2A - Groupe 3

BORDEAUX INP Ensc

Rapport du projet de

Génie logiciel

AUDEBERT Alex

BIRET Margo

8 Janvier 2023

I. Introduction

Aujourd'hui à l'école, la vie étudiante est organisée sur de nombreuses plateformes (mail, drive, réseaux sociaux) et il est facile de passer à côté d'informations importantes. Le but de notre projet dans le module génie logiciel est de centraliser l'information dans un site web et de donner accès aux différentes données à travers une API.

Pour centraliser les informations de la vie étudiante de l'ENSC, nous avons besoin de visualiser les différents bureaux et clubs existants, les membres qui les composent ainsi que les événements qu'ils organisent. Le but est qu'un élève de l'ENSC soit au courant de tous les événements passés et futurs et qu'il connaisse les membres de chaque bureau pour pouvoir les contacter.

II. Conception

A. Cas d'utilisation

- Un nouveau bureau veut naître à l'ENSC, le bureau des sports. L'utilisateur veut donc créer ce bureau. Pour que tous comprennent son rôle et ses objectifs, il renseigne une description et bien sûr il lui choisit un nom.
- Il a maintenant besoin d'ajouter des membres à son bureau. Pour cela, il choisit parmi les étudiants de l'ENSC et assigne un rôle.
- Parmis les rôles il ne trouve pas le "responsable mascotte", il doit donc le créer.
- Il remarque que dans la liste d'étudiants "Hugo Richard" n'est plus à l'école, il décide donc de le supprimer de la liste.
- L'utilisateur veut ajouter l'événement qu'il a choisi pour la semaine prochaine. Il renseigne bien que c'est un événement pour son bureau "BDS", ajoute une description avec toutes les informations pratiques et renseigne la date de l'événement.
- L'utilisateur veut vérifier qu'il n'y a pas un autre événement à la date, il va consulter tous les événements pas encore passés et remarque que le BDE à un événement. Il va voir sa page et remarque que c'est Mathys son président. Il décide de le contacter hors de l'application pour voir comment s'arranger.
- Il se demande s'il a assez de membres pour s'organiser. Il va voir combien de membres ont le BDE. Il veut de plus voir qui est leur responsable "coordination" pour le mettre en contact avec le sien.
- L'utilisateur veut créer une application mobile contenant toutes les informations du site, il a besoin d'ajouter, supprimer et modifier
    - Un groupe
    - Un membre

- Un événement
- Un rôle
- Un étudiant

B. Contraintes

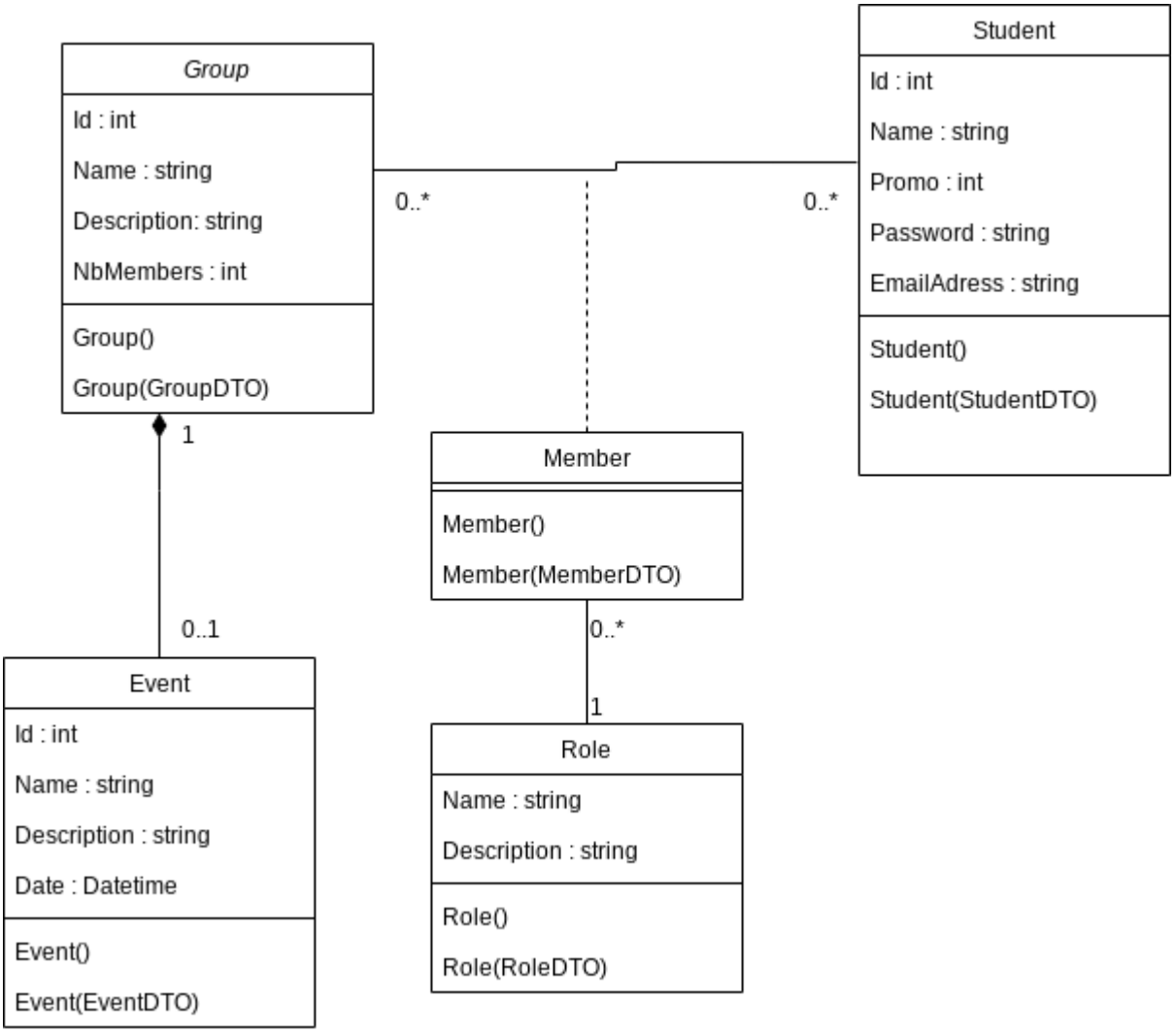
- L'application est réalisée à l'aide de la technologie ASP.NET Core MVC
- L'application utilise la version 6 du framework .NET.
- Les données persistantes sont stockées dans une base de données relationnelle SQLite.
- L'interface entre les classes métier et la SGBDR exploite l'outil Entity Framework Core.

- La création de sessions ainsi que de comptes étant difficile, nous avons dû contourner le problème. En effet, nous souhaitions à l'origine que chaque élève se connecte et puisse voir certaines choses ou non selon son rôle (par exemple des évènements "réunions" accessibles seulement aux élèves du club concerné).

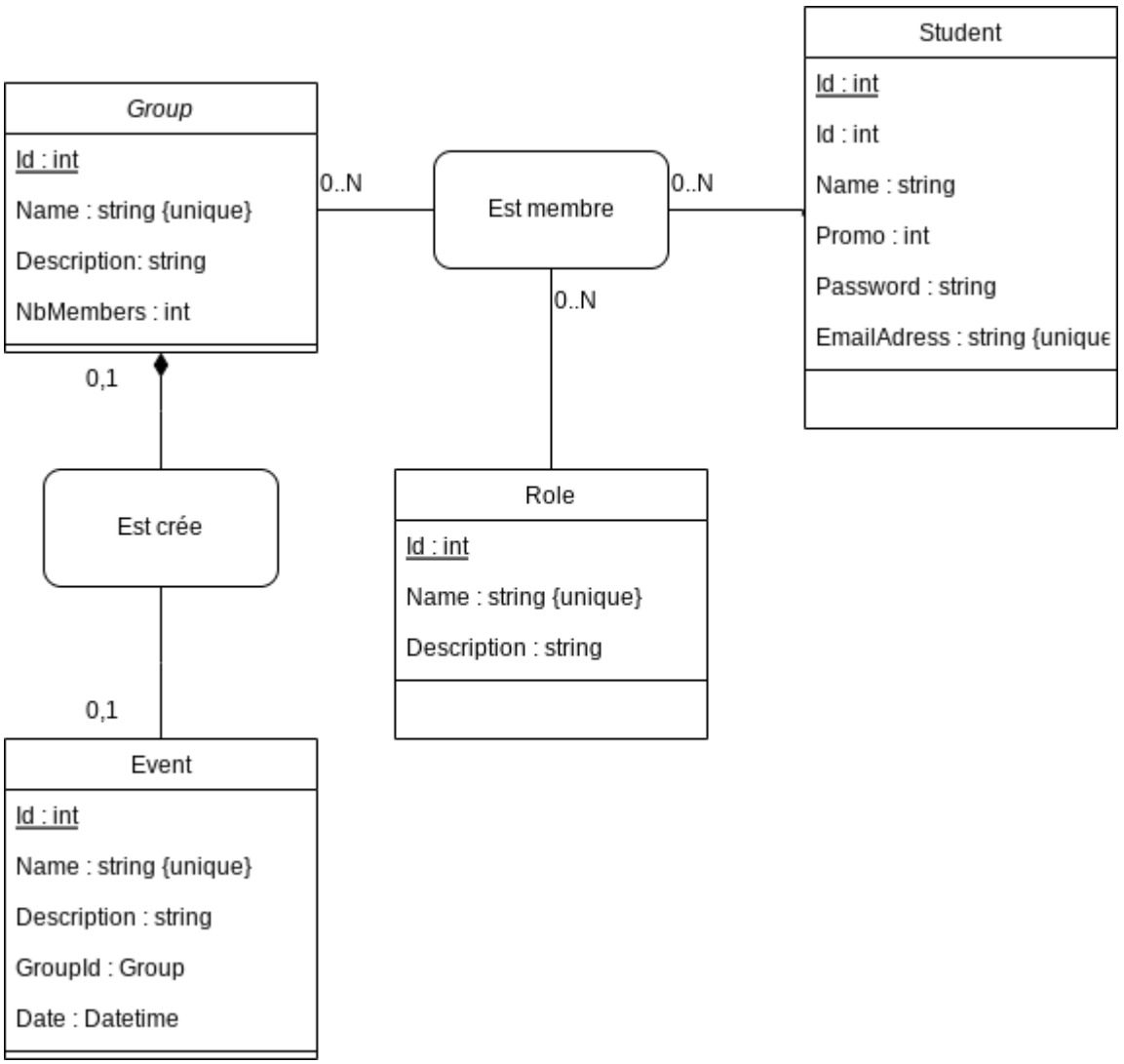
III. Réalisation technique

A. Modélisation des données

Pour commencer nous avons réalisé un diagramme de classe :



Qui nous a ensuite permis de déboucher sur un MCD et son modèle relationnel :



Student (Id,Name,Promo,Password,EmailAdress)
Group (Id,Name,Description,NbMembers,)
Event(Id,Name,Description,DateTime,#GroupId)
Role(Id,Name,Description)
Member(#IdGroup,#IdStudent, #IdRole)

La clé étrangère GroupId dans Event fait référence à la clé primaire Id de la table Group.
La clé étrangère IdGroup de Member fait référence à la clé primaire Id de la table Group.
La clé étrangère IdStudent de Member fait référence à la clé primaire Id de la table Student.

B. API

Les points d'entrée de l'API sont répartis sur différents controller selon l'objet qu'ils traitent.

- Pour les événements on peut accéder aux fonctions suivantes :
    - GetEvents : renvoie tous les évènements sous forme d'une liste.
    - GetEvent : renvoie uniquement l'événement dont l'Id a été donné, ainsi que les informations sur le groupe qui a créé cet événement.
    - UpdateEvent : modifie un événement qui existe déjà dans la base de données à partir de son Id.
    - CreateEvent : crée un nouvel événement. Une fois celui-ci créé, son détail nous est renvoyé.
    - DeleteEvent : supprime un événement existant à partir de son Id.

- Pour les clubs les fonctions suivantes sont disponibles :
    - GetGroups : renvoie tous les évènements sous forme d'une liste.
    - GetGroup : renvoie tout le détail du groupe dont l'Id a été envoyé, notamment les événements créés et les membres.
    - UpdateGroup : modifie un groupe selon les données transmises en format JSON.
    - CreateGroup : crée un groupe avec les données passées en entrée.
    - DeleteGroup : supprime le groupe dont l'Id est transmise.
- Pour les étudiants, les fonctions suivantes sont disponibles :
    - GetStudents : renvoie tous les étudiants sous forme d'une liste.
    - UpdateStudent : modifie un étudiant selon les données transmises en format JSON.
    - CreateStudent : crée un étudiant avec les données passées en entrée.
    - DeleteStudent : supprime l'étudiant dont l'Id est transmise.
- Pour les rôles,, les fonctions suivantes sont disponibles :
    - GetRoles : renvoie tous les étudiants sous forme d'une liste.
    - GetRole : renvoie tout le détail du groupe dont l'Id a été envoyé.
    - UpdateRole : modifie un rôle selon les données transmises en format JSON.
    - CreateRole : crée un rôle avec les données passées en entrée.
    - DeleteRole : supprime le rôle dont l'Id est transmise.
- Pour les membres,, les fonctions suivantes sont disponibles :
    - GetMembers : renvoie tous les étudiants membres d'un group sous forme d'une liste.
    - GetMembersByGroup : renvoie tous les étudiants membre du group dont l'Id est passé en entrée.
    - CreateMember : crée un membre avec les données passées en entrée.
    - DeleteMember : supprime le membre dont l'Id est transmise.

C. Procédure d'installation et tests

Afin d'installer notre application web il faut tout d'abord la téléchrager depuis GitHub avec *git clone*. Ensuite il est nécessaire d'ajouter EntityFrameworkCore à l'aide des deux commandes suivantes :

- *dotnet add package Microsoft.EntityFrameworkCore.Design*
- *dotnet add package Microsoft.EntityFrameworkCore.Sqlite*

Enfin, il faut lancer l'application en exécutant la commande *dotnet run*.

Pour s'assurer que notre API fonctionnait correctement nous avons réalisé de nombreux tests à l'aide de Postman. Voici un tableau où chacun d'entre eux est résumé à l'aide du lien et d'une donnée à insérer quand c'est nécessaire.

	Fonction	Méthode	Lien	Données
Event	UdpateEvent	PUT	https://localhost:7179/api/EventApi/2	*{"id":1, "date": "2022-11-27T00:00:00", "name": "Interpromo", "description": "Tournoi sportif entre les promos ! Pense à amener ta gourde ;) ", "groupId": 1}*
	CreateEvent	POST	https://localhost:7179/api/EventApi	*{"date": "2023-01-25T00:00:00", "name": "Patinoire", "description": "Prêt à glisser comme un pingouin ?", "groupId": 1}*
	GetEvent	GET	https://localhost:7179/api/EventApi/1	
	GetEvents	GET	https://localhost:7179/api/EventApi	
	DeleteEvent	DELETE	https://localhost:7179/api/EventApi/1	

Group	UpdateGroup	PUT	https://localhost:7179/api/GroupApi/1	*{"id":1,* *"name": "BDS",* *"description": "Meilleur bureau, on est là pour te faire transpirer !"}*
	CreateGroup	POST	https://localhost:7179/api/GroupApi	*{"name": "Terra Terre",* *"description": "Le club écologique et humanitaire <3"}*
	GetGroups	GET	https://localhost:7179/api/GroupApi/GetGroups	
	GetGroup	GET	https://localhost:7179/api/GroupApi/1	
	DeleteGroup	DELETE	https://localhost:7179/api/GroupApi/3	
Student	GetStudents	GET	https://localhost:7179/api/StudentApi/GetStudents	
	CreateStudent	POST	https://localhost:7179/api/StudentApi	*{"name": "Jdupont",* *"emailadress": "jdupont@ensc.fr",* *"promo":2025}*
	DeleteStudent	DELETE	https://localhost:7179/api/StudentApi/5	
	UpdateStudent	PUT	https://localhost:7179/api/StudentApi/1	*{"id":1,* *"name": "Alaudebert",* *"emailadress":* "alaudebert@ensc.fr", *"promo":2023}*
Role	GetRole	GET	https://localhost:7179/api/RoleApi/2	
	GetRoles	GET	https://localhost:7179/api/RoleApi/GetRoles	
	UpdateRole	PUT	https://localhost:7179/api/RoleApi/1	*{"id":2,* *"name": "Président",*

				"description": "Référent du club"}
	CreateRole	POST	https://localhost:7179/api/RoleApi	*{"name": "Trésorier", "description": "Gère la moula"}*
	DeleteRole	DELETE	https://localhost:7179/api/RoleApi/3	
Member	GetMembers	GET	https://localhost:7179/api/MemberApi	
	GetMembersByGroup	GET	https://localhost:7179/api/MemberApi/1	
	CreateMember	POST	https://localhost:7292/api/MemberApi	*{ "GroupId":"1", "StudentId":"3", "RoleId":"1"}*
	DeleteMember	DELETE	https://localhost:7179/api/MemberApi/2/2	

IV. Organisation et gestion de projet

Pour commencer le projet nous avons réalisé la modélisation des données ensemble. La méthode de pair programming nous a permis de construire les classes ensemble et de réaliser la première migration. Par la suite nous avons séparé le travail en créant des branches sur git.

Le design et l'apparence des pages ont été réfléchis mutuellement. Pendant que la page Home était créée par Alex, Margo s'est occupée de l'affichage des événements. La mise en page réalisée a été ré-utilisée par la suite par Alex pour l'affichage des clubs.

De même pour les formulaires de création : c'est tout d'abord Alex qui en a construit un qui a servi d'exemple pour les autres.

Ci-dessous le planning de réalisation de l'application :

	21/11 au 27/11	28/11 au 04/12	05/12 au 11/12	12/12 au 18/12	19/12 au 25/12	26/12 au 01/01	02/01 au 08/01
Création du projet							

Création des classes et du SeedData							
Vues, Controller et API des events							
Vue page Home							
Vues, Controller et API des clubs							
Vues, Controller et API des étudiants							
Vues, Controller et API des étudiants							
Vues, Controller et API des étudiants							

V. Conclusion et perspectives

Pour conclure, ce projet nous a permis d'approfondir ce que nous avions vu lors du module de génie logiciel. En effet, le passage de nombreuses informations pour les vues web par exemple ont été quelque chose de nouveau pour nous.

Pour la suite, ce projet sera utilisé pour le développement d'une application mobile. Celle-ci nécessitera sûrement des modifications au sein de ce projet, car il a été difficile pour nous de penser à toutes les fonctionnalités dont nous pourrions avoir besoin de mettre dans l'API. Pour autant nous avons tenté d'être le plus exhaustives possible.

Vous devez également rédiger un rapport constitué des éléments suivants (pas nécessairement dans cet ordre) :

- *Procédure d'installation et de test de l'application.*
- ~~*Thématique de l'application.*~~
- *Modélisation des données : diagramme des classes métier et modèle relationnel.*
- *Documentation de l'API (liste des points d'entrée et description des données renvoyées).*
- *Organisation de l'équipe, répartition des tâches dans le groupe et planning.*
- *Bilan et perspectives sur le projet.*

Ce rapport sera ajouté au format PDF à la racine du dépôt GitHub du projet.

1. Organisation

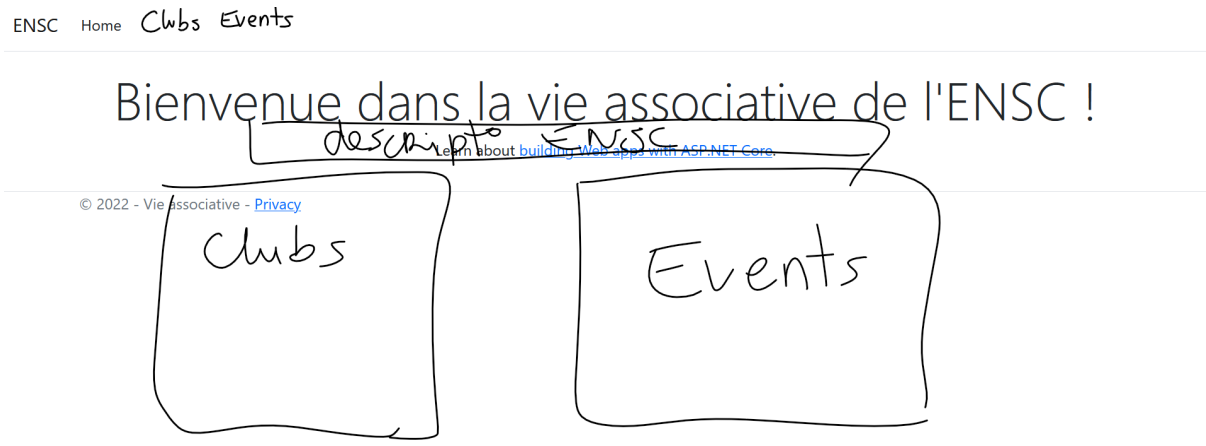
Diagramme de classe	pair programming
ORM	pair programming

2. Explication BDD

Event Visibility	
Description	Pour savoir à qui est destiné l'event
Type	List<Group>
Pourquoi	De base en utilisant l'authentification mais finalement pour créer des filtres

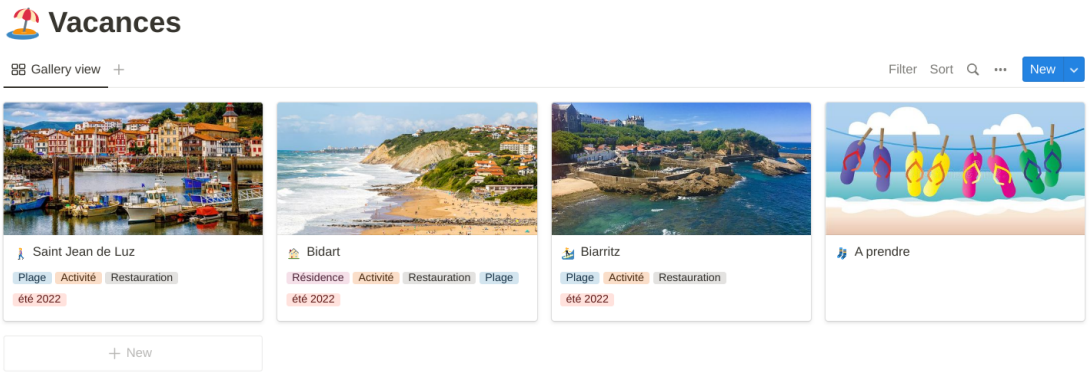
3. Organisation du site

a. Accueil



Dans chaque carte, possibilité de créer ou voir ceux qui existent. Présentation avec une photo, les deux cartes et enfin une description de l'école etc.

b. Présentation des events



Cartes avec photos, bureau organisateur, nom de l'event et date.
Quand on clique on a alors accès à la description, les comm si on en fait

/!\ Penser à mettre le code des requêtes curl dans le rapport /!

Requete API Group

1/ Création :

https://localhost:7292/api/GroupApi
{
"Name": "BDA",
"Description": "On fait de l'art",

}

Résultat :


```
{
  "id": 3,
  "name": "BDA",
  "students": null,
  "nbMembers": 0,
  "events": null,
  "description": "On fait de l'art"
}
```


https://localhost:7292/api/StudentApi/getStudents

```
[
  {
    "id": 1,
    "name": "Alaudebert",
    "emailAdress": "alaudebert@ensc.fr",
    "password": null,
    "promo": 2024,
    "groups": null,
    "group": null
  },
  {
    "id": 2,
    "name": "Mbiret",
    "emailAdress": "mbiret@ensc.fr",
    "password": null,
    "promo": 2024,
    "groups": null,
    "group": null
  },
  {
    "id": 3,
    "name": "CLasserre",
    "emailAdress": "classerre@ensc.fr",
    "password": null,
    "promo": 2025,
    "groups": null,
    "group": null
  },
```


https://localhost:7292/api/StudentApi

{
"Name": "Lubin",
"EmailAdress": "On fait de l'art",
"Promo":"2011"
}

```
{
  "id": 6,
  "name": "Lubin",
  "emailAdress": "On fait de l'art",
  "password": null,
  "promo": 2011,
  "groups": null,
  "group": null
}
```