



Plateforme pour amateur de Nutella

Du gras mais de bonne qualité ...

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



A screenshot of a web browser window titled "Pur Beurre". The page features a background image of yellow flowers. Text on the page includes "bienvenue, DU GRAS QU'", "Trouvez un produit de subs", "Rechercher un produit", and "Cole". A small text block at the bottom left states "Pur Beurre est l'initiative de c Colette Tatou et Rémy, que vo notre excell". At the bottom, there are two cartoon character avatars, one male and one female, and a small image of the Eiffel Tower in Paris.

SOMMAIRE

| | |
|--|----|
| CONTEXTE | 3 |
| TRELLO | 4 |
| MODELES | 5 |
| DJANGO users_account | 6 |
| FBC-->CBV | 7 |
| product Api et base de données | 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 --> lancement des tests | 16 |
| --> report && html | 17 |
| HEROKU --> settings | 18 |

Contexte

La société Pur Beurre désire créer une plateforme pour les amateurs de produits sains.

Nous devons proposer des produits de meilleures qualités en fonction d'une recherche effectuer par l'utilisateur.

Par exemple, si l'utilisateur recherche des chips, il faut trouver un produit possédant un nutriscore de meilleur qualité (A ou B).

Si l'utilisateur est satisfait de la proposition, une fois connecté à son compte, il peut l'enregistrer dans ses favoris.

Il peut aussi supprimer un favoris si celui ci ne lui convient plus.

The screenshot shows the homepage of the Pur Beurre website. The header features the logo "Pur Beurre" and a search bar with the placeholder "Rechercher". Below the header is a large banner with a background image of yellow flowers and some cookies. The banner contains the text "bienvenue, sain gastronome ! DU GRAS OUI, MAIS DE QUALITÉ !" in large white letters. A subtext below reads "Trouvez un produit de substitution pour ceux que vous consommez tous les jours.". There are two search input fields: "Rechercher un produit" and a red "rechercher" button. The footer section is orange and introduces "Colette et Rémi" as the initiators of the project. It mentions they are from the Ratatouille restaurant and provides a brief description of their initiative. At the bottom, there are three small images: a chef, the Eiffel Tower, and a cartoon character.

Trello

Nutella Fans

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 | L'équipe de Pur Beurre | Public | Inviter | Automatisation | Afficher le menu

Users Stories

A faire

En cours

Terminer

+ Ajouter une carte

+ Ajouter une carte

+ Ajouter une carte

Création du formulaire création du compte

Description

Création du compte utilisateur

Activité

Commentaires

Gabrielle Azadian a déplacé cette carte de A faire à Terminer

Gabrielle Azadian a terminé Créer la page Mon Compte (suive le cahier des charges) sur cette carte

Gabrielle Azadian a terminé Icône Crée un Compte nouvel utilisateur sur cette carte

Gabrielle Azadian a terminé Icône Mon Compte utilisateur connecté sur cette carte

Gabrielle Azadian a terminé Page de création de compte ainsi que le formulaire associé sur cette carte

Gabrielle Azadian a ajouté Création du compte utilisateur à cette carte

Gabrielle Azadian a ajouté cette carte à A faire

AJOUTER À LA CARTE

Membres

Étiquettes

Checklist

Dates

Pièce jointe

Image de couverture

Champs personnels

POWER-UPS

ACTIONS

Créer

Inviter

Créer un projet Django

Mise en forme -> graphisme

Test intégration, unitaires

Initialiser un repo

Création du formulaire création du compte

Requête Openfoodfacts

Enregistrement des aliments

Mise en ligne Heroku

Docstrings

+ Ajouter une carte

← 4

Modèles

```
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 with two main panes. The left pane displays a list of tables from the current schema, while the right pane shows a detailed view of the 'information_schema' and 'pg_catalog' databases.

Tables in Current Schema:

- 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

Databases Viewed:

- information_schema
- pg_catalog

Search Bar: Search

Buttons: + Table, + View, + Materialized View

Page Number: ← 5

users_account --> template

Page de connexion et d'inscription

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 overlay reads "Connectez-vous à votre profil !". Below the image, there is a form with fields for "Nom d'utilisateur" and "Mot de passe", and a red "se connecter" button. There are also links for "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 is "Création de votre compte Pur Beurre". Below it, there is a sub-headline "Inscrivez-vous au site Pur Beurre pour enregistrer vos favoris". The registration form includes fields for "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 "Mot de passe" field is a list of password requirements:

- 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.
- Votre mot de passe ne peut pas être entièrement numérique.

There is also 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 at the bottom of the form. There are also links for "Vous avez déjà un compte ?" and "Connectez-vous!".

FBC --> CBV

Function-Based Views -->

Class-Based View

-> scripts FBV pour le login

`urls.py`
`views.py`

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

```
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

-> scripts FBV
`urls.py`
`views.py`

Class generic LoginView

```
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 et base de données

```
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 JSON structure is as follows:

```

{
  "count": 3825,
  "tags": [
    {
      "id": "en:snacks",
      "known": 1,
      "name": "Snacks",
      "products": 71173,
      "url": "https://us.openfoodfacts.org/category/snacks"
    },
    {
      "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"
    },
    ...
    {
      "id": "...",
      "known": "...",
      "name": "...",
      "products": "...",
      "url": "..."
    }
  ]
}

```

The screenshot shows a PostgreSQL database viewer connected to the `openfoodfact_db` database. The `product_category` table is selected, displaying the following data:

| id | name |
|----|---|
| 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 |
| 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...ssons-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'])
    )
```

Test pour imiter et simuler une réponse à l'appel de l'api Openfoodfacts.

mock_get.return_value = mock_response --> valeur de retours simulées

product --> templates

app --> base

--> base.html

app --> product

--> product_list.html

--> substitute_list.html

The image shows a search interface on a website. At the top, there is a banner with yellow and white flowers and the word "QUALITE!" in large letters. Below the banner, a search bar contains the text "Biscuit". To the right of the search bar is a red button labeled "rechercher". A dropdown menu below the search bar also lists "Biscuit". The background of the page features a close-up image of a biscuit.

QUALITE!

Trouvez un produit de substitution pour ceux que vous consommez tous les jours.

Biscuit

Biscuit
biscuit

rechercher

Colette et Rémi

This section displays three product cards, each featuring a different biscuit brand and flavor. The cards are labeled A, B, and D from left to right.

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

The image shows a comparison between LU Thé biscuits and healthier alternatives. At the top, there is a banner with yellow and white flowers and the word "QUALITE!". Below the banner, a large image of LU Thé biscuits is shown next to a smaller image of Gerblé Biscuits aux pommes. The text "Thé - Biscuits" is displayed below the LU product. The bottom part of the image features a large, artistic photograph of a stack of biscuits with almond flakes.

QUALITE!

Trouvez un produit de substitution pour ceux que vous consommez tous les jours.

Colette et Rémi

LU Thé Biscuits

Gerblé Biscuits aux pommes

Gerblé Biscuits Goûter aux raisins

trouver des produits plus sains

Vous pouvez remplacer cet aliment par :

A

A

A

Gestion des produits --> django

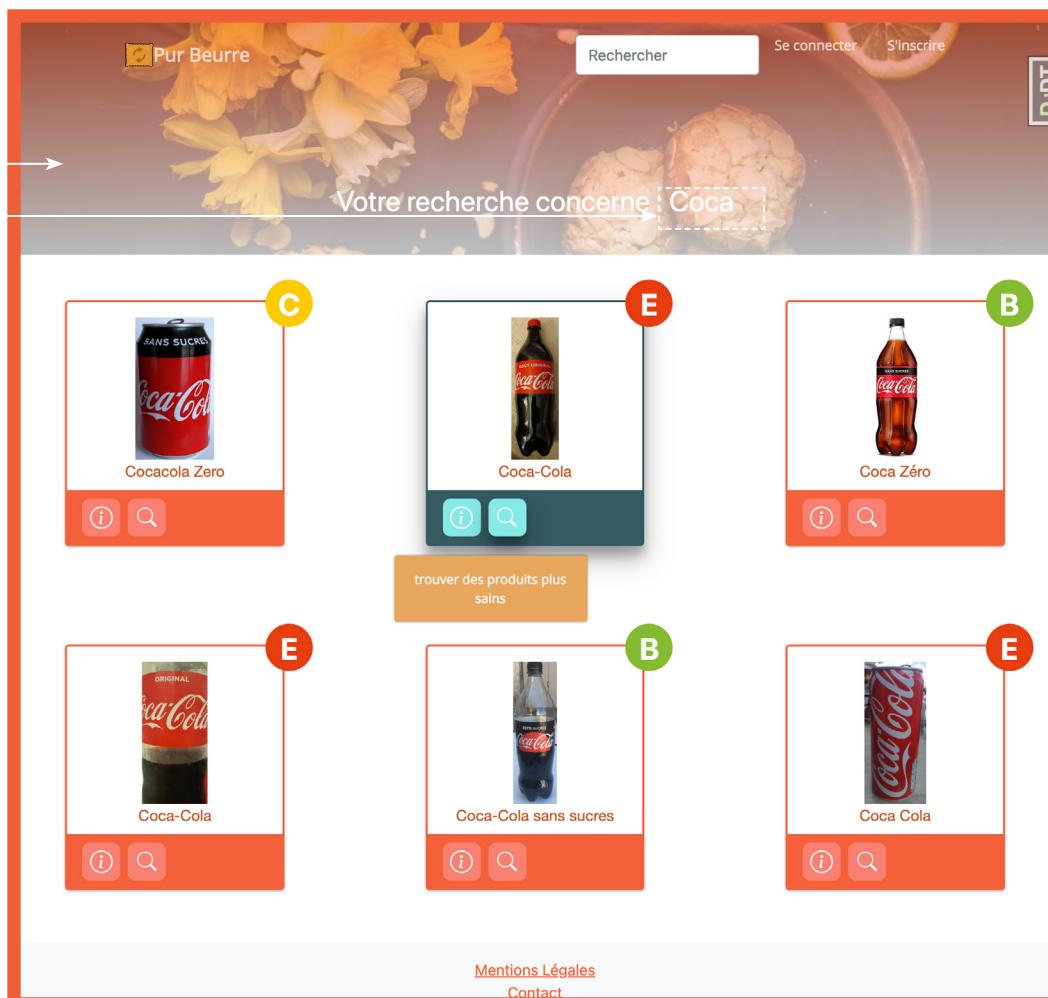
```
<div class=>col-lg-8 align-self-baseline>
    <p class=>text-white-75 mb-5> Trouvez un produit de substitution pour ceux que vous consommez tous les jours.</p>
    <form method=>GET action=>% url 'product_list' %>>
        <div class=>input-group mb-3>
            <input type=>text class=>form-control placeholder=>Rechercher un produit name=>search_product value='{{request.GET.search_product}}'>
            <input type=>submit class=>btn btn-primary value=>Rechercher</input>
        </div>
    </form>
</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

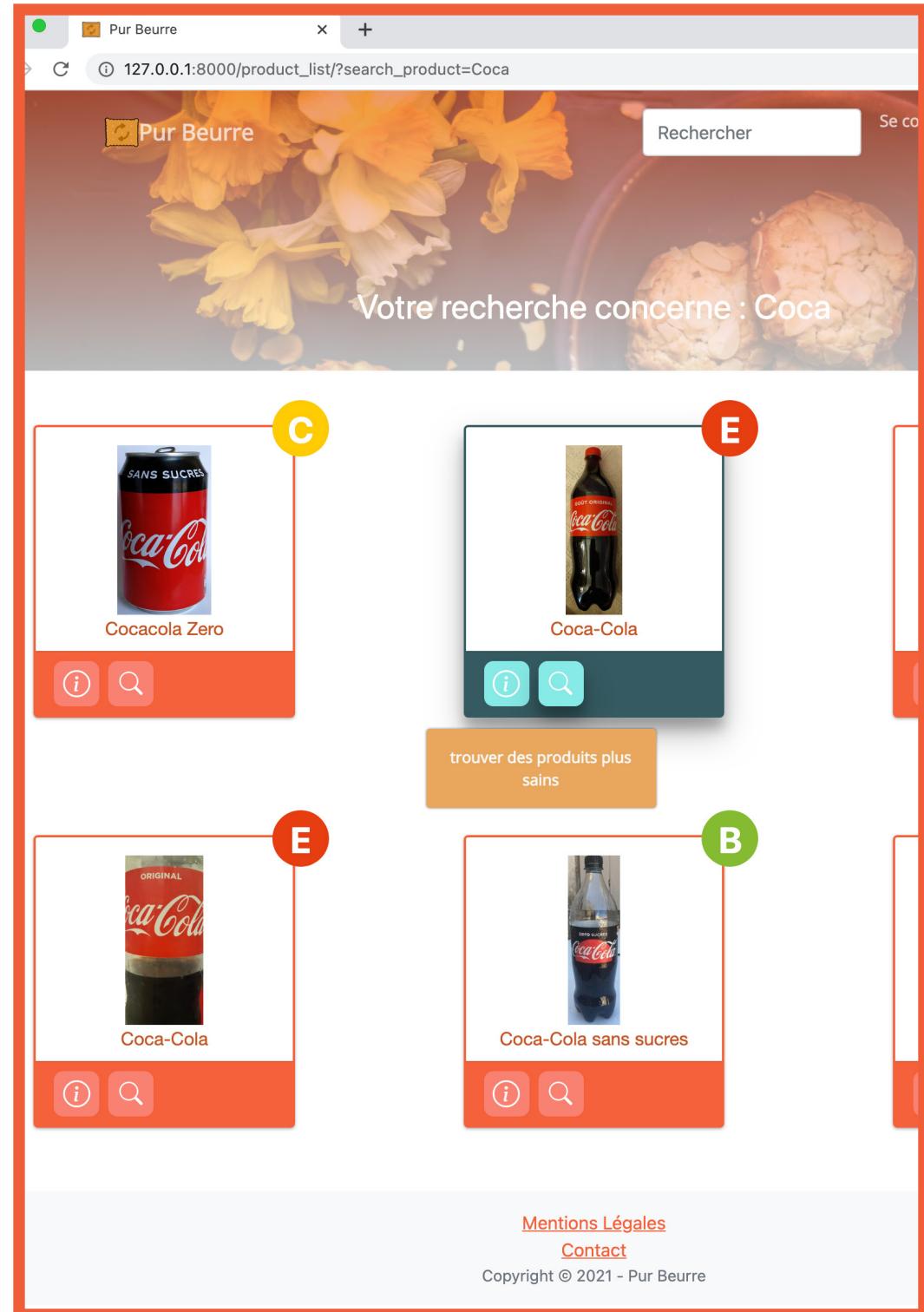
        Returns:
            Dict: list of products model queryset
        """
        search = self.request.GET.get('search_product', '').strip()
        qs = super().get_queryset()
        if search:
            return qs.filter(name__icontains=search)
        else:
            return qs
```



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 renvoie un **request**, Client renvoie un **response**.
RequestFactory --> c'est une fabrique pour créer request objets.
Client simule un cycle complet de **requête-reponse**.
Le Client exécute les vues, teste les vues, inspecte la réponse.

Test avec RequestFactory --> **get_queryset**
Création d'un object request utilisant request factory
On instancie la vue
J'attache la requête à la vue
J'appelle la méthode et je la compare au résultat

Test de la réponse HTTP du status_code

Gestion des 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

enregistrer ce produit sain dans vos favoris

←14

Gestion des 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` --> nom de variable retourner par Django pour afficher les listes. Je l'ai renommé en `product_list`

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

`def get_context_data` --> récupère les données dans l'url par l'attribut `kwargs`
`product = Product.objects.get(pk=self.kwargs.get('product_id'))`

`def get_queryset` --> on récupère l'id du produit dans l'url
on fait une requête afin de récupérer l'id dans la base de donnée ensuite on lui applique la fonction `get_substitutes()` pour afficher les meilleurs produits.

```
/ 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/301/762/042/2003/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
```


Heroku --> settings

Création de plusieurs settings
--> local (production)
--> développement (déploiement)

heroku run bash ou heroku run dans le terminal
git push heroku main

heroku run python3 manage.py migrate
Création des tables dans la base de données

heroku run manage.py import_off
importation des données avec l'appel à l'api

The screenshot shows the Heroku dashboard for the app 'openclassroom-nutella-fans'. The top navigation bar includes links for 'Vues génériques fondées sur le', 'openclassroom-nutella-fans | Heroku', and 'Salesforce Platform'. The dashboard has sections for 'Overview', 'Resources', 'Deploy', 'Metrics', 'Activity', 'Access', and 'Settings'. A prominent feature is the 'Heroku Pipelines' integration button. Below this, the 'Installed add-ons' section shows a single 'Heroku Postgres' instance on the 'Hobby Dev' plan named 'postgresql-cubic-98940'. The 'Latest activity' section shows three recent events: a deployment by gabrielleazadian@gmail.com at 1:53 PM, a build success by the same user, and another deployment by the same user at 1:53 PM. The bottom section provides a detailed view of the PostgreSQL database, including its service name 'heroku-postgresql', plan 'hobby-dev', and associated app 'openclassroom-nutella-fans'. It shows the database is available, primary status, version 13.4, created 10 days ago, and maintenance status. Utilization statistics include 0 of 20 connections, 18 tables, and 7,263 rows out of 10,000, which is close to the row limit. Data size is listed as 11.1 MB.