
CO7SFGL0 2A Génie Logiciel

NUDGE



Rapport du projet de génie logiciel

Adrien Bourdin & Nanor Otabachian

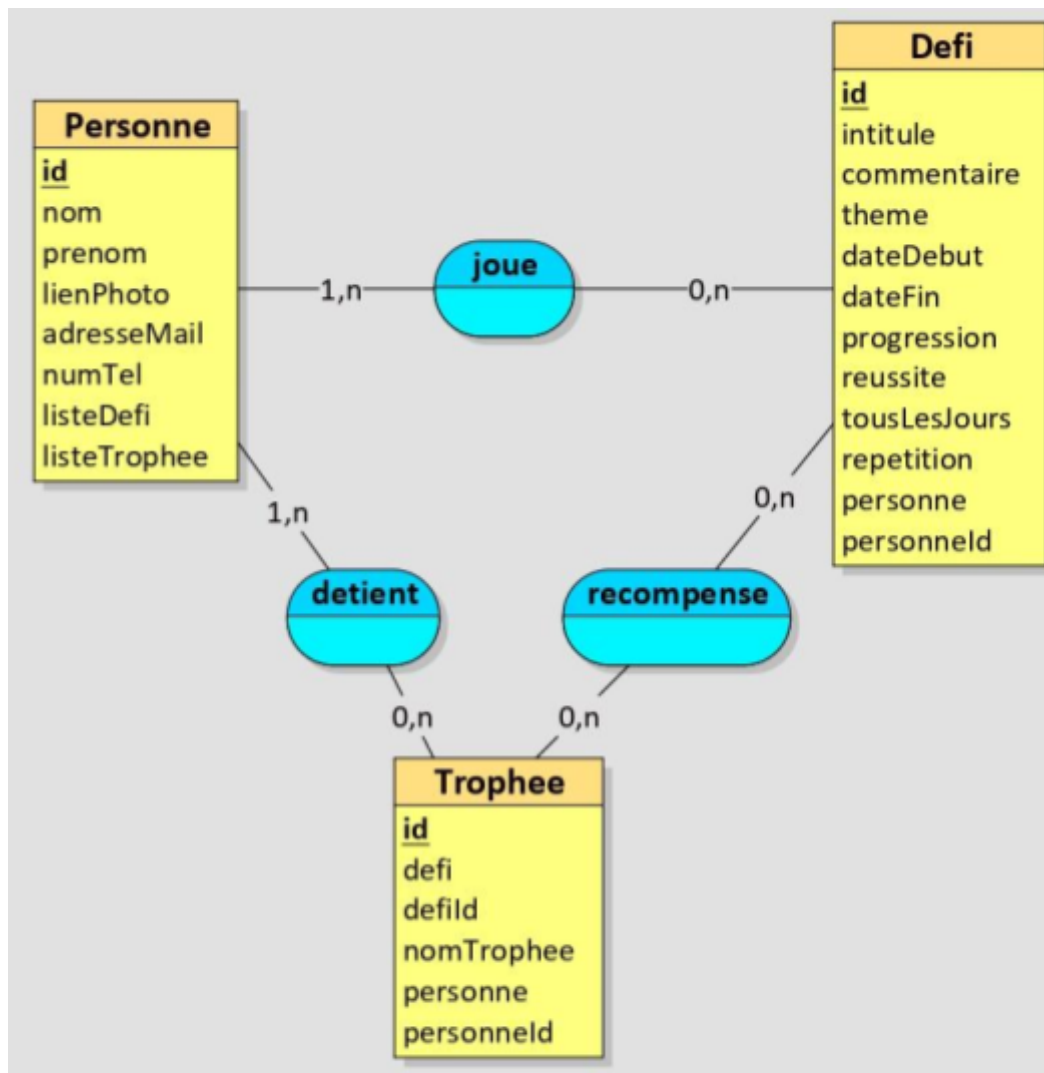
SOMMAIRE

Introduction et thématique de l'application	3
Modélisation des données	4
Documentation de l'API	7
Répartition des tâches et planning	9
Tests automatisés	10
Bilan et perspectives	13

I. Introduction et thématique de l'application

Dans le cadre de l'unité d'enseignement de génie logiciel de deuxième année à l'École Nationale Supérieure de Cognitique il nous a été demandé de créer la partie back-end d'une future application mobile dont le thème était au choix, à l'aide de la technologie ASP.NET Core MVC. Nous avons décidé de développer un site web autour de la notion de défi. Baptisé "Nudge", le concept du site est de se lancer des défis, ainsi qu'à ses amis, et de renseigner au jour le jour sa progression. Il est aussi possible de discerner des trophées en fonction des réalisations de chacun. Le site est pour l'instant à utiliser en "petite comité" puisque tout le monde peut rajouter des utilisateurs, des défis ou des trophées mais aussi les modifier et/ou les supprimer.

II. Modélisation des données



Modèle relationnel

Nom	Désignation	Type
nom	Nom de la personne	string
prenom	Prénom de la personne	string
lienPhoto	Le lien de l'image définie en photo de profil	string
adresseMail	Courriel	string
numTel	Mot de passe	string
listeDefi	Liste des défis de la personne	Defi
listeTrophee	Liste des trophées de la personne	Trophee
NomTrophee	Nom du trophée	string
intitule	Nom du défi	string
commentaire	Commentaire décrivant le défi	string
theme	Thématique du défi	string
dateDebut	Date de début du défi	date
dateFin	Date de fin du défi	date
progression	Score de progression au défi	int
reussite	Statut de validation du défi	bool
tousLesJours	Statut quotidien ou non du défi	bool
repetition	Nombre de répétition du défi	int

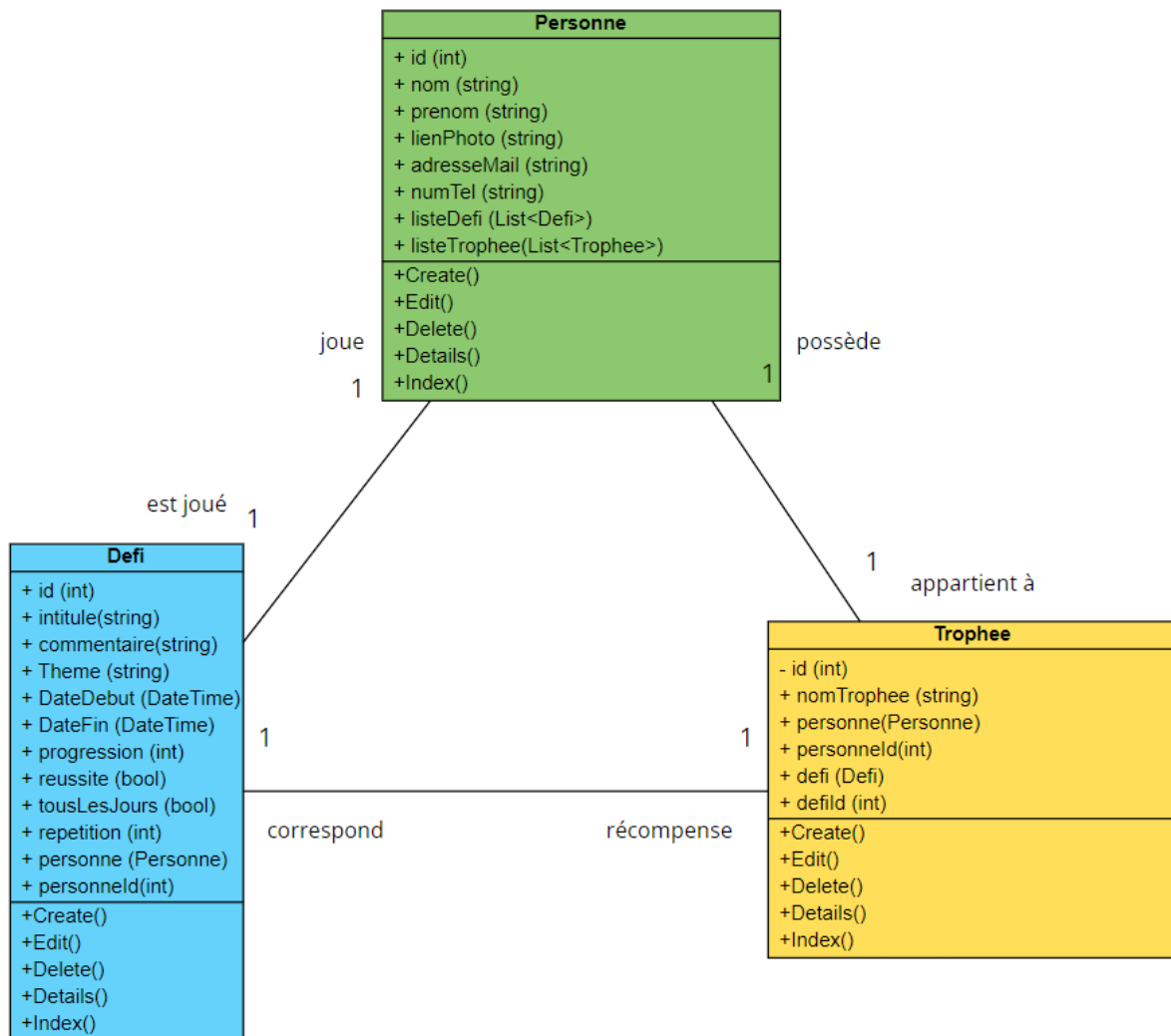


Diagramme de classe métier

III. Documentation de l'API

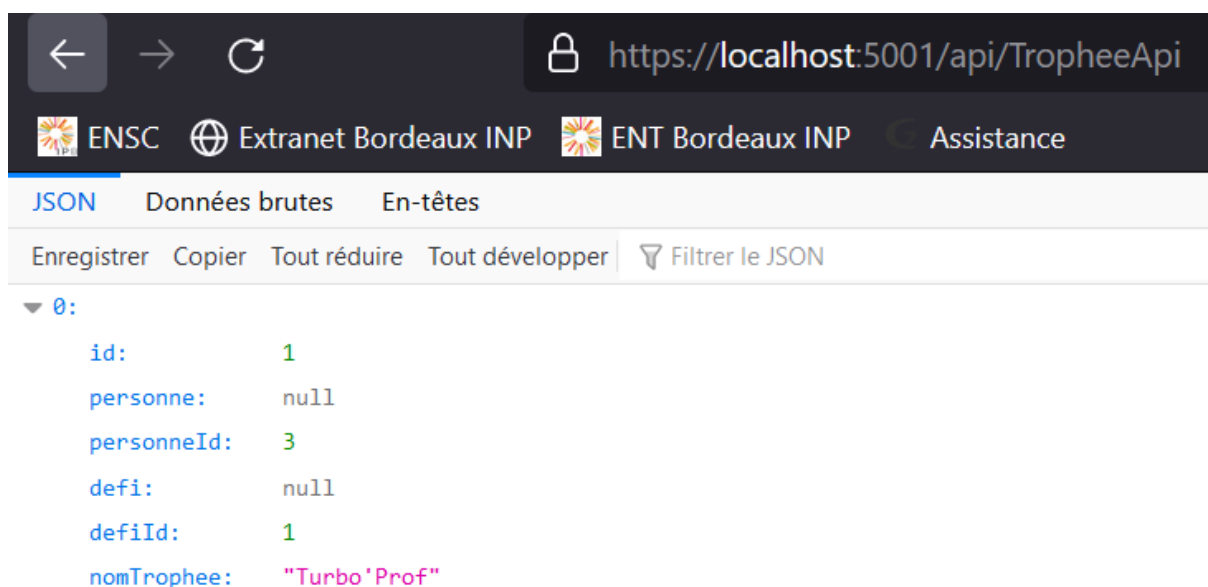
Afin de mettre en place un système de consultation des API Web nous avons créé tout notre projet et une fois tout notre modèle et notre BDD mis en plus, nous sommes venus ajouter nos contrôleurs des api grâce à un 3 "scaffoldings" en plus pour chacun des controllers déjà présents.

Nous avons ainsi généré les ApiControllerers suivants :





- PersonneApiController
- DefiApiController
- TrophéeApiController

Ceux-ci nous ont permis de voir les données de la BDD suivant le format JSON. Afin d'y accéder il suffit de suivre les URLs suivants, et voici quelques illustrations :


- <https://localhost:<port>/api/PersonneApi>
- <https://localhost:<port>/api/DefiApi>
- <https://localhost:<port>/api/TropheeApi>



← → ↻ <https://localhost:5001/api/PersonneApi>

 ENSC  Extranet Bordeaux INP  ENT Bordeaux INP  Assistance

JSON Données brutes En-têtes

Enregistrer Copier Tout réduire Tout développer  Filtrer le JSON





▼ 0:

```


id: 1
nom: "Pesquet"
prenom: "Baptiste"
lienPhoto: "Pesquet.jpg"
adresseMail: null
numTel: null
listeDefi: null
listeTrophee: null

```

← → ↻ <https://localhost:5001/api/DefiApi>

 ENSC  Extranet Bordeaux INP  ENT Bordeaux INP  Assistance

JSON Données brutes En-têtes

Enregistrer Copier Tout réduire Tout développer  Filtrer le JSON

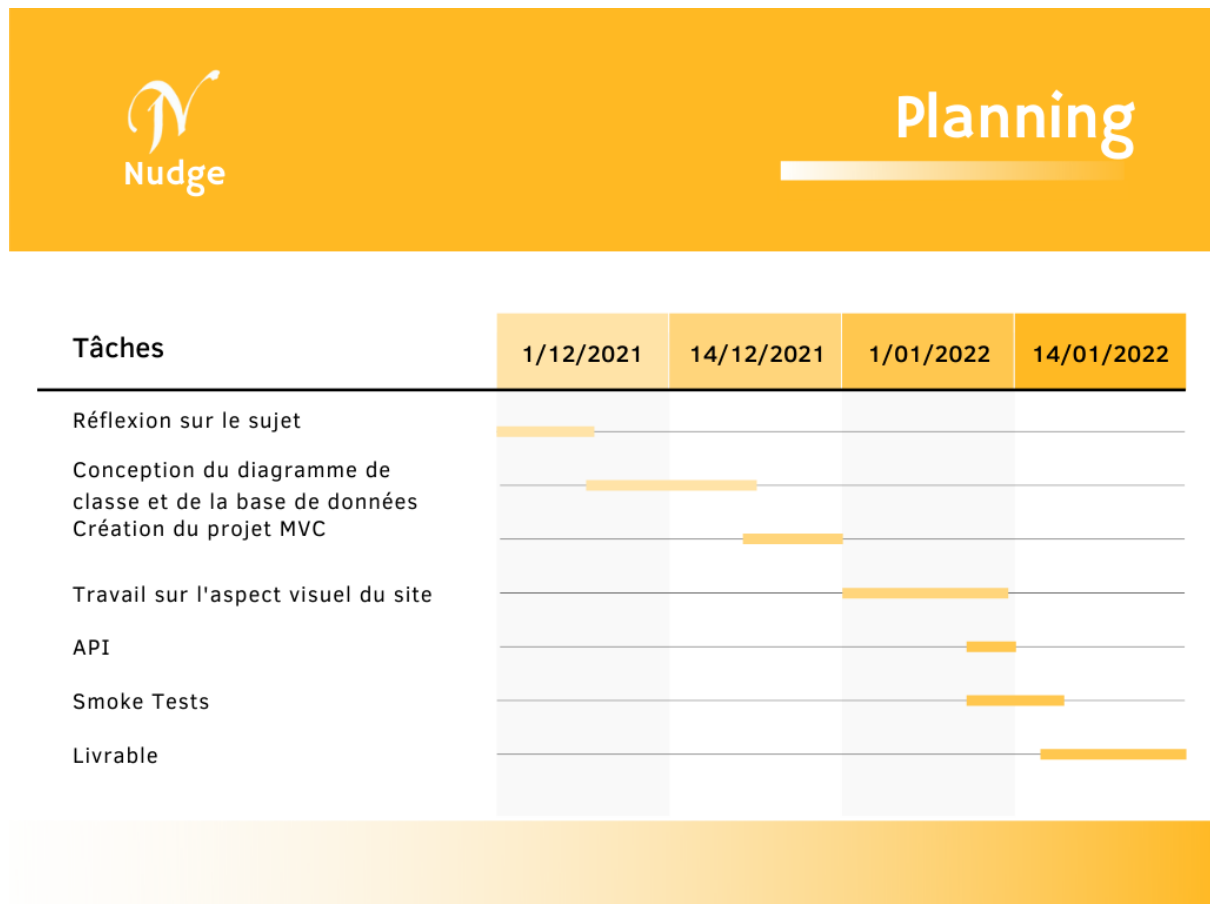
▼ 0:

```

id: 1
personne: null
personneId: 3
intitule: "Corriger le projet"
commentaire: "Bon courage !!"
theme: "Scolaire"
dateDebut: "2022-01-14T00:00:00"
dateFin: "2022-02-24T00:00:00"
progression: 1
reussite: false
tousLesJours: true
repetition: 0

```


IV. Répartition des tâches et planning



Nous avons travaillé en binôme pour le choix du sujet, et toute la partie conceptuelle du site. Une fois le projet généré, nous avons réparti les tâches de la manière suivante : Adrien s'est occupé de l'affichage des défis et des joueurs et plus particulièrement de l'aspect visuel en général, Nanor a fait le lien entre les trophées, les joueurs et les défis, puis les API. Nous nous sommes ensuite consacrés aux smoke tests et nous avons fini en travaillant sur le livrable.

V. Tests automatisés

Nous avons développé une série de classes de smoke tests afin de tester les méthodes de chacun des contrôleurs : HomeController, PersonneController, DefiController et TropheeController. Ces tests simulent un client et envoient une requête visant à tester chacune des méthodes des contrôleurs et vérifient si une réponse est présente oui ou non.

Smoke test	Description
HomeControllerTests	Cette classe de test vise à éprouver les méthodes des contrôleurs de la classe Home.
DefiControllerTests	Cette classe de test vise à éprouver les méthodes des contrôleurs de la classe Defi.
PersonneControllerTests	Cette classe de test vise à éprouver les méthodes des contrôleurs de la classe Personne.
TropheeControllerTests	Cette classe de test vise à éprouver les méthodes des contrôleurs de la classe Trophee.

Nous obtenons des résultats similaires pour chacun des tests lancés : les méthodes Index, et Create renvoient une réponse avec cependant des erreurs de type interne (500). Cependant les méthodes Edit, Details et Delete ne renvoient pas de réponses et génèrent des erreurs de type 404.

Nous avons tenté de comprendre l'origine des erreurs pendant un long moment en vain. Le site fonctionnait très bien ainsi que toutes les requêtes, nous n'avons pas réussi à résoudre le problème.

Vous trouverez ci-dessous un exemple de l'un de nos tests : la classe HomeControllerTests qui envoie une requête à chaque méthode de notre contrôleur Home. Comme vous pouvez le voir sur la capture d'écran du terminal, chaque requête (4) a obtenu une réponse.

```
PS C:\DossiersGit\Nudge.Tests> dotnet test
  Identification des projets à restaurer...
  Tous les projets sont à jour pour la restauration.
  Nudge -> C:\DossiersGit\Nudge\bin\Debug\net5.0\Nudge.dll
  Nudge -> C:\DossiersGit\Nudge\bin\Debug\net5.0\Nudge.Views.dll
  Nudge.Tests -> C:\DossiersGit\Nudge.Tests\bin\Debug\net5.0\Nudge.Tests.dll
  Série de tests pour C:\DossiersGit\Nudge.Tests\bin\Debug\net5.0\Nudge.Tests.dll (.NETCoreApp,Version=v5.0)
  Outil en ligne de commande d'exécution de tests Microsoft (R), version 16.10.0
  Copyright (c) Microsoft Corporation. Tous droits réservés.

  Démarrage de l'exécution de tests, patientez...
  Au total, 1 fichiers de test ont correspondu au modèle spécifié.

  Réussi! - échec :    0, réussite :    4, ignorée(s) :    0, total :    4, durée : 628 ms - Nudge.Tests.dll (net5.0)
  PS C:\DossiersGit\Nudge.Tests> █
```

Voici un exemple de notre classe de test du contrôleur Home :

```

0 references | Run All Tests | Debug All Tests
public class HomeControllerTests
: IClassFixture<WebApplicationFactory<Nudge.Startup>>
{
    2 references
    private readonly WebApplicationFactory<Nudge.Startup> _factory;

    0 references
    public HomeControllerTests(WebApplicationFactory<Nudge.Startup> factory)
    {
        _factory = factory;
    }

    [Theory]
    [InlineData("/")]
    [InlineData("/Home/Index")]
    [InlineData("/Home/Credit")]

    0 references | Run Test | Debug Test
    public async Task Get_EndpointsReturnSuccessAndCorrectContentType(string url)
    {
        // Arrange
        var client = _factory.CreateClient();

        // Act
        var response = await client.GetAsync(url);

        // Assert
        response.EnsureSuccessStatusCode(); // Status Code 200-299
        Assert.Equal("text/html; charset=utf-8",
            response.Content.Headers.ContentType.ToString());
    }
}

```

VI. Bilan et perspectives

Travailler avec ASP.NET core s'est avéré relativement complexe au début mais au fur et à mesure, nous avons saisi l'intérêt de cet outil, qui permet d'avoir une vision globale de son projet et de centraliser de nombreuses fonctionnalités (base de données, classes et méthodes, visuels) au sein d'un même environnement de travail. De plus, il permet d'économiser beaucoup de temps en proposant une structure de site fonctionnelle. Nous avons beaucoup aimé développer ce projet autour de la notion de défi et pensons qu'une application mobile viendra parfaitement compléter cette expérience et pourrait susciter un réel intérêt.

Nous aurions souhaité mettre en place un système d'authentification mais cela requiert l'utilisation d'Identity que nous n'avons encore jamais pris en main, ce qui était difficile au vu du temps que nous avions. Nous avons également essayé de faire cela par le biais d'un système d'identification et de vérification des identifiants dans la BDD, cela fonctionnait, mais il n'était pas possible de maintenir et la connexion en fonction d'un profil utilisateur donné (ou bien nous n'avons pas réussi avec la logique MVC) c'est pourquoi nous sommes restés sur un schéma plus "accessible à tous les utilisateurs".