**Diagrammes d’activité**

* **Création d’un tocken**

Le **token** assure l’authentification sécurisée de l’utilisateur et constitue la clé d’accès aux différentes fonctionnalités de l’application. Ainsi, tout utilisateur désirant accéder aux fonctionnalités de l’application doit se procurer un tocken. Comme l’illustre le diagramme ci-dessous, l’utilisateur fait une requête, le système vérifie son existence dans la base ; puis le cas échéant, s’il ne dispose pas d’un tocken valide, le système génère et lui retourne un token valide ; si l’utilisateur n’existe pas, un message d’erreur ou d’invitation à créer un compte est renvoyé.

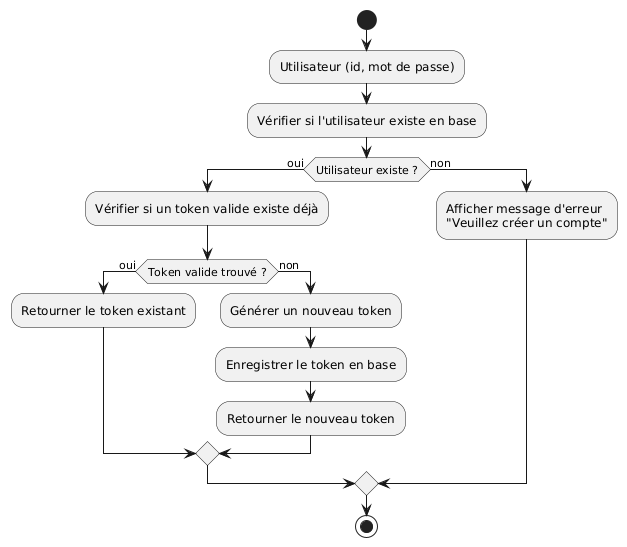
****

Figure 1: diagramme d'activité de création d'un tocken

* **Création de Qr code suivi**

Pour générer un QrCode suivi, l’utilisateur fournit les données à encoder ainsi que, éventuellement, des informations de suivi telles que la date d’expiration, le type de QR code ou le nombre maximal d’utilisations. Le serveur commence par vérifier le **token** de l’utilisateur pour s’assurer de son authentification : si le token est valide, le QR code est généré, enregistré avec ses métadonnées dans la base de données, puis retourné à l’utilisateur ; si le token est absent ou invalide, un message d’erreur ou une invitation à créer un compte est renvoyé.

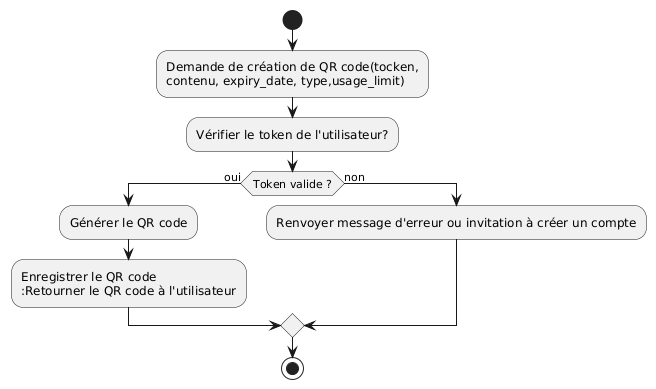
****

Figure 2: diagramme d'activité de création d'un qr code

* **Scan d’un qrcode**

Lorsqu’on scanne un QR code suivi, le système reçoit une requête contenant l’identifiant du QR code. Une vérification est d’abord effectuée pour s’assurer que ce QR code existe bien dans la base de données et qu’il est encore valide.

* Si le QR code est valide, le système enregistre un nouvel événement de type **Scan**, contenant la date, l’heure, l’adresse IP ou d’autres informations contextuelles. Le compteur de scans associé au QR code est mis à jour. Ensuite, l’utilisateur est automatiquement redirigé vers l’URL cible liée au QR code.
* Si le QR code est invalide (introuvable ou expiré), le système renvoie un message d’erreur à l’utilisateur au lieu d’effectuer la redirection.

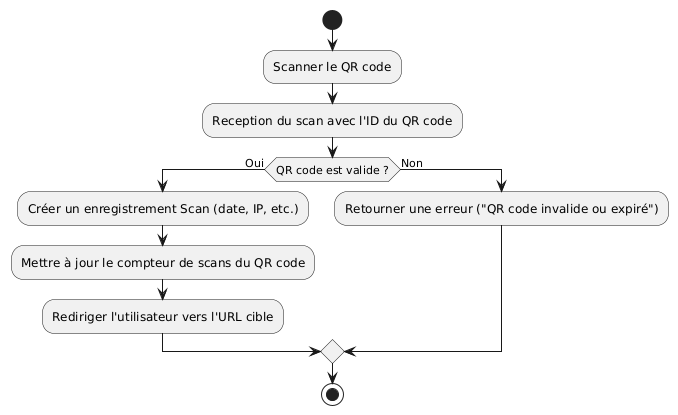
****

Figure 3: diagramme de séquence de scan d'un QrCode

**Les diagrammes de séquence**

Donner une def générale

* **Diagramme de séquence de création de tocken**

Le diagramme illustre la procédure par laquelle un utilisateur obtient un **token d’authentification** pour accéder aux services sécurisés de l’API.

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 metier *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*.

Enfin, le *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.

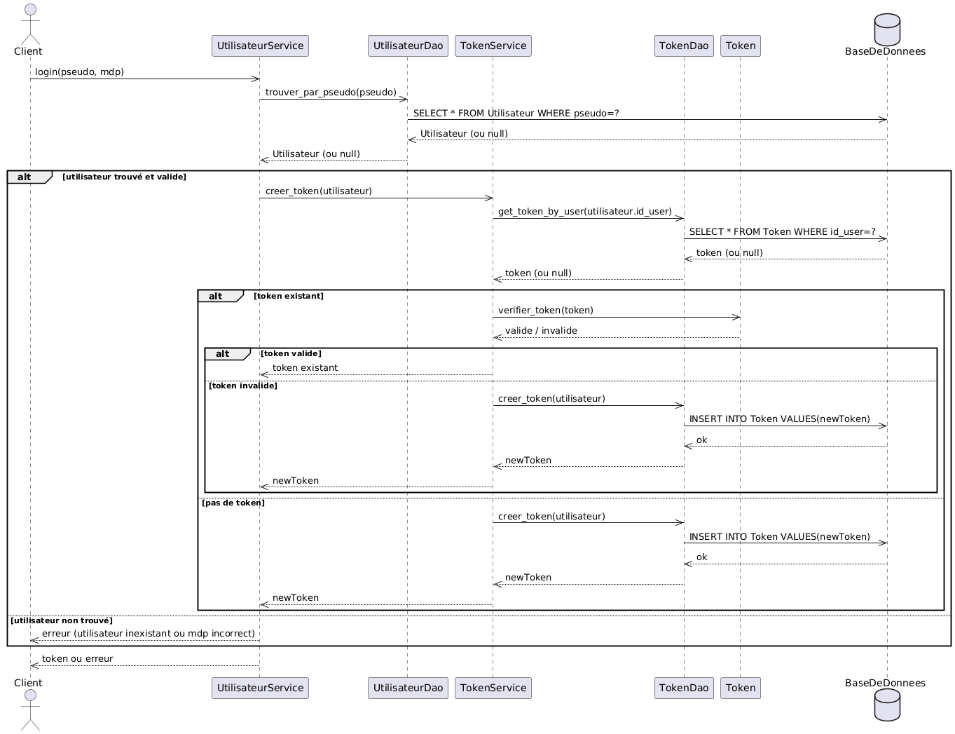


Figure 4: séquence de création de tocken

transition

* **Diagramme de séquence de création d’un QR Code suivi**

Lorsqu’un **client** demande la création d’un QR Code, il transmet les informations nécessaires (URL, type, couleur, logo) ainsi qu’un **token d’authentification**.

1. **Vérification du token**

* Le *QrcodeService*, chargé de la logique métier relative aux QR Codes, délègue la vérification du token au *TokenService.*
* Le *TokenService* appelle la méthode *get\_token\_by\_value* du *TokenDao,* qui interroge la base de données afin de déterminer si le token fourni existe et renvoie éventuellement toutes les informations y afférentes.
* 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.

1. **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.

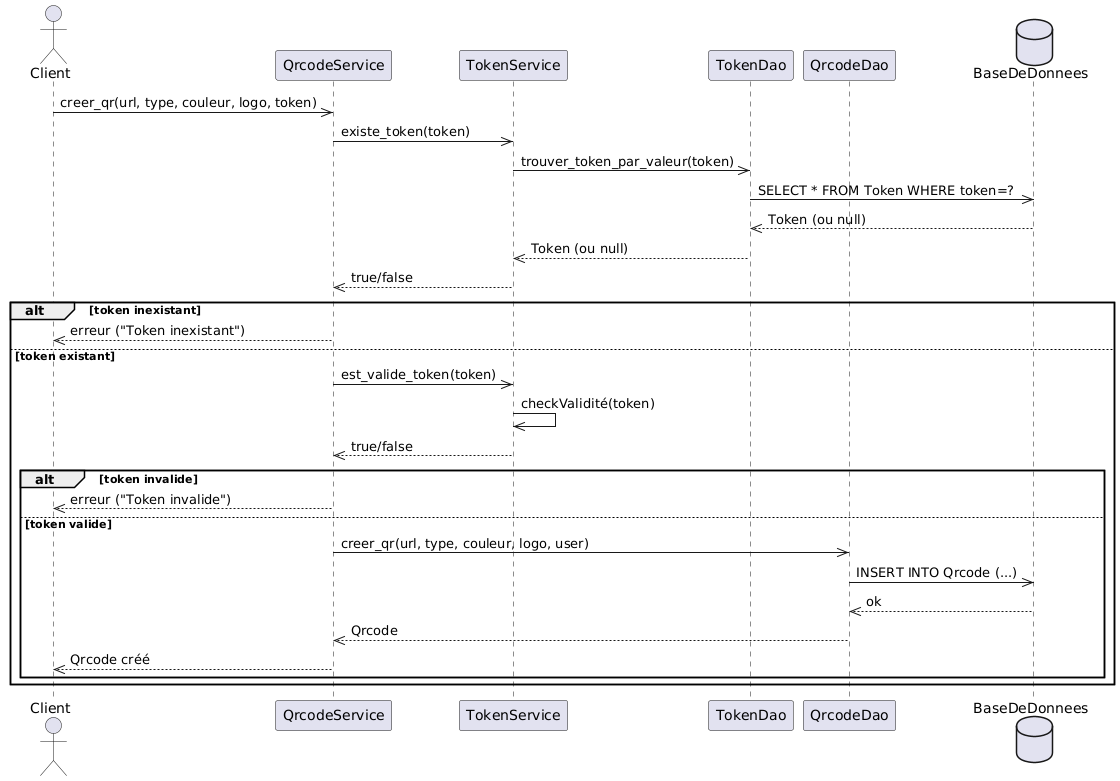
****

Figure 5: séquence de création d'un QRCode suivi

* **Diagramme de séquence de la consultation des statistiques d’un QrCode**

Lorsqu’un utilisateur souhaite consulter les statistiques liées à un QrCode, il initie la requête en appelant la méthode *afficher\_statistiques(qr:Qrcode, token:Token)* exposée par la classe *StatistiqueService*. Cette requête contient l’identifiant du QR Code concerné ainsi que le jeton d’authentification (token). Le *StatistiqueService* délègue la vérification du token au *TokenService.* Ce dernier interroge le *TokenDao* via la méthode *get\_token\_by\_value(token:string).*

* + 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 *get\_user\_id\_by\_token(token:string)* 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(id\_qrcode:int)).* 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(id\_qrcode:int)*). Ces informations sont ensuite renvoyées à l’utilisateur sous forme de résultats exploitables.

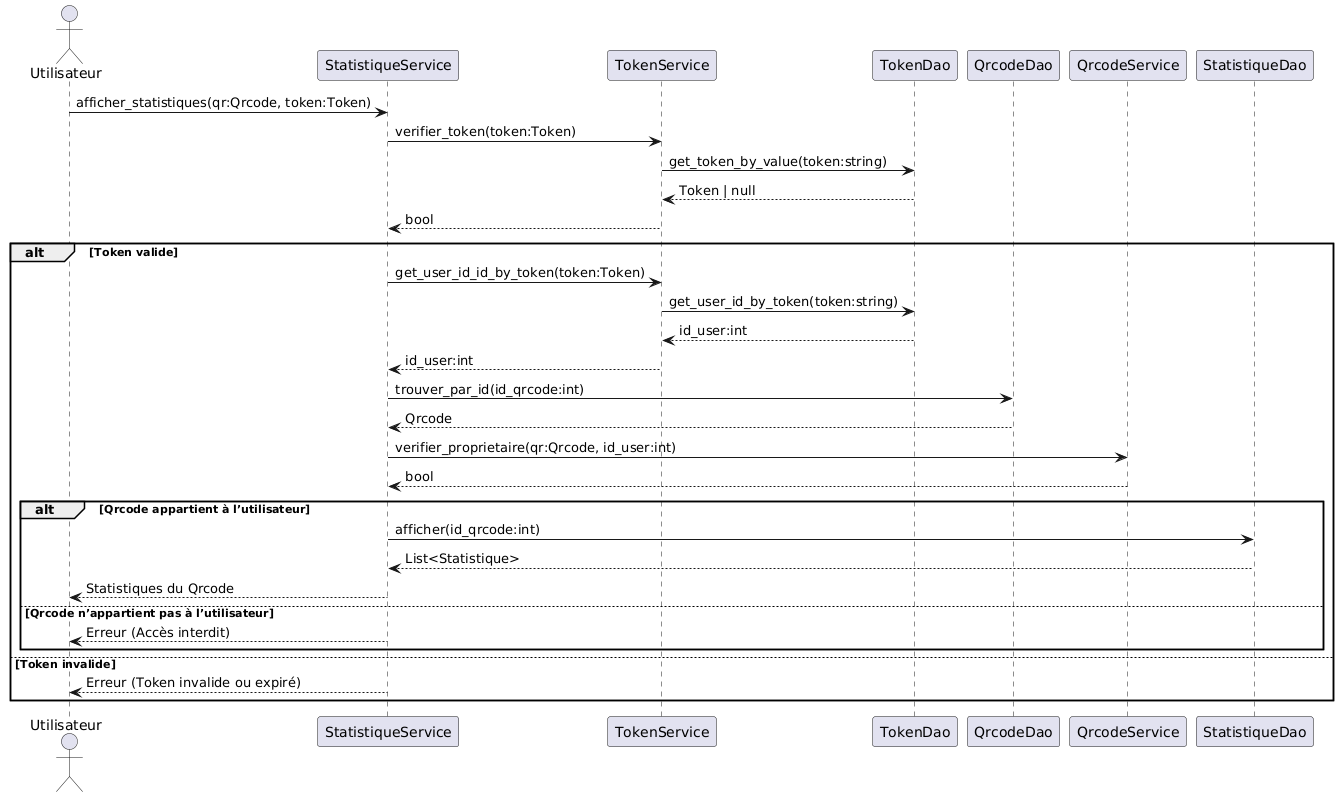


Figure 6: séquence de consultation des statistiques d'un QrCode

* **Diagramme de séquence de scan d’un QrCode**
* Lorsqu’un *ClientExterne* 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 *BaseDeDonnees* 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, la date du scan et l’adresse IP. Une fois l’enregistrement validé, le flux de retour propage un message de succès vers le *QRCodeService*, qui redirige alors le *ClientExterne* vers l’URL associée au QR code. En revanche, si aucun QR code n’est trouvé, le *QRCodeService* renvoie directement au *ClientExterne* un message d’erreur signalant un QR code invalide.

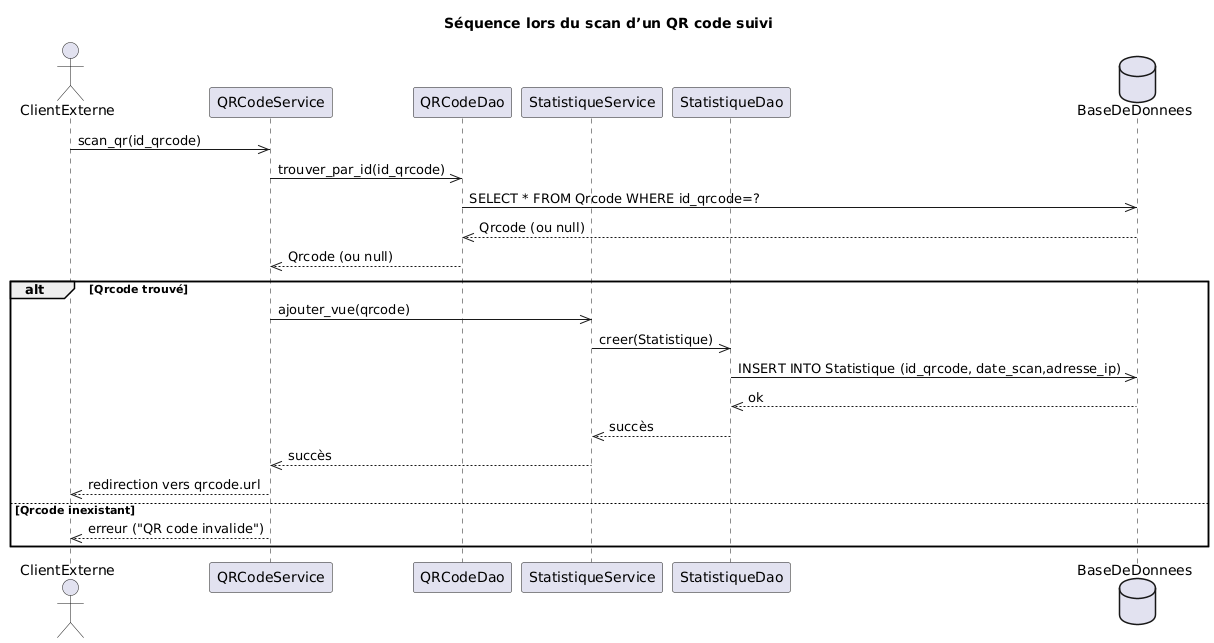


Figure : séquence de scan d'un QrCode