PROJET GÉNIE LOGICIEL

Gestion d'une course pédestre

Haddad Rime & Rieunier Roman

Groupe 2

Table des matières

I. Spécifications générales

A. Analyse fonctionnelle

Toutes les fonctionnalités principales nous ont été donné en début de projet via Trello, l'outil de gestion de projet en ligne collaboratif.

Notre projet a pour objectif de permettre la gestion de courses pédestres, et donc, d'assister les organisateurs durant les phases d'inscriptions des participants tout en permettant la saisie puis l'analyse des résultats obtenus.

Pour ce faire, différentes fonctions ont été créé afin d'atteindre nos objectifs.

Tout d'abord, nous avons permis à l'utilisateur de créer son compte, via un identifiant (son adresse mail) et un mot de passe. Concernant le mot de passe, celui ci est sécurisé (cf partie sur le mot de passe).

Une fois connecté, l'utilisateur de l'application peut alors afficher les informations relatives aux participants sur un tableau. Une fois la course terminée, l'utilisateur peut avoir accès au classement global de la course, sous forme tabulaire.

Notre application devant permettre à l'utilisateur de gérer une course, il était nécessaire de pouvoir lui laisser la possibilité d'inscrire les joueurs au sein d'une course. C'est pourquoi notre application permet d'importer un ou plusieurs participants à partir d'informations fournies dans un fichier CSV.

De même en ce qui concerne les résultats de la course, pouvant être importés sur l'application via un fichier CSV. Toutes ces fonctionnalités font parties des fonctionnalités principales définies sur Trello.

Ensuite, nous avons rendu possible la recherche du résultat d'un coureur à partir de son nom ou de son numéro de dossard.

Sachant qu'un utilisateur peut gérer plusieurs courses, il nous fallait rendre possible le choix de la gestion d'une course en particulier pour l'utilisateur.

Enfin, pour les dernières fonctionnalités, on peut constater que notre application rend possible l'affichage du classement pour une certaine tranche d'âge (de 10 ans choisie), tout en donnant à l'utilisateur la possibilité de s'authentifier et d'apporter des modifications concernant les données de l'application, telle que la suppression ou l'édition.

B. Analyse des besoins pour l'interface

Lors de ce projet, un de nos objectifs était l'utilisabilité de notre application, qui se voulait la plus ergonomique et accessible possible. Ce projet étant réalisé sous Windows via la technologie WinForms, technologie n'étant pas la plus optimisée en ce qui concerne l'ergonomie (car manque d'outils de personnalisation), il s'agissait alors de prendre en compte du mieux que possible l'utilisateur, et donc la facilité d'utilisation de notre application.

Ainsi, afin que l'utilisateur puisse utiliser au mieux notre application, nous avons fait en sorte que nos boutons et nos boîtes de saisies soient les plus parlantes pour l'utilisateur, afin qu'il puisse en comprendre aisément l'utilité.

L'application a été conçue de sorte à rendre la navigabilité la plus intuitive possible pour l'utilisateur, en limitant le nombre de page, mais également en proposant de nombreux feedbacks à celui-ci afin qu'il puisse à chaque fois prendre connaissance de ses actions.

D'un point de vue visuel, nous avons conçu les fenêtres de rapport 16:9, de couleur grises, avec des sections différentes pour chaque élément.

Notre page d'accueil se veut simple, avec peu d'informations pour ne pas perdre l'utilisateur. Celui-ci a d'ailleurs la possibilité d'augmenter la taille d'une fenêtre et de la fermer grâce aux icônes situées en haut à droite de chaque fenêtre.

Chaque bouton "actif" voit ses bordures s'éclairer en bleu, afin de comprendre que celui-ci est bien sélectionné.

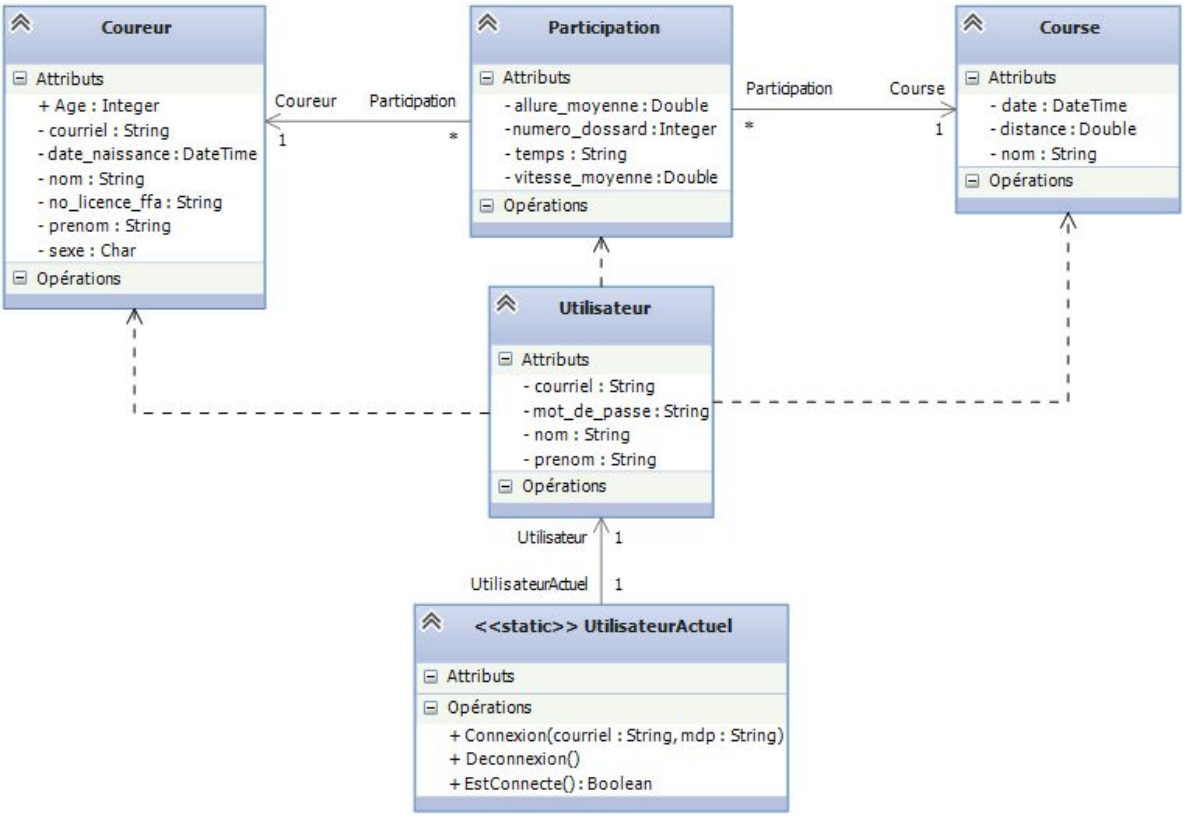
Pour faciliter les recherches, nous avons créé des barres de recherches, permettant par exemple de retrouver un participant en fonction de la course à laquelle il appartient, ou encore de retrouver une course en fonction de sa date. Les barres de recherches rendent l'application plus ergonomique et facilite donc l'utilisation générale de notre outil de gestion de course.

Lorsqu'une fonctionnalité ne peut être utilisée, elle est affichée en gris clair.

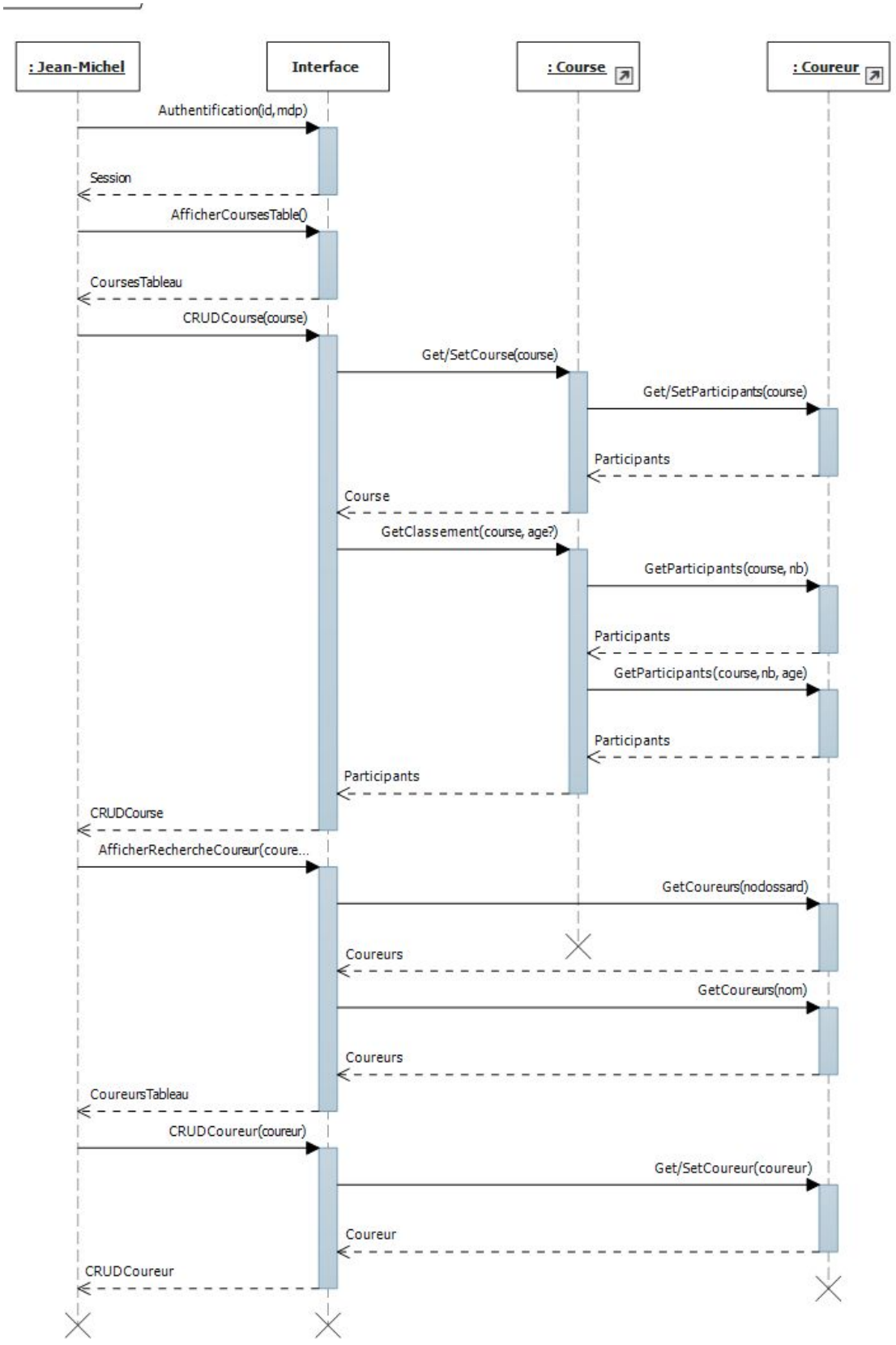
Enfin, nous avons mis en place des scrollbar afin de permettre à l'utilisateur d'accéder à toutes les informations du coureur par exemple, tout en évitant de surcharger la fenêtre de données, permettant ainsi d'aérer notre application et de rendre la lecture de l'information agréable.

C. Schémas UML

1. Diagramme de classes



2. Diagramme de séquences



D. Description de l'architecture

Nous avons opté pour une architecture en couches pour ce projet plutôt qu'un Model View Presenter. Notre solution se découpe en 3 couches :

- **DAL (Data Access Layer) :** Ce projet contient
    - les repositories, des classes permettant de manipuler les données issues des classes métiers, ils sont aussi responsables de la persistance dans la base de données.
- **Domain :** Ce projet contient
    - les fichiers de mapping en .hbm.xml, qui permettent de relier les classes métiers du DAL et leurs attributs aux tables et leurs colonnes dans la base de données.
    - les classes métier, les surcouches en POO qui représentent les tables contenues dans notre base de données ;
- **App :** Ce projet contient
    - les formulaires permettant à l'utilisateur d'interagir avec le programme et les données ;
    - les composants "maisons", qui sont utilisés dans les formulaires et permettent d'améliorer l'expérience utilisateur ;
    - les classes d'extension, qui permettent de rajouter des méthodes aux types primitifs et ainsi de factoriser le code.

E. Gestion de projet

1. Répartition des tâches et planning

Durant ce projet, nous nous sommes servi de l'outil de gestion de projet en ligne Trello pour nous répartir les tâches principales à effectuer. La plateforme Github nous a permis de partager notre code, afin de suivre l'avancée de chacun durant le projet.

En début de projet nous avons travaillé en pair programming. Cependant, nous avons dû nous diviser pour avancer au plus vite sur ce projet et attribuer des tâches précises à chaque membre du groupe, tâches que l'on s'est réparties sur Trello. La répartition des tâches apparaît sur le GANTT mis en annexe.

II. Spécifications détaillées

A. Formulaires

- **Formulaire principal :** page d'accueil permettant d'accéder à l'authentification et à la gestion des courses et des coureurs ;
- **Formulaire d'authentification :** fenêtre modale permettant à l'utilisateur de se connecter afin d'accéder aux fonctionnalités d'édition de contenu ;
- **Formulaire d'édition de compte :** fenêtre modale permettant à l'utilisateur de modifier ses informations de connexion ;
- **Formulaire de gestion des courses :** page permettant d'accéder à la liste des courses et de leurs participants. La page permet aussi de créer, éditer supprimer et importer des courses ou des participants ;
    - **Formulaire de création de course :** fenêtre modale permettant de créer et de persister une course dans la base de données après validation ;
    - **Formulaire d'édition de course :** fenêtre modale permettant de modifier et de persister la course sélectionnée dans la base de données après validation ;
    - **Formulaire d'importation de courses :** fenêtre modale permettant de sélectionner un fichier CSV dans le disque dur de l'utilisateur et de persister la liste des courses générées dans la base de données après validation ;
    - **Formulaire de création de participant :** fenêtre modale permettant de créer et de persister un participant dans la base de données après validation ;
    - **Formulaire d'édition de participant :** fenêtre modale permettant de modifier et de persister le participant sélectionné dans la base de données après validation ;
    - **Formulaire d'importation de participants :** fenêtre modale permettant de sélectionner un fichier CSV dans le disque dur de l'utilisateur et de persister la liste des participants générés pour la course sélectionnée dans la base de données après validation ;

- **Formulaire de gestion des coureurs :** page permettant d'accéder à la liste des coureurs et de leurs participations. La page permet aussi de créer, éditer, supprimer et importer des coureurs ou des participants ;
    - **Formulaire de création de coureur :** fenêtre modale permettant de créer et de persister une course dans la base de données après validation ;
    - **Formulaire d'édition de coureur :** fenêtre modale permettant de modifier et de persister le coureur sélectionné dans la base de données après validation ;
    - **Formulaire d'importation de coureurs :** fenêtre modale permettant de sélectionner un fichier CSV dans le disque dur de l'utilisateur et de persister la liste des coureurs générés dans la base de données après validation ;
    - **Formulaire de création de participation :** fenêtre modale permettant de créer et de persister une participation dans la base de données après validation ;
    - **Formulaire d'édition de participation :** fenêtre modale permettant de modifier et de persister la participation sélectionnée dans la base de données après validation ;
    - **Formulaire d'importation de participations :** fenêtre modale permettant de sélectionner un fichier CSV dans le disque dur de l'utilisateur et de persister la liste des participations générées pour la course sélectionnée dans la base de données après validation ;

B. Justification des choix de conception et production

1. Conception

Concernant les choix effectués au niveau de la conception, nous avons opté pour un développement en couche. En effet, ce dernier permet de simplifier notre application en structurant au mieux notre code. Étant découpée en plusieurs couches, le code est alors divisé en une partie spécifique à l'interface, aux accès aux données et aux classes métiers. Pour lier notre base de données relationnelle aux classes métiers, nous avons utilisé l'ORM NHibernate, permet l'automatisation des liens entre la base de données et nos objets.

2. Production

Chaque attributs et méthodes privés dans les classes n'ont pas été écrits en **CamelCase** mais en **_snake_case** pour respecter les conventions générales de programmation.

YAGNI (You Ain't Gonna Need It)

Nous étions contre l'usage d'interfaces dans notre code mais finalement nous en avons créées pour les repositories. De ce fait nous avons pu stocker des IRepository dans certaines classes afin d'y instancier de vrais repositories ou des stubs suivant nos besoins. Finalement les interfaces peuvent être utiles, même dans un petit projet. Il faut aussi reconnaître que leur utilisation rend le code plus élégant.

DRY (Don't Repeat Yourself)

Nous avons décidé de créer un Form Component, la PlaceholderTextBox. Ce composant permet d'afficher un texte dans une TextBox qui disparaît quand l'utilisateur est sur le point de rentrer du contenu. Ce composant est particulièrement utile, voire indispensable, pour la bonne expérience utilisateur des entrées dans les barres de recherche. En effet, avec nos contraintes d'espace, nous ne pouvions pas nous permettre d'associer un Label à chacune des TextBox, d'où l'importance du placeholder. D'autre part, nous avons créer une surcouche de TextBox pour éviter les répétitions dans le code des gestionnaires d'événements communs à toutes les PlaceholderTextBox.

Le pattern Singleton

Notre programme comporte plusieurs classes implémentant le pattern Singleton.

- **Toutes les classes fournissant des données tests (Stub*Repository) :** Elles ne doivent être instanciées qu'une seule fois afin d'éviter une perte de données entre les différents Stubs à travers des copies ;

- **La classe UtilisateurCourant :** Cette classe n'est pas elle-même un singleton, mais c'est une classe statique qui contient une instance unique d'Utilisateur, ainsi que des méthodes permettant de le manipuler (connexion, déconnexion).

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

En ce qui concerne les modalités de calcul de la force du mot de passe, nous avons fait des recherches sur le site de l'agence nationale de la sécurité des systèmes d'informations(ANSSI).

Ce site nous explique que la “force de calcul” d’un mot de passe désigne sa capacité à résister à une énumération de tous les mots de passe possibles.

Cette « force » dépend de la longueur L du mot de passe et du nombre N de caractères possibles. Elle suppose que le mot de passe est choisi de façon aléatoire. Elle se calcule aisément par la formule N^L. Mais il est plus difficile d’estimer si la valeur ainsi obtenue est suffisante ou pas.

Ainsi, l’ANSSI propose les choix suivants concernant les mots de passe à choisir:

Type de mot de passe	Taille de clé équivalente	Force	Commentaire
Mot de passe de 8 caractères dans un alphabet de 70 symboles	49	Très faible	Taille usuelle
Mot de passe de 10 caractères dans un alphabet de 90 symboles	65	Faible	
Mot de passe de 12 caractères dans un alphabet de 90 symboles	78	Faible	Taille minimale recommandée par l’ANSSI pour des mots de passe ergonomiques ou utilisés de façon locale.
Mot de passe de 16 caractères dans un alphabet de 36 symboles	82	Moyen	Taille recommandée par l’ANSSI pour des mots de passe plus sûrs.
Mot de passe de 16 caractères dans un alphabet de 90 symboles	104	Fort	
Mot de passe de 20 caractères dans un alphabet de 90 symboles	130	Fort	Force équivalente à la plus petite taille de clé de l’algorithme de chiffrement standard AES (128 bits).

Sachant que notre application est une application de course pédestre, nous avons opté pour un mot de passe de l’ordre de 12 caractères. D’après l’ANSSI, 12 caractères semble correspondre à la taille minimum pour un mot de passe ergonomique et utilisé de façon locale, ce qui correspond plutôt bien aux attentes de notre application.

III. Résultats et tests

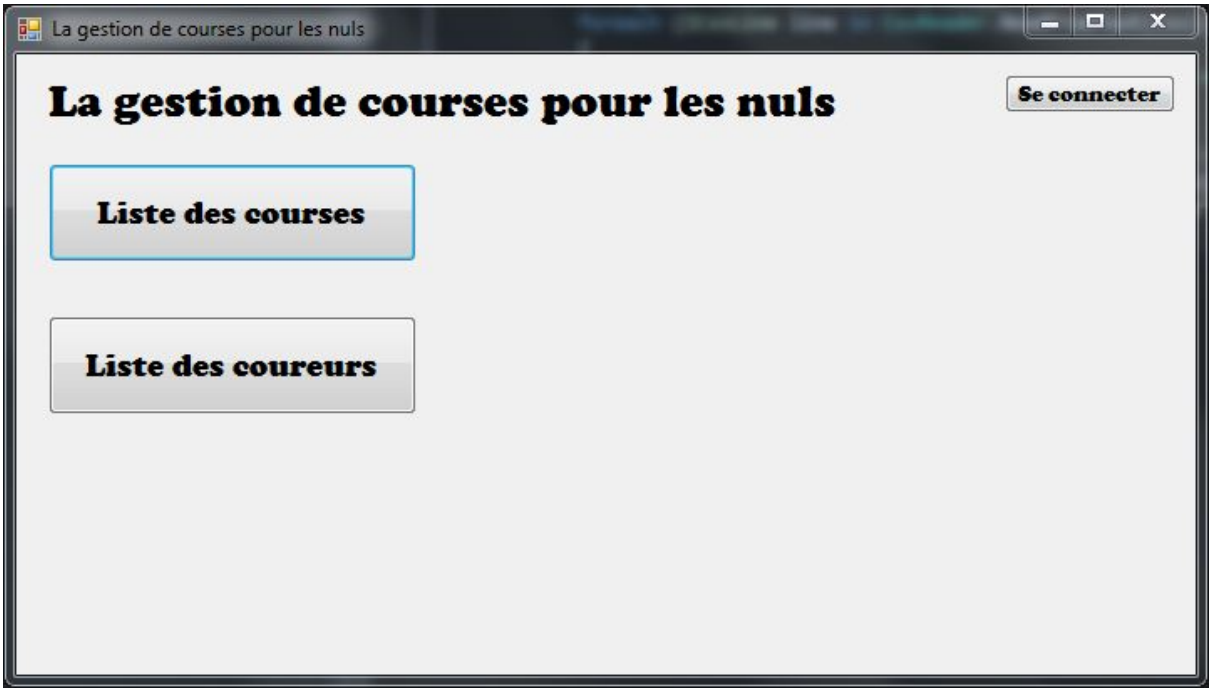
A. Exigences métiers respectées

Code	Description
EF_01	En tant qu'utilisateur, je peux afficher les informations sur les participants sous forme tabulaire
EF_02	En tant qu'utilisateur, je peux consulter le classement global de la course sous forme tabulaire

EF_03	En tant qu'utilisateur, je peux importer un ou plusieurs participants à partir des informations fournies dans un fichier CSV
EF_04	En tant qu'utilisateur, je peux importer les résultats de la course fournis dans un fichier CSV
EF_05	En tant qu'utilisateur, je peux accéder au résultat d'un coureur par une recherche à partir de son nom ou de son numéro de dossard
EF_06	L'application gère plusieurs courses. En tant qu'utilisateur, je peux choisir la course gérée
EF_07	En tant qu'utilisateur, je peux afficher le classement pour une tranche d'âge de 10 ans choisie
EF_08	En tant qu'utilisateur, je dois m'authentifier avant toute opération de modification des données de l'application (import, édition, suppression)

B. Démonstration du logiciel

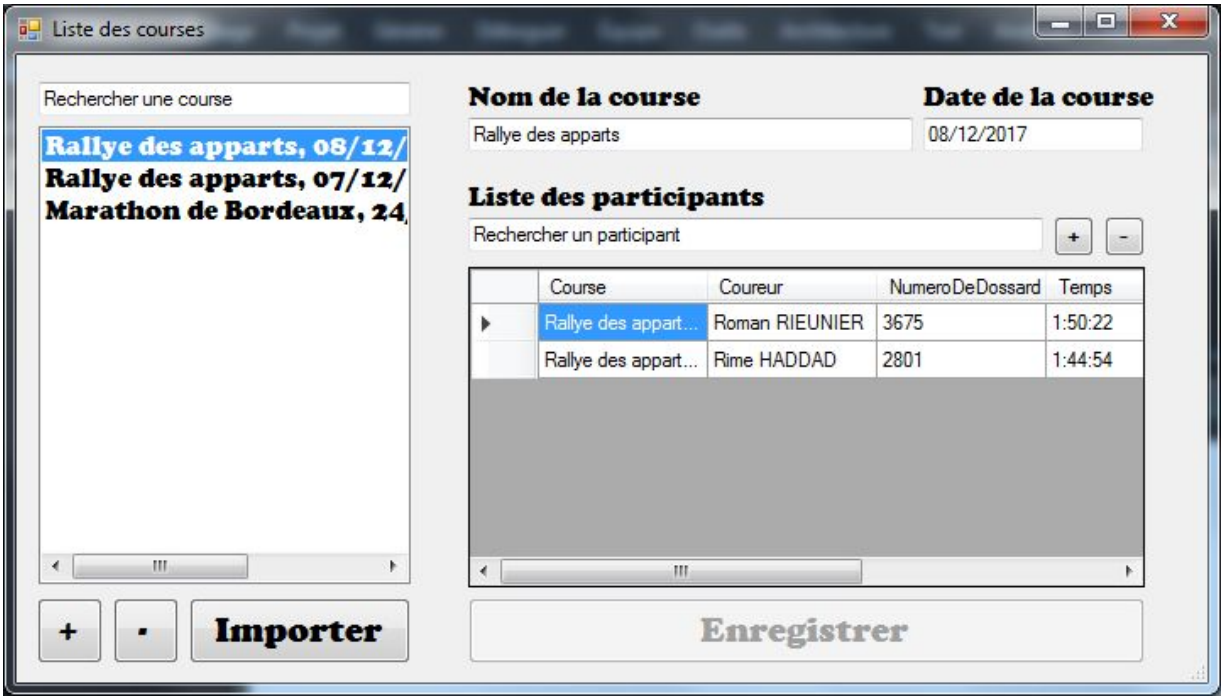
Voici la page d'accueil permettant à l'utilisateur de se connecter, d'accéder à la liste des courses et à la liste des coureurs. Les bordures du bouton cliqué deviennent bleues pour faciliter l'utilisation et la bonne compréhension de notre application à l'utilisateur.

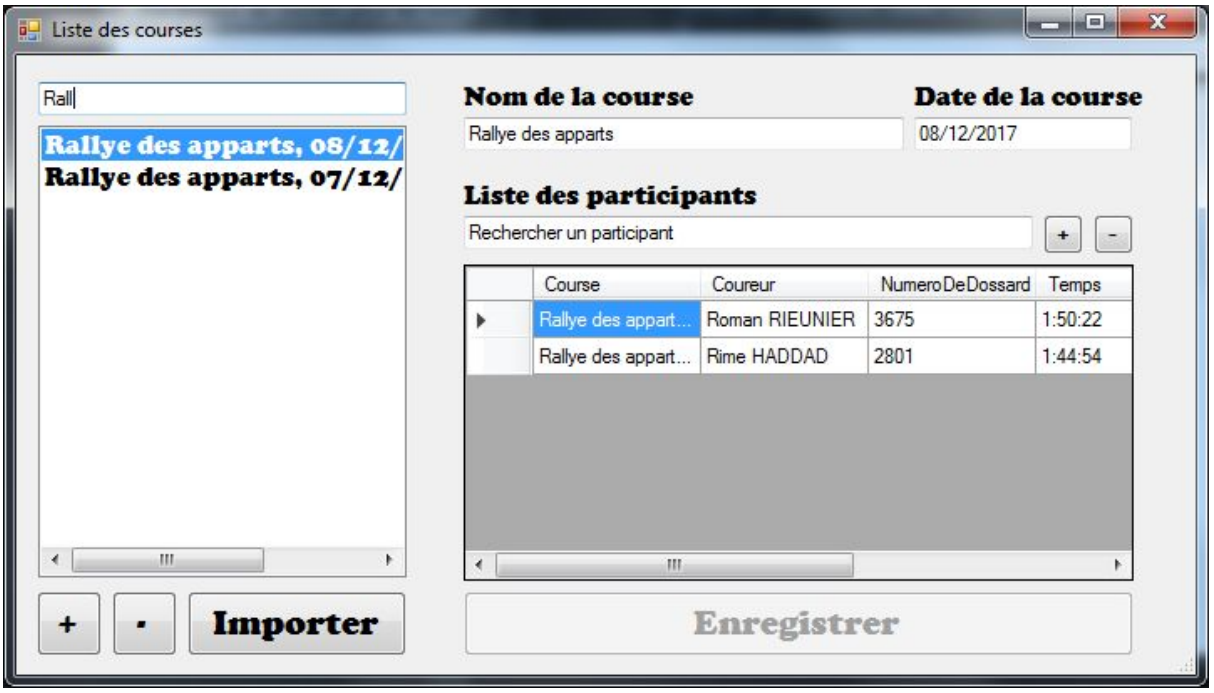


Une fois le bouton liste des courses cliqué, l'utilisateur peut accéder à différentes fonctionnalités de l'application. D'une part, il peut rechercher une course (à gauche) en tapant le nom de la course ou la date. L'utilisateur à la possibilité de glisser via une scrollbar. L'utilisateur peut aussi importer des participants à une course en utilisant le bouton importer. Si cette importation a réussie, une message box apparaît.

A droite, il est possible de rechercher un participant via ses différentes caractéristiques (telles que son numéro de dossard, son nom, la course à laquelle il appartient,...).

Les boutons plus et moins servent à ajouter et supprimer des participants. Par exemple, en cliquant sur le nom du coureur Roman Rieunier, celui-ci étant sélectionné, si l'on appuie sur le bouton -, alors ce coureur sera supprimé de la liste. De plus, on constate qu'à tout moment l'utilisateur peut agrandir la fenêtre ou la fermer (icônes en haut à droite de la fenêtre). La fenêtre est aussi appelée par sa fonction, ici "liste des courses").





Lorsque l'utilisateur souhaite importer des résultats, l'application lui demande de sélectionner un fichier csv. Une fois le fichier csv choisi, il apparaît alors à l'utilisateur ce qui lui permet ainsi de valider l'import.

C. Protocoles de test

Nous avons créer des stubs pour créer des formulaires sans avoir à mapper la base de données. De plus, nous avons effectué des tests de notre application afin de vérifier que notre application fonctionnait bien, comme nous pouvons le voir grâce aux captures ci-dessus.

IV. Bilan et perspectives

A. Points positifs et négatifs

Tout d'abord, nous sommes globalement satisfaits du rendu final de l'application : le logiciel fonctionne et toutes les fonctionnalités exigées ont été implémentées. Par ailleurs, son utilisation nous semble assez simple et intuitive. Du point de vue de la programmation, nous avons réussi à utiliser Nhibernate ce qui nous a permis de montrer nos compétences en mapping objet-relationnel dans le cadre d'une application C#.

Initialement, nous partions avec un a priori négatif de WinForms, technologie vue par un des membres du binôme lors de son cursus précédent. En effet cette technologie est réputée pour être peu modulable et manque de possibilités en terme de personnalisation de l'apparence. Nous avons donc été confrontés aux limites que supposait la technologie Winform en terme de fonctionnalités graphiques proposées. Cependant nous avons eu d'agréables surprises en explorant les formulaires et le framework .NET, et avons pu recréer des composants présents nativement dans d'autres technologies.

Enfin, étant équipés de MacBooks, nous n'avions d'autre choix que de se retrouver au 109 avenue Roul, pour profiter des capacités des supercalculateurs fournis par l'école, qui sont eux, compatibles avec WinForms. Nous avons fait le choix de ne pas installer de machine virtuelle sur nos ordinateurs pour des raisons relativement évidentes.

B. Évolutions possibles

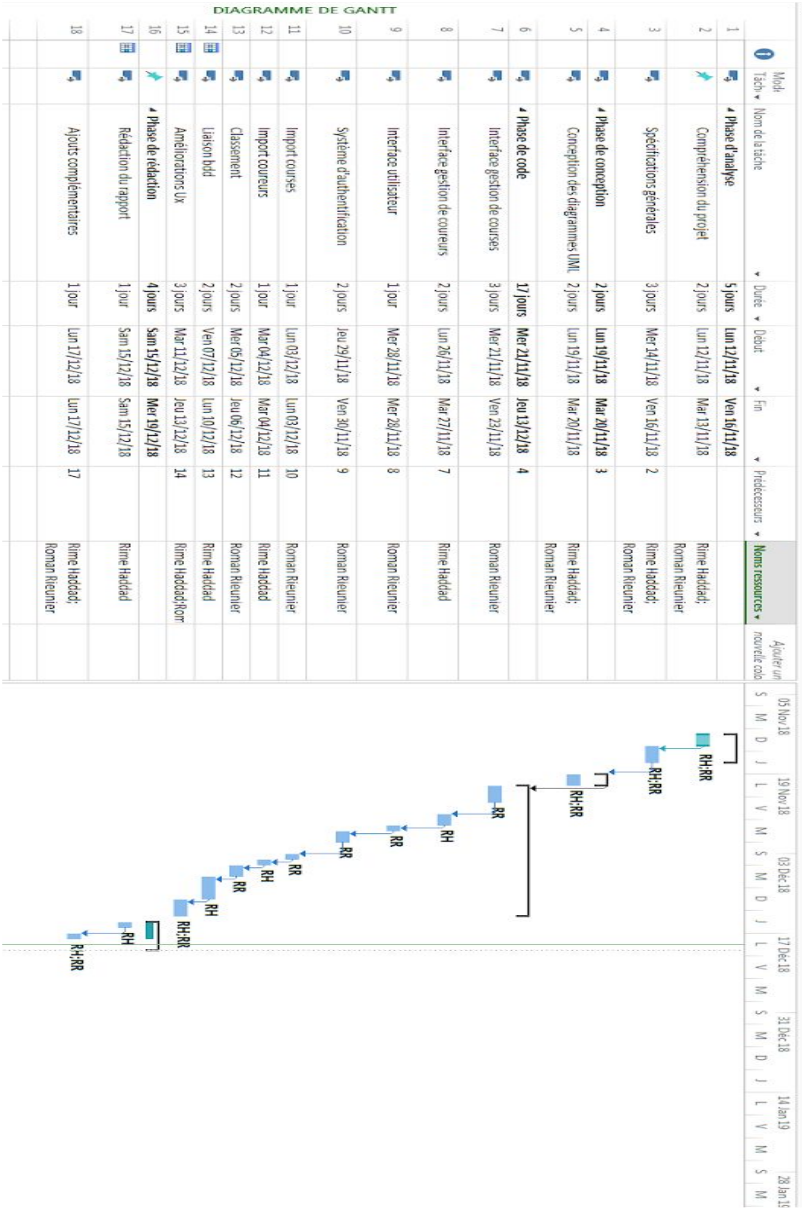
Il aurait été intéressant de pousser jusqu'au bout les fonctionnalités de WinForms. Des boutons tels que Annuler, Refaire, sur des actions critiques comme la suppression d'une course améliorerait sensiblement l'expérience utilisateur ainsi que notre connaissance de l'environnement .NET.

Nous pourrions de plus mettre en place une validation plus interactive des formulaires, avec notamment des ToolTips indiquant où sont les erreurs dans un formulaire, plutôt que de désactiver le bouton d'enregistrement tant que les contrôles ne sont pas valides.

D'autres possibilités d'import seraient souhaitables, en XML et JSON par exemple.

ANNEXES

ANNEXE 1 : GANTT



Annexe 2 : Bibliographie

- Pour le calcul de la force du mot de passe

https://www.ssi.gouv.fr/administration/precautions-elementaires/calculer-la-force-dun-mot-de-passe/