École nationale
de la statistique
et de l'analyse
de l'information

Projet informatique 2A - Groupe 32

QR Code Tracking

Dossier d'analyse

Étudiants :

Ahmed Beji
Lesline Méralda Kenne Yonta
Mayténa Labinsky
Adam Laporte
Louis Roux

Tuteur :

Thierry Mathe

27 septembre 2025

Table des matières

Introduction

Dans un monde où le numérique occupe une place de plus en plus importante, la rapidité et la simplicité d'accès à l'information sont devenues primordiales. Que ce soit pour partager un lien, consulter un menu de restaurant, accéder à un événement ou encore valider un billet de transport, les QR codes se sont imposés comme un outil incontournable de la vie quotidienne. Cela s'est encore plus accentué lors des dernières années, notamment en 2020, avec la pandémie mondiale, qui a accéléré la transition vers des solutions sans contact.

Cependant, si aujourd'hui, générer un QR code est assez simple pour que tout le monde puisse le faire, suivre les différentes utilisations reste encore plutôt limité. Pourtant, disposer de différentes données sur le nombre de scans, leurs localisations ou bien l'heure à laquelle le QR code est le plus scanné, peut être très utile, aussi bien pour les particuliers que pour une entreprise qui souhaite optimiser sa communication.

C'est en suivant cet objectif que ce projet nous a été proposé. En effet, coder une application, qui permettrait non seulement de générer des QR codes personnalisés, mais aussi de collecter et d'analyser toutes les données liées à leurs utilisations, contribuerait à répondre aux besoins de tout les utilisateurs, qu'ils soient professionnels ou non.

Ce dossier d'analyse s'articule en deux grandes parties : la première sera consacrée à la présentation du projet, aux choix fonctionnels ainsi qu'aux contraintes rencontrées, tandis que la seconde abordera les aspects techniques, incluant la conception de l'architecture, les modèles de données et les fonctionnalités implémentées.

1 Etude préalable

1.1 Cahier des charges

Le projet vise à développer une API Web capable de créer des QR codes et d'en suivre l'utilisation. Cette API permettra aux utilisateurs de générer des QR codes et de créer un compte utilisateur afin d'accéder aux fonctionnalités nécessitant une authentification. Les utilisateurs authentifiés pourront générer des QR codes suivis à partir d'une URL spécifique. Lorsqu'un QR code suivi est scanné, l'API enregistrera automatiquement l'appel et redirigera l'utilisateur vers le site associé, permettant ainsi de mesurer l'utilisation des QR codes.

Les utilisateurs disposant d'un compte pourront également consulter l'historique des QR codes suivis qu'ils ont créés, avec au minimum la date et l'heure de chaque utilisation. Cette fonctionnalité de suivi garantit une meilleure compréhension de l'engagement des utilisateurs et de l'efficacité des QR codes générés.

En complément des fonctionnalités de base, l'API pourra proposer des options avancées. L'authentification pourra être renforcée par l'utilisation de tokens. Les QR codes pourront également être personnalisés, par exemple en modifiant les couleurs ou en ajoutant un logo, afin d'améliorer l'esthétique et l'identité visuelle des codes générés.

1.2 Organisation du travail

Afin de structurer au mieux l'avancement de notre application, nous avons utilisé un diagramme de Gantt. Cet outil nous a permis de planifier les différentes étapes du projet, depuis l'analyse des besoins jusqu'à la soutenance finale.

Le projet a été découpé en trois grandes phases, en premier l'analyse et la modélisation. Cette étape sert surtout à définir les objectifs du projet, à réaliser quelques diagrammes nécessaires pour bien avancer (comme le diagramme de base de données, d'activités, de cas d'utilisation...). Elle inclut aussi la rédaction du rapport d'analyse. En résumé, cette étape a surtout servi à solidifier la base de notre projet afin de mieux pouvoir accomplir la seconde étape : le code. En effet, après que notre analyse soit validée, il faudra entamer la mise en place du code, que ce soit les classes métiers ou la DAO, mais aussi assurer une communication avec l'API et la base de données. Bien évidemment, il s'agira aussi de tester et de débugger pour que l'application soit la plus fiable possible. Enfin, on passera à la finalisation du projet. C'est une partie un peu moins technique que le reste. En effet, il faudra surtout rédiger le rapport final, produire un diaporama et répéter la soutenance.

Tout au long du projet, nous profiterons également de réunions avec le tuteur et nous nous organiserons pour faire des points d'équipe hebdomadaires pour suivre l'état d'avancement, ajuster la répartition des tâches et résoudre les difficultés rencontrées. Dans le diagramme, ces réunions, ainsi que les dates majeures (soutenance, rendus) ont été clairement identifiés pour avoir une meilleure vision d'ensemble des tâches à réaliser en priorité.

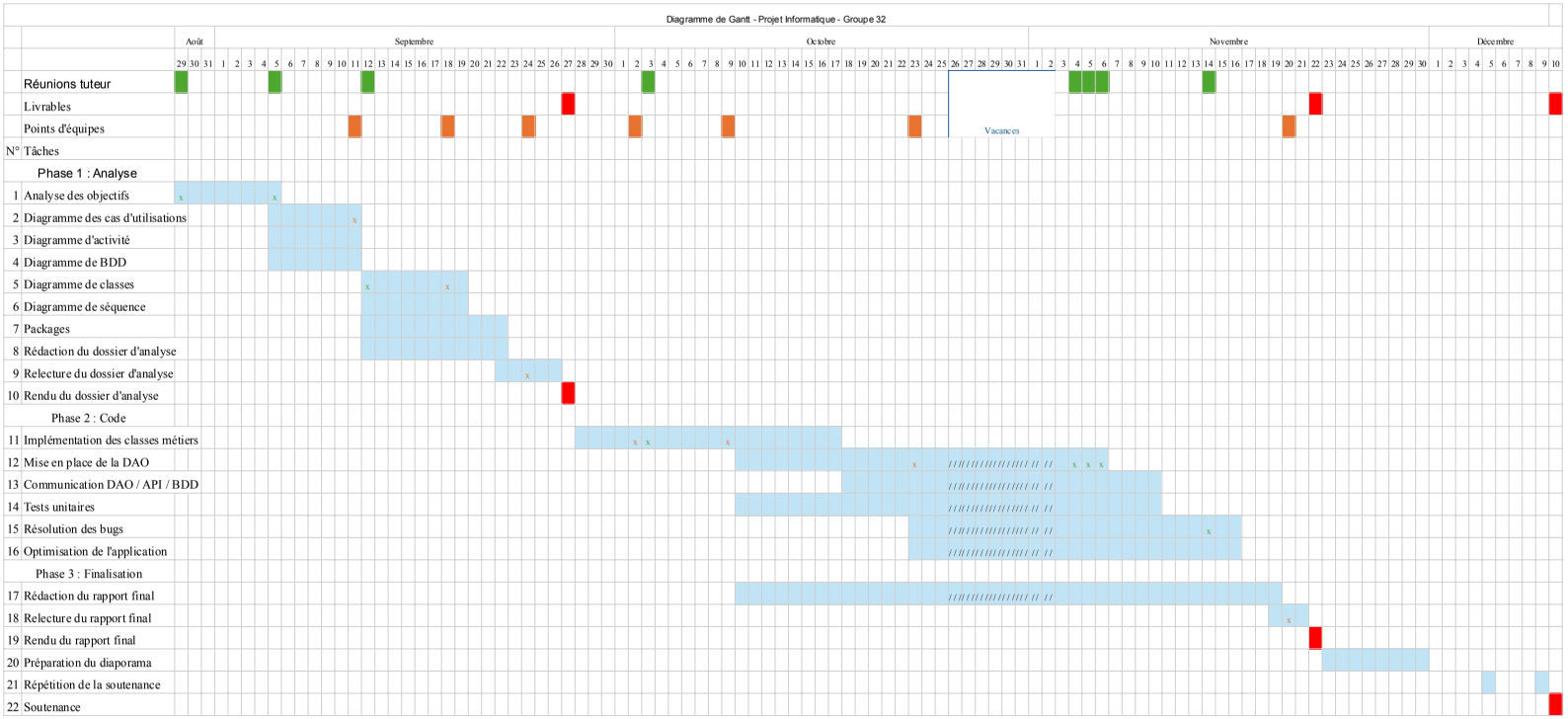

FIGURE 1 – Diagramme de Gantt du projet

2 Modèle de conception

2.1 Diagramme des cas d'utilisation

Pour notre application, le diagramme des cas d'utilisation se divise en deux parties distinctes. La première concerne l'utilisateur final, qui utilise l'application pour créer ou consulter les statistiques d'un QR code. La seconde concerne un utilisateur externe, qui se contente de scanner un QR code généré via l'application.

Étant donné que notre application est principalement orientée vers la simplicité d'usage, les fonctionnalités accessibles à un utilisateur standard restent limitées. Celui-ci peut créer un QR code classique ou un QR code suivi, avec la possibilité de le personnaliser en choisissant une couleur ou en y intégrant un logo. Il peut également consulter les statistiques liées à un QR code qu'il a préalablement créé. Cependant, l'accès aux fonctionnalités avancées, telles que la création d'un QR code suivi ou la consultation des statistiques, nécessite une authentification. Cette étape permet d'associer les QR codes créés à l'utilisateur afin d'en assurer le suivi et la gestion.

Quant à l'utilisateur externe, il n'interagit pas avec l'interface de l'application. Son unique action consiste à scanner le QR code. Dès lors, l'application met automatiquement à jour les statistiques associées à ce QR code et renvoie immédiatement l'utilisateur vers l'URL correspondante. Ainsi, à chaque scan, les statistiques sont actualisées en temps réel tout en assurant une redirection fluide vers le contenu prévu.

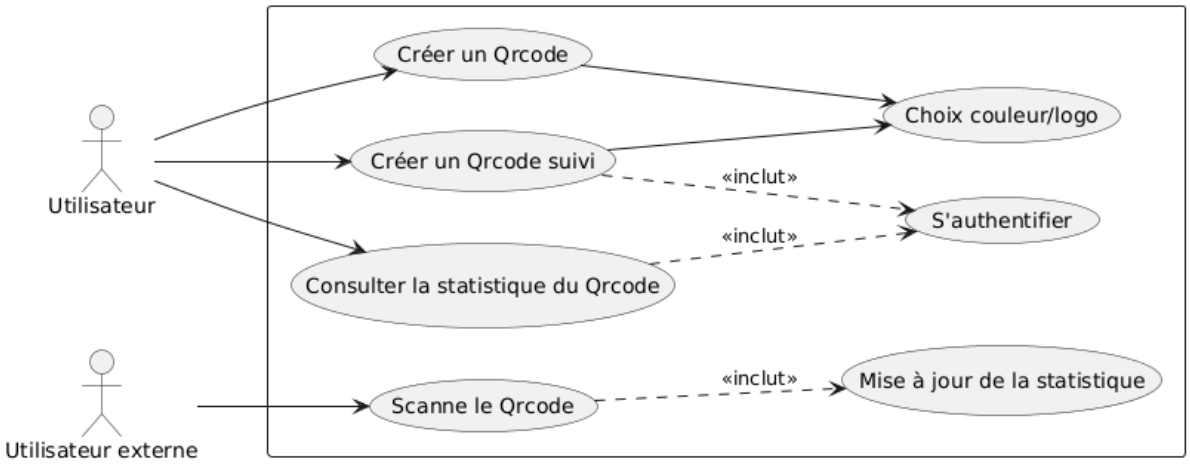

FIGURE 2 – Diagramme des cas d'utilisation

2.2 Diagramme de classe

Dans le cadre de ce projet, le diagramme de classe est organisé en quatre parties comprenant chacune une classe service et une classe DAO.

2.2.1 Utilistateur

En premier temps, la classe **Utilisateur** contient les attributs privés `id_user` qui est une clé primaire et correspond à l'identifiant ainsi que `mdp` qui correspond au mot de passe.

Ensuite, la classe **UtilisateurService** comprend les méthodes suivantes, qui prennent chacune en paramètre l'identifiant et le mot de passe de l'utilisateur :

- *creer_user()* qui permet de créer un nouvel utilisateur ;
- *se_connecter()* qui permet à un utilisateur de se connecter ;
- *modifier_user()* autorisant un utilisateur à modifier son compte (identifiant ou mot de passe) ;
- *supprimer_user()* offrant la possibilité de supprimer un compte.

Après, la classe **UtilisateurDao** contient les méthodes *creer_user()*, *modifier_user()*, *supprimer_user()* ainsi que *trouver_par_id_user()* qui permet de trouver un `Utilisateur` dans la base de donnée grâce à son identifiant.

2.2.2 Token

En deuxième temps, la classe **Token** contient les attributs privés `token` et `id_user`. De plus, la classe **TokenService** contient les méthodes :

- *creer_token()* permettant d'associer un token a un utilisateur ;
- *existe_token()* qui vérifie l'existence d'un token pour un utilisateur donné ;
- *est_valide_token()* qui vérifie cette fois si que la durée de validité du token n'est pas dépassée et qui supprime le token de la base de donnée si celui-ci n'est plus valide ;
- *trouver_id_user_par_token()* qui permet de trouver l'identifiant d'un utilisateur dans la base de donnée grâce à son token.

Par ailleurs, il y a la classe **TokenDao** qui contient les méthodes : *creer_token()*, *existe_token()*, *est_valide_token()*, *trouver_id_user_par_token()*, *trouver_token_par_id_user()* qui permet de trouver le token dans la base de donnée à partir de l'identifiant de l'utilisateur et *trouver_token_par_valeur()* qui déterminer si le token fourni existe et renvoie éventuellement toutes les informations liées.

2.2.3 QR code

Concernant la classe **QRCode**, celle-ci possède les attributs :

- `id_qrcode` qui est l'identifiant du QR code et une clé primaire ;
- `url` qui est l'URL du QR code ;
- `id_propriétaire` correspondant à l'identifiant du créateur du QR code ;
- `date_création` qui est la date de création du QR code ;
- `type` qui est un booléen tel que True correspondent à un QR code suivi et False un QR code simple ;

— `couleur` dans le cas d'un QR code coloré ;

— `logo` qui permet d'ajouter un logo au centre du QR code.

Par la suite, la classe **QRCodeService** possède les méthodes :

— *creer_ qrcode()* permettant de générer un QR code ;

— *trouver_ qrc_ par_ id_ user()* qui permet de donner la liste des QR code créer par un utilisateur donné ;

— *supprimer_ qrcode()* pour supprimer les statistique et la redirection du qrcode.

La classe **QRCodeDao** comprend *creer_ qrcode()*, *trouver_ qrc_ par_ id_ user()* et *supprimer_ qrcode()*.

2.2.4 Statistique

En dernier temps, la classe **Statistique** possède les attributs privés `id_qrcode` qui identifie le QR code, `nombre_vues` et `date_vues` qui liste les vues du QR code.

La classe **StatistiqueService** contient les méthodes *creer_ staistique()* qui permet de repertorier les statistiques d'un QR code, *modifier_ statistique()* pour les actualiser et *afficher()* pour pouvoir les consulter.

Finalement, la classe **StatistiqueDao** contient les méthodes *creer_ staistique()*, *modifier_ statistique()* et *afficher()*.

2.3 Diagrammes de séquence

2.3.1 Séquence de création d'un tocken

L'authentification par token garantit un accès sécurisé à l'ensemble des services de l'application de création de QR codes suivis. Cette approche permet de contrôler et tracer les opérations tout en maintenant la sécurité des données utilisateurs. Les diagrammes ci-dessous illustrent les processus d'authentification et de création d'un QR code suivi.
La création d'un token débute lorsqu'un utilisateur tente de se connecter via le **UtilisateurService**. Ce service vérifie d'abord l'existence de l'utilisateur en consultant le **UtilisateurDao**, lequel interroge la base de données. Si l'utilisateur n'est pas valide, la création du token est interrompue et un message d'erreur est renvoyé au client.

Pour un utilisateur valide, le **UtilisateurService** délègue la gestion du token au **TokenService**. Le **TokenService** consulte le **TokenDao** pour déterminer si un token existe déjà pour cet utilisateur.

— Si un token existant est trouvé : le **TokenService** utilise la classe métier **Token** pour vérifier sa validité. Si le token est valide, il est réutilisé, sinon, un nouveau token est créé par le **TokenService**, enregistré en base via le **TokenDao**, puis transmis au **UtilisateurService**.

— Si aucun token n'existe : le **TokenService** crée directement un nouveau token, le stocke via le **TokenDao**, et le renvoie au **UtilisateurService**.

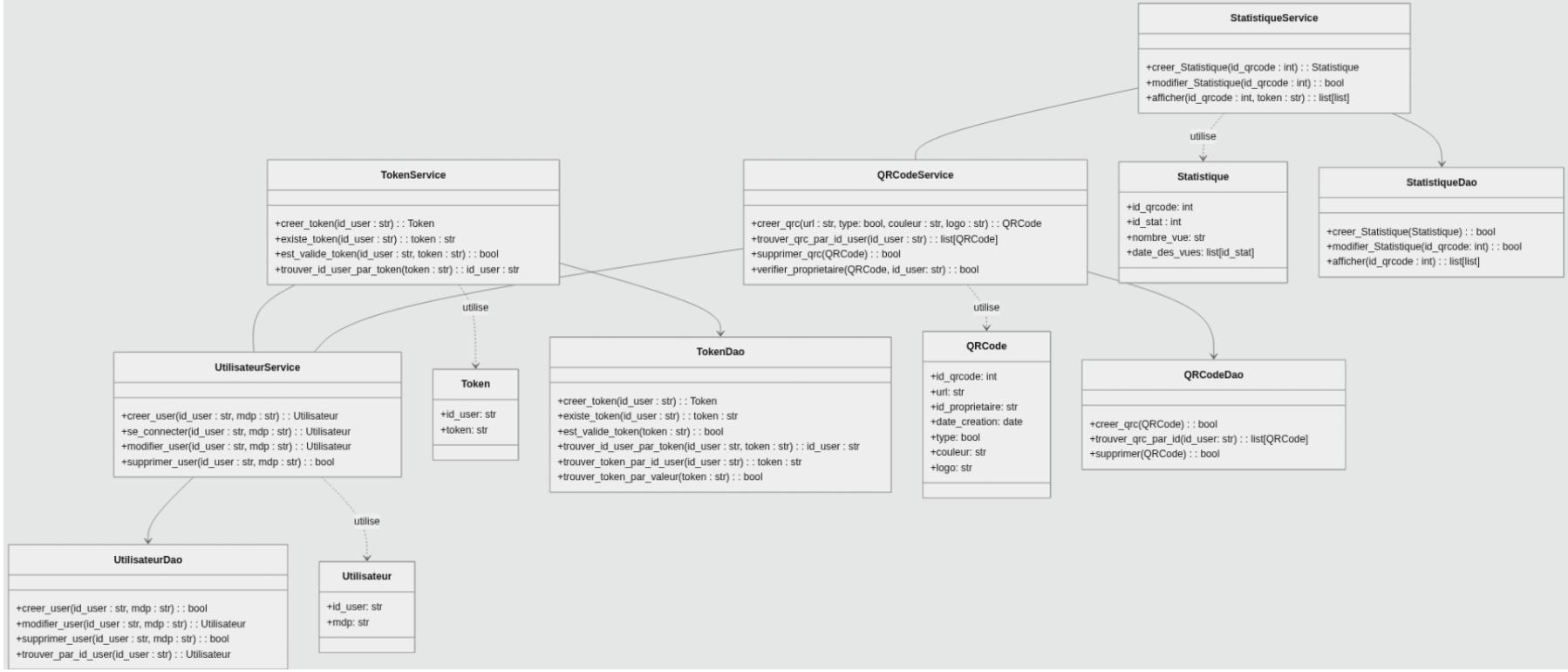

FIGURE 3 – Diagramme de classes

Enfin, le service **UtilisateurService** retourne au client le token valide ou un message d'erreur. Ce processus garantit que chaque utilisateur authentifié dispose d'un token unique et valide, tout en évitant la génération de tokens pour des utilisateurs non autorisés.

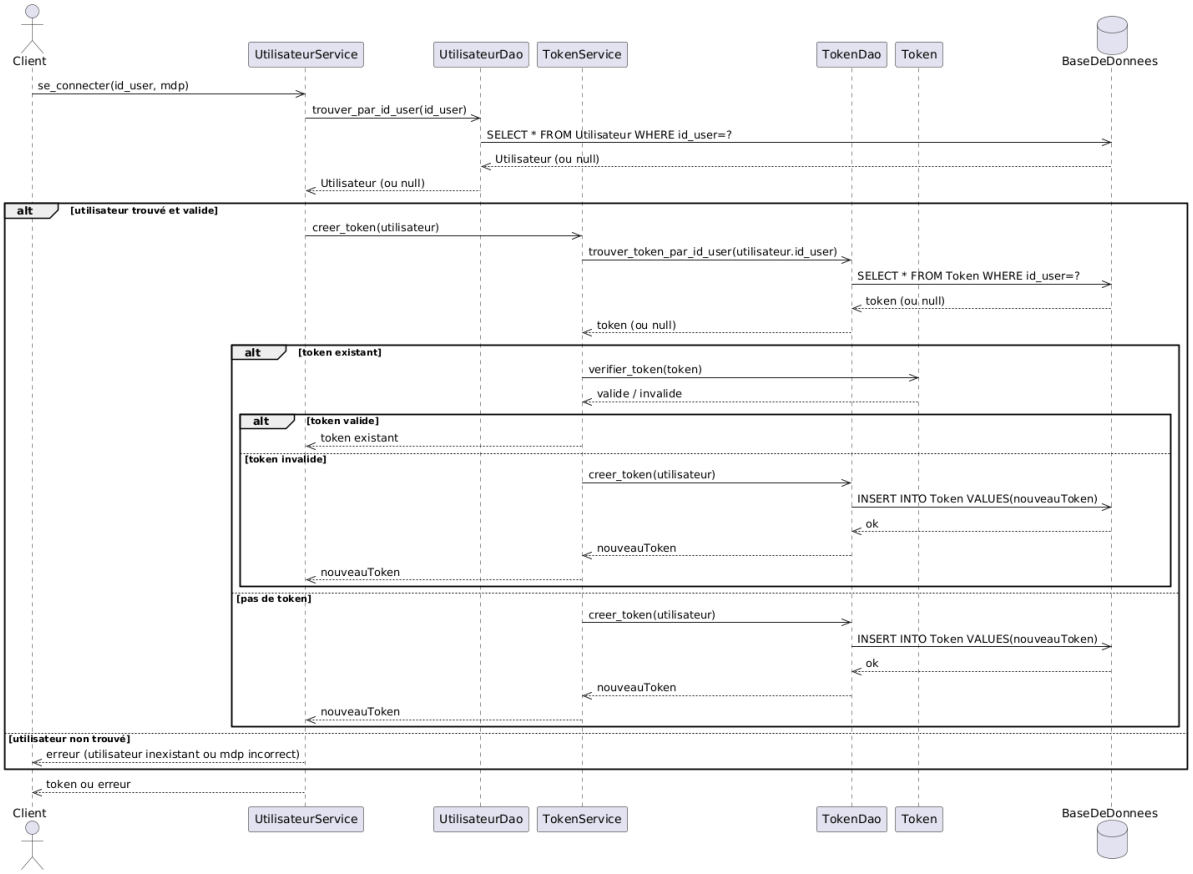

FIGURE 4 – Séquence de création d'un tocken

Une fois le système d'authentification établi, examinons maintenant comment ce token est utilisé concrètement lors de la création d'un QR code suivi.

2.3.2 Séquence de création d'un QR code suivi

Le processus de création d'un QR code suivi se déroule en deux phases principales : la vérification de l'authentification et la génération effective du code.

Lorsqu'un utilisateur demande la création d'un QR code, il transmet les informations nécessaires (URL, type, couleur, logo) ainsi qu'un token d'authentification.

i. Vérification du token

- Le service **QRCodeService**, chargé de la logique métier relative aux QR codes, délègue la vérification du token au **TokenService**.
- Le service **TokenService** appelle la méthode *trouver_token_par_valeur()* du **TokenDao**, qui interroge la base de données afin de déterminer si le token fourni existe et renvoie éventuellement toutes les informations liées.
- Si le token est absent, une erreur est immédiatement renvoyée au client.

— Dans le cas contraire, le **TokenService** procède à une vérification métier via sa méthode *est_valide_token()*, afin de confirmer que le token n'est pas expiré ou invalide. Si cette validation échoue, une erreur est également renvoyée.

ii. **Création du QR code**

— Si le token est reconnu comme existant et valide, le **QRCodeService** poursuit le traitement.

— Il appelle alors le **QRCodeDao**, qui se charge de l'insertion des données du QR code (URL, type, couleur, logo et propriétaire lié au token) dans la base de données.

— Une fois l'opération réussie, le **QRCodeDao** retourne l'objet QR code créé au **QRCodeService**, qui le renvoie finalement au client.

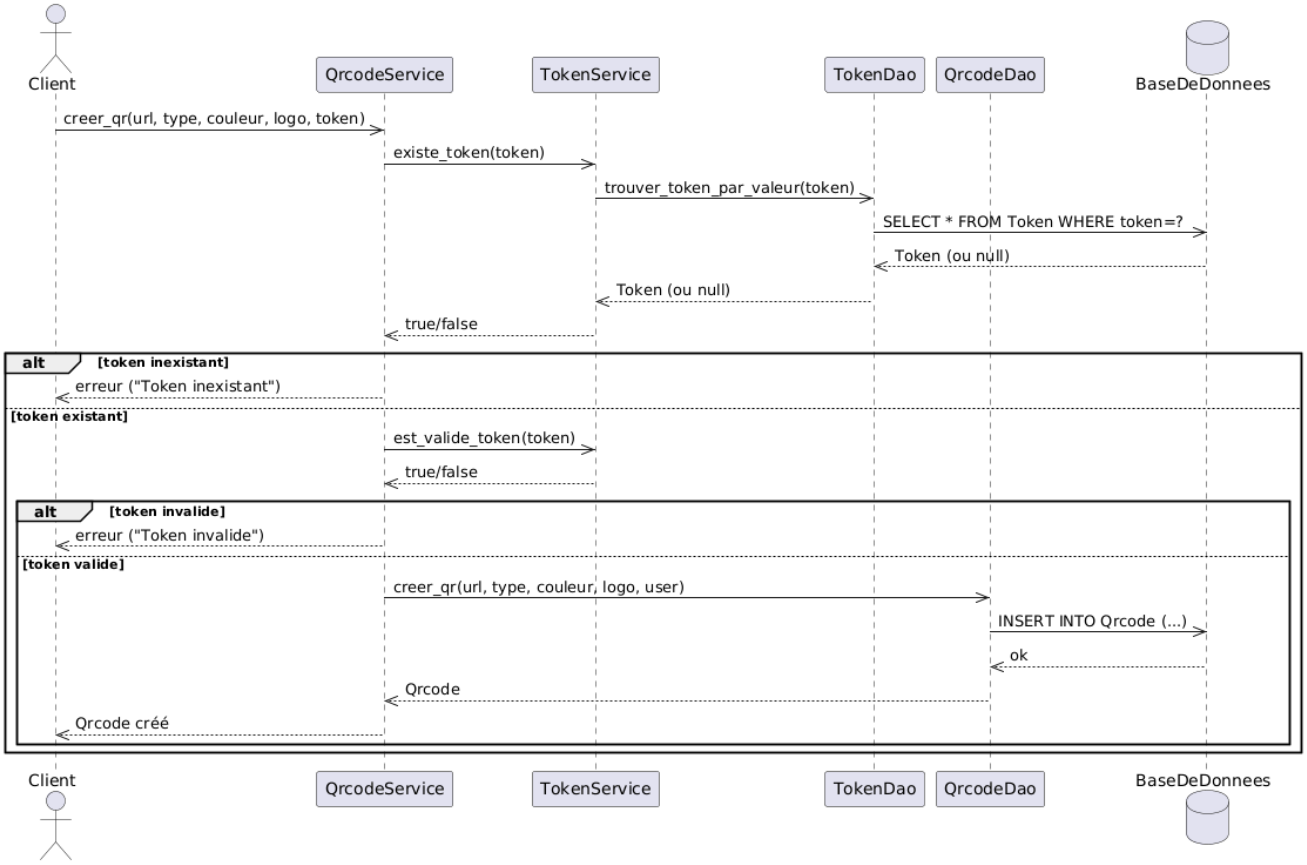

FIGURE 5 – Séquence de création d'un QRCode suivi

Une fois les QR codes créés et stockés dans le système, ils peuvent être scannés par des utilisateurs externes, déclenchant ainsi un processus de traçabilité et de redirection.

2.3.3 Séquence de tracking d'un QR code

Lorsqu'un utilisateur externe scanne un QR code, la requête est d'abord transmise au **QRCodeService** qui se charge de la logique de traitement. Ce service interroge le **QRCodeDao**, lequel effectue une requête SQL sur la base de données afin de rechercher

le QR code correspondant à l'identifiant fourni. Si le QR code est trouvé, le **QRCodeService** appelle le **StatistiqueService** pour enregistrer l'événement. Celui-ci sollicite à son tour le **StatistiqueDao**, qui insère dans la base de données une nouvelle ligne dans la table **Statistique** avec l'identifiant du QR code et la date du scan. Une fois l'enregistrement validé, le flux de retour propage un message de succès vers le **QRCodeService**, qui redirige alors le client externe vers l'URL associée au QR code. En revanche, si aucun QR code n'est trouvé, le **QRCodeService** renvoie directement au client externe un message d'erreur signalant un QR code invalide.

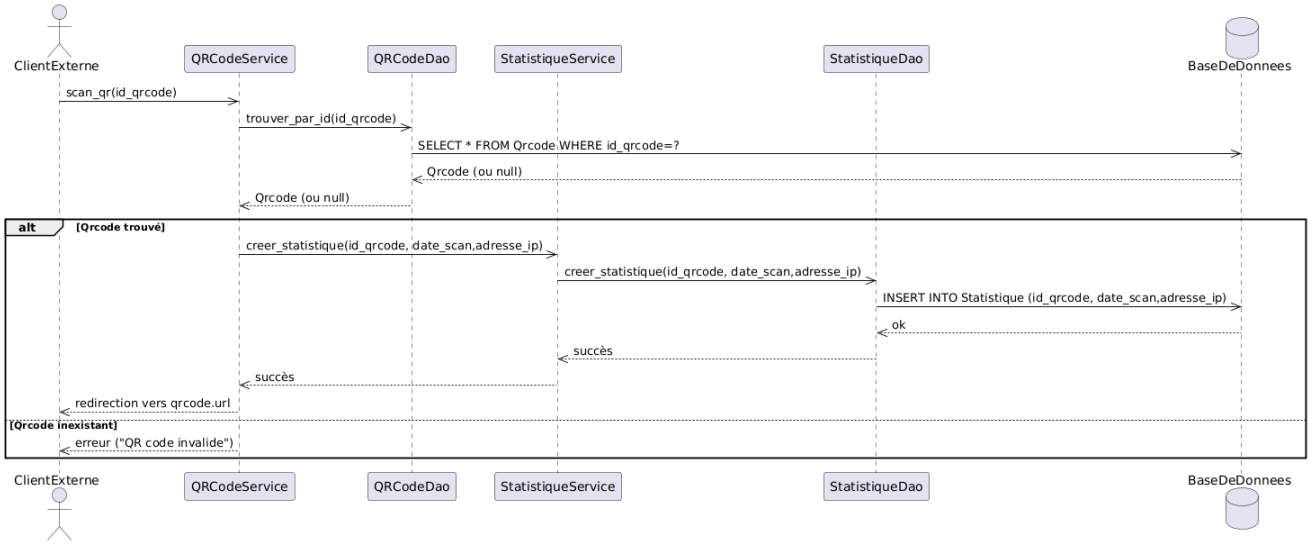

FIGURE 6 – Séquence de tracking d'un QR code suivi

Le but de la création des QR codes suivis étant de suivre son utilisation par les usagers, il est important de préciser la façon dont il pourra avoir accès aux statistiques d'utilisation de son QR code.

2.3.4 Séquence de la consultation des statistiques d'un QR code

Lorsqu'un utilisateur souhaite consulter les statistiques liées à un QR code, il initie la requête en appelant la méthode *afficher_statistiques()* exposée par la classe **StatistiqueService**. Cette requête contient l'identifiant du QR code concerné ainsi que le token. Le **StatistiqueService** délègue la vérification du token au **TokenService**. Ce dernier interroge le **TokenDao** via la méthode *trouver_token_par_valeur()*.

— Si aucun enregistrement n'est trouvé, ou si le token est expiré, une erreur est renvoyée indiquant un token invalide.

— Dans le cas contraire, le token est jugé valide et le processus se poursuit.

Une fois le token validé, le **TokenService** utilise la méthode *trouver_id_user_par_token()* du **TokenDao** pour identifier l'utilisateur propriétaire du token. L'identifiant de l'utilisateur (`id_user`) est alors transmis au **StatistiqueService**. Le **StatistiqueService** interroge ensuite le **QRCodeDao** pour récupérer le QR code ciblé (*trouver_par_id_user()*). Une vérification de cohérence est effectuée par le

QRCodeService afin de confirmer que ce QR code appartient bien à l'utilisateur dont l'identifiant a été obtenu via le token.

— Si le QR code n'appartient pas à cet utilisateur, une erreur est renvoyée indiquant un accès interdit.

— Si la vérification est concluante, le **StatistiqueService** fait appel au **StatistiqueDao** pour obtenir les données statistiques associées au QR code (*afficher()*). Ces informations sont ensuite renvoyées à l'utilisateur sous forme de résultats exploitables.

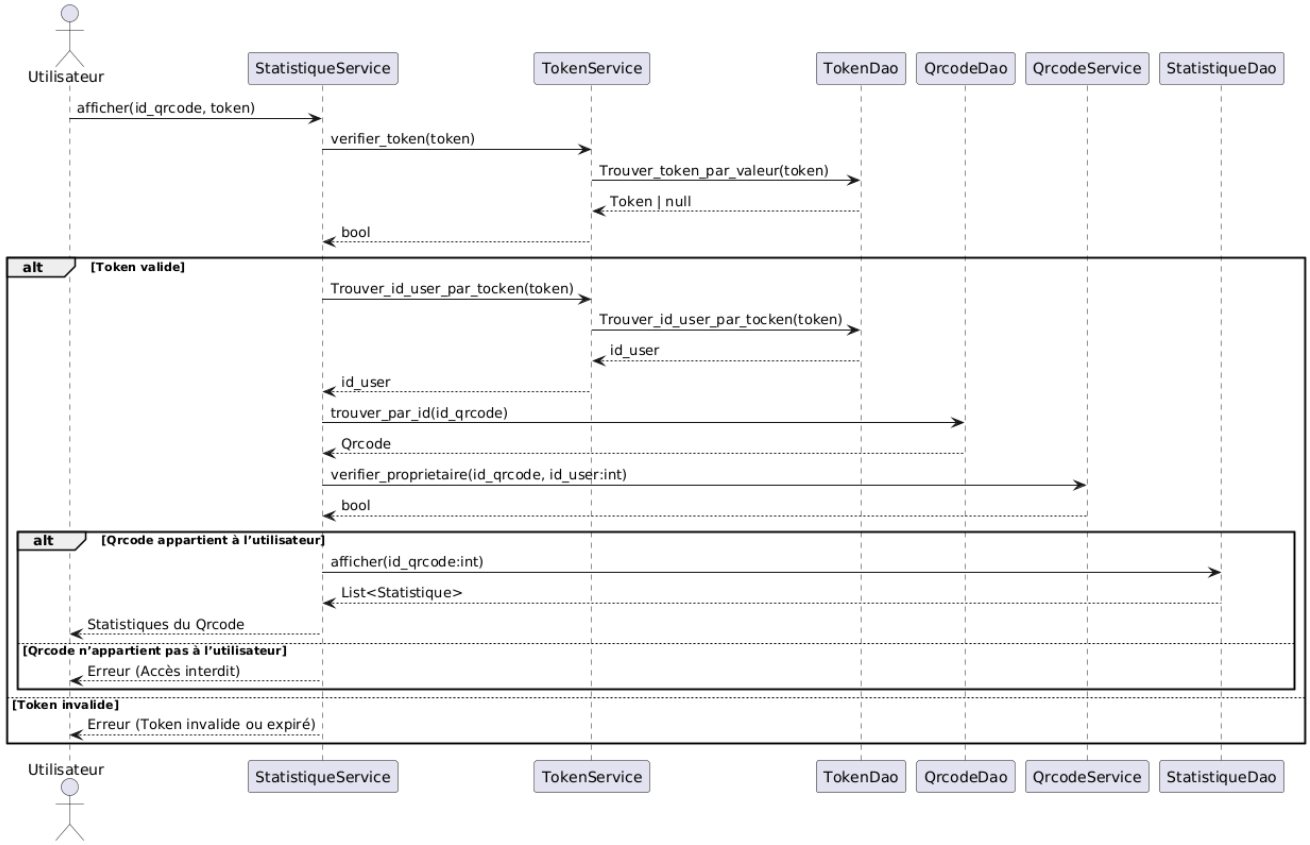

FIGURE 7 – Séquence de consultation d'un QR code suivi

2.4 Diagramme de base de données

Le modèle de données de notre application s'articule autour de quatre tables principales : **QRCode**, **Utilisateur**, **Statistique** et **Token**.

La table **QRCode** est la plus centrale. Elle contient l'ensemble des informations relatives à chaque code généré, comme l'identifiant unique, l'URL associée, le propriétaire, la date de création, le type de code (suivi ou non), ainsi que des éléments de personnalisation tels que la couleur ou le logo.

La table **Utilisateur** regroupe plutôt les informations nécessaires à l'identification et à l'authentification des membres de l'application, à travers un identifiant unique, un pseudonyme et un mot de passe. La table **Token** est directement reliée aux utilisateurs

et permet de gérer les connexions et l'authentification de manière sécurisée, chaque utilisateur se voyant attribuer un token unique lors de sa session.

Enfin, la table **Statistique** est associée aux QR codes et recense les données d'utilisation : le nombre total de vues ainsi que les dates précises de consultation.

Ce schéma relationnel assure ainsi la cohérence de l'ensemble du système. Chaque utilisateur peut générer plusieurs QR codes, chacun d'entre eux étant suivi par des statistiques détaillées, tandis que la table **Token** garantit la sécurité et la gestion des accès. L'organisation retenue permet donc à la fois de créer et de personnaliser des QR codes, de gérer les utilisateurs et d'exploiter les données liées à leur utilisation.

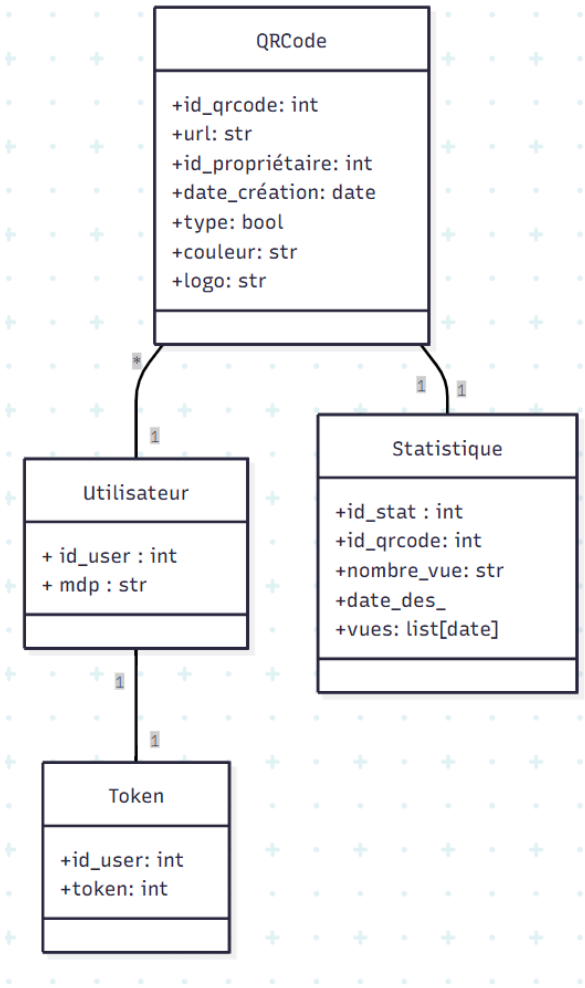

FIGURE 8 – Diagramme de base de données

3 Conclusion intermédiaire

Tout au long de cette première partie de projet, nous avons surtout appris à faire connaissance et à comprendre la manière de travailler de chacun. Cela nous permettra donc d'avancer plus rapidement dans les prochaines phases du projet.

Par ailleurs, en parallèle du projet, les cours de compléments d'informatique nous ont permis d'approfondir nos connaissances sur le fonctionnement d'une application. La plupart de ces notions nous étaient inconnues jusqu'à récemment, et elles nous permettent désormais d'aborder sereinement la suite du projet.

Enfin, cette période nous a également permis de consolider l'analyse et de définir clairement la suite du projet. Les différents diagrammes réalisés offrent une vision structurée de l'architecture et des interactions, tandis que le diagramme de Gantt trace la feuille de route des prochaines étapes, principalement axées sur l'implémentation et la validation des choix effectués.