

# Génie Logiciel

---

*Gestionnaire de mot de passe*



# Sommaire

---

<b>Sommaire</b>	<b>2</b>
<b>Introduction</b>	<b>3</b>
<b>Spécifications générales</b>	<b>4</b>
Analyse fonctionnelle	4
Analyse des besoins pour l'interface avec l'utilisateur	5
Schémas UML	6
Description de l'architecture de l'application	7
Répartition des tâches dans le groupe et planning	8
<b>Spécifications détaillées</b>	<b>10</b>
Formulaires de l'application	10
Justification des choix	15
Modalités de calcul de la force du mot de passe	17
<b>Résultats et tests</b>	<b>18</b>
Exigences métier respectées par le programme	18
Résultats de notre programme	19
Description des protocoles de tests	25
Résultats des tests	27
<b>Bilan et perspectives</b>	<b>28</b>
Points positifs et négatifs	28
Évolutions possibles	28
<b>Annexes</b>	<b>30</b>
Exigences techniques	30
Exigences métiers	30

# Introduction

---

La programmation est une composante principale de l'enseignement à l'ENSC. C'est pourquoi, dans le cadre du module de Génie Logiciel de deuxième année, se déroule un projet. Celui-ci a pour but de consolider nos connaissances récemment acquises en matière de création d'applications WinForm, de gestion d'événements et d'interaction avec une base de données.

D'un point de vue moins technique et plus des bonnes habitudes à prendre, c'est aussi l'occasion pour nous de mettre une fois de plus en application des principes comme "Don't Repeat Yourself", "Keep It Simple Stupid" ou les conventions types camelCase.

Ce projet consistera en la réalisation d'un gestionnaire de mots de passe, permettant de mémoriser et gérer l'ensemble des mots de passe d'un utilisateur donné. Ce gestionnaire devra alors bien évidemment répondre à certaines exigences, tant sur les plans techniques que ergonomiques ou métier. Par exemple, l'utilisation de MySQL pour stocker les données persistantes, la facilité d'utilisation de l'application ou encore la possibilité de supprimer un compte enregistré.

# I. Spécifications générales

---

## A. Analyse fonctionnelle

La première étape du projet a été d'identifier les principales fonctionnalités que devait posséder notre application.

Fonctions	Description
Connexion	Permet à l'utilisateur d'ouvrir sa session (login et mot de passe)
<b>Affichage</b>	
Affichage Contenu	Affiche le contenu de la session (Comptes et points wifi)
Détail compte	Affiche les détails du compte sélectionné
Détail Wifi	Affiche les détails du wifi sélectionné
<b>Mot de passe</b>	
CalculForce	Calcule la force du mot de passe
CalculDerniereModif	Calcule de la date de la dernière modification du mot de passe
Générer Mot de Passe	Générer un mot de passe aléatoire suivant les caractères imposés (Longueur, nombre de chiffres, nombres de symboles)
<b>Compte</b>	
AjouterCompte	Ajoute un nouveau compte à la BDD et un lien avec l'utilisateur connecté
ModifierCompte	Modifie un compte existant dans la base de données
SupprimerCompte	Supprime le lien utilisateur/compte de la BDD et le compte s'il n'est plus utilisé
<b>Wifi</b>	
AjouterWifi	Ajoute un point wifi dans la BDD et un lien avec l'utilisateur connecté
ModifierWifi	Modifie un point wifi existant dans la BDD
SupprimerWifi	Supprime le lien utilisateur/wifi de la BDD et le wifi s'il n'est plus utilisé

## B. Analyse des besoins pour l'interface avec l'utilisateur

Besoins	Description
Connexion	Avoir une interface de connexion pour accéder à son espace de gestion personnalisé et protégé par un mot de passe
<b>Mot de passe</b>	
Force du mot de passe	Afficher la force du mot de passe de manière visuelle
Mot de passe ancien	Afficher de manière visuelle si un compte a un mot de passe ancien (plus de 6 mois)
Mot de passe faible	Afficher de manière visuelle si un compte a un mot de passe faible
Générer Mot De Passe	Avoir une interface pour générer un mot de passe accessible à toutes actions concernant le mot de passe dans l'application et comprenant la saisie de l'utilisateur pour certains critères
<b>Comptes/Wifis</b>	
Sélectionner	Pouvoir sélectionner facilement un compte ou un wifi via l'interface principale
Détails	Voir facilement les détails d'un compte ou wifi sélectionné
Ajouter	Pouvoir ajouter un compte ou un wifi via l'interface principale
Modifier	Pouvoir modifier le compte/wifi sélectionné
Supprimer	Pouvoir supprimer un compte/wifi sélectionné via l'interface principale après un message de confirmation

## C. Schémas UML

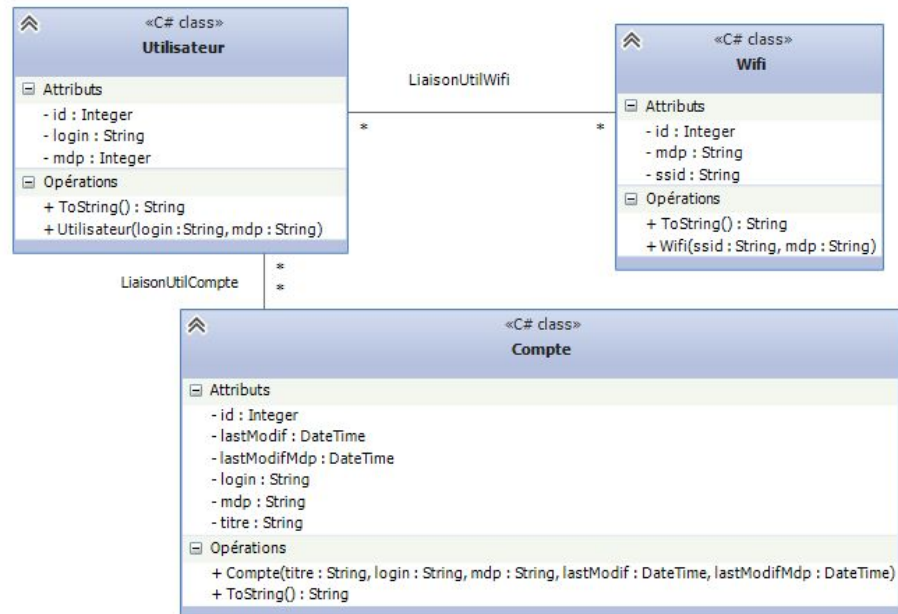


Diagramme des classes du Domain

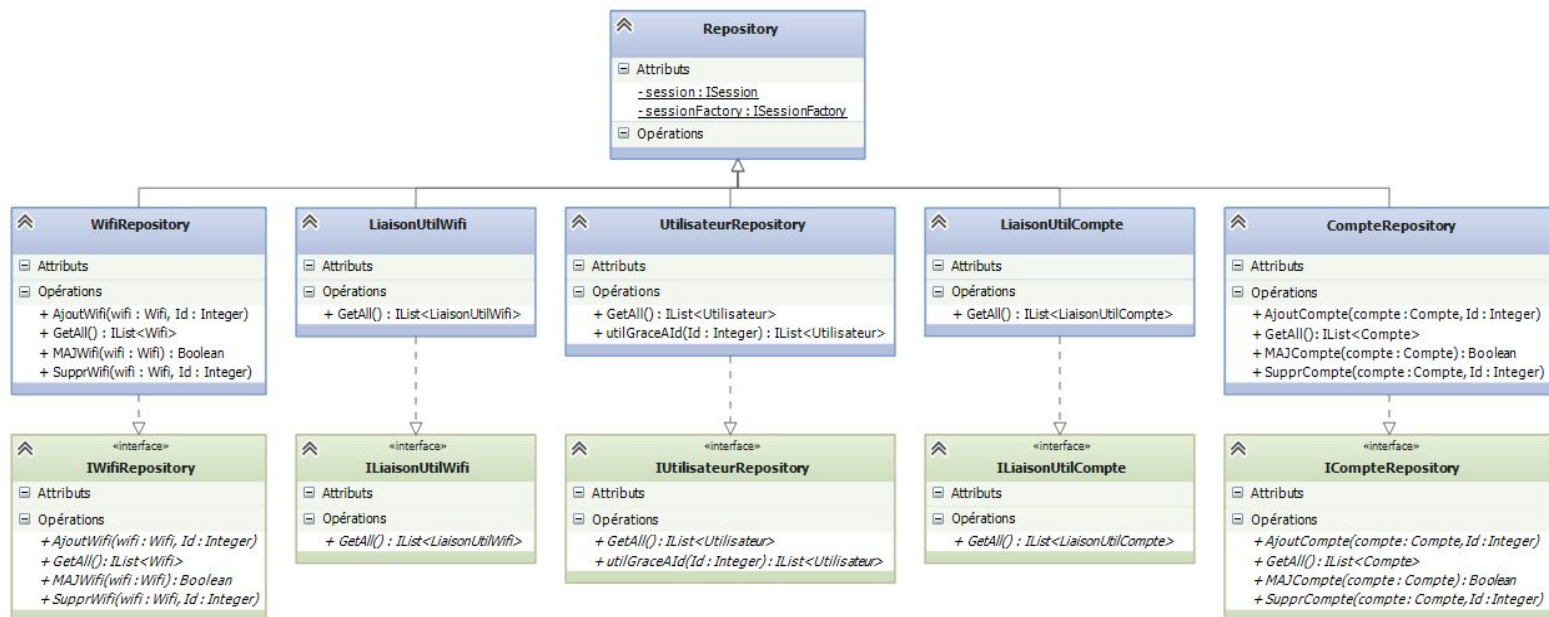


Diagramme des classes du DAL

## D. Description de l'architecture de l'application

Notre gestionnaire de mots de passe se décompose en 4 grandes parties : l'*App*, le *DAL*, le *Domain* et le *Test*. A ces parties s'ajoute un dossier à part nommé *NHibernate* qui rassemble les fichiers qui font fonctionner NHibernate ainsi que les fichiers nécessaires au bon fonctionnement des quatre autres cités précédemment (références).

### **App :**

Abréviation de "Application", l'*App* est l'endroit où ont été créés tous les designs de l'application. C'est également là où on retrouve, pour chaque interface, le code associé régissant les actions provoquées par les différentes commandes de l'utilisateur (clics, saisis au clavier, ...). C'est donc le lieu des interactions avec l'utilisateur et des actions que celles-ci déclenchent.

### **DAL :**

Le *DAL* (abréviation de Data Access Layer) est le coeur des interactions avec la BDD et se divise en 3 sous-dossiers.

On retrouve tout d'abord le dossier *DB* (pour DataBase) où se trouvent les scripts SQL permettant de créer la BDD et de la compléter avec quelques éléments qui nous serviront par la suite entre autre à tester nos requêtes.

Se trouve ensuite le dossier *Interfaces*. Sont rassemblées dans ce dossier toutes les interfaces qui permettent de donner un modèle que devront suivre les classes *Repositories* (classes du dernier sous-dossier). Elles fixent ainsi les méthodes qui devront impérativement être implémentées dans ces classes.

Le dernier dossier, comme dit précédemment, est le dossier *Repositories*, contenant toutes les classes *Repositories*. Une classe *Repository* n'a pas de constructeur et contient exclusivement des méthodes permettant l'exécution de certaines requêtes SQL. Elles auront pour pure vocation de fournir les données voulues en les récupérant dans la BDD.

### **Domain :**

Cette partie de notre solution est la base des différentes entités que l'on traite dans l'application. C'est donc naturellement dans le dossier que l'on retrouve les classes *Utilisateur*, *Compte*, *Wifi*, *LiaisonUtilCompte* et *LiaisonUtilWifi*, toutes munies de leurs constructeurs, leurs propriétés ainsi que leurs méthodes.

On retrouve également un dossier nommé *Mapping* qui décrit les correspondances entre les classes et la BDD. Par exemple, la table *Wifi* possède une colonne "SSID" et une colonne "mdp". On a donc défini la classe *Wifi* comme ayant un "SSID" et un "mdp" comme attributs puis décrit dans un fichier *Wifi.hbm.xml* dans le *Mapping* que ces colonnes correspondaient à ces attributs.

**Test :**

L'intérêt de cette partie de votre solution réside dans le seul fichier qui y soit présent, c'est-à-dire un fichier *Program.cs* dans lequel sont testées toutes les requêtes que l'on va utiliser dans votre application. Ces requêtes étant déclarées dans les classes Repositories du *Domain*, il a suffit d'instancier les objets dont nous avons besoins pour nos requêtes, lister ces dernières puis les tester en prenant en compte tous les cas possibles.

Finalement, l'*App* nous permet de recevoir certaines instructions de l'utilisateur, notamment sous forme de clics. Ces clics provoquent l'appel de méthodes des classes Repositories dans le *DAL*. Ces méthodes utilisent alors des requêtes SQL et renvoie les données voulues. Grâce à *NHibernate* et aux fichiers de Mapping dans le *Domain*, ces données trouvent une correspondance en C# (classes, listes de classes, etc.) sur laquelle on travaille par la suite pour transmettre par exemple des informations d'une classe à l'autre ou pour faire nos tests dans le programme dédié du dossier *Test*.

## E. Répartition des tâches dans le groupe et planning

Après une première période de réflexion commune sur la structure de notre base de données et des différentes fonctionnalités que devait posséder notre application, il nous a semblé évident de nous diviser les tâches pour travailler de façon parallèle et donc plus efficace.

Ainsi, l'un de nous s'est chargé de mettre en place la BDD, Nhibernate, les interfaces, les mappings, les classes implantées puis de tout faire communiquer entre eux. A cela s'est ajouté la réalisation du programme de test, faisant appel à toutes les requêtes qui implémentent certaines classes.

Pendant ce temps, l'autre a créé les interfaces de notre application avec Winform, avant de s'occuper entre autre de la gestion des évènements avec l'utilisateur, de la connexion et des affichages.

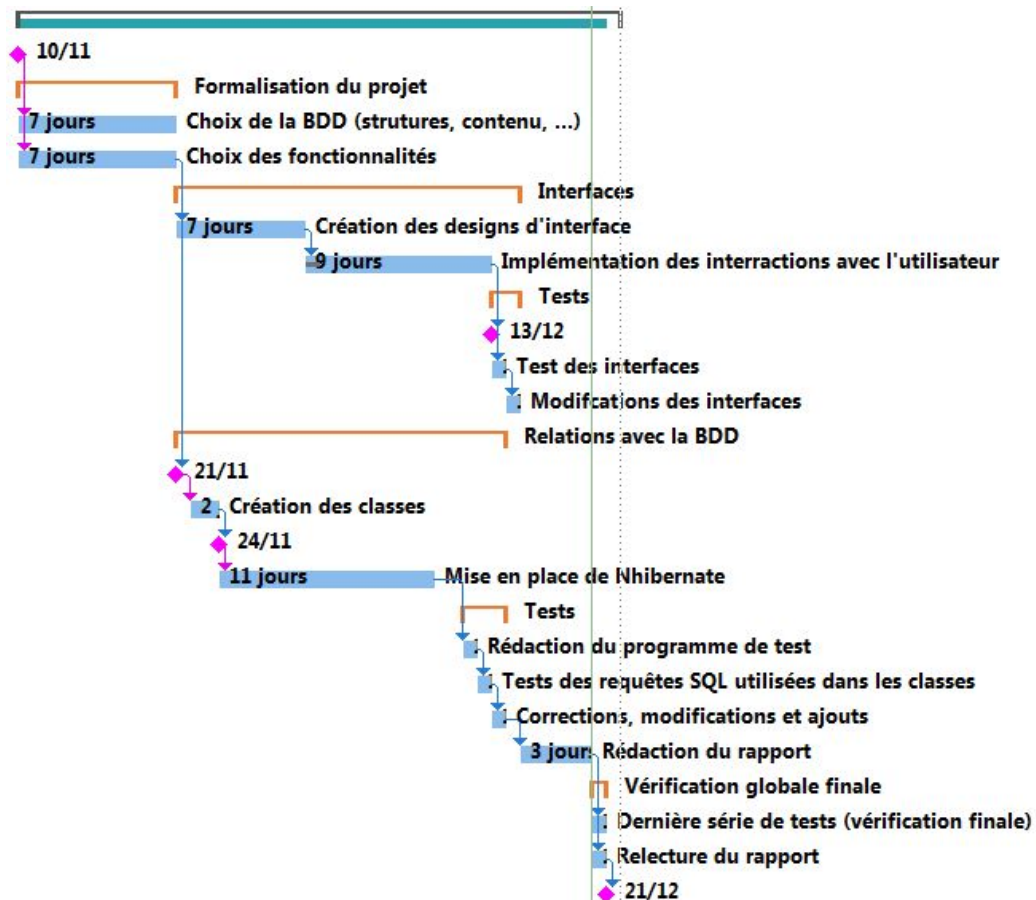
La communication au sein du projet a été la clé cruciale pour réaliser un projet cohérent et fournir assez vite de ce dont l'autre avait besoin lorsque c'était nécessaire. Pour cela, des tests réguliers de ce qui avait été réalisé ainsi qu'une utilisation quasi-permanente de GitHub se sont avérés indispensables et plus qu'utile. Cela nous a permis de ne pas perdre de temps et tenir ainsi le planning suivant :

	Mode Tâche	Nom de la tâche	Durée	Début	Fin	Prédécesseurs
1		▲ <b>Gestionnaire de mots de passe</b>	<b>30 jours</b>	<b>Ven 10/11/17</b>	<b>Jeu 21/12/17</b>	
2		Lancement des projets	0 jour	Ven 10/11/17	Ven 10/11/17	
3		▲ <b>Formalisation du projet</b>	<b>7 jours</b>	<b>Ven 10/11/17</b>	<b>Lun 20/11/17</b>	
4		Choix de la BDD (structures, contenu, ...)	7 jours	Ven 10/11/17	Lun 20/11/17	2
5		Choix des fonctionnalités	7 jours	Ven 10/11/17	Lun 20/11/17	2
6		▲ <b>Interfaces</b>	<b>18 jours</b>	<b>Mar 21/11/17</b>	<b>Jeu 14/12/17</b>	
7		Création des designs d'interface	7 jours	Mar 21/11/17	Mer 29/11/17	5
8		Implémentation des interactions avec l'utilisateur	9 jours	Jeu 30/11/17	Mar 12/12/17	7
9		▲ <b>Tests</b>	<b>2 jours</b>	<b>Mer 13/12/17</b>	<b>Jeu 14/12/17</b>	
10		Rédaction d'un protocole de test des interfaces	0 jour	Mer 13/12/17	Mer 13/12/17	8
11		Test des interfaces	1 jour	Mer 13/12/17	Mer 13/12/17	8
12		Modifications des interfaces	1 jour	Jeu 14/12/17	Jeu 14/12/17	11



13		➡	➡ Relations avec la BDD	17 jours	Mar 21/11/17	Mer 13/12/17	
14		➡	Création des scripts SQL	0 jour	Mar 21/11/17	Mar 21/11/17	5
15		➡	Création des classes	2 jours	Mer 22/11/17	Jeu 23/11/17	14
16		➡	Mapping	0 jour	Ven 24/11/17	Ven 24/11/17	15
17		➡	Mise en place de Nhibernate	11 jours	Ven 24/11/17	Ven 08/12/17	16
18		➡	➡ Tests	3 jours	Lun 11/12/17	Mer 13/12/17	
19		➡	Rédaction du programme de test	1 jour	Lun 11/12/17	Lun 11/12/17	17
20		➡	Tests des requêtes SQL utilisées dans les classes	1 jour	Mar 12/12/17	Mar 12/12/17	19
21		➡	Corrections, modifications et ajouts	1 jour	Mer 13/12/17	Mer 13/12/17	20
22		➡	Rédaction du rapport	3 jours	Ven 15/12/17	Mar 19/12/17	21
23		➡	➡ Vérification globale finale	1 jour	Mer 20/12/17	Mer 20/12/17	
24		➡	Dernière série de tests (vérification finale)	1 jour	Mer 20/12/17	Mer 20/12/17	22
25		➡	Relecture du rapport	1 jour	Mer 20/12/17	Mer 20/12/17	22
26		➡	Envoi du projet	0 jour	Jeu 21/12/17	Jeu 21/12/17	25

### Planning



### Diagramme de Gantt

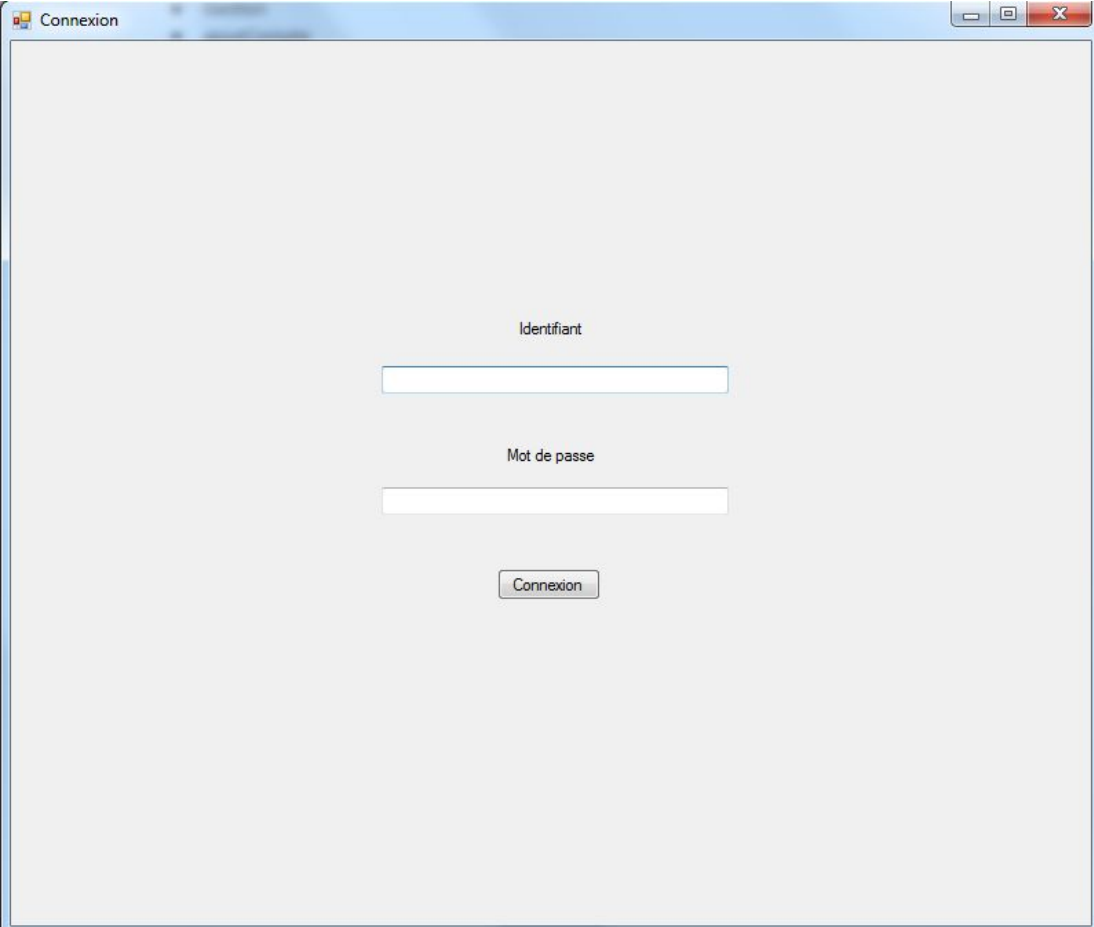
## II. Spécifications détaillées

---

### A. Formulaires de l'application

Notre application est composée de 9 formulaires distincts :

- Connexion
- Gestion
- ajoutCompte
- ajoutWifi
- modifierCompte
- modifierWifi
- supprimerCompte
- supprimerWifi
- mdp



The screenshot shows a window titled 'Connexion'. Inside the window, there are two text input fields. The first field is labeled 'Identifiant' and the second field is labeled 'Mot de passe'. Below these fields is a button labeled 'Connexion'.

Connexion.cs

Gestion

Comptes Points Wifi

Nom	Faible Mdp	Ancien Mdp
titre	*	

Ajouter un nouveau compte

Ajouter un nouveau point Wifi

INFORMATIONS SUR CE COMPTE

Nom : titre

Login : login

Mot de passe : mdp

Dernière modification : 23/11/2017 00:00:00

Modifier Supprimer

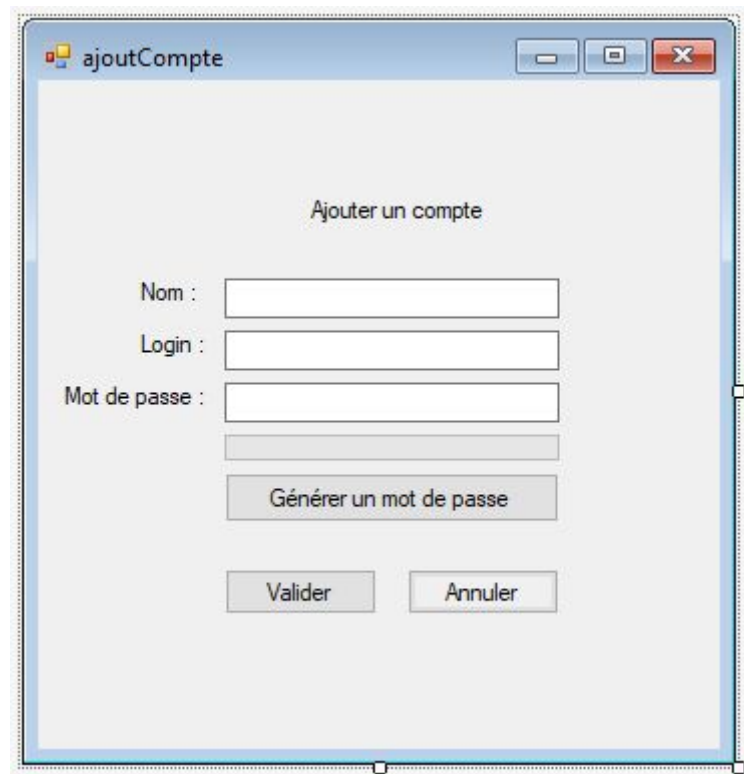
INFORMATIONS SUR CE WIFI

SSID :

Mot de passe :

Modifier Supprimer

Gestion.cs



A screenshot of a Java Swing window titled "ajoutCompte". The window has a standard Mac OS X-style title bar with minimize, maximize, and close buttons. The main content area has a light gray background and is titled "Ajouter un compte". It contains three text input fields labeled "Nom :", "Login :", and "Mot de passe :". Below the "Mot de passe :" field is a disabled grayed-out field. Below that is a button labeled "Générer un mot de passe". At the bottom are two buttons: "Valider" and "Annuler".

[ajoutCompte.cs](#)



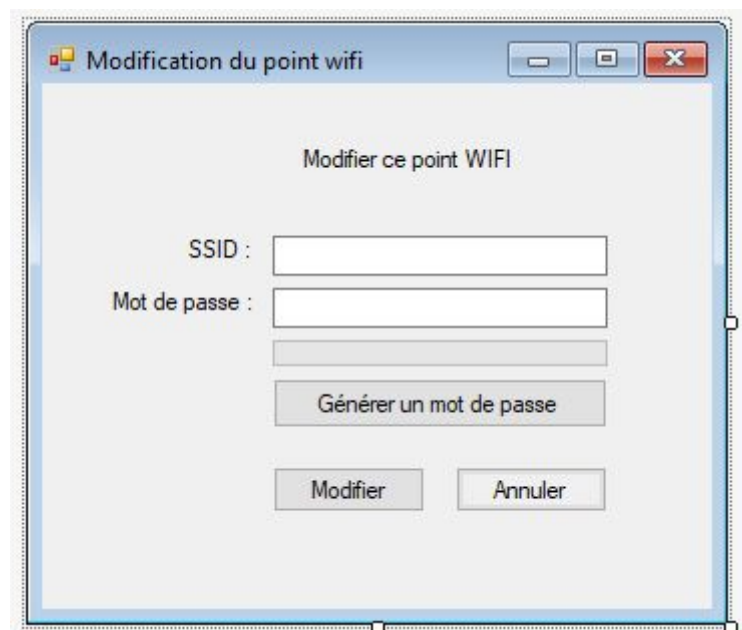
A screenshot of a Java Swing window titled "ajoutWifi". The window has a standard Mac OS X-style title bar with minimize, maximize, and close buttons. The main content area has a light gray background and is titled "Ajouter un point WIFI". It contains two text input fields labeled "SSID :" and "Mot de passe :". Below the "Mot de passe :" field is a disabled grayed-out field. Below that is a button labeled "Générer un mot de passe". At the bottom are two buttons: "Valider" and "Annuler".

[ajoutWifi.cs](#)



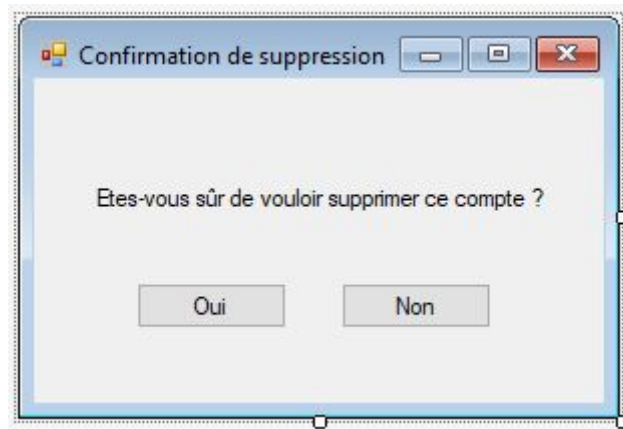
The screenshot shows a window titled "Modification du compte" with a standard Windows-style title bar (minimize, maximize, close buttons). The main content area has a title "Modifier le compte". Below this, there are three input fields labeled "Nom :", "Login :", and "Mot de passe :". The "Mot de passe" field is followed by a second, empty input field. Below these fields is a button labeled "Générer un mot de passe". At the bottom of the form are two buttons: "Modifier" and "Annuler".

[modifierCompte.cs](#)

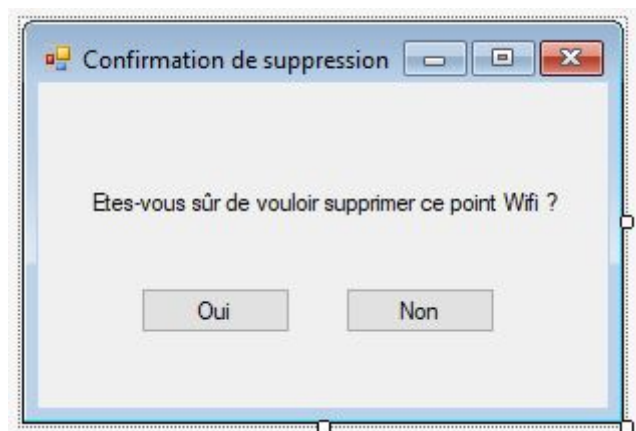


The screenshot shows a window titled "Modification du point wifi" with a standard Windows-style title bar. The main content area has a title "Modifier ce point WIFI". Below this, there are two input fields labeled "SSID :" and "Mot de passe :". The "Mot de passe" field is followed by a second, empty input field. Below these fields is a button labeled "Générer un mot de passe". At the bottom of the form are two buttons: "Modifier" and "Annuler".

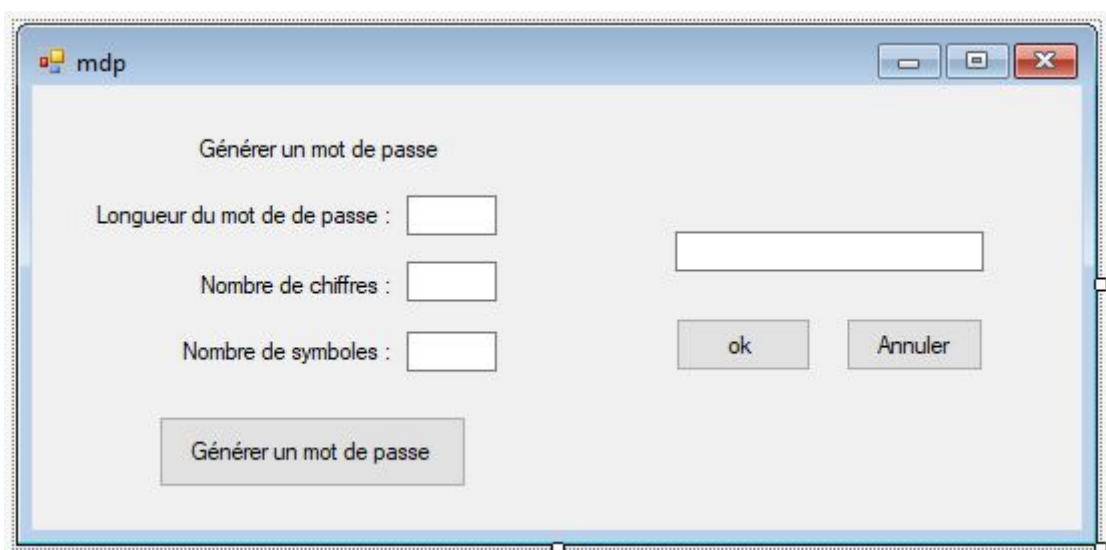
[modifierWifi.cs](#)



[supprimerCompte.cs](#)



[supprimerWifi.cs](#)



[mdp.cs](#)

## B. Justification des choix

### Base de données (BDD) :

La choix a été fait de placer des liaisons \*/\* entre les Utilisateurs et les Comptes d'une part et les Utilisateurs et les Wifi d'autre part. En effet deux utilisateurs différents pourraient parfaitement partager un même compte (Spotify par exemple) de même qu'un point Wifi pourrait être commun à deux personnes (habitants d'un même appartement par exemple). Donc si le mot de passe du compte est modifié par un utilisateur, il est modifié pour tous les utilisateurs ayant ce compte d'enregistré.

Il faut noter que la suppression ne supprime évidemment que le compte lié à l'utilisateur connecté et laisse donc encore aux autre utilisateurs l'accès aux données. En revanche lorsqu'un Compte ou un Wifi n'est plus lié à aucun Utilisateur, il est supprimé de la BDD.

### Application WinForm (App) :

#### Fichiers "Program.cs" / "Connexion.cs" :

Pour toutes les fonctions ayant besoin des données de la BDD, on a utilisé la fonction *GetAll()* de la classe qui nous intéressait puis nous avons travaillé sur la liste obtenu. Ceci n'est peut être pas la manière la plus optimisée mais nous a permis d'assurer l'accès aux informations de la BDD quand nous en avons besoin.

Lorsque l'action n'est pas possible ou ne s'effectue pas, l'application affiche un message d'erreur au sein d'une pop-up.

Lorsqu'une application Winform est lancée, la première fenêtre lancée est retenue comme étant la fenêtre principale. Cela signifie que si on la ferme, l'application s'arrête. Nous souhaitons en revanche que la page de connexion se ferme une fois l'utilisateur connecté. Pour modifier cela, nous avons donc annulé la fermeture de l'application dans "Program.cs" puis ajouté des conditions sur la fermeture de la page "Connexion.cs" ainsi que sur celle de "Gestion.cs", le formulaire principal de notre application.

Ainsi, l'application se ferme dans deux cas seulement : si l'utilisateur ferme la page de connexion avant même de se connecter ou s'il ferme la page de gestion une fois connecté.

Nous avons codé en premier la partie de connexion puisque c'était le plus important. En effet, toute la partie gestion des mots de passe et points wifi dépendent de l'utilisateur connecté.

**Fichier "mdp.cs" :**

Lorsqu'on génère un mot de passe (en cliquant sur le bouton dédié), la fonction prend en compte les paramètres "Longueur du mot de passe", "Nombre de chiffres" et "Nombre de symboles" contenus dans le mot de passe.

- Si l'utilisateur n'entre rien pour la longueur du mot de passe, le mot de passe est défini à 8 caractères dont 2 chiffres et 2 symboles.
- Si l'utilisateur ne rentre pas de longueur mais un nombre de chiffres et de symboles, la longueur du mot de passe est égale à la somme des deux.
- Si l'utilisateur n'entre rien pour le nombre de chiffres ou le nombre de symboles un nombre est choisi aléatoirement (sans dépasser la longueur totale fixée). Une case vide n'est donc pas interprétée comme un 0 mais plutôt comme si l'utilisateur ne sait simplement pas combien mettre, ou n'a pas de préférence particulière.
- Si l'utilisateur entre un chiffre négatif, le mot de passe prend en compte la valeur absolue.

Ce gestionnaire est évidemment accessible dès que l'utilisateur a la possibilité de modifier ou d'ajouter un mot de passe.

**Fichier "Gestion.cs" :**

Pour afficher les wifi, nous avons décidé d'utiliser une simple liste alors que pour les comptes nous avons utilisé une DataGridView. Cet outil nous permet d'afficher plusieurs colonnes et de les trier, l'utilisateur a donc la capacité, en cliquant sur le titre de la colonne, de trier ses comptes par titre, par mots de passe anciens ou par mots de passe faible. Le choix de faire une seule et même interface de gestion avec toutes les actions possibles est volontaire. En effet, il n'y a que peu d'actions à réaliser, ce qui ne surcharge pas l'utilisateur qui se retrouve alors avec toutes les fonctions à portée de main. De plus, si l'utilisateur désire quitter le gestionnaire, un message de confirmation s'affiche.

Tous les boutons de cette interface amène à des fenêtres qui s'ouvrent de manière modale, c'est-à-dire que l'utilisateur est obligé de les fermer en annulant ou validant son action. Ceci permet à l'utilisateur d'avoir un minimum de fenêtre ouverte à la fois et qu'il sache avec laquelle il doit interagir.

Le fichier "*Gestion.cs*" recharge également le contenu de l'affichage des Comptes ou Wifi si une modification est effectuée sur l'un des deux. L'affichage est séparé en deux pour que l'application n'est pas à recharger l'intégralité des affichages à chaque interaction.



**Fichiers “ajoutCompte.cs” / “ajoutWifi.cs” / “modifierWifi.cs” / “modifierWifi.cs” :**

Dans ces fichiers, un message d’erreur s’affiche si l’utilisateur ne remplit pas correctement les formulaires. Également, l’utilisateur ne peut pas ajouter/modifier un compte ou un wifi ayant respectivement un Titre ou un SSID déjà utilisé par un autre compte ou wifi. En effet, ceci générerait l’affichage des détails des comptes et wifis car les fonctions qui récupèrent les informations des comptes/wifis sélectionnés affichent le premier compte/wifi de l’utilisateur dont le Titre ou SSID correspond.

Toutes les textbox contenant un mot de passe sont liées à une barre de force. Lorsque le texte change, la force du mot de passe est calculée et automatiquement affichée. Ce calcul est détaillé dans la partie suivante.

## C. Modalités de calcul de la force du mot de passe

Notre programme considère qu’un mot de passe est composé de 4 paramètres différents : s’il a plus ou moins de 8 caractères, s’il a des lettres, des chiffres et/ou des symboles. Suivant le nombre de paramètres différents qu’il respecte ou non, un mot de passe peut être de force faible, modéré ou fort. Ainsi, si un mot de passe ne possède qu’un des paramètres, il est faible. S’il possède les quatre paramètres, le mot de passe est considéré fort. Et sinon, il est dit modéré.

On peut noter qu’en revanche, il ne prend pas en compte la différence entre les majuscules et les minuscules.

### III. Résultats et tests

Un cahier des charges ayant été fourni à l'origine du projet, il s'agissait, tout au long de la création de notre gestionnaire de mot de passe de régulièrement comparer le travail réalisé avec les exigences demandées, de tester ensuite celui-ci puis de le valider ou le retravailler selon les résultats aux tests.

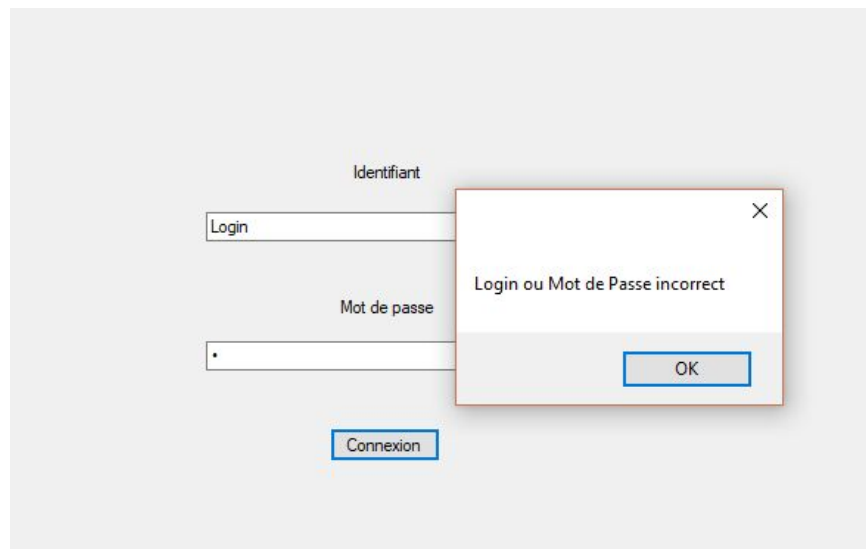
#### A. Exigences métier respectées par le programme

Code	Description	Exigences respectées
EF_01	L'utilisation de l'application est soumise à la saisie correcte d'un mot de passe principal	Oui, avant d'accéder aux comptes, l'utilisateur doit entrer son login et son mot de passe.
EF_02	Une fois authentifié, l'utilisateur a accès à la liste de ses différents comptes	Oui, une DataGridView permet d'afficher les titres des Comptes associés à l'utilisateur connecté.
EF_03	En sélectionnant le titre d'un compte dans la liste, les détails sur ce compte sont affichés	Oui, les éléments sont affichés dans des textbox.
EF_04	Les informations obligatoirement associées à un compte sont : titre, login, mot de passe	Oui, les informations sont sauvegardées dans la BDD.
EF_05	La force du mot de passe associé à un compte est évaluée et affichée, idéalement sous forme graphique	Oui, dès qu'un compte affiche ses informations associées, la force du mot de passe est calculée par une fonction et est affichée dans une barre de progression.
EF_06	L'application permet d'ajouter un nouveau compte en saisissant ses informations	Oui, une nouvelle fenêtre s'ouvre alors pour entrer les nouvelles informations.
EF_07	L'application permet de modifier les informations sur un compte	Oui, une nouvelle fenêtre s'ouvre alors pour entrer les modifications voulues.
EF_08	L'application permet, après confirmation, de supprimer un compte	Oui, la confirmation se fait via une fenêtre pop-up.

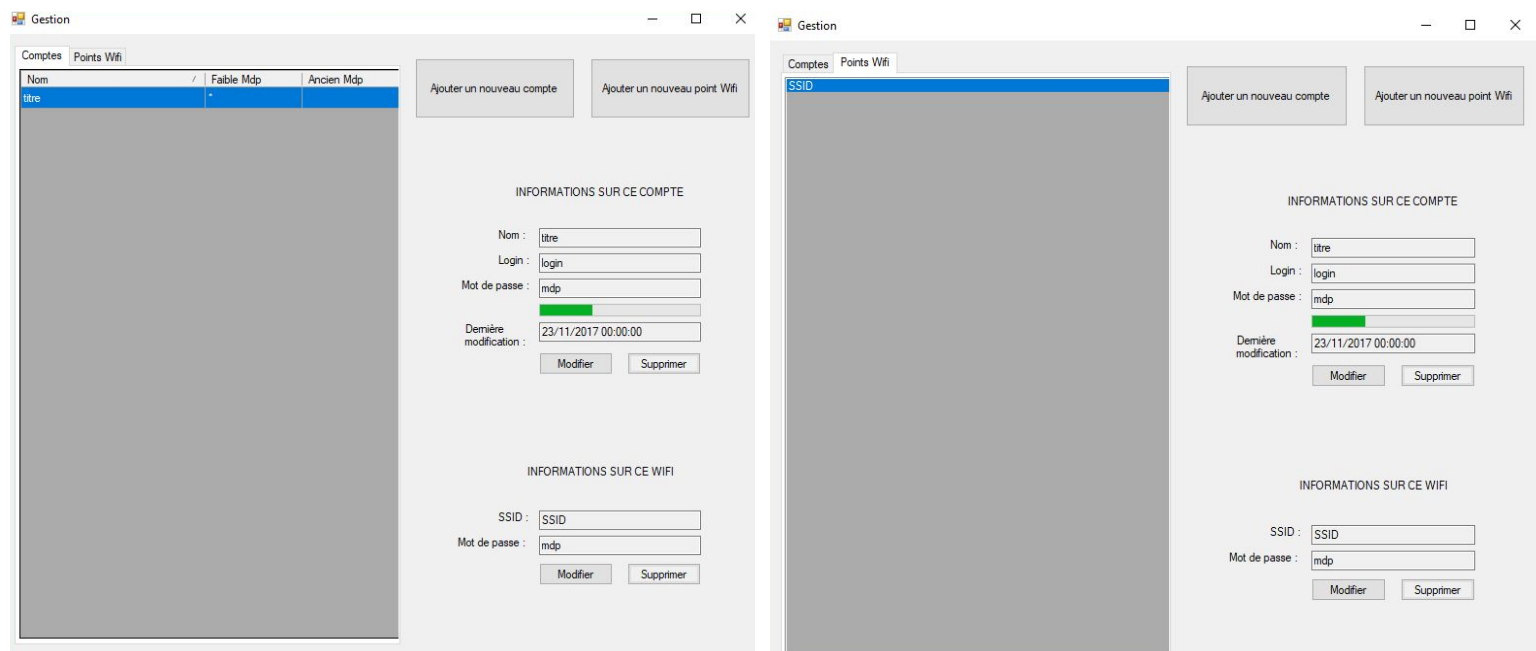
EF_09	L'application offre un mécanisme pour générer automatiquement un mot de passe aléatoire. L'utilisateur choisit la longueur (nombre de caractères) ainsi que le nombre de chiffres et de symboles dans le mot de passe généré	Oui, l'utilisateur peut générer un mot de passe quand il ajoute un compte/wifi ou lors de la modification d'un compte/wifi.
EF_10	L'application permet également de gérer la liste de points d'accès Wi-Fi de l'utilisateur (informations obligatoires pour un point d'accès : SSID et mot de passe)	Oui, un onglet permet d'accéder aux comptes ou aux points wifi de l'utilisateur connecté.
EF_11	L'application permet de gérer les mots de passe de plusieurs utilisateurs. Le mot de passe principal saisi permet d'identifier l'utilisateur	Oui, lors de la connexion le login et le mdp entrés permettent de déterminer qui est l'utilisateur connecté et d'ouvrir sa session.
EF_12	L'application gère la date de dernière modification d'un compte. Cette date est calculée automatiquement	Oui, une fonction séparée permet d'effectuer ce calcul.
EF_13	L'application affiche séparément la liste des comptes ayant un mot de passe faible	Oui, les comptes sont symbolisés par une astérisque dans la colonne associée du tableau (possibilité de trier le tableau en cliquant sur la colonne souhaitée).
EF_14	L'application affiche séparément la liste des comptes ayant un mot de passe non modifié depuis plus de 6 mois	Oui, les comptes sont symbolisés par une astérisque dans la colonne associé du tableau (possibilité de trier le tableau en cliquant sur la colonne souhaitée).

## B. Résultats de notre programme

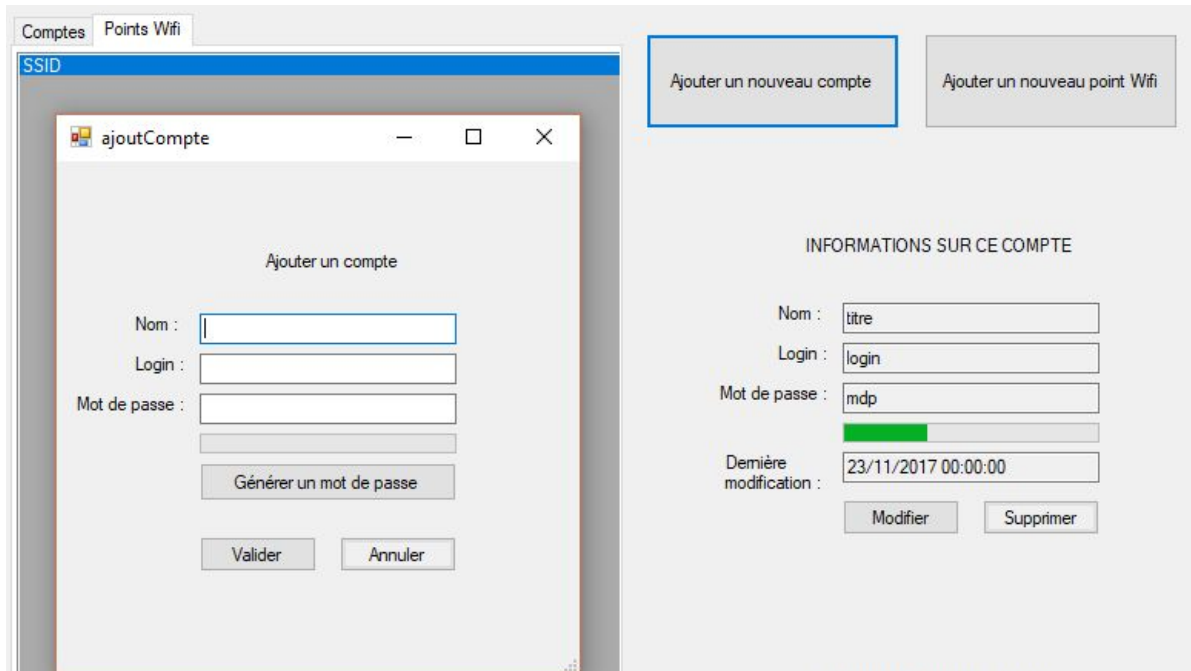
Ci-dessous se trouvent les résultats de notre application pour la gestion des comptes. Pour ne pas surcharger notre rapport, nous n'avons pas mis la gestion des points Wifi qui s'effectue de la même manière, avec des interfaces semblables.



Message d'erreur de connexion si le login ou mot de passe est incorrect



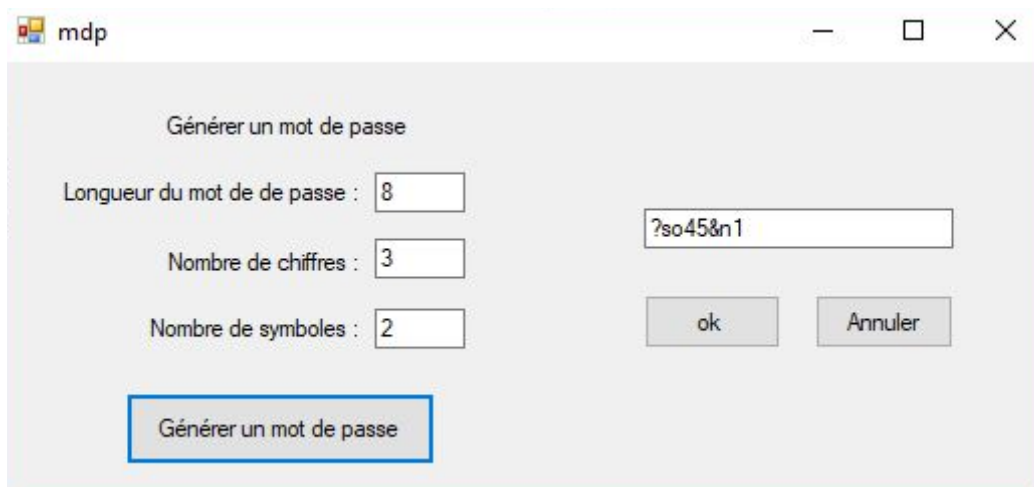
Affichage des comptes et points wifis après connexion



The screenshot shows a software interface with two tabs: 'Comptes' and 'Points Wifi'. The 'Comptes' tab is active, displaying a window titled 'ajoutCompte'. Inside this window, there are two main sections:

- Ajouter un compte:** This section contains input fields for 'Nom', 'Login', and 'Mot de passe'. Below the 'Mot de passe' field is a button labeled 'Générer un mot de passe'. At the bottom of this section are 'Valider' and 'Annuler' buttons.
- Informations sur ce compte:** This section displays the details of the account being added, including 'Nom' (titre), 'Login' (login), 'Mot de passe' (mdp), and 'Dernière modification' (23/11/2017 00:00:00). It also includes 'Modifier' and 'Supprimer' buttons.

Ajout d'un compte



The screenshot shows a window titled 'mdp' with a section titled 'Générer un mot de passe'. It contains three input fields for password generation criteria:

- Longueur du mot de de passe : 8
- Nombre de chiffres : 3
- Nombre de symboles : 2

Below these fields is a button labeled 'Générer un mot de passe'. To the right, there is a text box displaying the generated password '?so45&n1' and two buttons labeled 'ok' and 'Annuler'.

Génération d'un mot de passe de 8 caractères dont 3 chiffres et 2 symboles

ajoutCompte

Ajouter un compte

Nom :

Login :

Mot de passe :

Création d'un compte avec un mot de passe fort

Comptes Points Wifi

Nom	Faible Mdp	Ancien Mdp
titre	*	
CompteTest		

Ajouter un nouveau compte

Ajouter un nouveau point Wifi

INFORMATIONS SUR CE COMPTE

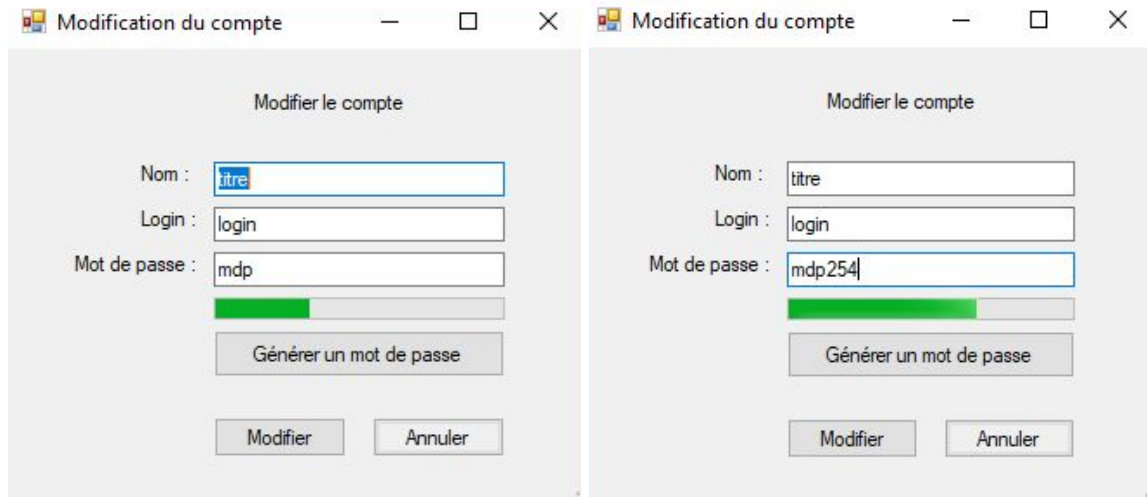
Nom :

Login :

Mot de passe :

Dernière modification :

Ajout du compte à la liste des comptes de l'utilisateur



Modification du compte

Modifier le compte

Nom : titre

Login : login

Mot de passe : mdp

Générer un mot de passe

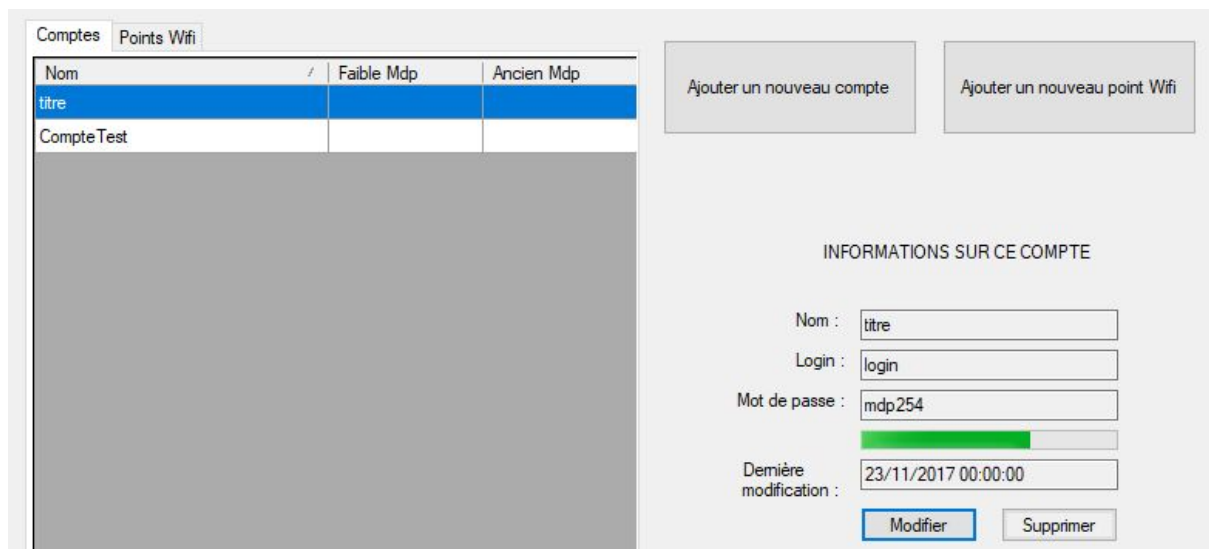
Modifier Annuler

Mot de passe : mdp254

Générer un mot de passe

Modifier Annuler

Récupération des données du compte à modifier et création d'un mot de passe moyen



Comptes Points Wifi

Nom	Faible Mdp	Ancien Mdp
titre		
Compte Test		

Ajouter un nouveau compte Ajouter un nouveau point Wifi

INFORMATIONS SUR CE COMPTE

Nom : titre

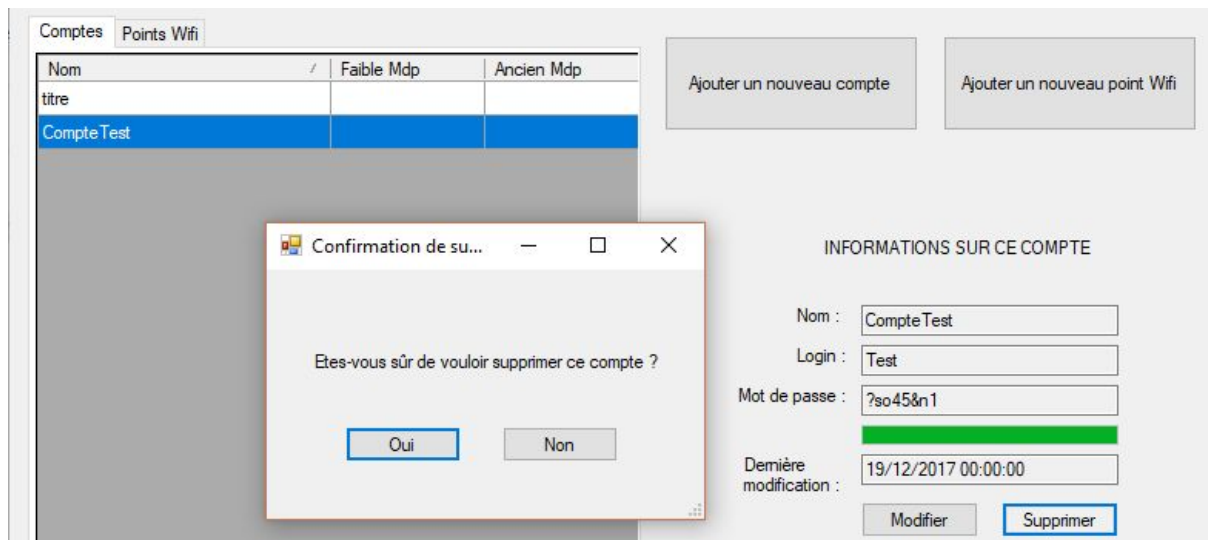
Login : login

Mot de passe : mdp254

Dernière modification : 23/11/2017 00:00:00

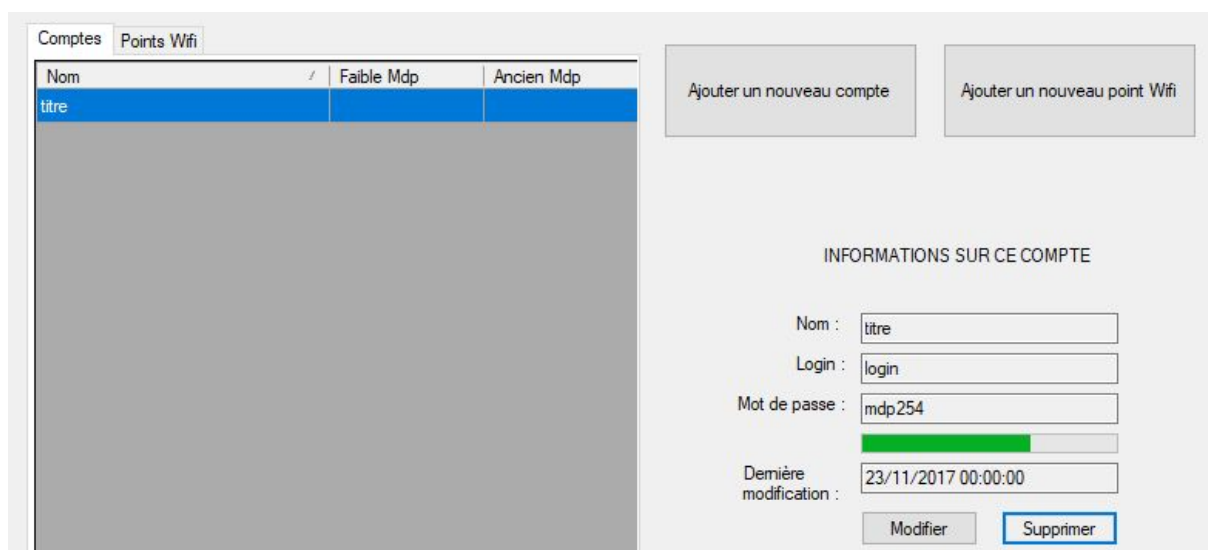
Modifier Supprimer

Le compte a bien été modifié, le mot de passe n'est plus faible



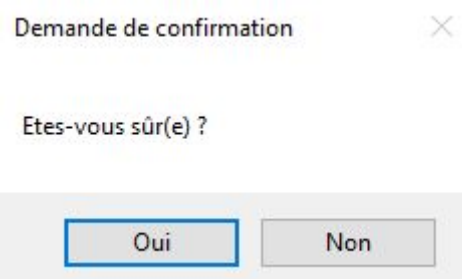
The screenshot shows the 'Comptes' application interface. A table lists accounts with columns 'Nom', 'Faible Mdp', and 'Ancien Mdp'. The 'Compte Test' account is selected. A modal dialog titled 'Confirmation de suppression' is displayed, asking 'Etes-vous sûr de vouloir supprimer ce compte ?' with 'Oui' and 'Non' buttons. To the right, the 'INFORMATIONS SUR CE COMPTE' section shows details for 'Compte Test' (Login: Test, Password: ?so45&n1, Last modification: 19/12/2017 00:00:00) and buttons for 'Modifier' and 'Supprimer'.

Demande de confirmation de suppression du compte sélectionné



The screenshot shows the 'Comptes' application interface after the deletion of 'Compte Test'. The table now lists the 'titre' account. The 'INFORMATIONS SUR CE COMPTE' section shows details for 'titre' (Login: login, Password: mdp254, Last modification: 23/11/2017 00:00:00) and buttons for 'Modifier' and 'Supprimer'.

Le compte a été supprimé



The screenshot shows a simple confirmation dialog box titled 'Demande de confirmation' with a close button (X). The text inside asks 'Etes-vous sûr(e) ?' and provides 'Oui' and 'Non' buttons.

Confirmation avant de quitter l'application



## C. Description des protocoles de tests

Nos protocoles de tests se sont décomposés en deux parties distinctes : dans un premier temps tester la bonne liaison avec la BDD, afin d'être sûr que les interactions qui allaient être effectuées fonctionnent correctement, et dans un second temps tester la gestion des événements via le WinForm, qui va d'ailleurs lui-même utiliser les requêtes testées précédemment.

### Protocole de Tests de la relation avec la BDD :

Ce protocole de test, que l'on retrouve entièrement dans le fichier "*Program.cs*" du projet "*Test*" se décompose en 3 parties, correspondantes à nos 3 tables de données principales : les tests liés aux utilisateurs, ceux liés aux comptes et enfin ceux liés aux points wifi.

On obtient ainsi le protocole suivant :

#### Utilisateurs :

- Récupérer la liste de tous les utilisateurs
- Récupérer un utilisateur dont on connaît l'identifiant

#### Comptes :

- Récupérer la liste de tous les comptes
- Récupérer la liste des doublets d'identifiant comptes/utilisateurs qui sont liés
- Créer un nouveau compte et le lier à son utilisateur
- Créer ce même compte mais avec un autre utilisateur et le lier à celui-ci (2 utilisateurs ont accès au même compte)
- Modifier ce compte
- Supprimer ce compte pour un utilisateur (ne doit pas retirer l'accès de l'autre utilisateur)
- Supprimer ce compte pour l'autre utilisateur également (le compte doit disparaître puisque plus personne ne l'utilise)

#### Wifis :

- Récupérer la liste de tous les points wifis
- Récupérer la liste des doublets d'identifiant wifi/utilisateurs qui sont liés
- Créer un nouveau wifi et le lier à son utilisateur
- Créer ce même wifi mais avec un autre utilisateur et le lier à celui-ci (2 utilisateurs ont accès au même wifi)
- Modifier ce wifi
- Supprimer ce wifi pour un utilisateur (ne doit pas retirer l'accès de l'autre utilisateur)
- Supprimer ce wifi pour l'autre utilisateur également (le wifi doit disparaître puisque plus personne ne l'utilise)

### Protocole de Tests sur l'App :

Ce protocole de test à servi non plus à tester les liaisons avec la BDD mais les interactions avec l'utilisateur, les différents scénarios qui pouvaient arriver, vérifier qu'il ne pouvait pas cliquer à des endroits qui auraient généré des erreurs ou, au contraire, qu'il pouvait bien effectuer toutes les actions possibles.

Ces tests se réalisant via le WinForm créé, c'est aussi l'occasion de vérifier le bon affichage des données, lorsqu'on ajoute un compte par exemple, que l'on sélectionne un wifi, etc.

Le protocole mis au point est donc le suivant :

Quitter l'App sans connexion

Relancer l'application

Se connecter avec un login et un mot de passe invalides

Se connecter avec un login et un mot de passe valides

Tests sur les Comptes :

- Ajouter un nouveau Compte avec un mot de passe faible
- Afficher le Compte ajouté
- Modifier le mot de passe du Compte
- Supprimer le Compte

Tests sur les Wifis :

- Ajouter un nouveau point wifi avec un mot de passe faible
- Afficher le point wifi ajouté
- Modifier le mot de passe du point Wifi avec un mot de passe généré
- Supprimer le point Wifi

Tests sur les mots de passe :

- Générer un mot de passe aléatoire
- Générer un mot de passe d'une longueur définie
- Générer un mot de passe avec un nombre de chiffres défini
- Générer un mot de passe avec un nombre de symboles défini
- Générer un mot de passe avec un nombre de chiffres et de symboles défini
- Générer un mot de passe avec une longueur définie et un nombre de chiffres ou de symboles défini
- Générer un mot de passe avec une longueur définie et un nombre de chiffres et de symboles défini

Quitter l'application

## D. Résultats des tests

Les tests sur l'application WinForm nous ont permis de vérifier le bon fonctionnement de toutes nos fonctions et mettre en lumière les erreurs présentes le cas échéant. Cela inclut donc les fonctions "*ajoutCompte.cs*", "*modifierCompte.cs*" et "*supprimerCompte.cs*", leur équivalents pour un point wifi et finalement les mises à jour et interactions avec la BDD.

En effet, on a pu remarquer et ainsi corriger plusieurs erreurs de cette manière. Par exemple, les liaisons lors de la modification des éléments se rajoutaient même si elles existaient déjà (duplication de l'information). Ce problème résultait en réalité d'une erreur de notre part lors d'une modification effectuée au préalable sur cette fonction. On a également pu s'apercevoir de certains problèmes d'affichage tels que des textboxes toujours remplies après désélection de tous les éléments.

## IV. Bilan et perspectives

---

Ce projet à présent terminé, il est possible de prendre un peu de recul pour effectuer un bilan rapide de ce qui a été fait ainsi que de ce qui pourrait être fait à posteriori pour améliorer l'ensemble de l'application réalisée.

### A. Points positifs et négatifs

Plusieurs points peuvent être cités dans ce paragraphe, témoignant des aspects positifs ou négatifs de notre rendu.

En effet, on peut tout d'abord parler de la gestion de situations pas forcément décrite dans le cahier des charges originel telles que le cas où deux utilisateurs A et B utilisent le même compte, si la personne A supprime alors son compte de son gestionnaire de mot de passe, ce compte ne doit pas être purement et simplement enlevé de la BDD. Seul l'accès de la personne A aux informations de ce compte doit être retiré. Il en va de même pour les points wifi.

On peut également parler d'un bon fonctionnement global de l'application ainsi que d'une relation avec la BDD vérifiée et fonctionnelle. Cette focalisation première sur le fait d'avoir une application qui fonctionne a en revanche été autant de travail en moins sur le design et l'ergonomie de l'interface. Ceci s'est fait dans l'optique de répondre au principe KISS vu en cours et faire, avant toute autre chose, quelque chose de fonctionnel.

Un bon point supplémentaire fut la bonne communication au sein de notre binôme ainsi que la motivation qui nous ont permis d'avancer efficacement et de tenir parfaitement les délais imposés, sans avoir à se précipiter.

Enfin, les différentes saisies et actions que peut effectuer l'utilisateur sur notre application étant très nombreuses, cela a compliqué la mise en place d'une application robuste. En effet, il est difficile d'imaginer toutes les combinaisons possibles. Des tests utilisateurs pourraient alors nous aider à trouver les failles que nous n'avons pas pu imaginer.

### B. Évolutions possibles

Logiquement, les évolutions possible d'un produit découlent de choses manquantes ou imparfaites de celui-ci. Ainsi un des points d'améliorations sera bien évidemment le design, l'intuitivité, l'ergonomie de notre gestionnaire de mot de passes.

Mais on peut également lister quelques fonctionnalités mineures non mises en place et dont les plus-value sont indéniables. On retrouvera par exemple pour les mots de passe le fait de mettre en place une différenciation entre les minuscules et majuscules ou de changer les couleurs de la barre de force (rouge, orange, vert) pour mieux différencier et comprendre les différents niveaux de force existants. Pourquoi pas imaginer également une plus grande quantité de symboles différents.

Un ajout qui nous semble également évident serait la possibilité d'ajouter un nouvel utilisateur, de supprimer un utilisateur, de gérer son login et son mot de passe une fois connecté ou encore de pouvoir récupérer ses informations par mail en cas d'oubli.

Parmi nos idées mais que nous n'avons pas mis en place il reste encore une meilleure communication avec l'utilisateur en cas d'échec de ses requêtes. Par exemple, si celui-ci veut modifier son compte mais entre des informations strictement identiques à un autre compte alors la modification n'aura pas lieu mais notre utilisateur n'est pas prévenu.

Enfin, une fois ces évolutions de l'application effectuées, la mise en place de tests utilisateurs sur un échantillon d'environ 10 individus serait un bon point pour valider ce qui a été fait et ainsi affirmer que notre solution est à la fois intuitive et facile à utiliser.

En cherchant, on pourrait encore trouver pleins d'autres détails à ajouter pour perfectionner notre rendu (lien direct vers le site internet du compte sélectionné, affichage du logo de ce site, de votre avatar sur celui-ci, ...) mais le produit parfait n'existant pas, ces améliorations sont infinies et on ne s'attardera donc pas à en lister plus ici.

# Annexes

## Exigences techniques

Code	Description
ET_01	L'application est réalisée sous Windows à l'aide de la technologie WinForms
ET_02	Les données persistantes sont stockées dans une base de données relationnelle MySQL
ET_03	L'application est structurée soit selon une architecture en couches, soit selon une architecture MVP
ET_04	Le lien entre la BD et les objets de l'application est fait soit manuellement, soit à l'aide d'un outil d'ORM
ET_05	L'application respecte autant que possible les grands principes de conception étudiés en cours : séparation des responsabilités, limitation de la duplication de code, KISS, YAGNI, etc
ET_06	L'ensemble du code source respecte la convention camelCase
ET_07	Les noms des classes, propriétés, méthodes, paramètres et variables sont choisis avec soin pour refléter leur rôle

## Exigences métiers

Code	Description
EF_01	L'utilisation de l'application est soumise à la saisie correcte d'un mot de passe principal
EF_02	Une fois authentifié, l'utilisateur a accès à la liste de ses différents comptes
EF_03	En sélectionnant le titre d'un compte dans la liste, les détails sur ce compte sont affichés
EF_04	Les informations obligatoirement associées à un compte sont : titre, login, mot de passe

EF_05	La force du mot de passe associé à un compte est évaluée et affichée, idéalement sous forme graphique
EF_06	L'application permet d'ajouter un nouveau compte en saisissant ses informations
EF_07	L'application permet de modifier les informations sur un compte
EF_08	L'application permet, après confirmation, de supprimer un compte
EF_09	L'application offre un mécanisme pour générer automatiquement un mot de passe aléatoire. L'utilisateur choisit la longueur (nombre de caractères) ainsi que le nombre de chiffres et de symboles dans le mot de passe généré
EF_10	L'application permet également de gérer la liste de points d'accès Wi-Fi de l'utilisateur (informations obligatoires pour un point d'accès : SSID et mot de passe)
EF_11	L'application permet de gérer les mots de passe de plusieurs utilisateurs. Le mot de passe principal saisi permet d'identifier l'utilisateur
EF_12	L'application gère la date de dernière modification d'un compte. Cette date est calculée automatiquement
EF_13	L'application affiche séparément la liste des comptes ayant un mot de passe faible
EF_14	L'application affiche séparément la liste des comptes ayant un mot de passe non modifié depuis plus de 6 mois