TP18 – Graphiques

Introduction, Contexte et Objectifs

Dans ce dernier TP on code pour l'administrateur des fonctionnalités statistiques :

- les revenus jour par jour sur les 7 derniers jours (graphique en barres verticales),
- les ventes pour chaque pizza sur les 7 derniers jours (graphique circulaire),
- …

Ces graphiques sont obtenus grâce au framework chart.js, par lecture de variables de gabarit.

Bien sûr, pour que ces graphiques aient un sens, il va falloir insérer beaucoup de commandes (via le site), et changer leur date (cette fois, dans sqlitebrowser) puisque par défaut c'est la date courante qui est donnée. Ce travail préalable est nécessaire.

Intégration de chart.js

Pour utiliser ce framework javascript, insérez dans le template base.html, juste avant la balise fermante </body>, la ligne suivante

```
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
```

L'application applistats

Créez la nouvelle application applistats, comme pour les trois précédentes applications. N'oubliez pas de l'intégrer dans INSTALLED_APPS, de créer le fichier urls.py d'applistats et d'insérer les urls d'applistats dans le fichier urls.py de webpizza. Créez aussi le répertoire de templates pour applistats.

Le graphique des revenus des 7 derniers jours

1. On va ici apprendre à coder une requête SQL sans passer par les mécanismes de django, ce qui peut être nécessaire pour des requêtes complexes.

   ATTENTION : toujours se méfier des requêtes non préparées. Ici, aucun risque car il n'y aura pas d'interaction avec le navigateur du client. Mais vous devez garder en tête que django, dans l'esprit framework, fait presque tout à votre place, y compris la sécurisation de la base de données. Sauf si vous construisez vous-même vos requêtes …

2. Créez, dans applistats, la view

   revenus(request)

   dont le code détaillé et commenté est le suivant :

```
def revenus(request) :
    # le user (ici ce sera un administrateur)
    user = PizzaUser.objects.get(id = request.user.id)

    # la requête écrite "à la main". Notez les triples
    # guillemets pour écrire une string sur plusieurs lignes
    requete = """
        SELECT *
        FROM applipanier_commande
        WHERE payee = 1
        AND julianday(date()) - julianday(dateCommande) <= 7
        ORDER BY dateCommande;
    """

    # exécution de la requête
    lesCommandesPayees = Commande.objects.raw(requete)

    # initialisation des variables qui seront utilisées
    # par le cript js via des variables de gabarit
    lesDatesDesCommandes = []
    lesMontantsDesCommandes = []
    CAmax = 0
```

```python
# l'algorithme pour qu'à un même indice i on ait
# - la date de la commande dans la première liste
# - le montant de la commande dans la deuxième liste
for commande in lesCommandesPayees :
    laDate = commande.dateCommande.strftime('%d/%m/%Y')
    leMontant = float(commande.prix)
    if laDate not in lesDatesDesCommandes :
        lesDatesDesCommandes.append(laDate)
        lesMontantsDesCommandes.append(leMontant)
    else :
        n = lesDatesDesCommandes.index(laDate)
        lesMontantsDesCommandes[n] += leMontant

# calcul du montant maximum pour adapter l'échelle des y du graphique
for montant in lesMontantsDesCommandes :
    if montant > CAmax :
        CAmax = montant

# calcul de la graduation max pour l'axe des y
# par exemple CAmax = 385 donne 385 // 50 = 7, 7 * 50 + 50 = 400
CAmax = ceil(CAmax // 50) * 50 + 50

# appel du template
return render(
    request,
    'applistats/revenus.html',
    {"dates" : lesDatesDesCommandes, "montants" : lesMontantsDesCommandes,
     "user" : user, "CAmax" : CAmax}
)
```

Concrètement :

a. on récupère les commandes des 7 derniers jours,
b. on crée en parallèle deux listes, une qui contient les dates (et qui alimentera l'axe des x), et une autre qui contient les montants (pour l'axe des y),
c. on calcule le montant maximum pour avoir une échelle adaptée en y.

3. Créez pour applistats l'url

path('stats/revenus/', views.revenus),

4. Dans le dashboard, faites en sorte que le lien des revenus pointe sur /stats/revenus/ si ce n'est pas déjà le cas.

5. Maintenant, nous allons examiner le code du premier template revenus.html d'applistats :

```
{% extends 'applipizza/base.html' %}
{% load static %}
{% block title %}revenus sur 7 jours{% endblock %}
{% block main %}
<div id="content">
  <h2>chiffre d'affaires par jour sur les 7 derniers jours</h2>
  <!-- la balise canvas qui va recevoir le graphique généré par le script -->
  <canvas id="chart_CA" width="100" height="40"></canvas>

  <!-- insertion de chart.js -->
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

  <!-- le script qui génère le graphique en utilisant les variables transmises par la view -->
  <script>
    let canvas_chart_CA = document.getElementById('chart_CA');
    let ctx = canvas_chart_CA.getContext('2d');
    let CA_chart = new Chart(ctx, {
      type: 'bar',
      data: {
        labels: {{dates|safe}},
        datasets: [{
          label: 'CA',
          data: {{montants|safe}},
          borderWidth: 1
        }]
      },
      options: {
        scales: {
          y: {
            beginAtZero: true,
            max: {{CAmax}}
          }
        }
      }
    });
  </script>
</div>
{% endblock %}
```

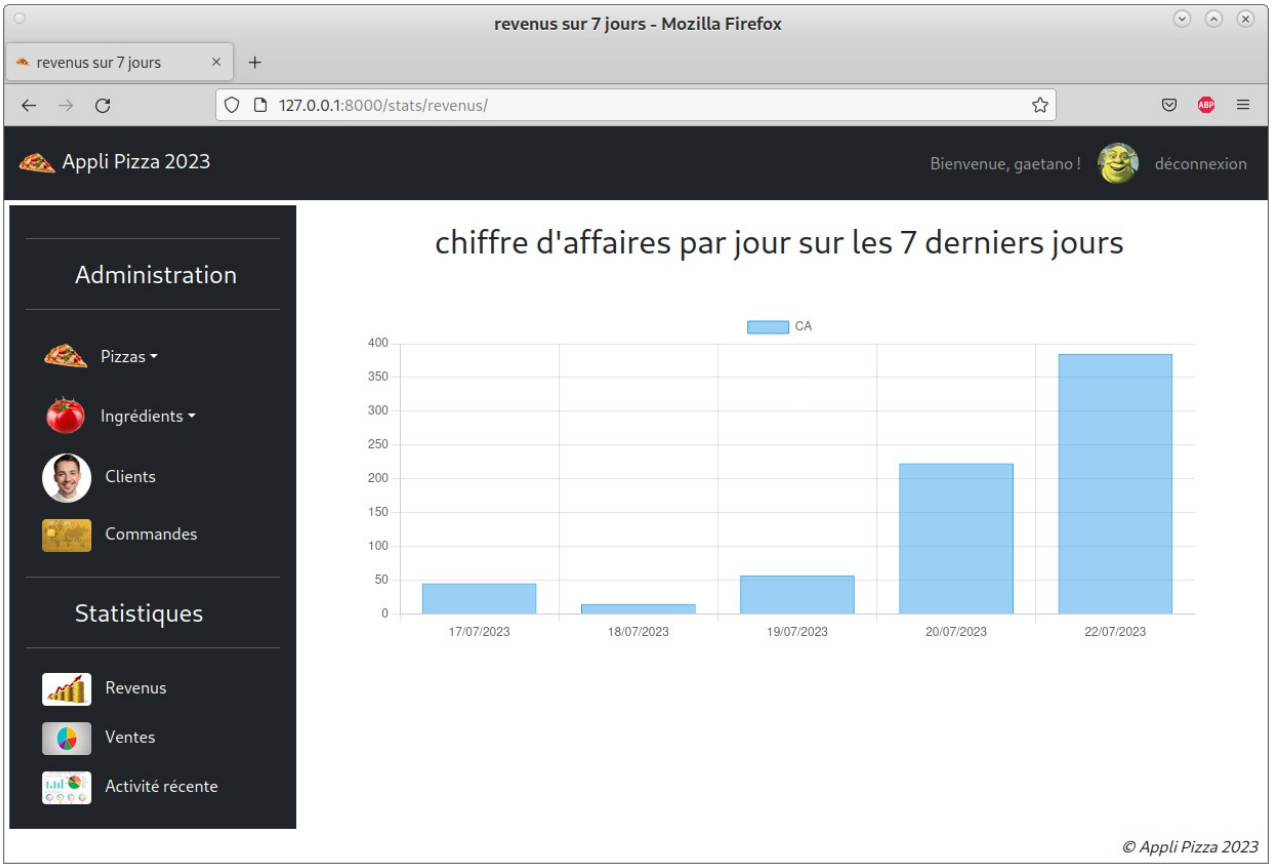
a. Le premier élément essentiel est la balise canvas qui recevra le graphique.

b. Le deuxième élément essentiel est l'insertion du code de chart.js

c. Enfin, il y a le script proprement dit qui crée le graphique dans le canvas, en utilisant les deux listes et le Camax transmis par la view.

Pour plus de documentation et d'exemples, voyez la documentation et les exemples de chart.js ou d'autres sites documentés : https://www.w3schools.com/ai/ai_chartjs.asp

6. Vérifiez que tout fonctionne bien. Vous devriez avoir un rendu de la forme suivante :



Le graphique des ventes de pizzas des 7 derniers jours

1. Pour la view ventes(request), le principe est exactement le même : on reprend la même requête, seules les variables à transmettre et l'algorithme changent :

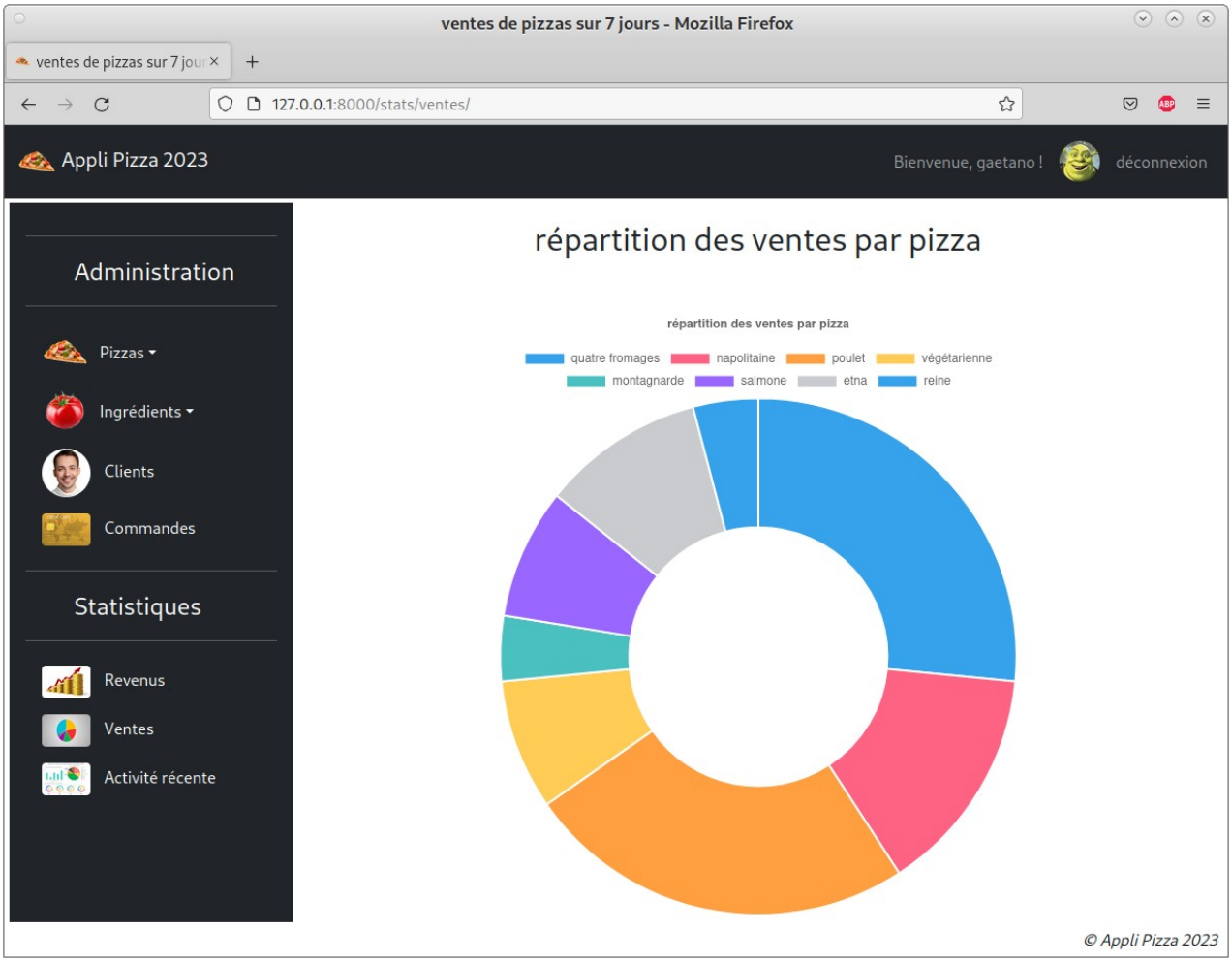
```python
# initialisation des variables qui seront utilisées
# par le script js via des variables de gabarit
nomsDesPizzas = []
ventesDesPizzas = []

# l'algorithme pour qu'à un même indice i on ait
# - le nom de la pizza dans la première liste
# - le nombre de ventes de cette pizza dans la deuxième liste
for commande in lesCommandesPayees :
    lesLignesPourCetteCommande = LigneCommande.objects.filter(commande = commande.idCommande)
    for ligne in lesLignesPourCetteCommande :
        nomDeLaPizza = ligne.pizza.nomPizza
        qte = ligne.quantite
        if nomDeLaPizza not in nomsDesPizzas :
            nomsDesPizzas.append(nomDeLaPizza)
            ventesDesPizzas.append(qte)
        else :
            n = nomsDesPizzas.index(nomDeLaPizza)
            ventesDesPizzas[n] += qte

# appel du template
return render(
    request,
    'applistats/ventes.html',
    {"noms" : nomsDesPizzas, "ventes" : ventesDesPizzas, "user" : user}
)
```

2. Vérifiez que vos url sont au point, et que le lien du dashboard l'est également.

3. Passons au code du template ventes.html d'applistats. Le titre change, et le type de graphique également :

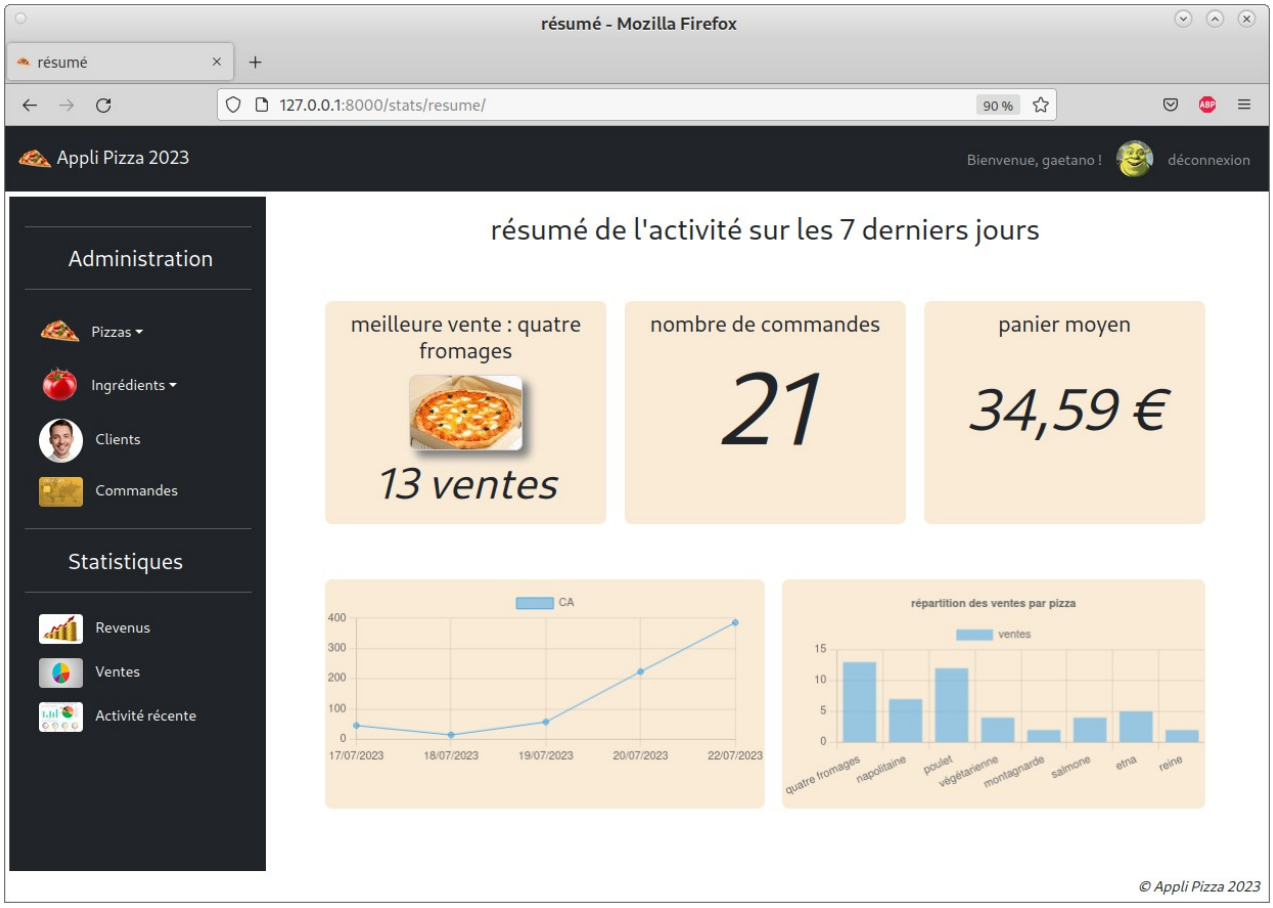
```
<!-- le script qui génère le graphique en utilisant les variables transmises par la view -->
<script>
  let canvas_chart_ventes = document.getElementById('chart_ventes');
  let ctx = canvas_chart_ventes.getContext('2d');
  let ventes_chart = new Chart(ctx, {
      type: "doughnut",
      data: {
          labels: {{noms|safe}},
          datasets: [{
              label : 'ventes',
              data: {{ventes|safe}}
          }]
      },
      options: {
          plugins: {
              legend: {
                  position: 'top',
              },
              title: {
                  display: true,
                  text: 'répartition des ventes par pizza'
              }
          }
      }
  });
</script>
```

4. Vérifiez que tout fonctionne bien. Vous devriez avoir un rendu de la forme suivante :


ventes de pizzas sur 7 jours - Mozilla Firefox
ventes de pizzas sur 7 jou
127.0.0.1:8000/stats/ventes/
Appli Pizza 2023
Bienvenue, gaetano !
déconnexion
Administration
Pizzas
Ingrédients
Clients
Commandes
Statistiques
Revenus
Ventes
Activité récente
répartition des ventes par pizza
répartition des ventes par pizza
quatre fromages
napolitaine
poulet
végétarienne
montagnarde
salmone
etna
reine
© Appli Pizza 2023

Le résumé de l'activité récente – travail en autonomie

Vous devez tout mettre en place pour arriver à ce visuel :



1. La meilleure vente est la pizza la plus vendue sur l'ensemble des commandes
2. Le nombre de commandes : depuis le début de l'activité
3. Idem pour le panier moyen
4. Les deux autres graphiques, faute d'imagination de ma part, sont une redite des ventes et des revenus. A vous d'imaginer mieux !