|  |  |  |
| --- | --- | --- |
| EPCF3 | 26 Septembre  2024 | |
| Dossier projet – Fresh ! L’application mobile Android, la conception et le développement d’une application multicouche répartie en intégrant les recommandations de sécurité. | | EXPERNET |

Table des matières

[I. Liste des compétences du référentiel qui sont couvertes par le projet 2](#_Toc156762393)

[II. Résumé du projet en anglais d'une longueur d’environ 5 à 10 lignes soit 100 à 120 mots, ou environ 600 caractères espaces non compris 3](#_Toc156762394)

[III. Cahier des charges, expression des besoins, ou spécifications fonctionnelles du projet 4](#_Toc156762395)

[IV. Spécifications fonctionnelles du projet 6](#_Toc156762396)

[V. Spécifications techniques du projet, élaborées par le candidat, y compris pour la sécurité 7](#_Toc156762397)

[VI. Réalisations du candidat comportant les extraits de code les plus significatifs, et en les argumentant, y compris pour la sécurité 11](#_Toc156762398)

[VII. Présentation du jeu d’essai élaboré par le candidat de la fonctionnalité la plus représentative (données en entrée, données attendues, données obtenues) 17](#_Toc156762399)

[VIII. Description de la veille, effectuée par le candidat durant le projet, sur les vulnérabilités de sécurité 21](#_Toc156762400)

[IX. Description d’une situation de travail ayant nécessité une recherche, effectuée par le candidat durant le projet 22](#_Toc156762401)

[X. Annexes 23](#_Toc156762402)

[Schéma de la base de donnée `symfapp\_fresh\_db` : 23](#_Toc156762403)

# Liste des compétences du référentiel qui sont couvertes par le projet

|  |  |
| --- | --- |
| Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité | Collaborer à la gestion d’un projet informatique et à l’organisation de l’environnement de développement |
| Concevoir une application |
| Développer des composants métier |
| Construire une application organisée en couches |
| Développer une application mobile |
| Préparer et exécuter les plans de tests d’une application |
| Préparer et exécuter le déploiement d’une application |

Le projet couvre obligatoirement les compétences suivantes :

* Collaborer à la gestion d’un projet informatique et à l’organisation de l’environnement de développement
* Concevoir une application
* Développer des composants métier
* Construire une application organisée en couches
* Développer une application mobile
* Préparer et exécuter les plans de tests d’une application
* Préparer et exécuter le déploiement d’une application

# Résumé du projet en anglais d'une longueur d’environ 5 à 10 lignes soit 100 à 120 mots, ou environ 600 caractères espaces non compris

Fresh is a mobile application that lets you create and manage shopping lists. There are two main categories: shopping to do and shopping already done. When you create an errand, you can set a date for it to be completed. You can add and modify product lines, indicating the product name, quantity and unit. Products can be marked as ‘taken’ to keep track of those still to be purchased. Once an errand has been completed, it can be marked as already done and will automatically be moved to the already done category with the date on which it was completed, simplifying the management of errands.

# Cahier des charges, expression des besoins, ou spécifications fonctionnelles du projet

Le présent cahier des charges définit les exigences et les spécifications pour le développement d'une application mobile de gestions de courses.

• Contexte

Fresh (Entreprise) souhaite mettre en place une solution permettant à n’importe qui, la possibilité de créer des listes de courses depuis son téléphone.

• Fonctionnalités

Les fonctionnalités permettent de savoir ce qui est fonctionnel pour l’utilisateur

1. L’application a une page d’accueil.

* Un bouton « Créer une course » est cliquable et permet de créer une course dans un pop-up.
* Listage des courses à faire.
* Chaque élément à une date à faire (sauf cas contraire)
* Chaque élément est cliquable qui permet d’ajouter la date à faire (si elle n’est pas encore attribuée) avec un pop-up ou afficher un pop-up récapitulatif de la course, où l’on peut modifier la course, manager les produits de la course et marquer la course comme « terminer »
* Un bouton « Voir les courses déjà effectuées » est cliquable et permet de voir la liste de courses ayant été terminées.

1. L’application a une page de management des produits de course

* Un bouton « Retour » est cliquable et permet de revenir à la page d’accueil.
* Un récapitulatif de la course est disponible avec la date de création de la course, la date à faire et la date quand la course est terminée (sauf cas contraire)
* Un bouton « Modifier » est cliquable et permet de modifier le nom de la course, la date à faire et de marquer la course comme « terminer »
* Listage des produits.
* Chaque élément affiche la date de création de la course
* Chaque élément qui n’a pas encore été pris est cliquable et permet d’afficher un pop-up qui contient un bouton « Modifier », qui permet de modifier la ligne du produit et un bouton « Marquer comme pris » qui permet de catégoriser le produit comme pris avec la date à laquelle il a été pris.
* Un bouton « Ajouter un produit » est cliquable et permet d’afficher un pop-up qui contient des champs libres pour indiquer le nom du produit, sa quantité et son unité (champ facultatif)
* Un bouton « Marquer comme terminer » qui permet de classer la course comme terminer avec la date à laquelle elle a été terminée.

• Technologies utilisées

* Développement de l’application
* Java
* Android SDK API 34

• Livrable et délais

* La plateforme complète et fonctionnelle doit être livrée conformément aux spécifications.
* La date de livraison prévue pour la plateforme est le 26/07/2024.

# Spécifications fonctionnelles du projet, élaborées par le candidat, y compris pour la sécurité

• Développement de l’application

* + Outil de version avec Git 2.45.1
  + Android Studio Koala

• Sécurité de l’application

* + L’application utilise un SDK à jour.
  + Il n’y a pas de fuite de mémoire
  + Les classes dans le code sont encapsulées.

Mission : Développer une application mobile android en répondant au cahier des charges

User-Story

# Spécifications techniques du projet, élaborées par le candidat, y compris pour la sécurité

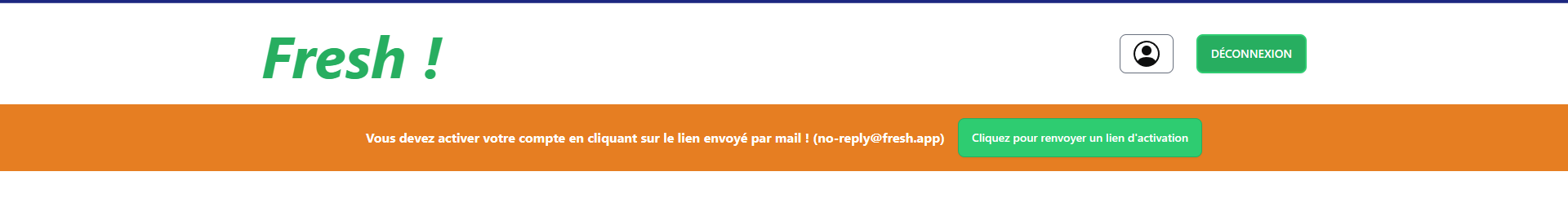
La persistance des données se fait à l’aide du cache en utilisant le « SharedPreferences ».

La classe WeakReference est utilisé pour le partage de données entre activités en sécurité.

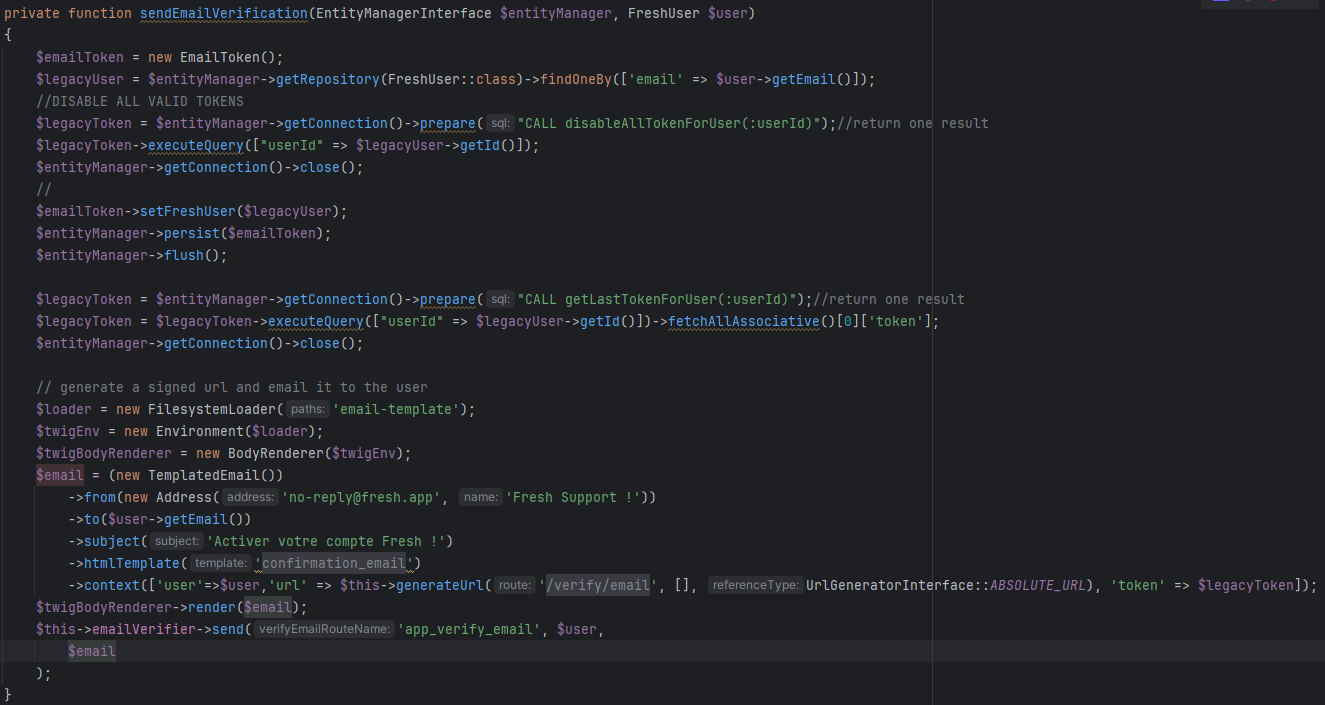
# Réalisations du candidat comportant les extraits de code les plus significatifs, et en les argumentant, y compris pour la sécurité

Développement de la vérification de compte :

L’utilisateur doit se créer un compte (sauf s’il possède déjà un) et se connecter.

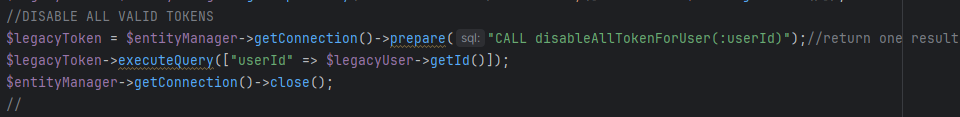
Il doit confirmer son compte pour pouvoir utiliser l’application.

Un courriel est envoyé à l’utilisateur, l’utilisateur peut cliquer sur le bouton.

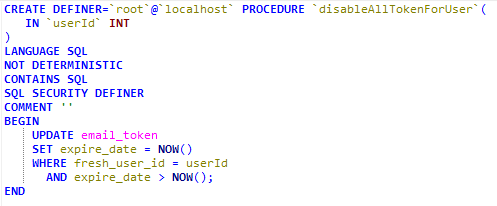


Un EmailToken est créé permettant l’utilisateur d’activer son compte sans se connecter.

Il faut d’abord désactiver tous ses EmailToken déjà existant.



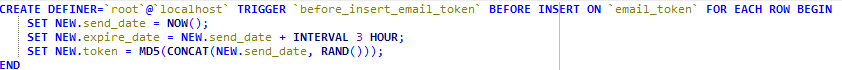
La procédure « disableAllTokenForUser(int userId) » est donc appelée avec une requête préparée (pour la sécurité).



Création de la procédure. (voir annexe 1)

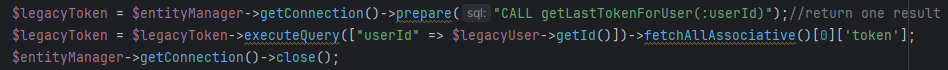
La procédure met toutes les « expire\_date » (date d’expiration) qui ont dépassé la date d’aujourd’hui, à la date d’aujourd’hui (la date d’expiration sera donc aujourd’hui), pour les lignes d’ « email\_token » ayant l’utilisateur. Une date d’expiration égale ou inférieure à la date d’aujourd’hui donnent des « email\_token » invalide qui ne permettent pas d’activer un compte utilisateur.

Lorsque qu’un « email\_token » est créé, un déclencheur s’exécute avant la persistance.

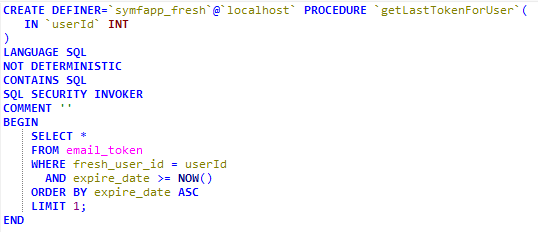


La date d’envoie est celle du jour d’exécution, la validité est de 3 heures et le token est généré automatiquement et hashé en MD5.

Puis on récupère le token d’« email\_token » généré par MySQL.



En faisant appel à la procédure « getLastTokenForUser(int userId) ».



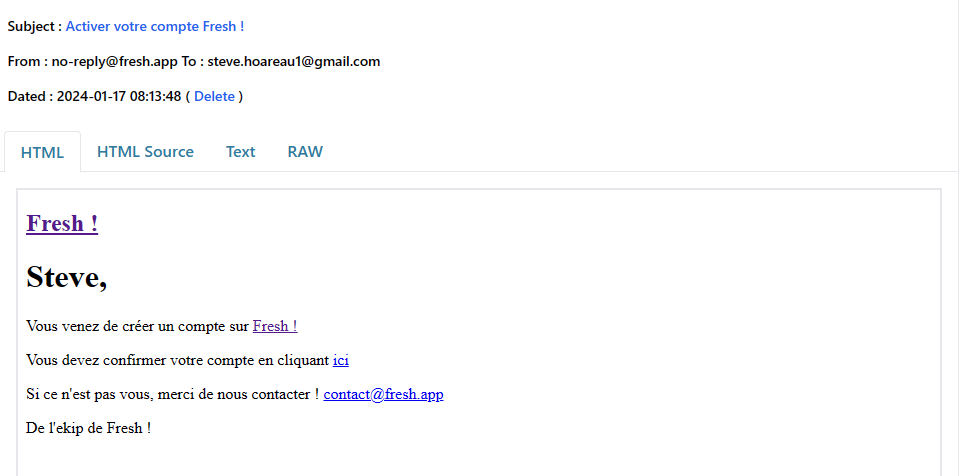
Création de la procédure. (voir annexe 1)

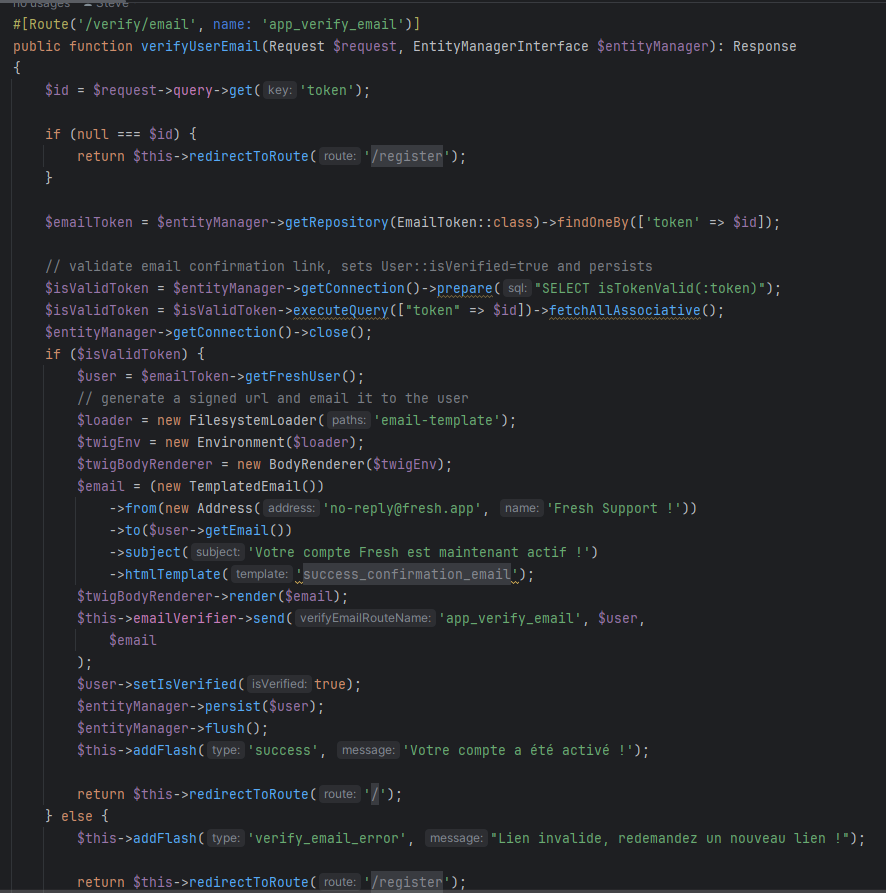
La procédure sélectionne le dernier « email\_token » généré pour l’utilisateur en prenant en compte que la date d’expiration doit être supérieure ou égale à la date d’aujourd’hui.

Rappel de la table « email\_token » :



Puis on envoie le courriel avec le token récupéré.

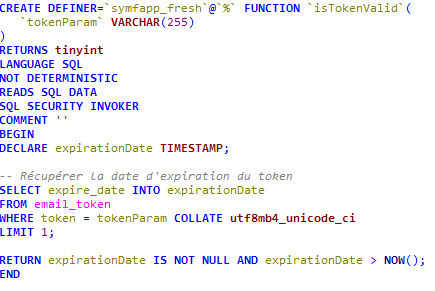


Redirige vers la route (avec l’url du site) /verify/email?token=f0f8dadf306b41ae96a73cd177187470

Lorsque l’utilisateur clique sur le lien, on vérifie que le token est valide.



On fait donc appel à la fonction « isTokenValid(varchar(255) tokenParam) » en requête préparée.



Création de la fonction. (voir annexe 1)

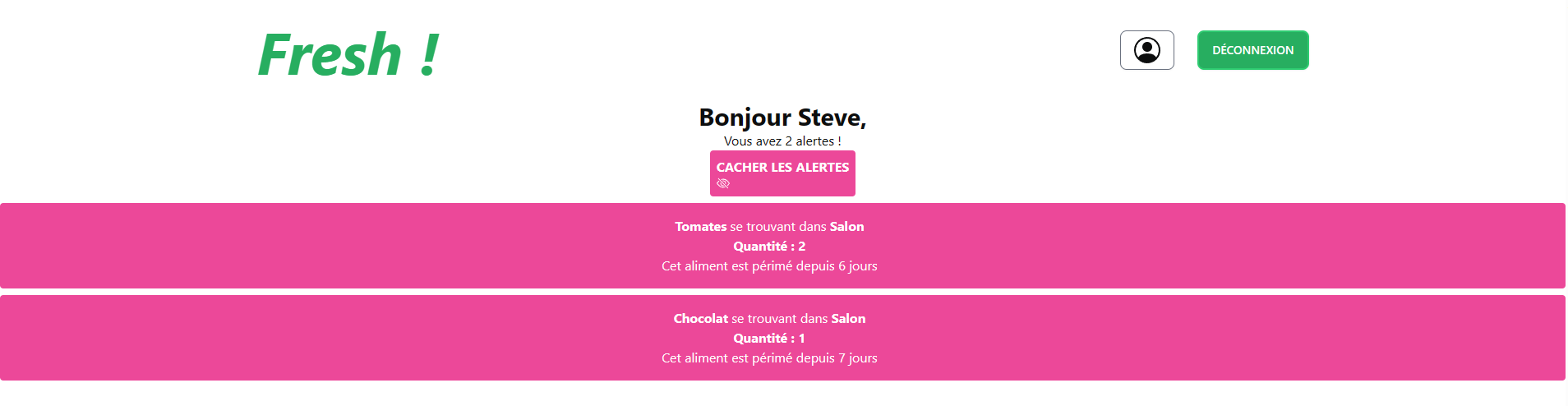
On vérifie juste pour le token (qui est unique) si sa date d’expiration est supérieur à la date d’aujourd’hui, ce qui renvoie vrai sinon faux en considérant que l’ « email\_token » est invalide.

Développement des alertes pour les aliments en phase de péremption :

L’utilisateur doit se créer un compte (sauf s’il possède déjà un) et se connecter.

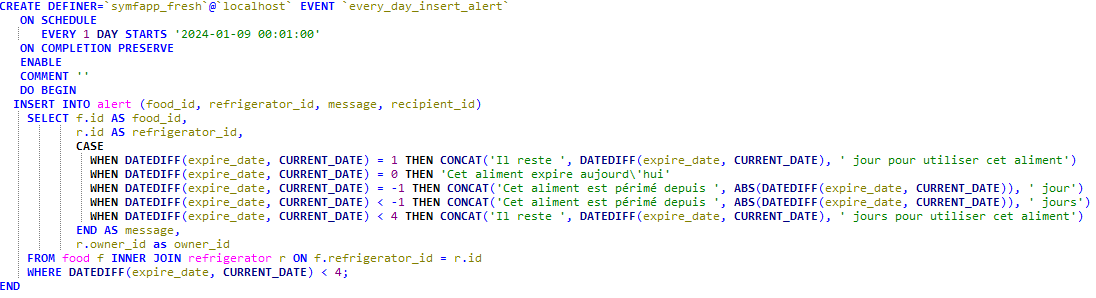
L’utilisateur doit avoir un compte vérifié.

L’utilisateur doit avoir au moins un frigo qui est créé et le frigo doit avoir au moins un aliment.



Des alertes s’affichent pendant au moins 10secondes.

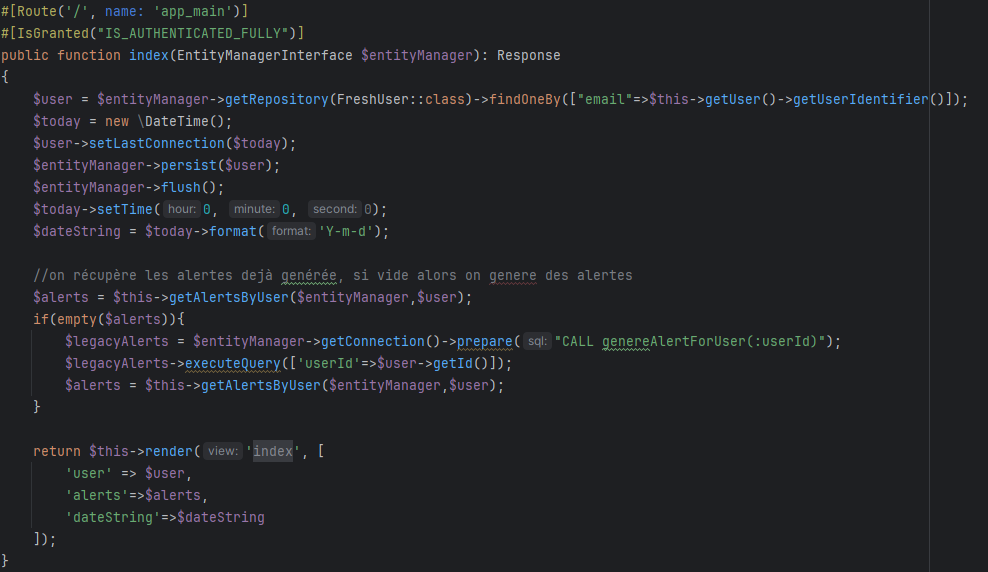
Tous les jours un événement sur la base de données MySQL est déclenché.



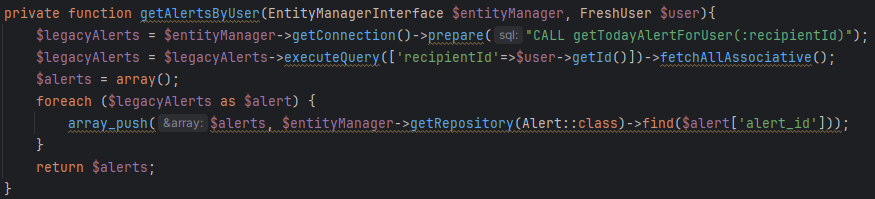
Création de l’événement.

L’événement va insérer automatiquement les aliments de tous les frigos pour ceux qui ont une date de péremption proche de 3 jours ou moins de la date d’aujourd’hui dans la table « alert ».

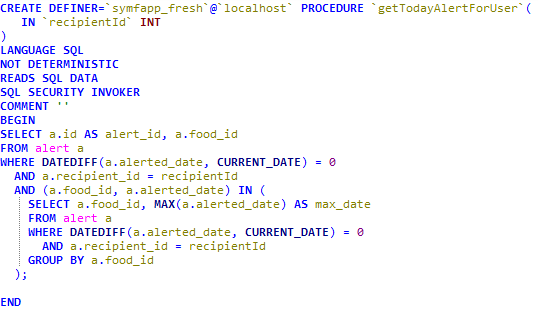
Et à chaque fois que l’utilisateur va sur la page d’accueil (page par défaut après un enregistrement ou une connexion)



On récupère les alertes pour l’utilisateur.



La procédure « getTodayAlertForUser(int recipientId) » est appelée en requête préparée.



Création de la procédure.

On récupère des alertes spécifiques dans la table « alert » pour l’utilisateur.

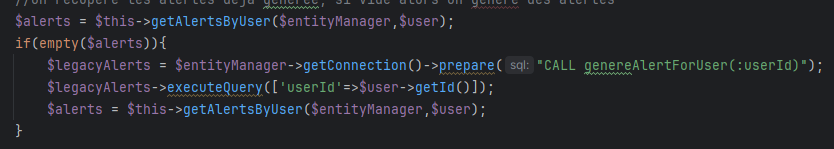
On veut les alertes qui concernent des aliments aujourd'hui (DATEDIFF(a.alerted\_date, CURRENT\_DATE) = 0).

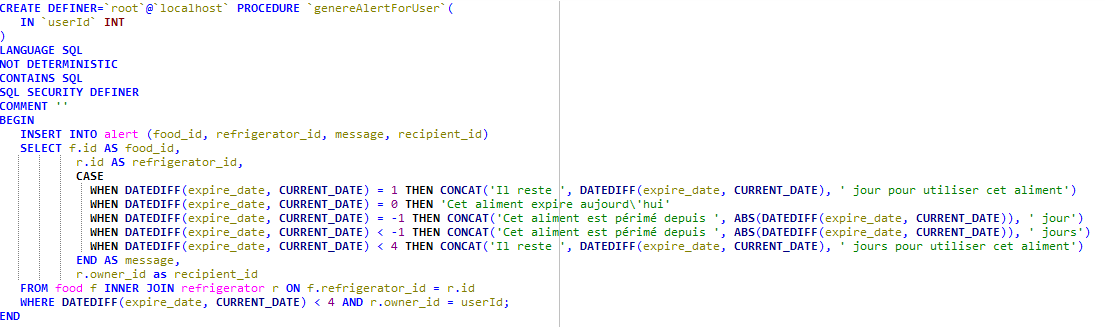
On cherche les alertes qui sont les dernières pour chaque type d'aliment. Donc, si la même alerte (même type d'aliment) a été émise plusieurs fois aujourd'hui, on récupère seulement la plus récente.

La requête utilise une sous-requête (SELECT a.food\_id, MAX(a.alerted\_date) AS max\_date ...) pour trouver la date la plus récente pour chaque type d'aliment, puis elle vérifie si l'alerte correspondante (type d'aliment et date) est également émise aujourd'hui pour l’utilisateur.

Ce qui permet d’avoir les alertes les plus récentes pour chaque aliment de l’utilisateur.

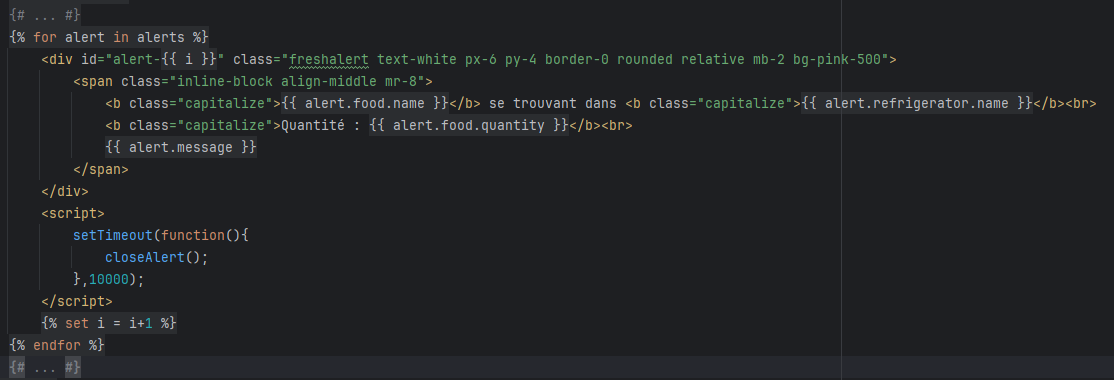
Les alertes sont générées à une heure précise. La procédure « genereAlertForUser(int userId ) » est appelée pour garantir que l’utilisateur a des alertes ou non.





Création de la procédure (Elle fait la même chose que l’événement).

Puis on va récupérer les alertes de l’utilisateur, et on les affiche.



# Présentation du jeu d’essai élaboré par le candidat de la fonctionnalité la plus représentative (données en entrée, données attendues, données obtenues)

Vérification du compte (email\_token):

*Rappel du trigger : before\_insert\_email\_token (Voir annexe 1)*

**BEGIN**

**SET** **NEW**.send\_date = **NOW**();

**SET** **NEW**.expire\_date = **NEW**.send\_date + **INTERVAL** 3 **HOUR**;

**SET** **NEW**.token = **MD5**(**CONCAT**(**NEW**.send\_date, **RAND**()));

**END**

* ***Données en entrée :***

L’identifiant d’un utilisateur.

**INSERT** **INTO** email\_token (`fresh\_user\_id`) **VALUES** (3);

* **Données attendues :**

Un entier positif.

* **Données obtenues :**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | fresh\_user\_id | send\_date | expire\_date | Token |
| 27 | 3 | 2024-01-17 15:38:33 | 2024-01-17 18:38:33 | 218b6ec2f5e8f2019fc13864bbeeb950 |

En ayant fait un **CALL** getLastTokenForUser(3);

A savoir que l’id est généré automatiquement (en incrémentation de 1).

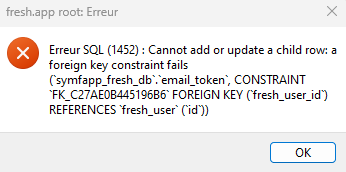
* **Si Données en entrée :**

L’identifiant d’un utilisateur qui n’existe pas.

**INSERT** **INTO** email\_token (`fresh\_user\_id`) **VALUES** (-1);

**Alors Données obtenues :**

Aucune (erreur)

****

Rappel de la fonction isTokenValid(varchar(255) tokenParam) : (voir annexe 1)

**BEGIN**

**DECLARE** expirationDate **TIMESTAMP**;

*-- Récupérer la date d'expiration du token*

**SELECT** expire\_date **INTO** expirationDate

**FROM** email\_token

**WHERE** token = tokenParam **COLLATE** **utf8mb4\_unicode\_ci**

**LIMIT** 1;

**RETURN** expirationDate **IS** **NOT** **NULL** **AND** expirationDate > **NOW**();

**END**

* **Données en entrée :**

« 218b6ec2f5e8f2019fc13864bbeeb950 »

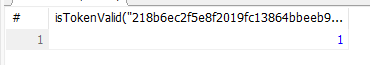
**SELECT** isTokenValid("218b6ec2f5e8f2019fc13864bbeeb950");

* **Données attendues :**

VARCHAR(255)

* **Données obtenues :**

Si la date d’expiration est supérieure à la date d’appel de la fonction :



Sinon :



* **Si données en entrée** :

« invalid\_token »

**SELECT** isTokenValid("invalid\_token");

**Alors données obtenue**:



Le token étant inexistant, la fonction retourne 0.

Insertion d’alertes :

*Rappel du trigger : before\_insert\_alert (voir annexe 1)*

**BEGIN**

**SET** **NEW**.alerted\_date = **NOW**();

**END**

*Rappel du trigger : after\_insert\_food*

**INSERT** **INTO** alert (food\_id, refrigerator\_id, message, recipient\_id)

**SELECT** f.id **AS** food\_id, r.id **AS** refrigerator\_id,

**CASE**

**WHEN** **DATEDIFF**(expire\_date, **CURRENT\_DATE**) = 1 **THEN** **CONCAT**('Il reste ', **DATEDIFF**(expire\_date, **CURRENT\_DATE**), ' jour pour utiliser cet aliment')

**WHEN** **DATEDIFF**(expire\_date, **CURRENT\_DATE**) = 0 **THEN** 'Cet aliment périme aujourd\'hui'

**WHEN** **DATEDIFF**(expire\_date, **CURRENT\_DATE**) = -1 **THEN** **CONCAT**('Cet aliment est périmé depuis ', **ABS**(**DATEDIFF**(expire\_date, **CURRENT\_DATE**)), ' jour')

**WHEN** **DATEDIFF**(expire\_date, **CURRENT\_DATE**) < -1 **THEN** **CONCAT**('Cet aliment est périmé depuis ', **ABS**(**DATEDIFF**(expire\_date, **CURRENT\_DATE**)), ' jours') **WHEN** **DATEDIFF**(expire\_date, **CURRENT\_DATE**) < 4 **THEN** **CONCAT**('Il reste ', **DATEDIFF**(expire\_date, **CURRENT\_DATE**), ' jours pour utiliser cet aliment')

**END** **AS** message,

r.owner\_id **as** owner\_id

**FROM** food f **INNER** **JOIN** refrigerator r **ON** f.refrigerator\_id = r.id

**WHERE** **DATEDIFF**(expire\_date, **CURRENT\_DATE**) < 4 **AND** f.id = **NEW**.id;

* **Données en entrée :**

Un aliment.

**INSERT** **INTO** food (`refrigerator\_id`,`name`,`quantity`,`expire\_date`) **VALUES** (8,"Caramel",4, **NOW**() + **INTERVAL** 3 DAY);

* **Données attendues :**

Un aliment qui appartient à un frigo existant et ayant une date d’expiration qui est inférieur à 4 jours de la date d’insertion.

* **Données obtenues :**

Une alerte.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | food\_id | message | alerted\_date | refrigerator\_id | recipient\_id |
| 61 | 32 | « Il reste 3 jours pour utiliser cet aliment » | 2024-01-18 13:30:41 | 8 | 2 |

# Description de la veille, effectuée par le candidat durant le projet, sur les vulnérabilités de sécurité

Toutes les sources de la veille de sécurité sur les vulnérabilités de la plateforme ont été vérifiée sur le site <https://owasp.org/Top10>.

A01:2021 – Contrôles d'accès défaillants

Pour réduire la vulnérabilité d’accès défaillants (par changement d’url, par api, sans authentification), on a rajouté une condition dans chacune des routes de Symfony (sauf login, register) pour vérifier si l’utilisateur qui fait la requête est identifié.

Pour le changement d’url, Symfony traite par défaut une URL invalide.

La modification ou suppression d’éléments dans la base de données depuis Symfony est sécurisé par un token (CSRF-Token) et par des requêtes préparées.

A02:2021 – Défaillances cryptographiques

Pour la vulnérabilité des défaillances cryptographiques il n’y a eu aucun traitement car les serveurs utilisés sont consacrés uniquement pour le développement et non la mise en production du projet.

A03:2021 – Injection

L’injection SQL est traitée par défaut par Symfony (composant Doctrine) en utilisant des requêtes préparées pour chaque traitement sur la base de données.

Les requêtes pour appeler les procédures/fonctions, sont des requêtes préparées.

A04:2021 – Conception non sécurisée

Le code de la plateforme est un code compilé, testé et fonctionnel.

A05:2021 – Mauvaise configuration de sécurité

La base de données peut être vulnérable si une personne a accès aux fichiers environnement de l’application. Cette vulnérabilité peut être réglée en délocalisant les identifiants de connexion.

A06:2021 – Composants vulnérables et obsolètes

Tous les composants sont à jour et ont été vérifiés par leurs auteurs (avec test d’intégrité, test unitaire avec une plateforme telle que Jenkins), mais on peut toujours trouver une faille.

En conclusion, la plateforme et sa base de données respecte la plupart des règles de sécurités et empêche-la plupart des failles. Les failles restantes doivent être régler avant la mise en production finale du projet.

Le projet contient des fichiers de journalisation permettant d’avoir un historique des requêtes (n’incluant pas les requêtes SQL).

# Description d’une situation de travail ayant nécessité une recherche, effectuée par le candidat durant le projet

Le site web [Stack Overflow](https://stackoverflow.com/) a été utilisé pour la majorité des recherches effectuées. La situation de travail pouvant justifier cette utilisation est pour automatiser la génération de token pour la vérification d’un compte utilisateur. Pour le trigger `before\_insert\_email\_token`, la fonction RAND() a été trouvée.

L’intelligence artificielle [Chat-GPT](https://chat.openai.com/) a été utilisé pour les différents blocages de code.

La documentation [daisyUi](https://daisyui.com/components/) a été utilisé pour trouver les bonnes classes a ajouté pour le front-end de l’application.

La documentation [Symfony](https://symfony.com/) a été utilisé pour l’envoie de mail et les différents blocages de code.

# Annexes

## 1. Schéma de la base de donnée `symfapp\_fresh\_db` :

-- --------------------------------------------------------

-- Hôte: 127.0.0.1

-- Version du serveur: 8.2.0 - MySQL Community Server - GPL

-- SE du serveur: Win64

-- HeidiSQL Version: 12.6.0.6765

-- --------------------------------------------------------

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET NAMES utf8 \*/;

/\*!50503 SET NAMES utf8mb4 \*/;

/\*!40103 SET @OLD\_TIME\_ZONE=@@TIME\_ZONE \*/;

/\*!40103 SET TIME\_ZONE='+00:00' \*/;

/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;

/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;

/\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;

-- Listage de la structure de la base pour symfapp\_fresh\_db

CREATE DATABASE IF NOT EXISTS `symfapp\_fresh\_db` /\*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci \*/ /\*!80016 DEFAULT ENCRYPTION='N' \*/;

USE `symfapp\_fresh\_db`;

CREATE USER IF NOT EXISTS `symfapp\_fresh`@`%` IDENTIFIED BY "(b-[D3GArJh\*qmO8";

GRANT USAGE ON \*.\* TO `symfapp\_fresh`@`%`;

GRANT ALL PRIVILEGES ON `symfapp\\_fresh\\_%`.\* TO `symfapp\_fresh`@`%`;

GRANT ALL PRIVILEGES ON `symfapp\\_fresh\\_db`.\* TO `symfapp\_fresh`@`%`;

-- Listage de la structure de la table symfapp\_fresh\_db. alert

CREATE TABLE IF NOT EXISTS `alert` (

`id` int NOT NULL AUTO\_INCREMENT,

`food\_id` int NOT NULL,

`message` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`alerted\_date` datetime NOT NULL,

`refrigerator\_id` int NOT NULL,

`recipient\_id` int NOT NULL,

PRIMARY KEY (`id`),

KEY `IDX\_17FD46C1BA8E87C4` (`food\_id`),

KEY `IDX\_17FD46C1915EAEB` (`refrigerator\_id`),

KEY `IDX\_17FD46C1E92F8F78` (`recipient\_id`),

CONSTRAINT `FK\_17FD46C1915EAEB` FOREIGN KEY (`refrigerator\_id`) REFERENCES `refrigerator` (`id`),

CONSTRAINT `FK\_17FD46C1BA8E87C4` FOREIGN KEY (`food\_id`) REFERENCES `food` (`id`),

CONSTRAINT `FK\_17FD46C1E92F8F78` FOREIGN KEY (`recipient\_id`) REFERENCES `fresh\_user` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=62 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- Listage des données de la table symfapp\_fresh\_db.alert : ~10 rows (environ)

INSERT INTO `alert` (`id`, `food\_id`, `message`, `alerted\_date`, `refrigerator\_id`, `recipient\_id`) VALUES

(10, 1, 'Il reste 1 jour pour utiliser cet aliment', '2024-01-10 09:25:00', 2, 2),

(11, 2, 'Cet aliment expire aujourd\'hui', '2024-01-10 09:25:00', 2, 2),

(13, 1, 'Il reste 1 jour pour utiliser cet aliment', '2024-01-10 09:31:55', 2, 2),

(14, 2, 'Cet aliment expire aujourd\'hui', '2024-01-10 09:31:55', 2, 2),

(16, 1, 'Il reste 1 jour pour utiliser cet aliment', '2024-01-10 10:06:34', 2, 2),

(17, 2, 'Cet aliment expire aujourd\'hui', '2024-01-10 10:06:34', 2, 2),

(56, 29, 'Cet aliment expire aujourd\'hui', '2024-01-12 16:40:35', 7, 3),

(57, 1, 'Cet aliment est périmé depuis 5 jours', '2024-01-16 11:31:10', 2, 2),

(58, 2, 'Cet aliment est périmé depuis 6 jours', '2024-01-16 11:31:10', 2, 2),

(59, 1, 'Cet aliment est périmé depuis 6 jours', '2024-01-17 12:05:41', 2, 2),

(60, 2, 'Cet aliment est périmé depuis 7 jours', '2024-01-17 12:05:41', 2, 2);

-- Listage de la structure de la procédure symfapp\_fresh\_db. disableAllTokenForUser

DELIMITER //

CREATE PROCEDURE `disableAllTokenForUser`(IN `userId` INT)

BEGIN

UPDATE email\_token

SET expire\_date = NOW()

WHERE fresh\_user\_id = userId

AND expire\_date > NOW();

END//

DELIMITER ;

-- Listage de la structure de la table symfapp\_fresh\_db. email\_token

CREATE TABLE IF NOT EXISTS `email\_token` (

`id` int NOT NULL AUTO\_INCREMENT,

`fresh\_user\_id` int NOT NULL,

`send\_date` datetime NOT NULL,

`expire\_date` datetime NOT NULL,

`token` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `UNIQ\_C27AE0B45F37A13B` (`token`),

KEY `IDX\_C27AE0B445196B6` (`fresh\_user\_id`),

CONSTRAINT `FK\_C27AE0B445196B6` FOREIGN KEY (`fresh\_user\_id`) REFERENCES `fresh\_user` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=30 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- Listage des données de la table symfapp\_fresh\_db.email\_token : ~27 rows (environ)

INSERT INTO `email\_token` (`id`, `fresh\_user\_id`, `send\_date`, `expire\_date`, `token`) VALUES

(1, 2, '2024-01-09 11:36:32', '2024-01-09 14:36:32', '09933ab69f6eca709dc3b9214614cff4'),

(2, 3, '2024-01-09 17:02:32', '2024-01-09 20:02:32', '44c1527ff9a16275dfb1cf3b5d0a66c0'),

(3, 3, '2024-01-12 10:14:57', '2024-01-12 13:14:57', '877515c2f5eaac91164603211daa2656'),

(4, 3, '2024-01-16 10:44:42', '2024-01-16 13:44:42', 'f3561dc7ffbd022e32c71a681cab1dbe'),

(5, 3, '2024-01-16 10:49:29', '2024-01-16 13:49:29', '1bce0f7993717ebc07494fe40bc6ad09'),

(6, 3, '2024-01-16 10:52:46', '2024-01-16 13:52:46', 'f6d3644647239cbb17512eb9e552b352'),

(7, 3, '2024-01-16 10:53:43', '2024-01-16 13:53:43', '9583a2b99d267962b04404de99908d52'),

(8, 3, '2024-01-16 10:58:52', '2024-01-16 13:58:52', 'd12722ebe482b5823bce144ff80ed931'),

(9, 3, '2024-01-16 10:59:09', '2024-01-16 13:59:09', '6ee7fd49db38dcae5d08ae48c2311b76'),

(10, 3, '2024-01-16 11:00:01', '2024-01-16 14:00:01', '60db75d813f72702bfa2c8daccbd8270'),

(11, 3, '2024-01-16 11:03:37', '2024-01-16 14:03:37', 'c19feda2dd5216e595c1bad92e3ccc11'),

(12, 3, '2024-01-16 11:14:16', '2024-01-16 14:14:16', '938a463afca918d474e8ad75e4174951'),

(13, 3, '2024-01-16 11:16:09', '2024-01-16 14:16:09', '53dcc6994393e4c0243476855021c79f'),

(14, 3, '2024-01-16 11:19:12', '2024-01-16 14:19:12', '82180b512cedb91dce9ba91cba765da2'),

(15, 3, '2024-01-16 11:28:49', '2024-01-16 14:28:49', '2fb95caef0b32d8e05ca3c750e703e3e'),

(16, 3, '2024-01-17 11:00:10', '2024-01-17 14:00:10', '2262b4733bfcc10f293588b8200baf3d'),

(17, 3, '2024-01-17 11:03:24', '2024-01-17 14:03:24', '0a7c37769d6c876f3874b57ed2cab511'),

(18, 3, '2024-01-17 11:03:35', '2024-01-17 14:03:35', 'b9ce3266fdead8fbbd7bd1b8a43df326'),

(19, 3, '2024-01-17 11:04:37', '2024-01-17 14:04:37', '30a68c53b0a8c1bd13c55fd1e611224f'),

(20, 3, '2024-01-17 11:19:51', '2024-01-17 14:19:51', '1c7072b229bf4bde158c7ff31fe3ddb9'),

(21, 3, '2024-01-17 11:21:33', '2024-01-17 14:21:33', '7e16f0c71b49a65a24372480f13cbcfe'),

(22, 3, '2024-01-17 11:46:52', '2024-01-17 14:46:52', '0046a1cca4c8b37cf3cada63271d0043'),

(23, 3, '2024-01-17 11:47:58', '2024-01-17 14:47:58', '224e9673e8f558ff7d1025f91c08898c'),

(24, 3, '2024-01-17 11:57:35', '2024-01-17 14:57:35', '032a2ccf94e00d635579676eaef469e5'),

(25, 2, '2024-01-17 12:11:45', '2024-01-17 12:13:46', 'f0f8dadf306b41ae96a73cd177187470'),

(26, 2, '2024-01-17 12:13:46', '2024-01-17 15:13:46', 'e6ec56eabb3aec67e18d852574243604'),

(27, 3, '2024-01-17 15:38:33', '2024-01-17 11:58:33', '218b6ec2f5e8f2019fc13864bbeeb950');

-- Listage de la structure de l'évènement symfapp\_fresh\_db. every\_day\_insert\_alert

DELIMITER //

CREATE EVENT `every\_day\_insert\_alert` ON SCHEDULE EVERY 1 DAY STARTS '2024-01-09 00:01:00' ON COMPLETION PRESERVE ENABLE DO BEGIN

INSERT INTO alert (food\_id, refrigerator\_id, message, recipient\_id)

SELECT f.id AS food\_id,

r.id AS refrigerator\_id,

CASE

WHEN DATEDIFF(expire\_date, CURRENT\_DATE) = 1 THEN CONCAT('Il reste ', DATEDIFF(expire\_date, CURRENT\_DATE), ' jour pour utiliser cet aliment')

WHEN DATEDIFF(expire\_date, CURRENT\_DATE) = 0 THEN 'Cet aliment expire aujourd\'hui'

WHEN DATEDIFF(expire\_date, CURRENT\_DATE) = -1 THEN CONCAT('Cet aliment est périmé depuis ', ABS(DATEDIFF(expire\_date, CURRENT\_DATE)), ' jour')

WHEN DATEDIFF(expire\_date, CURRENT\_DATE) < -1 THEN CONCAT('Cet aliment est périmé depuis ', ABS(DATEDIFF(expire\_date, CURRENT\_DATE)), ' jours')

WHEN DATEDIFF(expire\_date, CURRENT\_DATE) < 4 THEN CONCAT('Il reste ', DATEDIFF(expire\_date, CURRENT\_DATE), ' jours pour utiliser cet aliment')

END AS message,

r.owner\_id as owner\_id

FROM food f INNER JOIN refrigerator r ON f.refrigerator\_id = r.id

WHERE DATEDIFF(expire\_date, CURRENT\_DATE) < 4;

END//

DELIMITER ;

-- Listage de la structure de la table symfapp\_fresh\_db. food

CREATE TABLE IF NOT EXISTS `food` (

`id` int NOT NULL AUTO\_INCREMENT,

`refrigerator\_id` int NOT NULL,

`name` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`quantity` int NOT NULL,

`adding\_date` datetime DEFAULT NULL,

`expire\_date` datetime NOT NULL,

PRIMARY KEY (`id`),

KEY `IDX\_D43829F7915EAEB` (`refrigerator\_id`),

CONSTRAINT `FK\_D43829F7915EAEB` FOREIGN KEY (`refrigerator\_id`) REFERENCES `refrigerator` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=30 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- Listage des données de la table symfapp\_fresh\_db.food : ~4 rows (environ)

INSERT INTO `food` (`id`, `refrigerator\_id`, `name`, `quantity`, `adding\_date`, `expire\_date`) VALUES

(1, 2, 'Tomates', 2, '2024-01-09 15:36:16', '2024-01-11 00:00:00'),

(2, 2, 'Chocolat', 1, '2024-01-09 16:57:11', '2024-01-10 00:00:00'),

(29, 7, 'Tablettes de chocolat', 2, '2024-01-12 16:40:35', '2024-01-12 00:00:00');

-- Listage de la structure de la table symfapp\_fresh\_db. food\_recipe\_in\_refrigerator

CREATE TABLE IF NOT EXISTS `food\_recipe\_in\_refrigerator` (

`id` int NOT NULL AUTO\_INCREMENT,

`refrigerator\_id` int NOT NULL,

`food\_id` int NOT NULL,

`quantity` int NOT NULL,

`recipe\_id` int NOT NULL,

`unit` varchar(20) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `UNIQ\_22D2F3CD915EAEB` (`refrigerator\_id`),

UNIQUE KEY `UNIQ\_22D2F3CDBA8E87C4` (`food\_id`),

KEY `IDX\_22D2F3CD59D8A214` (`recipe\_id`),

CONSTRAINT `FK\_22D2F3CD59D8A214` FOREIGN KEY (`recipe\_id`) REFERENCES `recipe` (`id`),

CONSTRAINT `FK\_22D2F3CD915EAEB` FOREIGN KEY (`refrigerator\_id`) REFERENCES `refrigerator` (`id`),

CONSTRAINT `FK\_22D2F3CDBA8E87C4` FOREIGN KEY (`food\_id`) REFERENCES `food` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- Listage des données de la table symfapp\_fresh\_db.food\_recipe\_in\_refrigerator : ~1 rows (environ)

INSERT INTO `food\_recipe\_in\_refrigerator` (`id`, `refrigerator\_id`, `food\_id`, `quantity`, `recipe\_id`, `unit`) VALUES

(1, 2, 2, 1, 6, NULL);

-- Listage de la structure de la table symfapp\_fresh\_db. food\_recipe\_not\_in\_refrigerator

CREATE TABLE IF NOT EXISTS `food\_recipe\_not\_in\_refrigerator` (

`id` int NOT NULL AUTO\_INCREMENT,

`recipe\_id` int DEFAULT NULL,

`name` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`quantity` int NOT NULL,

`unit` varchar(20) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`can\_be\_regroup` tinyint(1) DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `IDX\_B0D4BE2259D8A214` (`recipe\_id`),

CONSTRAINT `FK\_B0D4BE2259D8A214` FOREIGN KEY (`recipe\_id`) REFERENCES `recipe` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=20 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- Listage des données de la table symfapp\_fresh\_db.food\_recipe\_not\_in\_refrigerator : ~19 rows (environ)

INSERT INTO `food\_recipe\_not\_in\_refrigerator` (`id`, `recipe\_id`, `name`, `quantity`, `unit`, `can\_be\_regroup`) VALUES

(1, 6, 'Sel', 1, 'pincée(s)', 0),

(2, 9, 'lait', 1, 'l', 0),

(3, 9, 'chocolat', 1, 'tablette', 1),

(4, 10, 'curry', 6, 'pincées', 0),

(5, 10, 'sel', 1, 'pincée(s)', 0),

(6, 10, 'poivre', 1, 'pincée(s)', 0),

(7, 10, 'crème fraîche', 12, 'cuillères', 0),

(8, 10, 'oeufs entier', 6, 'pincée(s)', 0),

(9, 10, 'chorizo', 12, 'rondelles', 0),

(10, 10, 'dés de jambon', 50, 'g', 0),

(11, 10, 'gruyère ou de parmesan', 50, 'g', 0),

(12, 11, 'curry', 6, 'pincées', 0),

(13, 11, 'sel', 1, 'pincée(s)', 0),

(14, 11, 'poivre', 1, 'pincée(s)', 0),

(15, 11, 'crème fraîche', 12, 'cuillères', 0),

(16, 11, 'oeufs entier', 6, 'pincée(s)', 0),

(17, 11, 'chorizo', 12, 'rondelles', 0),

(18, 11, 'dés de jambon', 50, 'g', 0),

(19, 11, 'gruyère ou de parmesan', 50, 'g', 0);

-- Listage de la structure de la table symfapp\_fresh\_db. fresh\_user

CREATE TABLE IF NOT EXISTS `fresh\_user` (

`id` int NOT NULL AUTO\_INCREMENT,

`email` varchar(180) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`roles` json NOT NULL,

`password` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`is\_verified` tinyint(1) NOT NULL,

`firstname` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`name` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`register\_date` datetime DEFAULT NULL,

`last\_connection` datetime DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `UNIQ\_569E4F03E7927C74` (`email`)

) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- Listage des données de la table symfapp\_fresh\_db.fresh\_user : ~0 rows (environ)

INSERT INTO `fresh\_user` (`id`, `email`, `roles`, `password`, `is\_verified`, `firstname`, `name`, `register\_date`, `last\_connection`) VALUES

(2, 'steve.hoareau1@gmail.com', '[]', '$2y$13$wRVKSpzyfyw2U9m5Fa1eIeXtSzSseIOTC3C5H2kDsSp3loOkRVE9S', 1, 'Steve', 'HOAREAU', '2024-01-09 11:36:32', '2024-01-17 10:24:27'),

(3, 'jeanpaul@gmail.com', '[]', '$2y$13$6w8T8paeacpf3.bpQIYQnOcVNyAxhkKEXveZ3UokfJOx6CvBfixtW', 0, 'Jean', 'PAUL', '2024-01-09 17:02:32', '2024-01-09 17:02:32');

-- Listage de la structure de la procédure symfapp\_fresh\_db. genereAlertForUser

DELIMITER //

CREATE PROCEDURE `genereAlertForUser`(IN userId INT)

BEGIN

INSERT INTO alert (food\_id, refrigerator\_id, message, recipient\_id)

SELECT f.id AS food\_id,

r.id AS refrigerator\_id,

CASE

WHEN DATEDIFF(expire\_date, CURRENT\_DATE) = 1 THEN CONCAT('Il reste ', DATEDIFF(expire\_date, CURRENT\_DATE), ' jour pour utiliser cet aliment')

WHEN DATEDIFF(expire\_date, CURRENT\_DATE) = 0 THEN 'Cet aliment expire aujourd\'hui'

WHEN DATEDIFF(expire\_date, CURRENT\_DATE) = -1 THEN CONCAT('Cet aliment est périmé depuis ', ABS(DATEDIFF(expire\_date, CURRENT\_DATE)), ' jour')

WHEN DATEDIFF(expire\_date, CURRENT\_DATE) < -1 THEN CONCAT('Cet aliment est périmé depuis ', ABS(DATEDIFF(expire\_date, CURRENT\_DATE)), ' jours')

WHEN DATEDIFF(expire\_date, CURRENT\_DATE) < 4 THEN CONCAT('Il reste ', DATEDIFF(expire\_date, CURRENT\_DATE), ' jours pour utiliser cet aliment')

END AS message,

r.owner\_id as recipient\_id

FROM food f INNER JOIN refrigerator r ON f.refrigerator\_id = r.id

WHERE DATEDIFF(expire\_date, CURRENT\_DATE) < 4 AND r.owner\_id = userId;

END//

DELIMITER ;

-- Listage de la structure de la procédure symfapp\_fresh\_db. getFoodAlreadyExistForUser

DELIMITER //

CREATE PROCEDURE `getFoodAlreadyExistForUser`(

IN `foodName` VARCHAR(255),

IN `expireDate` DATETIME,

IN `userId` INT

)

SQL SECURITY INVOKER

BEGIN

DECLARE foodCount INT;

-- Compter le nombre d'aliments avec le même nom et la même date d'expiration pour l'utilisateur donné

SELECT COUNT(\*) INTO foodCount

FROM food f

INNER JOIN refrigerator r ON f.refrigerator\_id = r.id

WHERE r.owner\_id = userId AND f.name LIKE CONCAT('%', foodName, '%') AND f.expire\_date = expireDate;

-- Si au moins un aliment existe, afficher la liste des aliments

IF foodCount > 0 THEN

SELECT f.id, f.name, f.quantity, f.adding\_date, f.expire\_date

FROM food f

INNER JOIN refrigerator r ON f.refrigerator\_id = r.id

WHERE r.owner\_id = userId AND f.name LIKE CONCAT('%', foodName, '%') AND f.expire\_date = expireDate;

ELSE

SELECT 0;

END IF;

END//

DELIMITER ;

-- Listage de la structure de la procédure symfapp\_fresh\_db. getLastTokenForUser

DELIMITER //

CREATE PROCEDURE `getLastTokenForUser`(IN `userId` INT)

SQL SECURITY INVOKER

BEGIN

SELECT \*

FROM email\_token

WHERE fresh\_user\_id = userId

AND expire\_date >= NOW()

ORDER BY expire\_date ASC

LIMIT 1;

END//

DELIMITER ;

-- Listage de la structure de la procédure symfapp\_fresh\_db. getTodayAlertForUser

DELIMITER //

CREATE PROCEDURE `getTodayAlertForUser`(

IN `recipientId` INT

)

READS SQL DATA

SQL SECURITY INVOKER

BEGIN

SELECT a.id AS alert\_id, a.food\_id

FROM alert a

WHERE DATEDIFF(a.alerted\_date, CURRENT\_DATE) = 0

AND a.recipient\_id = recipientId

AND (a.food\_id, a.alerted\_date) IN (

SELECT a.food\_id, MAX(a.alerted\_date) AS max\_date

FROM alert a

WHERE DATEDIFF(a.alerted\_date, CURRENT\_DATE) = 0

AND a.recipient\_id = recipientId

GROUP BY a.food\_id

);

END//

DELIMITER ;

-- Listage de la structure de la fonction symfapp\_fresh\_db. isTokenValid

DELIMITER //

CREATE FUNCTION `isTokenValid`(

`tokenParam` VARCHAR(255)

) RETURNS tinyint

READS SQL DATA

SQL SECURITY INVOKER

BEGIN

DECLARE expirationDate TIMESTAMP;

-- Récupérer la date d'expiration du token

SELECT expire\_date INTO expirationDate

FROM email\_token

WHERE token = tokenParam COLLATE utf8mb4\_unicode\_ci

LIMIT 1;

RETURN expirationDate IS NOT NULL AND expirationDate > NOW();

END//

DELIMITER ;

-- Listage de la structure de la table symfapp\_fresh\_db. messenger\_messages

CREATE TABLE IF NOT EXISTS `messenger\_messages` (

`id` bigint NOT NULL AUTO\_INCREMENT,

`body` longtext CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci NOT NULL,

`headers` longtext CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci NOT NULL,

`queue\_name` varchar(190) CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci NOT NULL,

`created\_at` datetime NOT NULL COMMENT '(DC2Type:datetime\_immutable)',

`available\_at` datetime NOT NULL COMMENT '(DC2Type:datetime\_immutable)',

`delivered\_at` datetime DEFAULT NULL COMMENT '(DC2Type:datetime\_immutable)',

PRIMARY KEY (`id`),

KEY `IDX\_75EA56E0FB7336F0` (`queue\_name`),

KEY `IDX\_75EA56E0E3BD61CE` (`available\_at`),

KEY `IDX\_75EA56E016BA31DB` (`delivered\_at`)

) ENGINE=InnoDB AUTO\_INCREMENT=2 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- Listage des données de la table symfapp\_fresh\_db.messenger\_messages : ~0 rows (environ)

INSERT INTO `messenger\_messages` (`id`, `body`, `headers`, `queue\_name`, `created\_at`, `available\_at`, `delivered\_at`) VALUES

(1, 'O:36:\\"Symfony\\\\Component\\\\Messenger\\\\Envelope\\":2:{s:44:\\"\\0Symfony\\\\Component\\\\Messenger\\\\Envelope\\0stamps\\";a:1:{s:46:\\"Symfony\\\\Component\\\\Messenger\\\\Stamp\\\\BusNameStamp\\";a:1:{i:0;O:46:\\"Symfony\\\\Component\\\\Messenger\\\\Stamp\\\\BusNameStamp\\":1:{s:55:\\"\\0Symfony\\\\Component\\\\Messenger\\\\Stamp\\\\BusNameStamp\\0busName\\";s:21:\\"messenger.bus.default\\";}}}s:45:\\"\\0Symfony\\\\Component\\\\Messenger\\\\Envelope\\0message\\";O:51:\\"Symfony\\\\Component\\\\Mailer\\\\Messenger\\\\SendEmailMessage\\":2:{s:60:\\"\\0Symfony\\\\Component\\\\Mailer\\\\Messenger\\\\SendEmailMessage\\0message\\";O:39:\\"Symfony\\\\Bridge\\\\Twig\\\\Mime\\\\TemplatedEmail\\":5:{i:0;s:41:\\"registration/confirmation\_email.html.twig\\";i:1;N;i:2;a:3:{s:9:\\"signedUrl\\";s:172:\\"http://127.0.0.1:8000/verify/email?expires=1704704530&id=1&signature=n59oRzYYJrxYygwvrA%2BHbpw7SV6zWSQp4j6sjK50b%2FU%3D&token=cp85vmRR7gGrjv32m3etHjw1EjtJ63hRjQWDAmIDm7Y%3D\\";s:19:\\"expiresAtMessageKey\\";s:26:\\"%count% hour|%count% hours\\";s:20:\\"expiresAtMessageData\\";a:1:{s:7:\\"%count%\\";i:1;}}i:3;a:6:{i:0;N;i:1;N;i:2;N;i:3;N;i:4;a:0:{}i:5;a:2:{i:0;O:37:\\"Symfony\\\\Component\\\\Mime\\\\Header\\\\Headers\\":2:{s:46:\\"\\0Symfony\\\\Component\\\\Mime\\\\Header\\\\Headers\\0headers\\";a:3:{s:4:\\"from\\";a:1:{i:0;O:47:\\"Symfony\\\\Component\\\\Mime\\\\Header\\\\MailboxListHeader\\":5:{s:50:\\"\\0Symfony\\\\Component\\\\Mime\\\\Header\\\\AbstractHeader\\0name\\";s:4:\\"From\\";s:56:\\"\\0Symfony\\\\Component\\\\Mime\\\\Header\\\\AbstractHeader\\0lineLength\\";i:76;s:50:\\"\\0Symfony\\\\Component\\\\Mime\\\\Header\\\\AbstractHeader\\0lang\\";N;s:53:\\"\\0Symfony\\\\Component\\\\Mime\\\\Header\\\\AbstractHeader\\0charset\\";s:5:\\"utf-8\\";s:58:\\"\\0Symfony\\\\Component\\\\Mime\\\\Header\\\\MailboxListHeader\\0addresses\\";a:1:{i:0;O:30:\\"Symfony\\\\Component\\\\Mime\\\\Address\\":2:{s:39:\\"\\0Symfony\\\\Component\\\\Mime\\\\Address\\0address\\";s:18:\\"no-reply@fresh.app\\";s:36:\\"\\0Symfony\\\\Component\\\\Mime\\\\Address\\0name\\";s:14:\\"Fresh Suport !\\";}}}}s:2:\\"to\\";a:1:{i:0;O:47:\\"Symfony\\\\Component\\\\Mime\\\\Header\\\\MailboxListHeader\\":5:{s:50:\\"\\0Symfony\\\\Component\\\\Mime\\\\Header\\\\AbstractHeader\\0name\\";s:2:\\"To\\";s:56:\\"\\0Symfony\\\\Component\\\\Mime\\\\Header\\\\AbstractHeader\\0lineLength\\";i:76;s:50:\\"\\0Symfony\\\\Component\\\\Mime\\\\Header\\\\AbstractHeader\\0lang\\";N;s:53:\\"\\0Symfony\\\\Component\\\\Mime\\\\Header\\\\AbstractHeader\\0charset\\";s:5:\\"utf-8\\";s:58:\\"\\0Symfony\\\\Component\\\\Mime\\\\Header\\\\MailboxListHeader\\0addresses\\";a:1:{i:0;O:30:\\"Symfony\\\\Component\\\\Mime\\\\Address\\":2:{s:39:\\"\\0Symfony\\\\Component\\\\Mime\\\\Address\\0address\\";s:24:\\"steve.hoareau1@gmail.com\\";s:36:\\"\\0Symfony\\\\Component\\\\Mime\\\\Address\\0name\\";s:0:\\"\\";}}}}s:7:\\"subject\\";a:1:{i:0;O:48:\\"Symfony\\\\Component\\\\Mime\\\\Header\\\\UnstructuredHeader\\":5:{s:50:\\"\\0Symfony\\\\Component\\\\Mime\\\\Header\\\\AbstractHeader\\0name\\";s:7:\\"Subject\\";s:56:\\"\\0Symfony\\\\Component\\\\Mime\\\\Header\\\\AbstractHeader\\0lineLength\\";i:76;s:50:\\"\\0Symfony\\\\Component\\\\Mime\\\\Header\\\\AbstractHeader\\0lang\\";N;s:53:\\"\\0Symfony\\\\Component\\\\Mime\\\\Header\\\\AbstractHeader\\0charset\\";s:5:\\"utf-8\\";s:55:\\"\\0Symfony\\\\Component\\\\Mime\\\\Header\\\\UnstructuredHeader\\0value\\";s:25:\\"Please Confirm your Email\\";}}}s:49:\\"\\0Symfony\\\\Component\\\\Mime\\\\Header\\\\Headers\\0lineLength\\";i:76;}i:1;N;}}i:4;N;}s:61:\\"\\0Symfony\\\\Component\\\\Mailer\\\\Messenger\\\\SendEmailMessage\\0envelope\\";N;}}', '[]', 'default', '2024-01-08 08:02:10', '2024-01-08 08:02:10', NULL);

-- Listage de la structure de la table symfapp\_fresh\_db. recipe

CREATE TABLE IF NOT EXISTS `recipe` (

`id` int NOT NULL AUTO\_INCREMENT,

`owner\_id` int NOT NULL,

`name` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`create\_date` datetime NOT NULL,

`last\_cooking\_date` datetime DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `IDX\_DA88B1377E3C61F9` (`owner\_id`),

CONSTRAINT `FK\_DA88B1377E3C61F9` FOREIGN KEY (`owner\_id`) REFERENCES `fresh\_user` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=12 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- Listage des données de la table symfapp\_fresh\_db.recipe : ~4 rows (environ)

INSERT INTO `recipe` (`id`, `owner\_id`, `name`, `create\_date`, `last\_cooking\_date`) VALUES

(6, 2, 'Test', '2024-01-16 11:57:19', NULL),

(9, 2, 'Chocolat chaud', '2024-01-16 12:30:36', NULL),

(10, 2, 'Oeuf cocotte chorizo et jambon', '2024-01-16 12:36:05', NULL),

(11, 2, 'Oeuf cocotte chorizo et jambon', '2024-01-16 12:36:55', NULL);

-- Listage de la structure de la table symfapp\_fresh\_db. refrigerator

CREATE TABLE IF NOT EXISTS `refrigerator` (

`id` int NOT NULL AUTO\_INCREMENT,

`owner\_id` int NOT NULL,

`name` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`adding\_date` datetime DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `IDX\_4619AF357E3C61F9` (`owner\_id`),

CONSTRAINT `FK\_4619AF357E3C61F9` FOREIGN KEY (`owner\_id`) REFERENCES `fresh\_user` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=8 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- Listage des données de la table symfapp\_fresh\_db.refrigerator : ~2 rows (environ)

INSERT INTO `refrigerator` (`id`, `owner\_id`, `name`, `adding\_date`) VALUES

(2, 2, 'Salon', '2024-01-09 14:31:25'),

(7, 3, 'Salon', '2024-01-12 13:29:58');

-- Listage de la structure de le déclencheur symfapp\_fresh\_db. after\_insert\_food

SET @OLDTMP\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

DELIMITER //

CREATE TRIGGER `after\_insert\_food` AFTER INSERT ON `food` FOR EACH ROW BEGIN

INSERT INTO alert (food\_id, refrigerator\_id, message, recipient\_id)

SELECT f.id AS food\_id,

r.id AS refrigerator\_id,

CASE

WHEN DATEDIFF(expire\_date, CURRENT\_DATE) = 1 THEN CONCAT('Il reste ', DATEDIFF(expire\_date, CURRENT\_DATE), ' jour pour utiliser cet aliment')

WHEN DATEDIFF(expire\_date, CURRENT\_DATE) = 0 THEN 'Cet aliment expire aujourd\'hui'

WHEN DATEDIFF(expire\_date, CURRENT\_DATE) = -1 THEN CONCAT('Cet aliment est périmé depuis ', ABS(DATEDIFF(expire\_date, CURRENT\_DATE)), ' jour')

WHEN DATEDIFF(expire\_date, CURRENT\_DATE) < -1 THEN CONCAT('Cet aliment est périmé depuis ', ABS(DATEDIFF(expire\_date, CURRENT\_DATE)), ' jours')

WHEN DATEDIFF(expire\_date, CURRENT\_DATE) < 4 THEN CONCAT('Il reste ', DATEDIFF(expire\_date, CURRENT\_DATE), ' jours pour utiliser cet aliment')

END AS message,

r.owner\_id as owner\_id

FROM food f INNER JOIN refrigerator r ON f.refrigerator\_id = r.id

WHERE DATEDIFF(expire\_date, CURRENT\_DATE) < 4 AND f.id = NEW.id;

END//

DELIMITER ;

SET SQL\_MODE=@OLDTMP\_SQL\_MODE;

-- Listage de la structure de le déclencheur symfapp\_fresh\_db. before\_insert\_alert

SET @OLDTMP\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

DELIMITER //

CREATE TRIGGER `before\_insert\_alert` BEFORE INSERT ON `alert` FOR EACH ROW BEGIN

SET NEW.alerted\_date = NOW();

END//

DELIMITER ;

SET SQL\_MODE=@OLDTMP\_SQL\_MODE;

-- Listage de la structure de le déclencheur symfapp\_fresh\_db. before\_insert\_email\_token

SET @OLDTMP\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

DELIMITER //

CREATE TRIGGER `before\_insert\_email\_token` BEFORE INSERT ON `email\_token` FOR EACH ROW BEGIN

SET NEW.send\_date = NOW();

SET NEW.expire\_date = NEW.send\_date + INTERVAL 3 HOUR;

SET NEW.token = MD5(CONCAT(NEW.send\_date, RAND()));

END//

DELIMITER ;

SET SQL\_MODE=@OLDTMP\_SQL\_MODE;

-- Listage de la structure de le déclencheur symfapp\_fresh\_db. before\_insert\_food

SET @OLDTMP\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

DELIMITER //

CREATE TRIGGER `before\_insert\_food` BEFORE INSERT ON `food` FOR EACH ROW BEGIN

SET NEW.adding\_date = NOW();

END//

DELIMITER ;

SET SQL\_MODE=@OLDTMP\_SQL\_MODE;

-- Listage de la structure de le déclencheur symfapp\_fresh\_db. before\_insert\_fresh\_user

SET @OLDTMP\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

DELIMITER //

CREATE TRIGGER `before\_insert\_fresh\_user` BEFORE INSERT ON `fresh\_user` FOR EACH ROW BEGIN

SET NEW.register\_date = NOW();

SET NEW.last\_connection = NOW();

END//

DELIMITER ;

SET SQL\_MODE=@OLDTMP\_SQL\_MODE;

-- Listage de la structure de le déclencheur symfapp\_fresh\_db. before\_insert\_refrigerator

SET @OLDTMP\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

DELIMITER //

CREATE TRIGGER `before\_insert\_refrigerator` BEFORE INSERT ON `refrigerator` FOR EACH ROW BEGIN

SET NEW.adding\_date = NOW();

END//

DELIMITER ;

SET SQL\_MODE=@OLDTMP\_SQL\_MODE;

/\*!40103 SET TIME\_ZONE=IFNULL(@OLD\_TIME\_ZONE, 'system') \*/;

/\*!40101 SET SQL\_MODE=IFNULL(@OLD\_SQL\_MODE, '') \*/;

/\*!40014 SET FOREIGN\_KEY\_CHECKS=IFNULL(@OLD\_FOREIGN\_KEY\_CHECKS, 1) \*/;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40111 SET SQL\_NOTES=IFNULL(@OLD\_SQL\_NOTES, 1) \*/;