

Synthèse de stage 2022 ARCNET



Contexte de stage :

J'ai effectué mon stage de 2ème année du 27/06/2022 au 05/08/2022 au sein de l'entreprise ARCNET.NC.

ARCNET.NC est une société de services spécialisée dans le domaine de l'ingénierie informatique tel que le développement informatique, les bases de données, la création et le référencement de sites internet et les formations professionnelles. Elle dispose d'une solide expérience dans les technologies Microsoft et Oracle grâce à une équipe de spécialistes certifiés.

ARCNET est un TPE composée de 9 personnes actuellement :

Stagiaire : Luca BEROULE

Analyste Développeur : Ayu PEDEVILLE, Bastien BOUCHETIERE, Steven FONGUE, Jonathan CAILLART, Chloé VIR.

Chef d'équipe : Erwan FAURE

Chef/Cheffe d'entreprise : Davy GRIGRI et Sophie PEZERON

La SLN étant le plus gros client de ARCNET où ils assurent le développement et la maintenance applicative Usine du suivi de production.

//PLUS DE DETAILS A FOURNIR\\ (CLIENTS - ENTREPRISE - CAPITAL)

Mission de stage :

J'ai travaillé sur le site de l'UPRA EQUINE (www.esirecal.nc) où j'ai dû créer un module de gestion documentaire pour un cheval, à la demande du client.
(Ticket Azure DevOps)

Les technologies utilisées : **Développement BackEnd** :

C#, .NetCore, Telerik.UI.for.ASPNET.Core, EntityFramework

Les technologies utilisées : **Développement FrontEnd** :

ASP.Net Razor Pages (HTML, CSS, JS + C#), Telerik.Kendo.UI (Bibliothèque JS)

Les technologies utilisées : **Gestion de projet / Equipe / Environnement de travail**
Azure DevOps, Microsoft Teams, Microsoft Visual Studio 2019

Sommaire :

- *Partie 1 : Formation avec les différents outils utilisés*
- *Partie 2 : La création du module*
- *Partie 3 : Aspect final du module + Extras*
- *Partie 4 : Déploiement du module*
- *Partie 5 : Conclusion personnelle*

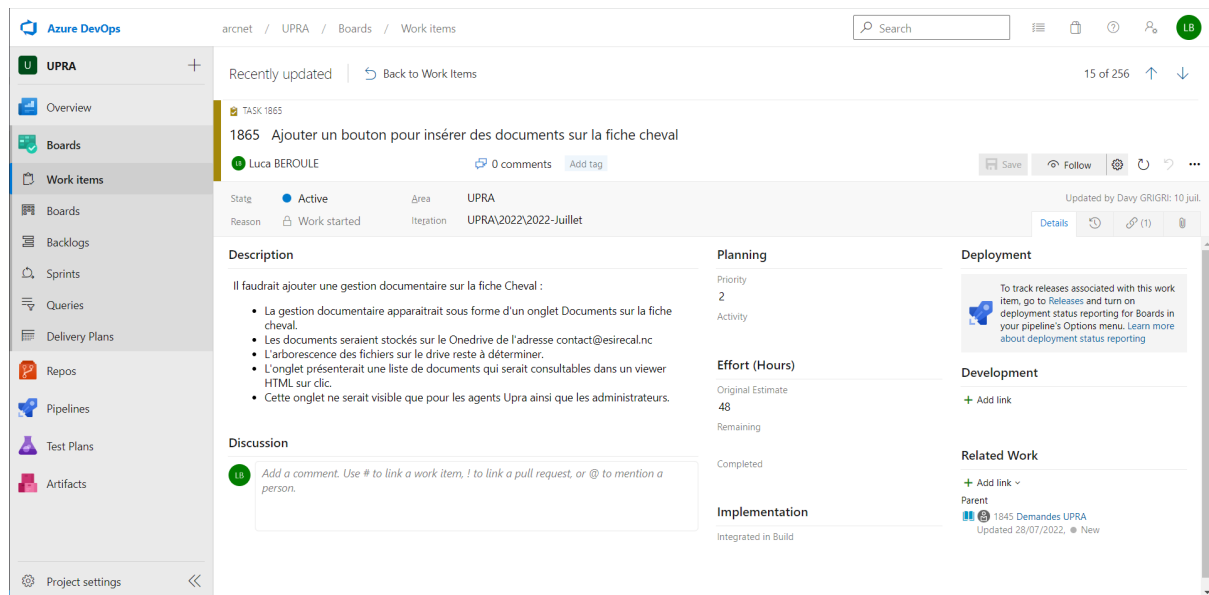
Partie 1 : Formation avec les différents outils utilisés

Durant la première semaine, j'ai pu découvrir les différents outils avec lesquels travaillent l'entreprise ARCNET. J'ai pu notamment être initié à l'utilisation de Azure DevOps.

Azure DevOps est une plate-forme de services de Microsoft qui fusionne la gestion de projet et les fonctionnalités de GitHub et plus encore. Elle est composée de 5 parties :

- Azure Pipelines (pour créer, tester et déployer en continu sur n'importe quelle plateforme et cloud avec n'importe quel langage),
- Azure Boards (pour planifier et suivre les tâches des équipes puis échanger à leur sujet),
- Azure Artifacts (pour créer, héberger et partager des packages),
- Azure Repos (pour accéder à un nombre illimité de dépôts Git privés hébergés dans le cloud),
- Azure Test Plans (pour tester et livrer, avec un kit de ressources pour les tests manuels et exploratoires).

Voici à quoi ressemble Azure DevOps et la tâche qui m'a été confiée



Ensuite, mon tuteur de stage m'a prêté un compte UDEMY avec différentes formations afin de monter en compétences et d'acquérir les bases surtout en C# et en MVC .NetCore car j'allais les voir constamment durant toute la durée du stage.

Partie 2 : La création du module

Après avoir suivi les deux formations, j'ai pu donc commencer à travailler sur la version de Développement du site de l'UPRA Equine.

Je me suis inspiré du code déjà intégré dans certaines fonctionnalités afin de parvenir à créer le module.

Mais avant, il fallait que je comprenne comment le site fonctionnait. Et c'est là que ma formation en MVC (Model-View-Controller) prend tout son sens. Bien que le site ne soit pas vraiment du MVC mais plutôt du MVVM (Model-View-View-Model) il n'y a pas grand chose qui change à vrai dire.

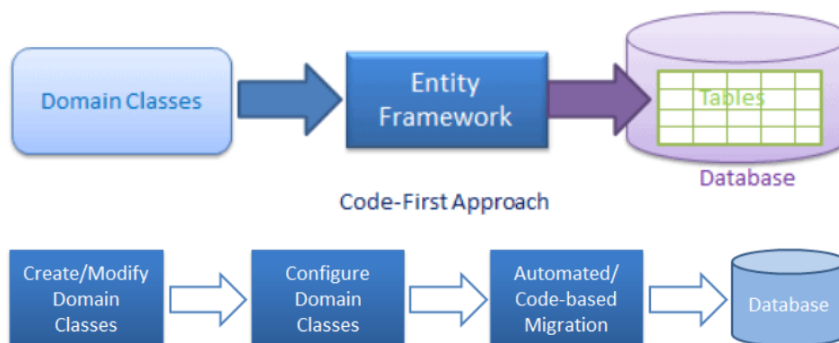
MVC veut dire : Model, c'est donc les données que l'on va utiliser, représenté avec une classe. Exemple ici avec DocumentCheval.cs : [Key] pour DocumentId indique la clé primaire de la table qui sera créé par EntityFramework.

```

1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel.DataAnnotations;
4
5 namespace Upra.Entities.Models
6 {
7     15 références | Davy GRIGRI, il y a 28 jours | 1 auteur, 3 modifications | 1 modification entrante
8     public partial class DocumentCheval
9     {
10         [Key]
11         5 références | Davy GRIGRI, il y a 30 jours | 1 auteur, 1 modification
12         public int DocumentId { get; set; }
13         3 références | Davy GRIGRI, il y a 28 jours | 1 auteur, 1 modification
14         public int IdTypeDocument { get; set; }
15         4 références | 0 modifications | 0 auteurs, 0 modifications
16         public string Ncheval { get; set; }
17         2 références | 0 modifications | 0 auteurs, 0 modifications
18         public DateTime Ddemande { get; set; }
19         2 références | 0 modifications | 0 auteurs, 0 modifications
20         public DateTime? Dreception { get; set; }
21         2 références | 0 modifications | 0 auteurs, 0 modifications
22         public DateTime? Dcreation { get; set; }
23         3 références | 0 modifications | 0 auteurs, 0 modifications
24         public DateTime? DmiseAJour { get; set; }
25         1 référence | 0 modifications | 0 auteurs, 0 modifications
26         public int? Copérateur { get; set; }
27         4 références | Davy GRIGRI, il y a 30 jours | 1 auteur, 1 modification
28         public string Nom { get; set; }
29         6 références | Davy GRIGRI, il y a 30 jours | 1 auteur, 1 modification
30         public string Url { get; set; }
31         1 référence | 0 modifications | 0 auteurs, 0 modifications | 1 modification entrante
32         public Cheval NchevalNavigation { get; set; }
33         4 références | Davy GRIGRI, il y a 28 jours | 1 auteur, 1 modification
34         public TypeDocument TypeDocument { get; set; }
35     }
36 }

```

Dans cet exemple, imaginons que nous avons créé une base de données qui ne contient pas de tables pour l'instant. Avec EntityFramework, on applique la méthode CodeFirst. C'est-à-dire que l'on crée la classe et ensuite Entity crée la table avec les données insérées dans cette classe.



La configuration se fait dans le fichier appsettings.json, Il faut bien renseigner le champ "DefaultConnection": Exemple :

```
"DefaultConnection": "Server=.;Database=InAndOut;Trusted_Connection=True;MultipleActiveResultSets=True"
```

Après il faut ajouter une migration qui elle va donc créer votre classe en table :

Dans Affichage - Autres fenêtres - Console du Gestionnaire de package

Puis entrer add-migration et le nom de votre migration et le tour est joué.

Ensuite, on a les Controllers, qui pour résumé, sont le BackEnd de notre application. C'est eux qui vont gérer nos fonctionnalités, or dans mon contexte de stage, l'ajout, la suppression, modification et l'affichage des données. Les Controllers sont assez différents dans le contexte de mon stage. En effet, ils sont séparés en deux parties. Les Controllers API et Non-API :

Les Controllers API gèrent la partie requête/procédure.

Les Controllers Non-API renvoie une view/partial view avec un objet

Exemple API pour ajouter un document dans une fiche cheval :

```
public async Task<IActionResult> AddChevalDocument( InsertDocumentViewModel model, IFormFile File )
{
    try
    {
        if ( ModelState.IsValid )
        {
            if ( !IFormFile.Equals( File, null ) )
            {
                // Chemin du dossier a rajouter.
                string newFolderName = @"wwwroot\Documents\Chevaux\" + model.NCheval + @"\";

                // Check si le dossier n'existe pas, si oui, crée le dossier.
                if ( !Directory.Exists(newFolderName) )
                {
                    Directory.CreateDirectory( newFolderName );
                }

                model.Url = @"Documents\Chevaux\" + model.NCheval + @"\\" + File.FileName;

                // Check si le fichier est en pdf, si oui, on ajoute le fichier.
                string ext = Path.GetExtension( model.File.FileName );
                if ( ext.ToLower() != ".pdf" )
                {
                    return BadRequest( "Veuillez saisir un fichier pdf." );
                }
                else
                {
                    await _documentChevalRepository.AddFile( File, model.Url );
                }
            }

            await _documentChevalRepository.CreateDocumentAsync( model );

            // On retourne un code 204.
            return Json( new { success = true, responseText = "Le document a bien été ajouté." } );
        }

        // Conversion des erreurs du ModelState en JSON.
        string errors = JsonConvert.SerializeObject( ModelState.Values
            .SelectMany( state => state.Errors )
            .Select( error => error.ErrorMessage ) );

        return BadRequest( errors );
    }
    catch ( Exception e )
    {
        if ( e.InnerException?.Message == "Remonté" )
        {
            return BadRequest( e.Message );
        }
        else { return StatusCode( 500, "Une erreur est survenue lors de la tentative d'ajout d'un document pour le cheval" ); }
    }
}
```

Méthode AddFile : Permet d'ajouter un fichier.

Méthode CreateDocumentAsync : Permet d'ajouter un document dans la fiche cheval.

Exemple Non-API(De la même fonctionnalité) :

```
public async Task<IActionResult> _FormInsertDocument( string NCheval )
{
    var newDocument = new InsertDocumentViewModel();
    newDocument.NCheval = NCheval;

    return PartialView( "_FormInsertDocument", newDocument );
}
```

Autre chose indispensable dans le contexte de mon stage : **Les Repositories.**
C'est eux qui sont responsables en grande partie des fonctionnalités du site.
Les Repositories sont un ensemble de méthodes que le Controller va pouvoir appeler et exécuter.

La Méthode AddFile :

```
public async Task AddFile(IFormFile File, string Url)
{
    await ContraintesFileName(Url);

    string destFile = _env.WebRootPath + Url;
    using (var stream = new FileStream(destFile, FileMode.Create))
    {
        File.CopyTo(stream); // je copie le fichier dans le dossier wwwroot\images\News de l'application
    }
}
```

La Méthode CreateDocumentAsync :

```
public async Task CreateDocumentAsync(InsertDocumentViewModel document)
{
    if (document == null)
        throw new ArgumentNullException(nameof(document));

    var model = mapper.Map<DocumentCheval>(document);
    model.Dcreation = DateTime.Now;
    model.Ddemande = DateTime.Now;
    model.Dreception = DateTime.Now;

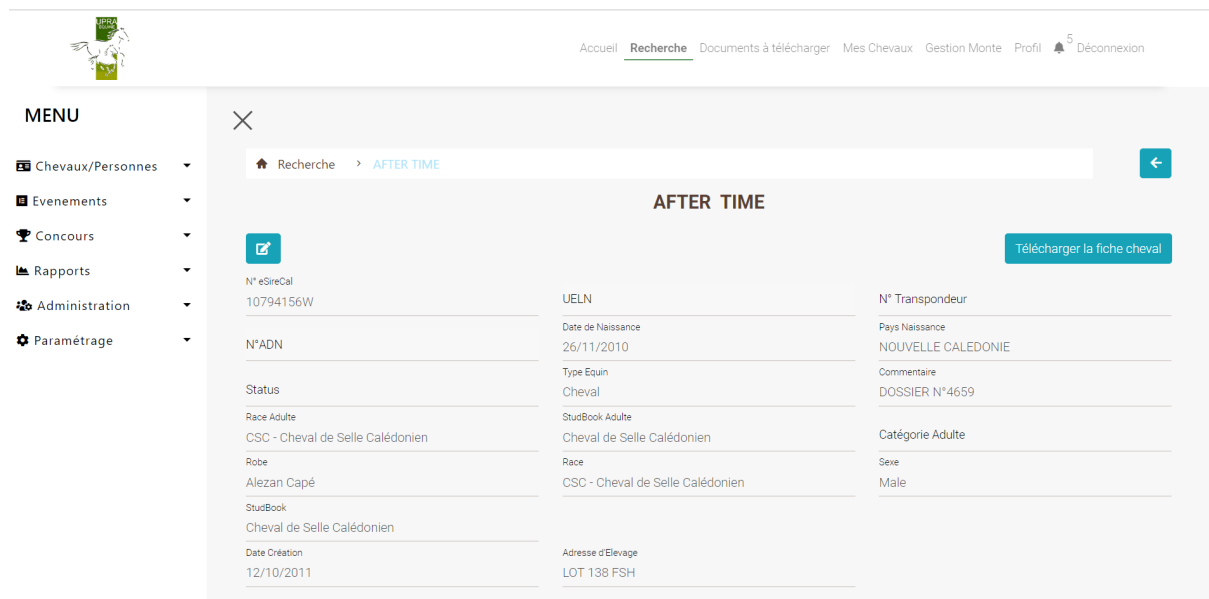
    await _context.DocumentCheval.AddAsync(model);

    await SaveChangesAsync();
}
```

La Méthode SaveChangesAsync (Grâce à EntityFramework) :

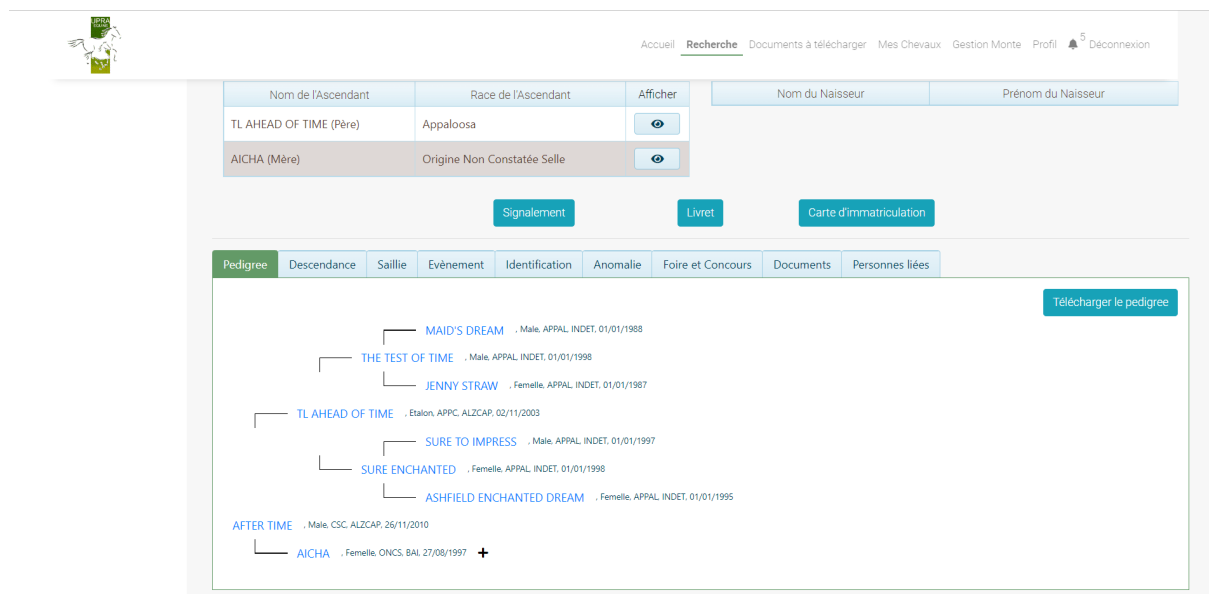
```
/// <summary>
/// Permet d'enregistrer les modifications effectuées
/// dans le contexte BDD.
/// </summary>
6 références
public async Task SaveChangesAsync() =>
    await _context.SaveChangesAsync();
```

Et pour finir, les Views qui correspondent aux vues, ce qui est affiché, l'interface utilisateur. C'est notre FrontEnd. Pour bien être dans le contexte voici une fiche cheval (en tant qu'administrateur ou agent du site)



The screenshot shows the 'AFTER TIME' horse profile page. The page has a header with navigation links: Accueil, Recherche, Documents à télécharger, Mes Chevaux, Gestion Monte, Profil, and Déconnexion. A left sidebar contains a 'MENU' with options: Chevaux/Personnes, Evenements, Concours, Rapports, Administration, and Paramétrage. The main content area is titled 'AFTER TIME' and includes a 'Télécharger la fiche cheval' button. Below this, there are three columns of information:

N° eSireCal 10794156W	UELN	N° Transpondeur
N°ADN	Date de Naissance 26/11/2010	Pays Naissance NOUVELLE CALEDONIE
Status	Type Equin Cheval	Commentaire DOSSIER N°4659
Race Adulte CSC - Cheval de Selle Calédonien	StudBook Adulte Cheval de Selle Calédonien	Catégorie Adulte
Robe Alezan Capé	Race CSC - Cheval de Selle Calédonien	Sexe Male
StudBook Cheval de Selle Calédonien		
Date Création 12/10/2011	Adresse d'Elevage LOT 138 FSH	



The screenshot shows the 'AFTER TIME' horse profile page with the pedigree chart and parentage information. The page has a header with navigation links: Accueil, Recherche, Documents à télécharger, Mes Chevaux, Gestion Monte, Profil, and Déconnexion. A left sidebar contains a 'MENU' with options: Chevaux/Personnes, Evenements, Concours, Rapports, Administration, and Paramétrage. The main content area is titled 'AFTER TIME' and includes a 'Télécharger la fiche cheval' button. Below this, there are three columns of information:

Nom de l'Ascendant	Race de l'Ascendant	Afficher	Nom du Naisseur	Prénom du Naisseur
TL AHEAD OF TIME (Père)	Appaloosa			
AICHA (Mère)	Origine Non Constatée Selle			

Below the parentage table, there are buttons: Signalement, Livret, and Carte d'immatriculation. Below these buttons, there is a tabbed interface with tabs: Pedigree, Descendance, Saillie, Evènement, Identification, Anomalie, Foire et Concours, Documents, and Personnes liées. The 'Pedigree' tab is selected, showing a pedigree chart for 'AFTER TIME'.

```

graph TD
    A["AFTER TIME  
Male, CSC, ALZCAP, 26/11/2010"] --- B["AICHA  
Femelle, ONCS, BAL, 27/08/1997"]
    A --- C["TL AHEAD OF TIME  
Etalon, APPC, ALZCAP, 02/11/2003"]
    C --- D["THE TEST OF TIME  
Male, APPAL, INDET, 01/01/1998"]
    C --- E["JENNY STRAW  
Femelle, APPAL, INDET, 01/01/1987"]
    D --- F["SURE TO IMPRESS  
Male, APPAL, INDET, 01/01/1997"]
    D --- G["SURE ENCHANTED  
Femelle, APPAL, INDET, 01/01/1998"]
    G --- H["ASHFIELD ENCHANTED DREAM  
Femelle, APPAL, INDET, 01/01/1995"]
  
```

Below the pedigree chart, there is a 'Télécharger le pedigree' button.

L'objectif de la mission était d'avoir un onglet en plus (Documents) sur cette fiche et aussi pour tous les autres chevaux.

Toute cette vue (deux précédentes captures d'écrans) sont gérés par le fichier _InfosCheval.cshtml (cshtml étant l'extension pour les **Pages Razor** intégré à ASP.Net Core qui permet d'intégrer du C# au pages HTML CSS JS basique)

Partie 3 : Aspect Final du module + Extras

Pedigree	Descendance	Saillie	Evènement	Identification	Anomalie	Foire et Concours	Documents	Personnes liées
Document :								Insérer un Document +
Titre du document				Type Document			Actions	
Bulletin N°1				Bulletin				
Livret du Cheval				Livret				
Tests du Cheval				Tests				
« « 1 » » 10 articles par page								Afficher les items 1 - 3 de 3

Bouton Insérer un Document : (Fichier, nom et type de document requis pour créer un document.)

Ajouter un Document

Charger un fichier

Aucun fichier n'a été sélectionné

Titre du document

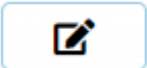
Selectionner le type de document

CRÉER

ANNULER

Bouton Voir :  (Affiche le pdf contenu dans le document.)



Bouton Modifier :  (Ouvre la fenêtre de modification, avec les données du document sélectionner, si un autre pdf est enregistré, l'ancien est écrasé.)



MODIFIER UN DOCUMENT

Aucun fichier n'a été sélectionné

Titre du document *
Bulletin N°1

Bouton Supprimer :  (Ouvre une fenêtre de confirmation de suppression.)



SUPPRIMER UN DOCUMENT

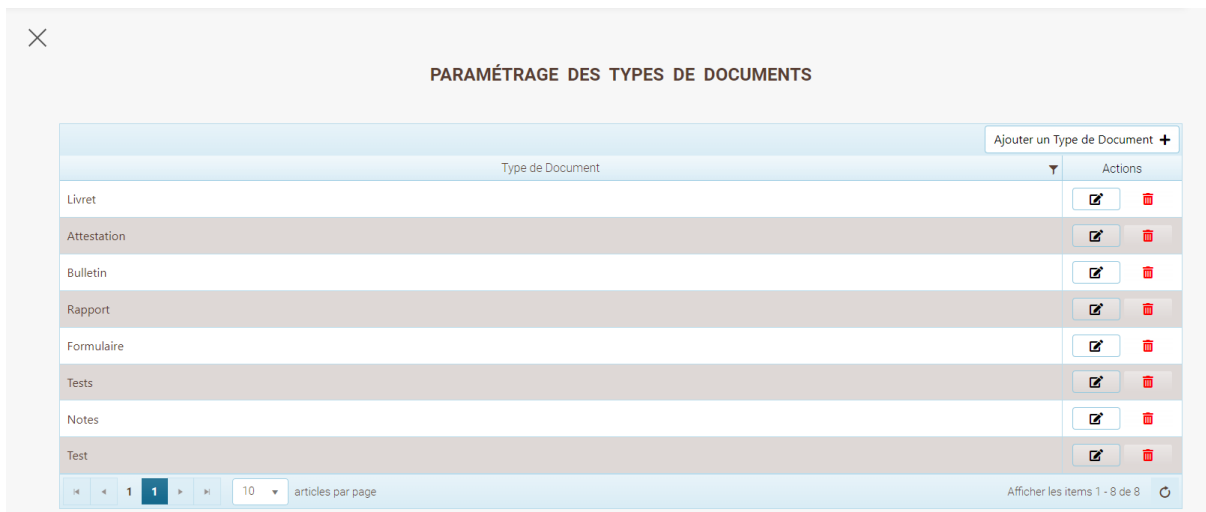
Êtes-vous sûr(e) de vouloir supprimer le document **Bulletin N°1** ?

La suppression est définitive.

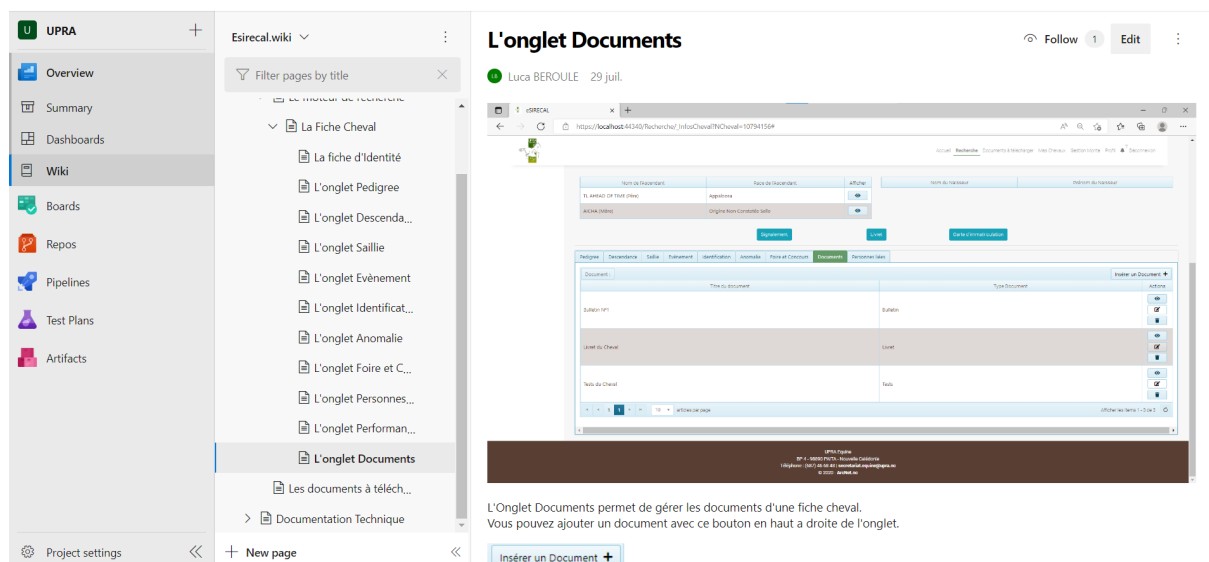
On m'a ensuite demandé de rajouter une gestion des types de documents similaire à la gestion des documents d'une fiche cheval mais avec des données différentes.

Paramétrage - Chevaux - Types Documents





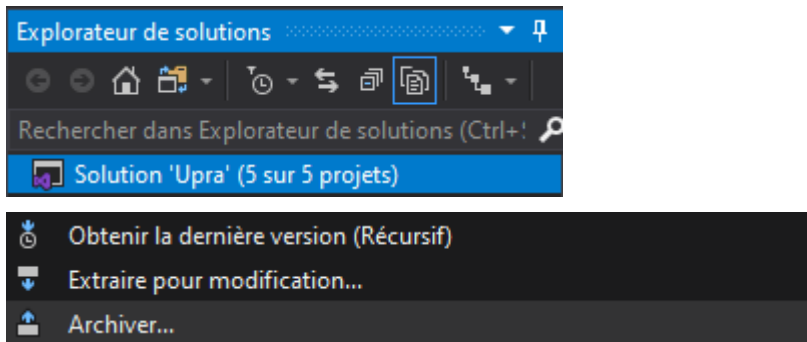
J'ai aussi rédigé une documentation utilisateur afin de pouvoir accompagner l'utilisateur dans l'utilisation du module.



Partie 4 : Déploiement du module

Le déploiement du module s'effectue grâce à Azure DevOps.

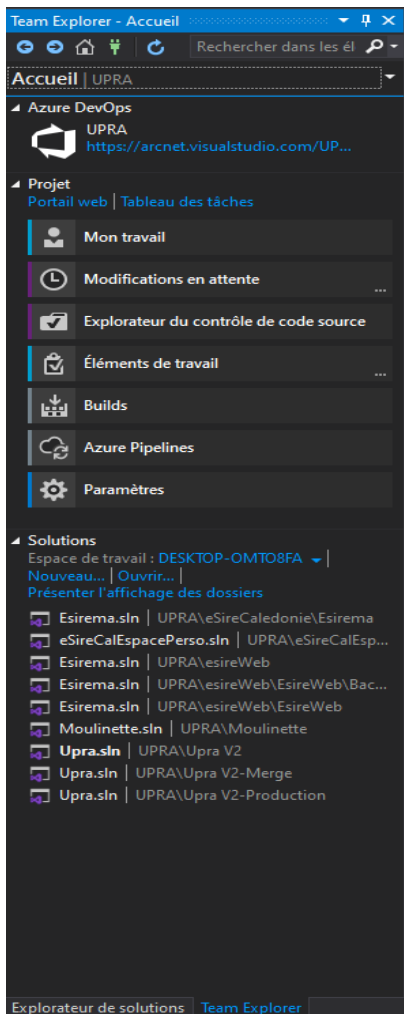
Récupération de la dernière version, Archivage (Résolution de conflit entre version locale et version serveur) sur Microsoft Visual Studio ancré dans Microsoft DevOps.



Dans l'explorateur de solutions à la racine du projet - click droit -

Récupérer la dernière version (mettre à jour la version locale) ou

Dans Team Explorer - Modifications en attente pour mettre en ligne les modifs, assigner une tâche du work items DevOps (Tickets).



Partie 5 : Conclusion personnelle

Ce stage a été une véritable découverte et une expérience très enrichissante pour moi. Les membres de l'équipe ont tous été très sympas avec moi. Découvrir une nouvelle facette du développement WEB était un véritable plaisir malgré mes difficultés.