**Structura Aplicației coworking\_nest**

**1. Obiectivele Aplicației**

* Utilizatorii pot vedea o listă de spații de coworking disponibile.
* Utilizatorii se pot autentifica/înregistra pentru a rezerva birouri.
* Administratorii pot gestiona spațiile (adăugare, editare, ștergere).
* Rezervările sunt stocate într-o bază de date, cu detalii precum data, utilizatorul și spațiul rezervat.
* Interfață simplă și intuitivă, cu pagini precum Home, About, Spații, Rezervări și Profil.

**2. Structura de Fișiere**

Vom extinde structura existentă a proiectului coworking\_nest pentru a include toate componentele necesare. Iată structura actualizată:

coworking\_nest/

├── coworking\_nest/

│ ├── \_\_init\_\_.py

│ ├── asgi.py

│ ├── settings.py

│ ├── urls.py

│ └── wsgi.py

├── deskdrop/

│ ├── migrations/

│ ├── templates/

│ │ ├── deskdrop/

│ │ │ ├── base.html

│ │ │ ├── home.html

│ │ │ ├── about.html

│ │ │ ├── spaces.html

│ │ │ ├── bookings.html

│ │ │ └── profile.html

│ ├── \_\_init\_\_.py

│ ├── admin.py

│ ├── apps.py

│ ├── models.py

│ ├── tests.py

│ ├── urls.py

│ └── views.py

├── static/

│ ├── css/

│ │ └── style.css

│ └── js/

│ └── scripts.js

├── manage.py

└── requirements.txt

**3. Structura Detaliată a Componentelor**

**3.1. Modele (deskdrop/models.py)**

Modelele definesc structura datelor stocate în baza de date (vom folosi SQLite implicit, dar poți trece la PostgreSQL ulterior).

* **Space**: Reprezintă un spațiu de coworking.
* **Booking**: Reprezintă o rezervare făcută de un utilizator.
* Vom folosi utilizatorii Django (django.contrib.auth.models.User) pentru autentificare.

from django.db import models

from django.contrib.auth.models import User

class Space(models.Model):

name = models.CharField(max\_length=100)

location = models.CharField(max\_length=200)

capacity = models.IntegerField()

price\_per\_day = models.DecimalField(max\_digits=10, decimal\_places=2) # Preț în RON

available = models.BooleanField(default=True)

created\_at = models.DateTimeField(auto\_now\_add=True)

def \_\_str\_\_(self):

return self.name

class Booking(models.Model):

user = models.ForeignKey(User, on\_delete=models.CASCADE)

space = models.ForeignKey(Space, on\_delete=models.CASCADE)

start\_date = models.DateField()

end\_date = models.DateField()

total\_price = models.DecimalField(max\_digits=10, decimal\_places=2) # Calculat automat

created\_at = models.DateTimeField(auto\_now\_add=True)

def \_\_str\_\_(self):

return f"{self.user.username} - {self.space.name} ({self.start\_date})"

**3.2. Views (deskdrop/views.py)**

Views-urile gestionează logica aplicației și randarea paginilor.

from django.shortcuts import render, redirect

from django.contrib.auth.decorators import login\_required

from .models import Space, Booking

from django.contrib import messages

from datetime import datetime

def home(request):

return render(request, 'deskdrop/home.html')

def about(request):

return render(request, 'deskdrop/about.html')

def spaces(request):

spaces = Space.objects.filter(available=True)

return render(request, 'deskdrop/spaces.html', {'spaces': spaces})

@login\_required

def bookings(request):

bookings = Booking.objects.filter(user=request.user)

return render(request, 'deskdrop/bookings.html', {'bookings': bookings})

@login\_required

def profile(request):

return render(request, 'deskdrop/profile.html')

@login\_required

def book\_space(request, space\_id):

space = Space.objects.get(id=space\_id)

if request.method == 'POST':

start\_date = request.POST['start\_date']

end\_date = request.POST['end\_date']

start = datetime.strptime(start\_date, '%Y-%m-%d')

end = datetime.strptime(end\_date, '%Y-%m-%d')

days = (end - start).days

total\_price = days \* space.price\_per\_day

booking = Booking.objects.create(

user=request.user,

space=space,

start\_date=start\_date,

end\_date=end\_date,

total\_price=total\_price

)

space.available = False

space.save()

messages.success(request, 'Rezervare realizată cu succes!')

return redirect('bookings')

return render(request, 'deskdrop/spaces.html', {'space': space})

**3.3. URL-uri (deskdrop/urls.py și coworking\_nest/urls.py)**

URL-urile definesc rutele aplicației.

* **deskdrop/urls.py**:

from django.urls import path

from . import views

urlpatterns = [

path('', views.home, name='home'),

path('about/', views.about, name='about'),

path('spaces/', views.spaces, name='spaces'),

path('bookings/', views.bookings, name='bookings'),

path('profile/', views.profile, name='profile'),

path('book/<int:space\_id>/', views.book\_space, name='book\_space'),

]

coworking\_nest/urls.py:

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

path('admin/', admin.site.urls),

path('', include('deskdrop.urls')),

path('accounts/', include('django.contrib.auth.urls')), # Pentru autentificare

]

**3.4. Template-uri (deskdrop/templates/deskdrop/)**

* **base.html**: Template de bază pentru layout.

<!DOCTYPE html>

<html lang="ro">

<head>

<meta charset="UTF-8">

<title>NestDrop - Spații de Coworking</title>

<link rel="stylesheet" href="{% static 'css/style.css' %}">

</head>

<body>

<nav>

<a href="{% url 'home' %}">Acasă</a>

<a href="{% url 'about' %}">Despre</a>

<a href="{% url 'spaces' %}">Spații</a>

{% if user.is\_authenticated %}

<a href="{% url 'bookings' %}">Rezervări</a>

<a href="{% url 'profile' %}">Profil</a>

<a href="{% url 'logout' %}">Deconectare</a>

{% else %}

<a href="{% url 'login' %}">Autentificare</a>

<a href="{% url 'signup' %}">Înregistrare</a>

{% endif %}

</nav>

<div class="container">

{% block content %}

{% endblock %}

</div>

</body>

</html>

* **home.html**, **about.html**, **spaces.html**, **bookings.html**, **profile.html**: Extind base.html și conțin conținut specific (ex. listă de spații, formular de rezervare).

**3.5. Setări (coworking\_nest/settings.py)**

Adaugă configurările necesare pentru autentificare, static files și fus orar (România).

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'deskdrop',

]

STATIC\_URL = '/static/'

STATICFILES\_DIRS = [BASE\_DIR / "static"]

TIME\_ZONE = 'Europe/Bucharest'

USE\_TZ = True

LOGIN\_REDIRECT\_URL = 'home'

LOGOUT\_REDIRECT\_URL = 'home'

3.6. Fișier Artifact cu Structura Completă

coworking\_nest/

├── coworking\_nest/

│ ├── init.py

│ ├── asgi.py

│ ├── settings.py

│ ├── urls.py

│ └── wsgi.py

├── deskdrop/

│ ├── migrations/

│ ├── templates/

│ │ ├── deskdrop/

│ │ │ ├── base.html

│ │ │ ├── home.html

│ │ │ ├── about.html

│ │ │ ├── spaces.html

│ │ │ ├── bookings.html

│ │ │ └── profile.html

│ ├── init.py

│ ├── admin.py

│ ├── apps.py

│ ├── models.py

│ ├── tests.py

│ ├── urls.py

│ └── views.py

├── static/

│ ├── css/

│ │ └── style.css

│ └── js/

│ └── scripts.js

├── manage.py

└── requirements.txt

**Pași Următori**

1. **Instalează dependențele**: Creează un requirements.txt cu django>=4.0 și instalează-le (pip install -r requirements.txt).
2. **Rulează migrațiile**: python manage.py makemigrations și python manage.py migrate pentru a crea baza de date.
3. **Creează un superuser**: python manage.py createsuperuser pentru a accesa panoul de admin.
4. **Testează aplicația**: python manage.py runserver și accesează http://127.0.0.1:8000.

Aplicația ta „NestDrop” are acum o structură funcțională! Poți adăuga spații prin panoul de admin, iar utilizatorii se pot autentifica pentru a rezerva birouri. Dacă vrei să extindem funcționalitățile (ex. sistem de plată, notificări), spune-mi!