

18

**Pimp My Unicorn**

**Alexandre BACA - Lucien BROUILLAUD - Jérémy EMO - Nicolas KAMPHAUS – Dorian LOISON**

Contenu

[Introduction 3](#_Toc532976483)

[Descriptif du projet 4](#_Toc532976484)

[Développement de l’idée 5](#_Toc532976485)

[Réalisation 6](#_Toc532976486)

[Partie Administrateur : 6](#_Toc532976487)

[Partie Utilisateur : 8](#_Toc532976488)

[Service WCF : 9](#_Toc532976489)

[Conclusion 10](#_Toc532976490)

# Introduction

Lors de notre formation, nous avons dû réaliser un projet C# en équipe. Nous avons voulu proposer un concept innovant et intéressant, et c’est pourquoi nous avons décidé de développer un outil de création de licorne personnalisée, ce qui nous permet de mettre en avant les compétences acquises durant la formation.

# Descriptif du projet

Pour réaliser ce projet, nous avions plusieurs prérequis :

* Développer un client Administrateur pouvant fonctionner en local sans aucune connexion à la base de données.
* Développer un client classique lisant la base de données située sur le serveur hébergeant un service WCF.
* Avoir un WCF permettant la synchronisation des données entre le client Administrateur et la base de données.

# Développement de l’idée

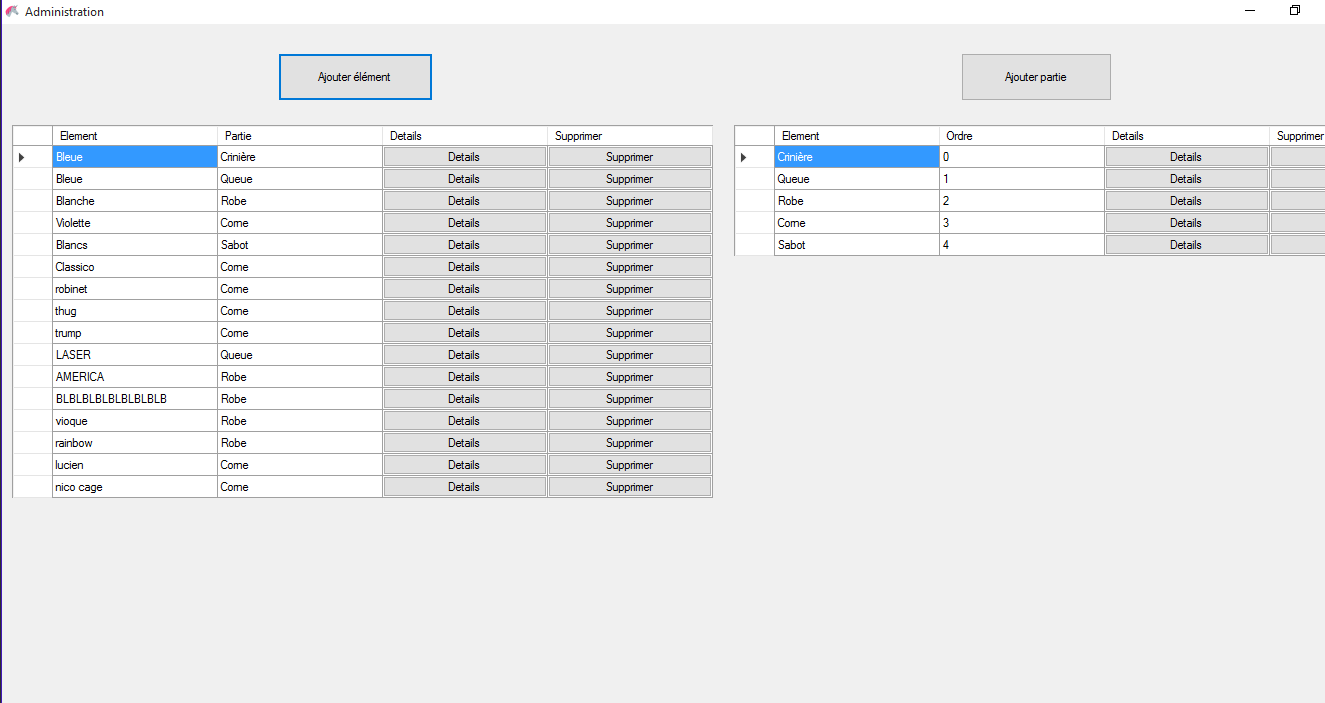
Afin de respecter ces consignes, nous sommes partis sur une application de création de licornes. Le client classique pourra donc créer la licorne de ses rêves en récupérant des parties de celle-ci dans la base de données tandis que le client administrateur pourra en ajouter. L’utilisateur pourra également enregistrer sa licorne au format PNG.

Le fonctionnement final reprends les principes d’une gestion de stock simplifiée où l’on gère une banque d’images.

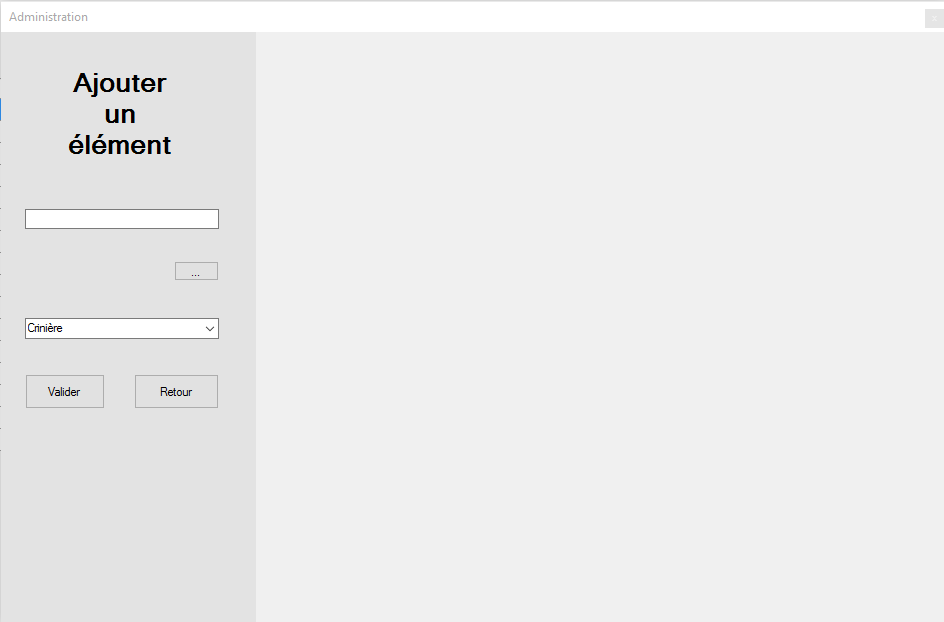
# Réalisation

## Partie Administrateur :

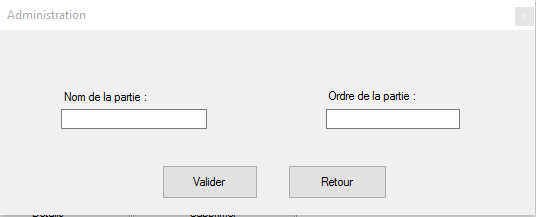
Le client administrateur est composé de 3 windowsForms servant de vues :



Ci-dessus la vue principale permettant de consulter toutes les entrées de la base de données.



Ci-dessus la fenêtre d’ajout d’éléments.



Et ci-dessus la fenêtre d’ajout de parties.

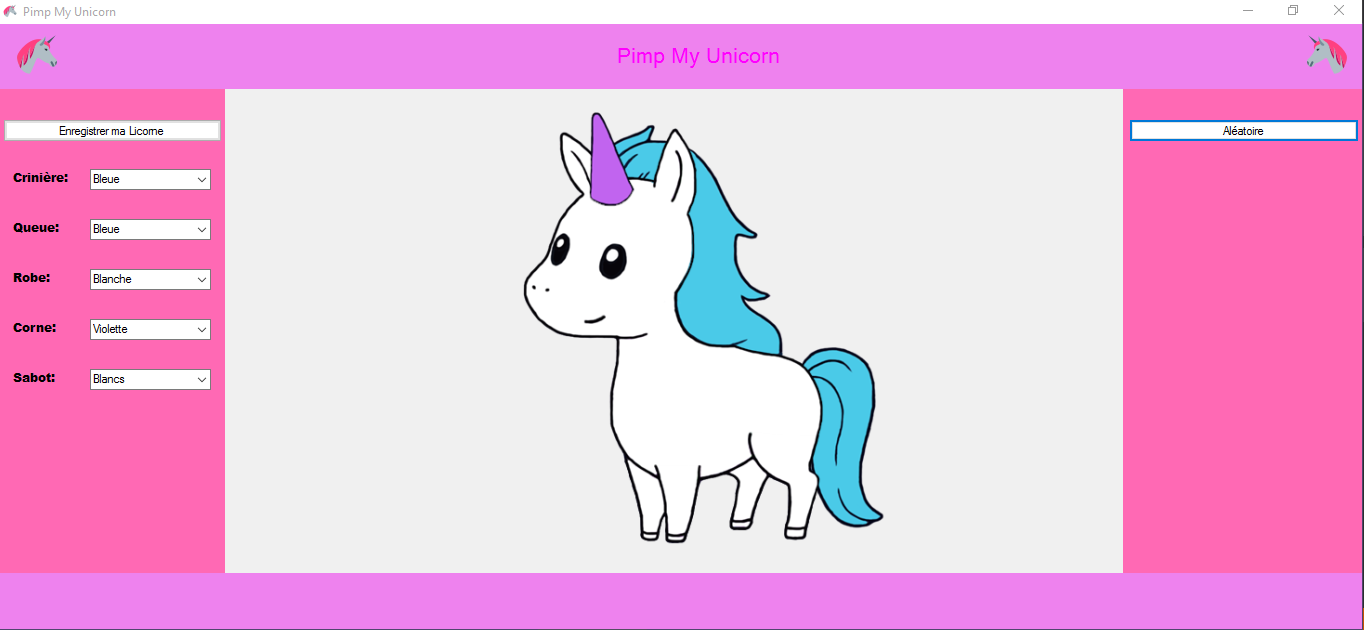
Afin d’avoir un client pouvant être utilisé en local, nous avons utilisé une base SQLite embarquée, sur laquelle nous importons toutes les données déjà existantes dans la base SQL à la première utilisation via le WCF.

Lors de l’ajout de nouvelles entrées dans la base SQLite, nous écrivons la requête dans un fichier texte récupéré par le WCF via un FileWatcher et qui l’exécute afin de la répliquer sur la base SQL.

Le FileWatcher est exécuté toutes les heures par défaut, et c’est configurable avec un scheduler.

## Partie Utilisateur :

Le client utilisateur est composé d’une seule vue, permettant la création de sa licorne :



Une fonctionnalité permet de générer sa licorne à partir d’éléments aléatoires. L’utilisateur peut également enregistrer sa licorne au format PNG.

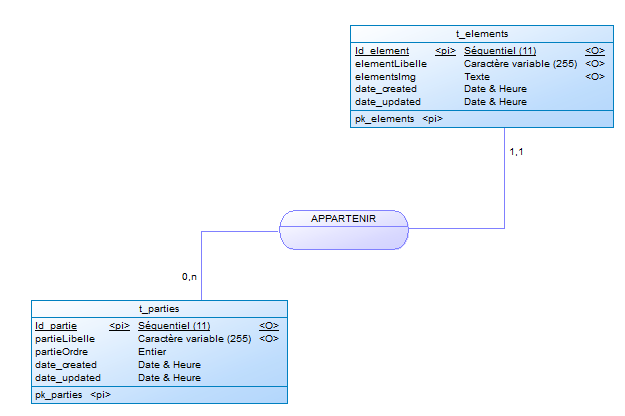
Le client est connecté directement à la base SQL, et les images sont récupérées directement depuis la base au format base64.

## Service WCF :

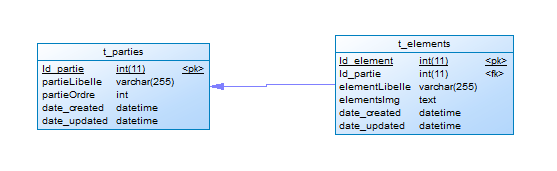
Le service WCF est composé de 3 fonctions, deux permettant de récupérer toutes les entrées de la base de données SQL, les parties et les éléments. Celles-ci sont utilisées lors de la première utilisation du client administrateur.

La troisième fonction permet de récupérer et d’exécuter les requêtes écrites dans le fichier texte du client administrateur. Cette fonction permet de créer, éditer ou supprimer des éléments ou des parties.

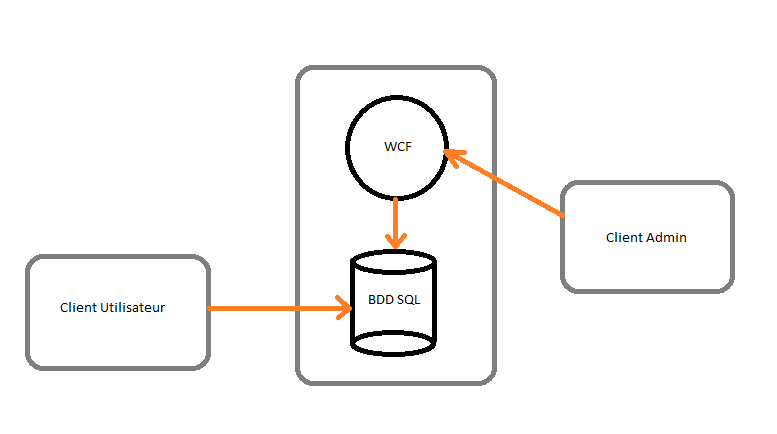
## Modèle Conceptuel de la base de données :



## Modèle Physique de la base de données :



## Fonctionnement final :



# Conclusion

Ce projet nous a permis de mettre en application ce que l’on a appris.