Projet 8

Plateforme Nutella

Jean baptiste Servais

Plan

- 1 - Première recherche
- 2 - Inscription
- 3 - Connexion
- 4 - Aliment
- 5 – remplacement
- 6 - database

Les pages HTML

- Inclus une navbarre navebarre.html
- Inclus un bottom bottom.html
- Relié à un fichier js
- Relié à un fichier css

1 - La première recherche

a) Simulation web

b) Les pages Html

→ *Navebarre.html*

→ *Home.html*

c) La fonction search() du fichier views.py

d) Et ses fonctions associées.

→ controle_data_food()

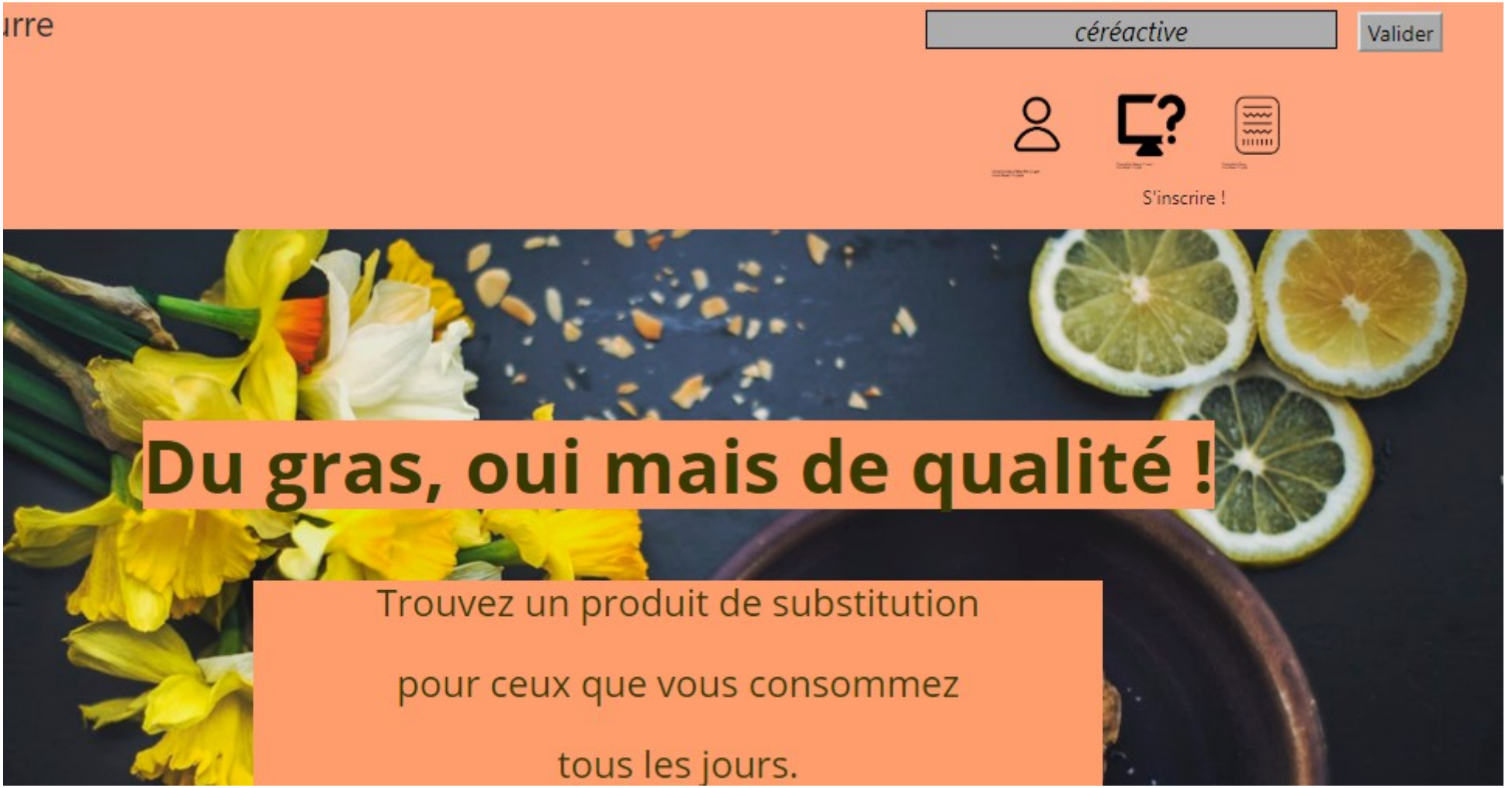
→ better_nutri()

→ image_food()

→ title_food()

a) la recherche - simulation web

L'utilisateur fait sa première recherche



1a) la recherche - simulation web
L'utilisateur fait sa première recherche

Connecte-toi afin de pouvoir les **enregistrer** et visualise les depuis **"Mon compte"**

Connecte-toi afin de pouvoir les **enregistrer** et visualise les depuis **"Mon compte"**

céréactive

japanese mochi "red bean" 210g

céréactive

1b) La recherche - les pages HTML

Input de la page navebarre.html et input de la page home.html

```
<form action ="/mes_aliments/recherche/" method="post">
  <input type="hidden" name="csrfmiddlewaretoken" value="ZpPGYe4an9kbhNTmpXYQUBxfWbFxQnWq20X3SVmFrLBY
  <input type="hidden" name="csrfmiddlewaretoken" value="EamjY5jhJJz2DK6tRyzNuimbCmgWR39MDQ8h0gSiKiGj
  <input name="cool" class="input_top_haut_droit" type="text" size:"30" placeholder="Produit"/>
  <input class="input_desous_h2" type="submit" value="Chercher"/>
</form>
```

```
</div>
<form action ="/mes_aliments/recherche/" method="post">
  <input type="hidden" name="csrfmiddlewaretoken" value="ZpPGYe4an9kbhNTmpXYQUBxfWbFxQnWq20X3SVmFrLBYDz1HyQacjnVhPE8nfBY7">
  <input type="hidden" name="csrfmiddlewaretoken" value="EamjY5jhJJz2DK6tRyzNuimbCmgWR39MDQ8h0gSiKiGjLb0OABbO443CiSMNOeH4">
  <input name="cool" class="input_top_haut_droit" type="text" size:"30" placeholder="Produit"/>
  <input class="input_desous_h2" type="submit" value="Chercher"/>
</form>
```

1c) la recherche - la fonction search()

```
if request.method == "POST":

    search = request.POST.get('cool')
    username = request.POST.get('username')
    validate = request.POST.getlist('data[]')
```

1c) la recherche - la fonction search()

```
if search:

    current_user = request.user

    if current_user.is_authenticated:
        stock = controle_data_food(current_user)

        if stock[1] == False:
            exceeded_stock = "oups vous avez trop d'aliment\
            en stock supprime en ! ou remplace le !"

    image = image_food(search)
    title = title_food(search)

    try:

        a = better_nutri(search)

        return render(request, 'recherche.html',
                      {"a":str(a[0][3]),
                       "b":str(a[1][3]),
                       "c":str(a[2][3]),
                       "d":str(a[3][3]),
                       "e":str(a[4][3]),
                       "f":str(a[5][3]),

                       "aa":str(a[0][0]),
                       "bb":str(a[1][0]),
                       "cc":str(a[2][0]),
                       "dd":str(a[3][0]),
                       "ee":str(a[4][0]),
                       "ff":str(a[5][0]),

                       "aaa":str(a[0][3]),
                       "bbb":str(a[1][3]),
                       "ccc":str(a[2][3]),
                       "ddd":str(a[3][3]),
                       "eee":str(a[4][3]),
```

1c) la recherche - la fonction search()
views.py

```
                                "aaaa":"/static/img/portfolio/nutriscore/" + str
                                "bbbb":"/static/img/portfolio/nutriscore/" + str
                                "cccc":"/static/img/portfolio/nutriscore/" + str
                                "dddd":"/static/img/portfolio/nutriscore/" + str
                                "eeee":"/static/img/portfolio/nutriscore/" + str
                                "ffff":"/static/img/portfolio/nutriscore/" + str

                                "image":str(image),
                                "titre":str(title),
                                "stock_depassé":exceeded_stock,
                                })

        except:
            message = "oups nous n'avons pas cet aliment en database"
            return render(request, 'error.html', {"message":message})

image = '/static/img/header1.jpg'
return render(request, 'recherche.html', {'image':image})
```

1d) la recherche - les fonctions associées
controle_data_food()

```
def controle_data_food(username):
    """Here we watch if user have 6 products,
    if he has -6 we ask him to select products
    else we warned him to modify his selection"""

    c = foodAccount.objects.get(name=username)

    liste = [c.name_aliment1, c.name_aliment2, c.name_aliment3,
             c.name_aliment4, c.name_aliment5, c.name_aliment6,]

    number = 0
    for i in liste:
        if i != "":
            number += 1

    if number >= 6:
        return "nombre de produit suppérieur a 6", False

    else:
        return "stockage du produit possible", True
```

1d) la recherche - les fonctions associées

image_food()

```
def image_food(para):
    """Here we search food picture """
    try:
        try:
            food = aliment.objects.get(name_aliment__contains='{}'.format(para))
            food = aliment.objects.get(name_aliment=para)
            image = food.image
            return image

        except:
            para = para.split()
            food = aliment.objects.get(name_aliment__contains=str(para[0]))
            image = food.image
            return image
    except:
        pass
```

1d) la recherche - les fonctions associées

title_food()

```
def title_food(para):
    """Here we search title picture """

    try:
        try:
            food = aliment.objects.get(name_aliment=para)
            title = food.name_aliment
            return title

        except:
            para = para.split()
            food = aliment.objects.get(name_aliment__contains=str(para[0]))
            title = food.name_aliment
            return title
    except:
        pass
```

1d) la recherche - les fonctions associées

better_nutri()

```
def better_nutri(para):
    """Here we search best nutriscore from category
    from food search"""

    food = aliment.objects.get(name_aliment=para)
    food_search = [food.name_aliment, food.id_categorie_id,
                        food.nutriscore, food.image, food.id]

    category = aliment.objects.filter(id_categorie_id=food.id_categorie_id).orde
        'nutriscore')

    liste = []

    count = 0
    for i in category:
        if count == 20:
            break
        else:
            a = []
            a = [i.name_aliment, i.id_categorie_id,
                 i.nutriscore, i.image]
            liste.append(a)

        count += 1


    liste = liste[:6]
    liste[0] = food_search

    return liste
```

2 – L'inscription

a) Simulation web

b) Les formulaires

→ *forms.py*

c) Les modèles

→ models.py

d) La fonction register_views() du fichier views.py

2a) l'inscription - simulation web

l'utilisateur s'inscrit

2b) L'inscription - les formulaires

Forms.py

```
class UserRegisterForm(forms.ModelForm):
    """This is form for register"""

    username = forms.CharField()
    email = forms.EmailField()
    email2 = forms.EmailField()

    password = forms.CharField(widget=forms.PasswordInput)

    class Meta:
        """We call username email and password from meta class"""

        model = User
        fields = [
            'username',
            'email',
            'email2',
            'password',
            ]

    def clean(self, *args, **kwargs):
        """We cleanning it"""

        email = self.cleaned_data.get('email')
        email2 = self.cleaned_data.get('email2')
        if email != email2:
            raise forms.ValidationError("pas les meme email")

        email_qs = User.objects.filter(email=email)
        if email_qs.exists():
            raise forms.ValidationError(
                "email existe deja")

        return super(UserRegisterForm, self).clean(*args, **kwargs)
```

2c) L'inscription - les modèles
models.py

```python
from django.db import models
#importation of basic model

class foodAccount(models.Model):
    """foodAccount model"""

    name = models.CharField(max_length=50)
    name_aliment1 = models.CharField(max_length=100, null=False)
    name_aliment2 = models.CharField(max_length=100, null=False)
    name_aliment3 = models.CharField(max_length=100, null=False)
    name_aliment4 = models.CharField(max_length=100, null=False)
    name_aliment5 = models.CharField(max_length=100, null=False)
    name_aliment6 = models.CharField(max_length=100, null=False)
```

2d) L'inscription - register_views()
views.py

```
def register_view(request):
    """Here we define the register view"""

    next = request.GET.get('next')
    form = UserRegisterForm(request.POST or None)

    if form.is_valid():

        user = form.save(commit=False)
        password = form.cleaned_data.get('password')
        user.set_password(password)
        user.save()

        data_food = foodAccount(name = user.username)
        data_food.save()


        create_database_user(user.username)
        insert_database_user(user.username)
        create_data_score_user(user.username)
        insert_data_score_user(user.username)

        new_user = authenticate(username=user.username, password=password)

        login(request, new_user)

        if next:
            return redirect(next)
        return redirect('/')

    context = {
        'form':form
    }

    return render(request, 'signup.html', context)
```

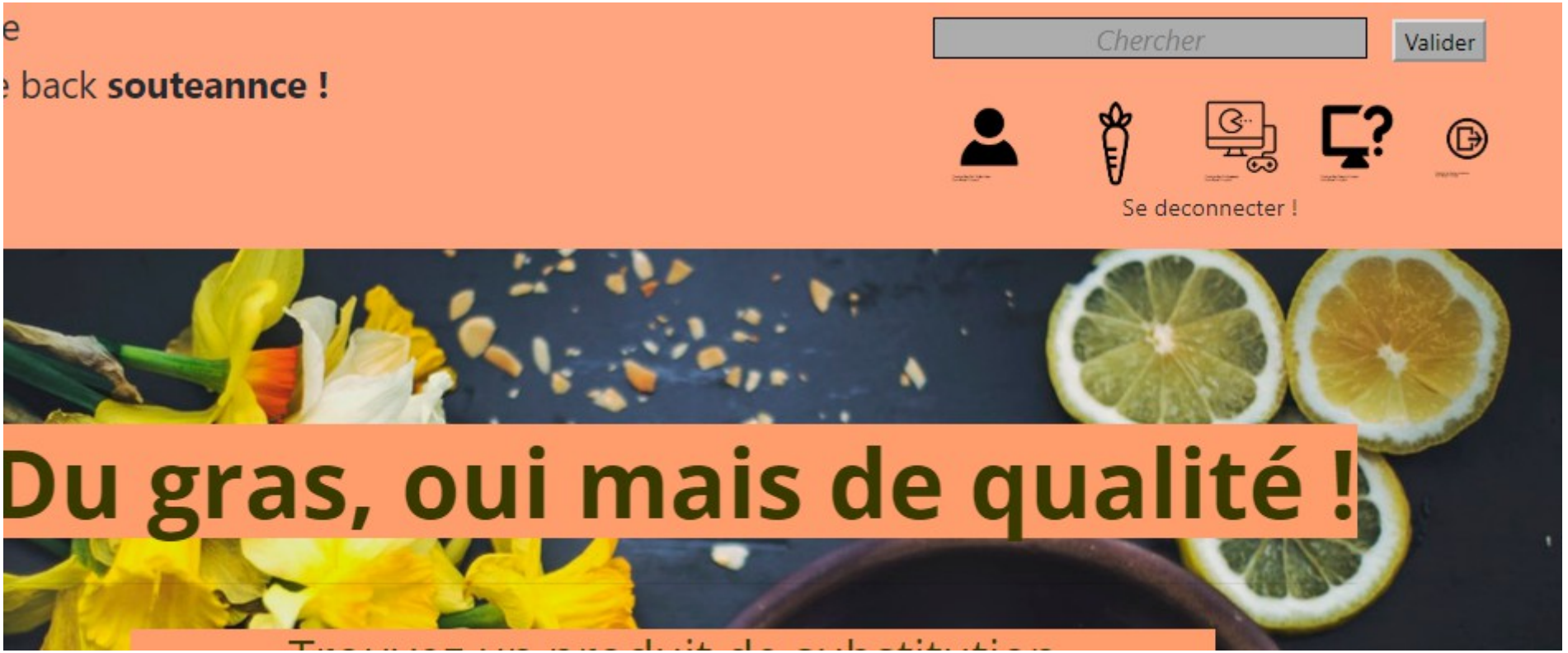
3 – La déconnexion

a) simulation web

b) la fonction logout_views() du fichier views.py

3a) Déconnexion – simulation web

l'utilisateur se déconnecte



3b) déconnexion - la fonction logout_views()

views.py

```
login_required
def logout_view(request):
    """Here we define logout session"""

    logout(request)
    print("déconnexion")
    return redirect('/')
```

4 – La connexion

a) Simulation web

b) les formulaires

→ *forms.py*

c) la fonction login_view()

→ *views.py*

4a) connexion – simulation web

L'utilisateur se connecte

4b) connexion – les formulaires
forms.py

```
class UserLoginForm(forms.Form):
    """this is form for login and run session"""

    username = forms.CharField()
    password = forms.CharField(widget=forms.PasswordInput)

    def clean(self, *args, **kwargs):
        """cleanning entrance"""

        username = self.cleaned_data.get('username')
        password = self.cleaned_data.get('password')

        if username and password:
            user = authenticate(username=username, password=password)
            if not user:
                raise forms.ValidationError('logins erronés')

            if not user.check_password(password):
                forms.ValidationError('password erronés')

        return super(UserLoginForm, self).clean(*args, **kwargs)
```

4c) connexion – la fonction login_views()

views.py

```
def login_view(request):
    """Here we define the login view"""

    next = request.GET.get('next')
    form = UserLoginForm(request.POST or None)
    if form.is_valid():
        username = form.cleaned_data.get('username')
        password = form.cleaned_data.get('password')
        user = authenticate(username=username, password=password)
        login(request, user)
        if next:
            return redirect(next)
        return redirect('/')

    context = {
        'form':form
    }

    return render(request, 'login.html', context)
```

L'utilisateur veut voir ses aliments pour la première fois

Veuillez remplir votre selection d'aliment de 6 produit svp =)

5 - La deuxième recherche

a) simulation web

b) les pages html

→ recherche.html

c) javascript

→ recherche.js

d) simulation web

e) javascript

→ recherche.js

f) la fonction search()

g) les fonctions associées

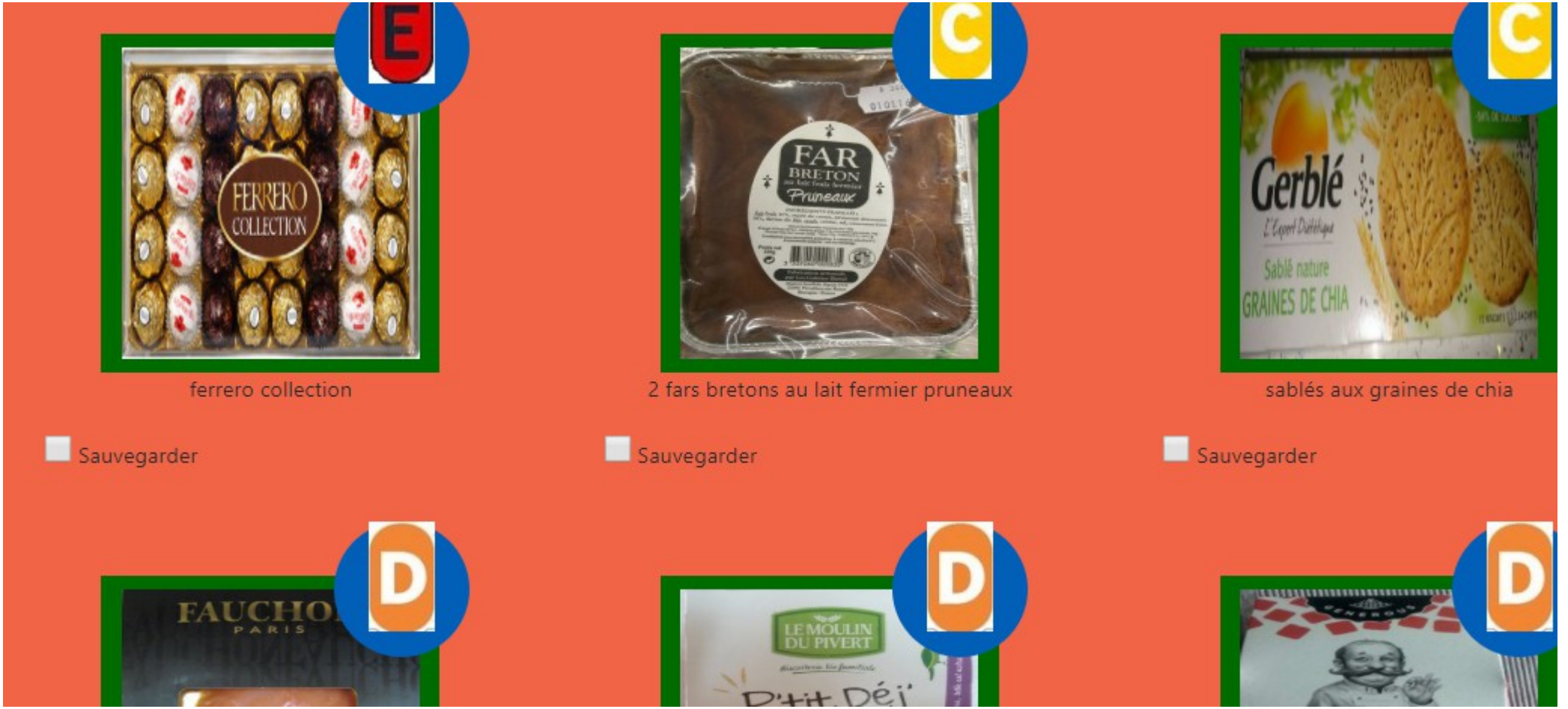
→ *controle_data_food()*

→ verification_not_two()

→ insert_food()

5a) La recherche - simulation web

L'utilisateur fait sa deuxième recherche



5b) La recherche - Page.html
Recherche.html

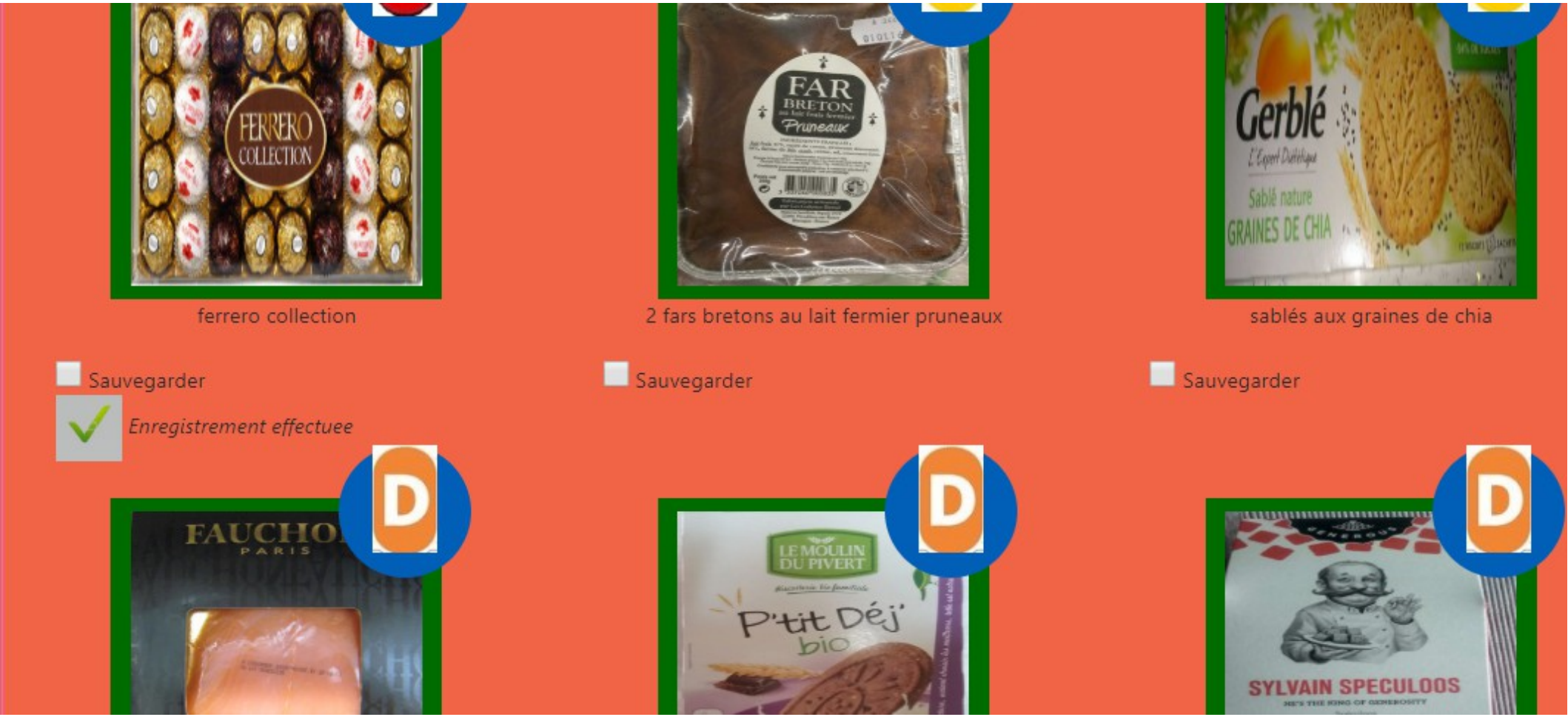
```
</div>
<div id="nomAliment2">
  {{bb}}
</div>
<br>
{% if user.is_authenticated %}
<input type="checkbox" style="width: 20px; height: 20px;"
id="product2" onclick='pushlist("product2", "user", "stock", "is_save2")' name="is_save2" value="{{bb}}" /> Sauvegarder
<div id="is_save2"></div>
{% endif %}
```

5c) la recherche – Javascript

recherche.js

```
$("#product1,#product2,#product3,#product4,#product5,#product6").on("click", function(e){
   e.preventDefault();

    $.ajax({
        data:{
            'data[]':LISTE[LISTE.length - 1],
            'username':USER[USER.length - 1],

        },
        type:"POST",
        url:"/mes_aliments/recherche/"
    })
    .done(function(data){
        if (data.error){
            $("#monCadreAlert").text(data.error);
            $("#is_save");


        }
        else{
            $("#is_save").html(data.data);
            $("#monCadreAlert");


        };

    });

});
```

5d) la recherche – simulation web
l'utilisateur sauvegarde des aliments



5e) la recherche – Javascript
recherche.js

```
function pushlist(id_product, user, stock, is_save){

  var a = document.getElementById(id_product).value;
  var b = document.getElementById(id_product).name;
  var c = document.getElementById(user).innerHTML;
  USER.push(c)
  LISTE.push(a)
  LISTE_NAME.push(b)

  d = document.getElementById(stock).innerHTML;
  console.log(d)

  if (d == "<center><strong>oups vous avez trop d'aliment en stock supprime en ! ou remplace le !</strong></center>"){
      console.log("trop d\'aliment pour ce compte")
      document.getElementById(is_save).innerHTML = "";
      document.getElementById(is_save).innerHTML = '<img style="width:15%;" src="/static/img/portfolio/recherche/croix.jpg" />'
  +'  <i>Vous avez trop d\'aliment</i>';
  }
  else{
      document.getElementById(is_save).innerHTML = "";
      document.getElementById(is_save).innerHTML = '<img style="width:15%;" src="/static/img/portfolio/recherche/validate.jpg" />'
  +'  <i>Enregistrement effectuee</i>';


  };
  };
```

5f) la recherche – la fonction search()

views.py

```
if request.method == "POST":

    search = request.POST.get('cool')
    username = request.POST.get('username')
    validate = request.POST.getlist('data[]')

    try:
        stock = controle_data_food(current_user)
    except:
        pass

    if validate and username:

        current_user = request.user
        stock = controle_data_food(username)

        if stock[1] == True:
            veri = verification_product_not_two(current_user,
                                                validate[0])

            if veri == True:
                insert_food(username, validate[0])

        elif stock[1] == False:
            exceeded_stock = "oups vous avez trop d'aliment en stock supprim

    if search:

        current_user = request.user

        if current_user.is_authenticated:
            stock = controle_data_food(current_user)

            if stock[1] == False:
                exceeded_stock = "oups vous avez trop d'aliment en stock sup

        image = image_food(search)
        title = title_food(search)
```

5g) la recherche – les fonctions associées

controle_data_food()

```
def controle_data_food(username):
    """Here we watch if user have 6 products,
    if he has -6 we ask him to select products
    else we warned him to modify his selection"""

    c = foodAccount.objects.get(name=username)

    liste = [c.name_aliment1, c.name_aliment2, c.name_aliment3,
             c.name_aliment4, c.name_aliment5, c.name_aliment6,]

    number = 0
    for i in liste:
        if i != "":
            number += 1

    if number >= 6:
        return "nombre de produit suppérieur a 6", False

    else:
        return "stockage du produit possible", True
```

5g) la recherche – les fonctions associées
verification_product_not_two()

```
def verification_product_not_two(username, product):
    """here we check that the food is not already present"""

    c = foodAccount.objects.get(name=username)
    if c.name_aliment1 == product or c.name_aliment2  == product or\
       c.name_aliment3  == product or c.name_aliment4  == product or\
       c.name_aliment5  == product or c.name_aliment1  == product:
        return False
    else:
        return True
```

5g) la recherche – les fonctions associées

insert_food()

```
def insert_food(username, food_name):
    """He we insert food"""

    c = foodAccount.objects.get(name=username)

    if c.name_aliment1 == "":
        c.name_aliment1 = food_name
        c.save()

    elif c.name_aliment2 == "":
        c.name_aliment2 = food_name
        c.save()

    elif c.name_aliment3 == "":
        c.name_aliment3 = food_name
        c.save()

    elif c.name_aliment4 == "":
        c.name_aliment4 = food_name
        c.save()

    elif c.name_aliment5 == "":
        c.name_aliment5 = food_name
        c.save()

    elif c.name_aliment6 == "":
        c.name_aliment6 = food_name
        c.save()

    else:
        print("to much food")
```

Voir ses aliments sans en avoir sauvegarder 6

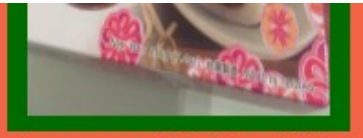
- Simulation web

L'utilisateur cherche un aliment avec 6 aliments en stock

L'utilisateur cherche un aliment avec 6 aliments en stock

céréactive

Sauvegarder



japanese mochi "red bean" 210g

Sauvegarder

céréactive

Sauvegarder

Vous avez trop d'aliment

chocolat noir excellence 99% de cacao

Sauvegarder

biscuits ptit déj bio

Sauvegarder

madeleines longues aux œufs

Sauvegarder

6 - Mes aliments

a) simulation web

b) la fonction my_food

→ *views.py*

c) les fonctions associées

→ *my_food_user()*

→ *display_food()*

6a) mes aliments – simulation web

l'utilisateur veut voir ses aliments

6a) mes aliments – simulation web

l'utilisateur veut voir ses aliments

E

ferrero collection

C

2 fars bretons au lait fermier pruneaux

C

sablés aux graines de chia

D

sylvain spéculoos -generou

D

biscuits ptit dej bio

D

fauchon

6b) mes aliments – la fonction my_food()

views.py

```
def my_food(request):
    """Here we acced to user product"""

    current_user = request.user

    try:
        food = my_food_user(request.user.username)
        a = display_food(food)

        return render(request, 'mes_aliments.html',{"a":str(a[0][4]),
                                                    "b":str(a[1][4]),
                                                    "c":str(a[2][4]),
                                                    "d":str(a[3][4]),
                                                    "e":str(a[4][4]),
                                                    "f":str(a[5][4]),

                                                    "aa":str(a[0][0]),
                                                    "bb":str(a[1][0]),
                                                    "cc":str(a[2][0]),
                                                    "dd":str(a[3][0]),
                                                    "ee":str(a[4][0]),
                                                    "ff":str(a[5][0]),

                                                    "aaaa":"/static/img/portfolio/nutriscore/" + str(a[0][3])
                                                    "bbbb":"/static/img/portfolio/nutriscore/" + str(a[1][3])
                                                    "cccc":"/static/img/portfolio/nutriscore/" + str(a[2][3])
                                                    "dddd":"/static/img/portfolio/nutriscore/" + str(a[3][3])
                                                    "eeee":"/static/img/portfolio/nutriscore/" + str(a[4][3])
                                                    "ffff":"/static/img/portfolio/nutriscore/" + str(a[5][3])

                                                    "aaaaa":str(a[0][0]),
                                                    "bbbbb":str(a[1][0]),
                                                    "ccccc":str(a[2][0]),
                                                    "ddddd":str(a[3][0]),
                                                    "eeeee":str(a[4][0]),
                                                    "fffff":str(a[5][0]),
                                                     })
    except:
        message = "Veuillez remplir votre selection d'aliment de 6 produit svp =) "
        return render(request, 'error.html', {"message":message})
```

6c) mes aliments – les fonctions associées
my_food_user()

```
def my_food_user(username):
    """Here we take user food"""

    liste = []
    c = foodAccount.objects.get(name=username)
    food = [c.name_aliment1, c.name_aliment2, c.name_aliment3,
            c.name_aliment4, c.name_aliment5, c.name_aliment6]

    for i in food:
        liste.append(i)

    return food
```

6c) mes aliments – les fonctions associées
display_food()

```
def display_food(food_list):
    """Here we take informations food for template"""

    liste_ali = []
    try:
        for i in food_list:
             z = aliment.objects.get(name_aliment=i)
             liste = []
             liste = [z.name_aliment, z.code_product_food,
                      z.description, z.nutriscore,
                      z.image, z.name_store, z.name_brand]

             liste_ali.append(liste)


        return liste_ali

    except:
        pass
```

7 - La description d'un produit

a) simuation web

b) Javascript

→ *choice()*

c) nanrien()

d) fonction food_details

→ *views.py*

e) fonction associées

→ *food_details*

L'utilisateur veut voir la description d'un produit

E

ferrero collection

C

2 fars bretons au lait fermier pruneaux

C

sablés aux graines de chia

D

sylvain spéculoos -generou

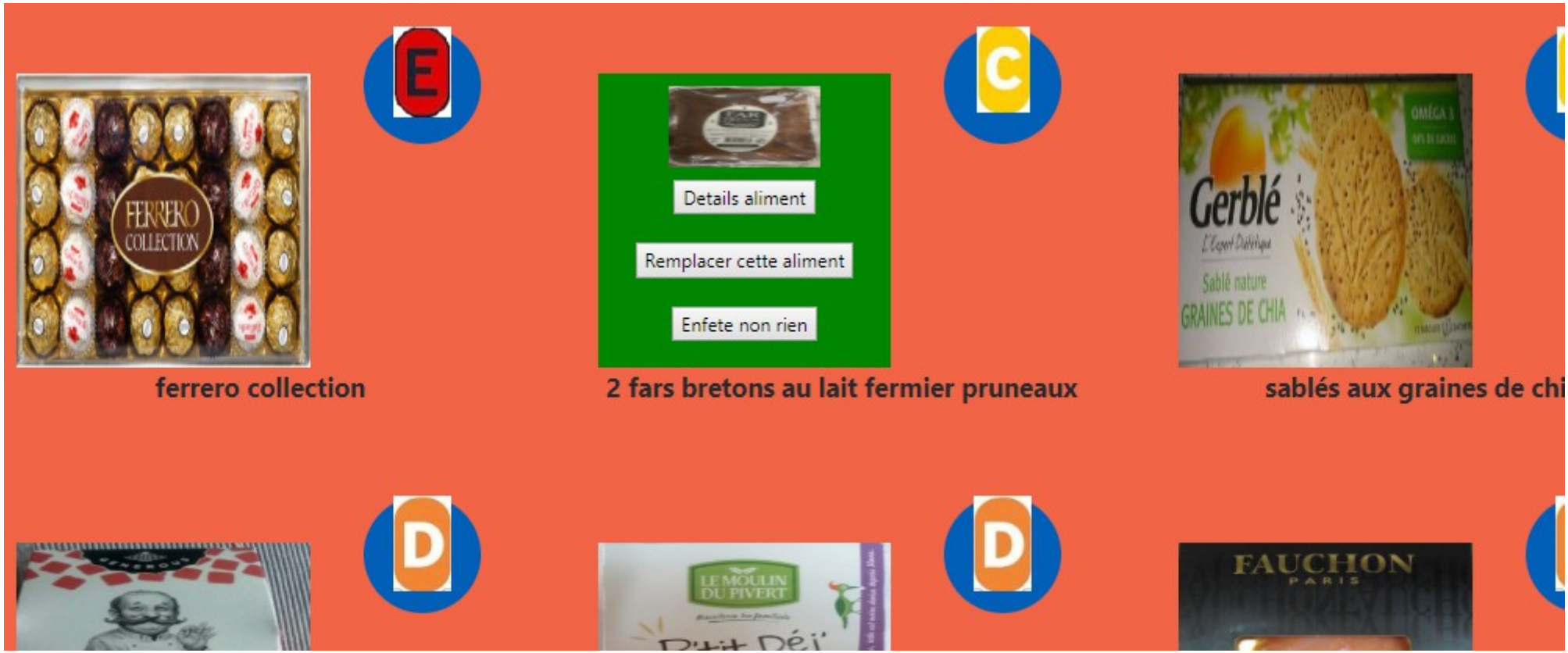
D

biscuits ptit dej bio

D

fauchon

L'utilisateur veut voir la description d'un produit



Les 3 choix

```
function choice(im, image, detail, remplacement, nan, gabarit){
    var a = "<center><img src=" + gabarit + " style='padding:10px;width:60%;height:90px;'></center>"
    document.getElementById(im).style.display = "none";
    document.getElementById(image).innerHTML = a
    document.getElementById(detail).innerHTML = "<center><input type='submit'" +
                                                "style='color:black;' value='Détails aliment' />" +
                                                "</center><br>";
    document.getElementById(remplacement).innerHTML = "<center><input type='submit' style='color:black;" +
                                                    "'value='Remplacer cette aliment' /></center><br>";


    var a =  "nanrien(" + "'" + im + "'" + ",'" + image + "','" + detail + "','" + remplacement + "','" + nan + "')" + '"'

   document.getElementById(nan).innerHTML = "<center><input type='button' onclick=" + '"' + a +
                                            "style='color:black;' value='Enfete non rien'/></center>";


};
```

nanrien()

E

ferrero collection

C

2 fars bretons au lait fermier pruneaux

C

sablés aux graines de chia

D

sylvain spéculoos -generou

D

biscuits ptit dej bio

D

fauchon

nanrien()

```
function nanrien(im, image, detail, remplacement, nan){

    document.getElementById(im).style.display = "block";
    document.getElementById(image).innerHTML = "";
    document.getElementById(detail).innerHTML = "";
    document.getElementById(remplacement).innerHTML = "";
    document.getElementById(nan).innerHTML = "";
    CHOISEN = []

    };
```

L'utilisateur veut voir la description d'un produit

View.py

```
def food_det(request):
    """This is function for food details"""

    if request.method == "POST":

        search = request.POST.get('produit')
        details = food_details(search)
        url_nutri = details.nutriscore
        food = details.name_aliment
        code = details.code_product_food
        image = details.image
        url_nutri = "/static/img/portfolio/nutriscore/" + str(url_nutri) + ".jpg >"

        return render(request, 'aliment_det.html',{'detail':details,
                                                   'url_nutri':url_nutri,
                                                   'code':code,
                                                   'image':image,
                                                   'food': food
                                                   })
    return render(request, 'aliment_det.html')
```

food_details()

```
def food_details(value):
    """Here we calling informations about product. Thank to that we
    can redirect to Openfactfood"""

    details = aliment.objects.get(image='{}'.format(value))

    return details
```

8 – remplacer un produit

a) simulation web

b) la fonction replacing

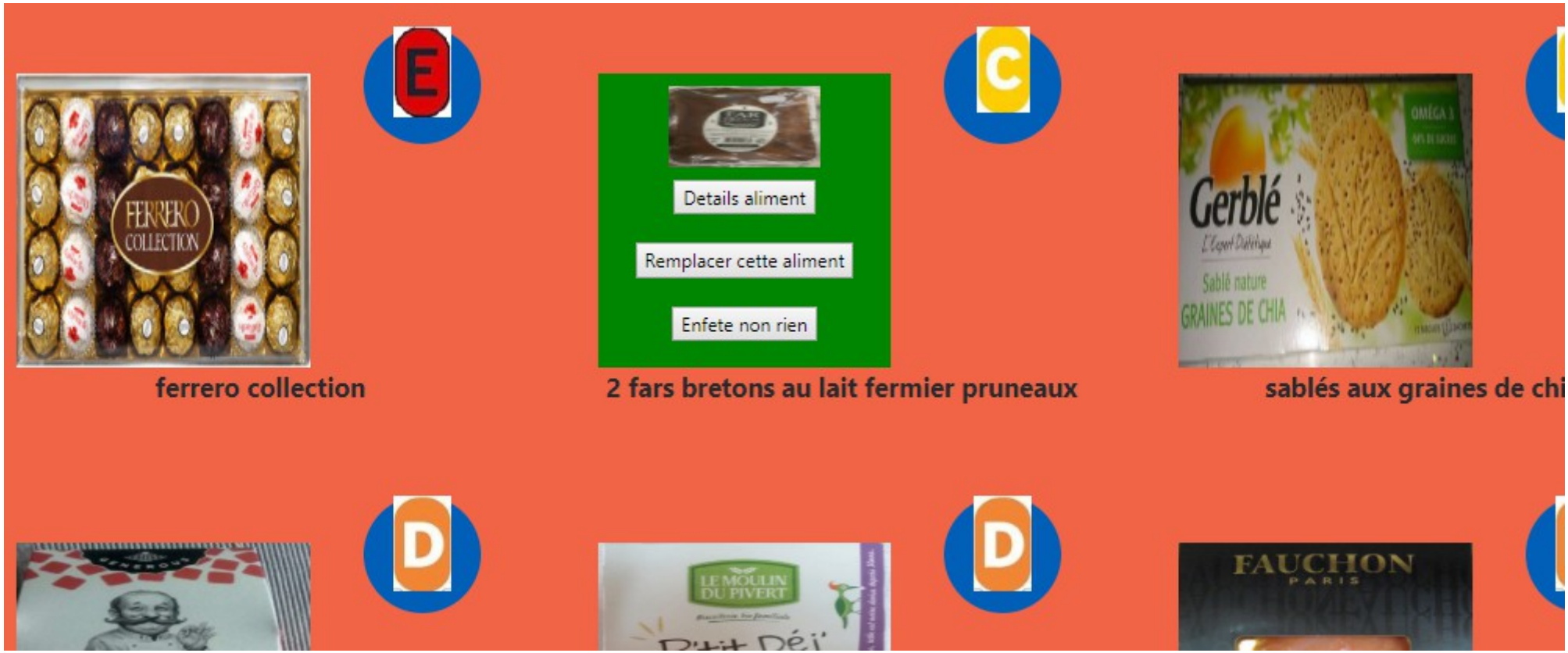
→ *views.py*

c) les fonctions associées

→ verification_replacement()

→ *data_replace()*

L'utilisateur veut remplacer un produit



Views.py

```
def replacing(request):
    """This is functionality for replace food from my food"""


    message = ''

    if request.method == "POST":
        replace_it = request.POST.getlist('remplace_food')
        if replace_it:
            current_user = request.user

            liste = [[],[]]
            element = []
            c=0
            for i in replace_it:
                for j in i:
                    if j == ",":
                        c+=1
                    else:
                        liste[c].append(j)
                c+=1
            for i in liste:
                i = "".join(i)
                element.append(i)

            b = verification_replacement(current_user, "".join(liste[-1]))

            if b == True:
                data_replace(request, current_user,
                             element[0], element[1])
            elif b == False:
                message = 'vous avez deja cet aliment'
```

Views.py

```
else:
    food = request.POST.get('rem')
    current_user = request.user
    image = image_food(food)
    title = title_food(food)
    a = replace(food)

    return render(request, 'remplacement.html', {"a":str(a[0][3]),
                                                 "b":str(a[1][3]),
                                                 "c":str(a[2][3]),
                                                 "d":str(a[3][3]),
                                                 "e":str(a[4][3]),
                                                 "f":str(a[5][3]),

                                                 "aa":str(a[0][0]),
                                                 "bb":str(a[1][0]),
                                                 "cc":str(a[2][0]),
                                                 "dd":str(a[3][0]),
                                                 "ee":str(a[4][0]),
                                                 "ff":str(a[5][0]),

                                                 "aaa":str(a[0][3]),
                                                 "bbb":str(a[1][3]),
                                                 "ccc":str(a[2][3]),
                                                 "ddd":str(a[3][3]),
                                                 "eee":str(a[4][3]),
                                                 "fff":str(a[5][3]),

                                                 "aaaa":"/static/img/portfolio/nutriscore/" + str(
                                                 "bbbb":"/static/img/portfolio/nutriscore/" + str(
                                                 "cccc":"/static/img/portfolio/nutriscore/" + str(
                                                 "dddd":"/static/img/portfolio/nutriscore/" + str(
                                                 "eeee":"/static/img/portfolio/nutriscore/" + str(
                                                 "ffff":"/static/img/portfolio/nutriscore/" + str(

                                                 "image":str(image),
                                                 "titre":str(title),
                                                 'message':message
```

Views.py

```
current_user = request.user
try:
    food = my_food_user(request.user.username)
    a = display_food(food)

    return render(request, 'mes_aliments.html', {"a":str(a[0][4]),
                                                 "b":str(a[1][4]),
                                                 "c":str(a[2][4]),
                                                 "d":str(a[3][4]),
                                                 "e":str(a[4][4]),
                                                 "f":str(a[5][4]),

                                                 "aa":str(a[0][0]),
                                                 "bb":str(a[1][0]),
                                                 "cc":str(a[2][0]),
                                                 "dd":str(a[3][0]),
                                                 "ee":str(a[4][0]),
                                                 "ff":str(a[5][0]),

                                                 "aaaa":"/static/img/portfolio/nutriscore/" + str(a[0][
                                                 "bbbb":"/static/img/portfolio/nutriscore/" + str(a[1][
                                                 "cccc":"/static/img/portfolio/nutriscore/" + str(a[2][
                                                 "dddd":"/static/img/portfolio/nutriscore/" + str(a[3][
                                                 "eeee":"/static/img/portfolio/nutriscore/" + str(a[4][
                                                 "ffff":"/static/img/portfolio/nutriscore/" + str(a[5][

                                                 "aaaaa":str(a[0][0]),
                                                 "bbbbb":str(a[1][0]),
                                                 "ccccc":str(a[2][0]),
                                                 "ddddd":str(a[3][0]),
                                                 "eeeee":str(a[4][0]),
                                                 "fffff":str(a[5][0]),
                                                 'message':message

                                                 })
except:
    return render(request, 'mes_aliments.html')
```

Views.py

```
def replacing(request):
    """This is functionality for replace food from my food"""


    message = ''

    if request.method == "POST":
        replace_it = request.POST.getlist('remplace_food')
        if replace_it:
            current_user = request.user

            liste = [[],[]]
            element = []
            c=0
            for i in replace_it:
                for j in i:
                    if j == ",":
                        c+=1
                    else:
                        liste[c].append(j)
                c+=1
            for i in liste:
                i = "".join(i)
                element.append(i)

            b = verification_replacement(current_user, "".join(liste[-1]))

            if b == True:
                data_replace(request, current_user,
                             element[0], element[1])
            elif b == False:
                message = 'vous avez deja cet aliment'
```

Verification_replacement()

```
def verification_replacement(username, product):
    """We verify food isnt already present"""

    c = foodAccount.objects.get(name=username)

    food = [c.name_aliment1, c.name_aliment2, c.name_aliment3,
            c.name_aliment4, c.name_aliment5, c.name_aliment6]


    for i in food:
        if product == i:
            return False

    return True
```

data_replace()

```
def data_replace(request, username, product, new_product):
    """ """

    c = foodAccount.objects.get(name=username)


    food = [c.name_aliment1, c.name_aliment2, c.name_aliment3,
            c.name_aliment4, c.name_aliment5, c.name_aliment6]


    if c.name_aliment1 == product:
        c.name_aliment1 = new_product
        c.save()

    elif c.name_aliment2 == product:
        c.name_aliment2 = new_product
        c.save()

    elif c.name_aliment3 == product:
        c.name_aliment3 = new_product
        c.save()

    elif c.name_aliment4 == product:
        c.name_aliment4 = new_product
        c.save()

    elif c.name_aliment5 == product:
        c.name_aliment5 = new_product
        c.save()

    elif c.name_aliment6 == product:
        c.name_aliment6 = new_product
        c.save()
```

9 -Construction de database

a) insertion de donnée dans category

b) insertion de donnée dans food

c) nettoyage de donnée

→ *charactere()*

→ *data_no_food()*

→ *delete_virgula()*

→ *visualisation()*

Construction de la database category

```
class data:
    """Here we insert data into tables"""

    def categorie(self):
        """Here we insert category"""

        conn = psycopg2.connect(database=DATABASE,
                                user=USER,
                                host=HOST,
                                password=PASSWORD)

        cursor = conn.cursor()
        self.liste = []
        path = "https://fr.openfoodfacts.org/categories"
        requete = requests.get(path)
        page = requete.content
        soup = BeautifulSoup(page, "html.parser")

        for tag in soup.find_all("td"):
            self.liste.append(tag.text)

        c = 0
        for i in range(3):
            print(self.liste[c])
            self.liste[c] = self.liste[c].replace(" ", "_")
            self.liste[c]= self.liste[c].replace("'", "")

            cursor.execute("INSERT INTO mes_aliments_categorie(name_categorie) \
            conn.commit()
            c+=3
        print("categorie faites")
```

Construction de la database insert_food

```
def insert_food(self):
    """Here we run api and we take informations necessary for tables insert

    conn = psycopg2.connect(database=DATABASE,
                            user=USER,
                            host=HOST,
                            password=PASSWORD)

    cursor = conn.cursor()


    c = 0
    d = 1
    self.liste_store = []
    self.liste_storel = [[], [], [], [], [], [], [], [],
                         [], [], [], [], [], [], [], [],
                         [], [], [], [], [], [], [], []]

    self.liste_brands = []
    self.liste_brandsl = [[], [], [], [], [], [], [], [],
                          [], [], [], [], [], [], [], [],
                          [], [], [], [], [], [], [], []]


    self.liste2 = []

    for i in range(3):

        print(self.liste[c])
        print("\n")

        path2 = "https://fr.openfoodfacts.org/categorie/{}".format(self.lis
        requete = requests.get(path2)
        page = requete.content
        soup = BeautifulSoup(page, "html.parser")
        for tag in soup.find_all("span"):
            self.liste2.append(tag.text)
```

Construction de la database insert_food

```
for i in range(3):

    print(self.liste[c])
    print("\n")

    path2 = "https://fr.openfoodfacts.org/categorie/{}".format(self.list
    requete = requests.get(path2)
    page = requete.content
    soup = BeautifulSoup(page, "html.parser")
    for tag in soup.find_all("span"):
        self.liste2.append(tag.text)

    for i in self.liste2[6:]:
        print(i)
        a = str(i).find(" - ")
        i = i[0:a]
        print(i)
        path3 = "https://fr.openfoodfacts.org/cgi/search.pl?search_terms
        search = path3.format(i)
        r = requests.get(search)
        data = json.loads(r.text)

        try:
            self.number_product = data["products"][0]['code']
        except:
            self.number_product = "No_found"
        try:
            self.description_product = data["products"][0]["ingredients_
        except:
            self.description_product = "No_found"

        try:
            self.nutriscore = data["products"][0]["nutrition_grades"]
            print("nutriscore=",self.nutriscore)
        except:
            self.nutriscore = "No_found"
```

Construction de la database insert_food

```
try:
    self.image = data["products"][0]["image_front_url"]
except:
    self.image = "No_found"


try:
    self.brandss = data["products"][0]["brands"]
    if self.brandss == '':
        self.liste_brands.append(self.brandss)
    else:
        self.liste_brands.append(self.brandss)
except:
    self.brandss = "No_found"
    self.liste_brands.append("No_found")

try:
    self.store_product = data["products"][0]["stores"]
    if self.store_product == '':
        self.liste_store.append("No_found")
    else:
        self.liste_store.append(self.store_product)
except:
    self.store_product = "No_found"
    self.liste_store.append("No_found")

i = i.replace("'", "")
print(i)
self.image = self.image.replace("'", "")
print(self.number_product)

self.description_product = self.description_product.replace("'

print(self.description_product)
self.nutriscore = self.nutriscore.replace("'", "")

print(self.nutriscore)
self.nutriscore = self.nutriscore.replace("'", "")
```

Construction de la database insert_food

```
                print(self.nutriscore)
                self.nutriscore = self.nutriscore.replace("'", "")

                print(self.image)
                self.image = self.image.replace("'", "")

                print(d)

                print(self.store_product)
                self.store_product = self.store_product.replace("'", "")

                print(self.brandss)
                self.brandss = self.brandss.replace("'", "")

                cursor.execute("INSERT INTO mes_aliments_aliment (name_aliment,\
                               image, code_product_food, description, nutriscor
                               id_categorie_id, name_store, name_brand)\
                               VALUES('{0}','{1}','{2}','{3}','{4}',{5},'{6}','
                              .format(i.lower(), self.image.lower(),
                                      self.number_product.lower(),
                                      self.description_product.lower(),
                                      self.nutriscore.lower(),
                                      d, self.store_product.lower(),self.brands

                conn.commit()

            d+=1
            c+=3
            self.liste2 = []


if __name__ == "__main__":

    yo = data()
    yo.categorie()
    yo.insert_food()
```

Construction de la database nettoyage virgule

```
import psycopg2
from config import DATABASE, USER, HOST, PASSWORD



def del_vir():
    """Here we search food picture """


    conn = psycopg2.connect(database=DATABASE,
                            user=USER,
                            host=HOST,
                            password=PASSWORD)
    cur = conn.cursor()

    cur.execute("""UPDATE mes_aliments_aliment
                set name_aliment = REPLACE(name_aliment, ',', '');""")

    conn.commit()
    cur.execute("""select * from mes_aliments_aliment""")

    rows = cur.fetchall()
    print(rows)

del_vir()
```

Construction de la database nettoyage charactère

```
def replace():
    """Here we search food picture """


    conn = psycopg2.connect(database=DATABASE,
                            user=USER,
                            host=HOST,
                            password=PASSWORD)
    cur = conn.cursor()

    cur.execute("""UPDATE mes_aliments_aliment
                set name_aliment = REPLACE(name_aliment, '\xa0', '');""")

    conn.commit()



replace()
```

Construction de la database nettoyage donnée

```
def dela():
    """Here we search food picture """


    conn = psycopg2.connect(database=DATABASE,
                            user=USER,
                            host=HOST,
                            password=PASSWORD)
    cur = conn.cursor()

    cur.execute("""DELETE FROM mes_aliments_aliment
                WHERE nutriscore = 'no_found' or
                image = 'no_found' or name_aliment='no_found'
                or code_product_food = 'no_found'""")

    conn.commit()
    cur.execute("""select * from mes_aliments_aliment""")

    rows = cur.fetchall()
    print(rows)

dela()
```

Visualisation database

```
import psycopg2

def visualisation():
    """Here we search food picture """


    conn = psycopg2.connect(database='deed0vc5ehln6g',
                            user='hylgxgwfjacgtj',
                            host='ec2-54-225-95-183.compute-1.amazonaws.com',
                            password='2509ffb8ec855b2a37e2d1b80e0521d942219ad4d5
    cur = conn.cursor()



    cur.execute("""select name_aliment from mes_aliments_aliment""")
    conn.commit()
    rows = cur.fetchall()
    print(rows)

visualisation()
```

test

- Direction le document de test

  → prend en compte les fichier tests.py de chaques applications.

  → ils testent le renvoie d'un statut 200 pour chaque template, le bon fonctionnement de chaque méthode que nous avons vu précédemment et le test de la requête vers l'api d'Openfactfood.