



OPEN FOOD

Pur Beurre

bienvenue,
DU GRAS
QU

Trouvez un produit de subs

Rechercher un produit

Cole

Pur Beurre est l'initiative de c
Colette Tatou et Rémy, que vo
notre excell

SOMMAIRE

CONTEXT	3
TRELLO	4
MODELS	5
DJANGO users_account	6
FBC-->CBV	7
product API and DataBase.....	8
test Mock	9
templates	10
gestion des produits --> django	11
--> html	12
tests --> test_views	13
gestion des substituts --> html / CSS.....	14
--> django.....	15
COUVERTURE --> tests launch	16
--> report && html	17
HEROKU --> settings	18

Context

Pur Beurre aims to establish a platform for health-conscious individuals who seek high-quality products.

Our goal is to provide products that match the user's search criteria and have a superior nutriscore rating (A or B) in case they are looking for crisps or any other product.

Once the user is satisfied with the proposal, they can log in to their account and save it to their favourites.

Additionally, they can remove any product from their favourites if it no longer suits their preferences.



Nutella Fans

Trello

<https://trello.com/b/iMvWuBbe/p8-creez-une-plateforme-pour-amateurs-de-nutella>

Pur Beurre [P8] Creez une plateforme pour amateurs de Nutella +

https://trello.com/b/iMvWuBbe/p8-creez-une-plateforme-pour-amateurs-de-nutella

Trello

Ce tableau est défini sur public. Vous pouvez modifier sa visibilité à tout moment. [En savoir plus ici](#)

Tableau [P8] Creez une plateforme pour amateurs de Nutella Public Inviter Automatisation Afficher le menu

Users Stories

A faire

En cours

Terminer

Création du formulaire création du compte

Description

Création du compte utilisateur

Activité

Commentaires

Actions

Automatisation

Membres

Étiquettes

Checklist

Dates

Pièce jointe

Image de couverture

Champs personnels

POWER-UPS

Ajouter des Pow...

Autres

Créer un bouton

Suivre

Créer un modèle

Archiver

Partager

Initialiser un repo

Créer un projet Django

Mise en forme -> graphisme

Test intégration, unitaires

Création du formulaire création du compte

Requête Openfoodfacts

Enregistrement des aliments

Mise en ligne Heroku

Docstrings

Ajouter une carte

← 4

Nutella Fans

Projet App

Folders and app tree

The screenshot shows a terminal window titled "untitled — nutella_fans". The left pane displays the "OPEN FILES" and "FOLDERS" structure of the project:

- FOLDERS**
 - nutella_fans
 - .pytest_cache
 - htmlcov
 - myvenv
 - nutella_fans
 - __pycache__
 - base
 - product
 - save_substitute
 - settings
 - staticfiles
 - users_account
 - /* __init__.py
 - /* asgi.py
 - /* urls.py
 - /* wsgi.py
- .coverage
- .coveragerc
- .gitignore
- .travis.yml
- db.sqlite3
- geckodriver.log
- manage.py
- Procfile
- README.md
- requirements.txt

The right pane of the terminal window is currently empty, showing only the number "1".

Nutella Fans

Models

```
from django.db import models

class Brand(models.Model):
    name = models.CharField(max_length=200)
    def __str__(self):
        return self.name

class Category(models.Model):
    name = models.CharField(max_length=200)
    def __str__(self):
        return self.name

class Store(models.Model):
    name = models.CharField(max_length=200)
    def __str__(self):
        return self.name

class Product(models.Model):
    name = models.CharField(max_length=200)
    nutriscore = models.CharField(max_length=1, null=True)
    nova = models.IntegerField(null=True)
    url = models.URLField(max_length=200)
    barcode = models.CharField(max_length=200, unique=True)
    description = models.TextField(null=True)
    picture = models.URLField(null=True)
    ...
    brand = models.ForeignKey(
        Brand, on_delete=models.CASCADE)
    categories = models.ManyToManyField(Category)
    stores = models.ManyToManyField(Store)

    def get_substitutes(self):
        product_categories = self.categories.all()
        result = Product.objects.filter(categories__in=product_categories) \
            .filter(nutriscore__lt=self.nutriscore) \
            .exclude(pk=self.pk) \
            .order_by('nutriscore').distinct()
        return result
```

The screenshot shows a database schema browser interface. On the left, there is a tree view of tables and schemas. On the right, there is a list of tables with their names and descriptions.

Tables:

- auth_group
- auth_group_permissions
- auth_permission
- django_admin_log
- django_content_type
- django_migrations
- django_session
- product_brand
- product_category
- product_product
- product_product_categories
- product_product_stores
- product_store
- save_substitute_substitute
- users_account_user
- users_account_user_groups
- users_account_user_substitutes
- users_account_user_user_permissions

Schemas:

- pg_catalog
- information_schema

Search Bar: Search

Buttons: + Table, + View, + Materialized View

Page Number: ← 6

users_account --> template

registration and login page

The screenshot shows a web browser window titled "Pur Beurre" with the URL "127.0.0.1:8000/account/login/". The page features a background image of yellow flowers and a bowl of butter. A large white text "Connectez-vous à votre profil !" is centered. Below it, a sub-header "Connectez vous à votre compte Pur Beurre !" is followed by two input fields: "Nom d'utilisateur :" and "Mot de passe :". A red "se connecter" button is positioned between them. Below the buttons, there are two links: "Vous n'avez pas de compte ?" and "Incrivez-vous !". The top right corner of the browser window has a "DJD" logo.

The screenshot shows a web browser window titled "Création de votre compte Pur Beurre" with the URL "127.0.0.1:8000/account/signup/". The background image is the same as the login page. The main heading "Création de votre compte Pur Beurre" is at the top. Below it, a sub-header "Inscrivez-vous au site Pur Beurre pour enregistrer vos favoris" is followed by three input fields: "Nom d'utilisateur :", "Email :", and "Mot de passe :". To the right of the "Nom d'utilisateur :" field is a note: "Requis. 150 caractères maximum. Uniquement des lettres, nombres et les caractères < @, < ., < +, < - et < >.". Below the input fields is a list of four bullet points: "Votre mot de passe ne peut pas trop ressembler à vos autres informations personnelles.", "Votre mot de passe doit contenir au minimum 8 caractères.", "Votre mot de passe ne peut pas être un mot de passe couramment utilisé.", and "Votre mot de passe ne peut pas être entièrement numérique.". Below the input fields is a "Confirmation du mot de passe" field with a note: "Saisissez le même mot de passe que précédemment, pour vérification." A red "créer mon compte" button is located below the confirmation field. At the bottom, there are two links: "Vous avez déjà un compte ?" and "Connectez-vous!".

Nutella Fans

FBV --> CBV

Function-Based Views -->

Class-Based View

-> log in scripts FBV

 urls.py
 views.py

request.POST['username'] --> accès à des dictionnaires envoyés par leurs clés (par le formulaire)

-> scripts FBV

 urls.py
 views.py

Class generic LoginView

The screenshot shows the source code for the generic.LoginView class. It includes sections for Descendants, Attributes, and Methods. The Descendants section lists various generic view classes like BaseDetailView, BaseCreateView, and BaseUpdateView. The Attributes section shows the template_name attribute. The Methods section contains the __init__, _allowed_methods, as_view, dispatch, http_method_not_allowed, options, and setup methods.

```
class View:
    from django.views.generic import View
    Intentionally simple parent class for all views. Only implements dispatch-by-method and simple sanity checking.
    Descendants
    Attributes
        template_name (str): template location
    Methods
        def _allowed_methods(self):
        def as_view(cls, **kwargs):
        def dispatch(self, request, *args, **kwargs):
        def http_method_not_allowed(self, request, *args, **kwargs):
        def __init__(self, *args, **kwargs):
        def options(self, request, *args, **kwargs):
        def setup(self, request, *args, **kwargs):
```

```
def login_request(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(request,
                            username=username, password=password)
        if user is not None:
            login(request, user)
            messages.info(request, f'You are now logged in as {username}.')
            return render(request, 'users_account/home.html')
        else:
            messages.error(request, 'Invalid username or password.')
            return render(request, 'users_account/registration/login.html')
```

Script Views pour le login --> FBV

```
urlpatterns = [
    path('', views.base, name='base'),
    path('sign_up/', views.sign_up, name='sign_up'),
    path('login/', views.login_request, name='login_request'),]
```

Script urls

class LoginView(View):

 '''Clas Based View for user's login

Attributes:

 template_name (str): template location

<<<

 template_name = «registration/login.html»

Script Views pour le login --> CBV

```
urlpatterns = [
    path('sign_up/', SignupView.as_view(), name='sign_up'),
    path('login/', LoginView.as_view(), name='login'),
    path('logout/', views.logout_request, name='logout'),
    path('profile/<int:pk>', UserDetailView.as_view(), name='profile'),]
```

Script urls

product --> API and DataBase

```

def handle(self, *args, **options):
    category_list = self.get_category()
    for category_dict in category_list:
        products = self.get_products(category_dict)
        for product in products:
            p = self.create_product(product)
            if not p:
                continue
            else:
                category_names = product.get(
                    «categories»).lower().split(«,»)
                for category in category_names:
                    c, created = Category.objects.get_or_create(
                        name=category)
                    p.categories.add(c)
    if product.get(«stores»):
        stores = product.get(«stores»).lower().split(«,»)
        for store in stores:
            s, created = Store.objects.get_or_create(
                name=store)
            p.stores.add(s)

```

def get_category(self):

«»» response of the request to extract data from API

Returns:

LIST: list of categories

```

«»»
response = requests.get(«https://fr.openfoodfacts.org/categories.json»)
result_category = response.json()
data_category = result_category.get('tags')
categories = [data.get('name') for data in data_category
              if data.get('name')]
category_name = categories[0:10]
return category_name

```

The screenshot shows a JSON viewer interface with the URL <https://us.openfoodfacts.org/categories.json>. The 'tags' section is expanded, showing 12 categories (0 to 11) with their details. Category 0 is 'en:snacks' (id: en:snacks, known: 1, name: "Snacks", products: 71173, url: <https://us.openfoodfacts.org/category/snacks>). Category 1 is 'en:plant-based-foods-and-beverages' (id: en:plant-based-foods-and-beverages, known: 1, name: "Plant-based foods and beverages", products: 70402, url: <https://us.openfoodfacts.org/category/plant-based-foods-and-beverages>). Other categories listed include 'aliments et boissons à base de végétaux', 'aliments d'origine végétale', 'céréales et pommes de terre', 'petit-déjeuners', 'céréales et dérivés', 'céréales pour petit-déjeuner', 'mueslis', 'céréales aux fruits', 'mueslis aux fruits', 'plant-based foods and beverages', 'plant-based foods', 'cereals and potatoes', 'breads', 'sliced breads', 'sliced breads without crust', 'wholemeal breads', 'wholemeal sliced breads', 'alimentos y bebidas de origen vegetal', 'alimentos de origen vegetal', 'cereales y patatas', 'desayunos', and 'cereales y derivados'.

The screenshot shows a PostgreSQL database table named 'product_category'. The table has two columns: 'id' and 'name'. The data consists of 22 rows, each representing a food category. The categories are: 1. aliments et boissons à base de végétaux, 2. aliments d'origine végétale, 3. céréales et pommes de terre, 4. petit-déjeuners, 5. céréales et dérivés, 6. céréales pour petit-déjeuner, 7. mueslis, 8. céréales aux fruits, 9. mueslis aux fruits, 10. plant-based foods and beverages, 11. plant-based foods, 12. cereals and potatoes, 13. breads, 14. sliced breads, 15. sliced breads without crust, 16. wholemeal breads, 17. wholemeal sliced breads, 18. alimentos y bebidas de origen vegetal, 19. alimentos de origen vegetal, 20. cereales y patatas, 21. desayunos, and 22. cereales y derivados.

	id	name
1	1	aliments et boissons à base de végétaux
2	2	aliments d'origine végétale
3	3	céréales et pommes de terre
4	4	petit-déjeuners
5	5	céréales et dérivés
6	6	céréales pour petit-déjeuner
7	7	mueslis
8	8	céréales aux fruits
9	9	mueslis aux fruits
10	10	plant-based foods and beverages
11	11	plant-based foods
12	12	cereals and potatoes
13	13	breads
14	14	sliced breads
15	15	sliced breads without crust
16	16	wholemeal breads
17	17	wholemeal sliced breads
18	18	alimentos y bebidas de origen vegetal
19	19	alimentos de origen vegetal
20	20	cereales y patatas
21	21	desayunos
22	22	cereales y derivados

product --> test mock

```
class GetCategory(TestCase):

    @mock.patch('requests.get')
    def test_get_categories(self, mock_get):
        mock_response = mock.Mock()
        mock_api = [
            {
                'tags': [
                    {
                        'id': 'en:plant-based-foods-and-beverages',
                        'known': 1,
                        'name': 'Aliments et boissons à base de végétaux',
                        'products': 107490,
                        'url': 'https://fr.openfoodfacts.org/categories-a-base-de-vegetaux'
                    }
                ]
            }
        ]

        mock_response.json.return_value = mock_api
        mock_response.status_code = 200
        mock_get.return_value = mock_response

        result = Command().get_category()
        self.assertEqual(result,
                         ['Aliments et boissons à base de végétaux'])
    
```

Openfoodfact's API request call test and response simulation.

mock_get.return_value = mock_response --> simulated values return

product --> templates

app --> base

--> base.html

app --> product

--> product_list.html

--> substitute_list.html



A screenshot of a search interface. At the top, there's a banner with yellow and white flowers and the word "QUALITE!". Below it is a search bar containing the text "Biscuit". A dropdown menu shows suggestions: "Biscuit" and "biscuit". To the right of the search bar is a red button labeled "rechercher". The background features a close-up image of a biscuit cookie.

Colette et Rémi



Three product cards are displayed side-by-side:

- D**: LU Thé Biscuits. The card includes an image of the product packaging, the name "Thé - Biscuits", and two small icons (info and search). Below the card is a button labeled "trouver des produits plus sains".
- A**: Gerblé Biscuits aux pommes. The card includes an image of the product packaging, the name "Biscuits aux pommes", and two small icons (info and search).
- B**: Gerblé Biscuits Goûter aux raisins. The card includes an image of the product packaging, the name "Biscuits Goûter aux raisins", and two small icons (info and search).



A large image of a LU Thé Biscuit box is shown at the top. Below it, the text "Thé - Biscuits" is displayed. The main message reads: "Vous pouvez remplacer cet aliment par :". Three smaller boxes below are labeled with letter "A":

- The first box contains Gerblé Biscuits aux pommes.
- The second box contains Gerblé Biscuits Goûter aux raisins.
- The third box contains Gerblé Biscuits aux fruits.

Gestion des produits --> django

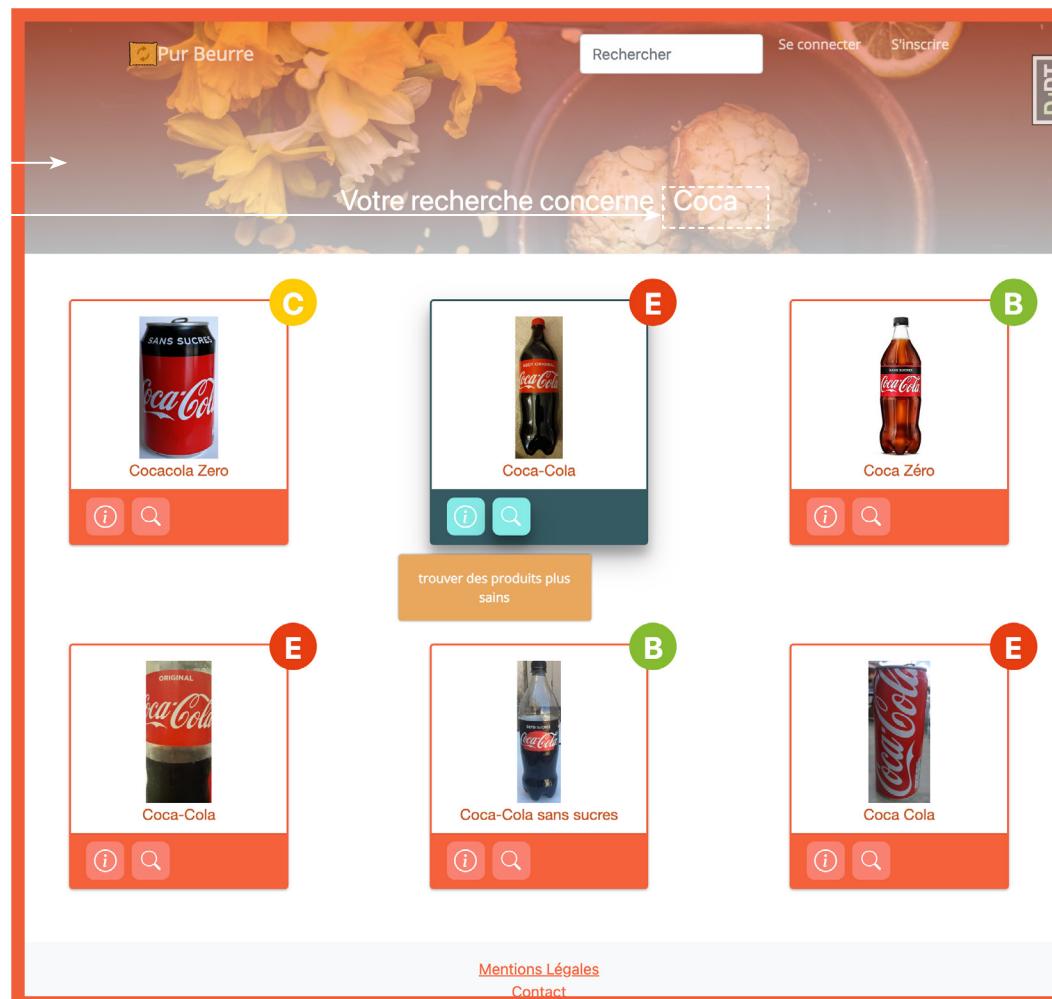
```
<div class=>col-lg-8 align-self-baseline>
    <p class=>text-white-75 mb-5> Find a substitute product.</p>
    <form method=>GET< form=> action=>% url 'product_list' %>>
        <div class=>input-group mb-3>
            <input type=>text< class=>form-control<
            placeholder=>Rechercher un produit< name=>search_product< value=>{{request.GET.search_product}}<
        </div>
        <input type=>submit< class=>btn btn-primary<
        value=>Rechercher<
    </div>
```

```
class ProductListView(ListView):
```

```
    template_name = 'product_list.html'
    model = Product

    def get_queryset(self):
        """ List of products according to user's research
        queryset --> get products filter by 'search_product' from base.html form

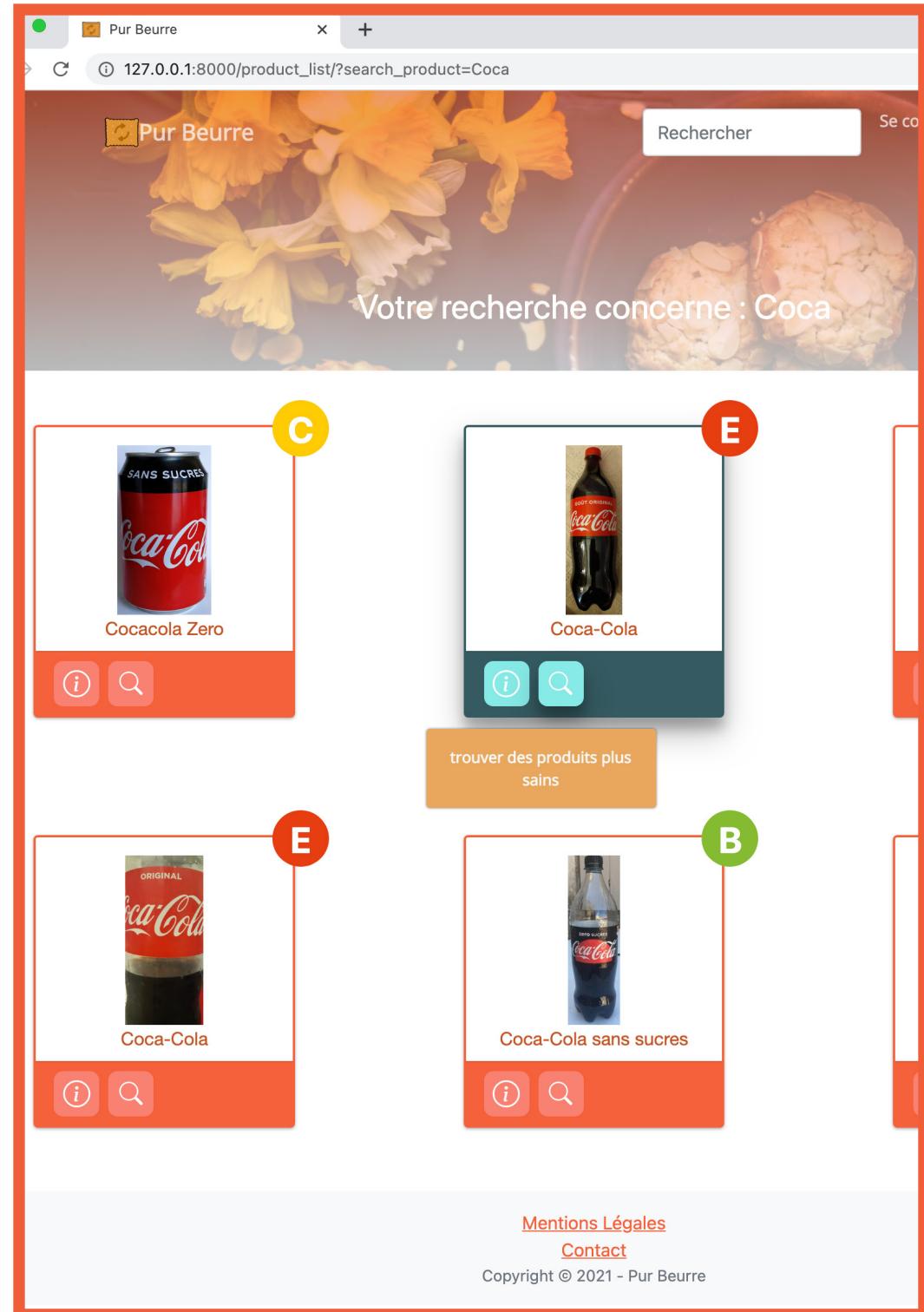
        Returns:
            Dict: list of products model queryset
        """
        search = self.request.GET.get('search_product', '').strip()
        qs = super().get_queryset()
        if search:
            return qs.filter(name__icontains=search)
        else:
            return qs
```



Gestion des produits -->html

```

<div class=>row>
    {% if product_list %}
        {% for product in product_list %}
            <div class=>col-lg-4 col-md-6 col-sm-12 d-flex col-card justify-content-center>
                <div class=>card product-card d-flex flex-column >
                    <img src=>{{product.picture}} class=>card-img-top product-img img-fluid rounded d-block align-m>
                    <p class=>card-title>>{{product.name}}</p>
                    <div class=>card-footer d-flex product align-content-around mt-auto>
                        <a class=>d-flex justify-content-center align-items-center btn btn-detail>
                            name=>detail_product type=>button href=>{% url 'detail_product' product.id %}>
                            data-toggle=>tooltip data-placement=>bottom>
                            <span>Afficher les détails du produit choisi</span>
                            <i class=>bi bi-info-circle></i>
                        </a>
                        <a class=>d-flex justify-content-center align-items-center btn btn-detail>
                            name=>search_substitute type=>button href=>{% url 'substitute_list' product_id=product.id %}>
                            value='{{request.GET.search_substitute}}' data-toggle=>tooltip data-placement=>bottom>
                            <span>Trouver des produits plus sains</span>
                            <i class=>bi bi-search></i>
                        </a>
                    </div>
                    <div class=>card-score card-score-color-{{ product.nutriscore }} text-center data-bs-toggle=>tooltip data-bs-placement=>right>
                        title=>Nutriscore, de A (très bon) à "E" (Très mauvais)>>{{product.nutriscore}}</div>
                </div>
            </div>
        {% endfor %}
    {% else %}
        <div class=>container h-100>
            <div class=>row justify-content-center h-100>
                <div class=>col-lg-4 col-md-6 col-sm-12 d-flex text-center>
                    <h2 class=>text-center no-favoris>Le produit n'existe pas !</h2>
                </div>
            </div>
        </div>
    
```



Tests --> test_views

```
from django.test import TestCase, RequestFactory, Client

class BaseTest(TestCase):
    def setUp(self):
        self.product_url = reverse('product_list')
        self.client = Client()
        self.factory = RequestFactory()
        return super().setUp()

class ProductListTest(BaseTest):

    def test_get_queryset(self):
        request = self.factory.get(self.product_url)
        view = ProductListView()
        view.request = request

        qs = view.get_queryset()

        self.assertQuerysetEqual(qs, Product.objects.all())

    def test_status_code(self):
        response = self.client.get(self.product_url)
        self.assertEqual(response.status_code, 200)
```

RequestFactory returns a **request**, Client returns a **response**.

RequestFactory --> this is a factory for creating request objects. Client simulates a complete **request-response** cycle.

Client executes views, tests views, inspects response.

RequestFactory 's Test --> **get_queryset**
Creating a request object using a request factory
Instantiate the view
I attach the request to the view
I call the method and compare it with the result

response Test HTTP --> **status_code**

Substituts --> templates, html, CSS

```
{% for substitute in product_list %}
    <div class=>col-lg-4 col-md-6 col-sm-12 d-flex col-card justify-content-center>
        <div class=>card product-card d-flex flex-column >
            <div class=>card-score card-score-color-{{ substitute.nutriscore }} text-center>
                data-bs-toggle=>tooltip data-bs-placement=>right>
                    title=>Nutriscore, de A (très bon) à "E" (Très mauvais)>>{{substitute.nutriscore}}
            </div>
            <img src=>{{substitute.picture}}> class=>card-img-top product-img img-fluid rounded d-block align-m>
            <p class=>card-title>{{substitute.name}}</p>
            <div class=>card-footer d-flex favoris align-content-around mt-auto>
                <a class=>d-flex justify-content-center align-items-center btn btn-detail name=>detail_product type=>button href=>{{ url 'detail_product' substitute.id }}>
                    data-toggle=>tooltip data-placement=>bottom>
                        <span>Afficher les détails du produit choisi</span>
                        <i class=>bi bi-info-circle></i>
                </a>
                {% if user.is_authenticated %}
                    <form method=>post action=>{{ url 'substitute_save' }}>
                        {% csrf_token %}
                        <input type=>hidden name=>product_id value=>{{ product.id }}>
                        <input type=>hidden name=>substitute_id value=>{{ substitute.id }}>
                        <button type=>submit class=>d-flex justify-content-center align-items-center btn btn-link btn-choice>
                            data-bs-toggle=>tooltip data-bs-placement=>bottom>
                                <span>Enregistrer ce produit sain dans vos favoris</span>
                                <i class=>bi bi-heart></i>
                        </button>
                    </form>
                {% endif %}
                </div>
            </div>
        </div>
    {% endfor %}
```

127.0.0.1:8000/substitutes/154

Pur Beurre

Rechercher

Coca-Cola

CSS

```
.product-card {
    position: relative;
    width: 65%;
    margin-bottom: 3.5rem;
    margin-top: 3rem;
    border-radius: 0.25rem;
    box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);
    transition: all 0.3s cubic-bezier(.25,.8,.25,1);
}

.product-card:hover {
    box-shadow: 0 14px 28px rgba(0,0,0,0.25), 0 10px 10px rgba(0,0,0,0.22);
    border: 2px solid #345a61;
```

A

B

B

B

Nutella Fans

←15

Substituts --> django

```
class SubstituteListView(ListView):

    template_name = 'product/substitute_list.html'
    model = Product
    context_object_name = 'product_list'

    def get_context_data(self):
        context = super().get_context_data()
        product = Product.objects.get(pk=self.kwargs.get('product_id'))
        context['product'] = product
        return context

    def get_queryset(self, *args, **kwargs):
        product_id = self.kwargs.get('product_id')
        product = Product.objects.get(pk=product_id)
        return product.get_substitutes()
```

context_object_name --> variable name returned by Django to display lists. I renamed it to `product_list`.

In `template_name = 'product/substitute_list.html'`
`{% for substitute in product_list %}`

def get_context_data --> get the data from the url using the attribute `kwargs`
`product = Product.objects.get(pk=self.kwargs.get('product_id'))`

def get_queryset --> retrieve the product id from the url,
make a query to retrieve the id from the database, then apply the `get_substitutes()` function to display the best products.

```
/ class SubstituteListTest(TestCase):
def test_same_category_better_nutriscore(self):
    brand_p1 = Brand.objects.create(name='ferrero').pk
    brand_p2 = Brand.objects.create(name='Funky Veggie').pk
    brand_p3 = Brand.objects.create(name='La Vache qui rit').pk
    category = Category.objects.create(name='Produits à tartiner')
    other_category = Category.objects.create(name='Produits laitiers')
    product = Product.objects.create(name='Nutella - Ferrero - 400 g',
                                      nutriscore='e', nova=4,
                                      url='https://fr.openfoodfacts.org/produit/'
                                      '3017620422003/nutella-ferrero',
                                      barcode=3017620422003,
                                      description='« Sucre, huile de palme, noisettes 13%»
                                      « lait écrémé en poudre 8,7%»
                                      «cacao maigre 7,4%, émulsifiants: lécithines [soja] ; vanilline. Sans gluten»,
                                      picture='https://images.openfoodfacts.org/images/'
                                      'products/301762/042/2003/front_en.288.400.jpg',
                                      brand_id=brand_p1)
    product_better = Product.objects.create(name='OUF! La pâte à tartiner Cacao Noisettes - Funky Veggie - 200 g',
                                            nutriscore='b', nova=3,
                                            url='https://fr.openfoodfacts.org/produit/'
                                            '3770008009653/ouf-la-pate-a-tartiner-cacao-noisettes-funky-veggie',
                                            barcode=3770008009653,
                                            description='«Purée de haricots rouges* 30% (eau, farine de haricots rouges*),
                                            «sucre de fleur de coco*»
                                            «eau, purée de noisettes torréfiées* 17,5%, poudre de cacao 4,8.%»
                                            «ingrédients issus de l'agriculture biologique»,
                                            picture='https://images.openfoodfacts.org/images/'
                                            'products/377/000/800/9653/front_fr.63.400.jpg',
                                            brand_id=brand_p2)
    product_worst = Product.objects.create(name='La Vache qui rit 32 portions - 535 g',
                                           nutriscore='d', nova=4,
                                           url='https://fr.openfoodfacts.org/produit/'
                                           '3073781070309/la-vache-qui-rit-32-portions',
                                           barcode=3073781070309,
                                           description='« LAIT écrémé réhydraté (origine : France),»
                                           « FROMAGES, BEURRE, protéines de LAIT,»
                                           « concentré des minéraux du LAIT,»
                                           « sels de fonte : polyphosphates, concentré LACTIQUE LAITIER.»,
                                           picture='https://images.openfoodfacts.org/images/'
                                           'products/307/378/107/0309/front_fr.64.400.jpg',
                                           brand_id=brand_p3)
    product.categories.add(category)
    product_same_category.categories.add(category)
    product_other_category.categories.add(other_category)
    response = product.get_substitutes()
    assert product_better in response
    assert product_worst not in response
```

Coverage --> launch

The Nutella lover's platform product has a test coverage of 90%.

python3 manage.py test test.users_account.
tests.test_views.SignUpTest.test_can_signup

coverage run manage.py test -v 2 && covert
report & html

```
nutella_fans -- bash -- 140x92
ast login: Sun Sep 12 10:28:48 on ttys000
(shpine standard)
he default interactive shell is now zsh.
o update your account to use zsh, please run `chsh -s /bin/zsh`.
or more details, please visit https://support.apple.com/kb/HT208050.
Mac-de-XavGab:~ xavgab$ cd /Users/xavgab/Documents/Python/08_Projet\ 8/nutella_fans
Mac-de-XavGab:nutella_fans xavgab$ source myenv/bin/activate
(myenv) iMac-de-XavGab:nutella_fans xavgab$ coverage run --source test -v 2 && coverage report
o such option: -v
se 'coverage help' for help.
ull documentation is at https://coverage.readthedocs.io
(myenv) iMac-de-XavGab:nutella_fans xavgab$ coverage run manage.py -v 2 && coverage report
nknown command: '-v'
ype 'manage.py help' for usage.
(myenv) iMac-de-XavGab:nutella_fans xavgab$ coverage run manage.py test -v 2 && coverage re
reating test database for alias 'default' ('test_openfoodfact_db')...
erations to perform:
Synchronize unmanaged apps: base, debug_toolbar, messages, staticfiles
Apply all migrations: admin, auth, contenttypes, product, save_substitute, sessions, users
ynchronizing apps without migrations:
Creating tables...
Running deferred SQL...
unning migrations:
Applying product.0001_initial... OK
Applying save_substitute.0001_initial... OK
Applying contenttypes.0001_initial... OK
Applying contenttypes.0002_remove_content_type_name... OK
Applying auth.0001_initial... OK
Applying auth.0002_alter_permission_name_max_length... OK
Applying auth.0003_alter_user_email_max_length... OK
Applying auth.0004_alter_user_username_opts... OK
Applying auth.0005_alter_user_last_login_null... OK
Applying auth.0006_require_contenttypes_0002... OK
Applying auth.0007_alter_validators_add_error_messages... OK
Applying auth.0008_alter_user_username_max_length... OK
Applying auth.0009_alter_user_last_name_max_length... OK
Applying auth.0010_alter_group_name_max_length... OK
Applying auth.0011_update_proxy_permissions... OK
Applying auth.0012_alter_user_first_name_max_length... OK
Applying users.account.0001_initial... OK
Applying admin.0001_initial... OK
Applying admin.0002_logentry_remove_auto_add... OK
Applying admin.0003_logentry_add_action_flag_choices... OK
Applying sessions.0001_initial... OK
ystem check identified no issues (0 silenced).
est_create_product (nutella_fans.product.tests.test_models.CreateProduct) ... ok
est_not_create_product (nutella_fans.product.tests.test_models.CreateProduct) ... None
k
est_without_product_name (nutella_fans.product.tests.test_models.CreateProduct) ... ok
est_get_categories (nutella_fans.product.tests.test_models.GetCategory) ... ok
est_get_product (nutella_fans.product.tests.test_models.GetProduct) ... ok
est_brand_name_is_name (nutella_fans.product.tests.test_models.ProductTestCase) ... ok
est_category_name_is_name (nutella_fans.product.tests.test_models.ProductTestCase) ... ok
est_lower_item (nutella_fans.product.tests.test_models.ProductTestCase) ... ok
est_name_label (nutella_fans.product.tests.test_models.ProductTestCase) ... ok
est_name_max_length (nutella_fans.product.tests.test_models.ProductTestCase) ... ok
est_store_name_is_name (nutella_fans.product.tests.test_models.ProductTestCase) ... ok
est_status_code_200 (nutella_fans.product.tests.test_views.ProductDetailTest) ... ok
est_get_queryset (nutella_fans.product.tests.test_views.ProductListTest) ... ok
est_status_code (nutella_fans.product.tests.test_views.ProductListTest) ... ok
est_same_category (nutella_fans.product.tests.test_views.SubstituteListTest) ... ok
est_status_code_200 (nutella_fans.product.tests.test_views.SubstituteListTest) ... ok
est_favorite_list (nutella_fans.save_substitute.tests.test_views.FavoriteListTest) ... ok
est_favorite_with_valid_user (nutella_fans.save_substitute.tests.test_views.SaveFavoriteTest) ... ok
est_login (nutella_fans.users.account.tests.test_forms.LoginTest) ... ok
est_invalid_password (nutella_fans.users.account.tests.test_forms.UserFormTest) ... ok
est_userform_valid (nutella_fans.users.account.tests.test_forms.UserFormTest) ... ok
est_can_view_page_correctly (nutella_fans.users.account.tests.test_views.LoginTest) ... ok
est_login_sucess (nutella_fans.users.account.tests.test_views.LoginTest) ... ok
est_user_logout (nutella_fans.users.account.tests.test_views.LogoutTest) ... ok
est_get_queryset (nutella_fans.users.account.tests.test_views.ProfilTest) ... ok
est_can_signup (nutella_fans.users.account.tests.test_views.SignUpTest) ... ok
est_invalid_email (nutella_fans.users.account.tests.test_views.SignUpTest) ... ok
est_unmatching_password (nutella_fans.users.account.tests.test_views.SignUpTest) ... ok
est_view_ok (nutella_fans.users.account.tests.test_views.SignUpTest) ... ok
est_login (nutella_fans.product.tests.test_functional.PurBeurreTest) ... ok
est_search_product (nutella_fans.product.tests.test_functional.PurBeurreTest) ... ok
+geometry+
an 32 tests in 18.157s
[89/616]
]
K
estroying test database for alias 'default' ('test_openfoodfact_db')...
```

```
Applying auth.0011_update_proxy_permissions... OK
Applying auth.0012_alter_user_first_name_max_length... OK
Applying users.account.0001_initial... OK
Applying admin.0001_initial... OK
Applying admin.0002_logentry_remove_auto_add... OK
Applying admin.0003_logentry_add_action_flag_choices... OK
Applying sessions.0001_initial... OK
ystem check identified no issues (0 silenced).
est_create_product (nutella_fans.product.tests.test_models.CreateProduct) ... ok
est_not_create_product (nutella_fans.product.tests.test_models.CreateProduct) ... None
k
est_without_product_name (nutella_fans.product.tests.test_models.CreateProduct) ... ok
est_get_categories (nutella_fans.product.tests.test_models.GetCategory) ... ok
est_get_product (nutella_fans.product.tests.test_models.GetProduct) ... ok
est_brand_name_is_name (nutella_fans.product.tests.test_models.ProductTestCase) ... ok
est_category_name_is_name (nutella_fans.product.tests.test_models.ProductTestCase) ... ok
est_lower_item (nutella_fans.product.tests.test_models.ProductTestCase) ... ok
est_name_label (nutella_fans.product.tests.test_models.ProductTestCase) ... ok
est_name_max_length (nutella_fans.product.tests.test_models.ProductTestCase) ... ok
est_store_name_is_name (nutella_fans.product.tests.test_models.ProductTestCase) ... ok
est_status_code_200 (nutella_fans.product.tests.test_views.ProductDetailTest) ... ok
est_get_queryset (nutella_fans.product.tests.test_views.ProductListTest) ... ok
est_status_code (nutella_fans.product.tests.test_views.ProductListTest) ... ok
est_same_category (nutella_fans.product.tests.test_views.SubstituteListTest) ... ok
est_status_code_200 (nutella_fans.product.tests.test_views.SubstituteListTest) ... ok
est_favorite_list (nutella_fans.save_substitute.tests.test_views.FavoriteListTest) ... ok
est_favorite_with_valid_user (nutella_fans.save_substitute.tests.test_views.SaveFavoriteTest) ... ok
est_login (nutella_fans.users.account.tests.test_forms.LoginTest) ... ok
est_invalid_password (nutella_fans.users.account.tests.test_forms.UserFormTest) ... ok
est_userform_valid (nutella_fans.users.account.tests.test_forms.UserFormTest) ... ok
est_can_view_page_correctly (nutella_fans.users.account.tests.test_views.LoginTest) ... ok
est_login_sucess (nutella_fans.users.account.tests.test_views.LoginTest) ... ok
est_user_logout (nutella_fans.users.account.tests.test_views.LogoutTest) ... ok
est_get_queryset (nutella_fans.users.account.tests.test_views.ProfilTest) ... ok
est_can_signup (nutella_fans.users.account.tests.test_views.SignUpTest) ... ok
est_invalid_email (nutella_fans.users.account.tests.test_views.SignUpTest) ... ok
est_unmatching_password (nutella_fans.users.account.tests.test_views.SignUpTest) ... ok
est_view_ok (nutella_fans.users.account.tests.test_views.SignUpTest) ... ok
est_login (nutella_fans.product.tests.test_functional.PurBeurreTest) ... ok
est_search_product (nutella_fans.product.tests.test_functional.PurBeurreTest) ... ok
+geometry+
an 32 tests in 18.157s
[89/616]
]
K
estroying test database for alias 'default' ('test_openfoodfact_db')...
```

Coverage --> report html

The Nutella lover's platform product has a test coverage of 90%

Average report: 90%

file	statements	missing	excluded	coverage
page.py	12	2	0	83%
cella_fans/__init__.py	0	0	0	100%
cella_fans/base/__init__.py	0	0	0	100%
cella_fans/base/apps.py	4	0	0	100%
cella_fans/base/urls.py	3	0	0	100%
cella_fans/base/views.py	10	2	0	80%
cella_fans/product/__init__.py	0	0	0	100%
cella_fans/product/admin.py	6	0	0	100%
cella_fans/product/apps.py	4	0	0	100%
cella_fans/product/management/__init__.py	0	0	0	100%
cella_fans/product/management/commands/__init__.py	0	0	0	100%
cella_fans/product/management/commands/import_off.py	55	16	0	71%
cella_fans/product/models.py	36	0	0	100%
cella_fans/product/urls.py	3	0	0	100%
cella_fans/product/views.py	29	1	0	97%
cella_fans/save_substitute/__init__.py	0	0	0	100%
cella_fans/save_substitute/admin.py	3	0	0	100%
cella_fans/save_substitute/apps.py	4	0	0	100%
cella_fans/save_substitute/models.py	7	1	0	86%
cella_fans/save_substitute/urls.py	3	0	0	100%
cella_fans/save_substitute/views.py	39	5	0	87%
cella_fans/settings.py	29	0	0	100%
cella_fans/urls.py	5	0	0	100%
cella_fans/users_account/__init__.py	0	0	0	100%
cella_fans/users_account/admin.py	4	0	0	100%
cella_fans/users_account/apps.py	4	0	0	100%
cella_fans/users_account/forms.py	14	0	0	100%
cella_fans/users_account/models.py	7	1	0	86%
cella_fans/users_account/urls.py	6	0	0	100%
cella_fans/users_account/views.py	34	3	0	91%
total	321	31	0	90%

verage.py v5.5, created at 2021-08-23 15:52 +0200

Coverage for nutella_fans/save_substitute/views.py : 87%

39 statements | 34 run | 5 missing | 0 excluded

```
from django.views.generic import CreateView
from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic import ListView
from django.views.generic import DeleteView
from django.urls import reverse_lazy
from django.shortcuts import redirect
from django.contrib import messages

from nutella_fans.save_substitute.models import Substitute, Product
from nutella_fans.users_account.models import User

class FavoriteListView(ListView):
    template_name = 'favorites_list.html'
    model = Substitute
    context_object_name = 'favorite_list'

    def get_queryset(self):
        queryset = super().get_queryset()
        return queryset.filter(user=self.request.user)

class SubstituteSaveView(LoginRequiredMixin, CreateView):
    template_name = 'substitute_list.html'
    model = Substitute
    fields = ['product', 'substitute']

    def post(self, request, *args, **kwargs):
        product = Product.objects.get(
            pk=request.POST.get('product_id'))
        substitute = Substitute.objects.filter(
            product_id=product.id, substitute_id=request.POST.get('substitute_id'))
        user_id = request.user
        user = User.objects.get(id=user_id.id)
        if not substitute.exists():
            substitute = Substitute.objects.create(product_id=product.id,
                                                    substitute_id=request.POST.get('substitute_id'))
            user.substitutes.add(substitute)
            messages.success(
                request, 'Le produit est bien dans votre liste de favoris')
        else:
            messages.warning(
                request, 'Le produit existe déjà dans votre liste de favoris')
        return redirect('favorites_list')

class FavoriteDeleteView(DeleteView):
    model = Substitute
    success_url = reverse_lazy('favorites_list')

    def get(self, request, *args, **kwargs):
        substitute_id = self.kwargs.get('fav_id')
        delete = self.post(request, Substitute.objects.filter(
            substitute_id=substitute_id).delete())
        messages.success(request, 'Le substitut a bien été supprimé')
        return delete
```

```
16/Sep/2021 16:12:09] "GET /static/assets/img/bg-purbeurre.jpg HTTP/1.1" 200 1955650
[16/Sep/2021 16:12:09] "GET /static/assets/favicon.ico HTTP/1.1" 200 679
[16/Sep/2021 16:12:14] "GET / HTTP/1.1" 200 21002
[16/Sep/2021 16:12:14] "GET /static/css/styles.css HTTP/1.1" 200 203956
[16/Sep/2021 16:12:14] "GET /static/js/scripts.js HTTP/1.1" 200 1779
[16/Sep/2021 16:12:14] "GET /static/debug_toolbar/css/toolbar.css HTTP/1.1" 200 11461
[16/Sep/2021 16:12:14] "GET /static/debug_toolbar/js/toolbar.js HTTP/1.1" 200 10452
[16/Sep/2021 16:12:14] "GET /static/debug_toolbar/js/utils.js HTTP/1.1" 200 2988
[16/Sep/2021 16:12:14] "GET /static/assets/img/logo_pur_beurre.png HTTP/1.1" 200 11846
[16/Sep/2021 16:12:14] "GET /static/assets/img/portfolio/thumbnails/remi-11.png HTTP/1.1" 200 276313
[16/Sep/2021 16:12:14] "GET /static/assets/img/portfolio/thumbnails/colette-6.jpg HTTP/1.1" 200 160090
[16/Sep/2021 16:12:14] "GET /static/assets/img/portfolio/thumbnails/remy-2.jpg HTTP/1.1" 200 198168
[16/Sep/2021 16:12:14] "GET /static/assets/img/bg-purbeurre.jpg HTTP/1.1" 200 1933650
[16/Sep/2021 16:12:14] "GET /static/debug_toolbar/css/print.css HTTP/1.1" 200 43
[16/Sep/2021 16:12:15] "GET /product_list/?search_product=Biscuit HTTP/1.1" 200 35454
[16/Sep/2021 16:12:15,101] - Broken pipe from ('127.0.0.1', 57687) to ('127.0.0.1', 57687)
```

Heroku --> settings

Several settings

- > local (production)
- > development (deployment)

heroku run bash ou heroku run dans le terminal

git push heroku main

heroku run python3 manage.py migrate

The process of generating tables within the database.

heroku run manage.py import_off

importing data through API calls.

The screenshot shows the Heroku dashboard for the 'openclassroom-nutella-fans' application. At the top, there are tabs for Overview, Resources, Deploy, Metrics, Activity, Access, and Settings. Below these, a message encourages using Heroku Pipelines for visualization. The 'Installed add-ons' section shows a single instance of Heroku Postgres Hobby Dev plan named 'postgresql-cubic-98940'. The 'Latest activity' section lists three recent events: a deployment by gabrielleazadian@gmail.com at 1:53 PM, a build success by the same user at 1:53 PM, and another deployment by the same user at 1:53 PM.

This screenshot shows the detailed view of the 'postgresql-cubic-98940' database within the Heroku Datastores section. The service is 'heroku-postgresql', the plan is 'hobby-dev', and it is associated with the 'openclassroom-nutella-fans' app. The 'HEALTH' status is 'Available'. Key metrics shown include: PRIMARY Yes, VERSION 13.4, CREATED 10 days ago, MAINTENANCE Unsupported, ROLLBACK Unsupported. In the 'UTILIZATION' section, it shows 0 of 20 connections, 18 tables, and 7,263 rows out of 10,000, which is noted as being close to the row limit. The 'DATA SIZE' is listed as 11.1 MB.