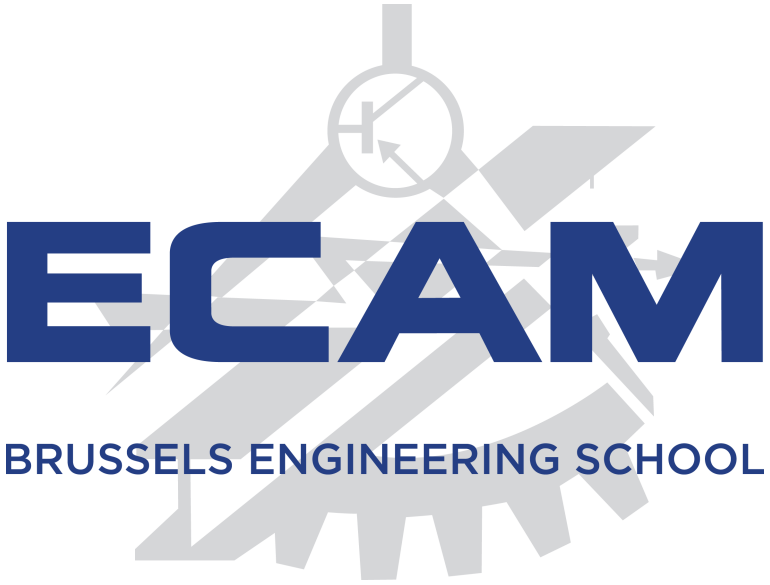

Projet Architecture Web

Projet Fridge - No Waste

Autheur:
Harold Snyers d'Attenhoven
16243

Professeur:
Mr. Philippe Dekimpe

Un projet en architecture web utilisant

les framework Symfony et Angular

Année académique 2019-2020

Contents

Contents i

1 Specification et Base de données 1
1.1 Specification . . . 1
1.1.1 Objectif du site . . . 1
1.1.2 Acteurs . . . 1
1.1.3 User storie . . . 1
1.1.4 Contraintes non-fonctionnelles . . . 2
1.1.5 Mockup . . . 2
1.2 Base de donnée . . . 8
1.2.1 Diagramme d'entité relation . . . 8
1.2.2 Diagramme relationnel . . . 8
1.2.3 Description des tables et des relations entre tables . . . 8
1.2.4 Description précise des champs . . . 9

2 Symfony - Twig 12
2.1 En tant qu'utilisateur, je veux pouvoir m'enregistrer . . . 12
2.1.1 Gestion des erreurs et sécurité . . . 13
2.1.2 partie de codes significative . . . 14
2.1.3 Lien vers script concerné . . . 14
2.1.4 Diagramme de séquence . . . 15
2.2 En tant qu'utilisateur, je veux pouvoir me connecter . . . 16
2.2.1 Gestion d'erreur et sécurité . . . 16
2.2.2 partie de codes significative . . . 17
2.2.3 Lien vers script concerné . . . 17
2.2.4 Diagramme de séquence . . . 18
2.3 En tant qu'utilisateur, je veux pouvoir retrouver les frigos que j'ai créé . . . 19
2.3.1 Gestion d'erreur et sécurité . . . 20
2.3.2 Partie de code significative . . . 20
2.3.3 Lien vers script concerné . . . 20
2.4 En tant qu'utilisateur, je veux pouvoir créer un frigo ou plusieurs . . . 21
2.4.1 Gestion d'erreur et sécurité . . . 22
2.4.2 Partie de code significative . . . 22
2.4.3 Lien vers script concerné . . . 22
2.5 En tant qu'utilisateur, je veux pouvoir modifier mon frigo . . . 23
2.5.1 Gestion d'erreur et sécurité . . . 24
2.5.2 Partie de code significative . . . 24
2.5.3 Lien vers script concerné . . . 24
2.6 En tant qu'utilisateur, je veux pouvoir supprimer mon frigo . . . 25
2.6.1 Gestion d'erreur et sécurité . . . 26
2.6.2 Partie de code significative . . . 26
2.6.3 Lien vers script concerné . . . 26
2.7 En tant qu'utilisateur, je veux pouvoir lister les étages de mon frigo . . . 27

2.7.1 Gestion d'erreur et sécurité 28
2.7.2 Partie de code significative 28
2.7.3 Lien vers script concerné 28
2.8 En tant qu'utilisateur, je veux pouvoir ajouter des étages à mon frigo . . 29
2.8.1 Gestion d'erreur et sécurité 29
2.8.2 Partie de code significative 29
2.8.3 Lien vers script concerné 30
2.9 En tant qu'utilisateur, je veux pouvoir modifier les étages de mon frigo . . 31
2.9.1 Gestion d'erreur et sécurité 32
2.9.2 Partie de code significative 32
2.9.3 Lien vers script concerné 32
2.10 En tant qu'utilisateur, je veux pouvoir supprimer les étages de mon frigo . 33
2.10.1 Gestion d'erreur et sécurité 33
2.10.2 Partie de code significative 34
2.10.3 Lien vers script concerné 34
2.11 En tant qu'utilisateur, je veux pouvoir lister les aliments se trouvant dans un étage 35
2.11.1 Gestion d'erreur et sécurité 36
2.11.2 Partie de code significative 36
2.11.3 Lien vers script concerné 36
2.12 En tant qu'utilisateur, je veux pouvoir afficher un aliment spécifique . . . 37
2.12.1 Gestion d'erreur et sécurité 38
2.12.2 Partie de code significative 38
2.12.3 Lien vers script concerné 38
2.13 En tant qu'utilisateur, je veux pouvoir ajouter des aliments à mon frigo . 39
2.13.1 Gestion d'erreur et sécurité 40
2.13.2 Partie de code significative 40
2.13.3 Lien vers script concerné 41
2.14 En tant qu'utilisateur, je veux pouvoir modifier des aliments de mon frigo 42
2.14.1 Gestion d'erreur et sécurité 43
2.14.2 Partie de code significative 44
2.14.3 Lien vers script concerné 44
2.15 En tant qu'utilisateur, je veux pouvoir supprimer des aliments de mon frigo 45
2.15.1 Gestion d'erreur et sécurité 46
2.15.2 Partie de code significative 46
2.15.3 Lien vers script concerné 46

3 API **47**
3.1 Tableau récapitulatif 47
3.2 Requêtes: Détaillées 48
3.2.1 User/register 48
3.2.2 User/login 52
3.2.3 fridge/index 55
3.2.4 fridge/add 58
3.2.5 fridge/edit 61
3.2.6 fridge/delete 64
3.2.7 floor/index 66

3.2.8 floor/add 69
3.2.9 floor/edit 71
3.2.10 floor/delete 74
3.2.11 food/index 76
3.2.12 food/add 80
3.2.13 food/edit 83
3.2.14 food/delete 86

4 Symfony - Angular 88
4.1 En tant qu'utilisateur, je veux pouvoir m'enregistrer 89
4.1.1 Gestion des erreurs et sécurité 90
4.1.2 partie de codes significative 91
4.1.3 Lien vers script concerné 91
4.1.4 Diagramme de séquence 92
4.2 En tant qu'utilisateur, je veux pouvoir me connecter 93
4.2.1 Gestion des erreurs et sécurité 93
4.2.2 partie de codes significative 94
4.2.3 Lien vers script concerné 94
4.2.4 Diagramme de séquence 95
4.3 En tant qu'utilisateur, je veux pouvoir retrouver les frigos que j'ai créé . . 96
4.3.1 Gestion d'erreur et sécurité 97
4.3.2 partie de codes significative 97
4.3.3 Lien vers script concerné 97
4.4 En tant qu'utilisateur, je veux pouvoir créer un frigo ou plusieurs 98
4.4.1 Gestion d'erreur et sécurité 98
4.4.2 Partie de code significative 99
4.4.3 Lien vers script concerné 99
4.5 En tant qu'utilisateur, je veux pouvoir modifier mon frigo 100
4.5.1 Gestion d'erreur et sécurité 101
4.5.2 Partie de code significative 101
4.5.3 Lien vers script concerné 101
4.6 En tant qu'utilisateur, je veux pouvoir supprimer mon frigo 102
4.6.1 Partie de code significative 103
4.6.2 Lien vers script concerné 103
4.7 En tant qu'utilisateur, je veux pouvoir lister les étages de mon frigo ainsi que leur aliments 104
4.7.1 Partie de code significative 106
4.7.2 Lien vers script concerné 106
4.7.3 Diagramme de séquence 107
4.8 En tant qu'utilisateur, je veux pouvoir ajouter des étages à mon frigo . . 108
4.8.1 Gestion d'erreur et sécurité 108
4.8.2 Partie de code significative 109
4.8.3 Lien vers script concerné 109
4.9 En tant qu'utilisateur, je veux pouvoir modifier les étages de mon frigo . . 110
4.9.1 Gestion d'erreur et sécurité 110
4.9.2 Partie de code significative 111
4.9.3 Lien vers script concerné 111

4.10 En tant qu'utilisateur, je veux pouvoir supprimer les étages de mon frigo . 112
4.10.1 Partie de code significative 113
4.10.2 Lien vers script concerné 113
4.11 En tant qu'utilisateur, je veux pouvoir ajouter des aliments à mon frigo . 114
4.11.1 Gestion d'erreur et sécurité 116
4.11.2 Partie de code significative 116
4.11.3 Lien vers script concerné 116
4.12 En tant qu'utilisateur, je veux pouvoir modifier des aliments de mon frigo 117
4.12.1 Gestion d'erreur et sécurité 118
4.12.2 Partie de code significative 118
4.12.3 Lien vers script concerné 119
4.13 En tant qu'utilisateur, je veux pouvoir supprimer des aliments de mon frigo 120
4.13.1 Partie de code significative 121
4.13.2 Lien vers script concerné 122
4.14 En tant qu'utilisateur, je veux pouvoir rechercher des aliments selon une caractéristique 123
4.14.1 Partie de code significative 123
4.14.2 Lien vers script concerné 123
4.15 En tant qu'utilisateur, je veux pouvoir lister les aliments en fonction de leur type ou leur date de fin 124
4.15.1 Lien vers script concerné 124

5 Conclusion 125
5.1 Objectifs atteints et non atteints 125
5.1.1 Symfony-Twig 125
5.1.2 Symfony-Angular 127
5.2 Piste d'amélioration 129

Specification et Base de données

1.1 Specification

1.1.1 Objectif du site

Le site doit permettre d'enregistrer les aliments du frigo, pouvoir faire un recherche dans un livre de recettes et de générer une liste de recette à partir d'aliments se trouvant dans le frigo.

1.1.2 Acteurs

Utilisateur : personne qui utilise le site

1.1.3 User storie

User Stories	priority
En tant qu'utilisateur, je veux pouvoir me connecter	1
En tant qu'utilisateur, je veux pouvoir m'enregister	1
En tant qu'utilisateur, je veux pouvoir retrouver les frigos que j'ai créé	1
En tant qu'utilisateur, je veux pouvoir créer un frigo ou plusieurs	1
En tant qu'utilisateur, je veux pouvoir modifier mon frigo	1
En tant qu'utilisateur, je veux pouvoir supprimer mon frigo	1
En tant qu'utilisateur, je veux pouvoir lister les étages de mon frigo	1
En tant qu'utilisateur, je veux pouvoir lister les étages d'un firgo ainsi que leurs aliments	2
En tant qu'utilisateur, je veux pouvoir ajouter des étages à mon frigo	1
En tant qu'utilisateur, je veux pouvoir modifier les étages de mon frigo	1
En tant qu'utilisateur, je veux pouvoir supprimer les étages de mon frigo	1

User Stories	priority
En tant qu'utilisateur, je veux pouvoir lister les aliments se trouvant dans un étage	1
En tant qu'utilisateur, je veux pouvoir afficher un aliment spécifique	2
En tant qu'utilisateur, je veux pouvoir ajouter des aliments à mon frigo	1
En tant qu'utilisateur, je veux pouvoir modifier des aliments de mon frigo	1
En tant qu'utilisateur, je veux pouvoir supprimer des aliments de mon frigo	1
En tant qu'utilisateur, je veux pouvoir lister les aliments en fonction de leur type ou leur date de fin	2
En tant qu'utilisateur, je veux pouvoir rechercher des aliments selon une caractéristique	2
En tant qu'utilisateur, je veux pouvoir trouver des recettes à partir d'alim du frigo	3
En tant qu'utilisateur, je veux pouvoir faire une recherche parmis les recettes	3

1.1.4 Contraintes non-fonctionnelles

Le site doit être ergonomique et responsive.

1.1.5 Mockup

1.1.5.1 Page Home

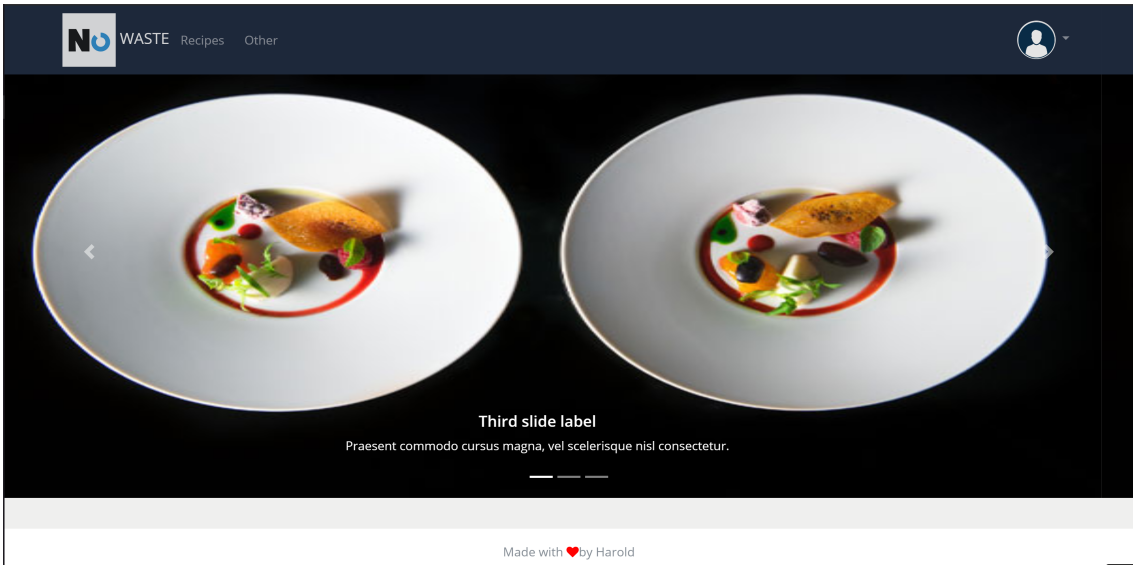

Figure 1.1

1.1.5.2 Page register

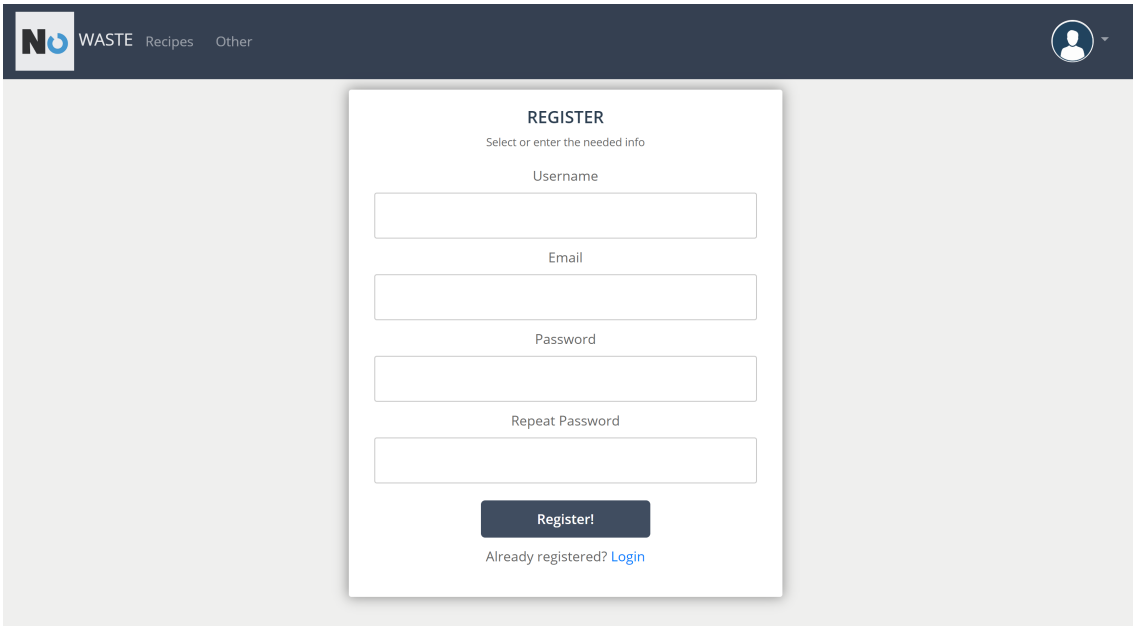

FIGURE 1.2

1.1.5.3 Page login

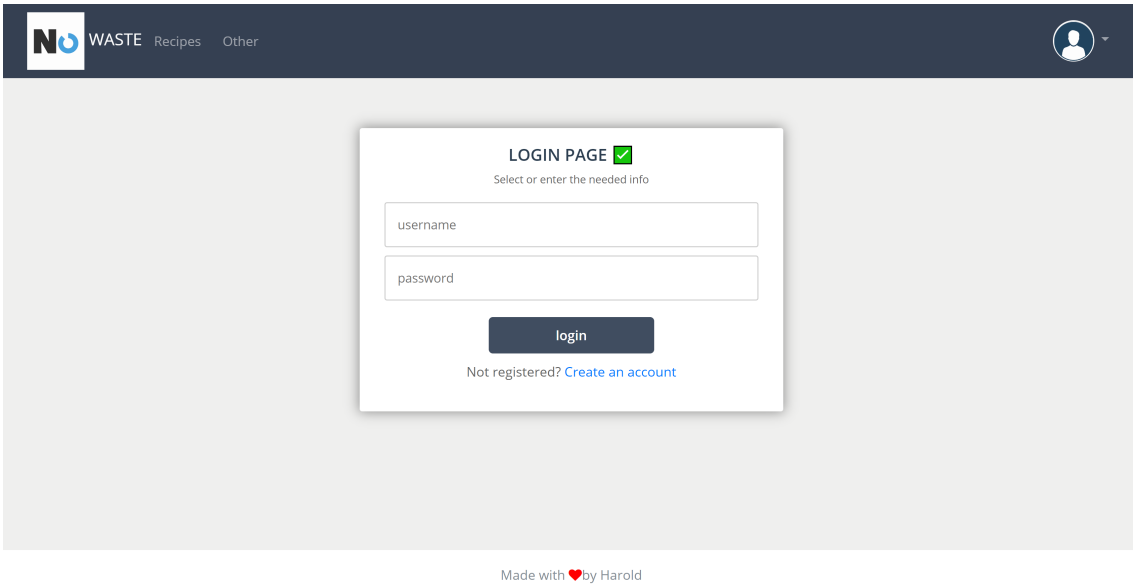

FIGURE 1.3

1.1.5.4 Page liste des frigo

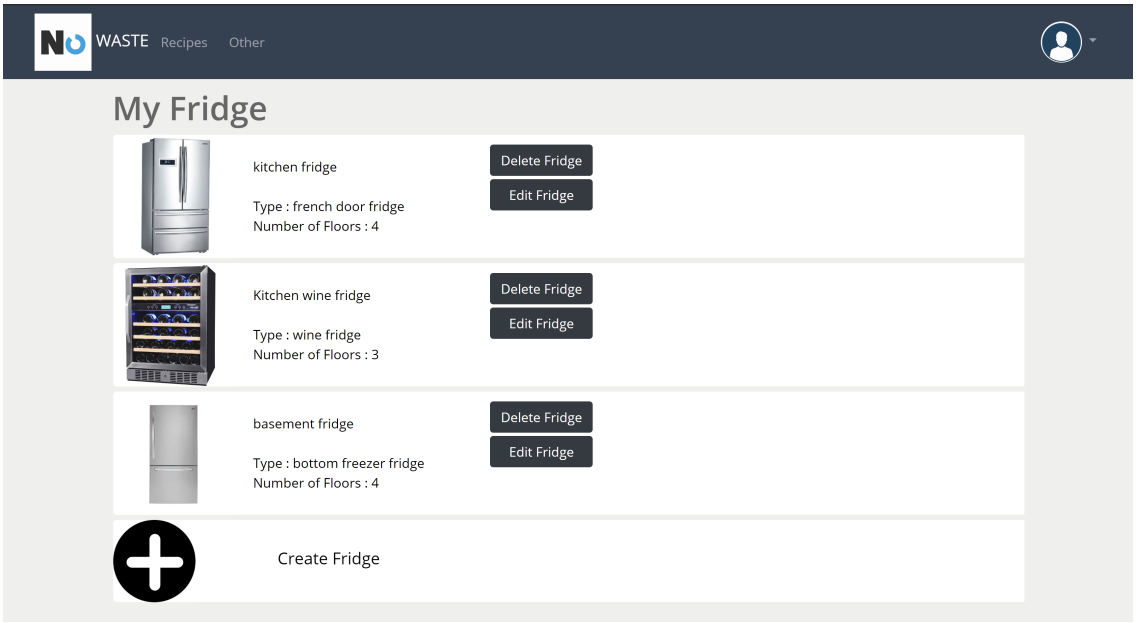

FIGURE 1.4

1.1.5.5 Page création et modification d'un frigo

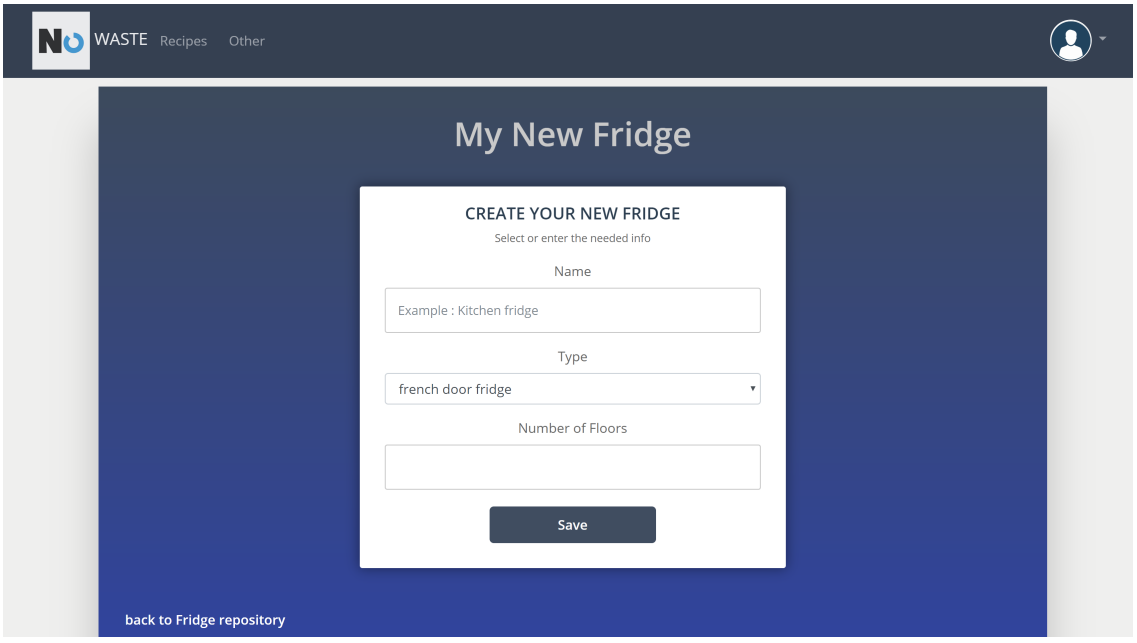

FIGURE 1.5

1.1.5.6 Page liste des étages d'un frigo

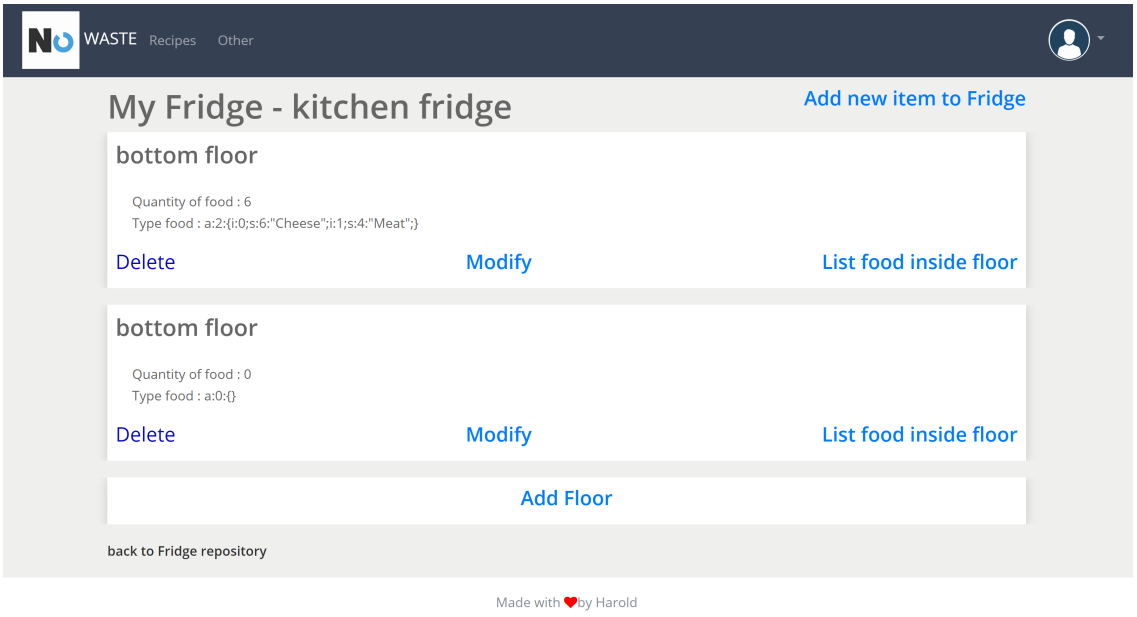

FIGURE 1.6

1.1.5.7 Page création et modification d'un étage

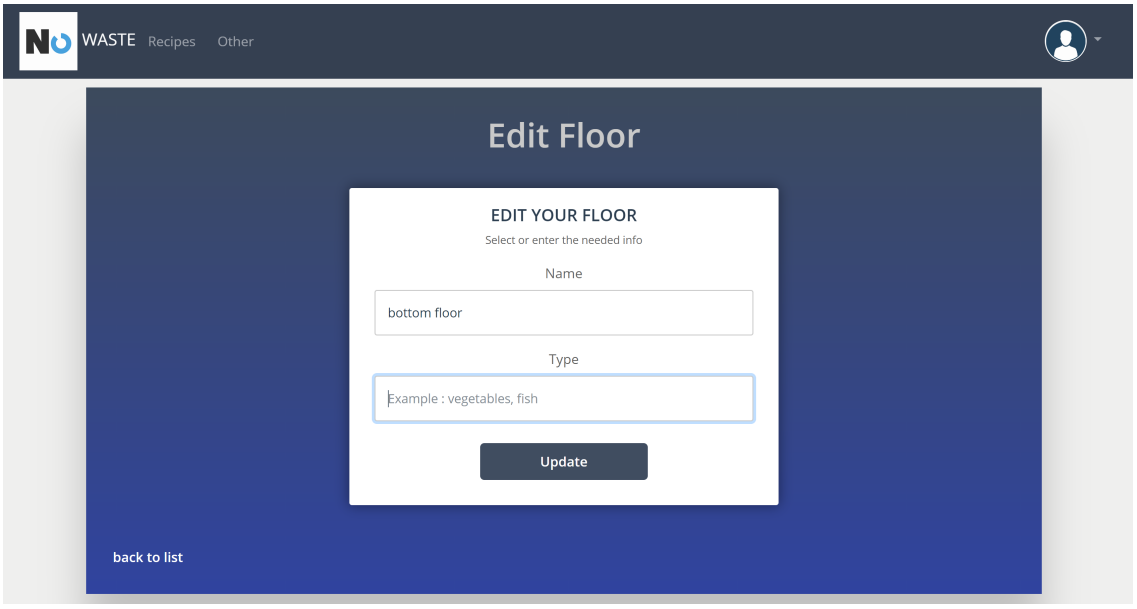

FIGURE 1.7

1.1.5.8 Page liste des aliment d'un étage

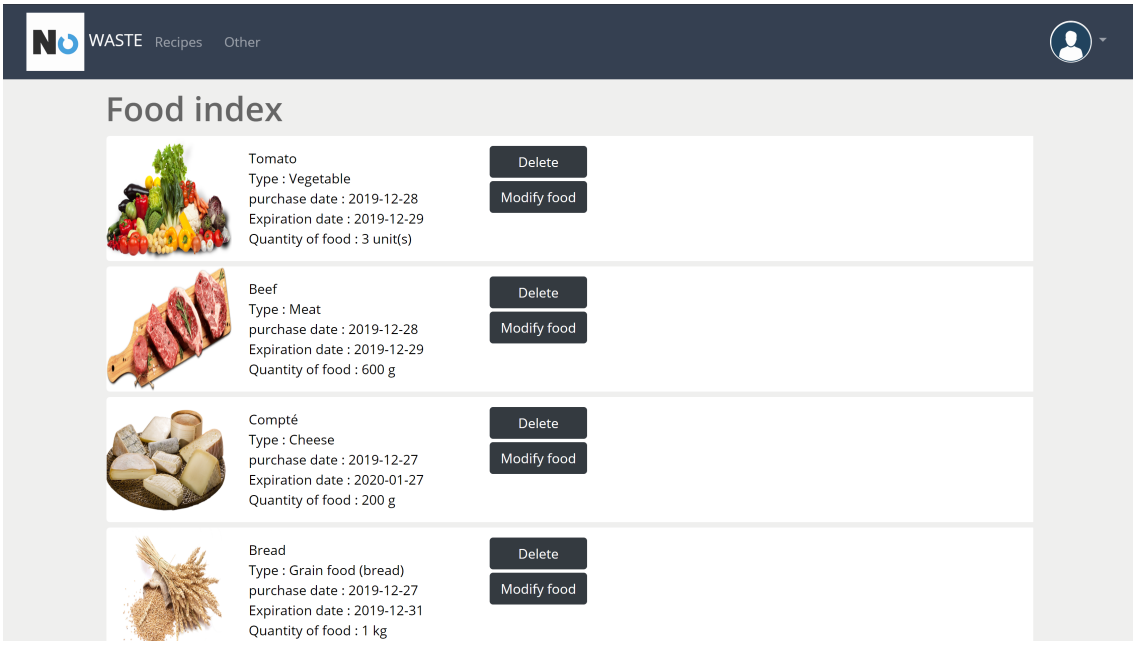

FIGURE 1.8

1.1.5.9 Page création et modification d'un aliment

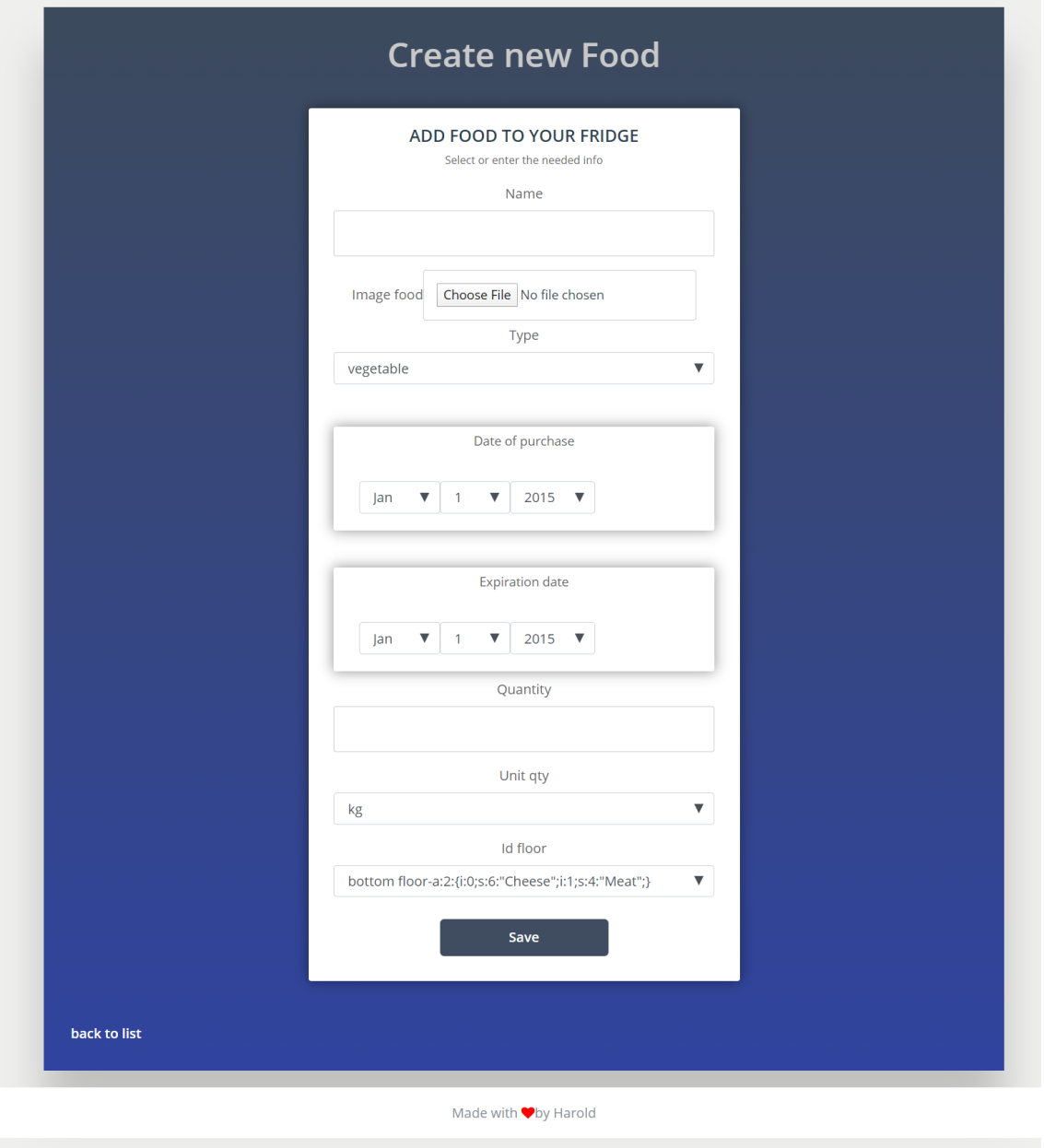

FIGURE 1.9

1.1.5.10 Page aliment spécifique

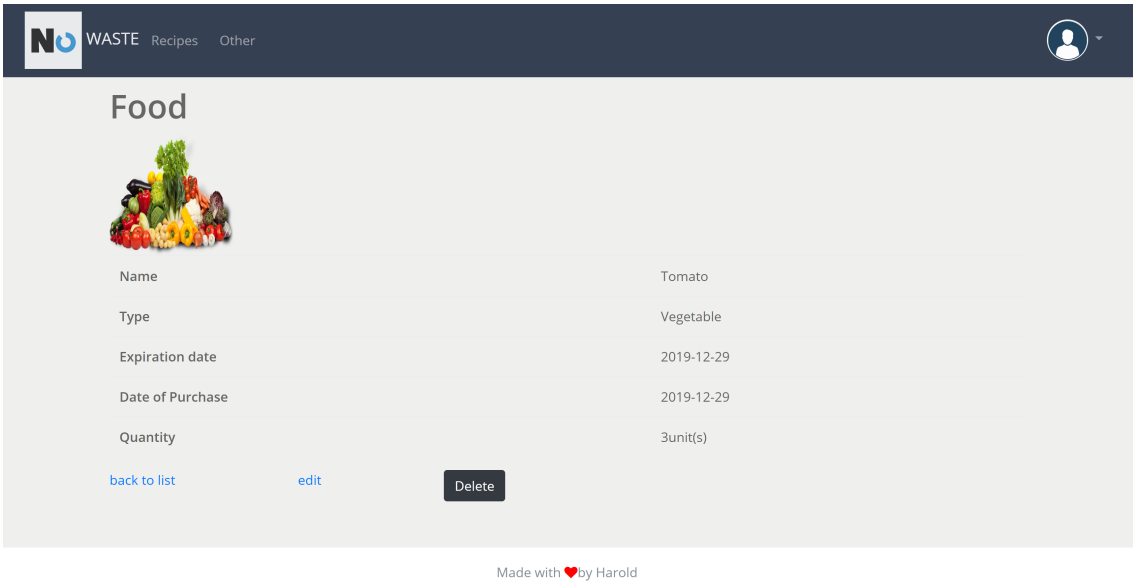

FIGURE 1.10

1.2 Base de donnée

1.2.1 Diagramme d'entité relation

1.2.2 Diagramme relationnel

1.2.3 Description des tables et des relations entre tables

Table	Table	Type	Description
user	fridge	0 à N	Un frigo à un utilisateur, un utilisateur peut avoir un ou plusieurs frigo
fridge	floor	0 à N	un floor (étage) à un frigo, un frigo peut contenir plusieurs floors
floor	food	0 à N	un food (un aliment) à un étages, un étages peut contenir plusieurs foods

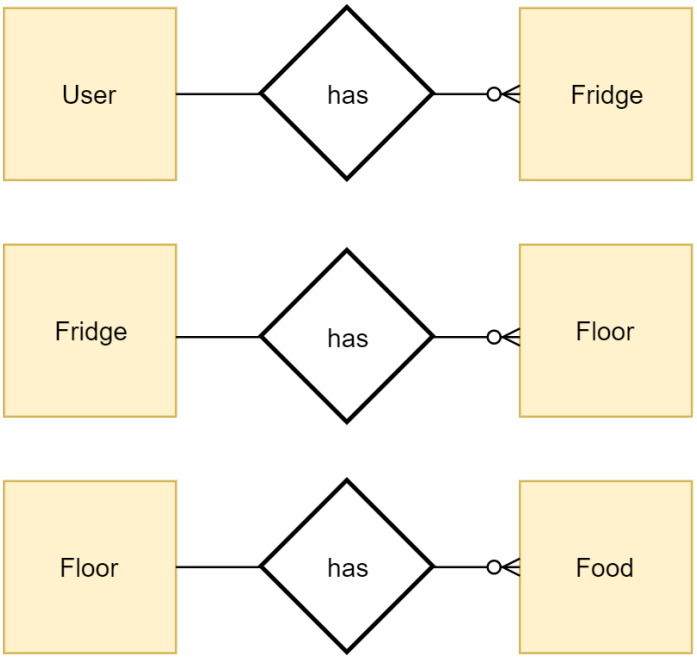

FIGURE 1.11: Diagramme d'entité relation

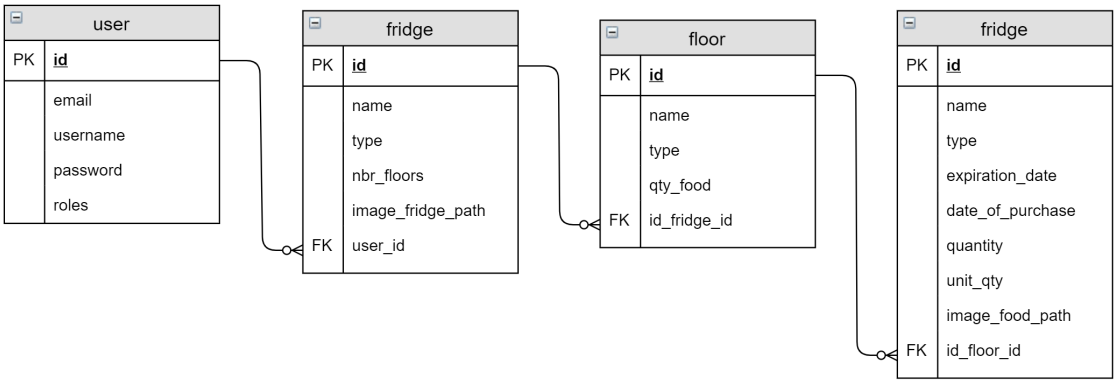

FIGURE 1.12: Diagramme Relationnel

1.2.4 Description précise des champs

1.2.4.1 Table: user

Nom	Key	Non null	Type	Référence	Intitulé	Commentaire
id	PK	O	int (11)		identifiant	unique
email		O	varchar (255)		email	unique
username		O	varchar (255)		username	unique
password		O	varchar (255)		password	taille 8 min
roles		N	text		roles	

1.2.4.2 Table: fridge

Nom	Key	Non null	Type	Référence	Intitulé	Commentaire
id	PK	O	int (11)		identifiant	unique
name		O	varchar (255)		Nom du frigo	
type		O	varchar (255)		type de frigo	
nbr_floors		O	int (11)		Nombre de floors	
user_id	FK	O	int (11)	id/user	lien avec l'utilisateur	
image_ fridge_ path		O	varchar (255)		chemin vers image du frigo selon le type	

1.2.4.3 Table: floor

Nom	Key	Non null	Type	Référence	Intitulé	Commentaire
id	PK	O	int (11)		identifiant	unique
type		O	text		type d'aliment dans l'étages	liste
qty_food		O	int (11)		quantité d'aliments dans l'étage	
id_fridge_ id	FK	O	int (11)	id/fridge	lien vers le frigo	
name		O	varchar (255)		nom de l'étage	

1.2.4.4 Table: food

Nom	Key	Non null	Type	Référence	Intitulé	Commentaire
id	PK	O	int (11)		identifiant	unique
name		O	varchar (255)		nom de l'aliment	
type		O	varchar (255)		type de l'aliment	
expiration_date		O	varchar (255)		date d'expiration	date
date_of_purchase		O	varchar (255)		date d'achat	date
quantity		O	int (11)		quantité de l'aliment	
unit_qty		O	varchar (255)		l'unité de la quantité	
id_floor_id	FK	O	int (11)	id/floor	lien vers le floor	

Symfony - Twig

2.1 En tant qu'utilisateur, je veux pouvoir m'enregistrer

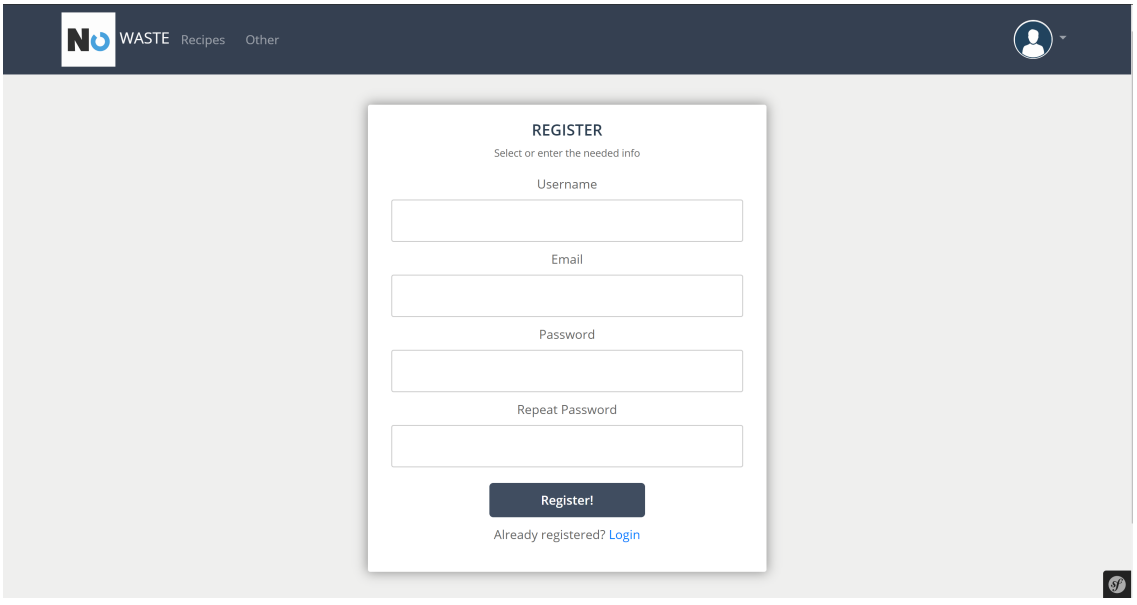

FIGURE 2.1

Ceci est donc la page rendu quand l'utilisateur veut s'enregistrer sur le site pour pouvoir créer un ou des frigo(s) par après. Quand l'utilisateur arrive sur le site, il est redirectionné vers la page d'acceuil qui est pour l'instant très modeste contenant seulement un caroussel comme montrer dans la section (...) dans mockup. À partir de cette page il peut se redirectionner vers la page "Sign up" ou "Register" en cliquant sur l'icone utilisateur dans le bar de menu en haut à droite. Puisque l'utilisateur ne sera pas connecté à ce moment-là, un dropdown contenant 2 boutons sera exposé comme montrer sur la page l'image suivante :

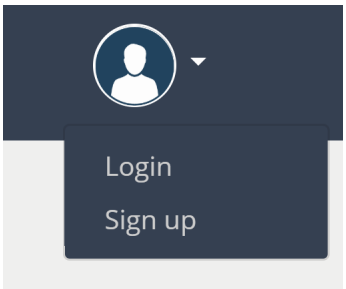

FIGURE 2.2

Un bouton pour aller sur la page "login" et l'autre pour aller sur la page "Register". L'utilisateur suit donc les étapes suivantes :

1. il clique sur le bouton "Register" dans le dropdown
2. le formulaire apparaît
3. il rempli le username et l'email qui doivent tous deux être unique
4. il rempli les password qui doivent être identiques
5. il clique sur "Register"
6. si tout se passe bien, il sera redirectionné vers la page de login.

Les forms sont tous sauf pour le form "login" généré à l'aide de "FormBuilderInterface". Ensuite il est aussi possible pour l'utisateur de se redirectionner vers la page "Login" s'il est déjà enregisté par le lien se trouvant en dessous du bouton "Register" ou par le drowdown.

2.1.1 Gestion des erreurs et sécurité

Si l'utilisateur entre un username ou email déjà utiliser, ceci causera une erreur et symfony renvoie pour chaque widget invalide l'erreur correspondante.
Si l'utilisateur entre deux password différent, ceci génèrera aussi une erreur (l'erreur affichée est à améliorer, ne spécifie pas le problème).

Pour ce qui est sécurité, quand le formulaire est valide, le controlleur s'occupe d'enrgister les valeur. Avant d'enregister le tout via doctrine, celui-ci va encoder le password avec l'algorithm de l'encoder mentionner dans le "security.yaml".

2.1.2 partie de codes significative

La partie la plus significative est :

- la fonction ***"register()"*** dans le UserController.php qui s'occupe de l'affichage de la page "register" avec le bon form et de l'enregistrement des données entrée par l'utilisateur dans la base de données.

2.1.3 Lien vers script concerné

- UserController.php
- security.yaml
- register.html.twig
- UserType.php
- User.php

2.1.4 Diagramme de séquence

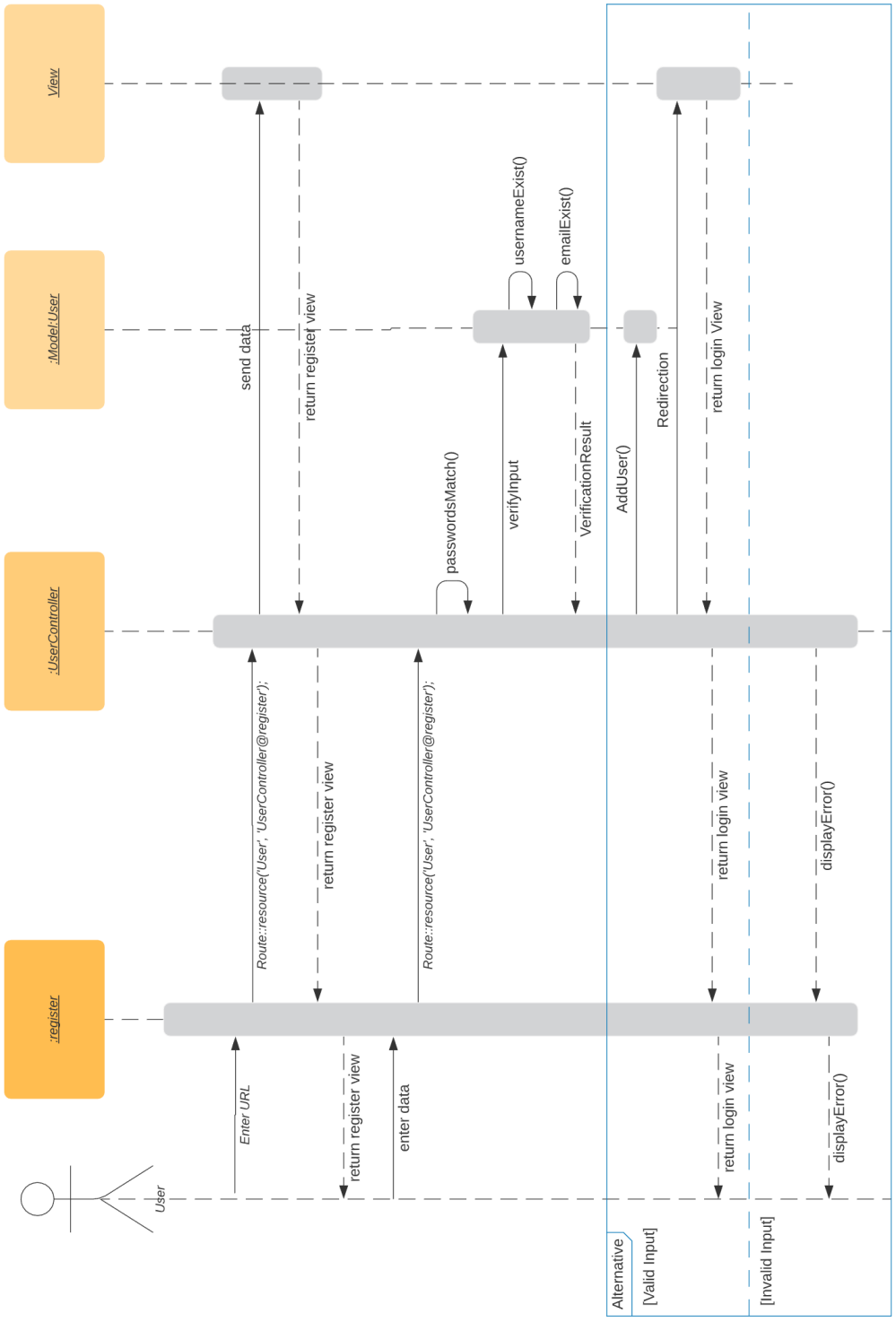

FIGURE 2.3: Diagramme de séquence pour l'enregistrement d'un utilisateur

2.2 En tant qu'utilisateur, je veux pouvoir me connecter

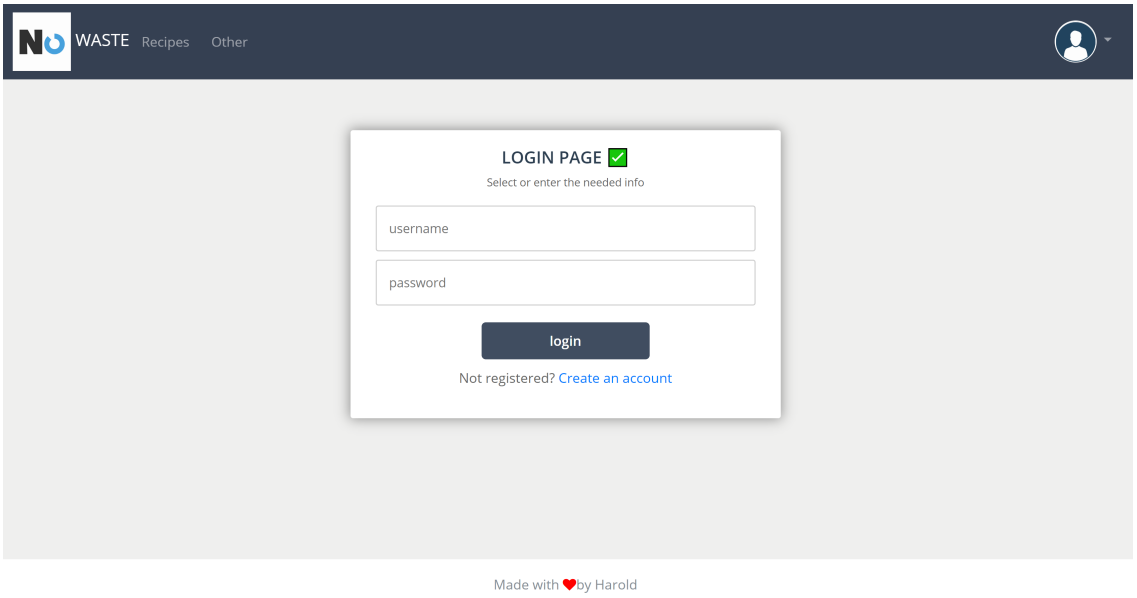

FIGURE 2.4

Ceci est donc la page rendu quand l'utilisateur veut se connecter. Pour s'y rendre, c'est la même démarche que pour s'enregister sauf qu'ici on clique sur le bouton "Login".

L'utilisateur suit dans cette partie les étapes suivantes :

1. il entre son username
2. il entre son password
3. il clique sur "login"

Comme pour "Register", il est possible via un lien ou via le dropdown, de se redirectionner vers la page "Register" si l'utilisateur n'est pas encore enregistré.

2.2.1 Gestion d'erreur et sécurité

Si le username et le password entrée par l'utilisateur ne correspondent pas, une erreur apparaît pour prévenir l'utilisateur.

2.2.2 partie de codes significative

Dans cette partie la plupart est réalisé par le bundle security de symfony. Il est tout de même nécessaire d'ajouter une fonction "login" pour pouvoir afficher la bonne page et y faire référence dans le security.yaml.

C'est d'ailleurs dans le sécurity.yaml que l'acces au différentes pages est authorisé. En effet, seul les pages "home", "login" et "register" sont accessibles sans être connecté.

2.2.3 Lien vers script concerné

- SecurityController.php
- security.yaml
- login.html.twig
- User.php

2.2.4 Diagramme de séquence

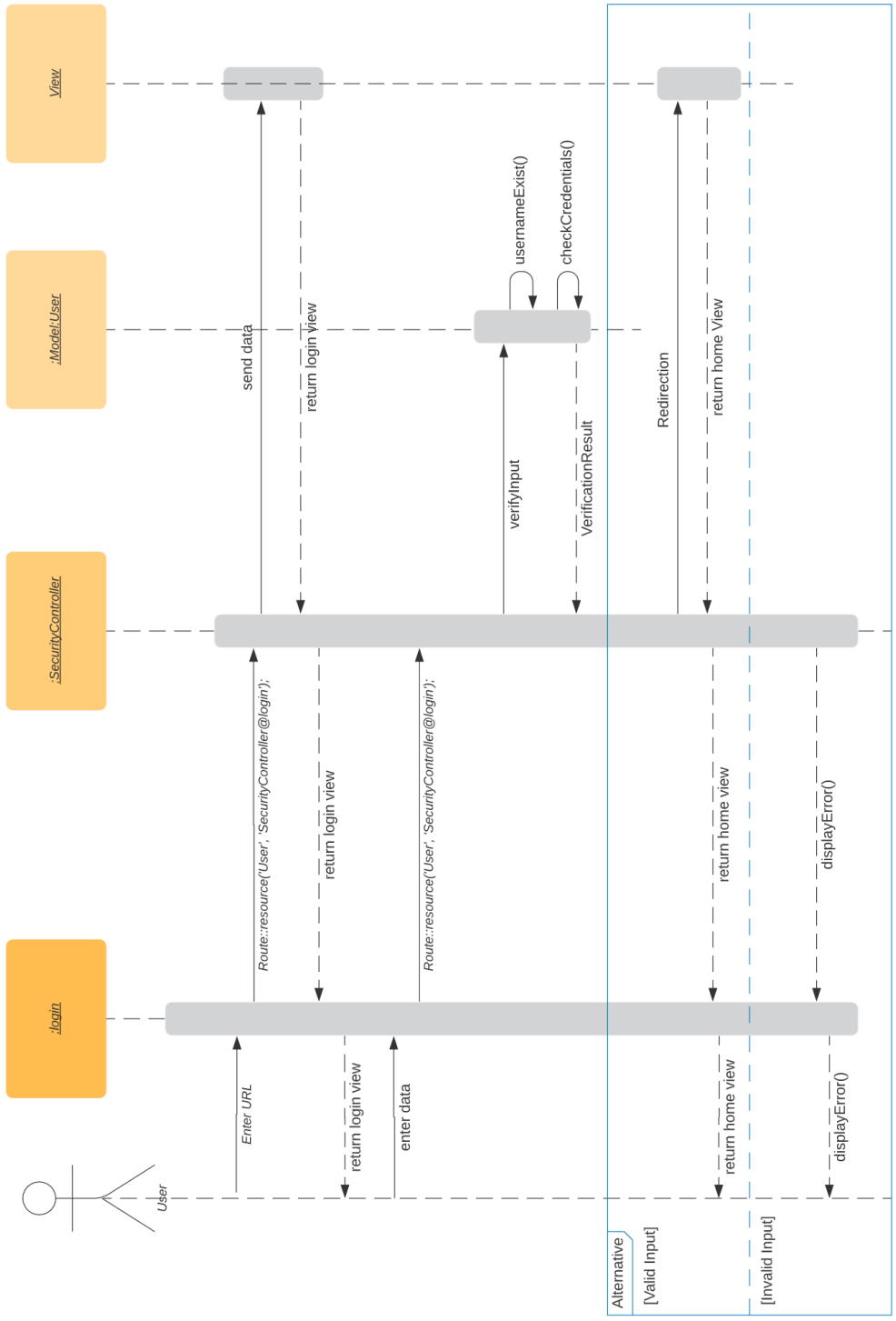

FIGURE 2.5: Diagramme de séquence pour connecter un utilisateur

2.3 En tant qu'utilisateur, je veux pouvoir retrouver les frigos que j'ai créé

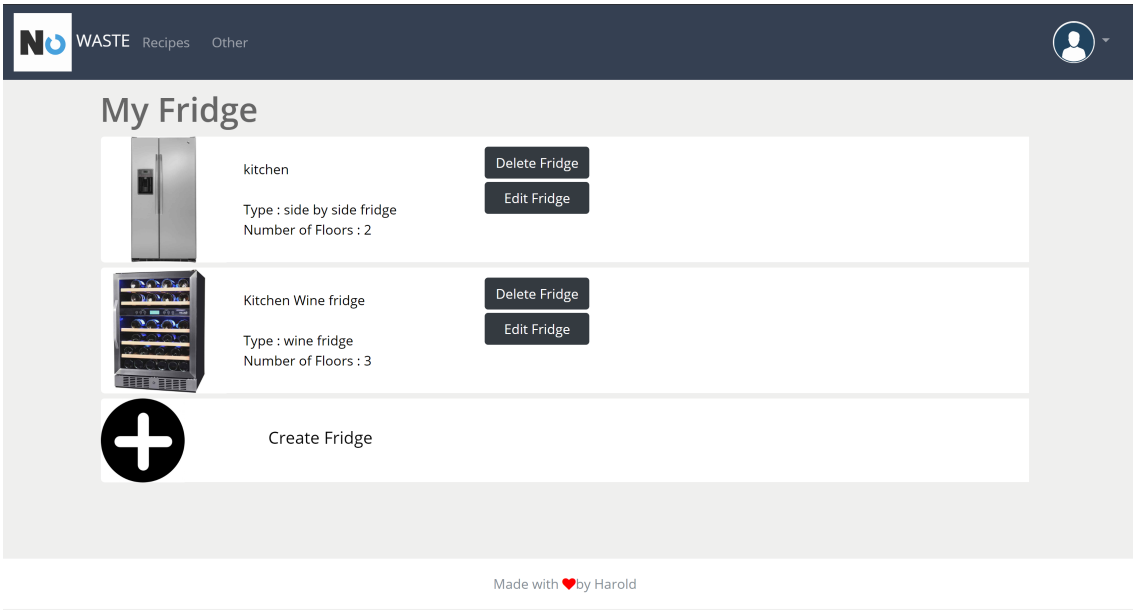

FIGURE 2.6

Ceci est donc la page rendu après s'être connecté ou quand l'utilisateur demande d'afficher la liste de ses frigos après avoir cliquer sur l'icône utilisateur qui affiche maintenant le dropdown suivant :

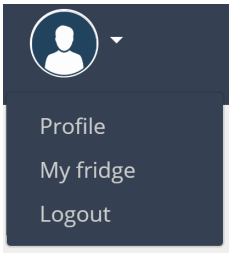

FIGURE 2.7

Chaque frigo est aligné l'un en dessous de l'autre contenant dans son box :

- une image selon le type de frigo
- son nom
- son type
- le nombre d'étages qu'il contient

- un bouton "delete" pour supprimer le frigo (voir plus loins)
- un bouton "edit" pour pouvoir modifier les caractéristique du frigo (voir plus loin)

2.3.1 Gestion d'erreur et sécurité

Une fois connecté, il n'est pas possible de modifier l'url pour afficher les frigos d'autres personnes. De plus la reqûete pour retrouver les frigos de l'utilisateur se fait par une fonction agissant sur l'utilisateur connecté.

2.3.2 Partie de code significative

La partie la plus significative est :

- la fonction ***"showFridgeRepo()"*** dans le FridgeController.php qui s'occupe de l'affichage et d'aller rechercher les bonne informations dans la base de donnée

2.3.3 Lien vers script concerné

- FridgeController.php
- showFridgeRep.html.twig
- Fridge.php

2.4 En tant qu'utilisateur, je veux pouvoir créer un frigo ou plusieurs

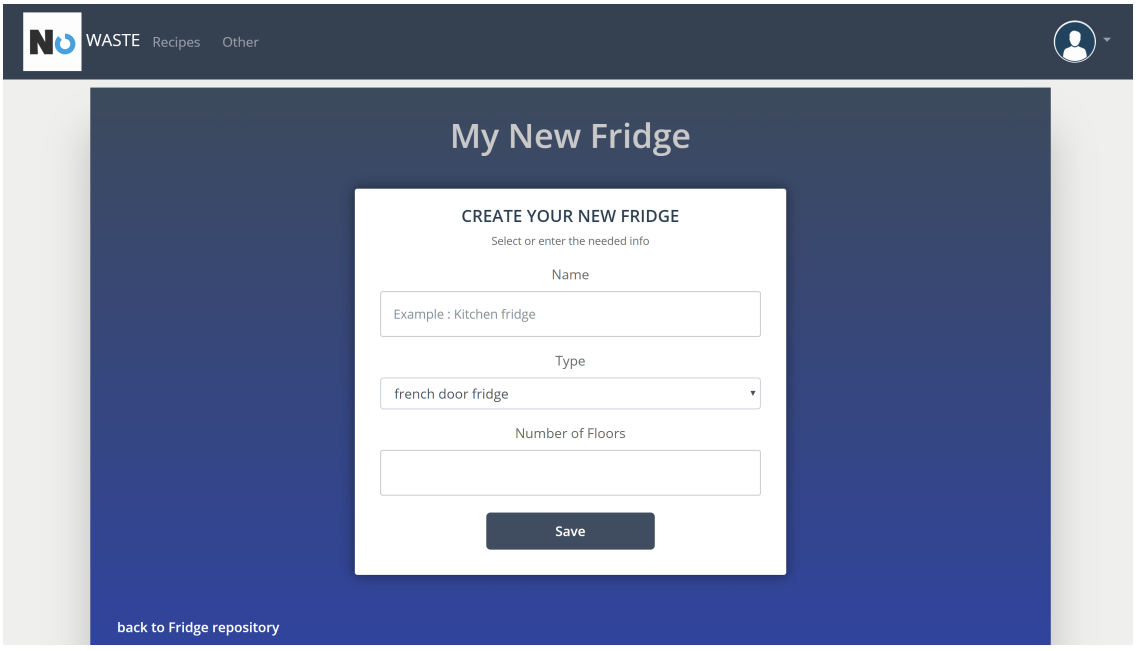

FIGURE 2.8

Pour se rendre sur et utiliser cette page, l'utilisateur suit les étapes suivantes (depuis la liste de frigo) :

1. Il clique sur le bloque "Create Fridge"

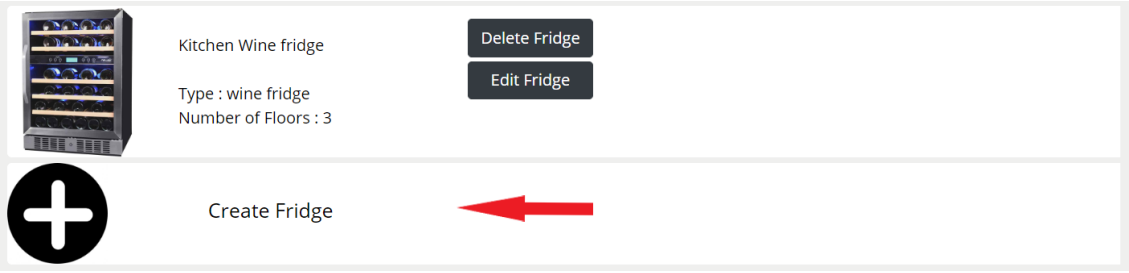

FIGURE 2.9

2. le formulaire apparaît
3. il encode le nom qu'il donnera à son frigo pour le différencier des autres
4. il choisit un type de frigo
5. il choisit enfin le nombre d'étages que le frigo contient

6. il clique sur save
7. si les données mentionnées sont bonnes, le submission à lieu
8. l'utilisateur est redirectionné vers la page contenant la liste des frigos

2.4.1 Gestion d'erreur et sécurité

Le form vérifie que les informations sont toutes encodées et correctes sinon une erreur s'affiche.

2.4.2 Partie de code significative

La partie la plus significative est :

- la fonction ***"newFridge(Request $request)"*** dans le FridgeController.php qui s'occupe de l'affichage et de créer un nouveau fridge dans la base de données. Celle-ci s'occupe aussi d'ajouter le bon lien vers l'image correspondante de frigo en fonction du type selectionné.

2.4.3 Lien vers script concerné

- FridgeController.php
- CreateFridge.html.twig
- _formFridge.html.twig
- FrideType.php
- Fridge.php

2.5 En tant qu'utilisateur, je veux pouvoir modifier mon frigo

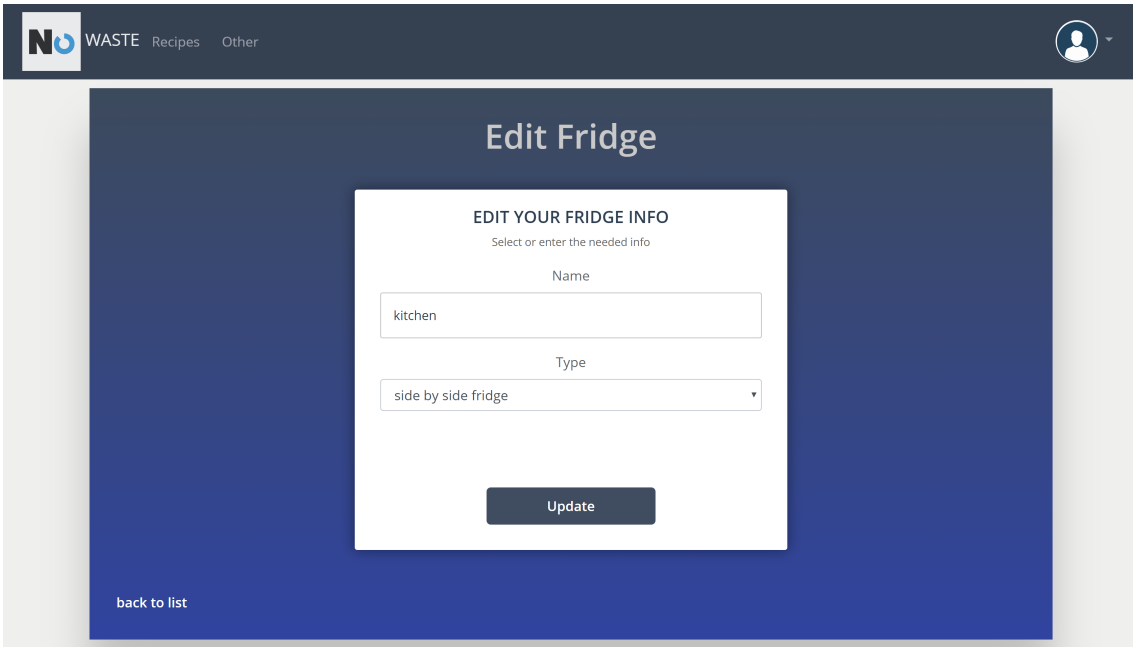

FIGURE 2.10

Pour se rendre sur la cette page, l'utilisateur doit avoir créer un fridge. Si c'est le cas, l'utilisateur suit les étapes suivantes (depuis la liste de frigo) :

1. Il clique sur le bouton ""Edit Fridge" du frigo en question

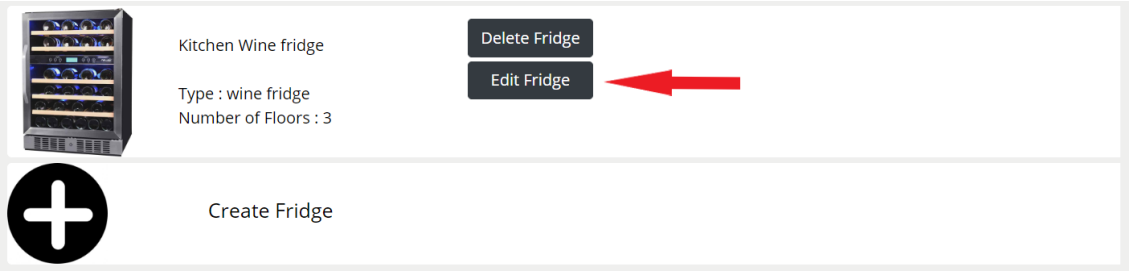

FIGURE 2.11

2. le formulaire apparaît
3. il modifie le nom de son frigo
4. il choisit un autre type de frigo s'il le souhaite
5. il clique sur save

6. si les données mentionnées sont bonnes, le submission à lieu et l'utilisateur est redirectionné vers la page contenant la liste des frigos

2.5.1 Gestion d'erreur et sécurité

Le form vérifie que les informations sont toutes encodée et correctes sinon une erreur s'affiche.

Vérification que c'est bien le détendeur du frigo qui essaie de modifier le frigo

2.5.2 Partie de code significative

La partie la plus significative est :

- la fonction ***"editFridge(Request $request, Fridge $fridge)"*** dans le FridgeController.php qui s'occupe de l'affichage et de modifier un fridge dans la base de données. Elle modifie le lien vers l'image frigo correspondant au type en même temps.

2.5.3 Lien vers script concerné

- FridgeController.php
- EditFridge.html.twig
- _formFridge.html.twig
- FrideType.php
- Fridge.php

2.6 En tant qu'utilisateur, je veux pouvoir supprimer mon frigo

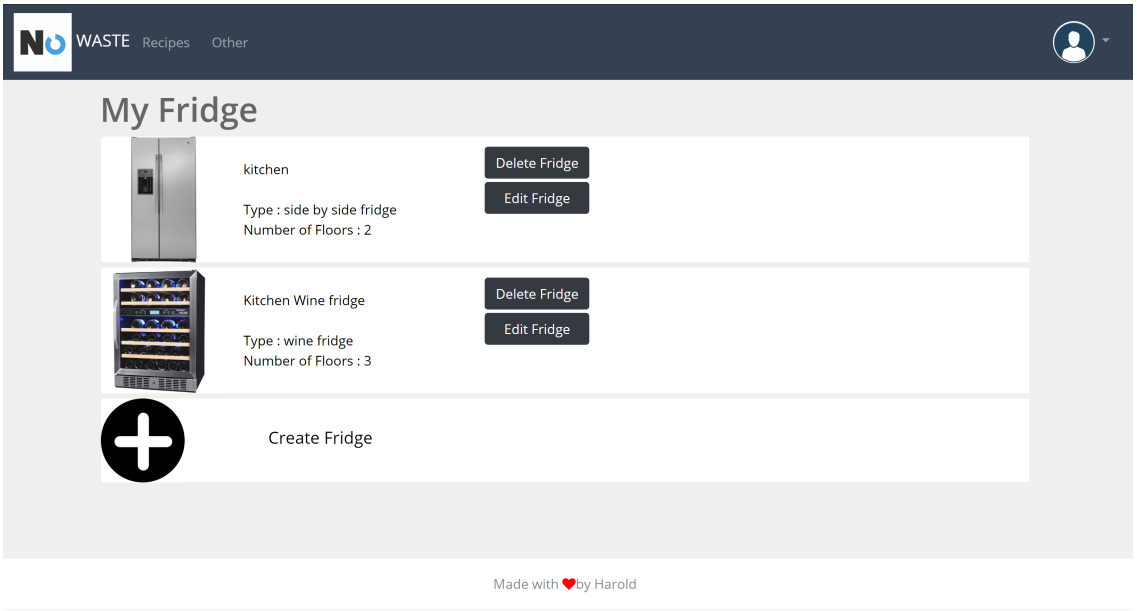

FIGURE 2.12

Pour pouvoir supprimer un frigo, l'utilisateur doit avoir créer un fridge. Si c'est le cas, l'utilisateur suit les étapes suivantes (depuis la liste de frigo) :

1. Il clique sur le bouton ""delete Fridge" du frigo en question

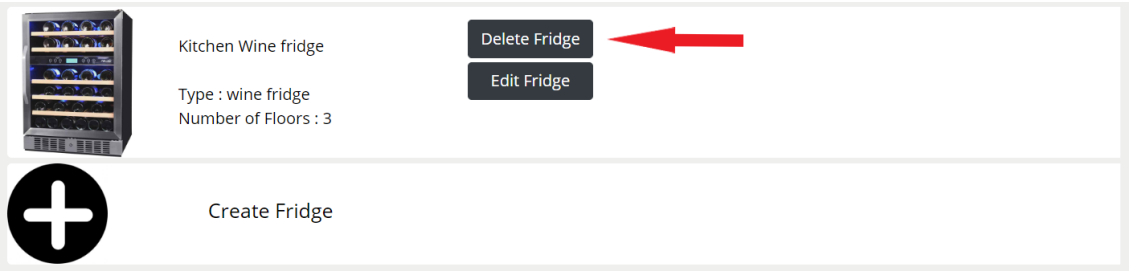

FIGURE 2.13

2. une alerte apparaît demandant le confirmation de l'action
3. il clique "OK" pour confirmer
4. si tout se passe bien, la page se recharge et le frigo est supprimé

2.6.1 Gestion d'erreur et sécurité

Vérification que c'est bien le détendeur du frigo qui essaie de modifier le frigo

2.6.2 Partie de code significative

La partie la plus significative est :

- la fonction ***"deleteFridge(Request $request, Fridge $fridge)"*** dans le FridgeController.php qui s'occupe de l'affichage et de supprimer un fridge dans la base de données.

2.6.3 Lien vers script concerné

- FridgeController.php
- _deleteFridge.html.twig
- Fridge.php

2.7 En tant qu'utilisateur, je veux pouvoir lister les étages de mon frigo

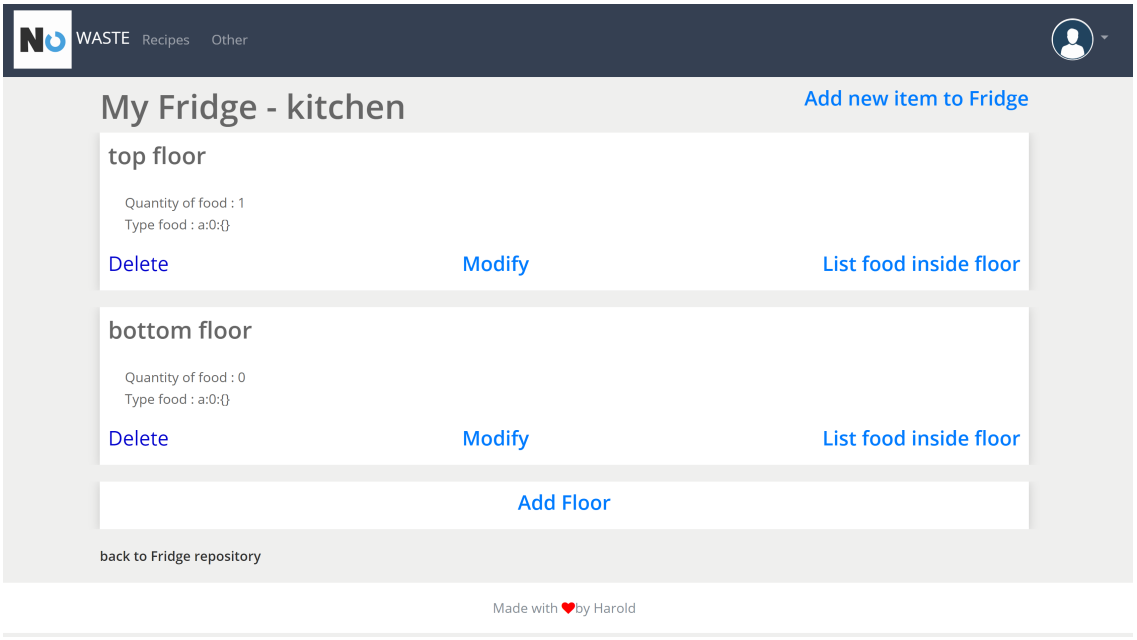

FIGURE 2.14

Pour pouvoir afficher les étages d'un frigo, l'utilisateur doit avoir créé un fridge et avoir créé un étage ou selectionné un nombre plus grand que zéro pendant la création d'un frigo pour le nombre d'étages. Si c'est le cas, l'utilisateur peut alors cliquer sur l'image du frigo (il doit se trouver dans la page de ses frigos) et il sera redirectionné vers une page contenant la liste des étages se trouvant dans le frigo.

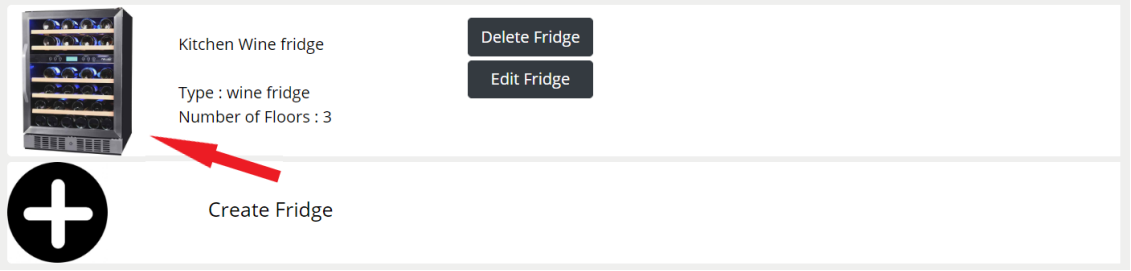

FIGURE 2.15

Comme pour les frigos, les étages sont alignés l'un en dessous de l'autre. Chaque bloque représente donc un étage du frigo. On peut y retrouver les choses suivantes :

- le nom de l'étage
- la liste des types de food se trouvant dedans
- le nombre d'aliment se trouvant dedans
- un bouton "Delete" pour supprimer l'étage (voir plus loins)
- un bouton "edit" pour pouvoir modifier les caractéristique de l'étage (voir plus loin)
- un bouton "List food inside floor" pour afficher la liste d'aliments se trouvant dans l'étage

2.7.1 Gestion d'erreur et sécurité

Vérification que c'est bien le détendeur du frigo qui essaie d'afficher les étages du frigo.

2.7.2 Partie de code significative

La partie la plus significative est :

- la fonction ***"index(...)"*** dans le FloorController.php qui s'occupe de l'affichage et d'aller chercher les données dans la base de données
- la fonction ***"generateFloorsFridge(...)"*** dans le FloorController.php qui qui s'occupe de générer les floors à la création du fridge ou de générer le floor ajouté par après.
- la fonction ***"getFloorsFridge($id_fridge)"*** dans le FloorController.php qui s'occupe d'aller rechercher les étages correspondant au fridge dans lequel on est.

2.7.3 Lien vers script concerné

- FloorController.php
- index.html.twig
- Floor.php

2.8 En tant qu'utilisateur, je veux pouvoir ajouter des étages à mon frigo

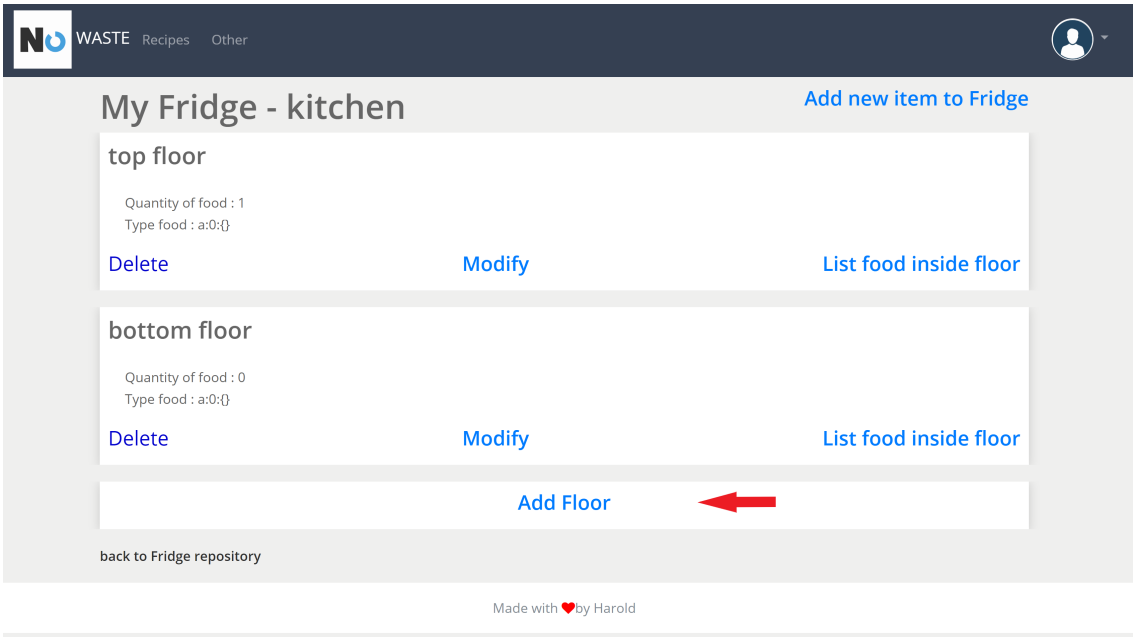

FIGURE 2.16

Pour pouvoir ajouter des étages à un frigo, l'utilisateur doit avoir créé un fridge. Après avoir suivi la première étape du User Storie précédent, il est redirectionné vers une page comme montrer à l'image 4.20. Il suffit à l'utilisateur de cliquer sur "Add Floor" et un nouveau floor se généré automatiquement. Il pourra personnalisé se fridge par après (expliquer plus loin)

2.8.1 Gestion d'erreur et sécurité

Vérification que c'est bien le détendeur du frigo qui essaie d'ajouter un étage au frigo.

2.8.2 Partie de code significative

La partie la plus significative est :

- la fonction ***"addFloor(...)"*** dans le FloorController.php qui s'occupe tout simplement d'incrémenté le nombre d'étages du frigo (grâce à une autre fonction nommé ***"changeNbrFloor(...))"*** et puis redirectionne vers la fonction "index()" qui refait le même processus que mentionné au dessus.

2.8.3 Lien vers script concerné

- FloorController.php
- FloorType.php
- Floor.php

2.9 En tant qu'utilisateur, je veux pouvoir modifier les étages de mon frigo

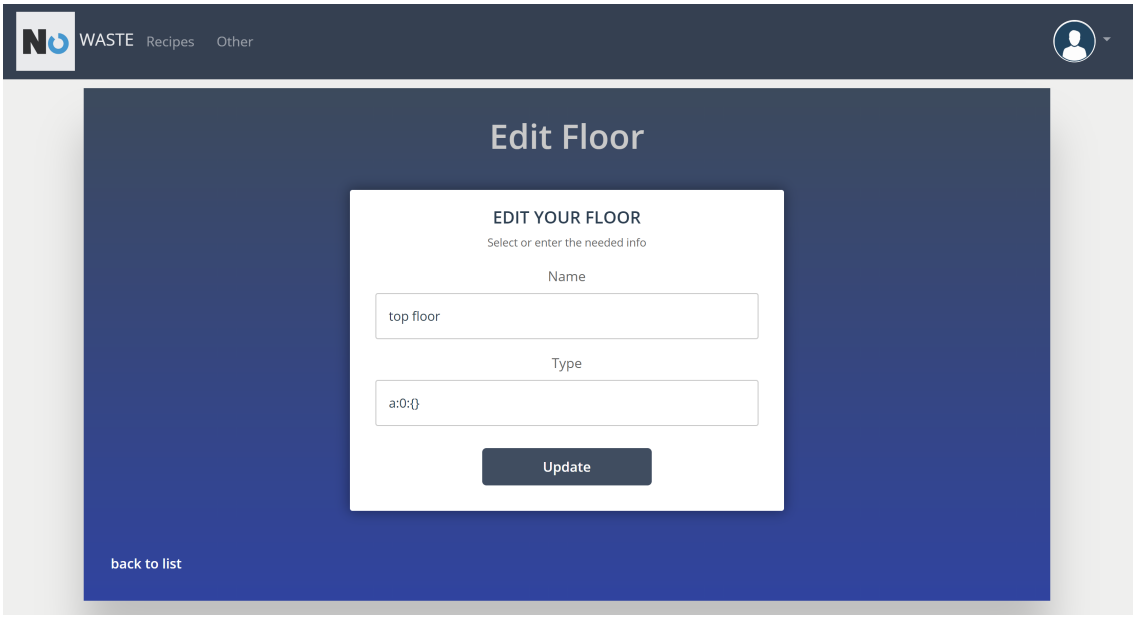

FIGURE 2.17

Pour pouvoir modifier un étage d'un frigo, l'utilisateur doit avoir créé un fridge et avoir créé un étage ou selectionné un nombre plus grand que zéro pendant la création d'un frigo pour le nombre d'étages.

Comme dit plus haut, les étages sont aligné l'un en dessous de l'autre. L'utilisateur peut suivre les étapes suivantes :

1. Il clique sur le bouton "Modify" du floor en question

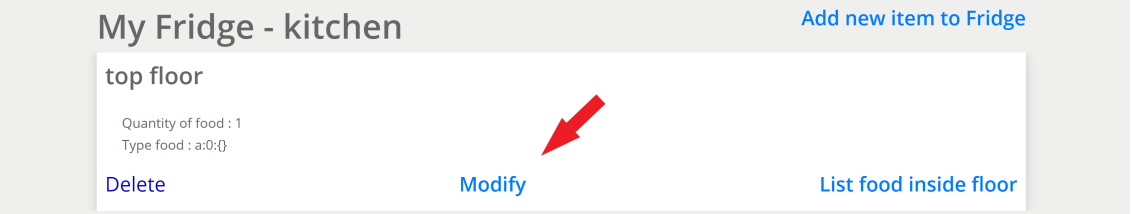

FIGURE 2.18

2. un formulaire apparaît
3. il modifie le nom ou les types comme il le veut

4. il clique "Update" pour confirmer
5. si tout se passe bien, il est redirectionné vers la liste d'étage

2.9.1 Gestion d'erreur et sécurité

Vérification que c'est bien le détendeur du frigo qui essaie de modifier l'étages du frigo.

2.9.2 Partie de code significative

La partie la plus significative est :

- la fonction ***"editFloor(...)"*** dans le FloorController.php qui s'occupe de l'affichage et d'aller chercher et modifier les données dans la base de données

2.9.3 Lien vers script concerné

- FloorController.php
- edit.html.twig
- _form.html.twig
- FloorType.php
- Floor.php

2.10 En tant qu'utilisateur, je veux pouvoir supprimer les étages de mon frigo

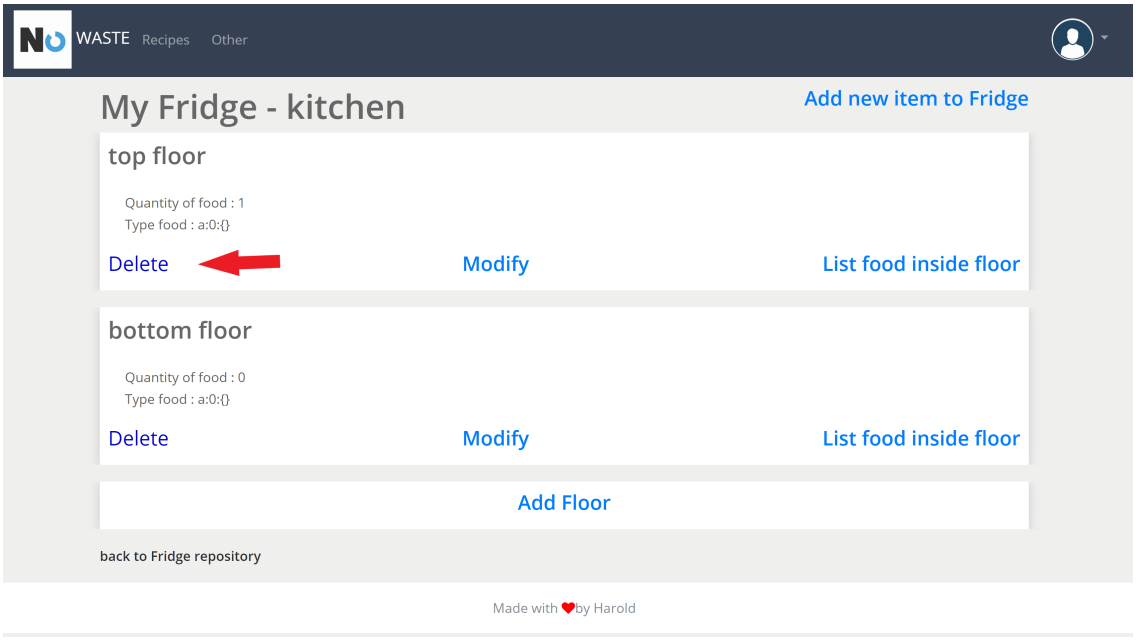

FIGURE 2.19

Pour pouvoir supprimer un étage d'un frigo, l'utilisateur doit avoir créé un fridge et avoir créé un étage ou selectionné un nombre plus grand que zéro pendant la création d'un frigo pour le nombre d'étages. Si c'est le cas, l'utilisateur peut suivre les étapes suivant depuis la liste des étages :

1. Il clique sur le bouton "Delete" du floor en question
2. un message d'alerte apparaît
3. il clique sur "OK" pour confirmer la suppression
4. si tout se passe bien, la page se recharge sans l'étage supprimé

2.10.1 Gestion d'erreur et sécurité

Vérification que c'est bien le détendeur du frigo qui essaie de supprimer un étage du frigo.

2.10.2 Partie de code significative

La partie la plus significative est :

- la fonction ***"deleteFloor(...)"*** dans le FloorController.php qui s'occupe de supprimer le floor en question dans la base de données

2.10.3 Lien vers script concerné

- FloorController.php
- _delete_form.html.twig
- Floor.php

2.11 En tant qu'utilisateur, je veux pouvoir lister les aliments se trouvant dans un étage

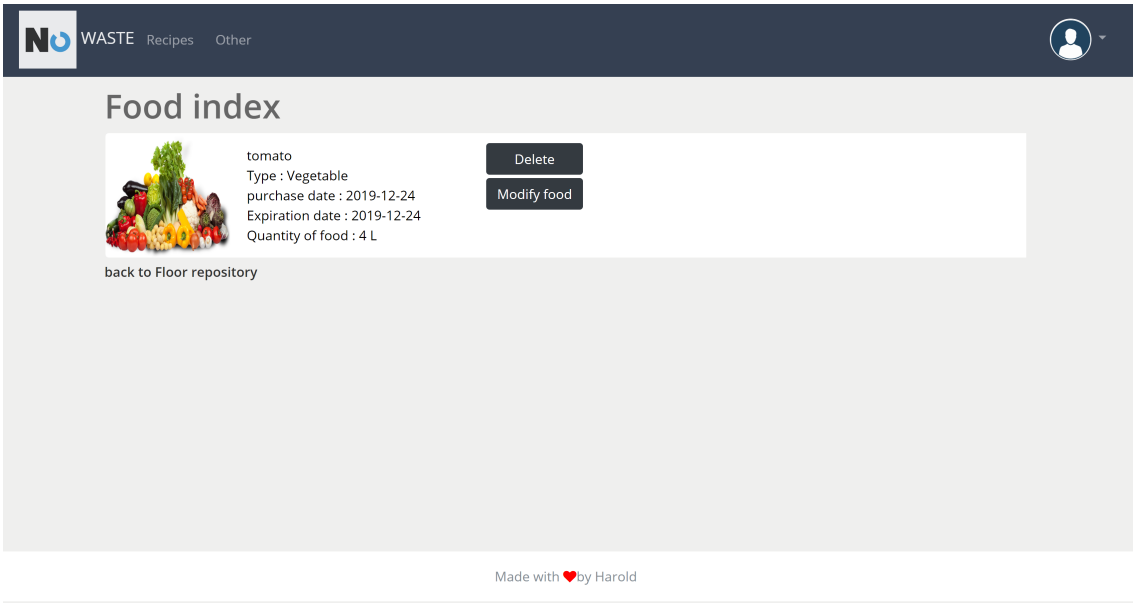

FIGURE 2.20

Pour pouvoir afficher les éléments d'un étage il faut évidemment avoir créer un frigo et au moins un étage. Si c'est le cas, l'utilisateur peut suivre les étapes suivante pour afficher la liste d'aliments d'un étage (depuis la liste d'étages) :

1. Il clique sur le bouton "List food inside floor" du floor en question

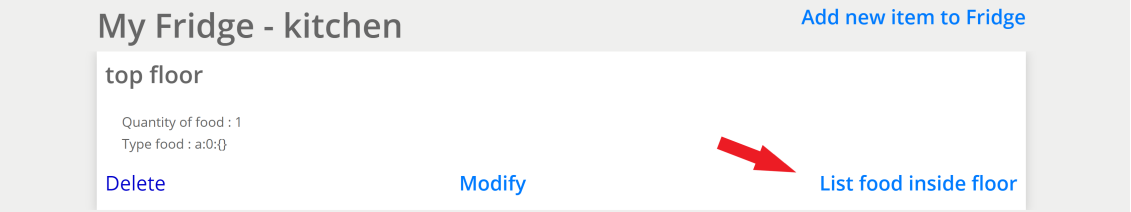

FIGURE 2.21

2. la liste des aliments apparaît aligné l'un en dessous de l'autre

Comme dit plus haut, les aliments sont aligné l'un en dessous de l'autre. Chaque box représente un aliment différent contenant les attributs suivants :

- le nom de l'aliment

- le type de l'aliment
- la date d'achat de l'aliment
- la date d'expiration de l'aliment
- la quantité de l'aliment
- un bouton "Delete"
- un bouton "Modify food"

2.11.1 Gestion d'erreur et sécurité

Vérification que c'est bien le détendeur du frigo qui essaie d'afficher les aliments d'un étages du frigo.

2.11.2 Partie de code significative

La partie la plus significative est :

- la fonction ***"index(...)"*** dans le FoodController.php qui s'occupe de l'affichage et d'aller chercher les données dans la base de données

2.11.3 Lien vers script concerné

- FoodController.php
- index.html.twig
- Food.php

2.12 En tant qu'utilisateur, je veux pouvoir afficher un aliment spécifique

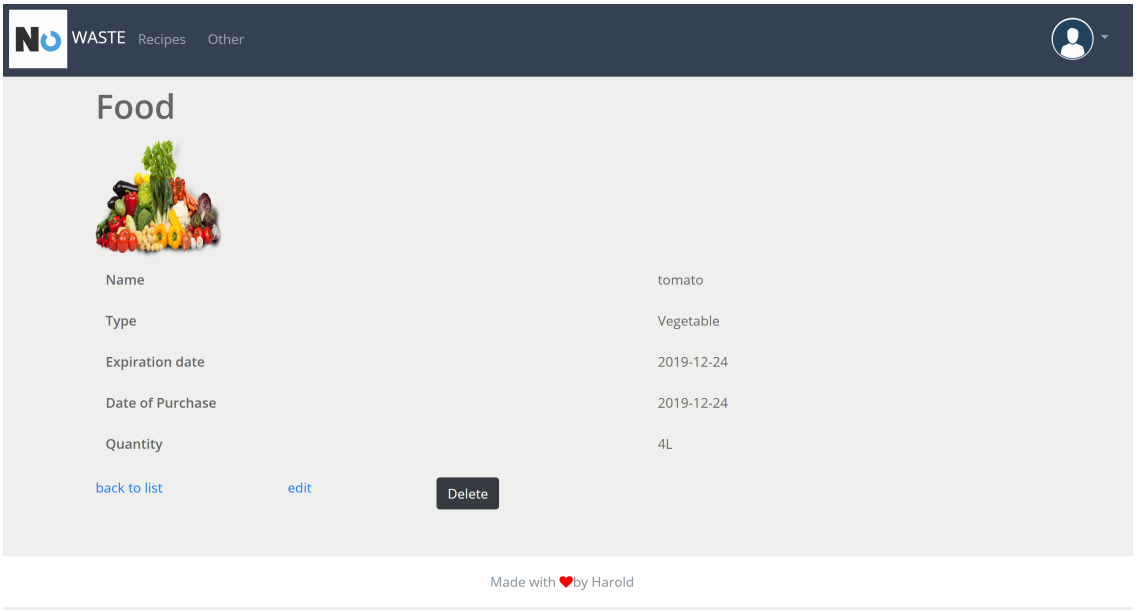

FIGURE 2.22

Pour pouvoir afficher un aliment d'un étage il faut évidemment avoir créer un frigo et au moins un étage. Si c'est le cas, l'utilisateur peut suivre les étapes suivante pour afficher la liste d'aliments d'un étage (depuis la liste d''aliments) :

1. Il clique sur l'image du food en question (en passant dessus le curseur détecte que c'est un lien)

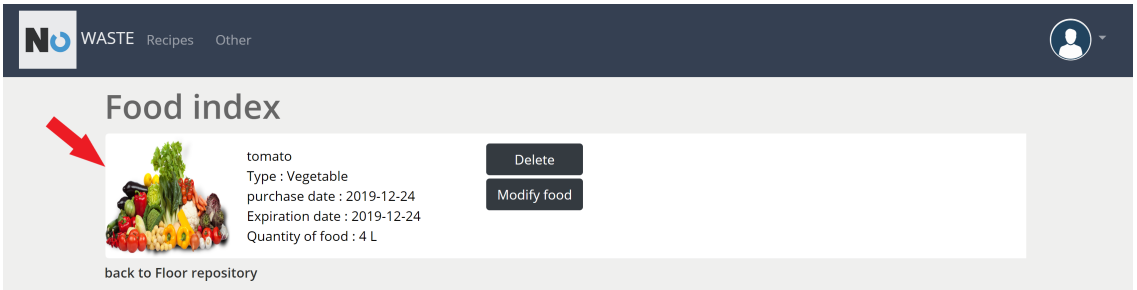

FIGURE 2.23

2. une nouvelle page apparait contenant l'aliment seul en détail

Cette page contient les mêmes éléments que la liste auparavant mais disposer d'une manière plus clair.

2.12.1 Gestion d'erreur et sécurité

Vérification que c'est bien le détendeur du frigo qui essaie d'afficher l'aliment d'un étages du frigo.

2.12.2 Partie de code significative

La partie la plus significative est :

- la fonction ***"show(...)"*** dans le FoodController.php qui s'occupe de l'affichage de l'aliment.

2.12.3 Lien vers script concerné

- FoodController.php
- show.html.twig
- Food.php

2.13 En tant qu'utilisateur, je veux pouvoir ajouter des aliments à mon frigo

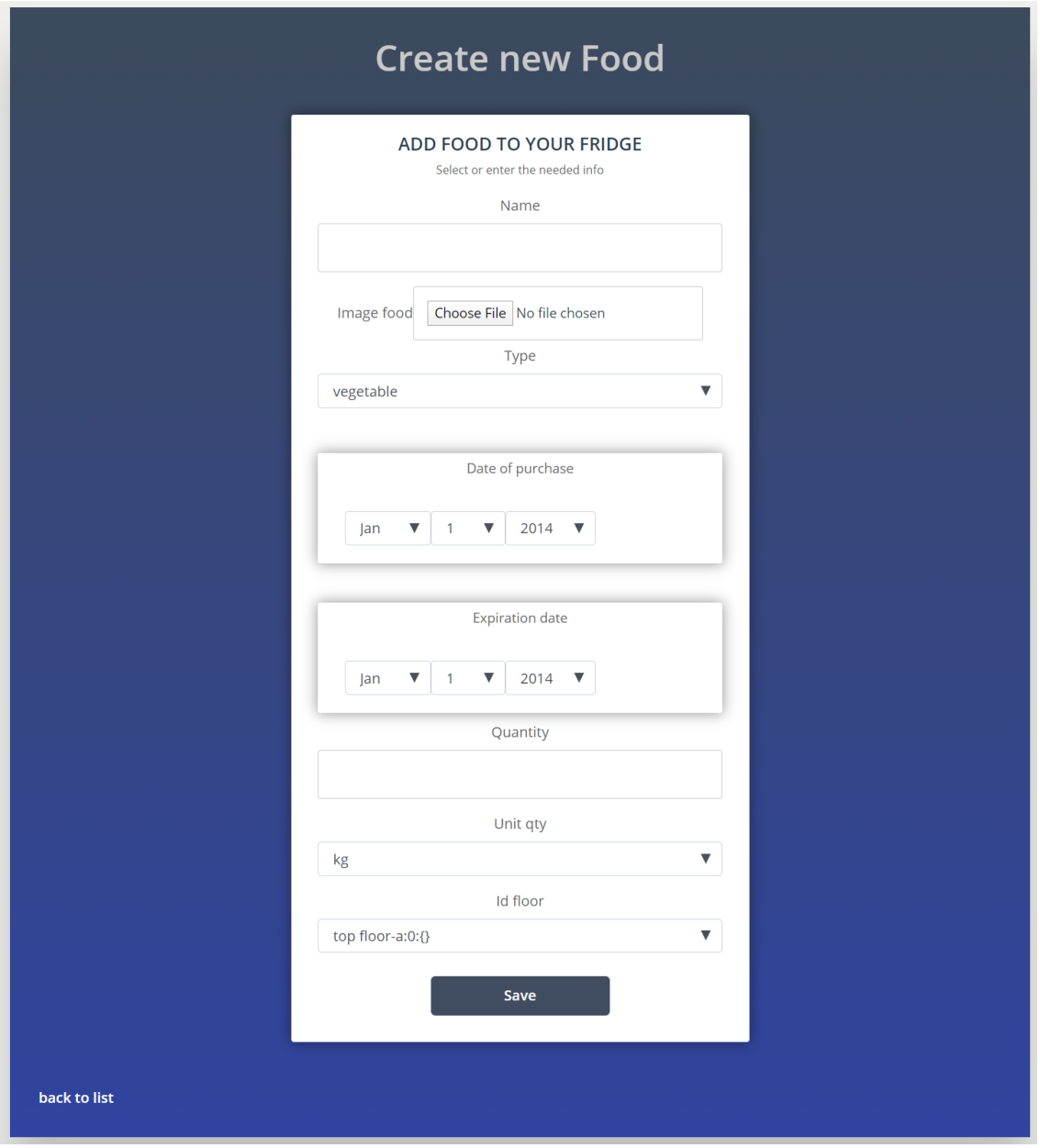

FIGURE 2.24

Pour pouvoir ajouter un élément à un étage, à nouveau il faut qu'un frigo ait été créé et de même pour un floor. Si c'est le cas, l'utilisateur peut suivre les étapes suivantes (depuis la liste des étages, pas possible pour l'instant depuis la liste des aliments) :

1. Il clique sur le bouton "Modify" du floor en question

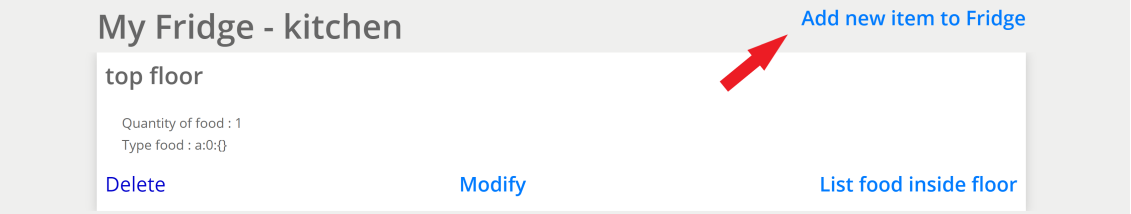

FIGURE 2.25

2. un formulaire apparaît
3. il encode le nom de l'aliment
4. il peut choisir de mettre sa propre image de son aliment (pas obligatoire)
5. il selectionne le type de l'aliment
6. il selectionne la date d'achat et date d'expiration
7. il encode la quantité de l'aliment et selectionne l'unité de la quantité
8. enfin il sélectionne l'étage auquel il veut rajouter l'aliment.
9. il clique "Save" pour confirmer
10. si tout se passe bien, il est redirectionné vers la liste d'étages

2.13.1 Gestion d'erreur et sécurité

Vérification que c'est bien le détendeur du frigo qui essaie d'ajouter un aliment à l'étages du frigo.

2.13.2 Partie de code significative

Les partie les plus significative sont :

- la fonction ***"new(...)"*** dans le FoodController.php qui s'occupe de l'affichage du form et d'aller créer un nouvel aliment dans la base de données
- la fonction ***"uploadImage(...)"*** dans le FoodController.php qui s'occupe de télécharger l'image selectionné par l'utilisateur et d'enregistrer le lien vers cette image dans la base de données. (l'image est enregistré dans la partie backend)

2.13.3 Lien vers script concerné

- FoodController.php
- new.html.twig
- _form.html.twig
- FoodType.php
- Food.php

2.14 En tant qu'utilisateur, je veux pouvoir modifier des aliments de mon frigo

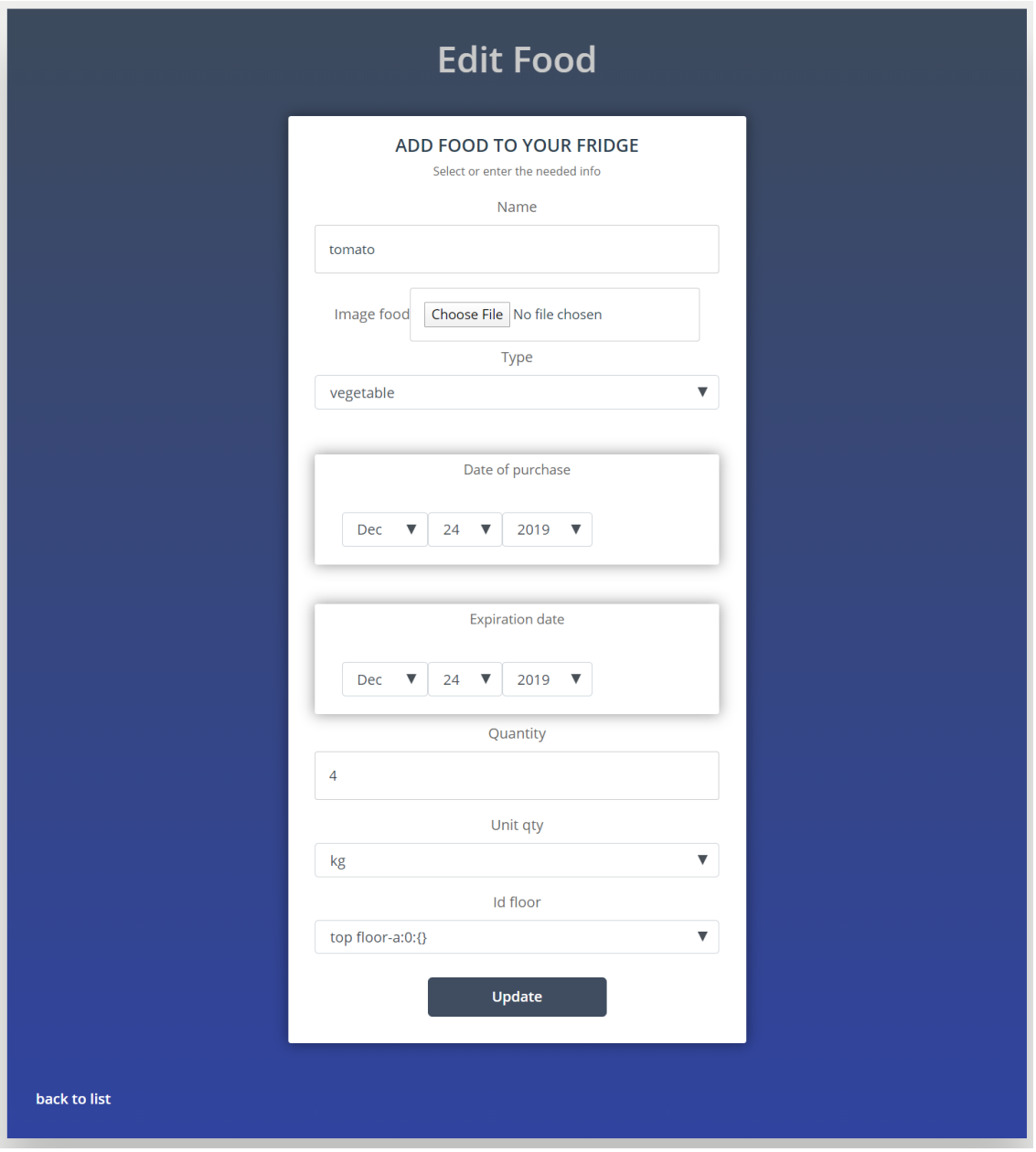

FIGURE 2.26

Pour pouvoir modifier un aliment d'un étage d'un frigo, l'utilisateur doit avoir créé un fridge et de même pour un étage. Si c'est le cas l'utilisateur peut suivre les étapes suivantes (depuis la liste d'aliments) :

1. Il clique sur le bouton "Modify food" du food en question

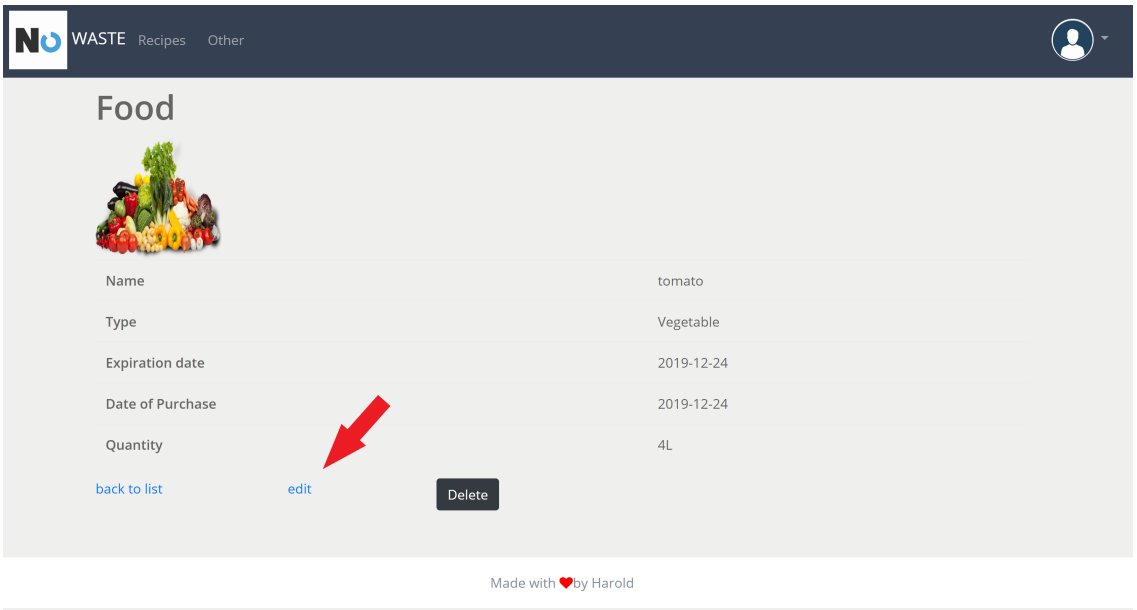

FIGURE 2.27

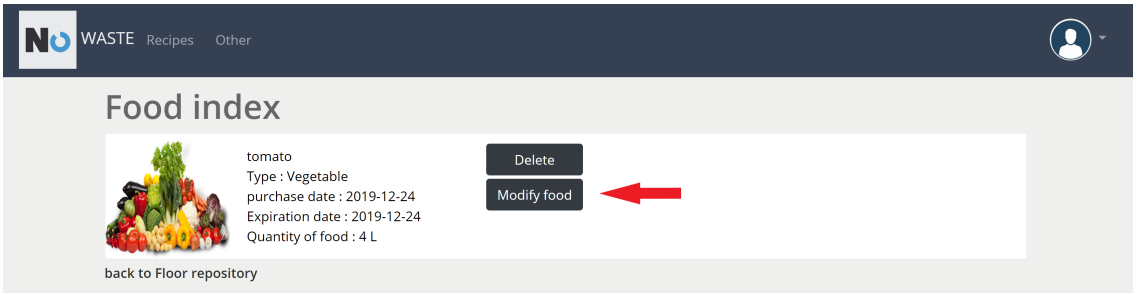

FIGURE 2.28

2. un formulaire apparaît
3. il modifie les éléments (voir creation food) qu'il veut
4. il clique "Update" pour confirmer
5. si tout se passe bien, il est redirectionné vers la liste des aliments

2.14.1 Gestion d'erreur et sécurité

Vérification que c'est bien le détendeur du frigo qui essaie de modifier l'aliment de l'étages du frigo.

2.14.2 Partie de code significative

La partie la plus significative est :

- la fonction ***"edit(...)"*** dans le FoodController.php qui s'occupe de l'affichage du form et d'aller chercher et modifier les données dans la base de données
- la fonction ***"uploadImage(...)"*** dans le FoodController.php qui s'occupe de télécharger l'image selectionné par l'utilisateur et d'enregistrer le lien vers cette image dans la base de données. (l'image est enregistré dans la partie backend)

2.14.3 Lien vers script concerné

- FoodController.php
- edit.html.twig
- _form.html.twig
- FoodType.php
- Food.php

2.15 En tant qu'utilisateur, je veux pouvoir supprimer des aliments de mon frigo

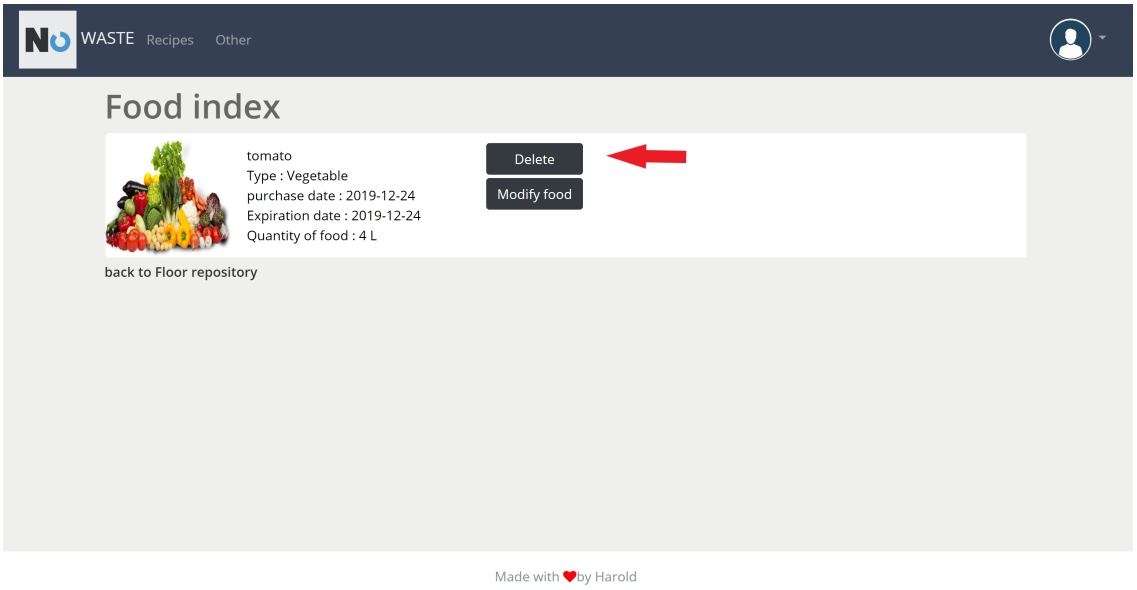

FIGURE 2.29

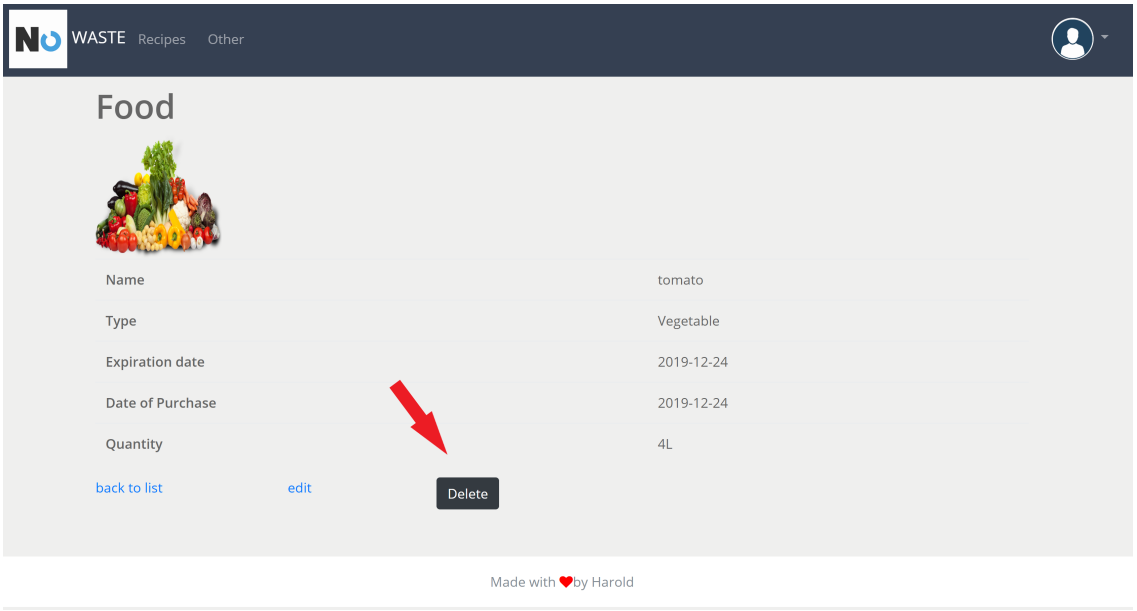

FIGURE 2.30

Pour pouvoir supprimer un aliment d'un étage d'un frigo, l'utilisateur doit avoir créé un fridge et de même pour un étage. Si c'est le cas l'utilisateur peut suivre les étapes suivantes (depuis la liste d'aliments) :

1. Il clique sur le bouton "Delete" du food en question

2. un message d'alerte apparaît demandant confirmation
3. il clique sur "OK" pour confirmer
4. si tout se passe bien, la page se recharge sans le food ou il est redirectionné dans le deuxième cas vers la liste des aliments

2.15.1 Gestion d'erreur et sécurité

Vérification que c'est bien le détendeur du frigo qui essaie de modifier l'aliment de l'étages du frigo.

2.15.2 Partie de code significative

La partie la plus significative est :

- la fonction ***"delete(...)"*** dans le FoodController.php qui s'occupe de supprimer l'aliment dans la base de données

2.15.3 Lien vers script concerné

- FoodController.php
- _delete_form.html.twig
- _form.html.twig
- FoodType.php
- Food.php

API

3.1 Tableau récapitulatif

Méthode	Requête	Contrôleur (.php)	Méthode	Action
POST	/api/user/register	UserControllor	register	register user in database
POST	/api/user/login	SecurityController	login	verify user exist and credentials are correct
GET	/api/fridge/	FridgeController	index	retourne la liste des fridge de l'utilisateur
POST	/api/fridge/	FridgeController	newFrigde	ajoute un fridge à l'utilisateur
PUT	/api/fridge/{id}	FridgeController	editFridge	modifie le fridge sous "id"
DELETE	/api/fridge/{id}	FridgeController	deleteFridge	supprime le fridge sous "id"
GET	/api/floors/	FloorController	index	retourne la liste des étage d'un certain fridge
POST	/api/floors/	FloorController	addFloor	ajoute un étage au fridge
PUT	/api/floors/{id}	FloorController	editFloor	modifie le floor sous "id"
DELETE	/api/floors/{id}	FloorController	deleteFloor	supprime le floor sous "id"
GET	/api/food/	FoodController	index	retourne la liste des food dans un étage
POST	/api/food/	FoodController	newFood	ajoute un ffood à l'étage
PUT	/api/food/{id}	FoodController	editFood	modifie le food sous "id"
DELETE	/api/food/{id}	FoodController	deleteFood	supprime le food sous "id"

3.2 Requêtes: Détaillées

3.2.1 User/register

- Ajout d'un utilisateur
- Méthode POST

3.2.1.1 Requête

Paramètre du path : aucun

Content-type : Application/json

Body

Propriété	Type	Obligatoire	Description
email	string	oui	email de l'utilisateur, doit être unique
username	string	oui	nom d'utilisateur
password	string	oui	password de l'utilisateur, doit être minimum de taille 8
passwordConfirmation	string	oui	password répété, doit être le même que password

exemple JSON :

```
{
        "email": "harold@mail.com",
        "username": "harold",
        "password": "harold9898",
        "passwordConfirmation": "harold9898"
}
```

3.2.1.2 Réponse: Réussite

Réussite Retourne code 200

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
message	string	oui	message confirmant la réussite de l'enregistrement

Exemple JSON :

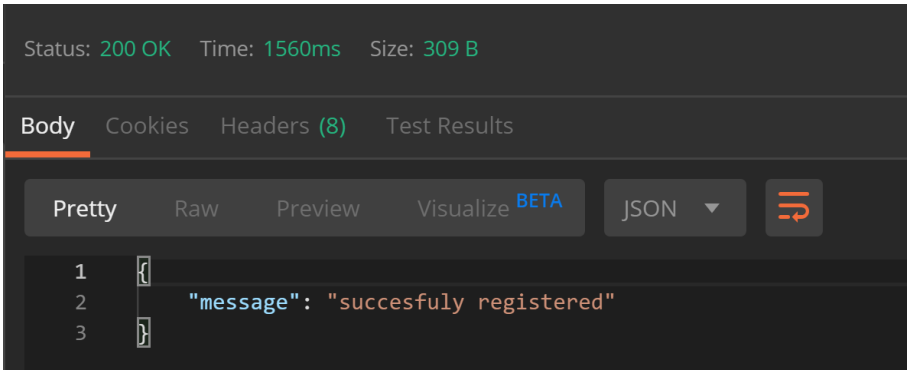

FIGURE 3.1

3.2.1.3 Réponse: erreurs

Erreur Retourne code 400

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
errors	Array	oui	Liste d'erreurs rencontré pendant l'enregistrement

Erreur en cas d'un utilisateur ou username déjà utilisé

Exemple JSON :

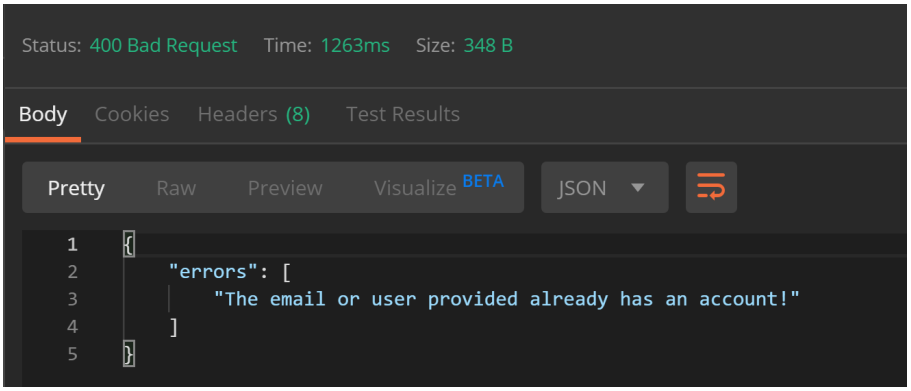

FIGURE 3.2

Erreur en cas d'un password trop court ou/et que les password donner ne match pas. (Cependant cet erreur ne devrait jamais être envoyé car l'erreur est géré dans le frontend.
Exemple JSON :

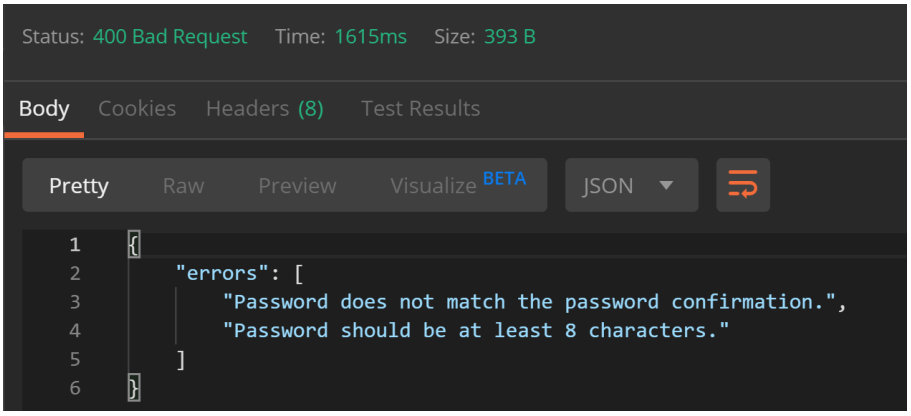

FIGURE 3.3

3.2.1.4 Extrait de code - liens vers script

- UserControllerApi.php

3.2.2 User/login

- connection d'un utilisateur
- Méthode POST

3.2.2.1 Requête

Paramètre du path : aucun

Content-type : Application/json

Body

Propriété	Type	Obligatoire	Description
email	string	oui	email de l'utilisateur, doit être unique
password	string	oui	password de l'utilisateur, doit être minimum de taille 8

exemple JSON :

```
{
        "email": "harold@mail.com",
        "password": "harold9898",
}
```

3.2.2.2 Réponse: Réussite

Réussite Retourne code 200

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
result	boolean	oui	message confirmant la réussite du login

Exemple JSON :

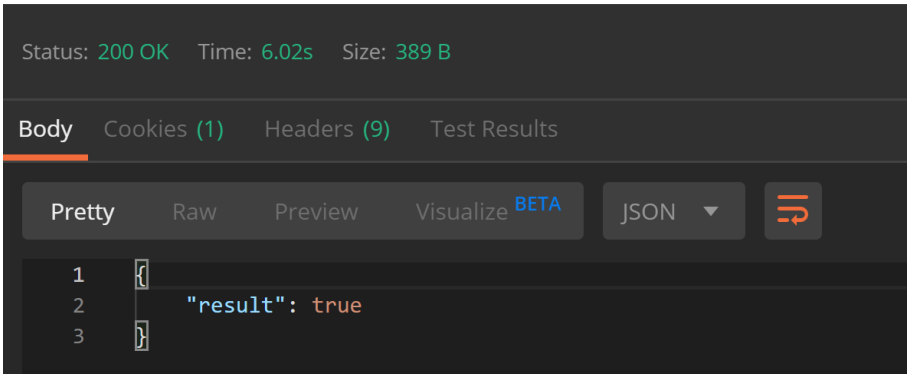

FIGURE 3.4

3.2.2.3 Réponse: erreurs

Erreur Retourne code 400

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
errors	string	oui	erreur rencontré pendant l'enregistrement

Erreur en cas d'un utilisateur invalide
Exemple JSON :

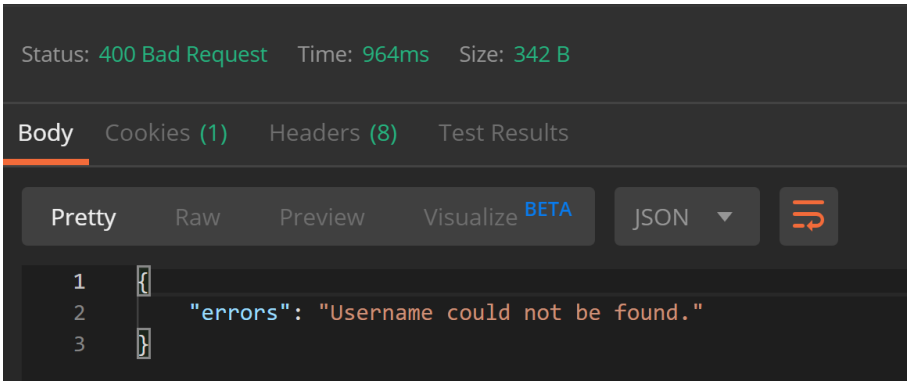

FIGURE 3.5

Erreur en cas d'un password incorrect

Exemple JSON :

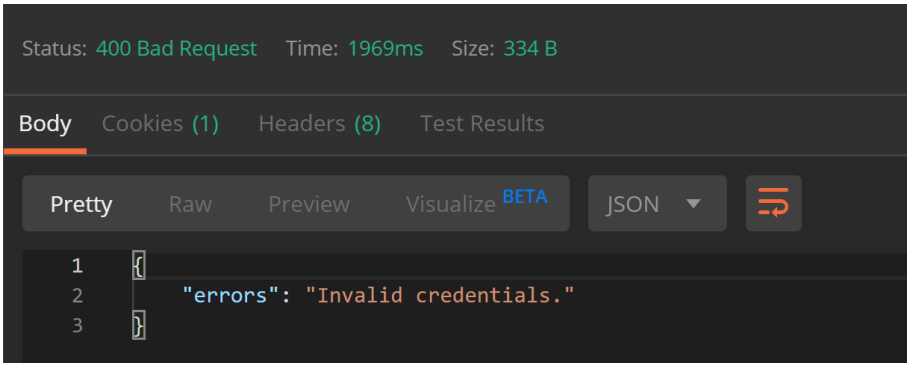

FIGURE 3.6

3.2.2.4 Extrait de code - liens vers script

- SecurityControllerApi.php
- security.yaml
- LoginAuthenticator.php (Guard)

3.2.3 fridge/index

- récupérer la liste des fridges d'un utilisateurs
- Méthode GET

3.2.3.1 Requête

Paramètre du path : aucun

Content-type : Application/json

Body : aucun

Attributes

Propriété	Type	Obligatoire	Description
email	string	oui	email de l'utilisateur pour lequel on veut récupérer les fridges

3.2.3.2 Réponse: Réussite

Réussite aucun code

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
id	int	oui	identifiant du fridge
name	string	oui	nom du frigo
type	string	oui	type de frigo
nbrFloors	int	oui	nombres d'étages dans le frigo
user	Objet	oui	object du user auquel le frigo appartient
id	int	oui	identifiant de l'utilisateur
imageFridgePath	string	oui	Lien vers l'image du fridge qui varie en fonction du type de frigo

Exemple JSON :


```
Status: 200 OK   Time: 1814ms   Size: 559 B
Body   Cookies   Headers (9)   Test Results
Pretty   Raw   Preview   Visualize BETA   JSON
1  [
2      {
3          "id": 40,
4          "name": "kitchen fridge",
5          "type": "french door fridge",
6          "nbrFloors": 5,
7          "user": {
8              "id": 28
9          },
10         "imageFridgePath": "http://127.0.0.1:8000/images/fridgeImage/
               french_door_fridge.jpg"
11     }
12 ]
```


FIGURE 3.7

3.2.3.3 Réponse: erreurs

Erreur Retourne code 400

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
errors	string	oui	message prévenant utilisateur qu'une erreur est survenu lors de la récupération des fridge

Exemple JSON :


```
Status: 400 Bad Request   Time: 1258ms   Size: 420 B
Body   Cookies   Headers (9)   Test Results
Pretty   Raw   Preview   Visualize BETA   JSON
1  {
2      "errors": "unable to fetch fridges"
3  }
```


FIGURE 3.8

3.2.3.4 Extrait de code - liens vers script

- FridgeController.php

3.2.4 fridge/add

- Ajout d'un nouveau fridge
- Méthode POST

3.2.4.1 Requête

Paramètre du path : aucun

Content-type : Application/json

Body

Propriété	Type	Obligatoire	Description
name	string	oui	nom du frigo
type	string	oui	type de frigo
nbrFloors	int	oui	nombres d'étages dans le frigo
imageFridgePath	string	oui	Lien vers l'image du fridge qui varie en fonction du type de frigo
userMail	string	oui	email du l'utilisateur

exemple JSON :

```
{
        "name": "kitchen",
        "type": "french door fridge",
        "nbrFloors": 2,
        "imageFridgePath": "http://127.0.0.1:8000/images/fridgeImage/french_door_fridge.jpg",
        "userMail": "test@mail.com"
}
```

3.2.4.2 Réponse: Réussite

Réussite Retourne code 200

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
message	string	non	message confirmant la réussite de l'enregistrement

Exemple JSON :

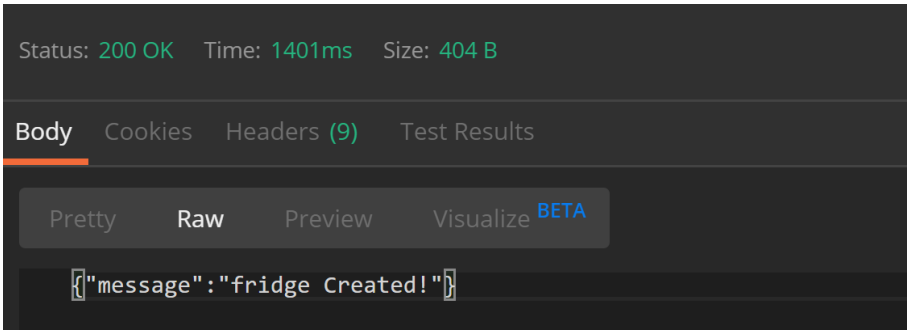

FIGURE 3.9

3.2.4.3 Réponse: erreurs

Erreur Retourne code 400

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
errors	string	oui	message prévenant utilisateur qu'une erreur est survenu lors de la creation du fridge

Exemple JSON :

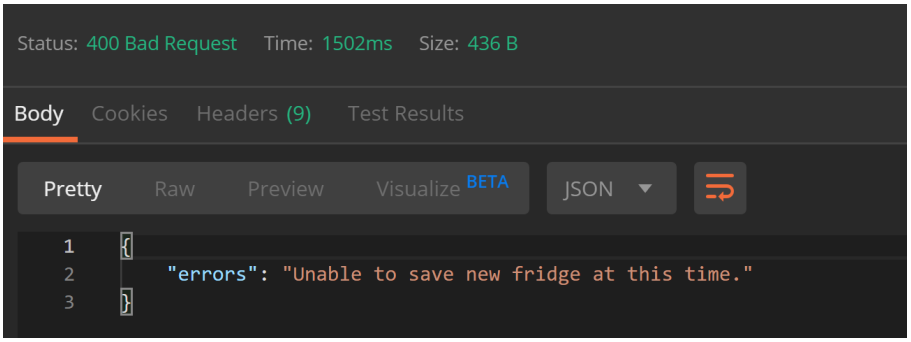

FIGURE 3.10

3.2.4.4 Extrait de code - liens vers script

- FridgeController.php

3.2.5 fridge/edit

- Modification d'un fridge
- Méthode PUT

3.2.5.1 Requête

Paramètre du path :

Propriété	Type	Obligatoire	Description
id	int	oui	identifiant du fridge qu'on modifie

Content-type : Application/json

Body

Propriété	Type	Obligatoire	Description
name	string	oui	nom du frigo
type	string	oui	type de frigo
nbrFloors	int	oui	nombres d'étages dans le frigo
imageFridgePath	string	oui	Lien vers l'image du fridge qui varie en fonction du type de frigo

exemple JSON :

```
{
        "name": "kitchen",
        "type": "french door fridge",
        "nbrFloors": 2,
        "imageFridgePath": "http://127.0.0.1:8000/images/fridgeImage/french_door_fridge.jpg"
}
```

3.2.5.2 Réponse: Réussite

Réussite Retourne code 200

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
message	string	non	message confirmant la réussite de l'enregistrement

Exemple JSON :

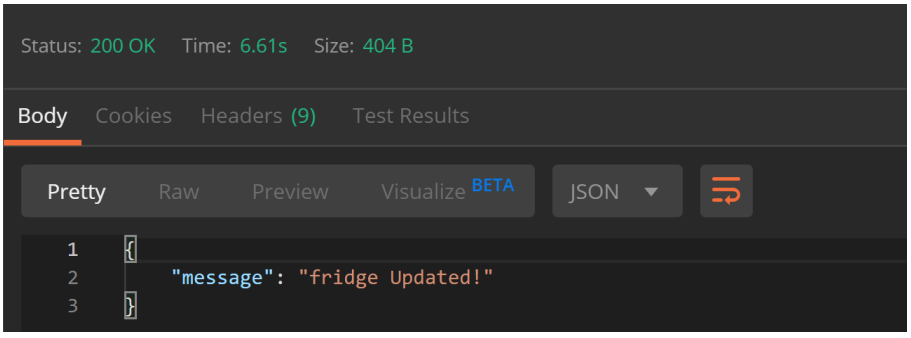

FIGURE 3.11

3.2.5.3 Réponse: erreurs

Erreur Retourne code 400

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
errors	string	oui	message prévenant utilisateur qu'une erreur est survenu lors de l'edition du fridge

Exemple JSON :

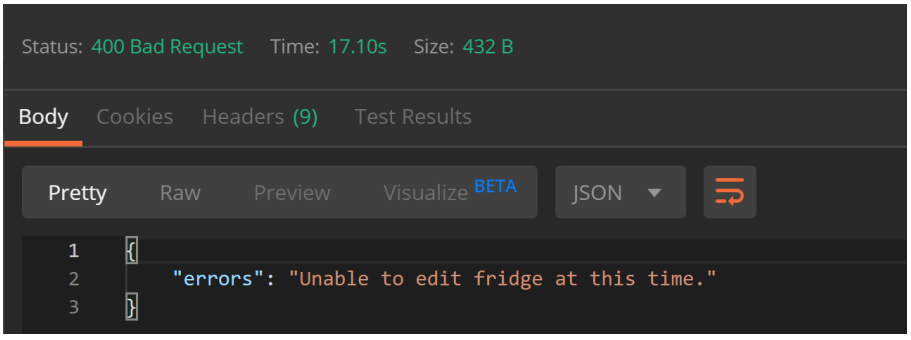

FIGURE 3.12

3.2.5.4 Extrait de code - liens vers script

- FridgeController.php

3.2.6 fridge/delete

- Suppression d'un fridge
- Méthode DELETE

3.2.6.1 Requête

Paramètre du path :

Propriété	Type	Obligatoire	Description
id	int	oui	identifiant du fridge qu'on modifie

Content-type : Application/json

Body : aucun

3.2.6.2 Réponse: Réussite

Réussite Retourne code 200

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
message	string	non	message confirmant la réussite de l'enregistrement

Exemple JSON :

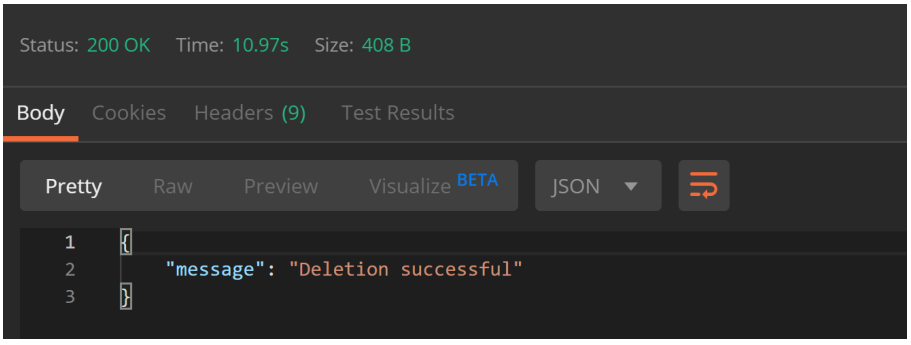

FIGURE 3.13

3.2.6.3 Réponse: erreurs

Erreur Retourne code 400

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
errors	string	oui	message prévenant utilisateur qu'une erreur est survenu lors de la suppression du fridge

Exemple JSON :

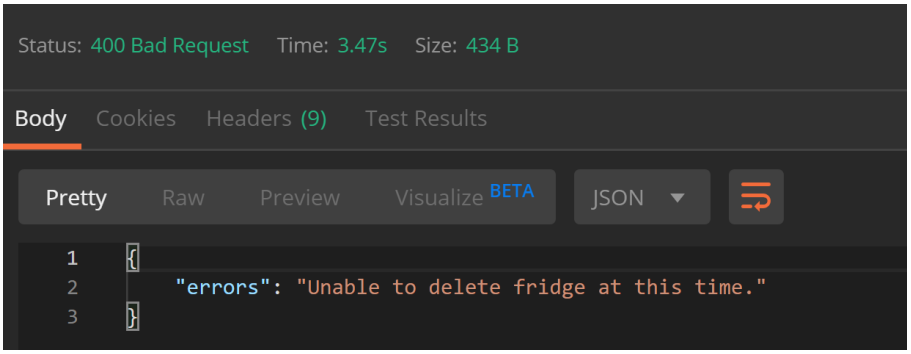

FIGURE 3.14

3.2.6.4 Extrait de code - liens vers script

- FridgeController.php

3.2.7 floor/index

- récupérer la liste des floors d'un fridge d'un utilisateur
- Méthode GET

3.2.7.1 Requête

Paramètre du path : aucun

Content-type : Application/json

Body : aucun

Attributes

Propriété	Type	Obligatoire	Description
idFridge	int	oui	identification du fridge pour lequel on veut récupérer les floors

3.2.7.2 Réponse: Réussite

Réussite aucun code

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
id	int	oui	identifiant du floor
name	string	oui	nom du floor
type	array	oui	liste des différents types de food à l'intérieur du floor
qtyFood	int	oui	nombres d'aliments dans le floor
idFridge	Objet	oui	object du fridge auquel le floor appartient
id	int	oui	identifiant du frigo
name	string	non	nom du frigo
type	string	non	type de frigo

Exemple JSON :

Status: 200 OK Time: 1346ms Size: 1.01 KB

Body Cookies Headers (9) Test Results

Pretty Raw Preview Visualize BETA JSON


```
[
    {
        "id": 25261,
        "type": [],
        "qtyFood": "0",
        "idFridge": {
            "id": 40,
            "name": "kitchen fridge",
            "type": "french door fridge"
        },
        "name": "bottom floor"
    },
    {
        "id": 25262,
        "type": [],
        "qtyFood": "0",
        "idFridge": {
            "id": 40,
            "name": "kitchen fridge",
            "type": "french door fridge"
        },
        "name": "bottom floor"
    },
```


FIGURE 3.15

3.2.7.3 Réponse: erreurs

Erreur Retourne code 400

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
errors	string	oui	message prévenant utilisateur qu'une erreur est survenu lors de la récupération des étage du fridge

Exemple JSON :

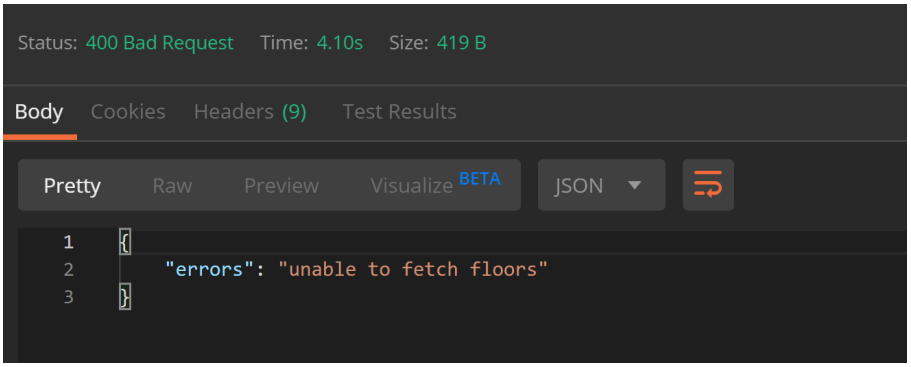

FIGURE 3.16

3.2.7.4 Extrait de code - liens vers script

- FloorController.php

3.2.8 floor/add

- Ajout d'un nouveau floor
- Méthode POST

3.2.8.1 Requête

Paramètre du path : aucun

Content-type : Application/json

Body

Propriété	Type	Obligatoire	Description
name	string	oui	nom du frigo
type	string	oui	type de frigo
id_fridge	int	oui	identifiant du frigo auquel on veut ajouter un étage

exemple JSON :

```
{
        "name": "bottom floor",
        "type": ["Vegetable"],
        "id_fridge": 40
}
```

3.2.8.2 Réponse: Réussite

Réussite Retourne code 200

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
message	string	non	message confirmant la réussite de la creation du floor

Exemple JSON :

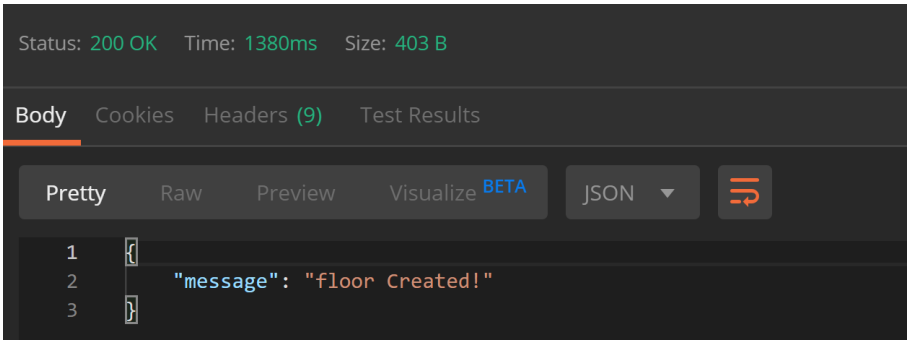

FIGURE 3.17

3.2.8.3 Réponse: erreurs

Erreur Retourne code 400

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
errors	string	oui	message prévenant utilisateur qu'une erreur est survenu lors de la creation du floor

Exemple JSON :

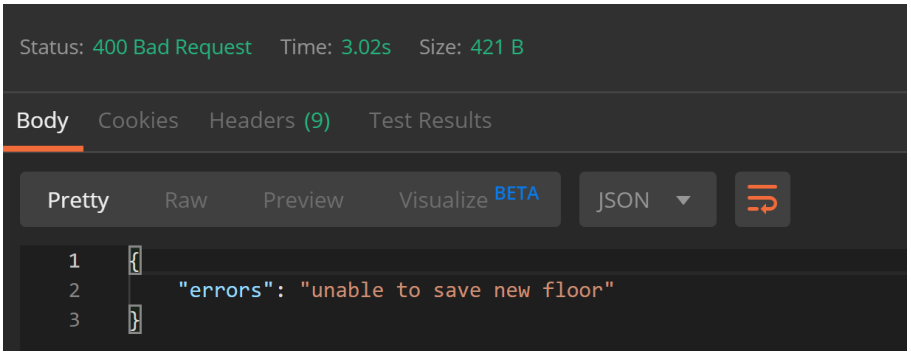

FIGURE 3.18

3.2.8.4 Extrait de code - liens vers script

- FloorController.php

3.2.9 floor/edit

- Modification d'un floor
- Méthode PUT

3.2.9.1 Requête

Paramètre du path :

Propriété	Type	Obligatoire	Description
id	int	oui	identifiant du fridge qu'on modifie

Content-type : Application/json

Body

Propriété	Type	Obligatoire	Description
name	string	oui	nom du floor
type	string	oui	type de floor

exemple JSON :

```
{
        "name": "bottom floor",
        "type": ["Vegetable"]
}
```

3.2.9.2 Réponse: Réussite

Réussite Retourne code 200

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
message	string	non	message confirmant la réussite de l'édition du floor

Exemple JSON :

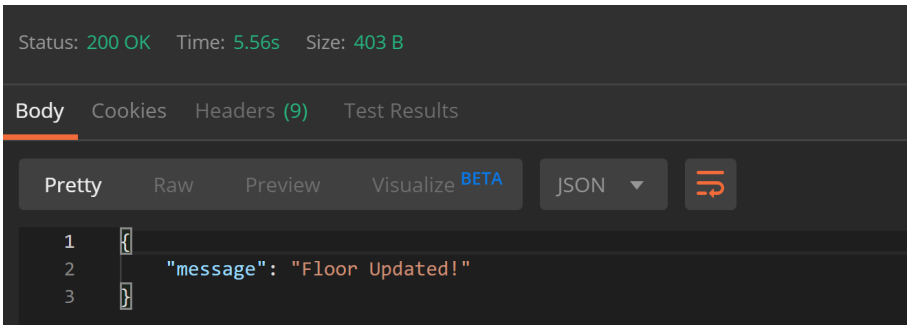

FIGURE 3.19

3.2.9.3 Réponse: erreurs

Erreur Retourne code 400

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
errors	string	oui	message prévenant utilisateur qu'une erreur est survenu lors de l'édition du floor

Exemple JSON :

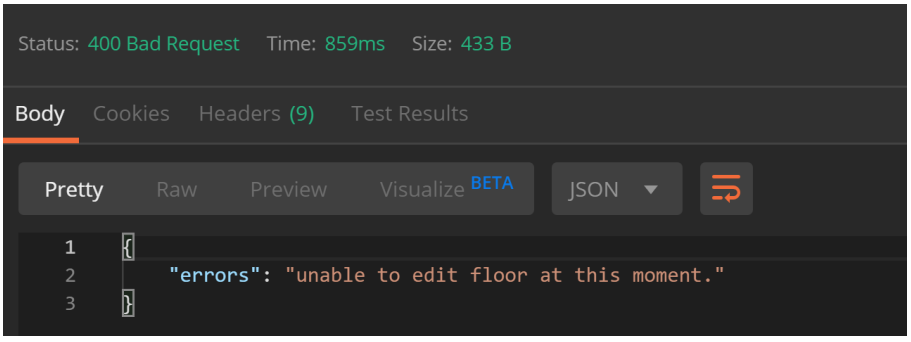

FIGURE 3.20

3.2.9.4 Extrait de code - liens vers script

- FloorController.php

3.2.10 floor/delete

- Suppression d'un floor
- Méthode DELETE

3.2.10.1 Requête

Paramètre du path :

Propriété	Type	Obligatoire	Description
id	int	oui	identifiant du floor qu'on modifie

Content-type : Application/json

Body : aucun

3.2.10.2 Réponse: Réussite

Réussite Retourne code 200

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
message	string	non	message confirmant la suppression du floor

Exemple JSON :

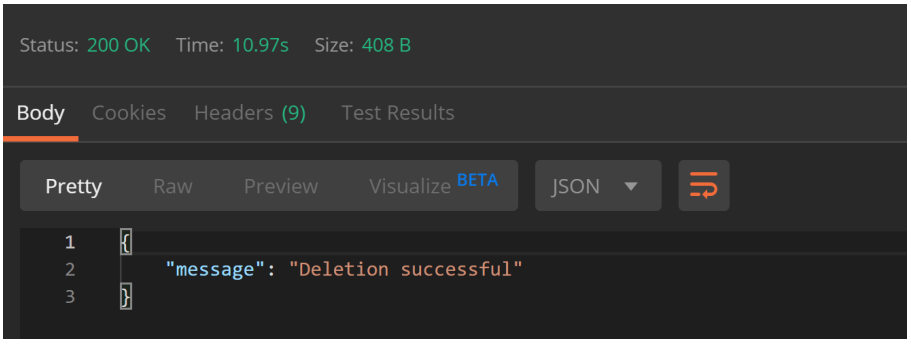

FIGURE 3.21

3.2.10.3 Réponse: erreurs

Erreur Retourne code 400

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
errors	string	oui	message prévenant utilisateur qu'une erreur est survenu lors de la suppression du floor

Exemple JSON :

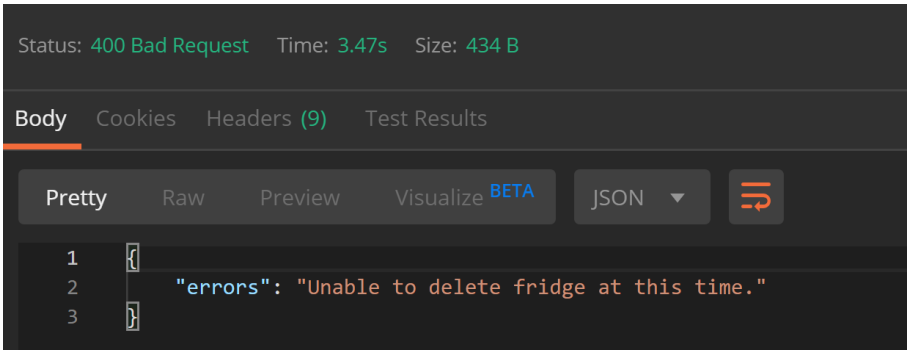

FIGURE 3.22

3.2.10.4 Extrait de code - liens vers script

- FloorController.php

3.2.11 food/index

- récupérer la liste des food d'un floor d'un fridge d'un utilisateur
- Méthode GET

3.2.11.1 Requête

Paramètre du path : aucun

Content-type : Application/json

Body : aucun

Attributes

Propriété	Type	Obligatoire	Description
idFloor	string	oui	identifiant du floor pour lequel on veut récupérer les aliments se trouvant dedans

3.2.11.2 Réponse: Réussite

Réussite aucun code

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
id	int	oui	identifiant du food
name	string	oui	nom du food
type	string	oui	type de food
expirationDate	string	oui	date d'expiration du food
quantity	int	oui	quantity de l'aliment dans le floor
idFloor	objet	oui	objet du floor auquel le food appartient
id	int	oui	identifiant du floor
type	array	non	liste des types de food se trouvant dans le floor
name	string	non	nom du floor auquel appartiennent les aliments
dateOfPurchase	string	oui	date d'achat de l'aliment
imageFoodPath	string	oui	lien vers l'image correspondant au type de food que le food est
unitQty	string	oui	unité pour la quanity de food

Exemple JSON :


```
Status: 200 OK   Time: 921ms   Size: 671 B   Save Respons
Body   Cookies   Headers (9)   Test Results
Pretty   Raw   Preview   Visualize BETA   JSON
1  [
2      {
3          "id": 69,
4          "name": "tomato",
5          "type": "Vegetable",
6          "expirationDate": "2019-12-24T23:00:00.000Z",
7          "quantity": 2,
8          "idFloor": {
9              "id": 25276,
10             "type": [],
11             "name": "bottom floor"
12         },
13         "dateOfPurchase": "2019-12-24T23:00:00.000Z",
14         "imageFoodPath": "http://127.0.0.1:8000/images/foodDefault/vegetables.png",
15         "unitQty": "L"
16     }
17 ]
```


FIGURE 3.23

3.2.11.3 Réponse: erreurs

Erreur Retourne code 400

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
errors	string	oui	message prévenant utilisateur qu'une erreur est survenu lors de la récupération des aliments du floor

Exemple JSON :

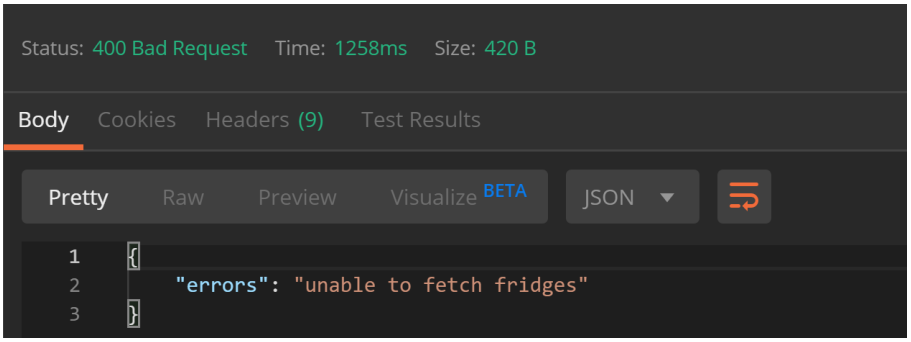

FIGURE 3.24

3.2.11.4 Extrait de code - liens vers script

- FoodController.php

3.2.12 food/add

- Ajout d'un nouveau food
- Méthode POST

3.2.12.1 Requête

Paramètre du path : aucun

Content-type : Application/json

Body

Propriété	Type	Obligatoire	Description
name	string	oui	nom du food
type	string	oui	type de food
expiration$_date$	string	oui	date d'expiration du food
quantity	int	oui	quantity de l'aliment dans le floor
id_floor	int	oui	identifiant du floor
date_of_purchase	string	oui	date d'achat de l'aliment
image_food_path	string	oui	lien vers l'image correspondant au type de food que le food est
unit_qty	string	oui	unité pour la quanity de food

exemple JSON :

```
{
        "name": "tomato",
        "type": "Vegetable",
        "expiration_date": "2019-12-24T23:00:00.000Z",
        "quantity": 2,
        "date_of_purchase": "2019-12-24T23:00:00.000Z",
        "image_food_path": "http://127.0.0.1:8000/images/foodDefault/vegetables.png",
        "unit_qty": "L",
        "id_floor": 25276
}
```

3.2.12.2 Réponse: Réussite

Réussite Retourne code 200

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
message	string	non	message confirmant la réussite de l'enregistrement

Exemple JSON :

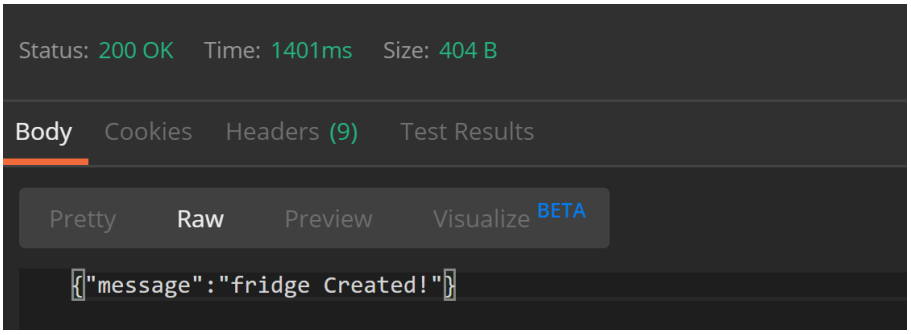

FIGURE 3.25

3.2.12.3 Réponse: erreurs

Erreur Retourne code 400

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
errors	string	oui	message prévenant utilisateur qu'une erreur est survenu lors de la creation du food

Exemple JSON :

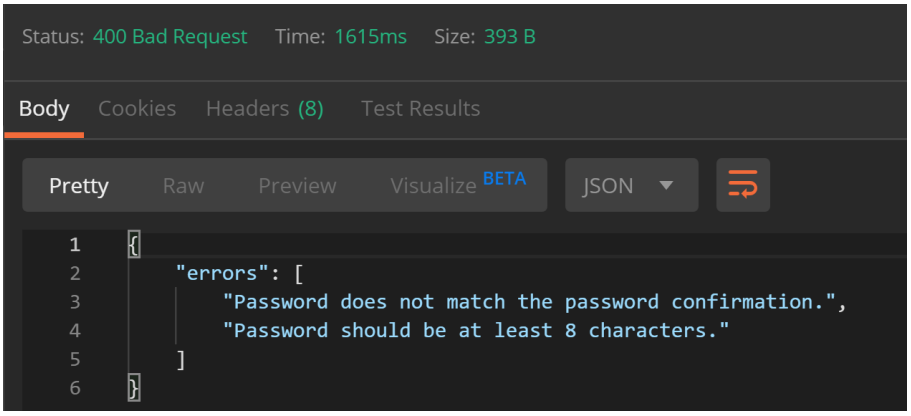

FIGURE 3.26

3.2.12.4 Extrait de code - liens vers script

- FoodController.php

3.2.13 food/edit

- Modification d'un food
- Méthode PUT

3.2.13.1 Requête

Paramètre du path :

Propriété	Type	Obligatoire	Description
id	int	oui	identifiant du fridge qu'on modifie

Content-type : Application/json

Body

Propriété	Type	Obligatoire	Description
id	int	oui	identifiant du food
name	string	oui	nom du food
type	string	oui	type de food
expiration_date	string	oui	date d'expiration du food
quantity	int	oui	quantity de l'aliment dans le floor
date_of_purchase	string	oui	date d'achat de l'aliment
image_food_path	string	oui	lien vers l'image correspondant au type de food que le food est
unit_qty	string	oui	unité pour la quanity de food

exemple JSON :

```
1 {
2         "name": "kitchen",
3         "type": "french door fridge",
4         "nbrFloors": 2,
5         "imageFridgePath": "http://127.0.0.1:8000/images/fridgeImage/french_door_fridge.jpg"
6 }
```

3.2.13.2 Réponse: Réussite

Réussite Retourne code 200

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
message	string	non	message confirmant la réussite de l'enregistrement

Exemple JSON :

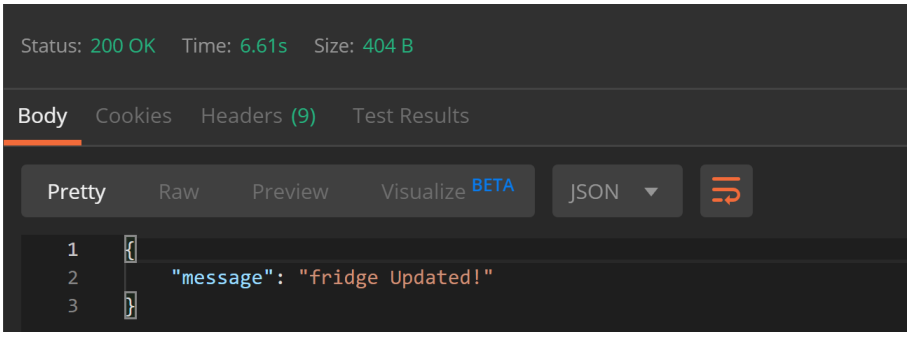

FIGURE 3.27

3.2.13.3 Réponse: erreurs

Erreur Retourne code 400

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
errors	string	oui	message prévenant utilisateur qu'une erreur est survenu lors de l'édition du food

Exemple JSON :

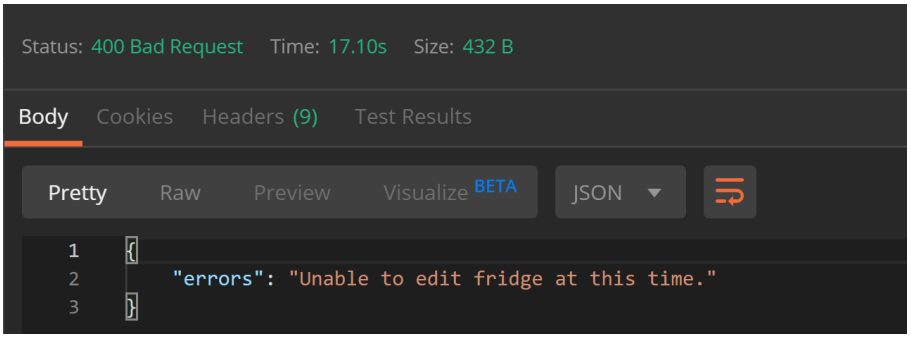

FIGURE 3.28

3.2.13.4 Extrait de code - liens vers script

- FoodController.php

3.2.14 food/delete

- Suppression d'un food
- Méthode DELETE

3.2.14.1 Requête

Paramètre du path :

Propriété	Type	Obligatoire	Description
id	int	oui	identifiant du fridge qu'on modifie

Content-type : Application/json

Body : aucun

3.2.14.2 Réponse: Réussite

Réussite Retourne code 200

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
message	string	non	message confirmant la réussite de l'enregistrement

Exemple JSON :

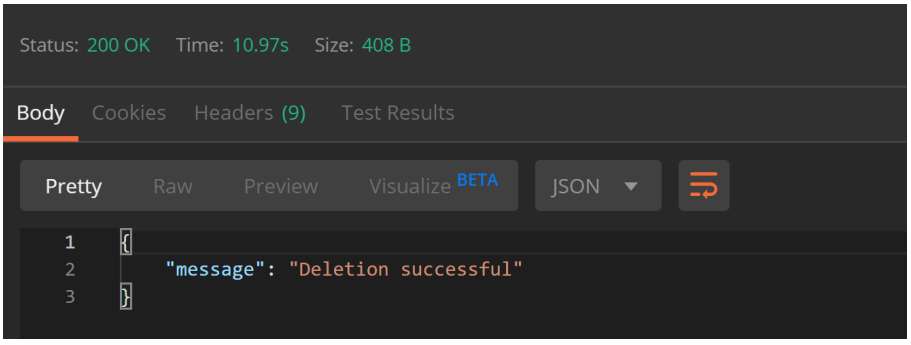

FIGURE 3.29

3.2.14.3 Réponse: erreurs

Erreur Retourne code 400

Content-type Application/json

Body

Propriété	Type	Obligatoire	Description
errors	string	oui	message prévenant utilisateur qu'une erreur est survenu lors de la suppression du food

Exemple JSON :

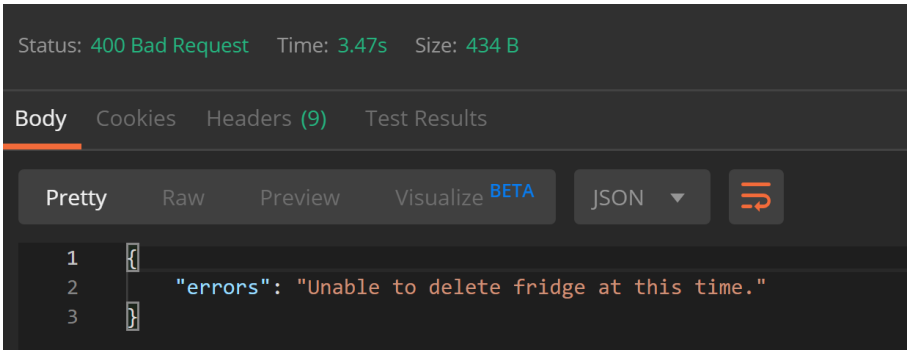

FIGURE 3.30

3.2.14.4 Extrait de code - liens vers script

- FoodController.php

Symfony - Angular

Comme dit plutot, cette partie porte sur le frontend de la partie de projet Symfony-API-Angular. Pour ce qu'il est de la partie symfony, elle est quasi la même que la partie symfony de la première partie du projet au niveau des missions des différentes requêtes vers la base de donnée. La séparation du backend et frontend se fait via un API REST décrit en détail au Chapitre 4.

Cette partie recouvre les différentes User stories définies au chapitre 2 et donc l'interface de l'application web qui elle est réaliser utilisant le framework Angular.

4.1 En tant qu'utilisateur, je veux pouvoir m'enregistrer

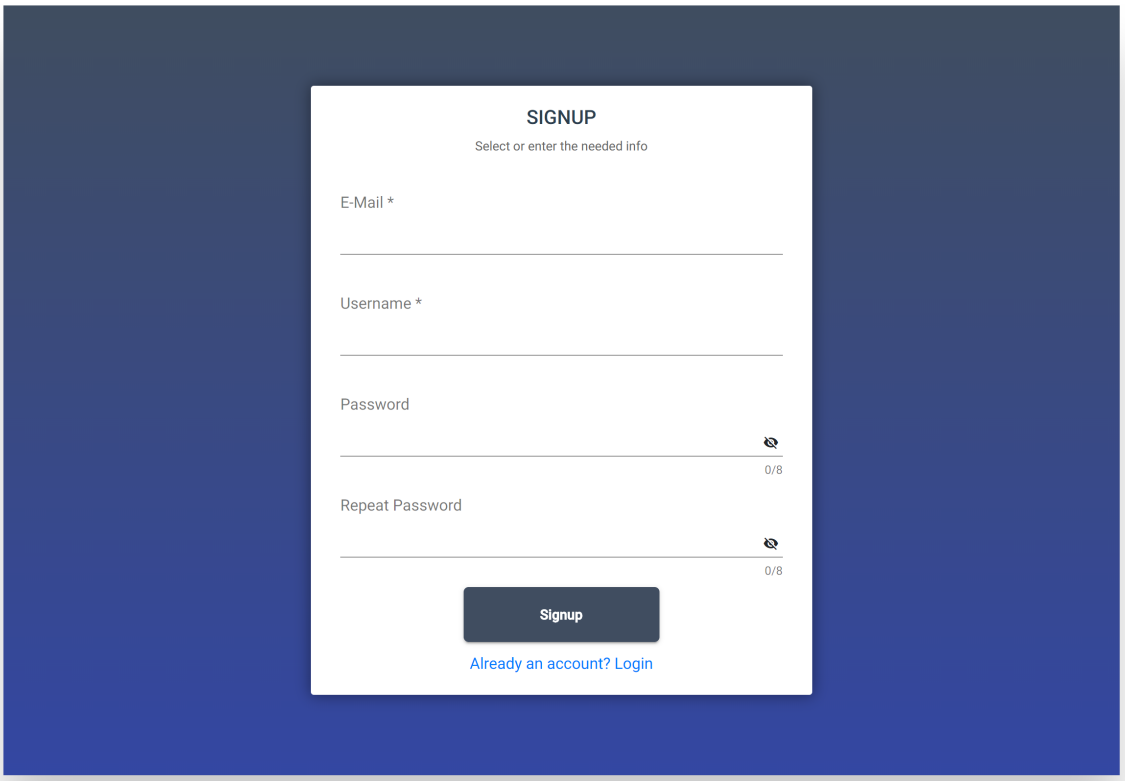

FIGURE 4.1

Ceci est donc la page rendu quand l'utilisateur veut s'enregistrer sur le site dans la partie Angular. La page d'accueil est la même et le processus pour arriver à cette page est la même que dans la partie symfony-twig (SyT), c'est à dire par un dropdown ou les liens vers login/signup en dessous des boutons.

En ce qui concerne l'utilisation, c'est les mêmes étapes que dans SyT. La différence est dans la gestion des erreurs et dans les fonctionnalités cachées. En effet pour la partie mot de passe, deux petites fonctionnalité ont été rajouté :

- show/hide password qui fonctionne tout simplement en cliquant sur le petit oeil
- une petite aide pour savoir combien de caractères sont nécessaire et à combien on est.

Une autre différence avec SyT est qu'après s'être enregistrer, l'utilisateur sera automatiquement connecté et ne devra pas passer par la phase de login.

La sécurité des pages est géré dans le "AppRoutingModule" grâce au guard "auth.guard.ts" qui vérifie si l'utilisateur est bien connecté.

4.1.1 Gestion des erreurs et sécurité

Il y a deux types d'erreurs

- erreurs en temps réel (Figure 4.2)
- erreurs après validation

Pour les erreurs en temps réel, on retrouve pour tous les champs une erreur si celui-ci est vide. Les erreurs en temps dans cette partie sont géré en plus par le module *"customValidator.module.ts"* qui s'occupe de vérifier si les mots de passe sont égaux. Il s'occupe aussi d'afficher des erreurs personnalisées.

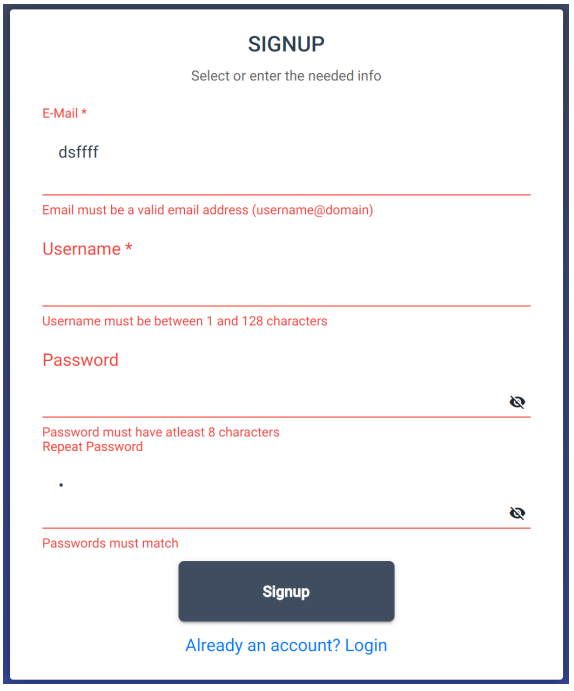

FIGURE 4.2: gestion des erreurs en temps réel

Pour les erreurs après validation, celle-ci s'afficheront en dessous du titre signup et sont des erreurs venant de l'api. Les données entrées par l'utilisateur sont enregistrées et donc récupérées si une erreur était survenu pour que l'utilisateur n'ai pas à tout refaire.

4.1.2 partie de codes significative

La partie significative se trouve dans le composant "SignupComponent" :

- la fonction ***"ngOnit()"*** s'occupe de la génération du form
- la fonction ***"ngAfterViewInit()"*** s'occupe d'afficher les erreurs venant de l'api
- la fonction ***"onSignup()"*** permet d'envoyer les données entrée par l'utilisateur vers l'api

Une autre partie significative se trouve dans "AuthService :

- la fonction ***"createUser()"*** s'occupe de créer l'utilisateur en envoyant les données et si tout va bien connecté l'utilisateur par après.

4.1.3 Lien vers script concerné

- signup.component.ts
- signup.component.html
- customValidator.module.ts
- auth.service.ts
- UserController.php
- User.php

4.1.4 Diagramme de séquence

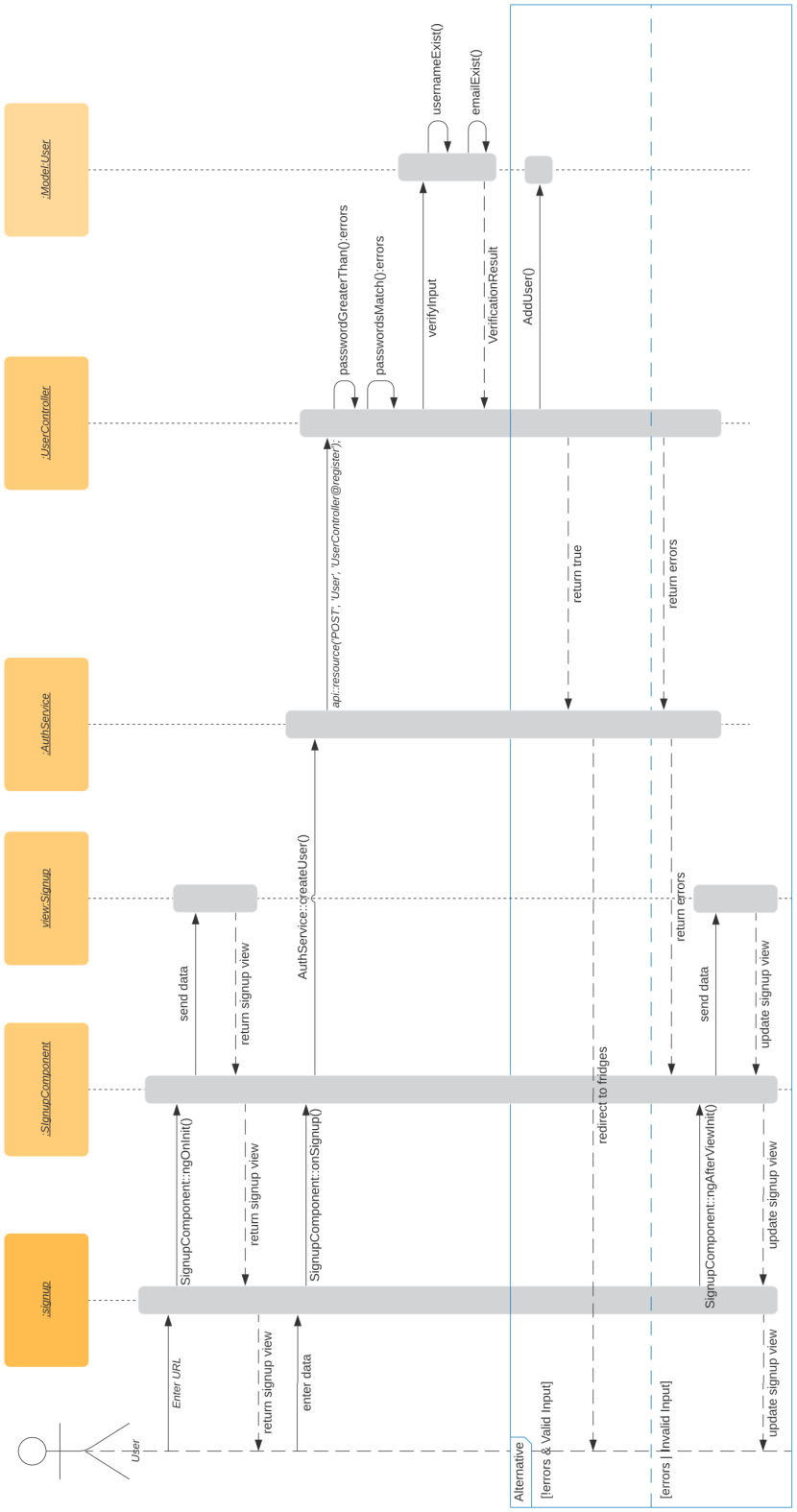

FIGURE 4.3: Diagramme de séquence pour l'enregistrement d'un utilisateur

4.2 En tant qu'utilisateur, je veux pouvoir me connecter

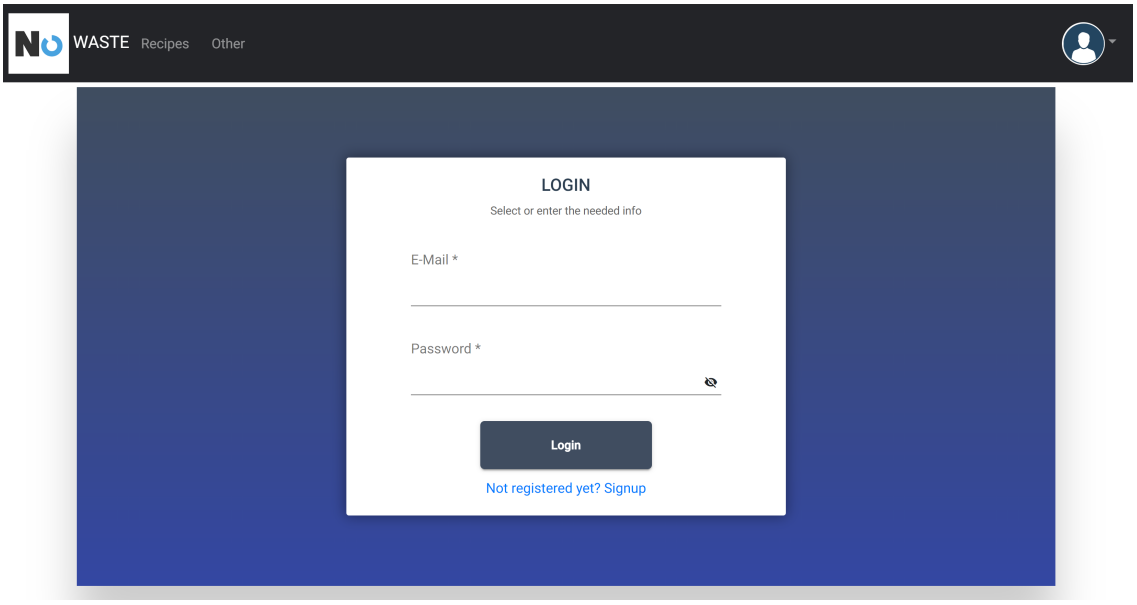

FIGURE 4.4

Même chose ici, seul différence dans les fonctionnalité est qu'on a un bouton "hide/show" pour le mot de passe.

Une chose importante à noter dans cette partie Angular, c'est que pour l'instant il n'y pas d'utilisation de cookies. Ce qui veut dire que quand la page est rechargé, l'utilisateur est automatiquement déconnecté vu que rien n'est enregistré.

4.2.1 Gestion des erreurs et sécurité

Comme pour le register nous avons des erreur en temps réel et des erreurs après submission.

Pour les erreurs venant de l'api, une nouvelle fonctionnalité a été implémenté pour se connecter. En effet un loginAuthenticator est utilisé ici. C'est lui qui va s'occuper de vérifier les "credentials" que l'utilisateur a entré.

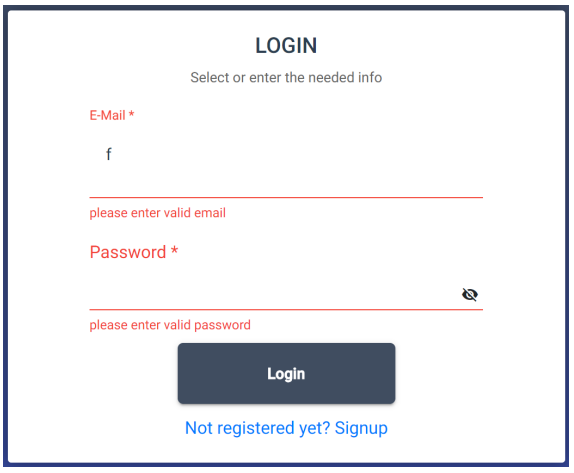

FIGURE 4.5

4.2.2 partie de codes significative

Dans SignupComponent :

- la fonction ***"ngOnit()"*** s'occupe de la génération du form
- la fonction ***"ngAfterViewInit()"*** s'occupe d'afficher les erreurs venant de l'api
- la fonction ***"onlogin()"*** permet d'envoyer les données entrée par l'utilisateur vers l'api en appelant le fonction *"loginUser()"*

Dans AuthService :

- la fonction ***"loginUser()"*** s'occupe de connecter l'utilisateur en envoyant les données sur l'api qui lui va vérifier grâce au *"LoginAuthenticator"* si les credentials sont juste.

4.2.3 Lien vers script concerné

- login.component.ts
- login.component.html
- auth.service.ts
- LoginAuthenticator.php
- SecurityController.php
- User.php

4.2.4 Diagramme de séquence

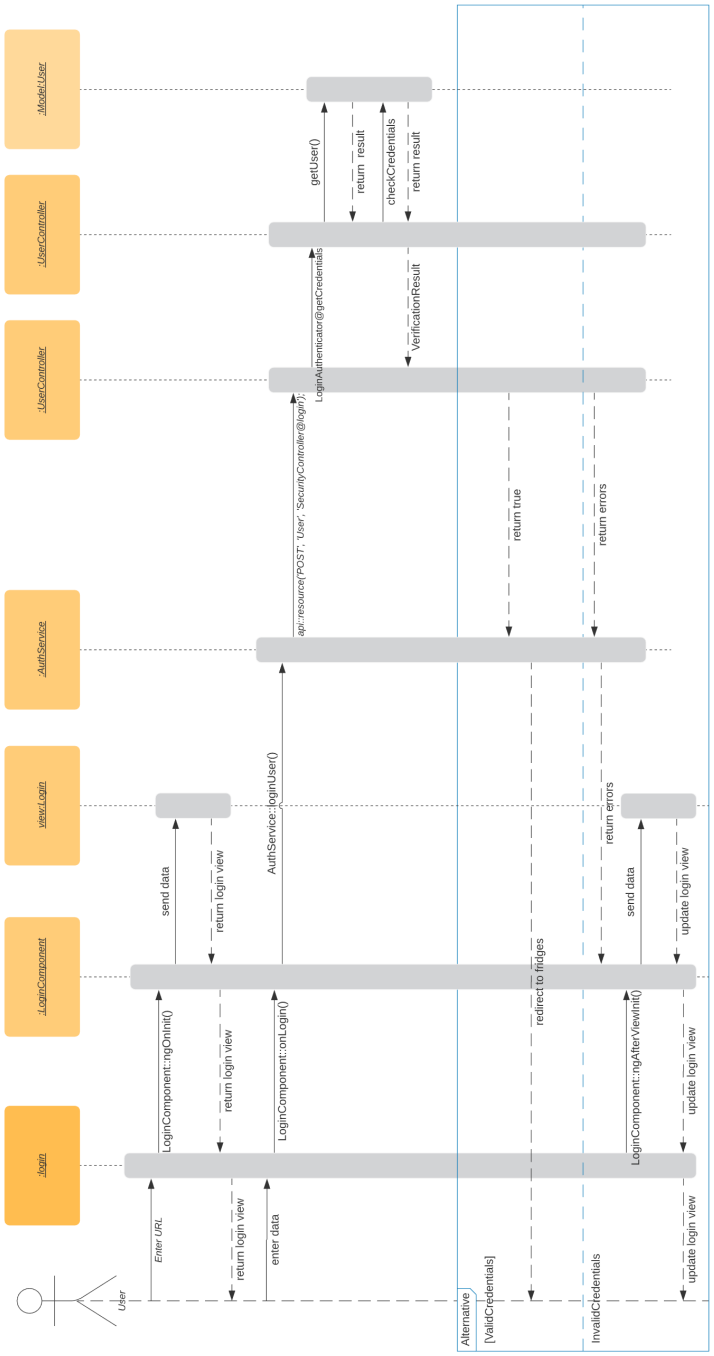

FIGURE 4.6: Diagramme de séquence pour connecter un utilisateur

4.3 En tant qu'utilisateur, je veux pouvoir retrouver les frigos que j'ai créé

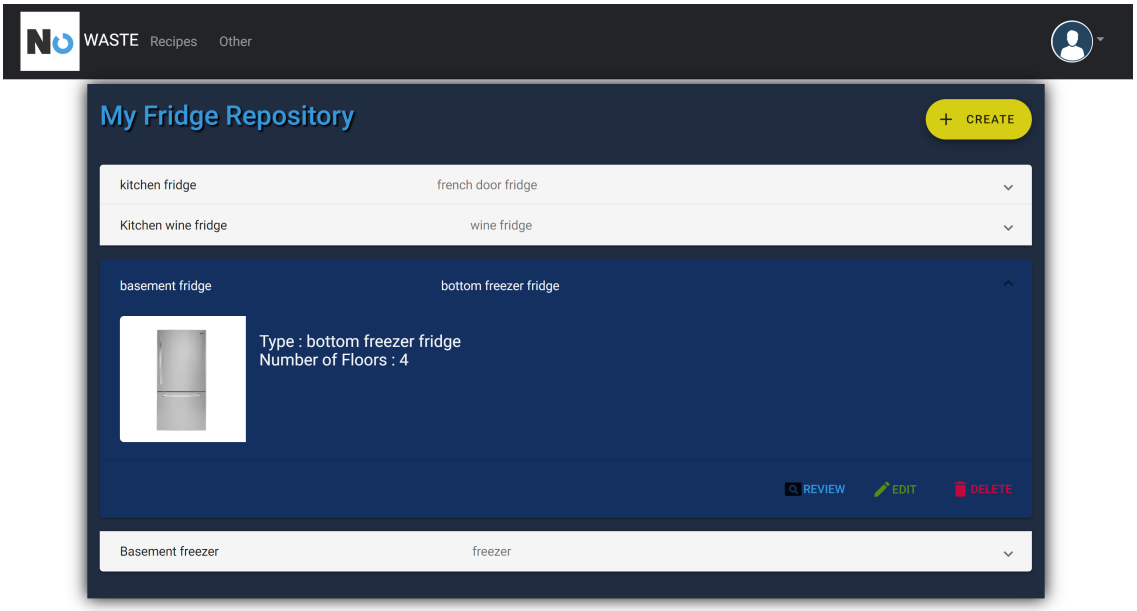

FIGURE 4.7

La seule vrai différence avec SyT est le design. Comme on peut le voir sur l'image, les frigos sont disposé dans des panneaux à expansion. Chaque panneau est muni d'un titre contenant le nom du frigo et son type. Si l'utilisateur veut avoir plus d'information sur le frigo en lui-même, il lui suffit de cliquer sur le panneau et le panneau s'ouvre.

À l'intérieur on peut retrouver une image du frigo, le type et le nombre d'étages que contient le frigo. Ensuite 3 boutons sont exposés en bas à droite de chaque panneau:

- **REVIEW** : permettant d'aller dans le frigo
- **EDIT** : permettant de modifier le frigo
- **DELETE** : permettant de supprimer un frigo

Ensuite en haut à droite, on peut voir un grand bouton jaune qui permet à l'utilisateurs de créer un nouveau frigo

4.3.1 Gestion d'erreur et sécurité

Si une erreur survenait pendant le charge des frigos de l'utilisateur, une erreur s'affichera pour prévenir l'utilisateur.

4.3.2 partie de codes significative

Dans SignupComponent :

- la fonction ***"ngOnit()"*** s'occupe d'aller chargé les données, donc les frigos de l'utilisateur. Une barre de chargement est toujours mise en place pendant le chargement des données (pas seulement ici mais à chaque fois qu'un chargement de données est fait pour avertir l'utilisateur que les données sont bien entrain de charger.

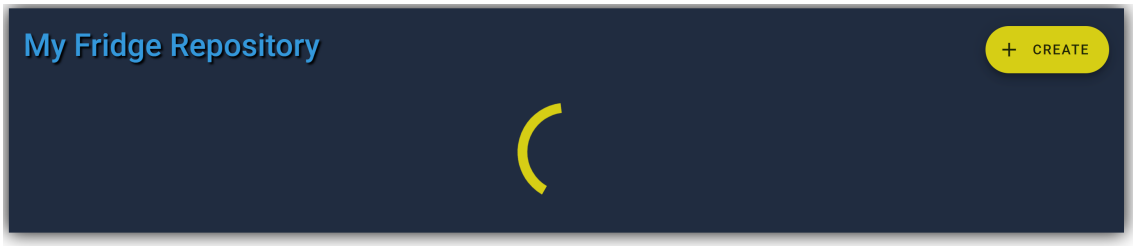

FIGURE 4.8

- la fonction ***"ngAfterViewInit()"*** s'occupe d'afficher les erreurs venant de l'api

Dans FridgeService :

- la fonction ***"getFridges()"*** s'occupe d'aller GET les fridges de l'utilisateur sur l'API.

4.3.3 Lien vers script concerné

- fridge.component.ts
- fridge-list.component.html
- fridge.service.ts
- FridgeController.php
- Fridge.php

4.4 En tant qu'utilisateur, je veux pouvoir créer un frigo ou plusieurs

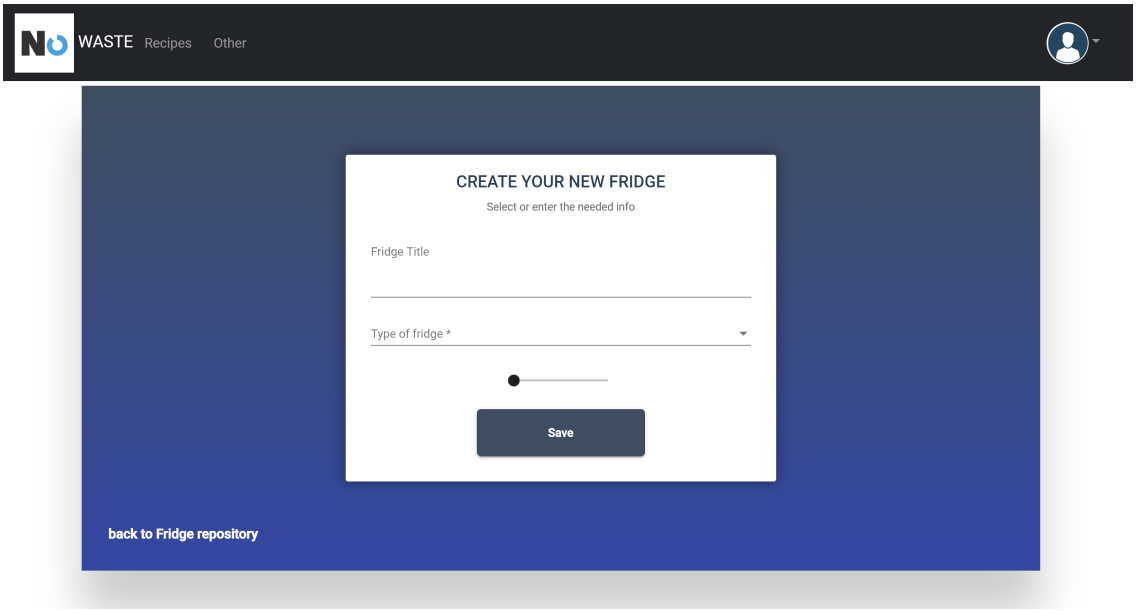

FIGURE 4.9

Pour se rendre sur cette page, l'utilisateur doit juste cliquer sur le bouton jaune nommé **"CREATE"** se trouvant en haut à droit dans la page "My fridge repository".

Le fonctionnement de cette page est identique à celle de SyT. Après confirmation, un snackbar apparaît pour confirmer ou non si l'action à fonctionner comme sur la figure suivante :

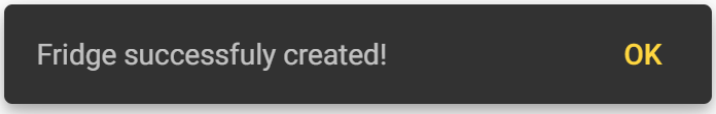

FIGURE 4.10

4.4.1 Gestion d'erreur et sécurité

Le form vérifie que les informations sont toutes encodées et correctes sinon une erreur s'affiche.

4.4.2 Partie de code significative

Dans FridgeCreateComponent :

- la fonction ***"ngOnit()"*** s'occupe de créer le form
- la fonction ***"ngAfterViewInit()"*** s'occupe d'afficher les erreurs venant de l'api
- la fonction **"onSaveFridge()"** envoie les données entrée par l'utilisateur vers l'api

Dans FridgeService :

- La fonction ***"addFridge()"*** s'occupe d'aller POST les données envoyer par le FridgeCreateComponent sur l'API

4.4.3 Lien vers script concerné

- fridge-create.component.ts
- fridge-create.component.html
- fridge.service.ts
- FridgeController.php
- Fridge.php

4.5 En tant qu'utilisateur, je veux pouvoir modifier mon frigo

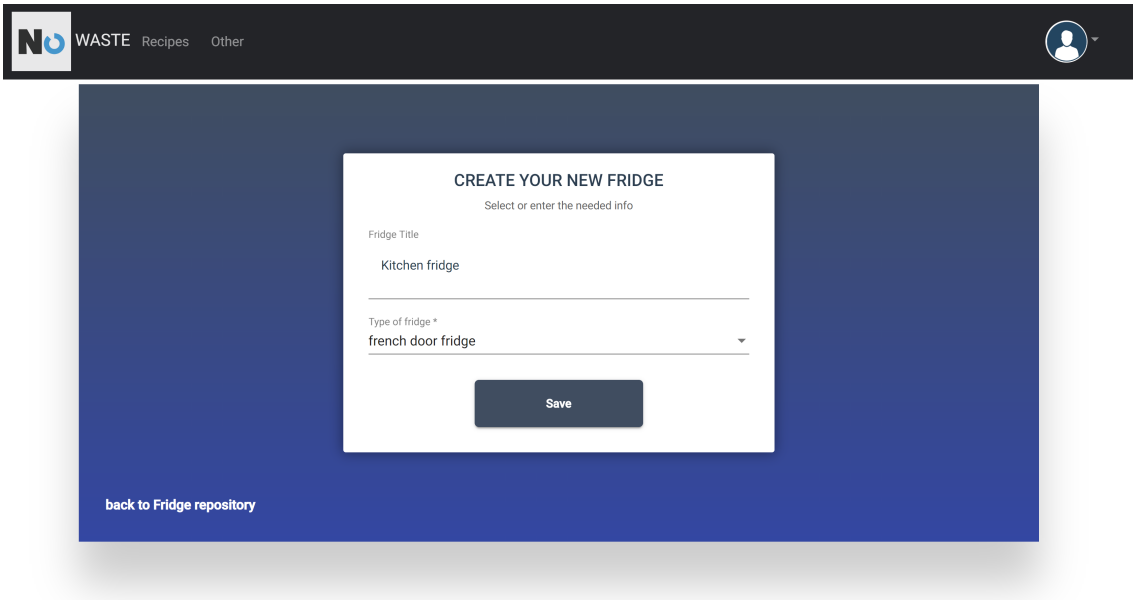

FIGURE 4.11

Pour se rendre sur cette page, l'utilisateur peut le faire à partir du "fridge repository" en cliquant sur "EDIT" dans un fridge (panneau à extension) ou cliquant sur le bouton (avec une info-bulle) comme sur le figure suivante :

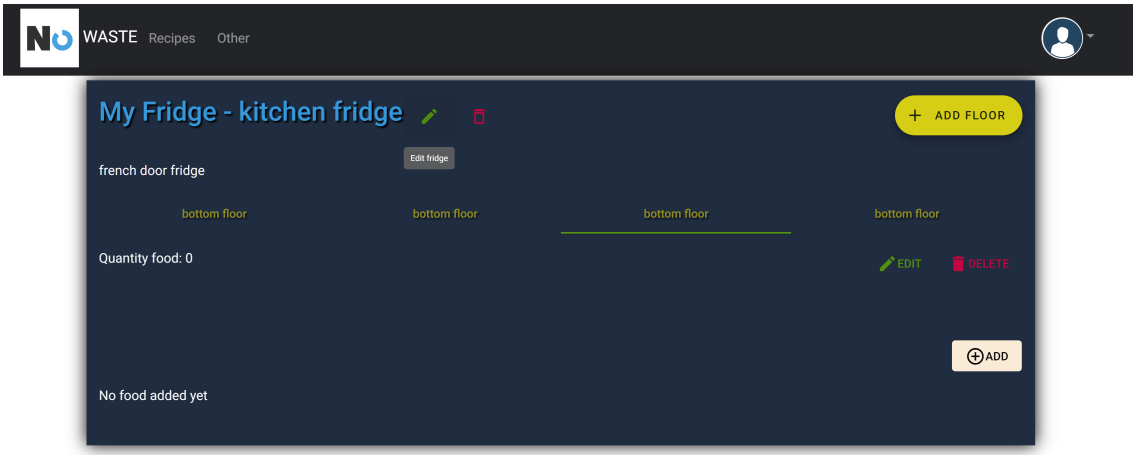

FIGURE 4.12

4.5.1 Gestion d'erreur et sécurité

Le form vérifie que les informations sont toutes encodée et correctes sinon une erreur s'affiche.

4.5.2 Partie de code significative

Dans FridgeCreateComponent :

- la fonction ***"ngOnit()"*** s'occupe de créer le form et d'y ajouter les caractéristiques du fridge à modifier.
- la fonction ***"ngAfterViewInit()"*** s'occupe d'afficher les erreurs venant de l'api
- la fonction **"onSaveFridge()"** envoie les données entrée par l'utilisateur vers l'api

Dans FridgeService :

- La fonction ***"updateFridge()"*** s'occupe d'aller PUT les données envoyer par le FridgeCreateComponent sur l'API

4.5.3 Lien vers script concerné

- fridge-create.component.ts
- fridge-create.component.html
- fridge.service.ts
- FridgeController.php
- Fridge.php

4.6 En tant qu'utilisateur, je veux pouvoir supprimer mon frigo

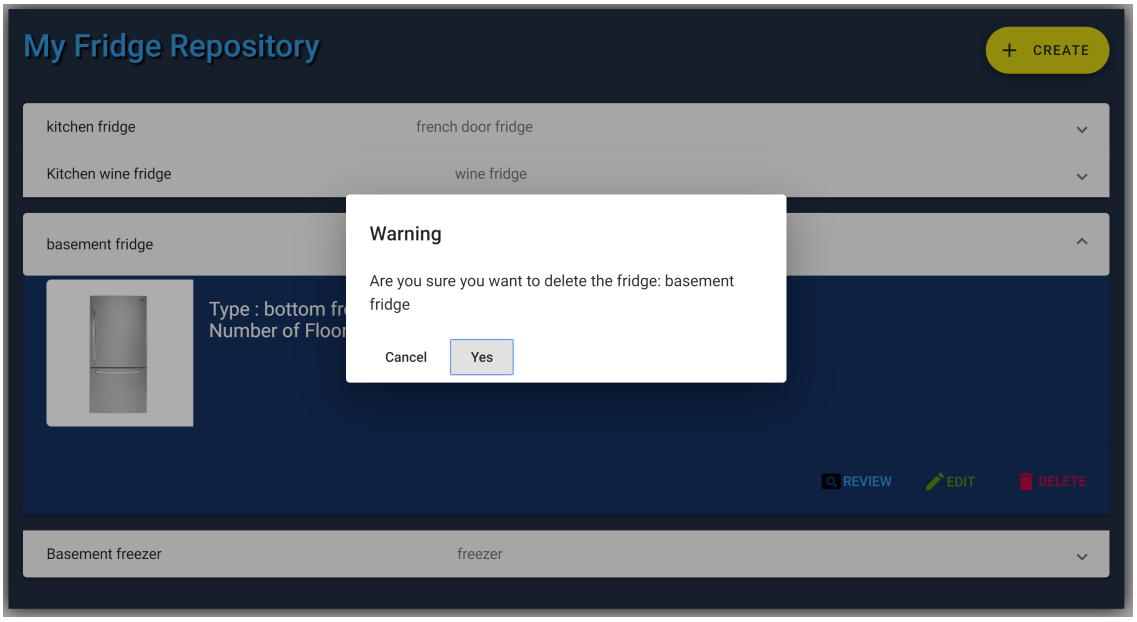

FIGURE 4.13

Pour supprimer un fridge, l'utilisateur à deux manière comme dans la partie "edit fridge" sauf dans ce cas le bouton "DELETE" ou comme sur l'image suivante :

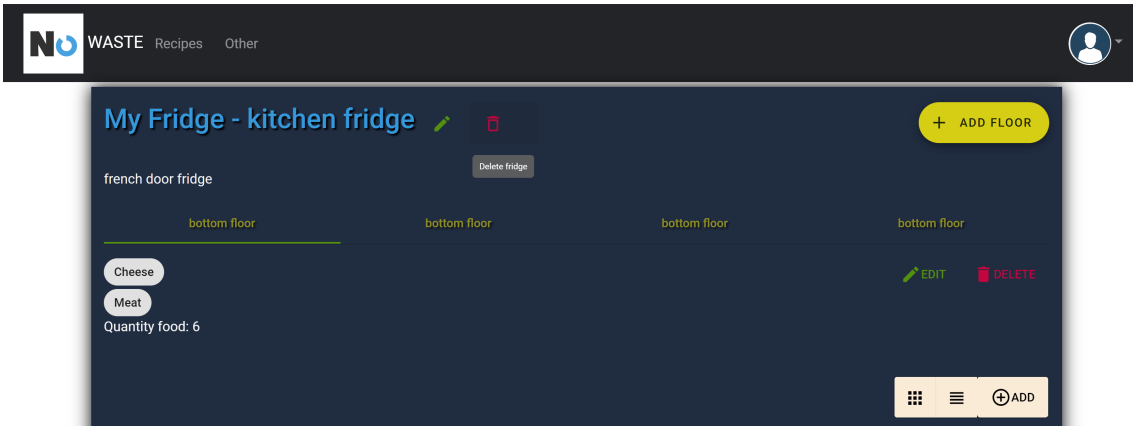

FIGURE 4.14

4.6.1 Partie de code significative

Dans FridgeListComponent :

- la fonction ***"openDialog()"*** s'occupe d'ouvrir un nouveau dialog pour la confirmation de la suppression

Dans DialogDeleteComponent :

- la fonction ***"ngAfterViewInit()"*** s'occupe d'afficher un snackbar négatif en cas d'erreur
- la fonction **"onYesClick()"** appelle la méthode "deleteFridge" du FridgeService et se souscrit à tout type d'erreur venant de l'api

Dans FridgeService :

- La fonction ***"deleteFridge()"*** s'occupe d'aller DELETE le fridge en question sur l'API

4.6.2 Lien vers script concerné

- dialog-delete.component.ts
- dialog-delete.component.html
- fridge.service.ts
- fridge.component.ts
- FridgeController.php
- Fridge.php

4.7 En tant qu'utilisateur, je veux pouvoir lister les étages de mon frigo ainsi que leur aliments

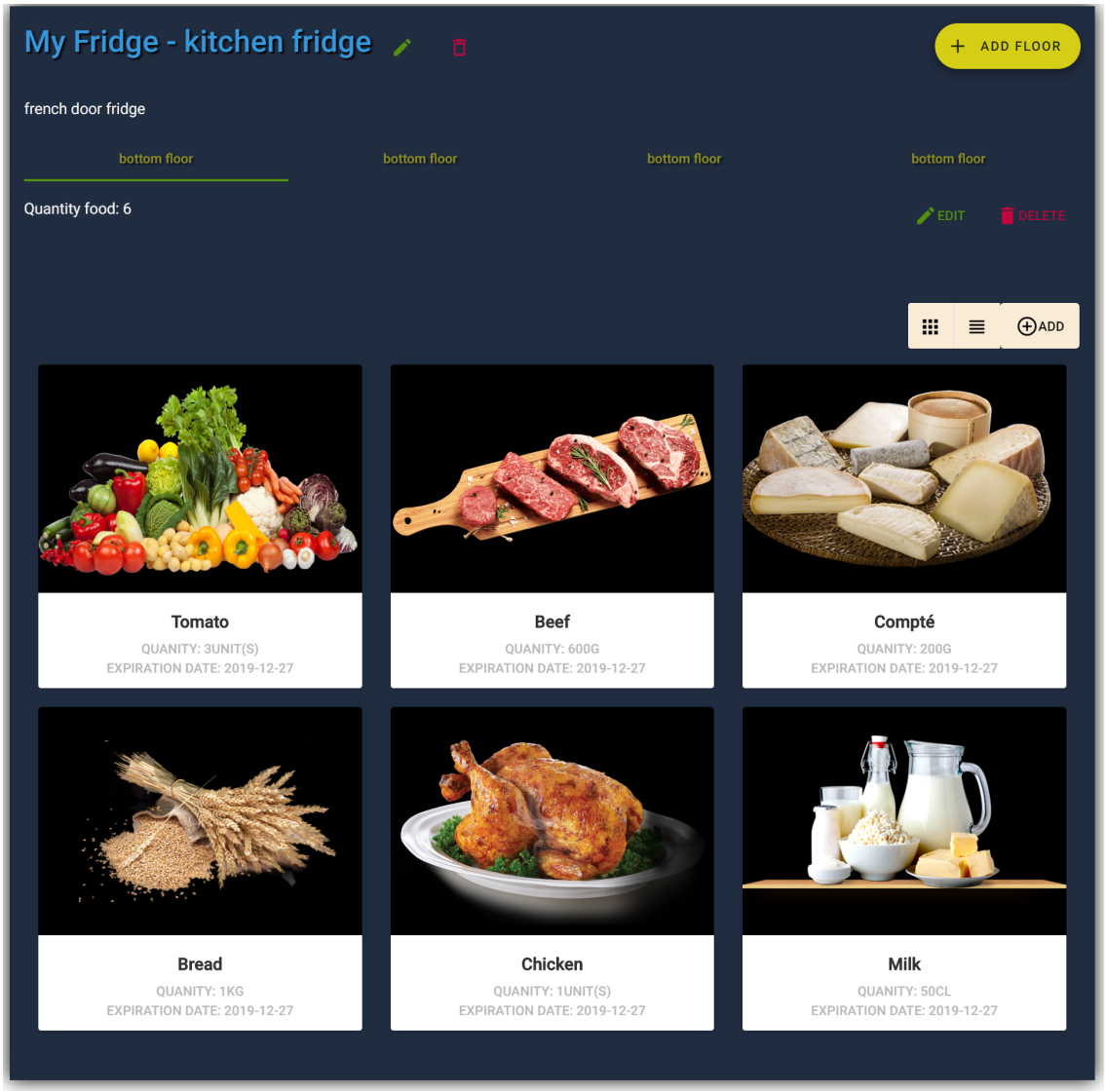

FIGURE 4.15

Pour s'y rendre, l'utilisateur doit juste cliquer sur le bouton "PREVIEW" du fridge. Ensuite, contrairement à SyT, le design de l'intérieur d'un fridge est complémetement différent. L'utilisateur peut accéder aux aliments du frigo et aux étages d'une façon beaucoup plus userfriendly grâce à l'utilisation d'onglet représentation chacun un étage. Chaque onglet contient alors les aliments de l'étage disposé de 2 manières différentes, en disposition "table" ou en disposition "grid" comme on peut voir à la figure 4.15 et la figure 4.16.

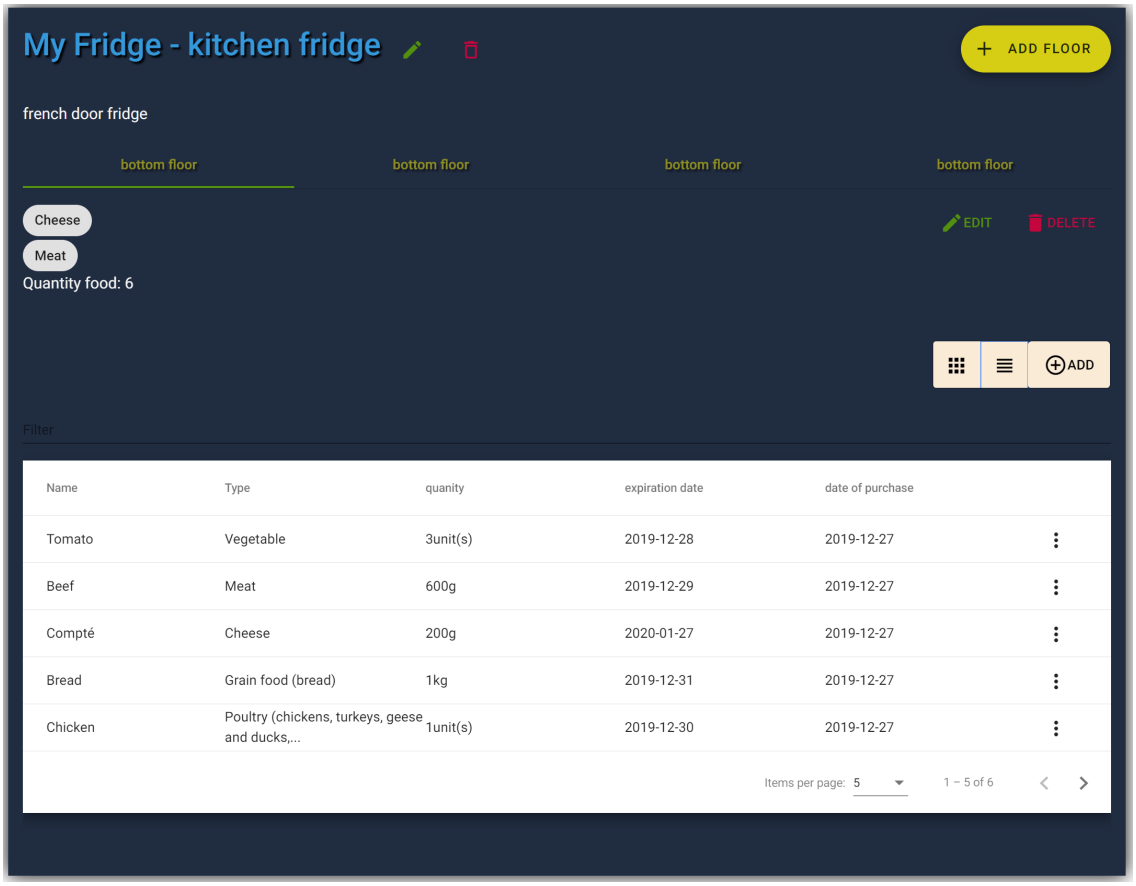

FIGURE 4.16

On peut donc accéder à ces deux disposition en cliquant sur les boutons blancs contenant des icones en forme de "grid" ou de "ligne".

FIGURE 4.17

La disposition table permet aussi à l'utilisateur de choisir combien d'aliments il veut afficher à la fois. Ceci se fait grâce à un "paginator" lier à la table qui nous permet de naviguer entre les différentes page de la table, de modifier le nombre d'aliment par page et finalement aussi de voir combien d'élement il y a dans la table.

4.7.1 Partie de code significative

Dans FridgeInsideListComponent :

- la fonction ***"ngOnit()"*** s'occupe d'aller demander les étages et les aliments du premier étage à l'API.
- la fonction ***"ngAfterViewInit()"*** s'occupe d'afficher les erreurs venant de l'api
- la fonction **"updateDataSource()"** permet d'aller chercher les aliments d'un étage de manière séquentiel. Il va aller demander les aliments de l'étage sélectionné (onglet).

Dans FloorService :

- La fonction ***"getFloors()"*** s'occupe d'aller GET les floors du fridge en question sur l'API
- La fonction ***"getFoodList()"*** s'occupe d'aller GET les foods d'étage en question sur l'API

4.7.2 Lien vers script concerné

- fridgeInsideList.component.ts
- fridgeInsideList.component.html
- floor.service.ts
- food.service.ts
- FloorController.php
- FoodController.php
- Floor.php
- Food.php

4.7.3 Diagramme de séquence

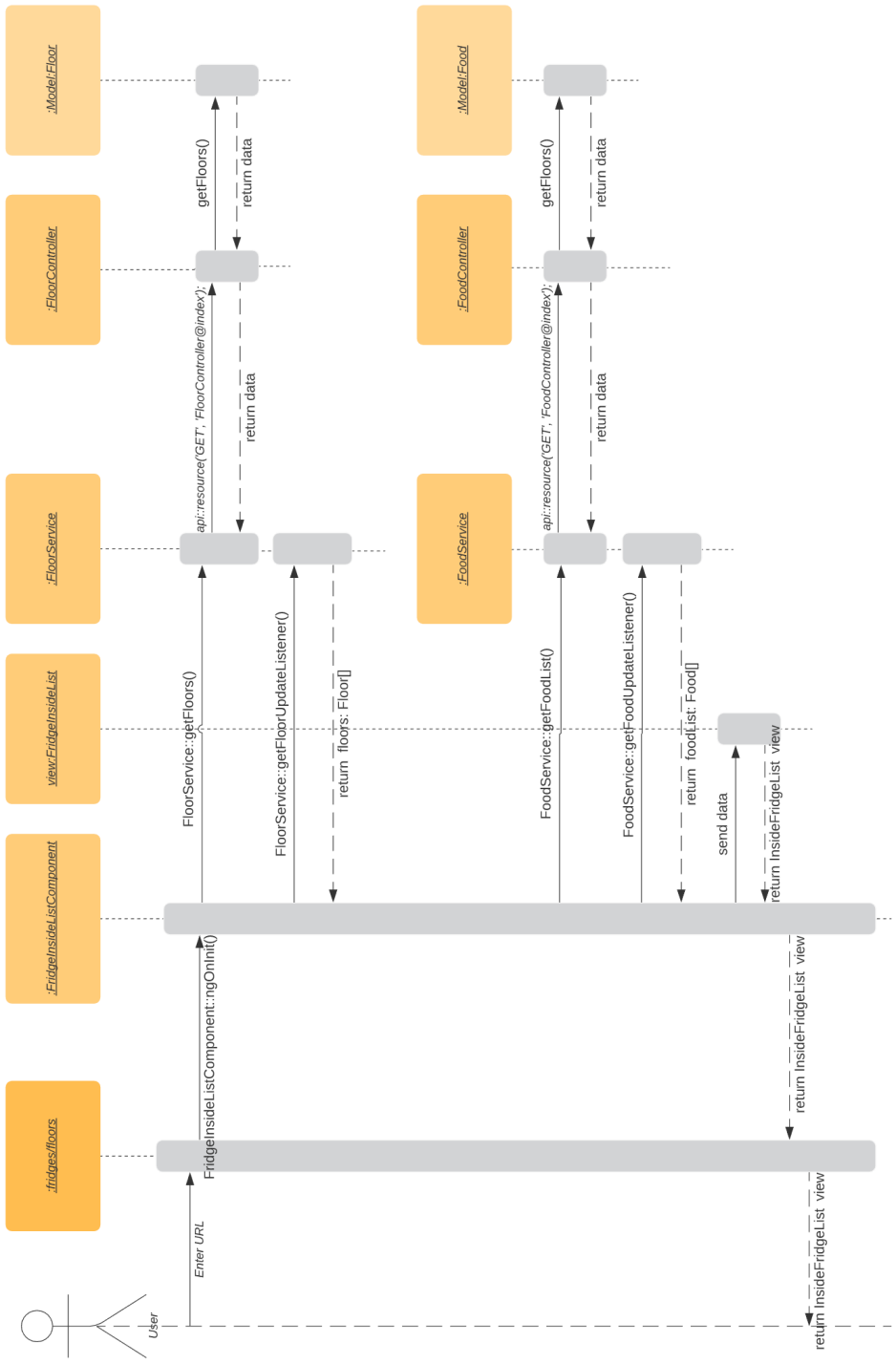

FIGURE 4.18: Diagramme de séquence pour la récupération et l'affichage des étages et des aliments d'un frigo

4.8 En tant qu'utilisateur, je veux pouvoir ajouter des étages à mon frigo

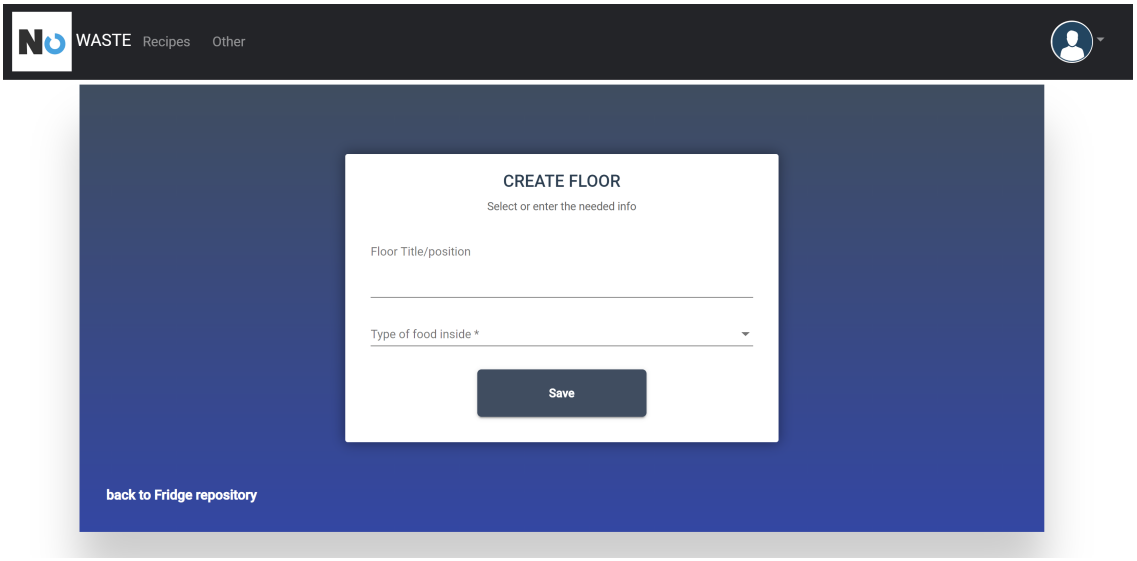

FIGURE 4.19

Pour ajouter un frigo, il suffit de cliquer sur le bouton comme montrer sur la figure suivante :

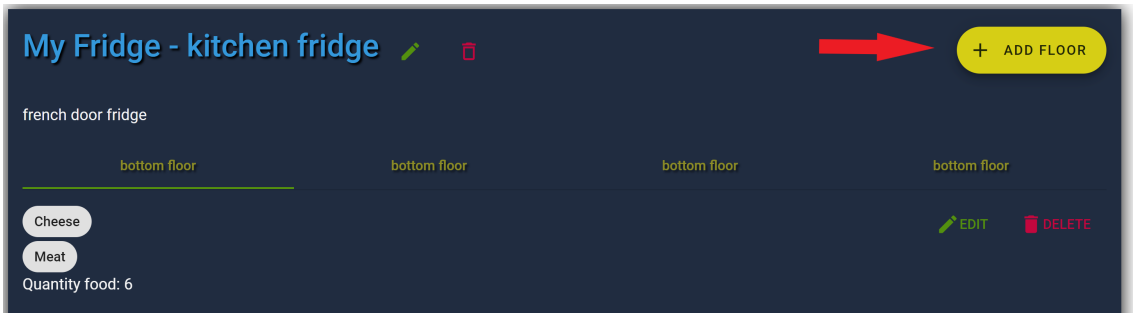

FIGURE 4.20

4.8.1 Gestion d'erreur et sécurité

Le form vérifie que les informations sont toutes encodée et correctes sinon une erreur s'affiche.

4.8.2 Partie de code significative

Dans FloorCreateComponent :

- la fonction ***"ngOnit()"*** s'occupe de créer le form
- la fonction ***"ngAfterViewInit()"*** s'occupe d'afficher les erreurs venant de l'api
- la fonction **"onSaveFloor()"** envoie les données entrée par l'utilisateur vers l'api

Dans FridgeService :

- La fonction ***"addFloor()"*** s'occupe d'aller PUT les données envoyer par le FloorCreateComponent sur l'API

4.8.3 Lien vers script concerné

- floor-create.component.ts
- floor-create.component.html
- floor.service.ts
- FloorController.php
- Floor.php

4.9 En tant qu'utilisateur, je veux pouvoir modifier les étages de mon frigo

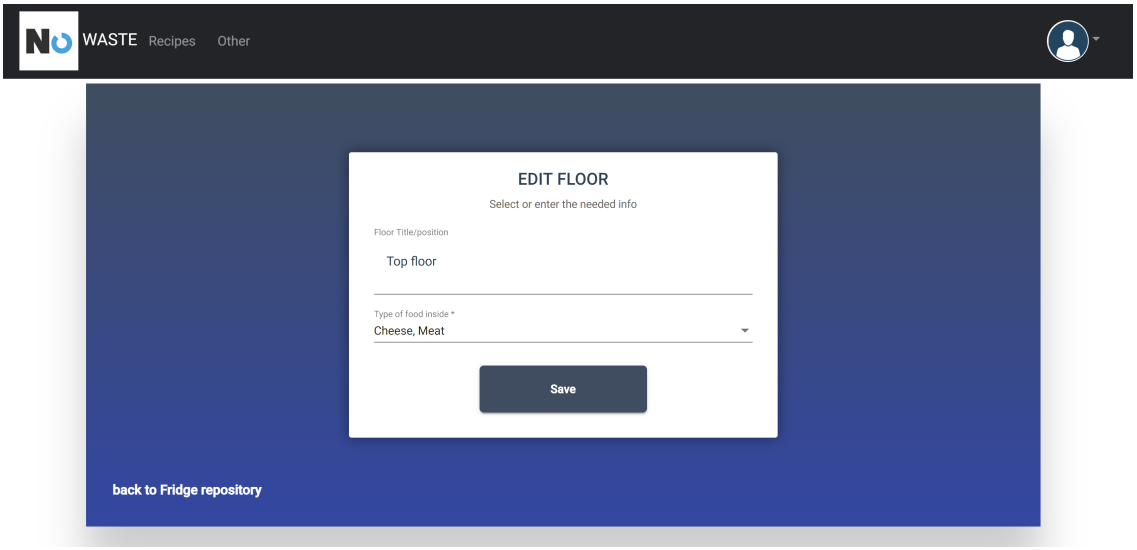

FIGURE 4.21

Pour se rendre sur cette page, il suffit à l'utilisateur de cliquer sur le bouton "EDIT" du floor en question comme sur la figure suivante :

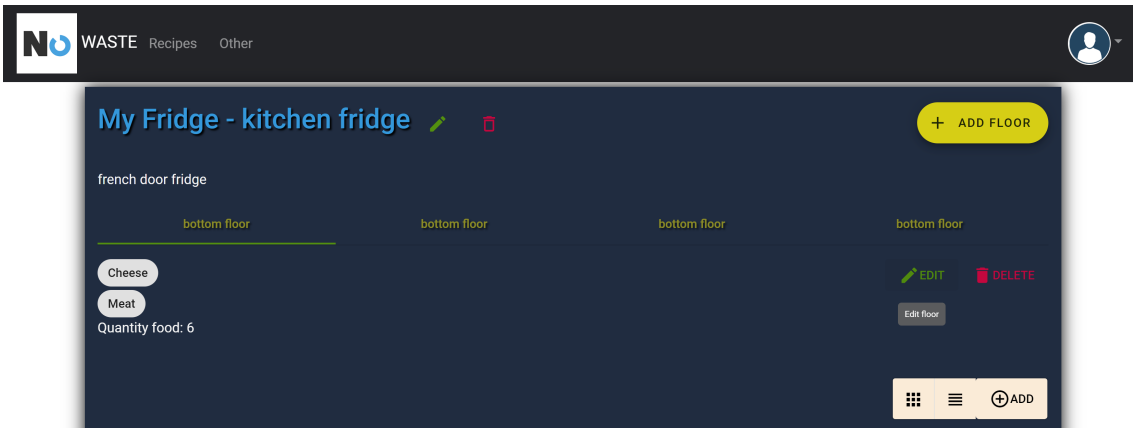

FIGURE 4.22

4.9.1 Gestion d'erreur et sécurité

Le form vérifie que les informations sont toutes encodée et correctes sinon une erreur s'affiche.

4.9.2 Partie de code significative

Dans FloorCreateComponent :

- la fonction ***"ngOnit()"*** s'occupe de créer le form et d'y ajouter les caractéristique de l'étage à modifier.
- la fonction ***"ngAfterViewInit()"*** s'occupe d'afficher les erreurs venant de l'api
- la fonction **"onSaveFloor()"** envoie les données entrée par l'utilisateur vers l'api

Dans FridgeService :

- La fonction ***"updateFloor()"*** s'occupe d'aller PUT les données envoyer par le FloorCreateComponent sur l'API

4.9.3 Lien vers script concerné

- floor-create.component.ts
- floor-create.component.html
- floor.service.ts
- FloorController.php
- Floor.php

4.10 En tant qu'utilisateur, je veux pouvoir supprimer les étages de mon frigo

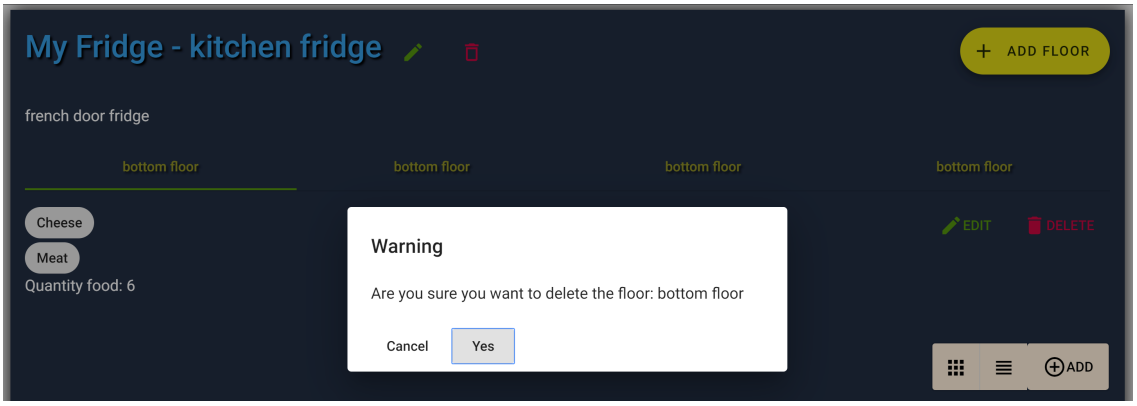

FIGURE 4.23

Pour supprimer un étage, il suffit à l'utisateurs de cliquer sur le bouton "DELETE" comme sur l'image suivante :

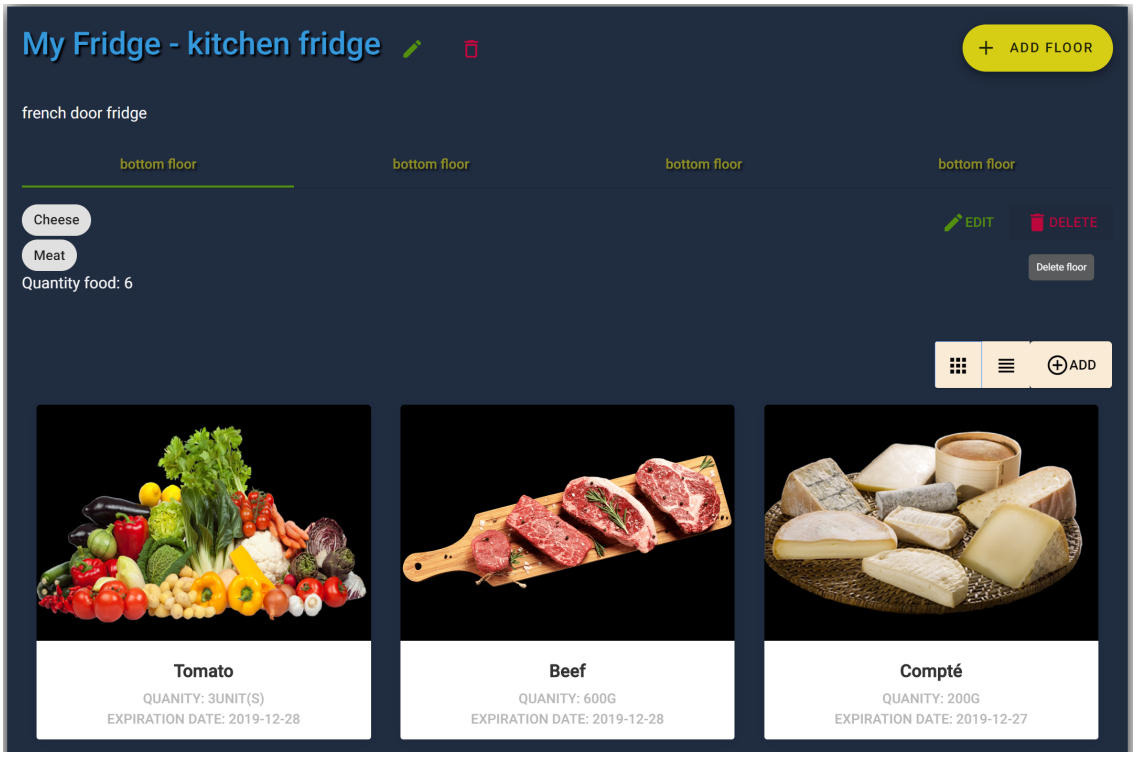

FIGURE 4.24

L'utilisateur n'a plus qu'à confirmer la suppression avec le dialog box qui aura apparu.

4.10.1 Partie de code significative

Dans FridgeInsideListComponent :

- la fonction ***"openDialog()"*** s'occupe d'ouvrir un nouveau dialog pour la confirmation de la suppression

Dans DialogDeleteComponent :

- la fonction ***"ngAfterViewInit()"*** s'occupe d'afficher un snackbar négatif en cas d'erreur
- la fonction **"onYesClick()"** appelle la méthode "deleteFloor" du FloorService et se souscrit à tout type d'erreur venant de l'api

Dans FloorService :

- La fonction ***"deleteFloor()"*** s'occupe d'aller DELETE le floor en question sur l'API

4.10.2 Lien vers script concerné

- dialog-delete.component.ts
- dialog-delete.component.html
- floor.service.ts
- fridgeInsideList.component.ts
- FloorController.php
- Floor.php

4.11 En tant qu'utilisateur, je veux pouvoir ajouter des aliments à mon frigo

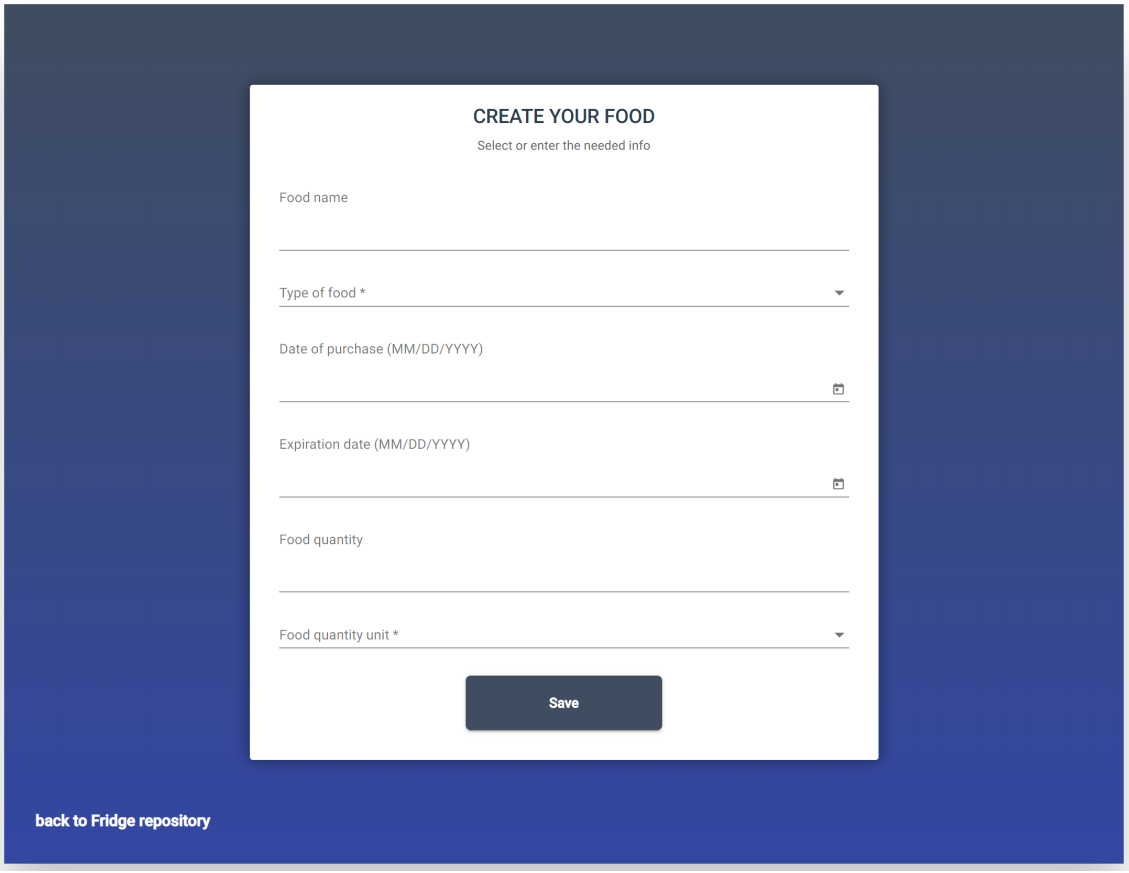

FIGURE 4.25

Pour ajouter un aliments à un étage, il suffit à l'utilisateur de cliquer sur le bouton "Add" comme montrer sur la figure 4.26.

Le form est muni dans la partie angular pour le choix de la date d'expiration et date d'achat d'un calendrier plus userfriendly comme montrer à la figure 4.27.

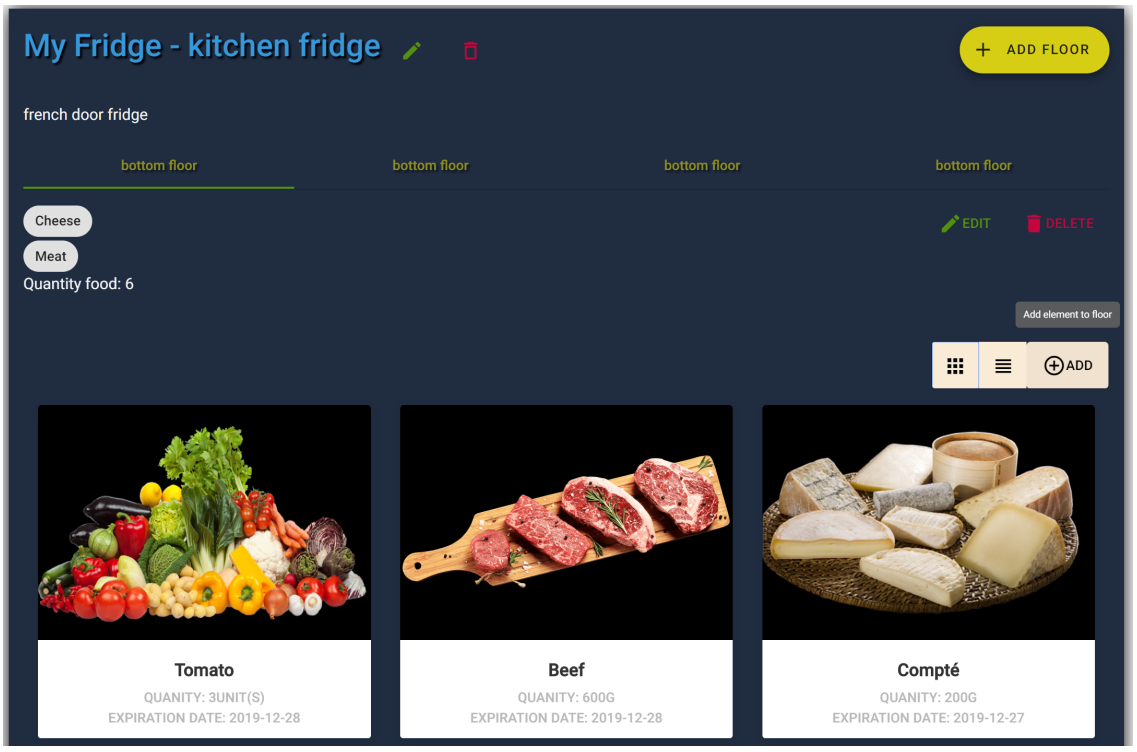

FIGURE 4.26

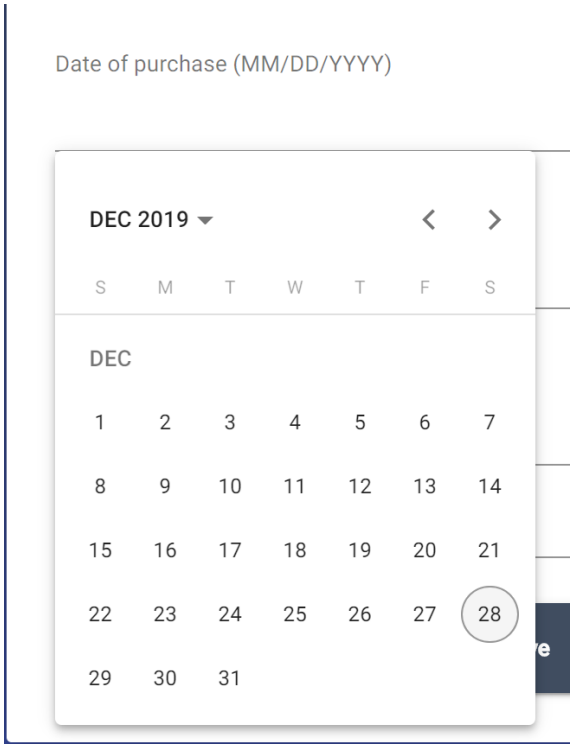

FIGURE 4.27

4.11.1 Gestion d'erreur et sécurité

Le form vérifie que les informations sont toutes encodée et correctes sinon une erreur s'affiche.

4.11.2 Partie de code significative

Dans FoodCreateComponent :

- la fonction ***"ngOnit()"*** s'occupe de créer le form
- la fonction ***"ngAfterViewInit()"*** s'occupe d'afficher les erreurs venant de l'api
- la fonction **"onSaveFood()"** envoie les données entrée par l'utilisateur vers l'api

Dans FoodService :

- La fonction ***"addFood()"*** s'occupe d'aller POST les données envoyer par le FoodCreateComponent sur l'API

4.11.3 Lien vers script concerné

- food-create.component.ts
- food-create.component.html
- food.service.ts
- FoodController.php
- Food.php

4.12 En tant qu'utilisateur, je veux pouvoir modifier des aliments de mon frigo

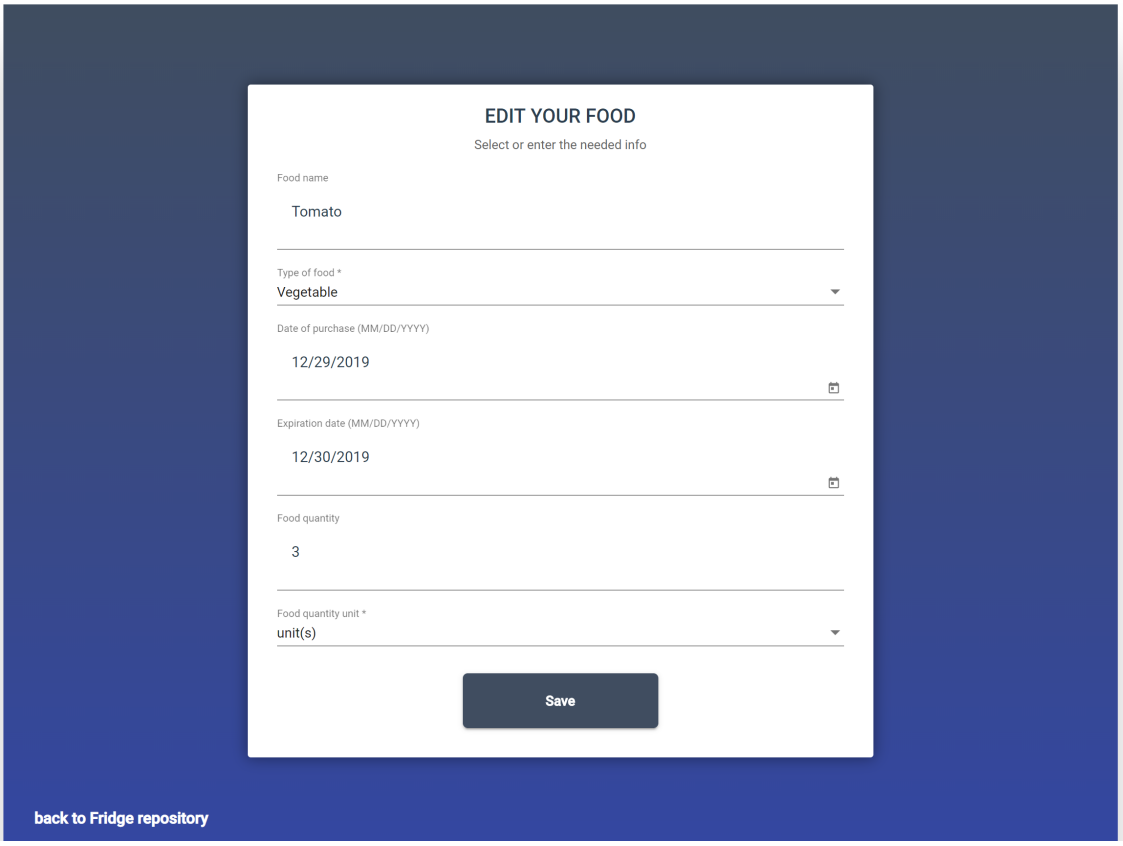

FIGURE 4.28

Pour y accéder, l'utilisateur clique sur bouton "Edit" du food en question qu'il veut modifier comme montrer sur les figures suivantes :

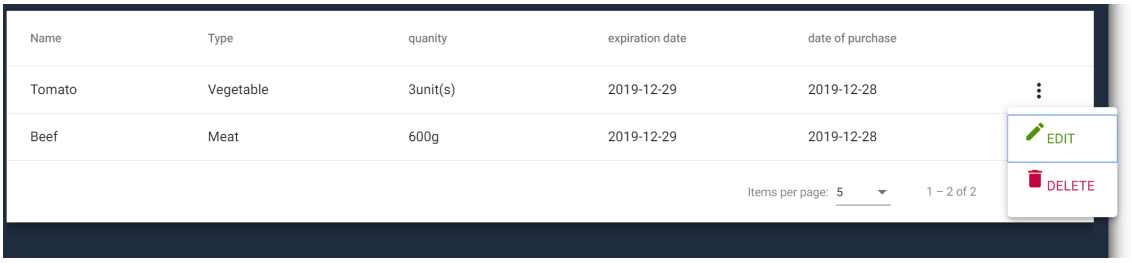

FIGURE 4.29

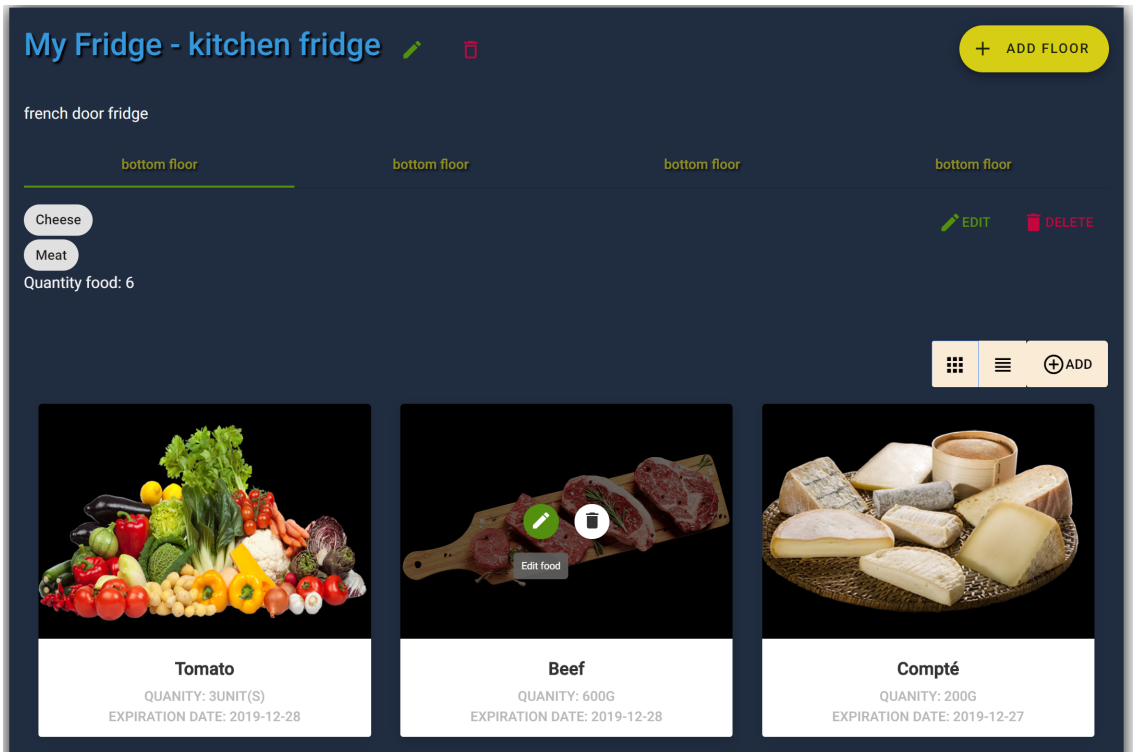

FIGURE 4.30

4.12.1 Gestion d'erreur et sécurité

Le form vérifie que les informations sont toutes encodée et correctes sinon une erreur s'affiche.

4.12.2 Partie de code significative

Dans FoodCreateComponent :

- la fonction ***"ngOnit()"*** s'occupe de créer le form et d'y ajouter les caractéristique du food en question
- la fonction ***"ngAfterViewInit()"*** s'occupe d'afficher les erreurs venant de l'api
- la fonction **"onSaveFood()"** envoie les données entrée par l'utilisateur vers l'api

Dans FoodService :

- La fonction ***"addFood()"*** s'occupe d'aller PUT les données envoyer par le FoodCreateComponent sur l'API

4.12.3 Lien vers script concerné

- food-create.component.ts
- food-create.component.html
- food.service.ts
- FoodController.php
- Food.php

4.13 En tant qu'utilisateur, je veux pouvoir supprimer des aliments de mon frigo

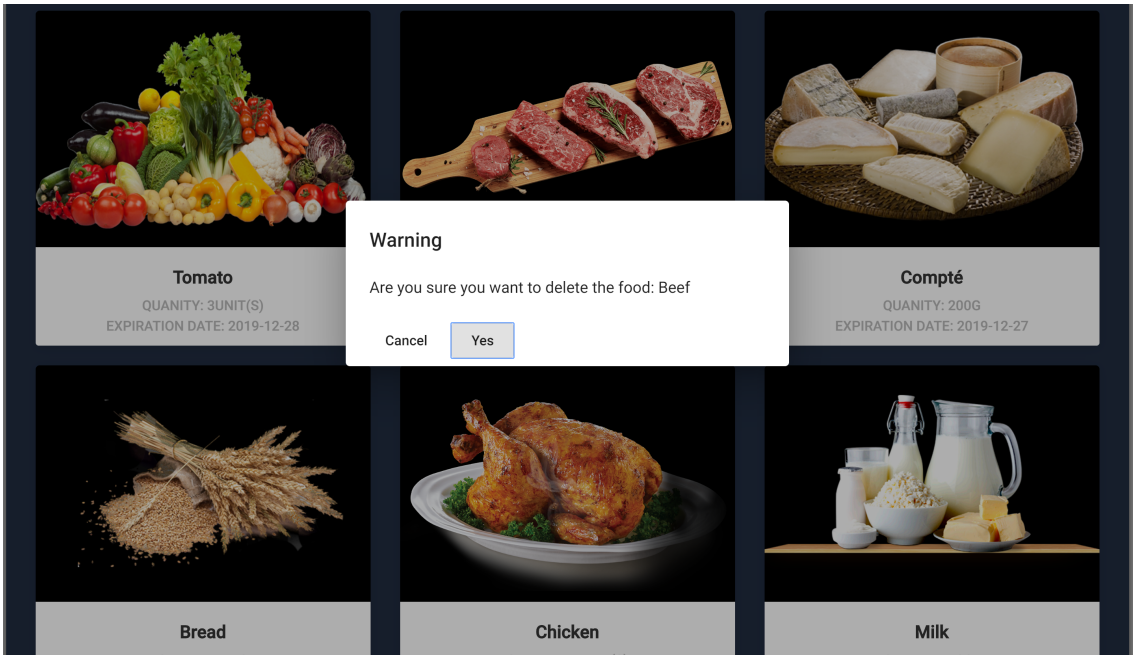

FIGURE 4.31

Pour supprimer un aliment, l'utilisateur clique sur bouton "Delete" du food en question qu'il veut supprimer comme montrer sur les figures suivantes :

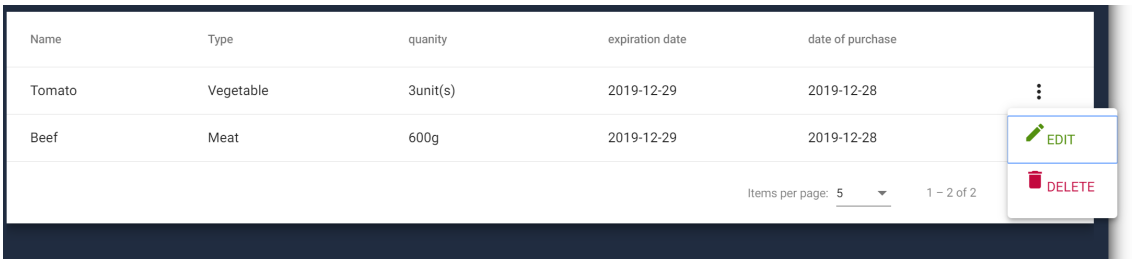

FIGURE 4.32

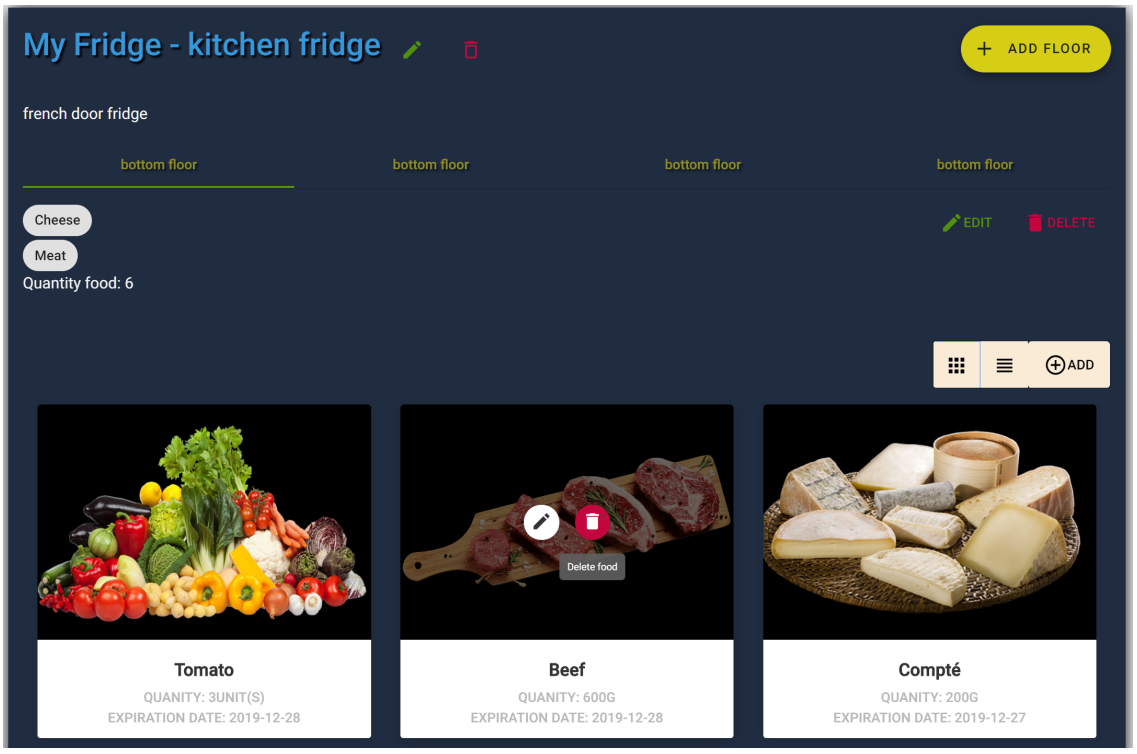

FIGURE 4.33

4.13.1 Partie de code significative

Dans FridgeInsideListComponent :

- la fonction ***"openDialog()"*** s'occupe d'ouvrir un nouveau dialog pour la confirmation de la suppression

Dans DialogDeleteComponent :

- la fonction ***"ngAfterViewInit()"*** s'occupe d'afficher un snackbar négatif en cas d'erreur
- la fonction **"onYesClick()"** appelle la méthode "deleteFood" du FoodService et se souscrit à tout type d'erreur venant de l'api

Dans FoodService :

- La fonction ***"deleteFood()"*** s'occupe d'aller DELETE le food en question sur l'API

4.13.2 Lien vers script concerné

- dialog-delete.component.ts
- dialog-delete.component.html
- food.service.ts
- fridgeInsideList.component.ts
- FoodController.php
- Food.php

4.14 En tant qu'utilisateur, je veux pouvoir rechercher des aliments selon une caractéristique

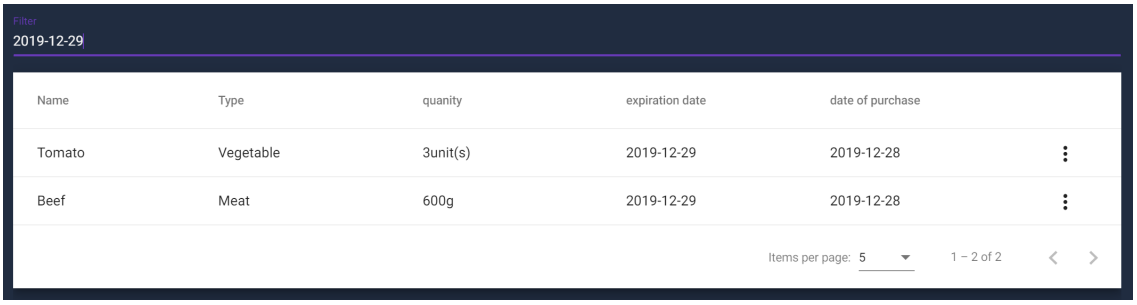

FIGURE 4.34

L'utilisateur est capable quand il a choisi la disposition table pour les aliments de faire une recherche par rapport à des parties de caractère se trouvant dans chaque colonne comme montrer à la figure 4.34. Grâce à se filtre, l'utilisateur est donc capable de faire des recherches précise dans un certain étage pour retrouver plus facilement des aliments.

4.14.1 Partie de code significative

Dans FridgeInsideListComponent :

- la fonction ***"applyFilter()"*** s'occupe d'appliquer un filtre sur la table.

4.14.2 Lien vers script concerné

- fridgeInsideList.component.ts
- fridgeInsideList.component.html

4.15 En tant qu'utilisateur, je veux pouvoir lister les aliments en fonction de leur type ou leur date de fin

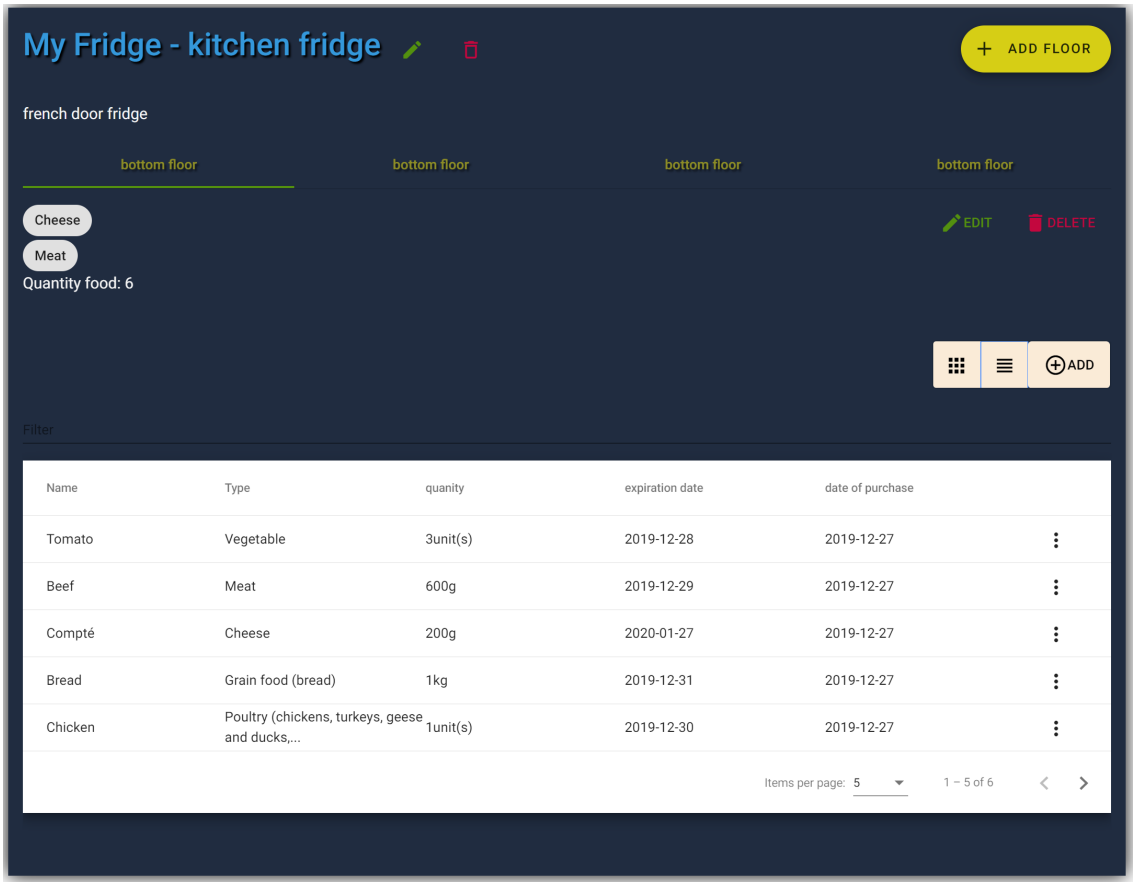

FIGURE 4.35

Cette fonction ne fonctionne pas pour l'instant pour une cause non identifier malgré que le tout à été implémenté permettant de faire un "sort" sur chaque colonne de la table et ainsi trier les colonnes selon les souhait de l'utilisateur, donc ici fonction de leur type ou leur date d'expiration.

4.15.1 Lien vers script concerné

- fridgeInsideList.component.ts
- fridgeInsideList.component.html

Conclusion

5.1 Objectifs atteints et non atteints

5.1.1 Symfony-Twig

User Stories	priority	Realisation	Reste à faire
En tant qu'utilisateur, je veux pouvoir me connecter	1	90%	Un moyen de récupérer un mot de passe oublié
En tant qu'utilisateur, je veux pouvoir m'enregister	1	90%	meilleur gestion des erreurs et aide comme dans Angular
En tant qu'utilisateur, je veux pouvoir retrouver les frigos que j'ai créé	1	100%	
En tant qu'utilisateur, je veux pouvoir créer un frigo ou plusieurs	1	90%	Ajout d'un moyen d'upload sa propre photo de frigo
En tant qu'utilisateur, je veux pouvoir modifier mon frigo	1	90%	Même problème que la création
En tant qu'utilisateur, je veux pouvoir supprimer mon frigo	1	100%	

User Stories	priority	Realisation	Reste à faire
En tant qu'utilisateur, je veux pouvoir lister les étages de mon frigo	1	90%	Affihchage propre des type d'aliments (1)
En tant qu'utilisateur, je veux pouvoir ajouter des étages à mon frigo	1	80%	Trouver une meilleur manière de créer un nouvel étage
En tant qu'utilisateur, je veux pouvoir modifier les étages de mon frigo	1	80%	Ajouter un dropdown list avec les types ou supprimer le type (voir partie affichage étage)
En tant qu'utilisateur, je veux pouvoir supprimer les étages de mon frigo	1	100%	
En tant qu'utilisateur, je veux pouvoir lister les aliments se trouvant dans un étage	1	90%	Manière plus facile de visualiser les aliments et plus spécifique
En tant qu'utilisateur, je veux pouvoir afficher un aliment spécifique	2	90%	Meilleur design
En tant qu'utilisateur, je veux pouvoir ajouter des aliments à mon frigo	1	100%	
En tant qu'utilisateur, je veux pouvoir modifier des aliments de mon frigo	1	100%	
En tant qu'utilisateur, je veux pouvoir supprimer des aliments de mon frigo	1	100%	

(1) Après multiple changement dans la base de données pour la partie Angular, Il a fallu modifier quelque peu la partie symfony. Cependant il reste encore à changer un truc dans la partie symfony, à savoir l'affichage et la modification du type. Ce dernier était initialement un string et a été changé en Array pour pouvoir choisir plusieurs types de food et y pouvoir accéder assez facilement par après.

5.1.2 Symfony-Angular

User Stories	priority	Realisation	Reste à faire
En tant qu'utilisateur, je veux pouvoir me connecter	1	90%	Un moyen de récupérer un mot de passe oublié
En tant qu'utilisateur, je veux pouvoir m'enregister	1	90%	Meilleur gestion des erreurs et aide comme dans Angular
En tant qu'utilisateur, je veux pouvoir retrouver les frigos que j'ai créé	1	100%	
En tant qu'utilisateur, je veux pouvoir créer un frigo ou plusieurs	1	90%	Ppouvoir upload sa propre image de frigo
En tant qu'utilisateur, je veux pouvoir modifier mon frigo	1	90%	Trouver une autre manière d'empecher l'utilisateur de modifier le nombre d'étage d'un frigo (pour l'instant css)
En tant qu'utilisateur, je veux pouvoir supprimer mon frigo	1	100%	
En tant qu'utilisateur, je veux pouvoir lister les étages de mon frigo ainsi que leur aliments	1	100%	
En tant qu'utilisateur, je veux pouvoir ajouter des étages à mon frigo	1	80%	Trouver une meilleur manière de créer un nouvel étage
En tant qu'utilisateur, je veux pouvoir modifier les étages de mon frigo	1	100%	
En tant qu'utilisateur, je veux pouvoir supprimer les étages de mon frigo	1	100%	

User Stories	priority	Realisation	Reste à faire
En tant qu'utilisateur, je veux pouvoir afficher un aliment spécifique	2	0%	pas encore possible
En tant qu'utilisateur, je veux pouvoir ajouter des aliments à mon frigo	1	90%	Pouvoir éventuellement ajouter sa propre photo de l'aliment (pas sûre pour une question de design)
En tant qu'utilisateur, je veux pouvoir modifier des aliments de mon frigo	1	100%	
En tant qu'utilisateur, je veux pouvoir supprimer des aliments de mon frigo	1	100%	
En tant qu'utilisateur, je veux pouvoir rechercher des aliments selon une caractéristique	2	90%	Pouvoir faire une recherche sur une colonne spécifique
En tant qu'utilisateur, je veux pouvoir lister les aliments en fonction de leur type ou leur date de fin	2	50%	Méthode en place mais pour raison inconnu ne s'active pas (1)

(1) Vu que les données du tableau sont chargé après création du tableau et change, il était nécessaire d'activer le sort et la paginator dans le ngAfterViewInit(). Cette méthode étant censé fonctionner dans les deux cas. Cependant cela fonctionne seulement pour le paginator et pas pour le sort.

5.2 Piste d'amélioration

- Optimisation des requêtes de base de données
- Nettoyage du code source (duplication)
- Renomage des clés json envoyées au frontend et de certain champ de la base de données pour compréhension plus simple
- Meilleur gestion des erreurs et affichage dans la partie symfony
- Faire en sorte que les types de food d'un étage soit automatique en fonction des des aliments qu'on y ajoute.
- Ajout d'une d'une partie recettes allant chercher des recettes sur une API.
- Ajout d'une partie création de recettes
- Ajout d'un système de recherche de recettes en fonction des aliments dans la frigo