

Projet n°9: Développez une application Web en utilisant Django

OpenClassRooms - parcours python

Bérenger Ossété Gombé

23 septembre 2022

Sommaire

- 1 Introduction
- 2 Exigences
- 3 Démonstration
- 4 Méthodes et Outils
- 5 Conclusion

Je me présente

Bérenger Ossété Gombé

- Bac Scientifique (2013).
- Maîtrise en informatique (2017).
- Reconversion web chez OpenClassRooms (janvier 2022).

Contexte du projet

Projet n°9, Parcours Python

- Développer une application web en utilisant Django
- Utiliser le rendu côté serveur dans Django

Contexte fictif



LITReview

- Équipe
 - Alix, UX *designer*.
 - Sam, le directeur technique.
 - Nous sommes *lead* développeur Python.
- Objectif : → Développement d'une application web.

L'application

Deux types d'utilisateurs

- Les utilisateurs qui **demandent des critiques** de documents.
- Les utilisateurs qui **recherchent des documents** guidés par les critiques.

Exigences fonctionnelles

L'utilisateur peut :

- Gérer son **compte** (inscription, connexion, déconnexion).
- Consulter son **flux**.
- Créer un **ticket**.
- Publier une **critique** répondant ou non à un ticket.
- **Gérer** ses **tickets et critiques** (consultation, édition, suppression)
- **S'abonner** et se **désabonner** d'un utilisateur.

Démonstration du site

Compte utilisateur

- Création d'un nouveau compte.
- Connexion et déconnexion.

Publications

- Création d'un ticket.
- Création d'une critique.
- Création groupée.
- Visualisation de mes publications.
- Édition de mes publications.

Interactions avec les utilisateurs

- Suivre un utilisateur.
- Ne plus suivre un utilisateur.
- Impact sur le flux.

Outils logiciels

Bibliothèques

- Django (4.1)
- Flake8 (5.0.4)
- Django SASS (1.2.1)
- Pillow (9.2.0)

Architecture du projet

Découpage en applications

authentication, *publication*, *home*

Définition des principales urls

- *login/*
- *logout/*
- *signup/*
- *tickets/*
- *tickets/<int:id>/edit*
- *tickets/<int:id>/review*
- *tickets/review*
- *reviews/<int:id>/edit*
- */* (*news feed*)
- *social/*
- *me/*

Les modèles

Utilisateur

```
class User( AbstractUser ):  
    pass
```

Les modèles

Les Tickets

```
class Ticket(models.Model):
    title = models.CharField(max_length=128)
    description = models.TextField(max_length=248,
    blank=True)
    user = models.ForeignKey(to=settings.AUTH.USER.MODEL,
    on_delete=models.CASCADE)
    image = models.ImageField(null=True, blank=True)
    time_created = models.DateTimeField(auto_now_add=True)
```

Les Critiques

```
class Review(models.Model):
    ticket = models.ForeignKey(to=Ticket, on_delete=models.CASCADE)
    rating = models.PositiveSmallIntegerField(validators=[
        MinValueValidator(0),
        MaxValueValidator(5)
    ])
    user = models.ForeignKey(to=settings.AUTH.USER.MODEL,
    on_delete=models.CASCADE)
    headline = models.CharField(max_length=128)
    body = models.TextField(max_length=8192, blank=True)
```

Les modèles

Les Abonnements

```
class UserFollows(models.Model):
    user = models.ForeignKey(to=settings.AUTH_USER_MODEL,
                             on_delete=models.CASCADE,
                             related_name='following')
    followed_user = models.ForeignKey(to=settings.AUTH_USER_MODEL,
                                      on_delete=models.CASCADE,
                                      related_name='followed_by')

    class Meta:
        unique_together = ('user', 'followed_user')
```

Les contrôleurs

Utilisation de vues génériques

```
class SignupPage(generic.CreateView):  
    form_class = forms.UserForm  
    success_url = reverse_lazy('login')  
    template_name = 'authentication/signup.html'  
  
class LoginPage(LoginView):  
    form_class = forms.LoginForm  
    template_name = 'authentication/login.html'  
    next_page = reverse_lazy('signup')  
  
class LogoutPage(LogoutView):  
    next_page = reverse_lazy('login')
```

Les contrôleurs

Utilisation de vues basées sur des classes

```
class CreateTicketReview(LoginRequiredMixin, View):
    def get(self, request, id):
        ticket = get_object_or_404(models.Ticket, id=id)
        author = User.objects.get(id=ticket.user_id)
        form = forms.ReviewForm()


        return render(request, 'publication/ticket_review.html', {
            'ticket': ticket,
            'author': author,
            'form': form
        })

    def post(self, request, id):
        ticket = get_object_or_404(models.Ticket, id=id)
        author = User.objects.get(id=ticket.user_id)
        form = forms.ReviewForm(request.POST)

        if form.is_valid():
            review = form.save(commit=False)
            review.ticket = ticket
            review.user = request.user
            review.save()
            return redirect('me')
```

Les vues

Vues partielles



Star Wars 7 et 8
Le 15 septembre 2022 à 01:23
Un film à voir ?

☆☆☆☆ **Médiocre**
Le 15 septembre 2022 à 01:25



Star Wars 7 et 8
Par user le 15 septembre 2022
Un film à voir ?

Un film qui démarre trop lentement et dont le dénouement me laisse perplexe

```
{% include 'publication/ticket.html' with ticket=pub.data %}  
{% include 'publication/review.html' with review=pub.data %}
```


Les vues

Vues héritées

```
{% extends 'base.html' %}  
{% block body %}  
    <!-- Code de la vue -->  
{% endblock %}
```

Conclusion

Travail effectué

- Configuration d'un projet Django.
- Implémentation d'une interface simple côté *frontend*.
- Mise en place d'une base de données.
- Gestion des utilisateurs et des abonnements.
- Développement d'opérations CRUD sur les ressources du projet.

Conclusion

Pistes d'améliorations

- Adopter une vraie politique de test (→ Projet 11).
- Rendre le site robuste et sécurisé (→ Projet 12).
- Améliorer l'adaptation mobile de l'interface.
- Fluidifier l'expérience utilisateur avec des comportements côté navigateur (SPA par exemple).
- Gérer l'hébergement ainsi que le déploiement du site.
- Configurer et utiliser un serveur d'intégration continue.

Merci pour votre attention

- 1 Introduction
- 2 Exigences
- 3 Démonstration
- 4 Méthodes et Outils
- 5 Conclusion