

ABIDA Wilem

Étudiant de troisième année

BACHELOR

Chargé de Projets Systèmes Informatiques Appliqués

RAPPORT DE PROJET

Mission : Application CRUD sur Symfony

Entreprise en charge de la réalisation : WA Services

Client : Union Nationale des Concerts

Année universitaire 2021-2022

SOMMAIRE

INTRODUCTION.....	3
1. PRÉSENTATION DU PROJET.....	4
1.1. Appel d'offre et mise en place interne.....	4
1.2. Technologies utilisées.....	5
1.3. Produit final.....	10
2. GESTION DE PROJET.....	12
2.1. Présentation de l'équipe.....	12
2.2. Méthode agile et planification.....	14
2.3. Étapes de réalisation.....	16
CONCLUSION.....	17
TABLE DES ILLUSTRATIONS.....	18
GLOSSAIRE.....	19
REFERENCES.....	22
ANNEXES.....	23

INTRODUCTION

Dans le cadre de l'augmentation du nombre d'événements après les confinements en France, le gouvernement a réalisé un appel d'offres* concernant une application qui permettra de gérer une base de données. Il est réalisé pour l'Union Nationale des Concerts, qui veut centraliser toutes les données et pouvoir éventuellement retracer certains acteurs des événements, voir ses clients en cas de troubles dans les concerts.

Après avoir réalisé l'appel d'offre, plusieurs entreprises ont réalisé des réponses, présenté leurs réalisations et ont proposé des prix à l'UNC*. C'est WA Services* qui s'est démarqué en répondant à cet appel d'offre. Il s'agit d'une petite entreprise, qui réalise des prestations de services et qui se compose d'une dizaine d'employés.

En obtenant cet appel d'offre, l'entreprise s'est engagée à livrer le produit sous un délai de 3 semaines, un délai plutôt court mais au vu de l'offre, l'entreprise a réuni tous ses moyens pour offrir le délai le plus court.

Ce rapport retrace le travail réalisé par WA Services, sa méthode de travail afin d'inspirer toute personne qui souhaite se former dans le domaine du développement. Il permettra de présenter le projet dans sa globalité, en commençant par l'appel d'offre et la mise en place des suites de son obtention. Les technologies utilisées y seront aussi expliquées dans le détail, ainsi que le produit final.

Par la suite, la lumière sera mise sur la gestion du projet, à travers une méthodologie agile*, qui est de plus en plus utilisée dans les équipes de développement. La planification, qui est indispensable pour un projet d'une telle ampleur, sera aussi expliquée dans le détail, à l'aide d'un diagramme de gantt*.

Enfin, seront présentées les étapes de réalisation et la présentation de l'équipe qui a travaillé sur ce projet.

1. PRÉSENTATION DU PROJET

1.1. Appel d'offre et mise en place interne

A la suite d'un appel d'offres du gouvernement pour l'UNC, notre société WA Services a répondu à cet appel et l'a obtenu. En faisant une offre abordable et donnant un délai de livraison rapide, la réponse du client a été unanime. Nous avons fait la promesse de livrer l'application sous un délai maximum de 3 semaines.

Cela est plutôt court pour le développement d'une application, mais nous avons réuni les meilleurs éléments de la société pour réaliser cette application. Cela garantit au client un produit de qualité et durable dans le temps. En effet, toutes nos solutions font état d'un suivi et d'une maintenance durant une période de 3 ans, renouvelable.

La mise en place interne a débuté dès l'obtention du développement de l'application. La première étape a été de réunir 4 personnes chargées du développement et des tests de l'application, un scrum master* et un product owner*. Durant les premiers jours le product owner s'est chargé de rédiger les user stories* et de réaliser les maquettes de l'application.

Une fois terminées, les travaux du product owner ont été par le biais du scrum master transmis à l'équipe avec qu'ils en prennent connaissance. Une réunion pour la planification des sprints* a ensuite eût lieu avec toute l'équipe, c'est lors de celle-ci qu'a été préparé le carnet de sprint qui contient les travaux à faire durant les 3 sprints. Les réunions de fin de sprint ont aussi été définies pour tous les jeudis, au vu de la durée des sprints qui est d'1 semaine.

1.2. Technologies utilisées

Dans le cadre de la réalisation de l'application, notre équipe a décidé d'utiliser Docker* pour pouvoir développer l'application depuis n'importe quel poste ou système d'exploitation et ne pas rencontrer de problème de versions. Le versioning de l'application a été réalisé à travers la plateforme Github*, cela permet à l'équipe de travailler à plusieurs en même temps sur le projet, de réaliser de la revue de code et ainsi garder une version stable sur la branche principale du projet. Enfin, l'équipe a décidé d'utiliser Symfony*, avec le bundle* EasyAdmin*. Cela permet un développement plus rapide et plus fluide.

```
mysql:
  container_name: "mysql-database"
  image: 'mysql:latest'
  environment:
    - MYSQL_DATABASE=${MYSQL_DATABASE}
    - MYSQL_USER=${MYSQL_USER}
    - MYSQL_PASSWORD=${MYSQL_PASSWORD}
    - MYSQL_ROOT_PASSWORD=${MYSQL_ROOT_PASSWORD}
  volumes:
    - mysql-data:/var/lib/mysql:rw
  ports:
    # To allow the host machine to access the ports below, modify the lines below.
    # For example, to allow the host to connect to port 3306 on the container, you would change
    # "3306" to "3306:3306". Where the first port is exposed to the host and the second is the container port.
    # See https://docs.docker.com/compose/compose-file/compose-file-v3/#ports for more information.
    - '3306:3306'
```

Figure 1 : Configuration docker pour la base de données

Pour l'utilisation de Docker, il y a été placé une base de donnée mySQL* ainsi qu'un conteneur qui contient php*. Cela est suffisant pour l'utilisation que nous allons faire de l'application. La configuration a été faite dans le fichier Dockerfile*, mais les commandes peuvent s'avérer longues et à rallonge. L'utilisation d'un Makefile a donc été indispensable dans le projet. Mais Docker est très complet et possède un système que l'on nomme : docker-compose*. Nous l'avons donc ajouté au projet, puisqu'il permet de rendre l'utilisation de Docker encore plus simple et les modifications des configurations encore plus. Cela a permis de mettre très vite en place le projet, malgré les bugs rencontrés, Docker tout comme Symfony possède une large communauté et les réponses se trouvent facilement avec quelques recherches.

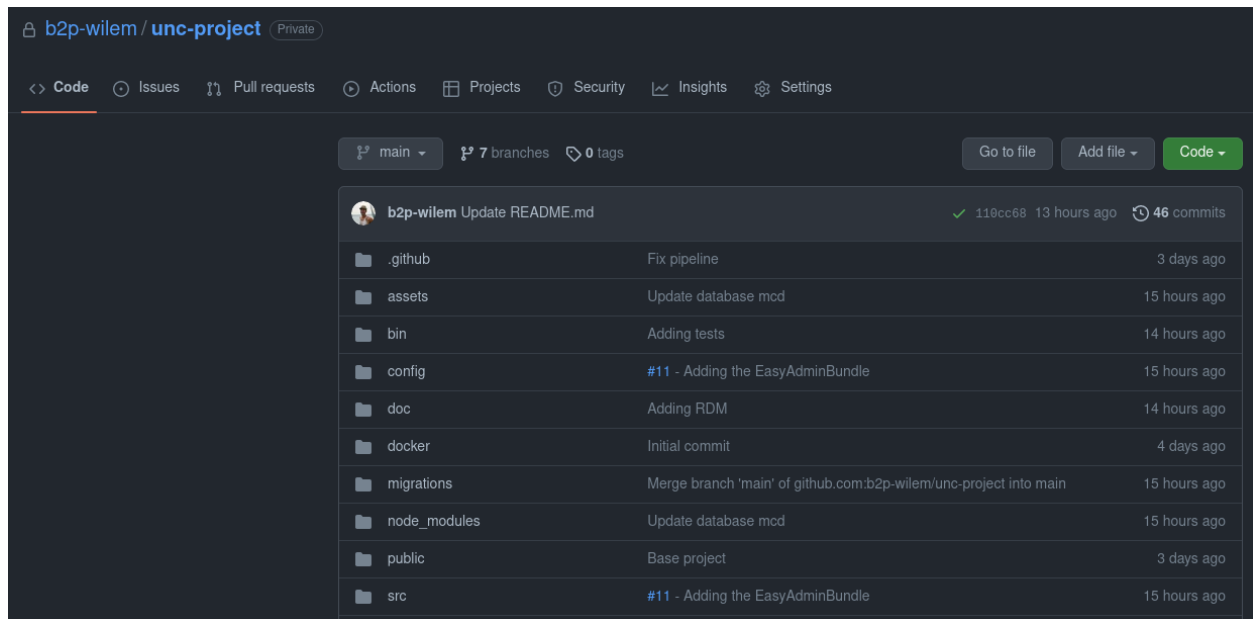


Figure 2 : Dépôt Github de l'application

Github a beaucoup servi dans la rédaction d'issue et le traitement de celles-ci à travers des merges request*. Le projet s'y trouve complètement dessus avec une documentation, et tous les schémas liés à l'application. Sa mise en place a été plutôt rapide avec notre équipe et celle-ci a suivi un Git Flow*, propre à notre société, qui revient à garder la branche principale toujours prête et fonctionnelle, et développer les fonctionnalités sur d'autres branches pour ainsi demander à les fusionner dans la branche principale une fois prêtes.

Une fonctionnalité importante et qui nous a beaucoup servi durant ce projet : l'approche CI/CD*, elle permet d'automatiser le développement des applications. Il s'agit d'éléments de surveillance qui vont s'assurer, lorsque l'on envoie sur la branche principale du projet du nouveau contenu, que l'application fonctionne toujours. Durant toute l'intégration et le développement, en plus de tests unitaires*, cela va vérifier que l'application est toujours fonctionnelle et prête à être déployée. C'est un système très

complet, qui, bien exploité peut permettre de trouver le moindre bug sur l'application et ainsi éviter de perdre beaucoup de temps à chercher la cause du problème.

52 workflow runs

Event ▾	Status ▾	Branch ▾	Actor ▾
✓ Update README.md CI #52: Commit 110cc68 pushed by b2p-wilem	main	14 hours ago 3m 59s	...
✓ Update README.md CI #51: Commit 369d3ce pushed by b2p-wilem	main	14 hours ago 4m 34s	...
✓ Merge pull request #13 from b2p-wilem/ajout-tests CI #50: Commit a2ac0b2 pushed by b2p-wilem	main	14 hours ago 3m 56s	...
✓ Adding tests CI #49: Pull request #13 opened by b2p-wilem	ajout-tests	14 hours ago 4m 20s	...

Figure 3 : CI/CD de l'application sur GitHub

Enfin, pour l'utilisation de Symfony, cela a été imposé par le client, mais notre équipe était plutôt satisfaite, puisqu'elle considère Symfony comme le meilleur framework* backend*. A travers une communauté riche, tout comme les fonctionnalités du framework, il est plutôt simple d'arriver au résultat attendu par le client. Dans notre contexte, nous avons tout d'abord créer les entités, puis intégrer le système de crud*, pour ensuite y ajouter EasyAdmin, pour l'UI* mais aussi pour une gestion plus compréhensive et plus simple des données par l'utilisateur.

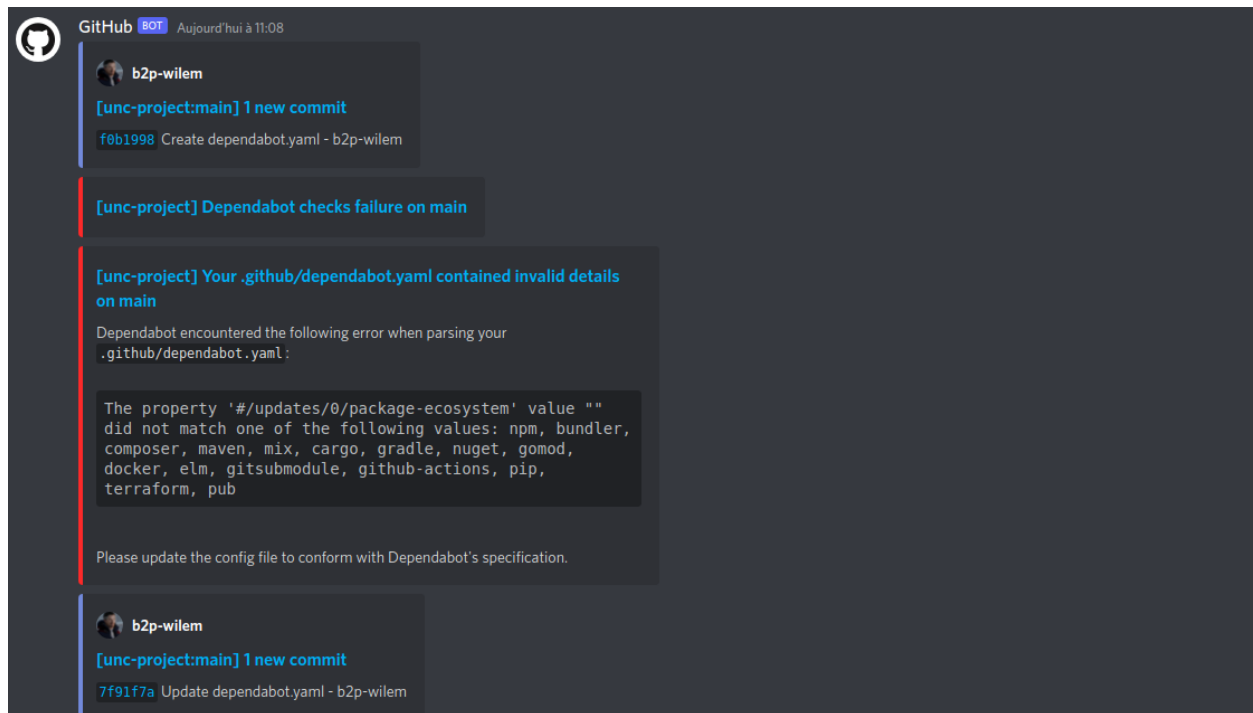


Figure 4 : Bot qui notifie l'activité sur GitHub sur un serveur Discord

Webhooks avec Github, qui sont liés au serveur discord de WA Services, sur lequel se passent toutes les communications et les daily. Cela permet d'avoir des notifications sur toute l'activité du dépôt Github et donc d'avoir un suivi en plus lorsque de l'activité s'y passe.

Cela peut être très utile pour l'équipe de développement, puisqu'elle peut retracer la version qui a provoqué le dysfonctionnement de l'application, en plus des tests, de la CI/CD et de l'historique d'activité de GitHub.

Ajouter ce type de fonctionnalités dans une application n'est pas primordiale, ni indispensable, mais permet d'assurer encore plus de couverture et de surveillance sur le projet.

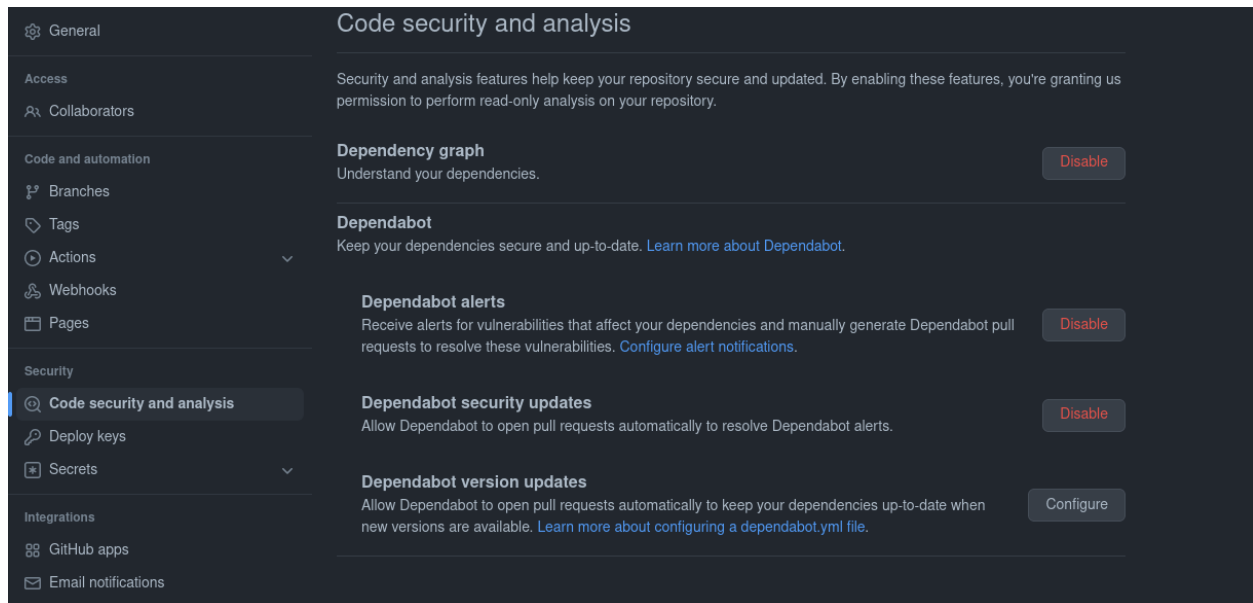


Figure 5 : Dependabot configuration

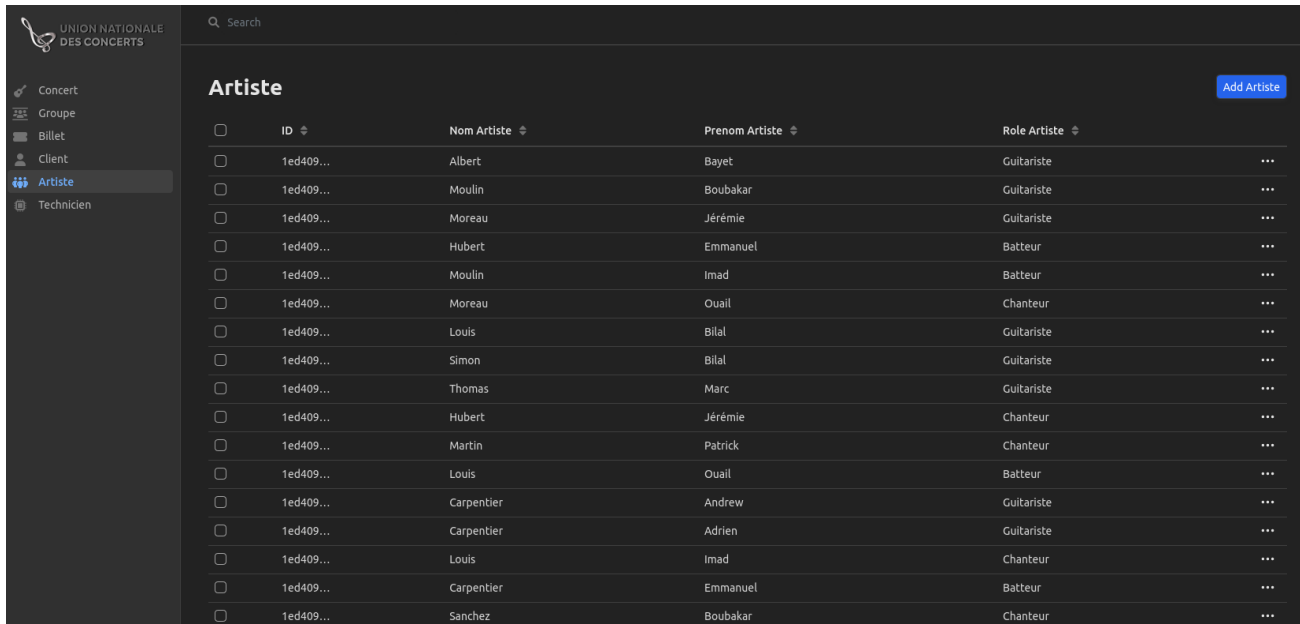
Dependabot est une aide précieuse dans un projet, puisqu'il permet de scanner les fichiers de dépendances, afin de vérifier s'ils ont des vulnérabilités ou s'ils nécessitent d'être mise à jour. L'ajouter permet de moins se soucier de mettre à jour les dépendances ou de passer à côté de vulnérabilités.

CVE, pour Common Vulnerabilities and Exposures, est la liste publique de failles de sécurité informatique. Si on ne prend pas en compte cela dans son application, il sera très compliqué d'assurer la sécurité dans celle-ci puisque les dépendances peuvent comprendre des failles.

Au stade actuel, notre application n'est pas entièrement sécurisée mais penser dès le départ à ce type de problème permet de les anticiper et mieux les gérer.

L'utilisation de dependabot est donc très utile en terme de sécurité, ou tout simplement de gestion des versions des dépendances.

1.3. Produit final



	ID	Nom Artiste	Prenom Artiste	Role Artiste	
<input type="checkbox"/>	1ed409...	Albert	Bayet	Guitariste	...
<input type="checkbox"/>	1ed409...	Moulin	Boubakar	Guitariste	...
<input type="checkbox"/>	1ed409...	Moreau	Jérémie	Guitariste	...
<input type="checkbox"/>	1ed409...	Hubert	Emmanuel	Batteur	...
<input type="checkbox"/>	1ed409...	Moulin	Imad	Batteur	...
<input type="checkbox"/>	1ed409...	Moreau	Ouail	Chanteur	...
<input type="checkbox"/>	1ed409...	Louis	Bilal	Guitariste	...
<input type="checkbox"/>	1ed409...	Simon	Bilal	Guitariste	...
<input type="checkbox"/>	1ed409...	Thomas	Marc	Guitariste	...
<input type="checkbox"/>	1ed409...	Hubert	Jérémie	Chanteur	...
<input type="checkbox"/>	1ed409...	Martin	Patrick	Chanteur	...
<input type="checkbox"/>	1ed409...	Louis	Ouail	Batteur	...
<input type="checkbox"/>	1ed409...	Carpentier	Andrew	Guitariste	...
<input type="checkbox"/>	1ed409...	Carpentier	Adrien	Guitariste	...
<input type="checkbox"/>	1ed409...	Louis	Imad	Chanteur	...
<input type="checkbox"/>	1ed409...	Carpentier	Emmanuel	Batteur	...
<input type="checkbox"/>	1ed409...	Sanchez	Boubakar	Chanteur	...

Figure 6 : Application finale

L'application finale, permet à travers une interface simple d'utilisation à l'utilisateur d'ajouter et de gérer les données de sa base de données. Cela sur toutes les entités de l'application. Chaque donnée ajoutée dans l'application est unique, à travers un UUID*, qui garantit qu'il n'y aura pas de doublon en termes d'identification d'une donnée.

Actuellement, l'application ne possède pas de sécurité ou d'identification, elle est accessible à tout le monde. Ce choix a été fait afin de faciliter le test de l'application dans un premier temps à notre clients, mais aussi à nos développeurs.

L'authentification peut être très facilement ajoutée, mais elle n'a pas été demandée par le client. L'ajout de cette fonctionnalité nécessitera une charge en plus pour le client.

```

1  #
2  # Fixtures for Artiste
3  #
4  App\Entity\Artiste:
5    artiste_1:
6      nomArtiste: Albert
7      prenomArtiste: Bayet
8      roleArtiste: Guitariste
9
10   artiste_{2..99}:
11     nomArtiste: <randomElement(['Martin', 'Bernard', 'Thomas', 'Moreau', 'Lo
12     prenomArtiste: <randomElement(['Paul', 'Pierre', 'Jean', 'Karim', 'Andrew',
13     roleArtiste: <randomElement(['Chanteur', 'Batteur', 'Guitariste'])>
14

```

Figure 7 : Fixtures de l'application pour la classe Artiste

Afin de permettre de tester les données facilement et rapidement, nous avons ajouté des données à travers des fixtures*. Ce sont des données qui sont ajoutés à la base de données à travers un fichier yaml*.

Ces fixtures de test permettent de jouer avec les données de l'application très simplement et très rapidement, en plus de cela, on peut ajouter un certain nombre de fixtures. Cela peut aussi permettre de tenter de faire dysfonctionner la base de données afin de voir ses limites en termes de charge.

Ici par exemple, on ajoute 100 artistes à l'intérieur de la table Artiste, avec des noms, prénoms et rôles, choisis parmi une liste rentrée en paramètre.

2. GESTION DE PROJET

2.1. Présentation de l'équipe

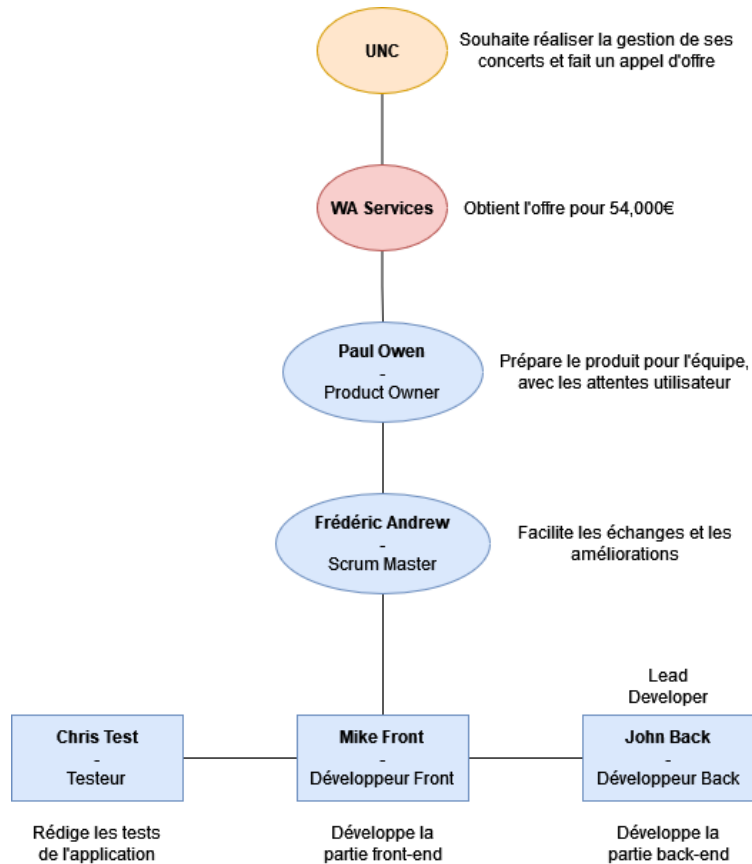


Figure 8 : Organigramme de l'équipe et acteurs

Le diagramme regroupe les différents acteurs, et l'équipe chargée du développement de l'application. On retrouve tout d'abord UNC, qui de par son appel d'offre souhaitait réaliser une application de gestion pour ses concerts. Par la suite WA Services, qui a obtenu l'appel d'offre pour la somme de 54,000€.

Par la suite, on entre dans l'équipe en charge, avec Paul Owen, le product owner. Il était chargé de préparer le produit pour l'équipe à travers des users stories, en lien avec les attentes de l'utilisateur, mais aussi les maquettes de l'application.

Frédéric Andrew était quant à lui le scrum master, il facilitait les échanges entre l'équipe de développement et le product owner. Il apportait aussi des suggestions quant à l'amélioration du processus.

Enfin, on regroupe l'équipe de développement, avec Chris Test, qui était chargé de rédiger les tests de l'application. Il s'est aussi occupé de la mise en place sur les plateformes telles que Git, ou encore la mise en place de Docker dans le projet. C'était aussi le DevOps* durant le projet. Mike Front était quant à lui chargé de développer les parties front-end de l'application, soit l'interface actuelle. Pour finir, John Back, le développeur backend et le lead developer* s'est chargé du développement de l'application sur Symfony.

2.2. Méthode agile et planification

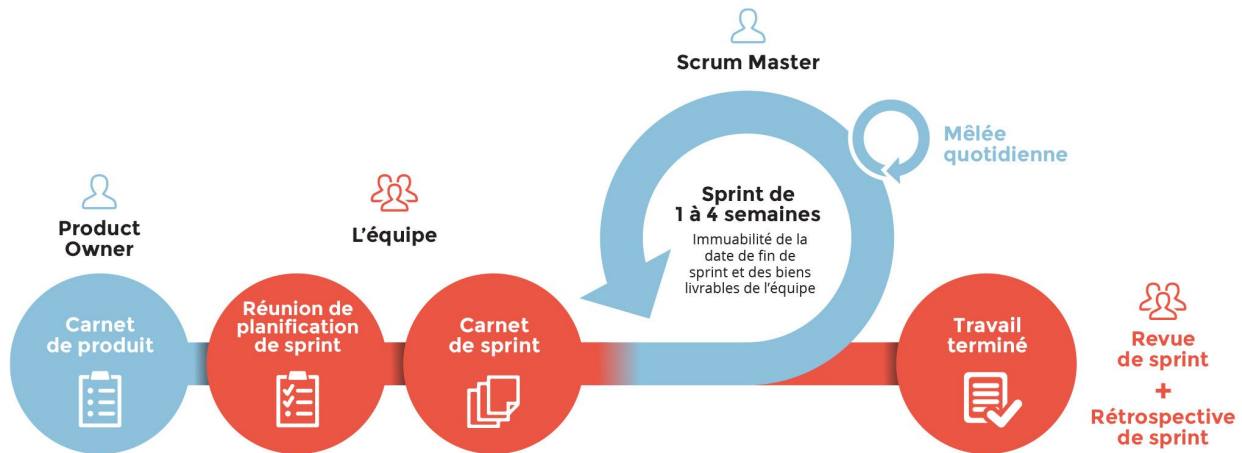


Figure 9 : Méthodologie agile

Pour la réalisation de l'application, l'équipe a suivi une méthodologie agile, elle s'est traduite initialement par la rédaction du carnet de produit par le product owner. Ce carnet contient principalement les users stories et les maquettes de l'application qui seront utiles au développement de l'application pour les développeurs.

A la suite de cela, des réunions de planifications de sprints ont eu lieu et un carnet de sprint par le biais de git a été transmis à l'équipe de développement. A chaque fin de sprint, des revues de sprint ont eu lieu afin de faire un point sur les user stories qui n'ont pas pu être développés et les rajouter au sprint suivant.

Tous les échanges entre le product owner et l'équipe se faisaient par le biais du scrum master, ou alors lors des revues de sprint. Le scrum master a beaucoup aidé durant le développement, puisqu'il a permis de faciliter les échanges avec le product owner, surtout lors d'incompréhension de certaines user stories.

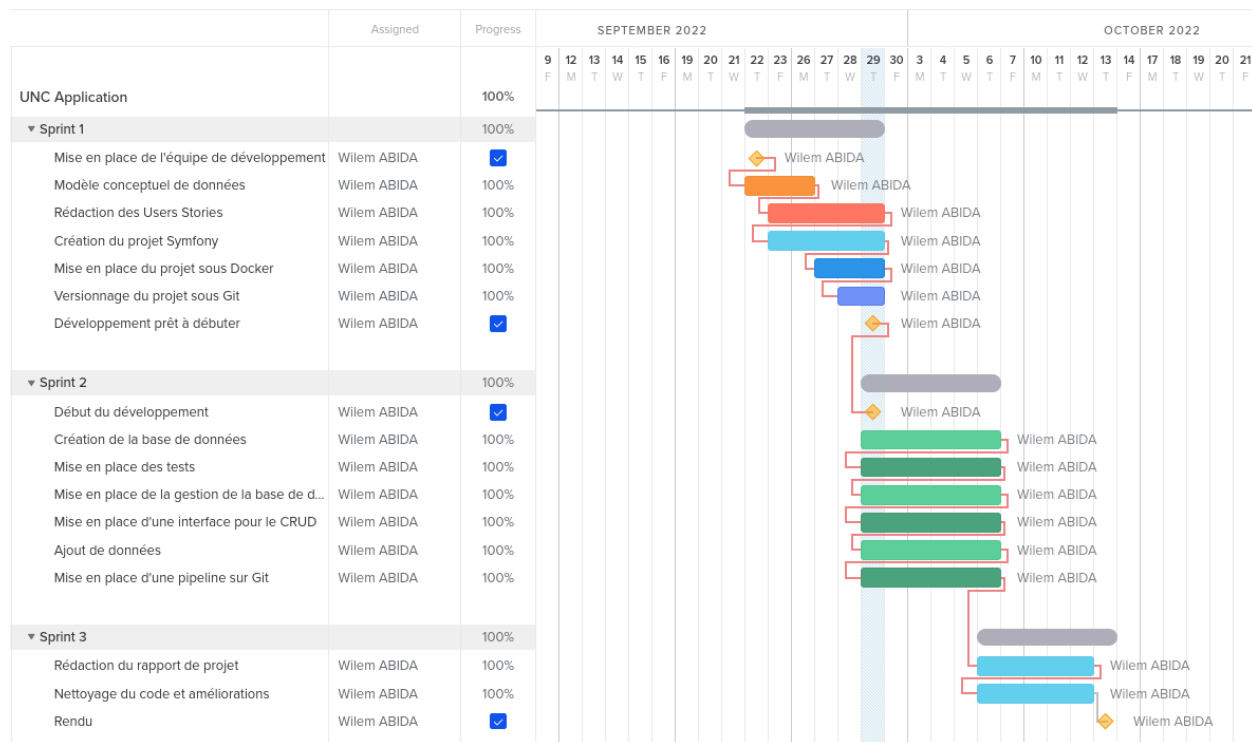


Figure 10 : Diagramme de Gantt du projet

Le diagramme de Gantt regroupe les différentes étapes du projet, notamment les plus importantes qui sont indispensables pour l'avancée de celui-ci. Il est divisé en trois parties, qui sont en réalité des sprints.

Chacun des sprints regroupe un certain nombre de tâches, le premier sprint était un sprint de mise en place, de conception pour faciliter la réalisation. C'est notamment durant celui-ci que deux activités se sont faites en simultanée. L'activité du côté du product owner, qui s'est occupé des Users Stories et des maquettes de l'application. Tandis que les développeurs se sont occupés de mettre en place le projet et sa configuration, afin de débiter le développement de l'application directement après que les maquettes et les users stories soient fournies.

Le second sprint était un sprint plutôt chargé pour l'équipe de développement, qui devait y réaliser toute l'application. Mais aussi rédiger les tests de l'application et

mettre en place des solutions qui permettraient de rendre le code facilement maintenable et de savoir quand celle-ci dysfonctionne après des changements.

Enfin, le dernier sprint a été un sprint durant lequel le rapport concernant l'application et sa réalisation a été rédigé, mais aussi le nettoyage du code, des améliorations et enfin le rendu du produit.

2.3. Étapes de réalisation

Au niveau des étapes de réalisation, le projet s'est principalement divisé en trois étapes. La mise en place des users stories, la planification et la mise en place d'une équipe a été la première chose à réaliser. Par cela s'est suivi la mise en place des postes et des différentes solutions utilisées pour les développeurs. Puis enfin, le développement et les tests de l'application.

Chacune des étapes de réalisation s'est faite durant les sprints, pour qui les durées ont été respectées dans un contexte agile. Sans structuration, le plan initial était avant tout la réalisation des diagrammes pour la base de données, puis le développement et enfin le rendu de l'application.

La structuration à travers plusieurs sprints a permis de mieux diviser les tâches et d'être plus organisé dans la réalisation.

CONCLUSION

La réalisation de cet appel d'offre a été une première pour WA Services, notre société est présente sur le marché du développement logiciel depuis seulement un an. L'obtention d'un appel d'offre de cet ampleur nous a permis d'augmenter la visibilité de notre groupe, mais aussi d'obtenir un gain d'expérience non négligeable.

Nos équipes ont pu travailler pour la première fois sur un vrai projet, qui se devait d'être organisé au vu du délai très court que nous avons donné au client. Les technologies que nous avons utilisées étaient pour la plupart inconnues de notre équipe. Il a donc fallu une période d'adaptation, durant laquelle des craintes quant aux délais de livraison se sont manifestées.

Néanmoins, le retard a très vite été rattrapé et a permis à l'équipe de faire manifester son savoir technique et son sens de l'organisation. La solution que nous avons obtenue aujourd'hui est celle que nous voulions dès le départ, tout comme le client. Malgré cela, des améliorations peuvent être faites, mais cela devra être facturé au client, comme toute nouvelle fonctionnalité.

En obtenant l'appel d'offre, notre Directeur Général, ne savait pas réellement quelles étaient les méthodes à utiliser, en termes de gestion d'équipe, de planning et de développement. En effet, notre société est initialement une petite agence web, qui ne fonctionne pas avec la méthode agile, ou encore avec une organisation sur les plateformes de versioning telles que GIT.

Il a donc fallu très vite s'organiser et trouver une solution, d'où le recrutement d'un product owner et d'un scrum master. Ils se sont rapidement intégrés à l'équipe et ont commencé la mise en place dès leur premier jour. Ils ont pu apporter leurs dernières expériences et nous permettre de rapidement débiter les développements.

Cela a donc été un plaisir de réaliser cette application pour l'Union Nationale des Concerts. En plus de fournir une application élégante et fonctionnelle, nous avons fourni à cet organisme un suivi et une maintenance pour une période de deux ans. Cela permettra de voir l'évolution de notre application, en plus de pouvoir en prendre soin durant les deux prochaines années.

TABLE DES ILLUSTRATIONS

Figure 1 : Configuration docker pour la base de données.....	5
Figure 2 : Dépôt Github de l'application.....	6
Figure 3 : CI/CD de l'application sur GitHub.....	7
Figure 4 : Bot qui notifie l'activité sur GitHub sur un serveur Discord.....	8
Figure 5 : Dependabot configuration.....	9
Figure 6 : Application finale.....	10
Figure 7 : Fixtures de l'application pour la classe Artiste.....	11
Figure 8 : Organigramme de l'équipe et acteurs.....	12
Figure 9 : Méthodologie agile.....	14
Figure 10 : Diagramme de Gantt du projet.....	15

GLOSSAIRE

Appel d'offre : Procédure par laquelle un client, qui peut être une entreprise, un particulier ou un organisme fait la demande auprès de plusieurs groupes, de lui faire une proposition quant à la réalisation d'un service.

Backend : développeur qui travaille sur des applications desktop, web ou mobiles, sur la partie qui n'est pas accessible aux utilisateurs finaux ou aux clients, par opposition à une application de front office (Front-End).

Bundle : Ensemble de fonctionnalités regroupées afin d'offrir une mise en place plus rapide d'application aux développeurs.

CI/CD : Méthode de travail qui vérifie constamment, que les modifications sur le code ne provoquent pas un dysfonctionnement au sein de l'application.

CRUD : Create, Read, Update, Delete. Ce sont les quatre opérations de base pour la persistance dans une base de données.

DevOps : Ensemble de pratique pour développer et livrer des logiciels ou changer d'infrastructure.

Diagramme de Gantt : Outil permettant la visualisation de l'état d'avancement des tâches dans un projet.

Docker : Solution informatique qui exécute la virtualisation au niveau du système d'exploitation, également connue sous le nom de conteneurisation. Docker permet d'embarquer une application dans un container virtuel qui pourra s'exécuter sur n'importe quelle machine.

Docker-compose : Outil qui permet d'exécuter et de définir des applications Docker avec plusieurs conteneurs, en une seule commande.

Dockerfile : Technologie permettant de faciliter le déploiement d'application ainsi que leur gestion.

EasyAdmin : Bundle symfony qui permet l'administration de ses entités, avec un système de CRUD.

Fixtures : Fausses données.

Github : Plateforme de gestion de versions et de collaborations pour les développeurs.

GitFlow : Stratégie pour gérer les branches de son application.

Symfony : Framework qui représente un ensemble de composants(ou librairies) PHP autonomes qui peuvent être utilisés dans des projets web privés ou open source.

Lead Dev : Développeur expérimenté qui dirige une équipe.

MySQL : My Structured Query Language est un serveur de base de données

Merge request : Demande de fusion d'une branche contenant une nouvelle fonctionnalité dans la branche principale d'un dépôt git.

Méthodologie Agile : Méthode qui met en avant la collaboration entre des équipes auto-organisées et pluridisciplinaires et leurs clients.

PHP : Langage informatique ou langage de script, utilisé principalement pour la conception de sites web dynamiques.

Product owner : Responsable de la définition et de la conception d'un produit.

Scrum master : Acteur chargé d'assurer que les processus scrum sont respectés mais aussi de faciliter les échanges entre le product owner et l'équipe de développement.

Sprints : Période durant laquelle un certain nombre de tâches doivent être effectuées, qui s'ensuit par une révision du travail effectué.

Symfony : Framework qui représente un ensemble de composants(ou librairies) PHP autonomes qui peuvent être utilisés dans des projets web privés ou open source.

Tests unitaires : Processus de vérification d'une unique unité de logiciel.

UI : User Interface, c'est l'environnement graphique de l'utilisateur.

UUID : Universally unique identifier, permet d'identifier de façon unique un objet.

UNC : Union nationale des concerts.

User stories : Récit utilisateur, c'est une description simple d'un besoin par un utilisateur.

WA Services : Wilem Abida Services, entreprise utilisée pour la simulation d'appel d'offres. Il s'agit de mon activité free-lance.

YAML : Yet Another Markup Language, c'est un format de représentation de données.

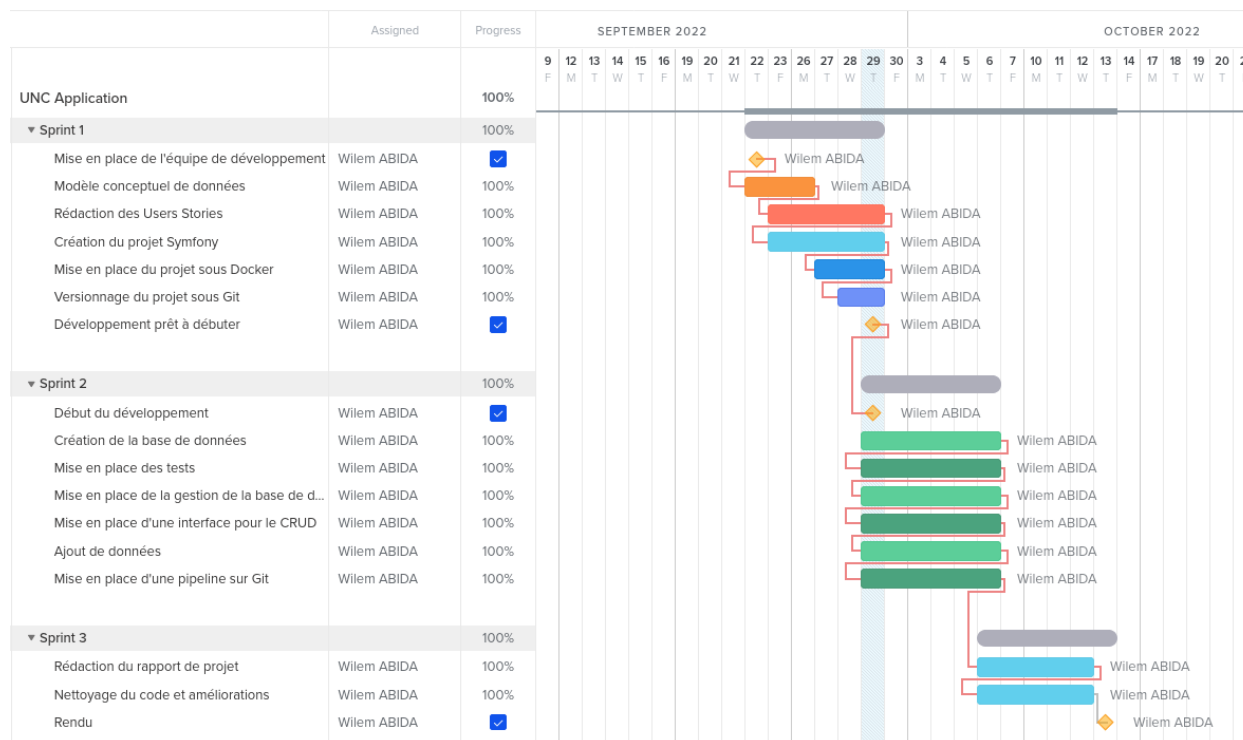
REFERENCES

- Réalisation des diagrammes : <https://app.diagrams.net/>
- Documentation Docker : <https://www.docker.com/>
- Documentation PHP : <https://www.php.net/>
- Documentation Symfony : <https://symfony.com/>
- Tutoriel W3Schools : <https://courses.w3schools.com/>
- Tutoriel OpenClassRooms : <https://openclassrooms.com/fr/>
- Github : <https://github.com/>
- Documentation du bundle EasyAdmin :
<https://symfony.com/bundles/EasyAdminBundle/current/index.html>
- Dépôt du projet : <https://github.com/b2p-wilem/unc-project>
- Symfony fixtures : <https://symfonycasts.com/screencast/alice-fixtures/fixtures>
- Dépôt dependabot : <https://github.com/dependabot>
- Tests pour Symfony :
<https://symfony.com/doc/current/the-fast-track/fr/17-tests.html>
- Plateforme pour les diagrammes de gantt : <https://www.teamgantt.com/>
- Makefile : <https://opensource.com/article/18/8/what-how-makefile>

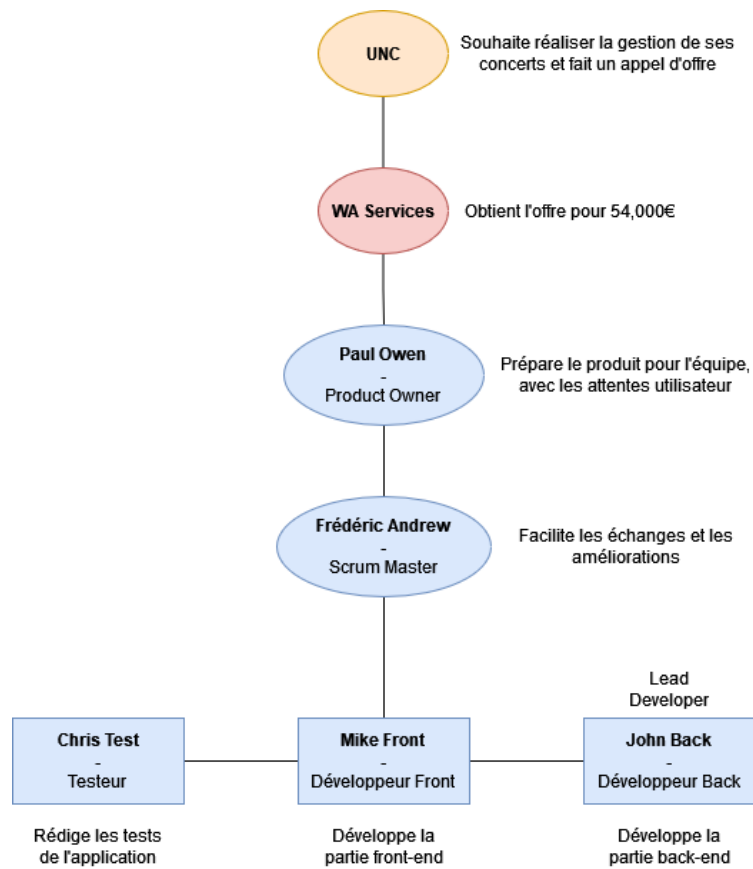
ANNEXES

TABLE DES ANNEXES


Annexe 1 : Diagramme de Gantt du projet.....	25
Annexe 2 : Organigramme du projet.....	26
Annexe 3 : Dashboard de l'application.....	27
Annexe 4 : Diagramme de la BDD sur Symfony.....	28
Annexe 5 : MCD de l'application.....	29
Annexe 6 : Modèle relationnel de l'application.....	30
Annexe 7 : Script SQL pour ajouter la base de données.....	31



Annexe 1 : Diagramme de Gantt du projet



Annexe 2 : Organigramme du projet



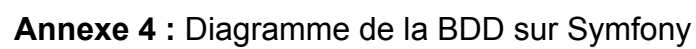
Concert
Groupe
Billet
Client
Artiste
Technicien

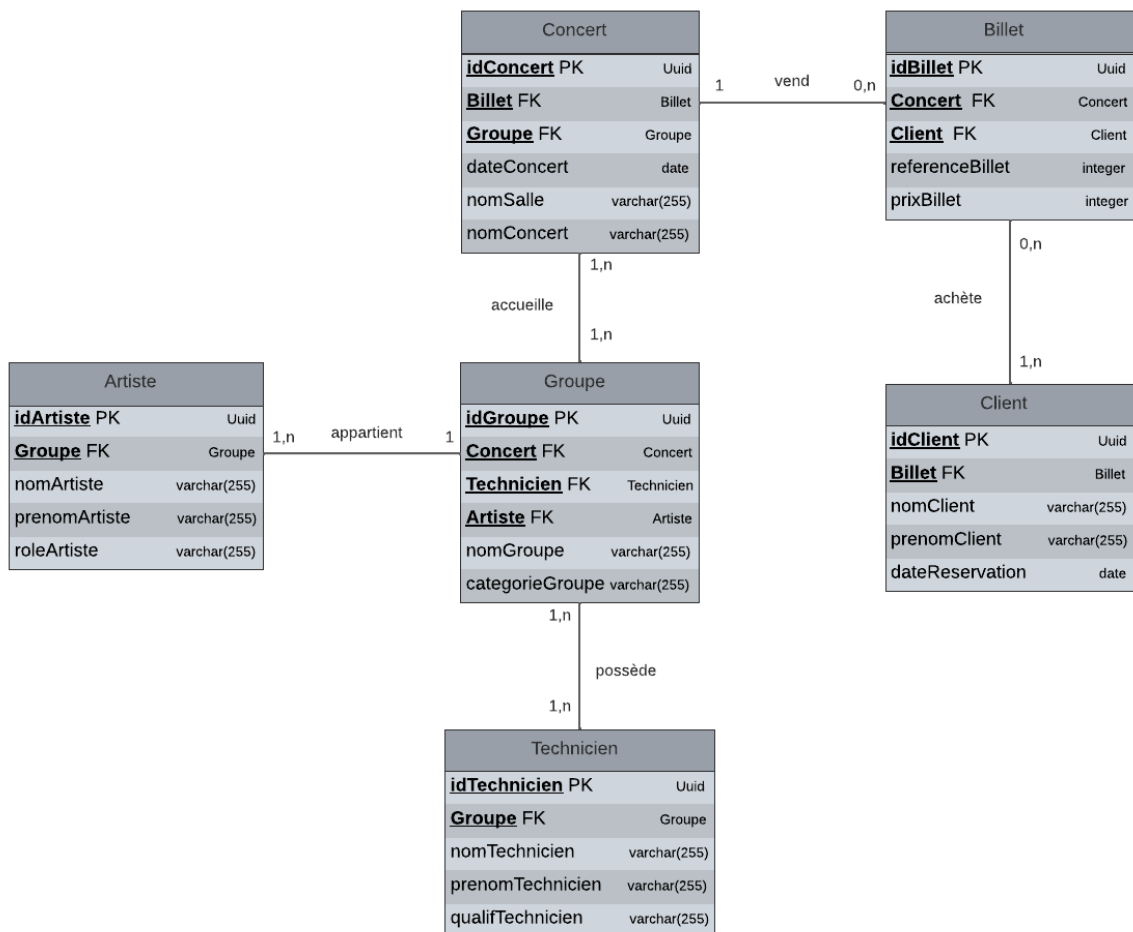
Search

Add Artiste

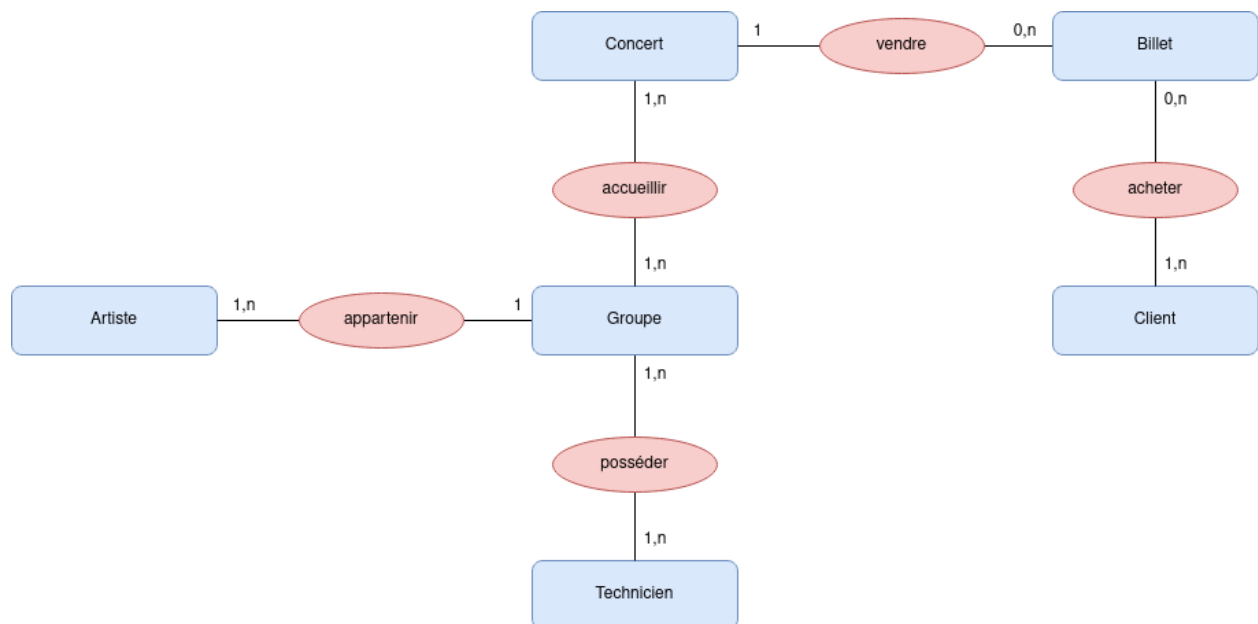
<input type="checkbox"/>	ID	Nom Artiste	Prenom Artiste	Role Artiste	
<input type="checkbox"/>	1ed409...	Albert	Bayet	Guitariste	...
<input type="checkbox"/>	1ed409...	Moulin	Boubakar	Guitariste	...
<input type="checkbox"/>	1ed409...	Moreau	Jérémie	Guitariste	...
<input type="checkbox"/>	1ed409...	Hubert	Emmanuel	Batteur	...
<input type="checkbox"/>	1ed409...	Moulin	Imad	Batteur	...
<input type="checkbox"/>	1ed409...	Moreau	Ouail	Chanteur	...
<input type="checkbox"/>	1ed409...	Louis	Bilal	Guitariste	...
<input type="checkbox"/>	1ed409...	Simon	Bilal	Guitariste	...
<input type="checkbox"/>	1ed409...	Thomas	Marc	Guitariste	...
<input type="checkbox"/>	1ed409...	Hubert	Jérémie	Chanteur	...
<input type="checkbox"/>	1ed409...	Martin	Patrick	Chanteur	...
<input type="checkbox"/>	1ed409...	Louis	Ouail	Batteur	...
<input type="checkbox"/>	1ed409...	Carpentier	Andrew	Guitariste	...
<input type="checkbox"/>	1ed409...	Carpentier	Adrien	Guitariste	...
<input type="checkbox"/>	1ed409...	Louis	Imad	Chanteur	...
<input type="checkbox"/>	1ed409...	Carpentier	Emmanuel	Batteur	...
<input type="checkbox"/>	1ed409...	Sanchez	Boubakar	Chanteur	...

Annexe 3 : Dashboard de l'application





Annexe 5 : MCD de l'application



Annexe 6 : Modèle relationnel de l'application

```

CREATE TABLE technicien (id BINARY(16) NOT NULL COMMENT '(DC2Type:uuid)',
nom_technicien VARCHAR(255) DEFAULT NULL, prenom_technicien
VARCHAR(255) DEFAULT NULL, qualif_technicien VARCHAR(255) DEFAULT NULL,
PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8mb4 COLLATE
`utf8mb4_unicode_ci` ENGINE = InnoDB;
CREATE TABLE concert (id BINARY(16) NOT NULL COMMENT '(DC2Type:uuid)',
nom_concert VARCHAR(255) DEFAULT NULL, date_concert DATE DEFAULT NULL,
nom_salle VARCHAR(255) DEFAULT NULL, PRIMARY KEY(id)) DEFAULT
CHARACTER SET utf8mb4 COLLATE `utf8mb4_unicode_ci` ENGINE = InnoDB;
CREATE TABLE concert_groupe (concert_id BINARY(16) NOT NULL COMMENT
'(DC2Type:uuid)', groupe_id BINARY(16) NOT NULL COMMENT '(DC2Type:uuid)',
INDEX IDX_A1B69CA083C97B2E (concert_id), INDEX IDX_A1B69CA07A45358C
(groupe_id), PRIMARY KEY(concert_id, groupe_id)) DEFAULT CHARACTER SET
utf8mb4 COLLATE `utf8mb4_unicode_ci` ENGINE = InnoDB;
CREATE TABLE client (id BINARY(16) NOT NULL COMMENT '(DC2Type:uuid)',
nom_client VARCHAR(255) DEFAULT NULL, prenom_client VARCHAR(255)
DEFAULT NULL, date_reservation DATE DEFAULT NULL, PRIMARY KEY(id))
DEFAULT CHARACTER SET utf8mb4 COLLATE `utf8mb4_unicode_ci` ENGINE =
InnoDB;
CREATE TABLE billet (id BINARY(16) NOT NULL COMMENT '(DC2Type:uuid)',
concert_id BINARY(16) DEFAULT NULL COMMENT '(DC2Type:uuid)', prix_billet INT
DEFAULT NULL, reference_billet VARCHAR(255) DEFAULT NULL, INDEX
IDX_1F034AF683C97B2E (concert_id), PRIMARY KEY(id)) DEFAULT CHARACTER
SET utf8mb4 COLLATE `utf8mb4_unicode_ci` ENGINE = InnoDB;
CREATE TABLE billet_client (billet_id BINARY(16) NOT NULL COMMENT
'(DC2Type:uuid)', client_id BINARY(16) NOT NULL COMMENT '(DC2Type:uuid)',
INDEX IDX_D8C0F82644973C78 (billet_id), INDEX IDX_D8C0F82619EB6921

```

```
(client_id), PRIMARY KEY(billet_id, client_id)) DEFAULT CHARACTER SET utf8mb4
COLLATE `utf8mb4_unicode_ci` ENGINE = InnoDB;

CREATE TABLE groupe (id BINARY(16) NOT NULL COMMENT '(DC2Type:uuid)',
nom_groupe VARCHAR(255) DEFAULT NULL, categorie_groupe VARCHAR(255)
DEFAULT NULL, PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8mb4 COLLATE
`utf8mb4_unicode_ci` ENGINE = InnoDB;

CREATE TABLE groupe_technicien (groupe_id BINARY(16) NOT NULL COMMENT
'(DC2Type:uuid)', technicien_id BINARY(16) NOT NULL COMMENT '(DC2Type:uuid)',
INDEX IDX_2EC51D907A45358C (groupe_id), INDEX IDX_2EC51D9013457256
(technicien_id), PRIMARY KEY(groupe_id, technicien_id)) DEFAULT CHARACTER
SET utf8mb4 COLLATE `utf8mb4_unicode_ci` ENGINE = InnoDB;

CREATE TABLE artiste (id BINARY(16) NOT NULL COMMENT '(DC2Type:uuid)',
groupe_id BINARY(16) DEFAULT NULL COMMENT '(DC2Type:uuid)', nom_artiste
VARCHAR(255) DEFAULT NULL, prenom_artiste VARCHAR(255) DEFAULT NULL,
role_artiste VARCHAR(255) DEFAULT NULL, INDEX IDX_9C07354F7A45358C
(groupe_id), PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8mb4 COLLATE
`utf8mb4_unicode_ci` ENGINE = InnoDB;

ALTER TABLE concert_groupe ADD CONSTRAINT FK_A1B69CA083C97B2E
FOREIGN KEY (concert_id) REFERENCES concert (id) ON DELETE CASCADE;
ALTER TABLE concert_groupe ADD CONSTRAINT FK_A1B69CA07A45358C
FOREIGN KEY (groupe_id) REFERENCES groupe (id) ON DELETE CASCADE;
ALTER TABLE billet ADD CONSTRAINT FK_1F034AF683C97B2E FOREIGN KEY
(concert_id) REFERENCES concert (id);
ALTER TABLE billet_client ADD CONSTRAINT FK_D8C0F82644973C78 FOREIGN
KEY (billet_id) REFERENCES billet (id) ON DELETE CASCADE;
ALTER TABLE billet_client ADD CONSTRAINT FK_D8C0F82619EB6921 FOREIGN
KEY (client_id) REFERENCES client (id) ON DELETE CASCADE;
```



```
ALTER TABLE groupe_technicien ADD CONSTRAINT FK_2EC51D907A45358C  
FOREIGN KEY (groupe_id) REFERENCES groupe (id) ON DELETE CASCADE;  
ALTER TABLE groupe_technicien ADD CONSTRAINT FK_2EC51D9013457256  
FOREIGN KEY (technicien_id) REFERENCES technicien (id) ON DELETE CASCADE;  
ALTER TABLE artiste ADD CONSTRAINT FK_9C07354F7A45358C FOREIGN KEY  
(groupe_id) REFERENCES groupe (id);
```

Annexe 7 : Script SQL pour ajouter la base de données