
	<b>Cahier des spécifications fonctionnelles</b> <b>10.02.22</b> <i>- Projet Répertoire musical API -</i>	Rédaction: FL - 09.02.2022
		Validation: EF


# Cahier de spécifications fonctionnelles

**PROJET: Répertoire musical API**

	<b>Cahier des spécifications fonctionnelles</b> <b>10.02.22</b> <i>- Projet Répertoire musical API -</i>	Rédaction: FL - 09.02.2022
		Validation: EF

## Table des matières

1.	Introduction et cadre du projet	3
I.	Résumé du projet	3
II.	Contexte et intervenant	3
III.	Objectifs	3
IV.	Enjeux	3
V.	Livrables	3
VI.	Critères de succès mesurables	3
VII.	Planning	3
VIII.	Risque lié au projet	3
2.	Schéma de base de données	4
3.	Fonctionnalités	4
I.	Étapes et fonctions	4
	Étape 1 : Compléter le schéma	4
	Étape 2 : Création des modèles	4
	Étape 3 : Route et contrôleurs	5
	Étape 4 : Identifier les requêtes	5
	Étape 5 : Ajout de fonctionnalité	5
	Étape 6 : Améliorer l'application (EN OPTION)	6
	Étape 7 : Déploiement	6
4.	Méthodologie	7
5.	Limitations et contraintes	7
I.	Choix technologique	7
II.	Installation et outils	7
6.	Compétences et savoir-faire visé	8

	<b>Cahier des spécifications fonctionnelles</b> <b>10.02.22</b> <i>- Projet Répertoire musical API -</i>	Rédaction: FL - 09.02.2022
		Validation: EF

## 1. Introduction et cadre du projet

### I. Résumé du projet

Créer une API back-end pour une application qui répertorie les musiques et les artistes

### II. Contexte et intervenant

Un client vous a demandé de mettre en place un répertoire qui va gérer toutes ses musiques préférées. Pour cela, vous avez reçu d'un ami un schéma de base de données un peu simplifié et il vous a également envoyé un code en PHP qu'il utilise pour ses APIs.

### III. Objectifs

A partir des éléments qui vous sont remis, créer un back-end qui permet d'offrir à votre futur front-end les endpoints nécessaires pour gérer et récupérer les informations de la base de données.

### IV. Enjeux

Création d'un back-end qui sera ensuite utiliser pour la réalisation d'un front-end

### V. Livrables

Code PHP

### VI. Critères de succès mesurables


A partir de Postman ou un autre logiciel similaire, vérifier que chaque endpoint répond avec succès.

### VII. Planning

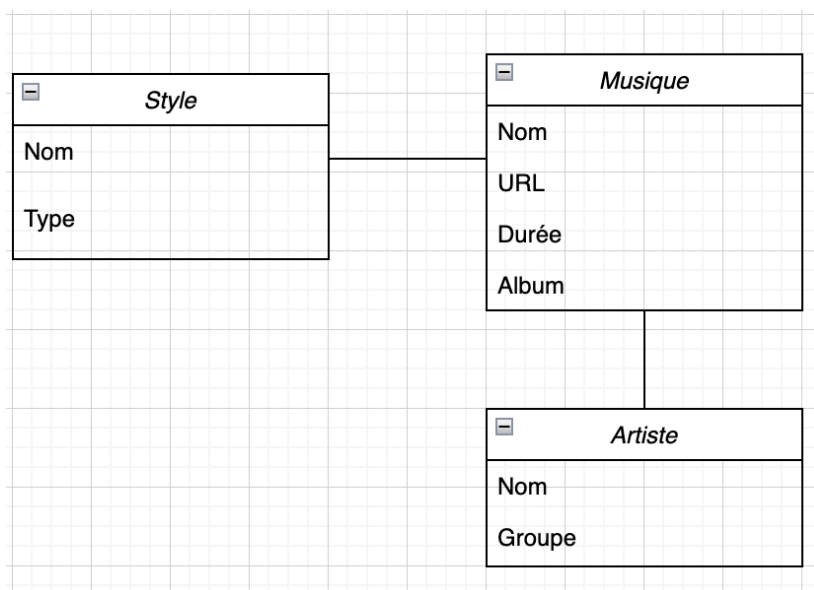
Le projet est sur 3,5 jours, c'est à vous de vous organiser entre équipe pour être dans les délais

### VIII. Risque lié au projet

Si le back-end n'est pas terminé, cela aura des conséquences pour la réalisation du projet. C'est la raison pour laquelle il sera essentiel de prioriser le développement en fournissant au minimum certains endpoints fonctionnels. Par ailleurs, il faudra prévoir avant la fin du projet de déployer le back-end sur un hébergement externe.

	<b>Cahier des spécifications fonctionnelles</b> <b>10.02.22</b> <i>- Projet Répertoire musical API -</i>	Rédaction: FL - 09.02.2022
		Validation: EF

## 2. Schéma de base de données



## 3. Fonctionnalités

### I. Étapes et fonctions


#### Étape 1 : Compléter le schéma

En fonction SQL qui vous a été fourni, compléter le schéma ci-dessus :

- Ajouter les champs « techniques » manquants
- Ajouter les relations (0..1, 1..1, 1..n ou n..m)
- Ajouter le type pour chaque champ selon les types suivant : **int(xx)**, **string(xx)**, **date()**, **time()**

#### Étape 2 : Création des modèles

- Créer 3 modèles « style », « musique » et « artiste » avec un CRUD pour chacun (Lister avec pagination, Créer, Récupérer une entrée d'après son ID, Mettre à jour, Supprimer). Attention ! Assurez-vous avant de supprimer une entrée liée à une autre relation que la relation soit préalablement supprimée.

	<b>Cahier des spécifications fonctionnelles</b> <b>10.02.22</b> <i>- Projet Répertoire musical API -</i>	Rédaction: FL - 09.02.2022
		Validation: EF

### Étape 3 : Route et contrôleurs

- Créer 3 contrôleurs (1 pour chaque modèle décrit ci-dessus)
- Ajouter les différentes routes pour chaque contrôleur

### Étape 4 : Identifier les requêtes

- Décrivez ce que les requêtes suivantes vont réaliser :

```
SELECT a.nom, a.groupe, m.nom FROM artiste a
INNER JOIN artiste_musique am ON am.artiste_id = a.id
INNER JOIN musique m ON m.id = am.musique_id
WHERE a.id = $artisteID
```


```
SELECT * FROM musique m
INNER JOIN artiste_musique am ON am.musique_id = m.id
INNER JOIN artiste a ON a.id = am.artiste_id
WHERE m.id = $musiqueID
```

```
SELECT m.id, m.nom, m.durée, m.album, CONCAT(s.nom, '/', s.type) as
style
FROM musique m
INNER JOIN style s ON s.id = m.style_id
ORDER BY m.nom ASC LIMIT $offset, $limit
```

```
SELECT m.id, m.nom, m.durée, m.album, CONCAT(s.nom, '/', s.type) as
style
FROM style s
INNER JOIN musique m ON s.id = m.style_id
WHERE s.id = $styleID
```

### Étape 5 : Ajout de fonctionnalité

- Au choix, adapter le code existant en modifiant les requêtes existantes par celle-ci-dessus ou rajouter des routes qui font appel à ces requête (via un contrôleur et le modèle adéquate)

	<b>Cahier des spécifications fonctionnelles</b> <b>10.02.22</b> <i>- Projet Répertoire musical API -</i>	Rédaction: FL - 09.02.2022
		Validation: EF

- Ajouter également 2 nouvelles routes qui permettent de rattacher une musique à un artiste ou de retirer une musique d'un artiste (sans supprimer ni la musique, ni l'artiste)


### Étape 6 : Améliorer l'application (EN OPTION)

Il existe plusieurs éléments pouvant être améliorés, notamment :

- Le routage ne respecte pas les normes REST
- Il n'existe pas de fonction générique pour la validation de champ
- L'application affiche des erreurs trop détaillées, ces informations peuvent être importantes pour un log (qu'elle soit enregistrée dans un fichier avec la date et l'heure) mais seul un message court devrait être retourné au front-end.

### Étape 7 : Déploiement

Déployer son site sur un hébergement externe

	<b>Cahier des spécifications fonctionnelles</b> <b>10.02.22</b> <i>- Projet Répertoire musical API -</i>	Rédaction: FL - 09.02.2022
		Validation: EF

## 4. Méthodologie

L'ensemble du projet doit être développé selon la méthode Agile + SCRUM.

La semaine sera organisée avec chaque début de matinée un stand-up :

- Prise de température du groupe
- Planification de la journée
- Partage des tâches
- Identification d'aide et soutien pour les tâches

Puis, à chaque fin de journée, une rétrospection :

- Quels ont été les obstacles (collaboratif ou technique) ?
- Y-a-t-il du retard ?
- Quels sont les succès ?

Chaque développeur-euse aura sa branche. On doit pouvoir remonter celles-ci pour repartir d'une étape en particulier.


## 5. Limitations et contraintes

### I. Choix technologique

- PHP
- MySQL

### II. Installation et outils

- Repository GitHub
- Docker avec Apache ou Nginx + PHP-FPM et un hébergement externe

	<b>Cahier des spécifications fonctionnelles</b> <b>10.02.22</b> <i>- Projet Répertoire musical API -</i>	Rédaction: FL - 09.02.2022
		Validation: EF

## 6. Compétences et savoir-faire visé

1. Lire des cahiers des charges/spécifications
2. Appréhender le cycle de vie d'un projet
3. Identifier les tâches dans des cahiers des charges et spécifications et les reporter dans les outils AGILE
4. Utiliser les outils de versioning et de collaboration
5. Traiter des données reçues
6. Lire et écrire dans un fichier
7. Vérifier le format, le type et la valeur des données reçues (validateur)
8. Reformater et nettoyer les données reçues pour les rendre compatibles
9. Accéder aux ressources de son site via différents protocoles
10. Déployer son site en local via un système de containerisation
11. Transposer un schéma de BDD en classe
12. Appliquer les concepts de POO avancé dans un langage front et/ou back
13. Lire des schémas de BDD
14. Comprendre et utiliser des fonctionnalités SCRUD existantes
15. Récupérer des données et les traiter