



## Dossier commun

### *Projet : Roofkit*



Élèves : Allan Defrenne et Kévin Once

# Sommaire

## 1- présentation du projet

1-1 Introduction	2
1-2 situation du projet dans son contexte	2
1-2-1 A qui le projet est-il destiné ?	2
1-2-1-1 Présentation d'Attila Système :	2
1-2-1-2 Activité d'Attila système :	3
1-2-2 pourquoi ce projet ?	5
1-2-3 le cahier des charges	6
1-3 l'architecture matériel	6
1-4 fonctionnalité offerte par le projet	7
1-5 Composition de l'équipe de projet	9

## 2 Conception préliminaire

2-1 Diagramme des cas d'utilisation	10
2-2 Description de l'acteur et des cas d'utilisation	10
2-2-1 Les acteurs :	10
2-2-2 : cas d'utilisation "éditer un devis"	11
2-2-3 : cas d'utilisation "Modifier un devis"	11
2-2-4 : cas d'utilisation "imprimer un devis"	11
2-2-5 : cas d'utilisation "sauvegarder un devis"	11
2-2-6 : cas d'utilisation "rechercher un devis"	11

## 3 solution envisagée

3-1 schéma de l'infrastructure de l'application :	12
3-2 scénario	13
3-3 schéma conceptuel de la base de donnée	14
3-3-1 description des tables	15

## 4 conception détaillée

4-1 la classe Commune : classe BDD	21
4-2 travail personnel : Once Kevin	22
4-2-1 : La page connexion	22
4-2-1-1 : description	22
4-2-1-2 :diagramme de séquence	23
4-2-1-3 : Test unitaire	25

4-2-2 : module d'édition devis	30
4-2-2-1 Page nouvelle toiture	32
4-2-2-1-1 : description	32
4-2-2-1-2 : test unitaire	33
4-2-2-2 : La page Récapitulatif:	40
4-2-2-2-1 : description	40
4-2-2-2-3 : test unitaire	40
4-2-2-3 : La Page FormulaireOpération	42
4-2-2-3-1 : description	42
4-2-2-3-3 : test unitaire	42
4-2-3 : module modifier devis	44
4-2-4 : Conclusion	45

#### 4-3 travail personnel : Defrenne Allan

4-3-1-Page d'accueil	46
4-3-1-1-Présentation	46
4-3-1-2-Description de la classe « WFAccueil »	46
4-3-1-3-Test unitaire	46
4-3-2-Page client	48
4-3-2-1-Présentation	48
4-3-2-2-Description de la classe « WFClient »	48
4-3-2-3-Test unitaire	49
4-3-3-Page de Recherche	54
4-3-3-1-Présentation	54
4-3-3-2-Description de la classe « WFRechercher »	54
4-3-3-3-Diagramme de séquence	56
4-3-3-4-Test unitaire	58
4-3-4-Conclusion	64

#### 5 Annexes

5-1 : Devis d'Attila système	65
5-2 : Code de la méthode PremierOperation de la page récapitulatif.	67
5-3 : Code de la méthode logmdp de la page de Connection.	70
5-4 : Code de la méthode ValidOp de la page FormulaireOperation.	71
5-5 : Code de la méthode DonneChantier de la page NouvelleToiture.	72
5-6 : Code de la méthode CalculDimension de la page NouvelleToiture.	74

# 1-Présentation du projet

## 1-1-Introduction

La seconde année de BTS IRIS comprends un projet en équipe permettant aux étudiants de mettre en application dans un contexte réel les modules logiciels et matériels étudié en cours et travaux pratique. L'objectif du projet ici est de permettre une gestion des devis plus rapide via une application web.

## 1-2-Situation du projet dans son contexte

### 1-2-1 A qui le projet est-il destiné ?

#### 1-2-1-1 Présentation d'Attila Système :

Le projet est destiné aux franchisés d'Attila Système qui sont répartis dans 23 franchises en France dans différentes villes.

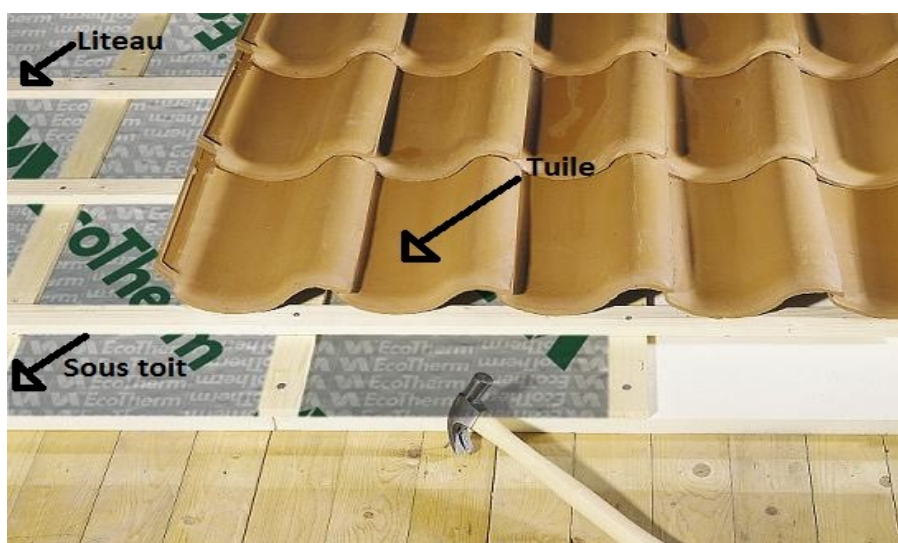
Alluets le roi, Ancenis, Arras, Auxerre, Billy Berclau Wingles, Brest est, Calais, Clermont Ferrand, Cote Basque, Evry, Limoges, Lyon nord, Lyon sud, Massy, Montargis, Nantes ouest, Nimes, Orleans, Rennes, Roissy, Rouen nord, Troyes et Villefranche.

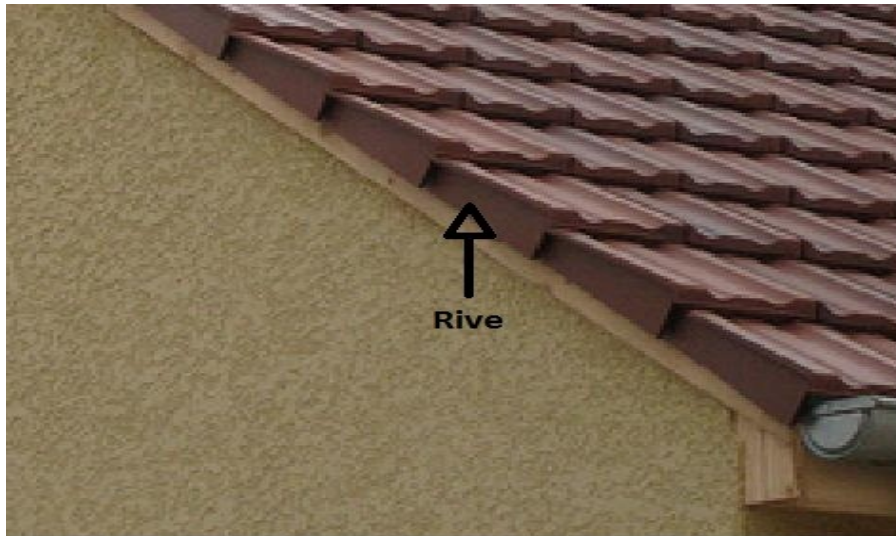


### 1-2-1-2 Activité d'Attila système :

Attila Système est spécialisé dans la création, l'entretien et la réparation de toitures. Dans le cas d'une création de toiture ils sont chargés de la couverture de la maison, la charpente n'est pas à leurs charge.

La création d'une toiture nécessite plusieurs matériaux dont les principaux sont : le sous-toit, les liteaux et les tuiles, à cela s'ajoute les éléments de finitions qui sont : les faîtières et les rives pour une forme simple (exemple toiture Rectangulaire à deux faces), voici un schéma illustrant ces principaux matériaux et leur emplacement dans une toiture :





Dans le cas d'un entretien, Attila système sont chargés de nettoyer les éléments de la toiture telle que les tuiles, les gouttières, cela peut concerner la suppression de mousse sur les tuiles ou le débouchages de gouttière par exemple.

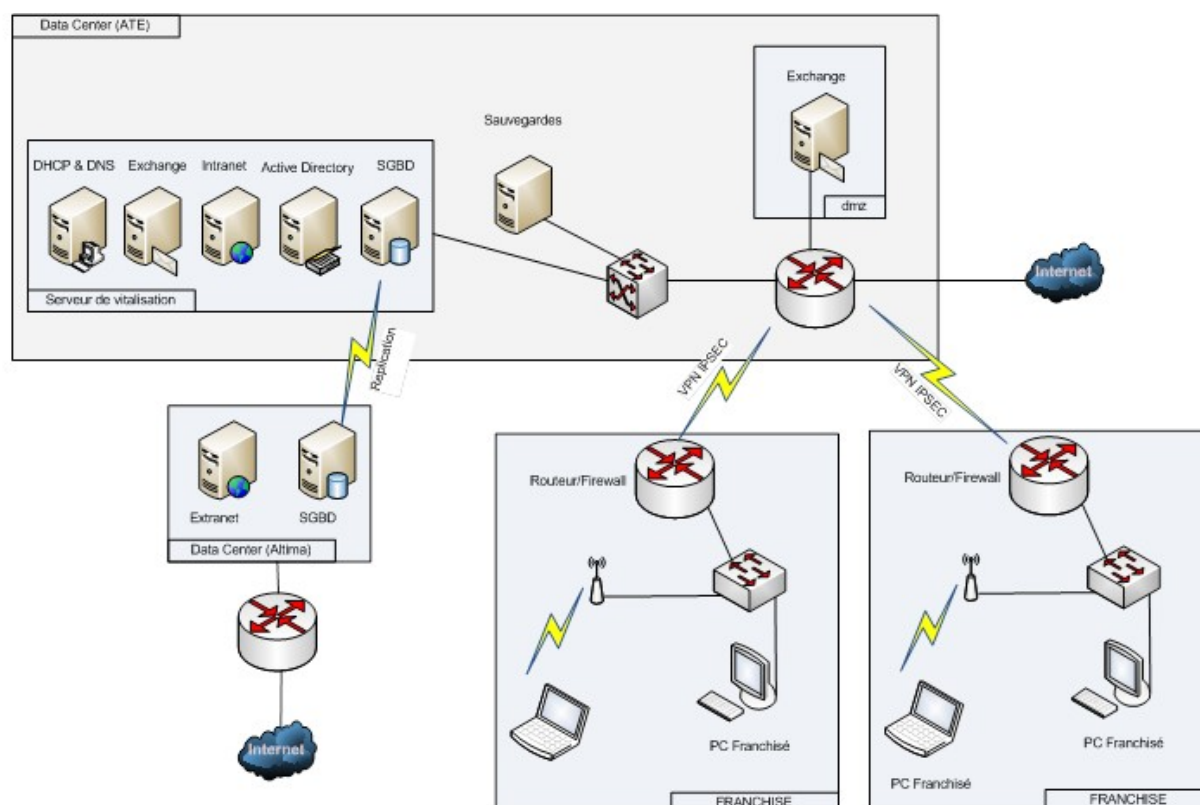
Dans le cas d'une Réparation, cela concerne la réparation de dégâts mineurs sur une toiture, comme par exemple le remplacement de tuiles cassées ou fissurées ou le changement d'une gouttière abîmée.

Et pour finir sur l'activité de l'entreprise, dans le cas d'une Rénovation, les franchisés d'Attila Système détruisent l'ancienne toiture et en créent une nouvelle, c'est le même fonctionnement que le cas de création d'une toiture avec les frais de destruction de l'ancienne en plus.



## 1-2-2 Pourquoi ce projet ?

En 2012, dix nouvelles agences doivent ouvrir pour Attila Système, devant cette augmentation d'activité Attila Système a décidé de se doter d'une nouvelle infrastructure informatique et d'une application Web qui permettra de centraliser le traitement de ses devis et ainsi gagner du temps. Le schéma ci-dessous présente cette infrastructure.



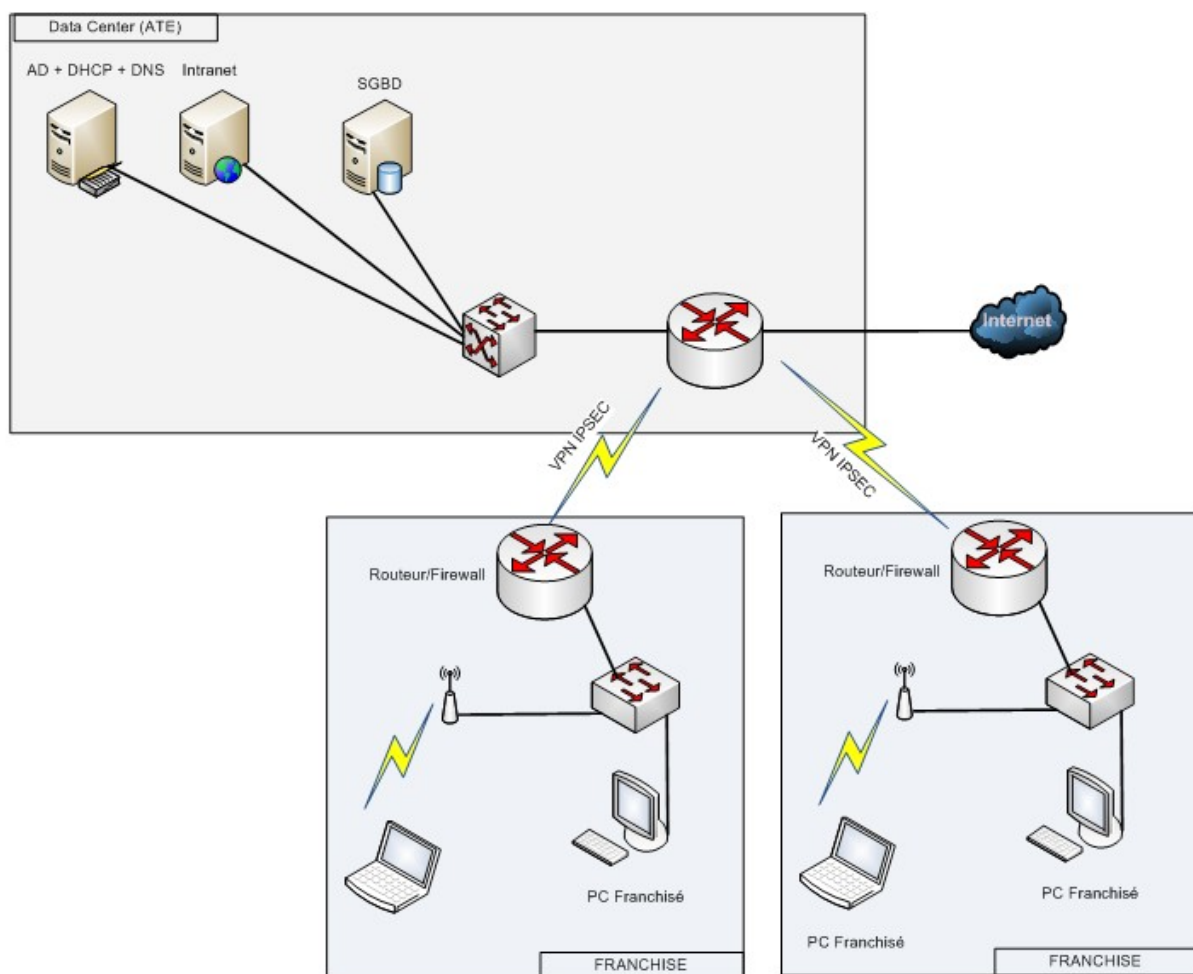
### 1-2-3-Le cahier des charges

Le cahier des charges du client montre les points importants à développer :

- Éditer un nouveau devis ;
- Valider ou modifier un devis existant ;
- Imprimer un devis.

### 1-3 L'architecture matériel

Voici l'architecture de test du projet, l'architecture de l'entreprise (présenté ci-dessus) possède plus d'éléments mais ils n'interviennent pas dans notre application web.





L'architecture générale se décompose ainsi :

- Les PC des franchisés utiliseront Windows comme système d'exploitation et Mozilla Firefox comme navigateur web.

- Le 'SGBD' est le serveur qui portera la base de donnée. Le système d'exploitation de ce serveur sera Windows XP et comme serveur SQL : SQL Serveur 2008.

-Le 'Intranet' est le serveur qui portera l'application web qui sera ici en ASP.net /C#.Le système d'exploitation sera Windows Serveur 2003 avec l'ajout du composant IIS (Internet Information Services) qui permet d'interpréter l'ASP.net .

- La liaison entre le Data Center et les différentes Franchises passera par internet de façon sécurisé via des tunnels VPN avec le protocole IPSec. Ces tunnels VPN (Virtual Private Network) permette de simuler un réseau unique avec deux réseaux distincts.

Notre application sera donc accessible de n'importe quelle franchise d'Attila système relié au Data Center. Le franchisé devra au préalable accéder à l'application intranet d'Attila système déjà en place et cliquer sur le lien de redirection vers notre application.

#### 1-4 Fonctionnalités offertes par le projet :

Les différentes fonctionnalités offerte par le projet devront être :

##### **Éditer un devis.**

Le franchisé peut éditer un nouveau devis, ce dernier peut porter sur quatre types : la rénovation , la réparation , l'entretien et la nouvelle réalisation ces types de devis correspondent aux différents types d'activités, citées plus haut, que réalise Attila système.

Cette fonctionnalité devra :

- Mettre automatiquement à jour les champs relatifs à la franchise qui édite le devis .
- Mettre automatiquement à jour le champ correspondant au collaborateur qui édite le devis .
- Générer automatiquement le numéro de devis, celui-ci est unique .
- Mettre à jour automatiquement le champ date .
- Renseigner les coordonnées du client .
- Renseigner les coordonnées du chantier (adresse, contact).
- Choisir les différentes opérations qui constituent le devis.

### **Rechercher un devis.**

Le franchisé pourra consulter tous les devis qu'il a édité , les devis pourront être triés selon les filtres suivants :

- nom de l'utilisateur.
- le nom du client.
- le numéro de devis
- la date d'édition

### **Valider un devis.**

Le franchisé pourra valider un devis , c'est a dire le rendre visible

### **Modifier un devis.**

Le franchisé pourra modifier un devis qu'il a édité c'est a dire ajouter , modifier ou supprimer des opérations.

### **Imprimer un devis.**

Le franchisé pourra imprimer le devis généré par l'application

### **Sauvegarder un devis**

Le franchisé pourra sauvegarder un devis sur sa machine au format PDF

## 1-5 Composition de l'équipe de projet :

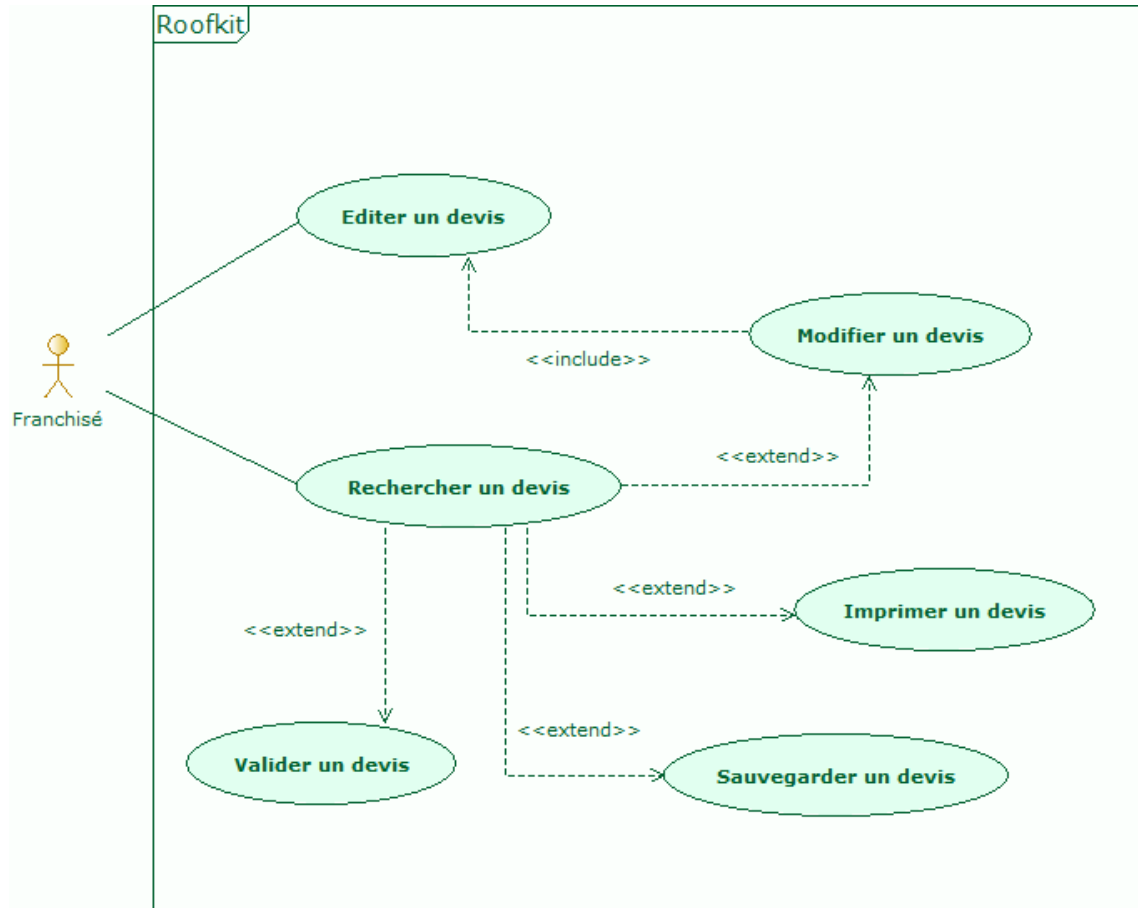
L'équipe est constituée de 2 étudiants de 2<sup>ème</sup> année de BTS IRIS :  
Once Kevin et Defrenne Allan.

Voici les différentes Tâches que nous devons réaliser :

Tâches à réaliser	Déploiement	Interaction et communication	Etudiant
<b>Structure matérielle.</b>			
Installation et configuration du serveur Intranet.	Serveur Intranet		Once Kevin
Installation et configuration du serveur SQL.	Serveur SQL		Once Kevin
Création de la base de données.	Serveur SQL		Once Kevin
Installation et configuration du serveur AD.	Serveur AD		Defrenne Allan
Installation et configuration des routeurs firewall	DFL - 800		Defrenne Allan
Création des tunnels VPN	DFL - 800		Defrenne Allan
<b>Edition de devis.</b>			
Développer et tester module logiciel : Editer les devis partie toiture	Serveur Intranet	Serveur SQL	Once Kevin
Développer et tester module logiciel : Editer les devis partie entretien	Serveur Intranet	Serveur SQL	Defrenne Allan
Développer et tester module logiciel : Editer les devis partie réparation	Serveur Intranet	Serveur SQL	Defrenne Allan
<b>Modifier un devis.</b>			
Développer module logiciel : Modifier un devis.	Serveur Intranet	Serveur SQL	Once Kevin
<b>Rechercher un devis.</b>			
Développer module logiciel : Modifier un devis.	Serveur Intranet	Serveur SQL	Defrenne Allan
<b>Sauvegarder un devis.</b>			
Développer module logiciel : Sauvegarder un devis.	Serveur Intranet	Serveur SQL	Defrenne Allan
<b>Valider un devis</b>			
Développer module logiciel : Valider un devis.	Serveur Intranet	Serveur SQL	Defrenne Allan

## 2- Conception préliminaire

### 2-1 Diagramme des cas d'utilisation :



### 2-2 Description des cas d'utilisation et de l'acteur :

#### 2-2-1 L'acteur :

Acteur	Description	Cas d'utilisation
Franchiser	Personnel d'une agence franchisée, cette personne est identifiée grâce à son login et son mot de passe (gérer par Attila Système). Le franchisé peut exécuter la totalité des opérations proposées par l'application.	-Éditer un devis. -Modifier un devis. -Imprimer un devis. -Sauvegarder un devis. -Rechercher un devis.

### 2-2-2 Cas d'utilisation « Éditer un devis » :

Ce cas d'utilisation correspond à la fonction d'usage normale de l'application. Le franchisé peut éditer un devis. Le devis peut porter sur quatre types de prestation, la réalisation d'une nouvelle toiture, la rénovation d'une toiture, la réparation d'une toiture ou l'entretien d'une toiture.

### 2-2-3 Cas d'utilisation « Modifier un devis » :

Le franchisé peut ajouter, supprimer ou modifier les opérations d'un devis existant.

### 2-2-4 Cas d'utilisation « Imprimer un devis » :

Le franchisé peut imprimer le devis sur son imprimante, le format du document est défini par Attila Système.

### 2-2-5 Cas d'utilisation « Sauvegarder un devis » :

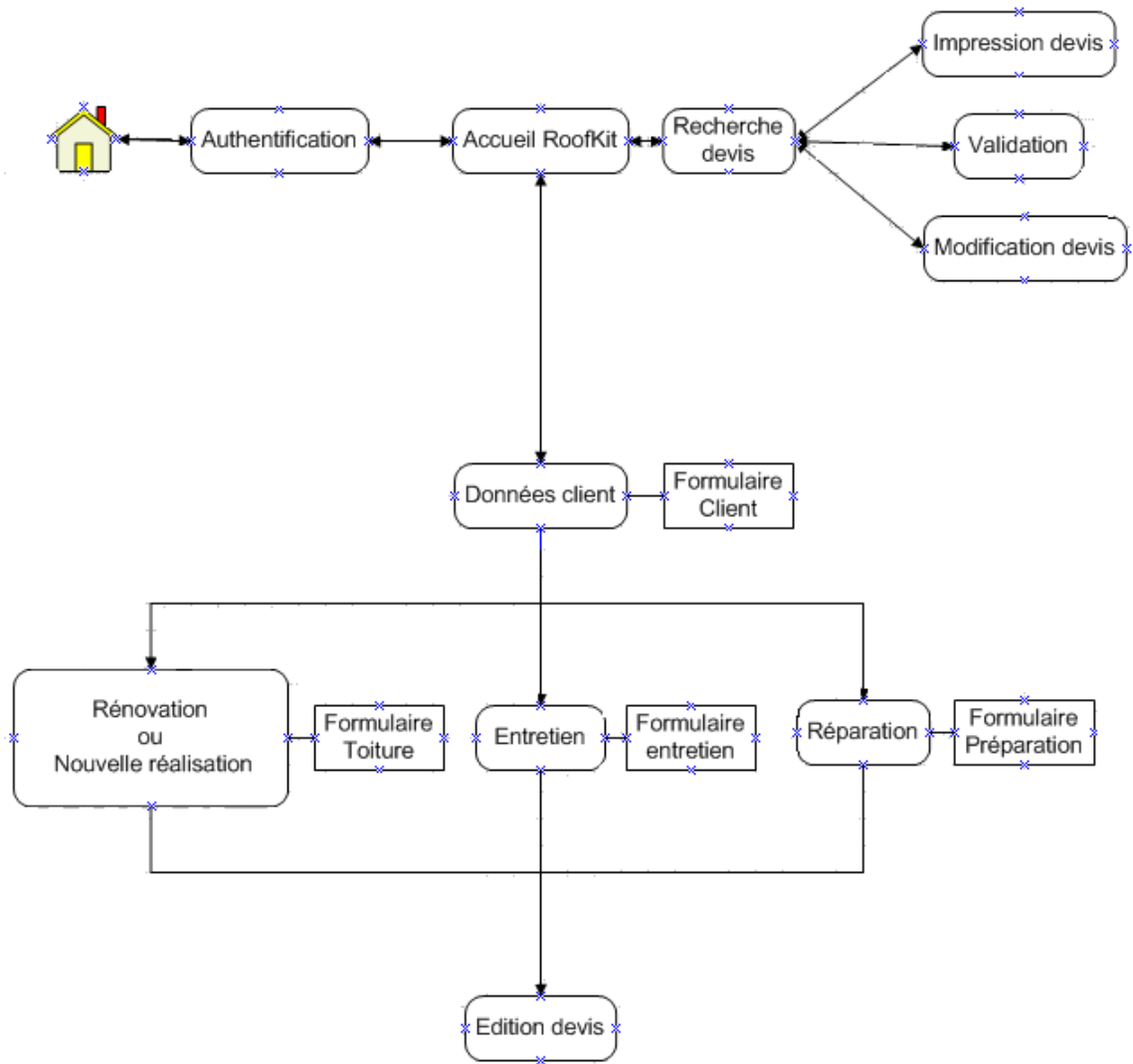
Le franchisé peut enregistrer un devis au format texte mis en page. Ces documents sont utilisés pour la transmission de devis par courriel.

### 2-2-6 Cas d'utilisation « Rechercher un devis » :

Cette fonctionnalité permet de rechercher un devis dans la base de données en utilisant différents filtres. Les filtres à utiliser sont : Le nom du client, le numéro de devis, la date d'édition.

### 3-Solution envisagée

#### 3-1 Schéma de l'infrastructure de l'application :





### 3-2 scénario

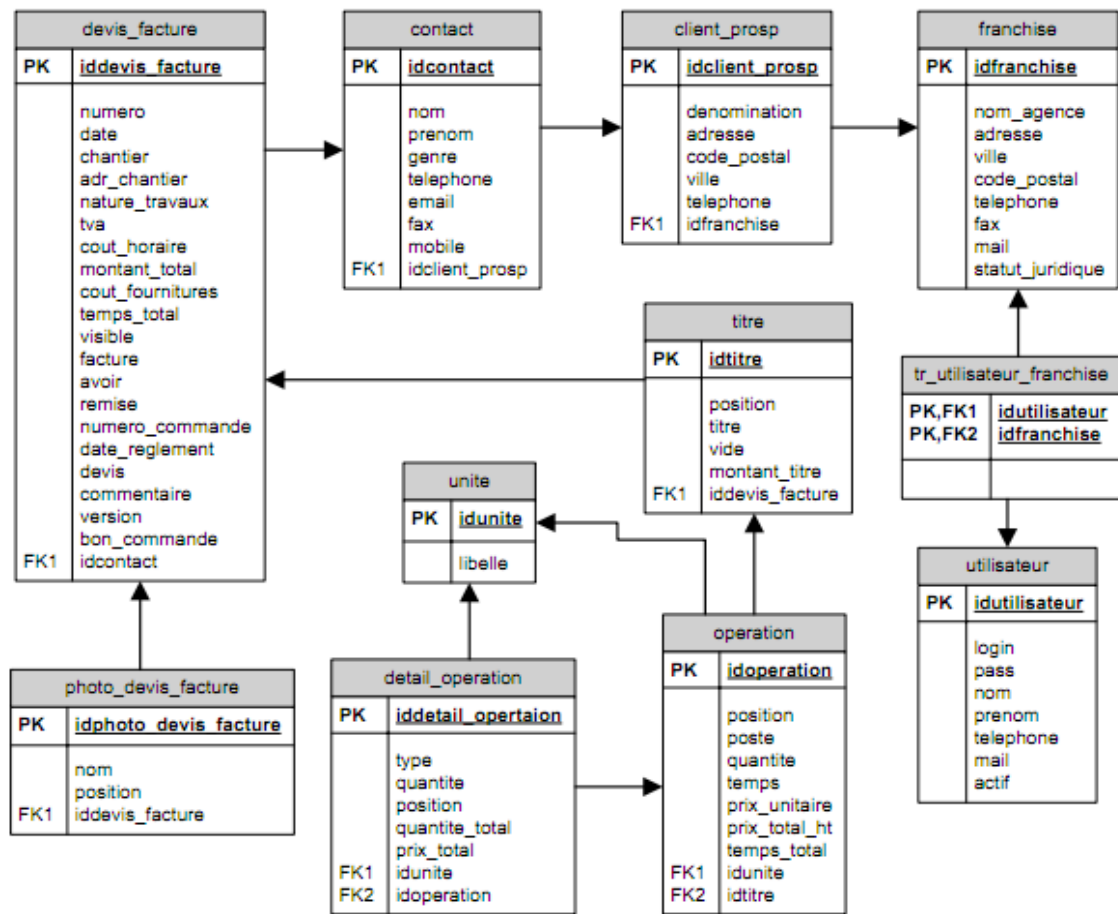
Le franchisé en arrivant sur notre application via l'application intranet d'Attila système se connecte , une fois connecté il est redirigé vers la page d'accueil , cette dernière comporte plusieurs choix qui sont :

- « éditer un devis nouvelle toiture ».
- « éditer un devis entretien ».
- « éditer un devis réparation ».
- « éditer un devis rénovation ».
- « rechercher devis ».

Si le franchisé choisi un des quatre cas d'édition de devis , il est dans un premier temps redirigé vers une Page client qui lui demande les informations client , une fois les données complétées il est redirigé vers la page du type de devis sélectionné , une fois toutes les informations du devis complétées un récapitulatif du devis est généré.

Dans le cas ou l'utilisateur choisi « Rechercher Devis » il est redirigé vers une page qui lui permet de rechercher un devis selon certains critères , une fois son devis trouvé et sélectionné il peut le modifier , le valider , l'imprimer et le sauvegarder. Si il choisi « modifier » son devis sélectionné apparaît et il peut le modifier directement.

### 3-3 Schéma Conceptuel de la base de donnée (fourni)



### 3-3-1 description des tables :

**La base de donnée est composé de 11 tables qui sont :**

**La table « franchise » :**

Elle contient toutes les informations sur les franchises d'Attila Système soit :

- son id dans la base de donnée (clé primaire)
- le nom de l'agence
- son adresse
- sa ville
- son code postal
- son numéro de téléphone
- son numéro de fax
- son adresse Mail
- son statut juridique.

**La table « utilisateur » :**

Elle contient toutes les informations sur les franchisés qui éditent les devis soit :

- son id dans la base de donnée (clé primaire)
- son login
- son mot de passe
- son nom
- son prénom
- son numéro téléphone
- son adresse mail
- son statut

**La table « tr\_utilisateur\_franchise » :**

Elle associe le franchisé à son entreprise ,elle contient donc :

- l'id de la table « utilisateur ». (clé primaire,clé étrangère)
- l'id de la table « franchise ». (clé primaire,clé étrangère)

### **La table « client\_prosp » :**

Elle contient toutes les informations sur le client pour le quel le franchisé a édité un devis, c'est à dire :

- l'id du client dans la base de donnée. (clé primaire)
- sa dénomination (ou son nom prénom).
- son adresse.
- son code postal.
- sa ville.
- son numéro de téléphone.
- l'id dans « franchise » de la franchise avec la quel il traite. (clé étrangère)

### **La table « contact » :**

Elle contient toutes les informations sur ceux qui ont édité des devis soit :

- l'id du contact dans la base de donnée. (clé primaire)
- son nom.
- son prénom.
- son genre.
- son numéro de téléphone.
- son email.
- son numéro de fax.
- son numéro de téléphone portable.
- l'id du client dans « client\_prosp » pour le devis concerné. (clé étrangère)

### **La table « devis\_facture » :**

Cette table contient toutes les informations sur le devis c'est a dire :

- l'id du devis dans la base de donnée (clé primaire)
- son numéro.
- sa date d'édition.
- le nom du chantier.
- l'adresse du chantier.
- la nature des travaux.
- le taux de TVA.
- le coût horaire.
- le montant total.
- le coût en fournitures.
- le durée total.
- la visibilité.
- facture (oui si c'est une facture , non dans le cas contraire).
- l'avoir.
- la remise.
- le numéro de commande.
- la date de règlement.
- devis (oui si c'est un devis , non dans le cas contraire).
- le commentaire.
- la version.
- le bon de commande
- l'id dans la table « contact » de celui qui édite le devis. (clé étrangère)

### **La table « photo\_devis\_facture » :**

Cette table contient les informations sur les photos des devis , c'est a dire :

- son id dans la base de donnée. (clé primaire)
- son nom.
- sa position
- l'id du devis dans « devis\_facture » à qui son rattaché les photos.(clé étrangère)

### **La table « titre » :**

Elle contient toutes les informations sur les titres des devis , c'est a dire :

- son id dans la base de donnée (clé primaire).
- sa position sur le devis.
- son titre.
- indication si il est vide ou non.
- le montant total du titre.
- l'id dans « devis\_facture » du devis qui lui est associé. (clé étrangère).

### **La table « unite »:**

Elle contient les libelles des unités utilisées dans les devis , c'est a dire :

- son id dans la base de donnée (clé primaire).
- son libelle.

### **La table « operation » :**

Elle contient toutes les informations sur les opérations du devis c'est à dire :

- l'id de l'opération dans la base de donnée (clé primaire).
- sa position dans le devis.
- le détail écrit de l'opération (poste).
- la quantité de matériaux de l'opération.
- son temps de réalisation (en minute).
- le prix unitaire des matériaux.
- le prix total hors taxe
- son temps de réalisation (en heure)
- l'id de l'unité utilisé dans l'opération venant de la table « unite »(clé étrangère)
- l'id du titre avec le quel elle est rattachée dans « titre »(clé étrangère) .

### **La table « detail\_operation » :**

Cette table contient tous les détails des opérations , c'est a dire :

- son id dans la base de donnée (clé primaire).
- le type de matériau.
- la quantité du type de matériau.
- sa position dans le devis.
- la quantité total du type de matériau.
- le prix total des matériaux.
- l'id de l'unité utilisé dans le détail venant de la table « unite » (clé étrangère)
- l'id de l'opération dans « operation » à la quelle il est rattaché (clé étrangère)



Exemple d'une partie d'un devis, fourni par Attila système, modifié pour illustrer l'intérêt des différentes tables :



**LE SPÉCIALISTE DES TOITURES**

Table client\_prosp

Le : 25-01-2011  
Devis N° : DE-2-2011-5.1

Table devis\_facture

Communauté de communes de suippes  
A l'attention de Monsieur Rochat  
pl Hôtel de Ville  
51600 - SUIPPES

Tel : 0326700855 ou 0627421552

**CHANTIER**  
TOITURE BASSE DE L'ÉGLISE DE SUIPPES

**Nature des travaux**  
DEMOUSSAGE ET TRAITEMENT FONGICIDE DE LA TOITURE DE L'ÉGLISE DE SUIPPES





table photo\_devis\_facture

table franchise

ATTILA FORMATION  
ZAC FICHET BAUCHE - 51110 - BAZANCORT  
- Mail : formation@attila-systeme.fr  
SARL au capital de : 7500 € - APE : 4391B - SIRET : 52184472000016  
**WWW.ATTILA-SYSTEME.FR**

Page 1/4



ATILA  
SYSTEME

table devis\_facture

LE SPÉCIALISTE DES TOITURES

Devis N° : DE-2-2011-5.1

table titre

## INSTALLATION DE CHANTIER

Déplacement, approvisionnement du chantier en matériaux et matériels.

table opération

Quantité	Unité	Prix UHT	total
1	ens	124.00 €	124.00 €

sous-total INSTALLATION DE CHANTIER : 124.00 €

table titre

table detail\_operation

## SECURITE

Mise ne sécurité par camion nacelle, harnais.

table opération

Quantité	Unité	Prix UHT	total
1	ens	186.00 €	186.00 €

sous-total SECURITE : 186.00 €

table titre

## NETTOYAGE DE LA TOITURE

Brossage de la toiture sur les parties basses de droite et de gauche de l'église.

table opération

Quantité	Unité	Prix UHT	total
330	m²	6.95 €	2 293.50 €

Traitement fongicide de la toiture basse de gauche et de droite de l'église

Quantité	Unité	Prix UHT	total
330	m³	3.40 €	1 122.00 €



table unite



sous-total NETTOYAGE DE LA TOITURE : 3415.50 €

ATILA FORMATION  
ZAC FICHET BAUCHE - 51110 - BAZANCORT  
- Mail : formation@attila-systeme.fr  
SARL au capital de : 7500 € - APE : 4391B - SIRET : 52184472000016  
**WWW.ATILA-SYSTEME.FR**

Page 2/4

(ce devis est fourni en annexe en version non modifié).

## 4 Conception détaillée

### 4-1 La classe Commune : classe BDD

Cette classe gère la lecture et l'écriture dans la base de donnée.  
Elle utilise l'espace de noms **System.Data.SqlClient** qui est le fournisseur de données .NET Framework pour SQL Server qui décrit une collection de classes utilisées pour accéder à une base de données SQL Server .

Méthodes de la classe :

Signatures	description
public bool Connection(string pseudo, string mdp)	Elle permet d'établir la connexion avec la base de donnée elle reçoit en paramètre le login et le mot de passe saisies par le franchisé et renvoie un booléen , true si connexion établie , false si erreur.
public void Deconnection(SqlConnection Connect)	Cette méthode permet de fermer la connexion avec la base de donnée.
public bool VerifNum(int Num)	Cette méthode Vérifie si le numéro de devis quelle reçoit en paramètre n'existe pas déjà dans la base de donnée ,elle renvoie false si il existe , true si il n'existe pas.
public string Select1Element(string Requete)	Cette méthode permet de récupérer plusieurs valeur dans la base de donnée.
public void RequeteNonSelect(string Requete)	Cette méthode permet d'écrire dans la base de donnée.
public void Remp_Table_Contact(string idC,string idF)	Cette méthode permet de remplir la table contact , elle reçoit en paramètre l'id client_prosp et l'id franchise
public void PremiereOperation(double PrixT,double tuile,double liteau,double soustoit)	Cette méthode permet de générer le premier titre , la première opération et les détails opération associé pour les matériaux principaux , elle reçoit en paramètre le prix des tuiles , la quantité de tuile,de liteau et de sous toit.

public void TitreBdd(string attributVal)	Cette méthode permet d'ajouter un titre au devis , elle reçoit en paramètre la valeur des attributs de la requête SQL.
public void OperationBdd(string attributOpe)	Cette méthode permet d'ajouter une opération au devis , elle reçoit en paramètre la valeur des attributs de la requête SQL.
public void DetailOperationBdd(string attributDetOpe)	Cette méthode permet d'ajouter un détail opération au devis , elle reçoit en paramètre la valeur des attributs de la requête SQL.
public string SelectElement(string Requete)	Cette méthode permet de récupérer une et une seul valeur dans la base de donnée.

## 4-2 : travail personnel : Once Kevin

### 4-2-1 La page Connexion :

#### 4-2-1-1 : Description

Cette page est la première que voit le franchisé quand il accède à notre application, elle lui permet de s'identifier.

Cette Page n'était pas prévu au projet à l'initial mais durant le développement nous nous sommes très vite rendu compte qu'un système d'authentification était indispensable pour deux raisons :

- L'application doit être sécurisé car l'accès au devis doit être restreint à un certain nombre de personnes répertoriées dans la base de donnée.

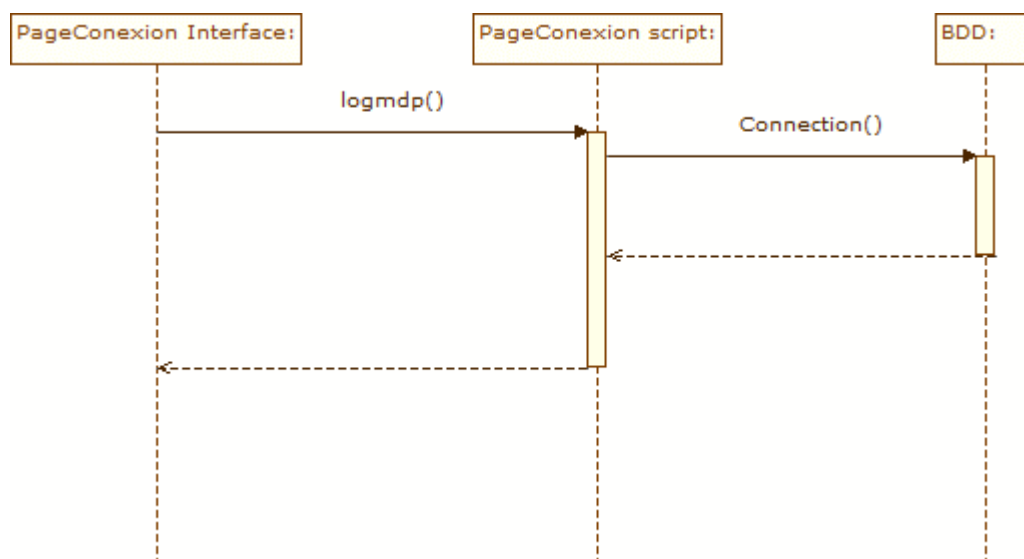
- le franchisé éditant le devis doit être identifié pour l'enregistrer dans la base de donnée.

Les Méthodes du script c# associé à la Page Connexion :

Signature	Description
public void logmdp(object sender, EventArgs e)	Cette méthode test si les identifiants entrés par l'utilisateur sont correcte
protected void LogText_TextChanged(object sender, EventArgs e)	Cette méthode permet de supprimer ou remplacer certains caractères

protected void MdpText_TextChanged(object sender, EventArgs e)	Cette méthode permet de supprimer ou remplacer certains caractères
--	---

#### 4-2-1-2 : Diagramme de séquence :



### Algorithme de la méthode logmdp :

```
debut
chaîne login ← ZoneDuLogin.contenu

chaîne motdepasse ← ZoneDuMotDePasse.contenu

BDD TestConnection

booléen test ← vrai
si(login!= Null & motdepasse!= Null)
alors
test ← TestConnection.Connection(login,motdepasse)

    si test = true

    alors

    VariableSession[Login ] = login;
    VariableSession[ID]=TestConnection.Select1Element(« Requete select » )

    RedirigerVers(« WFACCUEIL.aspx »)
    sinon

    AfficherMessageErreur
    fin si

sinon
AfficherMessageErreur
fin si

fin
```

(Code disponible en annexe)

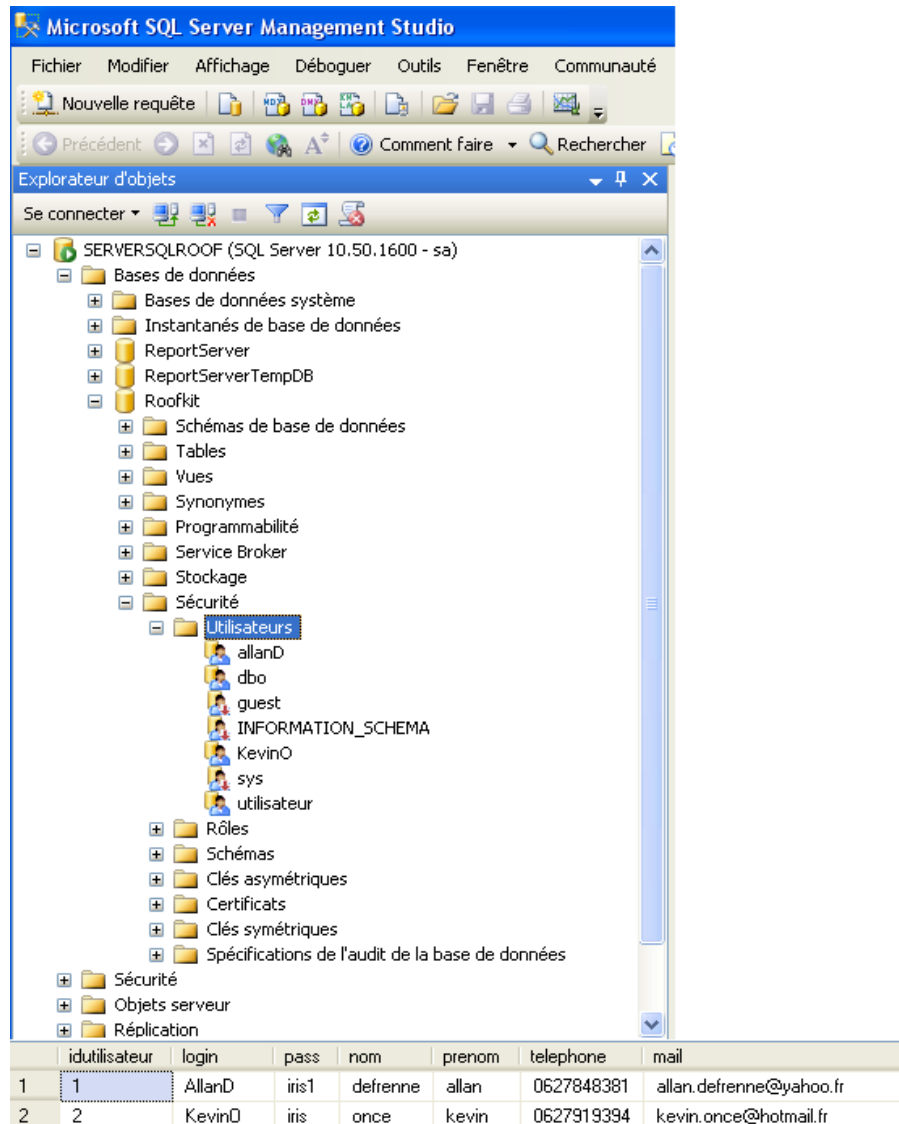
nous pouvons voir ici l'apparition de l'utilisation de variable de session , se sont des variables qui permettent d'enregistrer des données et de les utiliser dans différents page , pas forcément celle ou elle a été initialisé ,elle reste accessible le temps de la session (tant que l'application reste ouverte).



### 4-2-1-3 Test unitaire :

#### Situation :

voici une prise d'écran du contenu des utilisateurs pour notre base de donnée Roofkit suivie d'une prise d'écran du contenu de la table utilisateur.



The screenshot shows the Microsoft SQL Server Management Studio interface. The 'Explorateur d'objets' (Object Explorer) pane on the left displays the hierarchy of the 'SERVERSQLROOF (SQL Server 10.50.1600 - sa)' instance. The 'Roofkit' database is expanded, and the 'Utilisateurs' (Users) folder is selected, showing a list of users: allanD, dbo, guest, INFORMATION\_SCHEMA, KevinO, sys, and utilisateur. Below the screenshot, a table displays the data from the 'utilisateur' table.

	idutilisateur	login	pass	nom	prenom	telephone	mail
1	1	AllanD	iris1	defrenne	allan	0627848381	allan.defrenne@yahoo.fr
2	2	KevinO	iris	once	kevin	0627919394	kevin.once@hotmail.fr

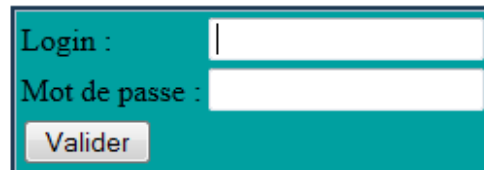
On peut voir que deux comptes (en excluant les comptes créés par défaut) sont créés dans la base de donnée dont les identifiants sont :

- AllanD iris1
- KevinO iris

Pour le test , je vais utiliser le compte KevinO.

Interface de test pour la Page connexion :

## AUTHENTIFICATION



Login :

Mot de passe :

Premier test : identifiants incorrects

scénario :

Je tente de me connecter en utilisant les identifiants « azerty » et « faux ».

## Résultat :

```
33 public void logmdp(object sender, EventArgs e)
34 {
35     string Login = LogText.Text;
36     string Mdp = MdpText.Text;
37     bool test = true;
38
39     BDD TestConec = new BDD();
40
41
42     if (Login != "" && Mdp != "")
43     {
44
45         Login = LogText.Text;
46         Mdp = MdpText.Text;
47
48         test = TestConec.Connection(Login, Mdp);
49
50
51         if (test == true)
52         {
53             Session["Login"] = Login;
54             Session["ID"] = TestConec.Select1Element("SELECT idutilisateur FROM utilisateur WHERE login= '" + Login + "'");
55
56
57
58
59
60             Response.Redirect("WFaccueil.aspx");
61
62         }
63         else
64         {
65
66             DetailErreur.Visible = true;
67             DetailErreur.Text = (string)Session["Erreur"];
68
69         }
```

Nom	Valeur	Type
this	{ASP.pageconnexion.aspx}	RoofKit.P
sender	{Text = "Valider"}	object {S
e	{System.EventArgs}	System.E
Login	"azerty"	string
Mdp	"faux"	string
test	false	bool

Liste d'erreurs: 0 erreurs

On peut voir que les variables login et mdp se sont correctement initialisé avec les identifiants entrés ,j'utilise la méthode Connection de la classe BDD pour tester si les identifiants sont correcte , la méthode renvoie false donc test s'initialise avec false, la condition du if n'est pas respectée le programme passe donc dans le else et affiche un message d'erreur :

# AUTHENTIFICATION

Login :	<input type="text" value="azerty"/>
Mot de passe :	<input type="password"/>
<input type="button" value="Valider"/>	

```
Échec de l'ouverture de session de  
l'utilisateur 'azerty'.
```

Deuxième test : identifiants corrects

scénario :

Je tente de me connecter en utilisant des identifiants existant dans la base de donnée , ici « KevinO » et « iris ».

## Résultat :

The screenshot displays a Visual Studio IDE with a C# code file and a 'Variables locales' (Local Variables) window. The code is a method named `logmdp` that takes `sender` and `e` as parameters. It initializes `Login` and `Mdp` with values from `LogText` and `MdpText`, sets `test` to `true`, and creates a `BDD` object `TestConec`. An `if` statement checks if both `Login` and `Mdp` are non-empty. If true, it updates the session with the login details, calls `TestConec.Connection`, and redirects the response to `WFAccueil.aspx`. The `Variables locales` window at the bottom shows the state of these variables: `this` is `{ASP.pageconnexion.aspx}`, `sender` is `{Text = "Valider"}`, `e` is `{System.EventArgs}`, `Login` is `"KevinO"` (type `string`), `Mdp` is `"Iris"` (type `string`), and `test` is `true` (type `bool`). The `Login`, `Mdp`, and `test` rows are highlighted with a red rectangle.

```
33 public void logmdp(object sender, EventArgs e)
34 {
35     string Login = LogText.Text;
36     string Mdp = MdpText.Text;
37     bool test = true;
38
39     BDD TestConec = new BDD();
40
41
42     if (Login != "" && Mdp != "")
43     {
44
45         Login = LogText.Text;
46         Mdp = MdpText.Text;
47
48         test = TestConec.Connection(Login, Mdp);
49
50
51         if (test == true)
52         {
53             Session["Login"] = Login;
54             Session["ID"] = TestConec.Select1Element("SELECT idutilisateur FROM utilisateur WHERE login= ' " + Login + "'");
55
56
57
58
59
60             Response.Redirect("WFAccueil.aspx");
61
62         }
63     }
64     else
65     {
66
```

Nom	Valeur	Type
this	{ASP.pageconnexion.aspx}	Roofkit.P
sender	{Text = "Valider"}	object {S
e	{System.EventArgs}	System.E
Login	"KevinO"	string
Mdp	"Iris"	string
test	true	bool

On peut voir que les variables login et mdp se sont correctement initialisé avec les identifiants entrés, la méthode Connection de la classe BDD renvoie true donc test s'initialise avec true, la condition du if est respectée le programme passe donc par le if et me redirige vers la page d'accueil.

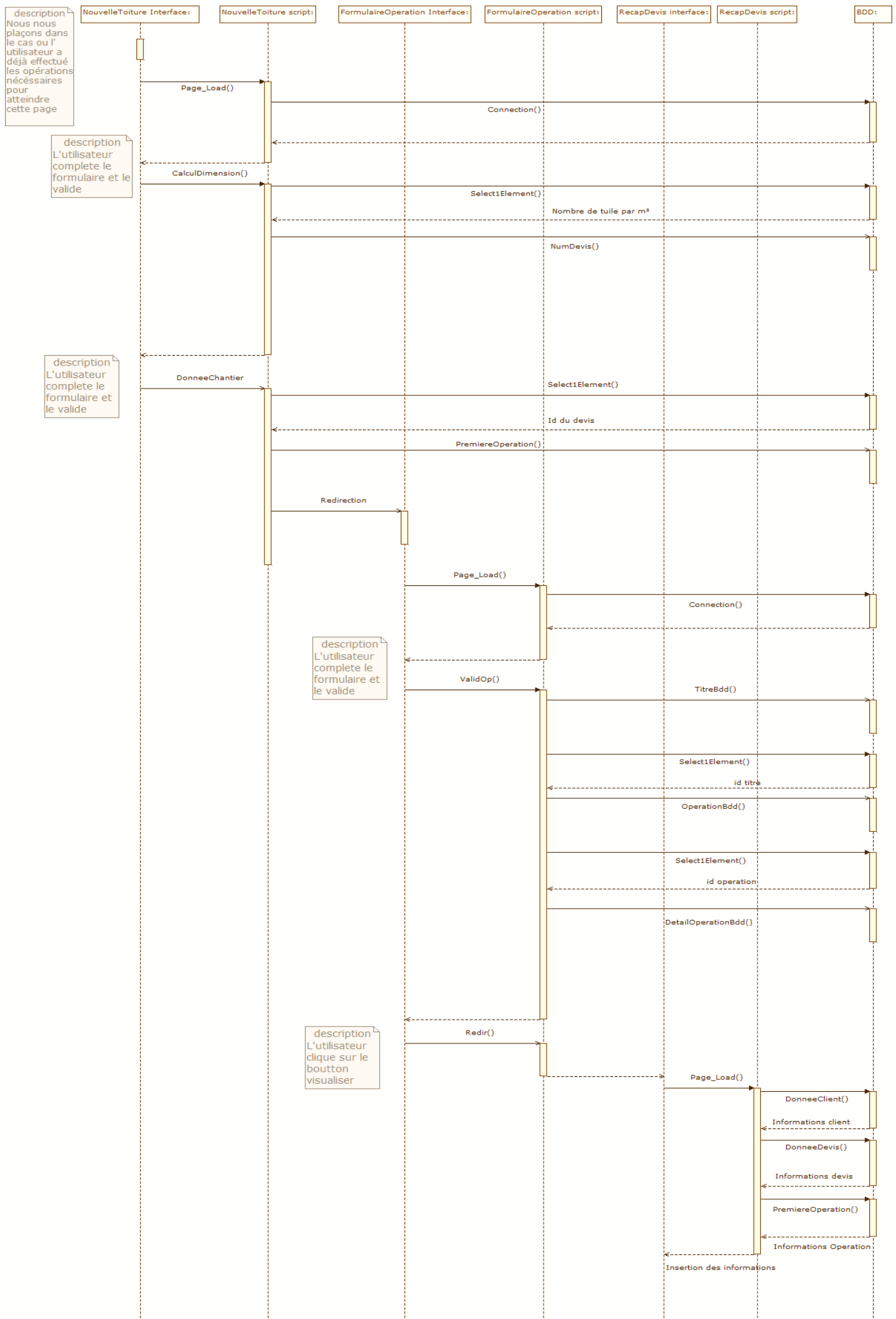
#### 4-2-2 Module d'édition devis

ce module permet de créer son devis dans la base de donnée , ceci se fait sous une suite de formulaire comportant des questions dans une premier temps sur la toiture (sa forme , son nombre de face ,ses dimensions , son type de tuile) puis des questions sur le chantier (son adresse , son nom, la nature des travaux et le taux de TVA) ensuite le franchisé est redirigé vers une page qui permet d'ajouter des opérations au devis ou de visualiser toutes les informations du devis.

#### Diagramme de séquence :

Pour des raisons de lisibilité, certaines méthodes non indispensables à la compréhension du fonctionnement du module ont été supprimé du diagramme tel que les méthodes de déconnexion ou celles qui servent à filtrer la saisie.





#### 4-2-2-1 Page nouvelle toiture :

##### 4-2-2-1-1 description :

C'est cette page qui comporte les formulaires sur la toiture et le chantier ,voici les méthodes du script c#associé à cette page :

protected void Page_Load(object sender, EventArgs e)	Au chargement de la page cette méthode établie la connexion entre certains composants et la base de donnée. Elle génère également la date
public void CalculDimension(object sender, EventArgs e)	Cette méthode réalise tous les calculs sur les matériaux principaux de la toiture (litage,tuile,sous toit) selon les informations quelle a récupéré et lance la méthode qui génère un numéro de devis
public void ImageToiture(object sender, EventArgs e)	Cette méthode choisi l'image qui est affiché dans l'asp selon la sélection du franchisé
public void Donneechantier(object sender, EventArgs e)	Cette méthode crée le devis dans la base de donnée selon les informations qu'elle a récupéré et lance la méthode de la classe bdd PremiereOperation.
public void NumDevis()	Cette méthode génère un numéro de devis
public void SelectNbr(object sender, EventArgs e)	Cette méthode gère le nombre de choix à afficher dans un composant selon les sélections du franchisé
protected void LongueurNbr_TextChanged(object sender, EventArgs e) protected void LargeurNbr_TextChanged(object sender, EventArgs e) protected void PrixHt_TextChanged(object sender, EventArgs e)	Ces méthodes Suppriment les caractères non numérique du champ à l'exception d'un point Ou d'une virgule
protected void NomChantier_TextChanged(object sender, EventArgs e) protected void NatureTravaux_TextChanged(object sender, EventArgs e) protected void AdresseChantier_TextChanged(object sender, EventArgs e)	Ces méthodes suppriment ou remplacent certains caractères spéciaux.

#### 4-2-2-1-2 Test unitaire :

##### scénario :

Je vais utiliser l'application réaliser dans son usage normal , en ne montrant que les parties concernant la Page nouvelle toiture.

##### Test :

Partie Calculs sur les matériaux. (cas d'une toiture rectangulaire à deux faces)

Voici l'interface de l'application pour la page nouvelle toiture partie formulaire information toiture avec les champs complétés :

### Nouvelle toiture

#### Information sur la toiture

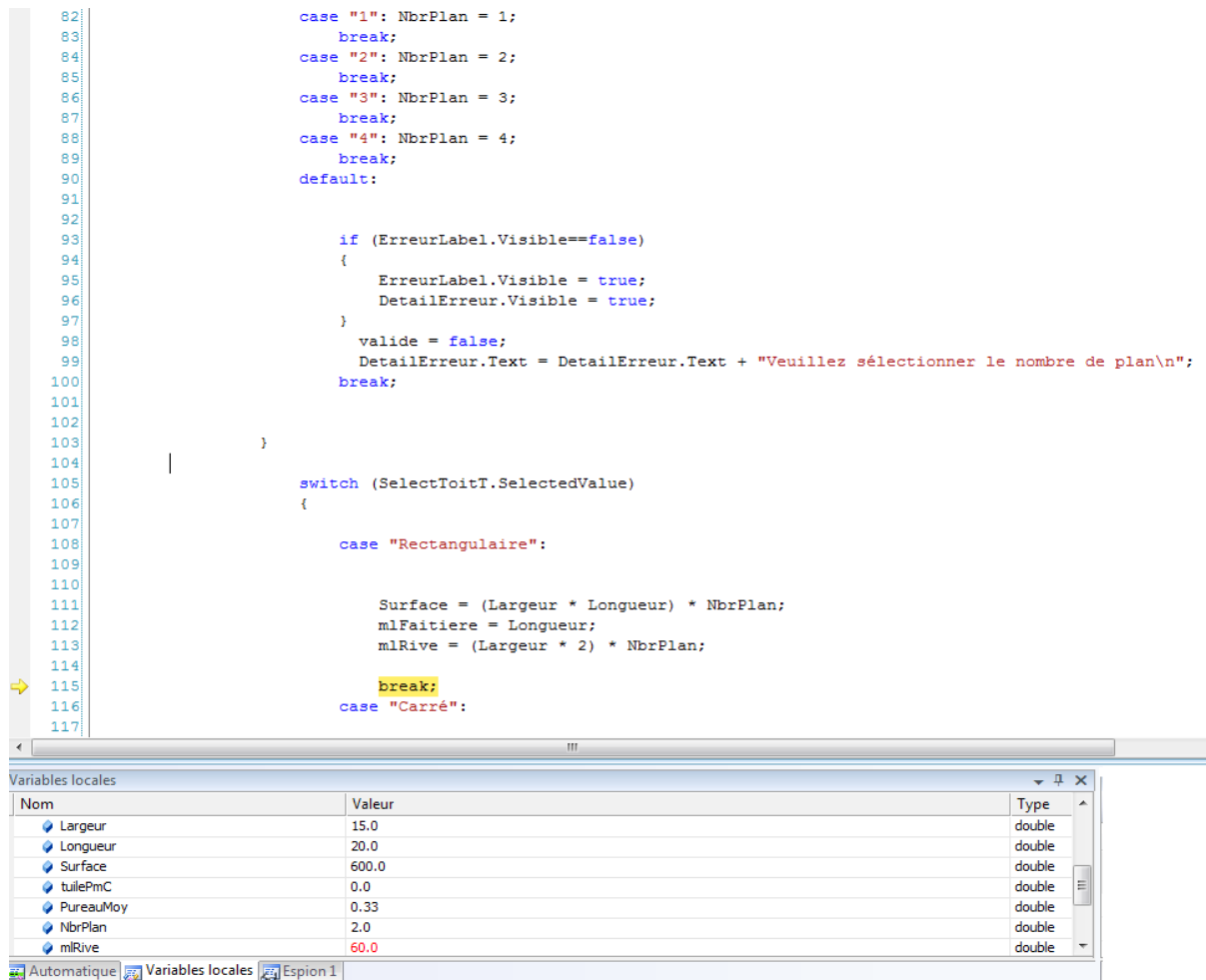
Forme de la toiture	Rectangulaire	Nombre de plan	2
			
Longueur	20	Largeur	15
Type de Tuile	HP13	Prix HT :	0.80
<button>Valider</button>			

A la validation du formulaire la méthode CalculDimension se lance :

```

82         case "1": NbrPlan = 1;
83         break;
84         case "2": NbrPlan = 2;
85         break;
86         case "3": NbrPlan = 3;
87         break;
88         case "4": NbrPlan = 4;
89         break;
90         default:
91
92
93         if (ErreurLabel.Visible==false)
94         {
95             ErreurLabel.Visible = true;
96             DetailErreur.Visible = true;
97         }
98         valide = false;
99         DetailErreur.Text = DetailErreur.Text + "Veuillez sélectionner le nombre de plan\n";
100        break;
101
102
103    }
104
105    switch (SelectToitT.SelectedValues)
106    {
107
108        case "Rectangulaire":
109
110
111        Surface = (Largeur * Longueur) * NbrPlan;
112        mlFaitiere = Longueur;
113        mlRive = (Largeur * 2) * NbrPlan;
114
115        break;
116        case "Carré":
117

```



Nom	Valeur	Type
Largeur	15.0	double
Longueur	20.0	double
Surface	600.0	double
tuilePmC	0.0	double
PureauMoy	0.33	double
NbrPlan	2.0	double
mlRive	60.0	double

On peut voir que les variables largeur et longueur se sont initialisées correctement , comme j'ai choisi Rectangulaire pour la forme de la toiture la surface a été calculé selon la formule  $(\text{Largeur} * \text{longueur}) * \text{Nombre de face}$  , la quantité en mètre linéaire de rive suis la formule :  $(\text{Largeur} * 2) * \text{Nombre de plan}$  et la quantité en mètre linéaire de faîtière est égal à la longueur.

```

146
147     switch (SelectTuileT.SelectedValeur)
148     {
149         case "ALPHA 10":
150
151             tuilePmC = Double.Parse(actionBDD.SelectElement("Select Nbrmc from materiel where element='tuile' AND nom='ALPHA 10'"));
152             break;
153
154         case "HP13":
155
156             tuilePmC = Double.Parse(actionBDD.SelectElement("Select Nbrmc from materiel where element='tuile' AND nom='HP13'"));
157             break;
158         default:
159
160             if (ErreurLabel.Visible == false)
161             {
162                 ErreurLabel.Visible = true;
163                 DetailErreur.Visible = true;
164             }
165
166             valide = false;
167             DetailErreur.Text = DetailErreur.Text + "veuillez sélectionner un type de tuile\n";
168             break;
169
170     }
171
172
173
174

```

Nom	Valeur	Type
Largeur	15.0	double
Longueur	20.0	double
Surface	600.0	double
tuilePmC	13.2	double
PureauMoy	0.33	double
NbrPlan	2.0	double
mRive	60.0	double

Automatique Variables locales Espion 1

Sur cette prise d'écran on peut voir que le nombre de tuile par mètre carré pour le type de tuile choisi est pris de la base de donnée et est égale à 13.2.

```

175
176     {
177         PrixUT = Double.Parse(PrixHt.Text.Replace('.', ','));
178
179     }
180     else
181     {
182         valide = false;
183
184         if (ErreurLabel.Visible == false)
185         {
186             ErreurLabel.Visible = true;
187             DetailErreur.Visible = true;
188         }
189
190         DetailErreur.Text = DetailErreur.Text + "veuillez indiquer un prix unitaire pour le type de tuile sélectionnée\n";
191     }
192
193

```

Nom	Valeur	Type
PureauMoy	0.33	double
NbrPlan	2.0	double
mRive	60.0	double
mFaitiere	20.0	double
PrixUT	0.8	double
mLit	0.0	double

Automatique Variables locales Espion 1

Sur la prise d'écran précédente, on peut voir que la variable PrixUT c'est correctement initialisé avec la valeur entré dans le formulaire (0,80).

```

196: {
197:     tuile = Math.Round(Surface * tuilePmC);
198:     Session["tuile"] = tuile;
199:
200:     if (SelectToitT.SelectedValues == "Rectangulaire")
201:     {
202:
203:
204:         litem = Math.Round(Largeur / PureauMoy);
205:         mLit = (litem * Longueur) * NbrPlan;
206:     }
207:     if (SelectToitT.SelectedValues == "Carré")
208:     {
209:         litem = Math.Round(Longueur / PureauMoy);
210:         mLit = ((litem * Longueur)) / 2 * NbrPlan;
211:
212:
213:     }
214:
215:
216:
217:     Session["litem"] = mLit;
218:     soustoit = Surface;
219:     Session["soustoit"] = soustoit;
220:     if (PrixHT.Text != null)
221:     {
222:         PrixUT = Double.Parse(PrixHT.Text.Replace('.', ','));
223:         PrixHT = tuile * PrixUT;
224:         Session["PrixTT"] = PrixHT;
225:
226:         form1.Visible = false;
227:
228:         form2.Visible = true;
229:         NumDevis();
230:     }
231: }

```

Nom	Valeur	Nom	Valeur
NumCommande	"none"	PureauMoy	0.33
NumeroDevis	null	NbrPlan	2.0
PrixHT	{System.Web.UI.WebControls.TextBox}	mRive	60.0
PrixHTT	6336.0	mFaitiere	20.0
PrixL	{Text = "Prix HT :"} "0"	PrixUT	0.8
RemiseTemp		mLit	1800.0
SelectToitT	{System.Web.UI.WebControls.DropDownList}		

Et pour finir sur la partie toiture , ici on peut voir que la quantité de tuile est calculé par la formule :  $\text{Surface} * \text{tuile par mètre carré}$  .Le nombre de litem est calculé grâce à la formule (en se plaçant dans le cas Rectangulaire) :

$\text{Largeur} / \text{pureau moyen}$  , et ainsi le mètre linéaire de litem est calculé par :

$\text{Nombre de litem} * \text{Longueur}$  . La quantité de sous toit est égale à la surface.

les données calculées sont enregistrées dans des variables de session pour pouvoir les récupérer dans d'autre page .La fonction qui génère un numéro de devis unique est lancé à la fin de la méthode.

Une fois cette fonction terminée, le formulaire contenant les questions sur la toiture disparaît et laisse place au formulaire information chantier (ici complété) :

## Nouvelle toiture

### Information sur le chantier

Nom du chantier :	Revue finale exemple
Adresse du chantier :	rue de la revue finale
Nature des travaux :	creation toiture
Taux TVA :	7
<input type="button" value="Valider"/>	

à la validation du formulaire la méthode `Donneechantier` se lance :

```

369
370 string attribut = "numero,date,chantier,adr_chantier,nature_travaux,tva,cout_horaire,montant_total,cout_fournitures,temps_total,visible,facture"
371 + ",avoir,remise,numero_commande,date_reglement,devis,commentaire,version,bon_commande,idcontact";
372 SqlCommand command = new SqlCommand("INSERT INTO devis_facture(" + attribut + ")VALUES('" + NumeroDevis + "','" + Date + "','" + Nchantier + "','"
373
374 command.Parameters.Add(new SqlParameter("@TVA", SqlDbType.Float));
375 command.Parameters["@TVA"].Value = "" + Convert.ToDecimal(tvaDec) + "";
376 command.Parameters.Add(new SqlParameter("@CoutH", SqlDbType.Float, 18));
377 command.Parameters["@CoutH"].Value = "" + Convert.ToDecimal(CoutHoraire) + "";
378 command.Parameters.Add(new SqlParameter("@MontantT", SqlDbType.Float, 18));
379 command.Parameters["@MontantT"].Value = "" + Convert.ToDecimal(MontantTotal) + "";
380 command.Parameters.Add(new SqlParameter("@CoutF", SqlDbType.Float, 18));
381 command.Parameters["@CoutF"].Value = "" + Convert.ToDecimal(MontantTotal) + "";
382 command.Parameters.Add(new SqlParameter("@RemiseT", SqlDbType.Float, 18));
383 command.Parameters["@RemiseT"].Value = "" + Convert.ToDecimal(Remise) + "";
384
385 command.ExecuteNonQuery();
386
387 Session["iddevis"] = actionBDD.Select1Element("select iddevis_facture from devis_facture where numero='" + NumeroDevis + "'");
388
389
390 tuile = (double)Session["tuile"];
391
392 liteau = (double)Session["liteau"];
393
394 soustoit = (double)Session["soustoit"];
395
396 PrixHT = (double)Session["PrixHT"];
397
398 actionBDD.PremiereOperation(PrixHT, tuile, liteau, soustoit);
399
400 actionBDD.Deconnection(connection);

```

Nom	Valeur
(double)Session["PrixHT"]	6336.0
PrixHT	6336.0
Session	{System.Web.SessionState}
Session["PrixHT"]	6336.0
actionBDD	{Roofkit.BDD}
liteau	1800.0
soustoit	600.0
this	{ASP.nouvelletoiture_aspx}
tuile	7920.0

Nom	Valeur	Type
AdrChantier	"rue de la revue finale"	string
Date	"2012-5-30"	string
NatureTrav	"creation toiture"	string
Nchantier	"Revue finale exemple"	string
NumeroDevis	"DF912982343"	string
SqlDbType.Float	Float	System.D
attribut	"numero,date,chantier,adr_chantier,nature_travaux,tva,cout_horaire,montant_total,cout_fourniture"	string

Automatique Variables locales Espion 1

On peut voir que les différents variables se sont correctement initialisés avec les informations entrées et qu'un numéro de devis généré par la méthode NumDevis a été récupéré, une requête SQL Insert into devis\_facture, contenant toutes les informations que j'ai entré ou qui est générées par l'application est exécutée, la table devis\_facture est éditée, ainsi que les tables titre, opération et détail opération grâce à la méthode PremiereOperation de la classe BDD.

Voici le résultat dans la base de données :

Table devis\_facture :

	iddevis_facture	numero	date	chantier	adr_chantier	nature_travaux	tva
33	131	DF731510701					
34	132	DF359031378					
35	133	DF729967567					
36	134	DF634952926					
37	135	DF171780107					
38	136	DF140622756					
39	138	DF912982343	2012-05-30	Revue finale exemple	rue de la revue finale	creation toiture	5,5

on peut voir que "Revue finale exemple", "rue de la revue finale", "création toiture" et "5,5" entrés ou sélectionnés précédemment apparaissent.

Table titre :

	idtitre	position	titre	vide	montant_titre	iddevis_facture
34	145					
35	146					
36	147					
37	148					
38	149					
39	150					
40	151					
41	153	1	Installation tuile,liteau,sous toit	non	8136	138

Un titre a été créé, on peut voir que l'iddevis\_facture correspond à la l'id de la ligne précédente précédente on peut également voir le montant total du titre qui est de 8136 €



Table operation :

	idoperation	poste	quantite	prix_total_ht	idunite	idtitre
34	133					
35	134					
36	135					
37	136					
38	137					
39	138					
40	139					
41	141	pose du sous toit ,des liteaux et des tuiles	3	8136	1	153

Une opération a été créée ,le montant total est egal au montant du titre puisqu'il n'y a qu'une opération.On peut voir que l'id titre correspond à la ligne vu precedemment.

Table detail\_operation :

	iddetail_operation	type	quantite	position	prix_total	idoperation
106	185					
107	186					
108	187					
109	188					
110	189					
111	193	tuile	7920	1	6336	141
112	194	liteau	1800	2	900	141
113	195	sous toit	600	3	900	141

On peut voir que trois details opérations ont été créés ,ils concernant les principaux matériaux de la toiture.L'id opération correspond à l'opération vu precedemment ,le prix total du titre correspond (si on additionne les trois "Prix\_total" ) au montant total du titre.

(Code de Donneechantier disponible en annexe)

#### 4-2-2-2 La page Récapitulatif:

##### 4-2-2-2-1 : description

Cette Page permet de visualiser les informations principales du devis (actuellement ) c'est a dire les informations client ,chantier , et l'opération générés de base par l'application.

Méthodes de la page :

Signature	description
public void DonneeClient()	Cette méthode récupère les informations du client dans la table client_prosp et les insères dans les différents composants d'affichages
public void DonneeDevis()	Cette méthode récupère les informations du devis dans la table devis_facture et les insères dans les différents composants d'affichages
public void PremiereOperation()	Cette méthode récupère les différentes lignes de l'opération correspondant aux principaux matériaux du devis, elle prend ses informations dans les tables titres , opération et détail opération et les insères dans les différents composants d'affichages.

(Code de PremiereOpération disponible en annexe )

##### 4-2-2-2- : Test unitaire

scénario :

Je visualise le devis en cours d'edition, le client que j'ai utilisé est Edmond labbé , voici une prise d'écran de la table client\_prosp pour les informations sur ce client :

	idclient_prosp	denomination	adresse	code_postal	ville	telephone	idfranchise
1	1	edmond labbe	817 rue Charles-Bourseul	59500	douai	0627000000	1
2	5						

## Récapitulatif

### Informations Client :

Nom Prénom (ou dénomination)	edmond labbe
Adresse :	817 rue Charles-Bourseul
Code postal :	59500
Ville :	douai
N° de Téléphone	0627000000

### Informations Chantier :

Numero du devis :	DF912982343
Date d'edition :	30/05/2012
Nom du chantier :	Revue finale exemple
Adresse du chantier :	rue de la revue finale
Nature des travaux :	creation toiture
taux de tva :	5.5
Montant total du devis :	8136

### Opération de base :

Titre :	Installation tuile,liteau,sous toit			
Montant total du titre :	8136			
	pose du sous toit ,des liteaux et des tuiles			
Poste :				
Montant total de l'opération :	8136			
Type :	Unite :	Quantite :	Prix Unitaire :	Prix total :
tuile	unite	7920	0.8	6336
liteau	ml	1800	0.5	900
sous toit	ml	600	1.5	900

On peut voir que les informations rentré précédemment et l'opération générée automatiquement avec les matériaux principaux apparaissent (les données correspondent aux données contenues dans les prises d'écrans des tables montrées plus haut )

### 4-2-2-3 : La Page FormulaireOpération

#### 4-2-2-3-1 : description

Cette Page permet d'ajouter un titre ,une opération et un detail opération (actuellement ) par validation du formulaire

Méthodes de la page :

Signature	Description
protected void Page_Load(object sender, EventArgs e)	Au chargement de la page cette méthode établie la connexion entre certains composants et la base de donnée
public void AddDetail(object sender, EventArgs e)	Cette méthode ajoute une ligne de détail opérations par clique
public void ValidOp(Object sender, EventArgs e)	Cette méthode récupère les informations entrées et lance la fonction pour ajouter une opération dans la base de donnée.
public void Redir(object sender, EventArgs e)	Cette méthode redirige vers la Page Récapitulatif

#### 4-2-2-3-2 : test unitaire

scénario :

Je complète les champs de la page comme ceci , je valide et vais voir le résultat dans la base de donnée.

Voici l'interface de cette Page complété :

**Nouvelle Opération**

**Titre :**

voici un test

**Details de l'operation (tache(s) effectuée(s)) :**

**Type :**

faitiere

**Unite :**

ml

**Quantite :**

20

**Prix Unitaire :**

1.10

selectionnez l'unité

selectionnez l'unité

selectionnez l'unité

selectionnez l'unité

à la validation du formulaire la méthode ValidOp (code disponible en annexe)

Resultat dans la base de donnée :

Table titre :

	idtitre	titre	vide	iddevis_facture
35	146			
36	147			
37	148			
38	149			
39	150			
40	151			
41	153	Installation tuile,liteau,sous toit	non	138
42	154	test	non	138

On peut voir qu'un titre a été ajouté et que son titre est "test" comme indiquer dans le test.

Table operation :

	idoperation	poste	quantite	prix_unitaire	prix_total_ht	idunite	idtitre
35	134						
36	135						
37	136						
38	137						
39	138						
40	139						
41	141	pose du sous toit ,des liteaux et des tuiles	3	1,15	8136	1	153
42	142	voici un test	20	1,1	22	1	154

On peut voir qu'une opérations a été ajouté , dont le poste est egal à "voici un test " comme indiquer dans le test, et dont le prix total est egal à 22 € , ce qui devrait corresponde dans la table detail\_operation.

Table detail\_operation :

	iddetail_operation	type	quantite	position	quantite_total	prix_total	idunite	idoperation
107	186							
108	187							
109	188							
110	189							
111	193	tuile	7920	1	7920	6336	1	141
112	194	liteau	1800	2	1800	900	4	141
113	195	sous toit	600	3	600	900	3	141
114	196	faitiere	20	1	20	22	1	142

On peut voir qu'une ligne dont son type est egale à "faitiere" a été ajouté comme indiquer dans le test. Le prix total correspond bien au prix total de l'opération precedente.

#### 4-1-3 Module «Modifier toiture »

Ce module reprend la page récapitulatif et la page FormulaireOpération du module nouvelle toiture, les méthodes utilisé sont donc les même.A l'heure actuelle seul l'ajout d'une opérations dans un devis est traité pour ce module, cette ajout fonctionne de la même façon que l'ajout pour un devis en cours d'edition à la difference près que l'id du devis à modifier est récupéré grâce à la selection du devis dans la Page Rechercher de mon collegue.

#### 4-1-4 Conclusion

Les fonctionnalités principales qui étaient à ma charge sont en partie réalisées. Concernant le module d'édition de devis, un devis peut être correctement édité dans la base de données néanmoins les notions de Coût horraire ne sont pas encore traités et donc des champs sont pour le moment en permanence à zéro, cela fausse du coup forcément le revenu total du devis qui ne prend en compte que le prix des matériaux. Il est également possible d'ajouter des opérations au devis, mais la visualisation de ces opérations ajoutées n'est pas encore réalisée je n'ai donc pas mis à jour le prix total du devis si on ajoute des opérations pour éviter d'avoir une incohérence au niveau du prix total du devis dans la visualisation.

Concernant le module de modification du devis, les problèmes des devis dans le module éditer un nouveau devis se posent également ici car les deux modules utilisent les mêmes pages, il est donc possible de visualiser les informations principales d'un devis contenues dans la base de données et possible d'ajouter des opérations. Il reste donc à traiter la suppression et la modification directement sur la visualisation.

Ce projet est très intéressant, malheureusement le temps risque de me manquer pour finir totalement les fonctionnalités demandées, il aura été envisageable d'améliorer l'esthétique de l'application, le fonctionnel a été privilégié sur le design. L'application pourrait être beaucoup plus dynamique surtout au niveau de la visualisation des devis, il serait possible de réaliser un formulaire dynamique dans lequel on pourrait sur la même page supprimer, ajouter ou modifier des opérations directement sans nécessiter le rechargement complet de la page.

Il sera également possible de gérer le timeout de la session en proposant à l'utilisateur de se reconnecter en cas de perte de la session et ainsi continuer le travail en cours, actuellement en cas de perte de la session il faut retourner sur la page de connexion de l'application pour réaccéder à notre application et passer par la recherche devis pour récupérer le devis en cours d'édition si l'utilisateur était suffisamment avancé dans son devis.

La connexion pourrait également être améliorée, car actuellement l'utilisateur accède à l'application intranet d'Attila Système, se connecte, clique sur le lien de redirection vers notre application puis se reconnecte sur notre application. Les identifiants entrés durant la connexion à l'application d'Attila pourraient être récupérés et entrés automatiquement dès l'accès à notre application.

## 4-3-Defrenne Allan

### 4-3-1Page d'accueil

#### 4-3-1-1-Présentation

La page d'accueil est la page de navigation de l'application web, elle permet de choisir a quelle fonctionnalité le franchisé veut accéder et le redirige vers cette fonctionnalité une fois le choix effectué.

Les différentes fonctionnalités que la page d'accueil propose sont l'édition d'un devis d'entretien, de réparation, de rénovation et de nouvelle toiture.

#### 4-3-1-2-Description de la classe « WFAccueil »

##### Début

##### Méthodes

Signature	Rôle	Paramètres	Retour
+Navigation_Accueil ( <a href="#">object</a> , <a href="#">eventargs</a> )	Permet de rediriger et de garder en mémoire le choix de l'utilisateur.	Les deux paramètres sont les paramètres requis pour appeler la méthode depuis un objet.	Ne retourne rien.

##### Fin

#### 4-3-1-3-Test unitaire

Pour le test de cette page nous suivons le chemin du franchisé qui veut accéder aux différentes fonctionnalités de l'application.

##### Page d'accueil :

## ACCUEIL

Editer devis nouvelle toiture	Editer devis rénovation	Editer devis entretien	Editer devis réparation	Rechercher devis
-------------------------------	-------------------------	------------------------	-------------------------	------------------



Après clic sur rechercher :

On accède a la page de recherche de devis de l'application.

### Rechercher un devis:

Nom et prénom du client :	<input type="text"/>
Le numéro de devis :	<input type="text"/>
La date d'édition du devis :	<input type="text" value="Ex: AAAA-MM-JJ"/> <input type="button" value="Rechercher"/>

	iddevis	facture	numero	date	chantier	adr chantier	nature travaux	tva	visible
<a href="#">Sélectionner</a>	98		DF321498887	2012-05-16	intermarche masny	rue de monchecourt	nouvelle toiture	19,6	non
<a href="#">Sélectionner</a>	100		DF442455683	2012-05-16	exemple revue 2	90 rue de la revue	demonstration	19,6	oui
<a href="#">Sélectionner</a>	101		DF139443583	2012-05-23	koko	oko	jojo	19,6	non
<a href="#">Sélectionner</a>	102		DF442982909	2012-05-23	poo	poou	poouo	19,6	non
<a href="#">Sélectionner</a>	103		DF624119390	2012-05-23	yip	yoei	yeas	19,6	non
<a href="#">Sélectionner</a>	104		DF477834855	2012-05-23	kkjh	uuhuh	uhuh	19,6	non
<a href="#">Sélectionner</a>	105		DF362070135	2012-05-23	kevin	next	jojo	19,6	non
<a href="#">Sélectionner</a>	106		DF744094148	2012-05-23	kkjh	oko	demonstration	10,57	non
<a href="#">Sélectionner</a>	107		DF236894228	2012-05-23	koko	next	demonstration	1,87	non
<a href="#">Sélectionner</a>	108		DF569346475	2012-05-23	poo	oko	uhuh	10,57	non
<a href="#">Sélectionner</a>	109		DF898150368	2012-05-23	intermarche masny	poou	next	19,6	non
<a href="#">Sélectionner</a>	110		DF965471323	2012-05-23	next	yoei	next	10,87	non
<a href="#">Sélectionner</a>	111		DF805061827	2012-05-24	plow	plowed	lowed	19,6	non
<a href="#">Sélectionner</a>	112		DF196093320	2012-05-24	cleb	clob	clib	19,6	non
<a href="#">Sélectionner</a>	113		DF960696862	2012-05-24	seot	soet	toes	19,6	non
1 2 3									

Après clic sur une des édition de devis :

On accède à la page client afin de commencer l'édition du devis.

### Informations client:

<input type="checkbox"/> Entreprise	<input type="checkbox"/> Le client est déjà enregistré
Nom :	<input type="text"/>
Prénom :	<input type="text"/>
Adresse :	<input type="text"/>
CodePostal :	<input type="text"/>
Ville :	<input type="text"/>
Téléphone :	<input type="text"/>
<input type="button" value="Annuler"/>	<input type="button" value="Suivant"/>

## 4-3-2-Page client

### 4-3-2-1-Présentation

La page client est la page permettant de remplir les données relatives aux client pour l'édition du devis. Les données demandés dans cette page sont le nom du client, son prénom, son adresse, son code postal, sa ville et son numéro de téléphone. Si le client est une entreprise, une organisation, une commune ou autre le prénom n'est plus requis.

Il est aussi possible de rechercher un client existant grâce a une interface de recherche. Avec cette interface le franchisé pourra modifier les données du client si besoins ait, puis de le sélectionner afin de continuer l'édition du devis avec les données du client existant.

### 4-3-2-2-Description de la classe « WFClient »

#### Début

#### Variables Globales

-string Reponse  
IdFranchise  
IdClient  
Requete

-BDD ActionBDD

#### Méthodes

Signature	Rôle	Paramètres	Retour
+Suivant_Click (object , eventargs)	Permet selon le choix de récupérer les paramètres requis pour remplir la table contact et remplir client si le client est un nouveau client.	Les deux paramètre sont les paramètre requis pour appeler la méthode depuis un objet.	Ne retourne rien.
+Annuler_Click (object , eventargs)	Permet de retourner a la page d'accueil.	Les deux paramètre sont les paramètre requis pour appeler la méthode depuis un objet.	Ne retourne rien.
+Rechercher_Click (object , eventargs)	Permet de récupérer les client correspondant au critères de recherche et de les afficher dans le data gridview.	Les deux paramètre sont les paramètre requis pour appeler la méthode depuis un objet.	Ne retourne rien.

+Check_change (object , eventargs)	Permet de gérer les élément visible ou non après les différents choix possible.	Les deux paramètre sont les paramètre requis pour appeler la méthode depuis un objet.	Ne retourne rien.
+Recup_id_client _Datagrid()	Permet de récupérer l'id du client sélectionné dans le datagrid.	Aucun paramètre	Ne retourne rien.
+Remp_Table_ Contact()	Permet de remplir la table contact.	Aucun paramètre	Ne retourne rien.
+Recup_id_franchise ( )	Récupère l'id	Aucun paramètre	Ne retourne rien.
+redirection()	Permet la redirection selon le choix fait a l'accueil entre les différentes éditions	Aucun paramètre	Ne retourne rien.

Fin

#### 4-3-2-3-Test unitaire

Pour un particulier :

### Informations client:

☐ Entreprise
☐ Le client est déjà enregistré

Nom :

Prénom :

Adresse :

CodePostal :

Ville :

Téléphone :

Une fois les différents champs remplis on clique sur suivant ce qui a pour effet le remplissage de la table client\_prosp avec les valeurs des différentes textbox, ainsi que le remplissage de la table contact avec les données de l'utilisateur connecté. Si le client existe déjà l'application se charge de ne pas le rentrer et utilise le client déjà créé afin de poursuivre, un système similaire est mis en place pour la table contact si le contact est déjà lié au client on utilise le contact déjà existant.

Voici la requête qui insère les données dans la table client\_prosp, la requête pour la table client est similaire.

```
Requete = "INSERT INTO client_prosp (denomination,adresse,code_postal,ville,telephone,idfranchise)VALUES('" + denomination + "','" + TbAdresse.Text + "','" + TbCodePostal.Text + "','" + TbVille.Text + "','" + TbTelephone.Text + "','" + IdFranchise + "')";
```

Une fois les tâches effectuées l'utilisateur est redirigé vers la page d'édition choisie précédemment sur la page d'accueil. Et le client est bien inséré dans la base de données.

	idclient_prosp	denomination	adresse	code_postal	ville	telephone	idfranchise
1	1	edmond labbe	817 rue Charles-Bourseul	59500	douai	0627000000	1
2	5	INTERMARCHE	Rue de Monchecourt	59176	masny	0627000000	1
3	6	Lecompte Pierre	56 rue la mote	59200	douai	0327564897	1

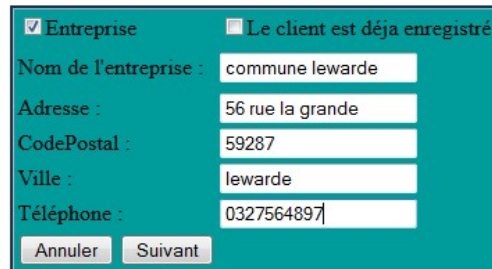
Ainsi que le contact qui lui est attribué.

	tact	nom	prenom	genre	telephone	email	fax	mobile	idclient
66		once	kevin	N/A	0627919394	kevin.once@hotmail.fr	0600000000	N/A	1
67		defrenne	allan	N/A	0627848381	allan.defrenne@yahoo.fr	0600000000	N/A	6

Pour une entreprise, organisation, une commune ... :

Seule l'interface change pour une entreprise le fonctionnement reste le même pour la suite à condition que les champs soient bien remplis.

## Informations client:



The screenshot shows a web form titled 'Informations client:'. It has two checkboxes at the top: 'Entreprise' (checked) and 'Le client est déjà enregistré' (unchecked). Below these are five text input fields: 'Nom de l'entreprise :', 'Adresse :', 'CodePostal :', 'Ville :', and 'Téléphone :'. The values entered are 'commune lewarde', '56 rue la grande', '59287', 'lewarde', and '0327564897' respectively. At the bottom are two buttons: 'Annuler' and 'Suivant'.

<input checked="" type="checkbox"/> Entreprise	<input type="checkbox"/> Le client est déjà enregistré
Nom de l'entreprise :	commune lewarde
Adresse :	56 rue la grande
CodePostal :	59287
Ville :	lewarde
Téléphone :	0327564897
<input type="button" value="Annuler"/> <input type="button" value="Suivant"/>	

Le changement d'interface s'effectue après le changement d'état de la propriété Checked de la checkbox et provoque la disparition du champs prénom inutile pour une entreprise.

Pour un client existant :

L'interface est le traitement vont alors êtres modifiés. Tout d'abord l'interface à la place du formulaire à remplir l'utilisateur ce retrouve une sur interface de recherche suite au changement d'état de la checkbox, afin de rechercher le client déjà enregistré via son nom et prénom.

Voici l'interface :

**Informations client:**

☐ Entreprise ☒ Le client est déjà enregistré

Nom :

Prénom :

<u>idclient_prosp</u>	<u>denomination</u>	<u>adresse</u>	<u>code postal</u>	<u>ville</u>	<u>telephone</u>	<u>idfranchise</u>
<a href="#">Modifier</a> <a href="#">Sélectionner</a>	6	Lecompte Pierre	56 rue la mote	59200	douai	0327564897 1

Sélectionner le client avant de valider

Requête permettant de récupérer les données a utiliser pour remplir le gridview.

```
Requete = "SELECT idclient_prosp,denomination,adresse,code_postal,ville,telephone,idfranchise FROM client_prosp WHERE denomination='" + Denomination + "'";
```

Le client recherché est le client précédemment enregistré par l'utilisateur les résultats de la recherche sont chargés dans le gridview. Une fois trouvé on peut effectuer deux actions à celui-ci tout d'abord l'action modifier qui permet de rendre les champs du gridview modifiable afin de permettre le changement des données si besoin (cette fonctionnalité est gérée par le gridview). Puis l'action sélectionner qui permet de sélectionner le client avec qui poursuivre le devis (cette fonctionnalité est gérée par le gridview).

Modification d'un champ par exemple l'adresse :

## Informations client:

☐ Entreprise ☒ Le client est déjà enregistré

Nom :

Prénom :

	<u>idclient prosp</u>	<u>denomination</u>	<u>adresse</u>	<u>code postal</u>	<u>ville</u>	<u>telephone</u>	<u>idfranchise</u>
Mettre à jour Annuler 6		Lecompte Pierre	59 rue la mote	59200	douai	0327564897	1

Sélectionner le client avant de valider

Clic sur mettre a jour puis vérification dans la base de donnée.

	idclient_prosp	denomination	adresse	code_postal	ville	telephone	idfranchise
1	1	edmond labbe	817 rue Charles-Bourseul	59500	douai	0627000000	1
2	5	INTERMARCHE	Rue de Monchecourt	59176	masny	0627000000	1
3	6	Lecompte Pierre	59 rue la mote	59200	douai	0327564897	1

Une fois les données mises à jour l'utilisateur peut poursuivre l'édition du devis si il a sélectionné le client avec lequel il veut poursuivre.

Le devis se poursuit donc avec la redirection vers l'édition choisie.

### 4-3-3-Page de Recherche

#### 4-3-3-1-Présentation

La page de recherche a pour fonction de permettre de retrouver un devis selon différents filtres qui sont le nom du client, le numéro de devis et la date d'édition, un quatrième filtre est mentionné dans le cas d'utilisation qui ne sera pas utilisé ici car il sera destiné à l'administrateur qui aura accès à tous les devis, alors que le franchisé lui n'aura accès qu'aux devis qu'il aura lui-même édités.

Une fois que le devis recherché est trouvé, on peut le sélectionner afin de lancer une action sur le devis. Il y a quatre actions possibles : modifier, imprimer, sauvegarder, valider. Après le choix de l'action effectuée, le franchisé est alors redirigé vers la page qui convient si besoin ou alors l'action est directement lancée (Ex : modifier entraîne une redirection, valider qu'en a-t-elle effectuée l'action directement sans redirection).

#### 4-3-3-2-Description de la classe « WFRechercher »

##### Début

##### Variables Globales

- string Requete
- Reponse
- BDD ActionBDD

##### Méthodes

Signature	Rôle	Paramètres	Retour
+Page_Load(object , EventArgs )	Permet au chargement de la page de charger le gridview avec tous les devis visibles par l'utilisateur,	Les deux paramètres sont les paramètres requis pour appeler la méthode depuis un objet.	Ne retourne rien.
+Rechercher_Click(object , EventArgs)	Permet de récupérer les devis recherchés par l'utilisateur et de charger le datagrid avec	Les deux paramètres sont les paramètres requis pour appeler la méthode depuis un objet.	Ne retourne rien.
+Choix_Requete()	Permet de mettre en place la requête de recherche	Aucun paramètre	Ne retourne rien.
+valider_click(object, EventArgs)	Permet de valider le devis c'est-à-dire de mettre son champs	Les deux paramètres sont les paramètres requis pour appeler la	Ne retourne rien.

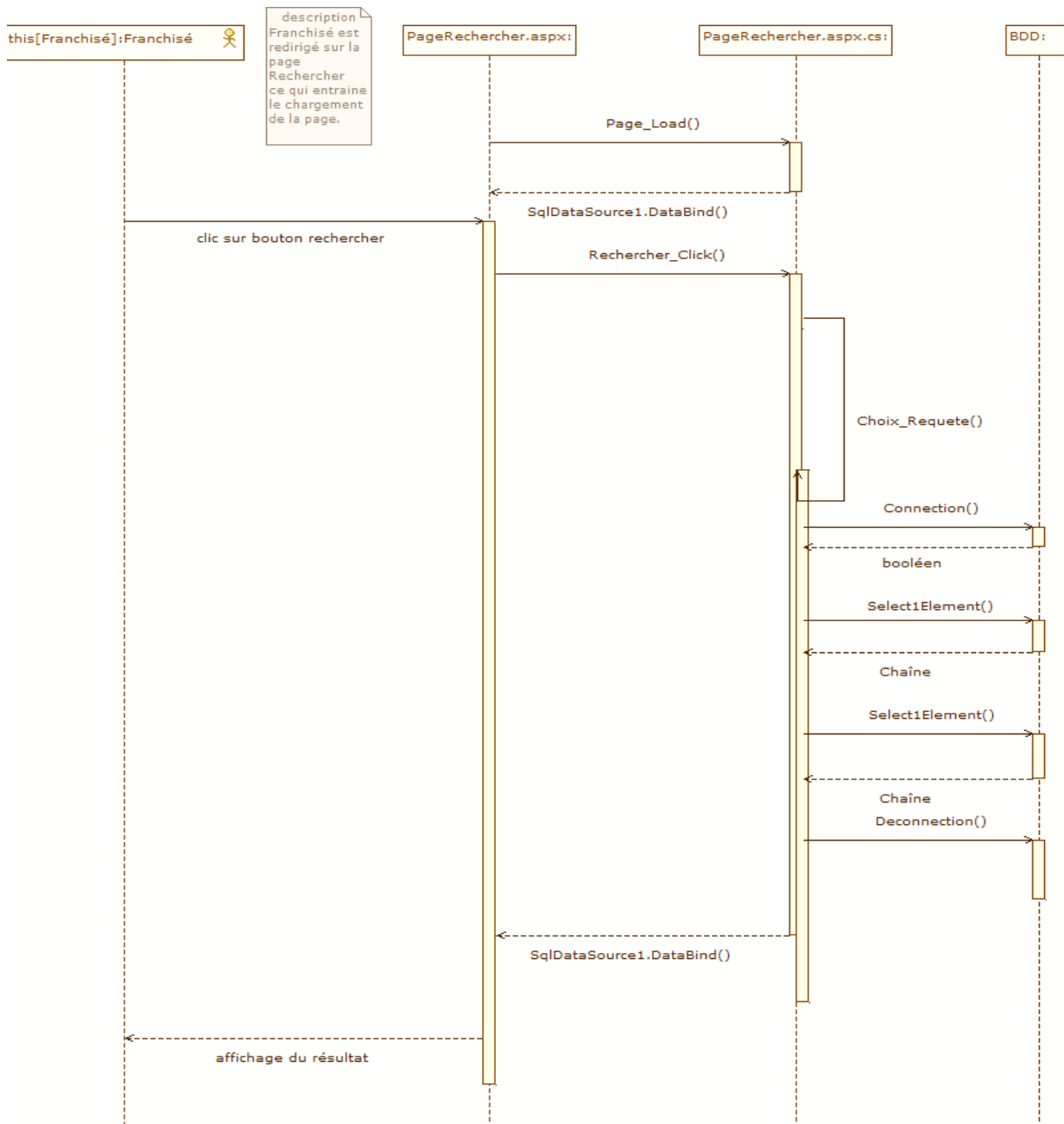


	visible a ouai.	méthode depuis un objet.	
Sauvegarder_Click ( <b>object</b> , <b>EventArgs</b> )	Permet de lancer la sauvegarde du PDF	Les deux paramètre sont les paramètre requis pour appeler la méthode depuis un objet.	Ne retourne rien.
Création_PDF( <b>string</b> )	Permet de créer le PDF via la classe CréateurPDF	Le paramètre est une chaîne comportant le Numéro de devis.	Ne retourne rien.

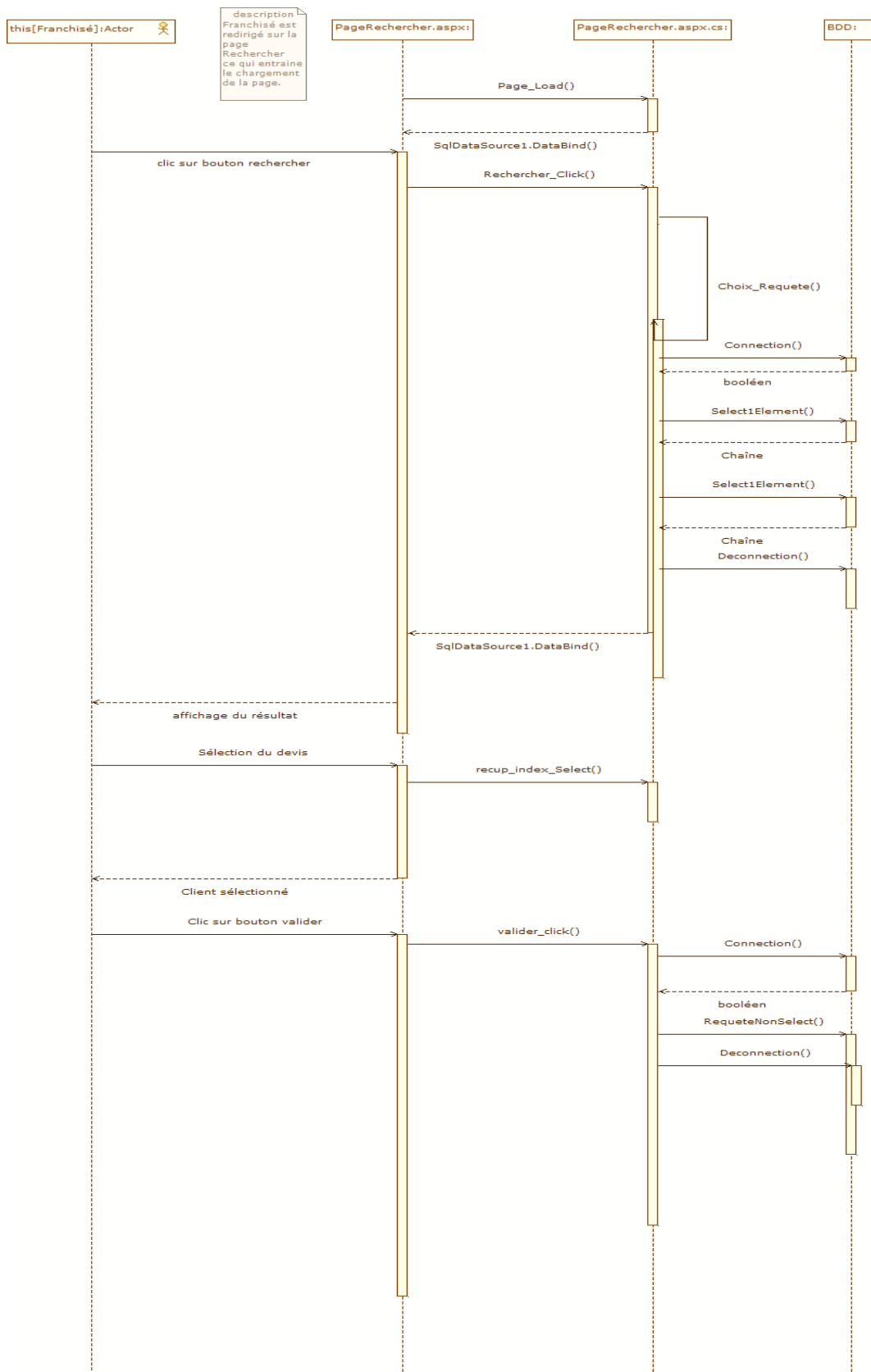
**Fin**

### 4-3-3-3-Diagramme de séquence

Diagramme pour le cas d'utilisation Rechercher



## Diagramme pour le cas d'utilisation Valider



### 4-3-3-4-Test unitaire

Arrivé sur la page la page rechercher l'utilisateur a directement tout ses devis chargés dans le gridview on obtient alors cette page.

Rechercher un devis:

Nom et prénom du client :   
 Le numéro de devis :   
 La date d'édition du devis : Ex AAAA-MM-JJ

	<u>iddevis</u>	<u>facture</u>	<u>numero</u>	<u>date</u>	<u>chantier</u>	<u>adr chantier</u>	<u>nature travaux</u>	<u>tva</u>	<u>visible</u>
<a href="#">Sélectionner</a>	98		DF321498887	2012-05-16	intermarche masny	rue de monchecourt	nouvelle toiture	19,6	non
<a href="#">Sélectionner</a>	100		DF442455683	2012-05-16	exemple revue 2	90 rue de la revue	demonstration	19,6	oui
<a href="#">Sélectionner</a>	101		DF139443583	2012-05-23	koko	oko	jojo	19,6	non
<a href="#">Sélectionner</a>	102		DF442982909	2012-05-23	poo	pooo	poooo	19,6	non
<a href="#">Sélectionner</a>	103		DF624119390	2012-05-23	yip	yoei	yeas	19,6	non
<a href="#">Sélectionner</a>	104		DF477834855	2012-05-23	kkjh	uuhuh	uhuh	19,6	non
<a href="#">Sélectionner</a>	105		DF362070135	2012-05-23	kevin	next	jojo	19,6	non
<a href="#">Sélectionner</a>	106		DF744094148	2012-05-23	kkjh	oko	demonstration	10,57	non
<a href="#">Sélectionner</a>	107		DF236894228	2012-05-23	koko	next	demonstration	1,87	non
<a href="#">Sélectionner</a>	108		DF569346475	2012-05-23	poo	oko	uhuh	10,57	non
<a href="#">Sélectionner</a>	109		DF898150368	2012-05-23	intermarche masny	pooo	next	19,6	non
<a href="#">Sélectionner</a>	110		DF965471323	2012-05-23	next	yoei	next	10,87	non
<a href="#">Sélectionner</a>	111		DF805061827	2012-05-24	plow	plowed	lowed	19,6	non
<a href="#">Sélectionner</a>	112		DF196093320	2012-05-24	cleb	clob	clib	19,6	non
<a href="#">Sélectionner</a>	113		DF960696862	2012-05-24	seot	soet	toes	19,6	non

1 2 3

Arrivé sur cette page l'utilisateur peut alors effectuer une recherche par exemple avec le filtre numéro de devis dans le cas présent DF442982909

Rechercher un devis:

Nom et prénom du client :   
 Le numéro de devis : DF442982909  
 La date d'édition du devis : Ex AAAA-MM-JJ

	<u>iddevis</u>	<u>facture</u>	<u>numero</u>	<u>date</u>	<u>chantier</u>	<u>adr chantier</u>	<u>nature travaux</u>	<u>tva</u>	<u>visible</u>
<a href="#">Sélectionner</a>	102		DF442982909	2012-05-23	poo	pooo	poooo	19,6	non

La requête pour chargé est construite de cette manière.

chaîne Réponse ← nul  
chaîne Requête ← nul  
chaîne ImbricRequete1 ← nul  
chaîne Date ← nul

Si ZoneDeLaDate.contenu ← Ex: AAAA-MM-JJ ou nul

Alors Date ← nul

Sinon Date ← ZoneDeLaDate.contenu

FinSi

Si ZoneDuNomClient.contenu ← nul

Alors ImbricRequete1 ← nul

Sinon Requête ← requête permettant la sélection de l'idclient\_prosp de la table  
                                client\_prosp quand dénomination ← TbNomClient

        lancement de la requête

        ImbricRequete1 ← résultat de la requête

        Requête ← requête permettant la sélection de l'idcontact de la table  
                                contact quand idclien\_prosp ← ImbricRequete1

        lancement de la requête

        Réponse ← résultat de la requête

FinSi

Requête ← requête permettant la sélection tout les champs de la table devis\_facture  
          quand numéro ← ZoneDuNumeroDeDevis.contenu et date ← Date  
          et idcontact ← Réponse

Vérification de sa présence dans la base de donnée.

	iddevis_facture	numero	date	chantier	adr_chantier	nature_travaux	tva	c
1	98	DF321498887	2012-05-16	intermarche masny	rue de monchecourt	nouvelle toiture	19,6	1
2	100	DF442455683	2012-05-16	exemple revue 2	90 rue de la revue	demonstration	19,6	1
3	101	DF139443583	2012-05-23	koko	oko	jojo	19,6	1
4	102	DF442982909	2012-05-23	poo	pooo	poooo	19,6	1
5	103	DF624119390	2012-05-23	yip	yoei	yeas	19,6	1
6	104	DF477834855	2012-05-23	kkjh	uuhuh	uhuh	19,6	1
7	105	DF362070135	2012-05-23	kevin	next	jojo	19,6	1
8	106	DF744094148	2012-05-23	kkjh	oko	demonstration	10,57	1
9	107	DF236894228	2012-05-23	koko	next	demonstration	1,87	1
10	108	DF569346475	2012-05-23	poo	oko	uhuh	10,57	1
11	109	DF898150368	2012-05-23	intermarche masny	pooo	next	19,6	1
12	110	DF965471323	2012-05-23	next	yoei	next	10,87	1
13	111	DF805061827	2012-05-24	plow	plowed	lowed	19,6	1
14	112	DF196093320	2012-05-24	cleb	clob	clib	19,6	1
15	113	DF960696862	2012-05-24	seot	soet	toes	19,6	1

Une fois le devis sélectionné l'utilisateur peut effectuer 4 actions sur le devis : modifier, sauvegarder, valider et imprimer.

Dans le cas de la validation, l'utilisateur n'a qu'à appuyer sur le bouton valider ce qui entraîne une update dans la base afin de changer son champ « visible » à oui.

## Rechercher un devis:

Nom et prénom du client :	<input type="text"/>
Le numéro de devis :	<input type="text" value="DF442982909"/>
La date d'édition du devis :	<input type="text" value="Ex: AAAA-MM-JJ"/> <input type="button" value="Rechercher"/>

	<u>iddevis</u>	<u>facture</u>	<u>numero</u>	<u>date</u>	<u>chantier</u>	<u>adr chantier</u>	<u>nature travaux</u>	<u>tva</u>	<u>visible</u>
<u>Sélectionner</u>	102		DF442982909	2012-05-23	poo	pooo	poooo	19,6	oui

Le devis numero DF442982909 est validé.

<input type="button" value="Modifier"/>	<input type="button" value="Valider"/>	<input type="button" value="Sauvegarder"/>	<input type="button" value="Imprimer"/>
---	--	--	---

Vérification dans la base de données :

	iddevis_facture	numero	visible
1	98	DF321498887	non
2	100	DF442455683	oui
3	101	DF139443583	non
4	102	DF442982909	oui
5	103	DF624119390	non
6	104	DF477834855	non

Dans le cas de la sauvegarde et de l'impression du devis au format client l'application doit tout d'abord générer un PDF contenant le résumé du devis. Pour cela la classe CreateurPDF sera utilisée. Elle sera constituée de méthodes utilisant les méthodes des dll suivantes [iTextSharp.text.pdf.dll](#) et [iTextSharp.text.dll](#) qui sont des bibliothèques permettant la création de PDF ainsi que leur édition.

La classe CreateurPDF est à l'heure actuelle celle-ci.

## Début

### Variables Globales

-[iTextSharp.text.Document](#) Doc  
 -[PdfWriter](#) Writer;  
 -[iTextSharp.text.pdf.PdfContentByte](#) cb;

### Méthodes

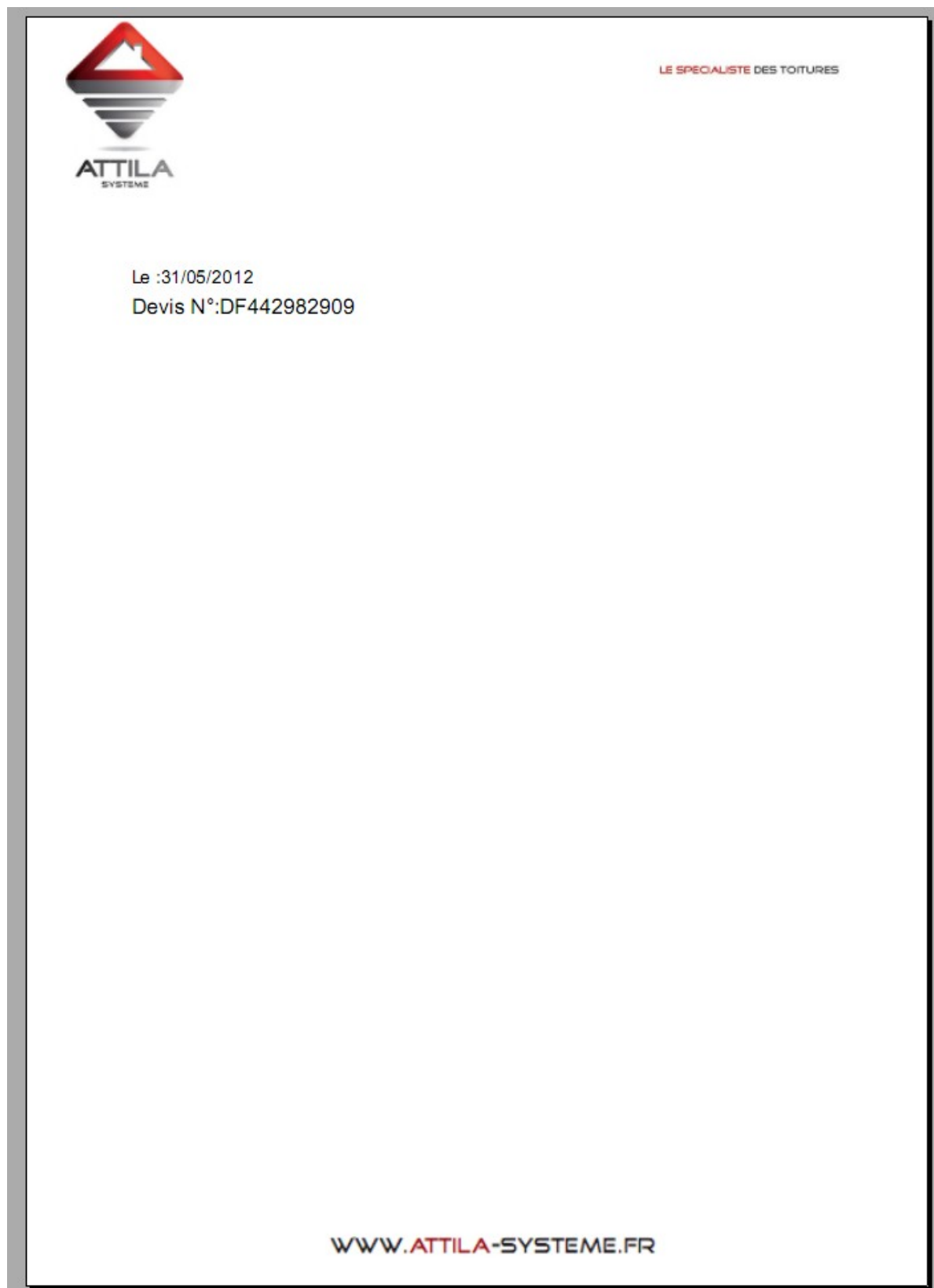
Signature	Rôle	Paramètres	Retour
+CreationDocument ( <a href="#">string</a> )	Permet la création d'un document PDF	La chaîne correspond au chemin du répertoire ou vous voulez créer le fichier se terminant par le nom du fichier ex: @"C:\Users\Nom_du_compte_utilisateur\Desktop\Nom_fichier.pdf"	Ne retourne rien

+Ajout_Image (string , float , float )	Permet d'ajouter une image à un endroit donné	La chaîne correspond au chemin menant à l'image ex: @”C:\Users\Nom_du_compte_utilisateur\Desktop\Nom_image.son_extention” le premier float est la position de l'image sur l'axe X et l'autre la position sur l'axe Y (attention le 0 est en bas de page)	Ne retourne rien
+Ajout_Text (string,float,float,float, float , float)	Permet d'ajouter du texte à un endroit donné	La chaîne correspond au texte à afficher les float correspondent au valeur permettant de gérer les dimensions la position de la cellule où est stocké le texte	Ne retourne rien
+Ouvrir_Document()	Permet d'ouvrir le document PDF c'est à dire de permettre le travail sur celui-ci	Pas de paramètres	Ne retourne rien
+Fermer_Document()	Permet de fermer le document PDF c'est à dire de ne plus permettre le travail celui-ci	Pas de paramètres	Ne retourne rien
+Ajout_Haut_Bas_Page()	Permet de mettre automatiquement les élément en haut et bas de page	Pas de paramètres	Ne retourne rien

Fin



Après le clic sur le bouton Sauvegarder l'application génère un PDF actuellement celui-ci avec l'exemple du devis N° DF442982909 de même pour l'impression.



#### 4-3-4 Conclusion

Sur les fonctionnalités finies il reste quelques détails à corriger exemple la date demandée en version anglaise ainsi que le design et l'ajout de dynamisme mais des choses plus importantes aussi tel que la gestion des exceptions. Pour le reste l'édition du devis au format PDF est en cours, une fois celle-ci finie l'impression et la sauvegarde pourront être mises en place puis les 2 éditions mises en place.

Pour ma part je pense que beaucoup de fonctionnalités de ma partie sont encore à faire, des recherches trop longues m'auront fait perdre du temps du fait de la mauvaise organisation de celle-ci et une mauvaise gestion de mon temps.

## 5 Annexes :

### 5-1 : Devis d'attila système (Page 1 et 2 non modifiés)



LE SPÉCIALISTE DES TOITURES

Le : 25-01-2011

Devis N° : DE-2-2011-5.1

Communauté de communes de suippes  
A l'attention de Monsieur Rochat  
pl Hôtel de Ville  
51600 - SUIPPES

Tel : 0326700855 ou 0627421552

#### CHANTIER

TOITURE BASSE DE L'ÉGLISE DE SUIPPES

#### Nature des travaux

DEMOUSSAGE ET TRAITEMENT FONGICIDE DE LA TOITURE DE L'ÉGLISE DE SUIPPES



ATTILA FORMATION  
ZAC FICHET BAUCHE - 51110 - BAZANCORT  
- Mail : formation@attila-systeme.fr  
SARL au capital de : 7500 € - APE : 4391B - SIRET : 52184472000016  
**WWW.ATTILA-SYSTEME.FR**

Page 1/4



**LE SPÉCIALISTE DES TOITURES**

Devis N° : DE-2-2011-5.1

## INSTALLATION DE CHANTIER

Déplacement, approvisionnement du chantier en matériaux et matériels.

Quantité	Unité	Prix UHT	total
1	ens	124.00 €	124.00 €
sous-total INSTALLATION DE CHANTIER : 124.00 €			

## SECURITE

Mise ne sécurité par camion nacelle, harnais.

Quantité	Unité	Prix UHT	total
1	ens	186.00 €	186.00 €
sous-total SECURITE : 186.00 €			

## NETTOYAGE DE LA TOITURE

Brossage de la toiture sur les parties basses de droite et de gauche de l'église.

Quantité	Unité	Prix UHT	total
330	m²	6.95 €	2 293.50 €

Traitement fongicide de la toiture basse de gauche et de droite de l'église

Quantité	Unité	Prix UHT	total
330	m³	3.40 €	1 122.00 €



sous-total NETTOYAGE DE LA TOITURE : 3415.50 €

ATTILA FORMATION  
ZAC FICHET BAUCHE - 51110 - BAZANCORT  
- Mail : formation@attila-systeme.fr  
SARL au capital de : 7500 € - APE : 4391B - SIRET : 52184472000016  
**WWW.ATTILA-SYSTEME.FR**

Page 2/4

## 5-2 : Code de la méthode PremierOperation de la page récapitulatif.

```
//instanciation de l'objet bdd pour utiliser les methodes de la classe.
BDD actionBDD = new BDD();

//instanciation des differentes variables et initialisation de certaine avec des id dans la base de donnée.
string iddevis = (string)Session["iddevis"];
string unite = "";
string idFirstTitre = actionBDD.SelectElement("select idtitre from titre where iddevis_facture="+iddevis+""");
string idFirstOperation = actionBDD.SelectElement("select idoperation from operation where idtitre=" + idFirstTitre + "");
string idFirstDoperation = actionBDD.SelectElement("select iddetail_operation from detail_operation where
idoperation="+idFirstOperation+""");
string idSecDoperation = (int.Parse(idFirstDoperation) + 1).ToString();
string idThrDoperation = (int.Parse(idSecDoperation) + 1).ToString();

//initialisation des differentes zone de texte avec leur valeur prise dans la base de donnée selon les conditions dans la requete
TitreOpD.Text=actionBDD.Select1Element("select titre from titre where idtitre="+idFirstTitre+""");
MontantTitreD.Text = actionBDD.Select1Element("select montant_titre from titre where idtitre=" + idFirstTitre + "");

PosteD.Text=actionBDD.Select1Element("select poste from operation where idtitre="+idFirstTitre+""");
MontantD.Text=actionBDD.Select1Element("select prix_total_ht from operation where idtitre="+idFirstTitre+""");
PrixTopD.Text = MontantD.Text;

TypeC.Text=actionBDD.SelectElement("select type from detail_operation where idoperation="+idFirstOperation+""");

unite= actionBDD.SelectElement("select idunite from detail_operation where idoperation=" + idFirstOperation + "");

switch (unite)
{
    case "1":

        UniteC.Text = actionBDD.Select1Element("select libelle from unite where idunite='1'");

        break;
    case "2":

        UniteC.Text = actionBDD.Select1Element("select libelle from unite where idunite='2'");

        break;
    case "3":

        UniteC.Text = actionBDD.Select1Element("select libelle from unite where idunite='3'");

        break;
    case "4":

        UniteC.Text = actionBDD.Select1Element("select libelle from unite where idunite='4'");

        break;
}

QuantiteC.Text = actionBDD.SelectElement("select quantite from detail_operation where idoperation=" + idFirstOperation + "");
PrixTotalC.Text = actionBDD.SelectElement("select prix_total from detail_operation where idoperation=" + idFirstOperation + "");
PrixUnitC.Text = (Math.Round((Double.Parse(PrixTotalC.Text)) / (Double.Parse(QuantiteC.Text)), 3)).ToString();
```

```

TypeC1.Text = actionBDD.SelectElement("select type from detail_operation where idoperation=" + idFirstOperation + " and
iddetail_operation=" + idSecDoperation + "");
unite = actionBDD.SelectElement("select idunite from detail_operation where idoperation=" + idFirstOperation + " and
iddetail_operation=" + idSecDoperation + "");

switch (unite)
{
    case "1":

        UniteC1.Text = actionBDD.Select1Element("select libelle from unite where idunite='1'");

        break;
    case "2":

        UniteC1.Text = actionBDD.Select1Element("select libelle from unite where idunite='2'");

        break;
    case "3":

        UniteC1.Text = actionBDD.Select1Element("select libelle from unite where idunite='3'");

        break;
    case "4":

        UniteC1.Text = actionBDD.Select1Element("select libelle from unite where idunite='4'");

        break;
}

QuantiteC1.Text = actionBDD.SelectElement("select quantite from detail_operation where idoperation=" + idFirstOperation + "
and iddetail_operation=" + idSecDoperation + "");

PrixTotalC1.Text = actionBDD.SelectElement("select prix_total from detail_operation where idoperation=" + idFirstOperation + "
and iddetail_operation=" + idSecDoperation + "");

PrixUnitC1.Text = (Math.Round((Double.Parse(PrixTotalC1.Text)) / (Double.Parse(QuantiteC1.Text)), 3)).ToString();


TypeC2.Text = actionBDD.SelectElement("select type from detail_operation where idoperation=" + idFirstOperation + " and
iddetail_operation=" + idThrDoperation + "");
unite = actionBDD.SelectElement("select idunite from detail_operation where idoperation=" + idFirstOperation + " and
iddetail_operation=" + idThrDoperation + "");

switch (unite)
{
    case "1":

        UniteC2.Text = actionBDD.Select1Element("select libelle from unite where idunite='1'");

        break;
    case "2":

        UniteC2.Text = actionBDD.Select1Element("select libelle from unite where idunite='2'");

        break;
    case "3":

        UniteC2.Text = actionBDD.Select1Element("select libelle from unite where idunite='3'");

        break;
    case "4":

        UniteC2.Text = actionBDD.Select1Element("select libelle from unite where idunite='4'");

        break;
}

```

```

        QuantiteC2.Text = actionBDD.SelectElement("select quantite from detail_operation where idoperation=" + idFirstOperation + ""
and iddetail_operation=" + idThrDoperation + "");

        PrixTotalC2.Text = actionBDD.SelectElement("select prix_total from detail_operation where idoperation=" + idFirstOperation + ""
and iddetail_operation=" + idThrDoperation + "");

        PrixUnitC2.Text = (Math.Round((Double.Parse(PrixTotalC2.Text)) / (Double.Parse(QuantiteC2.Text)), 3)).ToString();
    }
}

```

### 5-3 : Code de la méthode logmdp de la page de Connection.

```
public void logmdp(object sender,EventArgs e)
{
    string Login = LogText.Text; //Recuperation des identifiants dans des variables
    string Mdp = MdpText.Text;
    bool test = true;

    BDD TestConec = new BDD();

    if (Login != "" && Mdp != "") //verifications que le login et le mot de passe ne sont pas vide
    {

        Login = LogText.Text;
        Mdp = MdpText.Text;

        test = TestConec.Connection(Login,Mdp); //test si les identifiants existent dans la base de donnée

        if (test == true)          //si ils existent initialisation des variables de session avec le login et l'id utilisateur correspondant
        {
            Session["Login"] = Login;
            Session["ID"] = TestConec.Select1Element("SELECT idutilisateur FROM utilisateur WHERE login= " + Login + "");

            Response.Redirect("WFAccueil.aspx"); //redirection vers la page d'accueil
        }
        else          //Affichage du message d'erreur
        {

            DetailErreur.Visible = true;
            DetailErreur.Text = (string)Session["Erreur"];

        }

    }
    else //Affichage du message d'erreur
    {

        DetailErreur.Visible = true;
        DetailErreur.Text = "Veuillez entrer un nom d'utilisateur et un mot de passe";

    }
}
```



## 5-4 : Code de la méthode ValidOp de la page FormulaireOperation.

```
public void ValidOp(Object sender, EventArgs e)
{
    BDD actionBDD = new BDD(); //instanciation d'un objet BDD pour utiliser les methodes de la classe

    //initialisaiton des variables avec le contenu des différents TextBox
    string titreE = TitreC.Text, posteE = PosteC.Text, UniteE = UniteC.Text, quantiteE = QuantiteC.Text, PrixU =
    PrixUnitC.Text, TypeE=TypeC.Text;

    double MontantTitre = 0;
    //Calcul du prix total selon la quantite et le prix unitaire récupérés
    string Prix_TotalE = (Double.Parse(PrixU.Replace('.', ',')) * Double.Parse(quantiteE.Replace('.', ','))).ToString();

    if(Prix_TotalE.Contains(',')==true)
    {
        Prix_TotalE = Prix_TotalE.Replace('.', ',');
    }

    string idevis = (string)Session["idevis"];

    string attributValT="2,'" + titreE + "','non','" + MontantTitre + "','" + idevis + """;
    actionBDD.TitreBdd(attributValT); // utilisation de la methode TitreBdd pour remplir la table titre

    string idtitre=actionBDD.Select1Element("select max(idtitre) from titre ");

    string attributValO = "2,'" + posteE + "','" + quantiteE + "','500,'" + PrixU + "','" + Prix_TotalE + "','500','1,'" + idtitre + """;
    actionBDD.OperationBdd(attributValO); // utilisation de la methode ToperationBdd pour remplir la table operation

    string idoperation = actionBDD.Select1Element("select max(idoperation) from operation");

    string attributValDo = "" + TypeE + "','" + quantiteE + "','1,'" + quantiteE + "','" + Prix_TotalE + "','1,'" + idoperation + """;

    actionBDD.DetailOperationBdd(attributValDo); // utilisation de la methode DetailOperationBdd pour remplir la table
    Detail_Operation

}
```

## 5-5 : Code de la méthode DonneChantier de la page NouvelleToiture.

```
public void Donneechantier(object sender, EventArgs e)
{
    string idcontact = (string)Session["idcontact"]; //initialisation d'une variable avec l'idcontact contenu dans la variable de session
    Nchantier = NomChantier.Text; //initialisation des différents variables avec le contenu des différentes textbox ou droplist
    AdrrChantier = AdresseChantier.Text;
    NatureTrav = NatureTravaux.Text;
    tvaDecTemp = TauxTva.Text;
    NumeroDevis = (string)Session["NumeDevis"]; //initialisation d'une variable avec le numero de devis contenu dans la variable de session
    Double tvaDec, CoutHoraire, MontantTotal, CoutFourniture, Remise;

    if (Nchantier != "" && AdrrChantier != "" && NatureTrav != "" && tvaDecTemp != "") //on test si les champs ne sont pas vides
    {
        actionBDD.Connection((string)Session["Pseudo"], (string)Session["MotDePass"]); //on se connecte à la base grâce à la methode
        Connection de la classe BDD
        SqlConnection connection = (SqlConnection)Session["Connection"];

        if (tvaDecTemp.Contains('.') == true) //on test si les différentes variables contiennent un . Et le remplace par une virgule
        {
            tvaDec = Double.Parse(tvaDecTemp.Replace('.', ','));
        }
        else
        {
            tvaDec = Double.Parse(tvaDecTemp);
        }

        if (CoutHoraireTemp.Contains('.') == true)
        {
            CoutHoraire = Double.Parse(CoutHoraireTemp.Replace('.', ','));
        }
        else
        {
            CoutHoraire = Double.Parse(CoutHoraireTemp);
        }

        if (MontantTotalTemp.Contains('.') == true)
        {
            MontantTotal = Double.Parse(MontantTotalTemp.Replace('.', ','));
        }
        else
        {
            MontantTotal = Double.Parse(MontantTotalTemp);
        }

        if (CoutFournitureTemp.Contains('.') == true)
        {
            CoutFourniture = Double.Parse(CoutFournitureTemp.Replace('.', ','));
        }
        else
        {
            CoutFourniture = Double.Parse(CoutFournitureTemp);
        }

        if (RemiseTemp.Contains('.') == true)
        {
            Remise = Double.Parse(RemiseTemp.Replace('.', ','));
        }
        else
        {
            Remise = Double.Parse(RemiseTemp);
        }
    }
}
```

```

    }

    string attribut =
"numero,date,chantier,adr_chantier,nature_travaux,tva,cout_horaire,montant_total,cout_fournitures,temps_total,visible,facture"
    + ",avoir_remise,numero_commande,date_reglement,devis,commentaire,version,bon_commande,idcontact";
    SqlCommand command = new SqlCommand("INSERT INTO devis_facture(" + attribut + ")VALUES(" + NumeroDevis + "," +
Date + "," + Nchantier + "," + AdrChantier + "," + NatureTrav + ", @TVA, @CoutH,@MontantT,@CoutF," + TempsTotal + "," +
Visibilité + "," + FactureNbr + "," + AvoirNbr + ",@RemiseT," + NumCommande + "," + DatePay + "," + DevisN + "," + Com + "," +
VersionDec + "," + BonCommande + "," + idcontact + ")", connection);

    command.Parameters.Add(new SqlParameter("@TVA", SqlDbType.Float));
    command.Parameters["@TVA"].Value = "" + Convert.ToDecimal(tvaDec) + "";
    command.Parameters.Add(new SqlParameter("@CoutH", SqlDbType.Float, 18));
    command.Parameters["@CoutH"].Value = "" + Convert.ToDecimal(CoutHoraire) + "";
    command.Parameters.Add(new SqlParameter("@MontantT", SqlDbType.Float, 18));
    command.Parameters["@MontantT"].Value = "" + Convert.ToDecimal(MontantTotal) + "";
    command.Parameters.Add(new SqlParameter("@CoutF", SqlDbType.Float, 18));
    command.Parameters["@CoutF"].Value = "" + Convert.ToDecimal(MontantTotal) + "";
    command.Parameters.Add(new SqlParameter("@RemiseT", SqlDbType.Float, 18));
    command.Parameters["@RemiseT"].Value = "" + Convert.ToDecimal(Remise) + "";
    //Execution de la requete Insert into qui crée le devis
    command.ExecuteNonQuery();

    Session["iddevis"] = actionBDD.Select1Element("select iddevis_facture from devis_facture where numero=" + NumeroDevis + "");

    tuile = (double)Session["tuile"];

    liteau = (double)Session["liteau"];

    soustoit = (double)Session["soustoit"];

    PrixHfT = (double)Session["PrixTT"];

    actionBDD.PremiereOperation(PrixHfT, tuile, liteau, soustoit); //création du titre, de l'operation et les details operation grâce à la
methode PremiereOperation de la classe BDD
    actionBDD.Deconnection(connection);

    Response.Redirect("FormulaireOperation.aspx"); //redirection vers la page FormulaireOperation
}
else // affiche Message d'erreur
{
    if ( LErreurF2.Visible == false)
    {
        LErreurF2.Visible = true;
        TErreurF2.Visible = true;
    }

    TErreurF2.Text = "Veuillez compléter la totalité des champs";

}
}
}

```

## 5-6 : Code de la méthode CalculDimension de la page NouvelleToiture.

```
public void CalculDimension(object sender, EventArgs e)
{
    bool valide = true;
    ErreurLabel.Visible = false;
    DetailErreur.Visible = false;
    DetailErreur.Text = "";
    string tempA, tempB, PrixH = PrixHt.Text;
    Double Largeur = 0, Longueur = 0, Surface = 0, tuilePmC = 0, PureauMoy = 0.33, NbrPlan = 0, mlRive = 0, mlFaitiere = 0, PrixUT =
0, mlLit = 0;
    tempA = LargeurNbr.Text; //recuperation de la longueur et la largeur des textbox
    tempB = LongueurNbr.Text;

    if (tempA != "" && tempB != "" || tempB!=" " && LargeurNbr.Visible==false) //test si les champs ne sont pas vides
    {
        if(LargeurNbr.Visible!=false)
            Largeur = Double.Parse(tempA.Replace('.', ','));

        Longueur = Double.Parse(tempB.Replace('.', ','));

        switch (NbrPlanS.Selected.Value) //initialisation du nombre de plan grâce a la selection dans le droplist
        {
            case "1": NbrPlan = 1;
                break;
            case "2": NbrPlan = 2;
                break;
            case "3": NbrPlan = 3;
                break;
            case "4": NbrPlan = 4;
                break;
            default:

                if (ErreurLabel.Visible==false)
                {
                    ErreurLabel.Visible = true;
                    DetailErreur.Visible = true;
                }
                valide = false;
                DetailErreur.Text = DetailErreur.Text + "Veuillez sélectionner le nombre de plan\n";
                break;
        }
        switch (SelectToitT.Selected.Value) //Calcul de la surface, et du mètre linéaire de matériaux selon le type de toiture sélectionné
        {
            case "Rectangulaire":

                Surface = (Largeur * Longueur) * NbrPlan;
                mlFaitiere = Longueur;
                mlRive = (Largeur * 2) * NbrPlan;

                break;
            case "Carré":

                Surface = Math.Round(((Longueur * Longueur) * Math.Sqrt(3) / 4) * NbrPlan);
                mlFaitiere = Longueur * 4;
                mlRive = mlFaitiere;
                break;
            default:

                if (ErreurLabel.Visible == false)
                {
                    ErreurLabel.Visible = true;
                    DetailErreur.Visible = true;
                }
            }
        }
    }
}
```

```

    }

    valide = false;
    DetailErreur.Text = DetailErreur.Text + "Veuillez sélectionner une forme\n";

    break;
}

switch (SelectTuileT.Selected.Value) //recuperation du nombre de tuile par m² selon la selection
{
    case "ALPHA 10":

        tuilePmC = Double.Parse(actionBDD.Select1Element("Select Nbrmc from materiel where element='tuile' AND
nom='ALPHA 10'"));
        break;

    case "HP13":

        tuilePmC = Double.Parse(actionBDD.Select1Element("Select Nbrmc from materiel where element='tuile' AND
nom='HP13'"));
        break;
    default:

        if (ErreurLabel.Visible == false)
        {
            ErreurLabel.Visible = true;
            DetailErreur.Visible = true;
        }

        valide = false;
        DetailErreur.Text = DetailErreur.Text + "veuillez sélectionner un type de tuile\n";
        break;
}

if (PrixH != "")
{
    PrixUT = Double.Parse(PrixHt.Text.Replace('.', ','));
}
else
{
    valide = false;

    if (ErreurLabel.Visible == false)
    {
        ErreurLabel.Visible = true;
        DetailErreur.Visible = true;
    }

    DetailErreur.Text = DetailErreur.Text + "veuillez indiquer un prix unitaire pour le type de tuile sélectionnée\n";
}

if (valide == true) //Calcul du nombre de tuile, du ml de lisseau, de sous toit et du prix unitaire des tuiles
{
    tuile = Math.Round(Surface * tuilePmC);
    Session["tuile"] = tuile;

    if (SelectToitT.Selected.Value == "Rectangulaire")
    {

        lisseau = Math.Round(Largeur / PureauMoy);
        mlLit = (lisseau * Longueur) * NbrPlan;
    }
    if (SelectToitT.Selected.Value == "Carré")
    {
        lisseau = Math.Round(Longueur / PureauMoy);
        mlLit = ((lisseau * Longueur))/2 * NbrPlan;
    }
}

```

```

    }

    Session["liteau"] = mLLit;
    soustoit = Surface;
    Session["soustoit"] = soustoit;
    if (PrixHt.Text != null)
        PrixUT = Double.Parse(PrixHt.Text.Replace('.', ','));
    PrixHtT = tuile * PrixUT;
    Session["PrixTT"] = PrixHtT;

    form1.Visible = false;
    form2.Visible = true;
    NumDevis(); //génération du numero de devis
}
}

else
{
    if (ErreurLabel.Visible == false)
    {
        ErreurLabel.Visible = true;
        DetailErreur.Visible = true;
    }

    DetailErreur.Text=DetailErreur.Text+"Veuillez entrer les dimensions de la toiture\n";
}

}
}

```