



MINISTÈRE CHARGÉ
DE L'EMPLOI

DOSSIER PROFESSIONNEL (DP)

- Nom de naissance* ▶ Guevaer
Nom d'usage ▶ Guevaer
Prénom ▶ Arnaud
Adresse ▶

Titre professionnel visé

Développeur web et web mobile

MODALITÉ D'ACCÈS :

- 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.



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

DOSSIER PROFESSIONNEL^(DP)



DOSSIER PROFESSIONNEL^(DP)

Sommaire

Exemples de pratique professionnelle

Développer la partie front-end d'une application web ou web mobile sécurisée

- Maquetter des interfaces utilisateur web ou web mobile p.5
- Réaliser des interfaces utilisateur statiques web ou web mobile p.8
- Développer la partie dynamique des interfaces utilisateur web ou web mobile p.11

Développer la partie back-end d'une application web ou web mobile sécurisée

- Mettre en place une base de données relationnelle p.14
- Développer des composants d'accès aux données SQL et NoSQL p.17
- Développer des composants métier côté serveur p.20

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

p.23

Déclaration sur l'honneur

p.24

Documents illustrant la pratique professionnelle (*facultatif*)

p.25

Annexes (*Si le RC le prévoit*)

p.26

DOSSIER PROFESSIONNEL ^(DP)

EXEMPLES DE PRATIQUE

PROFESSIONNELLE

DOSSIER PROFESSIONNEL (DP)

Développer la partie front-end d'une application web ou web mobile sécurisée

Activité-type 1

Exemple n°1 - Maquetter des interfaces utilisateur web ou web mobile

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

Pendant la formation, j'ai commencé à coder un projet personnel que j'ai amélioré au fur et à mesure de mon apprentissage. Il s'agit d'un jeu incrémental sur l'univers de Dragon Ball que j'ai appelé **Z Warriors Clicker**, dont nous parlerons tout au long de ce dossier.

Le but du jeu est d'accumuler de la puissance par le biais d'un bouton qui incrémente un compteur et de vaincre des ennemis en réduisant leurs points de vie à zéro. Des techniques sont disponibles avec comme coût la puissance afin d'en gagner de plus en plus ou infliger des dégâts à l'ennemi.

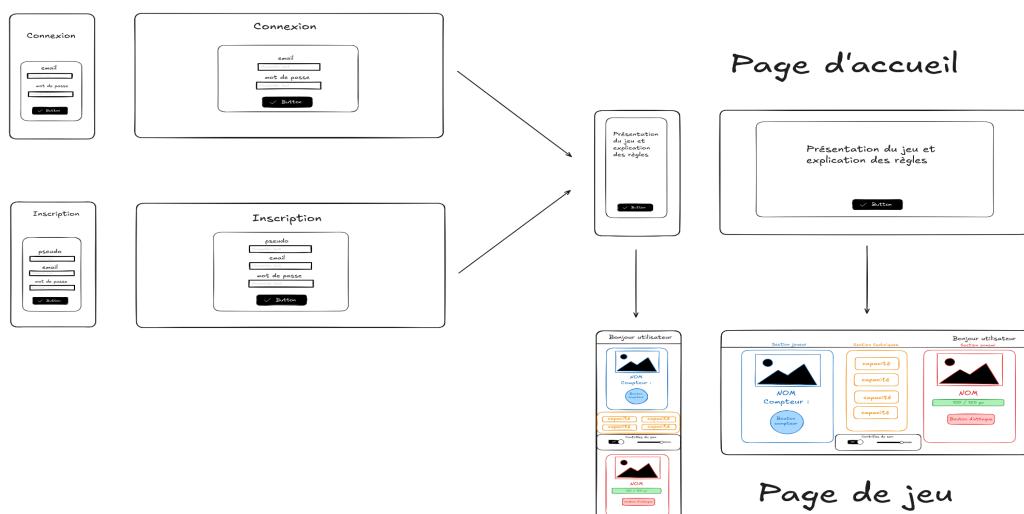
Voici la structure de l'application :

- Une page de connexion / inscription
- Une fois connecté, l'utilisateur arrive sur la page d'accueil, qui explique les règles du jeu, avec un bouton pour commencer.
- La page du jeu, composée d'une barre avec le nom de l'utilisateur en haut, une partie joueur avec son personnage, une partie avec les techniques du personnage, une partie ennemi et une barre pour contrôler le volume du jeu en bas.

2. Précisez les moyens utilisés :

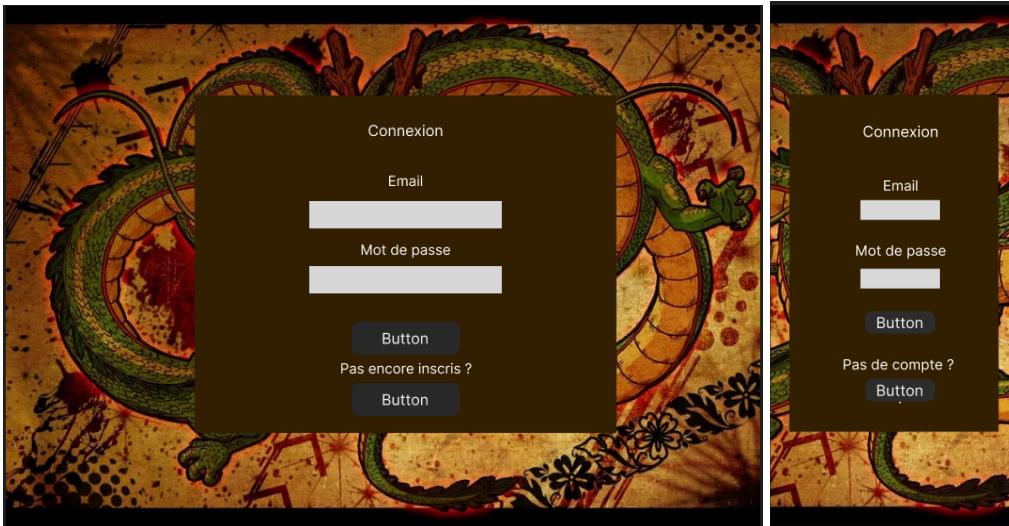
J'ai commencé par réaliser un wireframe sur l'outil en ligne **Excalidraw**.

Pages d'inscription / connexion

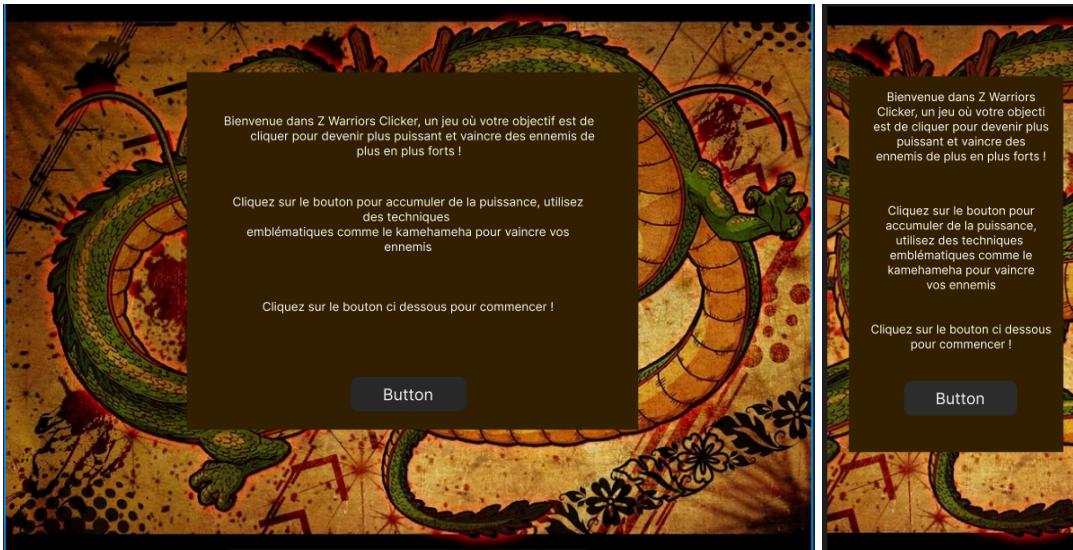


DOSSIER PROFESSIONNEL (DP)

J'ai ensuite réalisé un prototype du projet en utilisant l'outil en ligne **Figma** pour les versions desktop et mobile .

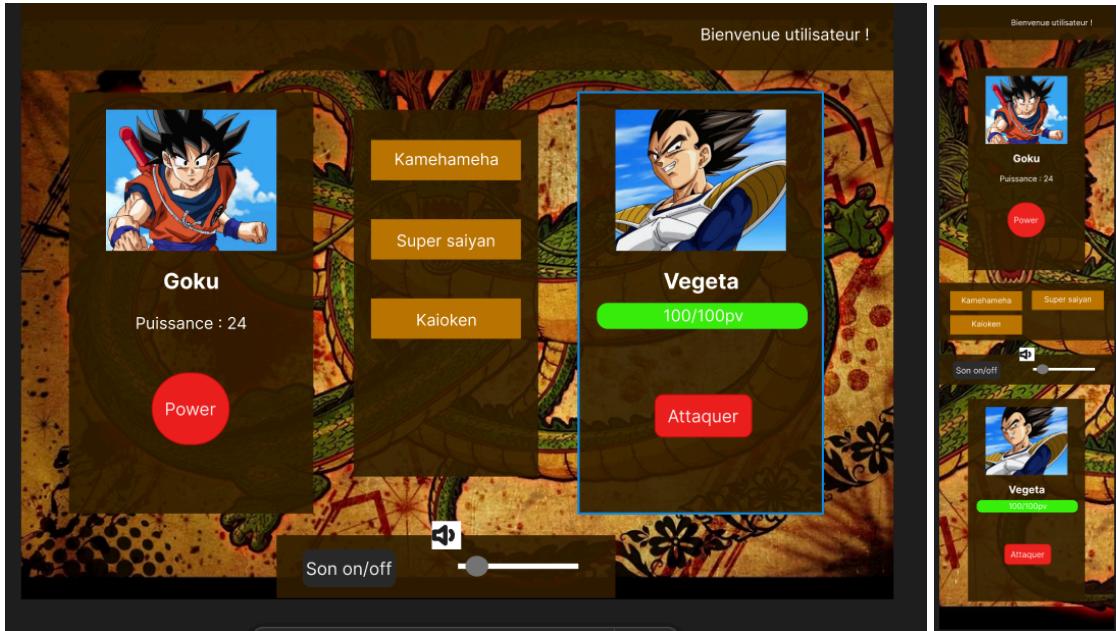


Page de connexion



Page d'accueil

DOSSIER PROFESSIONNEL (DP)



Page de jeu

3. Avec qui avez-vous travaillé ?

J'ai travaillé seul sur ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ Wild Code School

Chantier, atelier, service ▶ Projet personnel réalisé en cours de formation

Période d'exercice ▶ Du 23/09/2024 au 21/02/2025

5. Informations complémentaires (facultatif)

Le Dépôt Github du projet est accessible avec ce lien : <https://github.com/Arnaud6216/Z-Warriors-Clicker>

DOSSIER PROFESSIONNEL (DP)

Activité-type 1 Développer la partie front-end d'une application web ou web mobile sécurisée

Exemple n°2 - Réaliser des interfaces utilisateur statiques web ou web mobile

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

Le wireframe et le prototype étant réalisés et validés, j'ai commencé à coder les différentes pages du projet.

J'ai choisi de débuter par la version desktop, étant plus à l'aise sur ordinateur. Pour la gestion de l'authentification, j'ai créé deux composants distincts : **Login** pour la connexion et **Register** pour l'inscription.

Le composant **Login** contient deux champs de saisie (email et mot de passe) ainsi qu'un bouton de validation permettant à l'utilisateur de se connecter. De son côté, le composant **Register** reprend cette structure avec un champ supplémentaire pour le pseudo.

Chaque page intègre également un bouton permettant de basculer entre la connexion et l'inscription si l'utilisateur possède déjà un compte

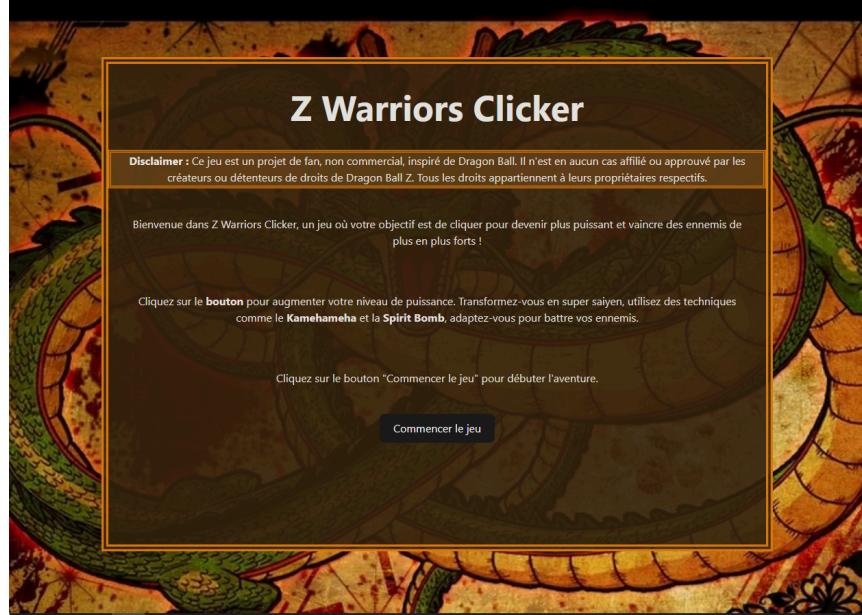
J'ai ensuite créé un nouveau composant pour la page d'accueil, composée de plusieurs paragraphes expliquant les règles du jeu, et d'un bouton pour naviguer vers la page du jeu.

Une fois la structure en place, il ne me restait plus qu'à styliser le tout grâce au CSS. Afin d'améliorer la lisibilité et la maintenabilité du code, j'ai créé une feuille de style CSS dédiée à chaque composant.



Pages de connexion et d'inscription

DOSSIER PROFESSIONNEL (DP)



Page d'accueil

Pour le responsive, j'ai utilisé différents breakpoints afin que l'application s'adapte pour tout type de téléphone et tablettes, en mode portrait et paysage.

2. Précisez les moyens utilisés :

Pour mener à bien ce projet, j'ai utilisé les outils et technologies suivants :

- **Langages et frameworks** : React.js pour la création des composants et la gestion de l'interface utilisateur, avec du CSS pour la stylisation.
- **Environnement de développement** : Visual Studio Code pour écrire et organiser le code.
- **Gestion de version** : Git et GitHub pour suivre l'évolution du projet et conserver un historique des modifications.

DOSSIER PROFESSIONNEL^(DP)

3. Avec qui avez-vous travaillé ?

J'ai travaillé seul sur ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ Wild Code School

Chantier, atelier, service ▶ Projet personnel réalisé en cours de formation

Période d'exercice ▶ Du 23/09/2024 au 21/02/2025

5. Informations complémentaires (*facultatif*)

Le Dépôt Github du projet est accessible avec ce lien : <https://github.com/Arnaud6216/Z-Warriors-Clicker>

DOSSIER PROFESSIONNEL (DP)

Activité-type 1

Développer la partie front-end d'une application web ou web mobile sécurisée

Exemple n°3 -

Développer la partie dynamique des interfaces utilisateur web ou web mobile

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



Page de jeu

Il ne me restait qu'une page à créer : celle du jeu. J'ai structuré son affichage avec le composant principal **Gameboard**, divisé en cinq sous-composants pour plus de clarté :

- **Navbar** : Affiche le pseudo et la progression du joueur.
- **PlayerCard** : Montre les infos du joueur (image, nom, puissance) et un bouton pour accumuler de la puissance.
- **Tech** : Liste les techniques disponibles.
- **AudioController** : Gère l'ambiance sonore (lecture, volume musique/effets).
- **EnnemyCard** : Affiche l'ennemi (image, nom, barre de vie) avec deux boutons pour l'attaquer.

Un **Context** centralise les données essentielles pour l'application.

Nous verrons ici le composant **EnnemyCard**

DOSSIER PROFESSIONNEL (DP)

EnnemyCard : La partie ennemi affiche l'image de celui-ci, sa barre de vie avec un nombre de points de vie, et deux boutons d'attaques : légère (lightAttack) et lourde (strongAttack).

L'attaque légère (lightAttack) : peut être enchaînée sans restriction et inflige des dégâts minime initialisés à 1 . Un effet sonore est joué à chaque clic. La fonction vérifie si les points de vie de l'ennemi ne descendent pas en dessous de 0. Si c'est le cas, la fonction **ennemyDefeated** s'exécute et passe à l'ennemi suivant (nous reviendrons plus tard sur cette fonction).

```
const handleClickLightAttack = () => {
  soundEffectList[0].play(effectVolume);
  if (ennemyLife > lightAttack) {
    setEnnemyLife(Math.max(ennemyLife - lightAttack, 0));
    // Verify if the ennemy's life do not go below 0
  } else {
    ennemyDefeated();
  }
};
```

L'attaque lourde (strongAttack) : dispose d'un temps de recharge mais inflige des dégâts plus élevés initialisés à 5 .

Affichage d'une barre de progression :

Un setInterval exécute une mise à jour toutes les 100 ms, augmentant la progression de 100 / 30 pour simuler un temps de recharge de 3 secondes. Lorsque la barre atteint 100%, l'intervalle est arrêté avec clearInterval(interval).

Réactivation du bouton : Après 3,4 secondes, un setTimeout réactive le bouton et réinitialise la barre de progression.

```
const handleClickStrongAttack = () => {
  if (isButtonDisabled) return;
  setIsButtonDisabled(true);
  setBarProgress(0);

  // disable the button for 3 seconds and display a progress bar
  let currentProgress = 0;
  const interval = setInterval(() => {
    currentProgress += 100 / 30;
    setBarProgress(currentProgress);
    if (currentProgress >= 100) {
      clearInterval(interval);
    }
  }, 100);

  setTimeout(() => {
    setIsButtonDisabled(false);
    setBarProgress(0);
  }, 3400);

  soundEffectList[1].play(effectVolume);
  if (ennemyLife > strongAttack) {
    setEnnemyLife(Math.max(ennemyLife - strongAttack, 0));
  } else {
    ennemyDefeated();
  }
};
```

DOSSIER PROFESSIONNEL (DP)

2. Précisez les moyens utilisés :

j'ai utilisé plusieurs technologies et outils afin d'assurer une structure propre, efficace et maintenable :

- React avec TypeScript : Pour construire les différents composants de l'interface utilisateur avec une meilleure gestion des types et une meilleure maintenabilité.
- Hooks React (useState, useEffect, useContext) :
 - **useState** pour gérer l'état local des composants (ex : compteur de puissance, styles des boutons).
 - **useEffect** pour exécuter des actions après le rendu des composants, notamment pour l'incrémentation automatique de la puissance avec setInterval.
 - **useContext** pour centraliser et partager certaines données importantes (comme les compteurs) entre plusieurs composants.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seul sur ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ Wild Code School

Chantier, atelier, service ▶ Projet personnel réalisé en cours de formation

Période d'exercice ▶ Du 23/09/2024 au 21/02/2025

5. Informations complémentaires (facultatif)

Le Dépôt Github du projet est accessible avec ce lien : <https://github.com/Arnaud6216/Z-Warriors-Clicker>

DOSSIER PROFESSIONNEL (DP)

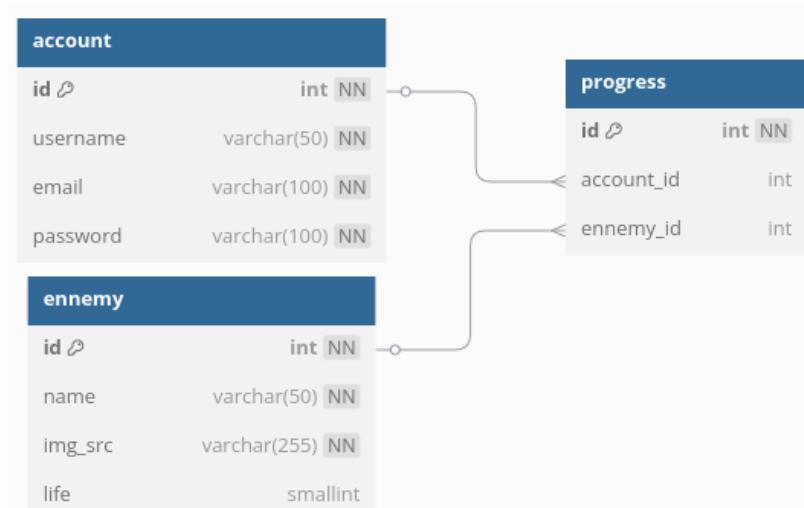
Activité-type 2

Développer la partie back-end d'une application web ou web mobile sécurisée

Exemple n°1 - Mettre en place une base de données relationnelle

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

La base de données de **Z Warrior Clicker** est conçue pour gérer les comptes des joueurs, les ennemis rencontrés et la progression de chaque utilisateur. J'ai commencé par établir le Modèle Conceptuel de données (MCD).



La table **account** stocke les informations essentielles des joueurs, notamment un identifiant unique (**id**), un nom d'utilisateur (**username**), une adresse e-mail (**email**) et un mot de passe haché (**hashed_password**).

La table **ennemy** contient les ennemis que les joueurs affronteront, chacun ayant un identifiant unique (**id**), un nom (**name**), une image associée (**img_src**) et un nombre de points de vie (**life**).

La relation entre ces deux tables est many to many et peut se traduire comme ceci : Un compte peut affronter plusieurs ennemis et un ennemi peut être affronté par plusieurs comptes.

Une nouvelle table **progress** est donc née de cette relation afin de pouvoir sauvegarder la progression du joueur.

Celle-ci associe chaque compte à un ennemi spécifique via les clés étrangères **account_id** et **ennemy_id**, avec des règles de suppression en cascade pour garantir l'intégrité des données en cas de suppression d'un compte ou d'un ennemi. Un *trigger* (`add_default_progress`) est mis en place pour initialiser automatiquement la progression des nouveaux joueurs en leur assignant leur premier ennemi dès la création de leur compte.

DOSSIER PROFESSIONNEL (DP)

J'ai créé un fichier schéma.sql pour la base de données, que j'ai donc complété selon mon MCD vu précédemment.

```
CREATE TABLE account (
    id INT UNSIGNED PRIMARY KEY AUTO_INCREMENT NOT NULL,
    username VARCHAR(50) NOT NULL UNIQUE,
    email VARCHAR(100) NOT NULL UNIQUE,
    hashed_password VARCHAR(100) NOT NULL
);

CREATE TABLE ennemy (
    id INT UNSIGNED PRIMARY KEY AUTO_INCREMENT NOT NULL,
    name VARCHAR(50) NOT NULL UNIQUE,
    img_src VARCHAR(255) NOT NULL UNIQUE,
    life SMALLINT UNSIGNED NOT NULL
);

CREATE TABLE progress (
    id INT UNSIGNED PRIMARY KEY AUTO_INCREMENT NOT NULL,
    account_id INT UNSIGNED NOT NULL,
    ennemy_id INT UNSIGNED NOT NULL,
    FOREIGN KEY (account_id) REFERENCES account(id) ON DELETE CASCADE,
    FOREIGN KEY (ennemy_id) REFERENCES ennemy(id) ON DELETE CASCADE
);
```

Création des différentes tables.

```
INSERT INTO ennemy (name, img_src, life)
VALUES
    ("Nappa", "src/assets/nappa.webp", 50),
    ("Vegeta", "src/assets/vegeta.webp", 500),
    ("Guldo", "src/assets/guldo.webp", 150),
    ("Burter", "src/assets/burter.webp", 160),
    ("Jeice", "src/assets/jeice.webp", 170),
    ("Recome", "src/assets/recome.webp", 180),
    ("Ginyu", "src/assets/ginyu.webp", 200),
    ("Freezer", "src/assets/freezer.webp", 1000),
    ("C 17", "src/assets/c17.webp", 500),
    ("C 18", "src/assets/c18.webp", 600),
    ("Cell", "src/assets/cell.webp", 3000),
    ("Buu", "src/assets/buu.webp", 5000);
```

```
CREATE TRIGGER add_default_progress
AFTER INSERT ON account
FOR EACH ROW
BEGIN
    INSERT INTO progress (account_id, ennemy_id)
    VALUES (NEW.id, 1);
END;
```

Insertion des ennemis et création du trigger afin que chaque nouveau joueur ait une progression de base en ajoutant un ennemy_id de 1 à chaque nouveau compte créé.

DOSSIER PROFESSIONNEL (DP)

2. Précisez les moyens utilisés :

J'ai utilisé le site **DB Diagram**(dbdiagram.io) pour modéliser ma base de données et définir ses relations. Ensuite, j'ai implémenté cette structure en utilisant le langage **SQL** avec le Système de Gestion de Base de Données Relationnelle (SGBDR) **MySQL**, que j'ai installé et configuré dans le cadre de mes cours à la Wild Code School.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seul sur ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association - **Wild Code School**

Chantier, atelier, service - Projet personnel réalisé en cours de formation

Période d'exercice - Du 23/09/2024 au 21/02/2025

5. Informations complémentaires (*facultatif*)

Le Dépôt Github du projet est accessible avec ce lien : <https://github.com/Arnaud6216/Z-Warriors-Clicker>

DOSSIER PROFESSIONNEL (DP)

Activité-type 2

Développer la partie back-end d'une application web ou web mobile sécurisée

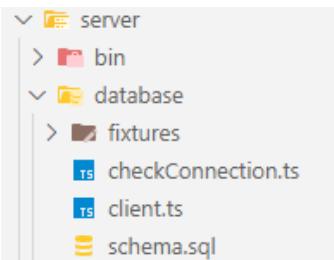
Exemple n°2 -

Développer des composants d'accès aux données SQL et NoSQL

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

J'ai utilisé l'architecture **MVC (Modèle - Vue - Contrôleur)** afin de structurer l'accès aux données de manière claire et organisée. Cette approche permet de séparer la logique métier, la gestion des données et l'interface utilisateur, facilitant ainsi la maintenance et l'évolution de l'application.

Nous allons voir la partie Modèle plus en détail.



Un module d'accès aux données **client.ts** assure une interaction efficace entre l'application web et la base de données.

```
// Get variables from .env file for database connection
const { DB_HOST, DB_PORT, DB_USER, DB_PASSWORD, DB_NAME } = process.env;

// Create a connection pool to the database
import mysql from "mysql2/promise";

const client = mysql.createPool({
  host: DB_HOST,
  port: Number.parseInt(DB_PORT as string),
  user: DB_USER,
  password: DB_PASSWORD,
  database: DB_NAME,
});

// Ready to export
export default client;

// Types export
import type { Pool, ResultSetHeader, RowDataPacket } from "mysql2/promise";

type DatabaseClient = Pool;
type Result = ResultSetHeader;
type Rows = RowDataPacket[];

export type { DatabaseClient, Result, Rows };
```

DOSSIER PROFESSIONNEL (DP)

Pour sécuriser les informations sensibles, le module **dotenv** stocke les identifiants de connexion dans un fichier .env, évitant ainsi leur exposition dans le code.

Un **pool de connexions** est géré avec mysql2/promise pour optimiser les performances en réutilisant les connexions existantes plutôt que d'en ouvrir de nouvelles à chaque requête. Le client est exporté pour être utilisé dans l'application.

Les **types** TypeScript garantissent la fiabilité des requêtes SQL :

- **DatabaseClient** : Gère l'accès à la base via le pool de connexions.
- **Result** : Contient les infos des requêtes INSERT, UPDATE, DELETE (ex. lignes affectées).
- **Rows** : Représente les résultats des SELECT sous forme de tableau.

Le fichier **checkConnexion.ts** sert à vérifier que la connexion à la base de données s'effectue correctement.

```
import client from "./client";

// Try to get a connection to the database
client
  .getConnection()
  .then((connection) => {
    console.info(`Using database ${process.env.DB_NAME}`);

    connection.release();
  })
  .catch((error: Error) => {
    console.warn(
      "Warning:",
      "Failed to establish a database connection.",
      "Please check your database credentials in the .env file if you need a database access.",
    );
    console.warn(error.message);
  });
}
```

La fonction **getConnection()** du pool de connexion teste la connexion avec la base de données.

Si tout se passe bien, un message est renvoyé utilisant une variable d'environnement avec le nom de la base de données.

Sinon, un message d'erreur est renvoyé dans la console.

DOSSIER PROFESSIONNEL (DP)

2. Précisez les moyens utilisés :

J'ai utilisé la bibliothèque mysql2/promise pour créer le pool de connexion.

Le module dotenv pour sécuriser les données sensibles dans des variables d'environnements.

Utilisation de Typescript pour la fiabilité du code.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seul sur ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association - Wild Code School

Chantier, atelier, service - Projet personnel réalisé en cours de formation

Période d'exercice - Du 23/09/2024 au 21/02/2025

5. Informations complémentaires (facultatif)

Le Dépôt Github du projet est accessible avec ce lien : <https://github.com/Arnaud6216/Z-Warriors-Clicker>

DOSSIER PROFESSIONNEL (DP)

Activité-type 2

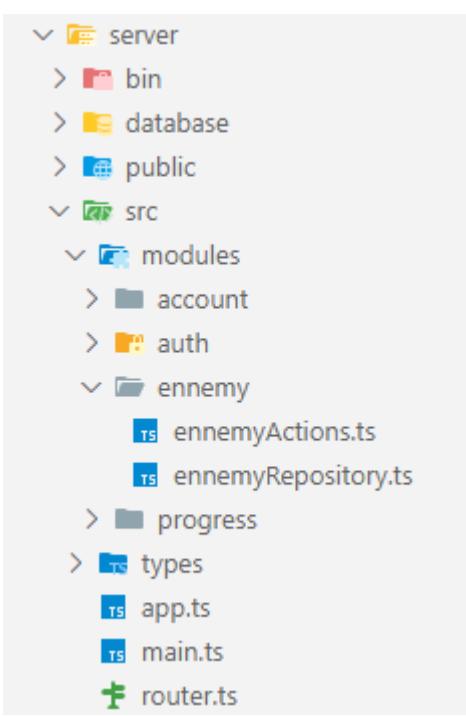
Développer la partie back-end d'une application web ou web mobile sécurisée

Exemple n°3 -

Développer des composants métier côté serveur

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

Après avoir la couche Modèle, nous allons voir maintenant la partie Contrôleur avec comme exemple la table ennemy.



```
import type { RequestHandler } from "express";
import ennemyRepository from "./ennemyRepository";

const browse: RequestHandler = async (req, res, next) => {
  try {
    const ennemy = await ennemyRepository.readAll();

    res.json(ennemy);
  } catch (err) {
    next(err);
  }
};

const read: RequestHandler = async (req, res, next) => {
  try {
    const ennemyId = Number(req.params.id);
    const ennemy = await ennemyRepository.read(ennemyId);

    if (ennemy == null) {
      res.sendStatus(404);
    } else {
      res.json(ennemy);
    }
  } catch (err) {
    next(err);
  }
};

export default { browse, read };
```

Le module **ennemyActions.ts** définit deux contrôleurs Express pour interagir avec la base de données :

- **browse** : Récupère tous les ennemis via ennemyRepository.readAll() et renvoie les données en JSON. En cas d'erreur, next(err) assure une gestion propre.
- **read** : Récupère un ennemi via son ID (Number(req.params.id)) en appelant ennemyRepository.read(ennemyId). Si aucun ennemi n'est trouvé, une réponse 404 est envoyée ; sinon, les données sont renvoyées en JSON. Les erreurs sont gérées via try/catch et next(err).

DOSSIER PROFESSIONNEL (DP)

```
import databaseClient from "../../database/client";
import type { Ennemy } from "../../types/express/index";
import type { Rows } from "../../database/client";

class EnnemyRepository {
    async read(id: number) {
        const [rows] = await databaseClient.query<Rows>(
            "select * from ennemy where id = ?",
            [id],
        );

        return rows[0] as Ennemy;
    }

    async readAll() {
        const [rows] = await databaseClient.query<Rows>("select * from ennemy");
        return rows as Ennemy[];
    }
}

export default new EnnemyRepository();
```

Le module **ennemyRepository.ts** définit la couche Modèle chargée de l'accès aux données des ennemis. Il repose sur la classe **EnnemyRepository**, qui exécute des requêtes SQL via `databaseClient`.

- **read(id: number)** : Récupère un ennemi par son ID via `SELECT * FROM ennemy WHERE id = ?`, sécurisé contre les injections SQL. Retourne le premier résultat (`rows[0]`) typé `Ennemy`.
- **readAll()** : Récupère tous les ennemis avec `SELECT * FROM ennemy` et les retourne sous forme de tableau typé `Ennemy[]`, assurant une stricte gestion des types en TypeScript.

Avec ces deux méthodes, la couche Modèle permet de manipuler les données des ennemis de manière sécurisée et structurée. Il ne reste plus qu'à définir les routes dans la couche Contrôleur (Controller), qui exposeront ces fonctionnalités à l'extérieur, pour que les clients puissent interagir avec ces données.

DOSSIER PROFESSIONNEL (DP)

Le module **router.ts** définit les **routes de l'API** de l'application en utilisant **Express**. Ces routes sont responsables de gérer les différentes requêtes envoyées par le client et de renvoyer les réponses appropriées. Il existe plusieurs groupes de routes pour interagir avec différentes parties de l'application : comptes utilisateurs, ennemis, progrès et authentification.

Pour les **ennemis**, deux routes sont définies :

1. **GET /api/ennemy** : Cette route permet de récupérer la liste complète des ennemis. Elle appelle la méthode `browse` du module `ennemyActions`, qui va récupérer toutes les données des ennemis et les renvoyer en réponse sous forme de JSON.
2. **GET /api/ennemy/:id** : Cette route permet de récupérer un ennemi spécifique en utilisant son identifiant, qui est passé en paramètre dans l'URL (`/api/ennemy/{id}`). La méthode `read` de `ennemyActions` est appelée pour obtenir les données de l'ennemi correspondant à cet identifiant et les renvoyer sous forme de JSON.

2. Précisez les moyens utilisés :

Utilisation de **TypeScript**, **Node.js** avec **express**, **MySQL** avec la bibliothèque **mysql2/promise**, et la librairie **dotenv**.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seul sur ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association - Wild Code School

Chantier, atelier, service - Projet personnel réalisé en cours de formation

Période d'exercice - Du 23/09/2024 au 21/02/2025

5. Informations complémentaires (facultatif)

Le Dépôt Github du projet est accessible avec ce lien : <https://github.com/Arnaud6216/Z-Warriors-Clicker>

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.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.

DOSSIER PROFESSIONNEL (DP)

Déclaration sur l'honneur

Je soussigné Arnaud GUEVAER ,

déclare sur l'honneur que les renseignements fournis dans ce dossier sont exacts et que je suis l'auteur des réalisations jointes.

Fait à LA COUTURE

le 23/02/2025

pour faire valoir ce que de droit.

Signature :



DOSSIER PROFESSIONNEL (DP)

Documents illustrant la pratique professionnelle

(facultatif)

Intitulé

Cliquez ici pour taper du texte.

ANNEXES

(Si le RC le prévoit)