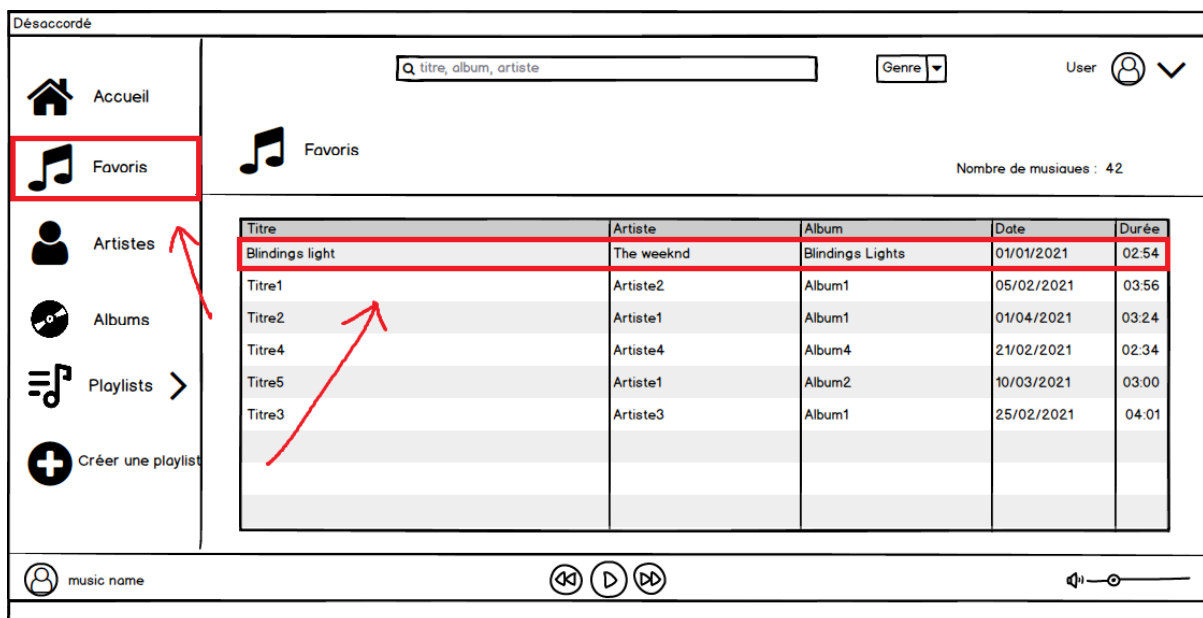


L'utilisateur se trouve n'importe où dans le site, il peut cliquer sur le bouton Favoris du menu principal qui reste toujours présent à l'écran.

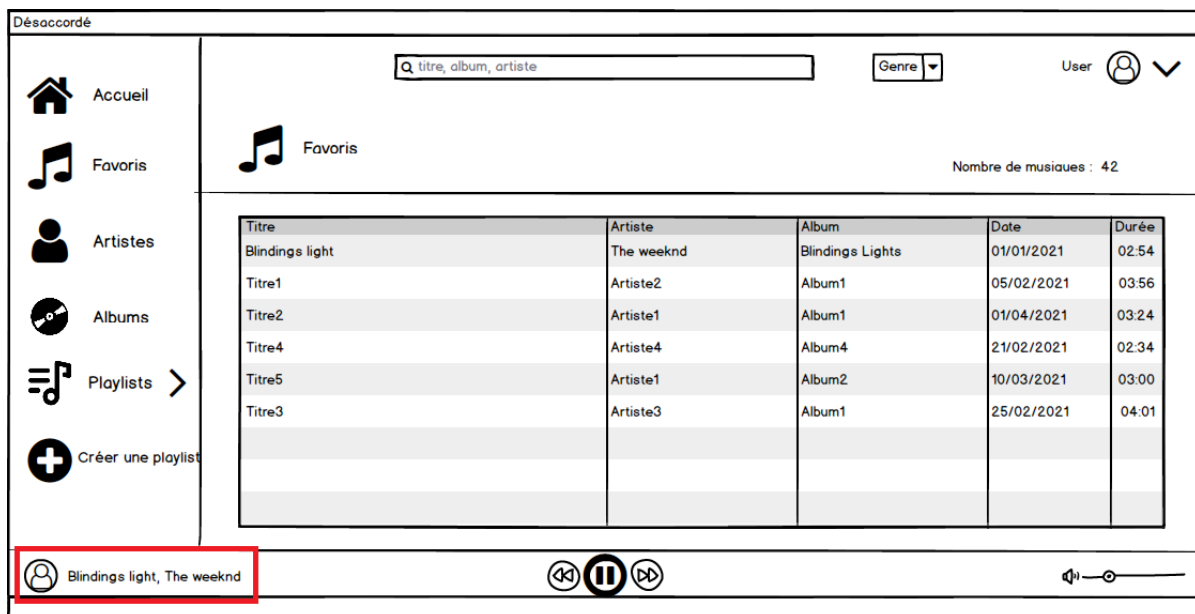


La page des morceaux favoris s'ouvre avec une liste de tous les morceaux et le nombre total de morceaux en haut.

Jouer une musique

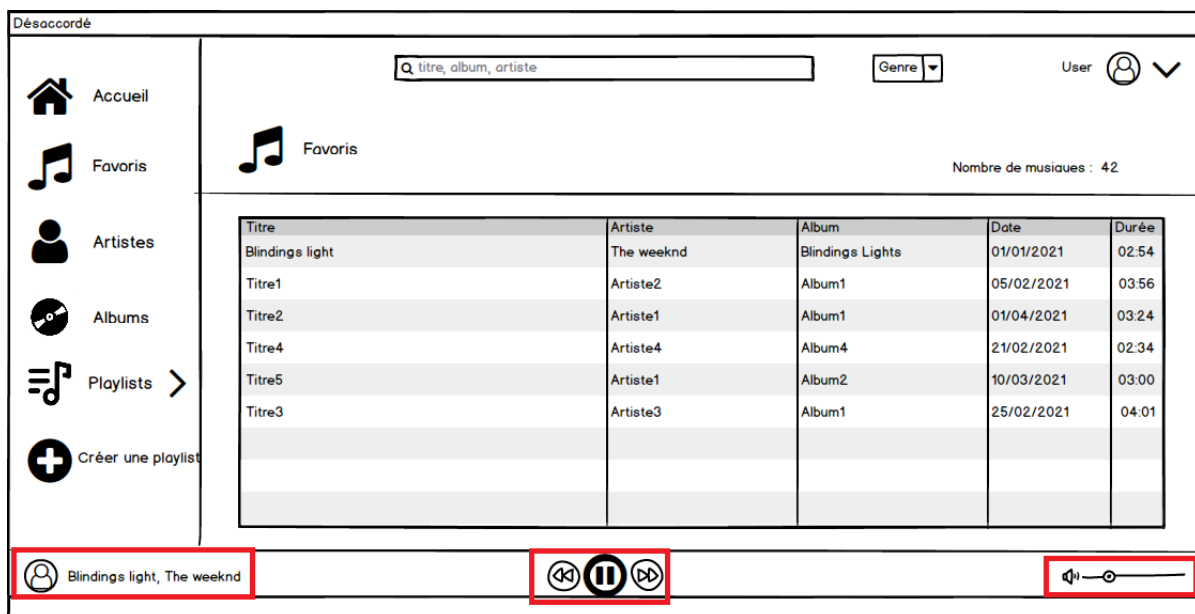


L'utilisateur se rend dans ses favoris ou dans une playlist, choisit sa musique et double clique dessus pour l'écouter.



Une fois la musique lancée, le nom et l'artiste apparaissent en bas à gauche.

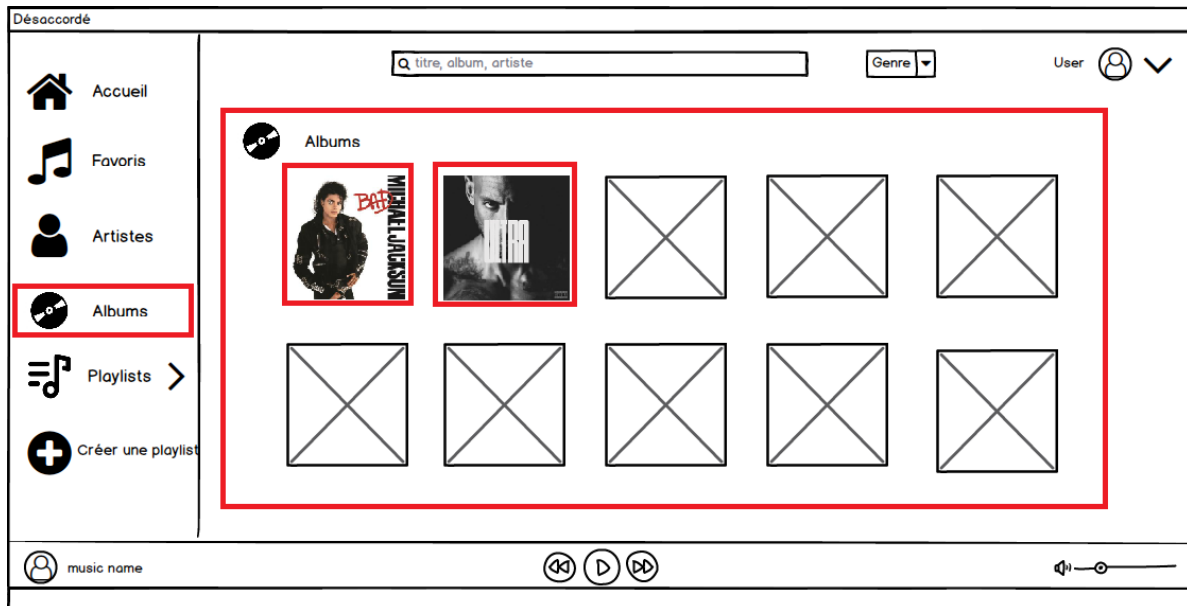
Boutons utilitaires



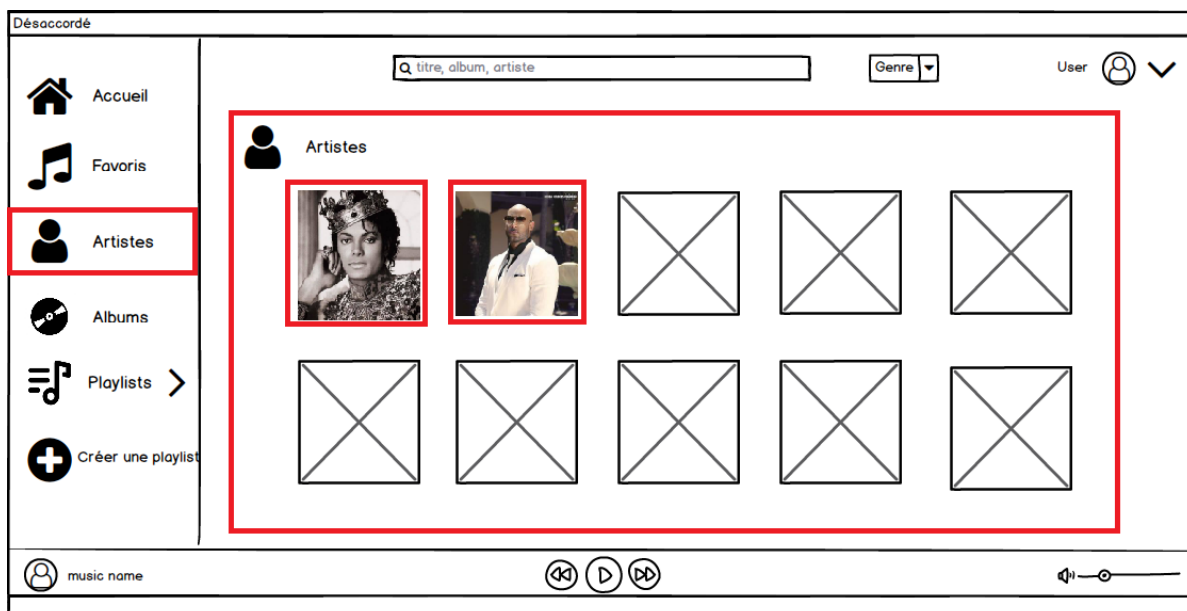
Une fois la musique lancée, on peut la stopper avec le bouton au milieu, passer à la musique suivante ou précédente avec les flèches autour du bouton stop. Enfin, le son peut être réglé avec le bouton de droite.

Accès aux artistes/albums favoris

L'accès aux albums favoris se fait à partir du menu avec le bouton "Albums". Une fois cliquée, une page s'affiche avec l'ensemble des albums favoris.

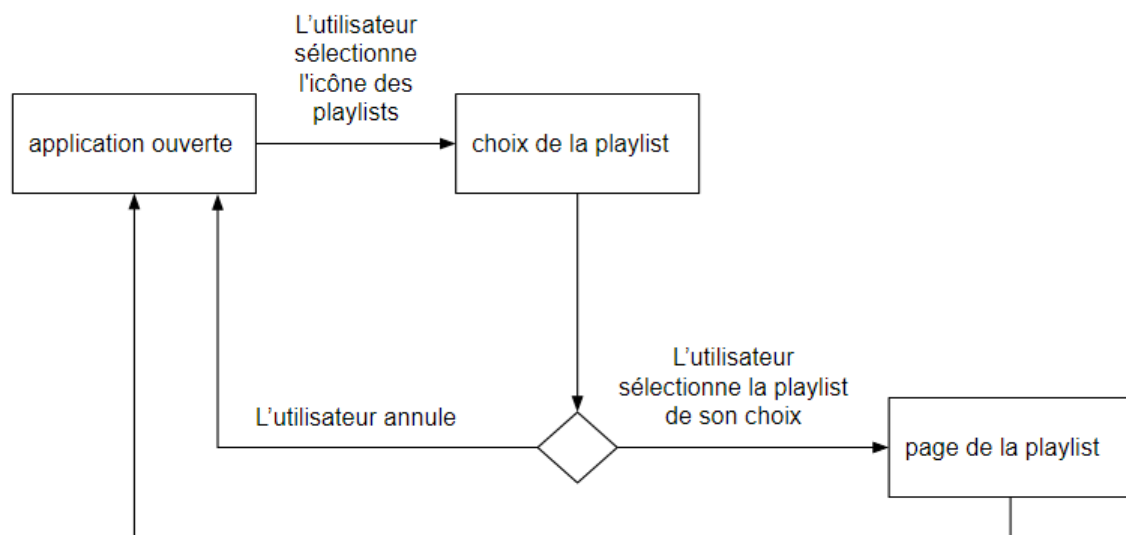


La même manipulation est nécessaire pour accéder aux artistes favoris, bouton "artistes" puis une page s'ouvre :

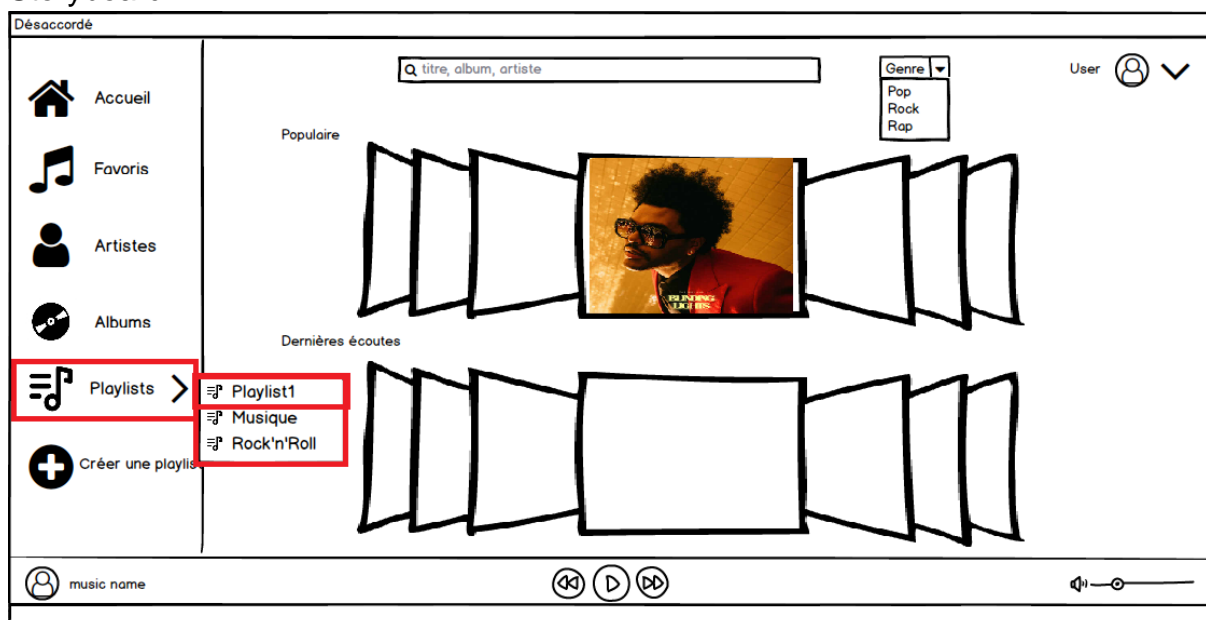


Accès aux playlists

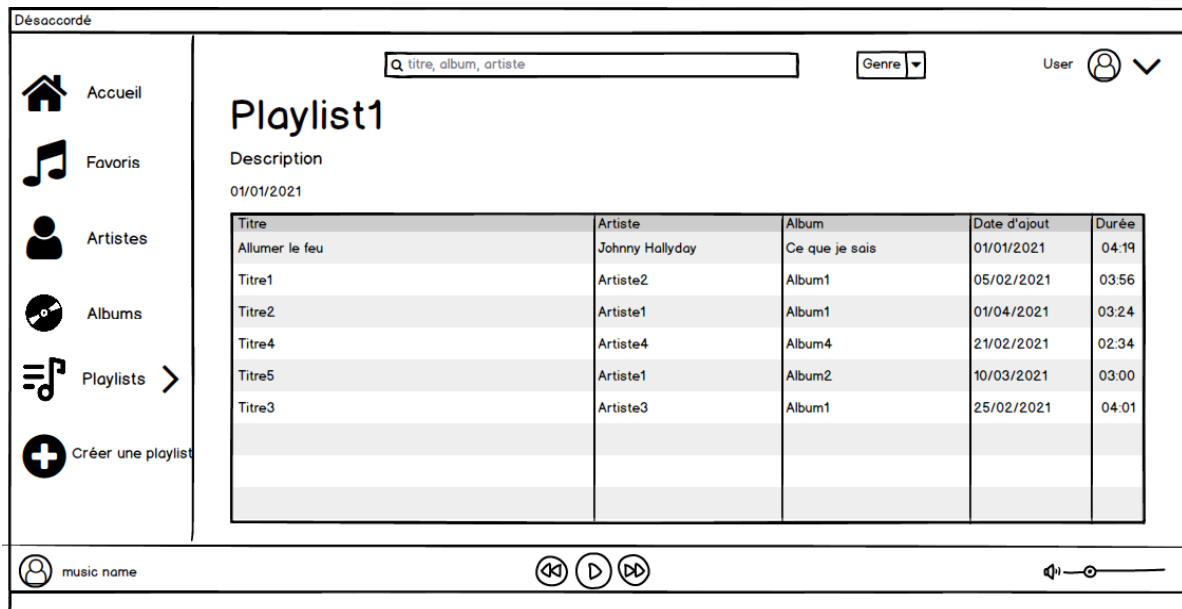
Diagramme de flux :



Storyboard :

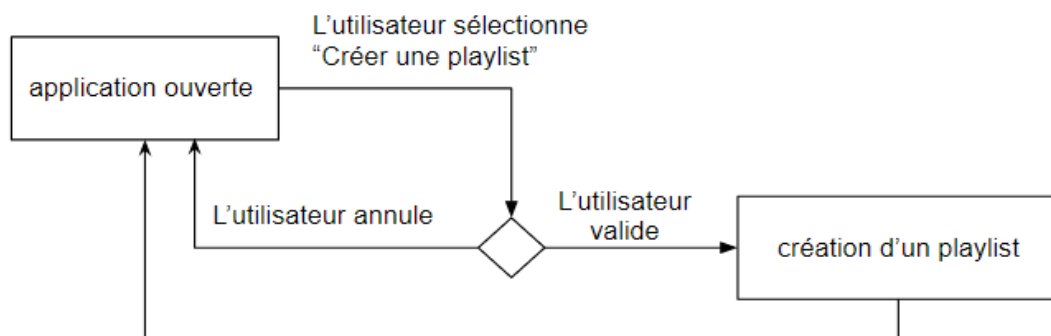


L'utilisateur se situe sur la page d'accueil ou n'importe où sur le site. Il clique sur Playlists dans le menu, puis choisit sa playlist dans le menu déroulant qui s'affiche. Une fois sa playlist choisie, il clique dessus. La page de la playlist choisie s'affiche alors avec l'ensemble des morceaux présents dessus :

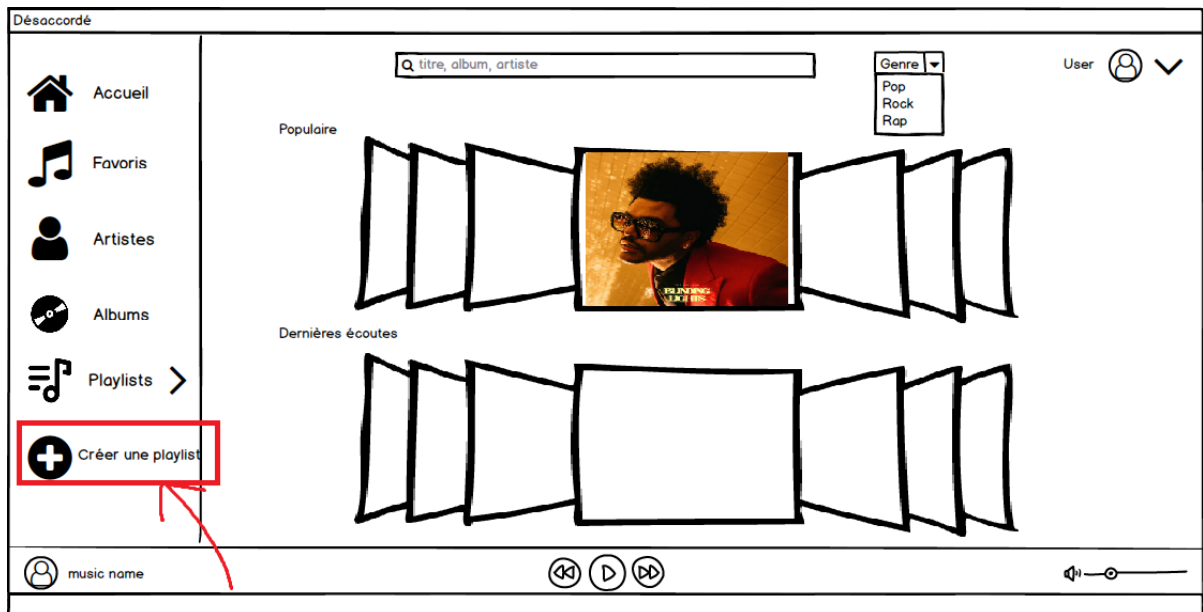


Création d'une playlist

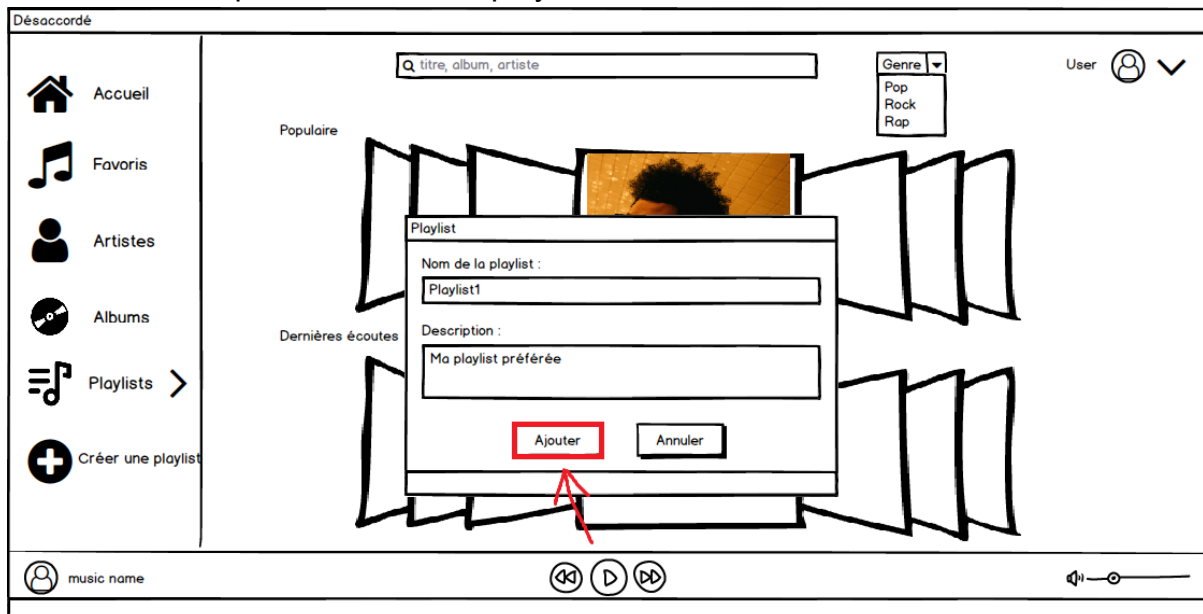
Diagramme de flux :



Storyboard :

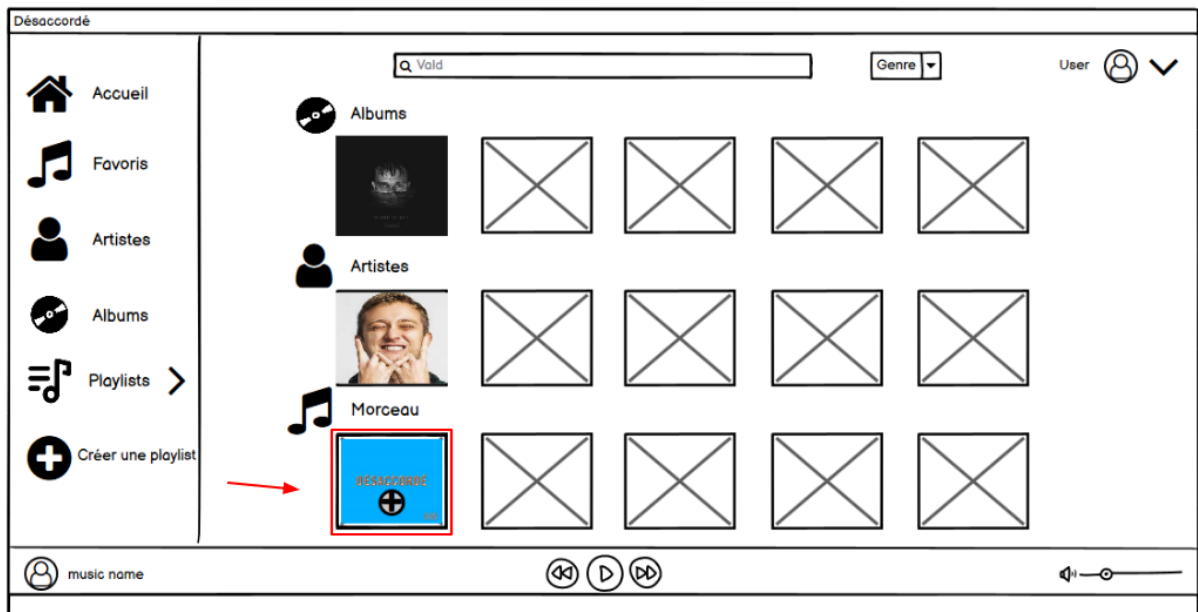


L'utilisateur clique sur Créer une playlist, une fenêtre s'affiche :

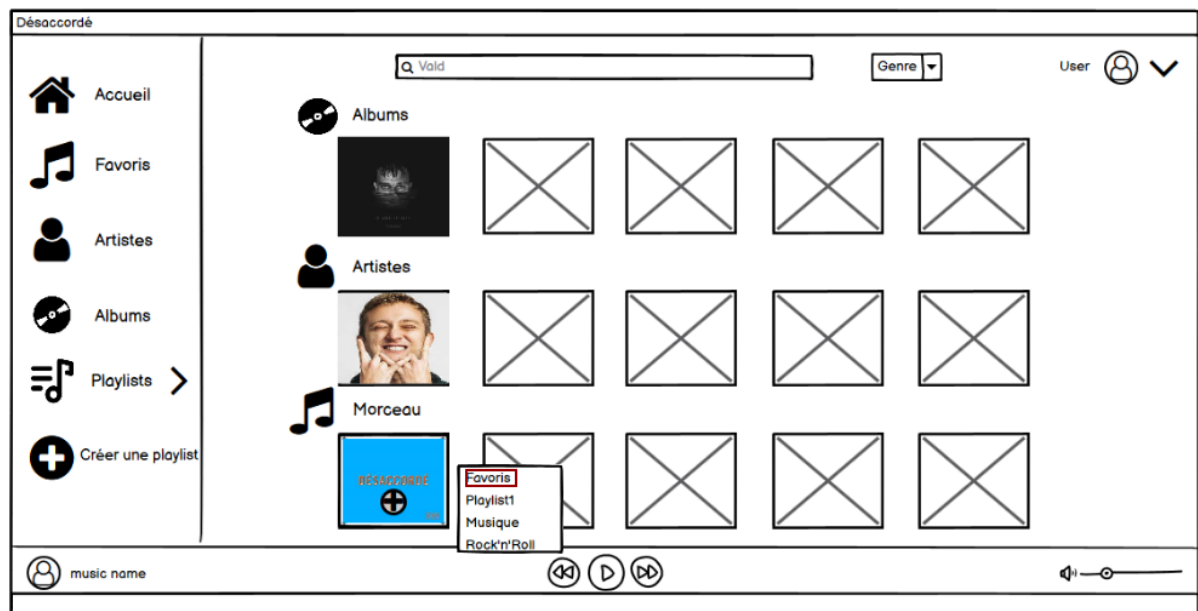


Il rentre le nom de sa playlist et sa description puis appuie sur Ajouter ou Annuler selon ses choix.

Ajout d'une musique dans une playlist

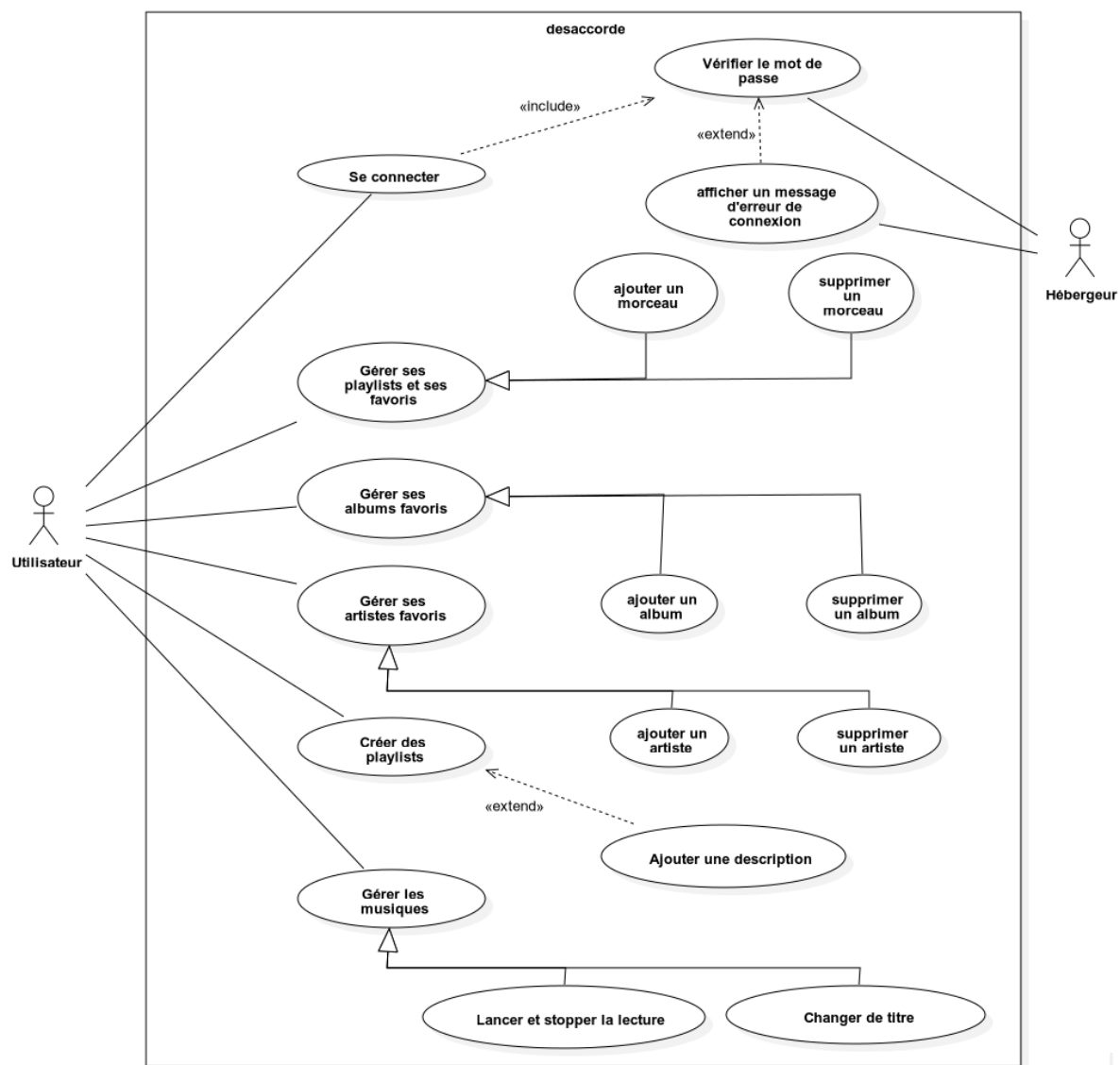


Lorsque l'on arrive sur la page de recherche, on entre un mot clé dans la barre de recherche. Ensuite, il suffit de faire un clic droit sur un morceau pour l'ajouter à ses favoris ou à une playlist. Une fenêtre s'affiche alors :



Il suffit alors de choisir ou ajouter le titre. Le même fonctionnement est repris pour les artistes et albums. Ils sont alors ajoutés d'office dans les favoris des albums et favoris.

Diagramme de cas d'utilisation



UI

Cas “Se connecter”

Nom	Se connecter
Objectif	S'authentifier sur son compte en utilisant ses identifiants
Acteurs principaux	Utilisateurs
Acteurs secondaires	Hébergeur
Conditions initiales	- L'utilisateur doit avoir un compte déjà créé
Scénario d'utilisation	- L'utilisateur entre ses identifiants - L'hébergeur vérifie ces informations [condition de fin 1] sinon [condition de fin 2]
Conditions de fin	1) L'utilisateur a entré les bons identifiants, il est connecté 2) L'utilisateur n'a pas entré les bonnes informations, affichage d'un message d'erreur

Cas "Vérifier le mot de passe"

Nom	Vérifier le mot de passe
Objectif	Valider la connexion d'un utilisateur
Acteurs principaux	Hébergeur
Acteurs secondaires	Utilisateur
Conditions initiales	- L'hébergeur doit avoir accès aux données des utilisateurs
Scénario d'utilisation	- L'hébergeur reçoit les données de connexion - Il vérifie si les données entrées correspondent aux données qu'il connaît - Si elles correspondent [condition de fin 1] sinon [condition de fin 2]
Conditions de fin	1) Les données sont valides, il autorise la connexion. 2) Les données ne sont pas correctes, il affiche un message d'erreur

Cas “afficher un message d’erreur de connexion”

Nom	Afficher un message d’erreur de connexion
Objectif	Informer l’utilisateur sur les données remplies
Acteurs principaux	Hébergeur
Acteurs secondaires	Utilisateurs
Conditions initiales	- L’hébergeur doit avoir vérifié les identifiants
Scénario d’utilisation	- L’hébergeur envoie un message si les données ne sont pas correctes [condition de fin 1] sinon [condition de fin 2]
Conditions de fin	1) les données sont erronées : message envoyé à l'utilisateur pour l'informer 2) les données sont correctes, aucun message envoyé.

Cas “Créer des playlists”

Nom	Créer des playlists
Objectif	Ajouter une nouvelle playlist à son compte
Acteurs principaux	Utilisateur
Acteurs secondaires	/
Conditions initiales	- L'utilisateur est authentifié
Scénario d’utilisation	- L'utilisateur clique sur “Créer une playlist” - Il choisit le nom et la description de la playlist et confirme [condition de fin 1] ou [condition de fin 2]
Conditions de fin	1) La playlist est créée 2) L'utilisateur annule la création

Cas “Gérer ses playlists et ses favoris”

Nom	Gérer ses playlists et ses favoris
Objectif	Permettre à l'utilisateur de modifier ses playlists et sa page de favoris
Acteurs principaux	Utilisateur
Acteurs secondaires	/
Conditions initiales	<ul style="list-style-type: none">- Avoir une playlist déjà créée et/ou une page de favoris- il doit être connecté
Scénario d'utilisation	<ul style="list-style-type: none">- Scénario 1 :<ul style="list-style-type: none">- L'utilisateur veut ajouter un morceau.- L'utilisateur cherche un morceau- Il le choisit et choisit où l'ajouter, playlist ou favoris [condition de fin 1] sinon [condition de fin 2]- Scénario 2 :<ul style="list-style-type: none">- L'utilisateur veut supprimer un morceau.- L'utilisateur choisit sa playlist ou sa page de favoris- Il choisit le morceau et le supprime [condition de fin 1] ou [condition de fin 2]
Conditions de fin	<p>Scénario 1 :</p> <ol style="list-style-type: none">1) La playlist est mise à jour avec le morceau choisi en plus.2) L'utilisateur annule <p>Scénario 2 :</p> <ol style="list-style-type: none">1) Le morceau est supprimé de la playlist / des favoris2) L'utilisateur annule

Notes : Ce cas inclut les deux autres cas : “ajouter” et “supprimer” un morceau d’une playlist / des favoris.

Cas “Gérer ses albums favoris”

Nom	Gérer ses albums favoris
Objectif	Permettre à l'utilisateur d'ajouter ou supprimer des albums à sa page de favoris
Acteurs principaux	Utilisateur
Acteurs secondaires	/
Conditions initiales	<ul style="list-style-type: none">- L'utilisateur doit être connecté- Un album au minimum doit déjà être en favoris- Scénario 1 : L'album ajouté ne doit pas être déjà présent dans la page de favoris- Scénario 2 : Il y a au moins un album déjà présent dans la page de favoris
Scénario d'utilisation	<ul style="list-style-type: none">- Scénario 1 :<ul style="list-style-type: none">- L'utilisateur cherche un album- Il le choisit et l'ajoute [condition de fin 1] ou [condition de fin 2]- Scénario 2 :<ul style="list-style-type: none">- L'utilisateur choisit son album- Le supprime [condition de fin 1] ou [condition de fin 2]
Conditions de fin	<p>Scénario 1 :</p> <ol style="list-style-type: none">1) L'album est ajouté aux favoris2) L'utilisateur annule <p>Scénario 2 :</p> <ol style="list-style-type: none">1) L'album est supprimé de ses favoris2) L'utilisateur annule

Notes : Ce cas inclut les deux autres cas : “ajouter” et “supprimer” un album.

Cas “Gérer ses artistes favoris”

Nom	Gérer ses artistes favoris
Objectif	Permettre à l'utilisateur d'ajouter ou supprimer des artistes à sa page de favoris
Acteurs principaux	Utilisateur
Acteurs secondaires	/
Conditions initiales	<ul style="list-style-type: none"> - L'utilisateur doit être connecté - L'utilisateur doit avoir au minimum un artiste favoris - Scénario 1 : L'artiste à ajouter ne doit pas être déjà présent dans la page de favoris - Scénario 2 : Au moins un artiste doit être présent dans la page de favoris
Scénario d'utilisation	<ul style="list-style-type: none"> - Scénario 1 : <ul style="list-style-type: none"> - L'utilisateur veut ajouter un artiste à ses favoris. - L'utilisateur cherche un artiste - Il le choisit et l'ajoute [condition de fin 1] ou [condition de fin 2] - Scénario 2 : <ul style="list-style-type: none"> - L'utilisateur veut supprimer un artiste de ses favoris. - L'utilisateur choisit son artiste à enlever - Le supprime [condition de fin 1] ou [condition de fin 2]
Conditions de fin	<p>Scénario 1 :</p> <ol style="list-style-type: none"> 1) L'artiste est ajouté aux favoris 2) L'utilisateur annule <p>Scénario 2 :</p> <ol style="list-style-type: none"> 1) L'artiste est supprimé de ses favoris 2) L'utilisateur annule

Notes : Ce cas inclut les deux autres cas : “ajouter” et “supprimer” un artiste.

Cas “Gérer les musiques”

Nom	Gérer les musiques
Objectif	Permettre à l'utilisateur de lancer, stopper la lecture ou de changer de titre
Acteurs principaux	Utilisateur
Acteurs secondaires	/
Conditions initiales	<ul style="list-style-type: none">- Une musique doit être déjà lancée pour changer de titre
Scénario d'utilisation	<ul style="list-style-type: none">- L'utilisateur veut mettre en pause ou relancer la lecture de son titre [condition de fin 1].- L'utilisateur veut changer de titre à l'écoute [condition de fin 2].
Conditions de fin	<ol style="list-style-type: none">1) cas “Lancer et stopper la lecture”.2) cas “Changer de titre”.

Cas “Lancer et stopper la lecture”

Nom	Lancer et stopper la lecture
Objectif	Permettre à l'utilisateur d'arrêter la musique et de la relancer
Acteurs principaux	Utilisateur
Acteurs secondaires	/
Conditions initiales	<ul style="list-style-type: none">- Une musique doit être choisie
Scénario d'utilisation	<ul style="list-style-type: none">- L'utilisateur clique sur sa musique [condition de fin 1]- Il clique sur pause [condition de fin 2]
Conditions de fin	<ol style="list-style-type: none">1) La musique se lance2) La lecture du morceau est mise en pause

Cas “Changer de titre”

Nom	Changer de titre
Objectif	Permettre à l'utilisateur de changer la musique en cours de lecture
Acteurs principaux	Utilisateur
Acteurs secondaires	/
Conditions initiales	- Une musique doit être en cours d'écoute ou sur pause
Scénario d'utilisation	- L'utilisateur choisit "musique précédente" [condition de fin 1] ou "musique suivante" [condition de fin 2]
Conditions de fin	1) La musique précédente est lancée 2) La musique suivante est lancée

Considération ergonomique



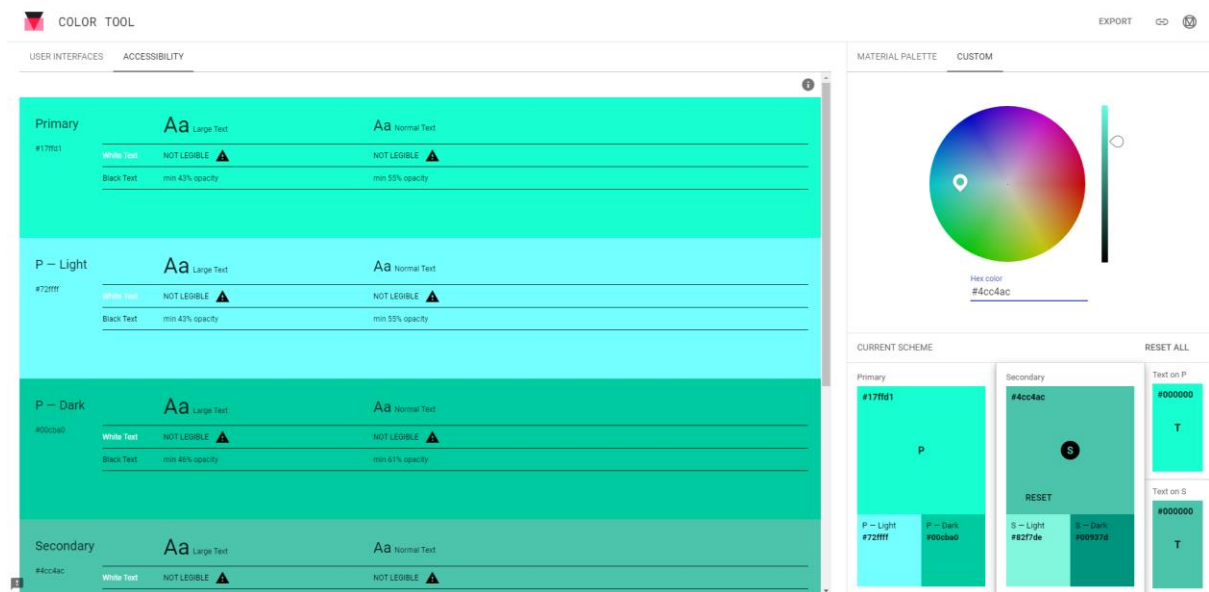
L'ergonomie de notre application respecte la loi de Fitts. Cette loi notifie que pour pouvoir atteindre sa cible dans un application, le chemin ne doit pas être trop compliqué. Sur notre application, chaque page et/ou fonctionnalité est accessible en moins de 3 clics.

La plupart des éléments de notre application sont cliquables, et quand l'utilisateur pour avoir des doutes sur l'utilisation d'un bouton, un petit texte apparaît pour lui expliquer l'utilisation de ce dernier. Aussi, des icônes sont présentes sur le bouton pour essayer de faire comprendre à l'utilisateur à quoi sert le bouton en question.

Les titres dans le menu sont les plus explicites possible pour que chaque utilisateur puisse comprendre et se retrouver dans la navigation au sein de l'application.

Prise en compte de l'accessibilité

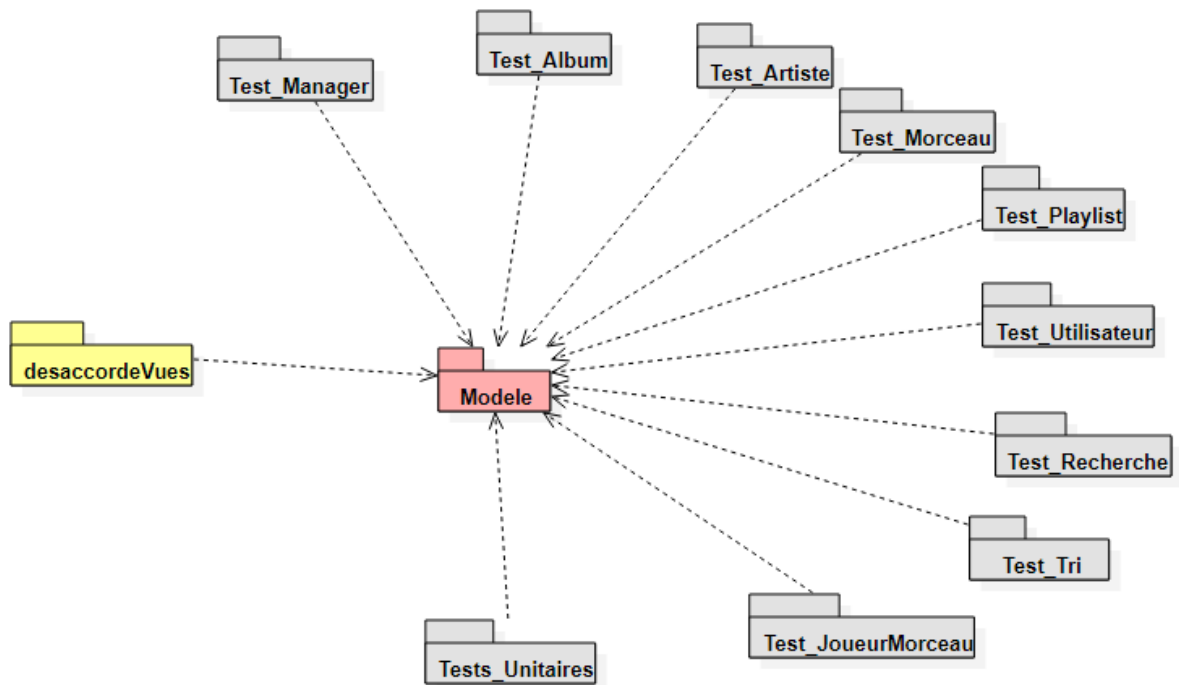
L'application a été désignée en faisant en sorte que n'importe quelle personne puisse utiliser confortablement votre application. Nous avons utilisé des outils comme Material.io, développé par Material Design pour pouvoir choisir les couleurs de l'application, en faisant en sorte qu'elles ne rentrent pas trop en conflit pour des personnes daltoniennes.



On peut voir sur cette capture d'écran les couleurs utilisées pour l'application. Pour chaque couleur de texte, des tests ont été effectués pour vérifier leur compatibilité avec la couleur de fond choisie.

Conception et Programmation Orientées Objets (C#, .NET)

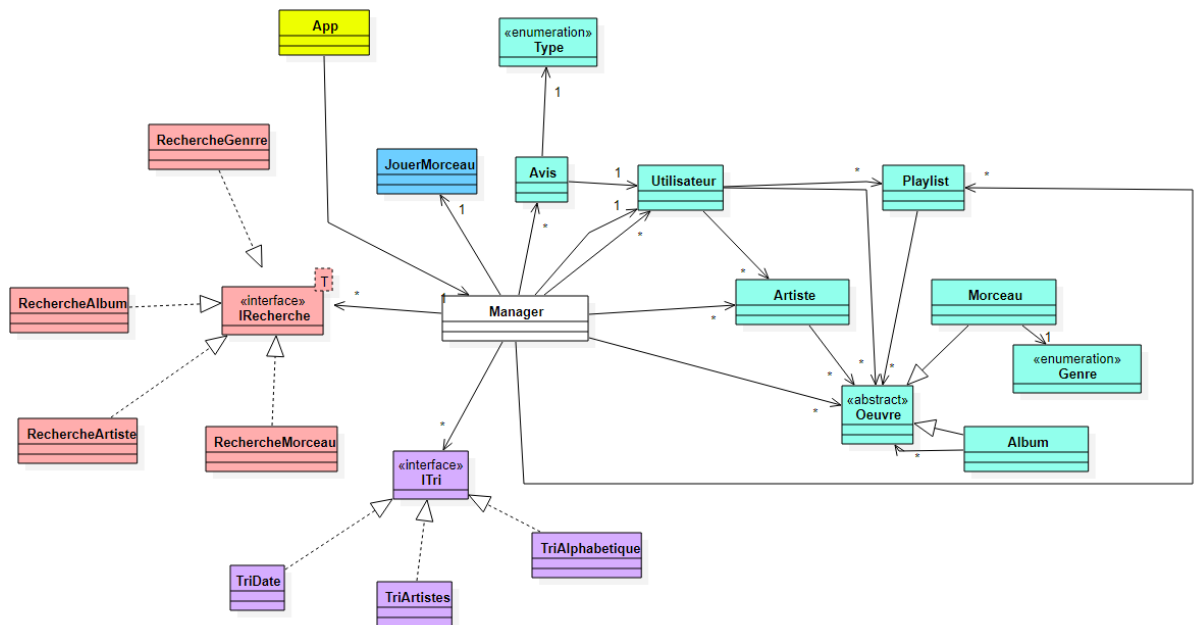
Diagramme de paquetage



Pour le diagramme de paquetage, chaque test fonctionnel ainsi que les tests unitaires font référence au modèle qui compose tout le code. Ainsi, chaque test fonctionnel permet de vérifier les fonctionnalités générales de la classe qui lui est associée. De plus, les tests unitaires permettent d'approfondir et de tester des parties très spécifiques de chaque classe. Cela permet de trouver un problème très rapidement et permet une vérification intégrale du code. Les vues font aussi référence au Modèle car ces dernières utilisent le code du Modèle.

Diagramme de classes

Diagramme général



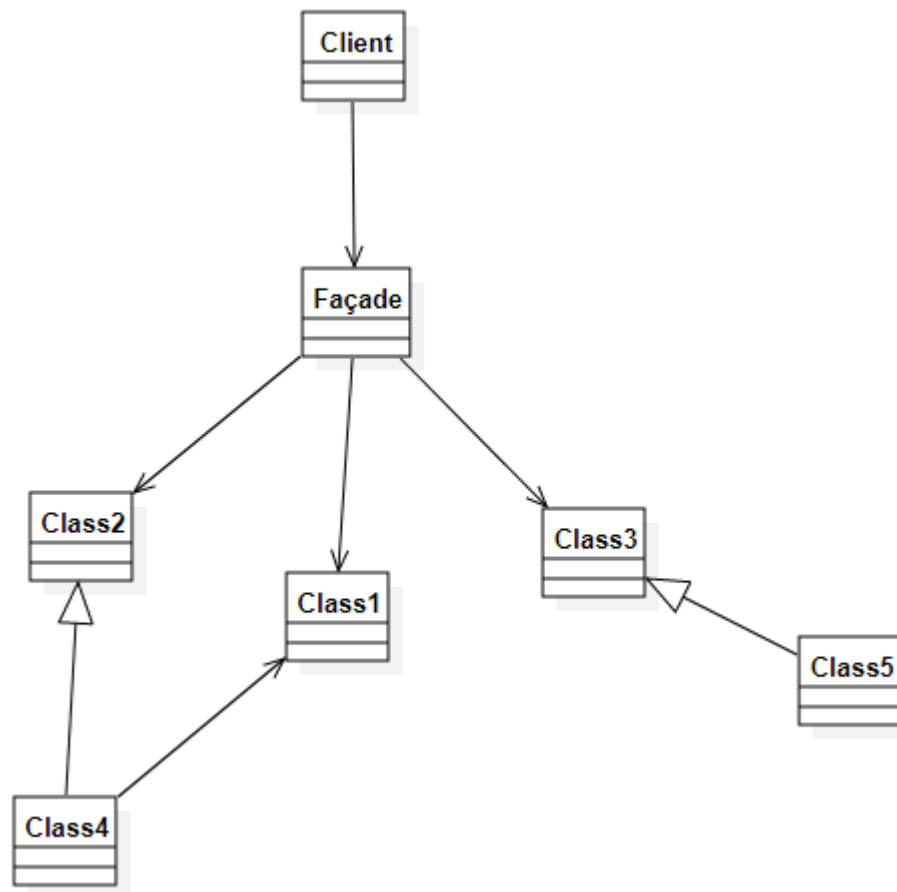
Le diagramme général présente le fonctionnement intégral de l'application. On voit ici une classe centrale qui est le Manager. Le Manager est basé sur le patron de conception de la Façade. Son fonctionnement est expliqué dans le paragraphe suivant. La classe Manager gère la plupart des autres classes principales, c'est elle qui gère le choix des données à afficher à travers l'application en appelant les différentes méthodes de tri et de recherche par l'intermédiaire des interfaces.

Elle enverra également les requêtes à la classe JoueurMorceau lorsque l'utilisateur veut écouter une musique ou modifier le son de cette dernière par exemple. Cette classe générale gère les utilisateurs ainsi que les listes d'artistes et des œuvres comprenant les Albums et les Playlists. Elle comporte en quelque sorte une bibliothèque d'utilisateurs, d'artistes et d'œuvres.

Dans ce diagramme, on voit apparaître la partie musique en cyan, la partie player en bleu, la partie recherche en rouge et pour finir la partie tri en violet.

Lors du lancement de l'application, App, qui définit l'application desaccordeVues, appelle un manager pour pouvoir mettre en application tout le code de ce dernier.

Patron de conception « Façade » du diagramme ci-dessus :



Le diagramme théorique ci-dessus présente le patron de conception “Façade”. Son but est de fournir une interface de plus haut niveau que les différentes interfaces d’une application, ce qui rend donc les différents sous-systèmes plus faciles à utiliser car tout est unifié au sein de la même classe.

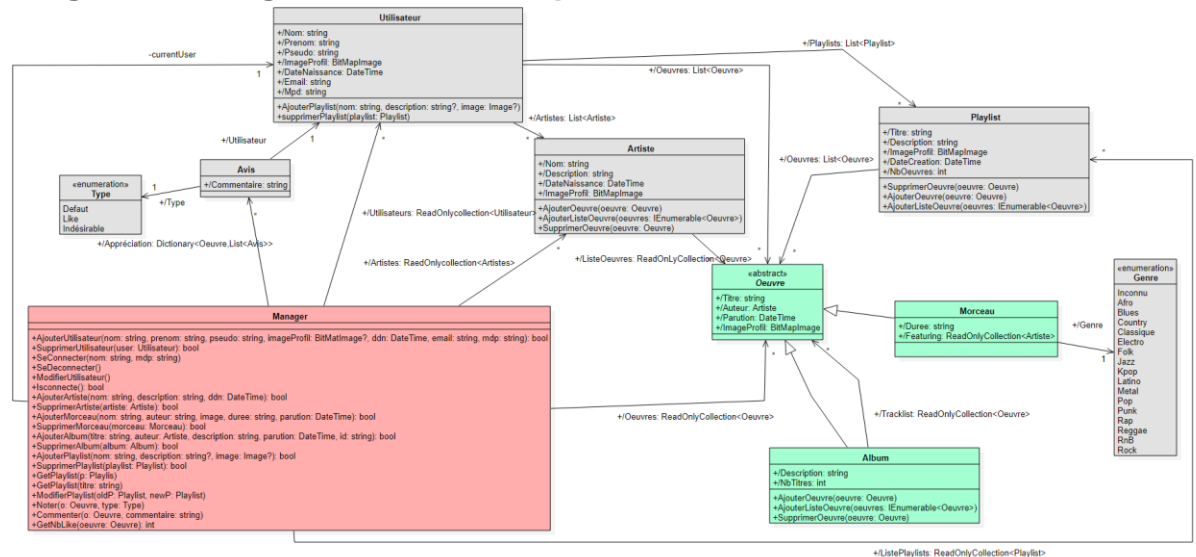
La façade, ici le manager, gère et permet d’intégrer une interface unique à l’ensemble des sous-classes. Ainsi, elle simplifie un système complexe en une seule classe qui permet de gérer le reste du code. Cette façade permet également de relier certaines classes entre elles et ainsi de déléguer des requêtes à d’autres classes. Ces sous-classes/systèmes ne font à aucun moment référence à la façade, ce sont eux qui vont recevoir les travaux et requêtes de la façade, ici le manager.

Dans notre cas, le Manager gère toutes les parties de l’application. Il gère la partie la plus complexe, la partie musique. Il simplifie son utilisation, génère une liste d’artistes et une liste d’œuvres, propose des méthodes permettant d’ajouter ou de supprimer des entités dans ces listes. Il permet aussi de faire cela avec des utilisateurs, fait appel aux classes utilisées. C’est également lui qui appellera les interfaces de tri et de recherche lorsque c’est nécessaire, faisant ainsi le lien entre la méthode de tri et les données à trier.

Ici, on a un client qui appelle la façade, ce dernier possède un élément façade, ce qui lui permet donc de posséder toutes les méthodes de cette façade. Le client

peut donc bénéficier de plusieurs façades par exemple sans que ces dernières ne connaissent leurs existences entre elles. Cela permet de simplifier le code du client en le fragmentant en plusieurs façades par exemple.

Diagramme de gestion des musiques



Ce diagramme est le plus gros et le plus important. Il comporte les collections et l'organisation des morceaux, artistes et listes de morceaux : Playlists et Albums. Tout part de la classe Manager qui gérera la plupart des collections. Avec ses méthodes, il pourra modifier chaque œuvre, artiste et playlist. C'est lui qui aura la liste générale des artistes et des playlists ainsi que la totalité des œuvres.

Pour commencer, le manager aura une liste d'utilisateurs qu'il parcourra lors d'une quelconque connexion. Il pourra créer un nouvel utilisateur lorsqu'un client sans compte voudra se connecter.

Chaque utilisateur possède une liste de playlists qui est celle qu'il aura créée. Il possède également une liste d'artistes et une liste d'œuvres. Ce sont les favoris choisis par l'utilisateur lui-même.

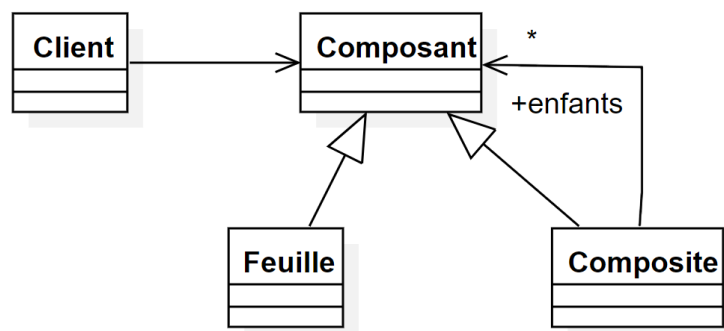
De plus, le manager possède un dictionnaire avec en clé une œuvre et en valeur, une liste d'avis. Ainsi, la clé œuvre aura comme valeur une liste d'avis qui lui fait référence. Ces avis sont écrits par les utilisateurs puis récupérés par le manager et ainsi classés dans le dictionnaire. Un avis aura un commentaire et un type qui peut être un Like ou un Indésirable, sinon il a une valeur défaut.

Enfin, la classe artiste aura une collection d'œuvres qui représente les albums et morceaux d'un artiste. Il possède plusieurs méthodes afin d'en ajouter/supprimer. La classe playlist aura une liste d'œuvres. Ainsi, l'utilisateur pourra créer des playlists personnalisées contenant différentes œuvres.

La partie en vert représente les classes morceaux et albums. Elles sont regroupées dans une classe œuvre à l'aide d'un composite que l'on verra ci-dessous.

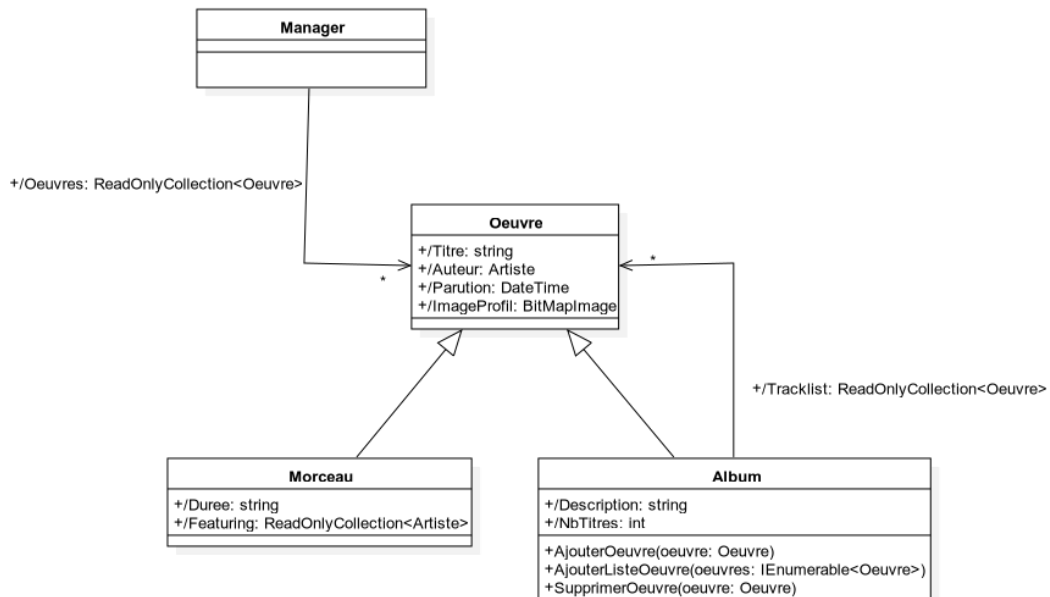
Patron de conception « Composite » du diagramme ci-dessus :

Dans le diagramme général, on utilise un patron de conception « Composite ». Il est illustré par le diagramme théorique suivant :



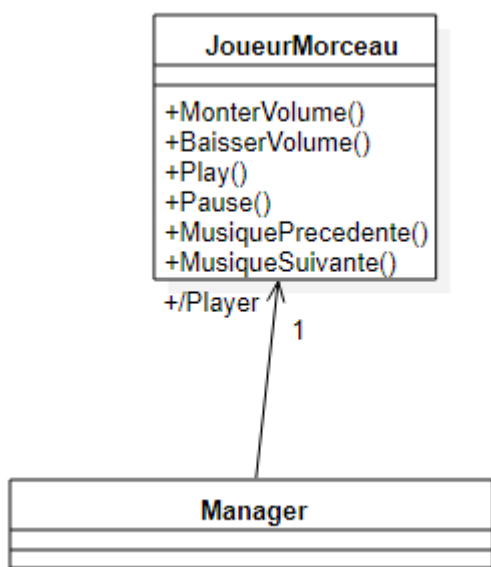
Le patron composite permet de structurer des objets en arborescence afin de créer une certaine hiérarchie. Dans notre cas, il est utilisé autour de la classe Œuvre qui est le composant, la classe album qui est le composite et la classe morceau qui est la feuille. Enfin, le manager gérera le tout et fera office de “client” sur le schéma ci-dessus.

Un composant est un élément commun aux classes qui héritent de cette dernière. Dans notre cas, un album ou un morceau sont tous deux des œuvres. Ils dérivent alors de cette classe pour pouvoir être définis comme des œuvres. Le composite hérite du composant, comme dit précédemment. Il peut aussi posséder ses propres méthodes mais il possède aussi une collection de composants qui sera ici une collection d'œuvres, donc soit des morceaux, soit des albums. Un album pourra être composé d'autres albums. On peut voir cela comme une extension d'un album. La feuille hérite simplement de “composant”, elle peut elle aussi avoir ses propres méthodes.



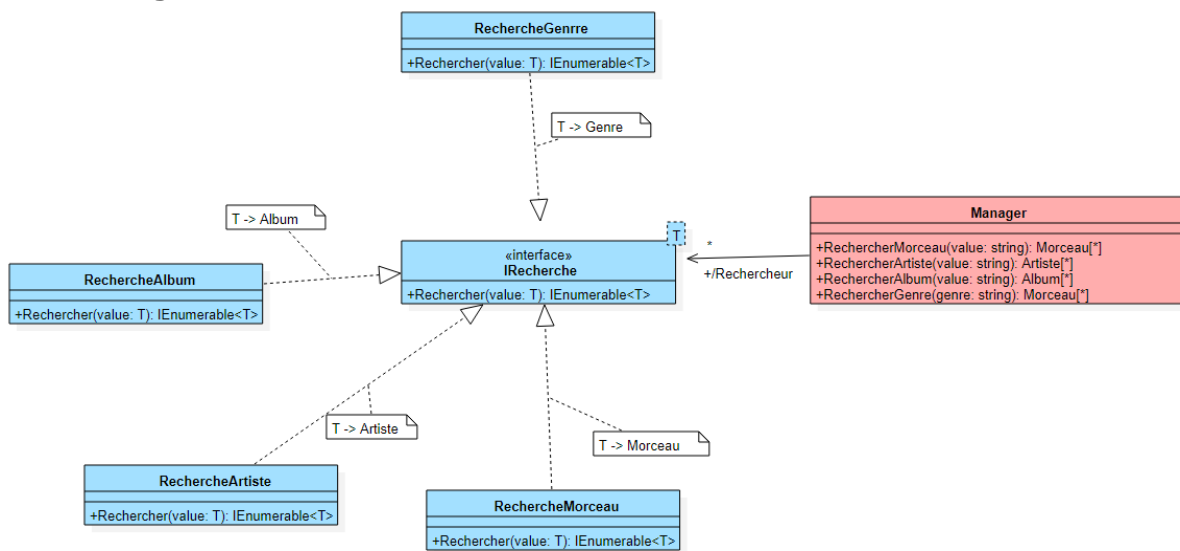
Chaque classe reste toutefois unique et possède ses propres méthodes. Ainsi, une œuvre peut être un album ou un morceau et un album possède une collection d'œuvres. Cette collection peut être composée de morceaux et/ou d'albums. Chaque œuvre possède un titre, un auteur, une date de parution et une image. Cependant, les morceaux possèdent une durée précise ainsi qu'un quelconque featuring s'il existe tandis que les albums possèdent en plus d'une liste d'œuvres une description et un nombre de titres. Les méthodes présentes dans la classe album permettent de gérer la liste d'œuvres.

Diagramme du player



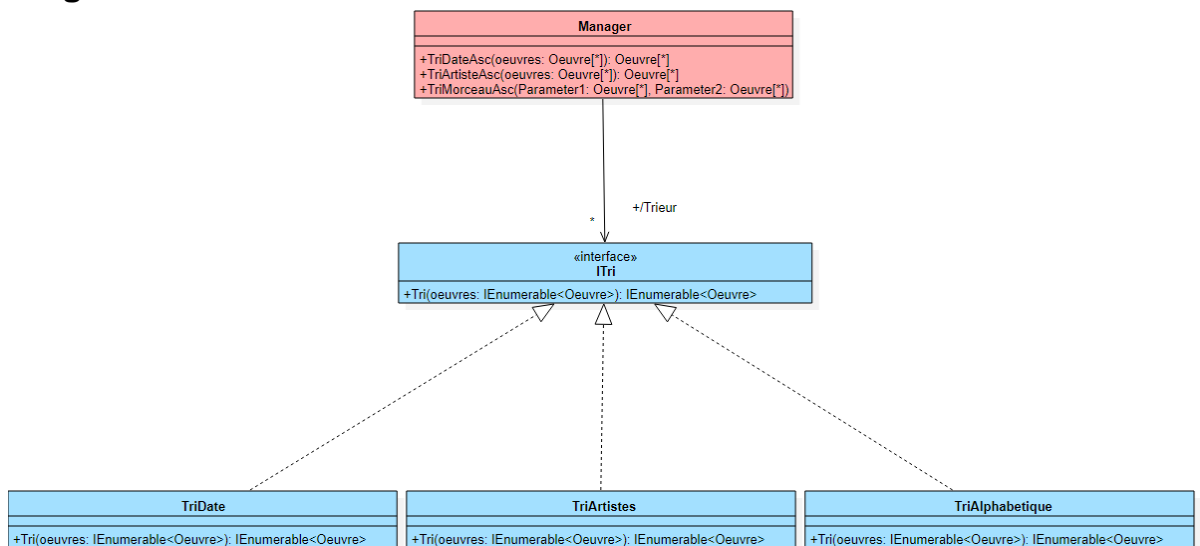
Le Player servira à lancer une musique lorsque l'utilisateur le demande. Chaque requête viendra donc du manager. La classe **JoueurMorceau** pourra alors effectuer différentes tâches à l'aide de méthodes comme `MonterVolume`, `BaisserVolume`, `Play` (qui lancera la musique), `Pause` (qui la mettra en pause) ou encore `MusiquePrecedente` et `MusiqueSuivante`.

Diagramme de recherche



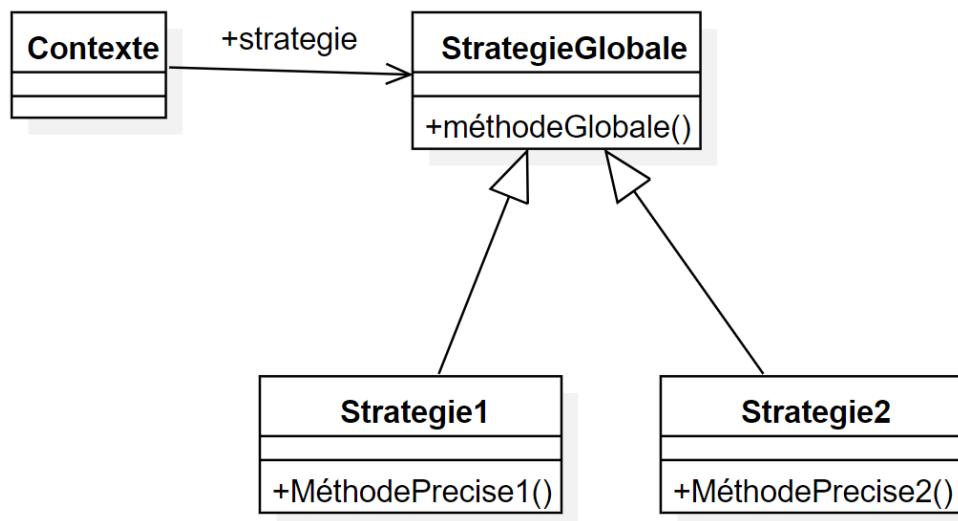
La recherche est déclenchée depuis la classe Manager. Toutes les recherches sont placées dans des classes différentes qui implémentent toutes l'interface **IRcherche**. Elles possèdent toutes une méthode de recherche demandant le type d'objet à rechercher et renvoie une collection de cet objet. Il y a ici un patron de conception "Stratégie" vu et expliqué en dessous.

Diagramme du tri



De même que pour la recherche, le tri est déclenché par le Manager. Chaque classe de tri implémente l'interface **ITri**. Le tri peut donc se faire par ordre alphabétique, par date ou par artiste. Chaque classe possède donc une méthode de tri propre demandant une collection et retournant la collection triée. Ici, comme pour le diagramme de recherche, un patron stratégie est réalisé :

Patron de conception « Stratégie » du diagramme ci-dessus :

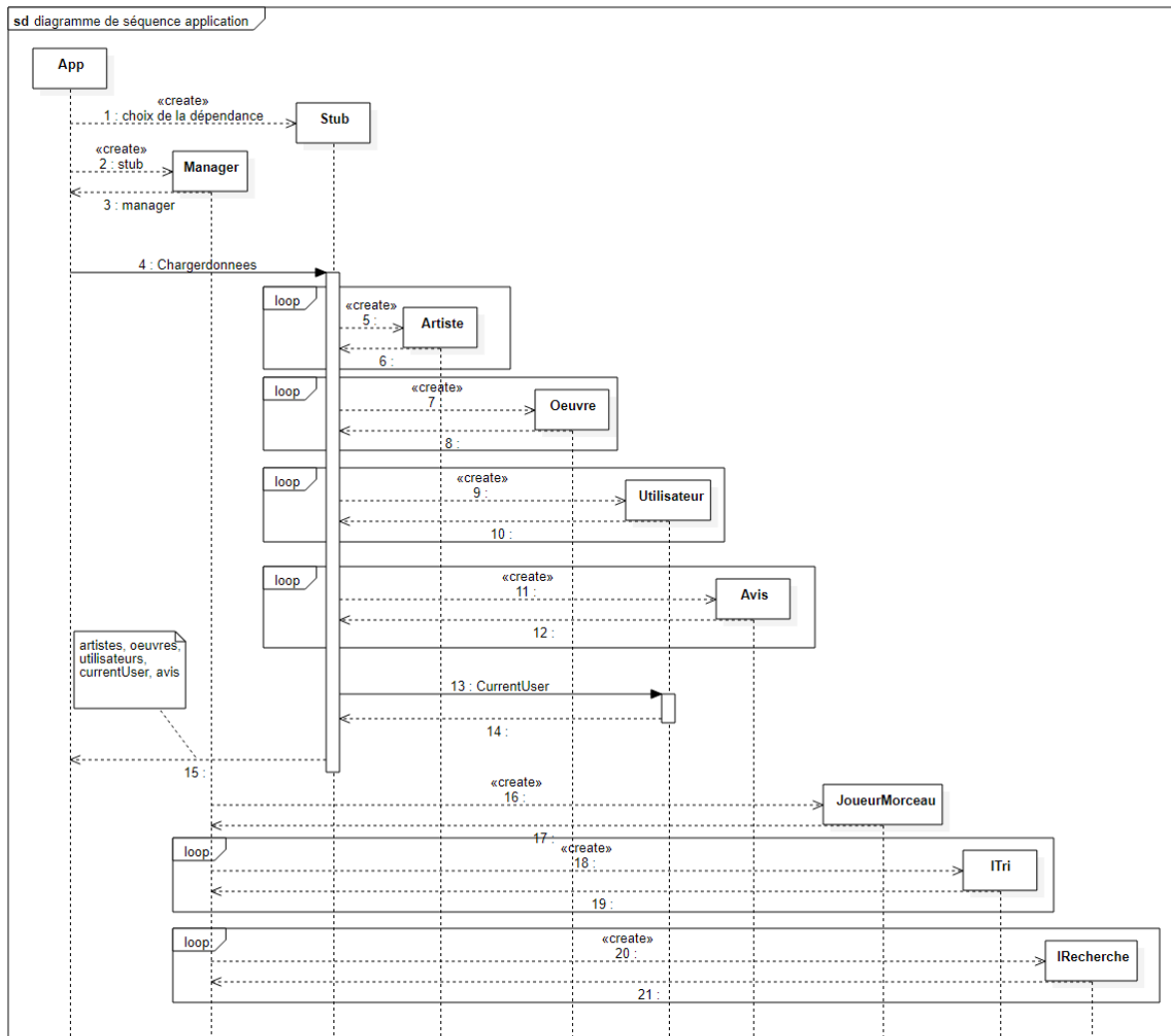


Le diagramme ci-dessus représente le patron Stratégie. Ce patron de conception consiste à encapsuler une famille d’algorithmes (dans notre cas, des algorithmes de tri et de recherche) et les rend interchangeables. Ainsi, chaque recherche est indépendante et évolue selon les requêtes que le manager enverra. La classe "StrategieGlobale" est une interface. Les classes stratégie 1 et 2 implémentent cette interface et précisent la méthode générale de l’interface. Ainsi, chaque classe se ressemble puisqu’elles ont une interface commune mais elles diffèrent dans les méthodes. La classe “contexte”, pour notre cas Manager, fera appel aux différentes stratégies par l’intermédiaire de l’interface et choisira ainsi quelle méthode il utilisera.

Dans le cas du diagramme de recherche, on va chercher dans les données chargées par l’injection de dépendance ce que l’on veut. On peut chercher parmi plusieurs types de données, des artistes, des œuvres ou alors des genres. On peut chercher un album, un morceau ou un artiste grâce à la barre de recherche, on tape le mot voulu et le chercheur nous présente alors différentes listes de morceaux, d’albums et d’artistes qui correspondent à la recherche. Pour la recherche par genre, on sélectionne le genre voulu dans la ComboBox et la recherche se lance automatiquement.

Dans le cas du diagramme de tri, on clique simplement sur l’intitulé de la ListView voulu, (titre, auteur, date, ...) pour pouvoir la trier selon la sélection de l’utilisateur.

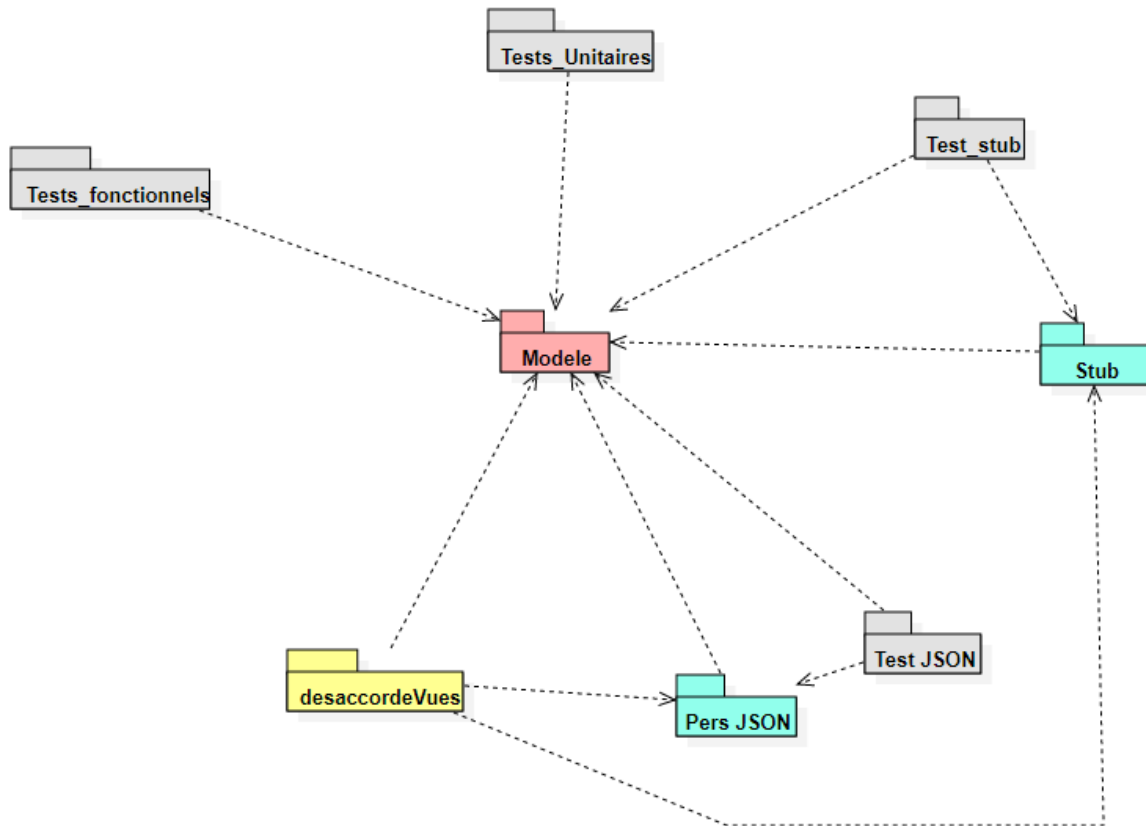
Diagramme de séquence



Le diagramme de séquence ci-dessus nous montre comment est générée l'application de cette dernière au lancement de l'application. L'application crée dans un premier temps et on choisit ensuite une dépendance. Le manager est ensuite créé avec la dépendance voulue. On appelle ensuite la méthode ChargerDonnees du manager, ce qui charge alors les données de la dépendance. On charge les collections d'artistes, d'œuvres, d'utilisateurs et d'avis. On charge aussi l'utilisateur qui était connecté lors de la dernière utilisation de l'application sous la variable CurrentUser. Le manager charge ensuite le player de musique puis les différentes stratégies de tri de morceaux, puis ensuite les différentes stratégies de recherche de l'application.

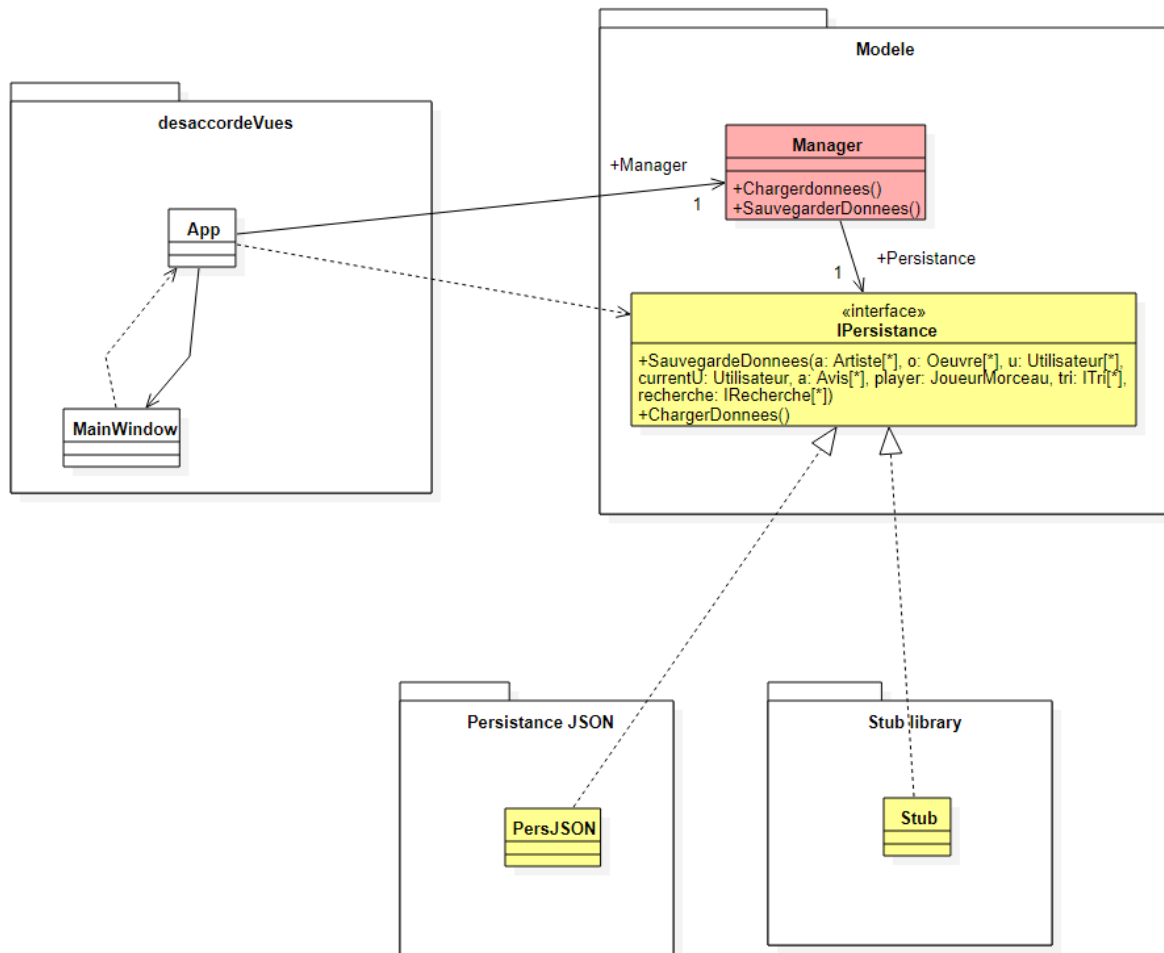
Projet Tutoré S2

diagramme de paquetage mettant en avant la partie persistance

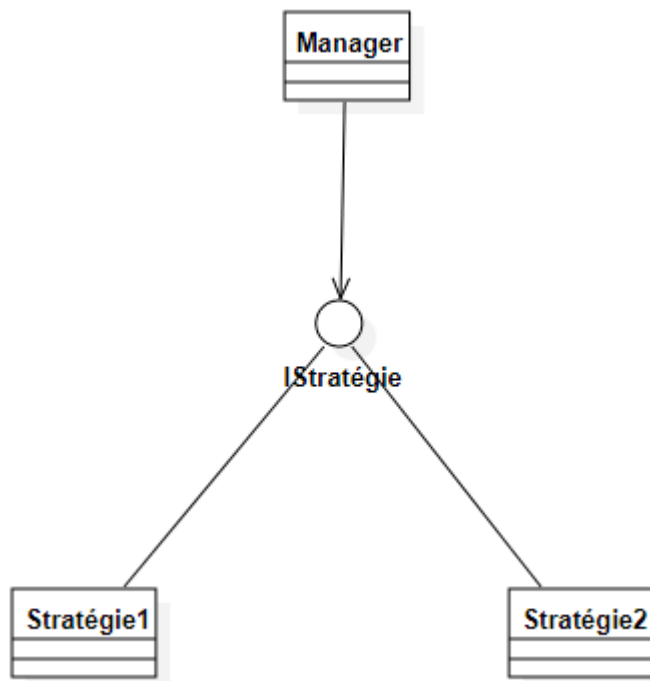


Le diagramme de paquetage orienté persistance nous montre comment fonctionne notre application avec des fichiers de données. Dans un premier temps, on utilise le stub avec des données fictives pour vérifier le bon fonctionnement de nos vues avec le binding des données. Ensuite, on met en place un fichier JSON avec cette fois-ci les vraies données de l'application, Ce fichier nous permettra de charger les données de l'application et de sauvegarder par la même occasion les changements que l'utilisateur a effectués pendant sa session d'utilisation de l'application.

diagramme de classes mettant en avant la partie persistance

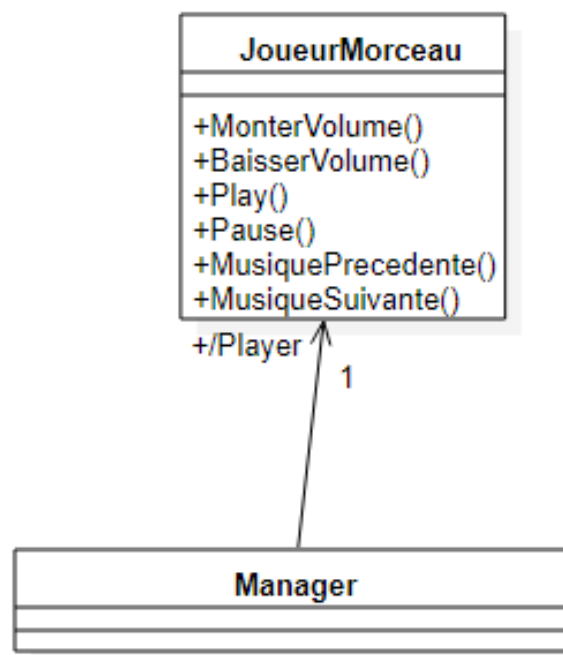


Le diagramme de classes orienté persistance représente le fonctionnement de la persistance. On voit dans un premier temps que L'application crée un Manager (dans le Modèle, seul le Manager est représenté car le reste du Modèle n'est pas le sujet à traiter ici). A son lancement, le manager se crée en appelant une méthode de persistance. La persistance est mise en place ici grâce au patron de conception stratégie. On voit que la persistance est définie dans le modèle grâce à l'interface IPersistence qui possède deux méthodes : SauvegarderDonnees et ChargerDonnees. Le but de cette interface est de pouvoir être redéfinie, ce qui nous permettra alors d'avoir différentes injections de données et donc différentes solutions pour charger et sauvegarder les informations de notre application. Pour faire des tests sur la partie fonctionnelle de notre application, on pourra utiliser des fausses données grâce au stub. Par la suite, on pourra utiliser des "vraies" injections telles que JSON ou alors une base de données.



La stratégie est utilisée ici pour pouvoir choisir différentes stratégies de persistance. Lorsque l'on crée un Manager, ce dernier crée une IPersistence. Cette IPersistence est implémentée par différentes stratégies de persistance, ce qui permet alors de choisir la persistance adéquate.

diagramme de classes sur votre (vos) partie(s) ajoutée(s)



La partie ajoutée pour notre application est un joueur de morceau qui lancera le morceau sélectionné. On pourrait changer le morceau en cours de lecture en utilisant les méthodes musique suivante/précédente. Si aucun morceau n'est relancé à la suite, le prochain morceau se lance automatiquement. On peut donc lancer une

liste de morceaux et ainsi l'écouter sans rien cliquer. Cela est donc possible pour les albums et les playlists. Les boutons play et pause arrêteront et lanceront la musique sélectionnée et deux autres méthodes permettent de modifier le volume.

On pourra aussi écouter les playlists en lecture aléatoire via un bouton dédié. Toutes ces méthodes auront un aspect visuel à travers des boutons en bas de l'écran sur un bandeau avec le nom de la musique sélectionnée, les différents boutons de navigation ainsi que de changement de volume.