



DOSSIER PROFESSIONNEL (DP)

Nom de naissance

▶ VERGULDEZOONE

Nom d'usage

▶ VERGULDEZOONE

Prénom

▶ JORIS

Adresse

▶ 139 rue François Mauriac, 7 résidences le Club,
13010 Marseille

Titre professionnel visé

Titre RNCP - Concepteur développeur d'applications

MODALITE D'ACCES :

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

DOSSIER PROFESSIONNEL (DP)

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.

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

DOSSIER PROFESSIONNEL ^(DP)



<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.	4
▶ Maquetter une application - Zill Event	p.	5
▶ Développer la partie front-end d'une interface utilisateur web – Zill Event	p.	9
▶ Développer la partie back-end d'une interface utilisateur web – Zill Event	p.	11
▶ Développer une interface de type desktop – Python	p.	13
Concevoir et développer la persistance des données en intégrant les recommandations de sécurité	p.	
▶ Concevoir une base de données	p.	15
▶ Développer des composants dans le langage d'une base de données	p.	18
▶ Intégrer les recommandations de sécurité		20
Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité	p.	
▶ Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement	p. p.	24
▶ Développer des composants métiers	p p.	27
▶ Construire une application organisée en couches		29
▶ Préparer et exécuter le déploiement d'une application		32
▶ Préparer et exécuter les plans de tests d'une application		35
▶ Développer une application		37
Titres, diplômes, CQP, attestations de formation (facultatif)	p.	
Déclaration sur l'honneur	p.	
Documents illustrant la pratique professionnelle (facultatif)	p.	

EXEMPLES DE PRATIQUE PROFESSIONNELLE

DOSSIER PROFESSIONNEL (DP)

Activité-type 1

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

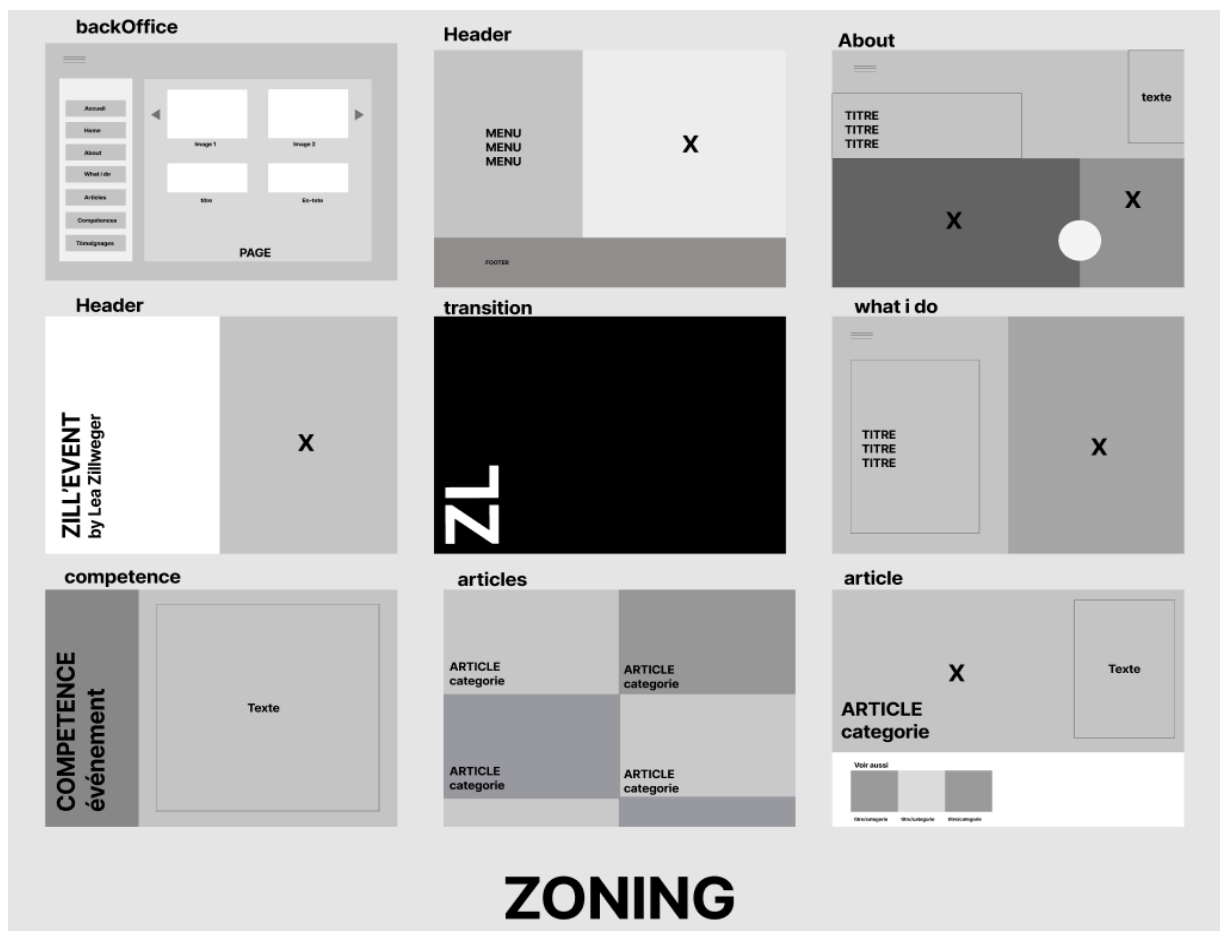
Exemple n°1 ► Maquetter une application

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

Zill Event est un projet professionnel que j'ai effectué dans le cadre d'une réponse au besoin d'un client de mon école. Léa Zill Weger est une auto entrepreneuse qui aide à l'organisation d'événement. Pour participer à la promotion de son activité, elle m'a demandé à moi et à une camarade de lui fournir un site internet.

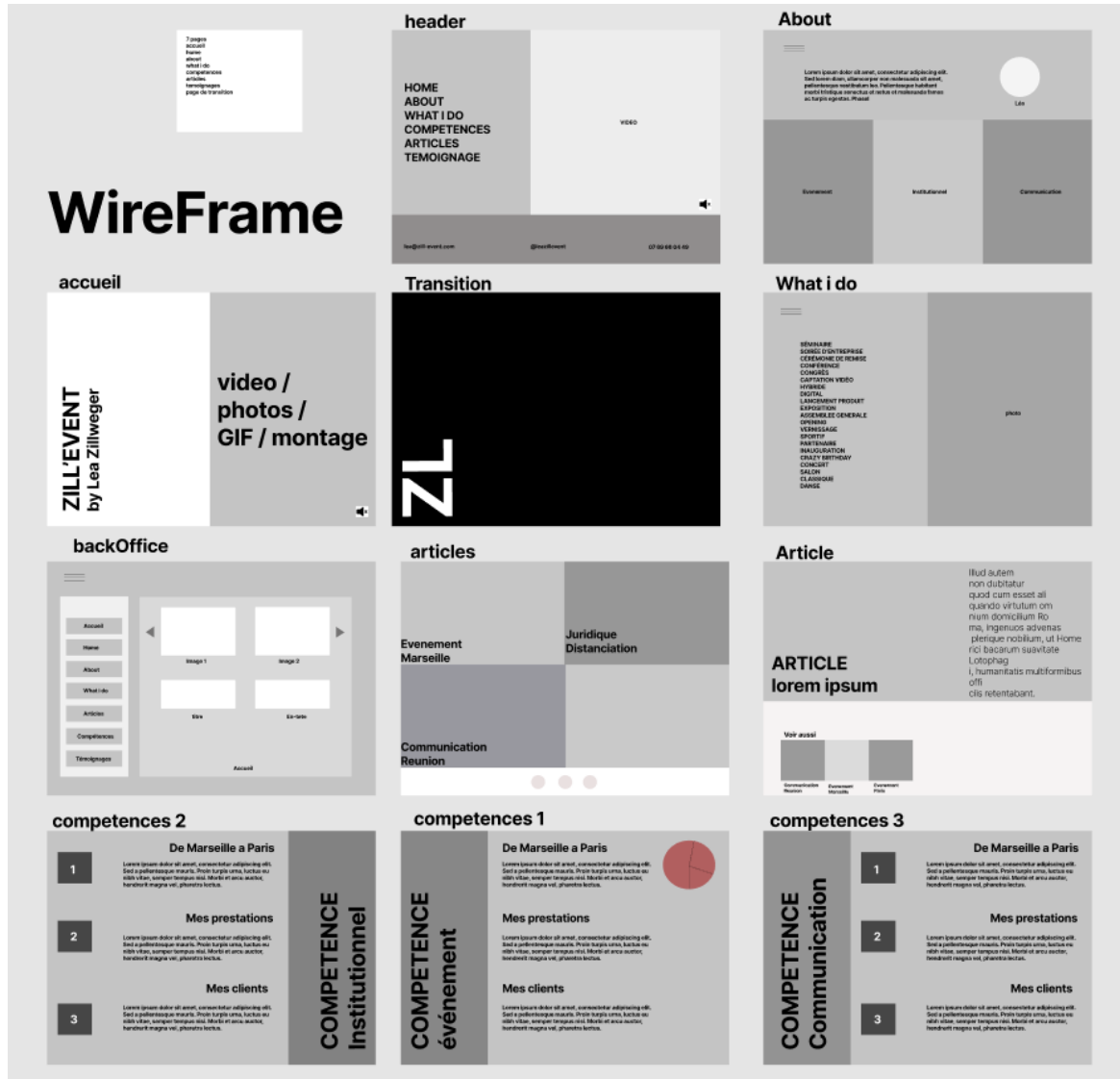
Après plusieurs réunions, nous nous sommes concertés sur les fonctionnalités qu'offrirait son site ainsi que sur le design qu'elle voudrait y voir apparaître.

Zoning :



DOSSIER PROFESSIONNEL (DP)

WireFrame :



DOSSIER PROFESSIONNEL (DP)

Maquette :



DOSSIER PROFESSIONNEL ^(DP)

2. Précisez les moyens utilisés :

J'ai réalisé cette maquette grâce à Figma.

3. Avec qui avez-vous travaillé ?

J'ai collaboré avec Huong-May Nguyen sur ce projet

4. Contexte

Nom de l'entreprise, organisme ou association ► *Zill Event.*

Chantier, atelier, service ► *Projet professionnel.*

Période d'exercice ► Du : *Mai 2021* au : *Décembre 2021*

5. Informations complémentaires (facultatif)

DOSSIER PROFESSIONNEL ^(DP)

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 :

Afin de décrire au mieux l'activité de notre cliente, nous avons développé diverses pages qui permettent de promouvoir ses compétences et son expérience

- Une page accueil a été créée
- Une page qui présente tous les types d'événements sur lesquels elle a déjà travaillée. Elle peut changer les images quand elle le souhaite
- Une page de blog sur laquelle elle peut émettre de nouveaux articles sur des sujets qui la concerne
- Une page qui présente ses compétences

Toutes ces pages sont affichées à l'aide du framework Slim qui produit des rendus via Twig (système de template)

C'est un système de routeur qui permet à l'utilisateur d'accéder aux diverses pages. Ainsi l'URL ne possède jamais l'extension des fichiers appelés.

Le site est quasiment totalement responsive aux téléphones

2. Précisez les moyens utilisés :

Slim, Twig, Php, Css, Html, Sql

3. Avec qui avez-vous travaillé ?

J'ai collaboré avec Huong-May Nguyen sur ce projet

DOSSIER PROFESSIONNEL ^(DP)

4. Contexte

Nom de l'entreprise, organisme ou association ► *Zill Event*

Chantier, atelier, service ► *Projet Professionnel*

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

5. Informations complémentaires (facultatif)

DOSSIER PROFESSIONNEL ^(DP)

Activité-type 1

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

Exemple n° 3 ► Développer la partie Back-end d'une interface utilisateur web

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

Afin de faire fonctionner l'interface web du site Zill Event, il a fallu implémenter des fonction sql relié a notre back-end et à notre base de donnée.

Notre site est développé en Model Vue Controller, ce qui nous permet d'organiser facilement nos requêtes selon les routes appelées.

Nos fonction vont permettre :

- L'upload d'image afin de faire changer la façon de promouvoir les événements que notre cliente organise.
- La création d'article de blog
- L'affichage des articles de blogs
- La création d'utilisateur admin
- Le fait de pouvoir changer de mot de passe
- Le fait de pouvoir se connecter en tant qu'admin

2. Précisez les moyens utilisés :

Slim, Twig, Php, Css, Html, Sql

3. Avec qui avez-vous travaillé ?

J'ai collaboré avec Huong-May Nguyen sur ce projet

4. Contexte

Nom de l'entreprise, organisme ou association ► Zill Event

Chantier, atelier, service ► Projet Professionnel

Période d'exercice ► Du : Mai 2021 au : décembre 2021

DOSSIER PROFESSIONNEL ^(DP)

5. Informations complémentaires *(facultatif)*

Activité-type 1 Cliquez ici pour entrer l'intitulé de l'activité

Exemple n°4 ► Développer une interface utilisateur de type desktop

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

Dans le cadre du développement d'un programme qui se lance sur le bureau j'ai réalisé un jeu en python. A l'aide de pygames j'ai généré une boucle de jeu dans laquelle on va calculer l'espace disponible pour se déplacer et ce qu'il s'y passe à chaque instant. Le but de mon jeu est de ramasser des objets, mais aussi de vaincre des adversaires à l'aide de sort.

Dans un premier temps j'ai construit un tableau qui me permet de représenter chaque case du jeu, pour savoir si c'est un mur, une case navigable ou un objet à ramasser sur une case où l'on peut se déplacer.

Mon personnage se déplace de case en case par un effet de remplacement des cases à l'aide de l'utilisation des sprite (images). Une fois la totalité des objets ramassés, le jeu s'arrête et affiche à l'écran "victoire".

Des fonctions permettent d'identifier le contenu des cases vers lesquels on veut naviguer et des fonctions permettent de lancer un sort. (voir annexes python : map, sprite, code)

Le sort est assez complexe puisqu'il est calculé par l'ordinateur. Quand j'appuie sur la touche du sort, il va se déplacer de 1 à 4 cases, cela dépend de si le sort rencontre un obstacle qui lui fera arrêter son animation. Ensuite je demande dans ma boucle de jeu pygames de rappeler plusieurs fonctions de manière récursive, pour que tous les calculs soient donnés en temps réel afin de ne jamais ralentir le jeu et sa fluidité. L'ordinateur va ainsi calculer autant d'image par seconde que possible pendant la durée de l'animation. Ce code dépend donc directement de la partie hardware de l'ordinateur et permet à l'animation de se comporter différemment en fonction de la puissance de chaque ordinateur.

On peut ainsi espérer avoir une animation totalement fluide à partir d'un pc moyen de gamme (plusieurs cœurs sur processeur).

Une fois que l'animation a fini de se déplacer, elle explose.

Sur le même principe de récursivité l'animation va calculer en fonction du temps actuel et du temps prévu pour sa finition. Ainsi, en fonction d'un coefficient et du temps écoulé, 4 images vont se succéder pour donner l'impression que le sort lancé explose une fois qu'il arrête de se déplacer.

DOSSIER PROFESSIONNEL ^(DP)

Une fois la partie terminée il suffit d'appuyer sur échappe pour que pygames détecte que l'on souhaite fermer le programme.

2. Précisez les moyens utilisés :

Pygames , python , Visual Code

3. Avec qui avez-vous travaillé ?

Seul

4. Contexte

Nom de l'entreprise, organisme ou association ► *Moi*

Chantier, atelier, service ►

Période d'exercice ► Du : *Mai 2022* au : *Mai 2022i*

5. Informations complémentaires (facultatif)

Activité-type 2

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

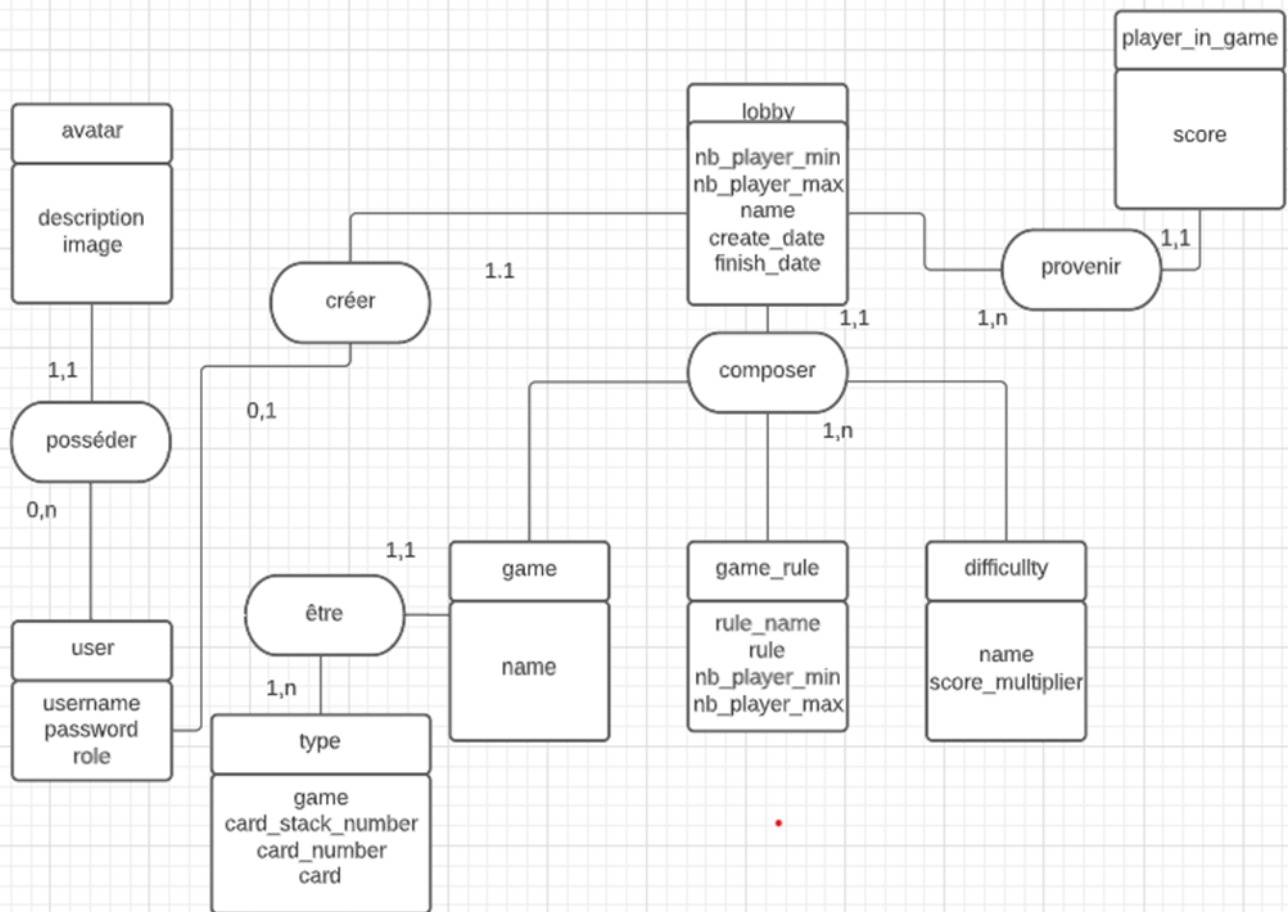
Exemple n° 1 ► Concevoir une base de données

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 table, 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.

DOSSIER PROFESSIONNEL (DP)



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

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)

DOSSIER PROFESSIONNEL (DP)

Exemple n° 2 ► Développer les composant 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 ligne 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 attendu.

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 appelé par n'importe quel champ sur n'importe quelle table. Certains champs en appellent d'autre 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 hermétiques aux actions indésirables.

Dans un autre domaine nous avons créé un server 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 server NestJS mais dans un soucis de compréhension nous avons préféré effectuer cette partie a part.

Pour accéder au server 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.

2. Précisez les moyens utilisés :

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

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 ► *LaPlateforme*

Chantier, atelier, service ►

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° 3 ► 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 }
18
```

(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 façons 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.

DOSSIER PROFESSIONNEL ^(DP)

2. Précisez les moyens utilisés :

Google, nestjs,

3. Avec qui avez-vous travaillé ?

4. Contexte

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'alternant et en dehors des cours que l'on recevait. Nous avons tout d'abord brainstormé sur les différents sujets qui nous intéresseraient 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 personnes 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 fixé. 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. À 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.

DOSSIER PROFESSIONNEL ^(DP)

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 mette 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 oblige 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 test 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, non-compatible 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 :

NestJS, React Native, Socket.io, SecureStore, Css , TypeScript , Javascript, NodeJs, sql, TypeOrm, Class-transformer, react navigation, virtual machine, linux , filleZilla, Trello , gantt...

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)

DOSSIER PROFESSIONNEL ^(DP)

Activité-type 3

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

Exemple n° 2 ► 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 évènement 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 associées au jeu sélectionné.

On nomme alors le lobby et on appuie sur le bouton créé. 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. Une fois ces signaux effectués, le Lobby figurera dans la liste des Lobbies 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 existants qui sont tous répertoriés par leur nom de lobby qui est aussi le nom de la room associée en temps réel. L'utilisateur clique,

DOSSIER PROFESSIONNEL ^(DP)

et rejoins ainsi le lobby et répète les signaux qu'a effectué 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 données via notre API utilisé avec axios dans le serveur socket.

Ainsi, la liste des lobby ne restera jamais remplie de serveur indisponible et sera toujours mise à jour avec soit les lobby disponibles soit les lobby pleins ou déjà en cours de jeu.

2. Précisez les moyens utilisés :

NodeJs, NestJS, TypeScript, Javascript, Socket.io, React Native

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° 3 ► 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 aillant 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 attribué.

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

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

Développé a 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 appel 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 composant métier réagissent en temps réel aux cliques de chacun (lobby, jeu de carte), il nous permet également de mettre en place tous 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'autre 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 même.

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 telephone afin de stocker des token 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 capable de générer une page modulaire (un organisme). Mais a 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 composant dans une navigation Stack (react-navigation) dans laquelle on appel un composant vue qui sera constitué de plusieurs composant qui effectueront des actions plus ou moins indépendante 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 a 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.

DOSSIER PROFESSIONNEL (DP)

2. Précisez les moyens utilisés :

nginx mariadb innodb, nestjs, typescript, js, sql, socket.io, react native , react navigation

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° 4 ► 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 fonctionnalités avec plusieurs types de données et plusieurs cas de figure afin de pouvoir s'assurer que

DOSSIER PROFESSIONNEL (DP)

notre application est bien sécurisée et marche comme on attend qu'elle marche

```
describe( name: 'UsersController', 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,
  });
});
```

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 bien 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 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 ►

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° 5 ► 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 server 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 server. Etant sur Window, nous ne possédons pas de terminal SSH, nous aurions pu en installer un léger mais nous avons préférés tester cela sur une VM. Une fois l'environnement mis en place nous avons pu accéder à notre server via les identifiants utilisateurs qui nous ont été fourni.

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

DOSSIER PROFESSIONNEL ^(DP)

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 transferts 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 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 ▶

Chantier, atelier, service ▶

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

5. Informations complémentaires (facultatif)

Activité- type 1

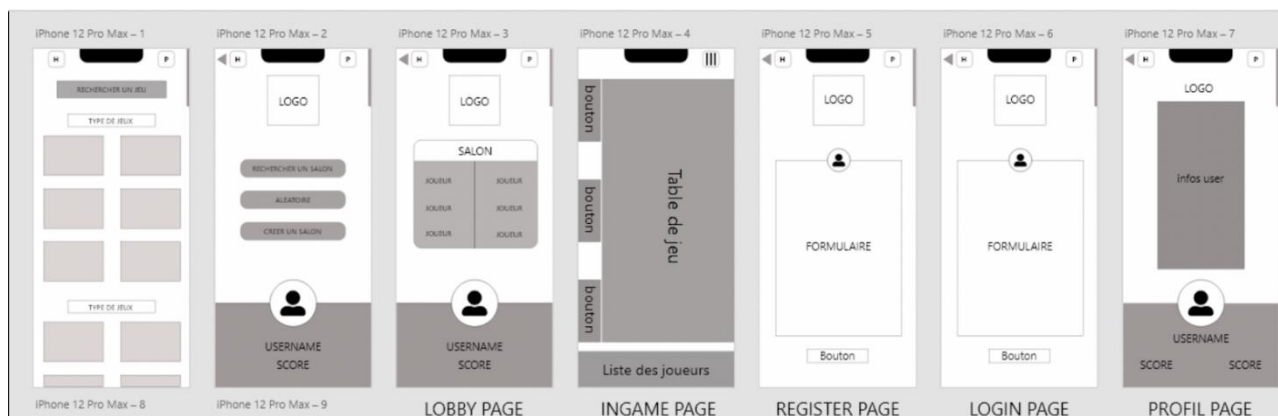
Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité

Exemple n°6 ► Développer une application

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

Après avoir réfléchi sur le concept de notre application, 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 maquettée afin d'avoir une vision d'ensemble sur la structure de l'application:



Nous avons également réalisé une charte graphique afin de définir l'identité visuelle de notre application:

- Nous avons sélectionnés les couleurs suivantes:

DOSSIER PROFESSIONNEL ^(DP)



- Nous avons créé un logo pour l'application:



- Tous les composants comme les boutons pour naviguer, les boutons pour annuler/valider, les entrées utilisateur, auront un style fixe et défini à l'avance.



Nous avons également pensé à utiliser un framework React Native pour styliser notre application, **'Native Base'**, qui est un framework semblable à Bootstrap pour styliser un site web.

2. Précisez les moyens utilisés : React Native, NestJS

DOSSIER PROFESSIONNEL ^(DP)

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 4 pour le maquettage.

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)

DOSSIER PROFESSIONNEL ^(DP)

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] [Cliquez ici pour taper du texte.](#) ,
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 [Cliquez ici pour taper du texte.](#) le [Cliquez ici pour choisir une date](#)
pour faire valoir ce que de droit.

Signature :

DOSSIER PROFESSIONNEL ^(DP)

Documents illustrant la pratique professionnelle

(facultatif)

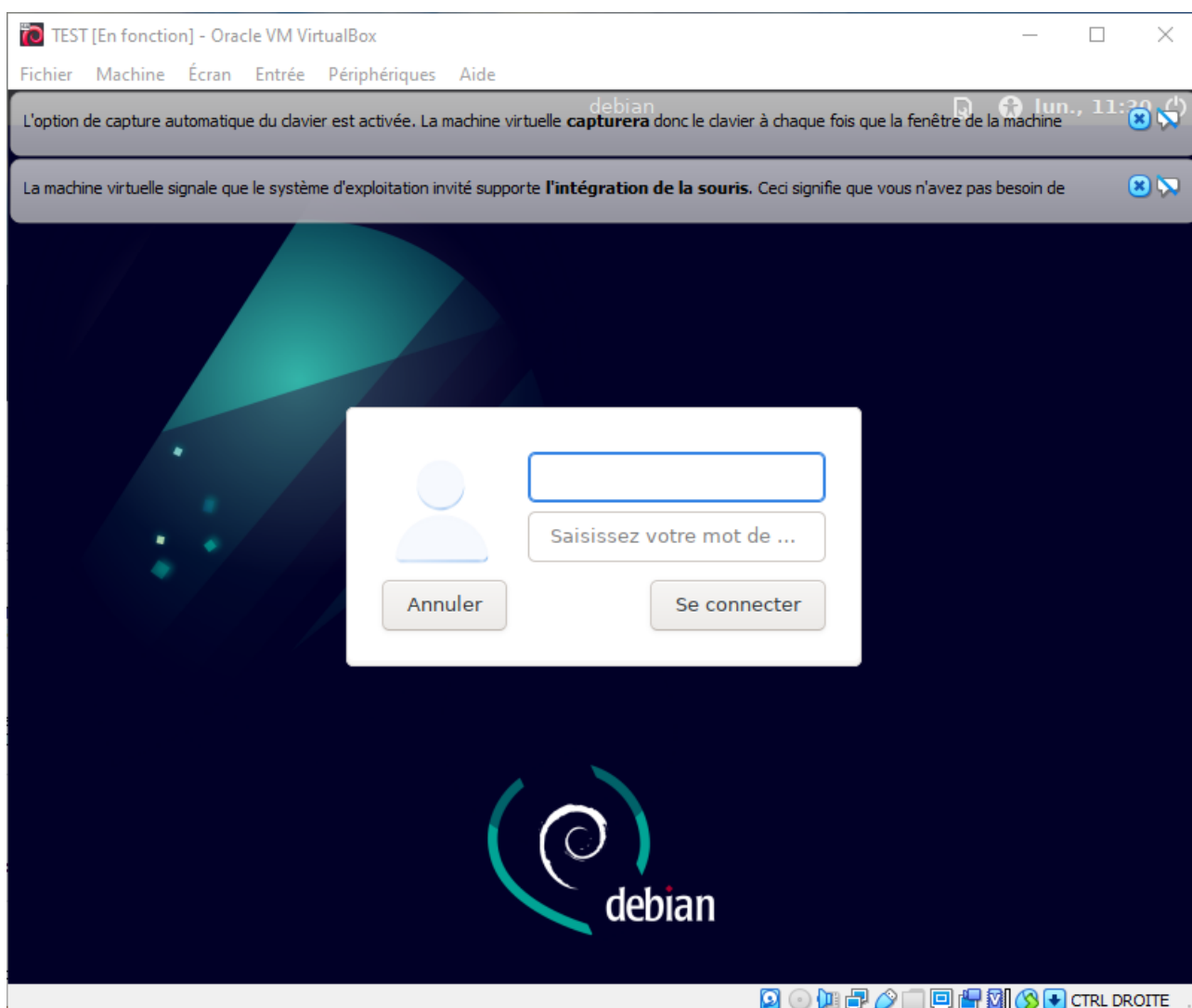
Intitulé
Déploiement
BDD
Développer les composant d'accès aux données :
Travail collaboratif

DOSSIER PROFESSIONNEL ^(DP)

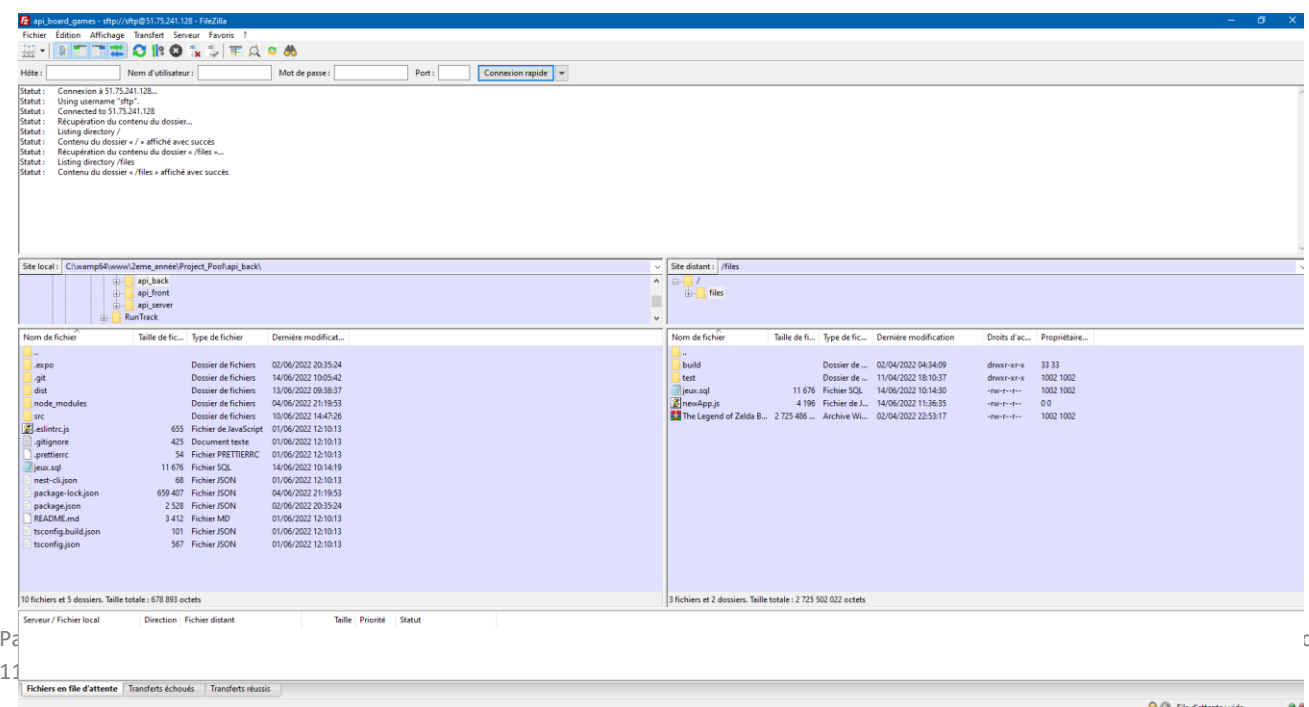
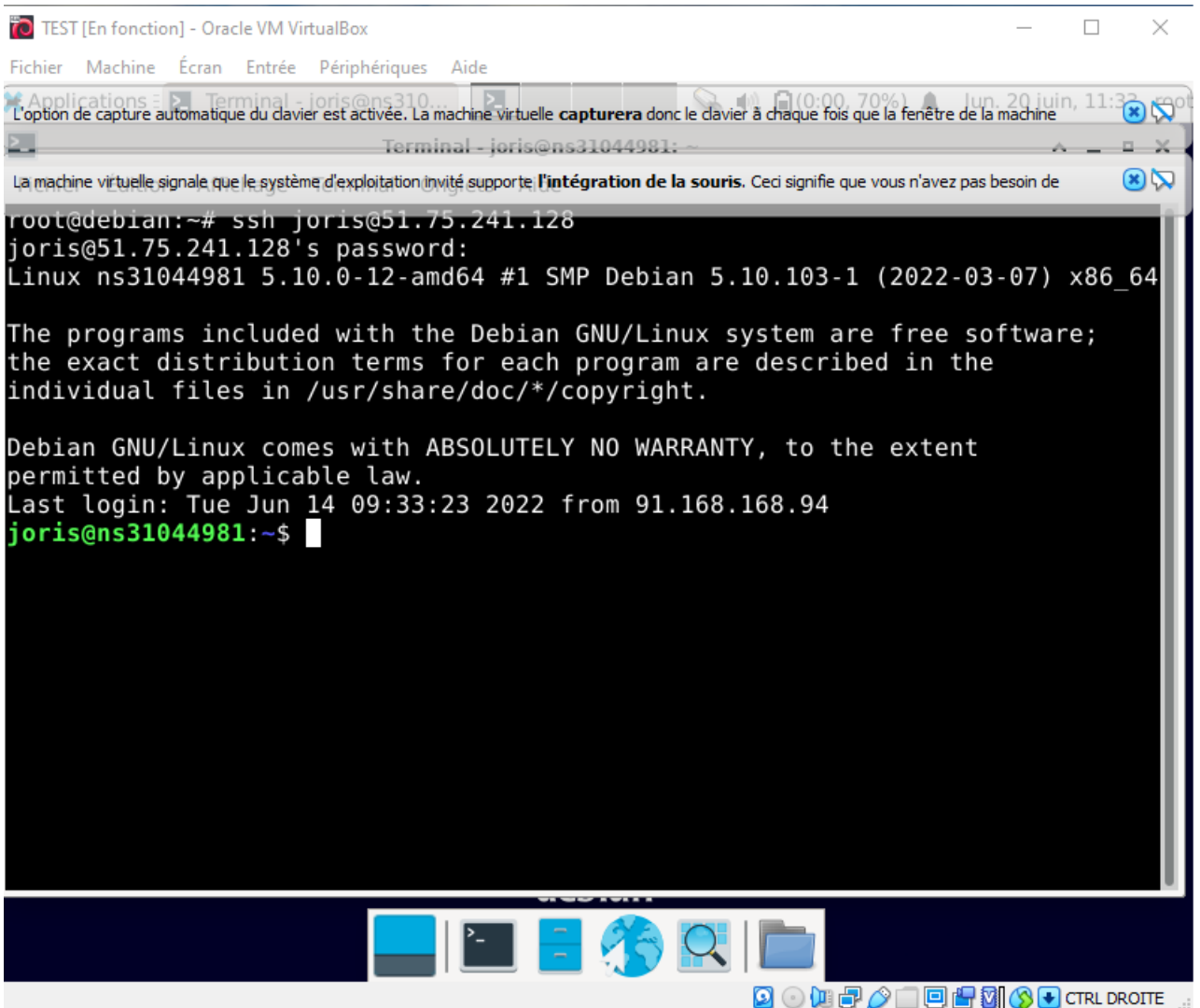
ANNEXES

(Si le RC le prévoit)

Déploiement :



DOSSIER PROFESSIONNEL (DP)

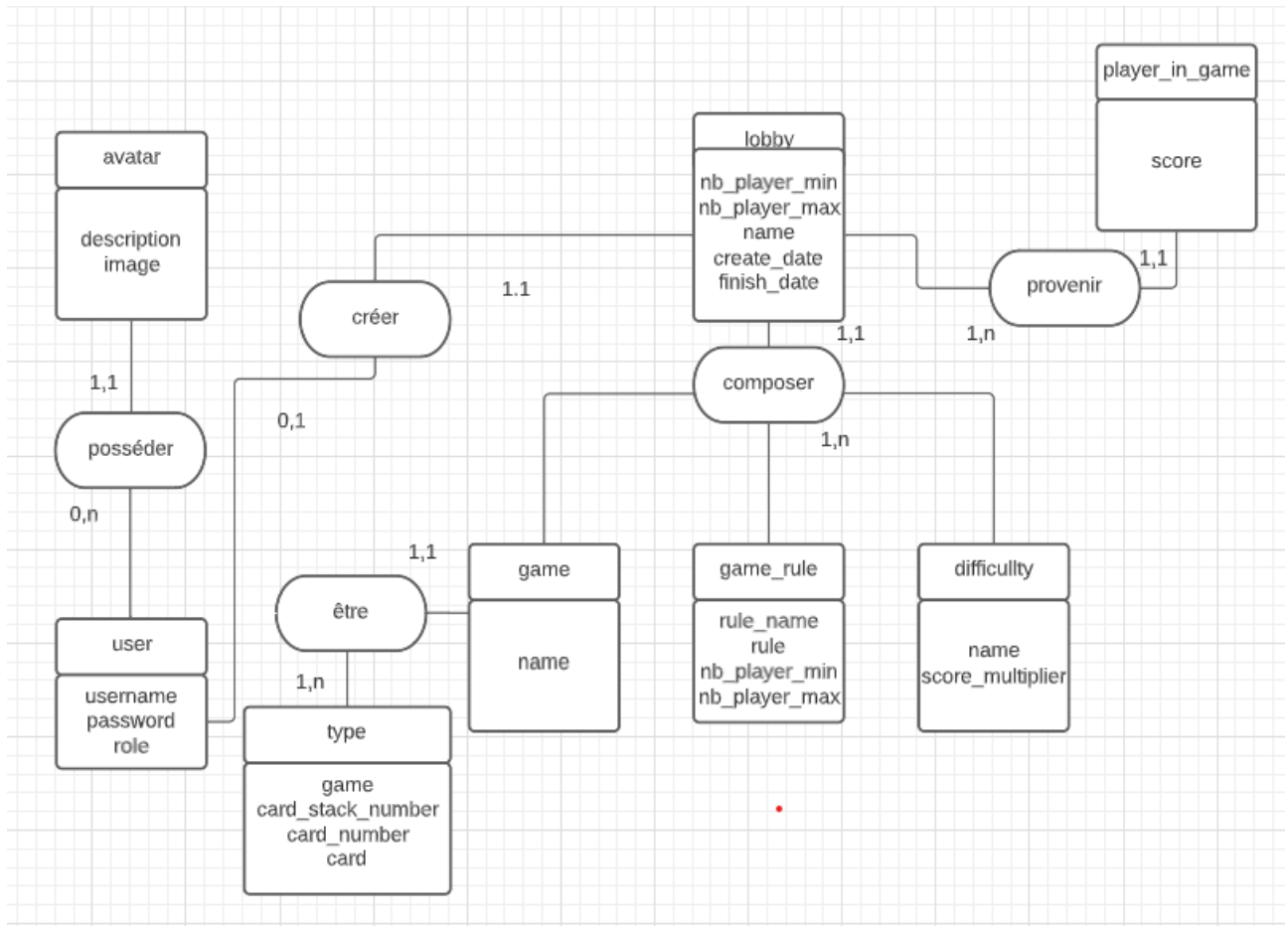


DOSSIER PROFESSIONNEL (DP)

```
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:~$
```

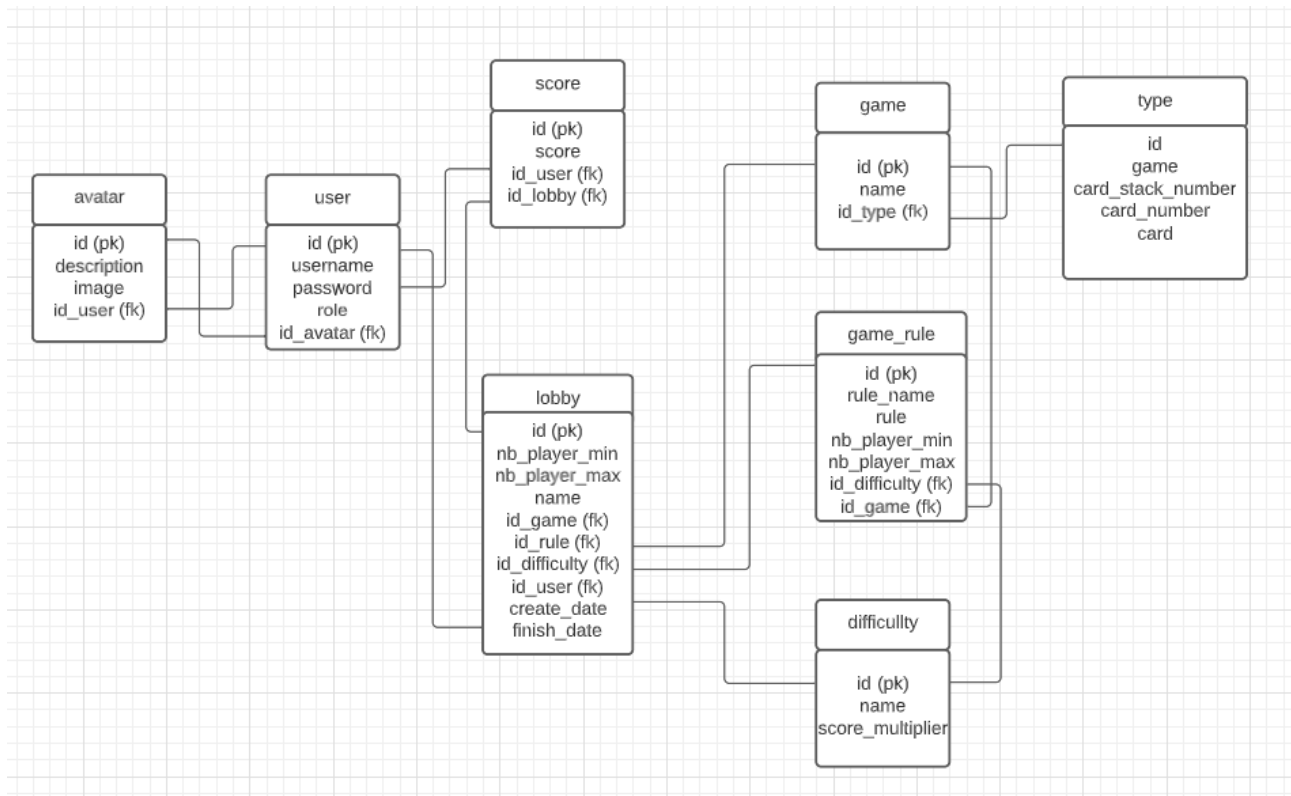
DOSSIER PROFESSIONNEL (DP)

Concevoir une base de donnée:



MCD

DOSSIER PROFESSIONNEL (DP)



MLD

DOSSIER PROFESSIONNEL (DP)

Développer les composant d'accès aux données :

DOSSIER PROFESSIONNEL (DP)

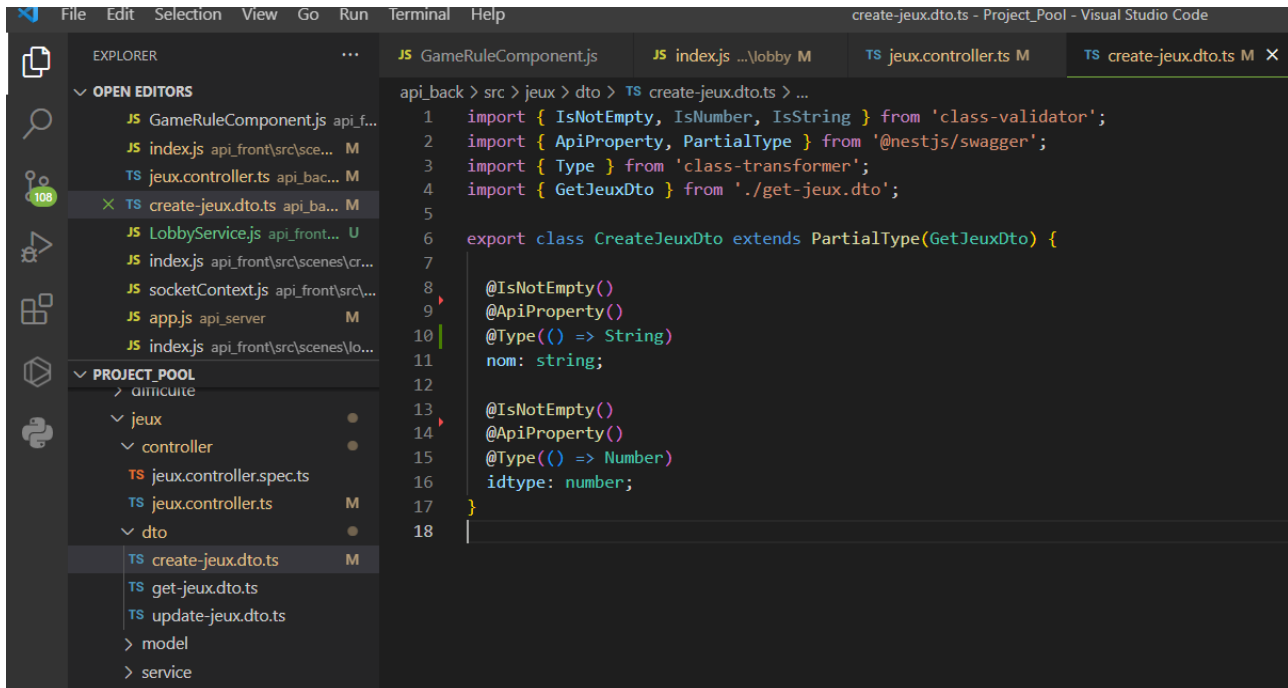
services - controller

```
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 }

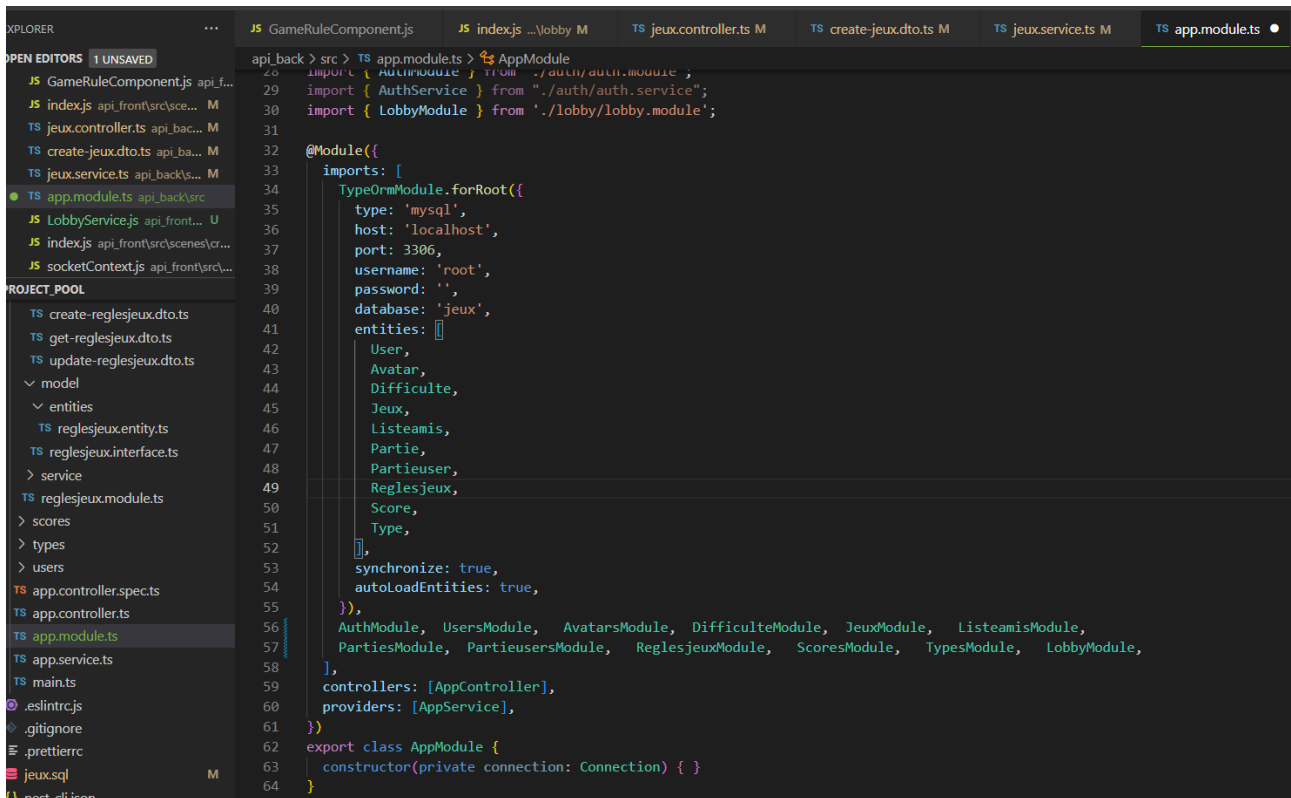
api_back > src > jeux > controller > TS jeux.controller.ts > ...
1 import { Controller, Get, Post, Body, Patch, Param, Delete, Put, Query, } from '@nestjs/common';
2 import { JeuxService } from '../service/jeux.service';
3 import { ApiTags } from '@nestjs/swagger';
4 import { Observable } from 'rxjs';
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         if (Object.keys(filterDto).length) {
36             return this.jeuxService.getGamesWithFilters(filterDto);
37         } else {
38             return this.jeuxService.findAll();
39         }
40     }
41 }
```

DOSSIER PROFESSIONNEL (DP)

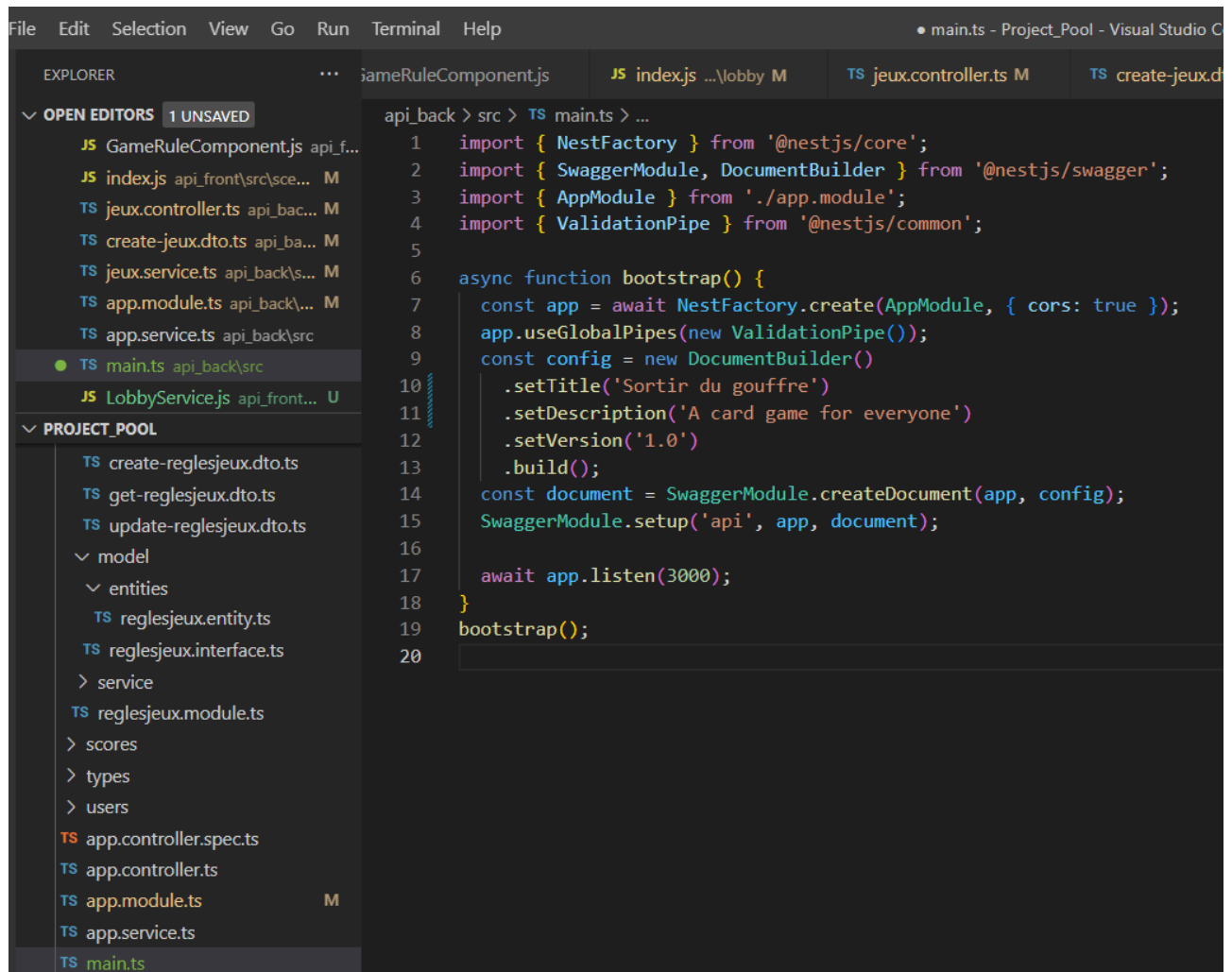
entity ORM



dto ORM



DOSSIER PROFESSIONNEL (DP)



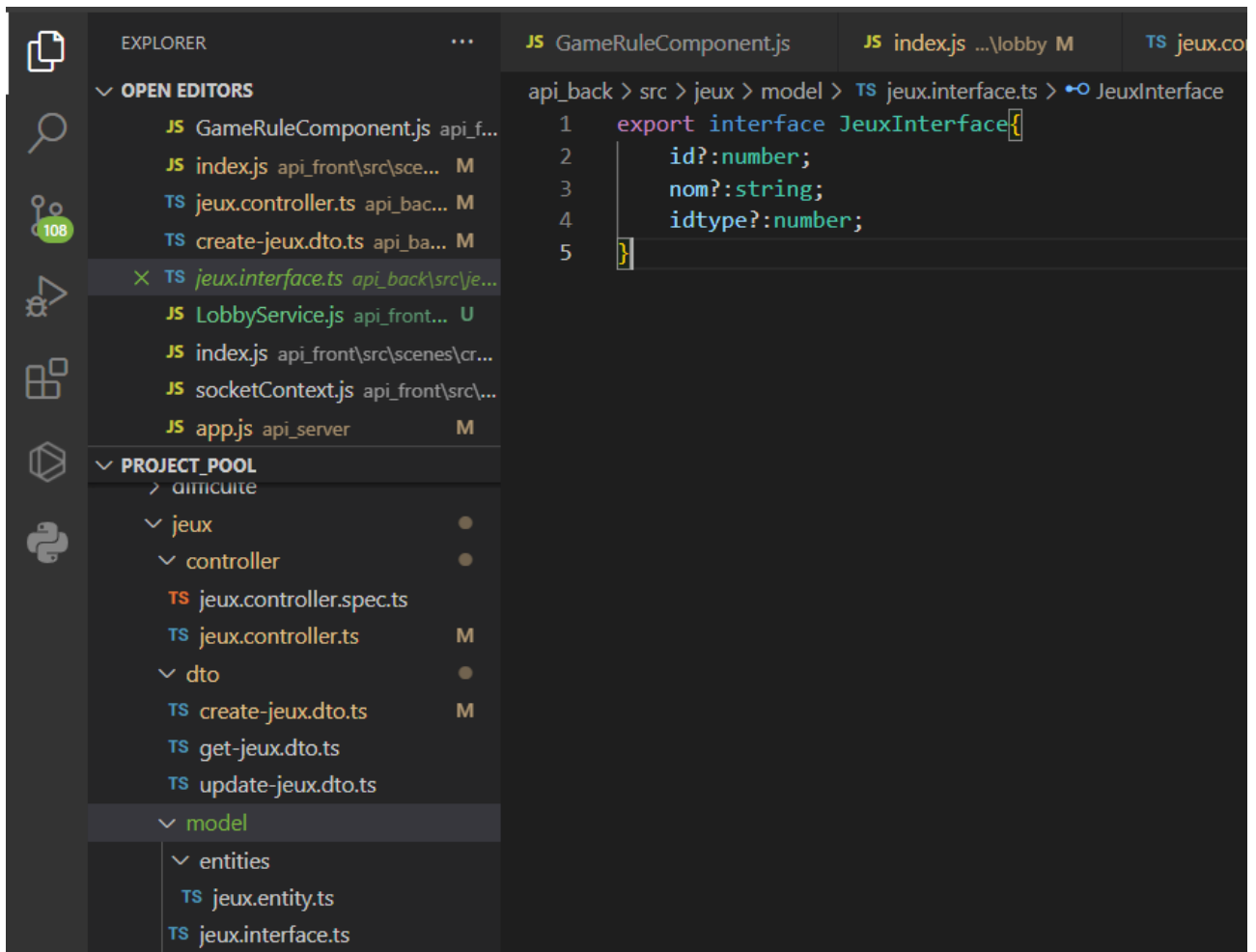
The screenshot shows the Visual Studio Code interface with the 'main.ts' file open. The Explorer sidebar on the left shows the project structure, including 'api_back' and 'api_front' folders. The main editor displays the following TypeScript code:

```
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

DOSSIER PROFESSIONNEL (DP)

interface ORM



DOSSIER PROFESSIONNEL (DP)


Travail collaboratif :

The screenshot displays a GitHub repository interface. At the top, there's a search bar and tabs for 'Overview', 'Yours', 'Active', 'Stale', 'All branches', and a 'New branch' button. Below this, the 'Default branch' is 'master', updated 7 days ago. The 'Your branches' section lists 'manoo' (updated 17 days ago), 'adjust_db_with_new_table' (updated last month), and 'payload' (updated 4 months ago). The 'Active branches' section lists 'manoo' and 'adjust_db_with_new_table'. The 'Stale branches' section lists 'payload'. Each branch entry includes a 'New pull request' button and edit/delete icons. Below the branches, the repository's file list is shown for the 'master' branch, with 4 branches and 0 tags. The file list includes folders like '.expo', 'src', and files like '.eslintrc.js', '.gitignore', '.prettierrc', 'README.md', 'jeux.sql', 'nest-cli.json', 'package-lock.json', 'package.json', 'tsconfig.build.json', and 'tsconfig.json'. Each file entry shows the commit message and the time since the last commit.

Branch	Updated	Commit	Actions
master	Updated 7 days ago by joris-verguldezoone	Default	
manoo	Updated 17 days ago by joris-verguldezoone	5 0	New pull request, edit, delete
adjust_db_with_new_table	Updated last month by joris-verguldezoone	13 0	New pull request, edit, delete
payload	Updated 4 months ago by joris-verguldezoone	18 0	New pull request, edit, delete

File	Commit	Time
.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...
Taitrement 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...

À faire

GETTER/SETTER--GENERIQUE--SOUS FORME DE TABLEAU

AXIOS LOCAL STORAGE--TOKEN

COMPONENT DRAG AND DROP

COMPONENT DE RANDOMISATION POUR LES CARTES

Avoir une structure atomique/ Revoir le concept et les molécules (mdr)

Revoir la navigation et la split en component pqq la c en procédurale

faire jwt pour le front, double check, pour emit vers la bdd et pour changer de component dans le front sans que le token soit modifié

Cryptage Mdp

Update profil

Veille pour mieux afficher les erreur avec expo

Liste d'amis ux, création des button, utilisation des routes, test avec le

En cours

FINIR LA NAV SHUN FDP

atomiser react

transition d'animation des pages

Faire les routes avec axios , lier le front et le back et faire register, connect modify profil

Quand on se connecte, affichage conditionnel pour appeler d'autre modules. On fait des useEffect dans app.js, si un lifecycle detecte un changement sur l'acquisition du token alors on donne l'accès a la page profil, on fetch tous les user pour pouvoir les ajouter a sa liste d'amis , on permet l'accès a la recherche de lobby/ creation de lobby

Terminé

installer variable d'environnement

n cvghj

shun le kinder fdp wih regarde sa va pas se passer comme ça jte ret

ouv jte souleve toi et t

a vue ta mere elle un chien e

finir deck-carte

homeScreen

Register Screen

a retravailler

useEffect pour async a formaliser

Améliorer la récupération de data a l'aise de fonction du style handleSubmit, ou handleChange

Ressources

<https://callstack.github.io/react-native-paper/radio-button.html>

ajout librairie pour <table> npm i react-native-table-component

NE PAS FAIRE DE NPM AUDIT FIX NI UPDATE

<https://reactnavigation.org/docs/screen-options/>

<https://reactnavigation.org/docs/navigation-prop>

<https://reactnavigation.org/docs/use-focus-effect/>

Connexion perdue. Nouvelle tentative...