**M2i Formation**

Décembre 2025

**Dossier Professionnel**

**Concepteur Développeur d’Applications**

**Ruben Trottein**

Table des matières

Contenu

[1. Introduction 2](#_Toc183697180)

[2. Description des projets réalisés 3](#_Toc183697181)

[A. Projets CDA 3](#_Toc183697182)

[APIs 3](#_Toc183697183)

[Applications 4](#_Toc183697184)

[API 6](#_Toc183697185)

[B. Projets perso 7](#_Toc183697186)

[3. Développement des compétences 8](#_Toc183697187)

[4. Résultats et bilan 9](#_Toc183697188)

[5. Annexes 10](#_Toc183697189)

# Introduction

*Présentez-vous : parcours, expériences, motivations pour devenir CDA.*



# Description des projets réalisés

*Chaque projet doit être décrit en détail. Incluez :*

*Titre et description du projet :*

*Nature du projet (application web, application mobile, etc.).*

*Contexte (professionnel, formation, stage, freelance, etc.).*

*Objectif (résoudre un problème spécifique, répondre à un besoin client).*

*Analyse préalable :*

*Expression des besoins du client ou de l'utilisateur.*

*Cahier des charges ou spécifications fonctionnelles.*

*Conception et organisation :*

*Diagrammes UML (use case, classes, séquence).*

*Modèle conceptuel de données (MCD).*

*Architecture choisie (MVC, microservices, etc.).*

*Outils ou méthodologies utilisés (Agile, Scrum, etc.).*

*Réalisation :*

*Technologies utilisées (langages, frameworks, bases de données).*

*Description des fonctionnalités développées.*

*Extraits de code significatifs (commentés et expliqués).*

*Gestion des versions (Git, GitHub/GitLab).*

*Problèmes rencontrés et solutions :*

*Décrivez les obstacles techniques ou organisationnels et comment vous les avez surmontés.*

*Tests et validation :*

*Types de tests réalisés (unitaires, fonctionnels, utilisateurs).*

*Résultats des tests.*

*Livrables :*

*Présentez ce qui a été livré (code source, documentation, déploiement).*

*Page notion :*

<https://mistertea.notion.site/Index-des-projets-CDA-65e4bd8d7cec45178ff38c3cbcfcc07e>

## Projets CDA

### APIs

#### Node.js noSql MongoDB (API / NoSql);

* API Zoologique d’identification d’Ours : SQL → CRUD;
* Movies / Netflix : SQL → CRUD;
* Foodtrucks : Node.js noSql MongoDB → CRUD ;
* Users : Node.js Sql + MongoDB → CRUD ;
* API Utilisateurs Securisée: Node.js noSql MongoDB → CRUD + JWT;

### Applications

#### Symfony

* PartageToo : Site de troc/commerce style LeBonCoin
* Actunews : Blog d’actualités
* ForumNews : Réseau social de réaction / commentaire sur des articles

#### Php

* Surf : Blog en Php procédural, (Sql, Bootstrap)
* Marmitard : Blog culinaire Php procédural orienté Objet

#### Java Spring (Web):

* PayMe : Site de paiements style Lydia (Java, Spring, Thymeleaf, Sql)
* PayMe Hacking BackEnd : Site de paiements avec backdoor pirate (Java, Spring, Thymeleaf, Sql)
* Festival version Web : Site de gestion evenementielle. Ajout de groupes avec leurs scène a un ou plusieurs festivals à venir. Version Thymeleaf / Spring / SQL
* FirstApp : application avec rendu sans templating (Spring uniquement)

#### React

* Pokedex-React : React+ interrogation API

#### NodeJS / Symfony (BDDs Relationnelles) :

* festival (Java + Sql)
* blog (Sql via MCD)
* Actunews ( Sql via Doctrine )

#### JavaScript (Full Front-end)

* Histoire interactive (HTML/CSS)
* Intégrations psd to html : recipe-page, Vitrines (Assurance / Boutique)
  + SunnySide (Défi FrontEnd Mentor)

<https://sunnyside-challenge-chi.vercel.app/>

* + W-Challenge (FrontEnd Mentor)

<https://rubentrottein.github.io/w-challenge/>

* To-Do List Full JS
* Quiz Full JS

#### Java (Console)

* Calculatrice
* Zelda
* Garage
* Festival version console
* Chifumi
* Potager
* Médiathèque
* Appli Notes

#### Kotlin (Mobile)

* Calculatrice
* To-Do list (Array)
* Simon
* UserList (ArrayList)

#### Ionic (Mobile)

* Liste Triable des directives Ionic
* Tableau Front-end de liste d'utilisateurs
* To-do list étendue avec API de sauvegarde
* Application de gestion utilisateurs (Ionic-storage / Local storage)

### API

#### FoodTruck.js :

API Restauration rapide: MongoDB→ CRUD :

Nature du projet*: API RESTful*

Contexte*: Formation*

Objectif*: Gérer une liste de FoodTrucks avec leur menu consultable*

Analyse préalable *:*

*On cherche à avoir de quoi identifier chaque Foodtruck et son menu , on les renvoie en JSON, et on peut créer de nouvelles fiches ou les modifier.*

Cahier des charges ou spécifications fonctionnelles*:*

*Création d’une application node.js en environnement de développement, récupération des informations avec postman.*

##### Conception et organisation :

##### Diagrammes UML (use case, classes, séquence).

##### Modèle conceptuel de données (MCD)

##### .

Architecture choisie *: MVC avec le rôle de la vue simplifié (retour JSON)*

*Outils ou méthodologies utilisés : Travaux dirigés*

##### Réalisation :

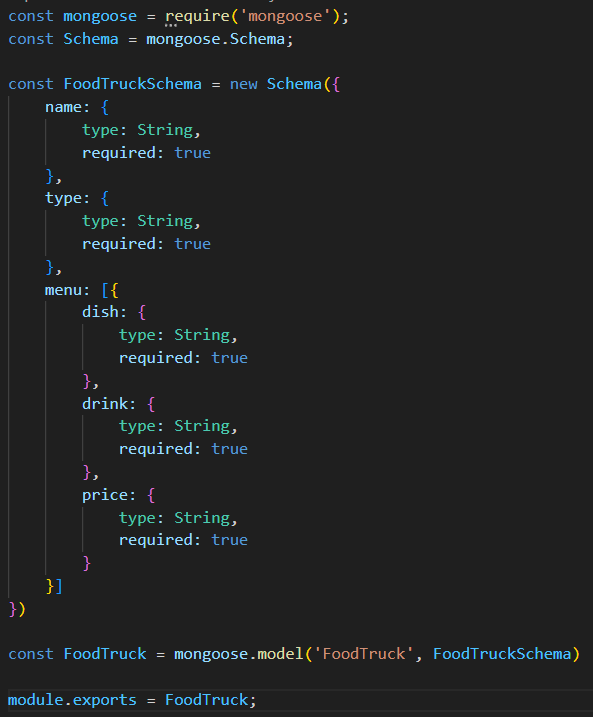
*Technologies utilisées : node.js, express.js, SQL, Postman*

*Description des fonctionnalités développées : CRUD simple*

*Extraits de code significatifs (commentés et expliqués).*

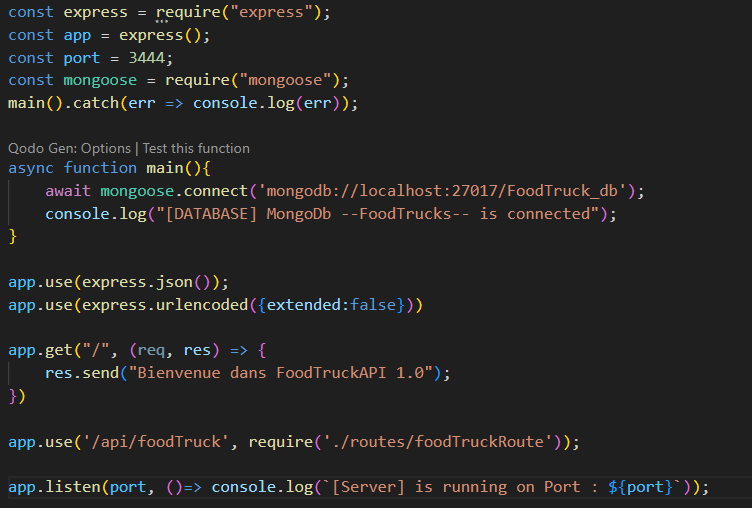
Modèle :

Notre modèle contient un array pour les menu. Il est au format des schémas Mongoose



Routes :

Notre routeur express nous permet une gestion RESTful des endpoint et gère l’accès aux données



##### Gestion des versions : Git

##### Problèmes rencontrés et solutions :

*Abstraction souhaitée sur l’application afin de récupérer la structure et la réutiliser. Apprentissage de node et express*

*Pas de déploiement*

##### Tests et validation :

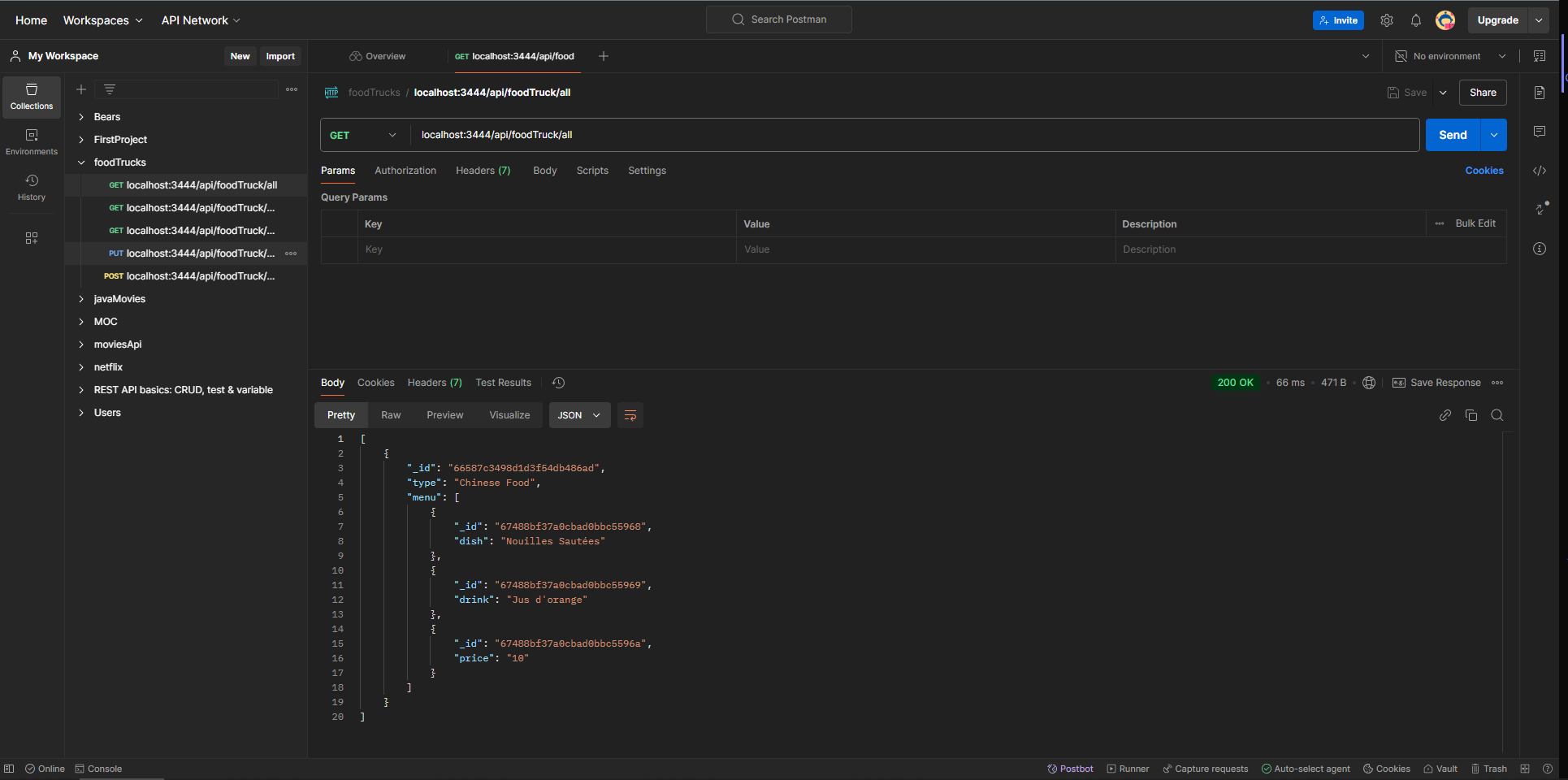
*Types de tests réalisés ; unitaires*

*Résultats des tests : Vérification du fonctionnement de l’API et du retour JSON*

##### Livrables :

Vue :

Retour JSON du GET All sous Postman



#### Bears.js :

API Zoologique d’identification d’Ours : SQL → CRUD :

Nature du projet*: API RESTful*

Contexte*: Formation*

Objectif*: Repertorier les espèces d’Ours et leurs caractéristiques*

Analyse préalable *:*

*On cherche a avoir quelques informations pertinentes pour chacun de nos Ours, on les affiche, et on peut créer de nouvelles fiches ou les modifier.*

Cahier des charges ou spécifications fonctionnelles*:*

*Création d’une application node.js en environnement de développement, récupération des information avec js fetch ou postman.*

##### Conception et organisation :

##### Diagrammes UML (use case, classes, séquence).

##### Modèle conceptuel de données (MCD).

Architecture choisie *(MVC, microservices, etc.). : MVC avec le rôle de la vue simplifié (retour JSON)*

*Outils ou méthodologies utilisés : Travaux dirigés*

##### Réalisation :

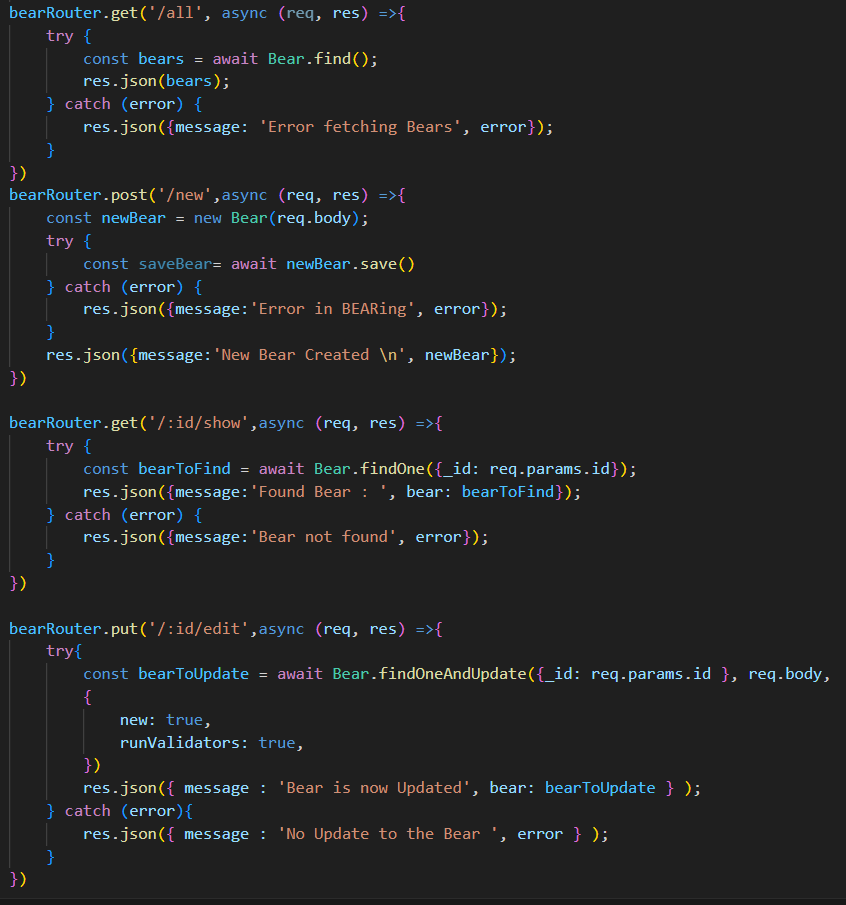
*Technologies utilisées : node.js, express.js, SQL, Postman, HTML, CSS, vanilla JS*

*Description des fonctionnalités développées : CRUD simple et affichage navigateur*

*Extraits de code significatifs (commentés et expliqués).*

Contrôleur :

Dans nos route on définit les endpoint du crud et on gère les erreurs. Ces endpoint sont utilisables directement via PostMan



Routes :

Notre routeur express nous permet une gestion RESTful des endpoint et gère l’accès aux données



##### Gestion des versions : Git

##### Problèmes rencontrés et solutions :

*Abstraction souhaitée sur l’application afin de récupérer la structure et la réutiliser. Apprentissage de node et express*

*Pas de déploiement*

##### Tests et validation :

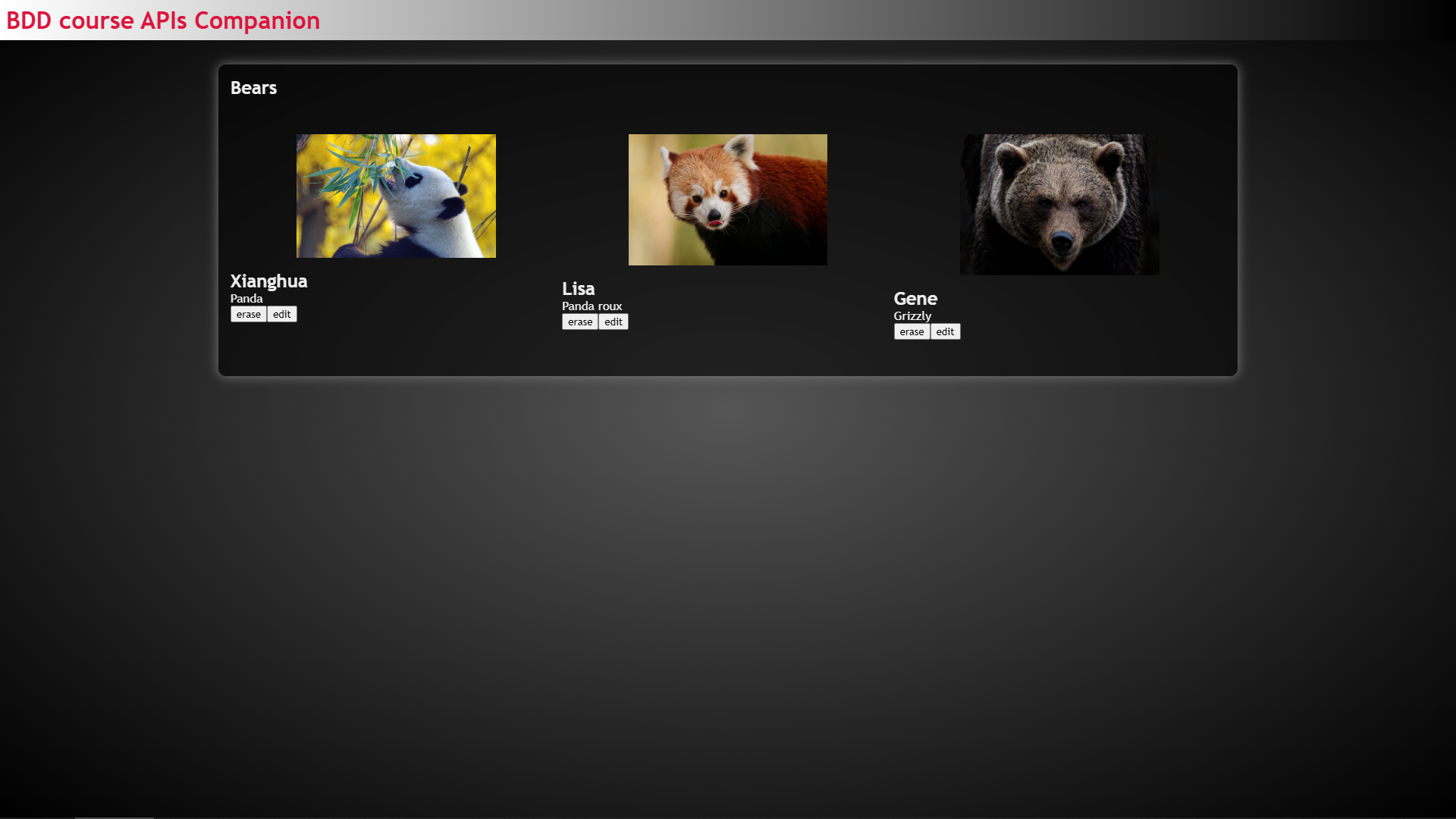
*Types de tests réalisés ; unitaires*

*Résultats des tests : Vérification du fonctionnement de l’API et du retour JSON*

##### Livrables :

Vue

Petite page web simple avec un fetch javascript pour afficher les Ours et les supprimer. Formulaire d’édition



#### MoviesApi

API repertoire de films : MongoDB → Application Web:

Nature du projet*: Application web Vanilla JS*

Contexte*: Formation*

Objectif*: Répertorier des films et leurs caractéristiques*

Analyse préalable *:*

*On cherche à avoir quelques informations pertinentes pour chacun de nos films, on les affiche, et on peut créer de nouvelles fiches ou les modifier / supprimer.*

Cahier des charges ou spécifications fonctionnelles*:*

*Création d’une application node.js en environnement de développement, récupération des informations avec js fetch et affichage sur une page web.*

##### Conception et organisation :

##### Diagrammes UML (use case, classes, séquence).

##### Modèle conceptuel de données (MCD).

Architecture choisie*: MVC*

*Outils ou méthodologies utilisés : Travaux dirigés*

##### Réalisation :

*Technologies utilisées : node.js, express.js, MongoDB, Postman, HTML, CSS, vanilla JS*

*Description des fonctionnalités développées : CRUD simple et affichage navigateur*

*Extraits de code significatifs (commentés et expliqués).*

##### Liste des endpoints ;

Chacun des endpoints permet une opération sur un des trois modèles disponibles (User, Movie ou Category)  
La route / utilise uniquement le GET des films tandis que les routes login et logout identifient un utilisateur test actuellement présent en dur dans la BDD

Les Sample Data créent un jeu de données rapide au premier lancement de l’application pour peupler la vue d’accueil.



##### Gestion des versions : Git

##### Problèmes rencontrés et solutions :

*Abstraction souhaitée sur l’application afin de récupérer la structure et la réutiliser. Utilisation d’un utilisateur autorisé*

*Pas de déploiement*

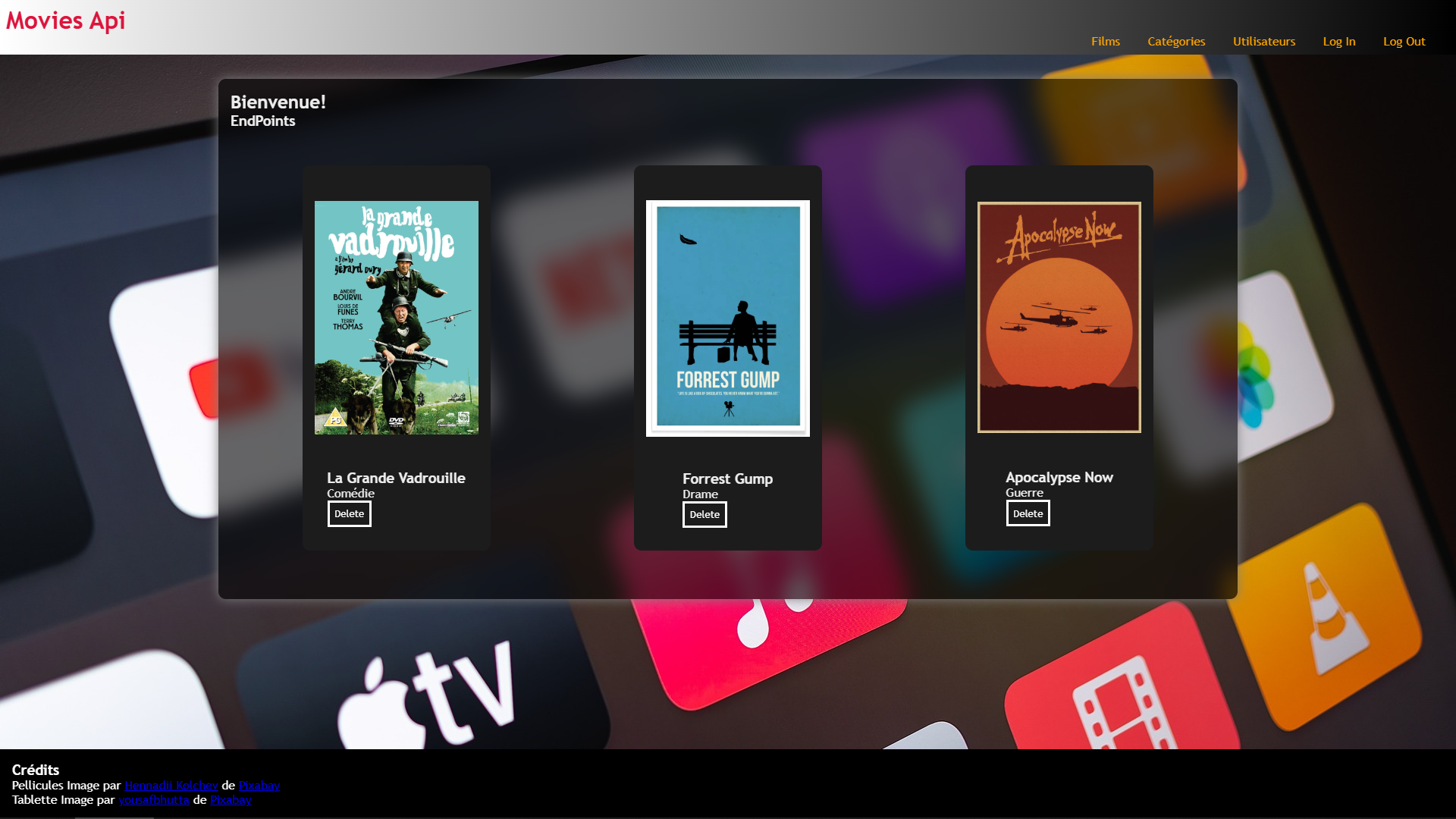
##### Tests et validation :

*Types de tests réalisés ; unitaires*

*Résultats des tests : Vérification du fonctionnement de l’API et du retour JSON*

##### Livrables :

Vue permettant de manager la liste de films. Impossible de fonctionner sans utiliser la commande login de la nav



#### UsersAPI :

API liste d’utilisateurs : Mongo DB + MySQL → Application Web:

Nature du projet*: API + Application web Vanilla JS*

Contexte*: Formation*

Objectif*: Gerer une liste d’utilisateurs répartis sur deux SGBD*

Analyse préalable *:*

*Liste très simple d’utilisateurs pouvant interroger et intervenir sur une BDD relationnelle et une BDD noSql simultanément*

Cahier des charges ou spécifications fonctionnelles*:*

*Création d’une application node.js hydratée à la fois par une BDD SQL et une BDD MongoDB en environnement de développement, récupération des information avec js fetch et affichage sur une page web.*

##### Conception et organisation :

##### Diagrammes UML (use case, classes, séquence).

##### Modèle conceptuel de données (MCD).

Architecture choisie*: MVC*

*Outils ou méthodologies utilisés : Travaux dirigés*

##### Réalisation :

*Technologies utilisées : node.js, express.js, MongoDB, SQL, Postman, HTML, CSS, vanilla JS*

*Description des fonctionnalités développées : Double connexion et affichage navigateur*

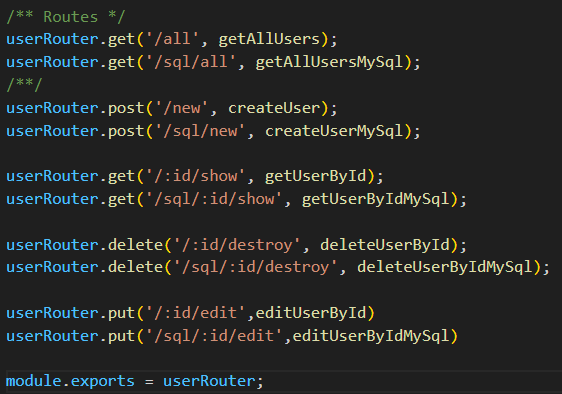
*Extraits de code significatifs (commentés et expliqués).*

##### Double connexion MySQL / Mongo DB :

##### 

##### Liste des endpoints ;

Chacun des endpoints est doublé pour une utilisation SQL ou Mongo DB



##### Gestion des versions : Git

##### Problèmes rencontrés et solutions :

*Abstraction souhaitée sur l’application afin de récupérer la structure et la réutiliser. Utilisation de deux SGBD par la même API*

##### Tests et validation :

*Types de tests réalisés ; unitaires*

*Résultats des tests : Vérification du fonctionnement de l’API et du retour JSON*

##### Livrables :

Vue récupérant les données de deux SGBD différent. Une liste MongoDb et une liste MySQL



## Projets perso

# Développement des compétences

*Pour chaque activité-type du titre professionnel, illustrez les compétences développées. Le référentiel est organisé en deux activités principales :*

*Développer des composants d'interface utilisateur.*

*Exemples : création d'une page web avec React ou Symfony, intégration d'APIs.*

*Lien avec vos projets.*

*Développer la persistance des données et des composants back-end.*

*Exemples : développement d'une API REST, gestion des bases de données.*

*Lien avec vos projets.*

## Objectifs de formation (m2i)

A l’issue de cette formation, vous serez capable de :

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

* Maquetter une application
* Développer une interface utilisateur de type desktop
* Développer des composants d'accès aux données
* Développer la partie front-end d'une interface utilisateur Web
* Développer la partie back-end d'une interface utilisateur Web

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

* Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement
* Concevoir une application
* Développer des composants métier
* Construire une application organisée en couches
* Développer une application mobile
* Préparer et exécuter les plans de tests d'une application
* Préparer et exécuter le déploiement d'une application

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

* Concevoir une base de données
* Mettre en place une base de données
* Développer des composants dans le langage d'une base de données

# Résultats et bilan

*Auto-évaluation :*

*Points forts et axes d'amélioration.*

*Compétences acquises pendant la formation ou le projet.*

*Retour client ou utilisateur :*

*Si possible, incluez des témoignages ou retours des clients.*

*Perspectives :*

*Objectifs professionnels futurs.*

*Domaines dans lesquels vous souhaitez évoluer (web, mobile, IA, DevOps, etc.).*

# Annexes

Documentation technique des projets.

Diagrammes (MCD, UML).

Extraits de cahiers des charges.

Captures d’écran des applications ou démonstrations vidéo (liens vers un dépôt GitHub ou un portfolio en ligne si applicable).