|  |
| --- |
| 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

[Table des matières 1](#_Toc41999760)

[Glossaire 3](#_Toc41999761)

[Analyse préliminaire 3](#_Toc41999762)

[Introduction 3](#_Toc41999763)

[Objectifs 3](#_Toc41999764)

[Planification initiale 5](#_Toc41999765)

[Analyse / Conception 6](#_Toc41999766)

[Concept 6](#_Toc41999767)

[Maquettes graphiques de la partie client 6](#_Toc41999768)

[Maquette graphique de la partie serveur 9](#_Toc41999769)

[Cas d’utilisation 10](#_Toc41999770)

[Partie client 10](#_Toc41999771)

[Partie serveur 12](#_Toc41999772)

[Analyse concurrentielle 13](#_Toc41999773)

[Partie client 13](#_Toc41999774)

[Partie Serveur 14](#_Toc41999775)

[14](#_Toc41999776)

[Base de données 14](#_Toc41999777)

[Modèle conceptuel des données 14](#_Toc41999778)

[Modèle logique des données 14](#_Toc41999779)

[Stratégie de test 15](#_Toc41999780)

[Risques techniques 15](#_Toc41999781)

[Planification détaillée 15](#_Toc41999782)

[Dossier de conception 16](#_Toc41999783)

[Outils 16](#_Toc41999784)

[Réalisation 17](#_Toc41999785)

[Modifications sur la partie graphique 17](#_Toc41999786)

[Partie serveur 20](#_Toc41999787)

[20](#_Toc41999788)

[Icônes utilisées 21](#_Toc41999789)

[Librairies utilisées 21](#_Toc41999790)

[SQLite 21](#_Toc41999791)

[Les sockets 22](#_Toc41999792)

[La cryptographie 22](#_Toc41999793)

[Description des tests effectués 23](#_Toc41999794)

[Erreurs restantes 23](#_Toc41999795)

[Liste des documents fournis 23](#_Toc41999796)

[Conclusions 23](#_Toc41999797)

[Annexes 24](#_Toc41999798)

[Résumé du rapport du TPI 24](#_Toc41999799)

[Journal de travail 24](#_Toc41999800)

[Manuel d'Installation 24](#_Toc41999801)

[Manuel d'Utilisation 24](#_Toc41999802)

[Archives du projet 24](#_Toc41999803)

[Table des illustrations 24](#_Toc41999804)

[Sources 24](#_Toc41999805)

[Bibliographie 24](#_Toc41999806)

# Glossaire

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

**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.

**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.

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

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

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

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

# 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

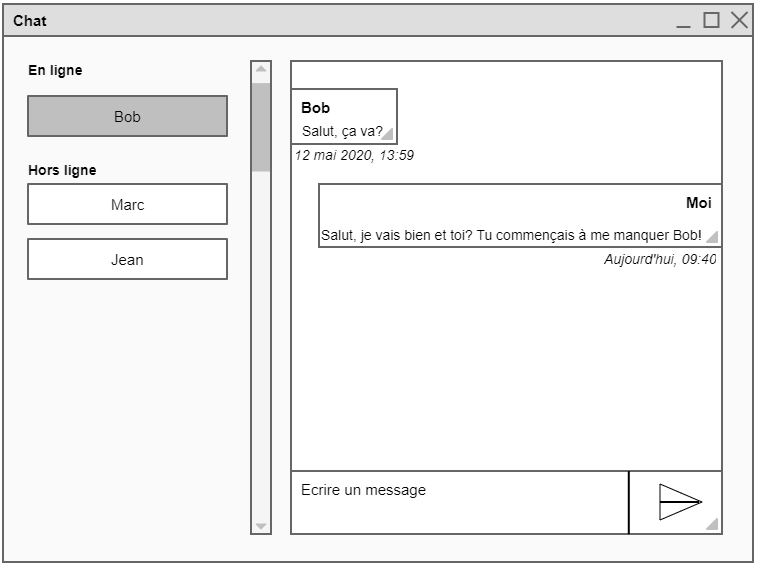
### Maquettes graphiques de la partie client

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

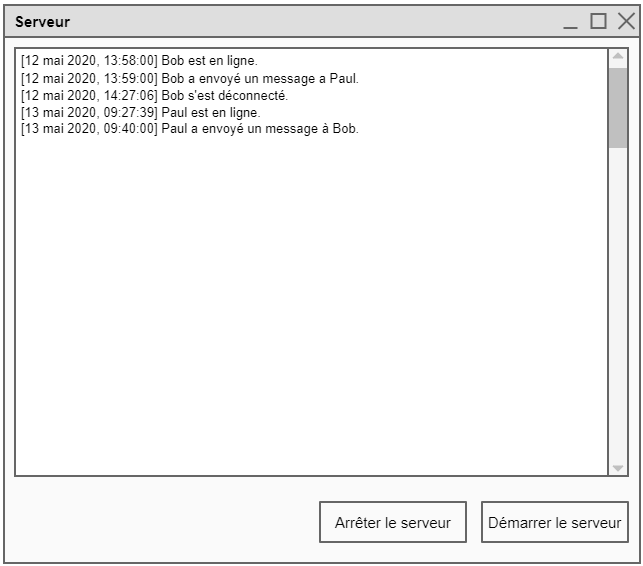
2 - Page de création d'un compte

3 - Page de chat

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

4 - 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.

5 - 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 chat 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 :

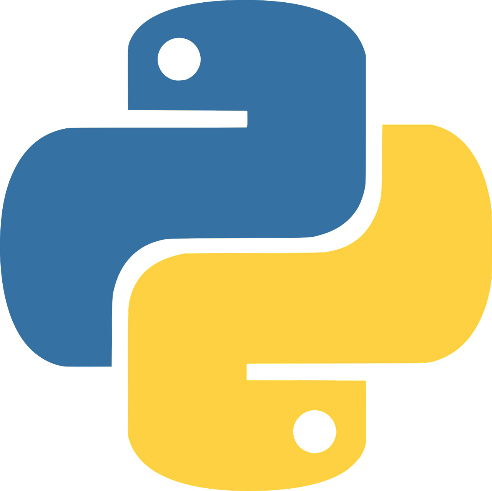
* Swift est moderne, les paramètres nommés sont exprimés dans une syntaxe propre.
* Swift est sécurisé, les tableaux et les intergers sont vérifiés pour que le débordement de mémoire soit automatiquement géré.
* Swift utilise la technologie haute performance LLVM.



6 - Kotlin

7 - Swift

### 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.

### Logiciels Java | Oracle Suisse

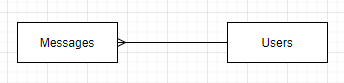
8 - Python

9 - 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.



10 - 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.

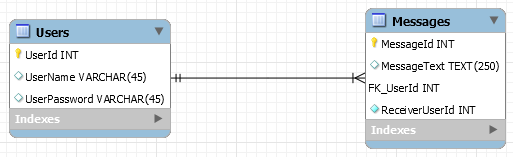
**Table « Messages »**

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.



11 - 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 :

* Le candidat
* Le chef de projet
* Trois 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

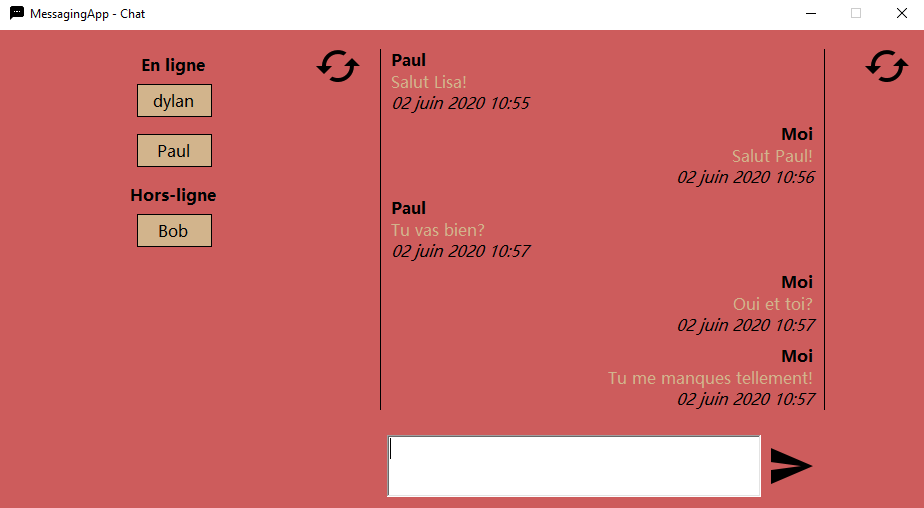
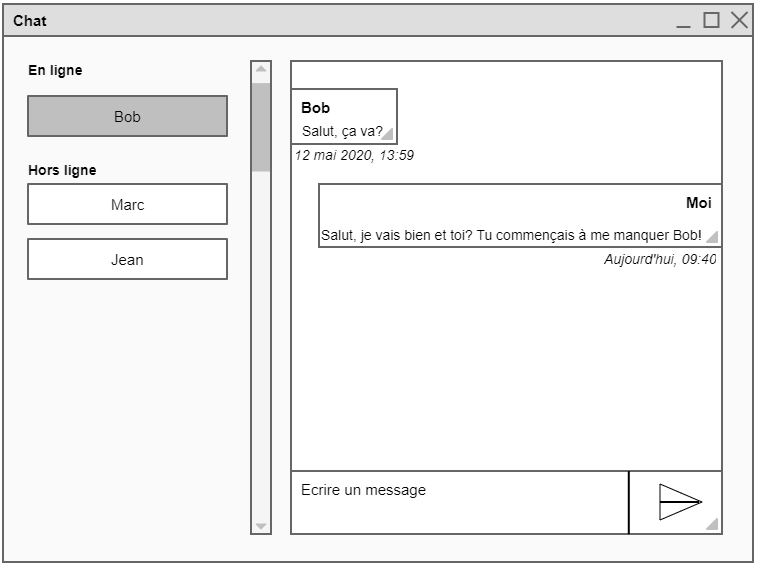
Partie client



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.

12 - 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.

13 - Page de création d’un compte (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).

14 - Page de chat (application)

### Partie serveur

### 

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

15 - Page de logs (application)

## Icônes utilisées

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

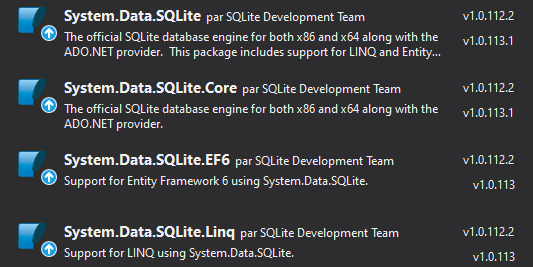
Lien :

<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 :

16 - 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

TESTER DEPUIS AUTRE PC

*Pour chaque partie testée de votre projet, il faut décrire:*

* *les conditions exactes de chaque test*
* *les preuves de test (papier ou fichier)*
* *tests sans preuve: fournir au moins une description*

## Erreurs restantes

*S'il reste encore des erreurs:*

* *Description détaillée*
* *Conséquences sur l'utilisation du produit*
* *Actions envisagées ou possibles*

## Liste des documents fournis

*Lister les documents fournis au client avec votre produit, en indiquant les numéros de versions*

* *le rapport de projet*
* *le manuel d'Installation (en annexe)*
* *le manuel d'Utilisation avec des exemples graphiques (en annexe)*
* *autres…*

# Conclusions

*Développez en tous cas les points suivants:*

* *Objectifs atteints / non-atteints*
* *Points positifs / négatifs*
* *Difficultés particulières*
* *Suites possibles pour le projet (évolutions & améliorations)*

# Annexes

## Résumé du rapport du TPI

## Journal de travail

## Manuel d'Installation

## Manuel d'Utilisation

## Archives du projet

*Media, … dans une fourre en plastique*

## Table des illustrations

[1 - Page de login 5](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc41999394)

[2 - Page de création d'un compte 6](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc41999395)

[3 - Page de chat 6](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc41999396)

[4 - Page de chat 7](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc41999397)

[5 - Page de logs 8](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc41999398)

[6 - Kotlin 12](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc41999399)

[7 - Swift 12](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc41999400)

[8 - Python 13](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc41999401)

[9 - Java 13](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc41999402)

[10 - MCD 13](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc41999403)

[11 - MLD 14](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc41999404)

[12 - Page de login (application) 16](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc41999405)

[13 - Page de création d’un compte (application) 17](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc41999406)

[14 - Page de chat (application) 18](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc41999407)

[15 - Page de logs (application) 19](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc41999408)

[16 - Packages NuGet SQLite 20](file:///C:\Users\Dylan\Desktop\MessagingApp\MessagingApp\Documentation\Rapport%20de%20projet.docx#_Toc41999409)

## Sources

# 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