2025

API et ASP.NET Core Web API

E-GESCOM
MATEO SAGE

PRO A PRO |1419 Av. d'Italie, 82000 Montauban, France

Table des matières

Introduction

Contexte du stage

Durant mon stage, j'ai été chargé d'établir un API interne du site E-Gescom de l'entreprise "Pro à Pro", en migrant l'intégralité (commencer), existante du backend d'une architecture PHP vers une API Web ASP.NET RESTful en C#, en adoptant une architecture MVC pour les employés internes.

Présentation de l'entreprise

Pro à Pro est un acteur majeur de la **distribution alimentaire pour les professionnels de la Restauration Hors Domicile (RHD)** en France. Filiale du groupe METRO, l'entreprise se positionne comme un grossiste alimentaire qui accompagne et livre quotidiennement des milliers de clients variés, incluant des établissements de santé, scolaires, d'entreprise, des maisons de retraite, ainsi que des restaurants traditionnels.

Elle propose une large gamme de produits couvrant l'épicerie, le frais, le surgelé, les boissons et les produits non-alimentaires, s'engageant à livrer ses clients sur l'ensemble du territoire français, souvent en 24h. Pro à Pro met en avant une dimension humaine et des valeurs fortes, favorisant l'intégration et l'épanouissement de ses collaborateurs.

Durant mon stage, j'ai intégré la **Direction des Systèmes d'Information (DSI)**, spécifiquement au sein du service de **Développement Web**. Ce service, crucial pour l'évolution et la performance des outils internes de l'entreprise, était placé sous la responsabilité de **Madame Yurelis**, qui m'a supervisé durant ma période ou j'étais présent.

Objectifs du projet

L'objectif principal de ce projet de stage était de **concevoir, développer et livrer une API Web RESTful en ASP.NET Core pour le site interne "E-Gescom" de Pro à Pro**. Cette nouvelle API visait à moderniser et à optimiser la gestion de l'ensemble des processus liés au secteur **agro-alimentaire** de l'entreprise, en offrant une interface de communication performante et sécurisée pour les applications internes utilisées par les employés.

Migrer l'API existante d'une technologie PHP vers un environnement C#/.NET, garantissant ainsi une meilleure maintenabilité, évolutivité et performance.

Structurer l'API selon les principes de l'architecture MVC, en développant des Modèles, des Contrôleurs et des Repositories robustes.

Sécurité (via authentification par jeton Bearer) pour permettre aux différents employés internes de Pro à Pro puissent interagir avec les données.

I. Architecture et Technologies Utilisées

Backend : API Web ASP.NET

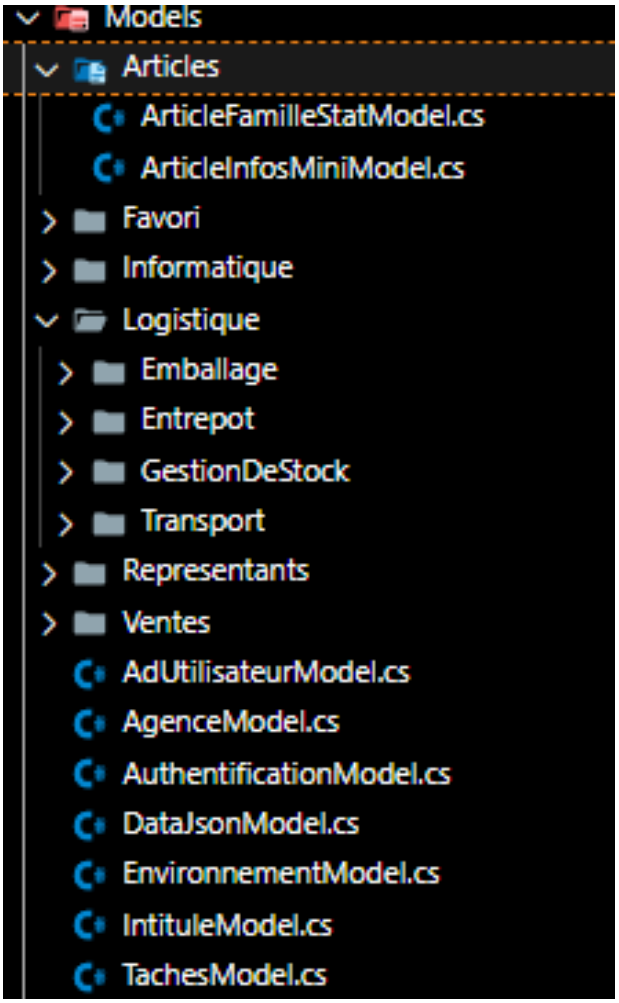
L'API a été développée en **ASP.NET Core**, un framework moderne et performant de Microsoft, choisi pour sa robustesse, sa flexibilité et sa capacité à construire des services web RESTful efficaces. Ce choix a permis de bénéficier de l'écosystème .NET et de ses nombreux avantages en termes de développement et de maintenance par rapport à l'architecture PHP préexistante.

Le projet a été organisé selon les principes de l'architecture **MVC (Model-View-Controller).**

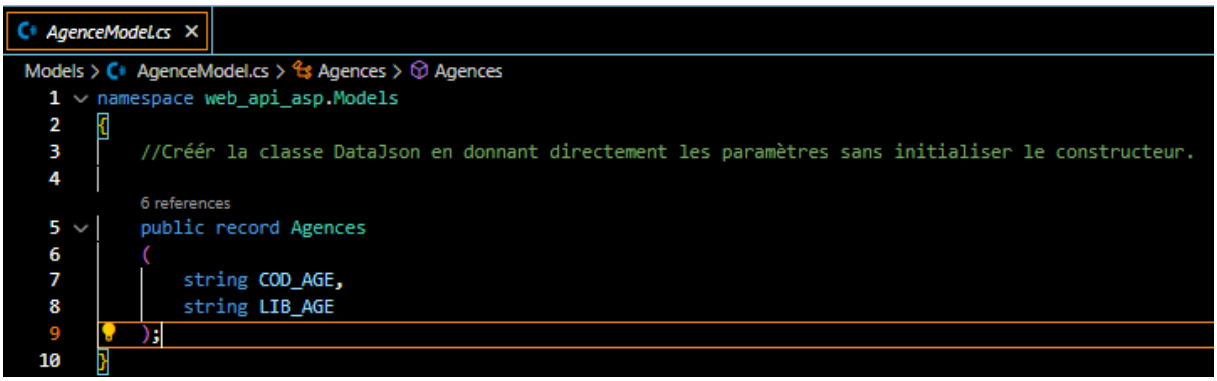
Organisation du projet

Modèles (Models) :

> Représentent la structure de données de l'application et les entités métier. Ils reflètent la base de données. Ils peuvent contenir des validations simples.

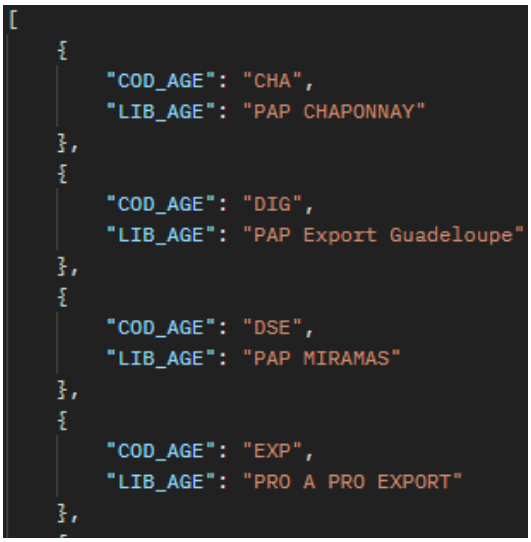


Exemple de modèles (AgenceModel.cs, AuthenticationModel.cs, etc.)

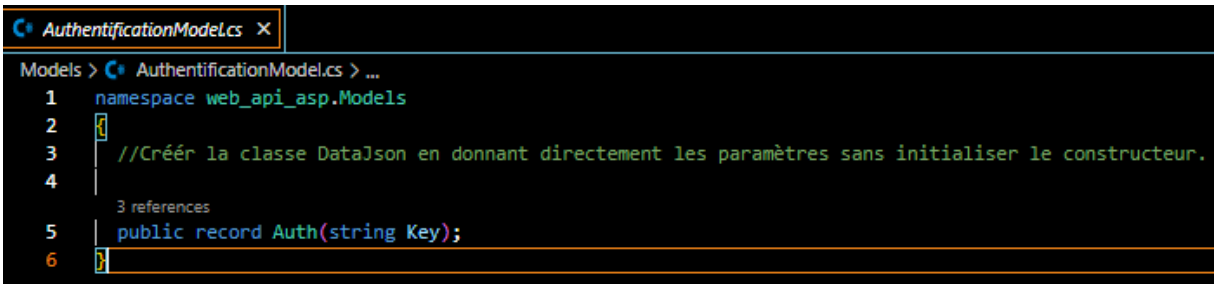


```cs
namespace web_api_asp.Models
{
    //Créér la classe DataJson en donnant directement les paramètres sans initialiser le constructeur.

    6 references
    public record Agences
    (
        string COD_AGE,
        string LIB_AGE
    );
}
```

Structure attendue :



```json
[
    {
        "COD_AGE": "CHA",
        "LIB_AGE": "PAP CHAPONNAY"
    },
    {
        "COD_AGE": "DIG",
        "LIB_AGE": "PAP Export Guadeloupe"
    },
    {
        "COD_AGE": "DSE",
        "LIB_AGE": "PAP MIRAMAS"
    },
    {
        "COD_AGE": "EXP",
        "LIB_AGE": "PRO A PRO EXPORT"
    },
```



```cs
namespace web_api_asp.Models
{
    //Créér la classe DataJson en donnant directement les paramètres sans initialiser le constructeur.

    3 references
    public record Auth(string Key);
}
```

Résultat attendue :

```
"Key": "[redacted]"
```

C’est le mot de passe qui permet d’accéder à l’environnement WS (AS400).

DataJsonModel.cs 5

Models > DataJsonModel.cs > Agence

```cs
using System.Text.Json.Serialization;

namespace web_api_asp.Models
{

    //Créér la classe DataJson et Agence en donnant directement les paramètres sans initialiser le constructeur.

    /*
    public record DataJson(string Name, string Description, List<Agence> Agences);
    public record Agence(string COD_AGE, string LIB_AGE);
    */

    4 references
    public class DataJson
    {
        [JsonPropertyName("name")]
        1 reference
        public string Name { get; set; }

        [JsonPropertyName("description")]
        1 reference
        public string Description { get; set; }

        [JsonPropertyName("agences")]
        1 reference
        public List<Agence> Agences { get; set; }
    }

    26 references
    public class Agence
    {

        /*
        Permet de mettre des propriétés individuelles en utilisant cet attribut =>  [JsonPropertyName("")]
        Dans notre requête les paramètres seront en majuscule
        */

        25 references
        public string COD_AGE { get; set; }
        25 references
        public string LIB_AGE { get; set; }

    }
}
```

WS / /datajson / **/datajson**

GET {{WS_URL}} /datajson

Params Headers Body

Query Params

Key	Value
Key	Value

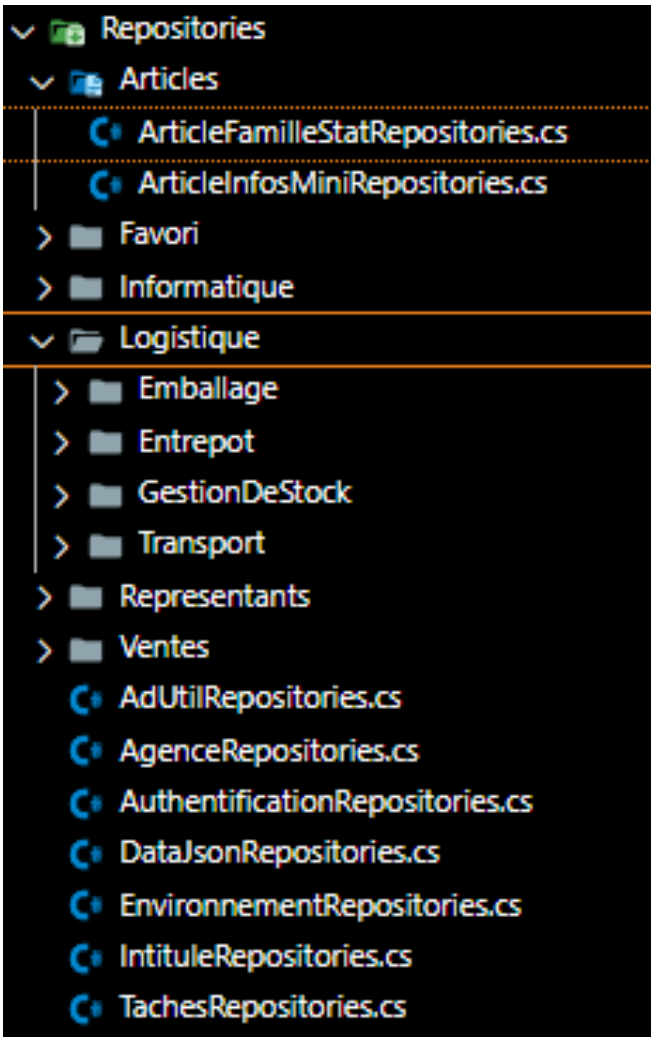
Body Headers (4)

{ } JSON Preview


```json
{
    "name": "datagescom",
    "description": "Liste de données en dur pour les applications",
    "agences": [
        {
            "COD_AGE": "MTB",
            "LIB_AGE": "MONTAUBAN"
        },
        {
            "COD_AGE": "PER",
            "LIB_AGE": "PERPIGNAN"
        },
        {
            "COD_AGE": "DSE",
            "LIB_AGE": "MIRAMAS"
        },
        {
            "COD_AGE": "DOL",
            "LIB_AGE": "DOLE"
        },
        {
            "COD_AGE": "BZH",
            "LIB_AGE": "SAINT GILLES"
        },
        {
            "COD_AGE": "HAR",
            "LIB_AGE": "CHALETTE"
        },
        {
            "COD_AGE": "DID",
            "LIB_AGE": "RUNGIS"
        },
```

Repositories (Couche d'accès aux données) :

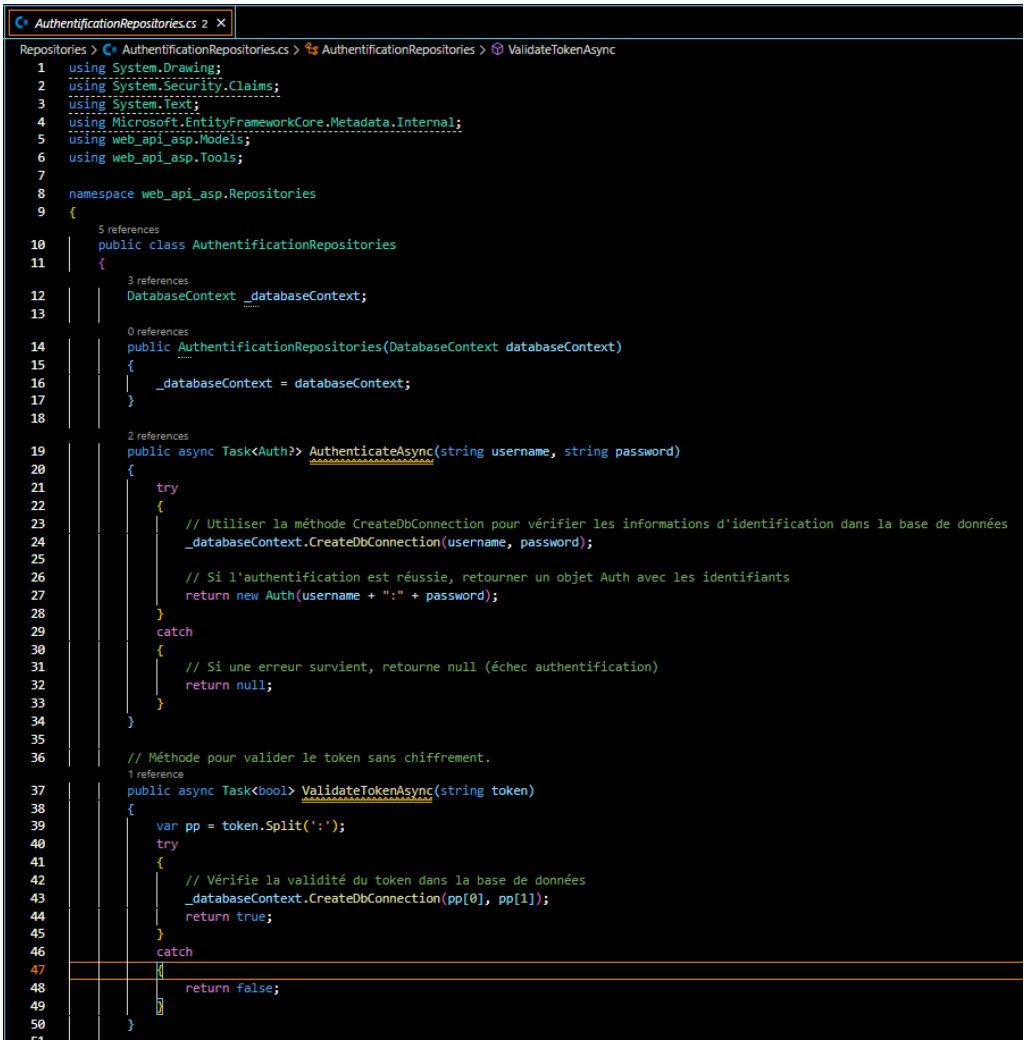
Encapsulent la logique d'accès aux données, servant d'intermédiaire entre les Contrôleurs et la source de données (base de données, services externes, etc.). Ils définissent les méthodes pour interagir avec les entités (CRUD : Create, Read, Update, Delete). L'utilisation de repositories permet d'isoler la logique d'accès aux données.



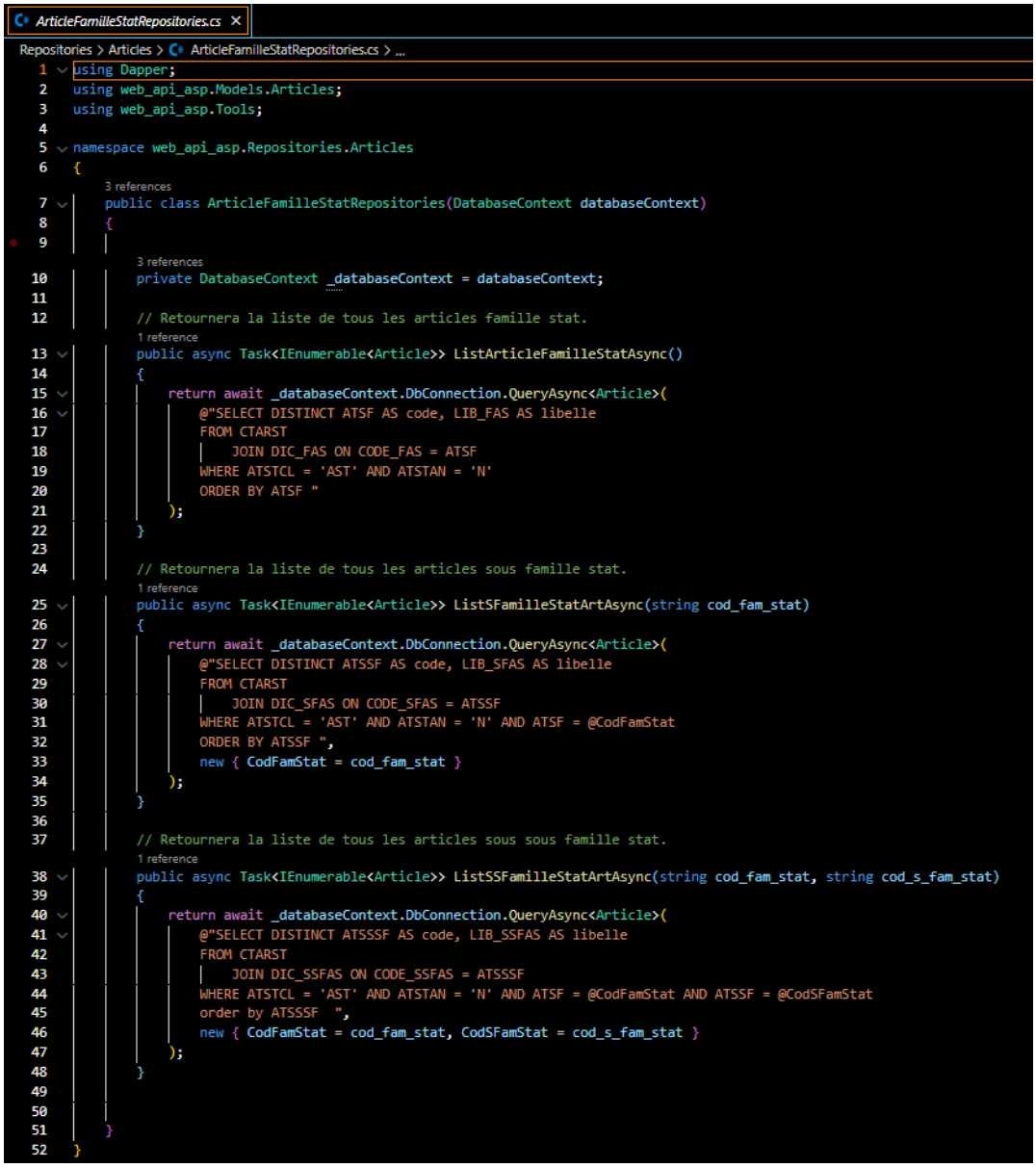
Exemple de repositories (AgenceRepositories.cs, AuthenticationRepositories.cs, etc.)

Contient les méthodes pour récupérer des informations sur les agences.

Gère la logique d'authentification des utilisateurs



Permet d'interagir avec les données des articles

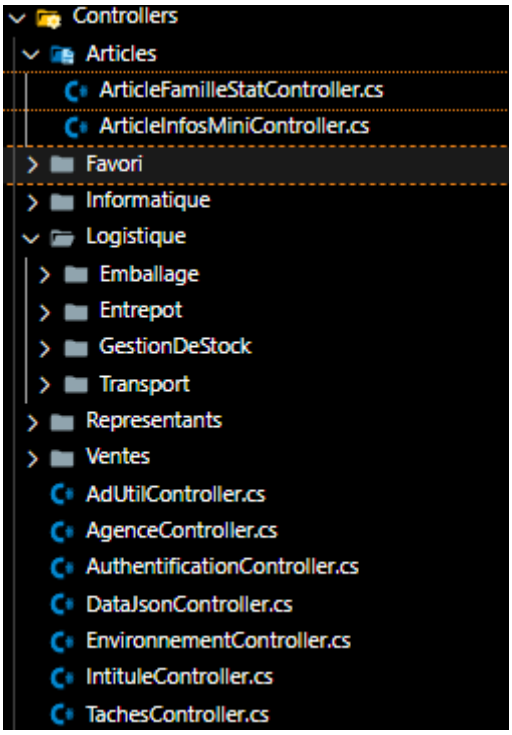


```
ArticleFamilleStatRepositories.cs
Repositories > Articles > ArticleFamilleStatRepositories.cs > ...
using Dapper;
using web_api_asp.Models.Articles;
using web_api_asp.Tools;

namespace web_api_asp.Repositories.Articles
{
    3 references
    public class ArticleFamilleStatRepositories(DatabaseContext databaseContext)
    {

        3 references
        private DatabaseContext _databaseContext = databaseContext;

        // Retournera la liste de tous les articles famille stat.
        1 reference
        public async Task<IEnumerable<Article>> ListArticleFamilleStatAsync()
        {
            return await _databaseContext.DbConnection.QueryAsync<Article>(
                @"SELECT DISTINCT ATSF AS code, LIB_FAS AS libelle
                FROM CTARST
                    JOIN DIC_FAS ON CODE_FAS = ATSF
                WHERE ATSTCL = 'AST' AND ATSTAN = 'N'
                ORDER BY ATSF "
            );
        }

        // Retournera la liste de tous les articles sous famille stat.
        1 reference
        public async Task<IEnumerable<Article>> ListSFamilleStatArtAsync(string cod_fam_stat)
        {
            return await _databaseContext.DbConnection.QueryAsync<Article>(
                @"SELECT DISTINCT ATSSF AS code, LIB_SFAS AS libelle
                FROM CTARST
                    JOIN DIC_SFAS ON CODE_SFAS = ATSSF
                WHERE ATSTCL = 'AST' AND ATSTAN = 'N' AND ATSF = @CodFamStat
                ORDER BY ATSSF ",
                new { CodFamStat = cod_fam_stat }
            );
        }

        // Retournera la liste de tous les articles sous sous famille stat.
        1 reference
        public async Task<IEnumerable<Article>> ListSSFamilleStatArtAsync(string cod_fam_stat, string cod_s_fam_stat)
        {
            return await _databaseContext.DbConnection.QueryAsync<Article>(
                @"SELECT DISTINCT ATSSSF AS code, LIB_SSFAS AS libelle
                FROM CTARST
                    JOIN DIC_SSFAS ON CODE_SSFAS = ATSSSF
                WHERE ATSTCL = 'AST' AND ATSTAN = 'N' AND ATSF = @CodFamStat AND ATSSF = @CodSFamStat
                order by ATSSSF  ",
                new { CodFamStat = cod_fam_stat, CodSFamStat = cod_s_fam_stat }
            );
        }


    }
}
```

Contrôleurs (Controllers) :

Reçoit les requêtes HTTP des clients, interagissent avec les Repositories pour récupérer ou manipuler les données, et renvoient les réponses HTTP (généralement au format JSON).

Ils contiennent la logique de haut niveau de l'API.



Exemple de controllers (AgencesController.cs, AuthenticationController.cs, etc.)

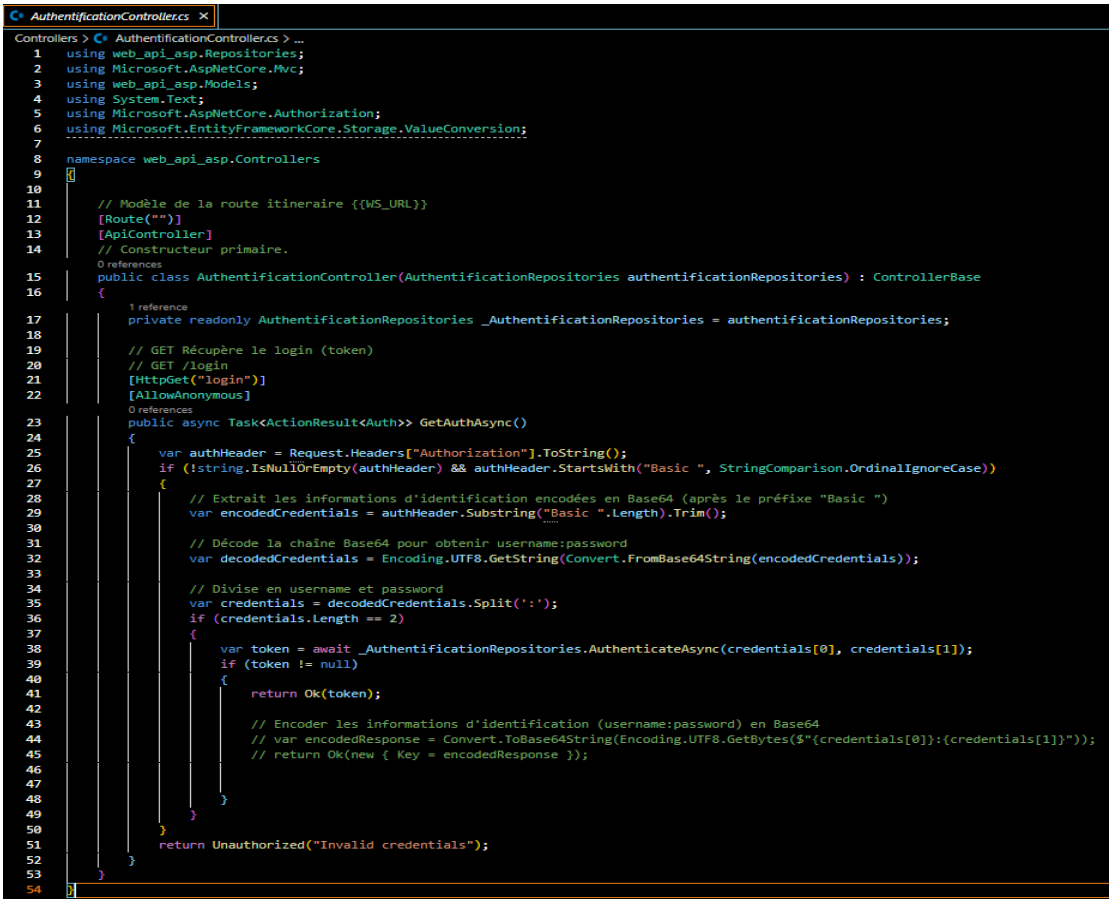
Gère les requêtes liées aux agences

AgenceController.cs

Controllers > AgenceController.cs > AgenceController > AgenceController

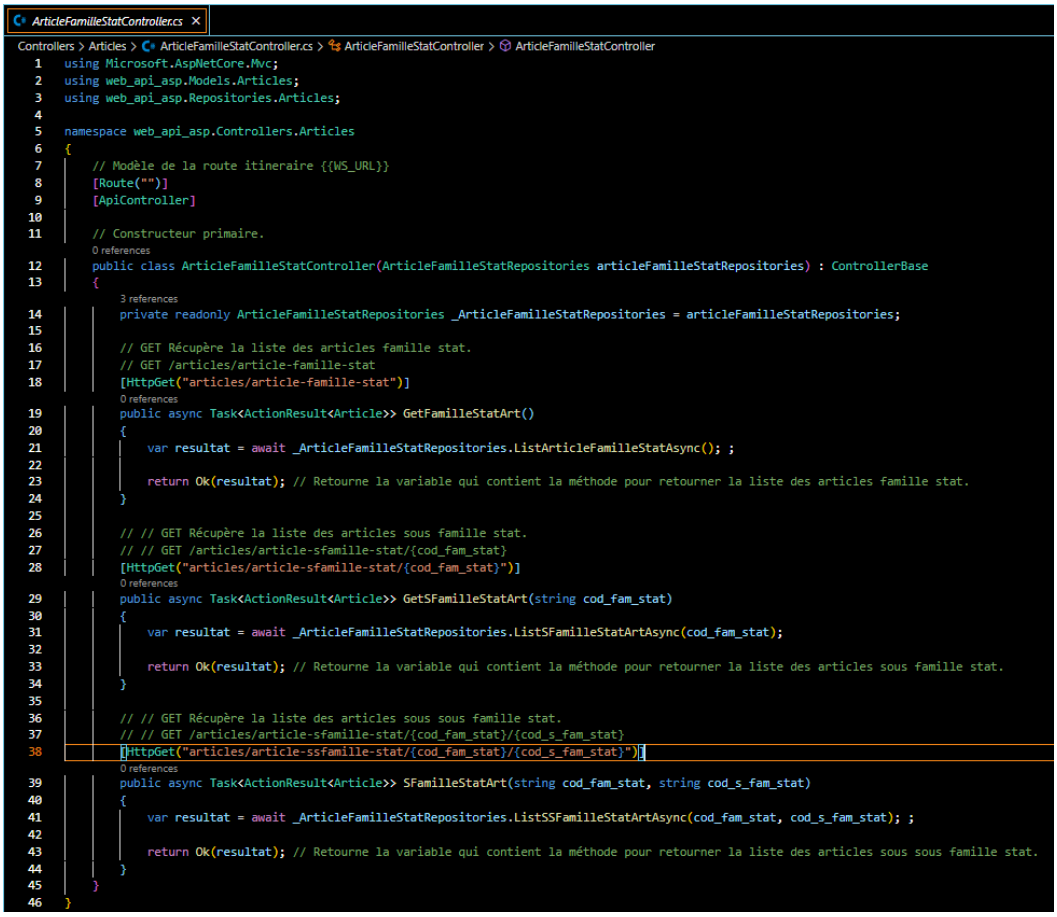
```
1   using Microsoft.AspNetCore.Mvc;
2   using web_api_asp.Repositories;
3   using web_api_asp.Models;
4
5   namespace web_api_asp.Controllers
6   {
7
8       // Modèle de la route itineraire {{WS_URL}}
9       [Route("")]
10      [ApiController]
11
12
13      // Constructeur primaire.
            0 references
14      public class AgenceController(AgenceRepositories agenceRepositories) : ControllerBase
15      {
                2 references
16          private readonly AgenceRepositories _AgenceRepositories = agenceRepositories;
17
18          // GET Récupère toutes les agences .
19          // GET /toutesagences
20          [HttpGet("toutesagences")]
                0 references
21          public async Task<ActionResult<Agences>> GetAllAgences()
22          {
23              var allAgences = await _AgenceRepositories.ListAgencesAsync();
24
25              return Ok(allAgences);  // Retourne la variable qui contient la méthode pour retourner toutes les agances.
26          }
27
28          // GET Récupère les agences d'un utilisateur
29          // GET /agences/{user}
30          [HttpGet("agences/{user}")]
                0 references
31          public async Task<ActionResult<Agences>> GetAgencesByUser(string user)
32          {
33              var agencesByUser = await _AgenceRepositories.ListAgencesByUserAsync(user);
34
35              return Ok(agencesByUser); // Retourne la variable qui contient la méthode pour retourner les agences d'un utilisateur.
36          }
37      }
38  }
```

Endpoint de connexion (get /login) et gère la génération des jetons d'authentification.



```
AuthentificationController.cs
Controllers > AuthentificationController.cs > ...
using web_api_asp.Repositories;
using Microsoft.AspNetCore.Mvc;
using web_api_asp.Models;
using System.Text;
using Microsoft.AspNetCore.Authorization;
using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

namespace web_api_asp.Controllers
{

    // Modèle de la route itineraire {{WS_URL}}
    [Route("")]
    [ApiController]
    // Constructeur primaire.
    0 references
    public class AuthentificationController(AuthentificationRepositories authentificationRepositories) : ControllerBase
    {
        1 reference
        private readonly AuthentificationRepositories _AuthentificationRepositories = authentificationRepositories;

        // GET Récupère le login (token)
        // GET /login
        [HttpGet("login")]
        [AllowAnonymous]
        0 references
        public async Task<ActionResult<Auth>> GetAuthAsync()
        {
            var authHeader = Request.Headers["Authorization"].ToString();
            if (!string.IsNullOrEmpty(authHeader) && authHeader.StartsWith("Basic ", StringComparison.OrdinalIgnoreCase))
            {
                // Extrait les informations d'identification encodées en Base64 (après le préfixe "Basic ")
                var encodedCredentials = authHeader.Substring("Basic ".Length).Trim();

                // Décode la chaîne Base64 pour obtenir username:password
                var decodedCredentials = Encoding.UTF8.GetString(Convert.FromBase64String(encodedCredentials));

                // Divise en username et password
                var credentials = decodedCredentials.Split(':');
                if (credentials.Length == 2)
                {
                    var token = await _AuthentificationRepositories.AuthenticateAsync(credentials[0], credentials[1]);
                    if (token != null)
                    {
                        return Ok(token);

                        // Encoder les informations d'identification (username:password) en Base64
                        // var encodedResponse = Convert.ToBase64String(Encoding.UTF8.GetBytes($"{credentials[0]}:{credentials[1]}"));
                        // return Ok(new { Key = encodedResponse });


                    }
                }
            }
            return Unauthorized("Invalid credentials");
        }
    }
}
```

Consulter ou manipuler les articles

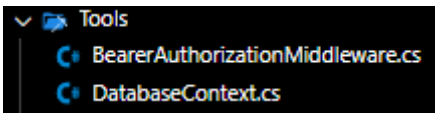


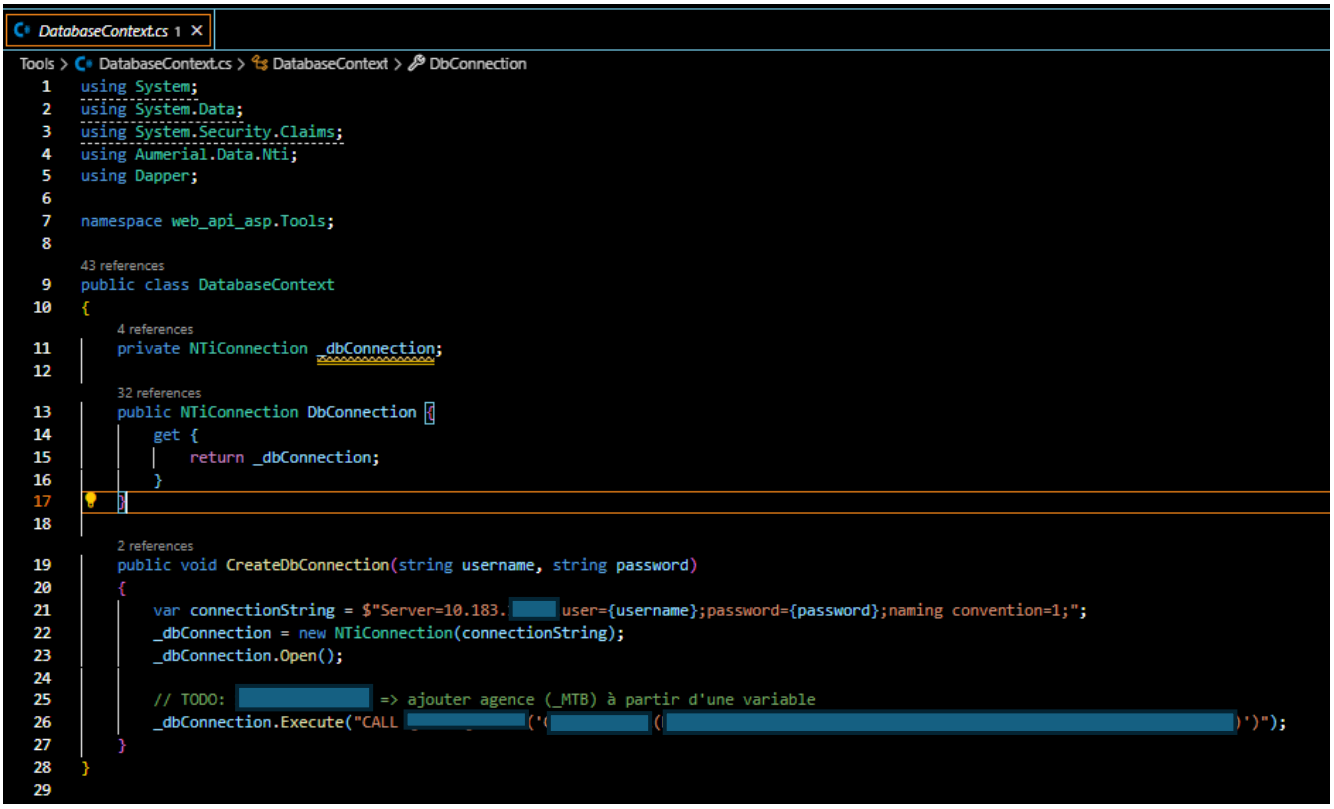
```cs
// ArticleFamilleStatController.cs
// Controllers > Articles > ArticleFamilleStatController.cs > ArticleFamilleStatController > ArticleFamilleStatController
using Microsoft.AspNetCore.Mvc;
using web_api_asp.Models.Articles;
using web_api_asp.Repositories.Articles;

namespace web_api_asp.Controllers.Articles
{
    // Modèle de la route itineraire {{WS_URL}}
    [Route("")]
    [ApiController]

    // Constructeur primaire.
    0 references
    public class ArticleFamilleStatController(ArticleFamilleStatRepositories articleFamilleStatRepositories) : ControllerBase
    {
        3 references
        private readonly ArticleFamilleStatRepositories _ArticleFamilleStatRepositories = articleFamilleStatRepositories;

        // GET Récupère la liste des articles famille stat.
        // GET /articles/article-famille-stat
        [HttpGet("articles/article-famille-stat")]
        0 references
        public async Task<ActionResult<Article>> GetFamilleStatArt()
        {
            var resultat = await _ArticleFamilleStatRepositories.ListArticleFamilleStatAsync(); ;

            return Ok(resultat); // Retourne la variable qui contient la méthode pour retourner la liste des articles famille stat.
        }

        // // GET Récupère la liste des articles sous famille stat.
        // // GET /articles/article-sfamille-stat/{cod_fam_stat}
        [HttpGet("articles/article-sfamille-stat/{cod_fam_stat}")]
        0 references
        public async Task<ActionResult<Article>> GetSFamilleStatArt(string cod_fam_stat)
        {
            var resultat = await _ArticleFamilleStatRepositories.ListSFamilleStatArtAsync(cod_fam_stat);

            return Ok(resultat); // Retourne la variable qui contient la méthode pour retourner la liste des articles sous famille stat.
        }

        // // GET Récupère la liste des articles sous sous famille stat.
        // // GET /articles/article-sfamille-stat/{cod_fam_stat}/{cod_s_fam_stat}
        [HttpGet("articles/article-ssfamille-stat/{cod_fam_stat}/{cod_s_fam_stat}")]
        0 references
        public async Task<ActionResult<Article>> SFamilleStatArt(string cod_fam_stat, string cod_s_fam_stat)
        {
            var resultat = await _ArticleFamilleStatRepositories.ListSSFamilleStatArtAsync(cod_fam_stat, cod_s_fam_stat); ;

            return Ok(resultat); // Retourne la variable qui contient la méthode pour retourner la liste des articles sous sous famille stat.
        }
    }
}
```

Gestion de la base de données (DbContext)

Le DbContext est la **passerelle principale** l'application C# et la base de données. Il représente la **session avec la base de données** et est responsable de la **traduction des requêtes LINQ (Language Integrated Query)** en requêtes SQL. J'ai utilisé le service **DapperNTI**.

AUMERIAL





Ce qui est caché sont les **bibliothèques AS/400** utilisées**.**

Authentification et autorisation (BearerAuthorizationMiddleware.cs)

La sécurité de l'API était primordiale pour garantir que seuls les employés internes autorisés puissent accéder aux données sensibles de l'E-Gescom. Pour cela, un middleware personnalisé, **BearerAuthorizationMiddleware.cs**, a été développé pour gérer le processus d'authentification et d'autorisation des requêtes.

Son rôle est de s’intercaler dans le pipeline de traitement des requêtes HTTP d'ASP.NET Core. Son objectif est d'inspecter chaque requête entrante avant qu'elle n'atteigne le contrôleur cible, afin de vérifier les informations d'authentification présentes dans l'en-tête Authorization.

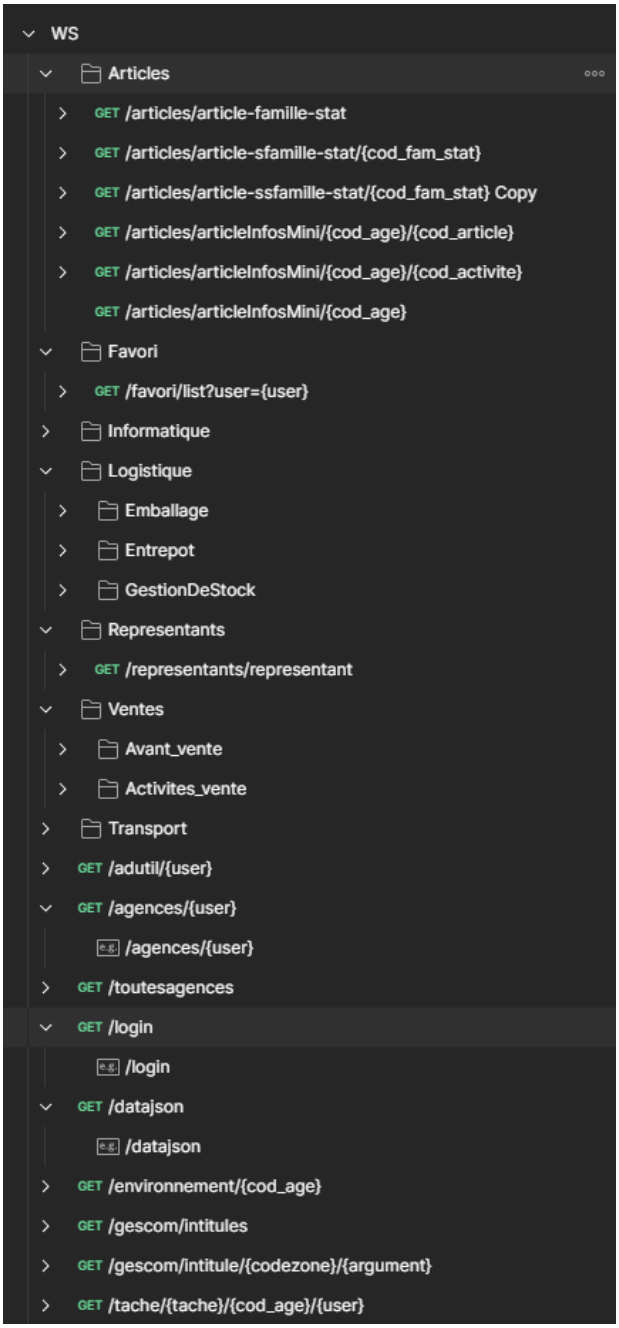
Visuallisation de la méthode

```
0 references
public async Task InvokeAsync(HttpContext context, AuthentificationRepositories authentificationRepositories)
{
    var endpoint = context.GetEndpoint();
    if (endpoint?.Metadata.GetMetadata<AllowAnonymousAttribute>() != null)
    {
        // Skip authentication if the [AllowAnonymous] attribute is present
        await _next(context);
        return;
    }

    // Get the token from the Authorization header
    var authorizationHeader = context.Request.Headers["Authorization"].FirstOrDefault();
    var auth = authorizationHeader?.Split(' ');
    if (auth == null || auth.Length != 2)
    {
        context.Response.StatusCode = 401; // Unauthorized
        await context.Response.WriteAsync("Missing or invalid Authorization header.");
        return;
    }

    var authMethod = auth[0].ToLower();
    var authData = auth[1];

    if (authMethod == "basic")
    {
        var decodedCredentials = Encoding.UTF8.GetString(Convert.FromBase64String(authData));
        var credentials = decodedCredentials.Split(':');
        if (await authentificationRepositories.AuthenticateAsync(credentials[0], credentials[1]) == null)
        {
            context.Response.StatusCode = 403; // Forbidden
            await context.Response.WriteAsync("Invalid credentials.");
            return;
        }
    }
    else if (authMethod == "bearer")
    {
        if (!await authentificationRepositories.ValidateTokenAsync(authData))
        {
            context.Response.StatusCode = 403; // Forbidden
            await context.Response.WriteAsync("Invalid or expired token.");
            return;
        }
    }
    else
    {
        context.Response.StatusCode = 403; // Forbidden
        await context.Response.WriteAsync("Invalid credentials.");
        return;
    }

    // Proceed to the next middleware
    await _next(context);
}
```

Exposition des Services (WS)

L'API RESTful développée expose un ensemble de services web structurés et organisés par domaines métier, permettant aux applications clientes internes de Pro à Pro (notamment le site E-Gescom) d'interagir avec les données et les fonctionnalités de gestion agro-alimentaire. Chaque "folder" de la structure des WS correspond généralement à un contrôleur ou à un ensemble de fonctionnalités regroupées logiquement.

Visualisation des WS (Postman)



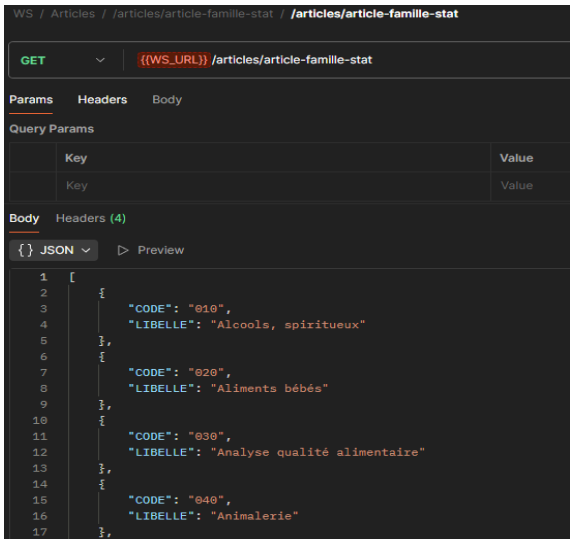
Structure des endpoints de l'API REST

L'API Web RESTful développée pour le site E-Gescom interne de Pro à Pro expose une série d'endpoints, organisés de manière logique par domaines métier. Cette structure permet aux applications clientes internes de Pro à Pro d'interagir de manière claire et cohérente avec les données et les fonctionnalités de gestion agro-alimentaire et d'administration interne.

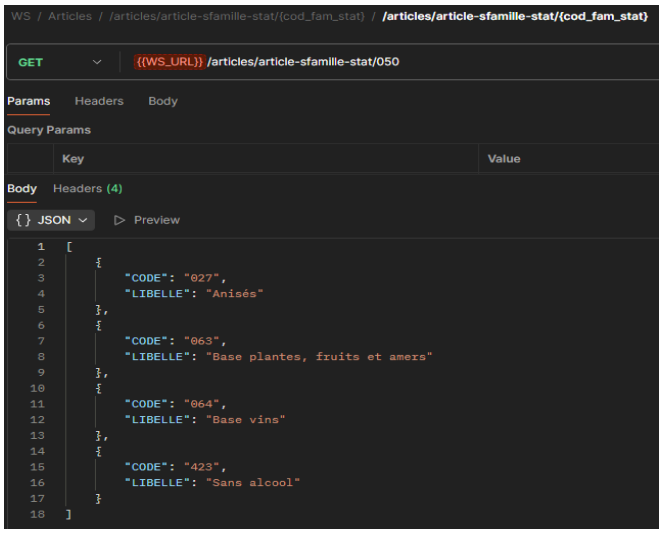
Exemples d'endpoints GET

Articles : Ces endpoints gèrent l'accès et la manipulation des informations relatives aux articles et à leurs classifications.

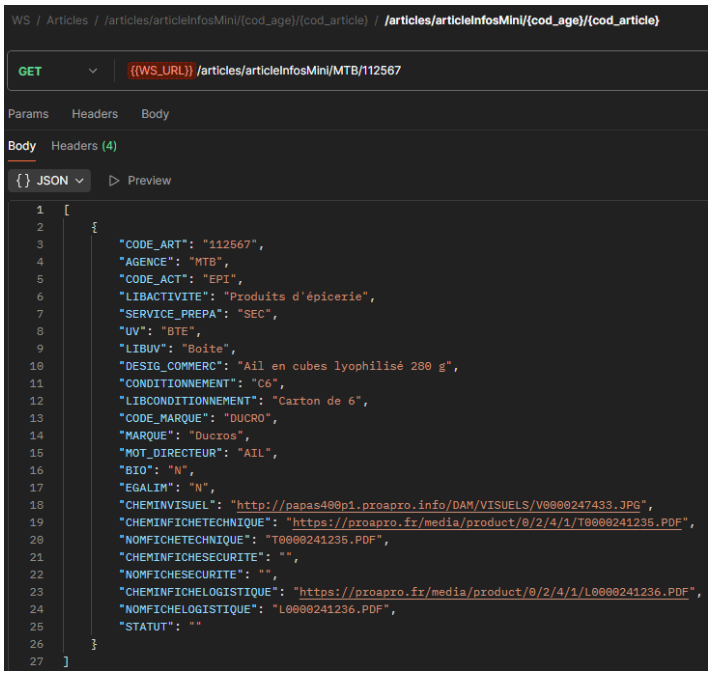
GET /articles/article-famille-stat : Permet d'obtenir les différentes familles statistiques d'articles.



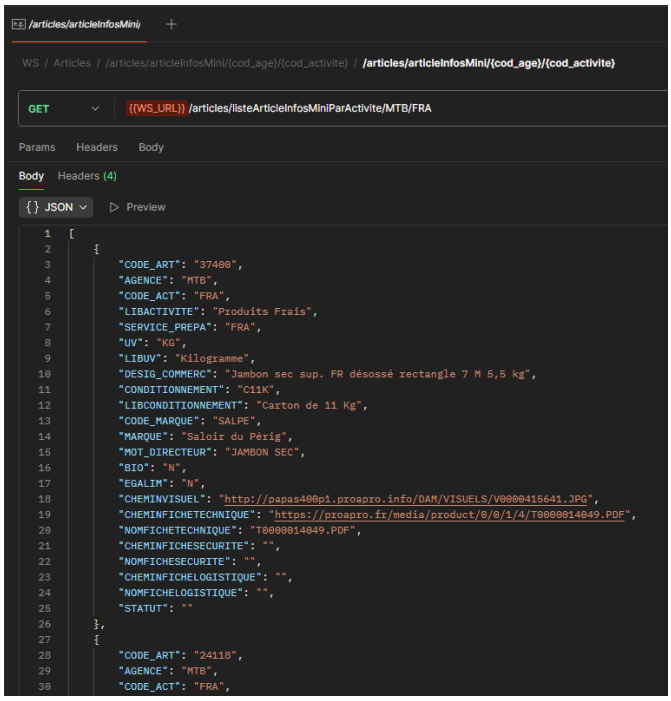
GET /articles/article-sfamille-stat/{cod_fam_stat} : Retourne les sous-familles statistiques associées à une famille statistique donnée.



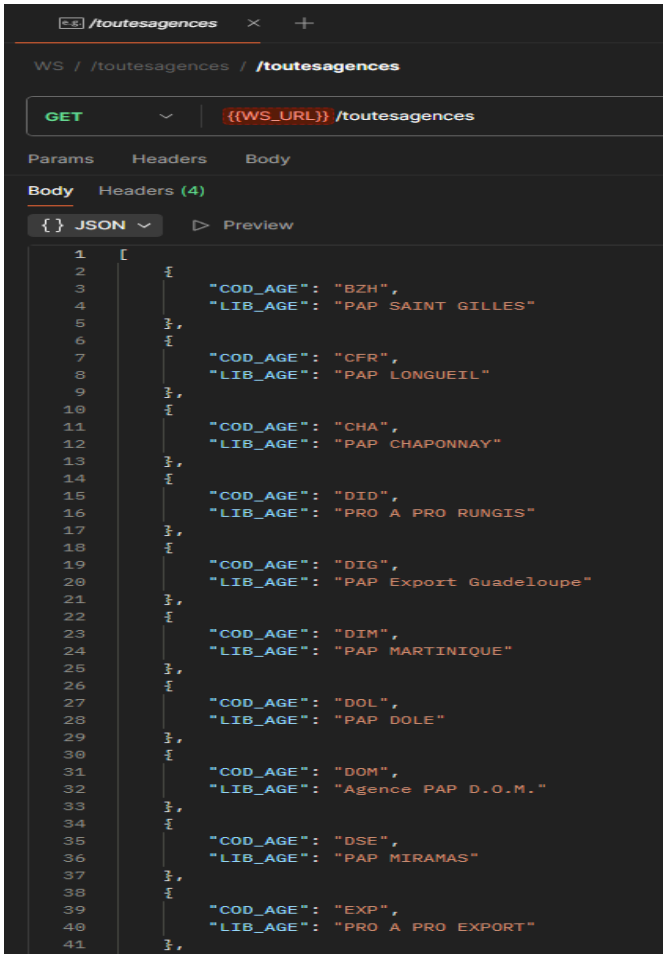
GET /articles/articleInfosMini/{cod_age}/{cod_article} : Fournit des informations synthétiques sur un article spécifique en fonction du code agence et du code article.



GET /articles/articleInfosMini/{cod_age}/{cod_activite} : Fournit des informations synthétiques sur un article en fonction du code agence et du code d'activité (peut-être lié à un type de produit ou service).



Agences : Offre un accès à la liste complète de toutes les agences.



Connexion et Déploiement

Tunnel SSH pour l'accès et le port forwarding

Pour le développement, les tests, la mise en place d'un accès sécurisé et d'une méthode de communication entre les environnements était essentielle. Cette section décrit l'utilisation d'un tunnel SSH avec redirection de ports pour établir cette connectivité.

Objectif du Tunnel SSH **:** Le tunnel SSH (Secure Shell) a été utilisé pour établir une connexion sécurisée vers la machine distante (10.183.X.1).

Il permet de :

- **Accéder au serveur distant :** En se connectant avec mon identifiant AS400 (M.....G@10.183.X.1).
- **Sécuriser les communications :** Toutes les données transitent via un canal chiffré.

Redirection de Port Inversée (-R 5056:localhost:5056) **:** La commande SSH spécifique utilisée était :

```
C:\Users\matheo.sage-ext>ssh -R 5056:localhost:5056 M   G@10.183.   .1
```

Configuration des URLs des services

Les URLs utilisées envoyant des requêtes HTTP aux services externes (par exemple, pour récupérer des données métiers depuis un système AS/400 via une API tierce, ou pour communiquer avec des services PHP existants pendant la phase de migration).

Dans le fichier **config.inc.php** nous y mettons nos urls

config.inc.php

```
// const BASE_PATH_WS = 'http://10.183.   .1:11100/ws';
// const BASE_PATH_WS = 'http://10.183.   .1:11106';
```

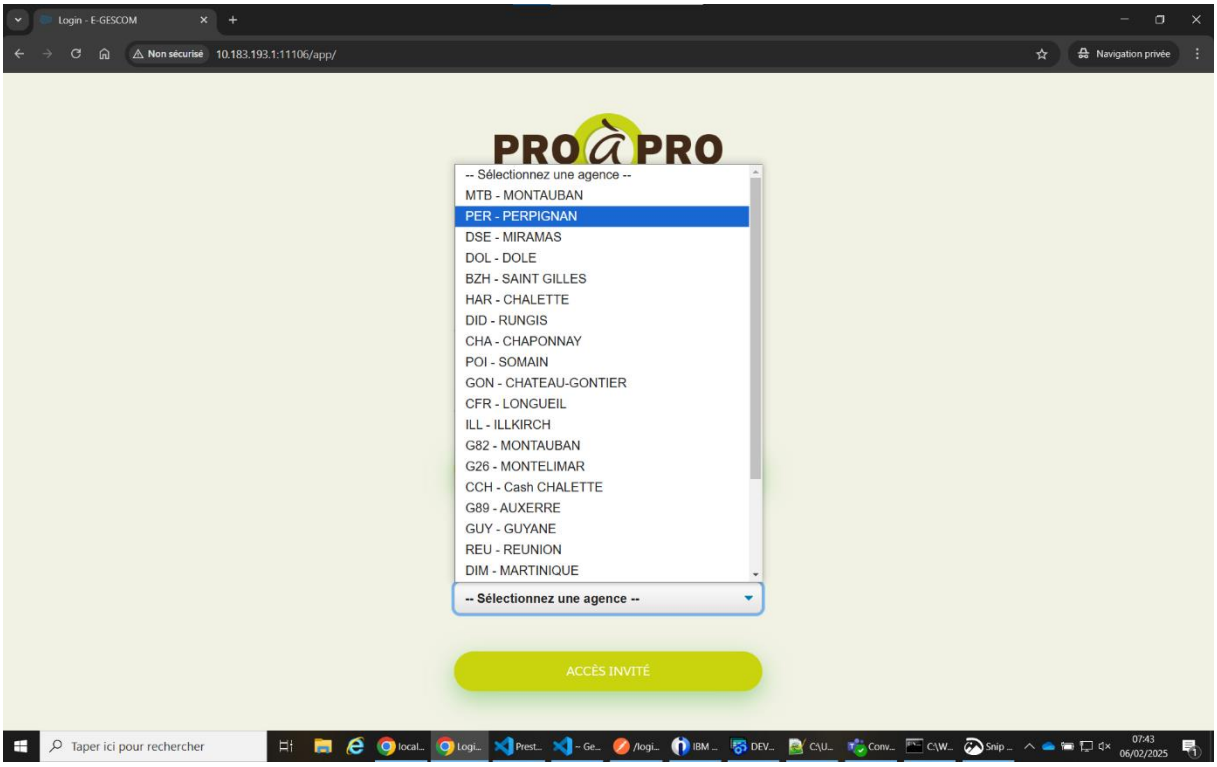
'http://10.183.X.1:11100/ws' => adresse et ports spécifiques pour utilise et joindre les "Web Services" existants (serveur développement).

‘http://10.183.X.1:11106' = adresse et port qui représente mon vhost personnel de développement pour tester mes requetes.

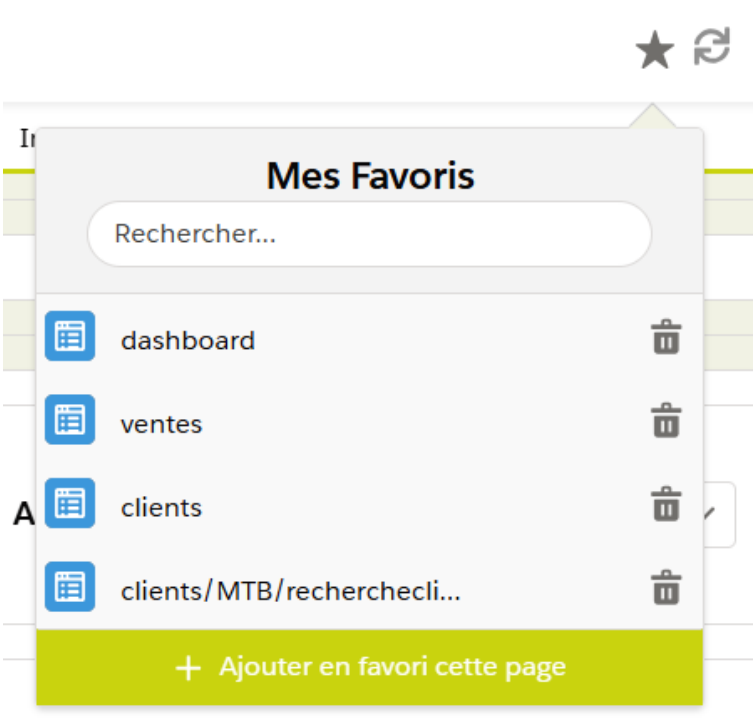
II. Tests

Résultats

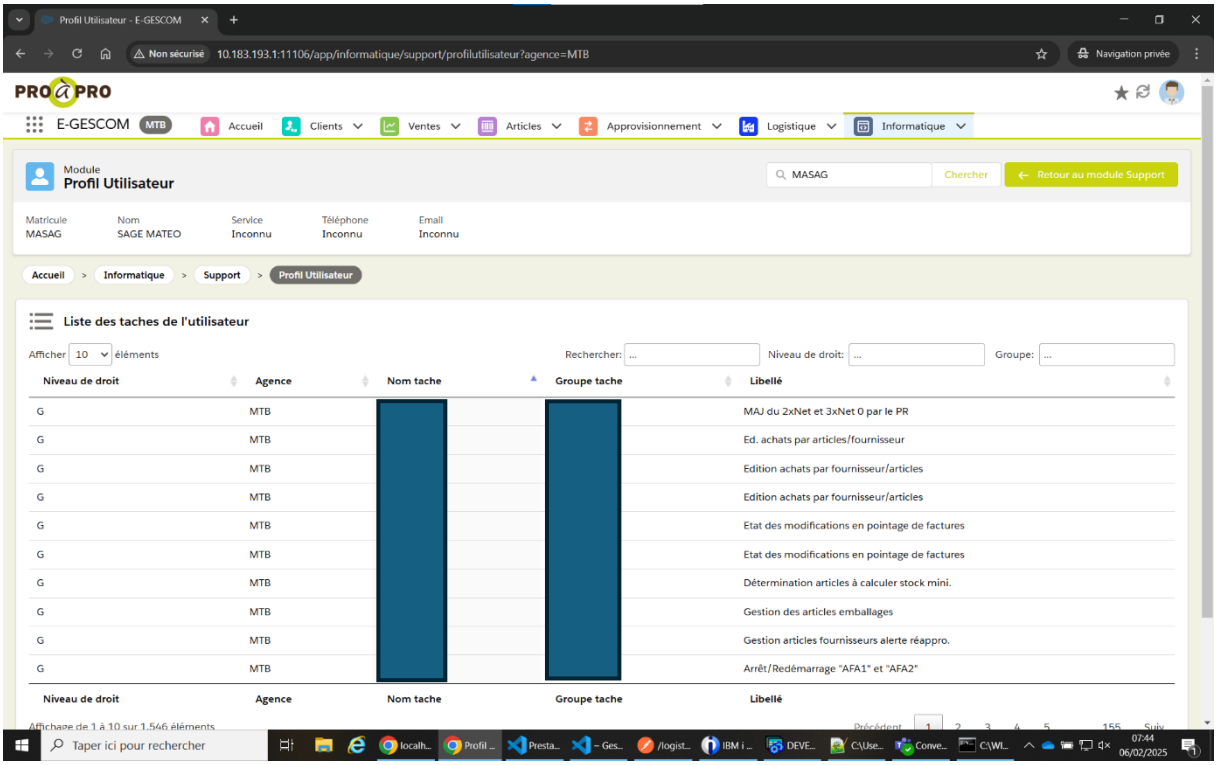
Login :

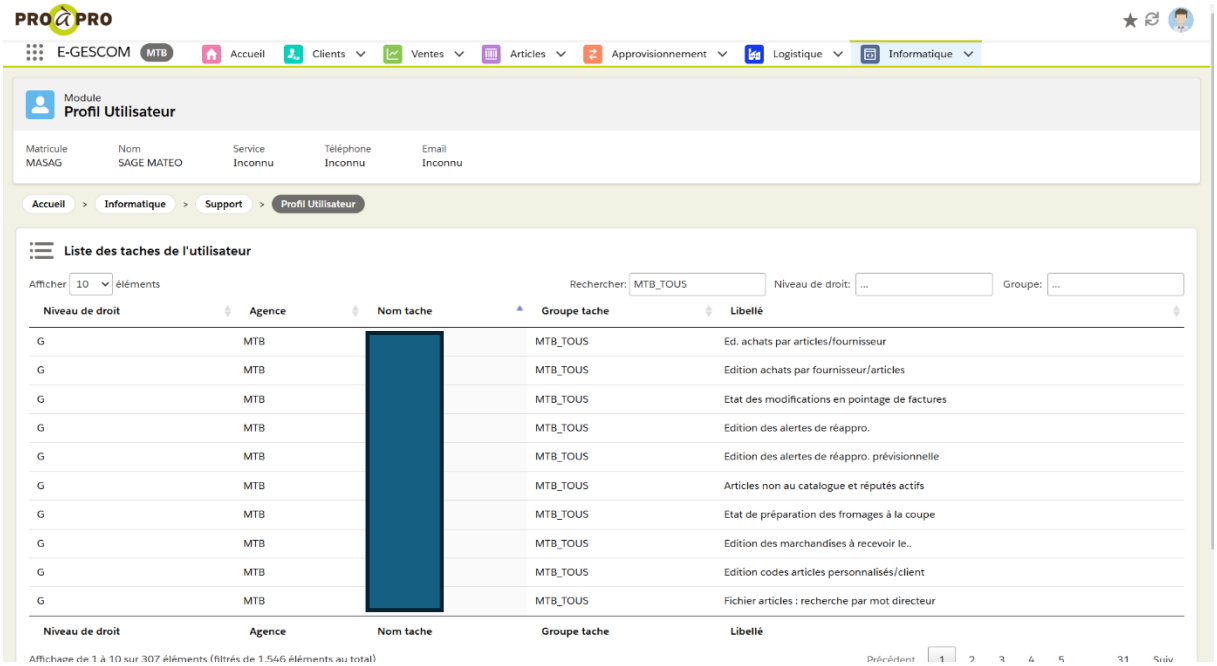


Favoris part rapport à un utilisateur :

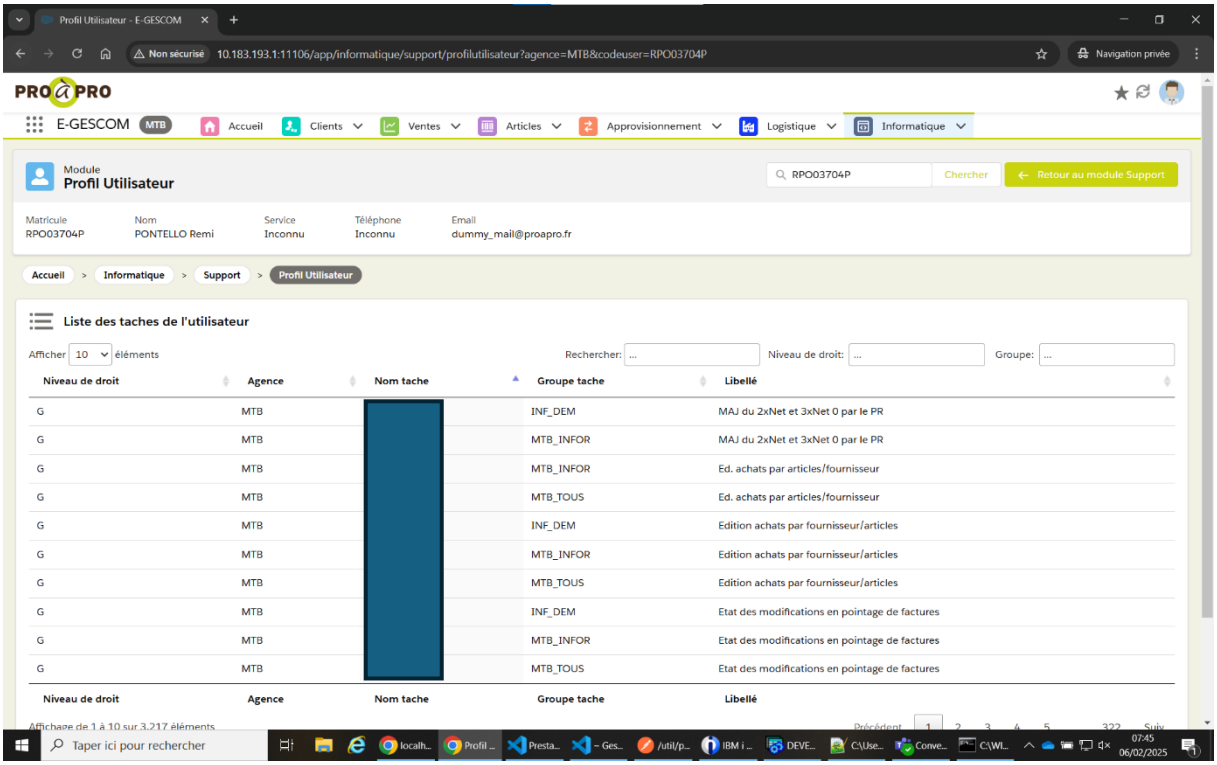


Profil utilisateur plusieurs résultats :

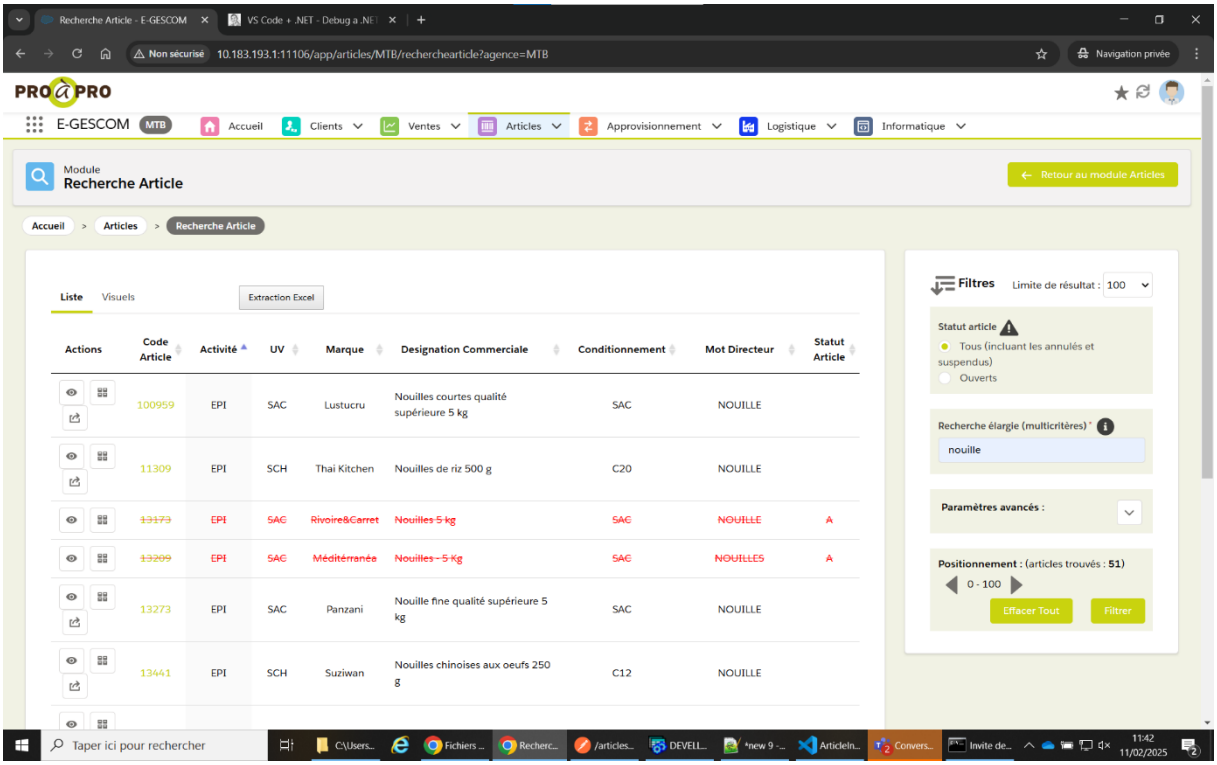


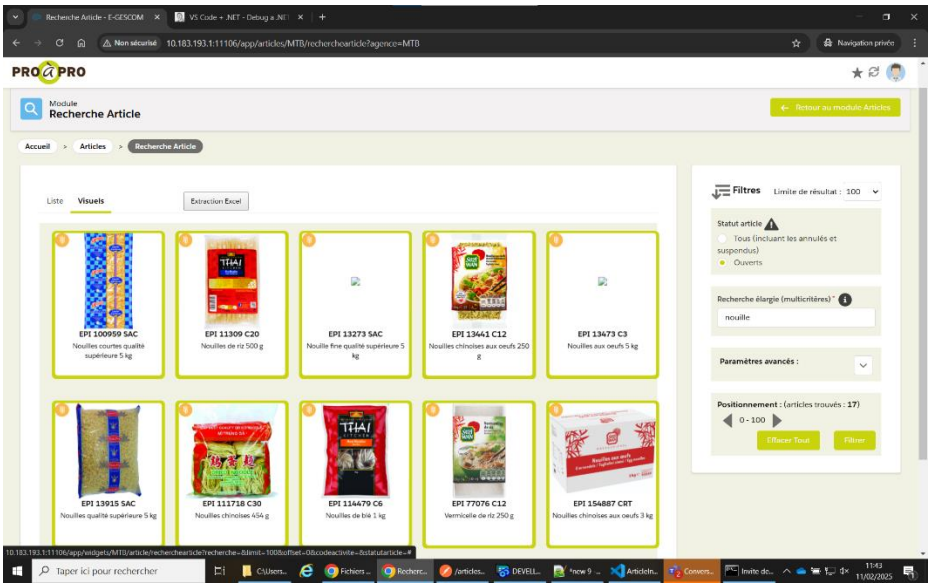


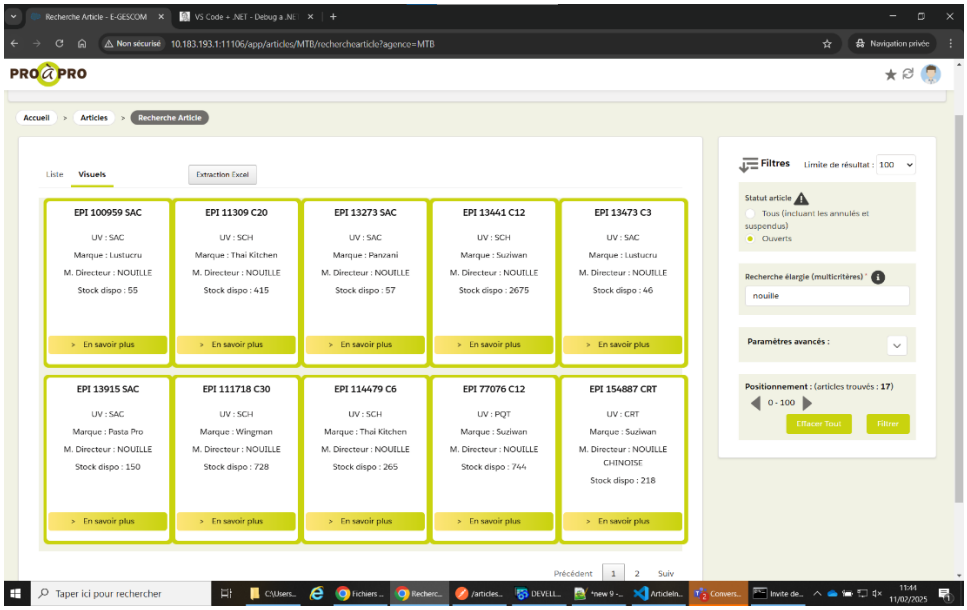
Profil utilisateur recherché :

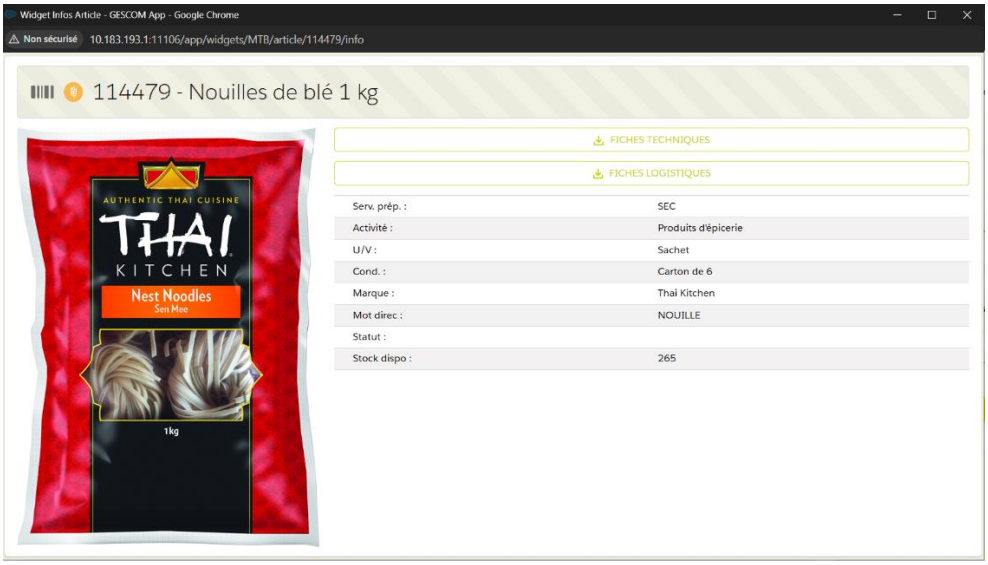


Section Articles => Recherche article :




PRO à PRO
Module
Recherche Article
Retour au module Articles
Accueil
Articles
Recherche Article
Liste
Visuels
Extraction Excel
EPI 100959 SAC
Nouilles courtes qualité supérieure 5 kg
EPI 11309 C20
Nouilles de riz 500 g
EPI 13273 SAC
Nouille fine qualité supérieure 5 kg
EPI 13441 C12
Nouilles chinoises aux oeufs 250 g
EPI 13473 C3
Nouilles aux oeufs 5 kg
EPI 13915 SAC
Nouilles qualité supérieure 5 kg
EPI 111718 C30
Nouilles chinoises 454 g
EPI 114479 C6
Nouilles de blé 1 kg
EPI 77076 C12
Vermicelle de riz 250 g
EPI 154887 CRT
Nouilles chinoises aux oeufs 3 kg
Filtres
Limite de résultat : 100
Statut article
Tous (incluant les annulés et suspendus)
Ouverts
Recherche élargie (multicritères)
nouille
Paramètres avancés :
Positionnement : (articles trouvés : 17)
0 - 100
Effacer Tout
Filtrer
Taper ici pour rechercher


PRO à PRO
Accueil
Articles
Recherche Article
Liste
Visuels
Extraction Excel
EPI 100959 SAC
UV : SAC
Marque : Lustucru
M. Directeur : NOUILLE
Stock dispo : 55
EPI 11309 C20
UV : SCH
Marque : Thai Kitchen
M. Directeur : NOUILLE
Stock dispo : 415
EPI 13273 SAC
UV : SAC
Marque : Panzani
M. Directeur : NOUILLE
Stock dispo : 57
EPI 13441 C12
UV : SCH
Marque : Suziwan
M. Directeur : NOUILLE
Stock dispo : 2675
EPI 13473 C3
UV : SAC
Marque : Lustucru
M. Directeur : NOUILLE
Stock dispo : 46
EPI 13915 SAC
UV : SAC
Marque : Pasta Pro
M. Directeur : NOUILLE
Stock dispo : 150
EPI 111718 C30
UV : SCH
Marque : Wingman
M. Directeur : NOUILLE
Stock dispo : 728
EPI 114479 C6
UV : SCH
Marque : Thai Kitchen
M. Directeur : NOUILLE
Stock dispo : 265
EPI 77076 C12
UV : PQT
Marque : Suziwan
M. Directeur : NOUILLE
Stock dispo : 744
EPI 154887 CRT
UV : CRT
Marque : Suziwan
M. Directeur : NOUILLE CHINOISE
Stock dispo : 218
En savoir plus
Filtres
Limite de résultat : 100
Statut article
Tous (incluant les annulés et suspendus)
Ouverts
Recherche élargie (multicritères)
nouille
Paramètres avancés :
Positionnement : (articles trouvés : 17)
0 - 100
Effacer Tout
Filtrer
Précédent
1
2
Suiv
Taper ici pour rechercher


Widget Infos Article - GESCOM App - Google Chrome
Non sécurisé
10.183.193.1:11106/app/widgets/MTB/article/114479/info
114479 - Nouilles de blé 1 kg
AUTHENTIC THAI CUISINE
THAI
KITCHEN
Nest Noodles
Sen Mee
1kg
FICHES TECHNIQUES
FICHES LOGISTIQUES
Serv. prép. : SEC
Activité : Produits d'épicerie
U/V : Sachet
Cond. : Carton de 6
Marque : Thai Kitchen
Mot direc : NOUILLE
Statut :
Stock dispo : 265

etc...

Conclusion

Bilan du projet

Le projet de développement d'une API Web ASP.NET Core RESTful pour le site interne E-Gescom de Pro à Pro a été une réussite pour moi. Cette initiative a permis de concrétiser la migration d'une architecture PHP existante vers une solution en C#, apportant des améliorations notables en termes de performance, de maintenabilité et de sécurité.

Sécurité renforcée : L'implémentation d'un middleware d'authentification par jeton (Bearer Token) a permis de sécuriser l'accès à l'API pour les utilisateurs internes, garantissant la confidentialité et l'intégrité des données sensibles.

Amélioration des performances : La transition vers ASP.NET Core devrait apporter des gains de performance par rapport à l'ancienne API PHP, améliorant la réactivité des applications utilisées par les employés.

Certes j’ai eu beaucoup de mal à démarrer mais ensuite avec des recherches sur internet et de l’aide de la part de toutes l’équipes de la DSI j’ai pu commencer et avancer sur ce projet qui m’a été donné pendant ma période de stage.

Compétences acquises

Compréhension de l’architecture MVC améliorée et du langage c#.