Compte Rendu des Travaux Pratiques

Développement Web Avancé - Année 2025-2026

Présenté par :
El Warraqi Imane
elwarraqiimane@gmail.com

Sous la supervision de :
Pr. Amal Ordu

Rendu le 11 novembre 2025

Résumé

Ce document présente le compte rendu détaillé des trois travaux pratiques réalisés dans le cadre du module de Développement Web Avancé. Les TP sont présentés dans l'ordre suivant : Jeu Pokémon avec l'API PokéAPI, application Express.js avec authentification, et Book Reading Tracker en TypeScript. Chaque projet démontre l'acquisition de compétences spécifiques en développement backend, frontend, intégration d'API et gestion de bases de données.

Table des matières

1 Introduction

1.1 Contexte Pédagogique

Ce compte rendu présente trois travaux pratiques réalisés dans le cadre du module de Développement Web Avancé. L'objectif était de maîtriser les technologies modernes du développement web à travers des projets concrets et progressifs.

1.2 Ordre de Présentation

1. **TP1 :** Jeu Pokémon avec l'API PokéAPI
2. **TP2 :** Application Express.js avec authentification et gestion de livres
3. **TP3 :** Book Reading Tracker en TypeScript

1.3 Objectifs Généraux

— Développer des applications web complètes avec différentes stack techniques
— Maîtriser l'intégration d'APIs externes
— Implémenter des systèmes d'authentification sécurisés
— Utiliser des bases de données NoSQL avec MongoDB
— Explorer les avantages de TypeScript pour le développement full-stack

2 TP1 : Jeu Pokémon avec l'API PokéAPI

2.1 Objectifs Spécifiques

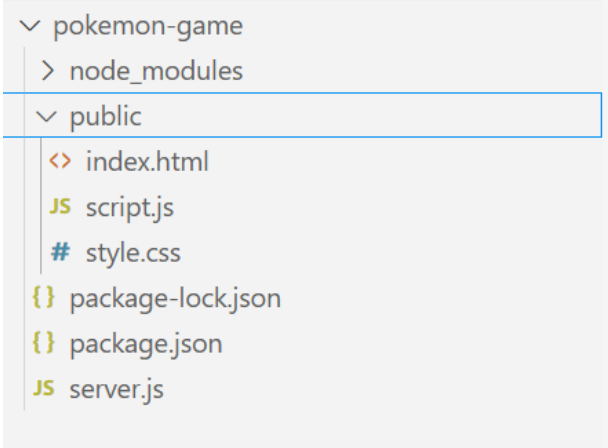
— Interagir avec une API REST externe (PokéAPI)
— Créer un jeu de combat Pokémon tour par tour
— Implémenter un système de combat avec gestion des PP (Power Points)
— Prendre en compte la précision des attaques
— Développer une interface utilisateur dynamique et interactive

2.2 Architecture Technique

2.3 Structure du Projet

Composant	Technologie
Backend	Express.js, Node.js
API Externe	PokéAPI (`https://pokeapi.co`)
Frontend	JavaScript vanilla, HTML5, CSS3
Communication	Fetch API, JSON
Serveur	Express.js avec routes API
Stockage	Variables en mémoire (session)

TABLE 1 – Stack technique du jeu Pokémon



2.4 Fonctionnalités Implémentées

2.4.1 Intégration de l'API PokéAPI

```
app.get('/api/pokemon/:name', async (req, res) => {
  try {
    const response = await axios.get(
      `https://pokeapi.co/api/v2/pokemon/${req.params.name}`
    );
    const pokemon = response.data;

    // Extraction des 5 premi res attaques
    const moves = [];
    for (let i = 0; i < Math.min(5, pokemon.moves.length); i++) {
      const moveResponse = await axios.get(pokemon.moves[i].move.url);
      const moveData = moveResponse.data;

      moves.push({
        name: moveData.name,
        power: moveData.power || 0,
        accuracy: moveData.accuracy || 100,
        pp: moveData.pp || 20,
        type: moveData.type.name
      });
    }

    res.json({
      name: pokemon.name,
      sprite: pokemon.sprites.front_default,
```

```
      hp: 300, // PV fixes pour le jeu
      moves: moves
    });
  } catch (error) {
    res.status(404).json({ error: 'Pok mon non trouv ' });
  }
});
```

Listing 1 – Récupération des données Pokémon

2.4.2 Système de Combat Tour par Tour

```
class PokemonGame {
  constructor() {
    this.playerPokemon = null;
    this.enemyPokemon = null;
    this.gameState = 'selection';
  }

  async playerAttack(moveIndex) {
    const playerMove = this.playerPokemon.moves[moveIndex];

    // V rification des PP
    if (playerMove.currentPp <= 0) {
      this.addToLog(`${this.playerPokemon.name} n'a plus de PP!`);
      return;
    }

    playerMove.currentPp--;

    // V rification de la pr cision
    const hitChance = Math.random() * 100;
    if (hitChance > playerMove.accuracy) {
      this.addToLog(`${this.playerPokemon.name} utilise ${playerMove.
    name}... mais rate!`);
    } else {
      const damage = playerMove.power || 10;
      this.enemyPokemon.currentHp = Math.max(0, this.enemyPokemon.
    currentHp - damage);
      this.addToLog(`${this.playerPokemon.name} inflige ${damage}
    d  g  ts!`);
    }

    this.updateDisplay();

    if (this.enemyPokemon.currentHp <= 0) {
      this.endGame('player');
      return;
    }

    setTimeout(() => this.enemyAttack(), 1000);
  }

  enemyAttack() {
    const availableMoves = this.enemyPokemon.moves.filter(move => move.
    currentPp > 0);
```

```
    if (availableMoves.length === 0) return;

    const randomMove = availableMoves[Math.floor(Math.random() *
    availableMoves.length)];
    // Logique similaire pour l'attaque ennemie...
  }
}
```

Listing 2 – Logique de combat

2.4.3 Interface Utilisateur

```
<div class="battle-screen">
  <div class="battle-field">
    <!-- Pok mon ennemi -->
    <div class="pokemon enemy">
      <img id="enemy-sprite" src="" alt="Pok mon ennemi">
      <div class="health-bar">
        <div id="enemy-health" class="health"></div>
      </div>
      <span id="enemy-hp">300/300 PV</span>
    </div>

    <!-- Pok mon joueur -->
    <div class="pokemon player">
      <img id="player-sprite" src="" alt="Ton Pok mon">
      <div class="health-bar">
        <div id="player-health" class="health"></div>
      </div>
      <span id="player-hp">300/300 PV</span>
    </div>
  </div>

  <div class="moves-container">
    <div id="moves-list" class="moves-grid"></div>
  </div>
</div>
```

Listing 3 – Structure HTML du jeu

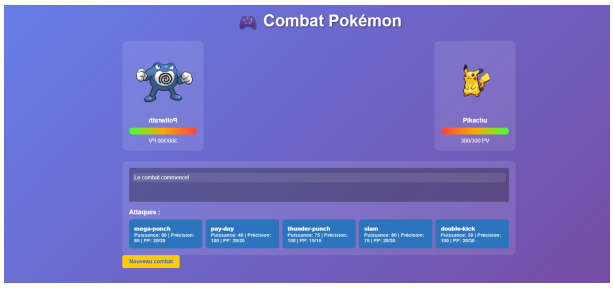
2.5 Difficultés Rencontrées et Solutions

2.6 Résultats et Validation

- Jeu de combat Pokémon entièrement fonctionnel
- Intégration réussie de l'API PokéAPI
- Système de combat équilibré avec PP et précision
- Interface utilisateur intuitive et responsive
- Gestion des erreurs et des cas limites

Problème	Solution
Structure complexe de l'API PokéAPI	Extraction sélective des données avec mapping personnalisé
Gestion asynchrone des appels API	Utilisation d'async/await avec gestion d'erreurs robuste
Synchronisation des tours de combat	Implémentation d'un système de callbacks avec setTimeout
Gestion des PP et précision	Création d'un système de règles métier complet
Interface utilisateur dynamique	Manipulation du DOM avec mise à jour en temps réel

TABLE 2 – Résolution des problèmes - TP1 Pokémon



3 TP2 : Application Express.js avec Authentification

3.1 Objectifs Spécifiques

- Créer une application web complète avec Express.js
- Implémenter un système d'authentification sécurisé
- Utiliser MongoDB pour le stockage des données utilisateurs
- Développer une interface avec le moteur de templates Pug
- Utiliser Tailwind CSS pour le styling moderne
- Protéger les routes avec des middlewares d'authentification

3.2 Architecture Technique

3.3 Structure du Projet

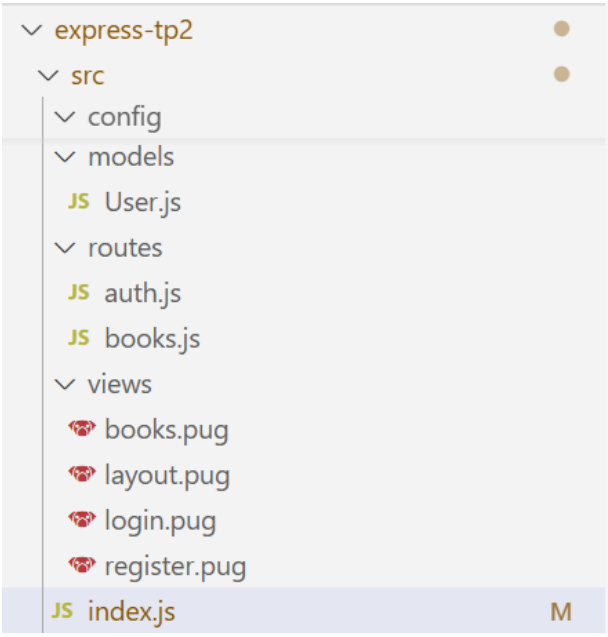
3.4 Fonctionnalités Implémentées

3.4.1 Système d'Authentification avec Passport.js

```
const passport = require('passport');
const LocalStrategy = require('passport-local').Strategy;
```

Composant	Technologie
Backend	Express.js, Node.js
Base de données	MongoDB avec Mongoose
Authentification	Passport.js, express-session
Templates	Pug (moteur de template)
Styling	Tailwind CSS
Sessions	connect-mongo
Sécurité	bcrypt pour le hashage des mots de passe

TABLE 3 – Stack technique de l'application Express.js




```
const User = require('../models/User');
const bcrypt = require('bcrypt');

passport.use(new LocalStrategy(
  async (username, password, done) => {
    try {
      const user = await User.findOne({ username });
      if (!user) {
        return done(null, false, { message: 'Utilisateur non trouv ' })
    ;
      }

      // V rification du mot de passe hash
      const isMatch = await bcrypt.compare(password, user.password);
      if (!isMatch) {
        return done(null, false, { message: 'Mot de passe incorrect' });
      }

      return done(null, user);
    } catch (error) {
      return done(error);
    }
  }
```

```
));

passport.serializeUser((user, done) => {
  done(null, user.id);
});

passport.deserializeUser(async (id, done) => {
  try {
    const user = await User.findById(id);
    done(null, user);
  } catch (error) {
    done(error);
  }
});
```

Listing 4 – Configuration Passport.js

3.4.2 Gestion des Sessions Sécurisées

```
const session = require('express-session');
const MongoStore = require('connect-mongo');

app.use(session({
  secret: process.env.SESSION_SECRET,
  resave: false,
  saveUninitialized: false,
  store: MongoStore.create({
    mongoUrl: process.env.DB_URL,
    ttl: 14 * 24 * 60 * 60 // 14 jours
  }),
  cookie: {
    secure: process.env.NODE_ENV === 'production',
    httpOnly: true,
    maxAge: 1000 * 60 * 60 * 24 // 1 jour
  }
}));
```

Listing 5 – Configuration des sessions

3.4.3 Middleware de Protection des Routes

```
function requireAuth(req, res, next) {
  if (req.isAuthenticated()) {
    return next();
  }
  res.redirect('/login');
}

function requireUnauth(req, res, next) {
  if (!req.isAuthenticated()) {
    return next();
  }
  res.redirect('/books');
}
```

```
14
15 // Application des middlewares
16 app.get('/books', requireAuth, (req, res) => {
17   res.render('books', { user: req.user });
18 });
19
20 app.get('/login', requireUnauth, (req, res) => {
21   res.render('login');
22 });
```

Listing 6 – Middleware d'authentification

3.4.4 Modèle Utilisateur avec Mongoose

```
1  const mongoose = require('mongoose');
2  const bcrypt = require('bcrypt');
3
4  const userSchema = new mongoose.Schema({
5    username: {
6      type: String,
7      required: true,
8      unique: true,
9      trim: true,
10     minlength: 3
11   },
12   password: {
13     type: String,
14     required: true,
15     minlength: 6
16   },
17   email: {
18     type: String,
19     required: true,
20     unique: true,
21     trim: true
22   },
23   createdAt: {
24     type: Date,
25     default: Date.now
26   }
27 });
28
29 // Hashage du mot de passe avant sauvegarde
30 userSchema.pre('save', async function(next) {
31   if (!this.isModified('password')) return next();
32   this.password = await bcrypt.hash(this.password, 12);
33   next();
34 });
35
36 module.exports = mongoose.model('User', userSchema);
```

Listing 7 – Modèle utilisateur MongoDB

3.5 Difficultés Rencontrées et Solutions

Problème	Solution
Configuration Passport.js complexe	Création d'un fichier de configuration dédié et documentation
Gestion des sessions MongoDB	Utilisation de connect-mongo avec configuration appropriée
Sécurité des mots de passe	Implémentation de bcrypt avec salt rounds appropriés
Protection des routes	Création de middlewares personnalisés pour différentes stratégies
Intégration Pug avec Express	Configuration correcte du moteur de templates et des paths

TABLE 4 – Résolution des problèmes - TP2 Express.js

3.6 Résultats et Validation

— Application Express.js complète et fonctionnelle
— Système d'authentification sécurisé avec hashage des mots de passe
— Sessions persistantes avec stockage MongoDB
— Interface moderne avec Pug et Tailwind CSS
— Routes protégées et gestion des accès
— Gestion des erreurs et messages utilisateur

4 TP3 : Book Reading Tracker en TypeScript

4.1 Objectifs Spécifiques

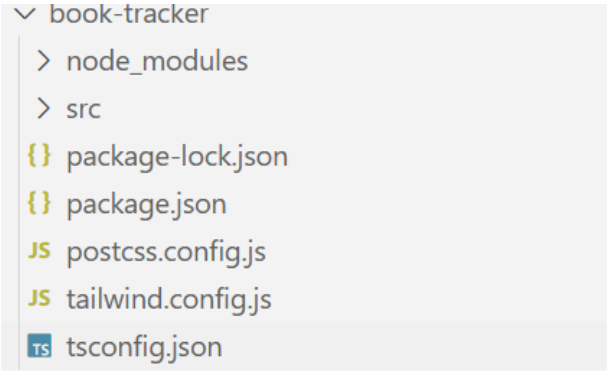
— Maîtriser TypeScript et ses fonctionnalités avancées
— Créer une application full-stack avec TypeScript
— Utiliser MongoDB avec Mongoose et typage TypeScript
— Implémenter des enums, interfaces et classes TypeScript
— Configurer un environnement de développement TypeScript complet
— Utiliser Tailwind CSS avec TypeScript

4.2 Architecture Technique

Composant	Technologie
Langage	TypeScript
Backend	Express.js avec TypeScript
Base de données	MongoDB avec Mongoose & TypeScript
Frontend	TypeScript compilé en JavaScript
Styling	Tailwind CSS
Outils de build	tsc, ts-node, nodemon
Validation	Types TypeScript, interfaces

TABLE 5 – Stack technique du Book Reading Tracker

4.3 Structure du Projet



4.4 Fonctionnalités Implémentées

4.4.1 Interfaces et Enums TypeScript

```
export enum BookStatus {
  Read = 'Read',
  ReRead = 're-read',
  DNF = 'DNF',
  CurrentlyReading = 'Currently reading',
  ReturnedUnread = 'Returned Unread',
  WantToRead = 'Want to read'
}

export enum BookFormat {
  Print = 'Print',
  PDF = 'PDF',
  Ebook = 'Ebook',
  AudioBook = 'AudioBook'
}

export interface IBook extends Document {
  title: string;
  author: string;
  numberOfPages: number;
  status: BookStatus;
  price: number;
  pagesRead: number;
  format: BookFormat;
  suggestedBy: string;
  finished: boolean;
  readingPercentage: number;
}

export interface BookStats {
  totalBooks: number;
  booksRead: number;
  totalPages: number;
  pagesRead: number;
  readingPercentage: number;
}
```

Listing 8 – Interfaces et enums TypeScript

4.4.2 Classe Book avec Méthodes Typées

```
export class Book {
  public finished: boolean;

  constructor(
    public title: string,
    public author: string,
    public numberOfPages: number,
    public status: BookStatus,
    public price: number,
    public pagesRead: number,
    public format: BookFormat,
    public suggestedBy: string,
    finished?: boolean
  ) {
```

```
    this.finished = finished !== undefined ? finished : this.pagesRead >= this.numberOfPages;
  }

  // Mettre    jour la progression de lecture
  currentlyAt(pagesRead: number): void {
    if (pagesRead < 0 || pagesRead > this.numberOfPages) {
      throw new Error('Nombre de pages invalide');
    }
    this.pagesRead = pagesRead;
    this.finished = this.pagesRead >= this.numberOfPages;
  }

  // Calculer le pourcentage de lecture
  getReadingPercentage(): number {
    return (this.pagesRead / this.numberOfPages) * 100;
  }

  // Marquer pour suppression
  deleteBook(): void {
    console.log(`Livre "${this.title}" marqu  pour suppression`);
  }

  // M thode statique pour cr er une instance depuis un objet
  static fromObject(obj: any): Book {
    return new Book(
      obj.title,
      obj.author,
      obj.numberOfPages,
      obj.status,
      obj.price,
      obj.pagesRead,
      obj.format,
      obj.suggestedBy,
      obj.finished
    );
  }
}
```

Listing 9 – Classe Book TypeScript

4.4.3 Modèle Mongoose avec Typage TypeScript

```
import { Schema, model, Document } from 'mongoose';
import { BookStatus, BookFormat, IBook } from '../types/Book';

const bookSchema = new Schema<IBook>({
  title: { type: String, required: true },
  author: { type: String, required: true },
  numberOfPages: { type: Number, required: true },
  status: {
    type: String,
    enum: Object.values(BookStatus),
    required: true
  },
  price: { type: Number, required: true },
```

```
  pagesRead: { type: Number, default: 0 },
  format: {
    type: String,
    enum: Object.values(BookFormat),
    required: true
  },
  suggestedBy: { type: String, required: true },
  finished: { type: Boolean, default: false },
  readingPercentage: { type: Number, default: 0 }
});

// Middleware pour calcul automatique
bookSchema.pre('save', function(next) {
  this.readingPercentage = (this.pagesRead / this.numberOfPages) * 100;
  this.finished = this.pagesRead >= this.numberOfPages;
  next();
});

export default model<IBook>('Book', bookSchema);
```

Listing 10 – Modèle Book avec Mongoose et TypeScript

4.4.4 Configuration TypeScript Complète

```
{
  "compilerOptions": {
    "target": "ES2020",
    "module": "CommonJS",
    "outDir": "./dist",
    "rootDir": "./src",
    "strict": true,
    "esModuleInterop": true,
    "skipLibCheck": true,
    "forceConsistentCasingInFileNames": true,
    "resolveJsonModule": true,
    "moduleResolution": "node",
    "typeRoots": ["./node_modules/@types"],
    "types": ["node"],
    "lib": ["ES2020", "DOM"],
    "declaration": true,
    "declarationMap": true,
    "sourceMap": true
  },
  "include": [
    "src/**/*.ts",
    "src/**/*.tsx"
  ],
  "exclude": [
    "node_modules",
    "dist",
    "**/*.test.ts",
    "**/*.spec.ts"
  ]
}
```

Listing 11 – Configuration tsconfig.json

4.4.5 Routes API Typées avec Express

```
import { Router, Request, Response } from 'express';
import Book, { IBook } from '../models/Book';
import { BookStats } from '../types/Book';

const router = Router();

// GET - Statistiques globales avec typage
router.get('/stats', async (req: Request, res: Response<BookStats | { error: string }>) => {
  try {
    const books = await Book.find();
    const totalBooks = books.length;
    const booksRead = books.filter(book => book.finished).length;
    const totalPages = books.reduce((sum, book) => sum + book.numberOfPages, 0);
    const pagesRead = books.reduce((sum, book) => sum + book.pagesRead, 0);

    const stats: BookStats = {
      totalBooks,
      booksRead,
      totalPages,
      pagesRead,
      readingPercentage: totalPages > 0 ? (pagesRead / totalPages) * 100 : 0
    };

    res.json(stats);
  } catch (error) {
    res.status(500).json({ error: 'Erreur lors du calcul des statistiques' });
  }
});

// POST - Nouveau livre avec validation de type
interface CreateBookRequest {
  title: string;
  author: string;
  numberOfPages: number;
  status: string;
  price: number;
  pagesRead: number;
  format: string;
  suggestedBy: string;
}

router.post('/', async (req: Request<{}, {}, CreateBookRequest>, res: Response) => {
  try {
    const bookData = req.body;
    const newBook = new Book(bookData);
    await newBook.save();
    res.status(201).json(newBook);
  } catch (error) {
    res.status(400).json({ error: 'Erreur lors de la cr ation du livre' });
```

```
  }
});

export default router;
```

Listing 12 – Routes Express avec typage TypeScript

4.5 Difficultés Rencontrées et Solutions

Problème	Solution
Erreurs de types avec les dépendances	Configuration de skipLibCheck et types explicites
Intégration Mongoose avec TypeScript	Création d'interfaces étendant Document
Compilation séparée frontend/backend	Configuration de multiples cibles et watch multiples
Gestion des enums dans MongoDB	Mapping des enums TypeScript vers strings MongoDB
Configuration complexe TypeScript	Création d'un tsconfig.json détaillé avec commentaires
Intégration Tailwind avec TypeScript	Configuration PostCSS et processus de build séparés

TABLE 6 – Résolution des problèmes - TP3 TypeScript

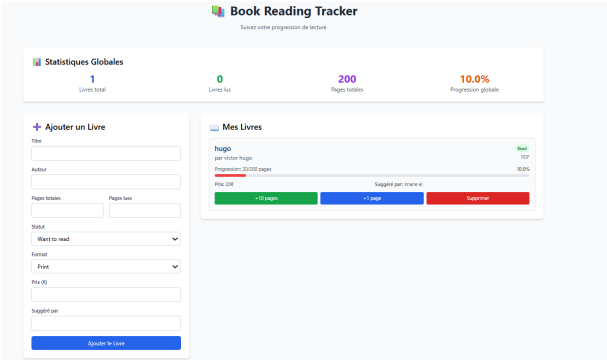
4.6 Résultats et Validation

- Application TypeScript full-stack entièrement fonctionnelle
- Typage fort avec interfaces, enums et génériques
- Intégration MongoDB avec typage complet
- Compilation et exécution sans erreurs de type
- Interface utilisateur moderne avec Tailwind CSS
- Gestion des livres avec statistiques avancées
- Code maintenable et extensible grâce au typage

5 Synthèse Comparative et Bilan

5.1 Analyse Comparative des Technologies

5.2 Évolution des Compétences



Aspect	TP1 Pokémon	TP2 Express	TP3 TypeScript
Complexité	Moyenne	Élevée	Très élevée
Type de projet	Application jeu	Application web	Application full-stack
Base de données	Aucune	MongoDB	MongoDB avec typage
Typage	Dynamique (JS)	Dynamique (JS)	Statique (TS)
Sécurité	Basique	Avancée	Très avancée
Maintenabilité	Moyenne	Bonne	Excellente
Évolutivité	Limitée	Bonne	Excellente

TABLE 7 – Comparaison des trois travaux pratiques

5.2.1 Compétences Techniques Acquises

— **JavaScript/TypeScript :** Maîtrise progressive des deux langages
— **Node.js/Express :** Architecture backend et API REST
— **Bases de données :** MongoDB avec Mongoose, schémas et modèles
— **Authentification :** Passport.js, sessions, sécurité
— **Frontend :** Templates Pug, Tailwind CSS, DOM manipulation
— **Outils :** npm, configuration, debugging, déploiement

5.2.2 Compétences Méthodologiques

— Gestion de projet et organisation du code * Résolution de problèmes techniques complexes * Documentation et commentaires * Tests et validation * Gestion des versions avec Git

5.3 Points Forts et Améliorations

5.3.1 Points Forts

— Progression logique dans la complexité technique * Diversité des technologies abordées * Approche pratique et concrète * Qualité du code produit * Documentation complète

5.3.2 Pistes d'Amélioration

— **Sécurité :** Implémentation de JWT, validation avancée * **Performance :** Mise en cache, optimisation des requêtes * **Testing :** Tests unitaires et d'intégration * **UI/UX :** Frameworks frontend modernes (React, Vue) * **DevOps :** Docker, CI/CD, déploiement cloud

6 Conclusion Générale

6.1 Bilan des Réalisations

À travers ces trois travaux pratiques, j'ai démontré ma capacité à :

— **TP1 Pokémon :** Créer une application interactive avec intégration d'API externe * **TP2 Express :** Développer une application web complète avec système d'authentification sécurisé * **TP3 TypeScript :** Maîtriser le développement full-stack typé avec architecture robuste

6.2 Apports Pédagogiques

Ces projets ont permis de :

* Comprendre l'écosystème JavaScript/Node.js dans sa globalité * Appréhender les avantages du typage statique avec TypeScript * Maîtriser les bonnes pratiques de sécurité web * Développer une approche méthodologique du développement

6.3 Perspectives

Les compétences acquises ouvrent la voie vers :

* Frameworks frontend modernes (React, Angular, Vue) * Architectures microservices et APIs GraphQL * Développement mobile (React Native, Ionic) * DevOps et déploiement cloud

Ce parcours a été extrêmement formateur et a considérablement renforcé mes compétences en développement web full-stack.

Annexes

Annexe A : Commandes de Démarrage par Projet

```
# Installation
npm install express axios

# D marrage
npm run dev
# ou
node server.js
```

Listing 13 – Commandes TP1 - Pokémon

```
# Installation
npm install express mongoose passport passport-local express-session
    connect-mongo bcrypt pug

# D marrage
npm run dev
# ou
npx nodemon src/index.js
```

Listing 14 – Commandes TP2 - Express.js

```
# Installation
npm install express mongoose
npm install -D typescript ts-node @types/express @types/node @types/
    mongoose

# Compilation et d marrage
npm run dev
# ou
npx tsc && node dist/server.js
```

Listing 15 – Commandes TP3 - TypeScript

Annexe B : Captures d'Écran des Applications

Note : Les captures d'écran des trois applications fonctionnelles sont disponibles séparément et démontrent :

— Interface du jeu Pokémon avec sélection et combat * Pages de connexion et gestion des livres de l'application Express * Interface du book tracker avec statistiques et progression

Annexe C : Références Techniques

— Documentation PokéAPI
— Documentation Express.js

— Documentation TypeScript
— Documentation Mongoose
— Documentation Tailwind CSS
— Documentation Passport.js

Annexe D : Code Source Complet

Les codes sources complets des trois projets sont disponibles aux adresses suivantes :

— **TP1 Pokémon :** `https://github.com/votre-compte/pokemon-game`
— **TP2 Express :** `https://github.com/votre-compte/express-auth`
— **TP3 TypeScript :** `https://github.com/votre-compte/book-tracker-ts`