Documentation technique

ToDo & Co - ToDoList

Table des matières

1. Présentation de l'application

1.1 Environnement de développement

L'application utilise le Framework ***Symfony***. Le version utilisée ici est la 4.4, qui est la version LTS (Long Term Support).
Le langage de programmation utilisé est ***PHP***. La version utilisée ici est la 7.4

Ces 2 versions garantissent un support optimisé pour une année afin de préparer une éventuelle évolution vers Symfony 5 et PHP 8 d'ici un an.

L'environnement d'hébergement consiste en un serveur ***Apache*** avec une base de données en ***MySql***.
Prévoir l'environnement de développement de développement adapté à la programmation sur Symfony, avec Notamment les outils suivants :

- Serveur Apache
- Mysql et phpMyAdmin
- php 7.4 avec les extensions suivantes :
  php7.4-gd php7.4-mysql php7.4-cli php7.4-json php7.4-common php7.4-mbstring php7.4-opcache php7.4-readline php7.4-bz2 php7.4-curl php7.4-intl php7.4-xml php7.4-xdebug php7.4-zip
- Git
- Composer
- SymfonyCLI (https://symfony.com/download)
  Optionnel mais très utile car il permet entre autres de valider l'environnement de développement avant d'y installer Symfony
- Logiciel de développement : PhpStorm
  Optionnel mais très utile pour développer en Symfony
  Plugins à prévoir :
  - Symfony support
  - Php Annotations
  - Php Toolbox
  - **PHP Unit code coverage**
  - Key Promoter X

1.2 Installation

1.2.1 Installation et paramétrages

- Cloner le repository Github :

```
git clone https://github.com/Reididsorg/test_8_local.git
```

- Cloner le fichier .env à la racine et le renommer en .env.local.
y renseigner le mode d'environnement : APP_ENV (Production) ou APP_dev (Développement)
Y renseigner aussi l'accès à la base de données selon :

```
DATABASE_URL=mysql://databaseUser:databasePassword@127.0.0.1:3306/databaseName
```

- Installer les dépendances :

```
composer install
```

- Installer la base de données ainsi que le jeu de données initiales (fixtures) :

```
bin/console doctrine:database:create (or manually create database)
```

```
bin/console doctrine:migrations:diff
```

```
bin/console doctrine:migrations:migrate
```

```
bin/console doctrine:fixtures:load
```

L'installation est terminée :)

On peut lancer le Symfony Local Web Server (fourni dans Symfony) :

```
symfony serve -d
```

----> La réponse indique un lien du type : http://127.0.0.1:8000/
C'est l'url de l'application :)

1.2.2 Mise à jour des dépendances

L'application utilise des librairies tierces à l'aide de ***Composer***.
Ces librairies sont listées dans le fichier composer.json, à la racine de l'application.
Une mise à jour régulière des librairies peut permettre d'éviter des failles de sécurité et de bénéficier d'éventuelles corrections de bugs et/ou bugs.

```
composer update
```

1.3 Outils de tests

Le composant PHPUnit est utilisé pour la réalisation de tests unitaires et fonctionnels de l'application.
A ce titre, l'extension PHP xDebug est requise.

Les mesures de la qualité de code utilisent Codacy.
Les mesures de performance de l'application utilisent Blackfire.

2. Base de données

2.1 Entités et Doctrine

L'application utilise l'ORM (Object-Relational Mapping) Doctrine.
Les entités Task et User sont définies dans src/Entity.
Leur manipulation est définie dans les classes TaskRepository et UserRepository dans src/Repository.
Les contrôleurs utilisent les entités pour exercer le code métier en invoquant les différents services nécessaires dans le dossier src/Controller.
Les entités sont reliées afin de pouvoir associer des utilisateurs à des tâches.

- OneToMany : User vers Task (Pour un utilisateur, plusieurs tâches)
- ManyToOne : Task vers User (Plusieurs tâches pour un utilisateur)

2.2 Fixtures

Les classes de fixtures sont situées dans le dossier src/DataFixtures.
Elles permettent de persister une quantité définie d'entités dans la base de données.
Dans le but de permettre une compréhension aisée des droits spécifiques à chaque utilisateur, 3 utilisateurs ont été créés avec un rôle spécifique :

- Admin : ROLE_ADMIN
- User : ROLE_USER
- Anonymous : IS_AUTHENTICATED_ANONYMOUSLY

Les tâches ont été attribuées à chacun des 3 utilisateurs, avec pour chacun d'eux, les 2 types de tâche : "A faire" et "Terminées".

2.3 Stockage des utilisateurs

Le champ "Username" de l'entité User est unique. C'est ce qui permet la récupération de l'utilisateur par la classe "LoginFormAuthenticator" lors de l'authentification.

L'unicité est spécifiée grâce à l'annotation :

```
/**
* @ORM\Column(type="string", length=25, unique=true)
```

Les relations entre les deux entités indiquées en annotations se traduisent en base par les jeux suivants :

- Table user :

id	username	password	email	roles (DC2Type:array)
1	Admin	$2y$13$fvzME9hKpG6aajPb8f3/AegUe6E.LZrIbiiqbkXYqXP...	admin@admin.fr	a:1:{i:0;s:10:"ROLE_ADMIN";}
2	User	$2y$13$uxY9QvcejOleN.dqLwBwSeXgQxztaxCWhBt9bQ.z8hN...	user@user.fr	a:1:{i:0;s:9:"ROLE_USER";}
3	Anonymous	$2y$13$9HYPs4zihPueQYwE3f7yqOgcWFBBroYj2bubrQbxra0...	anonymous@anonymous.fr	a:1:{i:0;s:28:"IS_AUTHENTICATED_ANONYMOUSLY";}

- Table task :

id	user_id	created_at	title	content	is_done
1	1	2021-10-01 11:34:08	Tâche 1	Provident placeat et similique velit quas. 1	0
2	3	2021-10-01 11:34:08	Tâche 2	Quos dolorum cumque delectus unde aliquid. 2	0
3	2	2021-10-01 11:34:08	Tâche 3	Rerum in quam nihil aliquam enim nobis vitae numqu...	0
4	2	2021-10-01 11:34:08	Tâche 4	Saepe impedit molestiae sint eos fuga illum simili...	1

Un utilisateur est associé à chacune des tâches.

Lors de la création d'une tâche, l'utilisateur connecté peut choisir de l'attribuer à un utilisateur ADMIN ou USER(par défaut)
Les données issues de fixtures comportent des tâches attribuées à l'utilisateur 'IS_AUTHENTICATED_ANONYMOUSLY', l'utilisateur 3.
Cela s'effectue dans le méthode createAction() du contrôleur src/Controller/TaskController.

3. Authentification

3.1 Fichiers d'authentification

- Configuration
config/packages/security.yaml
Configuration du processus d'authentification

- Entité
src/Entity/User.php
Entité des utilisateurs

- Contrôleurs
    - src/Controller/SecurityController.php
    Contrôleur de connexion / déconnexion

- src/Controller/UserController
  Contrôleur du formulaire de connexion

- Authentification
src/Security/LoginFormAuthenticator.php
Méthodes du processus d'authentification de l'application

- Vue
templates/security/login.html.twig
Template du formulaire de connexion

3.2 Page d'authentification

L'utilisateur accède à la page de connexion avec l'uri /login.
Cette route correspond à la classe src/Controller/SecurityController.php
La réponse est retournée à la vue "security/login" : templates/security/login.html.twig

3.3 Formulaire d'authentification et traitement

Une fois le formulaire rempli et validé, il est transmis à la classe src/Security/LoginFormAuthenticator.php.
Cette classe étend la classe AbstractFormLoginAuthenticator et implémente l'interface PasswordAuthenticatedInterface.
Les méthodes exécutent les opérations suivantes :

- getCredentials() :
  Récupère les données du formulaire.

- getUser() :
  Vérifie le token issu des données du formulaire puis récupère l'utilisateur correspondant en base de données.
  Renvoie le message d'erreur en cas de problème d'identification

- checkCredentials() :
  A partir de l'utilisateur récupéré, valide les caractéristiques de cryptage du mot de passe associé à l'utilisateur

- getPassword() :
  Re-encrypte le mot de passe au bout d'un certain temps

- onAuthenticationSuccess() :
  En cas d'authentification réussie, gère la réponse à renvoyer en redirigeant vers le nom de la route choisie.
  Ici, la route "homepage" a été choisie selon :

```
// After authentication success, redirect to homepage
return new RedirectResponse($this->urlGenerator->generate('homepage'));
```

- getLoginUrl() :
  retourne la route associée au formulaire de connexion

Plus d'infos sur : https://symfony.com/doc/current/security/form_login_setup.html

3.4 Cryptage du mot de passe

Le système de cryptage du mot de passe est défini dans config/packages/security.yaml

```
security:
   encoders:
       App\Entity\User:
           algorithm: bcrypt
```

3.5 Contrôle des accès

Le contrôle des accès est défini de 2 manières :

- Dans config/packages/security.yaml
- A travers les fichiers de template

L'accès des pages de gestion des utilisateurs réservé aux seuls administrateurs est défini ainsi dans config/packages/security.yaml :

```
access_control:
    - { path: ^/users, roles: ROLE_ADMIN }
    - { path: ^/tasks, roles: [ROLE_ADMIN, ROLE_USER] }
    - { path: ^/finished-tasks, roles: [ROLE_ADMIN, ROLE_USER] }
```

Si un utilisateur connecté, possédant un rôle autre que USER_ADMIN, tente d'accéder à des pages de gestion d'utilisateur (commençant pas /user*), une page d'erreur apparaît.
Cette page est définie dans templates/bundles/TwigBundle/Exception
Des templates sont utilisés en fonction du type de réponse http : 404, 403, 500.
Dans le cas présent, c'est error403.html.twig qui est appelé.
Les utilisateurs non-connectés sont redirigés vers la page d'accueil.

Les droits d'accès de suppression des tâches en fonction des utilisateurs sont mis en place :

- Dans le template templates/task/list.html.twig (Affichage ou non du bouton)

- Dans la méthode de suppression deleteTaskAction() (Au cas où l'url est devinée)

Le rôle de l'utilisateur connecté est confronté à celui de l'auteur de la tâche.
Le username de l'utilisateur connecté est comparé à celui de l'auteur de la tâche.

4. Tests unitaires et fonctionnels

4.1 Paramétrage de l'environnement de tests

- Déclaration de la base de tests

Il faut déclarer la base de données à tester en créant le fichier .env.tests.local à la racine de l'application.
On y copie la déclaration de base de données de .env, le nom de la base sera automatiquement changé en "todolist-test"

```
DATABASE_URL=mysql://USER:$PASSWORD@127.0.0.1:3306/todolist
```

- Réinitialisation automatique de la base avant chaque test

Le composant ***DAMADoctrineTestBundle*** a été installé à cet effet.
Il a été activé en tant qu'extension PHPUnit dans phpunit.xml.dist à la racine de l'application.

- Paramétrage de phpunit.xml

Exclusion de Kernel.php et /DataFixtures

```
<coverage processUncoveredFiles="true">
```

```
    <include>
        <directory suffix=".php">src</directory>
    </include>
    <exclude>
        <file>src/Kernel.php</file>
        <directory>
            src/DataFixtures
        </directory>
    </exclude>
</coverage>
```

- Utilisation de PHPUnit

Les tests utilisent des données fictives.
Le script de création de ces données (fixtures) se situe dans src/DataFixtures.

Il faut aussi activer l'extension xdebug de la version PHP utilisée.

4.2 Création de la base de données de tests

Le nom sera celui de la base existante, suffixé de "_test" :

- Créer la bdd

```
symfony console doctrine:database:create --env=test
```

- Créer les champs grâce à la migration des entités

```
symfony console doctrine:migrations:diff --env=test
```

```
symfony console doctrine:migrations:migrate --env=test
```

- Remplir la base avec des fixtures

```
symfony console doctrine:fixtures:load --env=test
```

(ci-dessus, remplacer "symfony" par "php bin/console" le cas échéant)

4.3 Présentation des scripts de tests

L'ensemble des routes disponibles est testé.
Pour lister ces routes :

```
php bin/console debug:router
```

Voici la liste actuelle des routes :

```
  homepage                  ANY      ANY      ANY    /
```

```
  app_login                 ANY      ANY      ANY    /login
  app_logout                ANY      ANY      ANY    /logout
  task_list                 ANY      ANY      ANY    /tasks
  finished_task_list        ANY      ANY      ANY    /finished-tasks
  task_create               ANY      ANY      ANY    /tasks/create
  task_edit                 ANY      ANY      ANY    /tasks/{id}/edit
  task_toggle               ANY      ANY      ANY    /tasks/{id}/toggle
  task_delete               ANY      ANY      ANY    /tasks/{id}/delete
  user_list                 ANY      ANY      ANY    /users
  user_create               ANY      ANY      ANY    /users/create
  user_edit                 ANY      ANY      ANY    /users/{id}/edit
```

En analysant chaque route, on détermine la classe contrôleur et la méthode associés.
C'est cette méthode associée à la route qui sera testée.
Exemple avec la page d'accueil :

```
php bin/console debug:router homepage
```

Voici les infos obtenues :

```
 +--------------+-------------------------------------------------------------------+
| Property     | Value                                                             |
+--------------+-------------------------------------------------------------------+
| Route Name   | homepage                                                          |
| Path         | /                                                                 |
| Path Regex   | #^/$#sD                                                           |
| Host         | ANY                                                               |
| Host Regex   |                                                                   |
| Scheme       | ANY                                                               |
| Method       | ANY                                                               |
| Requirements | NO CUSTOM                                                         |
| Class        | Symfony\Component\Routing\Route                                   |
| Defaults     | _controller: App\Controller\DefaultController::indexAction        |
| Options      | compiler_class: Symfony\Component\Routing\RouteCompiler           |
+--------------+-------------------------------------------------------------------+
```

On peut donc analyser la méthode ***indexAction()*** de la classe **DefaultController**
Ainsi, pour **indexAction()**, ce sera ***testIndexAction().***

Les autres tests vérifient le fonctionnement des formulaires, les interactions CRUD avec la base de données.
Se placer à la racine de l'application :
Pour lancer l'ensemble de tests avec résultat dans la console

```
vendor/bin/phpunit
```

Ou alors pour lancer un seul test avec résultat dans la console (exemple : indexAction()) :

```
vendor/bin/phpunit --filter=testIndexAction
```

Pour afficher les tests sur une page html :

```
XDEBUG_MODE=coverage vendor/bin/phpunit --coverage-html tests/coverage
```

/home/mexico/Projects/test8_lts_full_2_tests/src / (Dashboard)

	Code Coverage									
	Lines			Functions and Methods			Classes and Traits			
Total		94.94%	150 / 158		88.24%	45 / 51		54.55%	6 / 11
Controller		97.33%	73 / 75		86.67%	13 / 15		50.00%	2 / 4
Entity		88.57%	31 / 35		87.50%	21 / 24		0.00%	0 / 2
Form		100.00%	19 / 19		100.00%	2 / 2		100.00%	2 / 2
Repository		100.00%	4 / 4		100.00%	2 / 2		100.00%	2 / 2
Security		92.00%	23 / 25		87.50%	7 / 8		0.00%	0 / 1

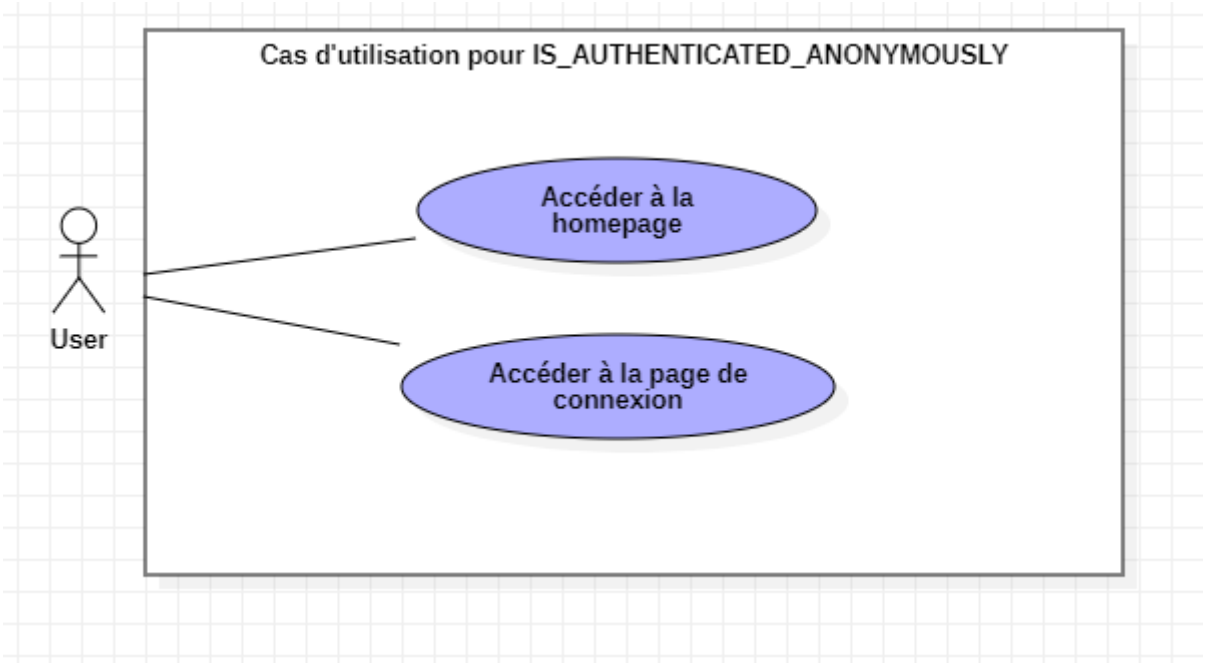
Legend

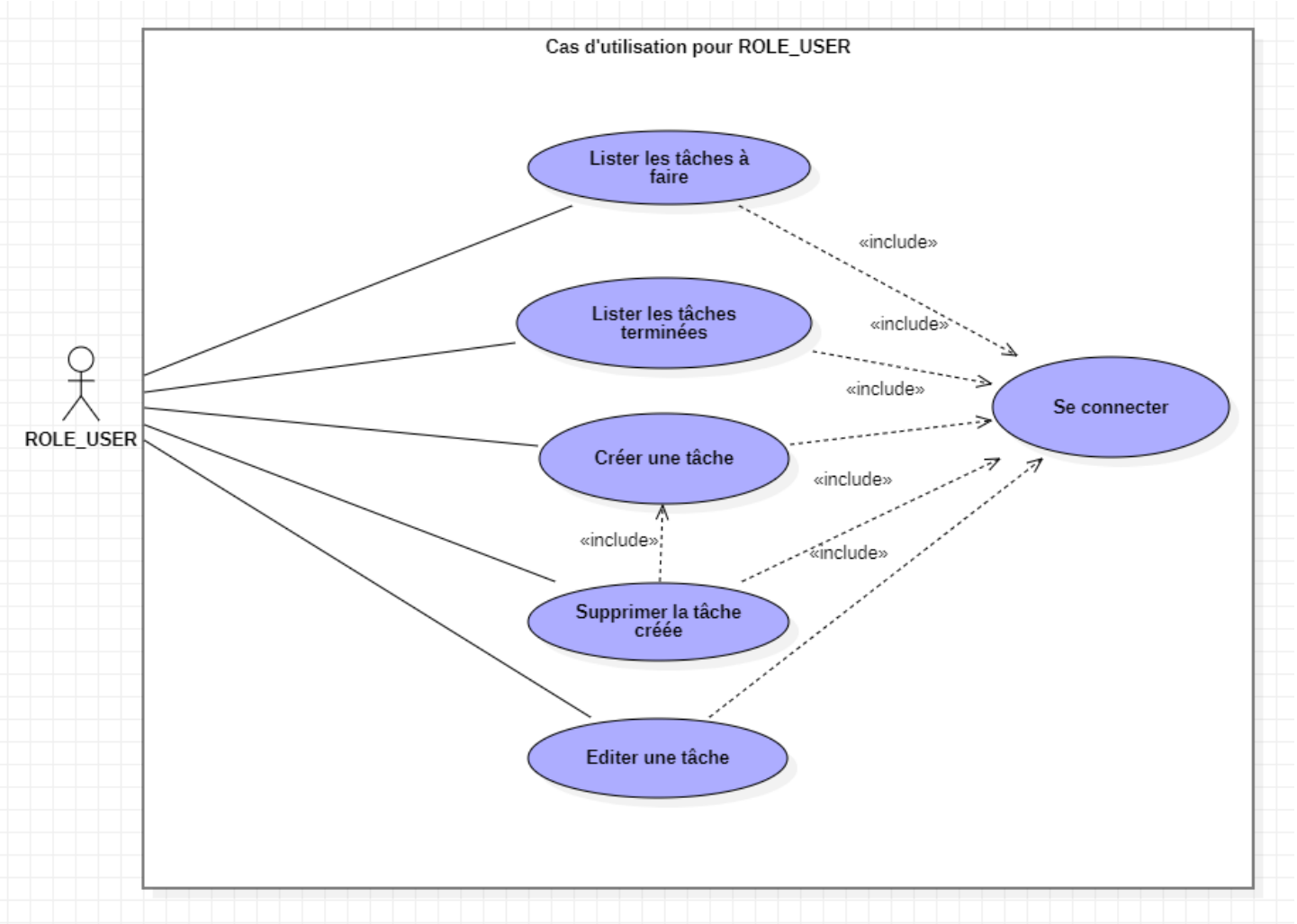
Low: 0% to 50% Medium: 50% to 90% High: 90% to 100%

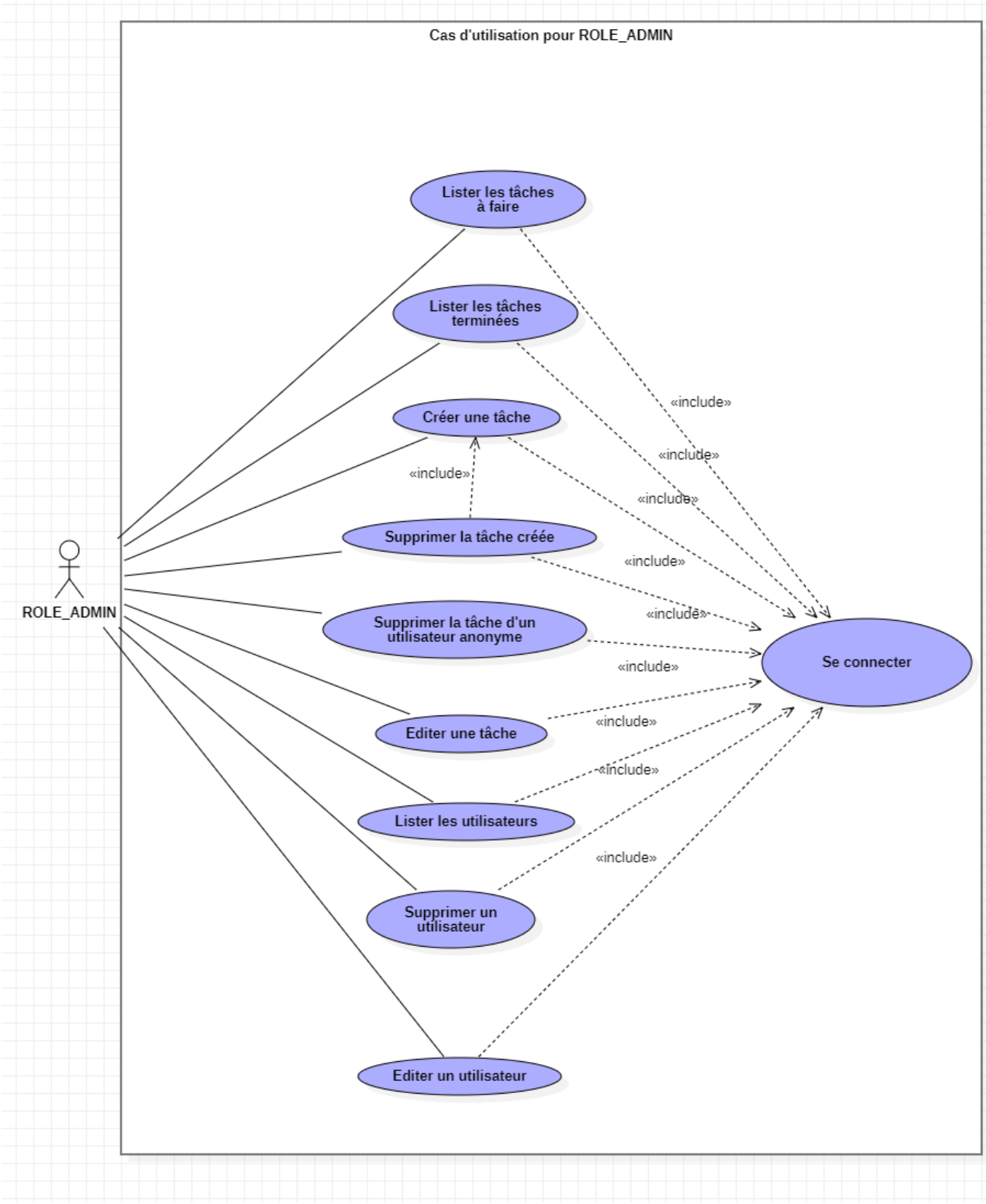
Generated by php-code-coverage 9.2.7 using PHP 7.4.24 and PHPUnit 9.5.9 at Mon Oct 11 18:54:31 UTC 2021.

5. Diagrammes UML

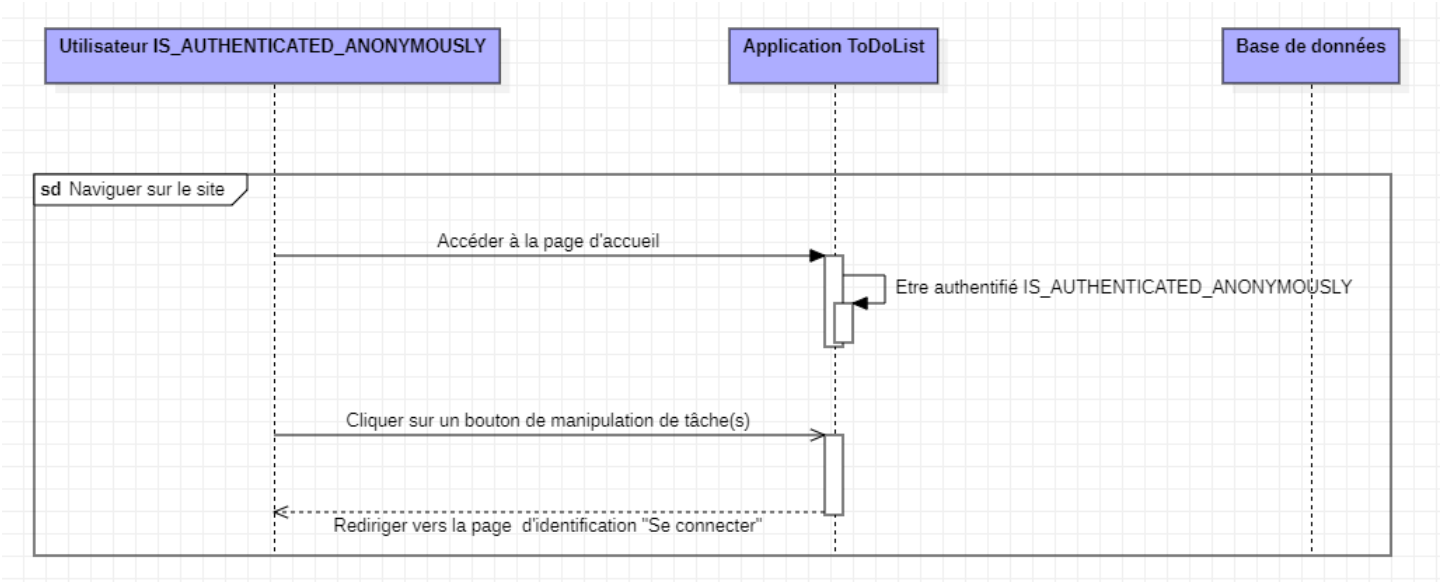
5.1 Cas d'utilisation

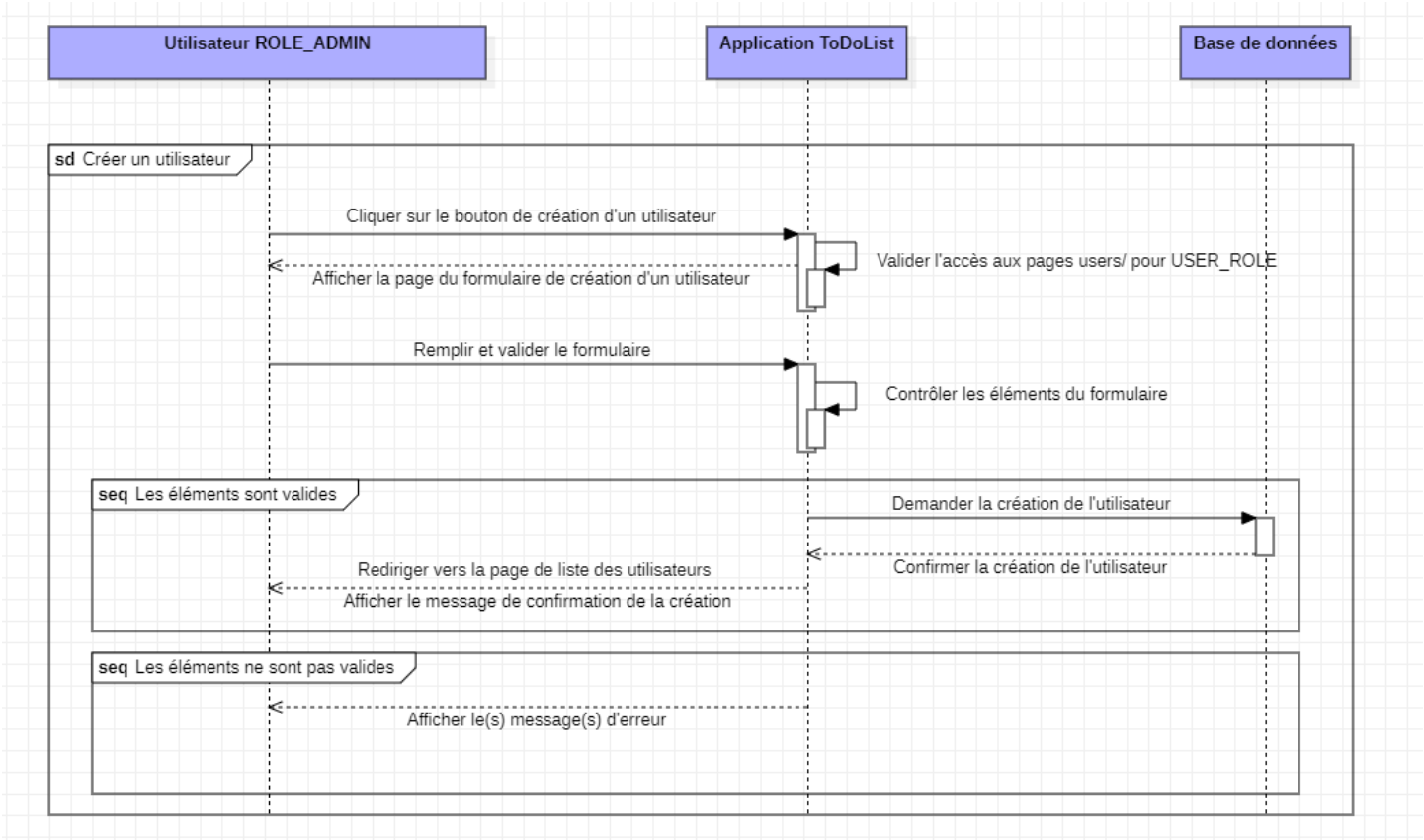


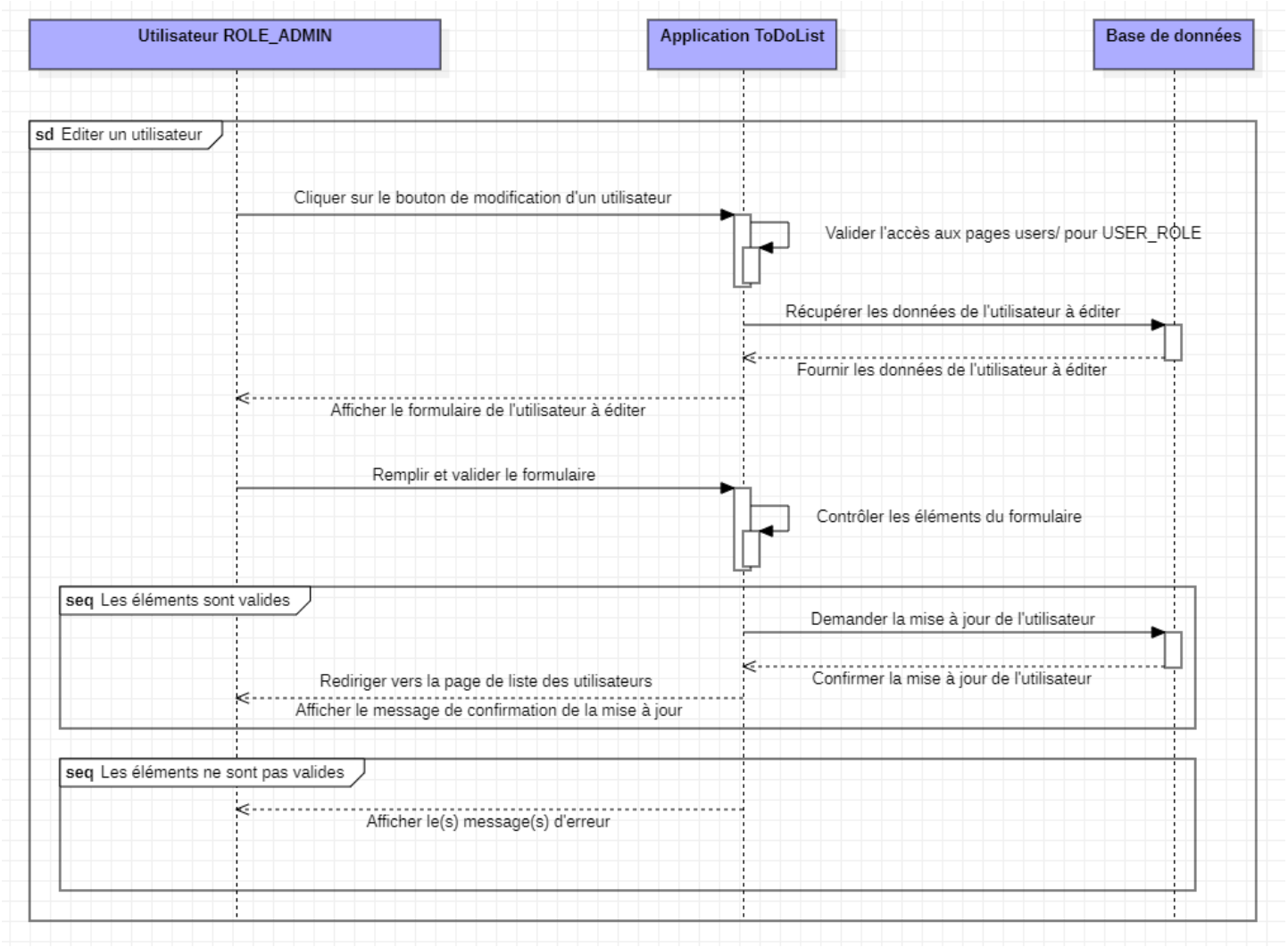

Cas d'utilisation pour ROLE_USER
ROLE_USER
Lister les tâches à faire
Lister les tâches terminées
Créer une tâche
Supprimer la tâche créée
Editer une tâche
Se connecter
«include»
«include»
«include»
«include»
«include»
«include»

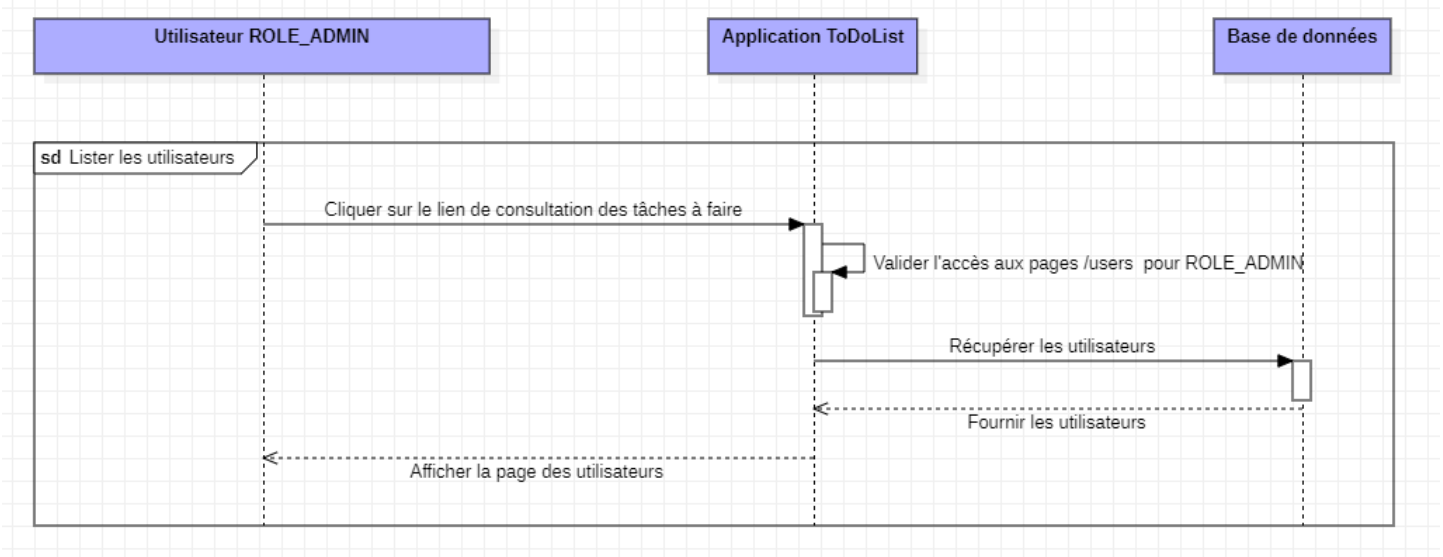

Cas d'utilisation pour ROLE_ADMIN
Lister les tâches à faire
Lister les tâches terminées
Créer une tâche
Supprimer la tâche créée
Supprimer la tâche d'un utilisateur anonyme
Editer une tâche
Lister les utilisateurs
Supprimer un utilisateur
Editer un utilisateur
Se connecter
ROLE_ADMIN
«include»
«include»
«include»
«include»
«include»
«include»
«include»
«include»
«include»
«include»

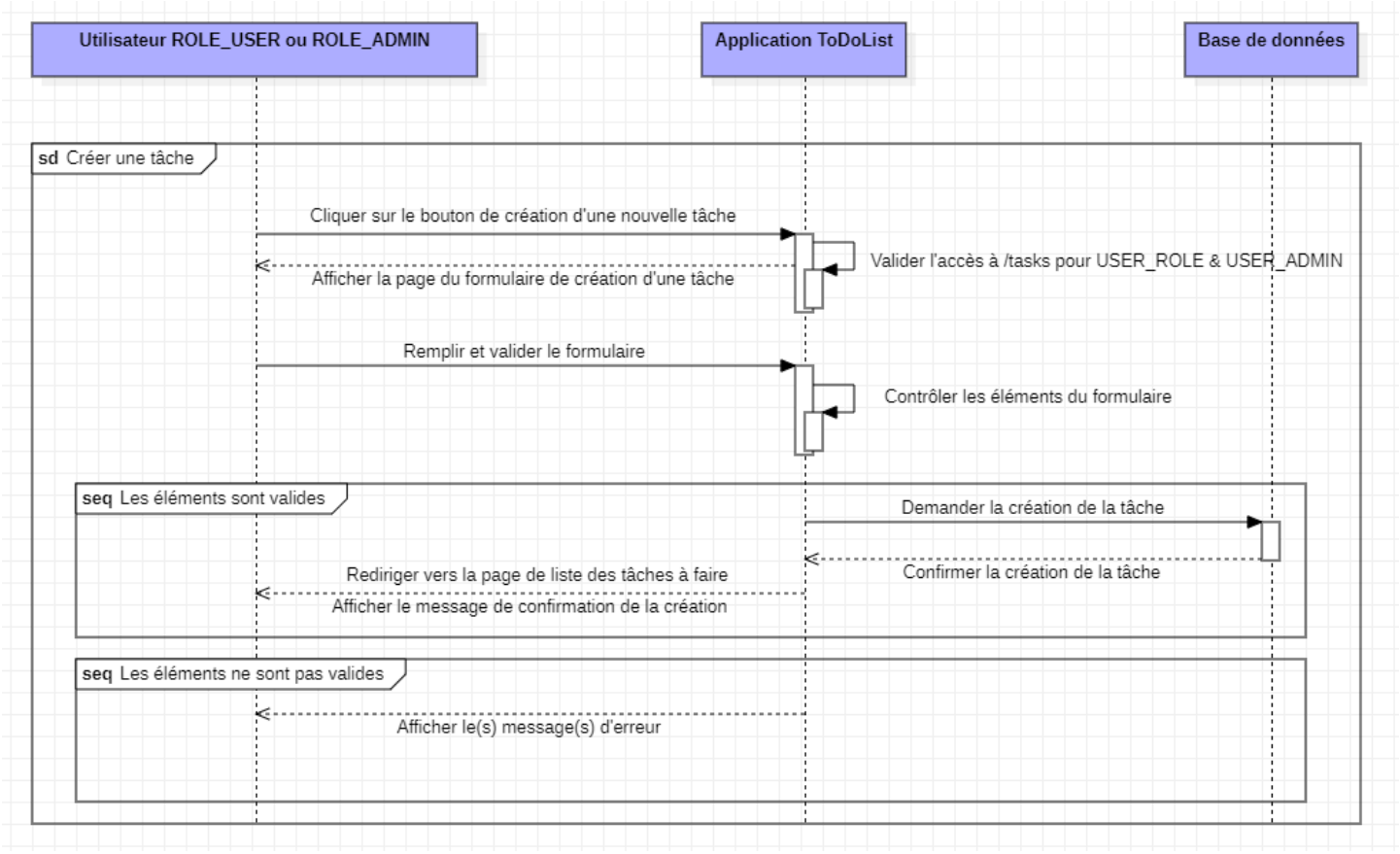
5.2 Séquence

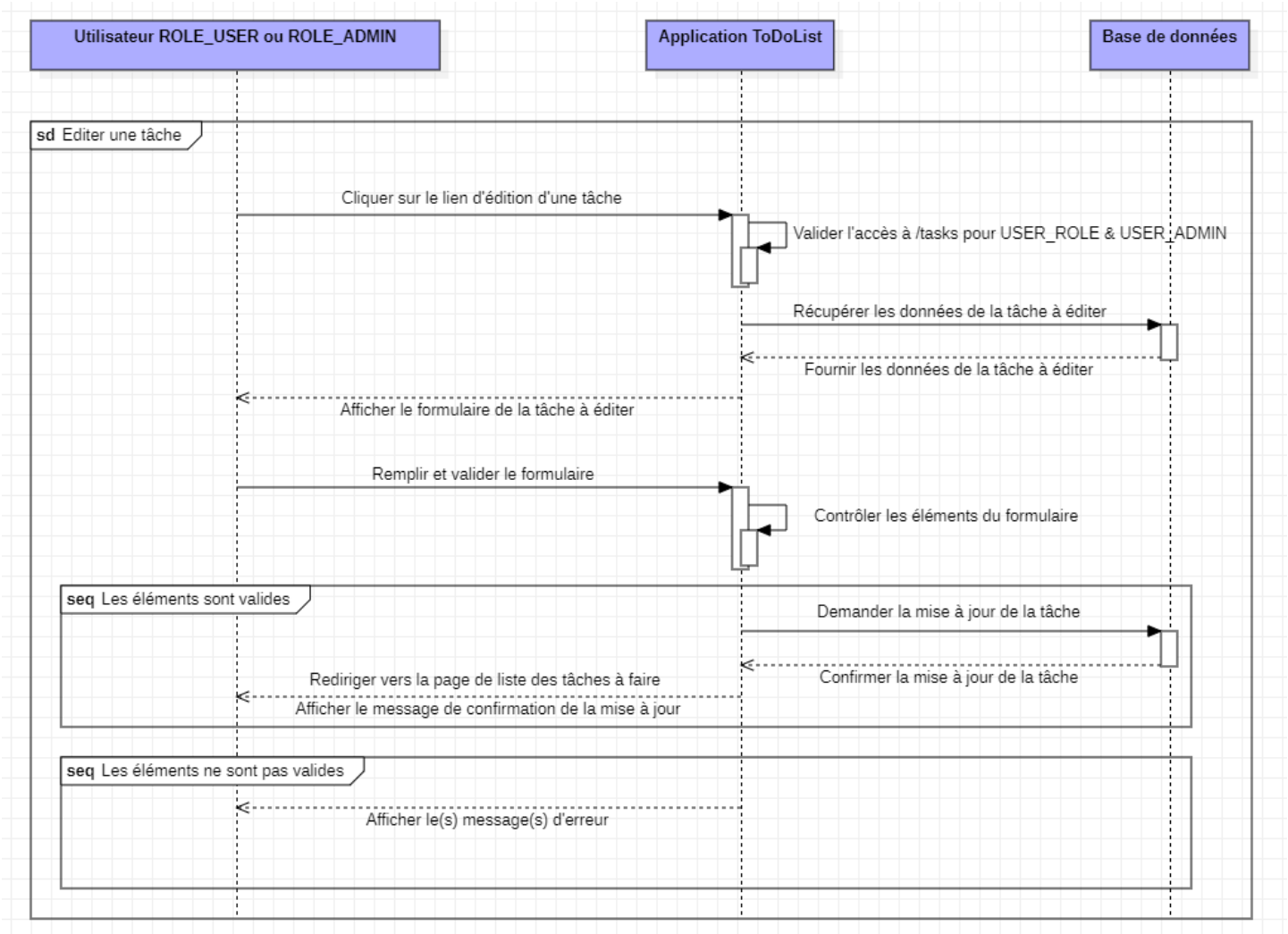

Utilisateur IS_AUTHENTICATED_ANONYMOUSLY
Application ToDoList
Base de données
sd Naviguer sur le site
Accéder à la page d'accueil
Etre authentifié IS_AUTHENTICATED_ANONYMOUSLY
Cliquer sur un bouton de manipulation de tâche(s)
Rediriger vers la page d'identification "Se connecter"

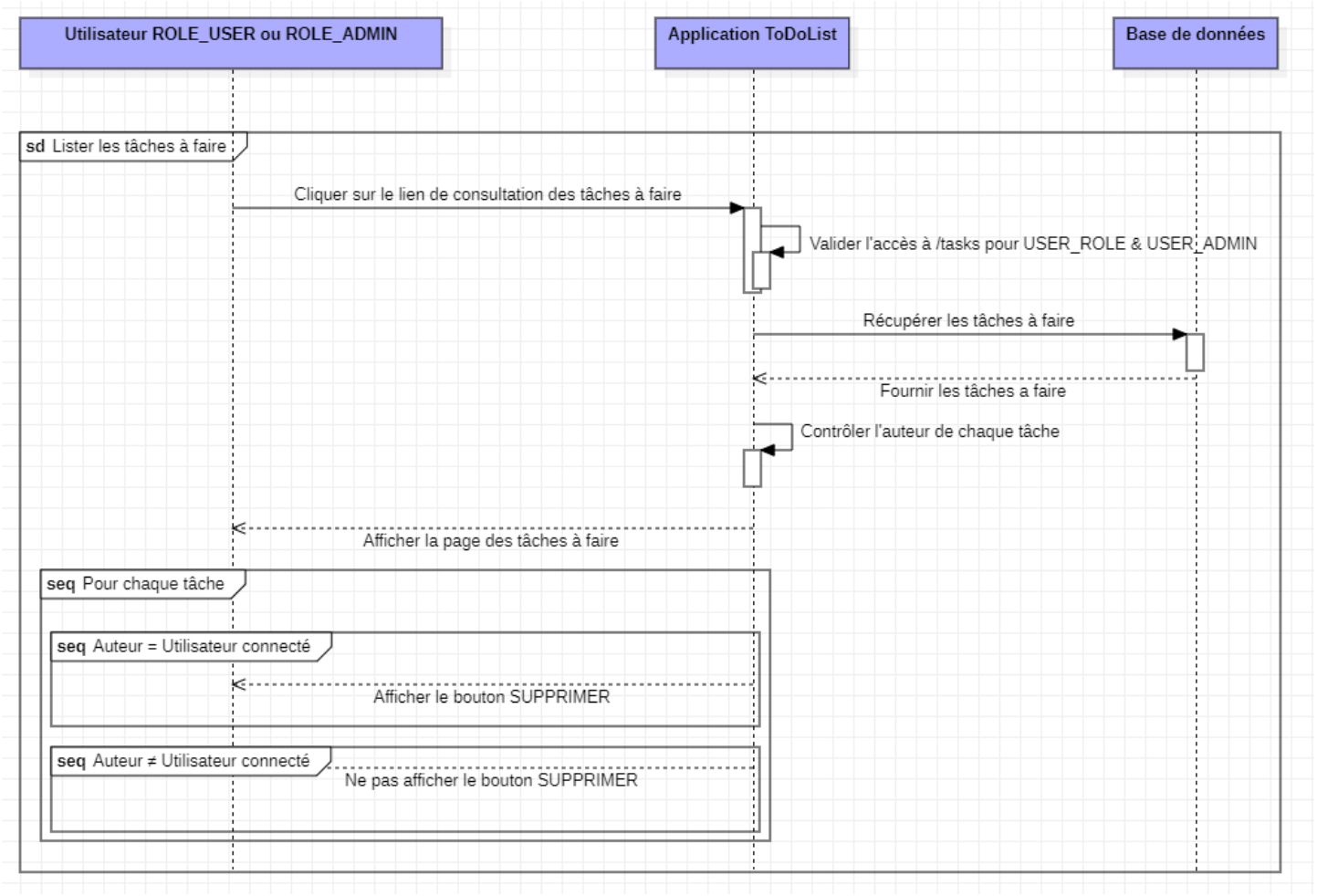

Utilisateur ROLE_ADMIN
Application ToDoList
Base de données
sd Créer un utilisateur
Cliquer sur le bouton de création d'un utilisateur
Valider l'accès aux pages users/ pour USER_ROLE
Afficher la page du formulaire de création d'un utilisateur
Remplir et valider le formulaire
Contrôler les éléments du formulaire
seq Les éléments sont valides
Demander la création de l'utilisateur
Confirmer la création de l'utilisateur
Rediriger vers la page de liste des utilisateurs
Afficher le message de confirmation de la création
seq Les éléments ne sont pas valides
Afficher le(s) message(s) d'erreur

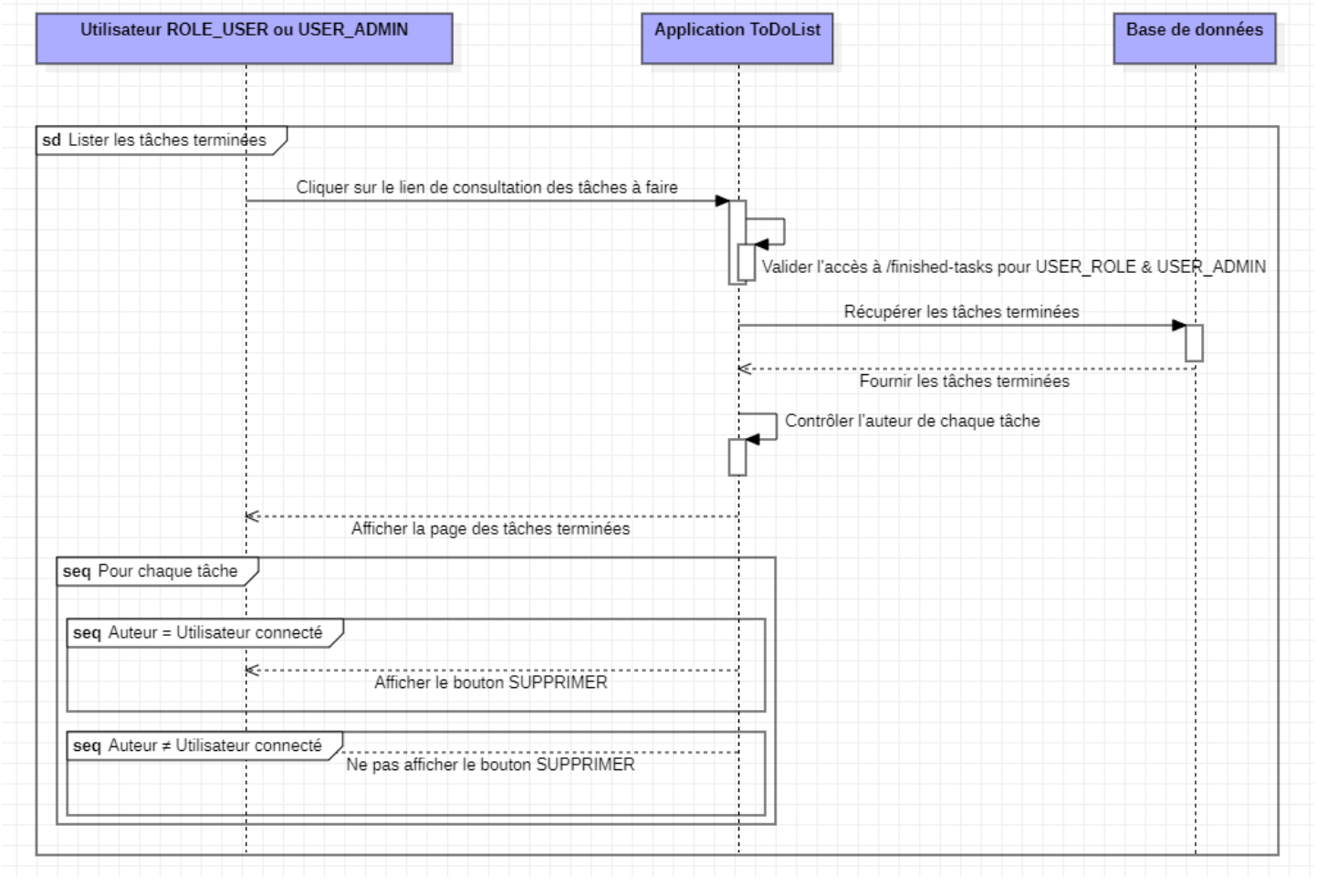

Utilisateur ROLE_ADMIN
Application ToDoList
Base de données
sd Editer un utilisateur
Cliquer sur le bouton de modification d'un utilisateur
Valider l'accès aux pages users/ pour USER_ROLE
Récupérer les données de l'utilisateur à éditer
Fournir les données de l'utilisateur à éditer
Afficher le formulaire de l'utilisateur à éditer
Remplir et valider le formulaire
Contrôler les éléments du formulaire
seq Les éléments sont valides
Demander la mise à jour de l'utilisateur
Confirmer la mise à jour de l'utilisateur
Rediriger vers la page de liste des utilisateurs
Afficher le message de confirmation de la mise à jour
seq Les éléments ne sont pas valides
Afficher le(s) message(s) d'erreur

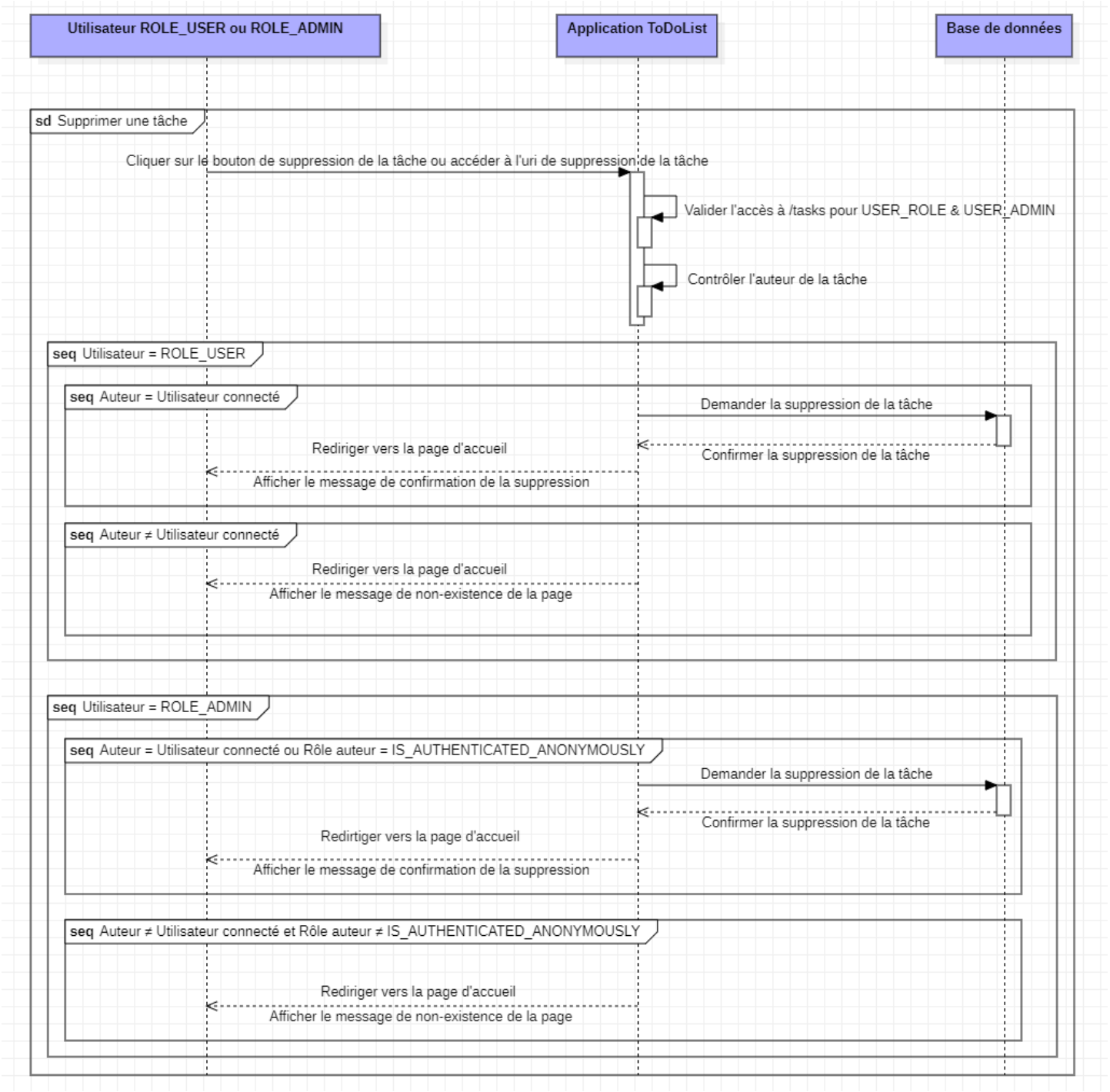

Utilisateur ROLE_ADMIN
Application ToDoList
Base de données
sd Lister les utilisateurs
Cliquer sur le lien de consultation des tâches à faire
Valider l'accès aux pages /users pour ROLE_ADMIN
Récupérer les utilisateurs
Fournir les utilisateurs
Afficher la page des utilisateurs


Utilisateur ROLE_USER ou ROLE_ADMIN
Application ToDoList
Base de données
sd Créer une tâche
Cliquer sur le bouton de création d'une nouvelle tâche
Valider l'accès à /tasks pour USER_ROLE & USER_ADMIN
Afficher la page du formulaire de création d'une tâche
Remplir et valider le formulaire
Contrôler les éléments du formulaire
seq Les éléments sont valides
Demander la création de la tâche
Confirmer la création de la tâche
Rediriger vers la page de liste des tâches à faire
Afficher le message de confirmation de la création
seq Les éléments ne sont pas valides
Afficher le(s) message(s) d'erreur

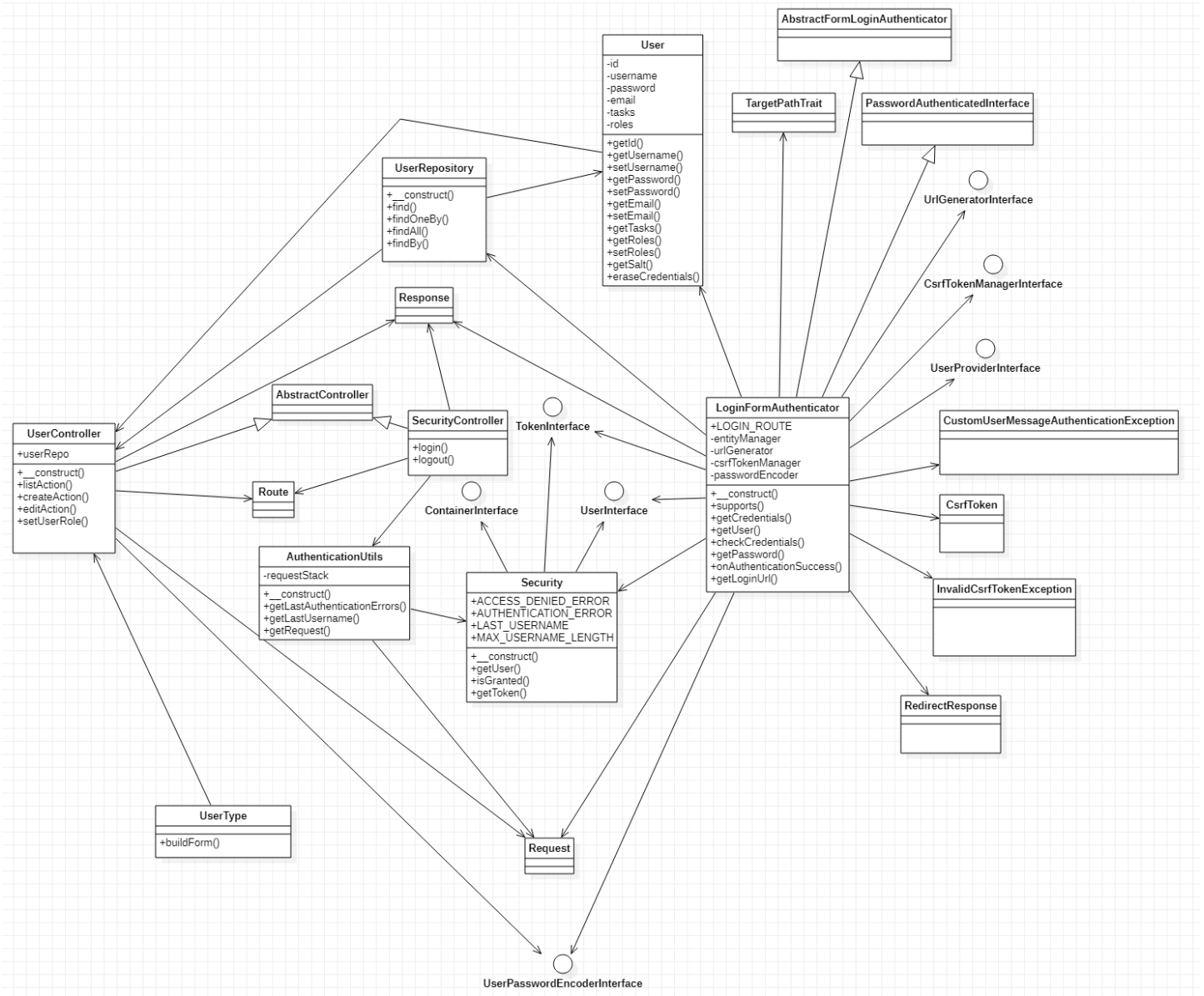

Utilisateur ROLE_USER ou ROLE_ADMIN
Application ToDoList
Base de données
sd Editer une tâche
Cliquer sur le lien d'édition d'une tâche
Valider l'accès à /tasks pour USER_ROLE & USER_ADMIN
Récupérer les données de la tâche à éditer
Fournir les données de la tâche à éditer
Afficher le formulaire de la tâche à éditer
Remplir et valider le formulaire
Contrôler les éléments du formulaire
seq Les éléments sont valides
Demander la mise à jour de la tâche
Confirmer la mise à jour de la tâche
Rediriger vers la page de liste des tâches à faire
Afficher le message de confirmation de la mise à jour
seq Les éléments ne sont pas valides
Afficher le(s) message(s) d'erreur


Utilisateur ROLE_USER ou ROLE_ADMIN
Application ToDoList
Base de données
sd Lister les tâches à faire
Cliquer sur le lien de consultation des tâches à faire
Valider l'accès à /tasks pour USER_ROLE & USER_ADMIN
Récupérer les tâches à faire
Fournir les tâches a faire
Contrôler l'auteur de chaque tâche
Afficher la page des tâches à faire
seq Pour chaque tâche
seq Auteur = Utilisateur connecté
Afficher le bouton SUPPRIMER
seq Auteur ≠ Utilisateur connecté
Ne pas afficher le bouton SUPPRIMER


Utilisateur ROLE_USER ou USER_ADMIN
Application ToDoList
Base de données
sd Lister les tâches terminées
Cliquer sur le lien de consultation des tâches à faire
Valider l'accès à /finished-tasks pour USER_ROLE & USER_ADMIN
Récupérer les tâches terminées
Fournir les tâches terminées
Contrôler l'auteur de chaque tâche
Afficher la page des tâches terminées
seq Pour chaque tâche
seq Auteur = Utilisateur connecté
Afficher le bouton SUPPRIMER
seq Auteur ≠ Utilisateur connecté
Ne pas afficher le bouton SUPPRIMER


Utilisateur ROLE_USER ou ROLE_ADMIN
Application ToDoList
Base de données
sd Supprimer une tâche
Cliquer sur le bouton de suppression de la tâche ou accéder à l'uri de suppression de la tâche
Valider l'accès à /tasks pour USER_ROLE & USER_ADMIN
Contrôler l'auteur de la tâche
seq Utilisateur = ROLE_USER
seq Auteur = Utilisateur connecté
Demander la suppression de la tâche
Confirmer la suppression de la tâche
Rediriger vers la page d'accueil
Afficher le message de confirmation de la suppression
seq Auteur ≠ Utilisateur connecté
Rediriger vers la page d'accueil
Afficher le message de non-existence de la page
seq Utilisateur = ROLE_ADMIN
seq Auteur = Utilisateur connecté ou Rôle auteur = IS_AUTHENTICATED_ANONYMOUSLY
Demander la suppression de la tâche
Confirmer la suppression de la tâche
Redirtiger vers la page d'accueil
Afficher le message de confirmation de la suppression
seq Auteur ≠ Utilisateur connecté et Rôle auteur ≠ IS_AUTHENTICATED_ANONYMOUSLY
Rediriger vers la page d'accueil
Afficher le message de non-existence de la page

5.3 Classe



5.4 Modèle physique de données