|  |
| --- |
| ENSPY |
| Authentification Handshake |
| Rapport de conception de la QRAPI |

|  |
| --- |
| Nghogué Taptué Franck Roddier  05/07/2025  Sous la supervision de Pr. DJOTIO Thomas |

Table des matières

[**INTRODUCTION :** 2](#_Toc203176115)

[**OBJECTIFS :** 3](#_Toc203176116)

[**ARCHITECTURE GENERALE :** 4](#_Toc203176117)

[**3.1 Architecture REST Spring Boot :** 4](#_Toc203176118)

[**MODELE DE DONNEES :** 5](#_Toc203176119)

[**4.1 Tables principales :** 5](#_Toc203176120)

[**DIAGRAMMES UML :** 6](#_Toc203176121)

[**5.1 Diagramme de Classes :** 6](#_Toc203176122)

[**5.2 Diagramme de package** 7](#_Toc203176123)

[**5.2 Diagramme de Cas d’Utilisation :** 8](#_Toc203176124)

[**5.2 Diagrammes de Séquence :** 9](#_Toc203176125)

[**5.3 Diagramme d’état-transition :** 11](#_Toc203176126)

[**5.3 Diagrammes d’activités :** 12](#_Toc203176127)

[**5.5 Diagramme de Déploiement :** 14](#_Toc203176128)

[**JUSTIFICATION DES CHOIX TECHNIQUES :** 15](#_Toc203176129)

[**SECURITE ET CONFORMITE :** 16](#_Toc203176130)

[**PLAN DE TEST :** 17](#_Toc203176131)

[**CONCLUSION :** 18](#_Toc203176132)

# **INTRODUCTION :**

Ce rapport présente la conception détaillée de l’API *QRAPI*, qui permet de générer et valider des QR codes pour la gestion sécurisée de courses. Il expose l’architecture logicielle, les choix techniques, les diagrammes UML associés, et les justifications de conception.

# **OBJECTIFS :**

* Générer des QR codes contenant des données de course signées.
* Scanner et vérifier l’authenticité de ces QR codes.
* Garantir la sécurité via JWT et le suivi via un historique horodaté.
* Concevoir une architecture modulaire, maintenable et sécurisée.

# **ARCHITECTURE GENERALE :**

## **3.1 Architecture REST Spring Boot :**

L’architecture repose sur les couches suivantes :

* **Contrôleur REST** (QRCodeController)
* **Service métier** (ScanService)
* **Utilitaire JWT** (JwtUtil)
* **Repositories Spring Data** (ScyllaDB/Cassandra)
* **Sécurité** : JWT Filter & CORS

# **MODELE DE DONNEES :**

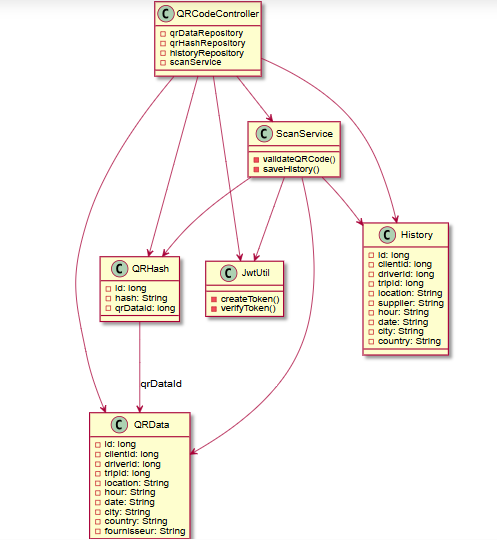
## **4.1 Tables principales :**

* **QRData** : Informations encodées dans le QR code.
* **QRHash** : Hash SHA-256 signé de QRData.
* **History** : Historique de scans.

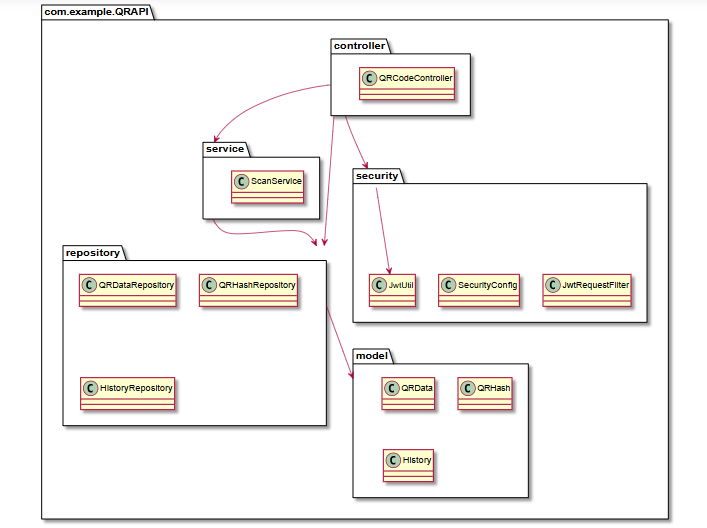
Voir le diagramme de classes ci-dessous.

# **DIAGRAMMES UML :**

## **5.1 Diagramme de Classes :**

**

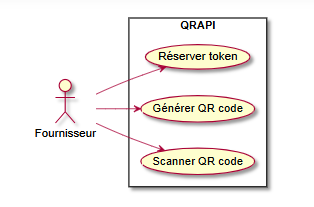
## **5.2 Diagramme de package**



## **5.2 Diagramme de Cas d’Utilisation :**

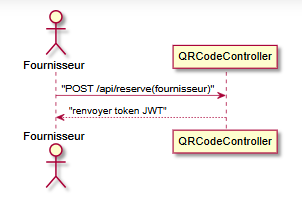
**Acteur principal** : Fournisseur  
**Cas** :

* Réserver token
* Générer un QR
* Scanner un QR

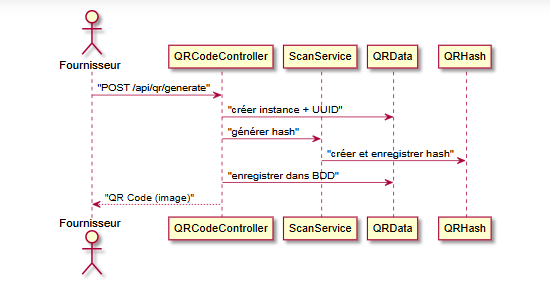


## **5.2 Diagrammes de Séquence :**

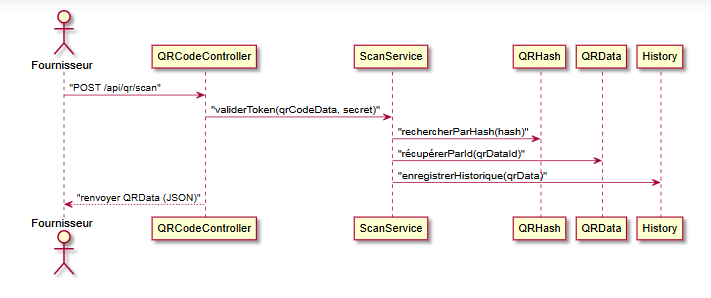
**Acteurs** : Fournisseur → API  
**Description** : Interaction entre l’utilisateur et le contrôleur pour générer un QR.



Réserver token

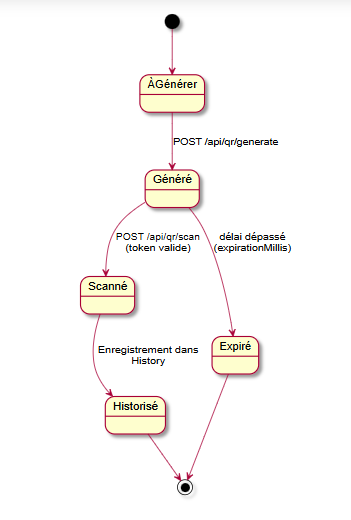
**

Générer QR code



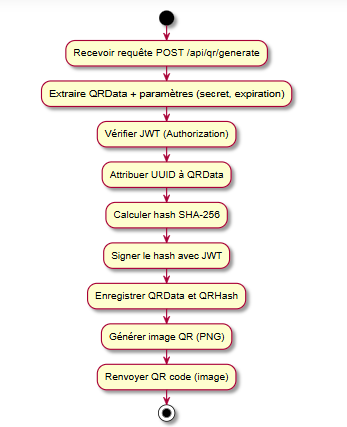
Scanner QR code

## **5.3 Diagramme d’état-transition :**

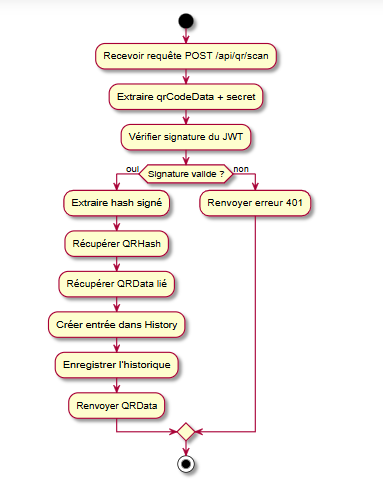


Cycle de vie d’un QR Code au sein de l’API

## **5.3 Diagrammes d’activités :**



Générer QR Code

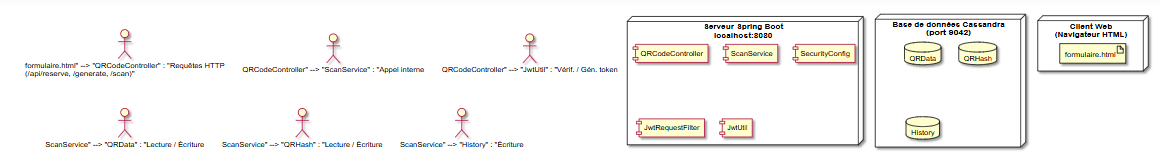


Scanner QR Code

## **5.5 Diagramme de Déploiement :**

**Composants** :

* Application Spring Boot (QRAPI)
* Base ScyllaDB/Cassandra
* Navigateur utilisateur



# **JUSTIFICATION DES CHOIX TECHNIQUES :**

* **Spring Boot** : Rapidité de développement et robustesse.
* **JWT + SHA-256** : Sécurité sans état et validation fiable.
* **ScyllaDB/Cassandra** : Scalabilité horizontale, forte résilience.
* **ZXing** : Génération rapide de QR en PNG.
* **Séparation des responsabilités** : Respect du principe SOLID.

# **SECURITE ET CONFORMITE :**

* CORS autorisé uniquement pour les origines nécessaires.
* Authentification obligatoire via /api/reserve avec token JWT.
* Tokens expirables (paramètre expirationMillis).
* Aucune donnée sensible exposée dans les QR codes (hash uniquement).

# **PLAN DE TEST :**

* Tests unitaires sur JwtUtil, ScanService
* Tests d’intégration entre contrôleur, service et base ScyllaDB/Cassandra
* Tests de charge pour évaluer la rapidité de génération
* Scénarios d’erreurs (QR invalide, token expiré, etc.)

# **CONCLUSION :**

Ce document détaille la conception d’une API fiable et sécurisée, conçue pour générer et valider des QR codes de façon performante. L'architecture adoptée assure modularité, sécurité, et extensibilité, adaptée à des contextes de production.