

Portfolio Dynamique en React.js et Symfony

1 Présentation du Projet

Ce projet consiste à développer un **portfolio interactif et dynamique**, permettant de présenter les projets de développement de manière professionnelle.

L'objectif est d'avoir un site **moderne, responsive et immersif**, avec une gestion centralisée des projets via une **base de données et une API Symfony**.

Points Clés du Projet

- **Technologies modernes** : React.js pour le frontend, Symfony pour le backend.
- **Gestion des projets** : Stockage des données en **base de données MySQL**.
- **Affichage dynamique** : Les projets sont récupérés via une **API Symfony** et affichés dynamiquement.
- **Carrousel d'images** : Chaque projet peut contenir plusieurs images affichées sous forme de **carrousel interactif** (clic + auto-scroll toutes les 5 secondes).
- **Expérience immersive** : Design inspiré des **bornes d'arcade rétro**, sans cadre visible pour une immersion totale.
- **Responsive Design** : Optimisation pour **desktop et mobile**.

2 Fonctionnalités

Frontend (React.js)

Page d'accueil

- Présentation rapide du développeur avec une mise en page immersive.

Section Projets

- **Liste dynamique des projets** récupérés depuis l'API Symfony.
- Chaque projet affiche un **titre** et permet d'ouvrir un **popup détaillé**.

Popup Projet

- **Titre** affiché en haut du popup.
- **Grande image du projet** (carrousel interactif au clic + auto-scroll toutes les 5 secondes).
- **Description et fonctionnalités** affichées à droite du popup.
- **Effets interactifs et design arcade** (hover, transitions fluides, effets néon verts).

Navigation et Téléchargement

- **Navbar dynamique** permettant une navigation fluide entre les sections.
- **Ajout d'un lien dans la navbar** pour télécharger un **CV en PDF** ([CV_louislegouge.pdf](#)).

Backend (Symfony)

API RESTful pour la Gestion des Projets

L'API a été développée avec **Symfony** et permet au **frontend React.js** d'accéder aux données des projets en **JSON**.

Méthode	Route	Action
GET	/api/projects	Retourne la liste des projets avec leurs images
POST	/api/projects	Permet d'ajouter un projet
PUT	/api/projects/{id}	Modifier un projet existant
DELETE	/api/projects/{id}	Supprime un projet

Base de Données (MySQL)

L'API repose sur une base de données relationnelle avec **deux tables** :

Table **projects** (Stocke les projets)

id	title	description	functionalities
1	"Projet A"	"Description..."	"React, Symfony, API..."

Table **project_images** (Stocke les images des projets)

id project_id image_url

1 1 "https://..."

2 1 "https://..."

3 Design & Expérience Utilisateur

Mise en Page du Popup Projet

Structure du popup :

- **Titre en haut**
- **Image principale à gauche (carrousel interactif)**
- **Description en haut à droite**
- **Liste des fonctionnalités en bas à droite**

Effets et animations :

- **Hover interactif** pour améliorer l'expérience utilisateur.
- **Animations fluides** pour les transitions entre les sections.
- **Effet néon vert** pour renforcer l'esthétique arcade.

Responsive Design :

- **Desktop** : Affichage en **grille à 2 colonnes**.
- **Mobile** : Passage en **une seule colonne** avec l'image en haut et la description en dessous.

4 Moodboard & Maquette

Moodboard

Un **moodboard** est ajouté pour illustrer l'ambiance graphique et l'inspiration visuelle du portfolio. Il sert de référence pour :

- Les **couleurs dominantes** (vert #00ff00; et noir).
- Le **style général** inspiré des bornes d'arcade.
- Les **animations et effets visuels** souhaités.

Maquette

Une **maquette détaillée** du site est réalisée pour visualiser la disposition des éléments avant l'implémentation.

Elle inclut :

- **La structure des pages** (sections, navbar, popup projet).
- **Les tailles et emplacements des éléments** (carrousel, boutons, textes).
- **L'organisation responsive** (desktop vs mobile).

Images du moodboard et de la maquette à intégrer une fois finalisées.

5 Constraintes Techniques

- **Frontend** : Développé avec **React.js**.
- **Backend** : Développé avec **Symfony**.
- **Utilisation d'Axios** pour la communication entre React et l'API Symfony.
- **Gestion de l'état** avec **useState** et **useEffect** pour les requêtes API et le carrousel.
- **Gestion des routes Symfony** (`debug:router` doit afficher `/api/projects`).
- Utilisation de CSS Grid/Flexbox pour la mise en page.

6 Planification & Prochaines Étapes

Étape 1 – Mise en place du Back-end

- ✓ **Création des entités Symfony** (`Project` & `ProjectImage`).
- ✓ **Création du contrôleur API** (`ProjectController`).
- ✓ **Configuration des routes et test de l'API** via `http://127.0.0.1:8000/api/projects`.

Étape 2 – Mise en place du Front-end

- ✓ **Intégration d'Axios** pour récupérer les projets.
- ✓ **Création du composant `Projects.js`** affichant la liste des projets.
- ✓ **Intégration du popup projet avec le carrousel interactif.**

Étape 3 – Finalisation et Tests

Optimisation CSS et animations interactives.
Tests sur mobile et ajustements responsive.
Vérification et ajout d'un système d'ajout/modification de projets.

7 Conclusion

Ce portfolio sera **moderne**, **dynamique** et **facilement extensible**. Une fois finalisé, il pourra être **hébergé en ligne** avec un backend accessible via **API**.

Évolution possible :

- Ajout d'une **interface d'administration** pour gérer les projets depuis une page dédiée.
- Ajout d'**effets visuels avancés** (glitches, filtres CRT).
- Optimisation pour **améliorer le référencement (SEO)** et la performance globale.

Objectif final :

Créer un **portfolio professionnel** qui met en avant les **compétences et projets** du développeur avec une **expérience unique et immersive**.