Documentation
BTS SIO SLAM

Organiser son développement professionel

Par Damien Monchaty, Avril 2023

Sommaire

Mettre en place son environnement d'apprentissage personnel

Pour me former sur divers domaines de l'informatique, je regarde des **videos – tutoriels** sur **Youtube** et **Internet**

Voici les liens utilisés :

- https://leetcode.com/
- https://grafikart.fr/tutoriels
- https://www.youtube.com/@Alphorm
- https://www.w3schools.com/
- https://openclassrooms.com/fr/
- https://code-maze.com/
- https://www.developpez.net

Pour toute autre information nécessaire, je me documente sur les **différentes documentations accessibles** sur Internet, comme par exemple :

- https://learn.microsoft.com/fr-fr/dotnet/csharp/programming-guide/
- https://sql.sh/
- https://www.nuget.org/

En cas de problèmes, je regarde sur les **forums** pour trouver des solutions comme :

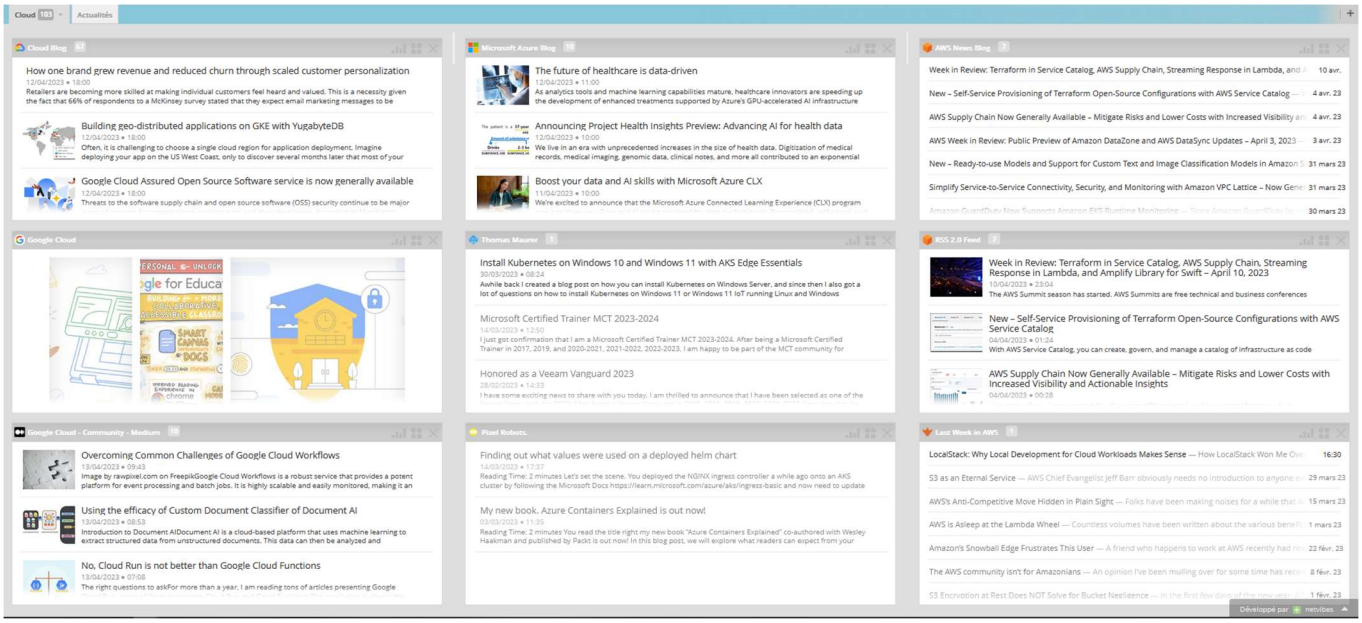
- https://stackoverflow.com/
- https://social.msdn.microsoft.com

Mettre en œuvre des outils et stratégies de veille informationnelle

NetVibes est un agrégateur de **flux RSS.**
Il s'agit d'un outil permettant de suivre en temp réel l'actualité de différent site internet et de regrouper cet ensemble de contenus dans un meme espace.

Ce service m'a permis de me constituer un tableau au thématique de mon choix, c'est-à-dire, le **Cloud Computing et leurs principaux acteurs : AWS, Google Cloud & Microsoft Azure.**

Vous pouvez constater ici que **NetVibes** montre bien les dernières nouvelles concernant le **Cloud Computing** :

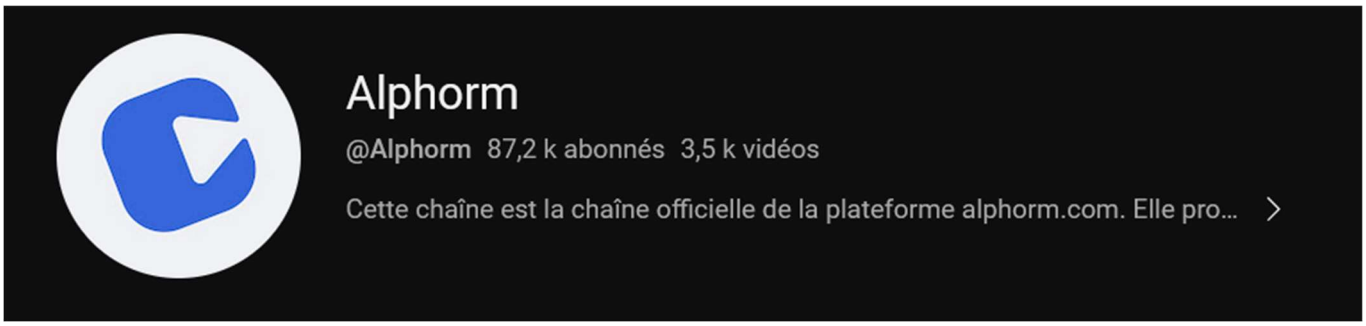


En complément, je regarde également les actualités sur les sites suivants :

- https://www.silicon.fr/actualites/cloud
- https://www.lemagit.fr/ressources/Services-Cloud
- https://www.usine-digitale.fr/cloud/
- https://siecledigital.fr/tag/aws/

De plus, je regarde également les chaines **Youtube** suivantes **:**

Google Cloud Tech présente les dernières mises à jour de **Google Cloud Platform, puis** propose des **tutoriels** pour tous les niveaux**.**



Alphorm propose des tutoriels pour tous les niveaux **d**ans différents domaines technologiques comme **la virtualisation, le cloud computing.**

Gérer son identité professionnelle

Vous pouvez trouver un lien hypertexte vers mon profil **LinkedIn** sur mon portfolio, en plus de mon compte **Github**.

En tant que plateforme, **LinkedIn** propose une approche ciblée, me permettant de me présenter aux entreprises avec précision.

► https://www.linkedin.com/in/damien-monchaty-691404b3/

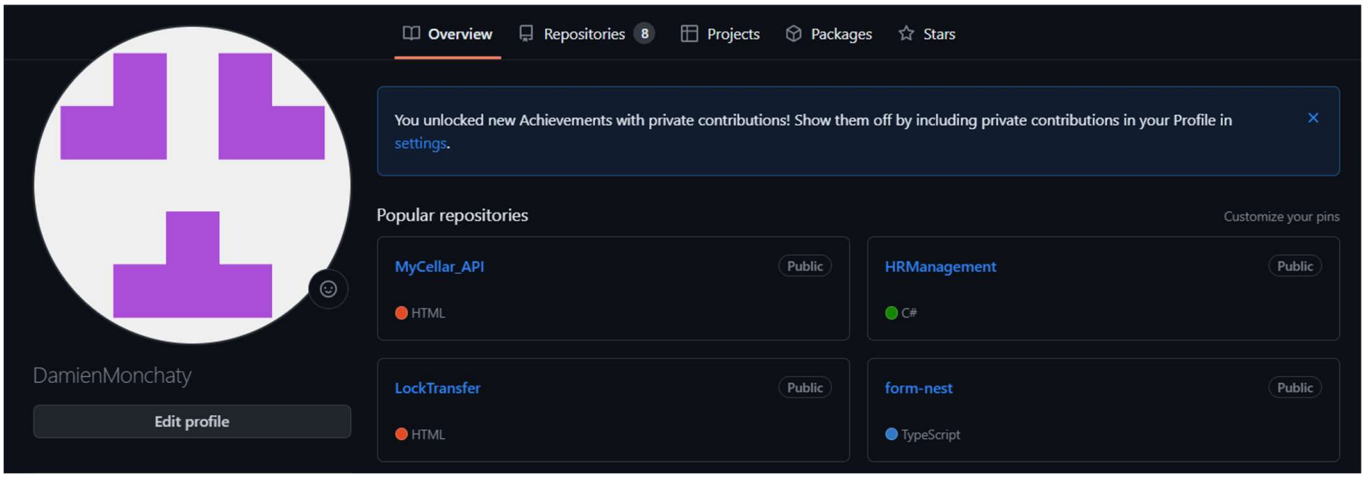
Github est une plateforme vitale pour les développeurs.

Il sert de **référentiel** pour les codes de projet et permet aux utilisateurs de les récupérer facilement en cas d'incident ou pour un travail d'équipe plus fluide grâce à une base de code partagée.

Un autre avantage est la possibilité de mettre en valeur nos compétences grâce à l'ajout de divers projets et engagements.

Cela nous permet de démontrer nos contributions et nos compétences dans différents domaines.

► https://github.com/DamienMonchaty



Développer son projet professionnel

Pourquoi le métier d'**analyste-programmeur** ?

J'ai toujours été intéressé par **l'informatique** depuis mon enfance et la façon d'exprimer des idées, des créations en des approches différentes. Ce qui m'intéresse le plus dans l'informatique, c'est de manier les langages pour y trouver une solution en tenant compte de nombreux besoins attendus par le client.

Mon **projet professionnel** est de m'épanouir dans le développement web et par la suite évoluer dans les Alpes-Maritimes dans des postes plus hiérarchiques (chef de projet…).

Durant ces années, j'ai étudié par le biais de tutoriels pour approfondir mes connaissances acquises tout au long de mon **BTS SIO** et les termes anglais spécifiques à l'informatique.

Je me suis renseigné sur les nouvelles technologies afin de rester au goût du jour comme les technologies **Front-End – Angular** et **Cloud**

Par conséquent, après le **BTS SIO**, je compte ensuite démarcher en tant que **Développeur Full-Stack.**

Voici un aperçu du projet **LockSelf** que j'ai fait durant mon deuxième stage dans la société de Consulting **Amri Code** :

Maître de stage : M. Mohamed Rezgui

Contexte : LockSelf est un **complément Outlook** qui permet d'envoyer des pièces jointes de manière sécurisée.

url : https://www.lockself.com/solutions/LockTransfer-transfert-fichier-securise

Durant ce stage, j'ai dû :

- Consommer un API https://api.lockself.com/api/doc
- Respecter le **Design Archtectural MVVM** Model – View – ViewModel
- Programmer des vues en **XAML,** qui est comme **HTML,** un langage de balisage déclaratif.

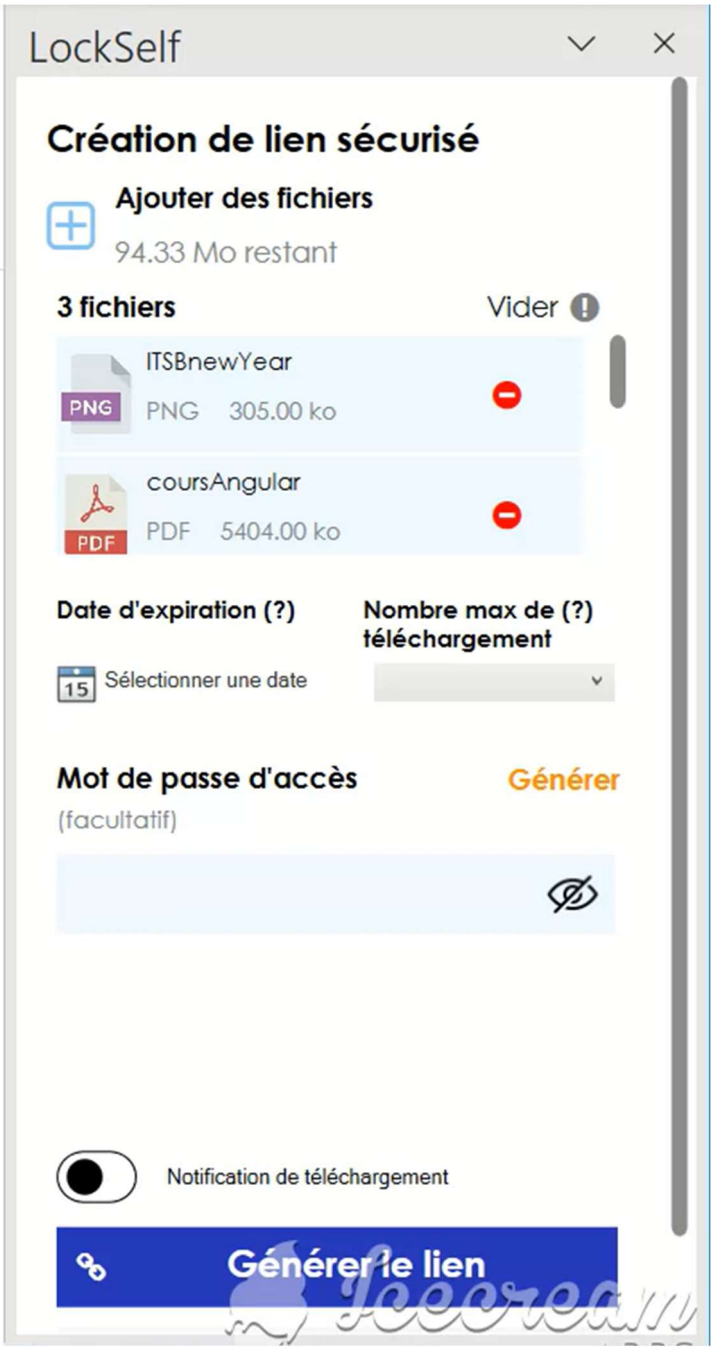
► Etape 1 :

Dans cette étape, l ’expéditeur pourra :

► Cliquer dans le conteneur pour qu’on puisse ajouter un ou plusieurs fichiers à partir de l’explorateur de fichier

► Glisser-déposer un ou plusieurs fichiers dans le conteneur

- Etape 2 :



Dans cette étape, l'expéditeur pourra préciser des **arguments**:

- Rajouter d'autres fichiers jusqu'à la limite de 2 Go.
- Supprimer un fichier
- Supprimer tous les fichiers
- Donner une date d'expiration au lien de téléchargement
- Ajouter un nombre maximum de téléchargement
- Sécuriser le téléchargement avec un mot de passe
- Saisir un numéro de téléphone pour envoyer le mot de passe
- Générer le lien de téléchargement

Voici le code de la méthode **UploadFileAsync(…)** en **C#** qui récupère tous ces **arguments** et les envoye en tant que paramètres de requête dans le chemin :

► https://api.lockself.com/api/transfers/password

Nous recevrons en tant que réponse le lien de téléchargement

```
public async Task<string> UploadFileAsync(string urlApi, User user,int uploadNumber, List<File> files, IProgress<int> progress, string expirationDate = "", string password = "", string phone = "")
{
    try
    {
        progress.Report(0);
        var request = new RestRequest($"/transfers/password", Method.POST);
        request.AlwaysMultipartFormData = true;
        request.AddHeader("Content-Type", "multipart/form-data");
        ServicePointManager.SecurityProtocol = (SecurityProtocolType)192 | (SecurityProtocolType)768 | (SecurityProtocolType)3072 | System.Net.SecurityProtocolType.Ssl3;
        string access_token = user.token;
        _restClient.Authenticator = new OAuth2AuthorizationRequestHeaderAuthenticator(
            access_token, "Bearer"
        );
        request.AddHeader("Accept", "application/json");
        using (ZipFile zip = new ZipFile())
        {
            List<string> filenames = new List<string>();
            foreach (var file in files)...
            zip.AddFiles(filenames, "projet");//Zip file inside filename
            string tmpPath = Path.GetTempPath();
            var tmpZipPath = Path.Combine(tmpPath, "test.zip");
            zip.Save(tmpZipPath);
            Console.WriteLine(tmpZipPath);
            request.AddFile("file", tmpZipPath);
            request.AddParameter("uuid", Guid.NewGuid().ToString());
            FileStream fs = new FileStream(tmpZipPath, FileMode.Open);
            request.AddParameter("totalSize", fs.Length);
            request.AddParameter("maxDl", uploadNumber);
            request.AddParameter("password", password);
            request.AddParameter("phones", phone);
            int chunckSize = 20971520; //20MB
            int totalChunks = (int)(fs.Length / chunckSize);
            if (fs.Length % chunckSize != 0)
            {
                totalChunks++;
            }
            for (int i = 0; i < totalChunks; i++)...
        }
    }
    catch (Exception ex)
```

Ce qui a été compliqué, ça a été de mettre en place un **« Chunk Upload », c'est-à-dire** permettre de télécharger de manière fiable des fichiers volumineux vers **LockSelf** en les divisant en une séquence de parties pouvant être téléchargées individuellement.

Pour cela, il a fallu que je me documente.

Voici les liens qui m'ont permis de resoudre cette tâche :

- ► **https://stackoverflow.com/questions/53123243/uploading-large-files-to-controller-in-chunks-using-httpclient-iformfile-always**
- ► **https://www.c-sharpcorner.com/article/upload-large-files-to-mvc-webapi-using-partitioning/**
- ► **https://learn.microsoft.com/en-us/rest/api/servicefabric/sfclient-api-uploadfilechunk**

Ici, la méthode **GenerateLink()** permettra de retourner un lien sécurisée dans le corps de l'email :

```
public async void GenerateLink()
{
    try {
        Inspector inspector = Globals.ThisAddIn.Application.ActiveInspector();
        MailItem mailItem = inspector.CurrentItem as MailItem;
        var user = LockSelf.Utils.Globals.LoggedInUser;

        List<File> files = Files.Select(vm => vm.File).ToList();
        Debug.WriteLine("UploadNumber" + UploadNumber.ToString());

        string s = string.Format("{0:yyyyMMdd}", DisplayDateStart);
        Debug.WriteLine("Date" +  s);

        Debug.WriteLine("PrefixPhone -> " + (string)PrefixPhone);

        var fullPhone = PrefixPhone + " " + Phone;
        Debug.WriteLine("fullPhone -> " + fullPhone);

        string urlApi = _authService.GetUrlApi(Email);

        var progress = new Progress<int>(x => CurrentProgress = x);

        ProgressLoading = true;
        JobSuccess = await Task.Run(() =>
        {
            var link = _transferService.UploadFileAsync(urlApi, user, UploadNumber, files, progress, s, PasswordTransfer, Phone);
```

Elle appelle la méthode **UploadFileAsync(…) qui retournera le lien sécurisée** dans le corps de l'email.

Etant dans une approche **MVVM,** nous devons utiliser **l'interface ICommand** pour fournir le comportement attendu au bouton **« Générer le lien »** dans la vue **TransferControlWpf.xaml**

```
GenLinkCommand = new RelayCommand(p =>  GenerateLink());
```

TransferControlWpf.xaml

```
<Button Height="40" Background="#2833b7" BorderThickness="0" Style="{StaticResource SomeButtonStyle}" x:Name="btnGen" Click="btn_GenLink" VerticalContentAlignment="Center" Margin="0 10">
    <Button.Resources>
        <Style TargetType="{x:Type Border}">
            <Setter Property="CornerRadius" Value="3"/>
        </Style>
    </Button.Resources>
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="95"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>
        <fa:ImageAwesome Grid.Column="0" Margin="10" Icon="Link" x:Name="LoadingStandart" HorizontalAlignment="Left" Width="14" Height="14" Foreground="White"/>
        <TextBlock Grid.Column="1" Text="Générer le lien" Margin="-10 7">
            <TextBlock.Style>
                <Style>
                    <Setter Property="TextBlock.FontFamily" Value="Century Gothic" />
                    <Setter Property="TextBlock.FontSize" Value="18" />
                    <Setter Property="TextBlock.Foreground" Value="White" />
                    <Setter Property="TextBlock.FontWeight" Value="Bold" />
                </Style>
            </TextBlock.Style>
        </TextBlock>
    </Grid>
</Button>
```

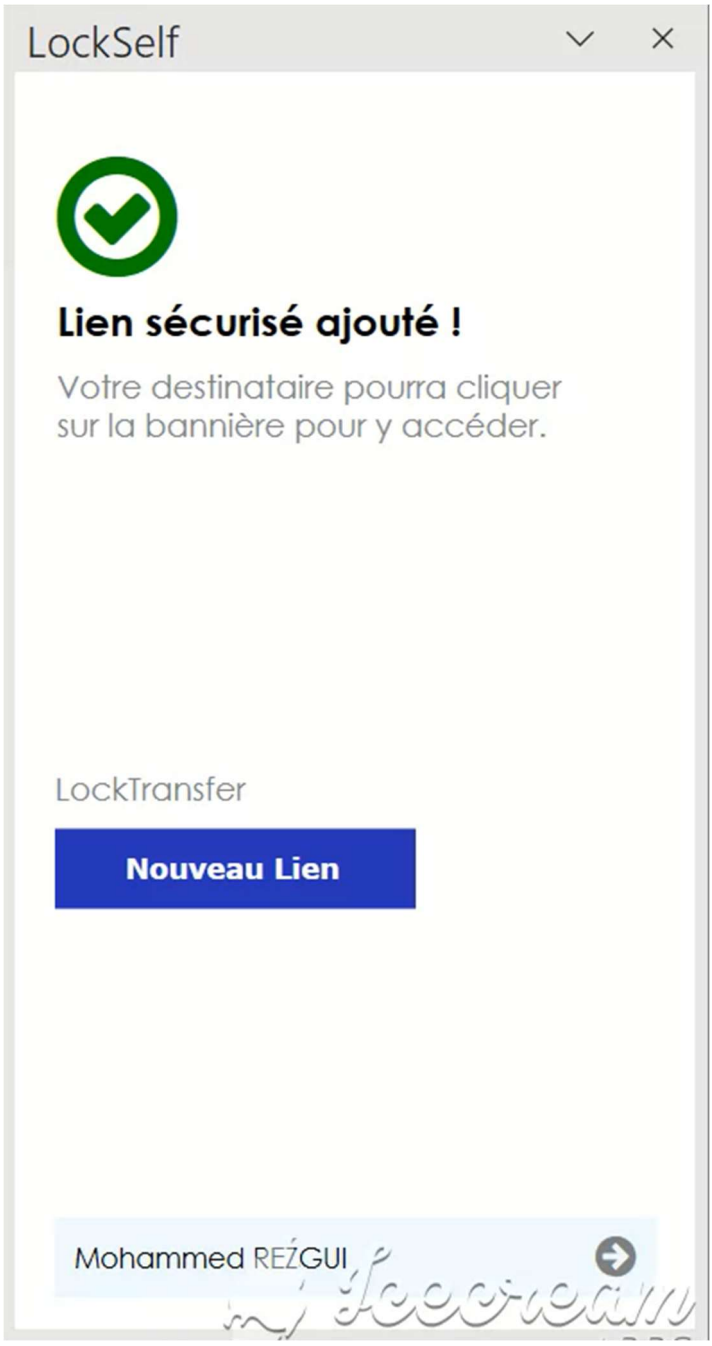
TransferControlWpf.xaml.cs

```
private void btn_GenLink(object sender, EventArgs e)
{
    TextBox txtPasswordTransfer = (TextBox)cp.ContentTemplate.FindName("txtPasswordTransfer", cp);
    TextBox txtPasswordTransfer2 = (TextBox)cp.ContentTemplate.FindName("txtPasswordTransfer2", cp);

    TransferViewModel vm = this.DataContext as TransferViewModel;
    if ((vm != null) && (vm.GenLinkCommand.CanExecute(null)))
    {
        cbProgress.IsChecked = vm.ProgressLoading;
        vm.GenLinkCommand.Execute(null);
        txtPasswordTransfer.Clear();

    }
    cbSuccess.IsChecked = vm.JobSuccess;
}
```

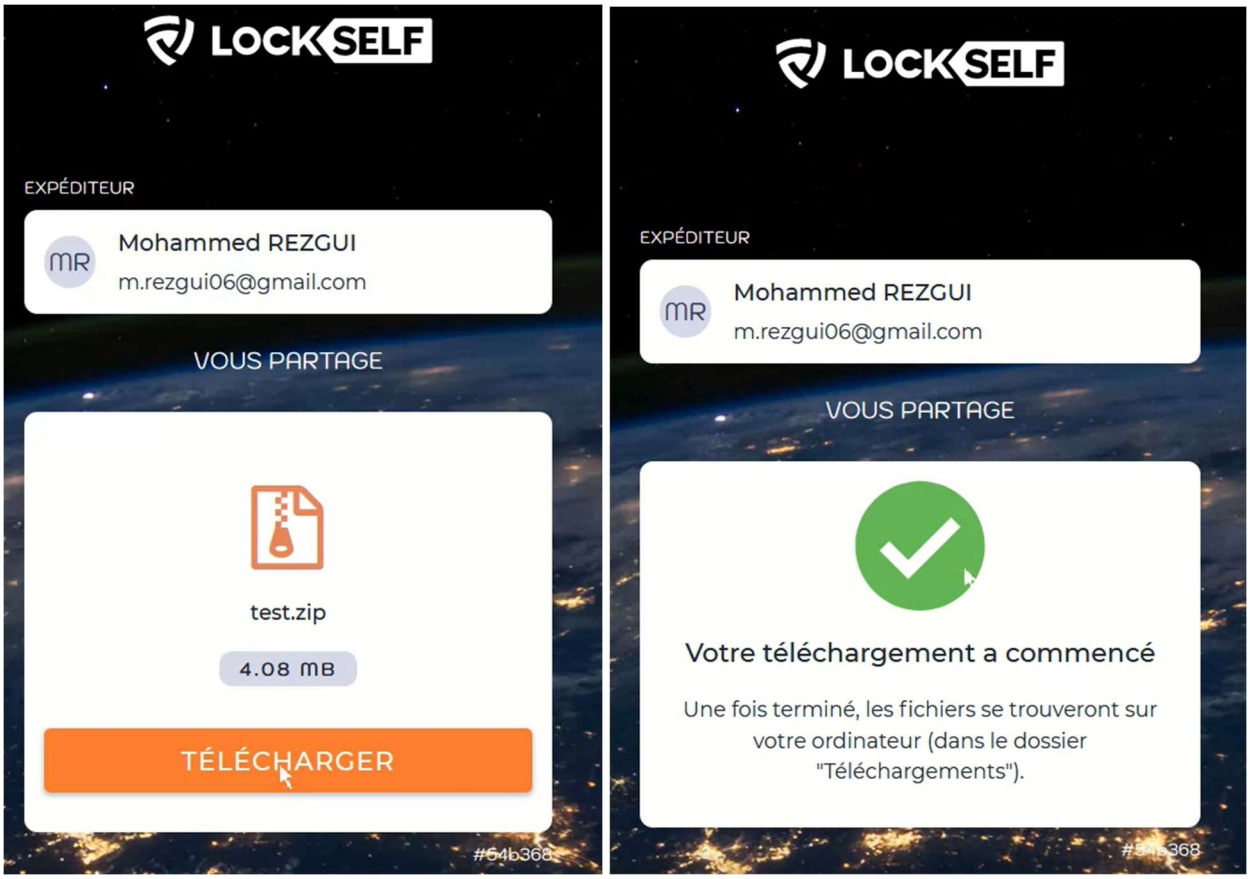
► Etape 3 :



Dans cette **étape**, l'expéditeur pourra :

► Retourner à l'étape 1 en appuyant sur le bouton **« Nouveau Lien »** pour **envoyer** un nouveau lien de téléchargement.
► Se déconnecter en appuyant sur le bouton du bas

► Etape 4 :



Dans cette **étape**, le destinataire pourra :

► Télécharger les pièces jointes en appuyant sur le bouton **« TELECHARGER »**