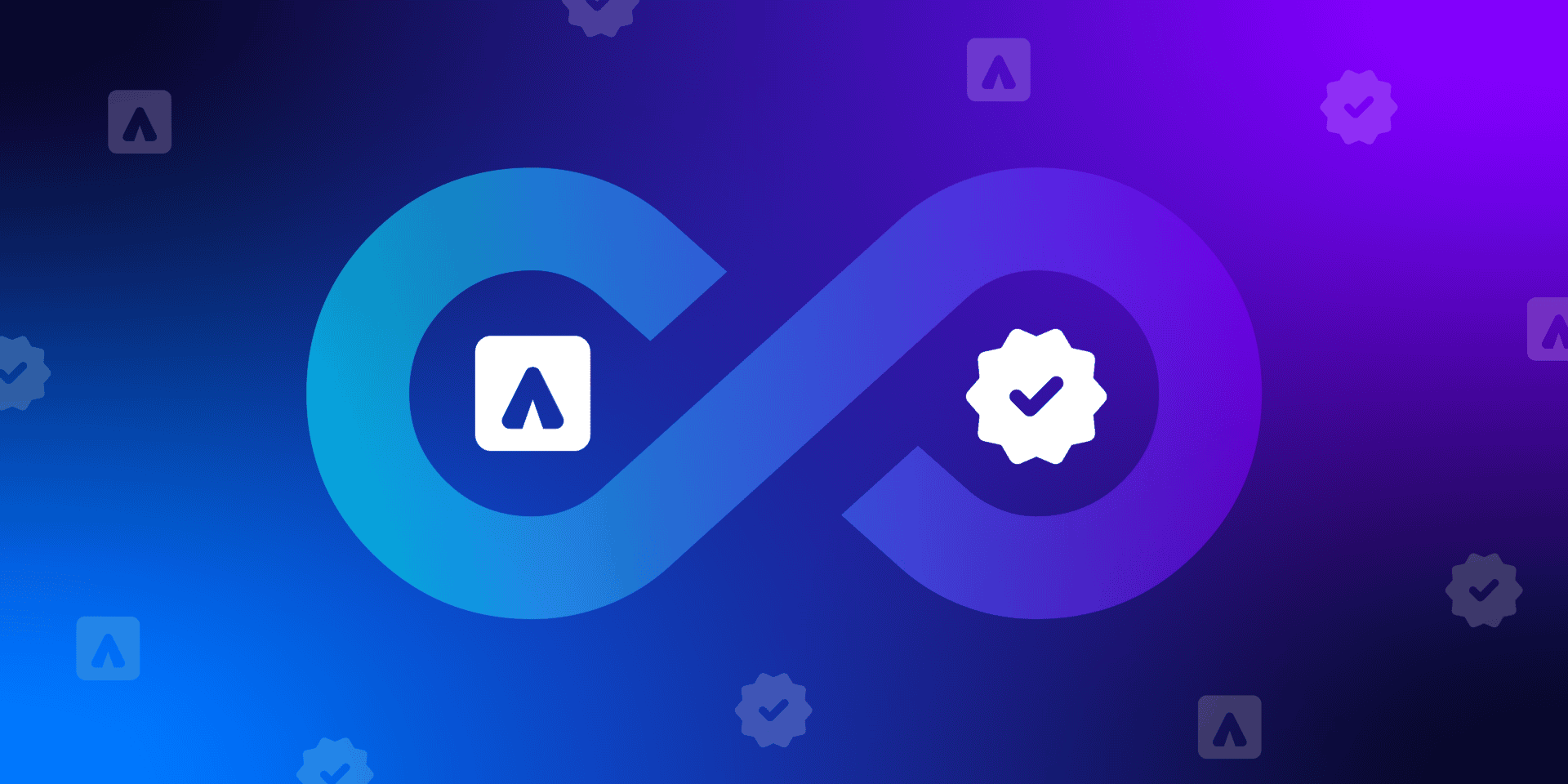
**PORTFOLIO**



William Trelles – Classe CID 3A

ETML-Lausanne-Vennes

Professeur : Mathieu Meylan

Table de matière

[1 Introduction 3](#_Toc187401906)

[2 Planification initiale 3](#_Toc187401907)

[3 Actions nécessaires 3](#_Toc187401908)

[4 Etapes du workflow 3](#_Toc187401909)

[4.1 Étapes de validation 4](#_Toc187401910)

[4.1.1 1.1 Backend-check 4](#_Toc187401911)

[4.2 Frontend-check 4](#_Toc187401912)

[4.3 Étapes de test 4](#_Toc187401913)

[4.3.1 Tests End-to-End (E2E) 4](#_Toc187401914)

[4.4 Étapes de construction 5](#_Toc187401915)

[4.4.1 Frontend-build 5](#_Toc187401916)

[4.4.2 Artefacts 5](#_Toc187401917)

[4.5 Étapes de déploiement 5](#_Toc187401918)

[4.5.1 Configuration et déploiement 5](#_Toc187401919)

[5 Diagramme de flux 5](#_Toc187401920)

[6 Préparation du serveur 6](#_Toc187401921)

[6.1 Installation et configuration de PM2 6](#_Toc187401922)

[6.2 Utilisation du port 80 avec authbind 6](#_Toc187401923)

[6.3 Configuration d’un alias pour authbind avec PM2 6](#_Toc187401924)

[7 Configuration SSH pour les VM 6](#_Toc187401925)

[8 Installation de NGINX pour servir le frontend 7](#_Toc187401926)

[8.1 Installation de nginx 7](#_Toc187401927)

[8.2 Créer le fichier où sera le config du serveur de l’application. 7](#_Toc187401928)

[8.3 Activer le fichier de configuration. 8](#_Toc187401929)

[8.4 Vérifier les erreurs de syntaxe : 8](#_Toc187401930)

[8.5 Recharger la configuration : 8](#_Toc187401931)

[9 Rapport de déploiement 8](#_Toc187401932)

[9.1 Environnements de déploiement 8](#_Toc187401933)

[URL de production 8](#_Toc187401934)

[URL de test 8](#_Toc187401935)

[9.2 Etat actuel de l'application 9](#_Toc187401936)

[9.3 Conclusion 9](#_Toc187401937)

[10 Schéma CI/CD 9](#_Toc187401938)

[11 Usage de l’IA 10](#_Toc187401939)

# Introduction

Le projet "addon-todo-app" est une extension développée pour le navigateur Firefox permettant aux utilisateurs de gérer leurs tâches de manière efficace. Ce portfolio présente les étapes de conception, de développement et de déploiement de cette application avec une approche CI/CD.

# Analyse et planification initiale

L'objectif principal de cette extension était de permettre :

La gestion des tâches personnelles sous forme de liste.

Un déploiement automatisé pour simplifier la maintenance et les mises à jour.

Pour atteindre cet objectif, plusieurs étapes ont été définies :

Mise en place d’un environnement de développement compatible avec les outils CI/CD.

Configuration des workflows GitHub Actions pour automatiser les processus de build et de déploiement.

La planification est accessible à travers ce [lien](https://github.com/users/witrellesp/projects/2/views/1).

# Configuration de l’environnement de développement

L'environnement de développement a été configuré avec les outils suivants :

* Node.js (v20) : Utilisé pour la gestion des dépendances et des scripts.
* wxt : Un outil de construction pour les extensions de navigateur.
* GitHub Actions : Configuration des workflows CI/CD pour automatiser les étapes de build, test et déploiement.
* Secrets GitHub : Stockage des clés sensibles et variables nécessaires au déploiement sécurisé de l'extension.

Procédure de mise en place :

Installation de Node.js.

Initialisation du projet avec les dépendances via npm ci.

Configuration du fichier wxt.config.ts pour inclure les variables dynamiques comme ADD\_ON\_FIREFOX\_VERSION et FIREFOX\_ADDON\_ID.

Configuration des secrets et des variables globales dans GitHub.

# Développement et tests

# Planification / jobs / étapes CI/CD

**Étape 1 : Checkout**

Le workflow utilise actions/checkout@v3 pour cloner le dépôt sur la machine virtuelle. Cela garantit que tout le code source est disponible pour les étapes suivantes.

**Étape 2 : Configuration de Node.js**

La version 20 de Node.js est installée pour exécuter les commandes npm. C'est une version récente et stable compatible avec les outils utilisés.

**Étape 3 : Installation des dépendances**

La commande npm ci installe toutes les dépendances nécessaires depuis package-lock.json. Cette méthode garantit que les versions exactes des dépendances sont utilisées.

**Étape 4 : Export des variables**

Les variables ADD\_ON\_FIREFOX\_VERSION (version de l’add-on) et FIREFOX\_ADDON\_ID (UUID de l’add-on) sont exportées dans l'environnement. Ces valeurs sont utilisées par les étapes suivantes, notamment pour configurer le fichier manifest.json.

**Étape 5 : Audit des vulnérabilités**

npm audit fix détecte et corrige automatiquement les vulnérabilités de sécurité dans les dépendances.

**Étape 6 : Build**

La commande npm run build:firefox génère les fichiers nécessaires à l’add-on dans le répertoire de sortie (.output/firefox).

**Étape 7 : Compression**

La commande npm run zip:firefox compresse l’add-on en un fichier .zip prêt à être déployé. Le fichier compressé est généré dans .output/firefox/.

**Étape 8 : Déploiement**

La commande utilise l’action wdzeng/firefox-addon@v1 pour uploader le fichier .zip sur le Firefox Add-ons Developer Hub.

Les secrets FIREFOX\_JWT\_ISSUER et FIREFOX\_JWT\_SECRET sont utilisés pour s’authentifier auprès de l’API de déploiement.

# Résultats finaux et conclusion

L’extension a été compilée et déployée avec succès. Voici les étapes clés obtenues :

Build et test locaux : L’extension a été testée localement avec npm run build:firefox et npm run zip:firefox.

Déploiement automatique : Le workflow CI/CD a permis d’automatiser le déploiement vers Firefox Add-ons avec un fichier .zip correct tous les fichiers nécessaires.

