
Développement d'une Application Web pour un expert comptable

Sommaire

1 - Introduction

1.1 - Synopsis	p.37
1.2 - Compétences du référentiel couvertes par le projet	p.38

2 - Présentation du projet

2.1 - Cahier des charges	p.39
2.1.1 - Contexte	p.39
2.1.2 - Présentation du client et du besoin	p.39
2.1.3 - Objectifs	p.39
2.1.4 - Mode d'accès de l'application	p.40
2.2 - Spécifications	p.40
2.2.1 - Fonctionnelles	p.40
2.2.2 - Techniques	p.41
2.3 - Gestion de projet	p.42
2.3.1 - Diagramme de GANTT	p.42
2.3.2 - Outils	p.43
2.3.3 - Cycle en V	p.44

3 - Réalisation

3.1 - Conception du projet	p.45
3.1.1 - Veille Technologique	p.45
3.1.2 - Diagramme de Séquence	p.46
3.2 - Maquettage des application	p.51
3.2.1 - Maquette web	p.51

3.3 -Développement des applications	p.54
3.3.1 - Mise en place de l'environnement	p.54
3.3.2 - Développement de l'application Web	p.61

4 - Conclusion

4 - Conclusion	p.85
----------------	------

Que tous ceux, qui de près ou de loin,
ont aidé ou encouragé mon efforts
trouvent ici le témoignage de ma plus sincère gratitude.

Introduction

1.1 Synopsis

Good morning everyone,

First of all, please excuse me for my accent and the mistakes I may make in English. I don't speak Shakespeare's language as well as Molière's. And on the other hand, I would like all those who, directly or indirectly, have encouraged my effort to find here the testimony of my most sincere gratitude.

My name is Khalil Saghour, I am 25 years old. So I have been coding daily for almost 3 years now.

At the end of 2019, thanks to the Region and Pole Emploi, I had the opportunity to start training as an Application Developer Designer.

I worked alone from initial conception to implementation. Even if there were no written specifications, I worked closely with my internship supervisor by using the Agile method to ensure that the platform, as the project progressed.

If I had to quickly summarize the skills I have acquired, I would tell you this:

I know HTML5, CSS3, BOOTSTRAP4, JAVASCRIPT5, JQUERY3, PHP7, SQL, CSHARP, JAVA, MERISE, UML and GIT. I know how to use SublimeText3, VS Code, Visual Studio, Blend, Eclipse, IntelliJ IDEA, Swing, JavaFx, Android Studio, Wampserver, MySQL, SQLServer, Oracle, Balsamiq, PowerAMC, ArgoUML.

1.2 Compétences du référentiel couvertes par le projet

Activité-type 1: Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité

- Maquetter une application
- Développer des composants d'accès aux données
- Développer la partie front-end d'une interface utilisateur web
- Développer la partie back-end d'une interface utilisateur web

Activité-type 2 : Concevoir et développer la persistance des données en intégrant les recommandations de sécurité

- Mettre en place une base de données

Activité-type 3 : Concevoir et développer une application multicouche répartie en intégrant les recommandation de sécurité

- Concevoir une application.
- Colaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement.
- Construire une application organisée en couches
- Développer une application mobile
- Préparer et exécuter le déploiement d'une application

Présentation du projet

2.1 Cahier des charges

2.1.1 Contexte

J'ai eu l'opportunité d'effectuer mon stage chez lc2l-informatique/
La Batisse,
de avril à aout 2020 sous la direction de mon maître de stage,
M. DEPORT Franck.

Lc2l-informatique et une startup au service des petits producteurs,
des TPE et des PME du secteur agroalimentaire, sis 87
Rue de la Picardière 01300 Virignin France.

2.1.2 Présentation du client et du besoin

La Batisse est un Société Coopérative et Participative régie selon les règles de l'Economie Sociale et Solidaire et membre de l'URSCOP .

Afin d'organiser ses projets et ses tâches, La Batisse souhaite créer une application Web de récupération de FEC (fichier écriture comptable) via une api quickbook et qui permettrait d'améliorer sa productivité.

Dans la continuité d'amélioration de productivité, l'application doit pouvoir permettre aussi de visualiser un aperçu des données du fichier sélectionné.

L'application doit être simple d'utilisation et très intuitive pour les utilisateurs.

2.1.3 Objectifs

L'objectif du projet est d'améliorer la productivité et l'efficacité du travail des collaborateurs internes de l'agence.

L'application sera réservée dans un premier temps aux équipes internes.

2.1.4 Mode d'accès des applications

Application Web

L'application doit être accessible qu'en interne dans un premier temps. Elle doit être déployée sur un windows server pour des tests et ensuite sur serveur local auquel les collaborateurs auront accès. toutefois cette dernière partie sera développée par mon maitre de stage.

2.2 Spécifications

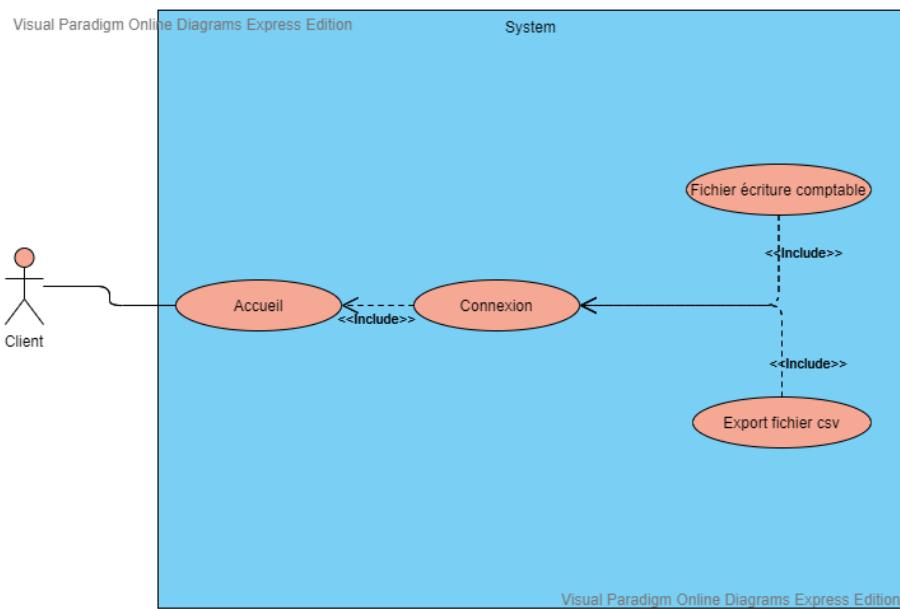
2.2.1 Fonctionnelles

Cette analyse démontre les différentes fonctions de l'application desktop.

a) Fonctionnalité de l'application:

- Connexion Api Quickbook
- Récupération de Fichier écriture comptable via l'api Quickbook
- Avoir un aperçut des données.
- Enregistrez le fichier en format CSV.

Voici un diagramme des cas d'utilisation



2.2.2 Techniques

Application Desktop

- Language: .NET core 3.1.

2.3 Gestion de projet

2.3.1 Diagramme de GANTT

Ce diagramme permet de lister les tâches à réaliser par semaine et est divisé en 2 étapes pour réaliser le projet en son entièreté.

Untitled Gantt Project

20 oct. 2020

<http://>

Chef de projet

Dates du projet

1 juin 2020 - 31 juil. 2020

Avancée

0%

Tâches

2

Ressources

0

Untitled Gantt Project

20 oct. 2020

2

Tâches

Nom	Date de début	Date de fin
Documentation	01/06/20	18/06/20
Développement	19/06/20	30/07/20

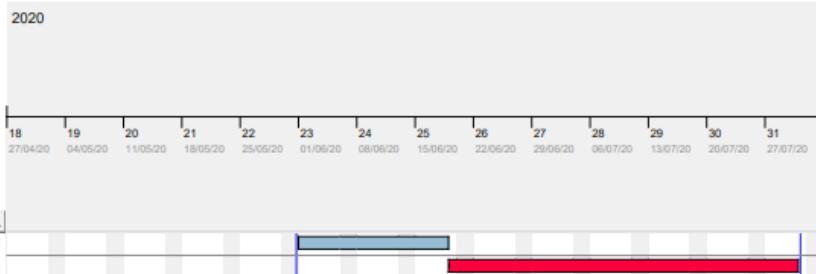
Untitled Gantt Project Diagramme de Gantt

20 oct. 2020

3



Nom	Date d... / Date d...
Documentation	01/06/... 18/06/...
Développement	19/06/... 30/07/...



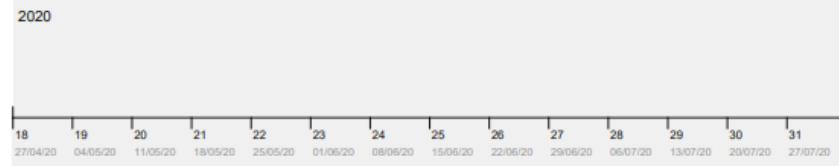
Untitled Gantt Project Diagramme des Ressources

20 oct. 2020

4



Nom	Rôle par ...



2.3.2 Cycle en V

Le développement de l'application desktop suit un processus séquentiel. L'utilisation du cycle en V permet de bien déterminer les étapes importantes et de respecter les objectifs qualités.

Il prend en compte 7 étapes:

1. L'analyse du besoin :

Il s'agit de l'expression du besoin client, c'est-à-dire la description de l'utilisation de l'application tel qu'il l'imagine. L'analyse du besoin doit s'accorder sur ce qui doitêtre fait dans le système.

2. Spécification fonctionnelle :

Elle correspond à tous les cas d'utilisations de l'application.

3. Spécification technique :

Traduction des spécifications fonctionnelles en termes techniques.

Selection des technologies à mettre en oeuvre pour le développement de l'application.

4. Réalisation:

Phase de réalisation pendant laquelle sont développés les composants qui sont ensuite assemblés pour créer l'application finie.

5. Tests unitaire:

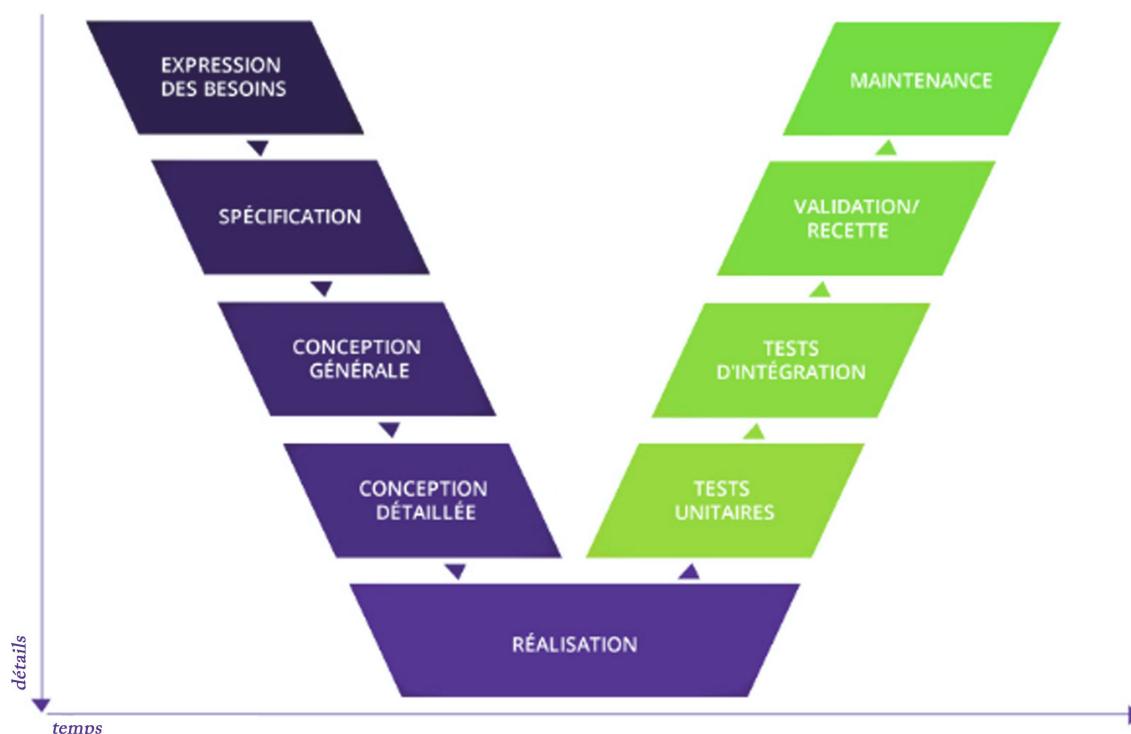
Réalisation de test sur les différentes parties codées : un test par fonctionnalité, pour ce projet il a été décidé qu'il n'y aura pas de test.

6. Tests d'intégration:

Réalisation de test sur l'ensemble de l'application afin d'assurer qu'il suit les indications des spécifications techniques, pour ce projet il a été décidé qu'il n'y aura pas de test.

7. Validation/Recette:

Avant mise en production, le client procède à la recette pour vérifier et valider que son expression de besoin est respectée.



Réalisation

3.1 Conception du projet

3.1.1 Veille Technologique

C'est dans le cadre de mon travail pour Lc2l-informatique, que j'ai pu développer avec le Framework .NET core 3.1, en effet la modularité prend une place prépondérante avec .NET Core car ce doit être un Framework qui fonctionne sur un très grand nombre de plateforme : mobile, desktop et tablette avec UWP, Web et Cloud avec ASP.NET, IoT avec Windows 10 IoT Core.

Selon le runtime, le Framework doit d'adapter afin d'utiliser uniquement les librairies qui l'intéressent, c'est pourquoi la grande majorité de ces dernières sont déployés via des paquets NuGet, optimisant la gestion des versions, la centralisation des paquets et la diffusion du code.

Le projet CoreRT, encore en développement, est un runtime dédié à .NET Core permettant d'accroître ses performances. Tout d'abord, CoreRT permet de faire de la compilation native, évitant ainsi l'étape du code intermédiaire que .NET a connu pendant longtemps. Cela veut dire que les performances de la compilation et de l'exécution d'un projet sont accrues.

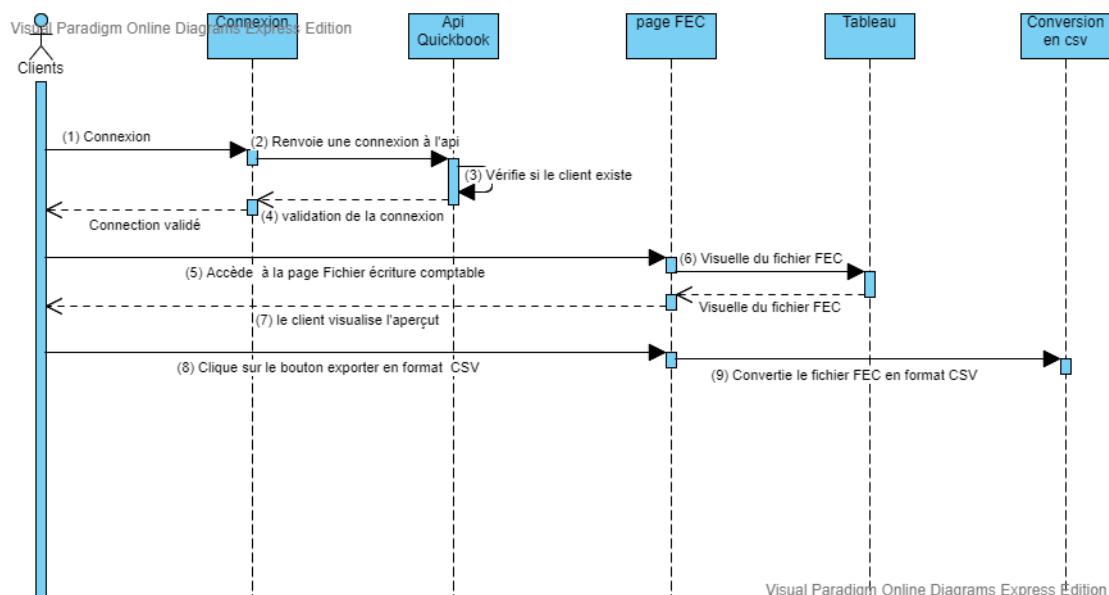
Avantages

deux des nombreux avantages d'utiliser .NET Core

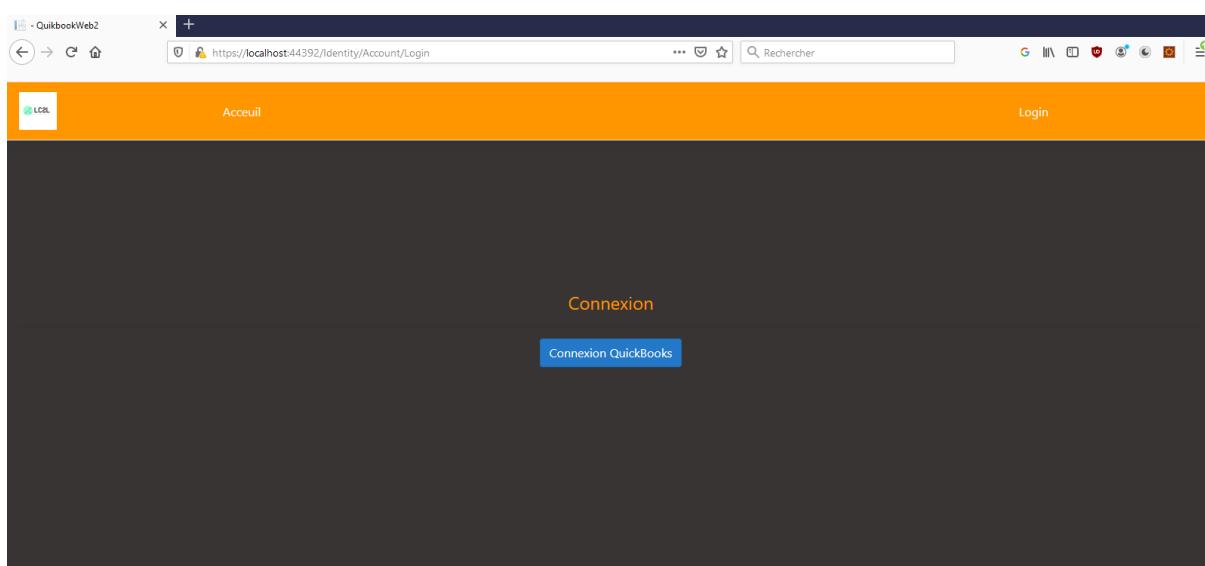
- bénéfice des fonctionnalités de .NET Core.
.NET Core constitue une version évolutive et open source de .NET. Désormais les applications WinForms et WPF sur Windows peuvent tirer profit des dernières fonctionnalités de .NET Core, qui incluent aussi plus de correctifs essentiels pour un support meilleur d'écrans à haute résolution.
- Tailles d'exécutables plus petites
Cette possibilité offerte par .NET Core 3 s'appuie sur une fonctionnalité d'analyse de code dénommée Linker. Elle inclut à une application .NET autonome uniquement les portions de .NET Core qui lui sont nécessaires

3.1.2 Diagramme de Séquence

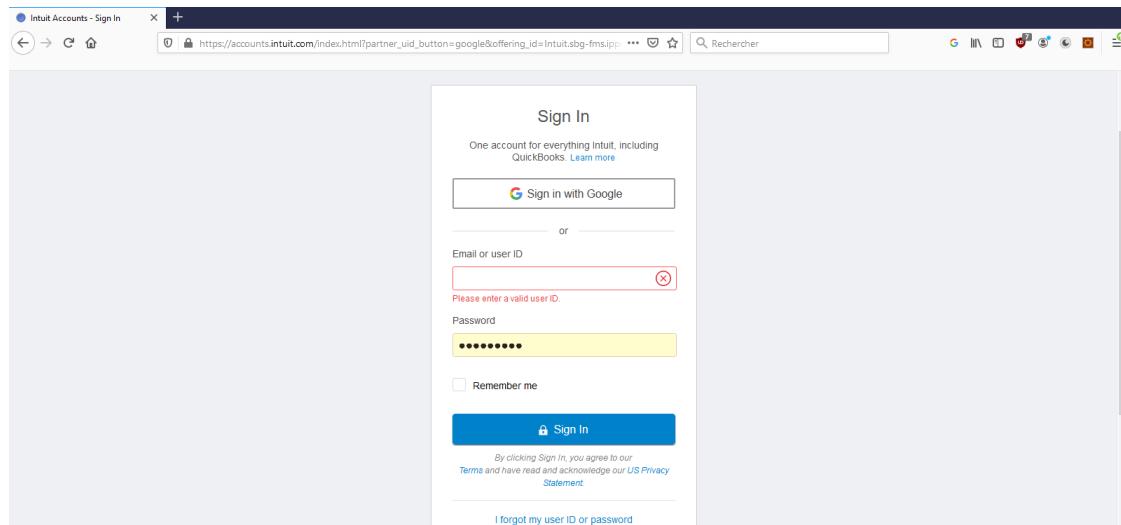
Pour représenter le fonctionnement de l'application, j'ai utilisé l'UML (Langage de modélisation unifié) afin de créer un schémas qui me permet d'apercevoir l'architecture logicielle de la future application.



1. L'utilisateur Clique sur le boutton Connexion.

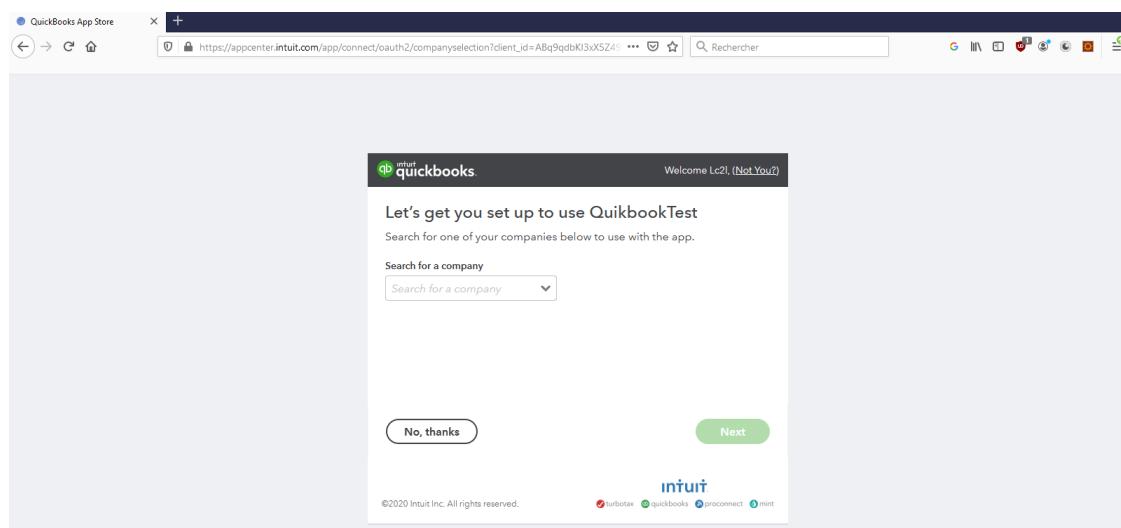


2. On est renvoyé sur la page de connexion de l'api Quickbook

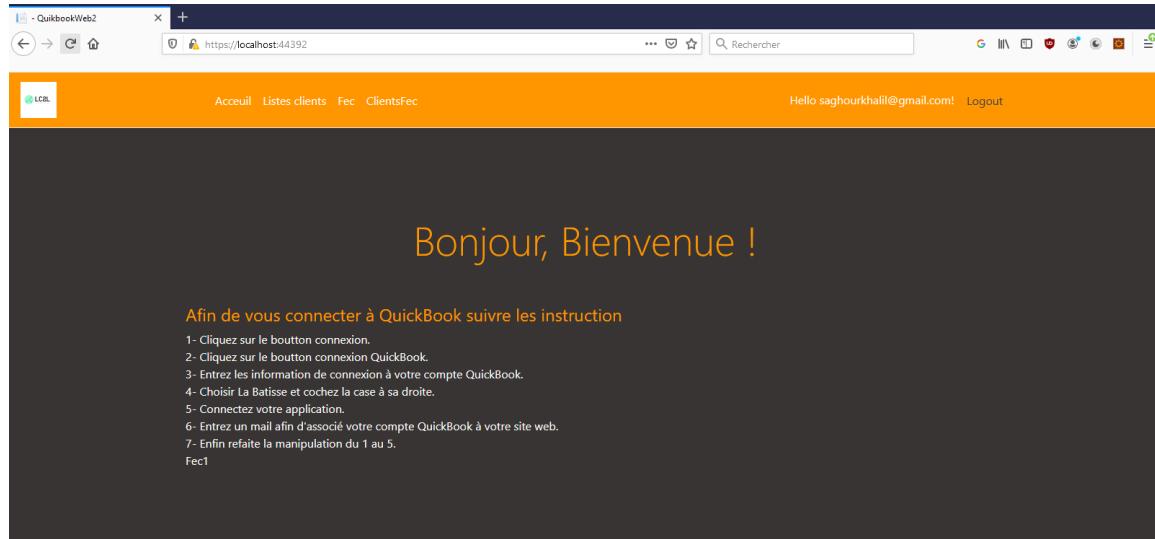


3. Une fois le client vérifier on accéde à une page de choix d'entreprise

Pour une question de sécurité des donnée du client, tout et produit en local avec des entreprise fictif créez au préalable sur le site officiel de Quickbook.

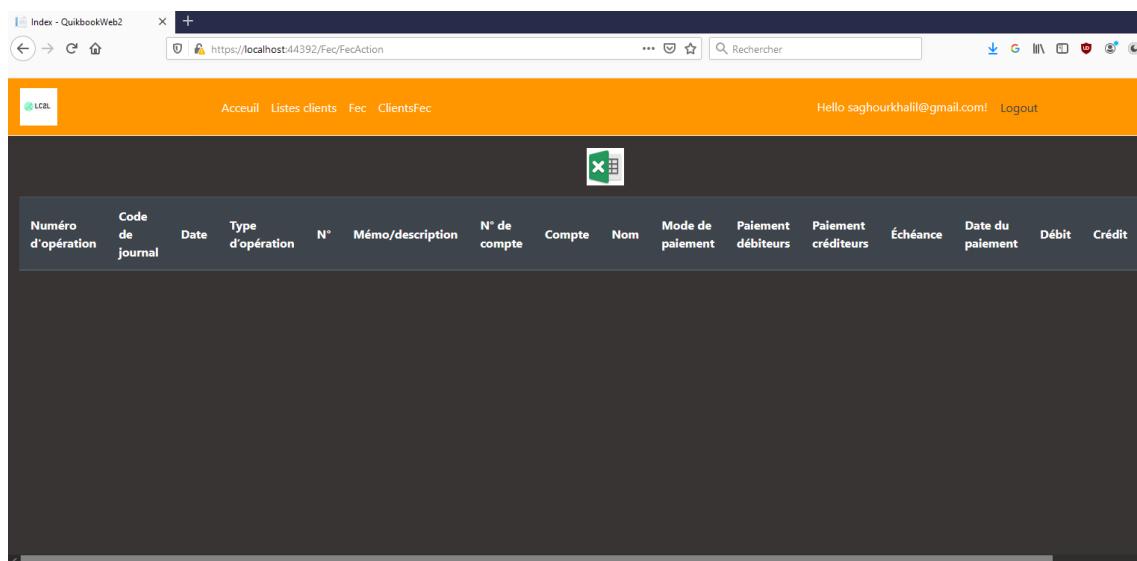


4. Connexion validée



Ici une fois la Connexion validée, on peut voir l'adresse mail du client et de nouvelles pages d'accès au fichier écriture comptable.

4. Le client accéde à la page FEC(Fichier écriture comptable)



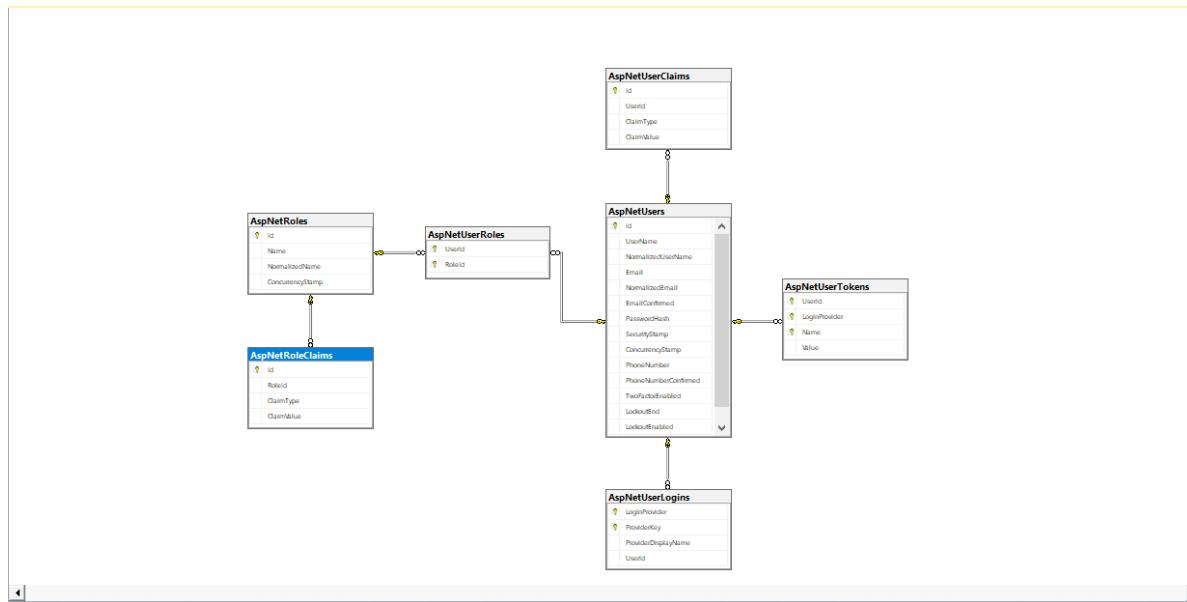
Ici aucune donnée n'est affiché car nous sommes avec une entreprise fictif qui ne possède pas de FEC.

6-7. Le client peut visualisé le FEC(Fichier écriture comptable)

8. Le client décide d'exporter le FEC(Fichier écriture comptable)

9. Le client visualise son FEC au format CSV.

3.1.2 Diagramme de Base de donnée



Lors du design d'une base de données, le Concepteur de bases de données permet de créer, de modifier ou de supprimer des tables, des colonnes, des clés, des index, des relations et des contraintes.

Style de ligne La ligne (et non ses extrémités) indique si le système de gestion de base de données (SGBD) met en œuvre l'intégrité référentielle de la relation quand de nouvelles données sont ajoutées à la table de clé étrangère. Si la ligne est pleine, le SGBD met en œuvre l'intégrité référentielle de la relation lorsque des lignes sont ajoutées ou modifiées dans la table de clé étrangère. Si la ligne est en pointillés, le SGBD ne met pas en œuvre l'intégrité référentielle de la relation lorsque des lignes sont ajoutées ou modifiées dans la table de clé étrangère.

Tables en relation La ligne indique si une relation de clé étrangère existe entre une table et une autre. Dans une relation un-à-plusieurs, la table de clé étrangère est celle qui se trouve près du symbole chiffre huit. Si les deux extrémités de la ligne s'attachent à la même table, il s'agit d'une relation réflexive.

3.2 Maquettage des applications

Le maquettage est l'étape indispensable, car elle force à se poser de nombreuses questions pour affiner la structure de l'application, l'organisation interne des pages et leur maillage.

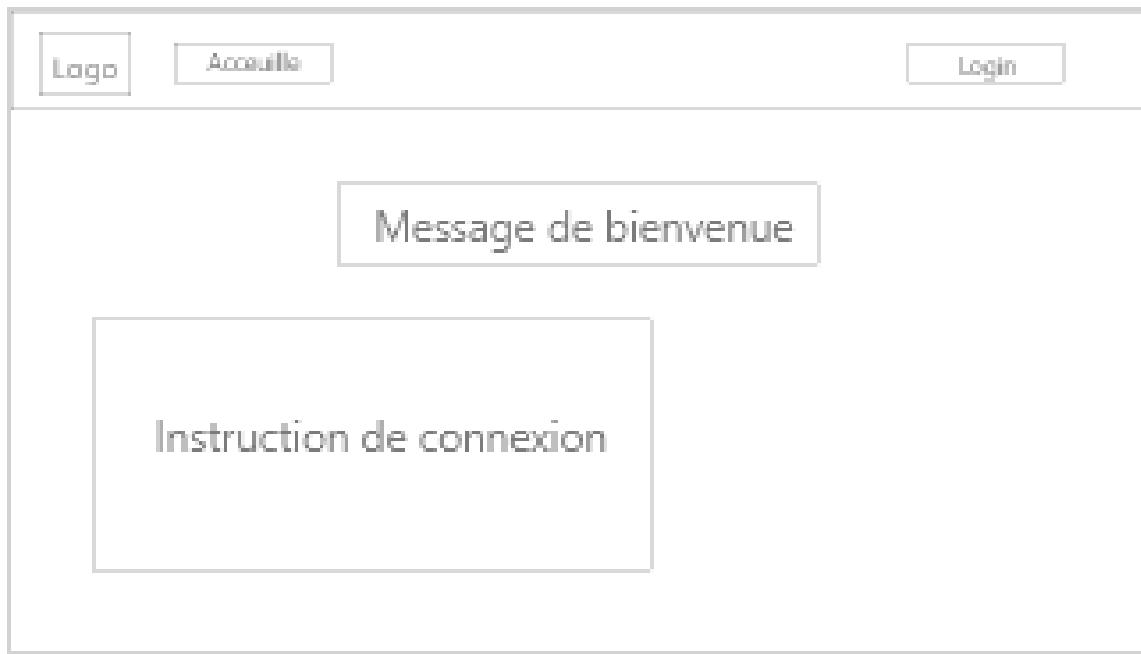
L'objectif de cette étape est de proposer un premier visuel d'interfaces conformes aux attentes et aux besoins.

Il a fallu penser à la structure de l'application Desktop en hiérarchisant les différents contenus, afin de faciliter au maximum l'accès aux informations. En m'appuyant sur le cahier des charges, j'ai tout d'abord réalisé les maquettes des interfaces graphiques grâce au logiciel Adobe XD.

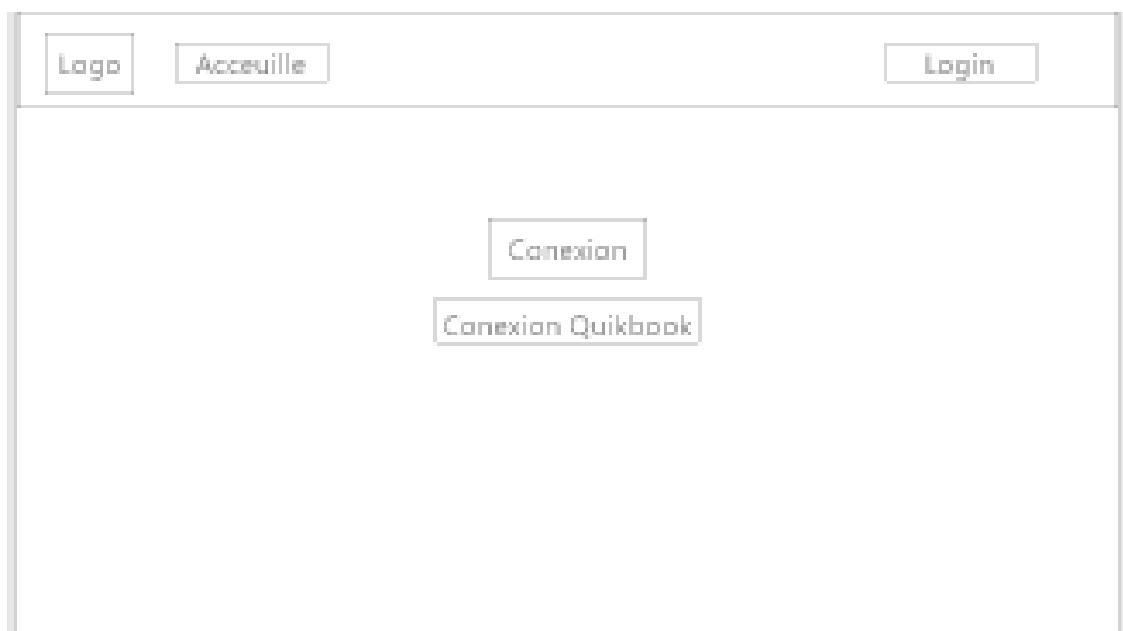
3.2.1 Maquette Application

Wifframe

Acceuil



Connexion

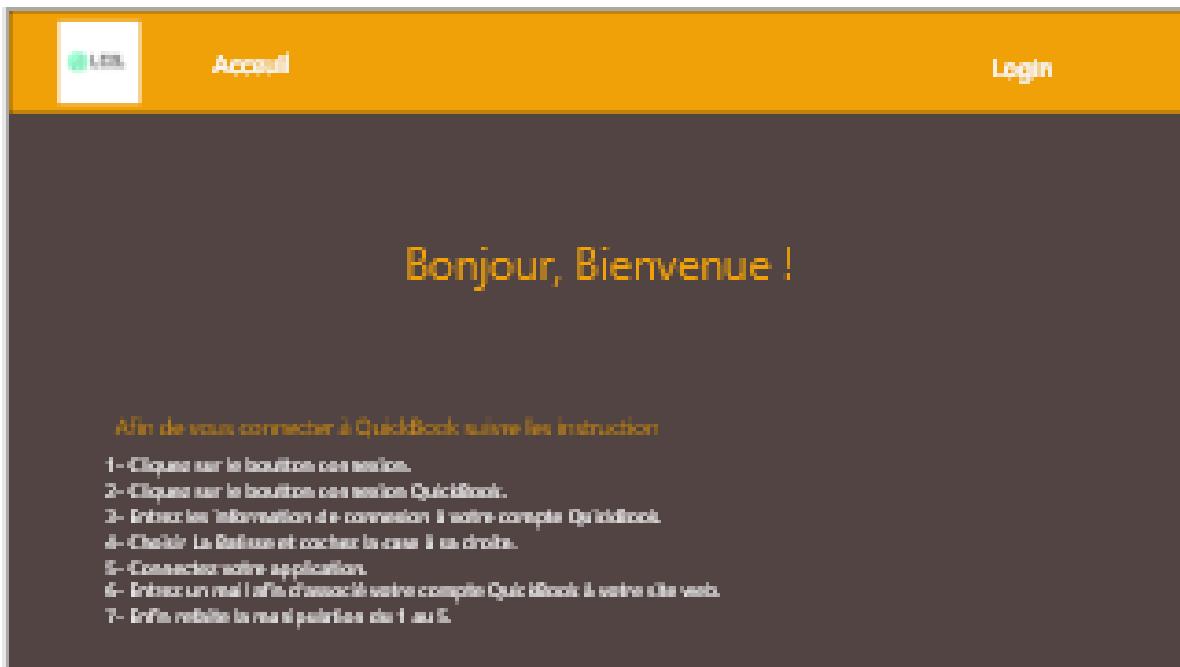


Page Fichier écriture comptable

Logo	Accueil	Fec	Mail connexion	Logout
Export Fec				
<p style="text-align: center;">Détails du FEC (Fichier expert comptable)</p>				

Prototype

Acceuil



ConnexionPage Fichier écriture comptableA screenshot of a web application showing a list of accounting entries. The header is orange with the logo 'LBB' on the left, 'Accueil' and 'Fac' on the right, and the user information 'Pseudo@gmail.com' and 'Logout' on the far right. Below the header is a blue button labeled 'Export Fac'. A table follows, with columns: DATE, OPERATION, NP, ECHEANCE, MONTANT, and SOLDE. The table contains five rows of data:

DATE	OPERATION	NP	ECHEANCE	MONTANT	SOLDE
15/12/2019	Facture	102%-Zara	21/12/2019	29,00	00,00
15/01/2020	Facture	102%-Zara	21/01/2020	49,00	00,00
20/01/2020	Facture	102%-Zara	26/01/2020	59,00	00,00
21/01/2020	Facture	102%-Zara	01/02/2020	79,00	00,00
01/02/2020	Facture	102%-Zara	15/02/2020	86,00	00,00

3.3 Développement de l'application

3.3.1 Mise en place de l'environnement

Machine Virtuelle

En premier temps, j'ai téléchargé un logiciel de Machine virtuelle, afin d'installer un environnement sain et spécifique au développement.

VMware Player

Mon tuteur ma conseiller de télécharger VMware Player, j'ai donc téléchargé VMware Player.

Je suis donc partie sur le site :

- <https://www.vmware.com/fr/products/workstation-player/workstation-player-evaluation.html>

VMware Workstation 15 Player



A ce moment j'ai choisie la version window.

Windows 10

Pour avoir un environnement sain et spécifique au développement, j'ai télécharger au format iso Window 10 professionnelle.

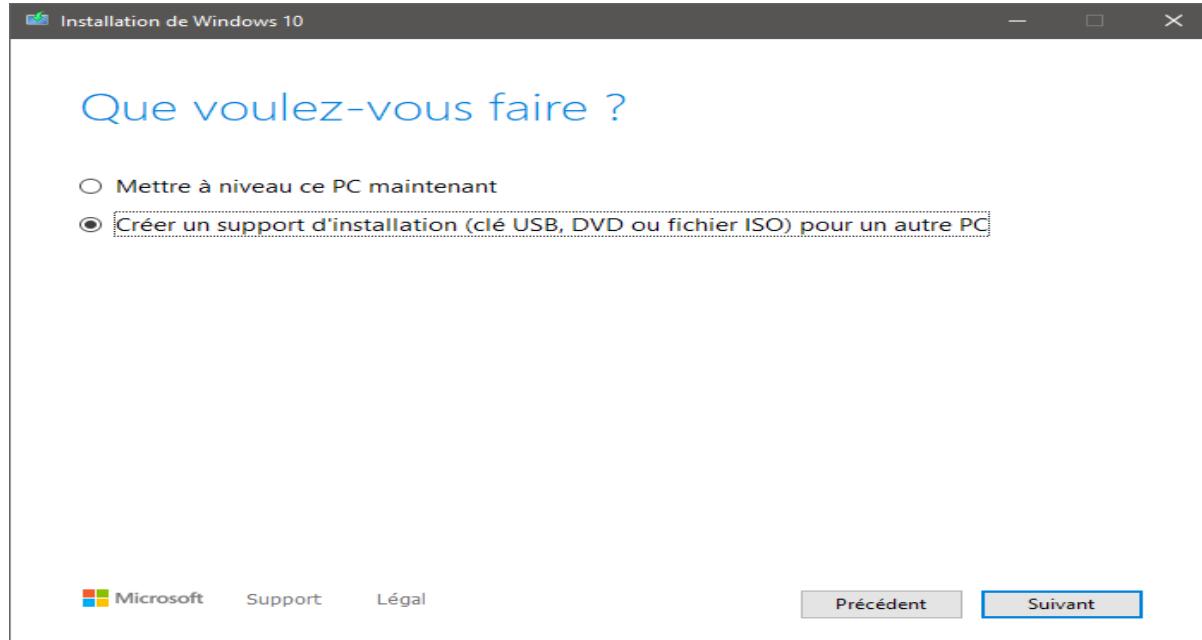
Je suis allez au site officielle de microsoft :

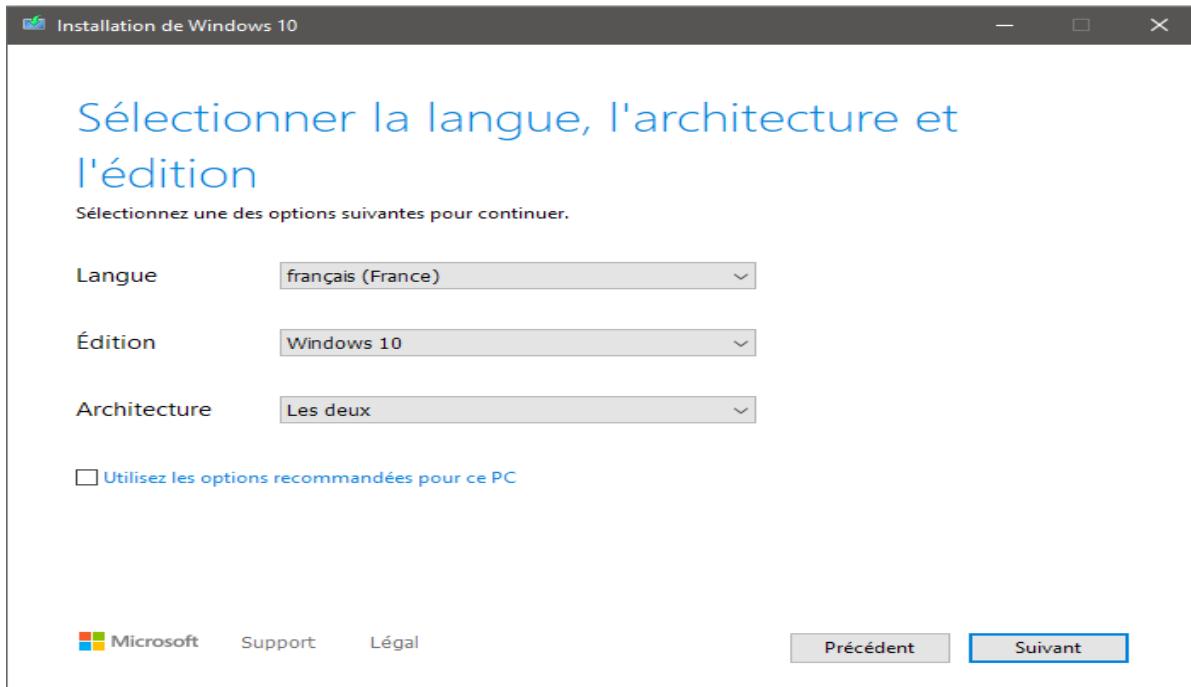
- <https://www.microsoft.com/fr-fr/software-download/windows10>

The screenshot shows the Microsoft Windows 10 download page. At the top, it asks if you want to install Windows 10 on your PC. Below that, it says you need a license and provides a link to download the tool. A large image of a laptop running Windows 10 is shown. At the bottom, there are three expandable options:

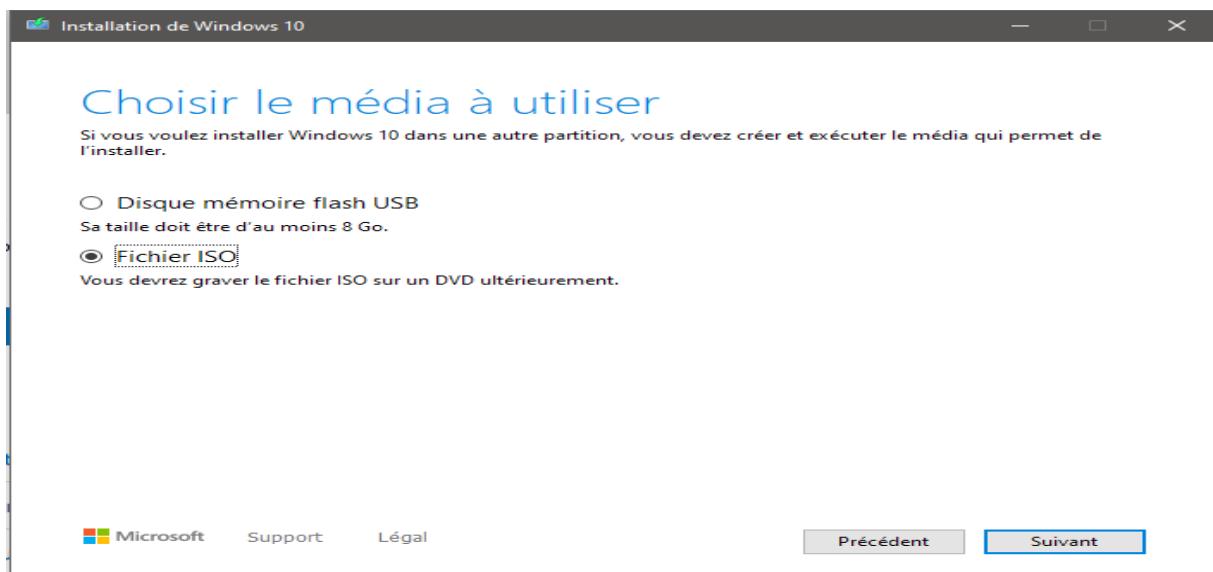
- Utilisation de l'outil pour mettre à niveau ce PC vers Windows 10 (cliquez pour afficher moins ou plus d'informations)
- Utilisation de l'outil pour créer vos supports d'installation (clé USB, DVD ou fichier ISO) afin d'installer Windows 10 sur un autre PC
- Options de téléchargement supplémentaires

Je suis la procédure d'installation.





Ici je conseille pour se qui ne savent pas quelle et l'architecture de son pc, de sélectionner les deux plutot que x64 ou x86.



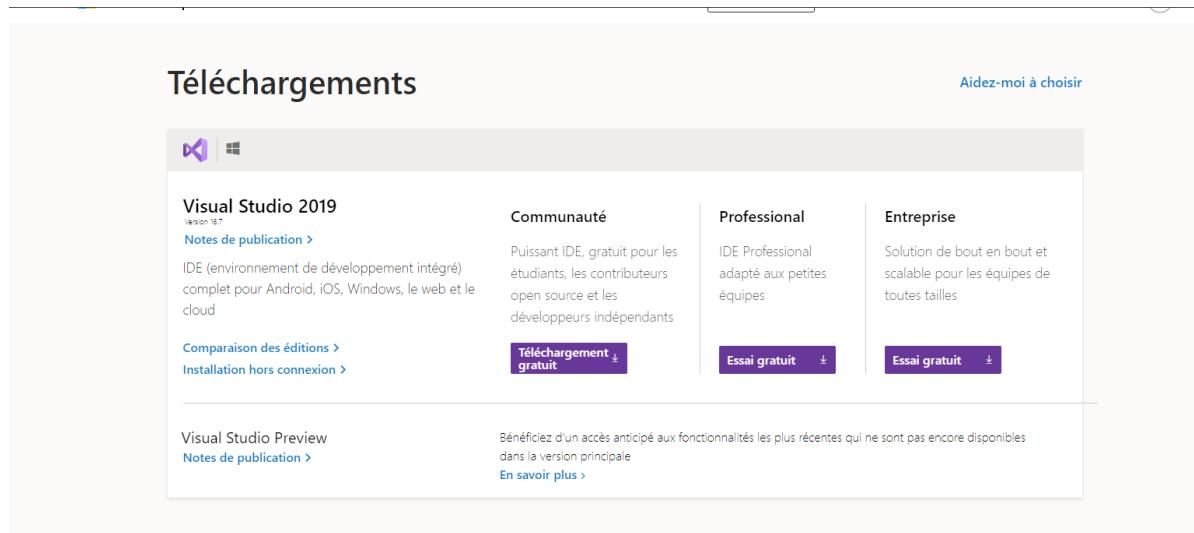
Pour finir je sélectionne le format iso et je lance l'installation.

Visual studio 2019

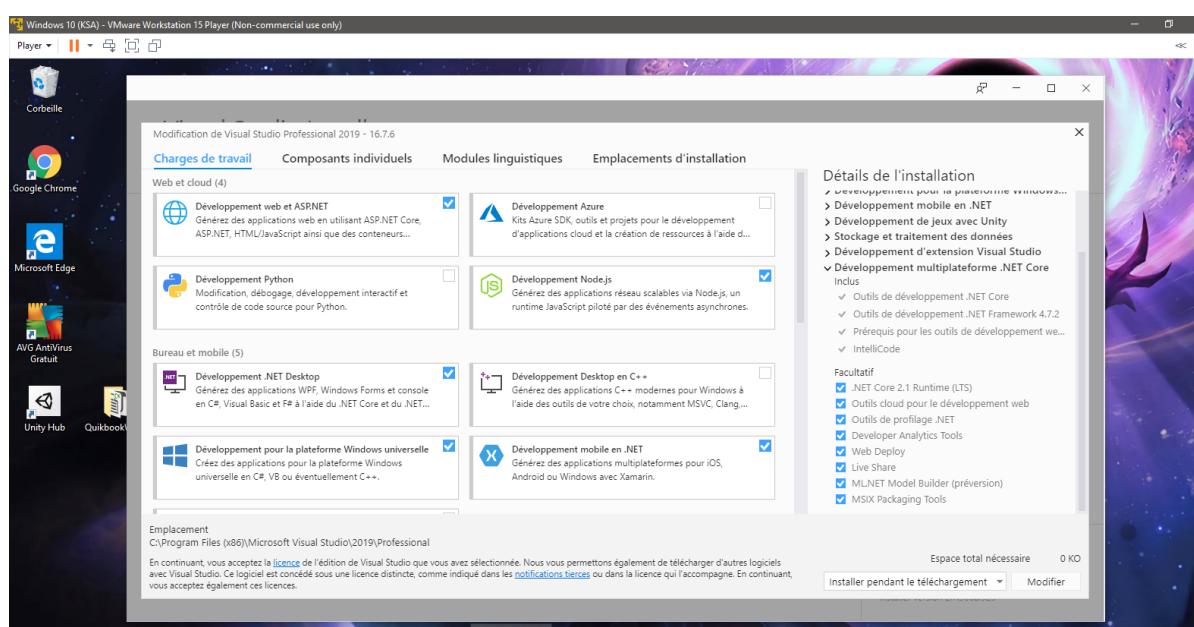
Pour avoir un IDE des plus performant j'ai télécharger visual studio 2019 ici :

- <https://visualstudio.microsoft.com/fr/downloads/>

Ici j'ai choisie la version essaie professionnal et j'ai suivie les étape d'installation.



Ici je vais choisir les language les mieux adapter a mon projet et futur projets.



Windows server

Afin de pouvoir héberger mon site sur un server j'ai donc décider de télécharger Window server 2019 ici :

- <https://www.microsoft.com/fr-fr/evalcenter/evaluate-windows-server-2019>

Ici j'ai choisie la version essaie Windows server 2019 et j'ai suivie les étape d'installation.



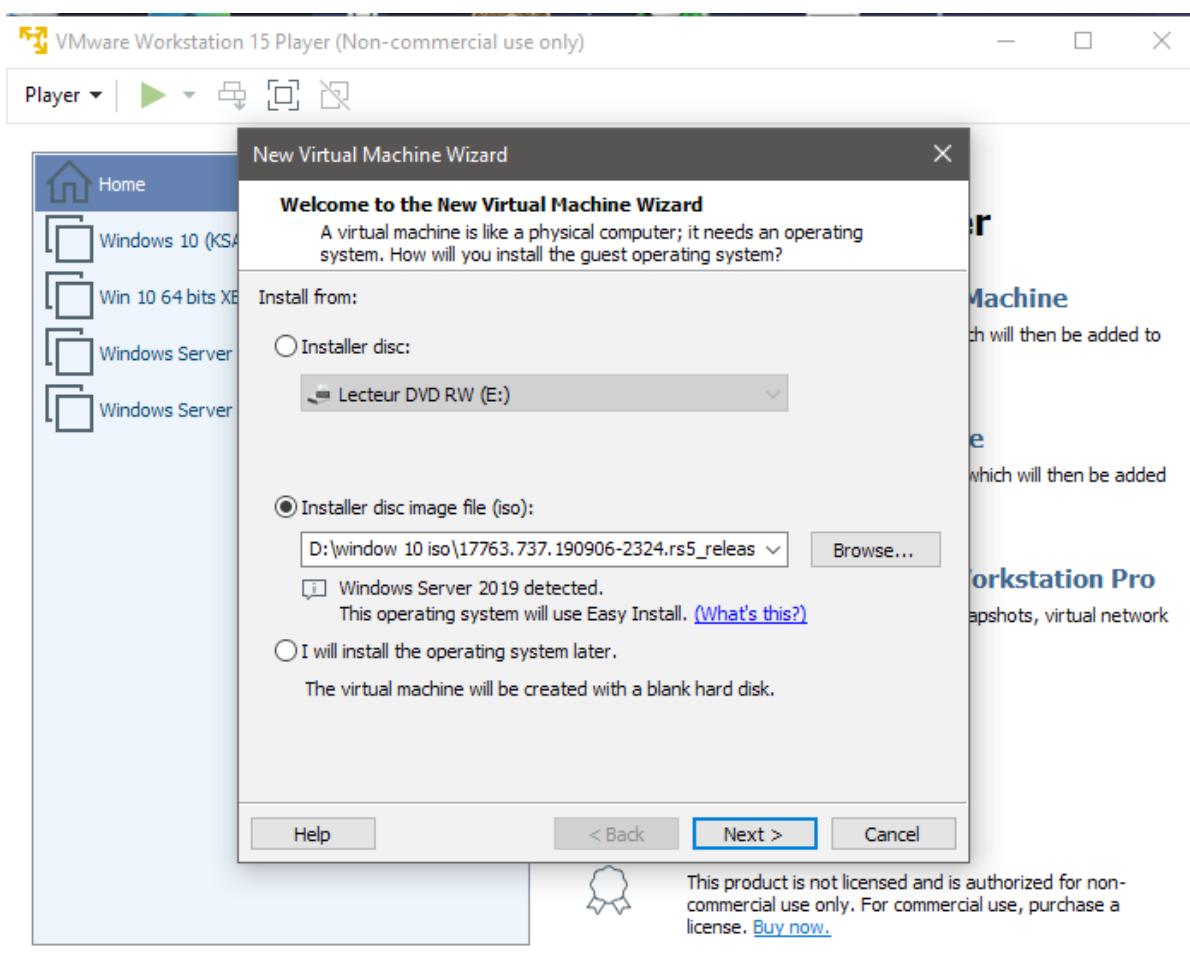
Windows Server Évaluations

- ⊕ Windows Server 2019
Évaluations | [180 jours](#)
- ⊕ Windows Server 2019 Essentials
Évaluations | [180 jours](#)
- ⊕ Microsoft Hyper-V Server 2019
Évaluations | [illimité](#)
- ⊕ Windows Admin Center
Évaluations
- ⊕ Windows Server 2016
Évaluations | [180 jours](#)
- ⊕ Microsoft Hyper-V Server 2016
Évaluations

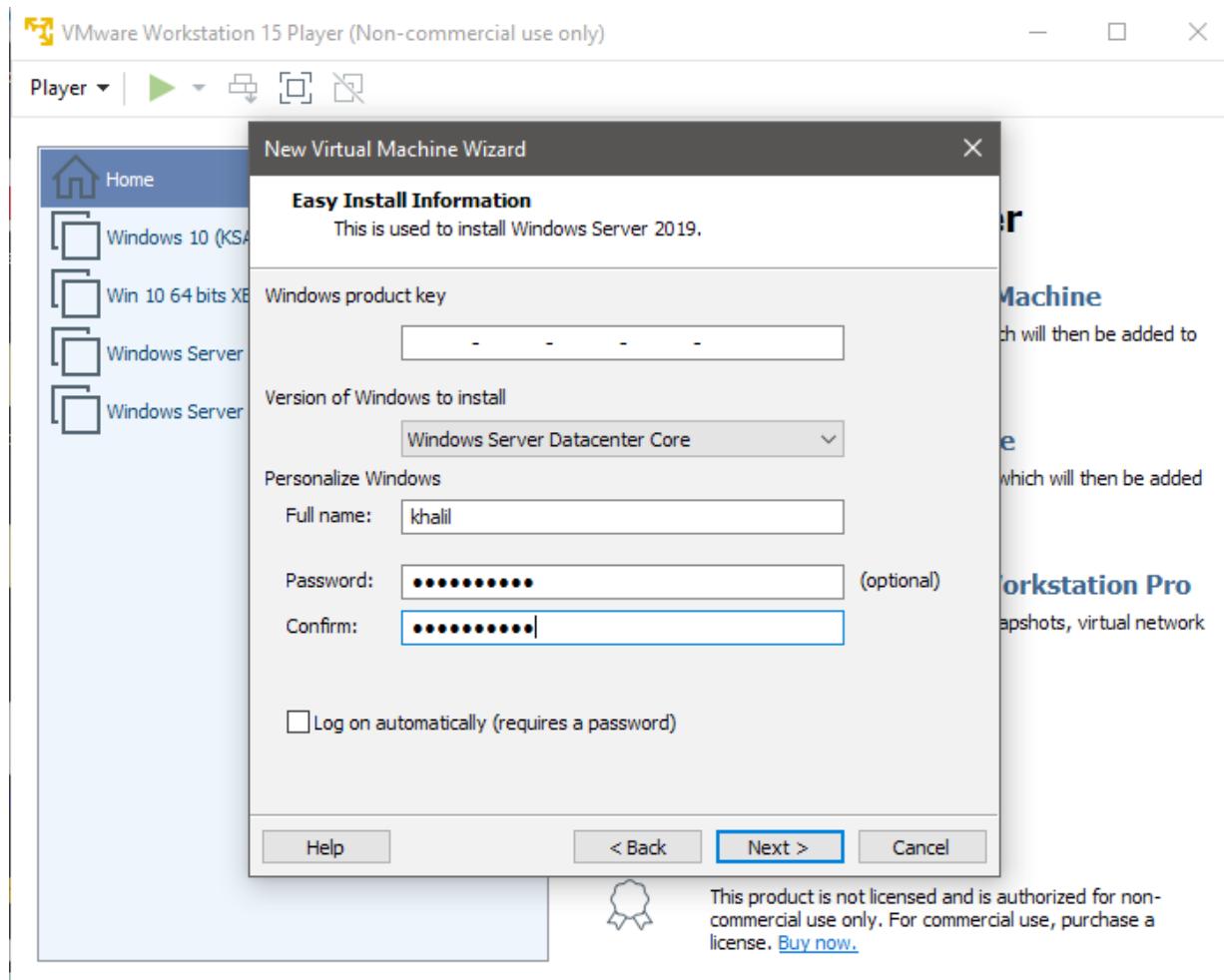
Je choisis donc le format iso afin de pouvoir l'intaller via ma machine virtuelle.

Une fois le téléchargement fini, j'ouvre ma VM et crée une nouvelle machine virtuelle.

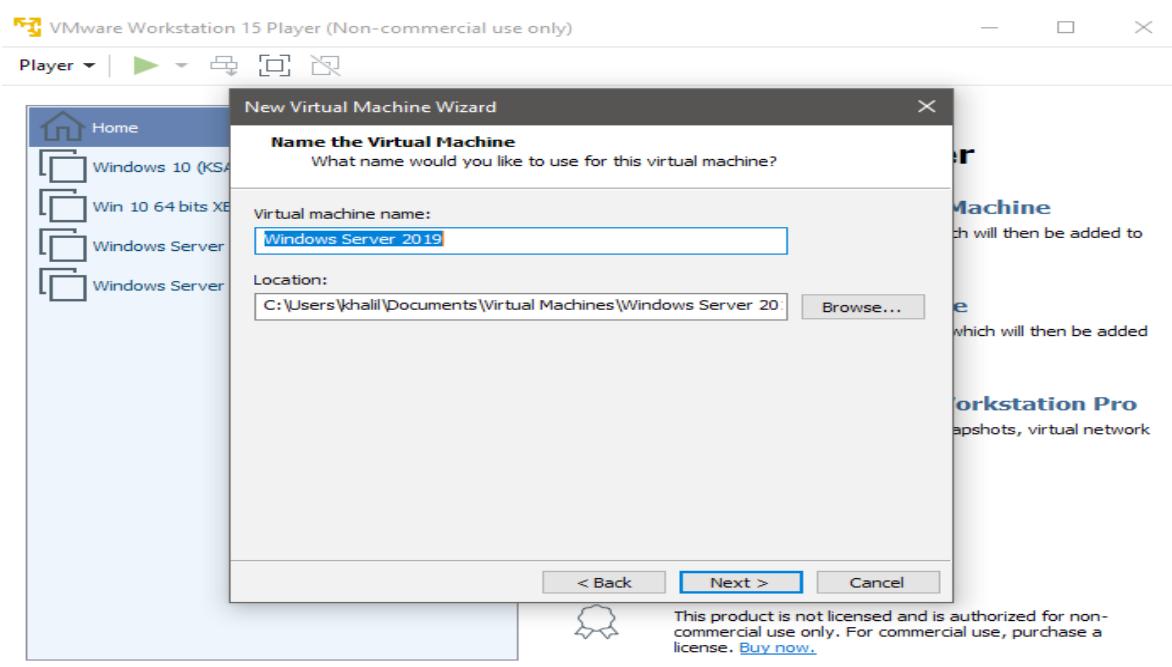
1. Indiquez l'emplacement de l'iso télécharger.



2. Indiquez un mots de passe facile.



3. Lancer le téléchargement.



3.3.2 Développement de l'application

Configuration et connexion à l'api quickbook

Ajouter à l'aide de l'instruction

```

16  using Microsoft.IdentityModel.Protocols.OpenIdConnect;
17  using Microsoft.AspNetCore.Authentication.OAuth;
18  using Microsoft.AspNetCore.Authentication.OpenIdConnect;
19  using Microsoft.AspNetCore.Authentication;
20  using Microsoft.IdentityModel.Protocols;
21

```

Ajouter un middleware OpenId Connect

Ajoutez le code ci-dessous à la méthode ConfigureServices pour configurer le fournisseur OpenId Connect afin que nous puissions utiliser l'authentification unique (SSO) sur le serveur Intuit OAuth 2.0. Il doit être placé au-dessus de la ligne services.AddMvc () .

```

45  services.AddAuthentication(sharedOptions => { })
46    .AddCookie()
47    .AddOpenIdConnect("QuickBooks", "Connexion QuickBooks", openIdConnectOptions =>
48    {
49      openIdConnectOptions.Authority = "QtretrretretrickBozffffoks";
50      openIdConnectOptions.UseTokenLifetime = true;
51      //openIdConnectOptions.ClientId = ""; //client id & client secret need to be set w/ your app keys 91.169.210.13
52      openIdConnectOptions.ClientId = ""; //client id & client secret need to be set w/ your app keys
53      //openIdConnectOptions.ClientSecret = "";
54      openIdConnectOptions.ClientSecret = "";
55      openIdConnectOptions.ResponseType = OpenIdConnectResponseType.Code;
56      openIdConnectOptions.MetadataAddress = "https://developer.api.intuit.com/.well-known/openid_sandbox_configuration/"; //développement path
57      //openIdConnectOptions.MetadataAddress = "https://developer.api.intuit.com/.well-known/openid_configuration"; //production path
58      openIdConnectOptions.ProtocolValidator.RequireNonce = false;
59      openIdConnectOptions.SaveTokens = true;
60      openIdConnectOptions.UseTokenLifetime = true;
61      openIdConnectOptions.Scope.Add("openid");
62      openIdConnectOptions.Scope.Add("phone");
63      openIdConnectOptions.Scope.Add("email");
64      openIdConnectOptions.Scope.Add("address");
65      openIdConnectOptions.Scope.Add("com.intuit.quickbooks.accounting");
66      openIdConnectOptions.GetClaimsFromUserInfoEndpoint = true;
67    });
68    services.AddRazorPages();

```

Ajouter un middleware OpenId Connect

Je définie `openIdConnectOptions.ClientId` et `openIdConnectOptions.ClientSecret` avec mon application ClientId et ClientSecret à partir du portail des développeurs Intuit.

The image shows two side-by-side screenshots. On the left, a code editor displays a portion of a C# file with the following code:

```

services.AddAuthentication(sharedOptions => { })
    .AddCookie()
    .AddOpenIdConnect("QuickBooks", "QuickBooks", openIdConnectOptions =>
{
    openIdConnectOptions.Authority = "QuickBooks";
    openIdConnectOptions.UseTokenLifetime = true;
    openIdConnectOptions.ClientId = "Q06cb1NwSHXscqjxqo2WkL..."; //client ID
    openIdConnectOptions.ClientSecret = "megMDCbbrMAlx4Kp41..."; //client secret
    openIdConnectOptions.ResponseType = OpenIdConnectResponseType.Code;
    openIdConnectOptions.MetadataAddress = "https://developer.api.intuit.com/v2/applications/open";
    openIdConnectOptions.ProtocolValidator.RequireNonce = false;
    openIdConnectOptions.SaveTokens = true;
    openIdConnectOptions.GetClaimsFromUserInfoEndpoint = true;
    openIdConnectOptions.Scope.Add("openid");
    openIdConnectOptions.Scope.Add("phone");
    openIdConnectOptions.Scope.Add("email");
    openIdConnectOptions.Scope.Add("address");
    openIdConnectOptions.Scope.Add("com.intuit.quickbooks.accounting");
    openIdConnectOptions.Scope.Add("com.intuit.quickbooks.payment");
});
services.AddMvc();
  
```

Two red arrows point from the highlighted `ClientId` and `ClientSecret` lines in the code to their respective fields in the `Keys` section of the Intuit developer portal on the right. The portal shows the following details:

- Development Keys** section:
 - Client ID:** Q06cb1NwSHXscqjxqo2WkL... (with a **Copy** button)
 - Client Secret:** megMDCbbrMAlx4Kp41... (with a **Copy** button)
 - Rotate Secret:** (link)
- Development Sandbox** section:
 - Sandbox Company_US_1:** (dropdown menu)
 - Accounting Sandbox Base URL:** sandbox-quickbooks.api.intuit.com (with a **Copy** button)
 - View Sandbox:** (link)
 - Manage sandboxes:** (link)

Ajouter un middleware OpenId Connect

The image shows a screenshot of the Intuit developer portal interface. On the left, there is a sidebar with application details and a navigation menu. The main content area is titled **Keys** and contains the following sections:

- Development Keys**: Shows Client ID and Client Secret fields with **Copy** buttons.
- Development Sandbox**: Shows a dropdown for **Sandbox Company_US_1**, a **View Sandbox** link, and a **Manage sandboxes** link.
- Redirect URIs**: A table with columns for **Redirect URI 1**, **Redirect URI 2**, etc. The first row contains:
 - Redirect URI 1:** https://localhost:44344/signin-oidc
 - Redirect URI 2:** https://developer.intuit.com/v2/OAuth2Playground/RedirectURI
 - Redirect URI 3:** http://localhost:59785/Default.aspx
 - Redirect URI 4:** https://www.getpostman.com/oauth2/callback
 - Redirect URI 5:** https://localhost:44359/login-oids

Two red circles are overlaid on the screenshot: circle 1 points to the **Client Secret** field in the Keys section, and circle 2 points to the **Save** button at the bottom of the Redirect URIs table.

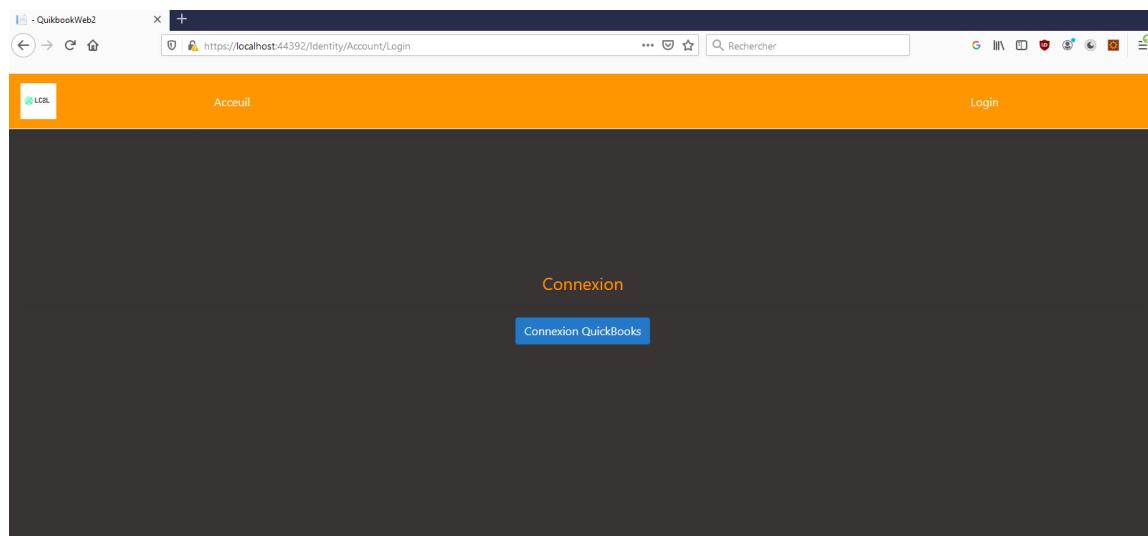
J'exécute l'application Web et je copie l'URL de mon applications Web (cela peut être différent de ce que vous voyez dans la capture d'écran).

(2) J'ajoute l'URL de mon applications Web plus / signin-oidc à la liste des URI de redirection sur le portail des développeurs

REMARQUE: Le /signin-oidc est un chemin magique utilisé par le middleware OpenId Connect pour le rappel d'Intuit.

Tester l'authentification unique

Je peux maintenant cliquer sur le lien Connexion de mon application Web et je vois QuickBooks un autre service pour me connecter.



Mettre à jour la connexion externe pour enregistrer les jetons

```
61     public async Task<IActionResult> OnGetCallbackAsync(string returnUrl = null, string remoteError = null)
62     {
63         if (remoteError != null)
64         {
65             ErrorMessage = $"Error from external provider: {remoteError}";
66             return RedirectToPage("./Login");
67         }
68         var info = await _signInManager.GetExternalLoginInfoAsync();
69         if (info == null)
70         {
71             return RedirectToPage("./Login");
72         }
73
74         // Sign in the user with this external login provider if the user already has a login.
75         var result = await _signInManager.ExternalLoginSignInAsync(info.LoginProvider, info.ProviderKey, isPersistent: false, bypassTwoStep: true);
76         if (result.Succeeded)
77         {
78             _logger.LogInformation("{Name} logged in with {LoginProvider} provider.", info.Principal.Identity.Name, info.LoginProvider);
79             await _signInManager.UpdateExternalAuthenticationTokensAsync(info);
80             return LocalRedirect(Url.GetLocalUrl(returnUrl));
81         }
82         if (result.IsLockedOut)
83         {
84             return RedirectToPage("./Lockout");
85         }
86         else
87         {
88             // ...
89         }
90     }
91 }
```

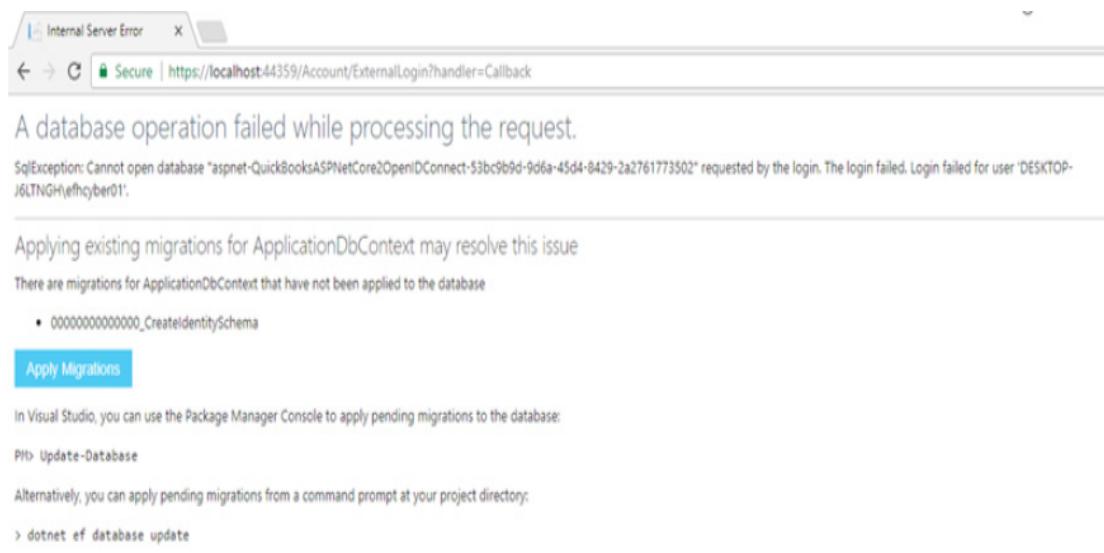
Configuration et connexion à la base de donnée

Une fois les jetons mis à jour je me connecte pour la première fois à Lapi Quickbook.

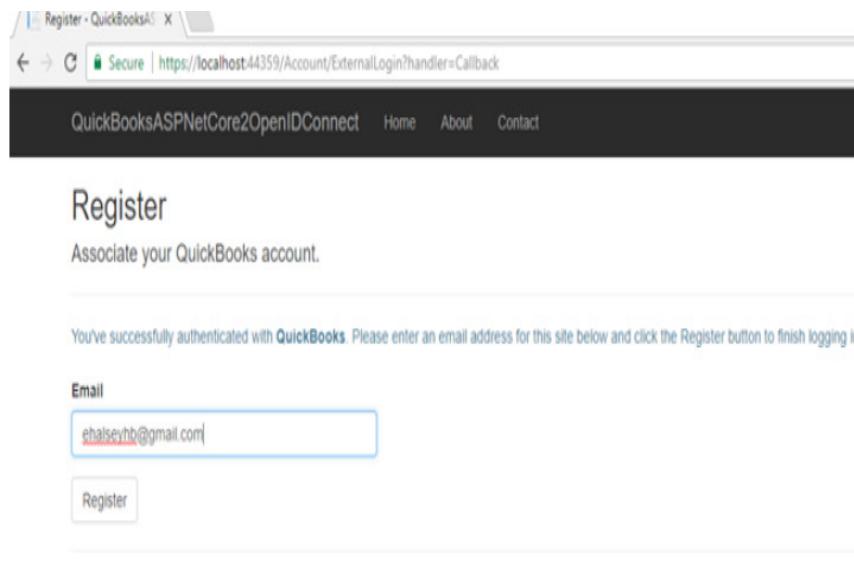
The image contains two screenshots of web browser windows. The top screenshot shows the 'Intuit Accounts - Sign In' page at https://accounts.intuit.com/index.html?partner_uid_button=google&offering_id=Intuit.sbg-fms.ip. It features a 'Sign in with Google' button and a form for 'Email or user ID' and 'Password'. The 'Email or user ID' field is highlighted in red with the error message 'Please enter a valid user ID.'. The bottom screenshot shows the 'QuickBooks App Store' at https://appcenter.intuit.com/app/connect/oauth2/companyselection?client_id=ABq9qdbK13xS24. It displays a search bar for 'Search for a company' and two buttons: 'No, thanks' and 'Next'. The Intuit logo is visible at the bottom.

Je sélectionne une entreprise et je met next afin d'autorisé l'application

REMARQUE La première fois que j'exécutez mon application Web, je dois appliquer les migrations. Cela crée les tables pour l'identité. Je clique sur Apply Migrations et j'actualise la page.



J'enregistre mon nouveau compte en indiquant une adresse mail que je verrais apparaitre après sur mon application.



Ensuite j'indique la connexion par default dans mon appsetting afin de pouvoir externalisé la connexion, pour plus de sécurité sur mon server svn je précise bien que se fichier n'est pas à push.

```
1  {
2    "ConnectionStrings": {
3      //DefaultConnection": "Server=SRVIPARLA-KSA;Database=aspnet-Quikbook_KSA;User Id=sa;Password=Y48mp*cGt;",
4      "DefaultConnection": "Server=(localdb)\\mssqllocaldb;Database=QBO-LBA;Trusted_Connection=True;"
5    },
6    "Logging": {
7      "LogLevel": {
8        "Default": "Information",
9        "Microsoft": "Warning",
10       "Microsoft.Hosting.Lifetime": "Information"
11      }
12    },
13    "AllowedHosts": "*"
14  }
```

Je configure dans mon startup.cs la connexion à récupére dans mon appsetting.cs.

```
35  public void ConfigureServices(IServiceCollection services)
36  {
37
38    services.AddDbContext<ApplicationContext>(options =>
39      options.UseSqlServer(
40        Configuration.GetConnectionString("DefaultConnection")));
41  }
```

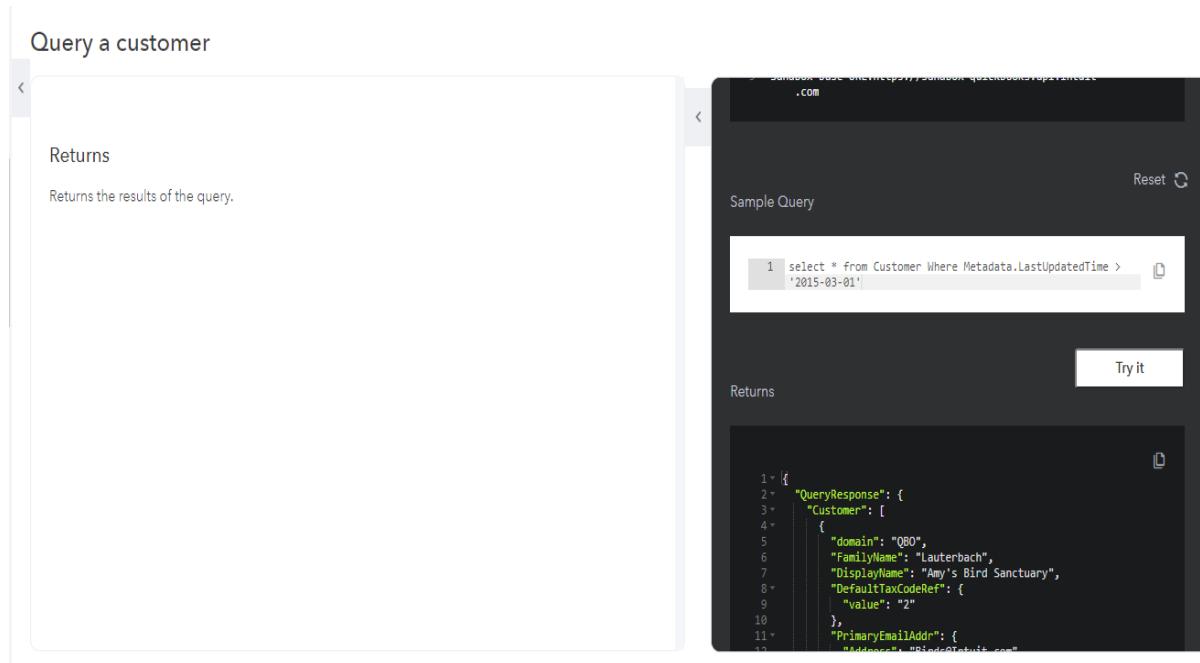
Ajouter un code pour afficher les clients

Maintenant que j'ai enregistré notre jeton d'accès, je peux les utiliser pour passer un appel API QBO

Ajouter un fichier de classe pour les classes POCO liées au client

On ma conseiller d'utiliser json2csharp pour générer automatiquement des classes C # à partir d'exemples Json. Je peux bien sûr construire ces classes manuellement ou extraire les données souhaitées de la chaîne Json si vous préférez. Mais pour une question de productivité j'ai utilisé un générateur automatique.

Je peux obtenir un exemple de Json à l'aide de QBO API Exploré . L' application Web va interroger les clients à l'aide du point de terminaison de l'API / query. Exécutez une requête à l'aide de l'explorateur d'API afin que nous puissions ensuite copier et coller les résultats de la requête dans json2csharp .



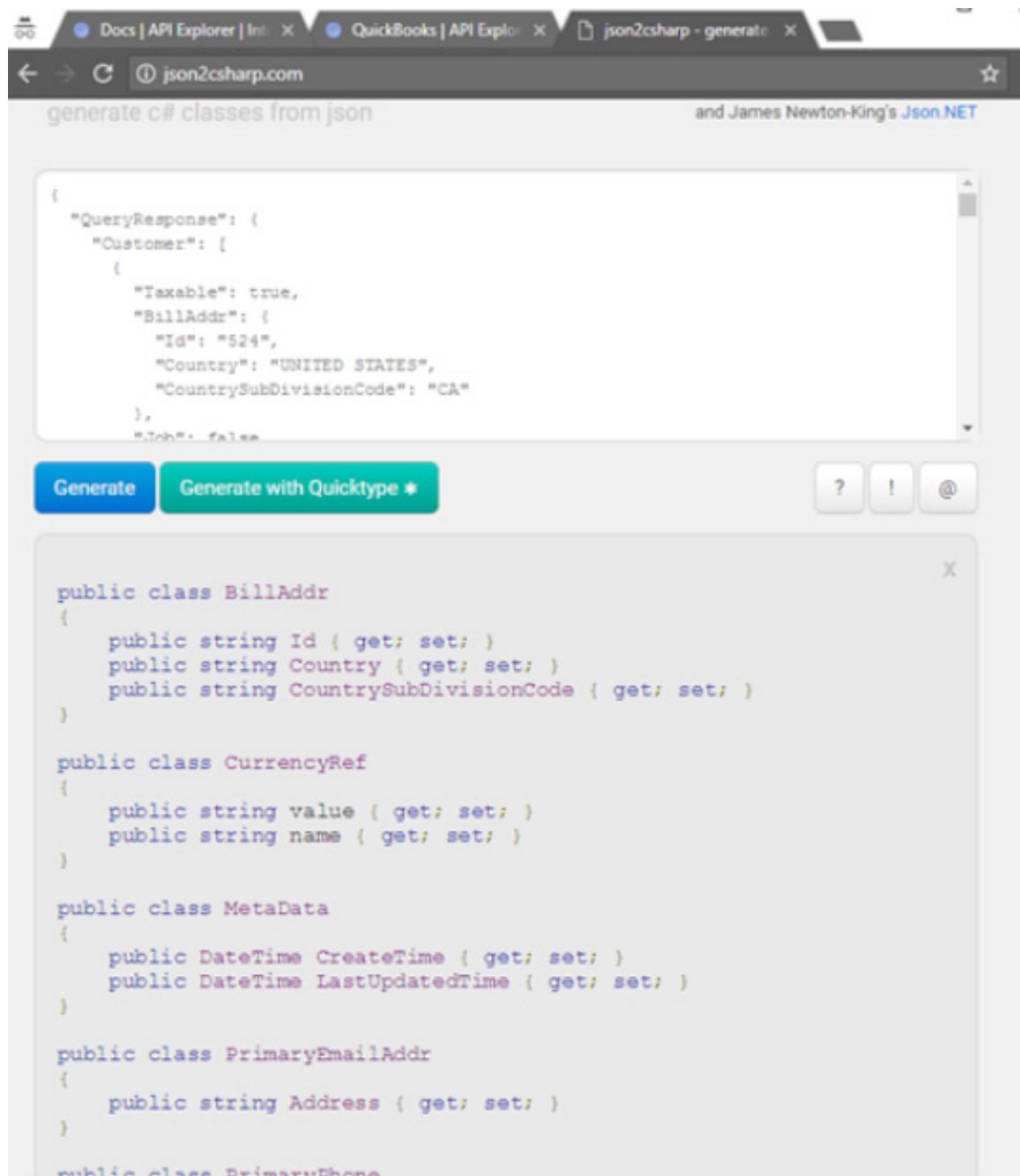
The screenshot shows the QBO API Explorer interface. On the left, there's a sidebar with a 'Query a customer' section. It has a 'Returns' section describing the results of the query. On the right, there's a main panel titled 'Sample Query'. It contains a code editor with the following JSON query:

```
1 select * from Customer Where Metadata.LastUpdatedTime > '2015-03-01'
```

Below the code editor is a 'Try it' button. Further down, another 'Returns' section shows the resulting JSON response:

```
1 {  
2   "QueryResponse": {  
3     "Customer": [  
4       {  
5         "domain": "QBO",  
6         "FamilyName": "Lauterbach",  
7         "DisplayName": "Amy's Bird Sanctuary",  
8         "DefaultTaxCodeRef": {  
9           "value": "2"  
10        },  
11        "PrimaryEmailAddr": {  
12          "Address": "Birds@Birds.com"  
13        }  
14      }  
15    ]  
16  }  
17 }
```

Collez les résultats dans json2csharp et cliquez sur Générer



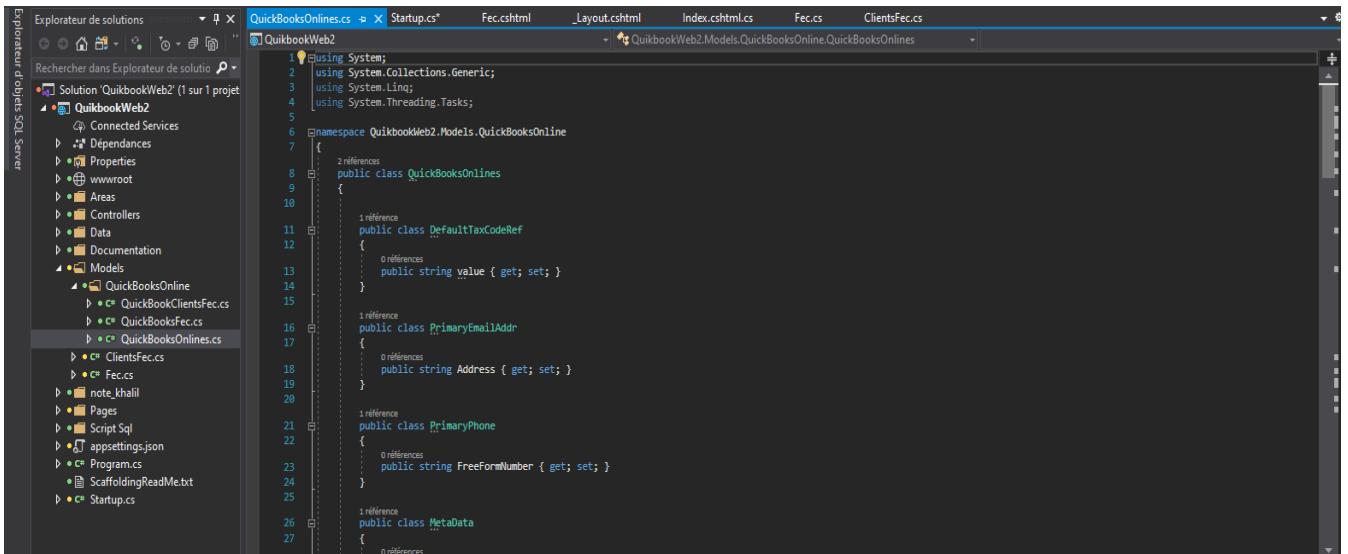
The screenshot shows a browser window with three tabs: "Docs | API Explorer | Int", "QuickBooks | API Explorer", and "json2csharp - generate". The main content area is the json2csharp website, which has a URL of "json2csharp.com" in the address bar. The page title is "generate c# classes from json" and includes a note "and James Newton-King's Json.NET". A JSON snippet is pasted into the input field:

```
{  
    "QueryResponse": {  
        "Customer": [  
            {  
                "Taxable": true,  
                "BillAddr": {  
                    "Id": "524",  
                    "Country": "UNITED STATES",  
                    "CountrySubDivisionCode": "CA"  
                },  
                "Tch": false  
            }  
        ]  
    }  
}
```

Below the input field are two buttons: "Generate" (blue) and "Generate with Quicktype *" (green). To the right of these buttons are three small icons: a question mark, an exclamation mark, and an '@' symbol. The generated C# code is displayed in a large text area:

```
public class BillAddr  
{  
    public string Id { get; set; }  
    public string Country { get; set; }  
    public string CountrySubDivisionCode { get; set; }  
}  
  
public class CurrencyRef  
{  
    public string value { get; set; }  
    public string name { get; set; }  
}  
  
public class MetaData  
{  
    public DateTime CreateTime { get; set; }  
    public DateTime LastUpdatedTime { get; set; }  
}  
  
public class PrimaryEmailAddr  
{  
    public string Address { get; set; }  
}  
  
public class PrimaryPhone
```

Je colle le code sharp C# généré dans un nouveau fichier de classe



```

Explorateur de solutions QuickBooksOnlines.cs ✘ Startup.cs* Fec.cshtml _Layout.cshtml Index.cshtml.cs Fec.cs ClientsFec.cs
Rechercher dans Explorateur de solution
Solution 'QuikbookWeb2' (1 sur 1 projet)
  QuikbookWeb2
    Connected Services
    Dépendances
    Properties
    wwwroot
    Areas
    Controllers
    Data
    Documentation
    Models
      QuickBooksOnline
        QuickBookClientsFec.cs
        QuickBooksfec.cs
        QuickBooksOnlines.cs
        ClientsFec.cs
        Fec.cs
        note_khalil
        Pages
        Script.Sql
        appsettings.json
        Program.cs
        ScaffoldingReadMe.txt
        Startup.cs

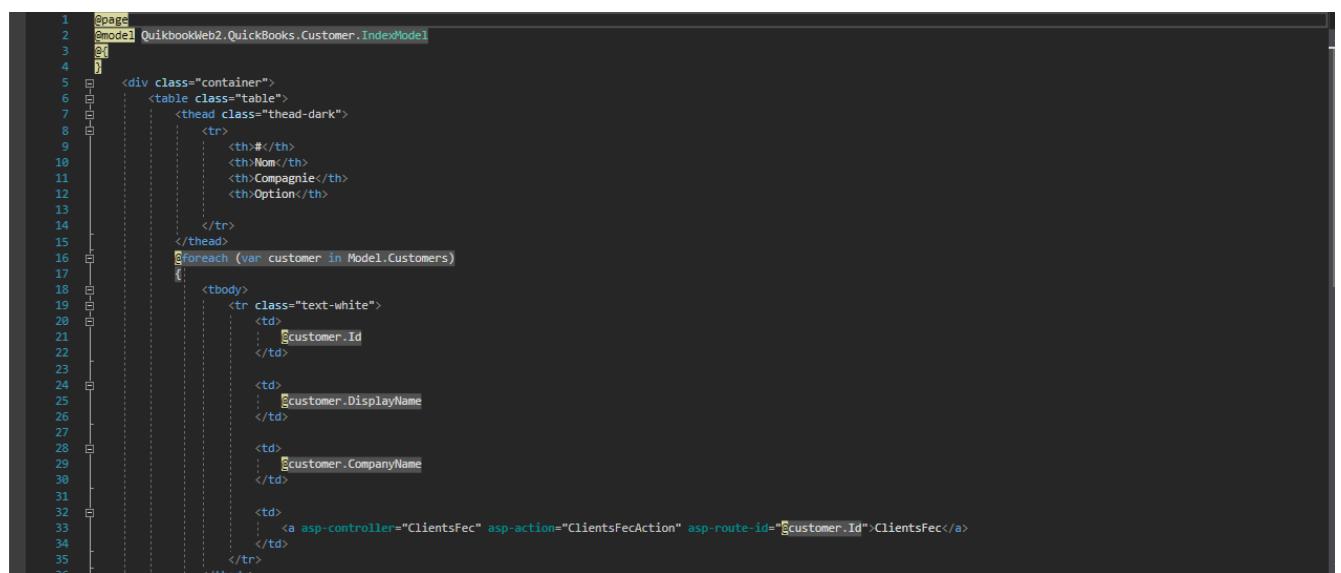
```

```

1 1 using System;
2 2 using System.Collections.Generic;
3 3 using System.Linq;
4 4 using System.Threading.Tasks;
5 5
6 6 namespace QuikbookWeb2.Models.QuickBooksOnline
7 7 {
8 8     2 réferences
9 9         public class QuickBooksOnlines
10 10     {
11 11         1 référence
12 12             public class DefaultTaxCodeRef
13 13             {
14 14                 0 références
15 15                     public string value { get; set; }
16 16             }
17 17             1 référence
18 18                 public class PrimaryEmailAddr
19 19                 {
20 20                     0 références
21 21                         public string Address { get; set; }
22 22             }
23 23             1 référence
24 24                 public class PrimaryPhone
25 25                 {
26 26                     0 références
27 27                         public string FreeFormNumber { get; set; }
28 28             }
29 29             1 référence
30 30                 public class MetaData
31 31                 {
32 32                     0 références
33 33             }
34 34         }
35 35     }
36 36 }

```

J'ajoute une page Razor ou je vais afficher mes clients.



```

1 @page
2 @model QuikbookWeb2.QuickBooks.Customer.IndexModel
3 @{
4 }
5 <div class="container">
6     <table class="table">
7         <thead class="thead-dark">
8             <tr>
9                 <th>#</th>
10                <th>Nom</th>
11                <th>Compagnie</th>
12                <th>Option</th>
13            </tr>
14        </thead>
15        <tbody>
16            @foreach (var customer in Model.Customers)
17            {
18                <tr class="text-white">
19                    <td>
20                        @customer.Id
21                    </td>
22
23                    <td>
24                        @customer.DisplayName
25                    </td>
26
27                    <td>
28                        @customer.CompanyName
29                    </td>
30
31                    <td>
32                        <a href="#" asp-controller="ClientsFec" asp-action="ClientsFecAction" asp-route-id="@customer.Id">ClientsFec</a>
33                    </td>
34                </tr>
35            </tbody>

```

Ma solution ressemble à sa

```

16  public class IndexModel : PageModel
17  {
18      private readonly UserManager<IdentityUser> _userManager;
19      private readonly IHttpContextAccessor _httpContextAccessor;
20
21      2 références
22      public IList<Models.QuickBooksOnline.QuickBooksOnlines.Customer> Customers { get; private set; }
23
24      0 références
25      public IndexModel(IHttpContextAccessor httpContextAccessor, UserManager<IdentityUser> userManager)
26      {
27          _httpContextAccessor = httpContextAccessor;
28          _userManager = userManager;
29      }

```

Ici je fais appelle à mon model auquelle il y a le code générer automatiquement.

Ensute j'insère une injection de dépendence qui va me permettre de ne pas se soucier de l'instanciation d'autres objets /modules dont il dépend et par conséquent de pouvoir le réutiliser dans d'autre classes.

```

29  public async Task OnGetAsync()
30  {
31      var user = await _userManager.GetUserAsync(_httpContextAccessor.HttpContext.User);
32      var accessToken = await _userManager.GetAuthenticationTokenAsync(user, "QuickBooks", "access_token");
33      //var baseUrl = $"https://quickbooks.api.intuit.com/v3/company/123146135740389/query?query=SELECT * FROM Customer ORDERBY Id";
34      var baseUrl = $"https://sandbox-quickbooks.api.intuit.com/v3/company/4620816365004689390/query?query=select * from customer";
35      HttpWebRequest qboApiRequest = (HttpWebRequest)WebRequest.Create(baseUrl);
36      qboApiRequest.Method = "GET";
37      qboApiRequest.Headers["Authorization"] = string.Format("Bearer {0}", accessToken);
38      qboApiRequest.ContentType = "application/json; charset=UTF-8";
39      qboApiRequest.Accept = "application/json";
40      try
41      {
42          // get the response
43          var response = await qboApiRequest.GetResponseAsync();
44          HttpWebResponse qboApiResponse = (HttpWebResponse)response;
45          //read qbo api response
46          using (var qboApiReader = new StreamReader(qboApiResponse.GetResponseStream()))
47          {
48              var result = qboApiReader.ReadToEnd();
49              var rootObj = JsonConvert.DeserializeObject<Models.QuickBooksOnline.QuickBooksOnlines.RootObject>(result);
50              Customers = rootObj.QueryResponse.Customer.ToList();
51          }
52      }
53      catch (WebException ex)
54      {
55          if (ex.Message.Contains("401"))
56          {
57          }
58          else
59          {
60          }
61      }

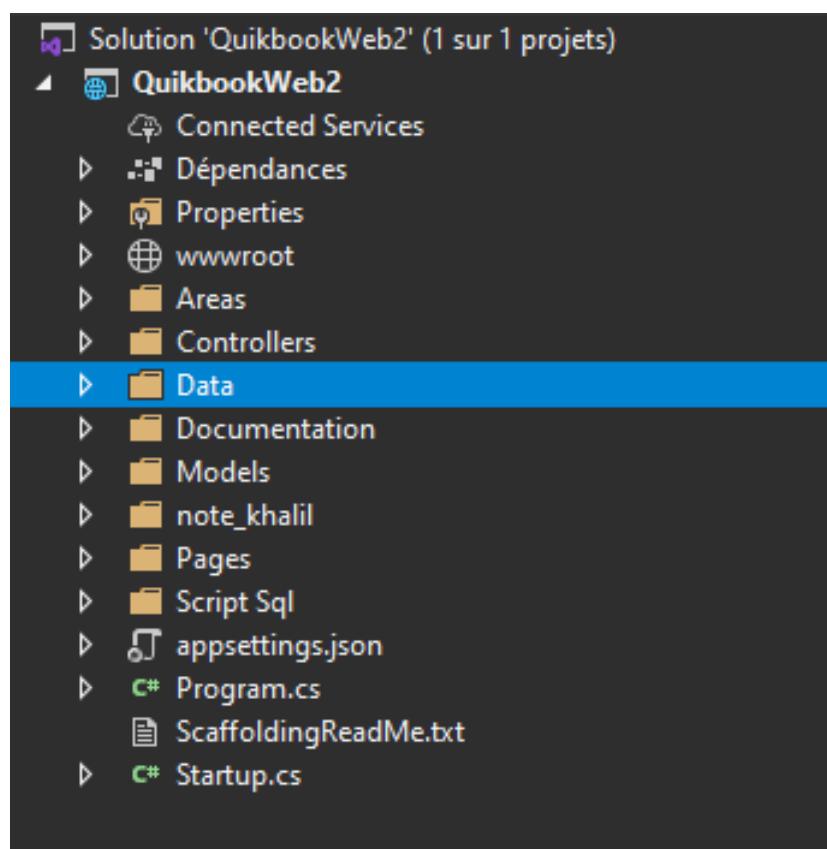
```

Ici je suis la doc quickbook et insère tout les donnée nécessaire à la communication entre mon application et Quickbook.

Ensute j'envoie une demande sous forme de requete sql et reçois une réponse que je deserialise via mon model et ma classe rootobjet et enfin j'insère c'est donner dans une variable .List().

Pattern MVC

J'utilise le patern MVC je crée donc un dossier Model, Vues et contrôleur que j'ai appeler ici pages car il à étais automatiquement généré par Razor.



Le pattern MVC permet de bien organiser son code source. Il va vous aider à savoir quels fichiers créer, mais surtout à définir leur rôle. Le but de MVC est justement de séparer la logique du code en trois parties que l'on retrouve dans des fichiers distincts.

Developpement du Fichier écriture comptable.

Pour mon fichier écriture comptable j'utilise le patern mvc, je crée donc un model que je vais appelé QuickbookClientFec() auquelle je vais suivre les même étapes du model QuickbookListClient().

Query a report

```

{
  "Header": {
    "ReportName": "JournalReportFR",
    "Option": [
      {
        "Name": "NoReportData",
        "Value": "false"
      }
    ],
    "DateMacro": "this month-to-date",
    "StartPeriod": "2016-05-01",
    "EndPeriod": "2016-05-16"
  }
}
  
```

Je copie donc le json et le transforme en classe C# à l'aide d'un générateur automatique.

```

public class BillAddr
{
    public string Id { get; set; }
    public string Country { get; set; }
    public string CountrySubDivisionCode { get; set; }
}

public class CurrencyRef
{
    public string value { get; set; }
    public string name { get; set; }
}

public class MetaData
{
    public DateTime CreateTime { get; set; }
}
  
```

Ensuite je crée une classe ClienFec() qui va contenir le code pour faire une requete à l'api Quickbook.

```

13  public class ClientsFec
14  {
15      private readonly UserManager<IdentityUser> _userManager;
16      private readonly IHttpContextAccessor _httpContextAccessor;
17
18      1 référence
19      public ClientsFec(IHttpContextAccessor httpContextAccessor, UserManager<IdentityUser> userManager)
20      {
21          _httpContextAccessor = httpContextAccessor;
22          _userManager = userManager;
23      }
24

```

Ici je fais appel a mon model auquelle il y a le code générer automatiquement.

Ensuite j'insére une injection de dépendence qui va me permettre de ne pas se soucier de l'instanciation d'autres objets /modules dont il dépend et par conséquent de pouvoir le réutiliser dans d'autre classes.

```

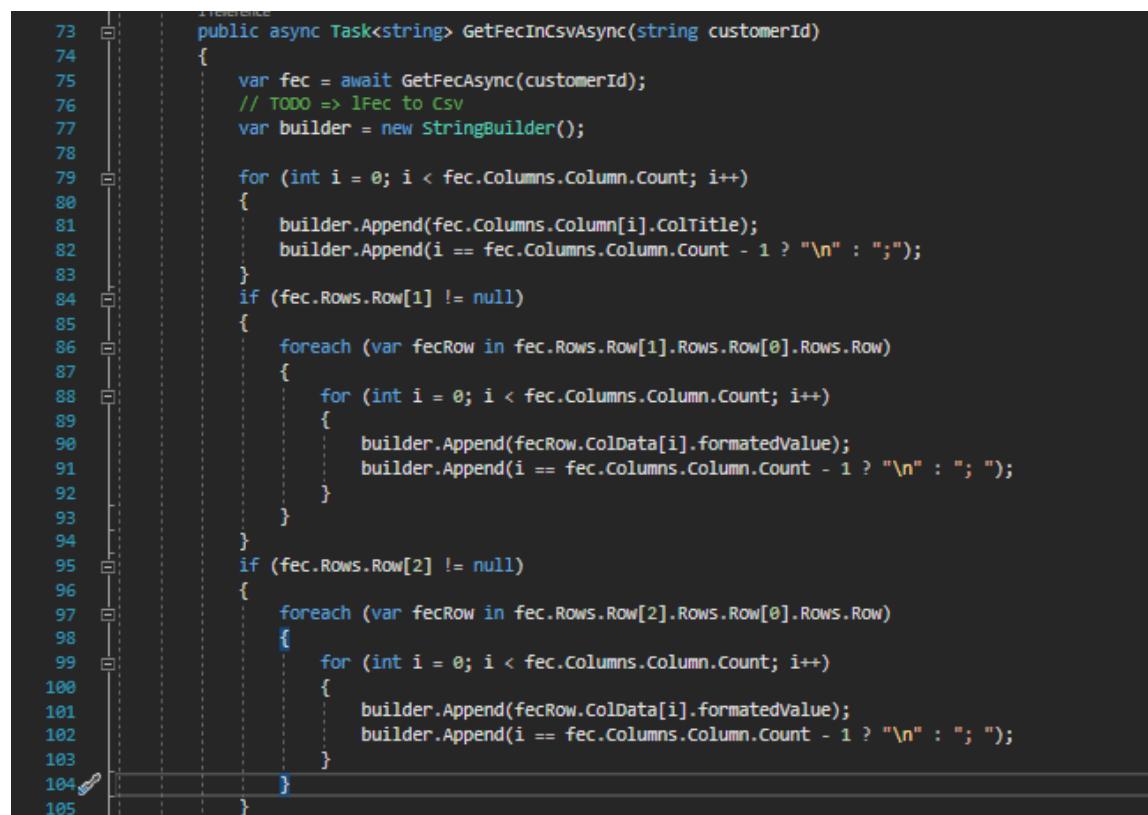
25  1 référence
26  public async Task<Models.QuickBooksOnline.QuickBookClientsFec.Root> GetFecAsync(string customerId)
27  {
28
29      Models.QuickBooksOnline.QuickBookClientsFec.Root IClientsFec = null;
30      var user = await _userManager.GetUserAsync(_httpContextAccessor.HttpContext.User);
31      var accessToken = await _userManager.GetAuthenticationTokenSync(user, "QuickBooks", "access_token");
32      //var baseUrl = $"https://quickbooks.api.intuit.com/v3/company/193514843689529/reports/JournalReportR?date_macro=Last Month";
33      var baseUrl = $"https://sandbox-quickbooks.api.intuit.com/v3/company/4620816365004689390/reports/JournalReportR?date_macro=Last Fiscal Year-to-date";
34      HttpWebRequest qboApiRequest = (HttpWebRequest)WebRequest.Create(baseUrl);
35      qboApiRequest.Method = "GET";
36      qboApiRequest.Headers["Authorization"] = string.Format("Bearer {0}", accessToken);
37      qboApiRequest.ContentType = "application/json;charset=UTF-8";
38      qboApiRequest.Accept = "application/json";
39      try
40      {
41          // get the response
42          var response = await qboApiRequest.GetResponseAsync();
43          HttpWebResponse qboApiResponse = (HttpWebResponse)response;
44          //read qbo api response
45          using (var qboApiReader = new StreamReader(qboApiResponse.GetResponseStream()))
46          {
47              var result = qboApiReader.ReadToEnd();
48              IClientsFec = JsonConvert.DeserializeObject<Models.QuickBooksOnline.QuickBookClientsFec.Root>(result);
49          }
50      }
51

```

Ici je suis la doc quickbook et insére tout les donnée nécessaire à la communication entre mon application et Quickbook.

Ensuite j'envoie une demande sous forme de requete sql et reçoie une réponse que je deserialize directement dans ma variable via mon model et ma classe root.

Je crée dans mon model une classe qui va me permettre de convertir les données au format CSV afin d'avoir un visuelle avec excel.



```
73     public async Task<string> GetFecInCsvAsync(string customerId)
74     {
75         var fec = await GetFecAsync(customerId);
76         // TODO => lFec to Csv
77         var builder = new StringBuilder();
78
79         for (int i = 0; i < fec.Columns.Column.Count; i++)
80         {
81             builder.Append(fec.Columns.Column[i].ColTitle);
82             builder.Append(i == fec.Columns.Column.Count - 1 ? "\n" : " ");
83         }
84         if (fec.Rows.Row[1] != null)
85         {
86             foreach (var fecRow in fec.Rows.Row[1].Rows.Row[0].Rows.Row)
87             {
88                 for (int i = 0; i < fec.Columns.Column.Count; i++)
89                 {
90                     builder.Append(fecRow.ColData[i].formatedValue);
91                     builder.Append(i == fec.Columns.Column.Count - 1 ? "\n" : " ");
92                 }
93             }
94         }
95         if (fec.Rows.Row[2] != null)
96         {
97             foreach (var fecRow in fec.Rows.Row[2].Rows.Row[0].Rows.Row)
98             {
99                 for (int i = 0; i < fec.Columns.Column.Count; i++)
100                {
101                    builder.Append(fecRow.ColData[i].formatedValue);
102                    builder.Append(i == fec.Columns.Column.Count - 1 ? "\n" : " ");
103                }
104            }
105        }
106    }
```

Ici je parcours mon tableau afin d'insérer les données une par une en remplaçant les saut de ligne par un point virgule qui est pour un fichier excel une nouvelle colonne.

En métant une condition de parcourir mais rows que si les données ne sont pas null.

Je bifurque donc dans mon controller afin d'envoyer les données à ma vues.

```

10  public class FecController : Controller
11  {
12      #region Fields injection dependance
13      private readonly UserManager<IdentityUser> _userManager;
14      private readonly IHttpContextAccessor _httpContextAccessor;
15
16      0 références
17      public FecController(IHttpContextAccessor httpContextAccessor, UserManager<IdentityUser> userManager)
18      {
19          _httpContextAccessor = httpContextAccessor;
20          _userManager = userManager;
21      }
22      #endregion

```

Ici je fais appel à nouveau a mes injection de dépendance tout sa en-glober par un #region et #endregion qui permet d'enpaqueter le code afin que mon contrôleur soit plus visible et enlaissant un petit commentaire après région pour savoir se que réalise ma méthode.

```

10  public class FecController : Controller
11  {
12      #region Fields injection dependance
13
14      #region Method envoie données tableau et bouton CSV
15      [HttpGet]
16      0 références
17      public IActionResult FecAction()
18      {
19          // KSA 07-07-2020 : A ce jour Quickbooks ne semble pas savoir
20          // filtrer les FEC par client. customerId est là pour préparer
21          // l'avenir.
22          var customerId = "";
23          var clientsFec = new Fec(_httpContextAccessor, _userManager);
24          var fec = clientsFec.GetFecAsync(customerId).Result;
25          return View("~/Pages/Fec/Fec.cshtml", fec);
26      }
27
28      [HttpGet]
29      0 références
30      public IActionResult FecToCsv()
31      {
32          var customerId = "";
33          var clientsFec = new Fec(_httpContextAccessor, _userManager);
34          var csvFec = clientsFec.GetFecInCsvAsync(customerId).Result;
35          return File(Encoding.UTF8.GetBytes(csvFec), "text/csv", "FecInfo.csv");
36      }
37      #endregion
38  }
39
40
41
42
43
44
45
46
47

```

Ici j'instancie ma classe Fec() en paametre mes injection de dépendance et j'envoie les données à ma vues fec.cshtml.

Ensuite j'instancie denouveau ma classe Fec() en fesant appelle à la méthode GetFecInCsvAsync() qui va me permettre de crée un bouton export en CSV et de convertir es donnée au format CSV.

Cette étape va me permettre d'afficher les données récupérées dans une page HTML.

```

1  @model dynamic
2
3  @{
4      Layout = "~/Pages/Shared/_Layout.cshtml";
5  }
6
7  @{
8      ViewBag.Title = "Index";
9  }
10
11 <a href="@Url.Action("FecToCsv", "Fec")">
12     <img src("~/picture/excel.png" class="center" title="Export compatible CSV" />
13 </a>
14

```

Ici je fais appel à mon layout qui va me permettre de récupérer le css et la navbar.

url.action va me permettre de faire appel à mon controller pour la conversion au format CSV.

```

15 <table class="table">
16     <thead class="thead-dark">
17         <tr>
18             @foreach (var columnscolumn in @Model.Columns.Column)
19             {
20                 <th>
21                     @columnscolumn.ColTitle
22                 </th>
23             }
24         </tr>
25     </thead>
26     <tbody>
27
28     </tbody>

```

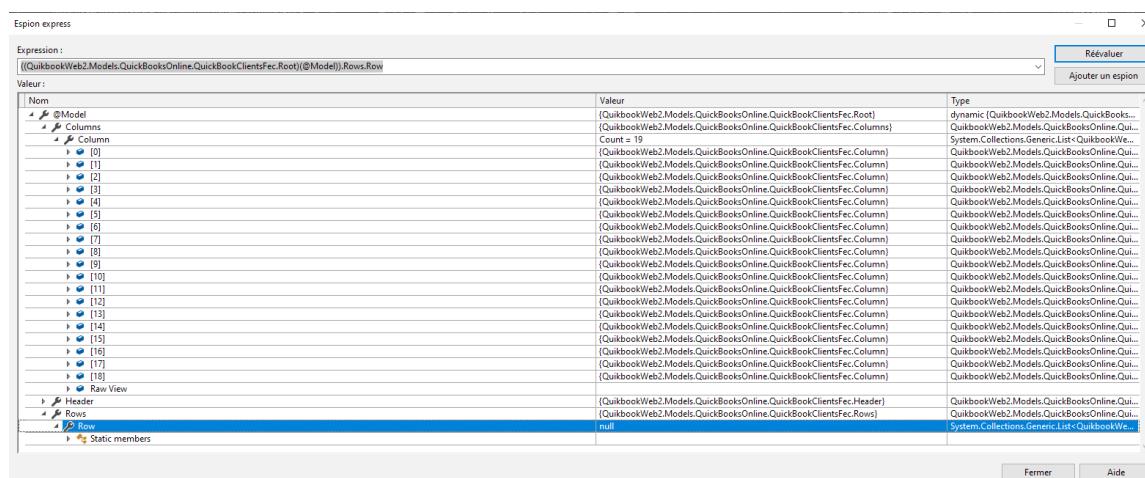
Je crée une balise Table qui va me permettre d'insérer mes données dans un tableau, pour mes colonnes j'utilise la balise thread suivie de tr et d'une boucle foreach qui me permet de parcourir mes données et de les afficher un par un.

Je suis cette méthodologie je crée une balise tbody qui va me permettre d'insérer les rows dans mon tableau. Avec la boucle foreach je parcour mes donnée et les insére un par un.

```

26     <tbody>
27         @if (@Model.Rows.Row.Count >= 3)
28     {
29         @foreach (var fecRow in @Model.Rows.Row[1].Rows.Row[0].Rows.Row)
30     {
31             <tr>
32                 @foreach (var colData in fecRow.ColData)
33             {
34                 <td class="text-white">
35                     @colData.formatedValue
36                 </td>
37             }
38         </tr>
39     }
40 }
41         @if (@Model.Rows.Row.Count >= 3)
42     {
43         @foreach (var fecRow in @Model.Rows.Row[2].Rows.Row[0].Rows.Row)
44     {
45             <tr>
46                 @foreach (var colData in fecRow.ColData)
47             {
48                 <td class="text-white">
49                     @colData.formatedValue
50                 </td>
51             }
52         </tr>
53     }
54 }
55     </tbody>
56 </table>
57 
```

En visualisant mes donnée et son arborescentce grâce à mon espion express, j'ai pu cibler mes donnée à la perfections.



3.3.2 Développement Front-end de l'application

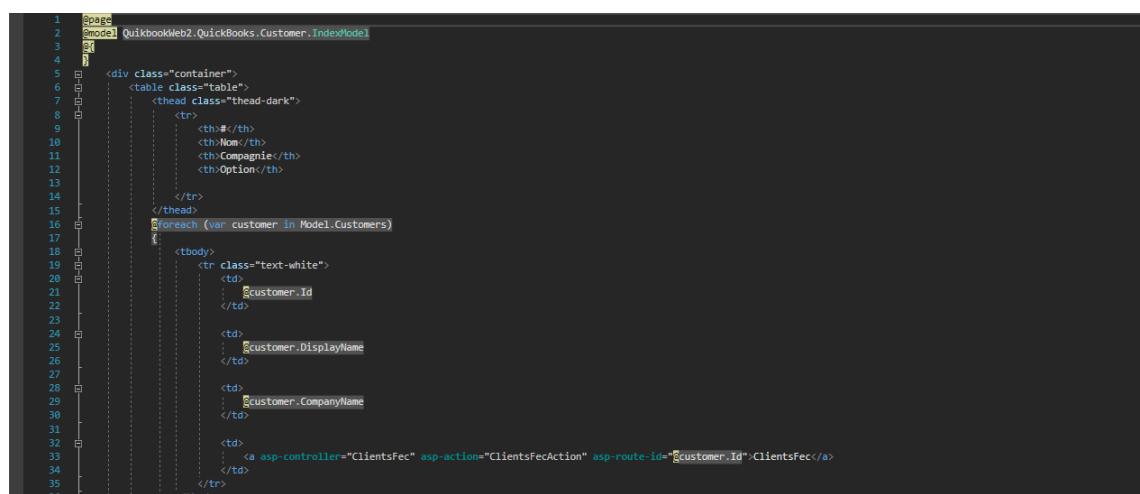
Rendu HTML

La langue par défaut Razor est html. Le rendu HTML à partir du Razor balisage n'est pas différent du rendu HTML d'un fichier html. Le balisage HTML dans les fichiers .cshtml Razor est rendu par le serveur inchangé.

Syntaxe de Razor

Razor prend en charge C# et utilise le @ symbole pour passer du code HTML à c#. Razor évalue les expressions C# et les restitue dans la sortie HTML.

Quand un @ symbole est suivi d'un Razor mot clé réservé, il passe à un Razor balisage spécifique. Sinon, il est converti en code C# brut.



```
1  @page
2  @model QuikbookWeb2.QuickBooks.Customer.IndexModel
3
4
5  <div class="container">
6      <table class="table">
7          <thead class="thead-dark">
8              <tr>
9                  <th>#</th>
10                 <th>Nom</th>
11                 <th>Compagnie</th>
12                 <th>Option</th>
13             </tr>
14         </thead>
15         <tbody>
16             @foreach (var customer in Model.Customers)
17             {
18                 <tr class="text-white">
19                     <td>
20                         @customer.Id
21                     </td>
22
23                     <td>
24                         @customer.DisplayName
25                     </td>
26
27                     <td>
28                         @customer.CompanyName
29                     </td>
30
31                     <td>
32                         <a asp-controller="ClientsFee" asp-action="ClientsFeeAction" asp-route-id="@customer.Id">ClientsFee</a>
33                     </td>
34                 </tr>
35             }
36         </tbody>
37     </table>
38 </div>
```

CSS

Pour le css je vais allez wwwroot et le dossier css et rajouté pour chaque partie de mon css un commentaire pour différencier chaque code et leur fonctionnement.

```

84 /*Tableaux css*/
85 .table-wrapper {
86   background: #fff;
87   padding: 20px 25px;
88   margin: 30px 0;
89   margin-top: 100px;
90   border-radius: 3px;
91   box-shadow: 0 1px 1px rgba(0, 0, 0, .05);
92 }
93
94 .table-title {
95   padding-bottom: 15px;
96   background: #435d7d;
97   color: #fff;
98   padding: 16px 30px;
99   margin: -20px -25px 10px;
100  border-radius: 3px 3px 0 0;
101 }
102
103 .table-title h2 {
104   margin: 5px 0 0;
105   font-size: 24px;
106 }
107
108 .table-title .btn-group {
109   float: right;
110 }
111
112 .table-title .btn {
113   color: #fff;
114   float: right;
115   font-size: 13px;
116   border: none;
117   min-width: 50px;
118   border-radius: 2px;
119   border: none;
120   outline: none !important;
121 }
```

```

4 a.navbar-brand {
5   white-space: normal;
6   text-align: center;
7   word-break: break-all;
8   background-color: darkorange;
9 }
10
11 nav{
12   background-color:darkorange;
13   width: auto !important;
14 }
15
16 .btn-primary {
17   color: #fff;
18   background-color: #1b6ec2;
19   border-color: #186iac;
20 }
21
22 .nav-pills .nav-link.active, .nav-pills .show > .nav-link {
23   color: white;
24   background-color: #1b6ec2;
25   border-color: #186iac;
26 }
```

3.3.3 Déploiement

Dans un premier temps, l'application Web doit être déployée sur un serveur local accessible depuis le réseau interne de l'entreprise. Elle sera déployée plus tard par un autre intervenant.

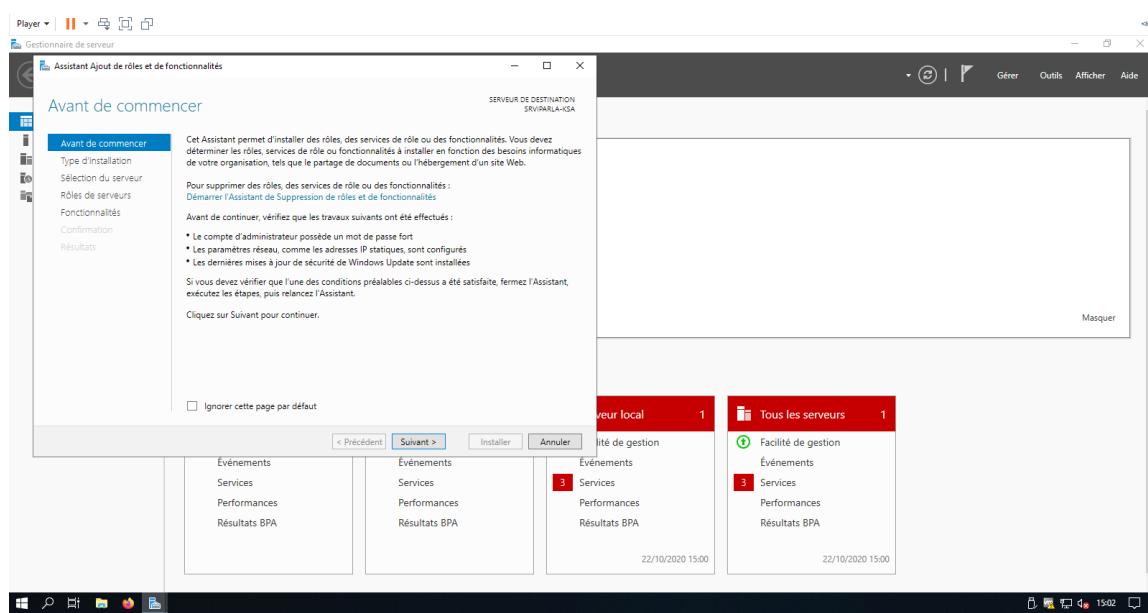
Dans un premier temps, j'installe sur ma machine virtuelle windows server qui servira de serveur, l'environnement dont à besoin l'application web: un windows server ainsi qu'un Serveur IIS Express.

Mise en place windows server

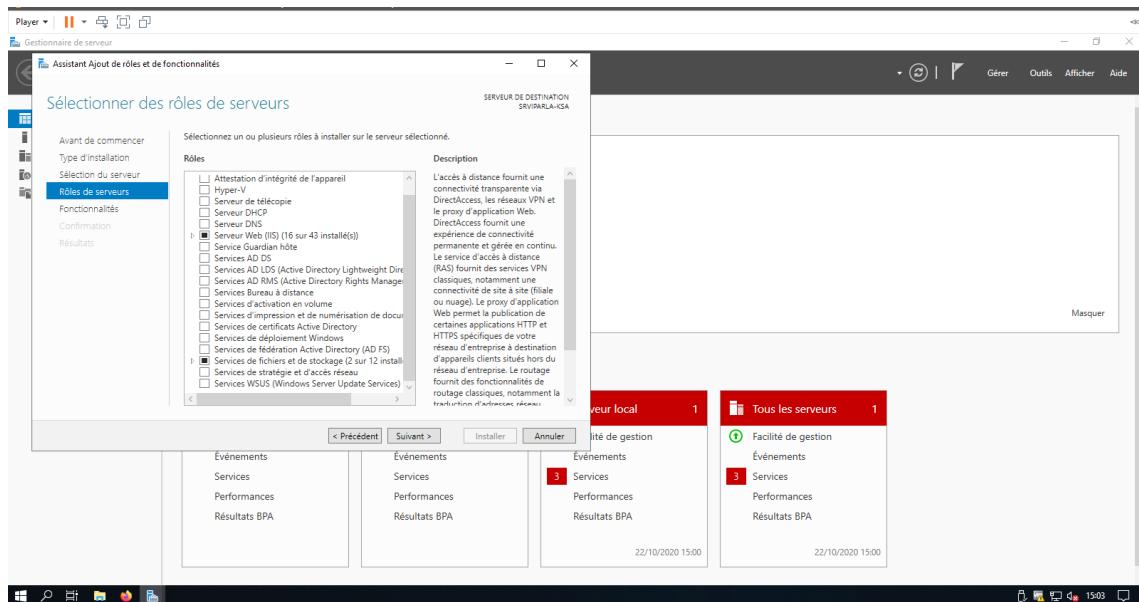
Se référer page 54.

Mise en place server IIS

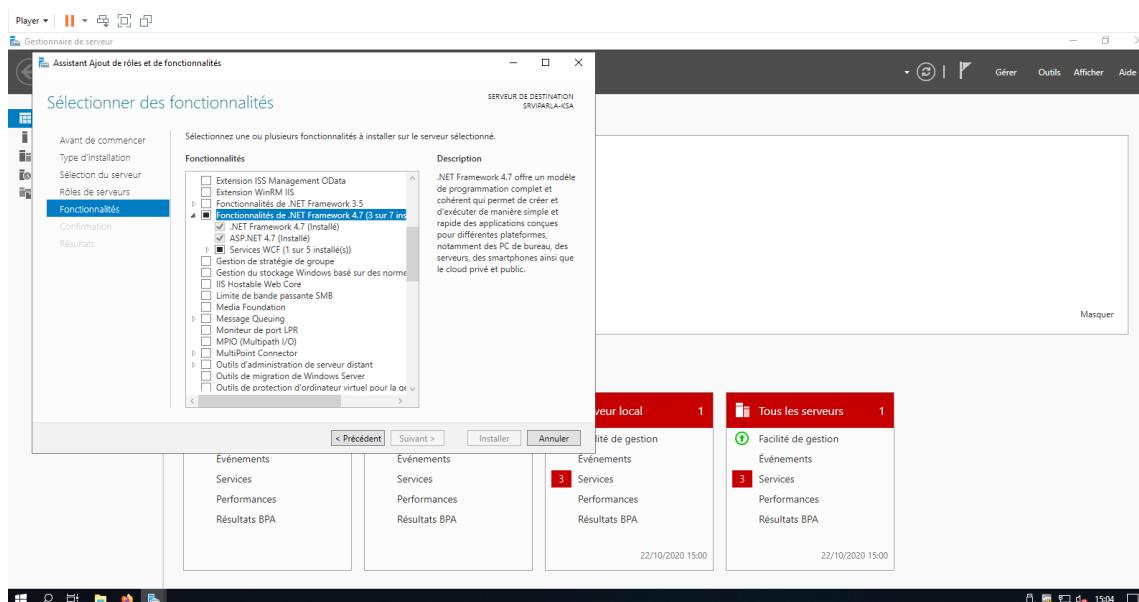
Pour la mise en place du server IIS, je vais sur mon gestionnaire de serveur ensuite je sélectionne ajout de fonctionnalité.



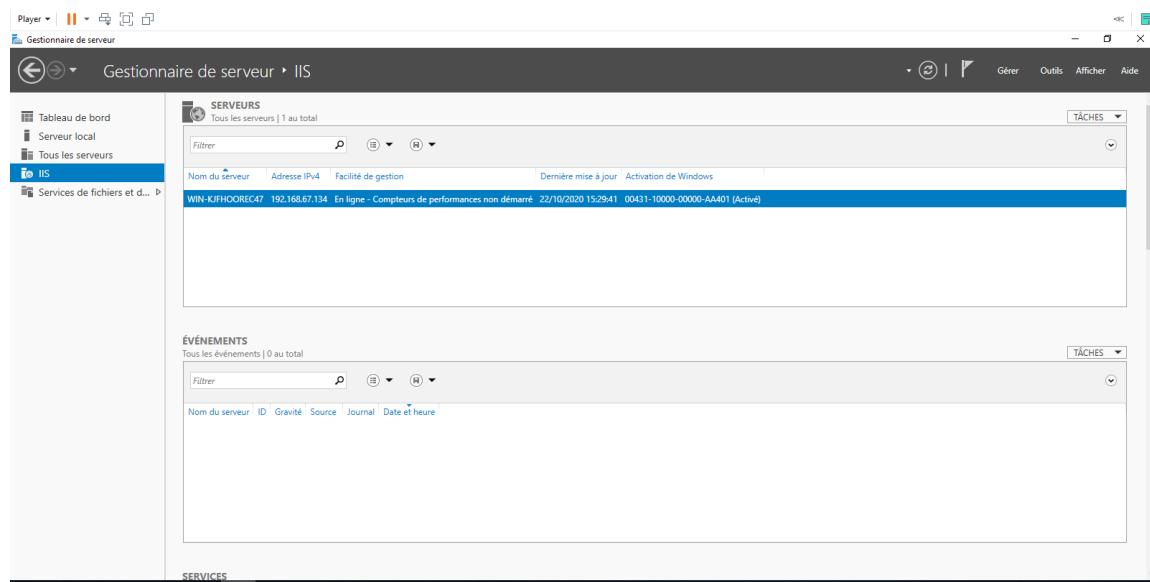
Je mets suivant, arriver au moment où je précise mon server web IIS.



En fonctionnalité je rajoute tous ce qui est en lien avec mon projet.



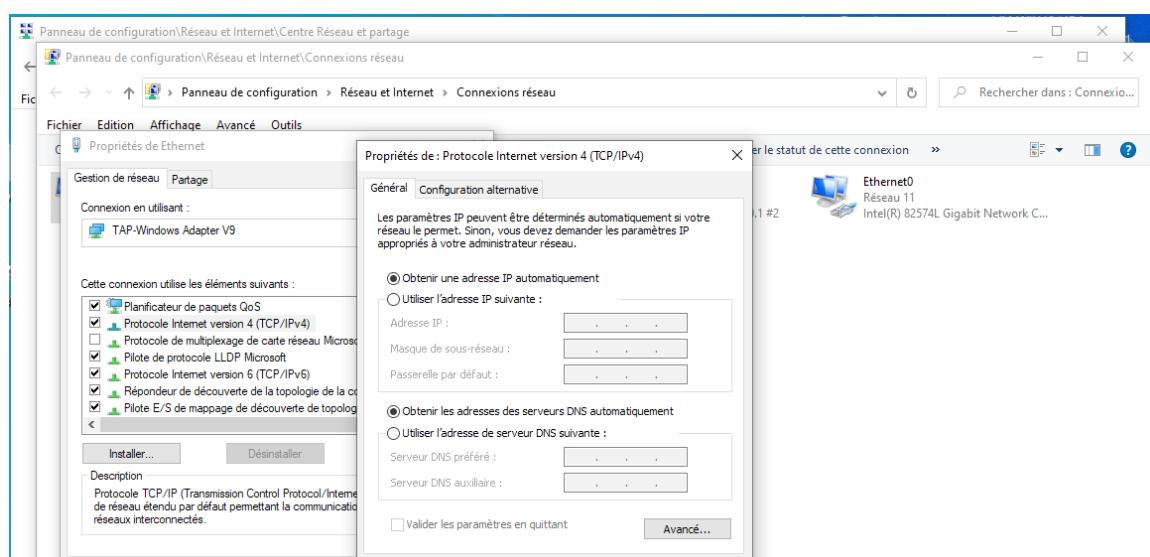
Ensuite je lance l'installation et redémarre mon windows server.



Ici mon installation à fonctionner car un onglet IIS est apparu.

Mise en place server DNS

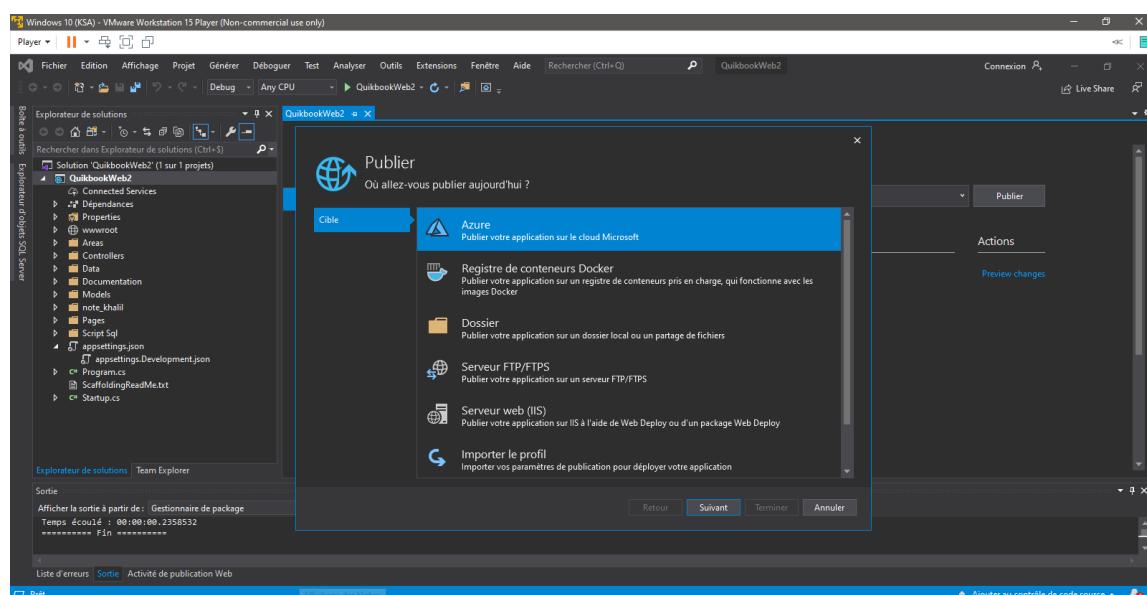
Le serveur DNS (Domain Name System, ou Système de noms de domaine en français) est un service dont la principale fonction est de traduire un nom de domaine en adresse IP. Pour simplifier, le serveur DNS agit comme un annuaire que consulte un ordinateur au moment d'accéder à un autre ordinateur via un réseau.



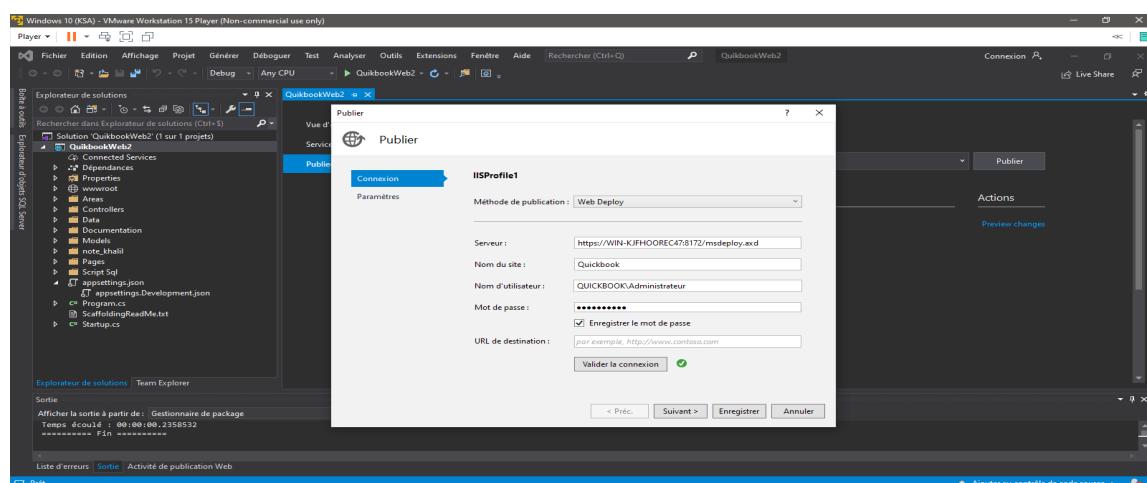
Ici j'insère mon adresse dns de mon windows server.
 Il faut absolument un windows Professional pour que cela fonctionne.

Déploiement sur windows server

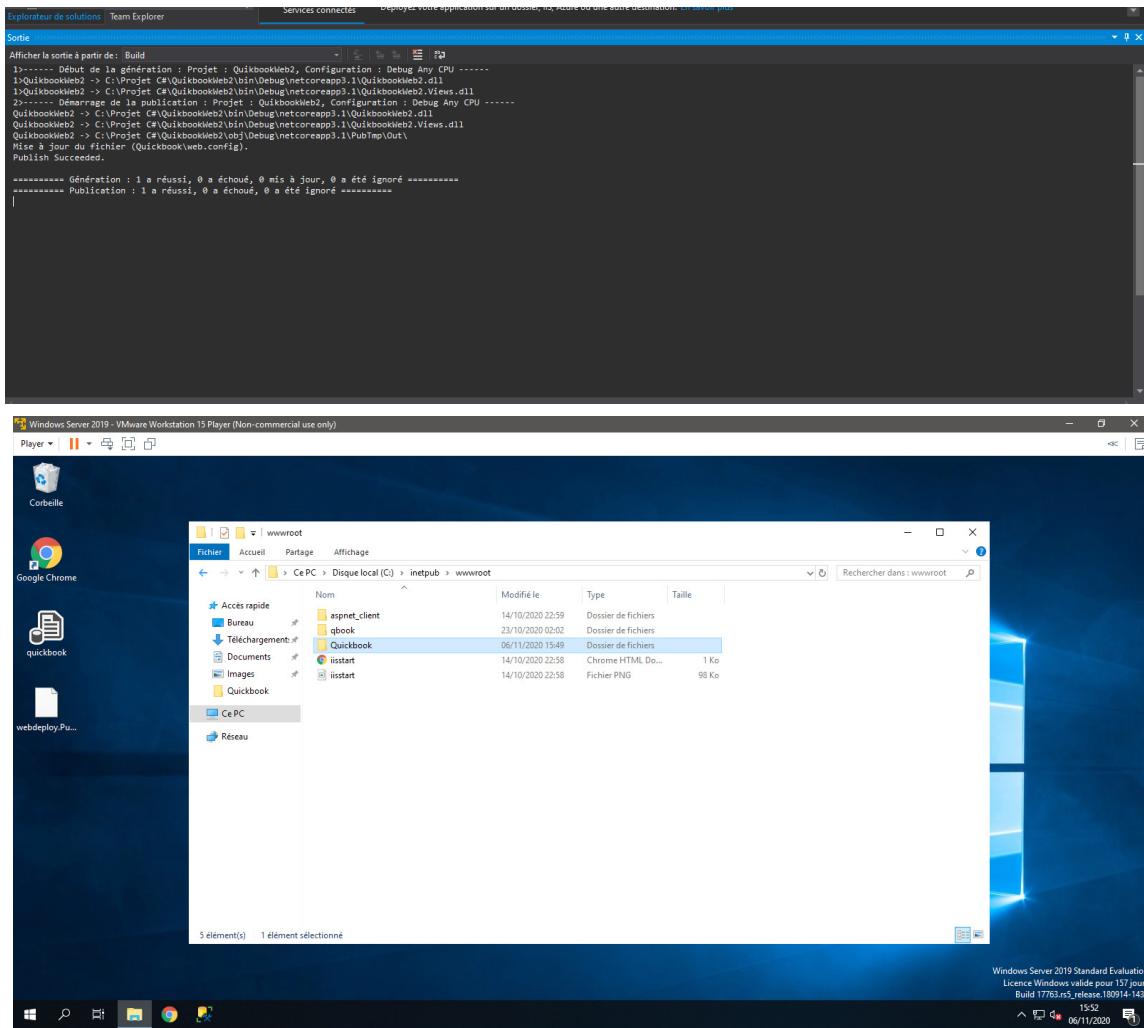
Ici je fais clic droit publier et on peut voir qu'on à différente façon de déployer son application.



Ici on remplit les champs et on clic sur validé la connexion.
 Une fois la connexion validé on publie.



Une fois la publication réussie on retourne sur notre windows server afin de vérifié si le projet à étais publié à l'endroit ciblé.



Voila notre site et publié.

4 - Conclusion

Pour conclure, je dirais que j'ai vraiment apprécié de mettre en pratique mes compétences dans un cadre professionnel. J'ai bien conscience que cette expérience ne couvre pas toutes celles demandées par le titre professionnel, mais elle m'a toutefois permis d'avoir un peu plus confiance en moi et d'appréhender ma recherche d'emploi dans le développement avec plus de sérénité.