|  |
| --- |
| MessagingApp |

**Projet :** Projet volontaire

**Auteur :** Oliveira Ramos Dylan

**Ecole :** Centre professionnel du Nord vaudois - filière informatique

**Période :** Du 12 mai au 5 juin 2020

Table des matières

[Glossaire 3](#_Toc42158481)

[Analyse préliminaire 4](#_Toc42158482)

[Introduction 4](#_Toc42158483)

[Objectifs 4](#_Toc42158484)

[Planification initiale 5](#_Toc42158485)

[Analyse / Conception 6](#_Toc42158486)

[Concept 6](#_Toc42158487)

[Maquettes graphiques de la partie client 6](#_Toc42158488)

[Maquette graphique de la partie serveur 9](#_Toc42158489)

[Cas d’utilisation 10](#_Toc42158490)

[Partie client 10](#_Toc42158491)

[Partie serveur 12](#_Toc42158492)

[Analyse concurrentielle 13](#_Toc42158493)

[Partie client 13](#_Toc42158494)

[Partie Serveur 14](#_Toc42158495)

[Base de données 14](#_Toc42158496)

[Modèle conceptuel des données 14](#_Toc42158497)

[Modèle logique des données 15](#_Toc42158498)

[Stratégie de test 15](#_Toc42158499)

[Risques techniques 15](#_Toc42158500)

[Planification détaillée 16](#_Toc42158501)

[Dossier de conception 16](#_Toc42158502)

[Outils 16](#_Toc42158503)

[Réalisation 17](#_Toc42158504)

[Modifications sur la partie graphique 17](#_Toc42158505)

[Partie client 17](#_Toc42158506)

[Partie serveur 20](#_Toc42158507)

[Modifications sur la base de données 21](#_Toc42158508)

[Avant 21](#_Toc42158509)

[Après 21](#_Toc42158510)

[Icônes utilisées 21](#_Toc42158511)

[Librairies utilisées 22](#_Toc42158512)

[SQLite 22](#_Toc42158513)

[Les sockets 23](#_Toc42158514)

[La cryptographie 23](#_Toc42158515)

[Description des tests effectués 24](#_Toc42158516)

[Testeurs 24](#_Toc42158517)

[Création d’un compte 24](#_Toc42158518)

[Connexion à un compte 26](#_Toc42158519)

[Utilisation du chat 27](#_Toc42158520)

[Utilisation du serveur 29](#_Toc42158521)

[Erreurs restantes 30](#_Toc42158522)

[Envoi de caractères spéciaux interdit 30](#_Toc42158523)

[Actualisation du chat et des contacts 30](#_Toc42158524)

[Problème d’affichage de certains caractères 30](#_Toc42158525)

[Conclusions 31](#_Toc42158526)

[Points positifs 31](#_Toc42158527)

[Projet 31](#_Toc42158528)

[Application 31](#_Toc42158529)

[Points négatifs 31](#_Toc42158530)

[Application 31](#_Toc42158531)

[Difficultés 31](#_Toc42158532)

[Suites possibles 31](#_Toc42158533)

[Annexes 32](#_Toc42158534)

[Journal de travail 32](#_Toc42158535)

[Manuel d'Utilisation 32](#_Toc42158536)

[Remerciements 32](#_Toc42158537)

[Table des illustrations 32](#_Toc42158538)

[Bibliographie 33](#_Toc42158539)

# Glossaire

**ASCII :** American Standard Code for Information Interchange, c’est un code américain normalisé pour l'échange d'informations.

**Cryptage :** Opération par laquelle un message est rendu inintelligible à quiconque ne possède pas la clé permettant de retrouver la forme initiale.

**Hachage :** Fonction particulière qui, à partir d'une donnée fournie en entrée, calcule une empreinte numérique servant à identifier rapidement la donnée initiale.

**IDE** : Integrated Development Environment, il s'agit de l'environnement de développement. C'est un ensemble d'outils aidant au développement de logiciels.

**JVM :** Java Virtual Machine, c’est un appareil informatique fictif qui exécute des programmes compilés sous forme de bytecode Java.

**LLVM :** Low Level Virtual Machine, c’est une infrastructure de compilateur conçue pour l’optimisation du code.

**Logs :** Désigne un fichier permettant de stocker un historique des évènements attachés à un processus.

**MCD :** Le modèle conceptuel de données est une description graphique pour représenter des modèles de données sous la forme de diagrammes contenant des entités et des associations.

**MLD :** Le modèle logique de données reprend le contenu du MCD, mais précise la volumétrie, la structure et l'organisation des données telles qu'elles pourront être implémentées.

**Open source :** L’accès au code source peut être redistribué au grand public.

**Sérialisation :** C’est le codage d'une information sous la forme d'une suite d'informations plus petites.

**Sockets :** Les sockets permettent à une ou plusieurs machines de communiquer avec un serveur à travers un réseau.

**Thread :** C’est un processus représentant l'exécution d'un ensemble d'instructions du langage machine d'un processeur.

**UTF-8 :** C’est un codage de caractères informatiques conçu pour coder l’ensemble des caractères du « répertoire universel de caractères codés ».

# Analyse préliminaire

## Introduction

Ce projet volontaire consiste à développer une application de messagerie instantanée réalisée par Dylan Oliveira Ramos pour le Centre professionnel du Nord vaudois de Sainte-Croix (filière informatique).

Etant donné la pandémie du COVID-19, le projet est entièrement réalisé à domicile. Le début de celui-ci est fixé au 12 mai 2020 et se termine le 5 juin 2020, sa durée est de 80 heures, soit 20 heures par semaine (4 semaines).

Le programme sera réalisé en C# sous forme de formulaires Windows en utilisant un IDE et sera connecté à un serveur avec une base de données.

## Objectifs

Le but de cette application est de pouvoir envoyer des messages d’un utilisateur connecté à un autre.

Afin que les utilisateurs puissent se connecter, un système de login doit être intégré à l’application. Pour les utilisateurs qui ne possèdent pas encore de compte, la création de celui-ci est indispensable, il faut donc intégrer un formulaire de création de compte.

Lorsqu’un message est envoyé, le destinataire le reçoit instantanément. Si le destinataire est hors-ligne, il recevra le message lors de sa prochaine connexion.

Tous les messages doivent être stockés dans une base de données afin qu’ils puissent être lus à tout moment.

Pour terminer, toutes les actions des utilisateurs doivent être enregistrées sous forme de logs sur le serveur.

## Planification initiale

# Analyse / Conception

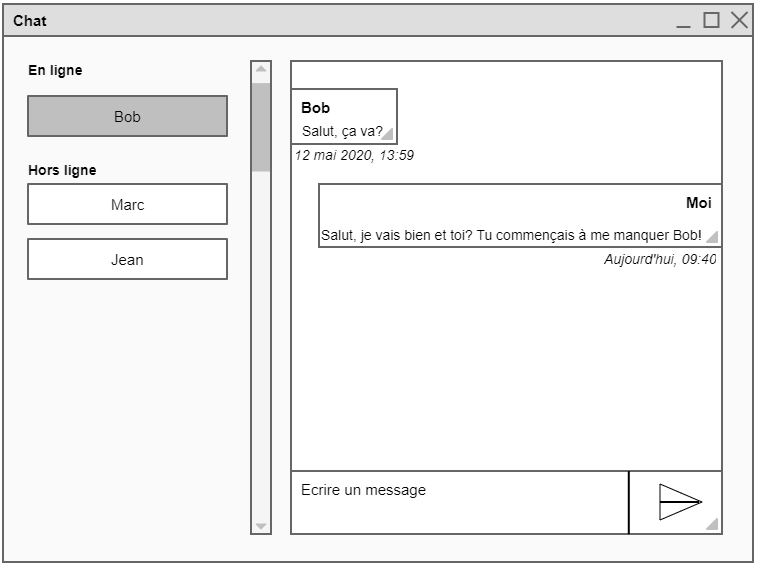
## Concept

### Maquettes graphiques de la partie client

La page ci-dessous permet à l’utilisateur de se connecter ou d’accéder à la page de création d’un compte s’il n’en a pas. La page ci-dessous permet à l’utilisateur de se créer un compte.

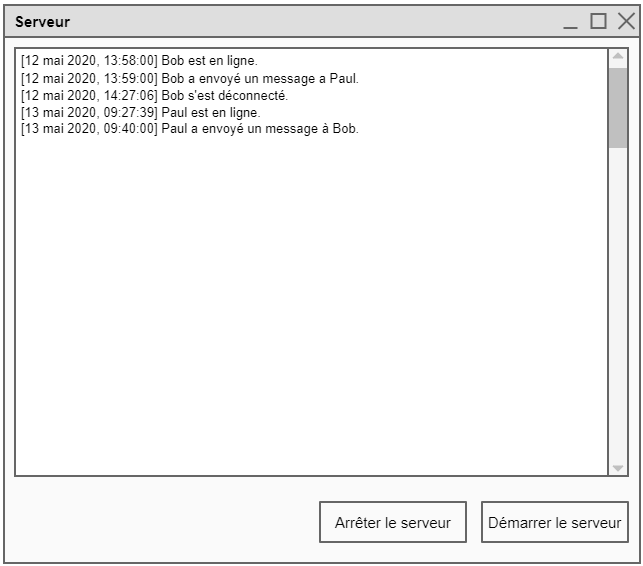
1 - Page de login

2 - Page de création d'un compte

La page ci-dessous permet aux utilisateurs de communiquer entre eux une fois qu’ils se sont connectés.

3 - Page de chat

### Maquette graphique de la partie serveur

La page ci-dessous permet de démarrer ou arrêter le serveur et de voir ce qu’il se passe sur les clients en temps réel.

4 - Page de logs

## Cas d’utilisation

### Partie client

|  |
| --- |
| Cas n°1 : Connexion à l’application de messagerie avec les bons identifiants |

|  |  |
| --- | --- |
| **Précondition** | L’utilisateur est sur la page de login |
| **Scénario** | 1. L’utilisateur insère son nom d’utilisateur 2. L’utilisateur insère son mot de passe 3. L’utilisateur clique sur « Se connecter » |
| **Performance attendue** | La page de chat apparaît en moins de 5 secondes. |

|  |
| --- |
| Cas n°2 : Connexion à l’application de messagerie avec les mauvais identifiants |

|  |  |
| --- | --- |
| **Précondition** | L’utilisateur est sur la page de login |
| **Scénario** | 1. L’utilisateur insère un nom d’utilisateur 2. L’utilisateur insère un mot de passe 3. L’utilisateur clique sur « Se connecter » |
| **Performance attendue** | Un message d’erreur apparaît instantanément. |

|  |  |
| --- | --- |
| Cas n°3 : Création d’un compte utilisateur | |
| **Précondition** | L’utilisateur est sur la page de création d’un compte |
| **Scénario** | 1. L’utilisateur insère un nom d’utilisateur 2. L’utilisateur insère un mot de passe 3. L’utilisateur insère le même mot de passe dans le champ de vérification 4. L’utilisateur clique sur « Créer » |
| **Performance attendue** | Le compte est créé et la page de login apparaît en moins de 5 secondes. |

|  |  |
| --- | --- |
| Cas n°4 : Création d’un compte utilisateur en insérant 2 mots de passe différents | |
| **Précondition** | L’utilisateur est sur la page de création d’un compte |
| **Scénario** | 1. L’utilisateur insère un nom d’utilisateur 2. L’utilisateur insère un mot de passe 3. L’utilisateur n’insère pas le même mot de passe dans le champ de vérification 4. L’utilisateur clique sur « Créer » |
| **Performance attendue** | Le compte n’est pas créé et un message d’erreur apparaît. |

|  |  |
| --- | --- |
| Cas n°5 : Envoi d’un message | |
| **Précondition** | L’utilisateur est sur la page de chat |
| **Scénario** | 1. L’utilisateur sélectionne un contact 2. L’utilisateur écrit un message 3. L’utilisateur clique sur le bouton d’envoi |
| **Performance attendue** | Le message est envoyé et il est affiché à l’écran avec la date d’envoi. |

|  |  |
| --- | --- |
| Cas n°5 : Réception d’un message | |
| **Précondition** | L’utilisateur est sur la page de chat |
| **Scénario** | 1. L’utilisateur sélectionne un contact |
| **Performance attendue** | Tous les messages envoyés sont affichés. |

### Partie serveur

|  |  |
| --- | --- |
| Cas n°1 : Démarrage du serveur | |
| **Précondition** | L’utilisateur est sur la page de logs |
| **Scénario** | 1. L’utilisateur clique sur « Démarrer le serveur » |
| **Performance attendue** | Un message disant que le serveur a démarré apparaît et toutes les actions de utilisateurs à venir apparaissent. |

|  |  |
| --- | --- |
| Cas n°2 : Fermeture du serveur | |
| **Précondition** | L’utilisateur est sur la page de logs |
| **Scénario** | 1. L’utilisateur clique sur « Arrêter le serveur » |
| **Performance attendue** | Un message disant que le serveur a été arrêté apparaît et les utilisateurs ne peuvent plus s’envoyer de messages. |

## Analyse concurrentielle

### Partie client

Si nous avions voulu nous diriger vers une application mobile, il aurait fallu utiliser Kotlin pour Android et Swift pour IOS.

Avantages de Kotlin :

* La quantité de code passe-partout est considérablement réduite grâce aux classes.
* Il existe plusieurs librairies pour faciliter notre travail tel que JVM, Android ou des librairies WEB.
* Facile à utiliser car il est possible de choisir n’importe quel build Java à partir de lignes de commande.

Avantages de Swift :

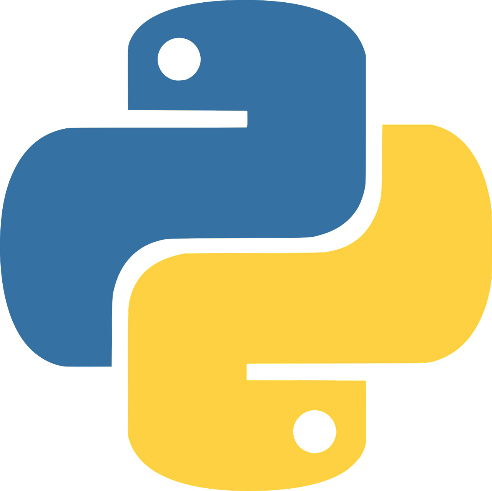
* Swift est moderne, les paramètres nommés sont exprimés dans une syntaxe propre.
* Swift est sécurisé, les tableaux et les intergers sont vérifiés pour que le débordement de mémoire soit automatiquement géré.
* Swift utilise la technologie haute performance LLVM.



5 - Swift

6 - Kotlin

### Partie Serveur

Pour la partie serveur, le langage utilisé est peu important. Ce qui est important, c’est que le langage puisse utiliser les Sockets. Pour cela, nous pouvons donc utiliser le Java ou le Python.



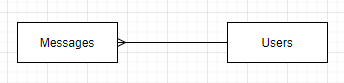
7 - Python

8 - Java

## Base de données

### Modèle conceptuel des données

Le MCD pour cette application est très simple, il faut une table « Utilisateurs » et une table « Messages ». Un utilisateur doit pouvoir envoyer et recevoir plusieurs messages.



9 - MCD

### Modèle logique des données

**Table « Users »**

Le champ « UserId » contient le numéro d’identifiant unique de chaque utilisateur.

Le champ « UserName » contient le nom d’utilisateur unique de chaque utilisateur.

Le champ « UserPassword » contient le mot de passe haché de chaque utilisateur.

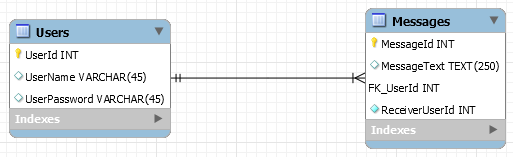
**Table « Messages »**

Le champ « MessageId » contient le numéro d’identifiant unique de chaque message.

Le champ « MessageText » contient le message.

Le champ « FK\_UserId » contient le numéro d’identifiant de l’utilisateur qui a envoyé le message.

Le champ « ReceiverUserId » contient le numéro d’identifiant de l’utilisateur destinataire.



10 - MLD

## Stratégie de test

Les tests s’effectuent une fois les fonctionnalités terminées. Ceux-ci sont repris des cas d’utilisation et se réalisent avec des données fictives.

Les tests doivent être effectués par :

* Deux personnes non informaticiennes

## Risques techniques

Etant donné que je n’ai jamais créé d’application en réseau, je risque de devoir utiliser du temps pour aller m’informer sur internet.

## Planification détaillée

La planification détaillée est accessible sur Github : <https://github.com/dylanramos/MessagingApp/projects>

Celle-ci contient les tâches à faire, en cours et terminées. Les tâches sont ajoutées chaque jour à mesure que le projet avance.

## Dossier de conception

### Outils

**Système d’exploitation**

* Windows 10 Professionnel 64 bits.

**Réseau**

* Accès à un routeur pour obtenir des adresses IP

**IDE**

* Microsoft Visual Studio 2019

**Base de données**

* Package NuGet pour Visual Studio : SQLite

# 

# Réalisation

## Modifications sur la partie graphique

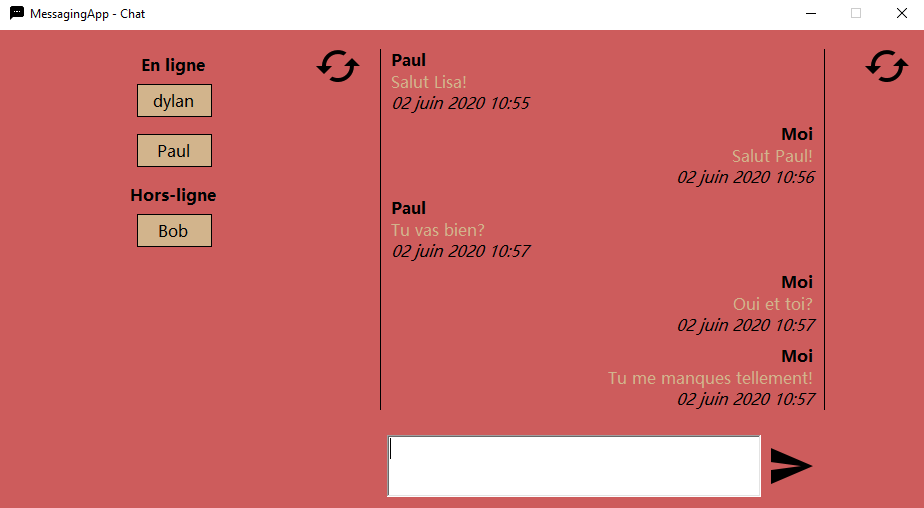
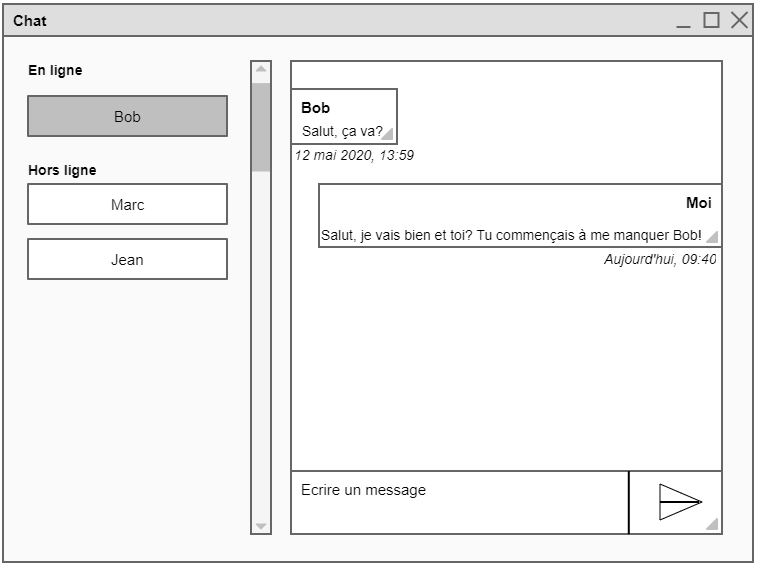
### Partie client



La page de login est modernisée, il y a des couleurs et des icônes.

Le bouton « Créer un compte » a été modifié en tant que lien hypertexte afin de rendre le formulaire plus propre.

11 - Page de login (application)

Comme pour la page précédente, la page de création de compte est elle aussi modernisée avec des couleurs et des icônes.

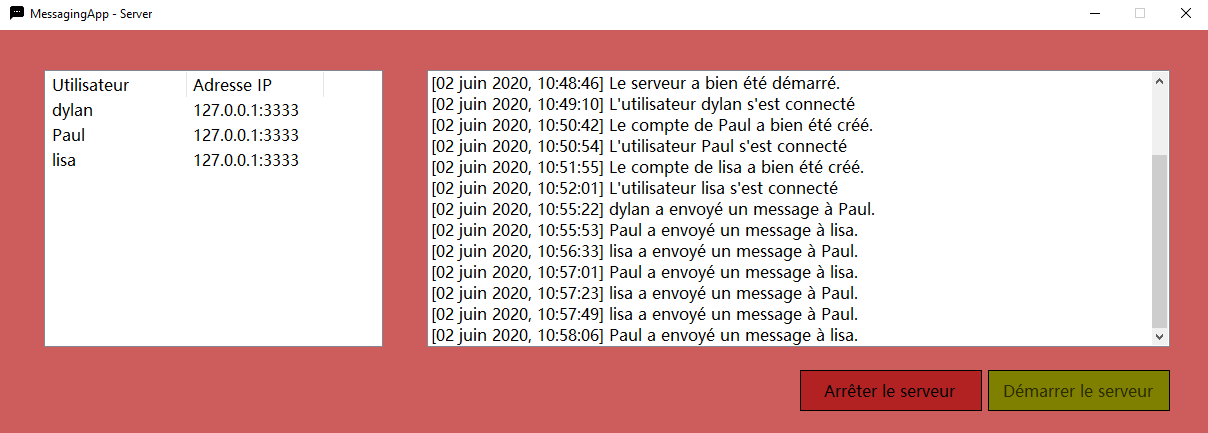
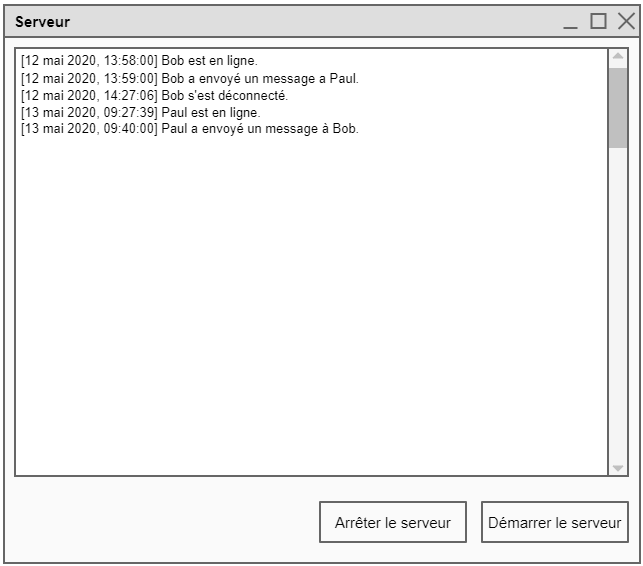
12 - Page de création d’un compte (application)

13 - Page de chat (application)

Comme pour la page précédente, la page de chat elle aussi modernisée avec des couleurs et des icônes. Cependant j’ai dû y ajouter deux boutons de rafraîchissement car j’ai eu quelques problèmes (ils seront expliqués par la suite).

Comme pour la page précédente, la page de chat elle aussi modernisée avec des couleurs et des icônes. Cependant j’ai dû y ajouter deux boutons de rafraîchissement car j’ai eu quelques problèmes (ils seront expliqués par la suite).

### Partie serveur



14 - Page de logs (application)

Pour la page de logs, j’ai trouvé nécessaire d’ajouter une fenêtre pour pouvoir voir qui est connecté avec quelle adresse IP.

Pour la page de logs, j’ai trouvé nécessaire d’ajouter une fenêtre pour pouvoir voir qui est connecté avec quelle adresse IP.

## Modifications sur la base de données

### Avant

### Après

1. Les longueurs des champs « UserName » et « UserPassword » sont passées à 25.
2. Un champ « MessageDate » a été ajouté pour stocker la date d’envoi du message.

## Icônes utilisées

Les icônes viennent d’un site open source appartenant à google, elles sont gratuites pour tout le monde.

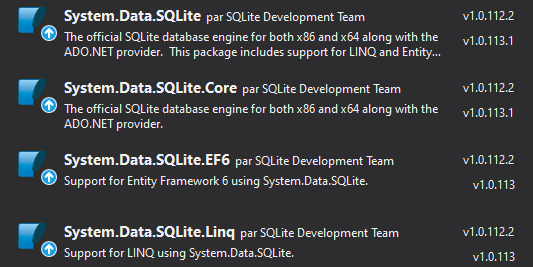
Lien :

<https://material.io/resources/icons/?icon=verified_user&style=baseline>

## Librairies utilisées

### SQLite

SQLite est un petit moteur de base de données très complet, il est rapide, autonome et de haute fiabilité. Je l’ai utilisé car je le connais maintenant très bien étant donné que je l’ai utilisé pour mes projets précédents.

Packages NuGet à installer :

15 - Packages NuGet SQLite

Exemple de code :

using System.Data.SQLite;

/// <summary>

/// Gets the requested data from the database

/// </summary>

/// <param name="sqlRequest"></param>

/// <returns></returns>

private SQLiteDataReader ExecuteQuery(string sqlRequest)

{

\_command = new SQLiteCommand(sqlRequest, \_dbConnection);

return \_command.ExecuteReader();

}

### Les sockets

Les sockets permettent la communication entre le client et le serveur, celle-ci peut se faire soit de manière synchrone soit de manière asynchrone. Pour cette application, la meilleure manière est l’asynchrone car elle permet d’envoyer et de recevoir les données en arrière-plan pendant que l’utilisateur utilise l’application.

Exemple de code :

using System.Net ;

using System.Net.Sockets ;

/// <summary>

/// Starts the connection with the remote server

/// </summary>

private void StartConnection()

{

try

{

\_socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,

ProtocolType.Tcp);

\_socket.BeginConnect(new IPEndPoint(IPAddress.Parse(SERVER\_IP),

SERVER\_PORT), new AsyncCallback(ConnectCallback), null);

}

catch (Exception)

{

MessageBox.Show("Le serveur distant est inaccessible.", "Erreur",

MessageBoxButtons.OK, MessageBoxIcon.Error);

Application.Exit();

}

}

### La cryptographie

La cryptographie permet de sécuriser le mot de passe en le cryptant. Pour cette application, le mot de passe est seulement haché.

Exemple de code :

using System.Security.Cryptography

// Password hash

var sha1 = new SHA1CryptoServiceProvider();

var data = Encoding.ASCII.GetBytes(passwordToCheck);

var hashedPassword = sha1.ComputeHash(data);

string password = Encoding.ASCII.GetString(hashedPassword);

## 

## Description des tests effectués

### Testeurs

Comme convenu avec le chef de projet, deux personnes non informaticiennes de ma famille ont testé mon application.

Mon père : **José Fonseca Ramos**

Ma sœur : **Océane Oliveira Ramos**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Synopsis** | | Création d’un compte | | | | |
| **Environnement** | | Visual Studio / Exécutable fourni | | | | |
| **Objectif** | | Vérifier si la création d’un compte est possible | | | | |
| **Données** | | Données fictives | | | | |
| **Prérequis** | | Être sur la page de création d’un compte | | | | |
| **Auteur** | | Dylan Oliveira Ramos | | | | |
|  | | | | | | |
| **N°** | **Action** | **Résultat attendu** | **Résultat** | **Testé par** | **Date** | **Commentaire** |
| 1 | L’utilisateur clique sur « Créer » alors qu’un ou plusieurs champs sont vides. | Un message d’erreur apparaît lui disant de remplir tous les champs. | OK | José | 03.06.2020 |  |
| OK | Océane | 03.06.2020 |  |
| 2 | L’utilisateur clique sur « Créer » alors qu’il a entré des caractères interdits : ; / " ( ) = , ' \ | Un message d’erreur apparaît lui disant de ne pas utiliser des caractères interdits. | OK | José | 03.06.2020 |  |
| OK | Océane | 03.06.2020 |  |
| 3 | L’utilisateur clique sur « Créer » alors que le mot de passe entré est inférieur à 10 caractères. | Un message d’erreur apparaît lui disant que le mot de passe ne respecte pas la taille minimum de 10 caractères. | OK | José | 03.06.2020 |  |
| OK | Océane | 03.06.2020 |  |
| 4 | L’utilisateur clique sur « Créer » alors que le mot de passe et la vérification du mot de passe ne sont pas identiques | Un message d’erreur apparaît lui disant que les deux mots de passe ne sont pas identiques. | OK | José | 03.06.2020 |  |
| OK | Océane | 03.06.2020 |  |
| 5 | L’utilisateur clique sur « Créer » alors que le serveur est arrêté. | Un message d’erreur apparaît lui disant que le serveur distant est inaccessible et l’application s’arrête. | OK | José | 03.06.2020 | L’application ne devrait pas s’arrêter. |
| OK | Océane | 03.06.2020 |  |
| 6 | L’utilisateur clique sur « Créer » alors que le nom d’utilisateur existe déjà. | Un message d’erreur apparaît lui disant que le nom d’utilisateur existe déjà. | OK | José | 03.06.2020 |  |
| OK | Océane | 03.06.2020 |  |
| 7 | L’utilisateur clique sur « Créer » alors que tout est rempli correctement. | Un message apparaît lui disant que son compte a bien été créé puis la page de login s’affiche.  Un message (log) apparaît sur le serveur en montrant le nom du compte créé. | OK | José | 03.06.2020 |  |
| OK | Océane | 03.06.2020 |  |
| 8 | L’utilisateur clique sur « Annuler ». | La page de login s’affiche. | OK | José | 03.06.2020 |  |
| OK | Océane | 03.06.2020 |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Synopsis** | | Connexion à un compte | | | | |
| **Environnement** | | Visual Studio / Exécutable fourni | | | | |
| **Objectif** | | Vérifier si la connexion au compte est possible | | | | |
| **Données** | | Données fictives | | | | |
| **Prérequis** | | Être sur la page de login | | | | |
| **Auteur** | | Dylan Oliveira Ramos | | | | |
|  | | | | | | |
| **N°** | **Action** | **Résultat attendu** | **Résultat** | **Testé par** | **Date** | **Commentaire** |
| 1 | L’utilisateur clique sur « Se connecter » alors qu’un ou plusieurs champs sont vides. | Un message d’erreur apparaît lui disant de remplir tous les champs. | OK | José | 03.06.2020 |  |
| OK | Océane | 03.06.2020 |  |
| 2 | L’utilisateur clique sur « Se connecter » alors que le serveur est arrêté. | Un message d’erreur apparaît lui disant que le serveur distant est inaccessible. | OK | José | 03.06.2020 |  |
| Ok | Océane | 03.06.2020 |  |
| 3 | L’utilisateur clique sur « Se connecter » alors que les identifiants sont incorrects. | Un message d’erreur apparaît lui disant que les identifiants sont incorrects. | OK | José | 03.06.2020 | Il faudrait ajouter un lien pour le mot de passe oublié. |
| OK | Océane | 03.06.2020 |  |
| 4 | L’utilisateur clique sur « Se connecter » alors que les identifiants sont corrects. | La page de chat s’affiche et un message (log) apparaît sur le serveur en montrant qui s’est connecté. | OK | José | 03.06.2020 |  |
| OK | Océane | 03.06.2020 |  |
| 5 | L’utilisateur clique sur « Créer un compte ». | La page de création d’un compte s’affiche. | OK | José | 03.06.2020 |  |
| OK | Océane | 03.06.2020 |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Synopsis** | | Utilisation du chat | | | | |
| **Environnement** | | Visual Studio / Exécutable fourni | | | | |
| **Objectif** | | Vérifier que l’envoi et la réception des messages est possible | | | | |
| **Données** | | Données fictives | | | | |
| **Prérequis** | | Être sur la page de chat (il faut se connecter à un compte) | | | | |
| **Auteur** | | Dylan Oliveira Ramos | | | | |
|  | | | | | | |
| **N°** | **Action** | **Résultat attendu** | **Résultat** | **Testé par** | **Date** | **Commentaire** |
| 1 | L’utilisateur clique sur un contact. | La conversation entre l’utilisateur et le contact sélectionné s’affiche avec pour chaque message le nom, le message et la date.  S’il n’y a jamais eu de conversation aucun message ne s’affiche. | OK | José | 03.06.2020 |  |
| OK | Océane | 03.06.2020 |  |
| 2 | L’utilisateur clique sur le bouton d’envoi d’un message alors qu’il n’a rien écrit. | Rien n’est envoyé. | OK | José | 03.06.2020 | Si on met un ou plusieurs espaces le message est quand même envoyé alors qu’il est vide. |
| OK | Océane | 03.06.2020 |  |
| 3 | L’utilisateur clique sur le bouton d’envoi d’un message alors qu’il a écrit quelque chose. | Le message est envoyé au destinataire et il s’affiche dans la conversation.  Le champ de texte est vidé.  Le serveur affiche qu’un message a été envoyé. | OK | José | 03.06.2020 |  |
| OK | Océane | 03.06.2020 | Le « ç » et le « è » affichent un « ? ». |
| 4 | L’utilisateur clique sur le bouton d’envoi d’un message alors qu’il a entré des caractères interdits : ; / " ( ) = , ' \ | Un message d’erreur apparaît lui disant de ne pas utiliser des caractères interdits. | OK | José | 03.06.2020 |  |
| OK | Océane | 03.06.2020 | Dommage que ces caractères soient interdits car ils sont utiles. |
| 5 | L’utilisateur clique sur le bouton d’actualisation des contacts. | La liste des contacts s’actualise. | OK | José | 03.06.2020 | Dommage que la liste des contacts ne s’actualise pas toute seule. |
| OK | Océane | 03.06.2020 | Dommage que la liste des contacts ne s’actualise pas toute seule. |
| 6 | L’utilisateur clique sur le bouton d’actualisation des messages. | La conversation s’actualise. | OK | José | 03.06.2020 | Dommage qu’un chat ne s’actualise pas tout seul. |
| OK | Océane | 03.06.2020 | Dommage qu’un chat ne s’actualise pas tout seul. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Synopsis** | | Utilisation du serveur | | | | |
| **Environnement** | | Visual Studio / Exécutable fourni | | | | |
| **Objectif** | | Vérifier si le serveur fonctionne | | | | |
| **Données** | | Données fictives | | | | |
| **Prérequis** | | Être sur la page principale du serveur | | | | |
| **Auteur** | | Dylan Oliveira Ramos | | | | |
|  | | | | | | |
| **N°** | **Action** | **Résultat attendu** | **Résultat** | **Testé par** | **Date** | **Commentaire** |
| 1 | L’utilisateur clique sur « Démarrer le serveur ». | Un message apparaît disant que le serveur a bien été démarré.  Les actions à venir des utilisateurs s’affichent. | OK | José | 03.06.2020 |  |
| OK | Océane | 03.06.2020 |  |
| 2 | L’utilisateur clique sur « Arrêter le serveur ». | Un message apparaît disant que le serveur a bien été arrêté.  Les utilisateurs sont déconnectés.  Les actions à venir des utilisateurs ne s’affichent plus. | OK | José | 03.06.2020 |  |
| OK | Océane | 03.06.2020 |  |

## Erreurs restantes

### Envoi de caractères spéciaux interdit

L’insertion de caractères spéciaux tels que ; / " ( ) = , ' \ est interdite car il n’y a pas de sécurisation au niveau de la base de données (protection contre l’injection SQL). De plus le ; et le / sont utilisés pour l’échange des requêtes entre le client et le serveur, il faudrait donc utiliser une autre méthode de communication.

**Conséquence :** l’utilisateur ne peut pas utiliser ces caractères pour son nom de compte, son mot de passe et ses messages.

**Solution :** il faudrait envoyer et recevoir des objets sérialisés au lieu d’envoyer et de recevoir du texte.

### Actualisation du chat et des contacts

Sur cette application, l’actualisation des contacts (pour voir s’ils sont en ligne ou non) et l’actualisation des messages ne se fait pas automatiquement. Ce n’est pas vraiment une erreur étant donné que cela fonctionne mais ce n’était pas prévu initialement.

**Conséquence :** l’utilisateur doit actualiser les données manuellement dès qu’il veut voir un contact en ligne ou qu’il attend un message.

**Solution :** il faudrait qu’il y ait un socket connecté en permanence au serveur sur un thread différent (pour fonctionner en arrière-plan) qui demande les données toutes les X secondes.

### Problème d’affichage de certains caractères

Lorsqu’un message contenant un « ç » ou un « è » est envoyé, ces caractères sont remplacés par des « ? ». Ce problème est probablement dû à la conversion du texte en ASCII.

**Conséquence :** l’utilisateur ne peut pas envoyer ses messages comme il le désire.

**Solution :** il faudrait essayer de convertir le texte en UTF-8.

# Conclusions

## Points positifs

### Projet

Ce projet m’a permis d’apprendre à utiliser les sockets en C#. Avant de le commencer, je n’avais aucune connaissance en programmation réseau, je suis donc fier de ce que j’ai pu accomplir. De plus, je n’ai pas eu de retard, j’ai pu finir le projet dans les temps.

### Application

L’application est fonctionnelle avec toutes les fonctionnalités prévues initialement et elle a, selon moi, une belle interface graphique.

## Points négatifs

### Application

Le point négatif de mon application est l’actualisation des données de la page de chat. Elle doit se faire manuellement, cela serait donc très dérangeant si nous voulions l’utiliser en entreprise par exemple.

## Difficultés

En début de projet j’ai pris beaucoup de temps à m’informer sur la programmation avec des sockets et je n’y comprenais pas grand-chose. Mais petit à petit, en testant quelques bouts de code repris sur internet, j’ai finalement réussi à comprendre le fonctionnement des sockets.

## Suites possibles

L’une des suites possibles serait d’ajouter un système d’ajout d’amis car actuellement, un utilisateur connecté peut voir tous les contacts qui existent et ne peut pas choisir ses contacts favoris.

Une autre suite possible serait de pouvoir envoyer des pièces jointes telles que des photos ou des vidéos par exemple.

Ou encore, une autre suite possible serait de pouvoir exporter la liste des logs dans un document texte par exemple.

# Annexes

## Journal de travail

Le journal de travail est sous le nom de « Journal de travail.xlsx » dans le dossier rendu final, il contient toutes les tâches réalisées durant se projet.

## Manuel d'Utilisation

Le manuel d’utilisation est sous le nom de « Manuel d’utilisation.pdf » dans le dossier rendu final, il contient toutes les informations dont l’utilisateur a besoin pour utiliser l’application.

## Remerciements

Merci à Monsieur Claude ROCHAT le chef de projet, pour son soutien, sa disponibilité et ses conseils lors de la réalisation de mon projet.

Merci à José FONSECA RAMOS mon père, pour avoir testé et commenté mon application.

Enfin, merci à Océane OLIVEIRA RAMOS ma sœur pour avoir testé et commenté mon application.

## Table des illustrations

[1 - Page de login 6](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc42156832)

[2 - Page de création d'un compte 7](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc42156833)

[3 - Page de chat 8](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc42156834)

[4 - Page de logs 9](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc42156835)

[5 - Swift 13](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc42156836)

[6 - Kotlin 13](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc42156837)

[7 - Python 14](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc42156838)

[8 - Java 14](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc42156839)

[9 - MCD 14](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc42156840)

[10 - MLD 15](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc42156841)

[11 - Page de login (application) 17](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc42156842)

[12 - Page de création d’un compte (application) 18](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc42156843)

[13 - Page de chat (application) 19](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc42156844)

[14 - Page de logs (application) 20](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc42156845)

[15 - Packages NuGet SQLite 22](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc42156846)

## Bibliographie

Allexy. (2006, janvier 30). *TCP/IP Chat Application Using C#*. Récupéré sur Code Project: https://www.codeproject.com/Articles/12893/TCP-IP-Chat-Application-Using-C

Brian. (2012, août 22). *C# Async Sockets Part 1: Basics*. Récupéré sur YouTube: https://www.youtube.com/watch?v=Bq1JhTHlxek

CaptJiggly. (2012, mai 21). *C# Sockets Multiple Connection 1 - Accepting Connections*. Récupéré sur YouTube: https://www.youtube.com/watch?v=cHq2lYLA4XY

CaptJiggly. (2012, mai 22). *C# Sockets Multiple Connection 2 - Receiving Data/Handling Disconnection*. Récupéré sur YouTube: https://www.youtube.com/watch?v=p8Nlxtj0sV4

Dotnetperls. (consulté en mai 2020). *C# Split String Examples*. Récupéré sur Dotnetperls: https://www.dotnetperls.com/split

Microsoft. (2017, mars 30). *Asynchronous Client Socket Example*. Récupéré sur Microsoft Docs: https://docs.microsoft.com/en-us/dotnet/framework/network-programming/asynchronous-client-socket-example

Microsoft. (2017, mars 30). *Asynchronous Server Socket Example*. Récupéré sur Microsoft Docs: https://docs.microsoft.com/en-us/dotnet/framework/network-programming/asynchronous-server-socket-example

Stack Overflow. (2009, mars). *How do I update the GUI from another thread?* Récupéré sur Stackoverflow: https://stackoverflow.com/questions/661561/how-do-i-update-the-gui-from-another-thread

Stack Overflow. (2009, septembre 20). *Splitting a string into chunks of a certain size*. Récupéré sur Stackoverflow: https://stackoverflow.com/questions/1450774/splitting-a-string-into-chunks-of-a-certain-size

Stack Overflow. (2010, novembre 15). *How to hash a password*. Récupéré sur Stackoverflow: https://stackoverflow.com/questions/4181198/how-to-hash-a-password

Stack Overflow. (2012, novembre 14). *Preventing Winform from being maximized?* Récupéré sur Stackoverflow: https://stackoverflow.com/questions/13381127/preventing-winform-from-being-maximized

Stack Overflow. (2013, avril 18). *Converting string to byte array in C#*. Récupéré sur Stackoverflow: https://stackoverflow.com/questions/16072709/converting-string-to-byte-array-in-c-sharp

Stack Overflow. (2013, mars 22). *Remove the last three characters from a string*. Récupéré sur Stackoverflow: https://stackoverflow.com/questions/15564944/remove-the-last-three-characters-from-a-string/15564958

Stack Overflow. (2013, juillet 19). *Stackoverflow*. Récupéré sur How to Programmatically Scroll a Panel: https://stackoverflow.com/questions/17752970/how-to-programmatically-scroll-a-panel

Stack Overflow. (2013, mars 12). *What is the proper way of closing and cleaning up a Socket connection?* Récupéré sur Stackoverflow: https://stackoverflow.com/questions/15354946/what-is-the-proper-way-of-closing-and-cleaning-up-a-socket-connection

Stack Overflow. (2016, janvier). *Check Whether a TextBox is empty or not*. Récupéré sur Stackoverflow: https://stackoverflow.com/questions/34298857/check-whether-a-textbox-is-empty-or-not/34299121#34299121

TeapotDev. (2012, juillet 26). *Simple Instant Messenger with SSL Encryption in C#*. Récupéré sur Code Project: https://www.codeproject.com/Articles/429144/Simple-Instant-Messenger-with-SSL-Encryption-in-Cs

Tutorials, C. (2017, mars 11). *Client Server programming in C# (Chat application)*. Récupéré sur YouTube: https://www.youtube.com/watch?v=X16IyNbcAr0