

23 Janvier

EPCF2

2024

Dossier projet – Fresh ! L’application WEB, la conception et
développement de la persistance des données en intégrant les
recommandations de sécurité.

EXPERNET

Table des matières

I.	Liste des compétences du référentiel qui sont couvertes par le projet	2
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
III.	Cahier des charges, expression des besoins, ou spécifications fonctionnelles du projet	4
IV.	Spécifications fonctionnelles du projet	6
V.	Spécifications techniques du projet, élaborées par le candidat, y compris pour la sécurité	7
VI.	Réalisations du candidat comportant les extraits de code les plus significatifs, et en les argumentant, y compris pour la sécurité	11
VII.	Présentation du jeu d'essai élaboré par le candidat de la fonctionnalité la plus représentative (données en entrée, données attendues, données obtenues)	17
VIII.	Description de la veille, effectuée par le candidat durant le projet, sur les vulnérabilités de sécurité ²¹	
IX.	Description d'une situation de travail ayant nécessité une recherche, effectuée par le candidat durant le projet	22
X.	Annexes	23
	Schéma de la base de donnée `symfapp_fresh_db` :	23

I. Liste des compétences du référentiel qui sont couvertes par le projet

Concevoir et développer la persistance des données en intégrant les recommandations de sécurité	Mettre en place une base de données
	Concevoir une base de données
	Développer des composants dans le langage d'une base de données

Le projet couvre obligatoirement les compétences suivantes :

- Mettre en place une base de données
- Concevoir une base de données
- Développer des composants dans le langage d'une base de données

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

Fresh is a web application designed to manage refrigerator inventory by tracking expiration dates for each food item, thereby reducing waste. The principle is straightforward: users log in, create up to 2 fridges, and can add 100 food items to each fridge. A grouping tool prevents users from duplicating items within a fridge. Users have the ability to modify or delete items, providing a clear indication of when an item is consumed.

Additionally, users receive alerts notifying them if a food item has expired or is nearing its expiration date. This system empowers users to proactively manage their food inventory, minimizing waste and promoting efficient consumption practices.

III. 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 web d'inventaire d'aliments.

- Contexte

Fresh (Entreprise) souhaite mettre en place une solution permettant à n'importe qui, la possibilité d'avoir un inventaire d'aliments disponible dans leurs réfrigérateurs.

- Fonctionnalités

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

1. La plateforme a une page d'enregistrement
 - L'utilisateur doit se créer un compte pour accéder aux autres pages.
 - L'utilisateur doit mettre les informations tel que :
 - Son nom,
 - Son prénom,
 - Son adresse mail,
 - Son mot de passe,
 - Une case à cocher pour les conditions d'utilisations.
2. La plateforme a une page de connexion.
 - Identification de l'utilisateur connecté.
 - L'utilisateur peut se déconnecter.
 - L'utilisateur peut accéder à ses informations.
3. La plateforme a une page de compte utilisateur.
 - L'utilisateur peut modifier les informations de son compte.
4. La plateforme a un système qui permet d'activer son compte avec une vérification par courriel.
 - Un compte qui n'est pas activé n'a pas accès aux autres pages.
5. La plateforme a une page d'accueil.
 - Listage des alertes de péremption d'aliments.
 - Avec option de masquage/affichage.
 - 1^{er} masquage automatique après 10 secondes.
 - Listage des réfrigérateurs.
 - L'utilisateur peut cliquer sur un réfrigérateur pour accéder aux aliments disponibles dans ce dernier.
 - Possibilité de créer un réfrigérateur si l'utilisateur a moins de 2 réfrigérateurs.
6. La plateforme a une page pour chaque réfrigérateur de chaque utilisateur.
 - Listage des aliments dans le réfrigérateur avec :
 - Le nom,
 - La date d'ajout,
 - La quantité
 - La date de péremption,

- Un message permettant de savoir si l'aliment est périmé ou sera périmé (en-dessous de 4 jours supérieur à la date d'accès à la page).
- Possibilité d'ajouter un aliment dans le réfrigérateur.
- Possibilité de modifier un aliment dans le réfrigérateur.
- Possibilité de regrouper des aliments qui ont presque le même nom et qui ont la même date de péremption.
- Possibilité de supprimer des aliments.
- Possibilité de modifier le nom du réfrigérateur.
- Possibilité de supprimer le réfrigérateur avec tous ses aliments.

Fonctionnalité de la base de données

1. Une procédure est nécessaire pour récupérer une liste de dernières alertes de chaque aliment périmé ou qui seront périmés (en-dessous de 4 jours à la date) à partir du jour d'appel (généralement appelée JOUR-J).
2. Un déclencheur est nécessaire pour insérer les alertes pour les aliments périmé ou qui seront périmés (ce-dernier s'exécutera après l'insertion d'une nourriture).
3. Un événement est nécessaire pour insérer tous les jours, les alertes pour les aliments périmé ou qui seront périmés.

• Technologies utilisées

- Conception de la base de données
 - Modèle conceptuel de données avec Looping.
- Développement de la plateforme
 - Back-end en PHP 8.3.1, Framework Symfony 7.0.2
 - Front-end en CSS 3, NodeJs 21.6.0 (npm 9.6.5), Webpack Encore 4.5.0, TailwindCss 3.4.0, daisyUi 4.6.0
- Persistance des données
 - La base de données avec MySQL 8.3.0 en locale.

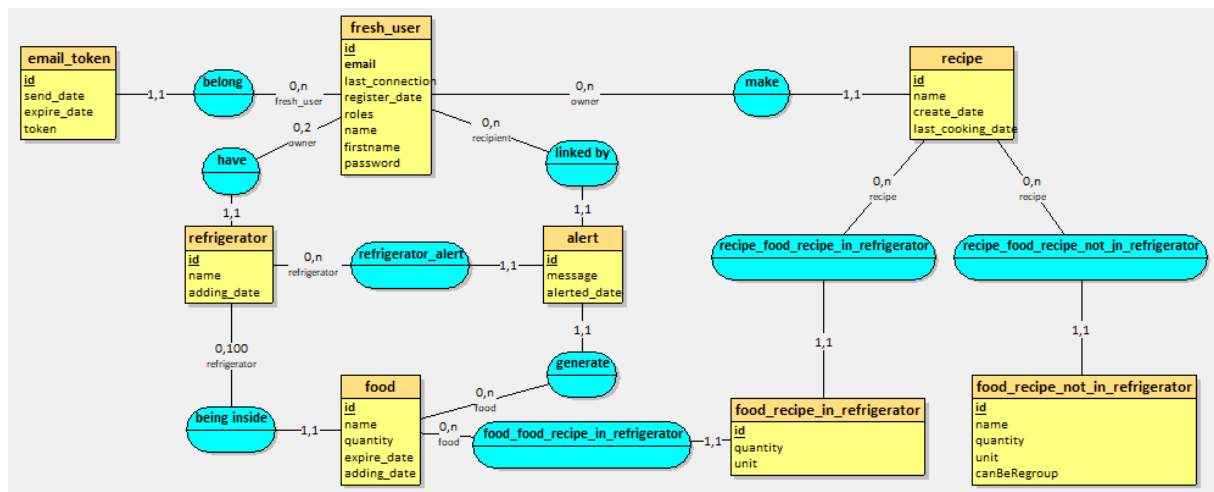
• Livrable et délais

- La plateforme complète et fonctionnelle doit être livrée conformément aux spécifications.
- La date de livraison prévue pour la plateforme est le 23/01/2024.

IV. Spécifications fonctionnelles du projet

- Développement de la plateforme
 - La gestion des composants de développement s'est faite avec
 - Composer 2.6.6 et NodeJs 21.6.0 (npm 9.6.5)
 - Les ressources (assets) utilisées sont traitées par
 - Webpack et PostCSS.
 - Outil de version avec Git 2.38.1
- Sécurité de la plateforme
 - Un système d'authentification est mis en place sur la plateforme avec une page de connexion.
 - Les mots de passes des utilisateurs sont chiffrés depuis la plateforme en utilisant bCrypt.
- Les routes dans le code sont sécurisées en vérifiant si l'utilisateur est connecté.
- La manipulation de la base de données dans le code est sécurisée avec des requêtes préparées.
- Les classes dans le code sont encapsulées.
- La base de données à un utilisateur root avec un mot de passe fort.
- L'application a son propre utilisateur authentifié avec un mot de passe fort, sur le serveur de gestion de base de données, il a toutes les permissions sur une seule base (base exclusivement dédiée à l'application)

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



Dans ce MCD, on retrouve 8 entités et 10 associations.

L'entité « fresh_user » possède un identifiant « id ».

L'entité « email_token » possède un identifiant « id » et est associé à 1 et 1 seul « fresh_user », l'entité « fresh_user » peut posséder 0 ou plusieurs « email_token ».

L'entité « refrigerator » possède un identifiant « id » et est associé à 1 et 1 seul « fresh_user », l'entité « fresh_user » peut avoir de 0 à 2 « refrigerator » au maximum.

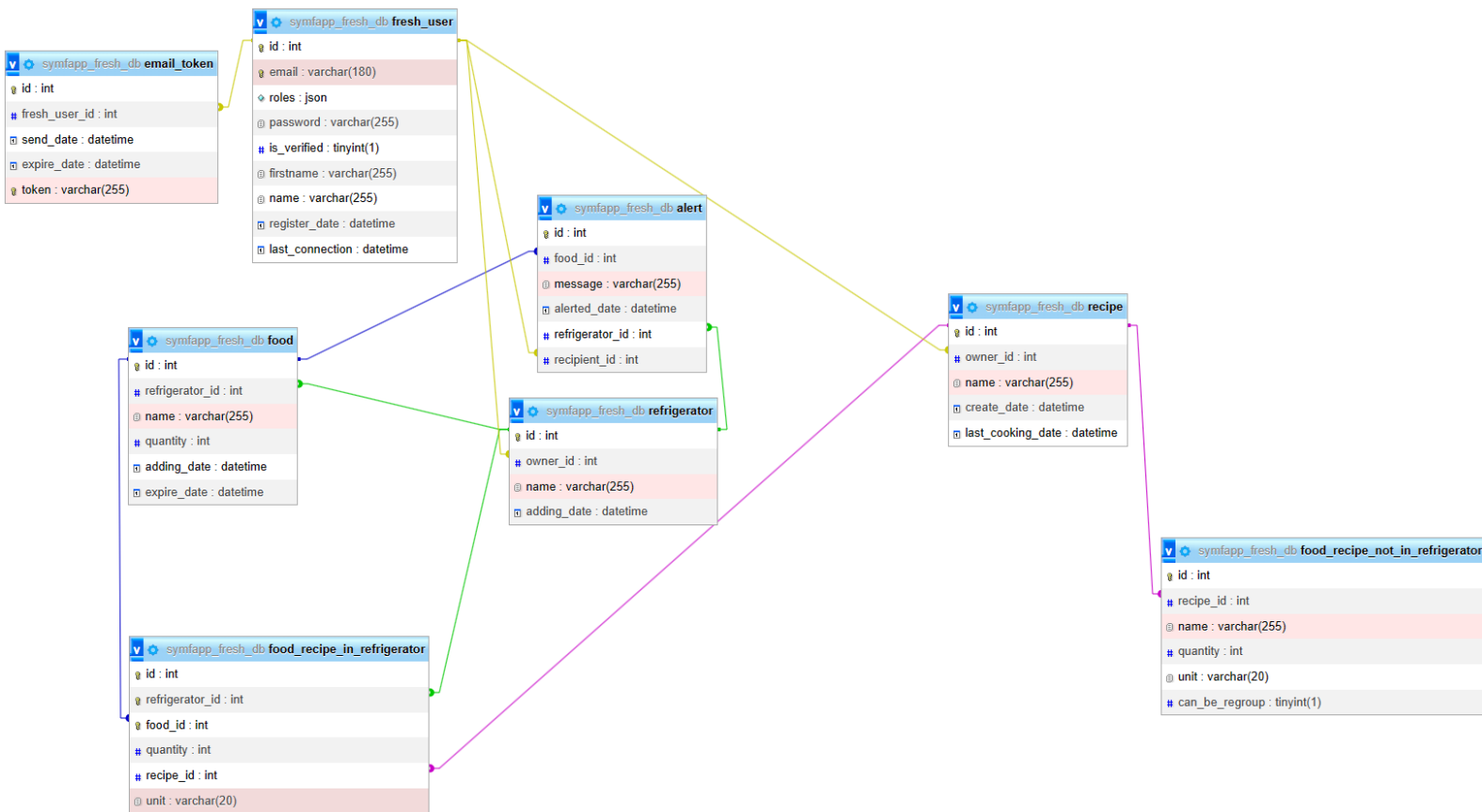
L'entité « food » possède un identifiant « id » et est associé à 1 et 1 seul « refrigerator », l'entité « refrigerator » peut avoir de 0 à 100 « food ».

L'entité « alert » possède un identifiant « id » et est associé à 1 et 1 seul « refrigerator », 1 et 1 seul « fresh_user » et 1 et 1 seul « food », les entités « refrigerator », « food », « fresh_user » peuvent apparaître dans 0 ou plusieurs « alert ».

L'entité « recipe » possède un identifiant « id » et est associé à 1 et 1 seul « fresh_user », l'entité « fresh_user » peut créer 0 ou plusieurs « recipe ».

L'entité « food_recipe_not_in_refrigerator » possède un identifiant « id » et est associé à 1 et 1 seul « recipe », l'entité « recipe » peut contenir 0 ou plusieurs « food_recipe_not_in_refrigerator ».

L'entité « food_recipe_in_refrigerator » possède un identifiant « id » et est associé à 1 et 1 seul « recipe » et à 1 et 1 seul « food », les entités « recipe » et « food » peuvent contenir 0 ou plusieurs « food_recipe_in_refrigerator ».



Dans ce Modèle Physique des Données, on observe 8 tables.

Voici en modèle textuel :

fresh_user = (id INT, email VARCHAR(255), last_connection DATETIME, register_date DATETIME, roles array, name VARCHAR(255), firstname VARCHAR(255), password VARCHAR(255));

refrigerator = (id INT, name VARCHAR(255), adding_date DATETIME, #id_owner);
#id_owner fait référence à id de la table fresh_user.

email_token = (id INT, send_date DATETIME, expire_date DATETIME, token VARCHAR(255), #id_fresh_user);
#id_fresh_user fait référence à id de la table fresh_user.

recipe = (id INT, name VARCHAR(255), create_date DATETIME, last_cooking_date DATETIME, #id_owner);
#id_owner fait référence à id de la table fresh_user.

food_recipe_not_in_refrigerator = (id INT, quantity INT, unit VARCHAR(20), canBeRegroup BOOLEAN, #id_recipe);
#id_recipe fait référence à id de la table recipe.

food = (id *INT*, name *VARCHAR(255)*, quantity *INT*, expire_date *DATETIME*, adding_date *DATETIME*, #id_refrigerator);
#id_refrigerator fait référence à id de la table refrigerator.

alert = (id *INT*, message *VARCHAR(255)*, #id_recipient, #id_food, #id_refrigerator);
#id_recipient fait référence à id de la table fresh_user.
#id_food fait référence à id de la table food.
#id_refrigerator fait référence à id de la table refrigerator.

food_recipe_in_refrigerator = (id *INT*, quantity *INT*, unit *VARCHAR(20)*, #id_recipe, #id_food);
#id_recipe fait référence à id de la table recipe.
#id_food fait référence à id de la table food.

Une fois transformé en base de données, elle se nomme « symfapp_fresh_db », la sécurisation d'accès a dû être faite. (voir annexe 1)

L'utilisateur « root » a un mot de passe fort. L'utilisateur « symfapp_fresh » a été créé et possède aussi un mot de passe fort, il a tous les droits sur la base de données « symfapp_fresh_db ».

L'utilisateur « root » peut se connecter qu'à partir de l'hôte localhost. L'utilisateur symfapp_fresh peut se connecter à partir de n'importe quel hôte.

Utilisateur	Permission
root	<p>GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, SHUTDOWN, PROCESS, FILE, REFERENCES, INDEX, ALTER, SHOW DATABASES, SUPER, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER, CREATE TABLESPACE, CREATE ROLE, DROP ROLE ON *.* TO `root`@`localhost` WITH GRANT OPTION;</p> <p>GRANT ALLOW_NONEXISTENT_DEFINER,APPLICATION_PASSWORD_ADMIN,AUDIT_ABORT_EXEMPT,AUDIT_ADMIN,AUTHENTICATION_POLICY_ADMIN,BACKUP_ADMIN,BINLOG_ADMIN,BINLOG_ENCRYPTION_ADMIN,CLONE_ADMIN,CONNECTION_ADMIN,ENCRYPTION_KEY_ADMIN,FIREWALL_EXEMPT,FLUSH_OPTIMIZER_COSTS,FLUSH_STATUS,FLUSH_TABLES,FLUSH_USER_RESOURCES,GROUP_REPLICATION_ADMIN,GROUP_REPLICATION_STREAM,INNODB_REDO_LOG_ARCHIVE,INNODB_REDO_LOG_ENABLE,PASSWORDLESS_USER_ADMIN,PERSIST_RO_VARIABLES_ADMIN,REPLICATION_APPLIER,REPLICATION_SLAVE_ADMIN,RESOURCE_GROUP_ADMIN,RESOURCE_GROUP_USER,ROLE_ADMIN,SENSITIVE_VARIABLES_OBSERVER,SERVICE_CONNECTION_ADMIN,SESSION_VARIABLES_ADMIN,SET_ANY_DEFINER,SHOW_ROUTINE,SYSTEM_USER,SYSTEM_VARIABLES_ADMIN,TABLE_ENCRYPTION_ADMIN,TELEMETRY_LOG_ADMIN,XA_RECOVER_ADMIN ON *.* TO `root`@`localhost` WITH GRANT OPTION;</p> <p>GRANT PROXY ON ``@`` TO `root`@`localhost` WITH GRANT OPTION;</p>
symfapp_fresh	GRANT USAGE ON *.* TO `symfapp_fresh`@`%`;

	<pre>GRANT ALL PRIVILEGES ON `symfapp_fresh_%.*` TO `symfapp_fresh`@`%`; GRANT ALL PRIVILEGES ON `symfapp_fresh_db`.* TO `symfapp_fresh`@`%`;</pre>
--	---

Une sauvegarde automatique journalière est exécutée pour sauvegarder la base de données.

Une sauvegarde sur un cloud privé est faite à chaque modification du répertoire.

Fichier de sauvegarde :

@echo off

set "backupdir=C:\Users\hoare\OneDrive\MSP2 - EPCF2\backups_sql\"

set "databases=symfapp_fresh_db"

set "mysqldumpcmd=mysqldump"

set "userpassword= --user=root --password=P*7k2UZ.bws6*X!E"

set "dumpoptions= --quick --add-drop-table --add-locks --extended-insert --lock-tables"

for /f "tokens=1-6 delims=/:. " %%a in ("%date% %time%") do (

set "TS=%%c%%b%%a%%d%%e%%f"

)

mkdir "%backupdir%" 2>nul

if not exist "%backupdir%" (

echo Not a directory: %backupdir%

exit /b 1

)

echo Backup symfapp_fresh_db

for %%d in (%databases%) do (

%mysqldumpcmd% %userpassword% %dumpoptions% %%d > "%backupdir%\%TS%-%%d.sql"

)

Dans le planificateur de tâche :

VI. 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.

Fresh !



DÉCONNEXION

Vous devez activer votre compte en cliquant sur le lien envoyé par mail ! (no-reply@fresh.app)

Cliquez pour renvoyer un lien d'activation

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

```
private function sendEmailVerification(EntityManagerInterface $entityManager, FreshUser $user)
{
    $emailToken = new EmailToken();
    $legacyUser = $entityManager->getRepository(FreshUser::class)->findOneBy(['email' => $user->getEmail()]);
    //DISABLE ALL VALID TOKENS
    $legacyToken = $entityManager->getConnection()->prepare( sql: "CALL disableAllTokenForUser(:userId)");//return one result
    $legacyToken->executeQuery(['userId' => $legacyUser->getId()]);
    $entityManager->getConnection()->close();
    //
    $emailToken->setFreshUser($legacyUser);
    $entityManager->persist($emailToken);
    $entityManager->flush();

    $legacyToken = $entityManager->getConnection()->prepare( sql: "CALL getLastTokenForUser(:userId)");//return one result
    $legacyToken = $legacyToken->executeQuery(['userId' => $legacyUser->getId()])->fetchAllAssociative()[0]['token'];
    $entityManager->getConnection()->close();

    // generate a signed url and email it to the user
    $loader = new FilesystemLoader( paths: 'email-template');
    $twigEnv = new Environment($loader);
    $twigBodyRenderer = new BodyRenderer($twigEnv);
    $email = (new TemplatedEmail())
        ->from(new Address( address: 'no-reply@fresh.app', name: 'Fresh Support !'))
        ->to($user->getEmail())
        ->subject( subject: 'Activer votre compte Fresh !')
        ->htmlTemplate( template: 'confirmation_email')
        ->context(['user'=>$user,'url' => $this->generateUrl( route: '/verify/email', [], referenceType: UrlGeneratorInterface::ABSOLUTE_URL), 'token' => $legacyToken]);
    $twigBodyRenderer->render($email);
    $this->emailVerifier->send( verifyEmailRouteName: 'app_verify_email', $user,
        $email
    );
}
```

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

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

```
//DISABLE ALL VALID TOKENS
$legacyToken = $entityManager->getConnection()->prepare( sql: "CALL disableAllTokenForUser(:userId)");//return one result
$legacyToken->executeQuery(['userId' => $legacyUser->getId()]);
$entityManager->getConnection()->close();
//
```

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

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `disableAllTokenForUser` (  
  IN `userId` INT  
)  
LANGUAGE SQL  
NOT DETERMINISTIC  
CONTAINS SQL  
SQL SECURITY DEFINER  
COMMENT ''  
BEGIN  
  UPDATE email_token  
  SET expire_date = NOW()  
  WHERE fresh_user_id = userId  
  AND expire_date > NOW();  
END
```

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.

```
CREATE DEFINER=`root`@`localhost` 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
```

La date d'envoi 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.

```
$legacyToken = $entityManager->getConnection()->prepare( sql: "CALL getLastTokenForUser(:userId)");//return one result  
$legacyToken = $legacyToken->executeQuery(["userId" => $legacyUser->getId()])->fetchAllAssociative()[0]['token'];  
$entityManager->getConnection()->close();
```

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

```
CREATE DEFINER=`symfapp_fresh`@`localhost` PROCEDURE `getLastTokenForUser` (
  IN `userId` INT
)
LANGUAGE SQL
NOT DETERMINISTIC
CONTAINS SQL
SQL SECURITY INVOKER
COMMENT ''
BEGIN
  SELECT *
  FROM email_token
  WHERE fresh_user_id = userId
    AND expire_date >= NOW()
  ORDER BY expire_date ASC
  LIMIT 1;
END
```

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 » :

#	id	fresh_user_id	send_date	expire_date	token
---	----	---------------	-----------	-------------	-------

Puis on envoie le courriel avec le token récupéré.

Subject : [Activer votre compte Fresh !](#)
From : no-reply@fresh.app To : steve.hoareau1@gmail.com
Dated : 2024-01-17 08:13:48 ([Delete](#))

[HTML](#) [HTML Source](#) [Text](#) [RAW](#)

Fresh !

Steve,

Vous venez de créer un compte sur [Fresh !](#)

Vous devez confirmer votre compte en cliquant [ici](#)

Si ce n'est pas vous, merci de nous contacter à contact@fresh.app

De l'ékip de Fresh !

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

```
#[Route('/verify/email', name: 'app_verify_email')]
public function verifyUserEmail(Request $request, EntityManagerInterface $entityManager): Response
{
    $id = $request->query->get('token');

    if (null === $id) {
        return $this->redirectToRoute('route: 'register');
    }

    $emailToken = $entityManager->getRepository(EmailToken::class)->findOneBy(['token' => $id]);

    // validate email confirmation link, sets User::isVerified=true and persists
    $isValidToken = $entityManager->getConnection()->prepare("SELECT isTokenValid(:token)");
    $isValidToken = $isValidToken->executeQuery(["token" => $id])->fetchOne();
    $entityManager->getConnection()->close();
```

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

```
$isValidToken = $entityManager->getConnection()->prepare( sql: "SELECT isValidToken(:token)");
$isValidToken = $isValidToken->executeQuery(["token" => $id])->fetchAssociative();
$entityManager->getConnection()->close();
if ($isValidToken) {
```

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

```
CREATE DEFINER=`symfapp_fresh`@`%` FUNCTION `isValidToken` (
  `tokenParam` VARCHAR(255)
)
RETURNS tinyint
LANGUAGE SQL
NOT DETERMINISTIC
READS SQL DATA
SQL SECURITY INVOKER
COMMENT ''
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
```

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.

Fresh !

 DÉCONNEXION

Bonjour Steve,

Vous avez 2 alertes !

CACHER LES ALERTES

Tomates se trouvant dans Salon
Quantité : 2
Cet aliment est périmé depuis 6 jours

Chocolat se trouvant dans Salon
Quantité : 1
Cet aliment est périmé depuis 7 jours

Des alertes s'affichent pendant au moins 10secondes.

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

```
CREATE DEFINER='symfapp_fresh'@'localhost' EVENT `every_day_insert_alert`
ON SCHEDULE
  EVERY 1 DAY STARTS '2024-01-09 00:01:00'
ON COMPLETION PRESERVE
ENABLE
COMMENT ''
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
```

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)

```
#[Route('/', name: 'app_main')]
#[IsGranted("IS_AUTHENTICATED_FULLY")]
public function index(EntityManagerInterface $entityManager): Response
{
    $user = $entityManager->getRepository(FreshUser::class)->findOneBy(["email"=>$this->getUser()->getUserIdentifier()]);
    $today = new \DateTime();
    $user->setLastConnection($today);
    $entityManager->persist($user);
    $entityManager->flush();
    $today->setTime( hour: 0, minute: 0, second: 0);
    $dateString = $today->format( format: 'Y-m-d');

    //on récupère les alertes déjà générées, si vide alors on genere des alertes
    $alerts = $this->getAlertsByUser($entityManager,$user);
    if(empty($alerts)){
        $legacyAlerts = $entityManager->getConnection()->prepare( sql: "CALL genereAlertForUser(:userId)");
        $legacyAlerts->executeQuery(["userId"=>$user->getId()]);
        $alerts = $this->getAlertsByUser($entityManager,$user);
    }

    return $this->render( view: 'index', [
        'user' => $user,
        'alerts'=>$alerts,
        'dateString'=>$dateString
    ]);
}
```

On récupère les alertes pour l'utilisateur.

```
private function getAlertsByUser(EntityManagerInterface $entityManager, FreshUser $user){
    $legacyAlerts = $entityManager->getConnection()->prepare( sql: "CALL getTodayAlertForUser(:recipientId)");
    $legacyAlerts = $legacyAlerts->executeQuery(["recipientId"=>$user->getId()])->fetchAllAssociative();
    $alerts = array();
    foreach ($legacyAlerts as $alert) {
        array_push( &array: $alerts, $entityManager->getRepository(Alert::class)->find($alert['alert_id']));
    }
    return $alerts;
}
```

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


```
CREATE DEFINER=`symfapp_fresh`@`localhost` PROCEDURE `getTodayAlertForUser`(
  IN `recipientId` INT
)
LANGUAGE SQL
NOT DETERMINISTIC
READS SQL DATA
SQL SECURITY INVOKER
COMMENT ''
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
```

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.

```
//on récupère les alertes déjà générées, si vide alors on genere des alertes
$alerts = $this->getAlertsByUser($entityManager,$user);
if(empty($alerts)){
    $legacyAlerts = $entityManager->getConnection()->prepare( sql: "CALL genereAlertForUser(:userId)");
    $legacyAlerts->executeQuery(['userId'=>$user->getId()]);
    $alerts = $this->getAlertsByUser($entityManager,$user);
}
```

```

CENTRE DE FORMATION
INFORMATIQUE
CREATE DEFINER='root'@'localhost' PROCEDURE `genereAlertForUser`(
  IN `userId` INT
)
LANGUAGE SQL
NOT DETERMINISTIC
CONTAINS SQL
SQL SECURITY DEFINER
COMMENT ''
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

```

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.

```

{# ... #}
{% for alert in alerts %}
  <div id="alert-{{ i }}" class="freshalert text-white px-6 py-4 border-0 rounded relative mb-2 bg-pink-500">
    <span class="inline-block align-middle mr-8">
      <b class="capitalize">{{ alert.food.name }}</b> se trouvant dans <b class="capitalize">{{ alert.refrigerator.name }}</b><br>
      <b class="capitalize">Quantité : {{ alert.food.quantity }}</b><br>
      {{ alert.message }}
    </span>
  </div>
  <script>
    setTimeout(function(){
      closeAlert();
    },10000);
  </script>
  {% set i = i+1 %}
{% endfor %}
{# ... #}

```

VII. 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).

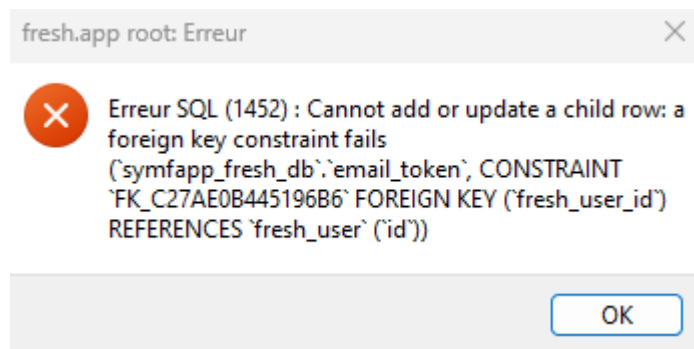
- **Si Données en entrée :**

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 :

#	isTokenValid("218b6ec2f5e8f2019fc13864bbeeb9...")
1	1

Sinon :

#	isTokenValid("218b6ec2f5e8f2019fc13864bbeeb9...")
1	0

- **Si données en entrée :**

« invalid_token »

```
SELECT isTokenValid("invalid_token");
```

Alors données obtenue:

#	isTokenValid("invalid_token")
1	0

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 périmé
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

VIII. 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 vérifiées 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 sécurité et empêche-la plupart des failles. Les failles restantes doivent être réglées 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).

IX. Description d'une situation de travail ayant nécessité une recherche, effectuée par le candidat durant le projet

Le site web [Stack Overflow](#) 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](#) a été utilisé pour les différents blocages de code.

La documentation [daisyUi](#) a été utilisé pour trouver les bonnes classes a ajouté pour le front-end de l'application.

La documentation [Symfony](#) a été utilisé pour l'envoi de mail et les différents blocages de code.

X. 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_17FD46C1915EAE8` (`refrigerator_id`),  
  KEY `IDX_17FD46C1E92F8F78` (`recipient_id`),  
  CONSTRAINT `FK_17FD46C1915EAE8` 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',  
'032a2ccf94e00d635579676eaeef469e5'),  
  
(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',  
'218b6ec2f5e8f2019fc13864bbbeb950');
```

-- 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_regroupe`) 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. isValidToken

DELIMITER //

CREATE FUNCTION `isValidToken` (

 `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:"Symfony\Component\Messenger\Envelope\Ostamps";a:1:{s:46:"Symfony\Component\Messenger\Stamp\BusNameStamp";a:1:{i:0;O:46:"Symfony\Component\Messenger\Stamp\BusNameStamp":1:{s:55:"Symfony\Component\Messenger\Stamp\BusNameStamp\ObusName";s:21:"messenger.bus.default";}}s:45:"Symfony\Component\Messenger\Envelope\Omessage";O:51:"Symfony\Component\Mailer\Messenger\SendMessage":2:{s:60:"Symfony\Component\Mailer\Messenger\SendMessage\Ome
```

```
ssage\\";O:39:\\\"Symfony\\\\\\\\Bridge\\\\\\\\Twig\\\\\\\\Mime\\\\\\\\TemplatedEmail\\\\\\\\":5:{i:0;s:41:\\\"registrati
on/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%2BHbpw7SV6zWSQp4j6sj
K50b%2FU%3D&token=cp85vmRR7gGrjv32m3etHjw1EjtJ63hRjQWDAmlDm7Y%3D\\\\\\\\";s:19:\\\"expires
AtMessageKey\\\\\\\\";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:\\\"\\\\0Symfon
y\\\\\\\\Component\\\\\\\\Mime\\\\\\\\Header\\\\\\\\AbstractHeader\\\\0name\\\\\\\\";s:4:\\\"From\\\\\\\\";s:56:\\\"\\\\0Symf
ony\\\\\\\\Component\\\\\\\\Mime\\\\\\\\Header\\\\\\\\AbstractHeader\\\\0lineLength\\\\\\\\";i:76;s:50:\\\"\\\\0Symfony
\\\\\\\\Component\\\\\\\\Mime\\\\\\\\Header\\\\\\\\AbstractHeader\\\\0lang\\\\\\\\";N;s:53:\\\"\\\\0Symfony\\\\\\\\Compon
ent\\\\\\\\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\\\\\\\\Componen
t\\\\\\\\Mime\\\\\\\\Address\\\\0address\\\\\\\\";s:18:\\\"no-
reply@fresh.app\\\\\\\\";s:36:\\\"\\\\0Symfony\\\\\\\\Component\\\\\\\\Mime\\\\\\\\Address\\\\0name\\\\\\\\";s:14:\\\"Fres
h Suport
\\\\\\\\";}}s:2:\\\"to\\\\\\\\";a:1:{i:0;O:47:\\\"Symfony\\\\\\\\Component\\\\\\\\Mime\\\\\\\\Header\\\\\\\\MailboxListHead
er\\\\\\\\":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\\\\\\\\Componen
t\\\\\\\\Mime\\\\\\\\Address\\\\0address\\\\\\\\";s:24:\\\"steve.hoareau1@gmail.com\\\\\\\\";s:36:\\\"\\\\0Symfony\\\\\\\\C
omponent\\\\\\\\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\\\\\\\\Compon
ent\\\\\\\\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\\\\\\\\SendMessageMessage\\\\0e
nvelope\\\\\\\\";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,ER
ROR_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,ER
ROR_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,ER  
ROR_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,ER  
ROR_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,ER  
ROR_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,ER  
ROR_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) */;
```