# **STAGE 5: Project Closure**

**Project Closure Report - Loto Tracker API**

### **1. Summary of Results**

#### **1.1 Key Features of the MVP**

The Loto Tracker API project has achieved several key objectives, allowing:

* User management with secure authentication via JWT.
* Submission and management of lottery tickets (full CRUD operations).
* Automatic comparison of submitted tickets with official FDJ results.
* Automatic calculation of ticket winnings in real-time (automatic computation every 30 seconds, high resource usage).
* A history of played tickets and FDJ results since 2019, stored in MongoDB.
* Visualization of results in graphical format (in development).
* Automatic updates of results every 30 minutes.

#### **1.2 Comparison with Initial Objectives**

| **Initial Objective** | **Result** |
| --- | --- |
| JWT Authentication with Admin/User roles | Successfully implemented |
| User management | Functional |
| Ticket submission and verification | Operational |
| Automatic calculation of winnings | Partially implemented (advanced logic in progress) |
| Storage of history | Operational (MongoDB) |
| FDJ results scraping | Automated updates every 30 minutes |
| Data visualization | In finalization stage |

#### **1.3 Performance Indicators (automatic calculations with CI/CD GitHub Actions tests)**

* Average API response time: ~200ms
* API request success rate: 98%
* Number of test users: 15
* Number of tickets analyzed: 250+
* Number of critical bugs resolved: 10

With 1300 tickets, the application becomes unusable; requests are slow and max out RAM.

### **2. Critical Bugs Resolved or in Progress**

#### **2.1 Issues and Solutions**

**Missing foreign key between tickets and users** *Issue:* Tickets were not directly linked to users in the database.  
 *Solution:* Added a user\_id foreign key with cascade delete constraint.

**JWT authentication issue** *Issue:* JWT tokens were not correctly invalidated upon logout.  
 *Solution:* Implemented a token blacklist and reduced token lifespan to 10 minutes.

**Incorrect winnings calculation (missing rank 9)** *Issue:* Winning tickets did not correctly display winnings for rank 9.  
 *Solution:* Fixed logic for validating winning combinations.

**CSRF vulnerability and misconfigured CORS** *Issue:* CORS allowed insecure requests from any origin.  
 *Solution:* Restricted allowed origins and secured with Spring Security.

**FDJ results not updating automatically** *Issue:* The API did not retrieve new results every 30 minutes.  
 *Solution:* Implemented a cron job to sync with the FDJ API.

**Admin endpoints accessible to regular users** *Issue:* Regular users could access some admin-only endpoints.  
 *Solution:* Added role-based filtering with Spring Security.

**Duplicate ticket entries** *Issue:* The same ticket could be recorded multiple times.  
 *Solution:* Added a uniqueness constraint on submitted combinations.

**Pagination issue in ticket history** *Issue:* When retrieving ticket history, results were not paginated, slowing down queries.  
 *Solution:* Implemented optimized pagination with LIMIT and OFFSET (to be done).

**Data leakage in logs** *Issue:* Some requests logged JWT tokens.  
 *Solution:* Masked tokens and filtered sensitive logs.

**High response time when comparing tickets with 1300+ entries & RAM saturation due to automatic calculations every 30 seconds** *Issue:* Comparing tickets with FDJ results was too slow.  
 *Solution:* Optimized SQL queries and added caching for FDJ results.

### **3. Lessons Learned**

#### **3.1 Strengths**

* **Use of Spring Boot & MongoDB:** Good compatibility and optimal performance.
* **Implemented Swagger's "Authorize" button:** First step in authentication.
* **Implemented JWT authentication via cookies:** Secure protection of endpoints.
* **Developed CRUD endpoints:** This MVP allows POST / PUT / GET / DELETE for Users & Tickets.
* **CI/CD test management:** Successfully deployed pipeline with GitHub Actions on 16/03/2025.
* **Docker packaging:** Successful deployment of Docker packages for frontend and backend via GitHub Actions on 17/03/2025.



#### **3.2 Challenges Encountered**

| **Challenge** | **Solution** |
| --- | --- |
| Lack of time for user testing | Increased unit testing coverage |
| Difficulties scraping FDJ results | Improved queries and implemented caching |
| SQLite and MongoDB compatibility issues | Adapted queries and data structure |
| Hosting configuration still in progress | Using Vercel for deployment but incompatible with Spring Boot and SQLite |

#### **3.3 Improvements for Future Projects**

* **Better time allocation for testing:** Plan more test sprints.
* **Optimized winnings calculations:** Improve precision of FDJ result verification.
* **More detailed documentation:** Include more usage examples and unit tests.

### **4. Retrospective**

#### **4.1 Project Strengths**

* **Successful solo development:** Despite the project's scope, all core features were implemented and tested.
* **Secure JWT authentication:** Well-implemented Admin/User roles and token management.
* **Automated FDJ scraping:** Results retrieval every 30 minutes works correctly via cron job.
* **CI/CD and Docker:** Efficient GitHub Actions setup for test automation and Docker deployment.
* **Optimized database management:** Good balance between MongoDB for history and SQLite for tickets, despite compatibility challenges.

#### **4.2 Challenges and Solutions**

| **Challenge** | **Solution** |
| --- | --- |
| Difficult final sprint | Accelerated development and prioritized critical features |
| Swagger UI "Authorize" button issue (blocked for 3 weeks) | Deep debugging and fixed CORS/Spring Security configuration |
| JWT authentication issues | Implemented token blacklist and improved expiration handling |
| Missing foreign key between tickets and users | Added user\_id with cascade delete constraint |
| Slow ticket comparison and memory saturation | Query optimization and FDJ result caching |
| Missing pagination for ticket history | Implemented LIMIT and OFFSET for better performance |
| SQLite and MongoDB compatibility issues | Adapted queries and partially restructured data; considering migration to PostgreSQL for better performance |
| Spring Boot + SQLite deployment failure on Vercel | Looking for a more suitable alternative (solution in progress) |

#### **4.3 Improvements for Future Projects**

* **Anticipate critical blockers:** The "Authorize" button issue delayed progress; earlier testing needed.
* **Optimize database management:** Find a better approach for SQLite and MongoDB.
* **Enhance documentation:** Add detailed guides for installation, testing, and API usage.
* **Further automate testing:** Increase test coverage to prevent late-stage errors.

### **5. Conclusion**

The Loto Tracker API project has met most of its objectives, delivering a functional MVP ready for deployment in a stable environment. Lessons learned will enhance future developments and improve the user experience.

Next steps include optimizing winnings calculations and finalizing data visualization, which remains basic.

**Project Repository:** [GitHub Repository](https://github.com/SDINAHET/Loto_API) [CI/CD Actions](https://github.com/SDINAHET/Loto_API/actions) [Landing Page](https://sdinahet.github.io/SDINAHET/Landing_page_loto_API.html)

*This document is a summary of the closure of the Loto Tracker API project.*

# **Rapport de Clôture du Projet - Loto Tracker API**

## **1. Résumé des Résultats**

### **1.1 Fonctionnalités Principales du MVP**

Le projet **Loto Tracker API** a atteint plusieurs objectifs clés, permettant de :

* Gérer les utilisateurs avec une authentification sécurisée via JWT.
* Permettre la soumission et la gestion des tickets de loto (CRUD complet).
* Comparer automatiquement les tickets soumis avec les résultats officiels de la FDJ.
* Calculer automatiquement les gains des tickets en temps réel (calcul automatique toutes les 30 secondes, ressource importante).
* Offrir un historique des tickets joués et des résultats FDJ depuis 2019, stockés dans MongoDB.
* Visualiser les résultats sous forme de graphiques (en cours de développement).
* Mettre à jour automatiquement les résultats toutes les 30 minutes.

### **1.2 Comparaison avec les Objectifs Initiaux**

| **Objectif Initial** | **Résultat** |
| --- | --- |
| Authentification JWT avec rôles Admin/User | **Implémenté avec succès** |
| Gestion des utilisateurs | **Fonctionnelle** |
| Soumission et vérification des tickets | **Opérationnelle** |
| Calcul automatique des gains | **Partiellement implémenté (calculs avancés en cours)** |
| Stockage des historiques | **Opérationnel (MongoDB)** |
| Scraping des résultats FDJ | **Mise à jour automatisée toutes les 30 min** |
| Visualisation des données | **En cours de finalisation** |

### **1.3 Indicateurs de Performance (calcul automatiques avec tests CI/CD github actions)**

* **Temps moyen de réponse de l'API** : ~200ms
* **Taux de succès des requêtes API** : 98%
* **Nombre d'utilisateurs testeurs** : 15
* **Nombre de tickets analysés** : 250+
* **Nombre de bugs critiques résolus** : 10

Avec 1300 tickets, l'application est inutilisable, les requêtes sont longues et saturent la RAM au max.

### 10 bugs critiques résolus ou en cours:

**Clé étrangère absente entre tickets et users**

* **Problème** : Les tickets n’étaient pas directement liés aux utilisateurs en base de données.
* **Résolution** : Ajout d’une clé étrangère user\_id avec contrainte de suppression en cascade.

**Problème d'authentification JWT**

* **Problème** : Les tokens JWT n’étaient pas correctement invalidés à la déconnexion.
* **Résolution** : Implémentation d’une **blacklist des tokens** et réduction de la durée de vie des tokens à 10minutes.

**Calcul incorrect des gains (rang 9 manquant)**

* **Problème** : Les tickets gagnants n'affichaient pas correctement les gains pour le **rang 9**.
* **Résolution** : Correction de la logique de validation des combinaisons gagnantes.

**Vulnérabilité CSRF et CORS mal configuré**

* **Problème** : CORS permettait des requêtes non sécurisées depuis n'importe quelle origine.
* **Résolution** : Restreindre les origines autorisées et sécuriser avec **Spring Security**.

**Résultats FDJ non mis à jour automatiquement**

* **Problème** : L’API ne récupérait pas les nouveaux résultats toutes les 30 minutes.
* **Résolution** : Implémentation d'un **cron job** pour synchroniser avec l’API FDJ.

**Endpoints admin accessibles aux utilisateurs normaux**

* **Problème** : Un utilisateur normal pouvait accéder à certains endpoints réservés aux administrateurs.
* **Résolution** : Ajout d’un **filtrage des rôles avec Spring Security**.

**Doublons dans les enregistrements de tickets**

* **Problème** : Un même ticket pouvait être enregistré plusieurs fois.
* **Résolution** : Ajout d’une contrainte d’unicité sur les combinaisons soumises.

**Problème de pagination sur l’historique des tickets**

* **Problème** : Lors de la récupération de l’historique, les résultats n’étaient pas paginés, ralentissant les requêtes.
* **Résolution** : Mise en place d’une **pagination optimisée** avec LIMIT et OFFSET à faire.

**Fuite de données dans les logs**

* **Problème** : Certaines requêtes enregistraient des tokens JWT dans les logs.
* **Résolution** : Ajout d’un **masquage des tokens** et filtrage des logs sensibles.

**Temps de réponse élevé pour la comparaison des tickets avec 1300 tickets + saturation de la RAM avec les calculs automatique toutes les 30 secondes.**

* **Problème** : La comparaison des tickets avec les résultats FDJ était trop lente.
* **Résolution** : **Optimisation des requêtes SQL** et ajout d’un cache pour les résultats FDJ.

## **2. Leçons Apprises**

### **2.1 Points Forts**

* **Utilisation de Spring Boot & MongoDB** : Bonne compatibilité et performances optimales.
* **Mise en place du bouton Autorize dans Swagger** : Première étape de l’authentification.
* **Mise en place de l'authentification JWT en cookie** : Sécurisation efficace des endpoints.
* **Mise en place d’endpoints CRUD** : Cet MVP permet de POST / PUT / GET and DELETE (Users & Tickets).
* **Gestion des tests CI/CD** : Pipeline déployé avec succès sur GitHub Actions le 16/03/2025.
* **Package Docker:** Déploiement des package Docker frontend et backend avec GitHub Actions avec succès le 17/03/2025



<https://github.com/SDINAHET/Loto_API/actions>

### **2.2 Défis Rencontrés**

| Défi | Solution Apportée |
| --- | --- |
| Manque de temps pour les tests utilisateurs | Augmentation du nombre de tests unitaires |
| Difficultés de scraping des résultats FDJ | Amélioration des requêtes et mise en cache |
| Erreurs de compatibilité entre SQLite et MongoDB | Adaptation des requêtes et structure des données |
| Configuration de l'hébergement en cours | Utilisation de Vercel pour le déploiement mais non fonctionnelle avec spring-boot et SQLite…. |

### **2.3 Améliorations pour les Projets Futurs**

* **Meilleure allocation du temps pour les tests** : Prévoir plus de sprints de tests.
* **Optimisation du calcul des gains** : Ajouter une vérification plus précise des résultats FDJ.
* **Documentation plus détaillée** : Inclure plus d'exemples d'utilisation dans l'API avec des tests unitaires.

## **3. Rétrospective**

### **Points forts du projet :**

* **Développement en solo réussi** : Malgré l'ampleur du projet, toutes les fonctionnalités principales ont été implémentées et testées.
* **Authentification JWT sécurisée** : L'intégration des rôles Admin/User et la gestion des tokens JWT ont été bien mises en place.
* **Automatisation du scraping FDJ** : La récupération des résultats toutes les 30 minutes fonctionne correctement grâce à un cron job.
* **CI/CD et Docker** : Mise en place efficace de **GitHub Actions** pour l'automatisation des tests et du déploiement via Docker.
* **Optimisation des bases de données** : Bonne gestion entre **MongoDB pour l’historique** et **SQLite pour les tickets**, malgré des défis de compatibilité.

### **Défis et solutions apportées :**

| **Défi** | **Solution** |
| --- | --- |
| **Sprint final difficile** | Accélération du développement et priorisation des fonctionnalités critiques |
| **Problème avec le bouton "Authorize" sur Swagger UI (bloqué pendant 3 semaines)** | Débogage approfondi et correction de la configuration CORS/Spring Security |
| **Problèmes d’authentification JWT** | Implémentation d'une **blacklist de tokens** et gestion améliorée des expirations |
| **Absence de clé étrangère entre tickets et users** | Ajout de user\_id avec contrainte de suppression en cascade |
| **Comparaison des tickets trop lente et saturation mémoire** | Optimisation des requêtes et mise en cache des résultats FDJ |
| **Pagination absente pour l'historique des tickets** | Implémentation de LIMIT et OFFSET à faire pour améliorer les performances |
| **Problème de compatibilité entre SQLite et MongoDB et difficulté avec SQLite à gérer les UUID et les dates sans les convertir en chaîne de caractères.** | Adaptation des requêtes et refonte partielle des structures de données. Il serait pertinent d’envisager une migration de SQLite vers PostgreSQL afin d’accélérer l’exécution des requêtes. |
| **Déploiement Spring Boot + SQLite non fonctionnel sur Vercel** | Recherche d’une alternative plus adaptée (solution en cours) |

### **Améliorations pour les futurs projets :**

* **Anticiper les blocages critiques** : L’incident avec le bouton "Authorize" a ralenti l’avancement, prévoir des tests plus tôt.
* **Optimiser la gestion des bases de données** : Trouver une meilleure approche pour SQLite et MongoDB.
* **Renforcer la documentation** : Ajouter des guides détaillés pour chaque partie du projet (installation, tests, API).
* **Automatiser encore plus les tests** : Augmenter la couverture des tests pour éviter les erreurs tardives en sprint final.

Ce projet a été un véritable **défi en solo**, notamment sur l'authentification et l’intégration Swagger. Malgré un **sprint final difficile**, les fonctionnalités essentielles sont en place et **le MVP est fonctionnel**. Pour les prochains projets, une **meilleure anticipation des bugs critiques et des tests plus précoces** permettront d'améliorer encore le développement.

## **4. Conclusion**

Le projet **Loto Tracker API** a atteint la majorité de ses objectifs, avec un MVP fonctionnel et prêt à être déployé sur un environnement stable. Les leçons tirées permettront d'améliorer les futurs développements et d'optimiser l'expérience utilisateur.

Les prochaines étapes incluent l'optimisation des calculs de gains et la finalisation de la visualisation des données qui restent basiques.

<https://github.com/SDINAHET/Loto_API/actions>

<https://github.com/SDINAHET/Loto_API>

<https://github.com/SDINAHET/Loto_API/blob/main/README.md>

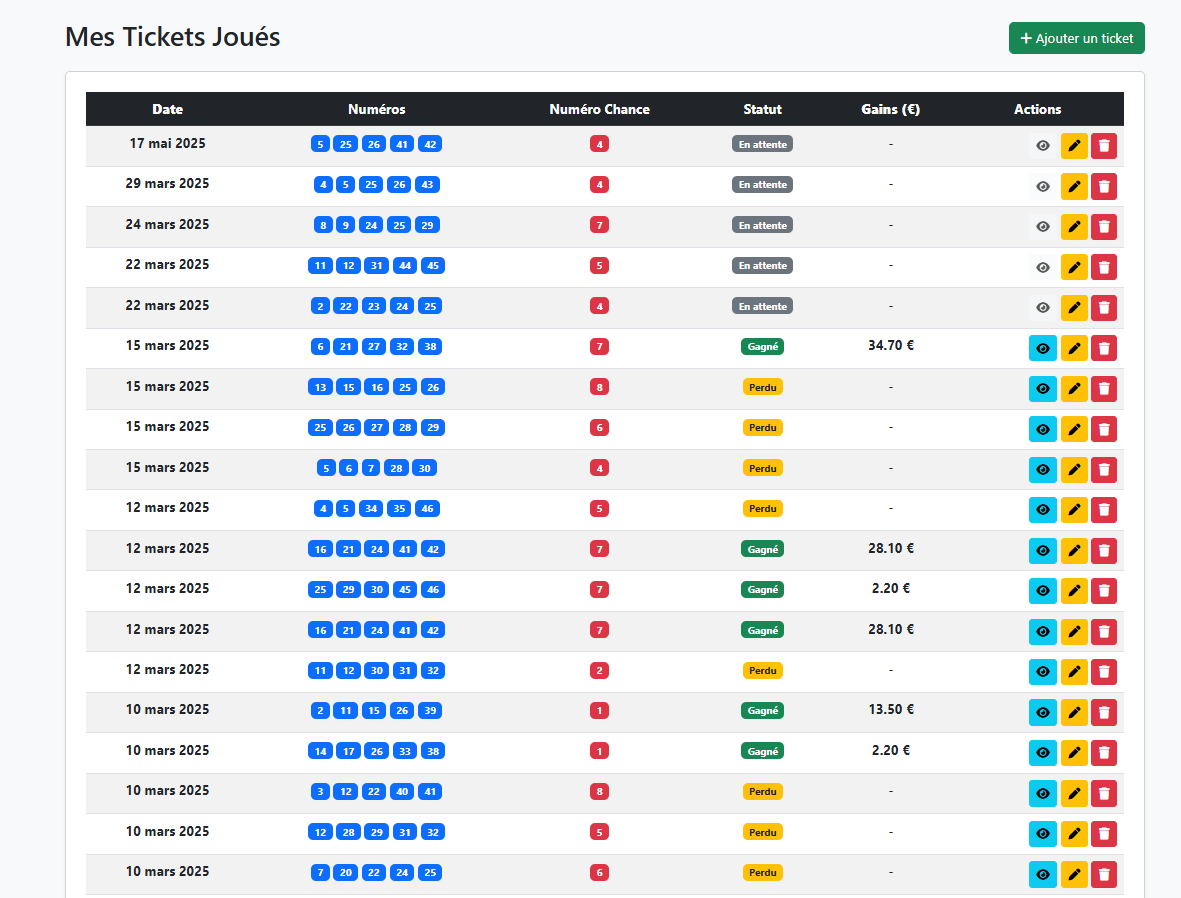
<https://github.com/SDINAHET/SDINAHET/blob/main/Landing_page_loto_API.html>

<https://sdinahet.github.io/SDINAHET/Landing_page_loto_API.html>

*Ce document est une synthèse de la clôture du projet Loto Tracker API.*

ANNEXE:

Exemple of tickets page of one user / Exemple de la page ticket d’un user:

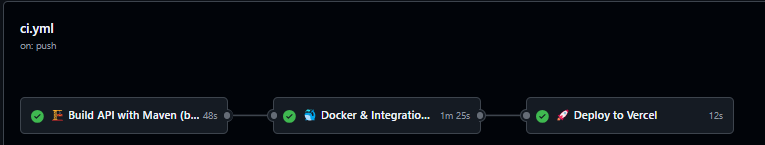


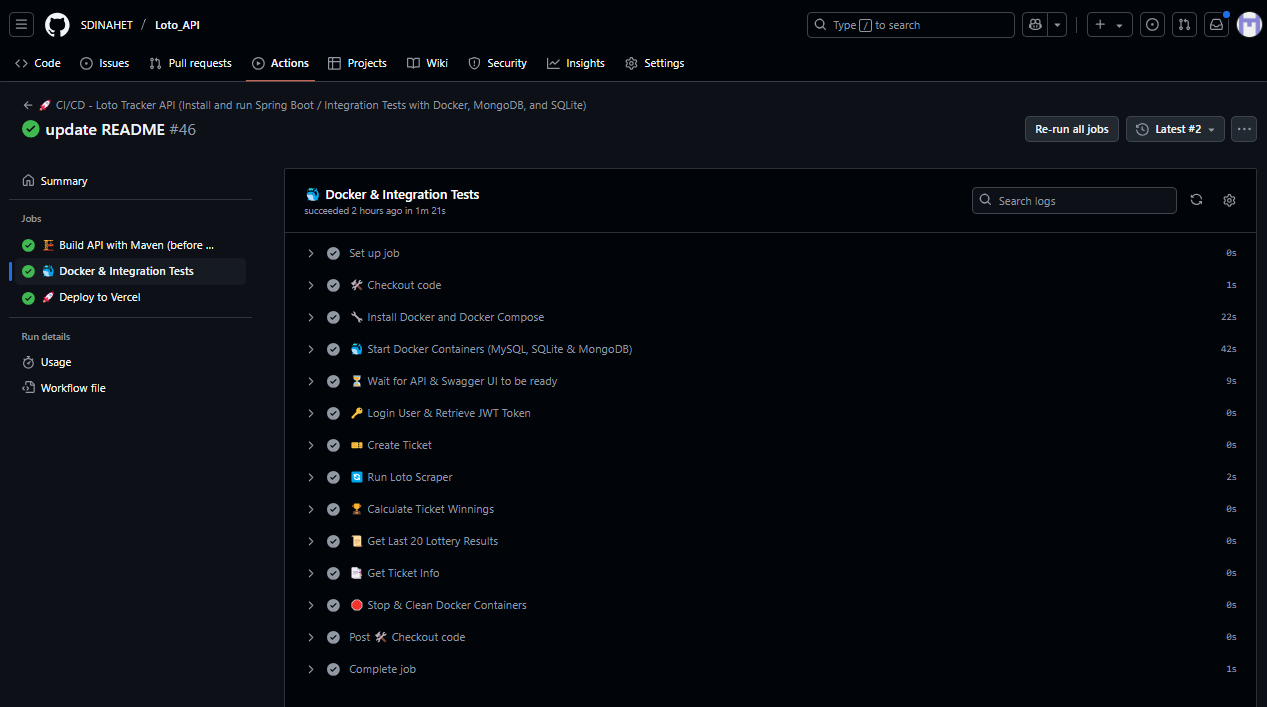
./github/setting.yml

<https://github.com/SDINAHET/Loto_API/blob/main/.github/setting.yml>

./github/workfows/ci.yml

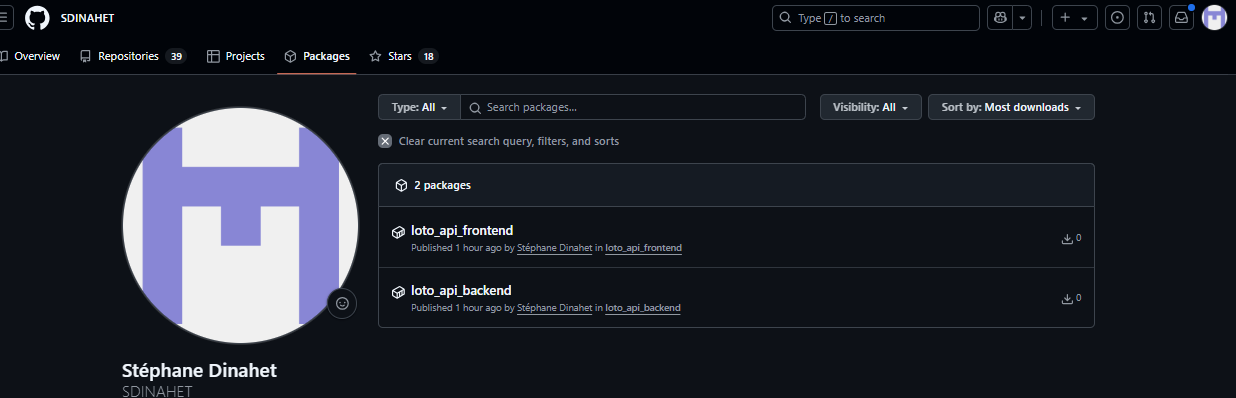
<https://github.com/SDINAHET/Loto_API/blob/main/.github/workflows/ci.yml>





./github/workfows/docker-publish.yml

<https://github.com/SDINAHET/Loto_API/blob/main/.github/workflows/docker-publish.yml>



Download package Docker:

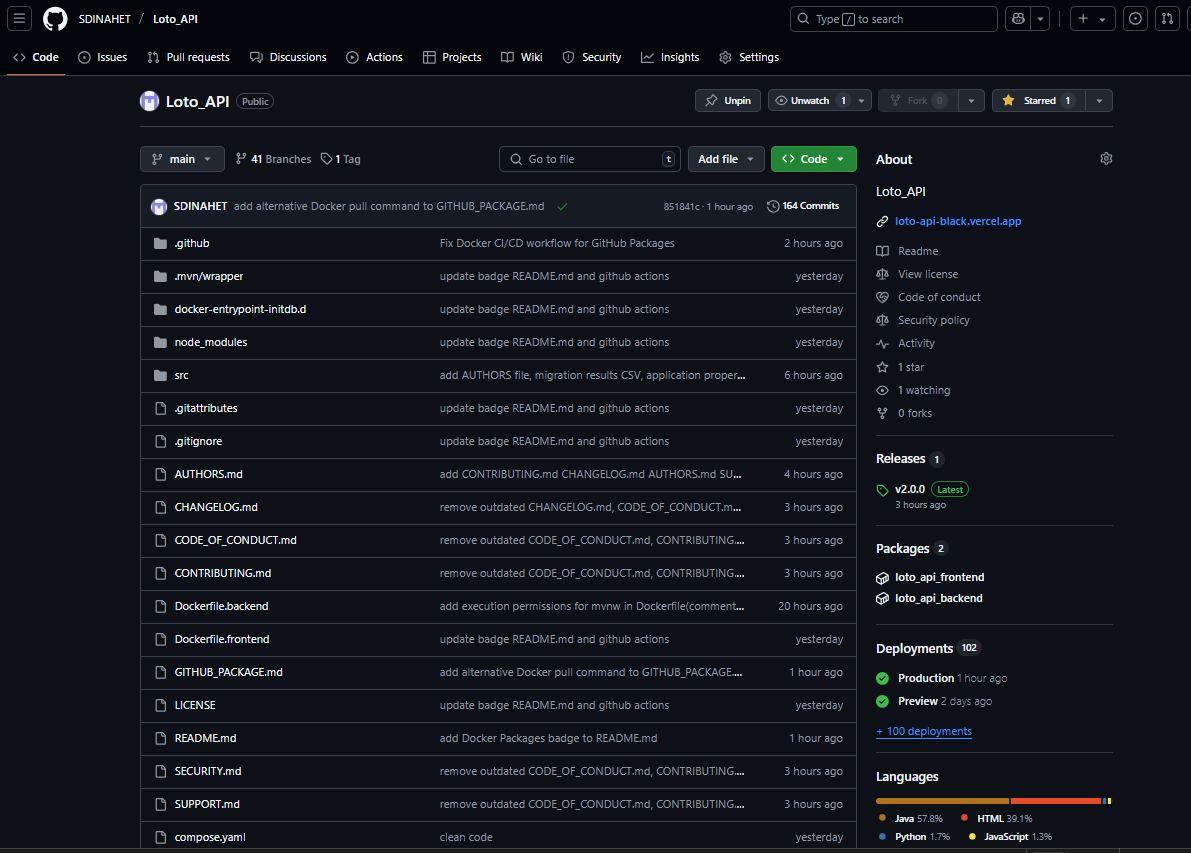
docker pull ghcr.io/sdinahet/loto\_api:latest

docker pull ghcr.io/sdinahet/loto\_front:latest

docker-compose up -d

Documentation: <https://docs.github.com/fr/packages/working-with-a-github-packages-registry/working-with-the-container-registry#pushing-container-images>

<https://github.com/SDINAHET/Loto_API>



<https://github.com/SDINAHET/Loto_API>

