

E-Banking

Solde : 2553.90

Date de début : mardi , 18 mai 2021

Date de fin : vendredi , 28 mai 2021

Filtrer

Liste des opérations

Destinataire	Envoyeur	Montant	Date	Information Transmise	Information Personnel
CA-456456	CA-455757	10.00	2021-05-27 14:38:12		
CA-456456	CA-455757	20.00	2021-05-27 14:39:39		
CA-456456	CA-455757	5.00	2021-05-27 14:40:59		
CA-455757	CA-456456	10.00	2021-05-27 15:21:00		
CA-455757	CA-456456	10.00	2021-05-28 00:00:00		
CA-455757	CA-456456	10.00	2021-05-28 08:40:16		
CA-455757	CA-456456	1.00	2021-05-28 08:42:07		

Rafraichir la liste Quitter Faire un versement

Robin Schmutz
Avenue de la gare 14
1450 Sainte-Croix
Robin.Schmutz@cpnv.ch

Table des matières

1	Introduction.....	4
1.1	<i>Cadre, description et motivation</i>	4
1.2	<i>Organisation</i>	4
1.3	<i>Objectifs</i>	4
1.4	<i>Planification initiale</i>	5
2	Analyse.....	6
2.1	Concept	6
2.1.1	Versement	6
2.1.2	Filtre de la liste des opérations	6
2.1.3	Module console.....	6
2.2	<i>Cahier des charges détaillé</i>	6
2.2.1	MCD	6
2.2.2	MLD	6
2.2.3	Maquettes :	7
2.2.4	Use case :	9
2.3	Stratégie de test.....	9
2.4	Budget initial	9
2.5	Convention de nommage.....	10
2.6	Planification	11
3	Implémentation	12
3.1	Dossier de conception	12
3.1.1	Hardware et système d'exploitation utilisé pour la réalisation et l'utilisation 12	
3.1.2	Le choix des outils logiciels pour la réalisation et l'utilisation	12
3.1.3	Database	12
3.1.4	Navigation des pages	13
3.1.5	Programmation et scripts	14
3.2	Risque Techniques	17
3.3	Dossier de réalisation	17
3.3.1	Logiciels utilisés.....	17
3.3.2	Arborescence du dépôt GitHub.....	17
3.3.3	Programmation et scripts.....	18
3.4	<i>Description des tests effectués</i>	21
3.4.1	Connexion Base de données.....	21
3.4.2	Connexion avec un compte	22
3.4.3	Création d'un versement.....	22
3.4.4	Trouver un compte par son numéro de compte	22
3.4.5	Trouver un compte par son id.....	22
3.4.6	Versement :	22
3.4.7	Module console :	23
3.4.8	Affichage des opérations :	24
3.4.9	Login.....	25
4	Mise en service.....	26
4.1	<i>Installation</i>	26
4.2	<i>Liste des documents fournis</i>	26

5	Conclusions.....	26
5.1	Objectifs atteints ?	26
5.2	Points positifs / négatifs	26
5.2.1	Négatif	26
5.2.2	Positif.....	26
5.3	Difficulté particulières.....	26
5.4	Suite pour le projet.....	27
6	Annexes.....	27
6.1	Sources – Bibliographie.....	27
6.2	Journal de bord.....	27
6.3	Manuel d'utilisation	30
6.4	Résumé de la documentation	30

1 Introduction

1.1 Cadre, description et motivation

Ce projet est réalisé dans le cadre de la formation d'informaticien au CPNV, pour mon TPI. L'objectif est d'avoir une application fonctionnel contenant toutes les fonctionnalités demandées par mon chef de projet et de mes 2 experts.

Ce projet sera sur la création d'une simulation d'un logiciel de E-Banking qui va permettra à un client de pouvoir faire des versements à d'autres comptes.

Le but de ce projet, c'est de montrer mon aptitude en programmation en C#, ma communication avec les experts et ma gestion de mon temps de travail.

1.2 Organisation

	Nom	Email	Numéro de Tél
Élève	Robin Schmutz	Robin.Schmutz@cpnv.ch	079 827 00 49
Chef de projet	Pascal Hurni	Pascal.hurni@cpnv.ch	078 616 48 08
Expert 1	Romain Gehrig	Romain.gehrig@gmail.com	079 714 43 58
Expert 2	Antoine Honore Mveng Evina	Antoine.mveng@eduvaud.ch	021 316 02 98

1.3 Objectifs

Les objectifs seront :

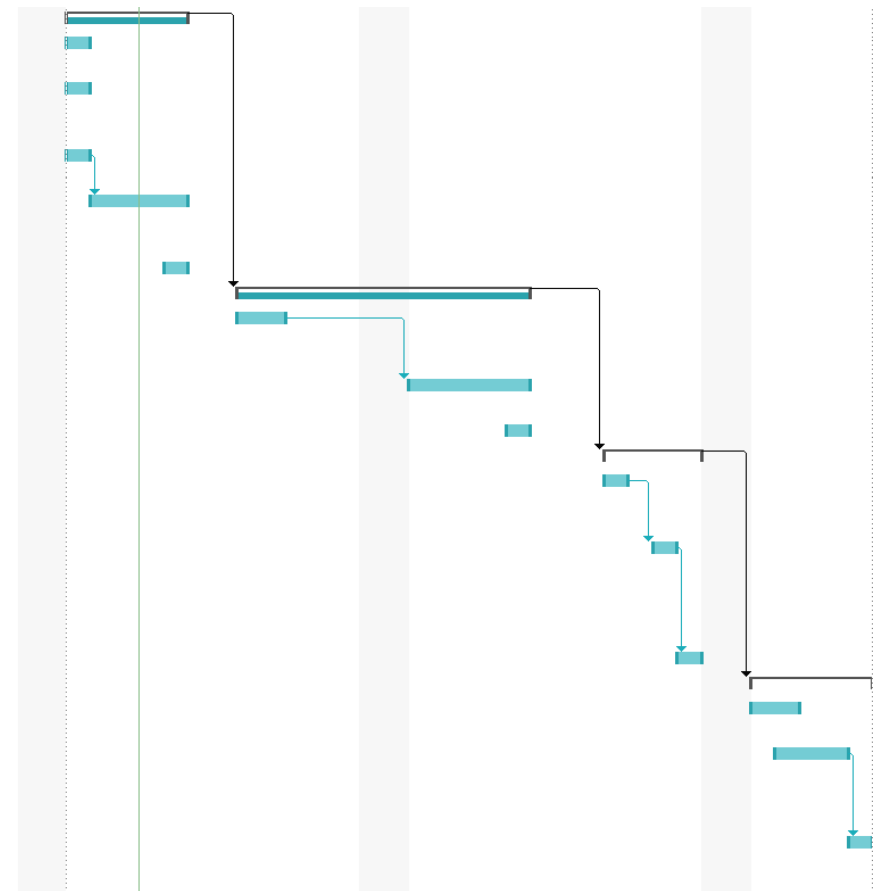
- S'identifier avec un login et mot de passe
- Afficher le solde actuel du compte
- Lister les opérations du mois courant
- Lister les opérations en indiquant une date de début et une date de fin
- Saisir un nouveau versement avec les données suivantes
 - Numéro de compte de destinataire
 - Montant du versement (contrôle sur le montant : ne peut dépasser le solde actuel)
 - Information transmise au destinataire (facultatif, une ligne de texte)
 - Remarque personnelle non transmise au destinataire (facultatif, plusieurs lignes de textes)
- Création d'un espace de versement pour les vendeurs avec les données suivantes :
 - Numéro de compte du vendeur
 - Numéro de compte du client
 - Montant du versement
- La gestion de plusieurs instances de l'interface client sur la même machine et réaliser des opérations, de même pour les vendeurs. Dont les effets des opérations seront visibles dans toutes les instances actives.

1.4 Planification initiale

Ce projet a commencé le 03 mai 2021 et finira le 04 juin 2021.

Il est partagé en 4 sprints. Je travaillerai tous les jours de la semaine dessus sauf les mercredis. Il y a aussi une pause le 13 et 14 mai à cause de l'ascension. Pareil pour le 24 mai qui est le lundi de pentecôte. Le jeudi 27 mai, je ne pourrai pas non plus travailler car j'ai un examen qui prévu ce jour-là.

♣ Sprint 1	4 jours	Lun 03.05.21	Ven 07.05.21			
Réception cahier des charges	1 jour	Lun 03.05.21	Lun 03.05.21			
Création Github + planification initiale	1 jour	Lun 03.05.21	Lun 03.05.21			
Création planning IceScrum	1 jour	Lun 03.05.21	Lun 03.05.21			
création BD avec Diagramme UML + Maquette	3 jours	Mar 04.05.21	Ven 07.05.21	4		
Documentation	1 jour	Ven 07.05.21	Ven 07.05.21			
♣ Sprint 2	8 jours	Lun 10.05.21	Ven 21.05.21	1		
Saisie d'un nouveau versement	2 jours	Lun 10.05.21	Mar 11.05.21			
Module console	4 jours	Lun 17.05.21	Ven 21.05.21	8		
Documentation	1 jour	Ven 21.05.21	Ven 21.05.21			
♣ Sprint 3	3 jours	Mar 25.05.21	Ven 28.05.21	7		
Lister les opérations du mois courant	1 jour	Mar 25.05.21	Mar 25.05.21			
Liste des opérations en indiquant une date de début et une date de fin	1 jour	Jeu 27.05.21	Jeu 27.05.21	12		
Documentation	1 jour	Ven 28.05.21	Ven 28.05.21	13		
♣ Sprint 4	4 jours	Lun 31.05.21	Ven 04.06.21	11		
Affichage du solde du compte	2 jours	Lun 31.05.21	Mar 01.06.21			
Identification d'un utilisateur avec login et mot de passe	2 jours	Mar 01.06.21	Jeu 03.06.21			
Documentation + résumé de la documentation	1 jour	Ven 04.06.21	Ven 04.06.21	17		



2 Analyse

2.1 Concept

2.1.1 Versement

Pour commencer le projet, je vais créer une base de données avec les tables compte et paiement. Pour faire un versement, il faut remplir un formulaire qui demande le numéro de compte du destinataire, le montant à envoyer.

2.1.2 Filtre de la liste des opérations

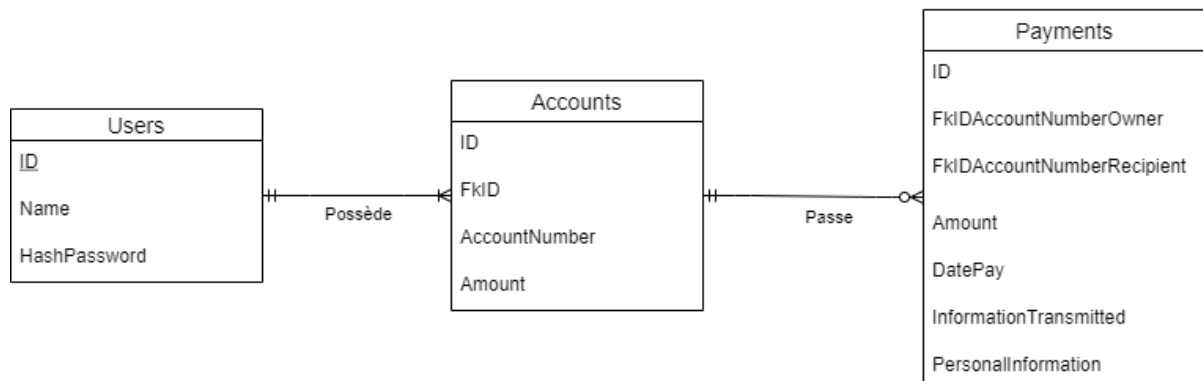
Le client peut filtrer ses opérations avec deux champs date dans la page principale. Dans les champs, on choisit les dates dans lesquels tous les versements qui ont été faits entre ces dates seront affichés dans la liste des opérations.

2.1.3 Module console

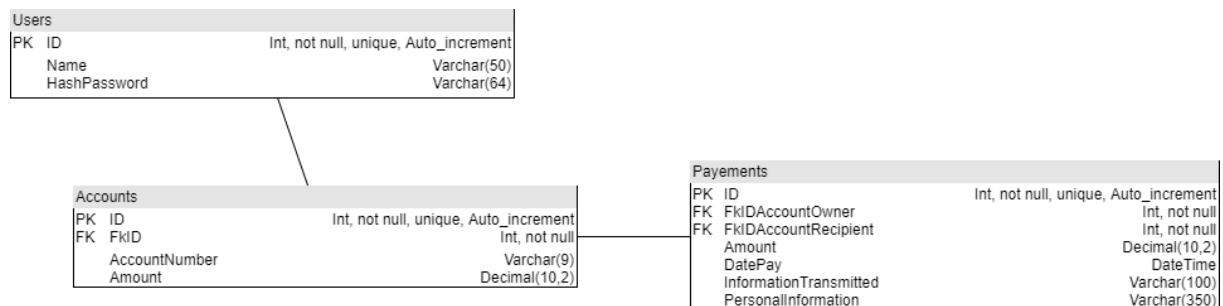
Le module console est une partie qui est réservé au vendeur. Ceci lui permettra d'avoir interface à lui où il pourra faire une demande des débits aux clients. Pour effectuer ceci, il doit donner trois informations : son numéro de compte, celui du client et le montant de la transaction.

2.2 Cahier des charges détaillé

2.2.1 MCD



2.2.2 MLD



2.2.3 Maquettes :

2.2.3.1 Login

Page de connexion

Identifiant

Mot de passe

Quitter

se connecter

2.2.3.2 HomePage

Page principale

Solde : 777 777 CHF

Date de début

15/05 /2021

Date de fin

15/05 /2021

Filtrer

Liste des opérations

Destinataire	Envoyeur	Montant	Date	Information Transmise	Information Personnel
CA-513232	CA-456456	40	15/05/2021	facture	
CA-986314	CA-456456	100	16/05/2021		
CA-123456	CA-456456	20	18/05/2021	facture	paiement en retard

Rafraichir la liste

Quitter


Faire un versement

2.2.3.3 Versement

Page de versement

Compte du destinataire

15/05 /2021



Montant

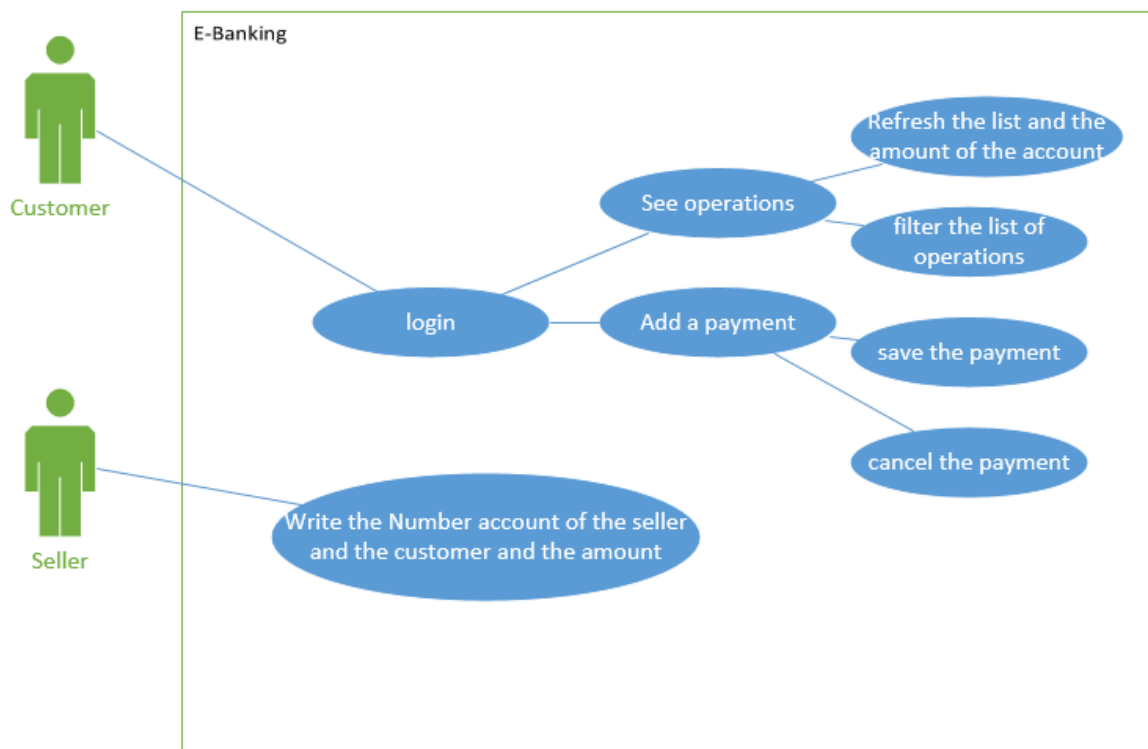
Information transmise

Remarque personnelle (non-transmise)

Annuler

Enregistrer

2.2.4 Use case :



2.3 Stratégie de test

Pour tester mon projet, je suis parti sur l'écriture d'un scénario de test que le code devrait passer pour être valide. Ces scénarios ont été écrits à chaque début de sprint et en fonction du cahier des charges.

2.4 Budget initial

Le budget de ce projet est de 90 heures réparties en 5 semaines. La date de début étant Lundi 03.05.2021 à 8h00 celle de fin vendredi 04.06.2021 à 14h15.

08:00	TPI SC-C213 ALTIERI Patrick SI-MI4a SI-C4a	Sport SC-Anrien Stand DAFELON Marc SI-C4a SI-MI4a	Economie d'entreprise 08:00 - 08:45 SC-C213 ZEN-RUFFINEN Xavier SI-C4a	TPI SC-C213 YKE SI-C4a SI-MI4a	08:00 - 08:45 SC-C213 YKE SI-C4a SI-MI4a	TPI SC-C213 ITHURBIDE Julien SI-MI4a SI-C4a
09:00			Mathématiques 08:50 - 09:35 SC-C213 DELAPORTE Stéphane SI-C4a	TPI SC-C213 GLASSEY Nicolas SI-C4a SI-MI4a		
10:00		TPI SC-C213 FAVRE Raphaël SI-MI4a SI-C4a	Anglais 09:50 - 10:35 SC-C213 RYSER Monika SI-C4a			
11:00	TPI SC-C213 ALTIERI Patrick SI-C4a SI-MI4a		Langue et communication 10:40 - 12:15 SC-C213 CRICCO Massimiliano SI-C4a			TPI SC-C213 KONOUTSE Yawo SI-C4a SI-MI4a
12:00						
13:00	TPI SC-C213 NGY SI-C4a SI-MI4a	TPI SC-C213 FAVRE Raphaël SI-C4a SI-MI4a	Société 13:30 - 15:05 SC-C213 CRICCO Massimiliano SI-C4a	Appui Maths 12:40 - 13:25 SC-M401 CSR SP-C2a SP-C1a SI-T1b		TPI SC-C213 KONOUTSE Yawo SI-C4a SI-MI4a
14:00	TPI SC-C213 ALTIERI Patrick SI-C4a SI-MI4a			TPI SC-C213 GLASSEY Nicolas SI-C4a SI-MI4a		
15:00						
16:00				TPI SC-C213 RFA SI-C4a SI-MI4a		Rattrapages - TE - Retenues SC-C181 CER SI-T2a SI-T2b SI-T2c SI-T1b SI-T1a SI-MI4b SI-MI4a SI-MI3b SI-MI3a SI-MI2b SI-MI2a SI-MI1b SI-MI1a SI-C2a SI-C1a SI-C4b SI-C4a SI-C3b

Pendant ces jours il y aura 3 jours de congé à cause du lundi de pentecôte et le vendredi de l'ascension. J'aurais aussi l'examen d'ECG (langue et communication et Société) qui se déroulera le matin du jeudi 27.05.2021, toutes ces heures ont été prise en compte dans les 90 heures.

Le 1^{er} juin je vais devoir m'absenter pendant 1 période pour un entretien. Cette période sera déduite de mon temps de travail. Au vu du travail que je dois effectuer, je pense pouvoir finir dans les temps déterminés.

2.5 Convention de nommage

J'ai utilisé la convention de nommage CamelCase qui est utilisé comme ceci :
nomDeMaVariable.

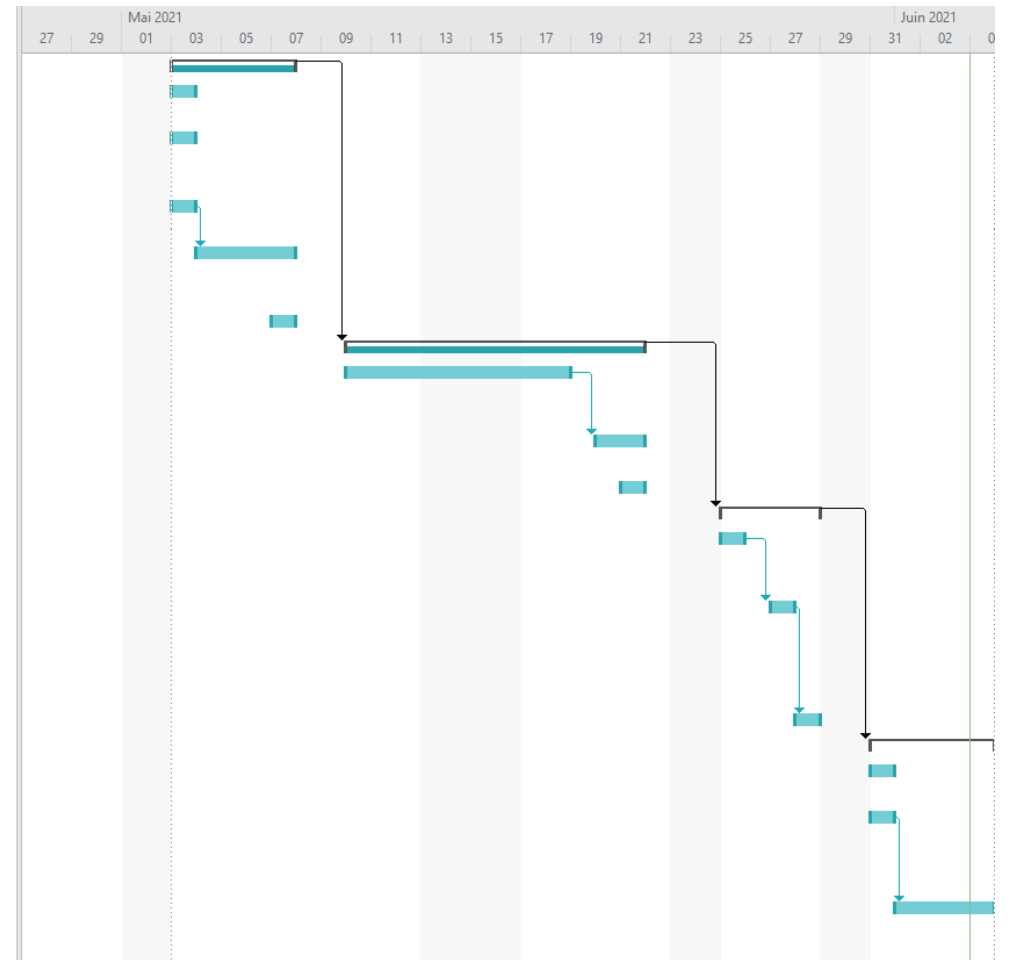
Elle est définie par une minuscule au début de nom et à une suite de majuscule à chaque mot.

J'utilise cette convention pour nommer mes variables ainsi que le nommage de mes entités de formulaire.

...

2.6 Planification

Nom de la tâche	Durée	Début	Fin	Prédécesseurs	Noms ressources	ter une nouvelle coli
▲ Sprint 1	4 jours	Lun 03.05.21	Ven 07.05.21			
Réception cahier des charges	1 jour	Lun 03.05.21	Lun 03.05.21			
Création Github + planification initiale	1 jour	Lun 03.05.21	Lun 03.05.21			
Création planning IceScrum	1 jour	Lun 03.05.21	Lun 03.05.21			
création BD avec Diagramme UML + Maquette	3 jours	Mar 04.05.21	Ven 07.05.21	4		
Documentation	1 jour	Ven 07.05.21	Ven 07.05.21			
▲ Sprint 2	6 jours	Lun 10.05.21	Ven 21.05.21	1		
Saisie d'un nouveau versement	4 jours	Lun 10.05.21	Mar 18.05.21			
Module console	2 jours	Jeu 20.05.21	Ven 21.05.21	8		
Documentation	1 jour	Ven 21.05.21	Ven 21.05.21			
▲ Sprint 3	3 jours	Mar 25.05.21	Ven 28.05.21	7		
Lister les opérations du mois courant	1 jour	Mar 25.05.21	Mar 25.05.21			
Liste des opérations en indiquant une date de début et une date de fin	1 jour	Jeu 27.05.21	Jeu 27.05.21	12		
Documentation	1 jour	Ven 28.05.21	Ven 28.05.21	13		
▲ Sprint 4	4 jours	Lun 31.05.21	Ven 04.06.21	11		
Affichage du solde du compte	1 jour	Lun 31.05.21	Lun 31.05.21			
Identification d'un utilisateur avec login et mot de passe	1 jour	Lun 31.05.21	Lun 31.05.21			
Documentation + résumé de la documentation	3 jours	Mar 01.06.21	Ven 04.06.21	17		



3 Implémentation

3.1 Dossier de conception

3.1.1 Hardware et système d'exploitation utilisé pour la réalisation et l'utilisation

Pour réaliser ce projet, j'ai utilisé un ordinateur fourni par le CPNV :

- Ordinateur fixe
 - OS : windows 10
 - CPU : intel Core i7-6700k
 - Ram : 16GB
 - GPU : intel (R)-HD Graphics 530

L'application ciblera les machines Windows mais sera seulement testé pour Windows 10.

3.1.2 Le choix des outils logiciels pour la réalisation et l'utilisation

Pour ce projet, j'ai utilisé le logiciel visual studio 2019, parce que j'ai des connaissances avec cet outil ce qui me permet de travailler directement sur le projet sans devoir perdre du temps en cherchant à comment utilisé le programme.

J'ai dû installer deux packages pour mon code : MySql.Data et Newtonsoft.Json
Mysql.Data me permet de communiquer avec la base de données depuis mon code.
Newtonsoft.Json me permet de lire le fichier Json qui fait la connexion à la DB.

J'ai utilisé l'outil heidi sql pour la gestion de la DB pour les mêmes raisons que celle de mon outil de programmation. Je connais l'outil ce qui me permet de ne pas perdre de temps et qu'il fait tout ce dont j'ai besoin pour la DB.

3.1.3 Database

La base de données contient trois tables :

- Users
- Accounts
- Payments

La table users contient les données des utilisateurs :

- ID
- Name
- HashPassword

La table accounts contient les données des comptes, elle est liée à la table users parce que un compte a forcément un client lié :

- ID
- FkID
- NumberAccount
- Amount

La table Payments contient les données des versements, elle est liée à la table accounts parce que un versement appartient à un compte :

- ID

- FkIDAccountOwner
- FkIDAccountRecipient
- Amount
- DatePay
- InformationTransmitted
- PersonalInformation

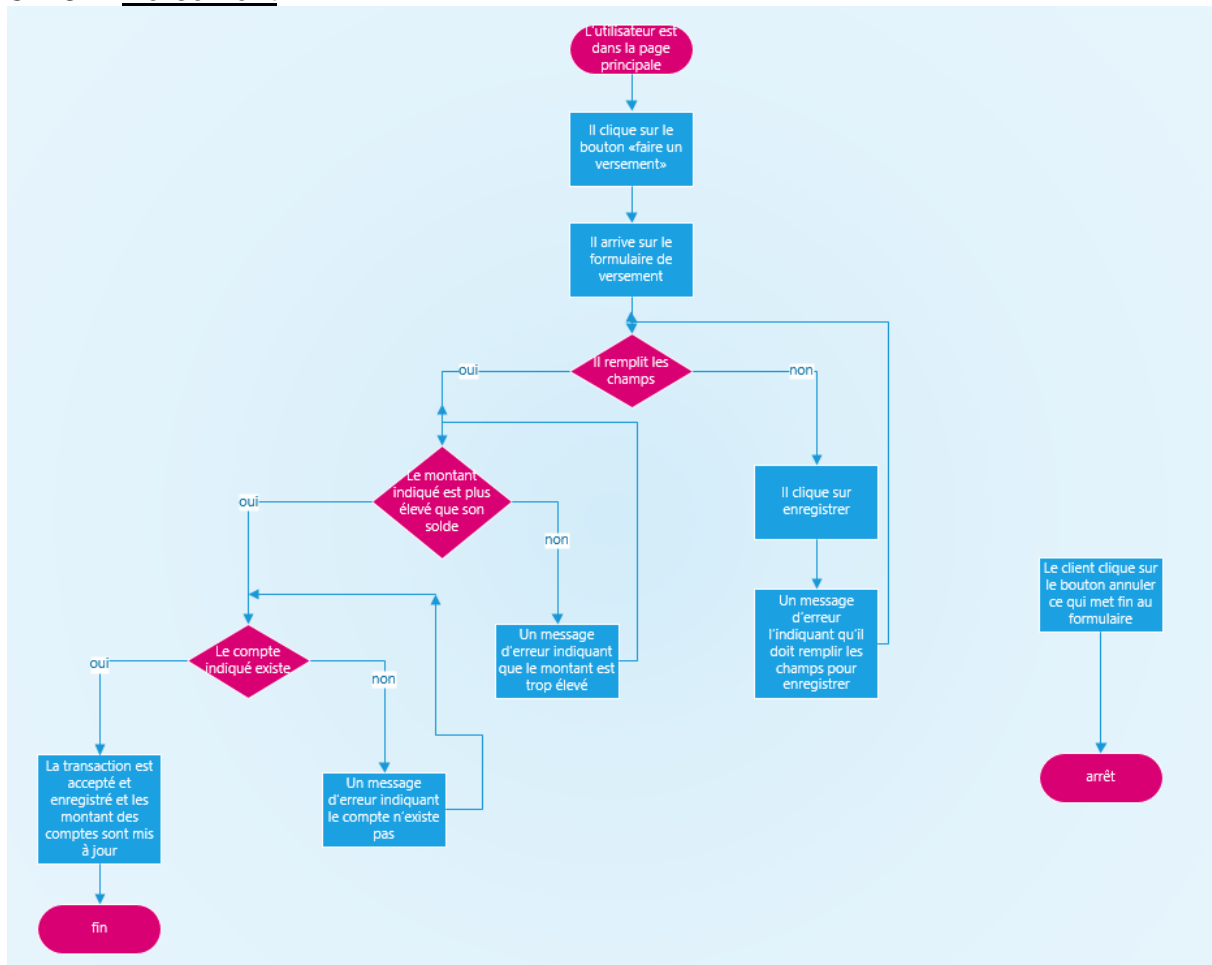
3.1.4 Navigation des pages

La navigation se fait de manière suivantes :

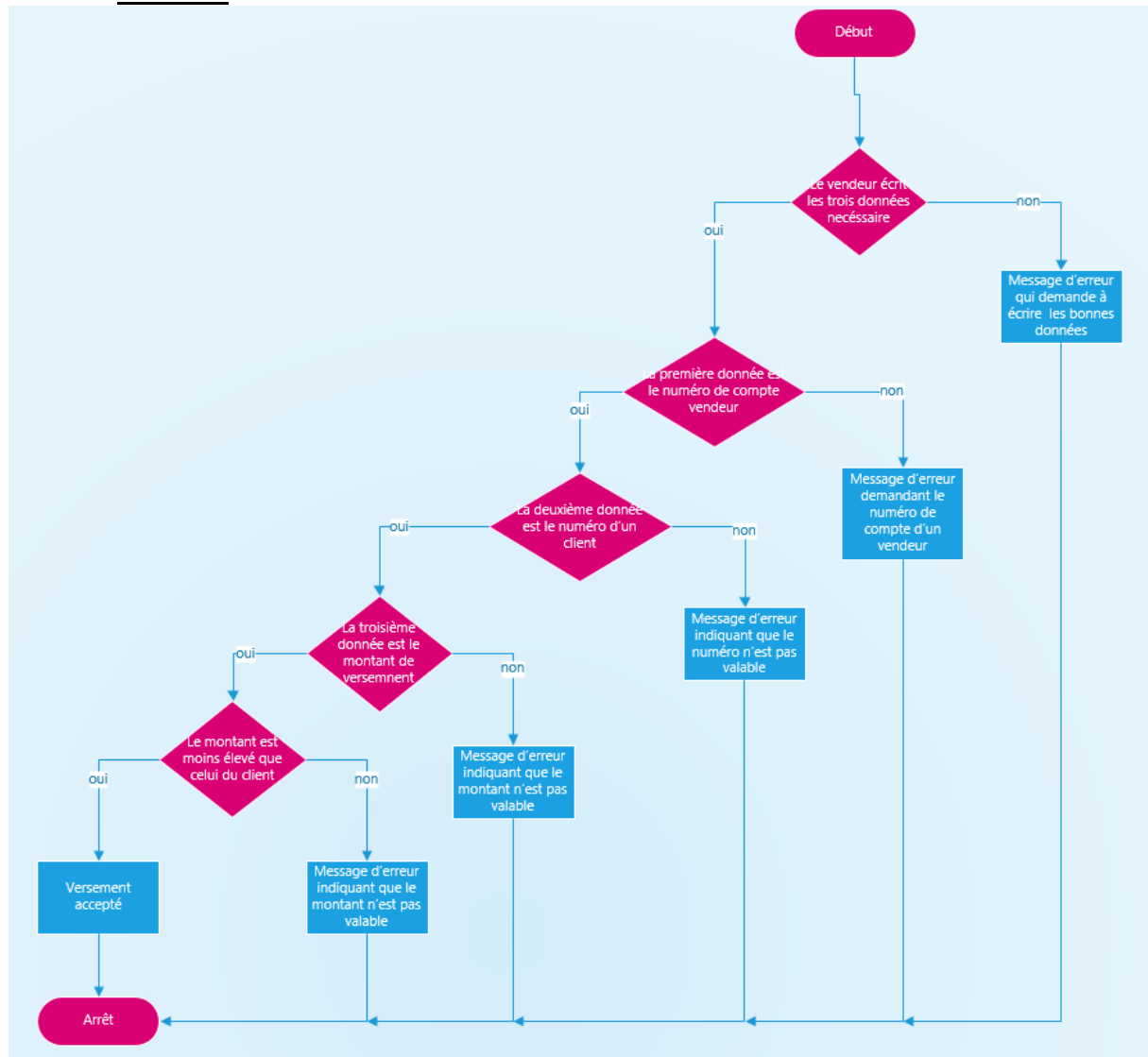


3.1.5 Programmation et scripts

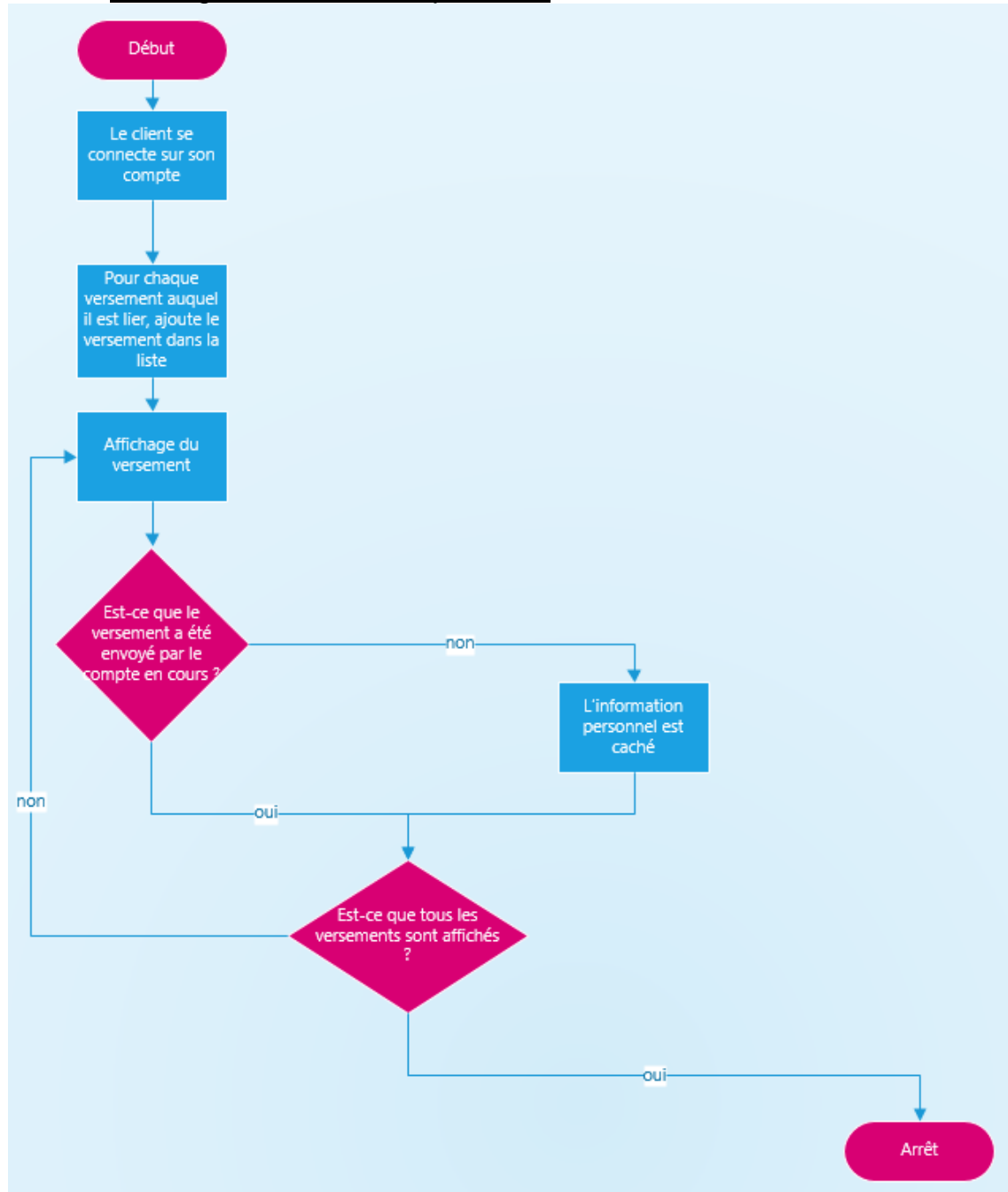
3.1.5.1 Versement



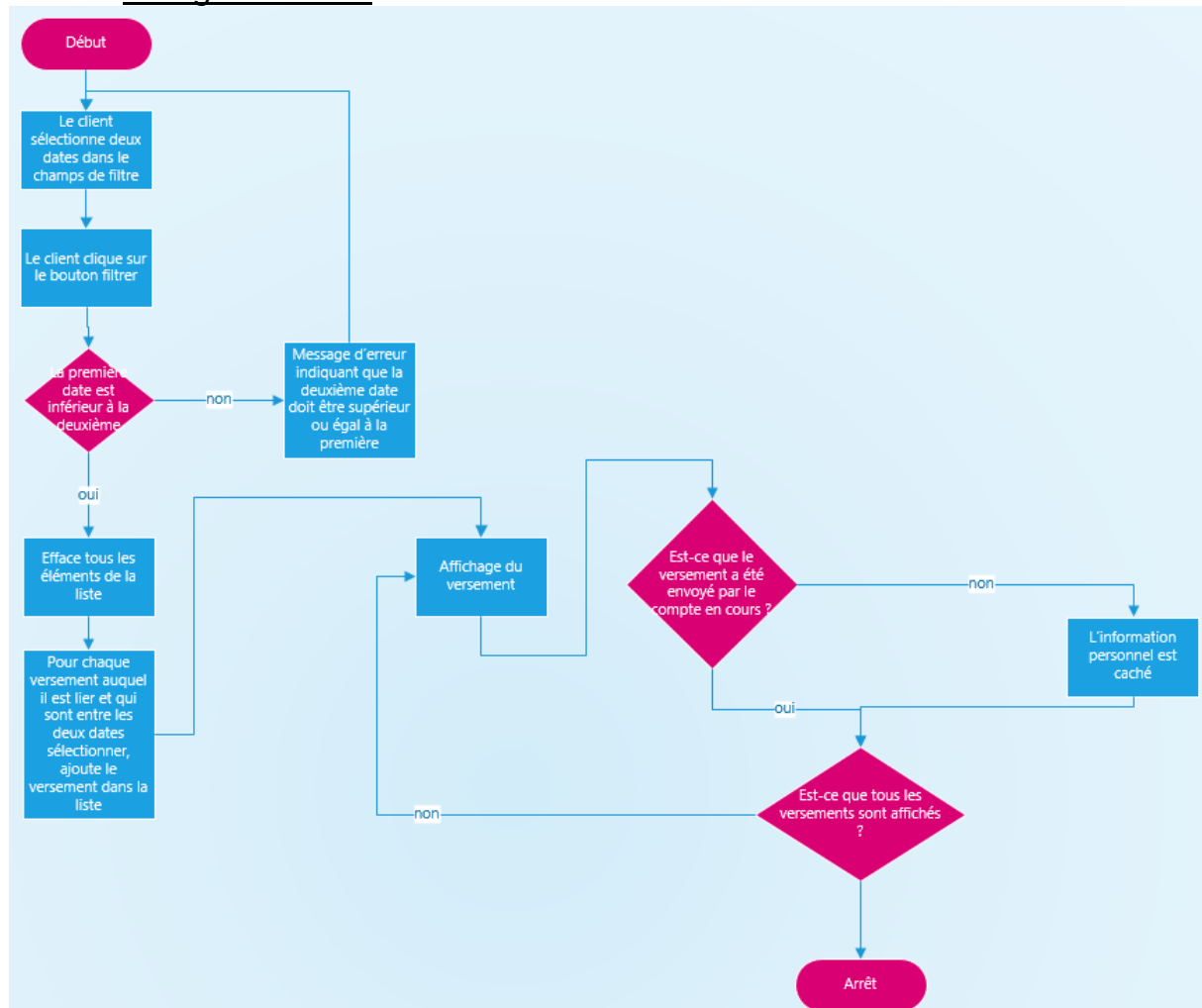
3.1.5.2 Console



3.1.5.3 Affichage de la liste des opérations



3.1.5.4 Filtrage de la liste



3.2 Risque Techniques

Le plus grand problème serait la partie console où je ne que très peu travaillé dessus donc il faudra que je me documente sur son utilisation.

3.3 Dossier de réalisation

3.3.1 Logiciels utilisés

Pour la réalisation de ce projet, j'ai utilisé visual studio Entreprise 2019 16.4.2

Pour l'écriture de la documentation du projet, j'ai utilisé office 2016.

3.3.2 Arborescence du dépôt GitHub

Dans l'arborescence de mon dépôt, vous trouverez les documents suivants :

.git	01.06.2021 15:58	Dossier de fichiers
Build	03.06.2021 08:07	Dossier de fichiers
Code	10.05.2021 08:25	Dossier de fichiers
Documentation	03.06.2021 08:02	Dossier de fichiers
script sql	06.05.2021 09:28	Dossier de fichiers

- Le dossier git avec les informations du répertoire
- Le dossier documentation où vous trouverez des versions modifiables de toute la documentation du projet.
- Le dossier code contenant le projet C#.
- Le dossier Build contient la solution finale du projet

Dans le dossier code, la structure est divisée en trois parties :

- View qui la partie visuelle de l'application
- Model qui est la partie business logic de l'application
- ConsoleMod qui est la partie console de l'application

3.3.3 Programmation et scripts

3.3.3.1 *Sprint 2*

Pour la partie versement j'ai créé une méthode qui est tout aussi valable pour la view que pour la console

```
public bool addPayment(Account activeAccount, int idAccountRecipient, DateTime datePayment, decimal amount, string informationSent, string personalInformation)
{
    ApplicationSettings settings = JsonSerializer.ReadAppSettings();
    DbConnector dbConnector = new DbConnector(settings.ConnectionString);
    string querySelect = "SELECT ID from accounts WHERE AccountNumber = " + "" + this.accountRecipient + "";

    List<List<object>> queryResultSelect = dbConnector.Select(querySelect);
    if (queryResultSelect.Count == 1)
    {
        this.idAccountRecipient = Convert.ToInt32(queryResultSelect[0][0]);
        string query = "INSERT INTO payments(`FkIDAAccountOwner`,`FkIDAAccountRecipient`,`Amount`,`DatePay`,`InformationTransmitted`,`PersonalInformation`) VALUES (" + this.idAccountOwner + "," + this.idAccountRecipient + "," + amount + "," + datePayment + "," + informationSent + "," + personalInformation + ")";
        bool queryResult = dbConnector.Insert(query);
        if (queryResult == false)
        {
            return false;
        }
        else
        {
            string querySelectOwner = "SELECT Amount from accounts WHERE AccountNumber = " + "" + activeAccount.AccountNumber + "";
            List<List<object>> queryResultSelectOwner = dbConnector.Select(querySelectOwner);
            if (queryResultSelectOwner.Count == 1)
            {
                decimal amountFinalOwner = Convert.ToDecimal(queryResultSelectOwner[0][0]) - amount;
                string queryUpdate = "UPDATE accounts SET Amount = " + amountFinalOwner + " WHERE ID = " + activeAccount.IdAccount.ToString();
                bool queryResultUpdate = dbConnector.Update(queryUpdate);
                if (queryResultUpdate == true)
                {
                    string querySelectRecipient = "SELECT Amount from accounts WHERE AccountNumber = " + "" + accountRecipient + "";
                    List<List<object>> queryResultSelectRecipient = dbConnector.Select(querySelectRecipient);
                    if (queryResultSelectRecipient.Count == 1)
                    {
                        decimal amountFinalRecipient = Convert.ToDecimal(queryResultSelectRecipient[0][0]) + amount;
                        string queryUpdateRecipient = "UPDATE accounts SET Amount = " + amountFinalRecipient + " WHERE ID = " + this.idAccountRecipient;
                        bool queryResultUpdateRecipient = dbConnector.Update(queryUpdateRecipient);
                        if (queryResultUpdateRecipient == true)
                        {
                            return true;
                        }
                    }
                }
            }
        }
    }
}
```

Il devait aussi il y avoir une connexion de compte donc j'ai fait une methode qui vérifie l'utilisateur connecté et qui sélectionne son compte pour qu'il soit le compte actif

```
public bool loadAccount(int idAccount, string accountNumber, decimal amount)
{
    ApplicationSettings settings = JsonSerializer.ReadAppSettings();
    DbConnector connector = new DbConnector(settings.ConnectionString);
    string accountQuery = "SELECT ID, AccountNumber, Amount FROM accounts WHERE FkID = " + ActiveUser.IdUser ;

    List<List<object>> queryResult = connector.Select(accountQuery);

    if (queryResult.Count == 1)
    {
        this.idAccount = Convert.ToInt32(queryResult[0][0]);
        this.accountNumber = queryResult[0][1].ToString();
        this.amount = Convert.ToDecimal(queryResult[0][2]);
        return true;
    }
    else { return false; }
}
```

Pour la partie console, puisqu'il n'y a pas de compte connecté j'ai dû faire deux méthodes qui permettait de trouver le compte du vendeur

```
public bool findAccount(int idAccount, string accountNumber, decimal amount, int idUser)
{
    ApplicationSettings settings = JsonSerializer.ReadAppSettings();
    DbConnector connector = new DbConnector(settings.ConnectionString);
    string accountQuery = "SELECT ID, AccountNumber, Amount, FkID FROM accounts WHERE AccountNumber = " + accountNumber + " ";

    List<List<object>> queryResult = connector.Select(accountQuery);

    if (queryResult.Count == 1)
    {
        this.idAccount = Convert.ToInt32(queryResult[0][0]);
        this.accountNumber = queryResult[0][1].ToString();
        this.amount = Convert.ToDecimal(queryResult[0][2]);
        this.idUser = Convert.ToInt32(queryResult[0][3]);
        return true;
    }
    else { return false; }
}
```

Et le compte du client

```
public bool findAccountById(int idAccount, string accountNumber, decimal amount)
{
    ApplicationSettings settings = JsonSerializer.ReadAppSettings();
    DbConnector connector = new DbConnector(settings.ConnectionString);
    string accountQuery = "SELECT ID, AccountNumber, Amount, FkID FROM accounts WHERE ID = " + idAccount;

    List<List<object>> queryResult = connector.Select(accountQuery);

    if (queryResult.Count == 1)
    {
        this.idAccount = Convert.ToInt32(queryResult[0][0]);
        this.accountNumber = queryResult[0][1].ToString();
        this.amount = Convert.ToDecimal(queryResult[0][2]);
        this.idUser = Convert.ToInt32(queryResult[0][3]);
        return true;
    }
    else { return false; }
}
```

3.3.3.2 Sprint 3

Pour la partie affichage j'ai dû utiliser deux méthodes une pour l'affichage normal

```
public bool displayPayment(Account activeAccount)
{
    string currentMonth = DateTime.Now.ToString("MM");

    allPayments = new List<Payment>();

    ApplicationSettings settings = JsonDataSaverReader.ReadAppSettings();
    DbConnector dbConnector = new DbConnector(settings.ConnectionString);

    string query = "SELECT * FROM payments WHERE DatePay LIKE '%" + currentMonth + "%' AND (FkIDAccountOwner = " + activeAccount.IdAccount + " OR FkIDAc";

    List<List<object>> queryResult = dbConnector.Select(query);
    if (queryResult.Count >= 1)
    {
        foreach (List<object> row in queryResult)
        {
            int idPayment = Convert.ToInt32(row[0]);
            int activeAccountId = Convert.ToInt32(row[1]);
            string accountRecipient = row[2].ToString();
            decimal amount = Convert.ToDecimal(row[3]);
            DateTime datePayment = (DateTime)row[4];
            string informationTransmitted = row[5].ToString();
            string personalInformation = row[6].ToString();
            allPayments.Add(new Payment(idPayment, activeAccountId, datePayment, accountRecipient, amount, informationTransmitted, personalInformation));
        }
        return true;
    }
}
```

L'affichage filtrer

```
public bool displayPaymentSort(Account activeAccount, DateTime firstDate, DateTime lastDate)
{
    firstDate = firstDate.Date;
    lastDate = lastDate.Date.AddDays(1).AddSeconds(-1);

    allPayments = new List<Payment>();

    ApplicationSettings settings = JsonDataSaverReader.ReadAppSettings();
    DbConnector dbConnector = new DbConnector(settings.ConnectionString);

    string query = "SELECT * FROM payments WHERE (FkIDAccountOwner = " + activeAccount.IdAccount + " OR FkIDAccountRecipient = " + activeAccount.IdAccount + ")";

    List<List<object>> queryResult = dbConnector.Select(query);
    if (queryResult.Count >= 1)
    {
        foreach (List<object> row in queryResult)
        {
            int idPayment = Convert.ToInt32(row[0]);
            int activeAccountId = Convert.ToInt32(row[1]);
            string accountRecipient = row[2].ToString();
            decimal amount = Convert.ToDecimal(row[3]);
            DateTime datePayment = (DateTime)row[4];
            string informationTransmitted = row[5].ToString();
            string personalInformation = row[6].ToString();
            allPayments.Add(new Payment(idPayment, activeAccountId, datePayment, accountRecipient, amount, informationTransmitted, personalInformation));
        }
        return true;
    }
}
```

3.3.3.3 Sprint 4

Pour la connexion d'un utilisateur, j'ai créé deux fonctions, une pour la vérification de l'utilisateur et de son mot de passe :

```
public bool Login(string userName, string userPassword)
{
    ApplicationSettings settings = JsonSerializer.ReadAppSettings();
    DbConnector connector = new DbConnector(settings.ConnectionString);
    string loginQuery = "SELECT HashPassword, ID FROM users WHERE Name = '" + userName + "'";

    List<List<object>> queryResult = connector.Select(loginQuery);

    if (queryResult.Count == 1)
    {
        List<object> row = queryResult[0];
        string userHashPsw = row[0].ToString();
        string hashedInputPsw = this.Hash(userPassword);

        if (hashedInputPsw == userHashPsw)
        {
            this.userName = userName;
            this.idUser = int.Parse(queryResult[0][1].ToString());
            return true;
        }
        else
        {
            return false;
        }
    }
    else { return false; }
}
```

Et l'autre est par la traduction du mot de passe hasher dans la DB

```
private string Hash(string dataToHash)
{
    using (SHA256 sha256Hash = SHA256.Create())
    {
        //hash the data
        byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(dataToHash));

        //convert the byte array to string
        StringBuilder builder = new StringBuilder();
        for (int i = 0; i < bytes.Length; i++)
        {
            builder.Append(bytes[i].ToString("x2"));
        }
        return builder.ToString();
    }
}
```

3.4 Description des tests effectués

3.4.1 Connection Base de données

Ce test permet de vérifier la connexion à la base de données.

Si la connexion est impossible l'application affiche un message disant que la DB n'est pas opérationnelle.

Si la connexion est possible alors elle laisse avancé.

3.4.2 Connexion avec un compte

Ce test permet de vérifier si un utilisateur peut se connecter avec son compte à la DB.

Si la connexion est réussie, cela retourne un bool True.

Si la connexion a échoué, cela retourne un bool False.

3.4.3 Création d'un versement

Ce test permet de vérifier si le versement est valable pour la DB.

Si l'insertion est réussie, cela retourne un bool True.

Si l'insertion a échoué, cela retourne un bool False.

3.4.4 Trouver un compte par son numéro de compte

Ce test permet de vérifier si le compte écrit existe dans la DB par son numéro

Si le select est réussi, cela retourne un bool True.

Si le select a échoué, cela retourne un bool False.

3.4.5 Trouver un compte par son id

Ce test permet de vérifier si le compte écrit existe dans la DB via son ID

Si le select est réussi, cela retourne un bool True.

Si le select a échoué, cela retourne un bool False.

3.4.6 Versement :

L'utilisateur Paul s'est connecté à son compte dont le numéro est CA-456456.

Il souhaite faire un versement à son ami anis dont son numéro est CA-455757.

Donc, depuis la page principale, il clique sur le bouton faire un versement et il est redirigé sur le formulaire de versement. Dans cette fenêtre, il devra remplir deux champs obligatoires ou alors le formulaire n'est pas valide et deux autres qui sont facultatifs. Dans les champs obligatoires, il y a le compte du destinataire qui doit être dans la base de données sinon le formulaire n'est pas valide et renverra une erreur indiquant que le compte n'existe pas et le montant du versement qui doit être inférieur ou égal à la valeur du montant de Paul sinon le versement n'est pas valide et un message s'affichera indiquant que le montant est au-dessus du solde de Paul. Paul pourra enregistrer son versement qui sera enregistré dans la Base de données et les soldes des comptes seront changés.

Paul pourra à ton moment, dans le formulaire, annuler le versement et retourner dans la page principale.

Given	When	Then
Le client est sur la page principale	Le client clique sur le bouton « Faire un versement »	Le client arrive sur le formulaire de versement
Le client est dans le	Le client remplir les	Le versement est accepté

formulaire de versement	champs : Numéro du destinataire, Montant du versement	et la envoyé dans la DB
Le client est dans le formulaire de versement	Le client écrit un montant dans le versement qui est au-dessus de son solde	Le versement n'est pas accepté et l'application renvoi un message d'erreur qui indique que le solde est insuffisant
Le client est dans le formulaire de versement	Le client écrit le numéro de compte d'un client qui n'existe pas	Le versement est refusé et l'application renvoi un message d'erreur qui indique que le client qui doit recevoir le montant n'existe pas
Le client est dans le formulaire de versement	Le client clique sur le bouton « annuler »	Le versement est annulé et le client est ramené sur la page principale
Le client est dans le formulaire de versement	Le client remplir les champs : Numéro du destinataire, Montant du versement, information transmise au destinataire, Remarque personnelle non transmise et clique sur le bouton « enregistrer »	Le versement est accepté et la envoyé dans la DB
Le client est dans le formulaire de versement	Le client écrit son numéro de compte comme destinataire	Le versement est refusé en renvoi un message d'erreur indiquant que le numéro de compte n'est pas accepté

3.4.7 Module console :

Le vendeur écrit les données qu'il veut faire passer dans la console par exemple : VA-694207 (un compte de vendeur) CA-456456 (un compte client, celui de Paul) 100 (le montant que le client doit au vendeur). En écrivant ces données et en exécutant la console, il y aura un versement de 100 CHF sur le compte VA-694207 par CA-456456. Ces données sont obligées d'être inscrit sinon la console ne s'exécutera pas. Il est obligé d'écrire un premier le compte d'un vendeur et en deuxième un compte de client sinon le console reportera une erreur indiquant que ce n'est pas un compte de vendeur valide. Pareillement si le numéro de compte n'existe pas. Si le montant est trop élevé par rapport au solde du client indiqué alors, un message d'erreur s'affichera qui dira que le montant n'est pas valide. Les transactions dans le console peuvent se faire en même temps qu'un client fait un versement sur le formulaire.

Given	When	Then
Le vendeur est dans la console	Le vendeur écrit un numéro de compte ainsi que celui d'un client et le montant à payer	La transaction est acceptée et les changements ont été faits dans la DB et la console renvoie un message qui dit que le versement a été accepté
Le vendeur est dans la console	Le vendeur écrit toutes les données mais le client n'existe pas	La transaction n'est pas acceptée et la console renvoie un message qui dit que le client n'est pas valable
Le vendeur est dans la console	Le vendeur écrit toutes les données mais le solde du client n'est pas assez élevé pour le montant écrit	La transaction n'est pas acceptée et la console renvoie un message qui dit que le montant n'est pas valide
Le vendeur utilise la console et le client utilise le formulaire	Le vendeur envoie sa requête avant le client ce qui change le montant du client	Le montant sera adapté pour le client donc si le montant de sa requête est devenu trop faible pour son solde alors elle ne sera pas acceptée
Le vendeur utilise la console et le client utilise le formulaire	Le client envoie sa requête avant le vendeur ce qui change le montant du vendeur	Le montant sera adapté pour le vendeur donc si le montant de sa requête est devenu trop faible pour le solde alors elle ne sera pas acceptée

3.4.8 Affichage des opérations :

Le client Paul vient de se connecter à son compte donc il est redirigé sur la page principale. Depuis ici, il pourra voir la liste de toutes les opérations dont il est lié. Si une des opérations a été envoyée par Paul, il pourra voir l'information personnel qui a été assignée et le destinataire ne pourra pas la voir. Paul pourra choisir de rafraîchir la liste des opérations. Ceci permettra à Paul de pouvoir afficher le versement qu'il vient de faire ou alors de pouvoir voir aussi la transaction qu'un vendeur venait juste de faire. Il pourra aussi filtrer sa liste grâce aux des champs date ce qui affichera tous ses versements qui ont été faits entre ces dates.

Given	Then	When
Le client veut voir sa liste des opérations	Le client va sur la page principale	Le client voit la liste des opérations qu'il a fait dans ce mois avec leurs données affichées (Destinataire, Envoyeur, Montant, Date, information transmise et l'information personnel) et seulement celle du mois courant

Le client est dans la page principale et le vendeur fait une transaction et le vendeur fait une transaction avec ce client dans la console	Le client clique sur le bouton « rafraichir la page »	Le client voit sa liste des opérations qui a été mise à jour avec le nouveau versement
Le client est dans la page principale	Le client choisi une date de début et une date de fin et clique sur le bouton « filtrer »	La liste des opérations affichera toutes les dates correspondant aux dates du filtre
Le client est dans la page principale	Le client choisi une date de début plus grande que celle de date de fin et clique sur le bouton « filtrer »	L'application renvoi un message d'erreur indiquant qu'il y a une erreur dans les champs filtre
Le client est dans la page principale	Le client indique la même date dans les deux champs et clique sur le bouton « filtrer »	La liste des opérations affichera seulement les opérations de la date indiqué

3.4.9 Login

Pour la connexion à l'application, Paul devra rentrer son nom ainsi que son mot de passe. Son mot de passe est hasher en SHA 256 dans la DB.

Given	When	Then
Le client est sur la page de connexion	Il y écrit son identifiant et son mot de passe et clique sur « se connecter »	Le client est connecté à l'application et est redirigé sur la page principale
Le client est sur la page de connexion	Le client écrit un mauvais mot de passe lors sa tentative de connexion	L'application renvoi un message d'erreur indiquant que le compte n'existe pas ou le mot de passe est incorrecte
Le client est sur la page de connexion	Le client écrit un mauvais identifiant lors de sa tentative de connexion	L'application renvoi un message d'erreur indiquant que le compte n'existe pas ou le mot de passe est incorrecte
Le client est sur la page principale de connexion	Le client clique sur le bouton « quitter »	L'application se ferme

4 Mise en service

4.1 Installation

Pour la connexion de l'application à la base de données, vous devrez en premier exécuter les scripts sql qui créeront la base de données ainsi qu'un utilisateur pour la DB. Il y a aussi le script CreateUserAccount qui crée les utilisateurs avec leurs comptes. Les mots de passes de tous les utilisateurs doivent être hasher en SHA256. Les mots de passe de ces utilisateurs sont 1234. Vous pourrez aussi modifier si besoin, le fichier applicationParameter.Json qui contient les données de connexion à la DB pour la code.

4.2 Liste des documents fournis

Comme demandé dans mon cahier des charges, les documents fournis et autres livrable sont les suivants :

- Une planification initial (disponible plus dans le rapport)
- Un rapport de projet
- Un journal de travail (livré deux fois par semaine)
- Le code source de l'application (fournis via un dépôt GitHub)
- L'exécutable de l'application

5 Conclusions

5.1 Objectifs atteints ?

Tous les objectifs du cahier des charges ont été atteints.

5.2 Points positifs / négatifs

5.2.1 Négatif

En terme de graphique, l'application n'est pas belle et ni attirante. Mais cela reste une simulation d'E-Banking.

J'ai pris un peu plus de temps que prévu sur le versement d'un client alors que j'avais déjà fait ce type de fonction auparavant.

5.2.2 Positif

J'ai pu rattraper le peu de retard que j'avais.

J'ai fait beaucoup de retour avec mon chef de projet ce qui m'a permis d'être sûr à chaque fois de ce qui était demandé.

Je pensais que la console allait me prendre plus de temps alors que j'ai réussi à vite la finir.

5.3 Difficulté particulières

La plupart du temps ma principale difficulté était la gestion de requête. J'ai passé plus de temps là-dessus car je l'ai réécrit souvent.

5.4 Suite pour le projet

Je pense que la principale amélioration à faire serait pour la connexion de l'utilisateur, où je pourrai essayer d'implémenter une double authentification.

6 Annexes

6.1 Sources – Bibliographie

Ce lien m'a permis de faire la lecture du mot de passe Hasher.

<https://www.c-sharpcorner.com/article/compute-sha256-hash-in-c-sharp/>

6.2 Journal de bord

Date	Durée	Activité	Remarques
04.juin	4.5	Finition de la documentation et Rendu Final	
03.juin	6.75	Continuation de la documentation	
01.juin	0.25	Entretien professionnel	J'ai dû faire un entretien pendant les heures de TPI
01.juin	1.50	Diagramme de classe	
01.juin	2.75	Continuation de la documentation	
31.mai	4.50	Continuation de la documentation	
31.mai	1.50	Coder le login et la lecture du mot de passe hash	
28.mai	0.75	Continuation de la documentation	
28.mai	1.00	Corriger une erreur de la console	Cette erreur faisait que l'on pouvait lancer l'exécutable de la console sans aucun argument ce qui la faisait s'arrêter
28.mai	1.00	Création diagramme de flux pour la console	
28.mai	1.00	Finition de l'implémentation du filtrage	
27.mai	2.50	Coder la fonction de filtrage de la liste des opérations	
27.mai	0.50	Changement de la maquette HomePage	
25.mai	1.50	Corriger les problèmes de versements	Les utilisateurs clients pouvaient écrire le numéro de compte dans le formulaire de versement et pouvait mettre un montant négatif
25.mai	0.25	Visite de Monsieur Hurni	Voir l'avancement du projet et test des fonctionnalités
25.mai	0.50	Coder l'actualisation de la liste des opérations	

25.mai	1.75	Coder l'affichage des versements du compte lier	
25.mai	0.25	Création des tâches dans icescrum pour le sprint 3	
25.mai	0.25	Modification de la maquette du la HomePage	
21.mai	2.25	Continuation de la documentation	
21.mai	3.00	Finition de la console	Il y a eu un petit problème sur la gestion des versements où si la console passait sa requête pendant que le client faisait la sienne. Même si le client n'avait plus d'argent pouvait quand-même passer son versement donc il se retrouvait avec un montant négatif
20.mai	6.25	Continuation du module console	Il y a eu un changement pendant la programmation de cette partie car je n'avais pas fait que la console obtienne les arguments. Je codais la console comme si c'était pour un utilisateur et qu'il puisse écrire ce qu'il voulait. Cela a été corrigé
20.mai	0.50	Visite de Monsieur Hurni	
18.mai	1.00	Commencement du module console	
18.mai	1.75	Fonction qui permet de prendre un compte du client à partir de l'utilisateur connecté	
18.mai	0.25	Visite de Monsieur Hurni	
18.mai	1.50	Finition de la fonction de versement et de ses tests	
17.mai	0.75	Visite de l'expert	
17.mai	3.25	Continuation de la fonction de versement	
17.mai	1.00	Changement du nommage de la base de données	
17.mai	1.00	Continuation de la documentation/écriture du début du scénario de test pour le versement	
11.mai	1.00	Modification de la base de données	
11.mai	0.50	Création des tâches sur Icescrum pour le sprint 2	
11.mai	0.50	Création de la HomePage qui redirige le client sur le formulaire de versement	

11.mai	1.00	Création de la class User	
11.mai	2.25	Continuation de la fonction versement	
11.mai	0.25	Création de la class Account	
11.mai	0.50	Mettre la référence du model à la view	J'ai eu un problème où le model ne voulait référencer la view car il avait un problème de compatibilité
10.mai	0.75	Changement du MCD et MLD	Changement des types des champs
10.mai	0.75	Visite de Monsieur Hurni	Nous avons fait une review du sprint 1
10.mai	2.50	Création classe PaymentManager	
10.mai	0.75	Création du formulaire de versement	
10.mai	0.50	Création Diagramme de séquence versement	
10.mai	0.25	Création de la view Versement	
10.mai	0.50	Création de la class DbConnector	
07.mai	0.75	Création du projet sur visual studio	J'ai eu un problème de licence donc je ne pouvais plus lancer l'application mais j'ai pu en trouver une nouvelle
07.mai	4.00	Continuation de la documentation	
07.mai	0.50	Correction MLD	
06.mai	0.75	Création des sprints sur icescrum	
06.mai	0.50	Modification de la base de données	
06.mai	1.00	Modification du planning IceScrum	
06.mai	0.25	Visite de Monsieur Hurni	Nous avons vu le planning Icescrum, le MCD et MLD et la base de données
06.mai	1.50	Avancement dans la documentation	
06.mai	0.75	Création des Diagrammes UML	
06.mai	0.50	Coder les scripts pour instaurer une base de données	
06.mai	0.75	Modification MCD et MLD	
06.mai	0.75	Création des maquettes	
04.mai	2.50	écriture des tests dans les stories du planning icescrum	
04.mai	0.50	Correction du planning initial	Monsieur Mveng m'a envoyé un email pour m'avertir que je n'avais pas spécifié que je ne travaillais pas sur le projet les

			mercredis.
04.mai	1.50	Création MCD et MLD	
03.mai	1.00	Création documentation du rapport de projet	
03.mai	1.00	Création du planning IceScrum	
03.mai	3.00	Création planification initiale et du dépôt github	
03.mai	1.00	Analyse et signature du cahier des charges	

6.3 Manuel d'utilisation

En premier, vous devrez vous connecter avec un utilisateur donc un qui est dans la base de données ou alors l'application retournera une erreur. Arrivé dans l'application vous trouverez la liste des opérations de cet utilisateur, son solde ainsi que des champs date qui vont vous permettre de filtrer la liste. Il y a aussi un bouton « Faire un versement » qui permet d'ajouter une opération.

Celle-ci se présentera comme un formulaire où vous devrez remplir avec un numéro de compte d'un autre client un montant qui est inférieur ou égal à celui de votre compte. Vous pourrez aussi écrire une information que vous voudrez transmettre à ce compte et une information personnel qui ne s'affichera seulement dans votre liste des opérations mais ces deux données sont facultatives.

Vous aurez aussi la partie console qui elle devra recevoir trois informations pour l'exécuté. Le numéro d'un compte vendeur (un compte qui commence par « VA ») qui existe dans la DB, un numéro de compte client et un montant qui est inférieur ou égal à celui du client.

6.4 Résumé de la documentation

Le résumé de la documentation se trouve dans le dossier documentation :
ResumueDocuementation.pdf