

Projet de Génie Logiciel

2017-2018

Rapport

Développeurs	Encadrants
Marine Lira	Baptiste Pesquet
Valentin Mancier	Pierre-Alexandre Favier
	Edwige Clermont

Table des matières

Table des matières	2
I - Spécifications générales	3
Analyse fonctionnelle	3
Besoins fonctionnels	3
Les exigences techniques	3
Modèle conceptuel de données	4
Diagramme de classes	4
Diagramme de séquences	7
Architecture de l'application	9
Planning et répartition des tâches	9
II - Spécifications détaillées	11
Formulaires	11
Connexion	11
Inscription	12
Formulaire principal	12
Ajout d'un compte	14
Ajout point wifi	14
Génération de mot de passe aléatoire	14
Mise à jour compte	15
Mise à jour point wifi	16
Choix de conception	16
Calcul de la force du mot de passe	17
III - Résultats et tests	18
Exigences métiers respectées	18
Protocoles de tests et résultats	19
IV - Bilan et perspectives	21

I - Spécifications générales

1. Analyse fonctionnelle

Besoins fonctionnels

L'application doit permettre à l'utilisateur de :

- S'inscrire
- S'identifier
- Accéder à la liste des comptes
- Accéder à la liste des comptes ayant un mot de passe faible
- Accéder à la liste des comptes ayant un mot de passe non modifié depuis plus de 6 mois
- Accéder aux détails d'un compte
- Ajouter un nouveau compte
- Modifier un compte
- Supprimer un compte
- Accéder à la liste des points d'accès wifi
- Accéder aux détails d'un point d'accès wifi
- Ajouter un nouveau point d'accès wifi
- Modifier un point d'accès wifi
- Supprimer un point d'accès wifi
- Générer automatiquement un mot de passe aléatoire à partir du nombre de caractères, de chiffres et de symboles.

Les exigences techniques

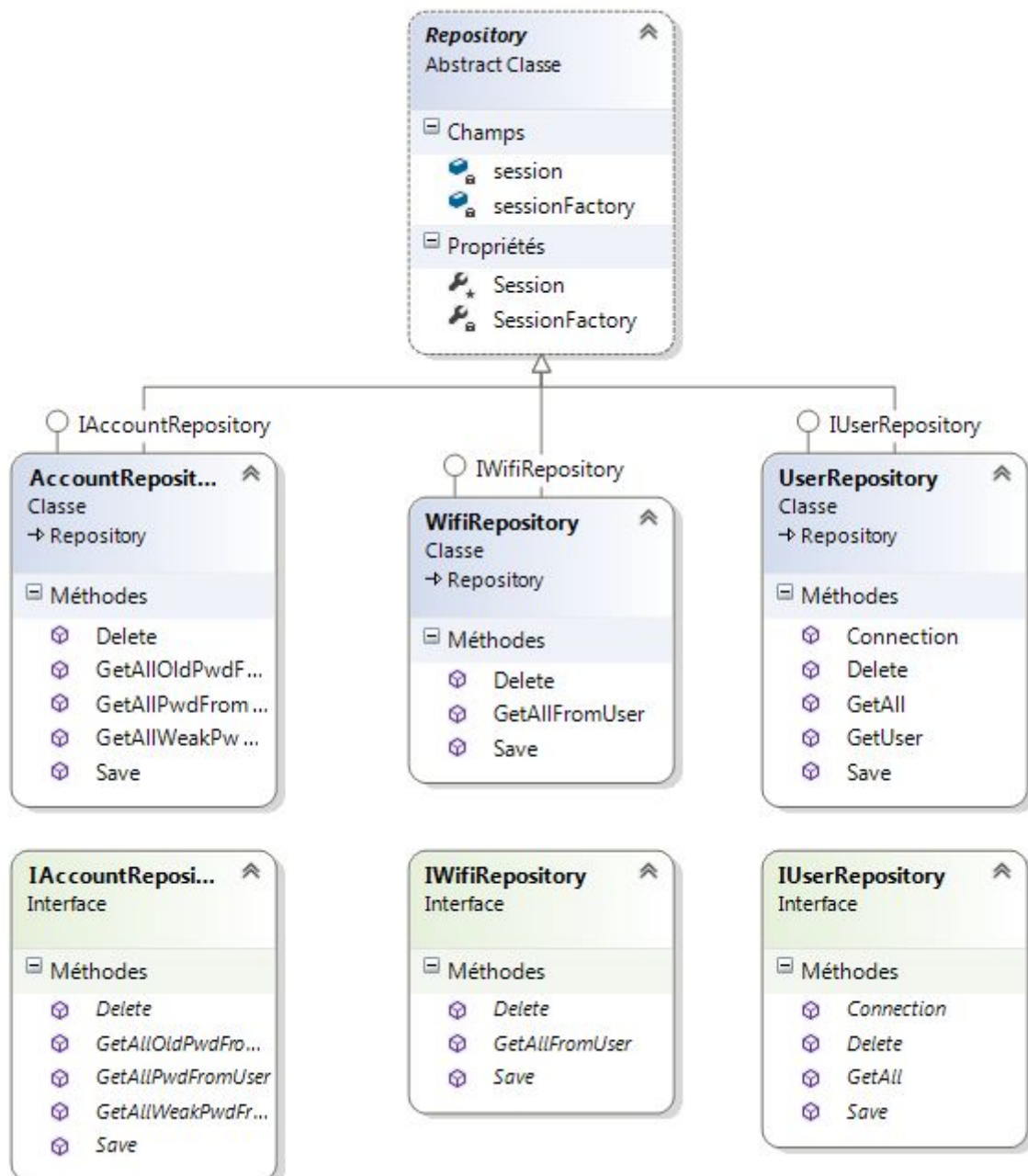
Code	Description
ET_01	L'application est réalisée sous windows à l'aide de la technologie WinForms
ET_02	Les données persistantes sont stockées dans une base de données relationnelle MySQL
ET_03	L'application est structurée soit selon une architecture en couches, soit selon une architecture MVP
ET_04	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 la conception étudiés en cours : séparation des responsabilités, limitation de 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

2. Modèle conceptuel de données

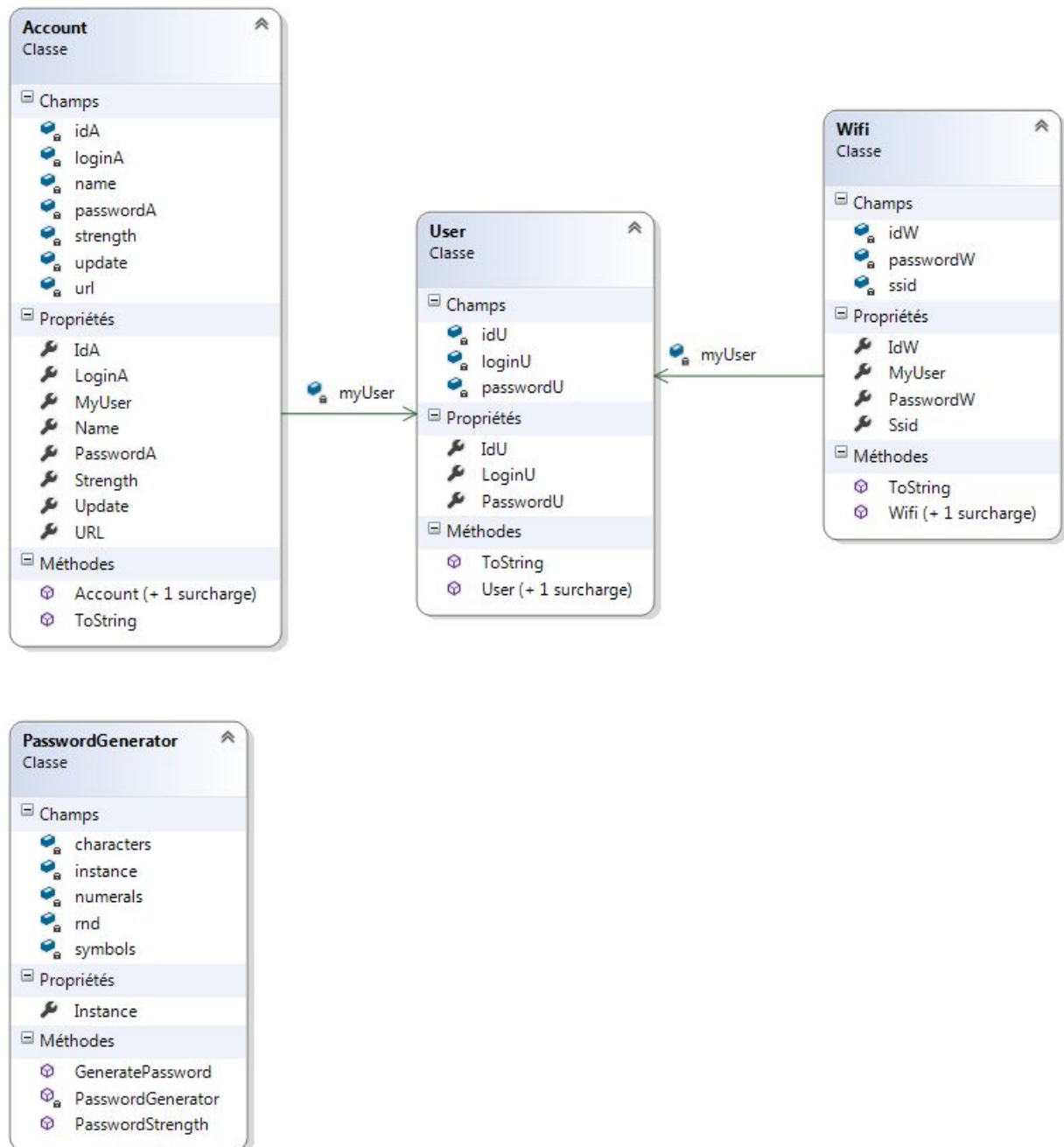


3. Diagramme de classes

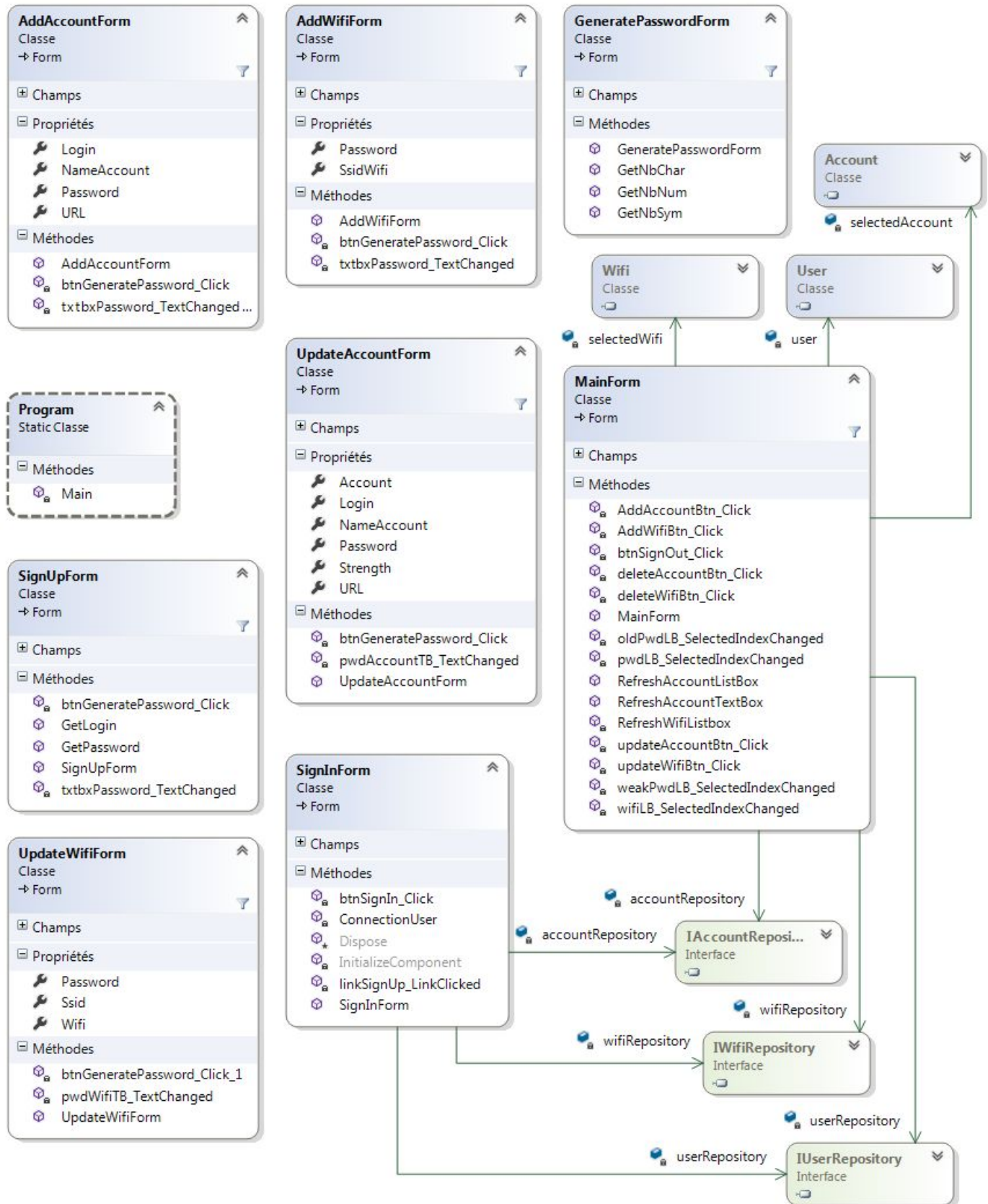
DAL



Domain

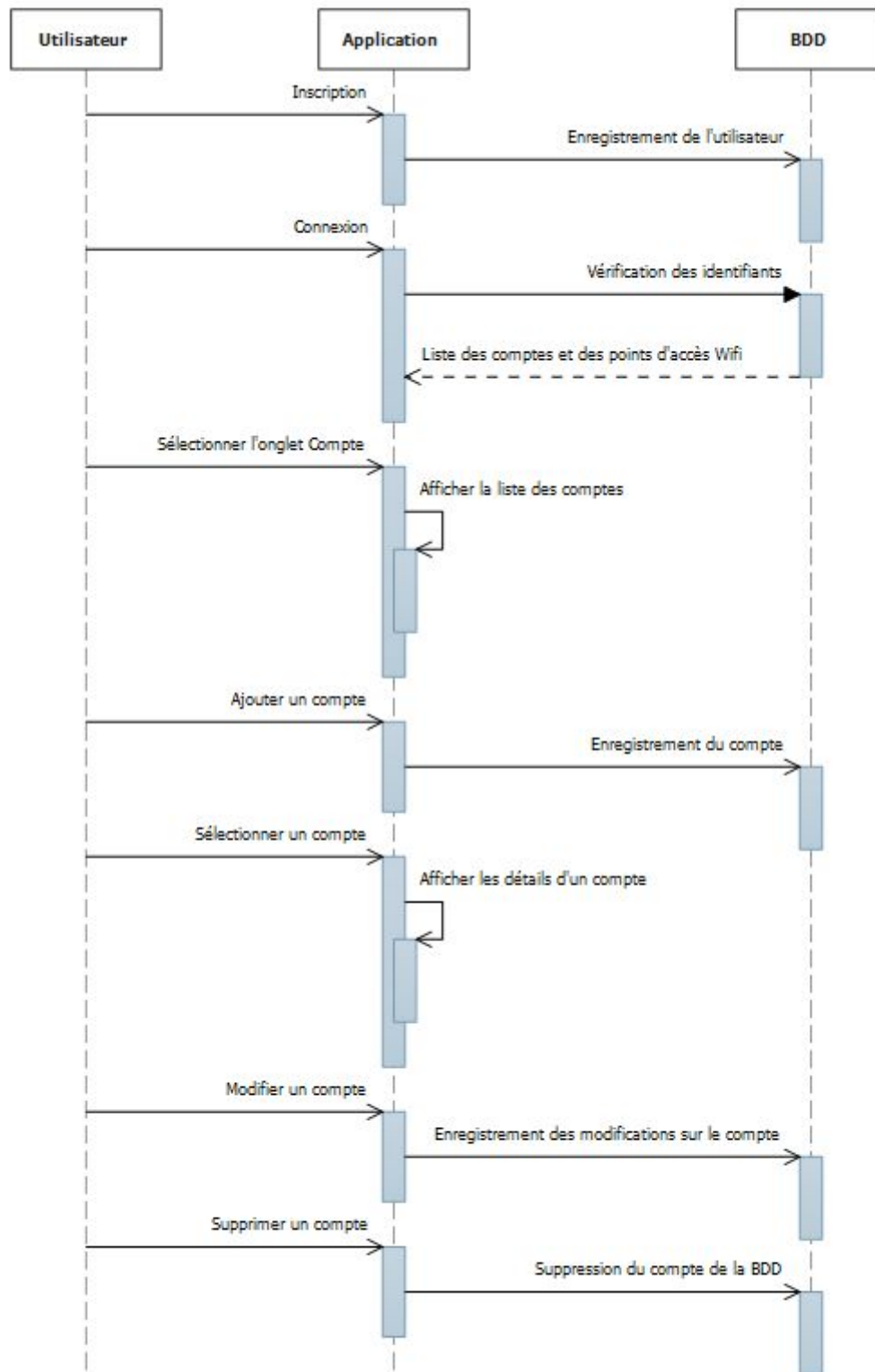


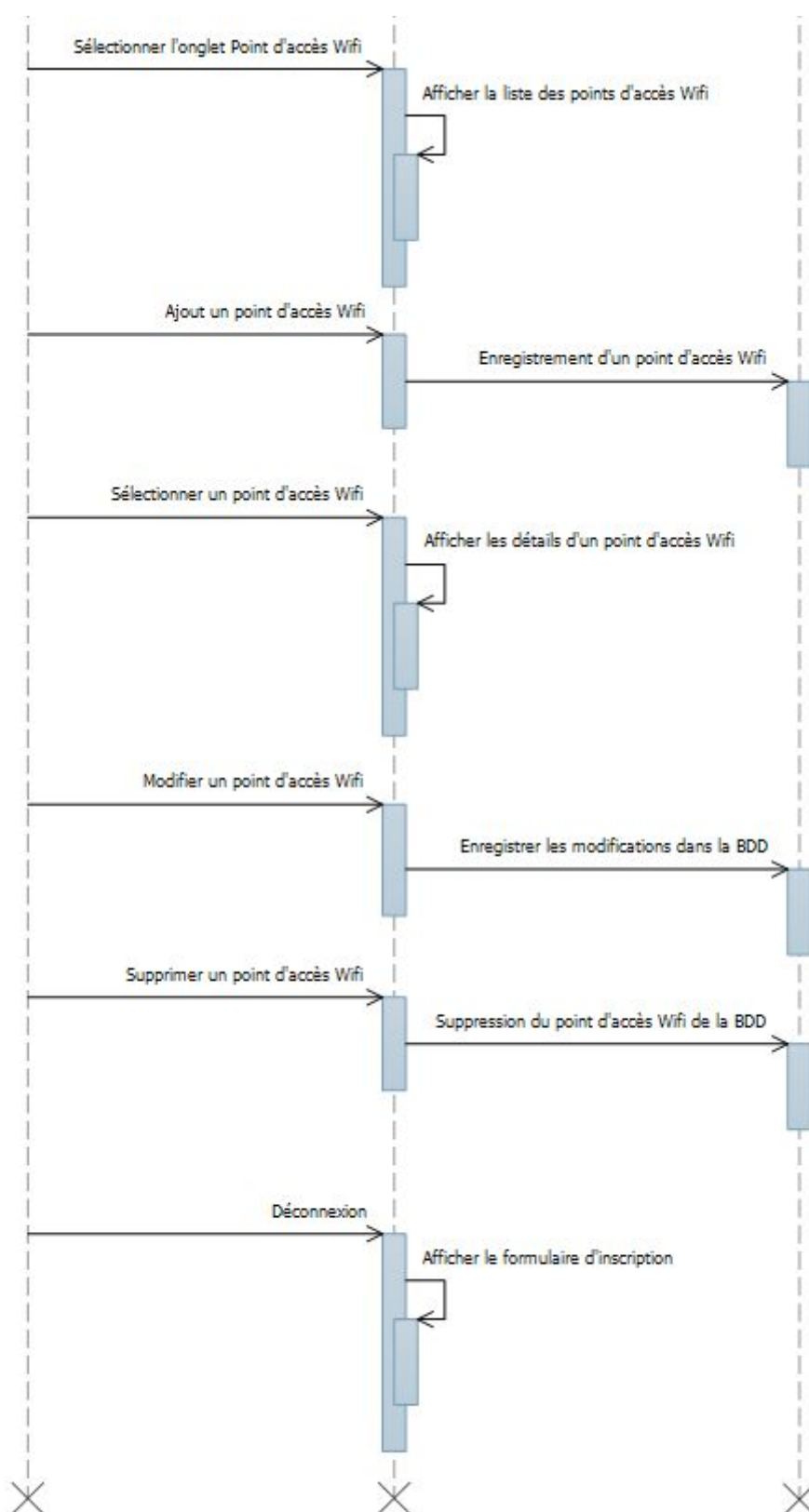
App



4. Diagramme de séquences

Le diagramme de séquences est deux parties : sur cette page et la suivante.





5. Architecture de l'application

L'application applique une architecture MVC construite de la manière suivante :

- **App** : contient la partie graphique, les formulaires
- **DAL** : contient la partie communication avec la base de données
- **Domain** : contient les classes métiers ainsi que les fichiers de mapping avec la BDD
- **Test** : contient le fichier de test

App fait appel à DAL et Domain pour communiquer avec la BDD et avoir accès aux classes métiers tandis que DAL fait uniquement appel à Domain pour les classes métiers.

6. Planning et répartition des tâches

Légende du planning de Gantt :



Jalons : début et fin du projet



Jalons : séances de TP



Phases du projet : architecture, code (forms, tests)



Tâches réalisées par Valentin



Tâches réalisées par Marine



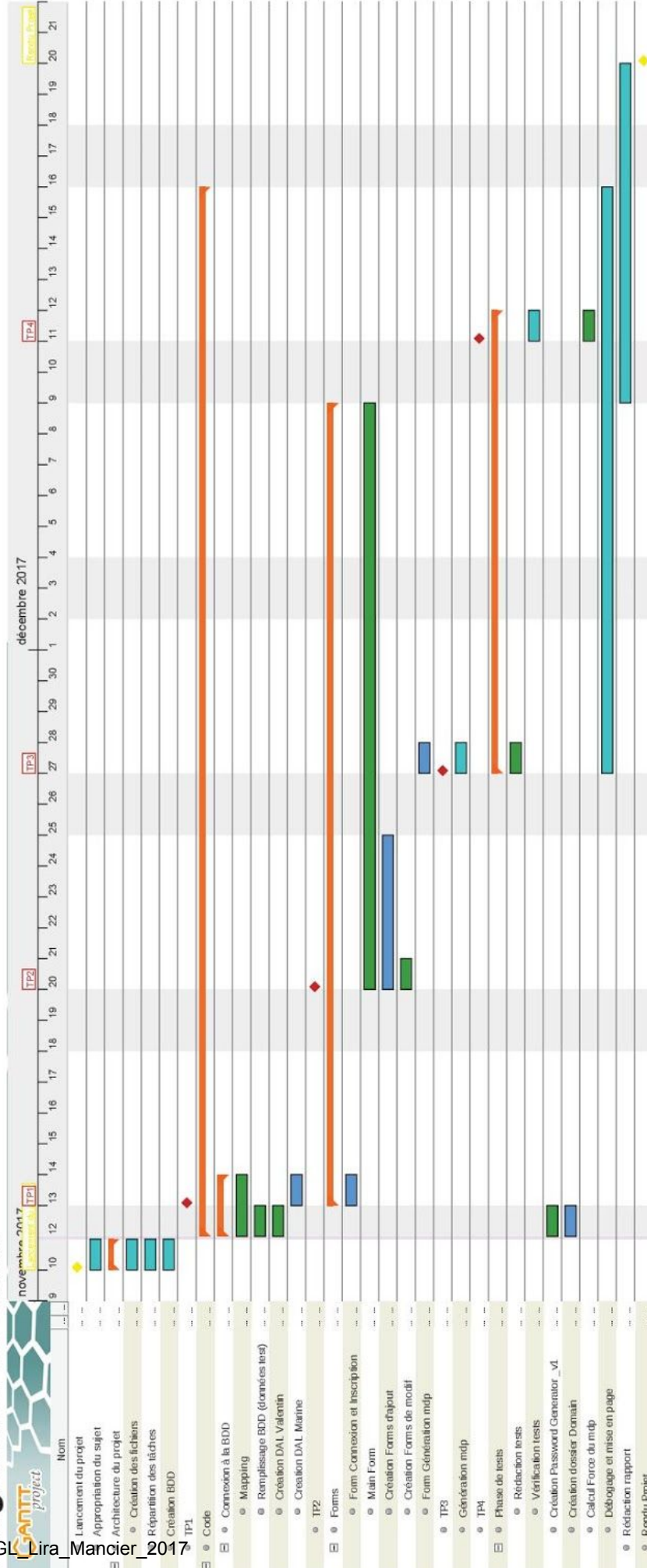
Tâches réalisées par Valentin et Marine

Projet_Genie_Log

20 déc. 2017

5

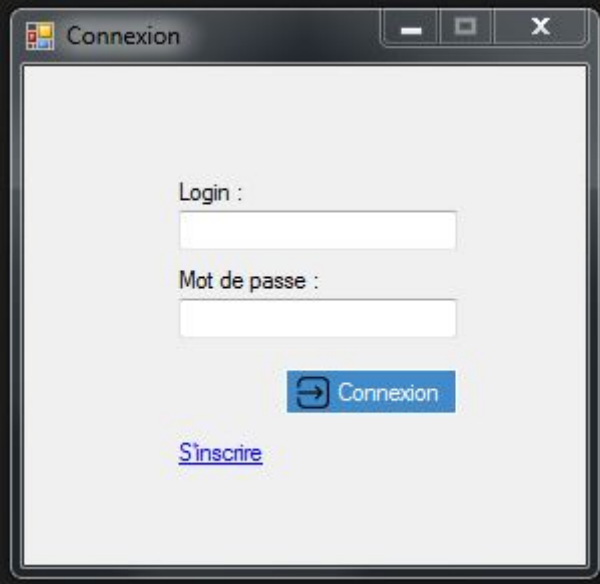
Diagramme de Gantt



II - Spécifications détaillées

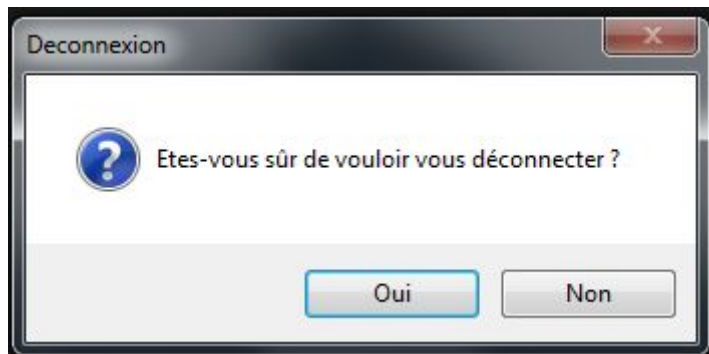
1. Formulaire

Connexion



The screenshot shows a window titled "Connexion" with a standard Windows-style title bar (minimize, maximize, close buttons). Inside the window, there are two text input fields. The first is labeled "Login :" and the second is labeled "Mot de passe :". Below these fields is a blue button with a right-pointing arrow icon and the text "Connexion". Below the button is a blue, underlined link that says "S'inscrire".

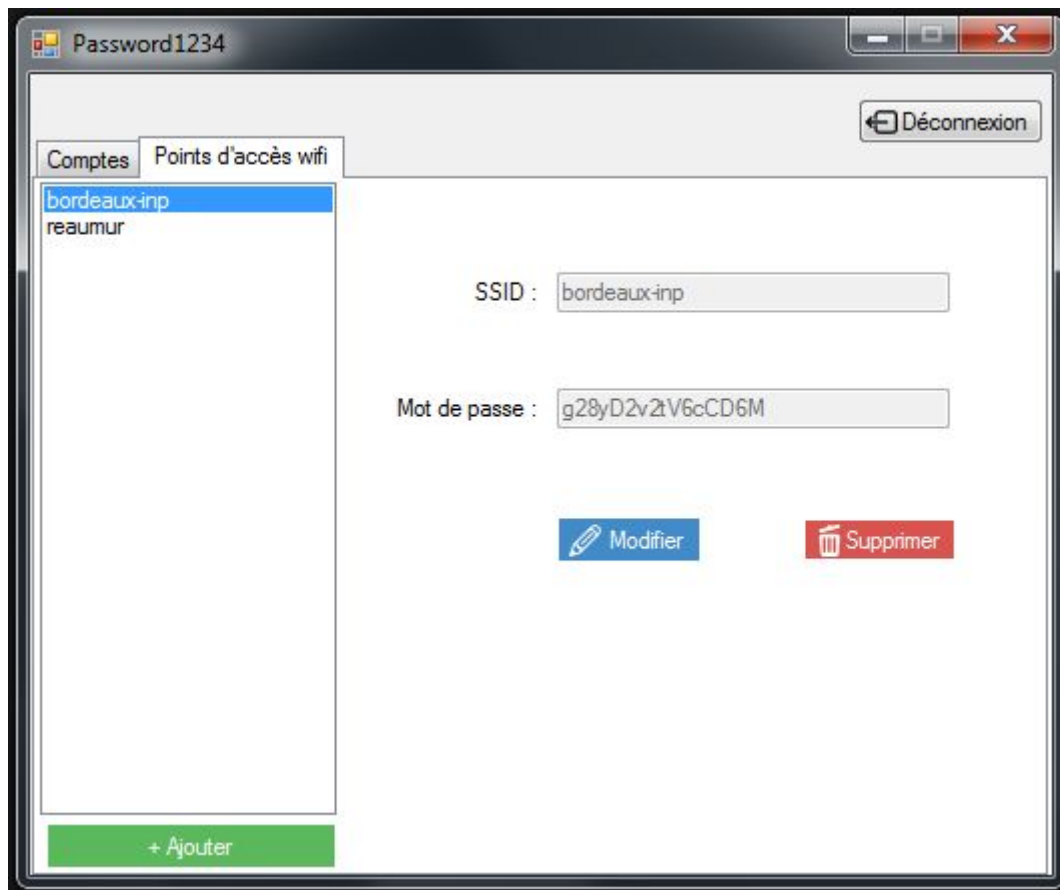
Lorsque l'utilisateur veut se déconnecter, un message de demande de confirmation apparaît.



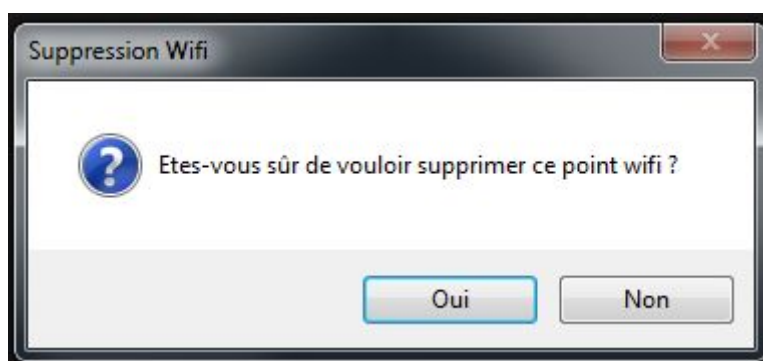
Inscription

Formulaire principal

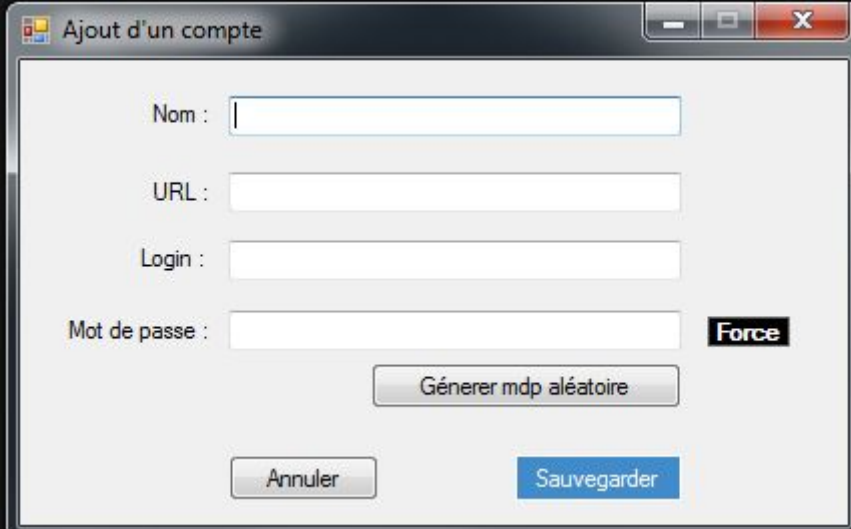
Lorsqu'on sélectionne l'onglet "Points d'accès wifi" :



Lorsque l'on clique sur un point d'accès wifi puis sur "supprimer", un message de demande de confirmation apparaît :



Ajout d'un compte



Ajout d'un compte

Nom :

URL :

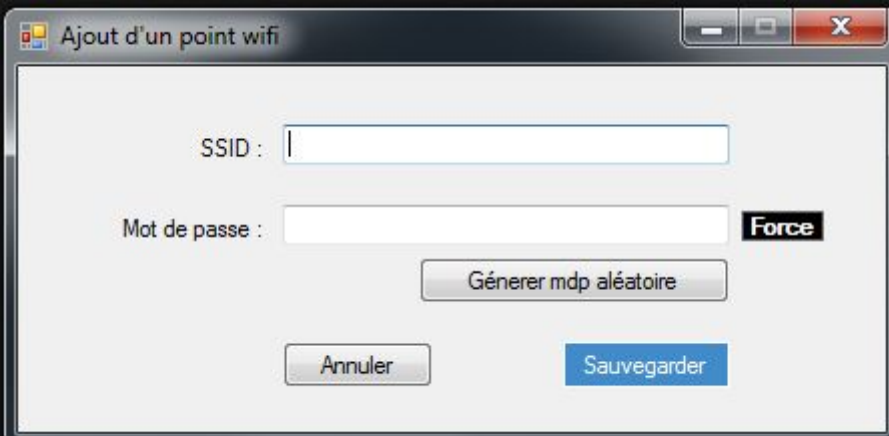
Login :

Mot de passe : **Force**

Générer mdp aléatoire

Annuler Sauvegarder

Ajout point wifi



Ajout d'un point wifi

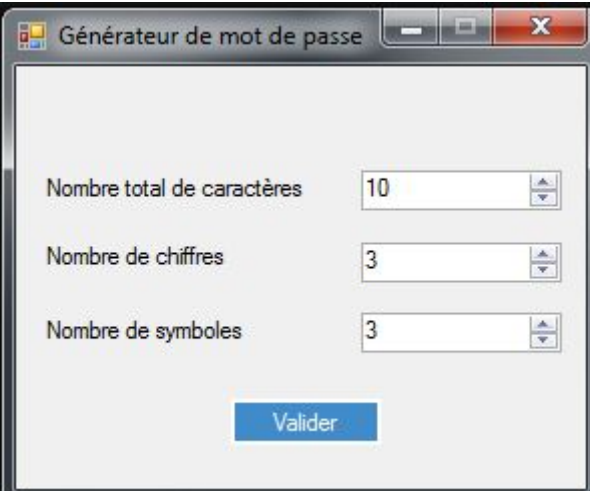
SSID :

Mot de passe : **Force**

Générer mdp aléatoire

Annuler Sauvegarder

Génération de mot de passe aléatoire



Générateur de mot de passe

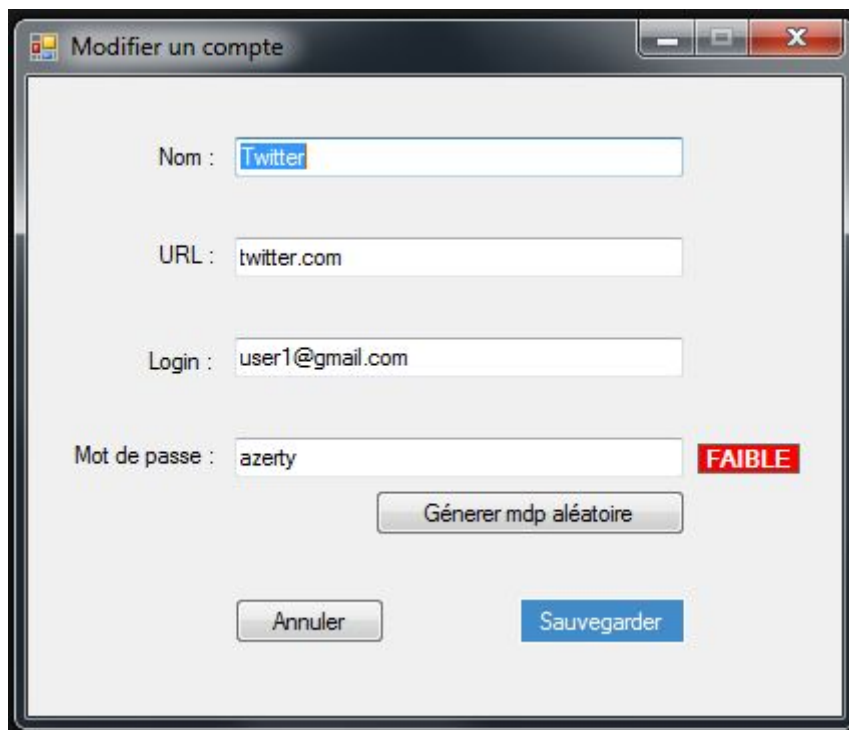
Nombre total de caractères

Nombre de chiffres

Nombre de symboles

Valider

Mise à jour compte



Modifier un compte

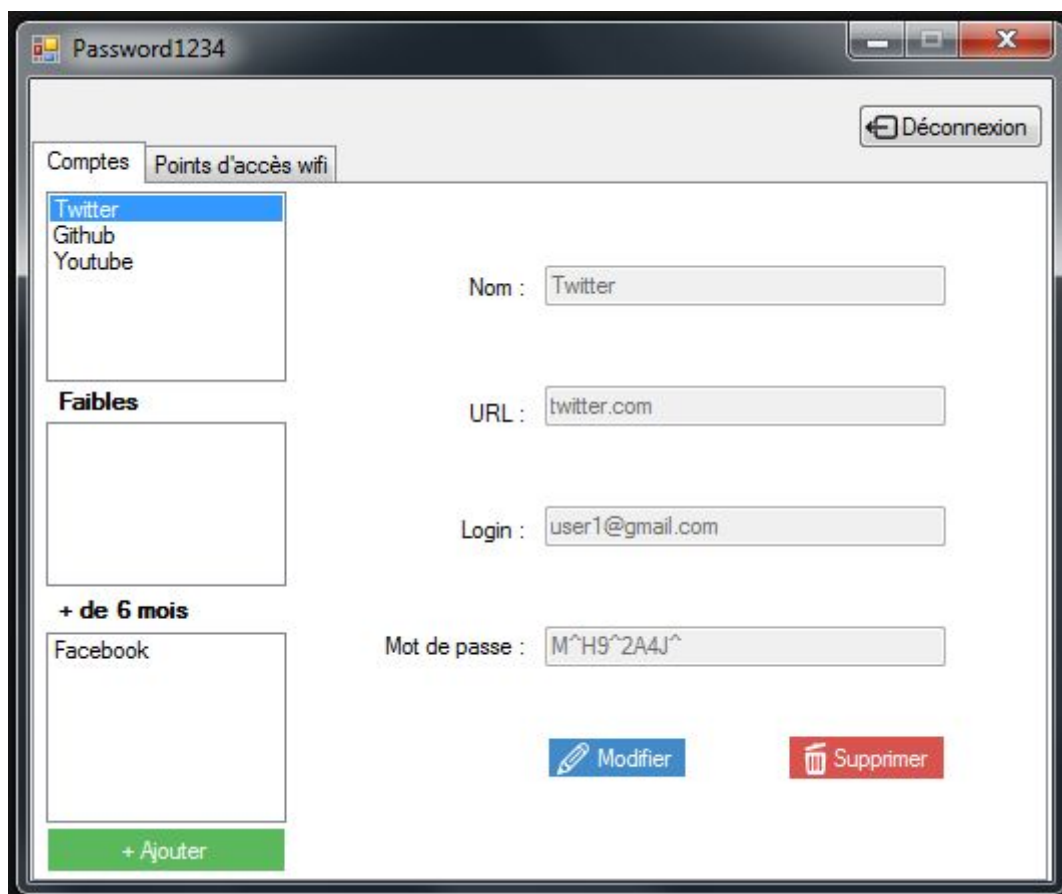
Nom :

URL :

Login :

Mot de passe : **FAIBLE**

Après modification du mot de passe associé au compte Twitter, on remarque que la modification a été prise en compte :



Password1234

Comptes Points d'accès wifi

Twitter
Github
Youtube

Faibles

+ de 6 mois

Facebook

+ Ajouter

Nom :

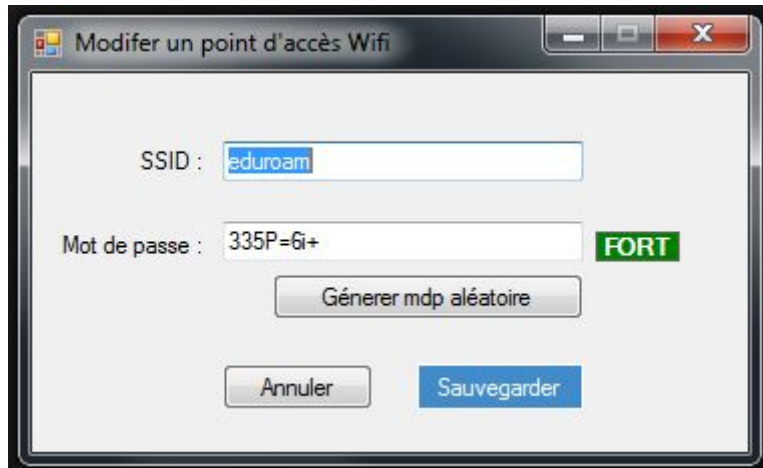
URL :

Login :

Mot de passe :

Le compte Twitter n'apparaît plus dans les mots de passe faibles.

Mise à jour point wifi



2. Choix de conception

Pour faire le lien entre la base de données et les objets manipulés par l'application, nous avons choisi de faire appel à l'outil d'ORM NHibernate auquel on va fournir un fichier de Mapping objet avec les correspondances entre les classes et les tables de la BDD. Cela permet d'améliorer les performances et de diminuer le volume de code.

Le paradigme du Chargement tardif est utilisé dans la classe Repository pour éviter les initialisations inutiles coûteuses en temps d'exécution et en bande passante, cela améliore les performances globales de l'application. On retarde l'accès à la base de données jusqu'au moment de sa première utilisation.

Le paradigme de l'Injection de dépendance utilisé dans les classes AccountRepository, UserRepository et WifiRepository permet quant à lui de limiter le couplage entre les classes au sein du projet non plus en déclarant les dépendances entre les classes mais en les injectant dynamiquement. De plus, nous avons utilisé des interfaces à la place des instances de classe.

Le paradigme du Singleton a été mis en place pour la classe PasswordGenerator qui est chargée de générer un mot de passe aléatoire et de calculer la force d'un mot de passe. Ce choix est motivé par plusieurs raisons : nous avons besoin de cette classe dans plusieurs formulaires mais une seule instance est nécessaire et cela pour la totalité du projet.

3. Calcul de la force du mot de passe

Dans un premier temps, on commence par parcourir le mot de passe pour récupérer les informations suivantes :

- le nombre de lettres majuscules
- le nombre de lettres minuscules
- le nombre de chiffres
- le nombre de symboles
- le nombre de lettres majuscules consécutives
- le nombre de lettres minuscules consécutives
- le nombre de chiffres consécutifs

Dans un second temps, on calcule un score :

- + 4 points par caractère
- + 2 points par caractère qui n'est pas une lettre minuscule
- + 2 points par caractère qui n'est pas une lettre majuscule
- + 4 points par chiffre
- + 6 points par symbole
- - 2 points par lettres minuscules consécutives
- - 2 points par lettres majuscules consécutives
- - 2 points par chiffres consécutifs
- - 1 point par caractère s'il ne contient que des nombres
- - 1 point par caractère s'il ne contient que des lettres

On attribue ensuite 2 points supplémentaire à chaque fois que le mot de passe remplit une des conditions suivantes :

- Avoir au moins 8 caractères
- Avoir au moins une lettre majuscule
- Avoir au moins une lettre minuscule
- Avoir au moins un chiffre
- Avoir au moins un symbole

Enfin, nous avons défini arbitrairement 3 niveaux de force de mot de passe :

- Faible pour un score inférieur 50
- Moyen pour un score entre 50 et 80
- Fort pour un score supérieur à 80

III - Résultats et tests

1. Exigences métiers respectées

Code	Description	Etat
EF_01	L'utilisation de l'application est soumise à la saisie correcte d'un mot de passe principal	Respectée
EF_02	Une fois authentifié, l'utilisateur a accès à la liste de ses différents comptes	Respectée
EF_03	En sélectionnant le titre d'un compte dans la liste, les détails sur ce compte sont affichés	Respectée
EF_04	Les informations obligatoirement associées à un compte sont : titre, login, mot de passe	Respectée
EF_05	La force du mot de passe associé à un compte est évaluée, affichée, idéalement sous forme graphique	Respectée
EF_06	L'application permet d'ajouter un nouveau compte en saisissant les informations	Respectée
EF_07	L'application permet de modifier les informations sur un compte	Respectée
EF_08	L'application permet, après confirmation, de supprimer un compte	Respectée
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	Respectée
EF_10	L'application permet également de gérer la liste des points d'accès wifi de l'utilisateur (informations : SSID et mot de passe)	Respectée
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	Respectée
EF_12	L'application gère la date de dernière modification d'un compte. Cette date est calculée automatiquement	Respectée
EF_13	L'application affiche séparément la liste des comptes ayant un mot de passe faible	Respectée
EF_14	L'application affiche séparément la liste des comptes ayant un mot de passe non modifié depuis plus de 6 mois	Respectée

2. Protocoles de tests et résultats

Les tests exécutés sont des tests fonctionnels, regroupés dans un fichier du projet Test.

Ce programme nous permet de :

1. récupérer la liste des utilisateurs
2. tester la connexion à la base de données d'un utilisateur test

Ces deux premiers tests permettent de tester la connexion à la base de données.

3. récupérer la liste des comptes pour lesquels le mot de passe est faible
4. récupérer la liste des comptes n'ayant pas été modifiés depuis plus de 6 mois
5. récupérer la liste des comptes n'appartenant à aucun des cas précédents (mots de passe moyens et forts, et comptes modifiés il y a moins de 6 mois)
6. récupérer les points d'accès Wifi

Ces tests permettent de gérer l'affichage du formulaire principal.

7. ajouter un nouvel utilisateur

Ce test nous permet de vérifier le formulaire d'inscription.

8. ajouter un nouveau compte
9. ajouter un nouveau point d'accès wifi

Ces tests nous permettent de vérifier les formulaires d'ajout de compte et de point d'accès wifi.

10. modifier un utilisateur (modification du mot de passe)
11. modifier un compte (modification du mot de passe)
12. modifier un point d'accès wifi (modification du mot de passe)

Le test de modification de l'utilisateur ne nous sert pas pour notre application, puisque nous ne proposons pas cette fonctionnalité. Les tests de modification de comptes et de point d'accès wifi nous servent à valider le fonctionnement des formulaires correspondants.

13. supprimer un utilisateur
14. supprimer un compte
15. supprimer un point d'accès wifi

Le test de suppression de l'utilisateur ne nous sert pas pour notre application, puisque nous ne proposons pas cette fonctionnalité. Les tests de suppression de

comptes et de point d'accès wifi nous servent à valider le fonctionnement des formulaires correspondants.

16. générer un mot de passe

Ce dernier test permet de vérifier le fonctionnement de la fonction de génération d'un mot de passe aléatoire, et donc le fonctionnement du formulaire de génération d'un mot de passe.

La réussite de ces tests est vérifiée grâce à un affichage console, qui indique si le test a réussi ou a échoué.

IV - Bilan et perspectives

Notre application vérifie toutes les exigences du projet et présente une interface simple et épurée, répondant au projet proposé.

Cependant, elle peut encore être améliorée, en effet, certaines fonctionnalités peuvent être ajoutées. Par exemple, nous ne vérifions pas lors de l'inscription si deux utilisateurs ont le même identifiant. Cette fonctionnalité n'est pas nécessaire (puisque le mot de passe permet d'identifier l'utilisateur) mais elle pourrait être proposée pour plus de confort à l'usage.

Pour permettre plus de confort lors de l'utilisation de notre application, il était possible par ailleurs de proposer une fonctionnalité "mot de passe oublié", qui contacterait l'utilisateur, sur une boîte mail qu'il aurait préalablement renseignée par exemple, afin de redéfinir un nouveau mot de passe.

Nous aurions également pu ajouter une fonctionnalité de gestion des utilisateurs, c'est à dire afficher tous les utilisateurs enregistrés lors de la connexion pour accélérer la connexion, permettre la modification des utilisateurs (login ou mot de passe), la suppression d'un utilisateur, etc.

La mise en place de blocs try/catch dans des portions de code qui peuvent être critiques aurait permis une meilleure gestion d'erreurs, notamment lors de la connexion à la base de données, sans laquelle le projet ne peut fonctionner.

Enfin, pour la phase de tests, nous avons choisi d'effectuer des tests fonctionnels. Il aurait pu être intéressant d'effectuer en plus des tests unitaires, ainsi que des tests automatisés sur la vue pour tester l'interface.