Rapport de Projet de Développement Front-end 3IIR Groupe 7: Crypto Portfolio Moderne en Vanilla Javascript

Par: Adam Radi 2025/12/17

Introduction

Au cours des dernières années, les cryptomonnaies ont connu une adoption croissante, tant auprès des investisseurs particuliers que des acteurs institutionnels. Cette évolution s’accompagne d’une diversification rapide des actifs numériques disponibles sur le marché, rendant la gestion d’un portefeuille de cryptomonnaies de plus en plus complexe. Dans ce contexte, les outils numériques permettant de suivre, analyser et organiser ces investissements jouent un rôle essentiel pour aider les utilisateurs à prendre des décisions éclairées.

Parallèlement à cette évolution du secteur financier, le développement web connaît lui aussi une transformation continue. Les interfaces web modernes doivent répondre à des exigences élevées en matière d’ergonomie, de performance et de clarté visuelle. Si de nombreux frameworks et bibliothèques JavaScript facilitent aujourd’hui la création d’applications complexes, il reste fondamental, notamment dans un cadre pédagogique, de maîtriser les technologies web natives. La compréhension approfondie de HTML, CSS et JavaScript constitue en effet la base de tout développement front-end robuste et maintenable.

C’est dans cette optique que s’inscrit ce projet, qui consiste en le développement d’une application web front-end dédiée à la gestion des portefeuilles de cryptomonnaies. L’application a été conçue exclusivement à l’aide de technologies web natives, à savoir HTML

pour la structuration du contenu, CSS pour la mise en forme et le design de l'interface, et JavaScript vanilla pour la gestion de la logique applicative et des interactions utilisateur. Aucun framework ou bibliothèque externe n'a été utilisé, sauf chart.js qui est explicitement mentionée dans le cahier de charges fourni, afin de mettre en évidence les capacités offertes par ces technologies fondamentales.

L'objectif principal de ce projet est de concevoir une interface moderne, intuitive et fonctionnelle permettant aux utilisateurs de suivre et de gérer leurs investissements en cryptomonnaies. L'application vise à centraliser les informations essentielles liées à un portefeuille, telles que la répartition des actifs, la visualisation des données et la navigation entre différentes vues de gestion. Elle se présente sous la forme d'une application monopage (Single Page Application / SPA), ce qui permet d'offrir une expérience utilisateur fluide sans rechargement complet de la page lors des changements de vue.

Le projet a été structuré autour de trois vues principales, chacune répondant à un besoin spécifique dans la gestion d'un portefeuille de cryptomonnaies. Cette organisation permet de séparer clairement les fonctionnalités tout en assurant une navigation cohérente et efficace. Le choix d'une architecture simple et modulaire vise également à faciliter la maintenance du code et son éventuelle évolution.

Au-delà de l'aspect fonctionnel, ce projet a pour ambition de démontrer les compétences en développement front-end acquises au cours de la formation et avant. Il met en avant la capacité à concevoir une application web complète, depuis la réflexion sur l'expérience utilisateur jusqu'à l'implémentation technique, tout en respectant les bonnes pratiques de développement. L'accent est mis sur la lisibilité du code, la séparation des responsabilités et l'optimisation des interactions côté client.

Enfin, ce rapport a pour objectif de présenter de manière détaillée les différentes étapes du développement de l’application. Il abordera dans un premier temps le contexte et les concepts théoriques liés aux portefeuilles de cryptomonnaies et au développement front-end. Il décrira ensuite les choix techniques et l’architecture de l’application, avant d’analyser les fonctionnalités implémentées et les résultats obtenus. Le rapport se conclura par une discussion des limites du projet ainsi que des pistes d’amélioration et d’évolution possibles.

Motivation du Projet

Choix du Sujet

Le choix de développer un portefeuille de cryptomonnaies découle de deux motivations principales. Premièrement, un intérêt personnel fort pour l'univers des cryptomonnaies et la technologie blockchain qui représente une innovation technologique fascinante. Deuxièmement, la volonté de créer quelque chose d'original et différent par rapport aux projets classiques que les autres développeurs pourraient réaliser.

La blockchain étant une technologie révolutionnaire qui transforme de nombreux secteurs, il était naturel de s'orienter vers un projet qui touche à cet écosystème. Cela permet également de se familiariser avec les concepts et les outils liés au monde de la finance décentralisée.

Technologies Choisies

Le projet utilise exclusivement du JavaScript vanilla, HTML et CSS sans frameworks externes. Cette approche permet de maîtriser les concepts fondamentaux du développement web et de créer une

application légère et performante. La seule exception concerne l'utilisation de Chart.js pour la création de graphiques, car implémenter des graphiques complexes depuis zéro représente une tâche particulièrement difficile et chronophage.

Architecture et Fonctionnalités

Structure de l'Application

L'application est développée comme une Single Page Application (SPA). Bien qu'elle présente trois sections distinctes qui pourraient être perçues comme des pages séparées, elles sont toutes contenues dans un seul fichier HTML et gérées par JavaScript. La navigation entre les sections se fait par l'affichage et le masquage d'éléments DOM selon les interactions de l'utilisateur.

Les Trois Sections Principales

- **Dashboard (Tableau de Bord)**

Le tableau de bord constitue la vue principale de l'application. Il présente plusieurs éléments informatifs :

- Un tracker de prix Bitcoin en temps réel qui affiche la valeur actuelle de la cryptomonnaie de référence
- Des données sur la valorisation totale des actifs possédés par l'utilisateur
- Un aperçu des différents portefeuilles créés
- Un graphique circulaire montrant la répartition des portefeuilles selon leur valeur monétaire
- Un graphique en barres détaillant la composition spécifique des actifs (cryptomonnaies) dans chaque portefeuille

- **Assets (Actifs)**

Cette section permet la gestion complète des actifs cryptographiques. L'utilisateur peut effectuer toutes les opérations CRUD (Create, Read, Update, Delete) sur ses actifs :

- Ajouter de nouveaux actifs en spécifiant le symbole de la cryptomonnaie
- Modifier les quantités possédées
- Supprimer des actifs du portefeuille
- Le prix de chaque cryptomonnaie est récupéré dynamiquement via l'API CoinGecko au moment où l'utilisateur saisit le symbole
- Une fonction de recherche basique permet de filtrer les actifs stockés localement selon des critères spécifiques

- **Portfolio**

Cette section offre une gestion simple des portefeuilles. Les fonctionnalités incluent :

- Création de nouveaux portefeuilles avec un nom personnalisé
- Suppression de portefeuilles existants
- Chaque portefeuille est identifié par un ID unique et un nom défini par l'utilisateur

Implémentation Technique

Gestion des Données

L'application utilise le **localStorage** du navigateur pour persister les données utilisateur. Cette approche permet de conserver les informations entre les sessions sans nécessiter de base de données externe ni de configuration côté serveur. Elle offre une solution simple et efficace pour stocker les données dans le cadre d’une application front-end autonome.

Le recours au localStorage présente également l’avantage d’une mise en oeuvre rapide et d’un accès direct aux données depuis le navigateur. Les informations sont stockées sous forme de paires clé-valeur et peuvent être lues ou modifiées à tout moment par l’application. Cette méthode convient particulièrement aux projets de taille réduite ou à vocation pédagogique, où la priorité est donnée à la démonstration des mécanismes de persistance plutôt qu’à la gestion avancée des données.

Cependant, cette solution comporte certaines limitations. Les données stockées sont spécifiques au navigateur et à l’appareil utilisé, ce qui empêche toute synchronisation entre plusieurs environnements. De plus, le localStorage n’est pas conçu pour gérer des volumes de données importants ni pour garantir un niveau de sécurité élevé. Malgré ces contraintes, son utilisation reste adaptée au périmètre du projet et permet de répondre efficacement aux besoins fonctionnels définis.

Le localStorage contient deux structures de données principales pour gérer les données du projet:

1. Tableau des Actifs: Chaque actif stocké comprend un identifiant unique, le prix par unité, la quantité possédée et l'identifiant du portefeuille auquel il appartient.
2. Tableau des Portefeuilles: Structure plus simple contenant uniquement l'identifiant unique et le nom du portefeuille.

Intégration API et KPI

L’application s’appuie sur l’API CoinGecko afin de récupérer les données de marché nécessaires au calcul des indicateurs clés de performance (KPI). Les données fournies par l’API, notamment les prix des cryptomonnaies, constituent des informations brutes qui sont ensuite traitées par l’application pour produire des indicateurs exploitables par l’utilisateur.

À partir de ces données, l’application calcule plusieurs KPI permettant d’évaluer la répartition et la valeur des investissements. Le premier indicateur correspond à la répartition de la valeur totale entre les différents portefeuilles, exprimée en valeur monétaire et en pourcentage. Cet indicateur offre une vision globale de l’allocation des fonds et permet d’identifier rapidement les portefeuilles dominants.

Le second indicateur met en évidence la composition détaillée des portefeuilles en fonction des cryptomonnaies détenues. Il permet de visualiser, pour chaque portefeuille, la contribution de chaque actif à la valeur totale. Cet indicateur apporte une lecture plus fine de l’exposition aux différentes cryptomonnaies et complète l’analyse globale fournie par le premier KPI.

Ces indicateurs sont représentés graphiquement afin d’en faciliter l’interprétation. La répartition de la valeur entre les portefeuilles est affichée à l’aide d’un graphique circulaire, tandis que la composition des portefeuilles par cryptomonnaie est présentée sous la forme d’un graphique en barres. L’utilisation de ces visualisations permet une compréhension rapide des données et aide l’utilisateur à analyser la structure de ses investissements

Précision sur le Calcul et la Mise à Jour des Indicateurs

Dans le cadre de ce projet, les indicateurs clés de performance sont calculés de manière ponctuelle lors de l’ajout d’un nouvel actif au portefeuille. Les valeurs obtenues, notamment le prix de la cryptomonnaie et la valeur associée à l’actif, sont ensuite stockées de manière persistante dans le **localStorage** du navigateur et ne sont pas recalculées automatiquement par la suite.

Ce choix de conception a été effectué afin de respecter les contraintes imposées par l’utilisation de la version gratuite de l’API CoinGecko, qui limite fortement la fréquence des requêtes autorisées. En limitant les appels à l’API aux actions explicites de l’utilisateur, l’application évite tout dépassement de quota tout en conservant des données cohérentes pour l’analyse des portefeuilles.

Bien que cette approche ne permette pas un suivi dynamique et en temps réel de l’évolution des valeurs, elle reste adaptée au périmètre du projet et à ses objectifs pédagogiques. Elle permet de démontrer la logique de calcul des indicateurs, la persistance des données et leur exploitation à travers des visualisations pertinentes, tout en garantissant la stabilité et la fiabilité de l’application.

Dans un contexte différent, notamment pour une application destinée à un usage réel, une mise à jour périodique ou en temps réel des indicateurs pourrait être envisagée, sous réserve de l’utilisation d’une infrastructure adaptée et d’un accès à des services de données moins contraints.

Design et Interface Utilisateur

Choix du Thème Sombre

L'application utilise un thème sombre pour plusieurs raisons pratiques :

- Modernité: Les interfaces sombres sont très populaires dans les applications contemporaines
- Confort visuel: Réduction de la fatigue oculaire, particulièrement lors d'utilisation prolongée
- Adéquation avec le domaine: Le thème sombre convient bien aux applications financières et techniques

Graphiques et Visualisations

L'utilisation de Chart.js permet de créer deux types de visualisations :

- Graphique circulaire: Représente la répartition proportionnelle de la valeur entre les différents portefeuilles
- Graphique en barres: Affiche la composition détaillée des cryptomonnaies dans chaque portefeuille

Ces visualisations rendent l'analyse des données plus intuitive et permettent une compréhension rapide de la répartition des investissements.

Fonctionnalités Techniques Avancées

Récupération Asynchrone des Données

L'application effectue des appels API asynchrones vers CoinGecko pour maintenir les prix à jour. Cette approche garantit que l'interface utilisateur reste réactive pendant les opérations de récupération de données.

Gestion de l'État de l'Application

La navigation entre les sections et la mise à jour de l'interface sont gérées entièrement en JavaScript. Le système d'affichage/masquage permet de créer une expérience utilisateur fluide sans rechargement de page.

Persistance Locale

L'utilisation du localStorage assure la persistance des données utilisateur entre les sessions. Les données sont automatiquement

sauvegardées lors de chaque modification et rechargées au démarrage de l'application.

Limites et perspectives d’évolution

Bien que l’application réponde aux objectifs définis dans le cadre du projet, certaines limites apparaissent lorsqu’on envisage une utilisation dans un contexte plus proche d’un environnement professionnel ou à grande échelle. Ces limites sont principalement liées aux contraintes techniques imposées par le cahier des charges et par le périmètre volontairement restreint du projet.

Tout d’abord, le choix d’utiliser exclusivement JavaScript vanilla, bien qu’il soit pertinent dans un cadre pédagogique, montre ses limites dès lors que l’application devient plus complexe. Dans un contexte réel, l’utilisation d’un framework moderne tel que React permettrait de bénéficier d’une meilleure gestion de l’état de l’application, d’une architecture plus modulaire et d’une maintenabilité accrue. React faciliterait également la réutilisation des composants et l’évolution future de l’interface utilisateur.

Ensuite, la persistance des données repose actuellement sur le localStorage du navigateur, ce qui constitue une solution simple mais limitée. Cette approche ne permet ni le partage des données entre plusieurs appareils, ni une gestion avancée des utilisateurs. Dans un projet réel, il serait préférable d’utiliser une base de données relationnelle telle que PostgreSQL, solution robuste et éprouvée, offrant une meilleure intégrité des données, des performances supérieures et une scalabilité adaptée à un usage multi-utilisateur.

Dans la continuité de cette évolution, l’application ne propose actuellement aucun système d’authentification. L’ajout d’un

mécanisme d’authentification et de gestion des utilisateurs serait indispensable dans un contexte professionnel afin de sécuriser l’accès aux portefeuilles et de permettre une personnalisation des données par utilisateur. Cette fonctionnalité ouvrirait également la voie à des rôles utilisateurs ou à des fonctionnalités avancées liées à la sécurité.

Enfin, l’utilisation de l’API CoinGecko constitue une autre limite du projet. L’application repose sur la version gratuite de l’API, qui impose des restrictions importantes en termes de fréquence d’appels. Ces limitations réduisent les possibilités d’actualisation des données et empêchent certaines fonctionnalités avancées, comme le suivi en temps réel à grande échelle ou l’analyse historique approfondie. Dans un projet destiné à un usage réel, il serait nécessaire d’envisager une offre payante ou une solution alternative afin de garantir des données fiables, fréquentes et exploitables de manière optimale.

En résumé, si les paramètres du projet avaient été différents, notamment en termes de contraintes techniques et d’objectifs fonctionnels, l’architecture globale aurait évolué vers une solution plus complète intégrant un framework front-end moderne, une base de données dédiée, un système d’authentification et une gestion plus avancée des données externes. Ces améliorations permettraient de transformer cette application pédagogique en un produit réellement exploitable dans un contexte professionnel.

Conclusion

Ce projet de portefeuille crypto démontre qu'il est possible de créer une application web complète et fonctionnelle en utilisant uniquement des technologies web natives. L'approche vanilla JavaScript permet une compréhension approfondie des mécanismes fondamentaux du développement web tout en créant une application pratique et moderne.

L'intégration avec l'API CoinGecko et l'utilisation du localStorage montrent comment combiner des services externes avec des solutions de stockage local pour créer une expérience utilisateur riche. Le choix du domaine des cryptomonnaies apporte une dimension moderne et pertinente au projet, en phase avec les évolutions technologiques actuelles.

Le projet atteint ses objectifs en proposant une interface intuitive pour la gestion de portefeuilles crypto, tout en démontrant les capacités du développement front-end sans frameworks complexes.