EXPLICATION TECHNIQUE

Table des matières

Table des illustrations

I. Architecture technique du projet

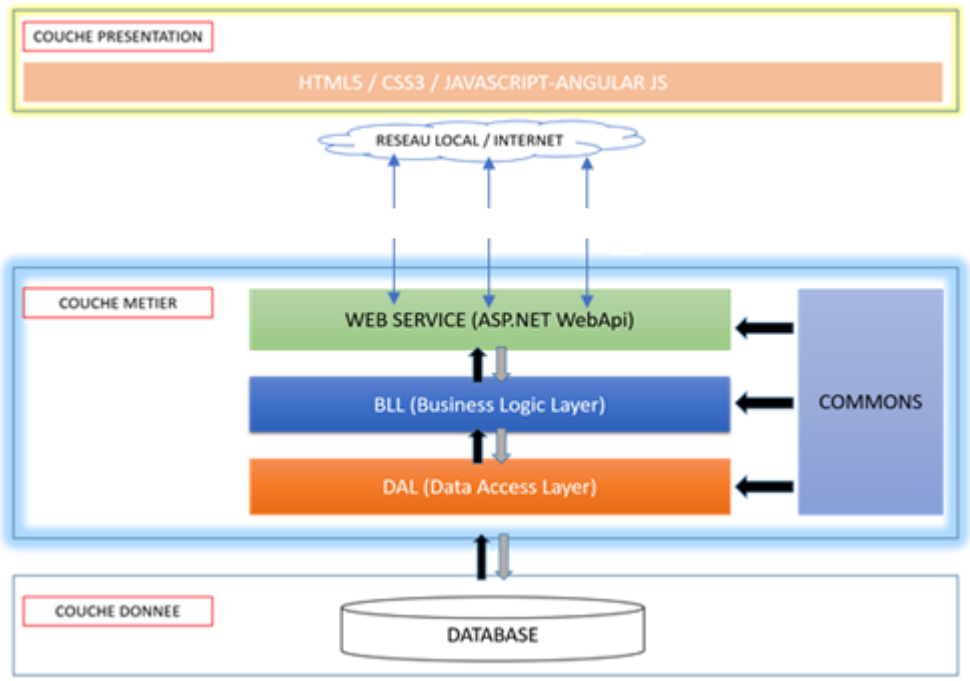

Figure 1: Architecture de la solution

La solution est basée sur l'architecture multitenant[1] proposée par le Framework **ASP .NET BOILERPLATE** disponible sur le lien : https://aspnetboilerplate.com. La base de données de la solution possède les tables de base de cette architecture. Celles-ci sont préfixées ABP.

La solution possède les spécifications suivantes :

La présentation des données

Les interfaces de l'application sont conçues en HTML5, CSS3, JAVASCRIT basé sur ANGULAR JS.

La couche métier

Le **traitement** métier des données : correspondant à la mise en œuvre de l'ensemble des règles de gestion et de la logique applicative. Les utilisateurs accèdent à cette couche depuis la couche de présentation et contactent les web services REST. Les utilisateurs sont identifiés à partir de l'adresse que ceux – ci utilisent pour se connecter aux web services. Le langage utilisé est le **C#** avec l'ORM **DAAPER.**

La couche donnée

Le système de gestion de bases de données utilisé est **Microsoft SQL SERVER**.

[1] Principe d'architecture permettant à un logiciel de servir plusieurs organisations clientes à partir d'un seul déploiement

II. Contenu de la solution

La solution est composée de 5 projets présent dans Quatre (4) répertoires :

- Un répertoire business contenant 3 projets : « CORE », « Data », « Service » pour la liaison avec la base de données et l'implémentation des services.
- Un répertoire Infrastructure avec le projet « Admin. Common » commun à tous les projets avec des constantes et des éléments de base.
- Un répertoire Présentation contenant un projet WEB contenant les web services (controllers ASP .NET) et l'interface utilisateur

III. Génération de fichiers

La génération des modèles est faite à partir d'entity developper en utilisant des templates présents dans le fichier _commons du projet. Elle permet d'ajouter toutes modifications de la base de données au projet. Ces modifications concernent l'ajout, la modification ou la suppression des tables et vues. Un mapping[2] est fait entre les tables de la base de données et les DTO[3] utilisées dans le projet.

La génération se fait au niveau du projet « DATA », dans le sous-dossier *« DbGeneration/Model/DataModel.edml ».*

1. Importation des données depuis la base de données

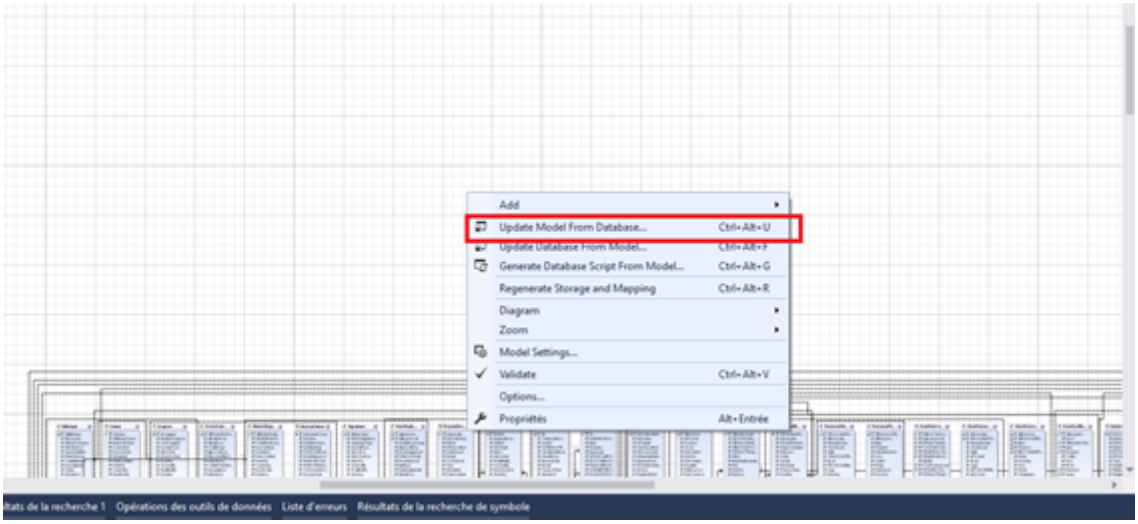

Figure 2 : importation des données

2. Résultat d'importation

A la génération trois (3) éléments sont créés automatiquement pour chaque table :

- Un DTO (Présent dans la classe DataModel.DTOS du fichier ***« Dto/Generated/DataModel.DTOS »*** du projet « CORE ».
- Un Provider : dans le fichier « Provider » du projet « Data » qui permet d'interagir avec la base de données.
- Un manager : dans le fichier « ***Manager*** » du projet « Service » qui devra contenir les règles de gestion. C'est ici que seront implémentés les règles métiers.

[2] Correspondance entre deux structures de données.
[3] Objet qui transporte les données entre les processus

IV. Constantes globales

Les données et indicateurs à valeur fixe pourront être identifiées dans le dossier « ***Enum*** » du projet « Admin.Common ».

Toutes les constantes utilisées dans le projet pourront être déclarées dans le fichier « ***Definition*** » contenu dans le dossier « ***Enum*** ».

V. Traitement des règles métiers

Toutes les règles métiers du projet doivent être implémentée dans le fichier « Manager » du projet « SERVICE ». Chaque fichier présent dans ce dossier présente les tables du projet suivie du nom « Manager ». Ces éléments ont été générés et peuvent être modifiés pour ajouter des conditions comme des champs obligatoires ou des valeurs non-conformes par exemple.

Le manager présente des actions standards sur les tables (Ajout, modification, Suppression, Lecture).

D'autres méthodes peuvent être ajoutées par le développeur.

1. Paramètre d'ajout et d'insertion

Les managers et les providers utilisent en commun des **BusinessRequest.** Ce sont des classes génériques qui permettent d'exécuter des requètes dans la base de données**.**

Un BusinessRequest est composé de :

- Un champ ItemToSearch de même type que l'objet mappé à la table sur laquelle l'on travail. Pour chacune des valeurs contenues dans l'objet, une recherche est faite dans la base de données sur celles-ci.
- Un champ ItemsToSearch qui est une liste d'objet dans le cas où l'on veut effectuer des recherches avec des conditions supplémentaires.
- Un champ ItemsToSave qui contient une liste d'objet qui permettront de mettre à jour la base de données dans le cas d'un ajout ou d'une modification de données sur une table.

2. Appel des providers

Les managers interagissent directement avec les providers pour l'interaction avec la base de données. Ceux-ci peuvent appeler les providers générés ou les providers créés pour les appels de procédures stockées (Providers).

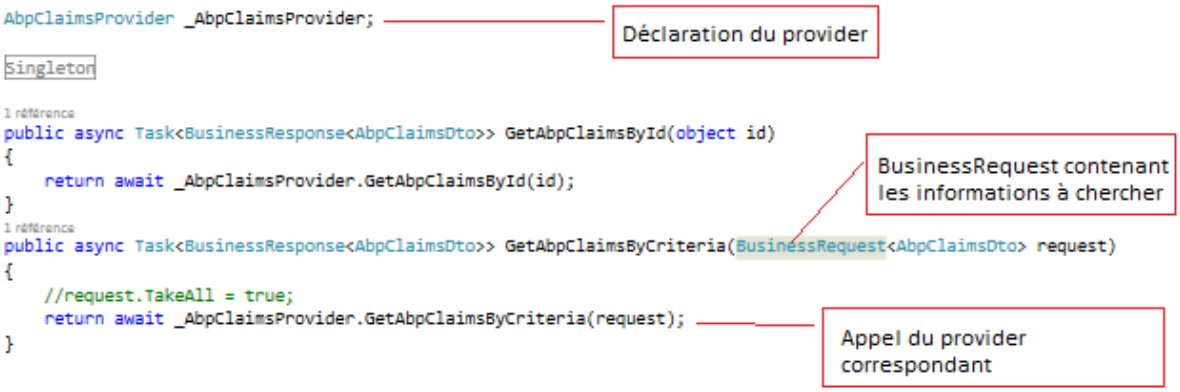

Figure 3: Exemple d'appel de manager (Cas d'une recherche)

Figure 4:Exemple de manager (Cas d'un ajout)

Remarque

Pour l'ajout, la modification et la suppression, uniquement les méthodes « Save » des providers Sont appelées :

- Pour la modification, l'id de l'élément et la propriété à modifier peuvent juste être spécifiés pour effectuer une modification en appelant la méthode « Save » correspondante
- Pour la suppression, tous les éléments en bases de données sont supprimés logiquement (il s'agit d'un booléen « isDeleted » présent dans toutes les tables du projet qui permet d'effectuer une suppression logique. La suppression est donc une modification du champ isDeleted de l'élément à supprimer.

VI. ViewModels

1. Role

Les viewmodels permettent de limiter les champs à envoyer aux webservices.

Ils font une correspondance avec les DTO. Chaque viewmodels doit posséder un DTO auquel il fait référence.

2. Déclaration

Les viewmodels sont déclarés dans le dossier « ***models*** » du projet « Web ». Un fichier modèle est disponible par module.

Les viewmodels se présentent comme suit :

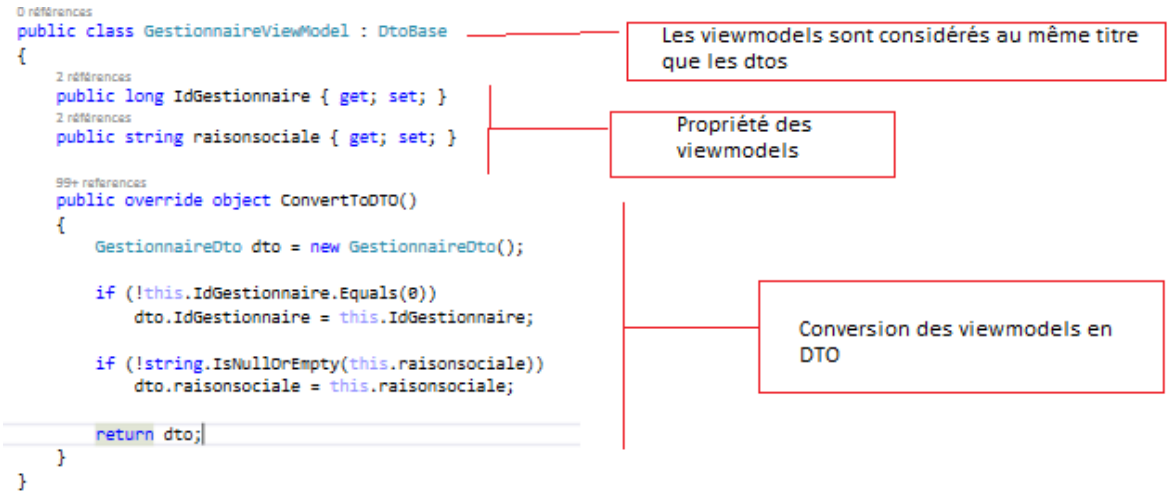

Figure 5: Présentation d'un viewmodel

VII. Controllers

1. Rôles

Les controllers sont les points d'entrées des requètes utilisateurs. Ils réceptionnent les requètes à partir des BusinessRequest couplés aux viewmodels et les transmet aux manager qui effectuent les traitements.

2. Déclaration

Les controllers sont déclarés dans le dossier « ***controllers*** » du projet « Web ». Un fichier Controller est disponible par module.


```
[AllowAnonymous]
[Route("SaveBilletage")]
[HttpPost]
0 références
public async Task<IHttpActionResult> SaveBilletage([FromBody]BusinessRequest<BilletageViewModel> request)
{
    BilletageManager manager = new BilletageManager();
    BusinessRequest<BilletageDto> convertedRequest = request.ConvertRequestToDto<BilletageViewModel, BilletageDto>();
    convertedRequest.IdCurrentTenant = this.Tenant.Id;
    try
    {
        var response = await manager.SaveBilletages(convertedRequest);
        if (response.HasError)
        {
            return Content(System.Net.HttpStatusCode.BadRequest, response).With(response.Message, c => c.Content = new StringContent(response.Message));
        }
        return Ok(response);

    }
    catch (Exception ex)
    {
        // throw new Exception(e.Message);
        //   ModelState.AddModelError("", ex.Message);
        return Content(System.Net.HttpStatusCode.BadRequest, ex).With(ex.Message, c => c.Content = new StringContent(ex.Message)); ;
    }
}
```

Figure 6:Exemple de Controller

Remarque :

L'adresse du service est : http://AdresseDuServeur/prefixe/NomDuService

VIII. IHM

L'IHM du projet est faite en AngularJS dans le projet « Web » de la solution.

1. Création de modules

Les étapes suivantes sont à suivre pour la création d'un module Dans le projet « Web » :

- Déclarer le module dans le fichier « config/install-features.js ».

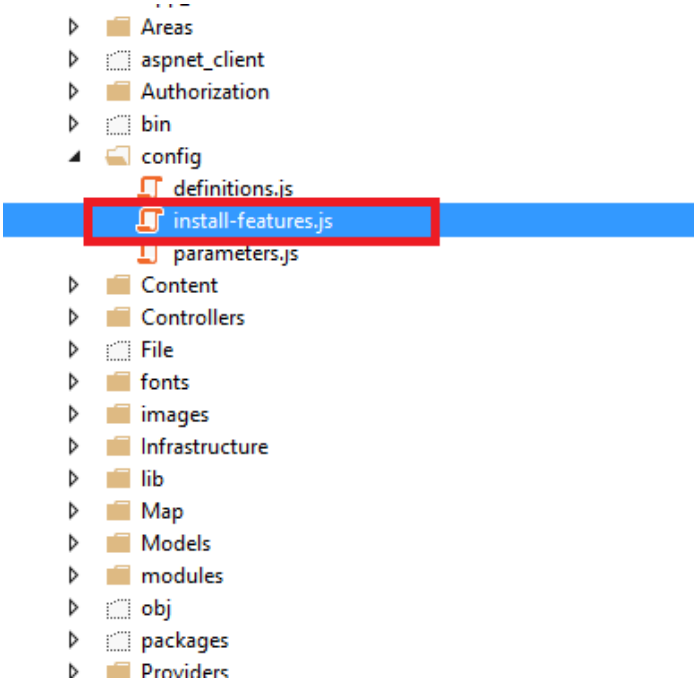

Figure 7: Fichier de déclaration des modules

```
paths: {

    //inserer le module
    'core-install-features': HelpersErp.getUrlByConfigBase("coreInstallFeatures", MODULE_ERP_CORE),
    'securite-install-features': HelpersErp.getUrlByConfigBase("securiteInstallFeatures", MODULE_ERP_SECURITE),
    'assurance-install-features': HelpersErp.getUrlByConfigBase("assuranceInstallFeatures", MODULE_ERP_ASSURANCE),
    'crud-install-features': HelpersErp.getUrlByConfigBase("crudInstallFeatures", MODULE_ERP_CRUD),
    'caisse-install-features': HelpersErp.getUrlByConfigBase("caisseInstallFeatures", MODULE_ERP_CAISSE),
    'reference-install-features': HelpersErp.getUrlByConfigBase("referenceInstallFeatures", MODULE_ERP_REFERENCE),
    'parametre-install-features': HelpersErp.getUrlByConfigBase("parametreInstallFeatures", MODULE_ERP_PARAMETRE),
    'tresorerie-install-features': HelpersErp.getUrlByConfigBase("tresorerieInstallFeatures", MODULE_ERP_TRESORERIE),
    'approvisionnement-install-features': HelpersErp.getUrlByConfigBase("approvisionnementInstallFeatures", MODULE_ERP_APPROVISIONNEMENT),
    'dossiermedical-install-features': HelpersErp.getUrlByConfigBase("dossiermedicalInstallFeatures", MODULE_ERP_DOSSIERMEDICAL),
    'prestataire-install-features': HelpersErp.getUrlByConfigBase("prestataireInstallFeatures", MODULE_ERP_PRESTATAIRE),

    'comptabilite-install-features': HelpersErp.getUrlByConfigBase("comptabiliteInstallFeatures", MODULE_ERP_COMPTABILITE),
    'workflow-install-features': HelpersErp.getUrlByConfigBase("workflowInstallFeatures", MODULE_ERP_WORKFLOW),
    'precompte-install-features': HelpersErp.getUrlByConfigBase("precompteInstallFeatures", MODULE_ERP_PRECOMPTE),

    'garant-install-features': HelpersErp.getUrlByConfigBase("garantInstallFeatures", MODULE_ERP_GARANT),
    'prestation-install-features': HelpersErp.getUrlByConfigBase("prestationInstallFeatures", MODULE_ERP_PRESTATION),

    'alerte-install-features': HelpersErp.getUrlByConfigBase("alerteInstallFeatures", MODULE_ERP_ALERTE),

    'erp-install-components': HelpersErp.getUrlByConfigBase("install-components")
},
shim: {

inserer le module
    'app': ["core-install-features", "crud-install-features", "assurance-install-features", "caisse-install-features", "reference-install-featu
    'app-controllers': ["core-install-features"],
    'definitions': ["angular"],
    'utilities': ["jquery"],
    'erp-directives': ["angular"],
    'erp-filters': ["angular"],
    'core-install-features': ["angular"],
    'parameters': ["angular"]
}
;
```

Figure 8: Zone de déclaration des modules

- Créer un dossier correspondant au nom du module dans le dossier « module » prévu
- Créer trois (3) sous dossier « config » « features » et « shared ».

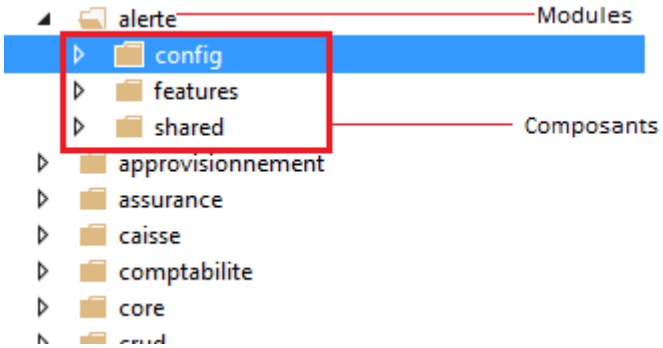

Figure 9: Element de modules angular

- Ajouter un fichier avec la nomenclature « NomduModuleInstallFeatures » qui permettra de référencer les éléments du module (services, modals, models, directives).

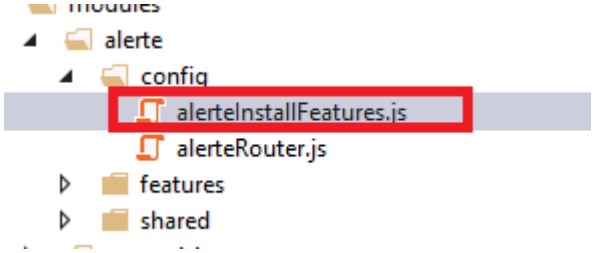

Figure 10: Fichier de référence des éléments du modules

```
define([], function () {
    'use strict';

    /* REFERENCE Install features */

    require.config({
        baseUrl: "",
        paths: {
            'alerte-router': HelpersErp.getUrlByConfigBase('alerteRouter', MODULE_ERP_ALERTE),
            'alerte-controllers': HelpersErp.getUrlBySharedBase('alerteControllers', MODULE_ERP_ALERTE),
            'alerte-directives': HelpersErp.getUrlBySharedBase('alerteDirectives', MODULE_ERP_ALERTE),
            'alerte-filters': HelpersErp.getUrlBySharedBase('alerteFilters', MODULE_ERP_ALERTE),
            'alerte-models': HelpersErp.getUrlBySharedBase('alerteModels', MODULE_ERP_ALERTE),
            'alerte-scripts': HelpersErp.getUrlBySharedBase('alerteScripts', MODULE_ERP_ALERTE),
            'alerte-services': HelpersErp.getUrlBySharedBase('alerteServices', MODULE_ERP_ALERTE)
            //'alerte-modal-services': HelpersErp.getUrlBySharedBase('alerteModalServices', MODULE_ERP_ALEF
        },
        shim: {
            'alerte-router': ['angular']
        },
        deps: ['alerte-router']
    });
});
```

Figure 11: Contenu de fichier install feature

- Ajouter un fichier avec la nomenclature « NomduModuleRouter » qui permettra de référencer les pages du module.

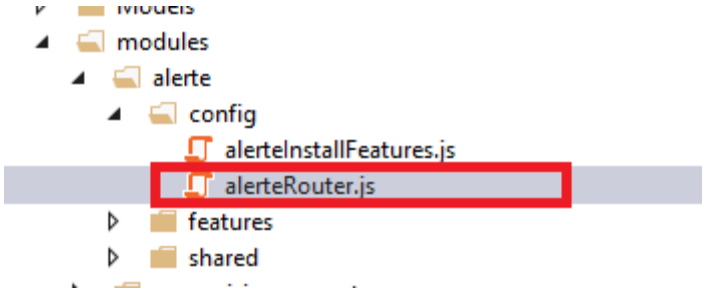

Figure 12: Fichier de référence des pages

```
define(['definitions'], function () {
    'use strict';

    /* CAISSE Router */

  erpRouter.config(['$stateProvider', function ($stateProvider) {
      $stateProvider
          .state('app.alerte', {
              url: '/alerte',
              templateUrl: HelpersErp.getUrlByFeaturesBase('alerte/alerteList.html', MODULE_ERP_ALERTE),
              resolve: {
                  loadMyCtrl: ['$ocLazyLoad', function ($ocLazyLoad) {
                      // you can lazy load files for an existing module
                      return $ocLazyLoad.load({
                          files: [HelpersErp.getUrlByFeaturesBase('alerte/alerteListCtrl.js', MODULE_ERP_ALERTE)]
                      });
                  }]
              }
          })


    }]);

});
```

Figure 13: Contenu du fichier de référence des pages du module

- Un dossier features qui contiendra les pages du module
- Un fichier shared regroupant tous les éléments référencés dans le fichier « InstallFeatures »(models, directives, modals, services)