



DOSSIER PROFESSIONNEL (DP)

Nom de naissance ▶ Cottet.
Nom d'usage ▶ Cottet.
Prénom ▶ Mathis.
Adresse ▶ 24 rue d'hozier 13002 Marseille.

Titre professionnel visé

Concepteur, développeur d'application.

MODALITÉ D'ACCÈS :

- ☐ Parcours de formation
- ☐ Validation des Acquis de l'Expérience (VAE)

Présentation du dossier

Le dossier professionnel (DP) constitue un élément du système de validation du titre professionnel.
Ce titre est délivré par le Ministère chargé de l'emploi.

Le DP appartient au candidat. Il le conserve, l'actualise durant son parcours et le présente **obligatoirement à chaque session d'examen.**

Pour rédiger le DP, le candidat peut être aidé par un formateur ou par un accompagnateur VAE.

Il est consulté par le jury au moment de la session d'examen.

Pour prendre sa décision, le jury dispose :

1. des résultats de la mise en situation professionnelle complétés, éventuellement, du questionnaire professionnel ou de l'entretien professionnel ou de l'entretien technique ou du questionnement à partir de productions.
2. du **Dossier Professionnel (DP)** dans lequel le candidat a consigné les preuves de sa pratique professionnelle.
3. des résultats des évaluations passées en cours de formation lorsque le candidat évalué est issu d'un parcours de formation
4. de l'entretien final (dans le cadre de la session titre).

[Arrêté du 22 décembre 2015, relatif aux conditions de délivrance des titres professionnels du ministère chargé de l'Emploi]

Ce dossier comporte :

- pour chaque activité-type du titre visé, un à trois exemples de pratique professionnelle ;
- un tableau à renseigner si le candidat souhaite porter à la connaissance du jury la détention d'un titre, d'un diplôme, d'un certificat de qualification professionnelle (CQP) ou des attestations de formation ;
- une déclaration sur l'honneur à compléter et à signer ;
- des documents illustrant la pratique professionnelle du candidat (facultatif)
- des annexes, si nécessaire.

DOSSIER PROFESSIONNEL ^(DP)

Pour compléter ce dossier, le candidat dispose d'un site web en accès libre sur le site.

 <http://travail-emploi.gouv.fr/titres-professionnels>

Sommaire

Exemples de pratique professionnelle

Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité	p.	5
- Maquetter une application	p. p.	6-7
- Développer une interface de type desktop	p. p.	8-11
- Développer des composants d'accès aux données	p p.	11-13
- Développer la partie front-end d'une interface utilisateur web		14-16
- Développer la partie back-end d'une interface utilisateur web		16-18
Concevoir et développer la persistance des données en intégrant les recommandations de sécurité	p.	19
- Concevoir une base de données	p. p.	19-21
- Intégrer les recommandations de sécurité	p. p.	21-25
- Développer des composants dans le langage d'une base données	p p.	19-21
Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité	p.	26
- Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement	p. p.	26-28
- Concevoir une application	p. p.	34-36
- Développer des composants métier	p p.	28-30
- Construire une application organisée en couches		30-33
- Développer une application mobile		34-36
- Préparer et exécuter les plans de tests d'une application		36-38
- Préparer et exécuter le déploiement d'une application		40-41

DOSSIER PROFESSIONNEL ^(DP)

Titres, diplômes, CQP, attestations de formation *(facultatif)*

p.

Déclaration sur l'honneur

p.

Documents illustrant la pratique professionnelle *(facultatif)*

p.

Annexes *(Si le RC le prévoit)*

p.

45-58

EXEMPLES DE PRATIQUE PROFESSIONNELLE

Activité-type 1

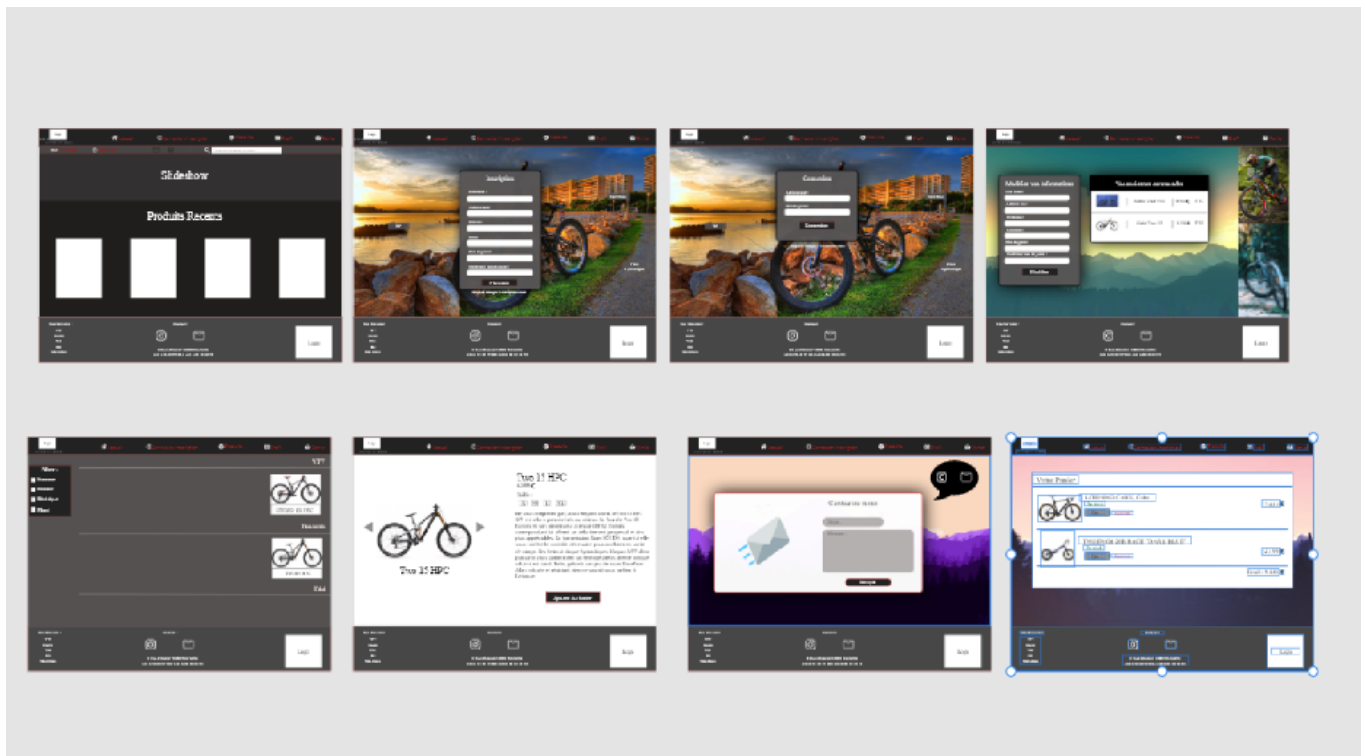
Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité

Exemple n°1 - Maquetter une application

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Après avoir réfléchi sur le concept de notre site web, il a fallu réfléchir aux pages nécessaires au bon fonctionnement de notre future application.

Nous avons alors réalisé une maquette de l'application, Chaque page a été maquetée afin d'avoir une vision d'ensemble sur la structure de l'application:



DOSSIER PROFESSIONNEL ^(DP)

- Nous avons créé un logo pour l'application:



- Tous les composants comme les boutons pour naviguer, les boutons pour ajouter au panier/voir plus, les entrées utilisateur, auront un style fixe et défini à l'avance.



Pour nous aider nous avons utilisé bootstrap. Bootstrap est une collection d'outils utiles à la création du design de sites et d'applications web. C'est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres éléments interactifs, ainsi que des extensions JavaScript en option.

2. Précisez les moyens utilisés :

Nous avons réalisé la maquette avec l'outil faisant partie de la suite adobe : adobe XD.

3. Avec qui avez-vous travaillé ?

Nous avons collaboré à 2 pour le maquetage.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ *La plateforme.*

Chantier, atelier, service ▶ *Projet professionnel.*

Période d'exercice ▶ Du : *mars 2021* au : *Juin 2022*

5. Informations complémentaires (facultatif)

Activité-type 1 Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité

Exemple n°1 - Développer une interface de type desktop

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

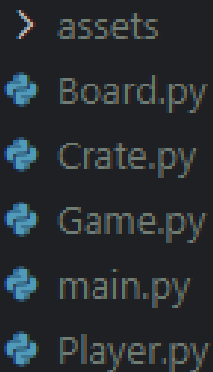
Pour développer une application de type desktop j'ai décidé de faire le jeu du sokoban en python. Tout d'abord je me suis renseigné plus amplement sur le fonctionnement exact du jeu du sokoban. Puis une fois cela fait j'ai consulté la documentation officiel python. Ensuite j'ai procédé à l'installation de python 5. Je me suis ensuite penché sur le design que j'allais donner à mon sokoban. Mon choix s'est porté sur le thème des Simpson. Puis je suis passé au développement du mini-jeu.

Plusieurs choix s'offraient à moi pour la définition de la map, du joueur, des caisses et des objectifs à atteindre. J'ai décidé de fonctionner avec un tableau qui se redessine à chaque action.

```
board = [  
    [1, 1, 0, 0, 0, 1, 1, 1, 1, 1],  
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 1],  
    [1, 0, 0, 0, 0, 0, 0, 0, 4, 0],  
    [0, 0, 0, 0, 4, 0, 0, 0, 0, 0],  
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
    [1, 0, 0, 2, 0, 3, 0, 0, 0, 0],  
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 1],  
    [0, 0, 0, 3, 0, 0, 3, 0, 4, 1],  
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 1],  
    [1, 1, 0, 1, 1, 1, 1, 1, 1, 1]  
]
```

Ensuite j'ai décidé de structurer mon code de façon à ce que chaque partie ait son propre fichier.

De la façon suivante :



```
> assets  
Board.py  
Crate.py  
Game.py  
main.py  
Player.py
```

Dans le dossier asset nous allons retrouver les images qui composent la partie graphique de mon sokoban. Dans le fichier board.py je déclare le tableau (taille..). Dans le fichier crate.py je déclare la caisse a poussée et les mouvements associée à celle-ci. Je check aussi si une action se passe par rapport à elle. Dans le fichier game.py je vérifie si le jeu est lancé, je défini la taille de la fenêtre, j'attribue le terrain et le joueur. Puis dans une autre fonction je lance le jeu et je charge les assets puis je gérer les événement de jeu avec l'aide de pygame. Pygame est une bibliothèque libre multiplate-forme qui facilite le développement de jeux vidéo temps réel avec le langage de programmation Python. Dans le fichier main.py j'importe mon game.py qui contient mon player.py et mon board.py ainsi que pygame et je dessine mon tableau d'origine. Dans le fichier player.py je déclare l'attribut qui stock la position du joueur. Ensuite je fais ma fonction pour les mouvements du joueur et je check s' il y a des actions. Ensuite je gère les mouvements du joueur avec la caisse.

Mon Sokoban final ressemble à cela :



DOSSIER PROFESSIONNEL ^(DP)

2. Précisez les moyens utilisés :

Pour réaliser ce sokoban j'ai eu besoin de python et de pygame.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seul sur ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ *La plateforme.*

Chantier, atelier, service ▶ *Réaliser une application desktop.*

Période d'exercice ▶ Du : *9/05/2022* au : *14/05/2022*

5. Informations complémentaires (facultatif)

Dans une version ultérieure le jeu sera amélioré et pourra être lancé en .exe.

Activité-type 1

Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité

Exemple n°1 ▶ Développer des composants d'accès aux données

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Dans le cadre du développement de notre API nous avons décidé d'utiliser NestJS (nodejs). A l'aide de lignes de commande nous avons pu générer nos premiers fichiers pour créer tout ce qui servira au routage et aux services. Avec le temps nous avons réussi à nous approprier la logique et le fonctionnement du framework. Avec la compréhension des DTO (Data Transfert Object) nous avons pu configurer des pipes basiques pour

contenir et vérifier nos données avant de les passer à des Controller. Cela nous a aussi permis de configurer correctement SWAGGER pour pouvoir tester convenablement notre API et plus rapidement qu'avec POSTMAN. A l'aide de TypeOrm nous pouvons gérer nos entités, configurer les champs attendus, lier les entités et personnaliser leurs requêtes en fonction des comportements attendus.

Nous avons réussi à maîtriser comme il se doit toute la partie des services de NestJS. Cela nous a permis de personnaliser complètement les appels de notre API. Elle peut ainsi être appelée par n'importe quel champ sur n'importe quelle table. Certains champs en appellent d'autres grâce à des jointure qui sont effectuées en fonction de certains paramètres.

Notre API décode également les tokens qu'elle reçoit pour vérifier leur conformité. Dans une version future, l'API pourra posséder un système de rôle qui permettrait aux utilisateurs d'accéder aux services (routes) de l'API en fonction du rôle qui est contenu dans leur token. Actuellement n'importe qui peut accéder et utiliser l'API. Avec un système de rôle cela la rendrait complètement hermétique aux actions indésirables.

Dans un autre domaine nous avons créé un serveur socket.io en nodejs qui nous permet de communiquer avec la partie front de notre application dans le cadre d'échange dynamique de signaux et de data légères. Pour implémenter socket.io sur un serveur nodejs il nous a fallu installer CORS, axios (accéder à notre API), express (partitionner notre serveur en route) et socket.io (recevoir et émettre des événements). Nous aurions pu directement l'intégrer à notre serveur NestJS mais dans un souci de compréhension nous avons préféré effectuer cette partie a part.

Pour accéder au serveur socket il suffit de l'URL du serveur sur laquelle est hébergée l'API et du numéro de port correspondant au serveur socket. A partir de là, l'utilisateur sera authentifiable par le serveur et pourra accéder au différents événement lui permettant d'émettre et de recevoir des données en temps réel.

DOSSIER PROFESSIONNEL ^(DP)

2. Précisez les moyens utilisés :

NestJS, NodeJs, Npm, Swagger, TypeORM, passport-jwt, class-transformer
Socket.io, axios, cors.

3. Avec qui avez-vous travaillé ?

Nous avons collaboré à 4 sur ce projet : Joris Verguldezoone, Mathis Cottet, Mohamed-marwane Bellagha, Shun Lassal.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ *La plateforme.*

Chantier, atelier, service ▶ *Projet Professionnel.*

Période d'exercice ▶ Du : *decembre 2021* au : *Juillet 2022*

5. Informations complémentaires *(facultatif)*

Activité-type 1

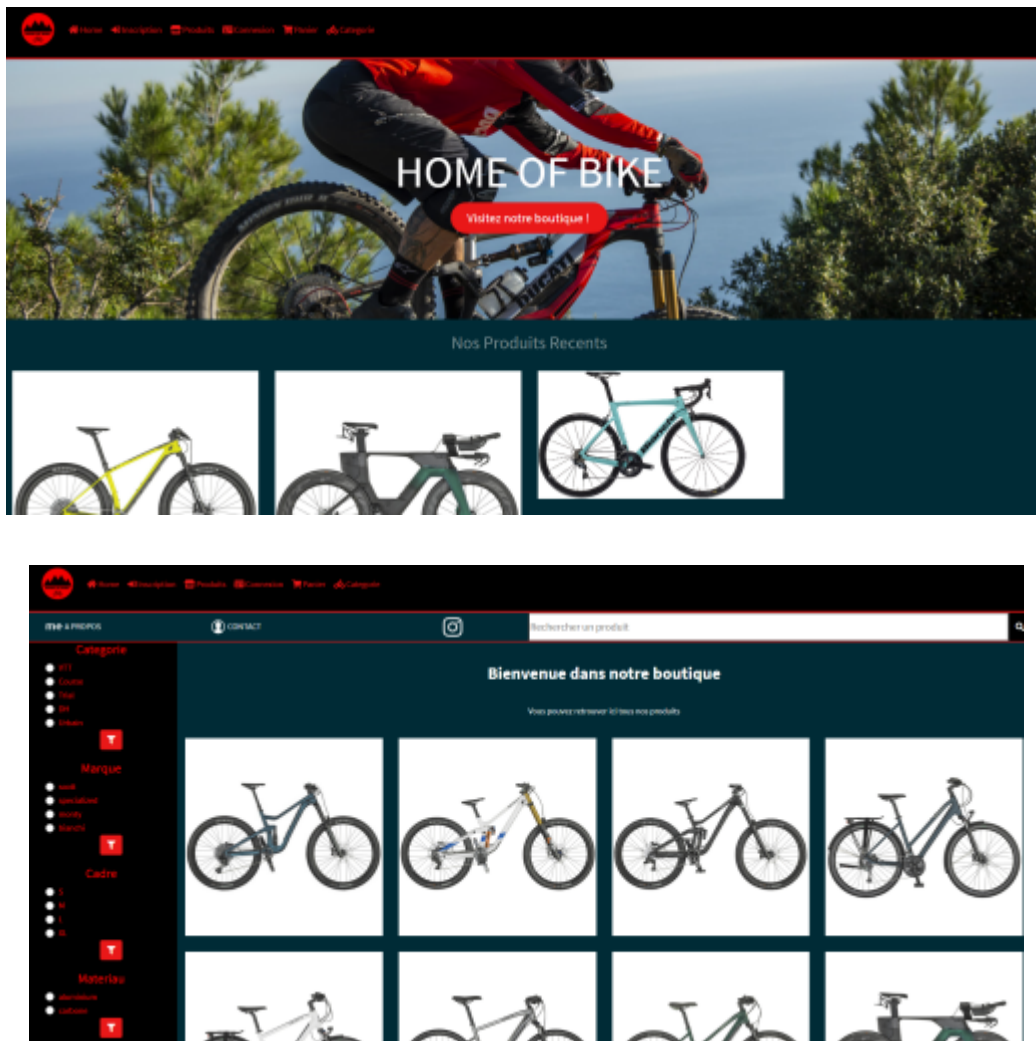
Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité

Exemple n° 2 - Développer la partie front-end d'une interface utilisateur web

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Pour ce projet j'ai décidé tout d'abord de prendre connaissance de l'étendue du travail à effectuer en faisant un wireframe et en choisissant avec mon collaborateur le thème de notre boutique (vente de vélos en ligne). Après le wireframe je fais la maquette du projet pour choisir et nous référer à ce design tout au long du projet. Je fais ensuite le mcd de la base de données. Je crée ensuite la base de données et discute de la maquette terminée avec mon collègue pour la valider. Grâce à cette maquette j'organise le développement des fonctionnalités. J'effectue la structure de mon code (dossiers....). Une fois cette étape effectuée, on se partage le travail par page. Et je commence par faire le HTML et le CSS de mes pages. Puis je fais les fonctionnalités (inscription/ connexion, dashboard admin, récupération des produits en base de données, search bar). J'implémente ensuite le système de paiement via une API (stripe). Pour terminer le projet j'effectue le test de toutes les fonctionnalités et fais le debug si besoin. - Le projet a été réalisé avec les langages suivant: HTML / CSS / PHP / SQL / JAVASCRIPT - Les pages présente sur la boutique: Accueil / inscription / connexion / produits / catégorie / panier / mon profil / paiement

DOSSIER PROFESSIONNEL ^(DP)



2. Précisez les moyens utilisés :

- Le wireframe a été réalisé sur un tableau afin de modifier facilement les éléments.
- La maquette du site a été réalisée sur adobe xd pour un rendu réaliste et complet.
- Le mcd a été réalisée sur lucid chart.
- J'utilise l'IDE Visual Studio Code pour développer mon site.
- Le système de paiement a été ajouté grâce à l'API stripe.
- J'utilise PHPMyAdmin pour ma base de données.
- Je me suis servi de Git et GitHub pour conserver mon travail et "merge" mon travail

DOSSIER PROFESSIONNEL ^(DP)

avec celui de mon collègue.
-Google pour toutes mes recherches.

3. Avec qui avez-vous travaillé ?

Nous avons collaboré à 2 sur ce projet : Mathis Cottet, Mohamed-marwane Bellagha.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ *La Plateforme*
Chantier, atelier, service ▶ *Projet Professionnel*
Période d'exercice ▶ Du : *Décembre 2021* au : *Juillet 2022*

5. Informations complémentaires (facultatif)

Ce projet m'a permis de gagner en compétences au niveau du maquettage, de la création d'un site web dynamique et d'aborder le e-commerce ainsi qu'un système de paiement. Mais aussi sur la sécurité des données utilisateurs. Ce projet me permet de valider la compétence de maquetter une application et développer une interface web utilisateur dynamique. Mais aussi de valider la compétence réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce.

Activité-type 1 Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité

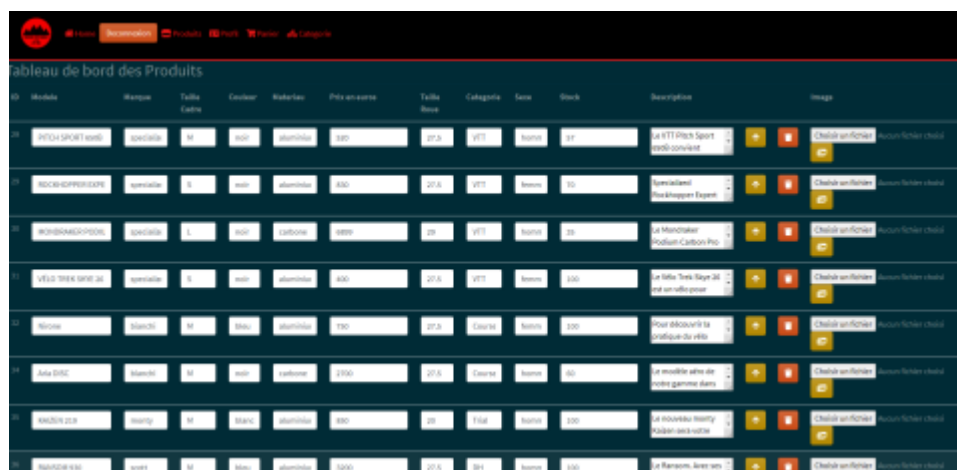
Exemple n° 2 ▶ Développer la partie Back-end d'une interface utilisateur web

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

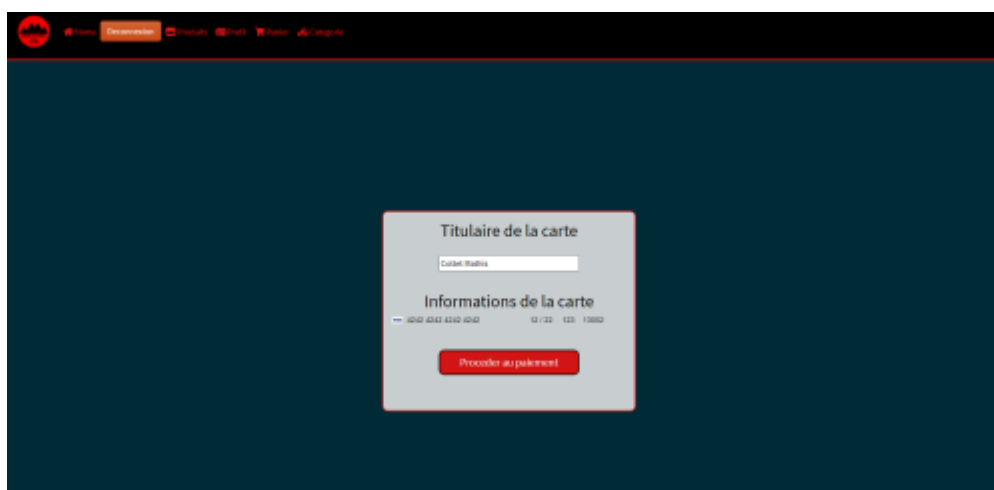
Pour ce projet j'ai décidé tout d'abord de prendre connaissance de l'étendue du travail à effectuer en faisant un wireframe et en choisissant avec mon collaborateur le thème de notre boutique (vente de vélos en ligne). Après le wireframe je fais la maquette du projet pour choisir et nous référer à ce design tout au long du projet. Je fais ensuite le mcd de la base de données. Je crée ensuite la base de données et discute de la maquette terminée avec mon collègue pour la valider. Grâce à cette maquette j'organise le développement des fonctionnalités. J'effectue la structure de mon code (dossiers...). Une fois cette étape effectuée, on se partage le travail par page. Et je commence par faire le HTML et le

DOSSIER PROFESSIONNEL ^(DP)

CSS de mes pages. Puis je fais les fonctionnalités (inscription/ connexion, dashboard admin, récupération des produits en base de données, search bar). J'implémente ensuite le système de paiement via une API (stripe). Pour terminer le projet j'effectue le test de toutes les fonctionnalités et fais le debug si besoin. - Les langages utilisés sont les suivants: HTML / CSS / PHP / JAVASCRIPT / SQL.



ID	Nom	Marque	Taille	Couleur	Matériau	Prix au stock	Taille (cm)	Catégorie	Sexe	Stock	Description	Image
1	PROTECTOR SPORT	PROTECTOR	M	Blanc	Aluminium	1500	17,5	MTB	Homme	10	Le MTB Protector Sport est un vélo de montagne	Choisir un fichier
2	PROTECTOR SPORT	PROTECTOR	M	Blanc	Aluminium	1500	17,5	MTB	Homme	10	Le MTB Protector Sport est un vélo de montagne	Choisir un fichier
3	PROTECTOR SPORT	PROTECTOR	M	Blanc	Aluminium	1500	17,5	MTB	Homme	10	Le MTB Protector Sport est un vélo de montagne	Choisir un fichier
4	PROTECTOR SPORT	PROTECTOR	M	Blanc	Aluminium	1500	17,5	MTB	Homme	10	Le MTB Protector Sport est un vélo de montagne	Choisir un fichier
5	PROTECTOR SPORT	PROTECTOR	M	Blanc	Aluminium	1500	17,5	MTB	Homme	10	Le MTB Protector Sport est un vélo de montagne	Choisir un fichier
6	PROTECTOR SPORT	PROTECTOR	M	Blanc	Aluminium	1500	17,5	MTB	Homme	10	Le MTB Protector Sport est un vélo de montagne	Choisir un fichier
7	PROTECTOR SPORT	PROTECTOR	M	Blanc	Aluminium	1500	17,5	MTB	Homme	10	Le MTB Protector Sport est un vélo de montagne	Choisir un fichier
8	PROTECTOR SPORT	PROTECTOR	M	Blanc	Aluminium	1500	17,5	MTB	Homme	10	Le MTB Protector Sport est un vélo de montagne	Choisir un fichier
9	PROTECTOR SPORT	PROTECTOR	M	Blanc	Aluminium	1500	17,5	MTB	Homme	10	Le MTB Protector Sport est un vélo de montagne	Choisir un fichier



Titulaire de la carte

Cédric Wablia

Informations de la carte

4000 0000 0000 0000 12 / 20 12 / 20 1234

Procéder au paiement

DOSSIER PROFESSIONNEL ^(DP)

2. Précisez les moyens utilisés :

- Le wireframe a été réalisé sur un tableau afin de modifier facilement les éléments.
- La maquette du site à été réalisée sur adobe xd pour un rendu réaliste et complet.
- Le mcd a été réalisé sur lucid chart. -Le site à été fait en PHP, HTML, CSS, SQL, JAVASCRIPT.
- J'utilise l'IDE Visual Studio Code pour développer mon site.
- Le système de paiement a été ajouté grâce à l'API stripe.
- J'utilise PHPMYAdmin pour ma base de données.
- Je me suis servi de Git et GitHub pour conserver mon travail et "merge" mon travail avec celui de mon collègue.

3. Avec qui avez-vous travaillé ?

Nous avons collaboré à 2 sur ce projet : Mathis Cottet, Mohamed-marwane Bellagha.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ *La Plateforme*

Chantier, atelier, service ▶ *Projet Professionnel*

Période d'exercice ▶ Du : *Décembre 2021* au : *Juillet 2022*

5. Informations complémentaires (facultatif)

Activité-type 2

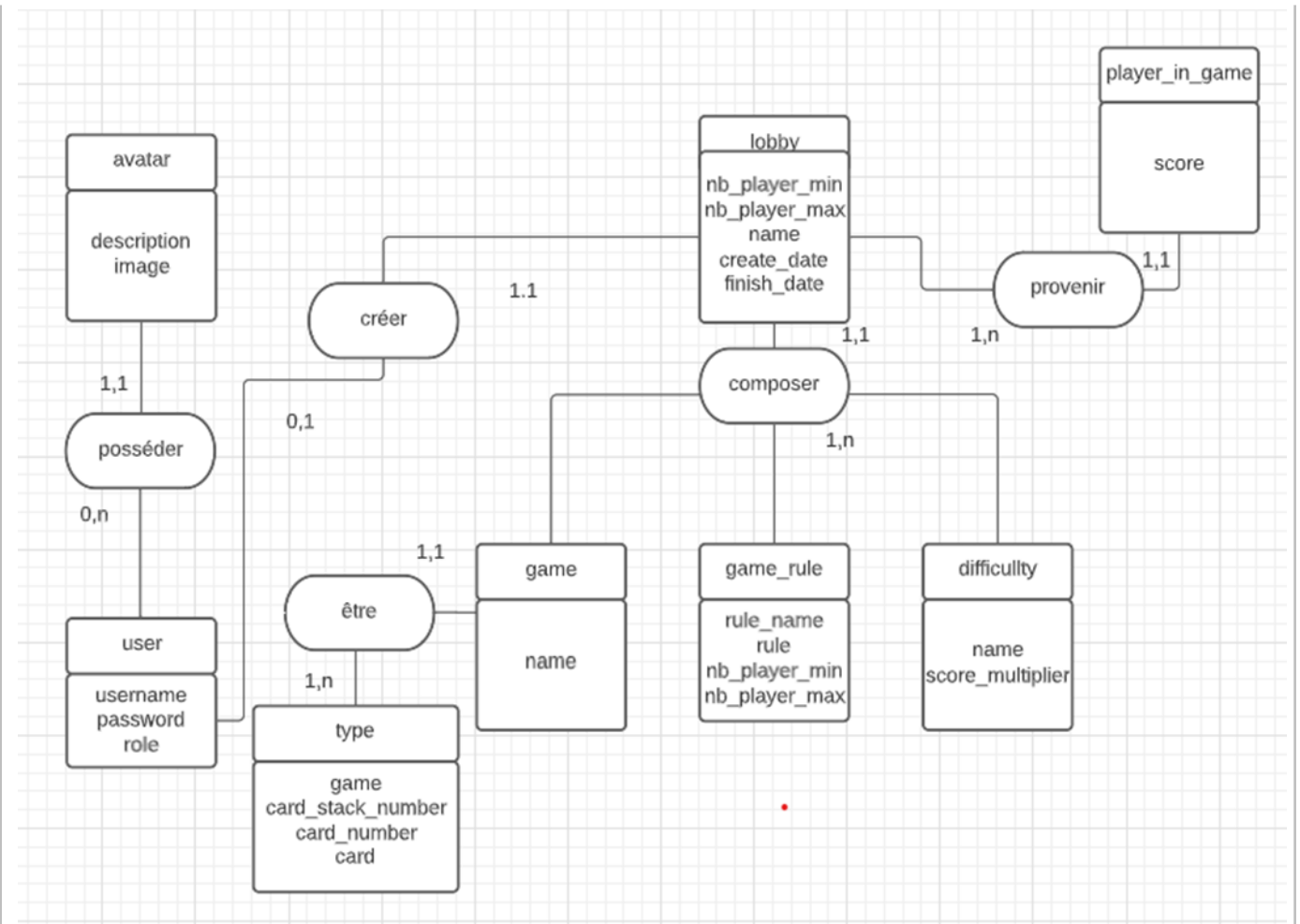
Concevoir et développer la persistance des données en intégrant les recommandations de sécurité

Exemple n° 1 - Concevoir une base de données / Mettre en place une base de données / Développer des composant dans le langage d'une base données

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Dans le cadre de notre projet d'application de jeux de carte mobile, nous avons imaginé un MCD et un MLD qui ont évolué au fil du développement de notre projet. Cette base de données possède une dizaine de tables, la plupart d'entre elles sont contraintes par les cascades via leurs clés étrangères et sont toutes accessibles et testables via notre environnement Swagger.

Avant de pouvoir développer la base de données nous nous sommes concertés avec un papier et un stylo pour imaginer à quoi elle devrait ressembler. En avançant dans le développement nous avons fait évoluer notre base de données pour répondre au nouveau besoin que nous avons rencontré et auxquels il fallait répondre.



Si create_date != null alors ce n'est plus un lobby mais une partie

Si finish_date != null alors la partie est terminée

Si * == null et que le serveur socket ne détecte aucun socket dans la room du lobby alors le lobby se supprimera automatiquement via le serveur socket pour annuler le lobby créer.

2. Précisez les moyens utilisés :

PhpMyAdmin, sql, InnoDB, MariaDB, Swagger, LucidChart

3. Avec qui avez-vous travaillé ?

DOSSIER PROFESSIONNEL ^(DP)

Nous avons collaboré à 4 sur ce projet : Joris Verguldezoone, Mathis Cottet, Mohamed-marwane Bellagha, Shun Lassal.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ *La Plateforme.*

Chantier, atelier, service ▶ *Projet Professionnel.*

Période d'exercice ▶ Du : *Decembre 2021* au : *Juillet 2022*

5. Informations complémentaires (facultatif)

Activité-type 2

Concevoir et développer la persistance des données en intégrant les recommandations de sécurité

Exemple n° 1 ▶ Intégrer les recommandations de sécurité

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Ensuite le problème de la sécurité de notre application s'est imposé. Pour répondre à cela, nous avons mis en place un certain nombre de moyens afin d'assurer une application sécurisée.

Les injections SQL et les failles XSS sont les premiers points de sécurité que nous avons abordés. L'injection SQL est généralement utilisée dans les formulaires présents sur votre site. Le pirate inclura une chaîne de caractère lui permettant de détourner votre requête SQL et ainsi de récupérer les informations de vos utilisateurs et bien d'autres choses que permet d'effectuer le SQL. La faille XSS consiste à injecter un script arbitraire dans une page pour provoquer une action bien définie. Les autres utilisateurs exécutent le script sans s'en rendre compte dès l'ouverture de la page. Cross veut également dire traverser, car l'un des buts de la faille est d'exécuter un script permettant de transmettre des données depuis un site vers un autre. Pour nous protéger de ces deux failles de sécurité

nous avons utilisé DTO (data transfert object). Son but est de simplifier les transferts de données entre les sous-systèmes d'une application logicielle. DTO nous a permis de simplifier et de sécuriser le transfert de données.

```
create(jeux: CreateJeuxDto): Promise<JeuxInterface> {  
  return this.jeuxRepository.save(jeux);  
}  
  
findAll(): Promise<Jeux[]> {  
  return this.jeuxRepository.find();  
}  
  
update(id: number, jeux: UpdateJeuxDto): Promise<any> {  
  return this.jeuxRepository.update(id, jeux);  
}  
  
remove(id: number): Promise<any> {  
  return this.jeuxRepository.delete(id);  
}
```

Dans ce screenshot on peut apercevoir la déclaration de plusieurs fonctions. Celles-ci sont contraintes par TypeScript et par l'usage des DTO. Si TypeScript ne rencontre pas l'objet demandé, il enverra une erreur. Si l'objet reçu ne correspond pas aux données l'interface alors la fonction ne pourra pas être exécutée.

Si les données ne correspondent pas aux contraintes contenues dans les DTO alors la fonction ne s'exécutera toujours pas.

Par exemple :

Ici, pour poster un nouveau jeu, il faut que les champs soit rempli et du bon type. Il faudra aussi que le nom des champs correspondent à ceux dans l'interface. Une fois toutes ces contraintes respectées, le nouveau jeu pourra être posté.

```
api_back > src > jeux > dto > TS create-jeux.dto.ts > ...
1  import { IsNotEmpty, IsNumber, IsString } from 'class-validator';
2  import { ApiProperty, PartialType } from '@nestjs/swagger';
3  import { Type } from 'class-transformer';
4  import { GetJeuxDto } from './get-jeux.dto';
5
6  export class CreateJeuxDto extends PartialType(GetJeuxDto) {
7
8      @IsNotEmpty()
9      @ApiProperty()
10     @Type(() => String)
11     nom: string;
12
13     @IsNotEmpty()
14     @ApiProperty()
15     @Type(() => Number)
16     idtype: number;
17 }
```

(a noter que ApiProperty sert à configurer Swagger (testeur d'api))

Ensuite nous nous sommes concentrés sur la faille CSRF (Cross-site request forgery). Il s'agit d'effectuer une action visant un site ou une page précise en utilisant l'utilisateur comme déclencheur, sans qu'il en ait conscience. On va deviner un lien qu'un utilisateur obtient habituellement, et tout simplement faire en sorte qu'il clique lui-même sur ce lien. Pour pallier ce problème un système de token a été mis en place. Grâce au JWT (Json Web Token). Il permet l'échange sécurisé de jetons (token) entre plusieurs parties. Cette sécurité se traduit par la vérification de l'intégrité et de l'authenticité des données. Un JWT se structure de la façon suivante :

- Un en-tête (header) : utilisé pour décrire le jeton (objet json).
- Une charge utile (payload) : représente les informations embarquées dans le jeton (objet json).
- Une signature numérique : générée à partir du payload et d'un algorithme.

```
const payload = {  
  sub: user.id,  
  username: user.username,  
  idavatar: user.idavatar,  
  role: user.role,  
  expiresIn: ''  
};  
return {  
  access_token: this.jwtService.sign(payload),  
};
```

Ici nous définissons le payload il est propre à ce que désire le développeur. Dans notre projet nous avons décidé de définir la payload avec l'id utilisateur, l'username, son avatar et son rôle. Toutes ces informations sont prises en base de données. Puis nous définissons la durée de validité du token (expiresIn). Puis grâce à jwtService nous formons le token final qui comportera les 3 grandes parties évoquées au-dessus (header, payload et signature). La fonction sign nous permet de créer la signature à partir ici du payload.

Pour finir nous nous sommes concentrés sur la faille upload puisque l'utilisateur aura la possibilité d'uploader son image de profil. Le principe de l'attaque est très simple. Le pirate essaie d'uploader un fichier qui contient du code malveillant ou un code PHP de sa création. Si la faille est là alors le fichier finira pas atterrir sur le serveur. Il suffit ensuite au pirate d'appeler son fichier pour que celui-ci s'exécute. Pour éviter cela, nous avons mis un filtre. C'est-à-dire que l'utilisateur ne pourra uploader d'autres formats que les suivants : png, jpg, jpeg. De plus, nous avons limité la taille des fichiers à uploader.

2. Précisez les moyens utilisés :

DOSSIER PROFESSIONNEL ^(DP)

3. Avec qui avez-vous travaillé ?

Nous avons collaboré à 4 sur ce projet : Joris Verguldezoone, Mathis Cottet, Mohamed-marwane Bellagha, Shun Lassal.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ *La Plateforme.*

Chantier, atelier, service ▶ *Projet Professionnel.*

Période d'exercice ▶ Du : *Decembre 2021* au : *Juillet 2022*

5. Informations complémentaires *(facultatif)*

Activité-type 3

Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité

Exemple n° 1 - Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Lors du développement de ce projet nous étions quatre. Il a fallu organiser les tâches en fonction de notre planning d'alternance et en dehors des cours que l'on recevait. Nous avons tout d'abord brainstormé sur les différents sujets qui nous intéressaient avant de choisir le thème du jeu de carte en ligne et en temps réel.

Ce projet comporte beaucoup de technologies que nous ne maîtrisons pas. Pour éviter de se laisser déborder, nous avons organisé des équipes tournantes. Deux personnes devaient mettre en place la partie BACK-END de l'application tandis que 2 autres devaient mettre en place le FRONT-END. Le but était de maîtriser les technologies front et back avant de faire un échange de connaissance entre les deux équipes. Ainsi en progressant chacun de notre côté, il ne nous resterait plus qu'à appliquer les conseils des uns et des autres pour compléter toutes les tâches que l'on s'était fixées. Pour connaître les tâches à long terme que nous aurions à réaliser nous avons produit un diagramme de Gantt. Celui-ci nous permet de visualiser toutes les tâches à effectuer sur une sorte de planning annuel. Ainsi à chaque fois que l'on rentrait d'alternance nous savions sur quoi nous concentrer et quelle serait la prochaine étape à remplir avant de pouvoir s'intéresser à une autre technologie ou fonctionnalité.

Dans une échelle de temps à court terme nous avons également utilisé Trello. Le but était que les 2 sous équipes BACK-END/FRONT-END puisse s'échanger leurs besoins à court terme. A chaque fois qu'une personne rencontrait un problème ou qu'elle remarquait qu'il fallait ajouter une fonctionnalité ou l'arranger à un endroit, une carte a été créée. Certaines cartes servent également aux ressources de documentation et nous permettent d'apprendre et d'avancer dans la même voie même si l'on ne se voyait pas régulièrement à cause de notre rythme d'alternance.

Nous possédons 3 répertoires GitHub afin de versionner notre Front-end, Back-end, et le serveur socket(back). React Native, NestJS et les socket (NodeJS, Socket.io) ont donc tous été développés indépendamment.

A chaque fois qu'une personne devait développer une nouvelle fonctionnalité il lui a été donné de créer une nouvelle branche sur GitHub.

Une fois que les deux membres de la même sous équipe se mettent d'accord ils peuvent merge sur le master ou l'un après l'autre si le travail a été bien intégré.

Concernant le Back-end, des tests sont effectués à chaque mise à jour de l'API à l'aide de SWAGGER que nous avons configuré.

Quand il s'agit du Front-end nous testons l'application sur téléphone et sur navigateur. La simulation sur navigateur nous permet de développer plus rapidement car nous n'avons pas besoin de télécharger les mises à jour sur téléphone qui sont longues. Cependant les compatibilités avec le simulateur du navigateur sont limitées et nous obligent à tester régulièrement avec le téléphone.

Le style dépend beaucoup du modèle de téléphone utilisé ce qui nous a obligé à faire attention lors des tests à ce que ce notre code soit d'autant plus compatible avec les autres supports.

Le serveur socket nécessite également une série de tests dans laquelle on ouvre plusieurs sessions sur un même navigateur pour accumuler des utilisateurs connectés. Cependant cette façon de tester fonctionne uniquement lorsque l'on désactive le système de Token qui est directement dépendant du LocalStorage du téléphone, noncompatible avec le stockage local du navigateur. Encore une chose qui nous oblige à rallonger nos tests en aillant plusieurs téléphones connectés à l'application. On doit ainsi désactiver une partie de notre application ou prendre beaucoup de temps avec plusieurs téléphones pour pouvoir mettre à jour notre serveur socket.

2. Précisez les moyens utilisés :

DOSSIER PROFESSIONNEL ^(DP)

3. Avec qui avez-vous travaillé ?

Nous avons collaboré à 4 sur ce projet : Joris Verguldezoone, Mathis Cottet, Mohamed-marwane Bellagha, Shun Lassal.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ *La Plateforme.*

Chantier, atelier, service ▶ *Projet professionnel.*

Période d'exercice ▶ Du : *Decembre 2021* au : *Juillet 2022*

5. Informations complémentaires (facultatif)

Activité-type 3

Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité

Exemple n° 1 ▶ Développer des composant métier

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Un utilisateur se connecte, émet naturellement un événement de connexion sur le serveur socket qui l'authentifie et l'utilisateur reçoit un token qui lui permet d'accéder à l'ensemble des espaces de l'application.

La création et l'accessibilité des lobbies :

Il a donc accès au bouton de création de lobby. Celui-ci appelle un composant général

CreateLobby dans lequel un composant CreateLobbyServices permet de fetch l'ensemble des jeux disponibles. Une fois le jeu sélectionné, le composant GameRule permet d'afficher les règles et les difficultés de jeu associé au jeu sélectionné.

On nomme alors le lobby et on appuie sur le bouton créer. Une redirection s'effectue, les données sont envoyées au serveur en POST via le composant CreateLobbyServices et l'on est dirigé directement dans le composant général Lobby que l'on vient de créer.

A ce moment-là un emit (envoi d'un signal socket comportant parfois une data) est effectué vers le serveur socket pour le notifier de notre présence. Le socket de l'utilisateur est alors directement associé à une room qui portera le nom du Lobby. Un fois ces signaux effectués, le Lobby figurera dans la liste des Lobby disponibles.

Un autre utilisateur se connecte et souhaite accéder au Lobby que l'on vient de créer. Il appuie sur le bouton liste des Lobbies et accède au composant général LobbyList. Ce composant affiche l'ensemble des lobbies existant qui sont tous répertoriés par leur nom de lobby qui est aussi le nom de la room associé en temps réel. L'utilisateur clique, rejoint ainsi le lobby et répète les signaux qu'a effectués le créateur du lobby pour se joindre à la room en y insérant son socket d'utilisateur.

Un troisième joueur souhaite se connecter. Mais le lobby est plein. Il recevra alors une réponse du serveur socket lors de sa tentative de connexion qui lui expliquera que le serveur est déjà rempli.

Si un utilisateur quitte alors il pourra rejoindre et si tous les utilisateurs quitte ou si la partie est lancée et terminée alors le lobby se supprimera par lui-même en socket avec l'événement disconnect. Lors de la déconnection en socket du lobby, si nodejs constate que le lobby est vide ou n'existe plus alors il va envoyer une requête de suppression du lobby en base de donnée via notre API utilisé avec axios dans le serveur socket.

Ainsi, la liste des lobby ne restera jamais rempli de serveur indisponible et sera toujours mis à jour avec soit les lobby disponible soit les lobby plein ou déjà en cours de jeu.

DOSSIER PROFESSIONNEL ^(DP)

2. Précisez les moyens utilisés :

3. Avec qui avez-vous travaillé ?

Nous avons collaboré à 4 sur ce projet : Joris Verguldezoone, Mathis Cottet, Mohamed-marwane Bellagha, Shun Lassal.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ *La Plateforme*

Chantier, atelier, service ▶ *Projet professionnel*

Période d'exercice ▶ Du : *Decembre 2021* au : *Juillet 2022*

5. Informations complémentaires (facultatif)

Activité-type 3

Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité

Exemple n° 1 ▶ Construire une application organisée en couches

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Notre application de jeu de carte mobile présente plusieurs couches afin de fonctionner. Tout d'abord un serveur (distribution Debian 11) nous permet d'héberger notre back-end. Celui-ci a été développé sur le Framework NestJS, en NodeJS et en TypeScript.

Un système de routeur intégré au framework nous a permis de construire notre API grâce à l'appel de fonction suivant les routes demandées. Ces routes exécutent des fonctions ayant des requêtes sql permettant de satisfaire les besoins de l'utilisateur de l'API

vis-à-vis de la base de données et ce qu'elle contient.

Pour chaque entité de notre back-end, NestJS va posséder un controller un model d'entité, une interface, des DTO , des services, un fichier .spec dans lequel on peut tester nos fonctions et un module qui va permettre d'assembler la logique entre chaque fichier, puis de tout rassembler dans le main.module qui est le fichier module racine du projet par lequel toutes les entités qui sont imbriquées au lancement de NestJS vont être appelées.

```
@Module({  
  
  imports: [TypeOrmModule.forFeature([User])],  
  
  exports: [TypeOrmModule],  
  
  providers: [UsersService],  
  
  controllers: [UsersController],  
  
})
```

Notre base de données est accessible via notre système de gestion de base de données MariaDB. Il utilise InnoDB qui est un moteur de stockage pour nous fournir des relations entre les tables. Notre base de données fonctionne à l'aide du serveur web Nginx et est relié à notre API via Un mapping objet-relationnel (en anglais object-relational mapping ou ORM, dans notre cas TypeOrm) qui est un type de programme informatique qui se place en interface entre un programme applicatif et une base de données relationnelle pour simuler une base de données orientée objet. Ce programme définit des correspondances entre les schémas de la base de données et les classes du programme applicatif.

Ainsi lorsqu'une requête s'exécute elle doit être validé par TypeOrm et la configuration qu'on lui a attribuée.

Notre API est disponible via un URL et le port 3001.

Celle-ci va être appelée par notre Front et une autre partie de notre back-end qui est le serveur socket.

Développé à l'aide de NodeJS et socket.io notre serveur socket est lui aussi hébergé sur le même serveur, sur le port 3002 et écoute en permanence les événements qu'il reçoit en provenance du front. Dans ses réponses, il appelle parfois l'API pour donner des informations à l'utilisateur.

Le serveur socket nous permet d'avoir une gestion des événements javascript en temps réel entre tous les utilisateurs.

Ainsi nos composants métier réagissent en temps réel aux cliques de chacun (lobby, jeu de carte), il nous permet également de mettre en place tout ce qui va permettre des interactions sociales entre les utilisateurs comme l'ajout d'amis, l'envoi et la réception de message, les notifications et encore d'autres fonctionnalités à venir...

Notre front-end est développé en React-native et construit à l'aide d'Expo.

React et react-native sont deux langages très proches, pour ne pas dire que ce sont les mêmes.

Cette proximité dans la compatibilité des deux langages nous permet d'émuler notre application mobile sur navigateur (notamment pour tester rapidement l'avancée de notre application) et sur mobile à l'aide d'un QR code à scanner. Afin de pouvoir accéder à l'API le front-end utilise la librairie AXIOS et utilise le SecureStore pour pouvoir utiliser le localStorage du téléphone afin de stocker des tokens ou des cookies.

Au début nous nous étions lancés dans un design pattern atomique. Notre code est donc divisé en organisme (ensemble d'une page), molécule (un composant appelé dans un organisme) et d'un atome (un tout petit composant appelé dans une molécule). Ainsi avec un ensemble de molécules et d'atomes nous sommes capables de générer une page modulaire (un organisme). Mais à terme nous n'avons pas utilisé cette architecture car

elle nous demandait de trop refactoriser le code.

Nous sommes donc restés sur une imbrication assez classique de nos composants dans une navigation Stack (react-navigation) dans laquelle on appelle un composant vue qui sera constitué de plusieurs composants qui effectueront des actions plus ou moins indépendantes du composant parent.

Nous faisons passer nos states dans notre stackNavigation qui alimente l'ensemble de nos pages. Parmi les states les plus partagés on a notamment le Token et le Socket de l'utilisateur. Ainsi notre utilisateur est identifié à la fois sur l'API et sur le serveur socket une fois qu'il a réussi sa connexion à l'aide de son compte utilisateur.

2. Précisez les moyens utilisés :

3. Avec qui avez-vous travaillé ?

Nous avons collaboré à 4 sur ce projet : Joris Verguldezoone, Mathis Cottet, Mohamed-marwane Bellagha, Shun Lassal.

4. Contexte

Nom de l'entreprise, organisme ou association ▶

Chantier, atelier, service ▶

Période d'exercice ▶ Du : *Decembre 2021* au : *Juillet 2022*

5. Informations complémentaires (facultatif)

Activité-type 3

Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité

Exemple n° 1 - Développer une application mobile / Concevoir une application mobile

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

A l'aide de la maquette, nous avons développé les pages de l'application en suivant le style graphique défini.

Nous avons créé des composants pour chaque pages tout en respectant les bonnes pratiques de développement Objet;

Ces composants sont des boutons, des entrées utilisateurs, tout ce qui permet à l'utilisateur d'interagir avec l'application, il vont être utilisés de nombreuses fois afin de développer notre interface utilisateur.

Voici la page d'accueil de l'application:



La page Connexion:



La page Inscription:



DOSSIER PROFESSIONNEL ^(DP)

2. Précisez les moyens utilisés :

Nous avons utilisé le framework React Native et le langage Javascript pour développer l'application.
En utilisant la structure de la maquette

3. Avec qui avez-vous travaillé ?

Nous avons collaboré à 4 sur ce projet : Joris Verguldezoone, Mathis Cottet, Mohamed-marwane Bellagha, Shun Lassal.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ *La Plateforme*

Chantier, atelier, service ▶ *Projet professionnel*

Période d'exercice ▶ Du : *Decembre 2021* au : *Juillet 2022*

5. Informations complémentaires (facultatif)

Activité-type 3

Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité

Exemple n° 1 ▶ Préparer et exécuter les plans de tests d'une application

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Afin de prévoir au mieux le déploiement de toute l'interface de programmation d'application ou application programming interface (API) que nous avons choisi de développer en utilisant Javascript et plus précisément le Framework NestJS, nous avons effectué une batterie de test unitaires ainsi qu'un test dit « end-to-end » sur l'intégralité de notre API.

Les test end-to-end sont des tests globaux réalisés sur l'intégralité d'un bout à l'autre de l'application et non plus sur chacune des fonctions de chacun des composants. Concrètement, lors d'un test dit end to end on recrée l'environnement de développement et d'utilisation de notre app et on test l'ensemble des fonctionnalité avec plusieurs types de données et plusieurs cas de figure afin de pouvoir s'assurer que notre application est bien sécurisé et marche comme on attend qu'elle marche

```
Test Suites: 2 failed, 2 passed, 4 total
Tests:      2 failed, 7 passed, 9 total
Snapshots:  0 total
Time:       10.217 s, estimated 12 s
Ran all test suites matching /users/i.

Watch Usage: Press w to show more.
```

2. Précisez les moyens utilisés :

Grâce à NestJS et la création automatique des fichiers de test utilisant le Framework de testing de javascript Jest, la création de test est facilitée. En effet, avec l'utilisation de Jest, la création de fausses données est facilitée pour vérifier que la fonction fonctionne correctement et renvoie exactement ce que nous attendions qu'elle renvoie. Il a fallu aussi effectuer des tests sur l'ensemble des fonctions de base c'est-à-dire l'ensemble des opérations possibles sur chacun de nos composants aussi bien sur la partie Controller

DOSSIER PROFESSIONNEL ^(DP)

que Service des modules de notre application.

3. Avec qui avez-vous travaillé ?

Nous avons collaboré à 4 sur ce projet : Joris Verguldezoone, Mathis Cottet, Mohamed-marwane Bellagha, Shun Lassal.

DOSSIER PROFESSIONNEL ^(DP)

4. Contexte

Nom de l'entreprise, organisme ou association ▶ *La Plateforme*

Chantier, atelier, service ▶ *Projet professionnel*

DOSSIER PROFESSIONNEL ^(DP)

Période d'exercice - Du : *Decembre 2021* au : *Juillet 2022*

5. Informations complémentaires (facultatif)

Activité-type 3

Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité

Exemple n° 1 - Préparer et exécuter le déploiement d'une application

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Dans le cadre du déploiement de notre API et de notre serveur socket nous avons utilisé un serveur distant. Dans un premier temps nous avons créé une Virtual Machine (VM) pour accéder au terminal SSH de notre serveur. Étant sur Window, nous ne possédons pas de terminal SSH, nous aurions pu en installer un léger mais nous avons préféré tester cela sur une VM. Une fois l'environnement mis en place nous avons pu accéder à notre serveur via les identifiants utilisateurs qui nous ont été fournis.

Avant de pouvoir procéder au déploiement de notre API il a fallu installer diverses technologies. On a tout d'abord installé NodeJS pour pouvoir utiliser NPM (gestion des paquets), NestJS (api sous nodejs). Puis apache2 même si par la suite on est passé sur Nginx. Et MariaDB pour gérer nos bases de données en SQL.

A l'aide d'un gestionnaire de port UFW (debian) nous avons ouvert les ports 3001 (API) et 3002 (socket). Puis est venu le temps de la migration sftp (Secure file transfert program) que l'on a effectué à l'aide de FileZilla.

Il nous a suffi de transférer nos fichiers sur un répertoire de linux configuré pour recevoir les transfert sftp, puis de déplacer les dossiers reçus dans le répertoire de notre utilisateur. Enfin nous avons pu lancer les npm install pour recevoir tous les modules

DOSSIER PROFESSIONNEL ^(DP)

nécessaires au lancement de nos deux serveurs et les tester.

Nous avons ainsi accès à notre serveur API via l'url <http://51.75.241.128:3001> et à notre serveur Socket via l'url <http://51.75.241.128:3002>

2. Précisez les moyens utilisés :

VirtualBox, Debian11, FileZilla, ufw (allow port debian), screen, npm, MariaDB, NestJS, socket.io

3. Avec qui avez-vous travaillé ?

Nous avons collaboré à 4 sur ce projet : Joris Verguldezoone, Mathis Cottet, Mohamed-marwane Bellagha, Shun Lassal.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ *La Plateforme*

Chantier, atelier, service ▶ *Projet professionnel*

Période d'exercice ▶ Du : *Decembre 2021* au : *Juillet 2022*

5. Informations complémentaires (facultatif)

Titres, diplômes, CQP, attestations de formation

(facultatif)

Intitulé	Autorité ou organisme	Date
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.

DOSSIER PROFESSIONNEL ^(DP)

Déclaration sur l'honneur

Je soussigné(e) [prénom et nom] **Mathis Cottet**,
déclare sur l'honneur que les renseignements fournis dans ce dossier sont exacts et que je suis
l'auteur(e) des réalisations jointes.

Fait à **Marseille** le **26/06/2022**
pour faire valoir ce que de droit.

Signature :

Documents illustrant la pratique professionnelle

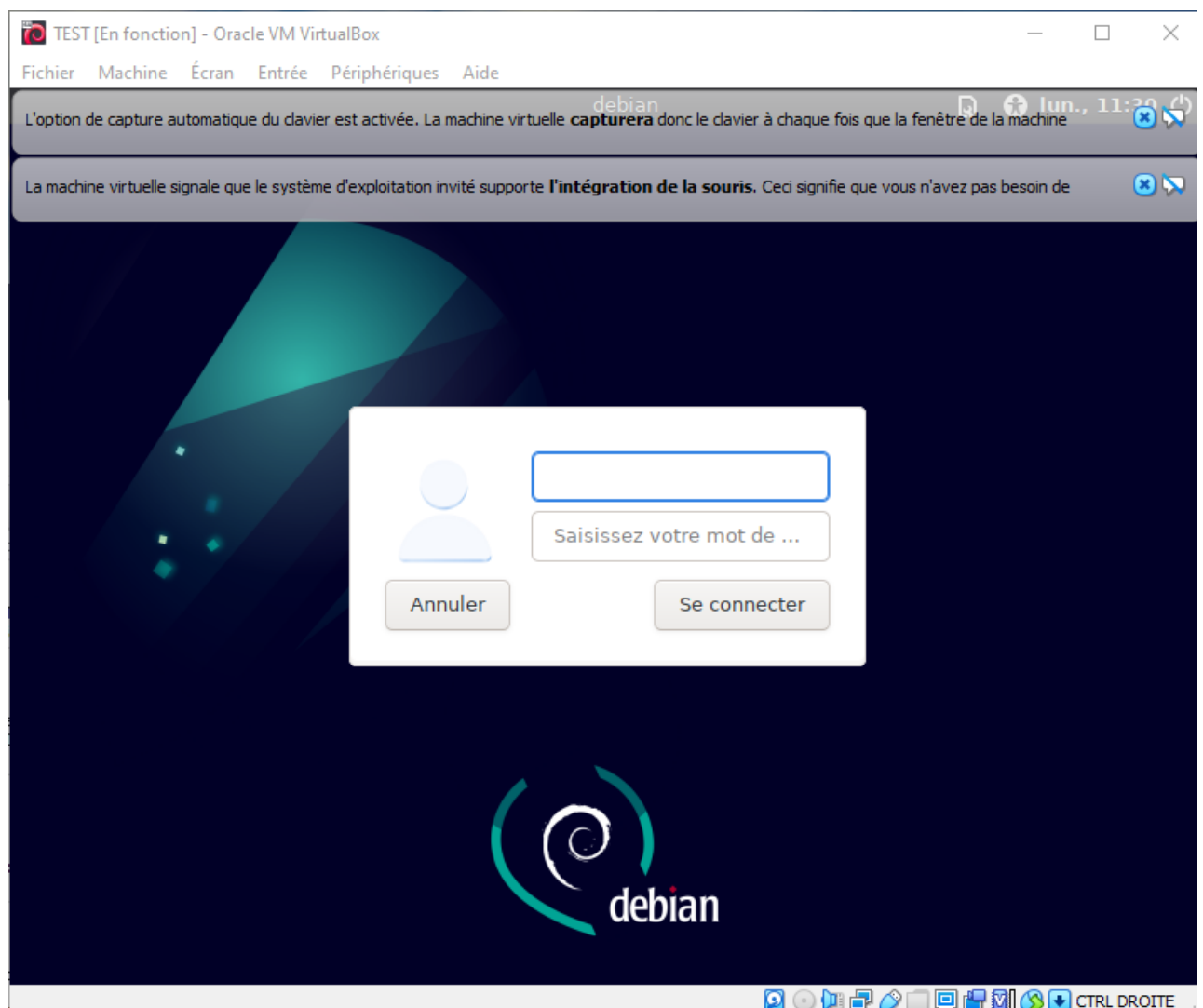
(facultatif)

Intitulé
Cliquez ici pour taper du texte.

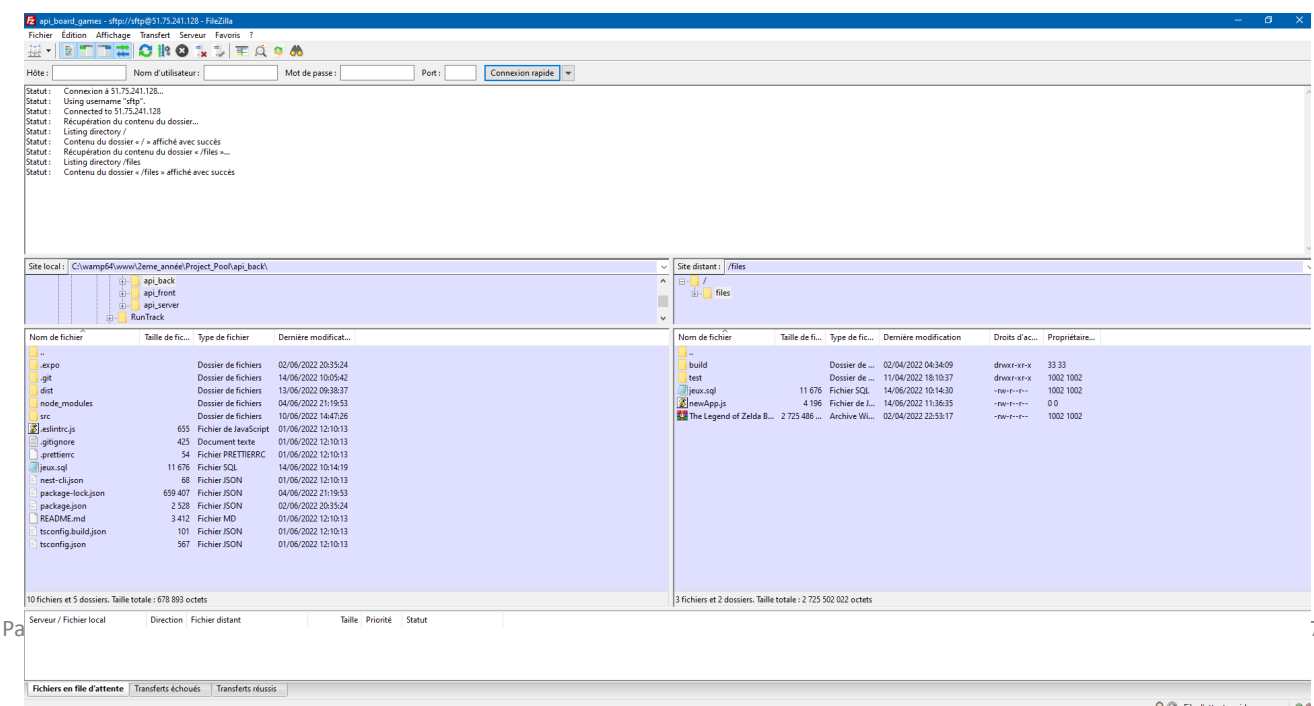
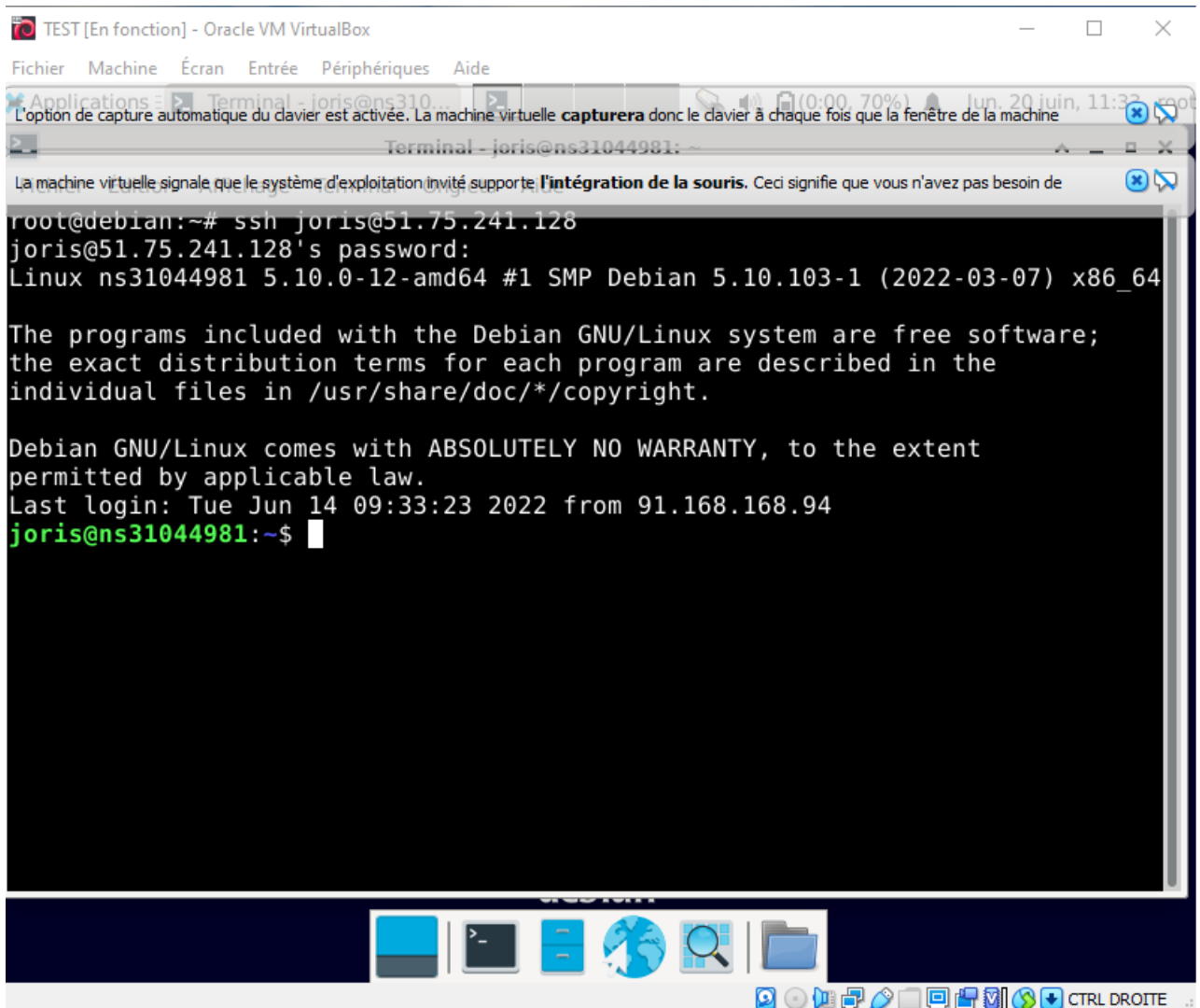
ANNEXES

(Si le RC le prévoit)

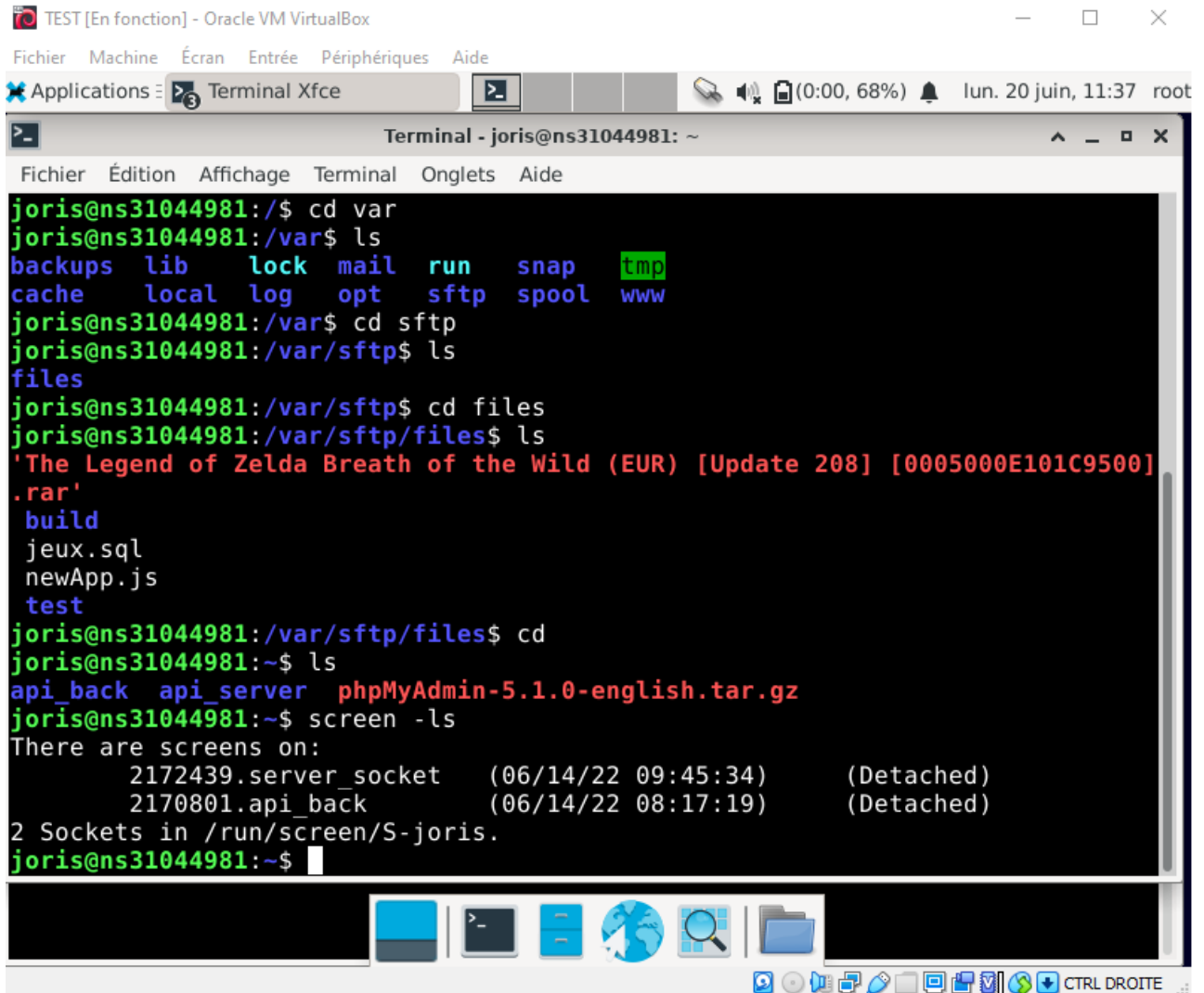
Déploiement :



DOSSIER PROFESSIONNEL (DP)

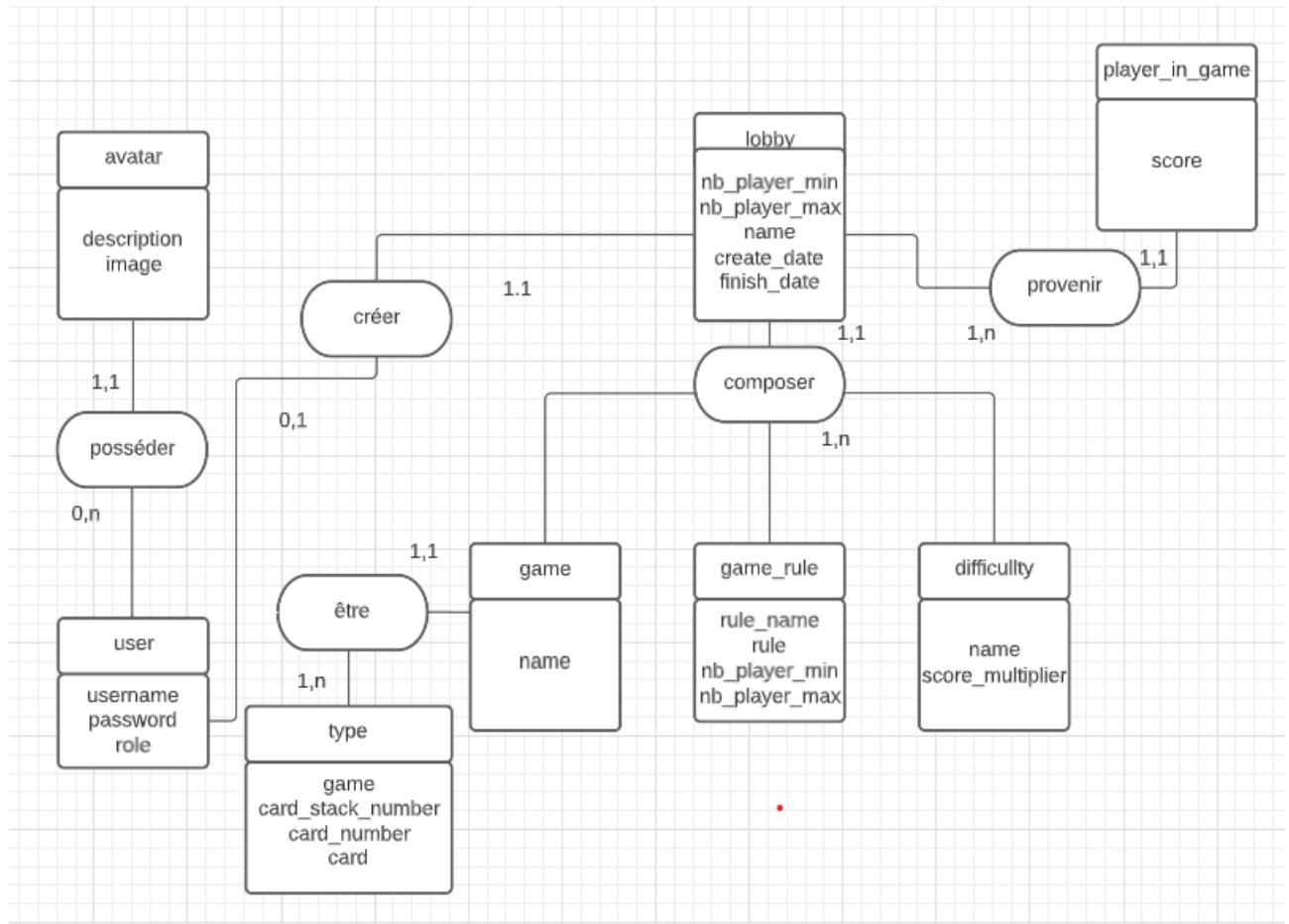


DOSSIER PROFESSIONNEL (DP)



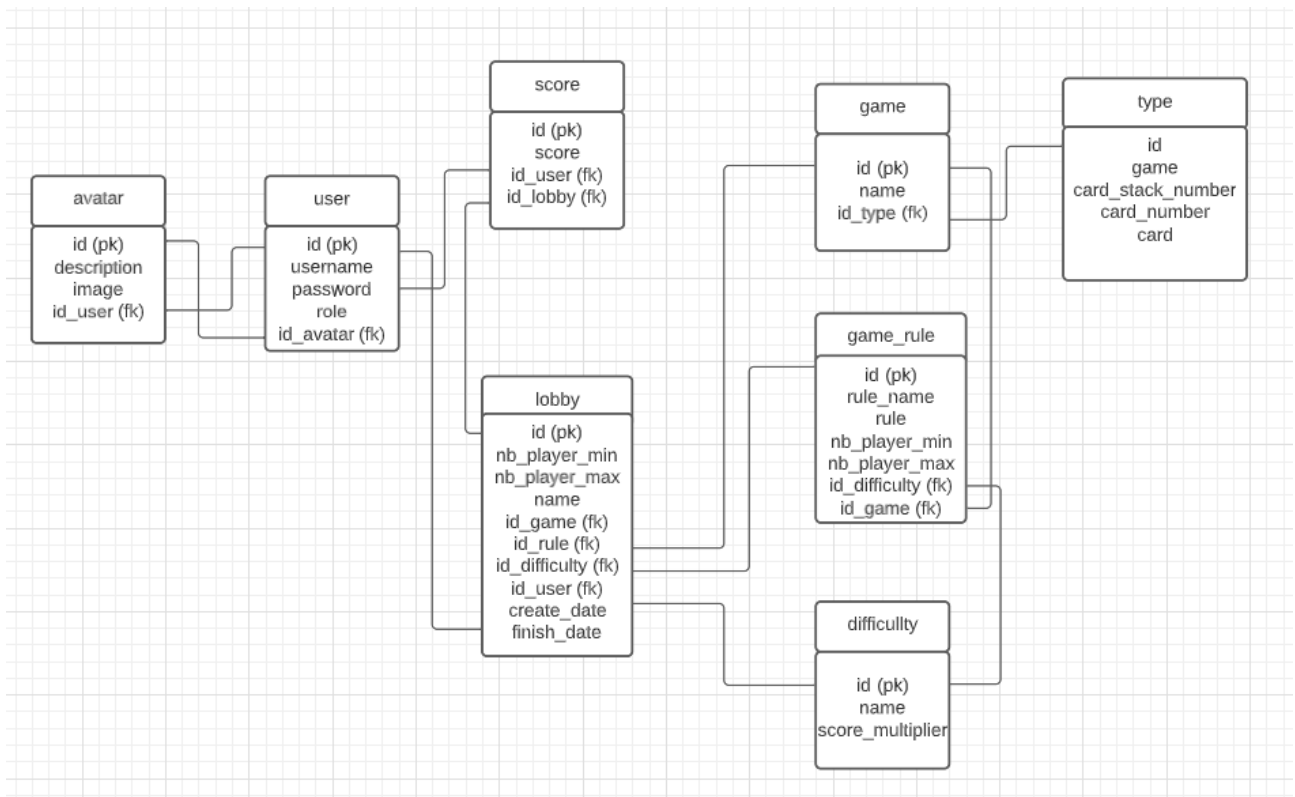
```
joris@ns31044981:/$ cd var
joris@ns31044981:/var$ ls
backups  lib      lock     mail     run      snap     tmp
cache    local    log      opt      sftp     spool    www
joris@ns31044981:/var$ cd sftp
joris@ns31044981:/var/sftp$ ls
files
joris@ns31044981:/var/sftp$ cd files
joris@ns31044981:/var/sftp/files$ ls
'The Legend of Zelda Breath of the Wild (EUR) [Update 208] [0005000E101C9500]
.rar'
build
jeux.sql
newApp.js
test
joris@ns31044981:/var/sftp/files$ cd
joris@ns31044981:~$ ls
api_back  api_server  phpMyAdmin-5.1.0-english.tar.gz
joris@ns31044981:~$ screen -ls
There are screens on:
      2172439.server_socket      (06/14/22 09:45:34)      (Detached)
      2170801.api_back          (06/14/22 08:17:19)      (Detached)
2 Sockets in /run/screen/S-joris.
joris@ns31044981:~$
```

Concevoir une base de donnée:



MCD

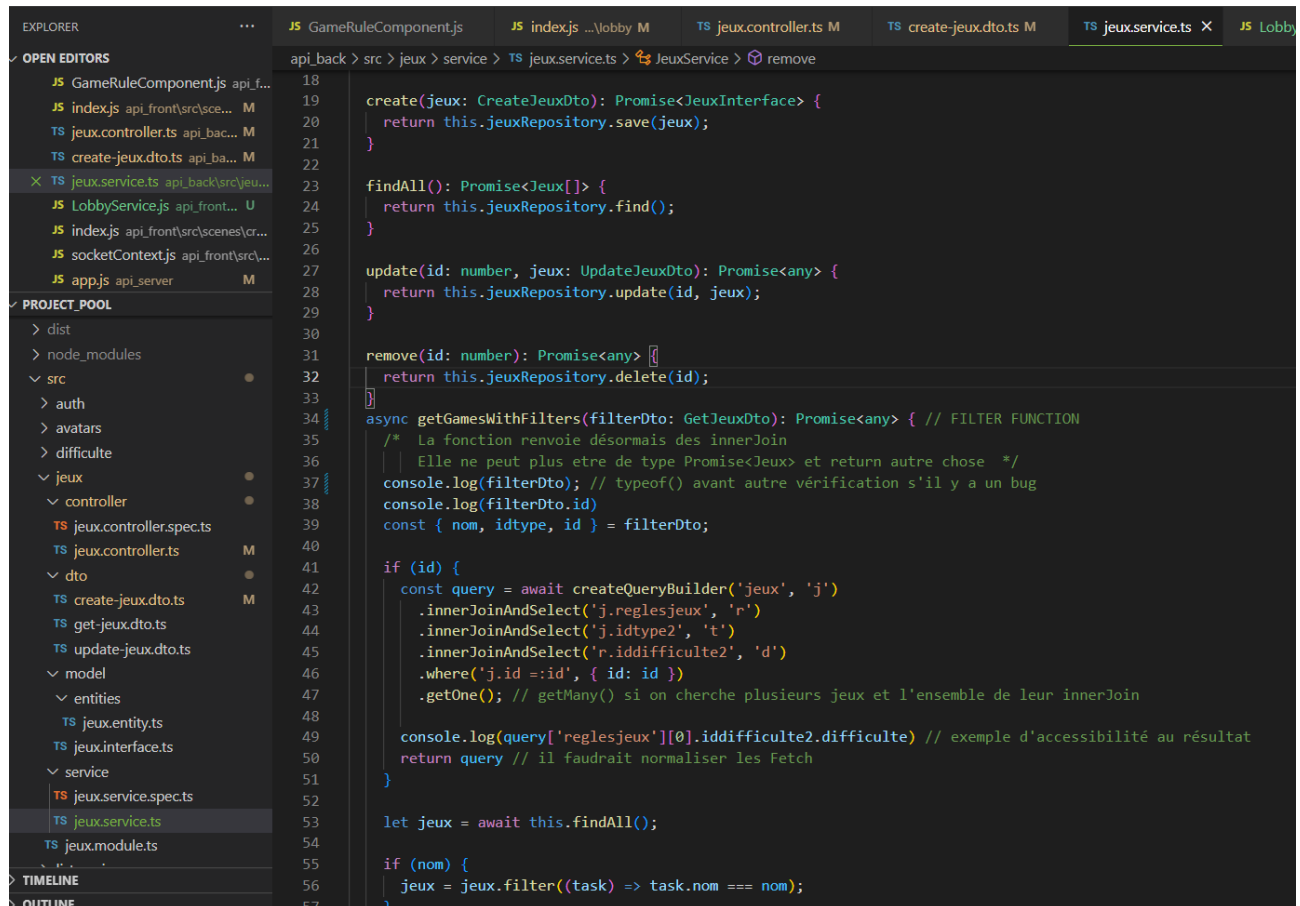
DOSSIER PROFESSIONNEL (DP)



MLD

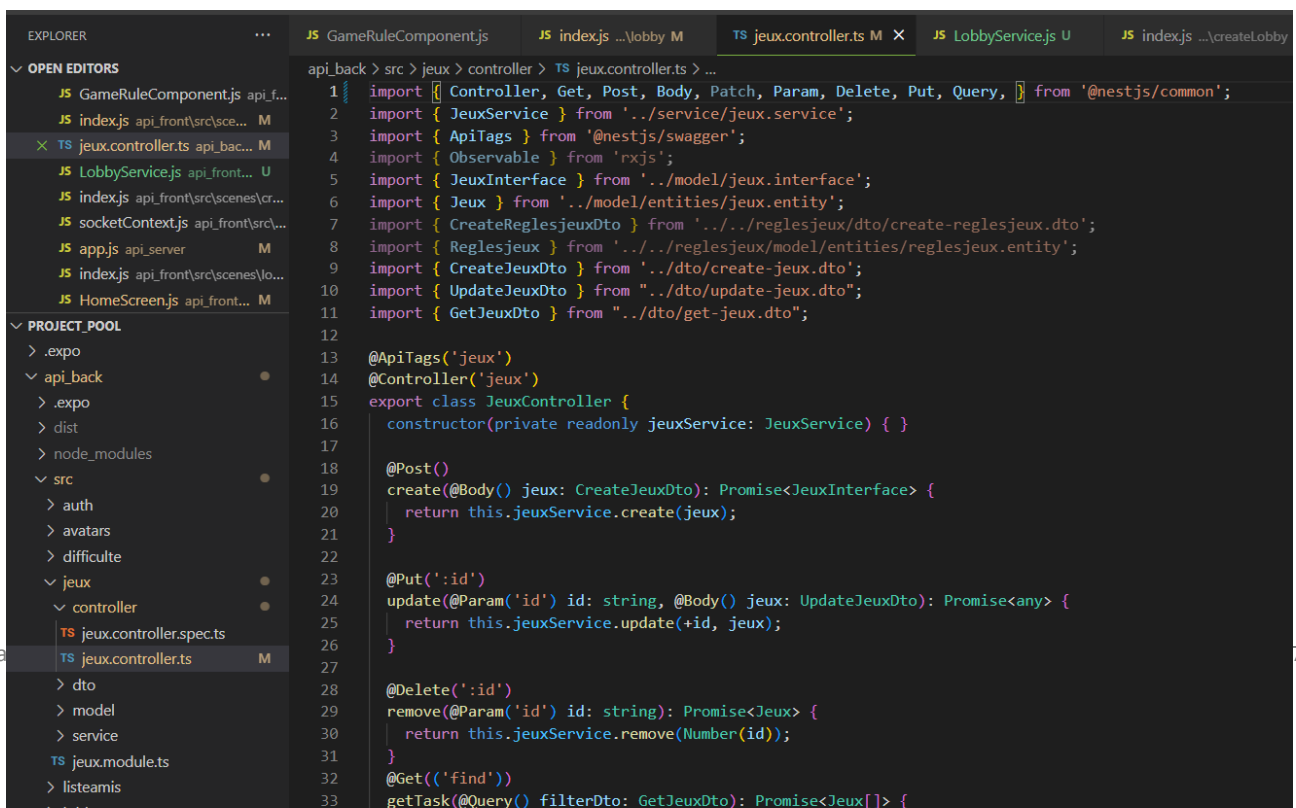
DOSSIER PROFESSIONNEL (DP)

Développer les composant d'accès aux données :



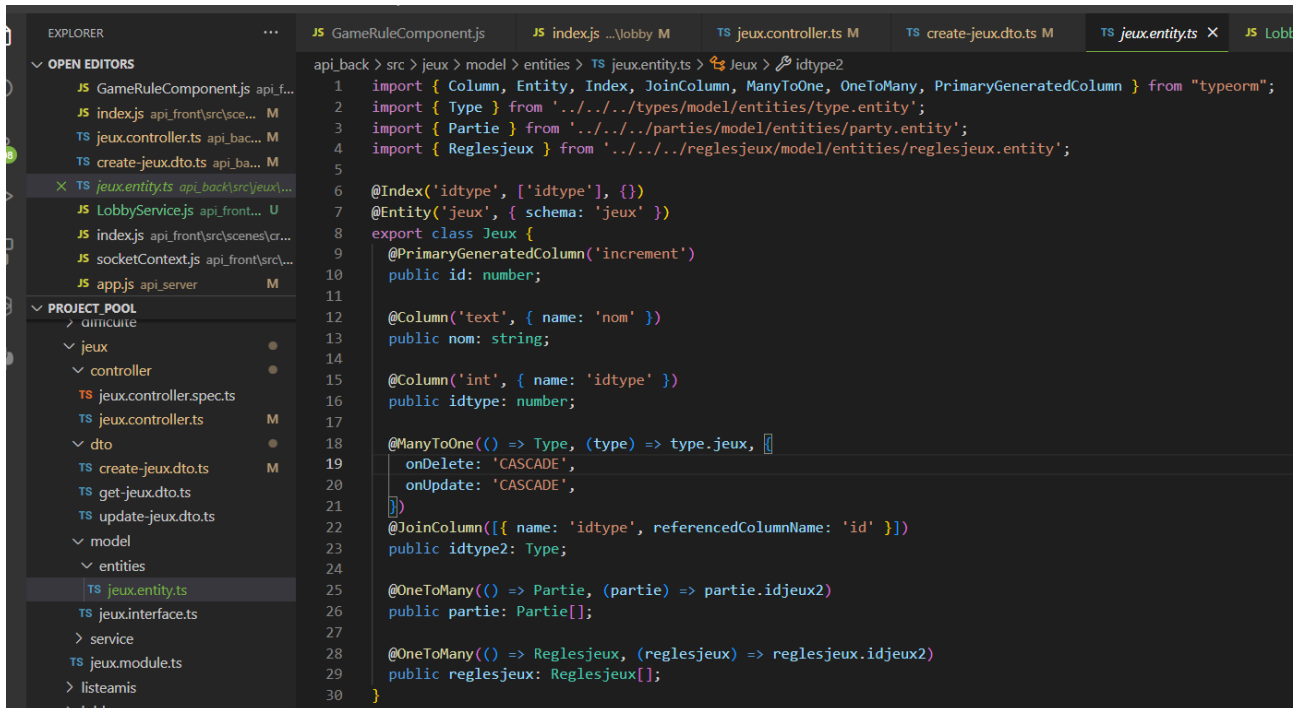
```
18
19
20 create(jeux: CreateJeuxDto): Promise<JeuxInterface> {
21   return this.jeuxRepository.save(jeux);
22 }
23
24 findAll(): Promise<Jeux[]> {
25   return this.jeuxRepository.find();
26 }
27
28 update(id: number, jeux: UpdateJeuxDto): Promise<any> {
29   return this.jeuxRepository.update(id, jeux);
30 }
31
32 remove(id: number): Promise<any> {
33   return this.jeuxRepository.delete(id);
34 }
35
36 async getGamesWithFilters(filterDto: GetJeuxDto): Promise<any> { // FILTER FUNCTION
37   /* La fonction renvoie désormais des innerJoin
38   Elle ne peut plus etre de type Promise<Jeux> et return autre chose */
39   console.log(filterDto); // typeof() avant autre vérification s'il y a un bug
40   console.log(filterDto.id)
41   const { nom, idtype, id } = filterDto;
42
43   if (id) {
44     const query = await createQueryBuilder('jeux', 'j')
45       .innerJoinAndSelect('j.reglesjeux', 'r')
46       .innerJoinAndSelect('j.idtype2', 't')
47       .innerJoinAndSelect('r.iddifficulte2', 'd')
48       .where('j.id=:id', { id: id })
49       .getOne(); // getMany() si on cherche plusieurs jeux et l'ensemble de leur innerJoin
50
51     console.log(query['reglesjeux'][0].iddifficulte2.difficulte) // exemple d'accessibilité au résultat
52     return query // il faudrait normaliser les Fetch
53   }
54
55   let jeux = await this.findAll();
56
57   if (nom) {
58     jeux = jeux.filter((task) => task.nom === nom);
59   }
60 }
```

services - controller



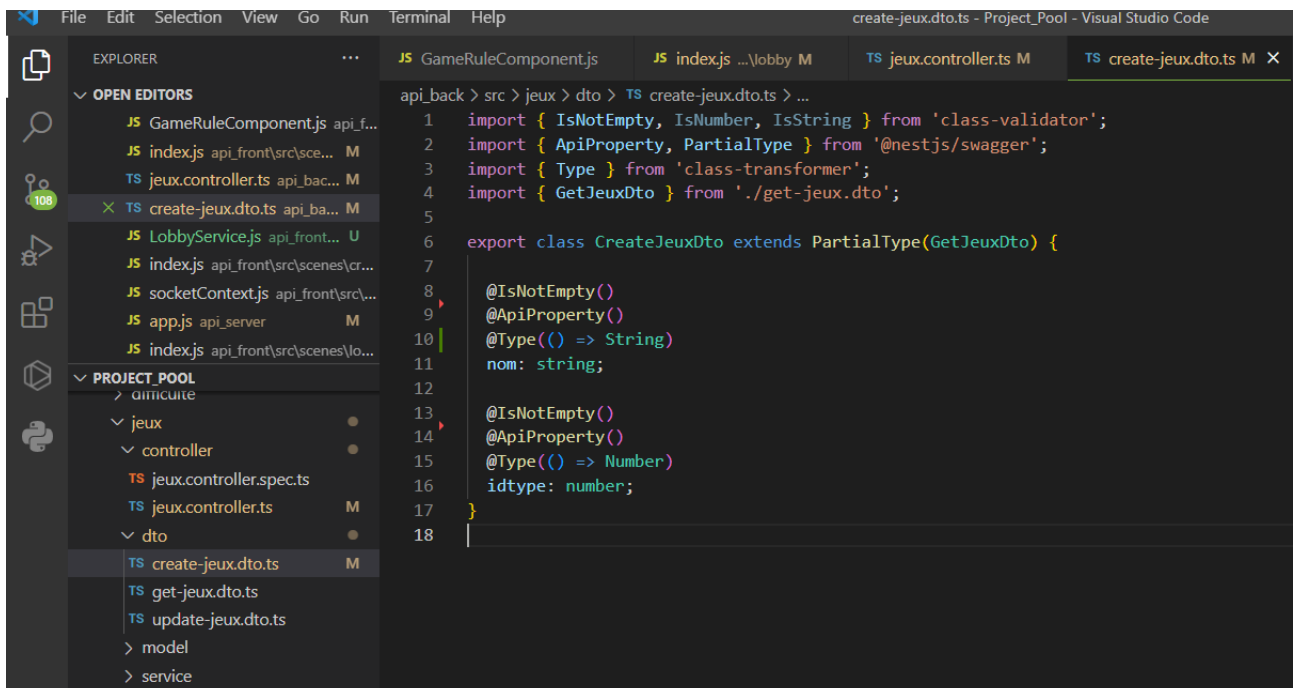
```
1 import { Controller, Get, Post, Body, Patch, Param, Delete, Put, Query, } from '@nestjs/common';
2 import { JeuxService } from '../service/jeux.service';
3 import { ApiTags } from '@nestjs/swagger';
4 import { Observable } from 'rxjs';
5 import { JeuxInterface } from '../model/jeux.interface';
6 import { Jeux } from '../model/entities/jeux.entity';
7 import { CreateReglesjeuxDto } from '../reglesjeux/dto/create-reglesjeux.dto';
8 import { Reglesjeux } from '../reglesjeux/model/entities/reglesjeux.entity';
9 import { CreateJeuxDto } from '../dto/create-jeux.dto';
10 import { UpdateJeuxDto } from '../dto/update-jeux.dto';
11 import { GetJeuxDto } from '../dto/get-jeux.dto';
12
13 @ApiTags('jeux')
14 @Controller('jeux')
15 export class JeuxController {
16   constructor(private readonly jeuxService: JeuxService) { }
17
18   @Post()
19   create(@Body() jeux: CreateJeuxDto): Promise<JeuxInterface> {
20     return this.jeuxService.create(jeux);
21   }
22
23   @Put('/:id')
24   update(@Param('id') id: string, @Body() jeux: UpdateJeuxDto): Promise<any> {
25     return this.jeuxService.update(+id, jeux);
26   }
27
28   @Delete('/:id')
29   remove(@Param('id') id: string): Promise<Jeux> {
30     return this.jeuxService.remove(Number(id));
31   }
32
33   @Get('/:find')
34   getTask(@Query() filterDto: GetJeuxDto): Promise<Jeux[]> {
35     return this.jeuxService.getGamesWithFilters(filterDto);
36   }
37 }
```

DOSSIER PROFESSIONNEL (DP)



```
api_back > src > jeux > model > entities > TS jeux.entity.ts > Jeux > idtype2
1 import { Column, Entity, Index, JoinColumn, ManyToOne, OneToMany, PrimaryGeneratedColumn } from "typeorm";
2 import { Type } from '.../types/model/entities/type.entity';
3 import { Partie } from '.../parties/model/entities/party.entity';
4 import { Reglesjeux } from '.../reglesjeux/model/entities/reglesjeux.entity';
5
6 @Index('idtype', ['idtype'], {})
7 @Entity('jeux', { schema: 'jeux' })
8 export class Jeux {
9   @PrimaryGeneratedColumn('increment')
10   public id: number;
11
12   @Column('text', { name: 'nom' })
13   public nom: string;
14
15   @Column('int', { name: 'idtype' })
16   public idtype: number;
17
18   @ManyToOne(() => Type, (type) => type.jeux, {
19     onDelete: 'CASCADE',
20     onUpdate: 'CASCADE',
21   })
22   @JoinColumn([ { name: 'idtype', referencedColumnName: 'id' } ])
23   public idtype2: Type;
24
25   @OneToMany(() => Partie, (partie) => partie.idjeux2)
26   public partie: Partie[];
27
28   @OneToMany(() => Reglesjeux, (reglesjeux) => reglesjeux.idjeux2)
29   public reglesjeux: Reglesjeux[];
30 }
```

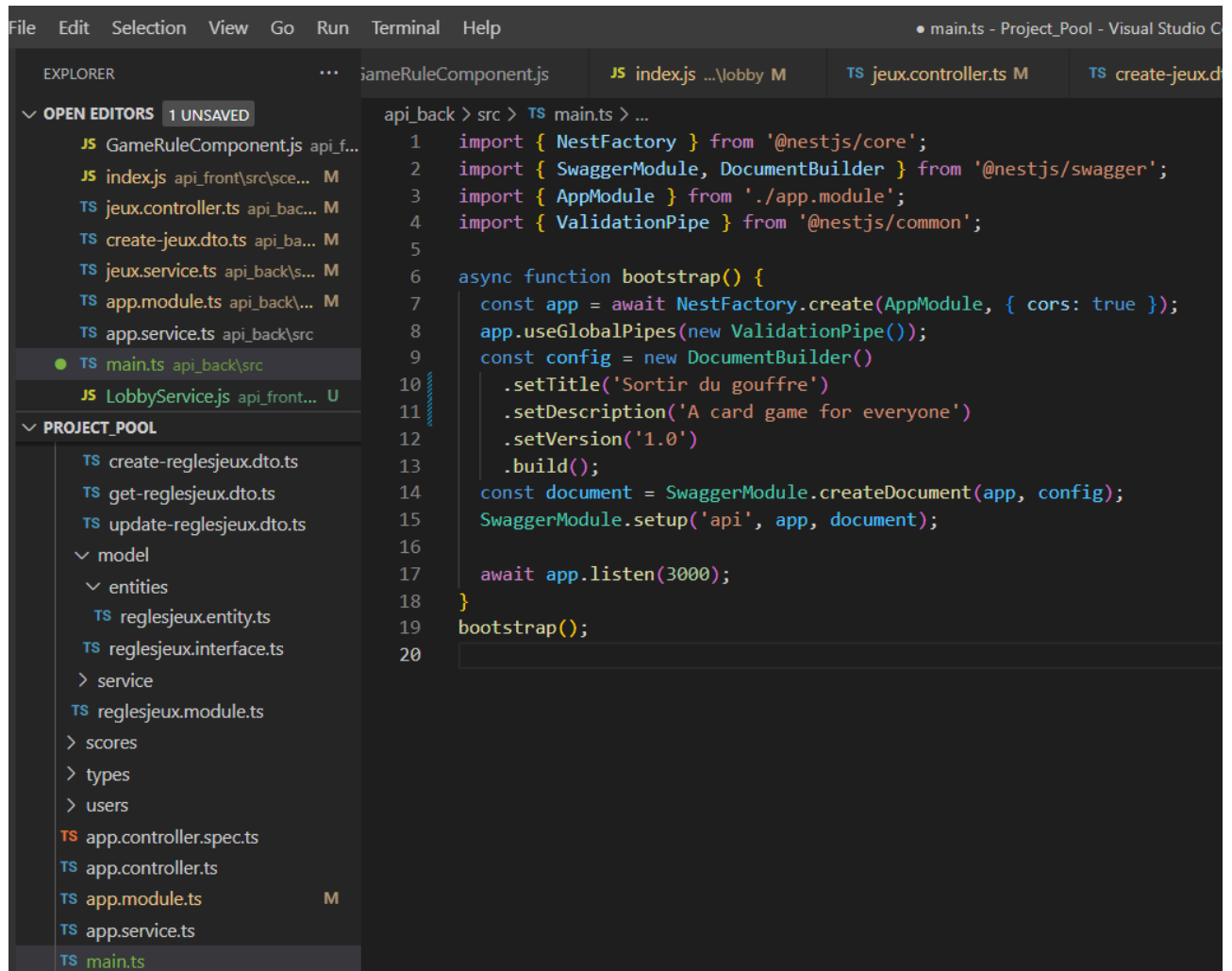
entity ORM



```
api_back > src > jeux > dto > TS create-jeux.dto.ts > ...
1 import { IsNotEmpty, IsNumber, IsString } from 'class-validator';
2 import { ApiProperty, PartialType } from '@nestjs/swagger';
3 import { Type } from 'class-transformer';
4 import { GetJeuxDto } from './get-jeux.dto';
5
6 export class CreateJeuxDto extends PartialType(GetJeuxDto) {
7
8   @IsNotEmpty()
9   @ApiProperty()
10   @Type(() => String)
11   nom: string;
12
13   @IsNotEmpty()
14   @ApiProperty()
15   @Type(() => Number)
16   idtype: number;
17 }
18
```

dto ORM

DOSSIER PROFESSIONNEL (DP)



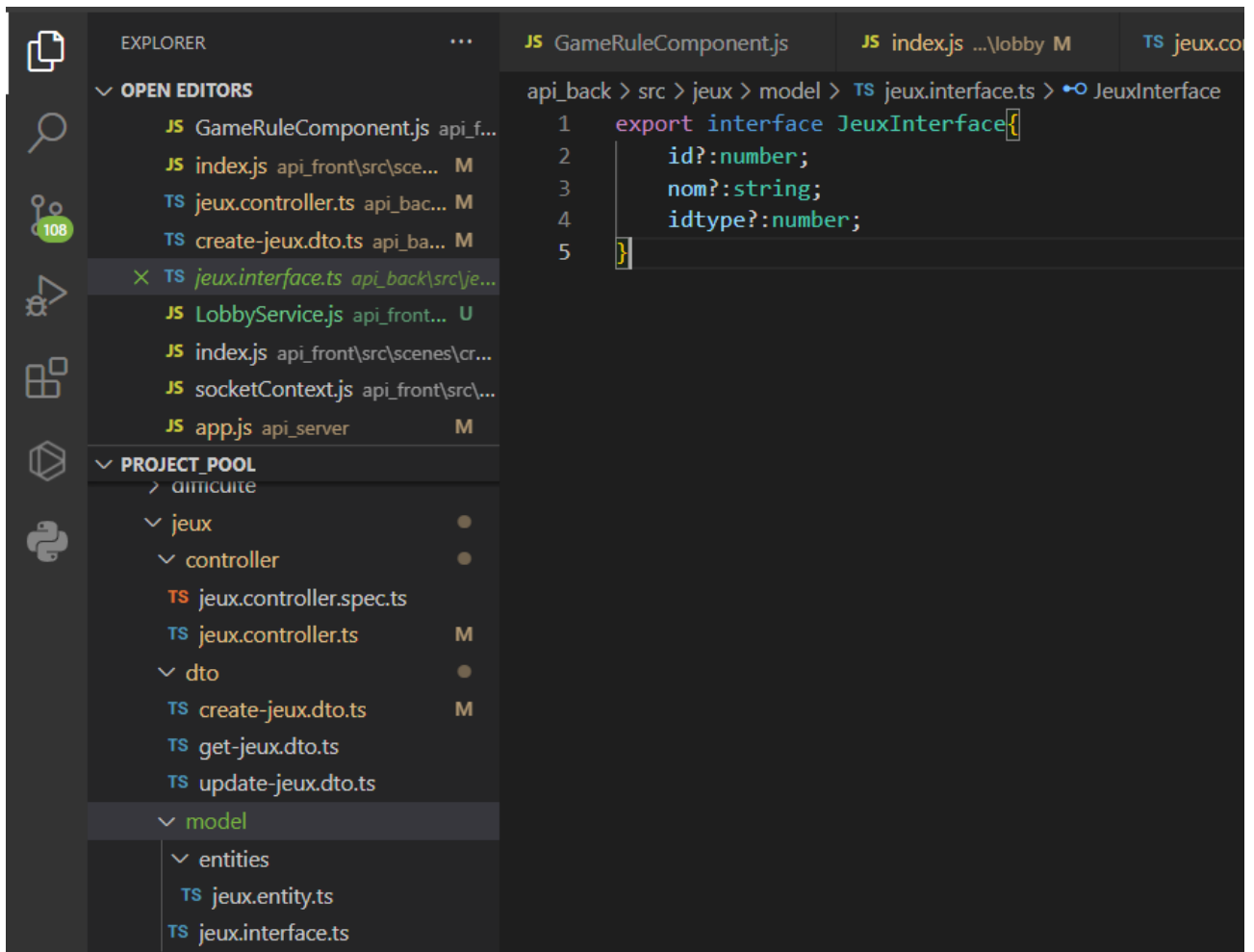
```
File Edit Selection View Go Run Terminal Help • main.ts - Project_Pool - Visual Studio C

EXPLORER
  OPEN EDITORS 1 UNSAVED
    JS GameRuleComponent.js api_f...
    JS index.js api_front\src\sce... M
    TS jeux.controller.ts api_bac... M
    TS create-jeux.dto.ts api_ba... M
    TS jeux.service.ts api_back\s... M
    TS app.module.ts api_back\... M
    TS app.service.ts api_back\src
    ● TS main.ts api_back\src
    JS LobbyService.js api_front... U
  PROJECT_POOL
    TS create-reglesjeux.dto.ts
    TS get-reglesjeux.dto.ts
    TS update-reglesjeux.dto.ts
    > model
    > entities
      TS reglesjeux.entity.ts
    TS reglesjeux.interface.ts
    > service
    TS reglesjeux.module.ts
    > scores
    > types
    > users
    TS app.controller.spec.ts
    TS app.controller.ts
    TS app.module.ts M
    TS app.service.ts
    TS main.ts

api_back > src > TS main.ts > ...
1 import { NestFactory } from '@nestjs/core';
2 import { SwaggerModule, DocumentBuilder } from '@nestjs/swagger';
3 import { AppModule } from './app.module';
4 import { ValidationPipe } from '@nestjs/common';
5
6 async function bootstrap() {
7   const app = await NestFactory.create(AppModule, { cors: true });
8   app.useGlobalPipes(new ValidationPipe());
9   const config = new DocumentBuilder()
10     .setTitle('Sortir du gouffre')
11     .setDescription('A card game for everyone')
12     .setVersion('1.0')
13     .build();
14   const document = SwaggerModule.createDocument(app, config);
15   SwaggerModule.setup('api', app, document);
16
17   await app.listen(3000);
18 }
19 bootstrap();
20
```

main config

interface ORM



DOSSIER PROFESSIONNEL ^(DP)

Travail collaboratif :

DOSSIER PROFESSIONNEL (DP)

Q Search branches...

Overview Yours Active Stale All branches New branch

Default branch

master Updated 7 days ago by joris-verguldezoone Default

Your branches

manoo Updated 17 days ago by joris-verguldezoone	5 0	New pull request	
adjust_db_with_new_table Updated last month by joris-verguldezoone	13 0	New pull request	
payload Updated 4 months ago by joris-verguldezoone	18 0	New pull request	

Active branches

manoo Updated 17 days ago by joris-verguldezoone	5 0	New pull request	
adjust_db_with_new_table Updated last month by joris-verguldezoone	13 0	New pull request	

Stale branches

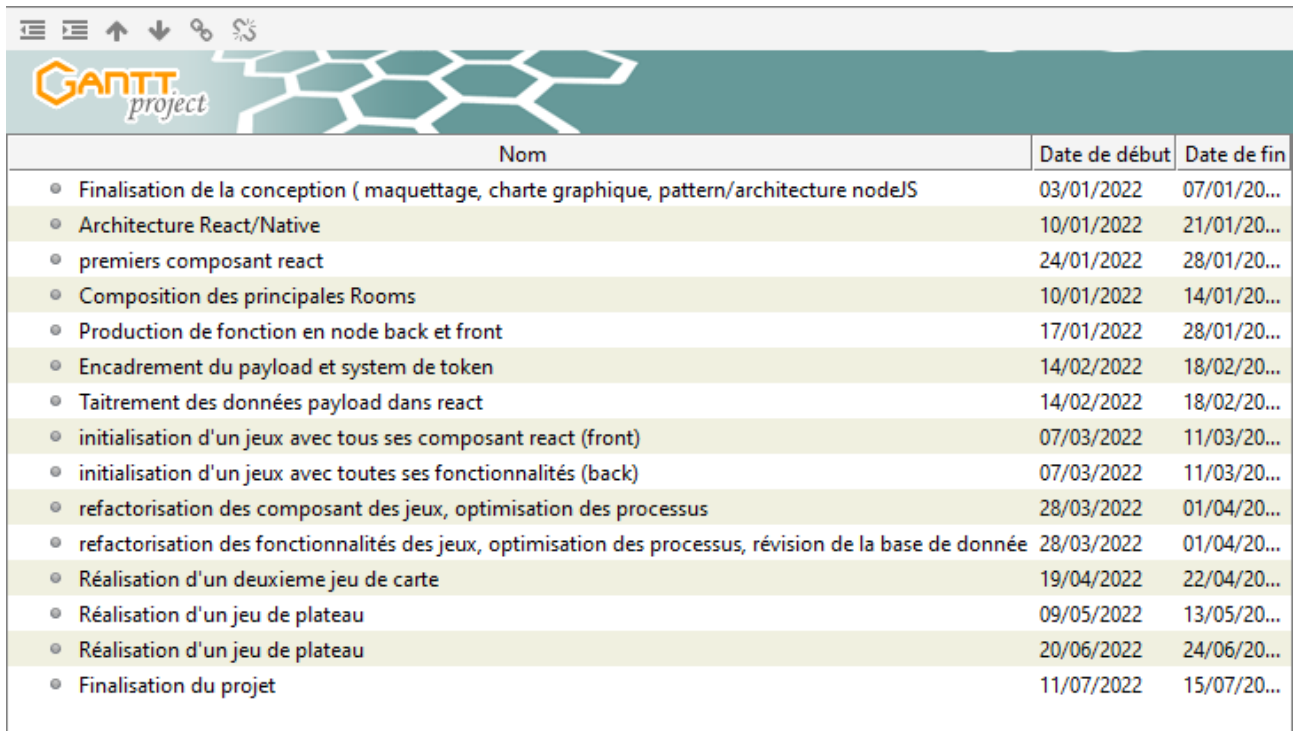
payload Updated 4 months ago by joris-verguldezoone	18 0	New pull request	
---	------	------------------	--

master 4 branches 0 tags Go to file Add file Code

joris-verguldezoone fix table partie 0aede7f 7 days ago 20 commits

.expo	prequel	2 months ago
src	fix table partie	7 days ago
.eslintrc.js	first commit	4 months ago
.gitignore	first commit	4 months ago
.prettierrc	first commit	4 months ago
README.md	first commit	4 months ago
jeux.sql	fix table partie	7 days ago
nest-cli.json	first commit	4 months ago
package-lock.json	ajout create and update dto avant test	last month
package.json	ajout create and update dto avant test	last month
tsconfig.build.json	first commit	4 months ago
tsconfig.json	first commit	4 months ago

DOSSIER PROFESSIONNEL (DP)



Nom	Date de début	Date de fin
• Finalisation de la conception (maquettage, charte graphique, pattern/architecture nodeJS	03/01/2022	07/01/20...
• Architecture React/Native	10/01/2022	21/01/20...
• premiers composant react	24/01/2022	28/01/20...
• Composition des principales Rooms	10/01/2022	14/01/20...
• Production de fonction en node back et front	17/01/2022	28/01/20...
• Encadrement du payload et system de token	14/02/2022	18/02/20...
• Traitement des données payload dans react	14/02/2022	18/02/20...
• initialisation d'un jeux avec tous ses composant react (front)	07/03/2022	11/03/20...
• initialisation d'un jeux avec toutes ses fonctionnalités (back)	07/03/2022	11/03/20...
• refactorisation des composant des jeux, optimisation des processus	28/03/2022	01/04/20...
• refactorisation des fonctionnalités des jeux, optimisation des processus, révision de la base de donnée	28/03/2022	01/04/20...
• Réalisation d'un deuxieme jeu de carte	19/04/2022	22/04/20...
• Réalisation d'un jeu de plateau	09/05/2022	13/05/20...
• Réalisation d'un jeu de plateau	20/06/2022	24/06/20...
• Finalisation du projet	11/07/2022	15/07/20...

Test Unitaires sur la base de données Users:

```
describe( name: 'UserController', fn: () => {
  let controller: UsersController;
  let service: UsersService;
  const mockUsersService = {
    create: jest.fn( implementation: (dto) => {
      return {
        id: Date.now(),
        ...dto,
      };
    }),
    update: jest.fn( implementation: (id, dto) => ({
      id,
      ...dto,
    })),
    getTask: jest
      .fn()
      .mockImplementation( fn: (user :any ) =>
        Promise.resolve( value: { id: Date.now(), ...user })),
    getUsersWithFilters: jest
      .fn()
      .mockImplementation( fn: (user :any ) => Promise.resolve( value: { ...user })),
    remove: jest.fn().mockResolvedValue( value: 1),
  };
});
```



```
beforeEach( fn: async () => {
  const module: TestingModule = await Test.createTestingModule( metadata: {
    controllers: [UsersController],
    providers: [UsersService, User],
  }) TestingModuleBuilder
    .overrideProvider(UsersService) OverrideBy
    .useValue(mockUsersService) TestingModuleBuilder
    .compile();
  service = module.get<UsersService>(UsersService);
  controller = module.get<UsersController>(UsersController);
});
```

```
it( name: 'should be defined', fn: () => {
  expect(controller).toBeDefined();
});
it( name: 'should create a user', fn: () => {
  const dto = {
    username: 'termti',
    password: 'termti',
    idavatar: 1,
    role: 0,
  };
  expect(controller.create(dto)).toEqual( expected: {
    id: expect.any(Number),
    username: 'termti',
    password: 'termti',
    idavatar: 1,
    role: 0,
  });
});
it( name: 'should update a user', fn: () => {
  const dto = {
    id: 1,
    username: 'termta',
    password: 'termta',
    idavatar: 1,
    role: 0,
  };
  expect(controller.update( id: '1', dto)).toEqual( expected: {
    id: 1,
    ...dto,
  });
});
```

Card board games is a school project that we made as a team of 4 students. We wanted to make an application able to host card and board games. We created a database and builded the front-end and the back-end in this way.

To implement these games we built a multi-layered application with NestJS , React-Native and a socket server. Our approach allows us to build real-time services and gaming. The majority of the data that make the players able to play are stored in the component of the react-native part and the database hydrates the component with the values that are selected by the player and shared with the others.

The cards game is the only type of game implemented but it's just a question of time because the system can easily handle the implementation of the boards game type.

Each cards has been designed by us. We also draw two playground to positionate players and cards.

At the end, player can create and join a lobby. Play a game with other people and then see the score of his game.

Everyting has been developped in javascript (NodeJs, TypeScript)