



MINISTÈRE CHARGÉ
DE L'EMPLOI

DOSSIER PROFESSIONNEL (DP)

<i>Nom de naissance</i>	- Gomez
<i>Nom d'usage</i>	- Gomez
<i>Prénom</i>	- James
<i>Adresse</i>	- 113 impasse des argelas, 83210, Sollies-Toucas

Titre professionnel visé

Concepteur et Développeur d'Application (CDA)

MODALITÉ D'ACCÈS :

- Parcours de formation



MINISTÈRE CHARGÉ
DE L'EMPLOI

DOSSIER PROFESSIONNEL (DP)

- 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 :

DOSSIER PROFESSIONNEL (DP)

- ▶ 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.

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

Sommaire

Exemples de pratique professionnelle

Concevoir et développer des composants d'interface	p. 5
▪ Maquetter une application	p.
▪ Développer une interface utilisateur de type desktop	p.
▪ Développer des composants d'accès aux données	p.
▪ Développer la partie front-end d'une interface utilisateur web	p.
▪ Développer la partie back-end d'une interface utilisateur web	p.
 Concevoir et développer la persistance des données	 p.
▪ Concevoir une base de données	p.

DOSSIER PROFESSIONNEL (DP)

- Mettre en place une base de données p. p.

- Développer des composants dans le langage d'une base de données p. p.

Concevoir et développer une application multicouche p.

- Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement p.

- Concevoir une application p.

- Développer des composants métier p.

- Construire une application organisée en couches p.

- Développer une application mobile p.

- Préparer et exécuter les plans de tests d'une application p.

- Préparer et exécuter le déploiement d'une application p.

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.

DOSSIER PROFESSIONNEL (DP)

EXEMPLES DE PRATIQUE PROFESSIONNELLE

DOSSIER PROFESSIONNEL (DP)

Activité-type 1

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

CP°1 - Maquetter une application

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

Pour maquetter une application, on commence par dessiner des plans simples de l'interface pour organiser les différents éléments. Ensuite, on définit le chemin que l'utilisateur va suivre dans l'application. On peut utiliser des outils pour créer des maquettes interactives et tester l'expérience. Il est aussi important de penser aux différents écrans (ordinateur, mobile).

2. Précisez les moyens utilisés :

On utilise des outils comme Figma pour créer des maquettes interactives. On dessine d'abord des esquisses pour organiser les menus, les boutons, et les visuels, en tenant compte du sens de lecture naturel des utilisateurs (gauche à droite, haut en bas). L'esthétique et l'ergonomie sont essentielles pour capter rapidement l'attention des joueurs. On teste ensuite différentes mises en page, en optimisant pour un accès facile aux fonctions principales

DOSSIER PROFESSIONNEL (DP)

3. Avec qui avez-vous travaillé ?

Collaborateurs de formation et Formateur

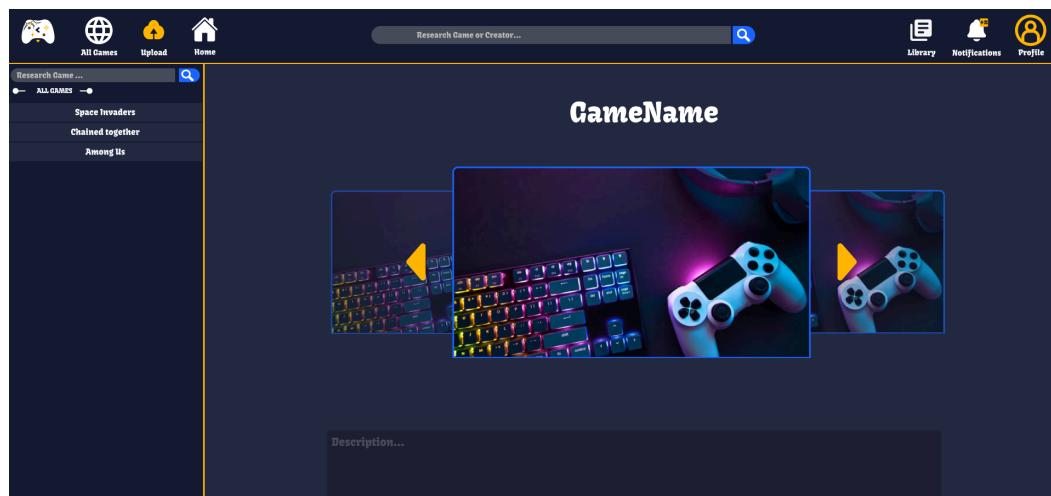
4. Contexte

Nom de l'entreprise, organisme ou association ➔ *LaPlateforme.io*

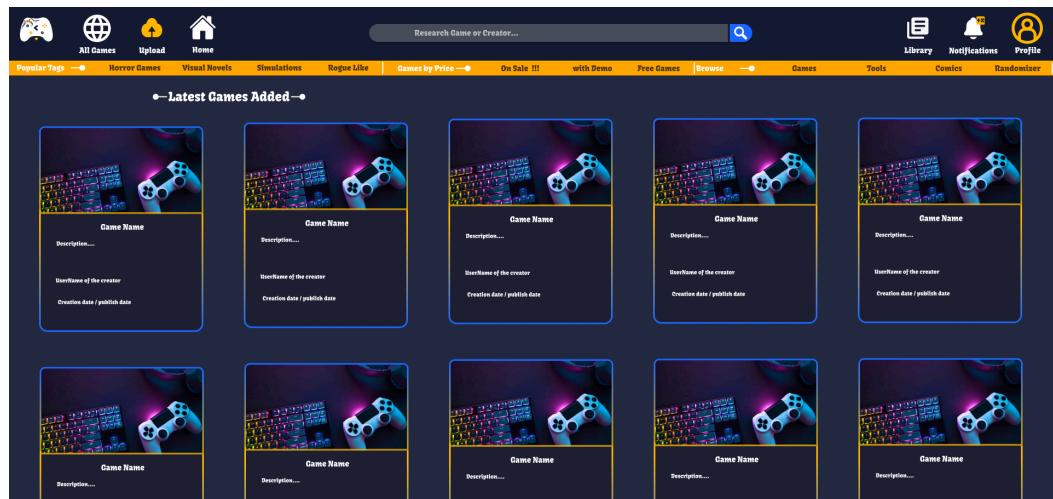
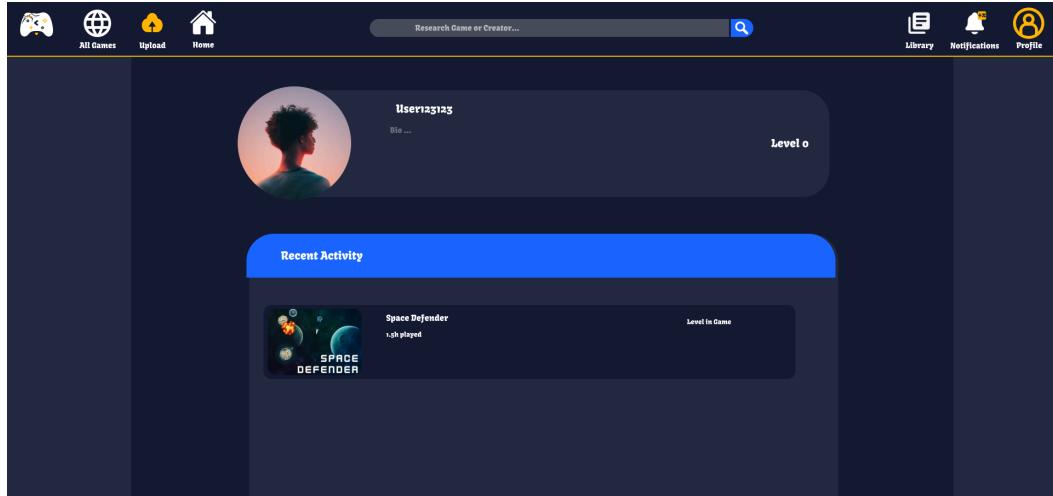
Chantier, atelier, service ➔ *Site de Formation*

Période d'exercice ➔ Du : *04/03/2024* au : *01/10/2024*

5. Informations complémentaires (*facultatif*)



DOSSIER PROFESSIONNEL (DP)



DOSSIER PROFESSIONNEL (DP)

Activité-

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

CP°2 - *Développer une interface utilisateur de type desktop*

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

Pour développer une interface utilisateur de type desktop, on commence par définir la structure de l'application avec des plans simples. Ensuite, on choisit les outils ou frameworks adaptés (comme Electron). On crée les différentes fenêtres, menus, et boutons, en organisant les éléments pour que l'utilisateur puisse naviguer facilement. On s'assure que le design est clair et fonctionnel, avec des boutons bien placés et une navigation fluide.

2. Précisez les moyens utilisés :

Les moyens utilisés sont des outils et frameworks adaptés comme Electron (pour des applications multiplateformes). Les maquettes sont réalisées avec des logiciels comme Figma pour définir l'apparence. Quant aux langages de programmation tels que JavaScript, HTML/CSS (pour Electron), permettent de construire l'interface. La bibliothèque de Bootstrap peut être utilisée pour styliser les composants.

3. Avec qui avez-vous travaillé ?

J'ai travaillé tout seul pour cette réalisation

DOSSIER PROFESSIONNEL (DP)

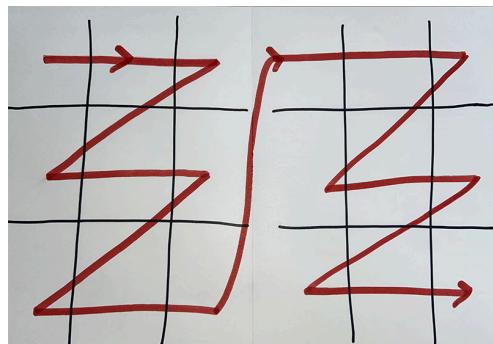
4. Contexte

Nom de l'entreprise, organisme ou association ▶ **LaPlateforme.io**

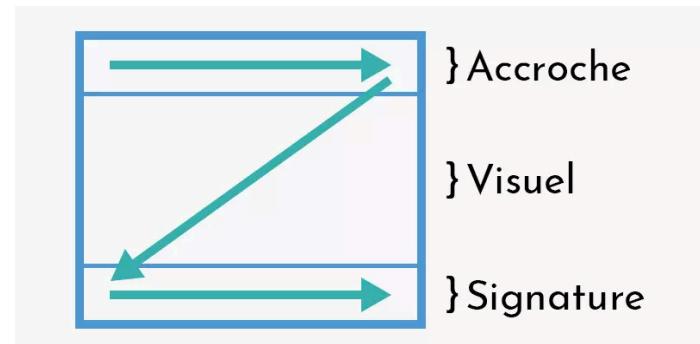
Chantier, atelier, service ▶ **Site de Formation**

Période d'exercice ▶ Du : **04/03/2024** au : **01/10/2024**

5. Informations complémentaires (facultatif)



Sens de lecture Occidentales(Gauche à Droite)



A diagram of a Japanese-style document layout grid. It consists of a 3x4 grid of boxes. The boxes are numbered 1 through 10. Box 1 is at the top center, box 2 is to its right, box 3 is below 1, box 4 is below 2, box 5 is below 4, box 6 is to the left of 5, box 7 is to the left of 6, box 8 is at the top left, box 9 is below 8, and box 10 is below 9. A red arrow points from the top right towards the bottom left, indicating the reading direction.

8	1
9	7
10	3
	2
	4
	6
	5

Sens de lecture Japonais / autres (Droite à Gauche)



Application/framework utilisés

DOSSIER PROFESSIONNEL (DP)

Activité-type 1

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

CP°3 - Développer des composants d'accès aux données

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

Pour développer des composants d'accès aux données SQL, j'ai d'abord analysé les besoins de l'application afin de déterminer quel type de données était mieux adapté à chaque type de base de données. **MySQL** a été choisi pour gérer les données relationnelles avec des entités bien définies et des relations structurées. J'ai conçu les tables SQL en utilisant un ORM comme **Sequelize** pour faciliter la gestion des requêtes et des transactions.

2. Précisez les moyens utilisés :

Pour développer ces composants d'accès, j'ai utilisé des outils simples et efficaces. Pour **MySQL**, j'ai opté pour **Sequelize**, un **ORM** qui facilite la création de tables et la gestion des relations entre les données. Cet outil m'a permis de structurer et d'exécuter des requêtes complexes de manière fluide, tout en assurant une organisation claire et cohérente des données. Sequelize a également simplifié la gestion des transactions et des interactions avec la base de données, garantissant une bonne maintenance et une évolutivité du projet.

3. Avec qui avez-vous travaillé ?

DOSSIER PROFESSIONNEL (DP)

Collaborateur de Formation et Formateur

4. Contexte

Nom de l'entreprise, organisme ou association ➔ *LaPlateforme.io*

Chantier, atelier, service ➔ *Site de Formation*

Période d'exercice ➔ Du : *04/03/2024* au : *01/10/2024*

5. Informations complémentaires (*facultatif*)

DOSSIER PROFESSIONNEL (DP)

```
import { Sequelize } from "../database.js";
import { DataTypes, DECIMAL, DOUBLE, Op, Sequelize, STRING } from "sequelize";
import { Router } from 'express';
import { Controller } from "./Controller.js";
import { Platform } from "./Platform.js";
import { Status } from "./Status.js";
import { Language } from "./Language.js";

export const GameRoute = Router();
export const Game = sequelize.define("Game", {
    title: {...},
    price: {...},
    authorStudio: {...},
    madewith: {...},
    description: {...},
    createdAt: {...},
    updatedAt: {
        type: DataTypes.DATE,
        defaultValue: DataTypes.NOW,
        field: 'updatedAt',
        allowNull: true,
    },
    StatusId: {
        type: DataTypes.INTEGER,
        allowNull: true, // Autoriser NULL pour éviter l'erreur
        references: {
            model: 'Statuses',
            key: 'id',
        },
        onDelete: 'SET NULL',
        onUpdate: 'CASCADE',
    },
    LanguageId: {...}
});

GameRoute.post('/new', async (req, res) => {...});
GameRoute.post('/new/manyGames', async (req, res) => {...});
GameRoute.get("/AllGames", async (req, res) => {...});
```

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

CP°4 - Développer la partie front-end d'une interface utilisateur web

DOSSIER PROFESSIONNEL (DP)

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

Le développement du front-end avec **Angular** commence par la création des composants pour structurer l'interface utilisateur. On définit les templates **HTML**, puis on applique les styles avec **CSS** pour concevoir une mise en page responsive. Les services Angular sont utilisés pour gérer les interactions avec le back-end via des API. La gestion des données se fait avec des **observables** et des fetch dans les services d'Angular. Ensuite, on intègre la navigation avec le **Router** et on ajoute des fonctionnalités interactives.

2. Précisez les moyens utilisés :

On commence par l'utilisation de la maquette réalisée (par exemple avec **Figma**) pour définir la structure visuelle. On crée les composants Angular en suivant la maquette, puis on organise les templates HTML. Les styles sont appliqués avec CSS pour respecter le design prévu. Les services Angular utilisent **fetch** pour effectuer des appels API et gérer les données sans passer par **HttpClient**. La navigation est gérée avec le **Router d'Angular**, et les tests avec **Jasmine** et **Karma** permettent de vérifier la fiabilité de l'interface avant le déploiement.

3. Avec qui avez-vous travaillé ?

Durant cette exercice j'ai travaillé tout seul avec les connaissances et des recherches webs

4. Contexte

Nom de l'entreprise, organisme ou association ▶

LaPlateforme.io

DOSSIER PROFESSIONNEL (DP)

Chantier, atelier, service ▶

Site de Formation

Période d'exercice ▶ Du : **04/03/2024** au : **01/10/2024**

5. Informations complémentaires (*facultatif*)

```
✓ NG-PLAYFORGE
  > .angular
  > .vscode
  > node_modules
  > public
    ✓ src
      > app
        ✓ components
          > dumb
          > smarts
            > carousel
              # carousel.component.css
              ⚡ carousel.component.html
              TS carousel.component.spec.ts
              TS carousel.component.ts
            > game
            > game-date-list
            > nav-item
            > select-components
            > sign-in
        > services
          TS data-fetch.service.spec.ts
          TS data-fetch.service.ts
          TS file-service.service.ts
          TS file.service.spec.ts
          TS game.service.spec.ts
          TS game.service.ts
          ⚡ index.html
          TS main.ts
          # styles.css
          .editorconfig
          .gitignore
        {} angular.json
```

DOSSIER PROFESSIONNEL (DP)

Activité-type 1

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

CP°5 - Développer la partie back-end d'une interface utilisateur web

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

Le développement du back-end d'une interface utilisateur web commence par la conception de l'architecture du serveur, souvent avec des frameworks comme Express / Cors. On crée des API pour gérer la communication entre le front-end et la base de données. La gestion des données se fait via des ORM comme Sequelize (pour SQL). On implémente des contrôleurs pour traiter les requêtes HTTP, et des middlewares pour la sécurité, l'authentification et la validation des données.

2. Précisez les moyens utilisés :

Pour le développement de la partie **back-end**, j'ai utilisé **Node.js** avec **Sequelize** pour gérer les interactions avec la base de données **MySQL**. J'ai structuré deux **API** distinctes : l'une pour le backend qui gère les données métiers, et l'autre dédiée à l'upload des fichiers (images, jeux, etc.). Avec Sequelize, j'ai défini les modèles et les relations entre les tables, facilitant ainsi la gestion des données. J'ai également intégré Cors pour permettre la communication sécurisée entre le front-end et le back-end. Enfin, chaque API gère ses fonctionnalités spécifiques tout en garantissant une architecture propre et modulaire.

DOSSIER PROFESSIONNEL (DP)

3. Avec qui avez-vous travaillé ?

J'ai travaillé cette partie avec mon formateur.

4. Contexte

Nom de l'entreprise, organisme ou association ➔ *LaPlateforme.io*

Chantier, atelier, service ➔ *Site de Formation*

Période d'exercice ➔ Du : *04/03/2024* au : *01/10/2024*

DOSSIER PROFESSIONNEL (DP)

5. Informations complémentaires (*facultatif*)

 api-boutique	27/08/2024 14:33	Dossier de fichiers
 api-upload	19/09/2024 16:06	Dossier de fichiers

```
import express from "express";
import cors from "cors";
import { FileRoute } from './Models/File.js';
import { ImageRoute } from './Models/Image.js';
import path from "path";

const app = express();
app.use(express.json());
app.use('/uploads', express.static(path.join(__dirname,'uploads')));

// Configuration CORS
app.use(cors({
  origin: 'http://localhost:4200', // Autorise uniquement l'origine de ton front-end
  methods: ['GET', 'POST', 'PUT', 'DELETE'],
  allowedHeaders: ['Content-Type', 'Authorization'],
  credentials: true // Permet l'envoi des cookies/sessions
}));

// Routes
app.use('/game', FileRoute);
app.use('/game', ImageRoute);

// Démarrer le serveur
app.listen(9091, () => {
  console.log("Server on port 9091");
});
```

Api-Upload/App.ts

```
import cors from "cors";
import express from "express";

import { Request, Response } from 'express';
import { Game, GameRoute } from './Models/Game.js';
import { User } from './Models/User.js';
import { Controller, ControllerRoute } from './Models/Controller.js';
import { Tag, TagRoute } from './Models/Tag.js';
import { Status, StatusRoute } from './Models>Status.js';
import { Platform, PlatformRoute } from './Models/Platform.js';
import { Genre, GenreRoute } from './Models/Genre.js';
import { DataRoute } from './Data.js';
import { Language, LanguageRoute } from './Models/Language.js';
import { Role } from './Models/Role.js';
import { Cart } from './Models/Cart.js';

const app = express();
app.use(express.json());

// error failed to fetch --> Cors head
app.use((req, res, next) => {
  res.setHeader('Access-Control-Allow-Origin', '*'); // Ou spécifiez le domaine explicitement
  res.setHeader('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE');
  res.setHeader('Access-Control-Allow-Headers', 'Content-Type, Authorization');
  next();
});

app.use(cors({
  origin: 'http://localhost:4200', // Remplace par l'URL de ton Front-end
  methods: ['POST', 'GET', 'PUT', 'DELETE'],
  credentials: true
}));
```

Api-Backend/App.ts

DOSSIER PROFESSIONNEL (DP)

Activité-type 2

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

CP° 1 - Concevoir une base de données

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

Pour concevoir une base de données, j'ai d'abord commencé par analyser les besoins du projet afin d'identifier les entités principales et leurs interactions. J'ai ensuite créé un diagramme de cas d'utilisation (**Use Case**) en UML pour modéliser les interactions entre les utilisateurs et le système. Ensuite, j'ai élaboré un diagramme entité-relation (**ER**) pour visualiser les entités, leurs attributs, et les relations entre elles. Enfin, j'ai construit un modèle conceptuel de données (**MCD**) en UML, représentant les entités avec leurs clés primaires et étrangères, ainsi que les relations pour structurer les tables et la base de données finale.

2. Précisez les moyens utilisés :

J'ai d'abord utilisé du support papier pour esquisser les premiers concepts et relations. Ensuite, j'ai créé les diagrammes UML, notamment le diagramme de cas d'utilisation (**Use Case**) et le modèle conceptuel de données (**MCD**), avec Draw.io pour formaliser les entités, attributs et relations. Pour documenter et partager les schémas, j'ai utilisé du Markdown sur GitHub, ce qui m'a permis d'organiser et présenter clairement le projet.

DOSSIER PROFESSIONNEL (DP)

3. Avec qui avez-vous travaillé ?

J'ai travaillé en collaboration avec mon formateur et des apprenants en formations également.

4. Contexte

Nom de l'entreprise, organisme ou association ➔ *LaPlateforme.io*

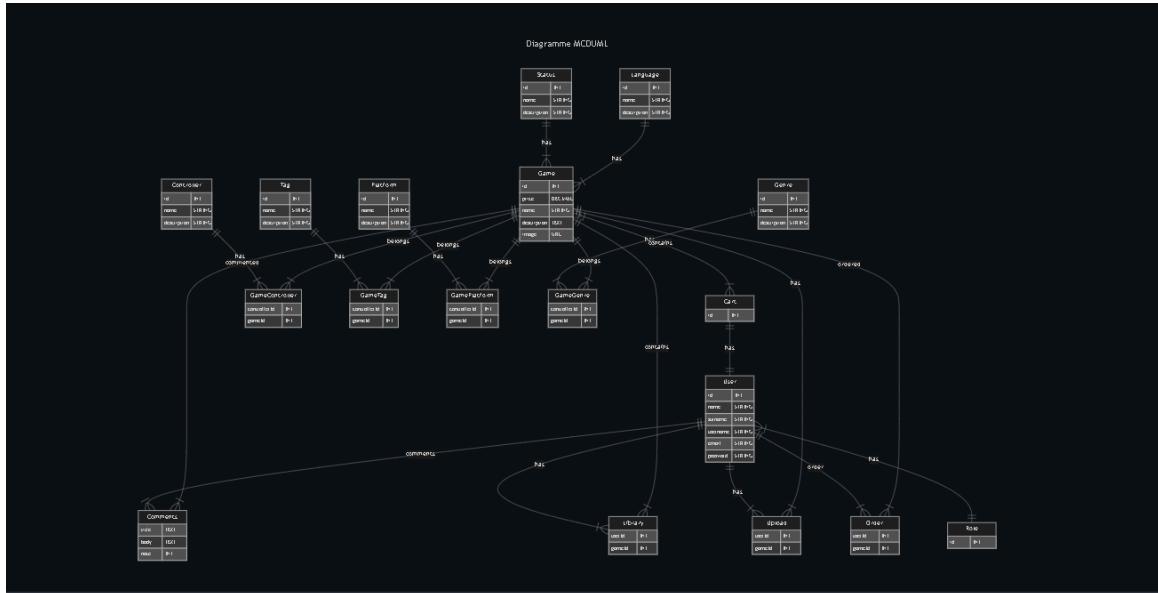
Chantier, atelier, service ➔ *Site de Formation*

Période d'exercice ➔ Du : *04/03/2024* au : *01/10/2024*

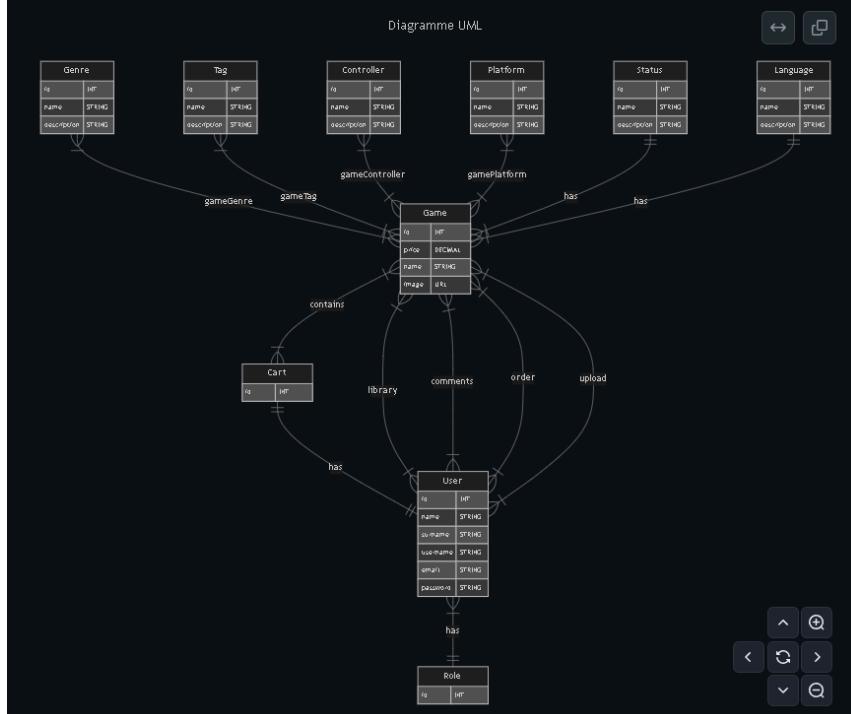
DOSSIER PROFESSIONNEL (DP)

5. Informations complémentaires (facultatif)

<https://github.com/QLFJameS/api-boutique?tab=readme-ov-file>



Entity Relation Diagram UML



DOSSIER PROFESSIONNEL (DP)

Activité-type 2 Concevoir et développer la persistance des données en intégrant les recommandations de sécurité

CP° 2 - Mettre en place une base de données

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

D'abord, j'ai défini les entités principales et leurs relations en réalisant des diagrammes UML avec Draw.io. Ensuite, j'ai utilisé le support papier pour structurer et vérifier l'architecture conceptuelle. Après validation, j'ai documenté ces étapes sur GitHub en utilisant Markdown pour partager les détails avec l'équipe. Une fois le schéma validé, j'ai utilisé Sequelize pour créer les modèles et relations dans la base de données MySQL, en m'assurant de bien configurer les clés primaires et étrangères. Enfin, j'ai testé l'intégrité des données et les relations entre les tables pour garantir le bon fonctionnement du système.

2. Précisez les moyens utilisés :

j'ai utilisé Sequelize comme **ORM** pour interagir avec MySQL. D'abord, après avoir validé les schémas des entités et relations avec les diagrammes UML sur Draw.io, j'ai créé les modèles Sequelize pour chaque entité. J'ai défini les champs, types de données, ainsi que les relations (one-to-many, many-to-many) en utilisant les méthodes **belongsTo**, **hasMany**, et **belongsToMany** de Sequelize. J'ai ensuite utilisé les migrations pour générer automatiquement les tables dans MySQL. Enfin, j'ai testé les relations et les requêtes pour assurer que les interactions entre les modèles et la base de données fonctionnaient correctement avec **Postman**.

DOSSIER PROFESSIONNEL (DP)

3. Avec qui avez-vous travaillé ?

J'ai testé mes relations et mes requêtes en utilisant Postman pour vérifier les réponses de l'API. Ensuite, j'ai validé leur bon fonctionnement en les intégrant avec mon front-end connecté, garantissant ainsi une communication fluide entre l'interface utilisateur et la base de données.

4. Contexte

Nom de l'entreprise, organisme ou association ➔ **LaPlateforme.io**

Chantier, atelier, service ➔ **Site de Formation**

Période d'exercice ➔ Du : **04/03/2024** au : **01/10/2024**

DOSSIER PROFESSIONNEL (DP)

5. Informations complémentaires (*facultatif*)

```
//relations
Game.belongsToMany(User, { through: "Comment" });
User.belongsToMany(Game, { through: "Comment" });

Game.belongsToMany(User, { through: "Library" });
User.belongsToMany(Game, { through: "Library" });

Game.belongsToMany(User, { through: "Upload" });
User.belongsToMany(Game, { through: "Upload" });

User.belongsToMany(Game, { through: "Order" });
Game.belongsToMany(User, { through: "Order" });

Game.belongsToMany(Controller, { through: 'GameControllers' });
Controller.belongsToMany(Game, { through: 'GameControllers' });

Game.belongsToMany(Tag,{through:"GameTag"});
Tag.belongsToMany(Game,{through:"GameTag"});

Game.belongsTo(Status, { foreignKey: 'StatusId', as: 'status' });
StatushasMany(Game, { foreignKey: 'StatusId', as: 'games' });

UserhasOne(Role);
RolehasMany(User);

Game.belongsToMany(Platform,{through:"GamePlatforms"});
Platform.belongsToMany(Game,{through:"GamePlatforms"});

Game.belongsTo(Language, { foreignKey: 'LanguageId' as 'language'});
LanguagehasMany(Game, { foreignKey: 'LanguageId' as 'games'});

Game.belongsToMany(Genre, { through: "GameGenre" });
Genre.belongsToMany(Game, { through: "GameGenre" });

Cart.belongsToMany(Game,{through:"GameCart"});
Game.belongsToMany(Cart,{through:"GameCart"});

UserhasOne(Cart);
Cart.belongsTo(User);
```

Relations de mon api-Backend

The screenshot shows the Postman interface with a POST request to `http://localhost:9090/statuses/new`. The 'Params' tab is selected, showing two query parameters: 'Key' and 'Key'. Other tabs include Authorization, Headers (8), Body, Pre-request Script, Tests, and Settings.

Test de Route avec Postman

(création de Status)

DOSSIER PROFESSIONNEL (DP)

Activité-type 2

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

CP° 3 - *Développer des composants dans le langage d'une base de données*

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

J'ai commencé par définir les modèles correspondant aux tables, en spécifiant les champs, types de données, et relations (clés primaires, étrangères). J'ai ensuite utilisé un ORM comme Sequelize pour automatiser la gestion des interactions entre les modèles et la base de données. J'ai créé des migrations pour générer les tables et maintenir la structure de la base. Enfin, j'ai mis en place des requêtes SQL personnalisées pour effectuer des opérations complexes (jointures, filtres) et testé ces composants via des outils comme Postman pour valider leur bon fonctionnement.

2. Précisez les moyens utilisés :

J'ai utilisé les méthodes Sequelize comme **define** pour structurer les modèles et **associate** pour gérer les relations (comme **belongsTo**, **hasMany**, **belongsToMany**). Ensuite, j'ai mis en place des migrations avec Sequelize CLI pour créer et mettre à jour les tables dans la base de données. Pour les requêtes complexes, j'ai utilisé des méthodes comme **findAll**, **findById**, et des jointures via **include**. Enfin, j'ai testé ces composants avec des appels API dans Postman pour vérifier la cohérence et les interactions entre les modèles.

3. Avec qui avez-vous travaillé ?

J'ai travaillé en collaboration avec mon formateur et des recherches webs

DOSSIER PROFESSIONNEL (DP)

4. Contexte

Nom de l'entreprise, organisme ou association ➤ **LaPlateforme.io**

Chantier, atelier, service ➤ **Site de Formation**

Période d'exercice ➤ Du : **04/03/2024** au : **01/10/2024**

5. Informations complémentaires (*facultatif*)

```
export const Game = sequelize.define("Game", {
  title: {...},
  price: {...},
  authorStudio: {...},
  madewith: {...},
  description: {...},
  createdAt: {...},
  updatedAt: {
    type: DataTypes.DATE,
    defaultValue: DataTypes.NOW,
    field: 'updatedAt',
    allowNull: true,
  },
  StatusId: {
    type: DataTypes.INTEGER,
    allowNull: true, // Autoriser NULL pour éviter l'erreur
    references: {
      model: 'Statuses',
      key: 'id',
    },
    onDelete: 'SET NULL',
    onUpdate: 'CASCADE',
  },
  LanguageId: {...}
});
```

Définition du Modèle Game

avec `sequelize.define`

DOSSIER PROFESSIONNEL (DP)

```
// Rechercher et associer les contrôleurs
const controllers = await Controller.findAll({ where: { id: controllerIds } });
if (!controllers.length) {
  return res.status(400).json({ error: 'Aucun contrôleur trouvé pour les IDs spécifiés' });
}
await sequelize.models.GameControllers.create({
  GameId : GameId,
  ControllerId : ControllerId
});
```

Association dans la table de jointure GameControllers de GameId et ControllerId

DOSSIER PROFESSIONNEL (DP)

Activité-type 3

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

CP° 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 :

Mon formateur m'a demandé de configurer un environnement de travail adapté à mon rôle de Concepteur Développeur d'Applications. J'ai donc accédé à un moteur de recherche installé sur l'ordinateur. J'y ai recherché les logiciels nécessaires, puis je les ai téléchargés et installés, notamment Postman, Visual Studio Code, IntelliJ IDEA, Docker Desktop, Git Bash et Unity, pour disposer de tous les outils indispensables à mon travail.

2. Précisez les moyens utilisés :

Pour configurer un environnement de travail en tant que Concepteur Développeur d'Applications, plusieurs outils sont indispensables :

1. **Postman** : pour tester et gérer les API en simulant des requêtes HTTP.
2. **Visual Studio Code** : un éditeur de code léger et performant, adapté à de nombreux langages.
3. **IntelliJ IDEA** : un IDE puissant, surtout utilisé pour le développement Java.
4. **Docker Desktop** : pour créer des environnements de développement en conteneurs, facilitant le déploiement et la gestion d'applications.
5. **Git Bash** : un terminal pour gérer des projets Git en ligne de commande.
6. **Unity** : pour le développement de jeux vidéo en 2D et 3D.

Ces outils permettent de couvrir les besoins de développement, test, gestion des versions, et environnement virtuel.

DOSSIER PROFESSIONNEL (DP)

3. Avec qui avez-vous travaillé ?

En Collaboration avec mon formateur

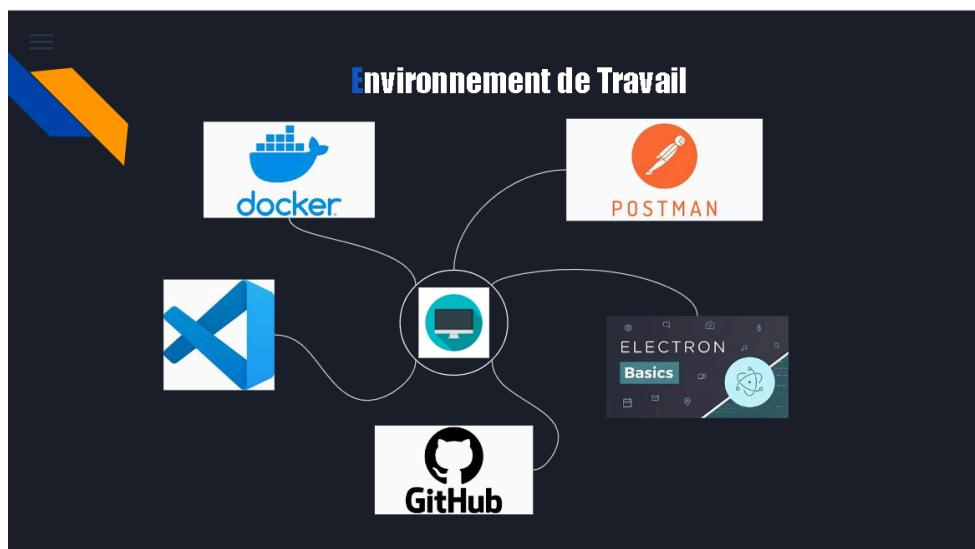
4. Contexte

Nom de l'entreprise, organisme ou association ➔ **LaPlateforme.io**

Chantier, atelier, service ➔ **Site de Formation**

Période d'exercice ➔ Du : **04/03/2024** au : **01/10/2024**

5. Informations complémentaires (*facultatif*)



DOSSIER PROFESSIONNEL (DP)

Activité-type 3 Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité

CP° 2 - *Concevoir une application*

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

Il faut d'abord analyser les besoins du projet et définir les fonctionnalités ainsi que le public cible. Ensuite, on planifie le développement en choisissant les technologies et outils nécessaires. Vient ensuite la phase de conception, où l'on crée des maquettes et des diagrammes pour structurer l'architecture logicielle. Le développement commence par le back-end (base de données, API), suivi du front-end (interface utilisateur). Des tests unitaires et d'intégration sont effectués pour valider le bon fonctionnement. Il est essentiel d'utiliser des outils comme Postman, VSCode et Git pour assurer un environnement de travail efficace.

2. Précisez les moyens utilisés :

voici les étapes clés de la méthode à suivre :

1. **Analyse des besoins** : Identifier les objectifs, le public cible, et les fonctionnalités attendues. Cette étape inclut la rédaction de spécifications fonctionnelles.
2. **Conception technique** : Définir l'architecture logicielle avec des outils comme UML pour créer des diagrammes (UseCase, ER, etc.). On conçoit également l'interface utilisateur via des maquettes (Figma, Draw.io).
3. **Choix des technologies** : Sélectionner les langages, frameworks (Angular, Node.js), et outils adaptés (Postman, Docker).
4. **Développement** : Implémenter d'abord le back-end (base de données, API), puis le front-end (interface utilisateur).
5. **Tests** : Effectuer des tests unitaires et d'intégration pour garantir le bon fonctionnement.
6. **Mise en production** : Préparer et déployer l'application dans l'environnement final.

DOSSIER PROFESSIONNEL (DP)

3. Avec qui avez-vous travaillé ?

Collaboration avec le formateur ainsi que des apprenants de la formation

4. Contexte

Nom de l'entreprise, organisme ou association ➔ *LaPlateforme.io*

Chantier, atelier, service ➔ *Site de Formation*

Période d'exercice ➔ Du : *04/03/2024* au : *01/10/2024*

DOSSIER PROFESSIONNEL (DP)

5. Informations complémentaires (facultatif)

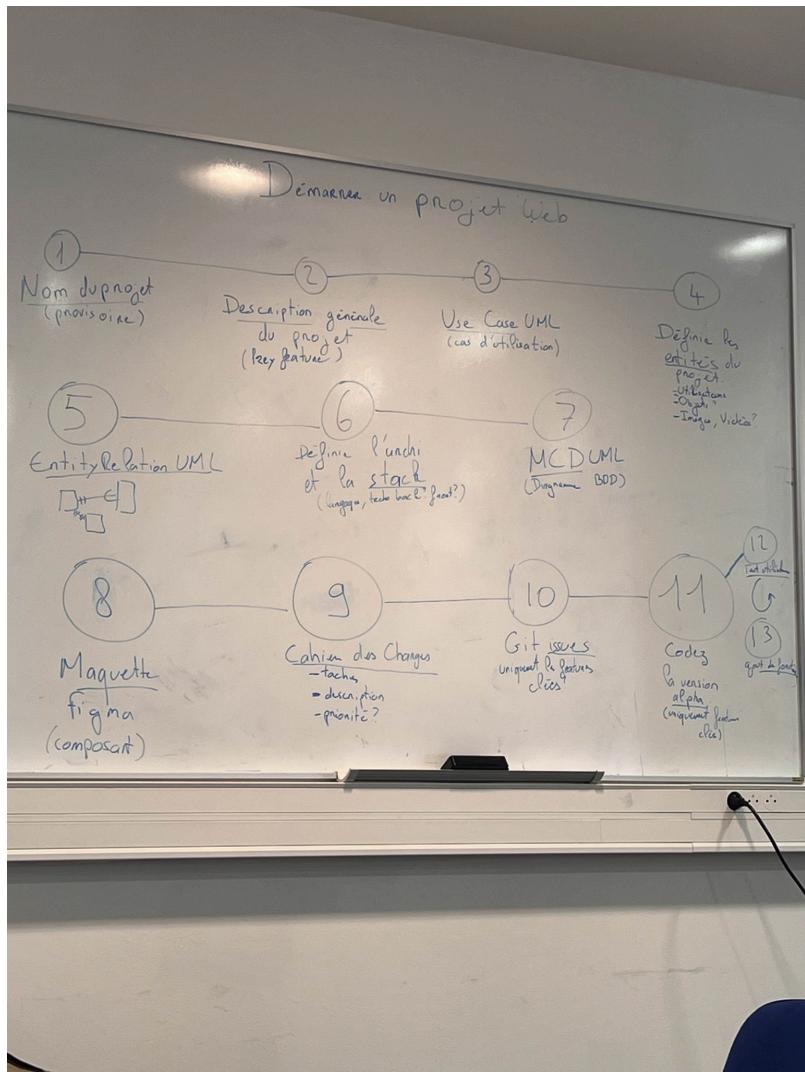


Schéma réalisé en cours de formation

DOSSIER PROFESSIONNEL (DP)

Activité-type 3

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

CP° 3 - Développer des composants métier

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

Afin de réaliser cette tâche j'ai dû comprendre les besoins métier et les fonctionnalités que le composant doit accomplir. Définir les cas d'utilisation et les règles métier et établir un cahier des charges fonctionnel qui décrit les spécifications du composant métier.

Déterminer les responsabilités du composant :

Ce que le composant fait exactement (calcul, transformation de données, validation, etc.).

Définir les interfaces du composant :

- Quels services ou méthodes ce composant exposera à d'autres parties de l'application.

Modéliser les objets métier :

- Créer des classes/objets qui représentent des entités métier (ex. :jeux, commande, utilisateur,catégories,etc.).

Créer un **diagramme MCD UML** et éventuellement un **diagramme de classe** à l'aide des diagrammes **d'Entités relations** et du **Cas d'utilisations** pour représenter le comportement du composant.

DOSSIER PROFESSIONNEL (DP)

2. Précisez les moyens utilisés :

Pour accomplir cette tâche, j'ai commencé par analyser les besoins métier et les fonctionnalités que le composant devait couvrir. J'ai établi un cahier des charges fonctionnel décrivant précisément les spécifications du composant. Ensuite, j'ai déterminé ses responsabilités (calculs, transformations de données, validations), et défini les interfaces, c'est-à-dire les services ou méthodes qu'il expose aux autres parties de l'application. J'ai également modélisé les objets métier sous forme de classes représentant des entités (jeux, utilisateurs, commandes, catégories). Pour finir, j'ai créé des diagrammes UML, notamment un MCD et un diagramme de classes, pour formaliser le comportement et les relations du composant.

3. Avec qui avez-vous travaillé ?

En Collaboration avec un apprenant de la formation

4. Contexte

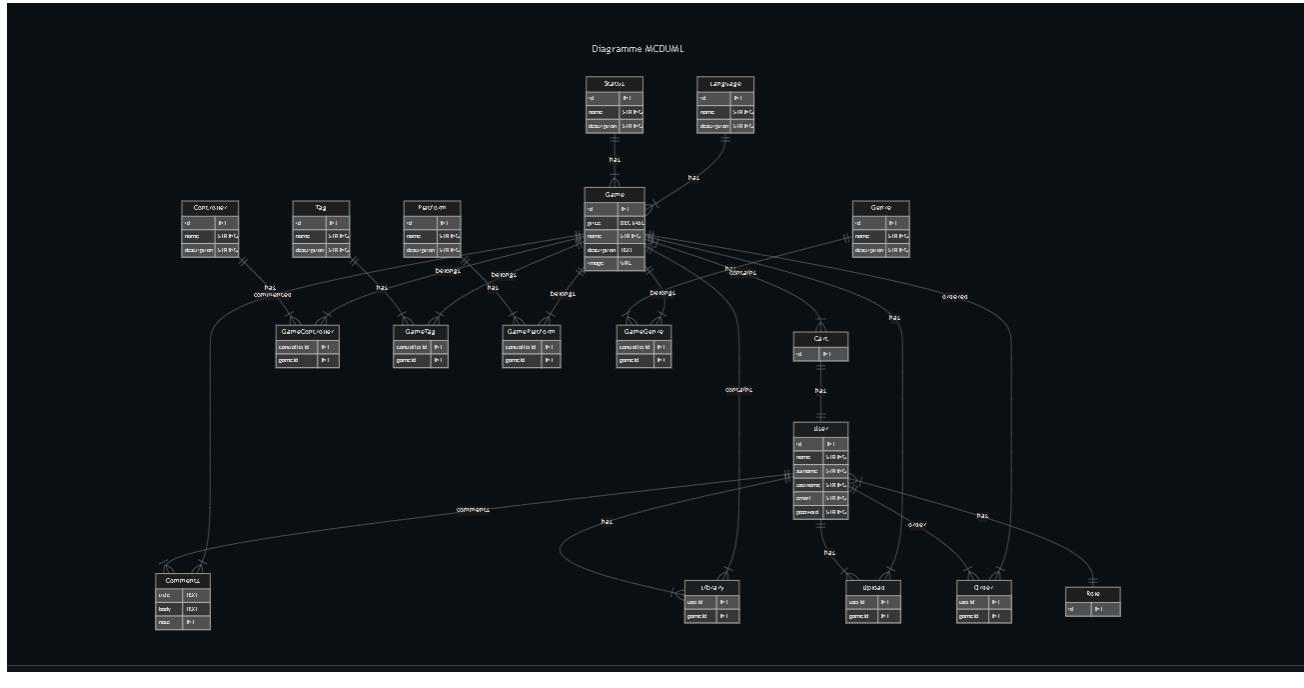
Nom de l'entreprise, organisme ou association ➔ *LaPlateforme.io*

Chantier, atelier, service ➔ *Site de Formation*

Période d'exercice ➔ Du : *04/03/2024* au : *01/10/2024*

DOSSIER PROFESSIONNEL (DP)

5. Informations complémentaires (*facultatif*)



DOSSIER PROFESSIONNEL (DP)

Activité-type 3

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

CP° 4 - *Construire une application organisée en couches*

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

Pour construire une application en couches, on commence par définir une architecture comprenant une couche présentation (UI), une couche métier (logique) et une couche d'accès aux données. Chaque couche a des responsabilités distinctes : l'UI gère l'interaction utilisateur, la couche métier applique les règles logiques, et la couche d'accès aux données interagit avec la base de données. Les couches communiquent via des interfaces bien définies. Ensuite, on effectue des tests pour valider chaque couche avant le déploiement. Cette structure garantit modularité, organisation claire et meilleure maintenance de l'application.

Cette organisation correspond au Modèle MVC (Modèle-Vue-Contrôleur).

2. Précisez les moyens utilisés :

Définir l'architecture : Organiser l'application en trois couches : présentation (interface), métier (logique), et données (base de données).

Concevoir l'interface utilisateur (Vue) : Créer l'interface qui affiche les données et permet à l'utilisateur d'interagir avec l'application.

Développer la logique métier (Contrôleur) : Écrire le code qui gère les règles et les processus de l'application.

Gérer l'accès aux données (Modèle) : Créer les fonctions qui communiquent avec la base de données pour récupérer et modifier les données.

Faire communiquer les couches : Assurer que chaque couche échange des informations correctement via des interfaces.

Tester : Valider que chaque partie fonctionne bien, seule et ensemble.

Déployer : Mettre l'application en ligne une fois testée.

DOSSIER PROFESSIONNEL (DP)

3. Avec qui avez-vous travaillé ?

en Collaboration avec le formateur ainsi que mes connaissances

4. Contexte

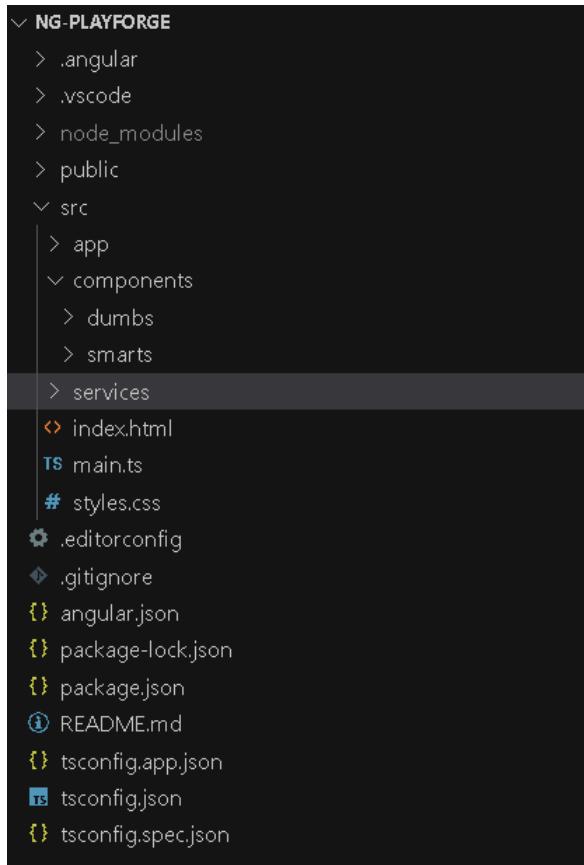
Nom de l'entreprise, organisme ou association ➔ *LaPlateforme.io*

Chantier, atelier, service ➔ *Site de Formation*

Période d'exercice ➔ Du : *04/03/2024* au : *01/10/2024*

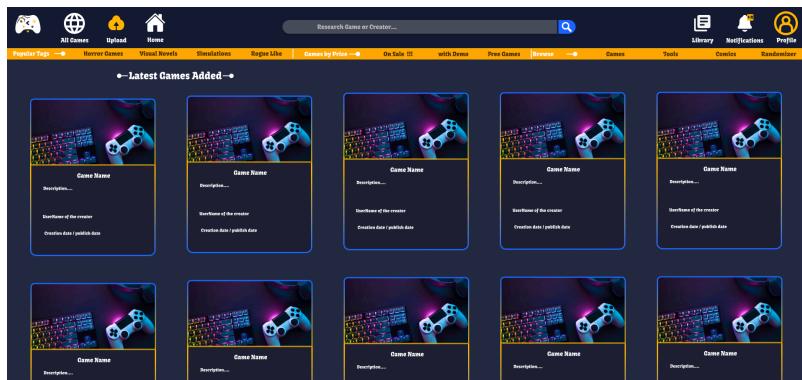
DOSSIER PROFESSIONNEL (DP)

5. Informations complémentaires (*facultatif*)



```
< NG-PLAYFORGE
> .angular
> .vscode
> node_modules
> public
< src
| > app
| < components
| | > dumbs
| | > smarts
| > services
< index.html
TS main.ts
# styles.css
.editorconfig
.ignored
.angular.json
package-lock.json
package.json
README.md
tsconfig.app.json
tsconfig.json
tsconfig.spec.json
```

**Application organisé en couche avec des composant
qui sont gérés par des services qui vont aller chercher
les informations du Backend ou de l'Upload**



Vue

DOSSIER PROFESSIONNEL (DP)

```
export const GameRoute = Router();
export const Game = sequelize.define("Game", {
    title: {...},
    price: {...},
    authorStudio: {...},
    madewith: {...},
    description: {...},
    createdAt: {...},
    updatedAt: {
        type: DataTypes.DATE,
        defaultValue: DataTypes.NOW,
        field: 'updatedAt',
        allowNull: true,
    },
    StatusId: {
        type: DataTypes.INTEGER,
        allowNull: true, // Autoriser NULL pour éviter l'erreur
        references: {
            model: 'Statuses',
            key: 'id',
        },
        onDelete: 'SET NULL',
        onUpdate: 'CASCADE',
    },
    LanguageId: {...}
});
```

Modèle

```
async getAllGames(): Promise<any> {
    try {
        const response = await fetch(this.apiUrlAllGames, {
            method: 'GET',
            headers: {
                'Content-Type': 'application/json',
            },
        });

        if (!response.ok) {
            throw new Error('Erreur lors de la récupération des jeux : ' + response.statusText);
        }

        const data = await response.json();
        return data;
    } catch (error) {
        console.log('erreur getAllGames()')
        console.error('Erreur:', error);
        throw error;
    }
}
```

Contrôleur

DOSSIER PROFESSIONNEL (DP)

Activité-type 3

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

CP° 5 - *Développer une application mobile*

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

Développer une application mobile avec Android Studio, on commence par analyser les besoins et définir les fonctionnalités. Ensuite, on configure le projet dans Android Studio et on conçoit l'interface utilisateur avec XML. La logique métier est implémentée en Java. On connecte l'application à une API pour l'accès aux données, puis effectue des tests sur différents appareils avec l'émulateur intégré. Enfin, l'application est optimisée et déployée sur le Google Play Store pour sa distribution.

2. Précisez les moyens utilisés :

1. **Android Studio** : IDE principal pour configurer le projet et gérer tout le cycle de développement.
2. **Kotlin ou Java** : Langages de programmation pour implémenter la logique métier et les fonctionnalités de l'application.
3. **XML** : Pour concevoir l'interface utilisateur (UI) en définissant les layouts des écrans.
4. **API RESTful** : Pour connecter l'application à des services web et gérer les données à distance.
5. **Emulateur Android** : Outil intégré à Android Studio pour tester l'application sur différents types d'appareils virtuels.
6. **Google Play Console** : Pour déployer et gérer la distribution de l'application sur le Google Play Store.

3. Avec qui avez-vous travaillé ?

DOSSIER PROFESSIONNEL (DP)

En collaboration avec un formateur

4. Contexte

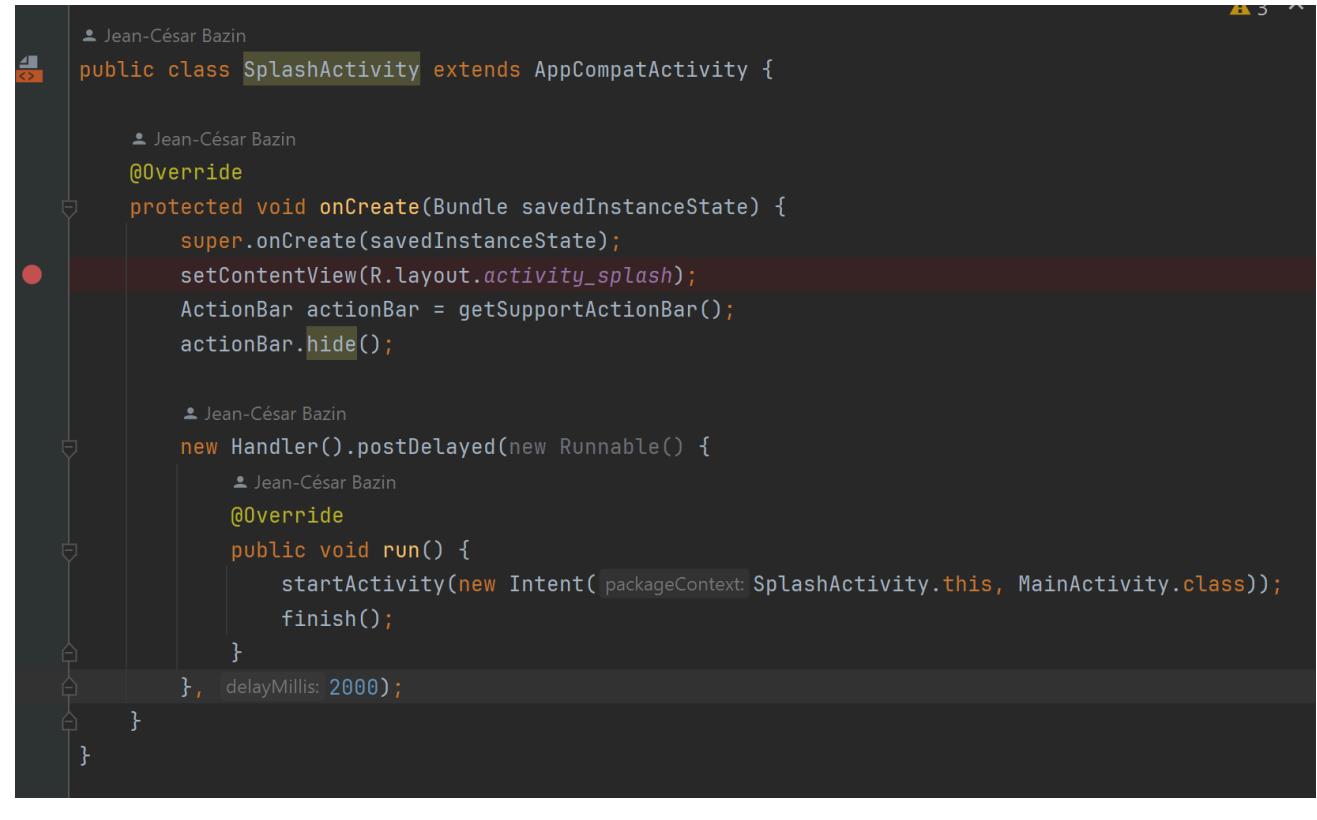
Nom de l'entreprise, organisme ou association ➔ *LaPlateforme.io*

Chantier, atelier, service ➔ *Site de Formation*

Période d'exercice ➔ Du : *04/03/2024* au : *01/10/2024*

5. Informations complémentaires (*facultatif*)

DOSSIER PROFESSIONNEL (DP)



The screenshot shows the code editor of an Android application. The current file is `SplashActivity.java`. The code implements a splash screen that displays for 2 seconds before transitioning to the main activity.

```
public class SplashActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash);
        ActionBar actionBar = getSupportActionBar();
        actionBar.hide();

        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                startActivity(new Intent(getApplicationContext(), MainActivity.class));
                finish();
            }
        }, 2000);
    }
}
```

DOSSIER PROFESSIONNEL (DP)

Activité-type 3

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

CP° 6 - *Préparer et exécuter les plans de tests d'une application*

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

J'ai utilisé des outils de test intégrés pour écrire et exécuter des scénarios de test. J'ai structuré les tests de manière à regrouper les cas similaires et à valider les résultats attendus. Pour isoler les composants ou services, j'ai utilisé des techniques de simulation et d'injection de dépendances. J'ai également employé des assertions pour comparer les résultats obtenus avec ceux attendus. Enfin, j'ai utilisé un système de lancement de tests pour exécuter et vérifier l'ensemble des tests, garantissant ainsi la qualité du code.

2. Précisez les moyens utilisés :

J'ai configuré Karma et Jasmine, qui sont déjà intégrés dans le framework. J'ai créé des fichiers de test `.spec.ts` pour chaque composant ou service. Ensuite, j'ai utilisé `describe()` pour regrouper mes tests et `it()` pour définir les cas de test spécifiques. J'ai employé `expect()` pour vérifier que les résultats correspondent à ce qui est attendu. Pour simuler les services ou dépendances, j'ai utilisé `TestBed` et des mocks. Enfin, j'ai exécuté les tests avec `ng test` pour valider le bon fonctionnement de mes composants et services.

3. Avec qui avez-vous travaillé ?

En collaboration avec des apprenants

DOSSIER PROFESSIONNEL (DP)

4. Contexte

Nom de l'entreprise, organisme ou association ➤ **LaPlateforme.io**

Chantier, atelier, service ➤ **Site de Formation**

Période d'exercice ➤ Du : **04/03/2024** au : **01/10/2024**

5. Informations complémentaires (facultatif)

```
src > services > 18 file.service.spec.ts > describe(FileService) callback
1 import { TestBed } from '@angular/core/testing';
2 import { HttpClientTestingModule, HttpTestingController } from '@angular/common/http/testing';
3 import { FileService } from './file-service.service';
4
5 describe('FileService', () => {
6   let service: FileService;
7   let httpMock: HttpTestingController;
8
9   beforeEach(() => {
10     TestBed.configureTestingModule({
11       imports: [HttpClientTestingModule],
12       providers: [FileService]
13     });
14     service = TestBed.inject(FileService);
15     httpMock = TestBed.inject(HttpTestingController);
16   });
17
18   afterEach(() => {
19     httpMock.verify(); // Vérifie qu'il n'y a pas de requêtes non exécutées
20   });
21
22   it('devrait uploader un fichier avec succès', async () => {
23     const mockFile = new File(['dummy content'], 'example.txt', { type: 'text/plain' });
24     const mockResponse = { fileUrl: 'http://localhost:9091/files/example.txt' };
25     const gameId = 123;
26
27     service.uploadFile(mockFile, gameId).then((response) => {
28       expect(response.fileUrl).toBe('http://localhost:9091/files/example.txt');
29     });
30
31     const req = httpMock.expectOne('http://localhost:9091/game/upload/file');
32     expect(req.request.method).toBe('POST');
33     req.flush(mockResponse); // Simule la réponse de l'API
34   });
35
```

DOSSIER PROFESSIONNEL (DP)

Activité-type 3

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

CP° 7 - *Préparer et exécuter le déploiement d'une application*

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

Pour préparer et exécuter le déploiement de mon projet Angular avec deux API (une pour l'upload et une pour le backend), j'ai d'abord optimisé et testé le code pour m'assurer du bon fonctionnement. Ensuite, j'ai créé le build Angular en production via la commande **ng build**. J'ai configuré les environnements de production pour l'API backend et l'API d'upload, en m'assurant que les routes et connexions étaient correctement configurées. Enfin, j'ai déployé les deux API sur un serveur, tout en reliant le front-end Angular à ces services pour une interaction fluide.

2. Précisez les moyens utilisés :

1. **Optimisation du code** : J'ai nettoyé et testé le projet Angular pour m'assurer qu'il fonctionne correctement.
2. **Build Angular** : J'ai utilisé la commande `ng build --prod` pour générer le projet optimisé en production.
3. **Configuration des environnements** : J'ai ajusté les fichiers `environment.prod.ts` pour les connexions à l'API backend et à l'API d'upload.
4. **Plesk** : J'ai utilisé Plesk pour déployer les API et le front-end Angular sur un serveur, en configurant les domaines, SSL et les connexions serveur (Node.js pour les API et le serveur web pour Angular).
5. **Gestion des connexions** : J'ai configuré correctement les routes dans Angular pour pointer vers les API backend et upload hébergées sur le serveur via Plesk.

DOSSIER PROFESSIONNEL (DP)

3. Avec qui avez-vous travaillé ?

J'ai travaillé avec l'aide de recherche Web et Plesk, un hébergeur fournit par l'organisme de formation

4. Contexte

Nom de l'entreprise, organisme ou association ➔ **LaPlateforme.io**

Chantier, atelier, service ➔ **Site de Formation**

Période d'exercice ➔ Du : **04/03/2024** au : **01/10/2024**

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] **Gomez James**,
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 **Sollies-Toucas** le **25/09/2024**

pour faire valoir ce que de droit.

Signature :



DOSSIER PROFESSIONNEL (DP)

Documents illustrant la pratique professionnelle

(*facultatif*)

Intitulé

Cliquez ici pour taper du texte.

DOSSIER PROFESSIONNEL (DP)

ANNEXES

(Si le RC le prévoit)