Session 2: Basics of Django

Xander Warszawski

GitHub

IEEE Student Branch
KU Leuven Campus Brugge

0

Contents of last session

1. Introduction
2. Installing tools
3. Git and GitHub
4. Django setup
5. Docker setup

GitHub

IEEE Student Branch
KU Leuven Campus Brugge

1

Contents of this session

1. Django Architecture
2. Environment variables
3. Pages app
4. Static assets
5. Common app
6. Albums app

GitHub IEEE Student Branch KU Leuven Campus Brugge

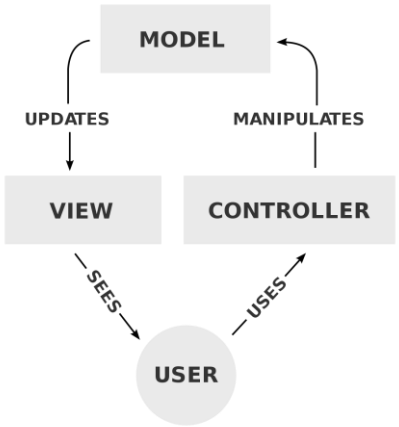
2

1. Django Architecture

3

MVC: Model-View-Controller

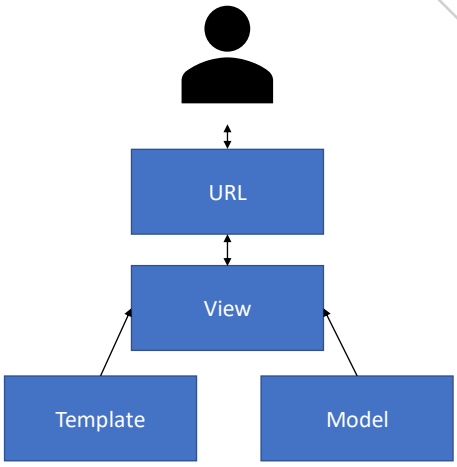
- Software architectural pattern
- Used by all web frameworks
- **Model**: manages data & core business logic
- **View**: renders data from model
- **Controller**: accepts user input and performs application logic



GitHub IEEE Student Branch KU Leuven Campus Brugge

MVTU: Model-View-Template-Url

- Model: manages data & core business logic
- **View: describes which data is sent to user**
- **Template: presentation of data**
- **Url: Regex components configured to a view**
- Flow from request to response:
  1. User requests certain URL
  2. In urls.py a url pattern is found that matches it, this is linked to one view
  3. A view combines data from model (from models.py) and styling (*.html file)
  4. A response is returned



GitHub IEEE Student Branch KU Leuven Campus Brugge

Generated files by startproject

- **manage.py**: command line utility for interaction with django
- **config/**
  - **__init__.py**: empty file that tells python this is a package
  - **settings.py**: settings for this project
  - **urls.py**: Main url declarations
  - **asgi.py**: Async Server Gateway Interface: useful for websockets (used for e.g. chatting, real-time apps)
  - **wsgi.py**: Web Server Gateway Interface: we will use this one

GitHub IEEE Student Branch KU Leuven Campus Brugge

Contents of settings.py

- **BASE_DIR**: base directory of project
- **SECRET_KEY**: used to provide cryptographic signing
- **DEBUG**: when enabled will provide detailed error pages, unsecure in prod
- **ALLOWED_HOSTS**: host/domains that can serve this site, to prevent Host header attacks
- **INSTALLED_APPS**: list of all apps that are enabled in this project
- **MIDDLEWARE**: list of enabled middleware; these are low-level “plugins”
- **ROOT_URLCONF**: represents full python import path to root URLconf
- **TEMPLATES**: contains all settings for templates
- **WSGI_APPLICATION**: path to WSGI Application that Django servers will use
- **DATABASES**: database settings
- **AUTH_PASSWORD_VALIDATORS**: validators used to check password strength
- **STATIC_URL**: url to use when referring to static files
- **DEFAULT_AUTO_FIELD**: default primary key field

GitHub IEEE Student Branch KU Leuven Campus Brugge

Generated files by startapp

- **__init__.py**
- **admin.py**: config file for built-in Django admin app
- **apps.py**: config file for app itself
- **migrations/**: keeps track of changes to models, stays synced with db
- **models.py**: where we define our db models
- **tests.py**: for app-specific tests
- **views.py**: where we handle request/response logic

GitHub IEEE Student Branch KU Leuven Campus Brugge

Questions?

GitHub IEEE Student Branch KU Leuven Campus Brugge

2. Environment Variables

10

Environment variables

- Variables that will be different according to the environment
- Eg:
  - Database_url
  - Debug
  - Secret_Key
  - …
- Part of the Twelve-Factor App Design: https://12factor.net/

11

Git checkpoint

1. Create new feature branch

GitHub IEEE Student Branch KU Leuven Campus Brugge

Environment variables

1. Pip install **environs[django]** & freeze
2. Update **docker-compose.yml**
    1. Take <value> for DJANGO_SECRET_KEY from **config/settings.py**
        1. If $ is present in secret key: add another $; $ → $$

GitHub IEEE Student Branch KU Leuven Campus Brugge

docker-compose.yml

```
depends_on:
  - db
environment:
  - "DJANGO_SECRET_KEY=<value>"
  - "DJANGO_DEBUG=True"
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

Environment variables

1. Pip install **environs[django]** & freeze
2. Update **docker-compose.yml**
3. Update **config/settings.py**

GitHub IEEE Student Branch KU Leuven Campus Brugge

config/settings.py

```
from environs import Env
from django.core.management.utils import get_random_secret_key

env = Env()
env.read_env()

SECRET_KEY = env("DJANGO_SECRET_KEY", get_random_secret_key())

DEBUG = env.bool("DJANGO_DEBUG", default=False)

ALLOWED_HOSTS = env("DJANGO_ALLOWED_HOSTS", "127.0.0.1,localhost").split(",")

DATABASES = {
    'default': env.dj_db_url("DATABASE_URL", default="postgres://postgres@db/postgres")
}
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

Git checkpoint

1. Commit changes
2. Create PR

GitHub IEEE Student Branch KU Leuven Campus Brugge

Questions?

GitHub IEEE Student Branch KU Leuven Campus Brugge

18

3. Pages app

19

Flow for creating an application

1. Figure out what you will make
2. Design your models before coding them, think about relationships
    1. One-to-one: Table 1 has ID that points to entry in Table 2
    2. One-to-many: Table 2 has ID that points to entry in Table 1
    3. Many-to-many: Linking table that has IDs of Table 1 and Table 2
3. Code models
4. Code views
5. Code templates
6. Code urls
7. Write tests (with a bit of practice this can be step 3 → TDD)

GitHub IEEE Student Branch KU Leuven Campus Brugge

Music store: model design

- We sell albums
- One album has
    - One or more artists
    - Cover (image)
    - Price (decimal)
    - Record label (string)
- One artist has:
    - One or more albums
    - Artist name (string)
- CustomUser has:
    - One or more paid albums
- So:
    - Album-Artist: many-to-many relationship (1 album can have +1 artists, 1 artist can have +1 albums)
    - CustomUser-Album: many-to-many (1 user can have +1 albums, 1 album can be paid by +1 users)

GitHub IEEE Student Branch KU Leuven Campus Brugge

Some good practices

- When structuring your code, work by:
  - Fat models, utility modules, thin views, stupid templates
- **Fat models & thin views**: business logic in models instead of views
- **Utility modules**: create (a) separate module(s) for shared code
  - Eg app named **common** or core
- **Stupid templates**: templates should absolutely NOT contain logic, they should only 'print' data
- Also:
  - Try to keep **apps as small as possible**, they should be focused on one task

GitHub IEEE Student Branch KU Leuven Campus Brugge

Git checkpoint

1. Create new feature branch

GitHub IEEE Student Branch KU Leuven Campus Brugge

Pages app

1. Execute: **docker-compose exec web python manage.py startapp pages**
2. Configure templates:
    1. Update **config/settings.py**

GitHub IEEE Student Branch KU Leuven Campus Brugge

config/settings.py (1)

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'pages.apps.PagesConfig'
]
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

config/settings.py (2)

```
TEMPLATES = [
    {
        …
        'DIRS': [BASE_DIR / "templates"],
        …
    },
]
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

Pages app

1. Execute: **docker-compose exec web python manage.py startapp pages**
2. Configure templates:
   1. Update **config/settings.py**
   2. Add **templates/_base.html**

GitHub IEEE Student Branch KU Leuven Campus Brugge

templates/_base.html (1)

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>{% block title %}Music store{% endblock title %}</title>
    </head>
    <body>
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

28

templates/_base.html (2)

```
        <div>
            <ul>
                <li><a href="">Home</a></li>
                <li><a href="">Music</a></li>
                <li><a href="">My music</a></li>
                <li><a href="">Log out</a></li>
                <li><a href="">Log in</a></li>
                <li><a href="">Sign up</a></li>
            </ul>
        </div>
        <div>
        {% block content %}
        {% endblock content %}
        </div>
    </body>
</html>
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

29

Pages app

1. Execute: **docker-compose exec web python manage.py startapp pages**
2. Configure templates
3. Add **templates/pages/home.html**

GitHub IEEE Student Branch KU Leuven Campus Brugge

templates/pages/home.html

```
{% extends '_base.html' %}

{% block title %}Home{% endblock title %}

{% block content %}
    <h1>Welcome to the music store!</h1>
{% endblock content %}
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

Pages app

1. Execute: **docker-compose exec web python manage.py startapp pages**
2. Configure templates
3. Add **templates/pages/home.html**
4. Update **pages/views.py**

GitHub IEEE Student Branch KU Leuven Campus Brugge

pages/views.py

```
from django.views.generic import TemplateView

class HomePageView(TemplateView):
    template_name = "pages/home.html"
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

Pages app

1. Execute: **docker-compose exec web python manage.py startapp pages**
2. Configure templates
3. Add **templates/pages/home.html**
4. Update **pages/views.py**
5. Add **pages/urls.py**

GitHub IEEE Student Branch KU Leuven Campus Brugge

pages/urls.py

```
from django.urls import path
from .views import HomePageView

urlpatterns = [
    path('', HomePageView.as_view(), name="home")
]
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

Pages app

1. Execute: **docker-compose exec web python manage.py startapp pages**
2. Configure templates
3. Add **templates/pages/home.html**
4. Update **pages/views.py**
5. Add **pages/urls.py**
6. Update **config/urls.py**

GitHub IEEE Student Branch KU Leuven Campus Brugge

config/urls.py

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('pages.urls'))
]
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

Configure tests

1. Check if Testing icon is visible on left bar in VSCode
    1. Otherwise Right-click > Testing, to make it show up
    2. Then Configure Python Tests
    3. Pick: 'pytest'
    4. Pick: Root directory
    5. Add **.vscode** to **.gitignore**
    6. Pip install **pytest-django** & freeze
    7. Add **pytest.ini** next to manage.py

GitHub IEEE Student Branch KU Leuven Campus Brugge

pytest.ini

```
[pytest]
DJANGO_SETTINGS_MODULE = config.settings.test
python_files = tests.py test_*.py *_tests.py
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

Configure tests

1. Check if Testing icon is visible on left bar in VSCode
    1. Otherwise Right-click > Testing, to make it show up
    2. Then Configure Python Tests
    3. Pick: 'pytest'
    4. Pick: Root directory
    5. Add **.vscode** to **.gitignore**
    6. Pip install **pytest-django** & freeze
    7. Add **pytest.ini** next to manage.py
    8. Add **settings** folder in **config**
        1. Copy **settings.py** as **base.py** into this folder
        2. Add **__init__.py**

GitHub IEEE Student Branch KU Leuven Campus Brugge

config/settings/__init__.py

```
from .base import *
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

Configure tests

1. Check if Testing icon is visible on left bar in VSCode
    1. Otherwise Right-click > Testing, to make it show up
    2. Then Configure Python Tests
    3. Pick: 'pytest'
    4. Pick: Root directory
    5. Add **.vscode** to **.gitignore**
    6. Pip install **pytest-django** & freeze
    7. Add **pytest.ini** next to manage.py
    8. Add **settings** folder in **config**
        1. Copy **settings.py** as **base.py** into this folder
            1. Change **BASE_DIR** to: `Path(__file__).resolve().parent.parent.parent`
        2. Add **__init__.py**
        3. Add **test.py**

GitHub IEEE Student Branch KU Leuven Campus Brugge

config/settings/test.py

```
from .base import *

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': 'testdb'
    }
}

DEBUG = False

# Needed for later
SECURE_SSL_REDIRECT = False
SECURE_HSTS_SECONDS = 0
SECURE_HSTS_INCLUDE_SUBDOMAINS = False
SECURE_HSTS_PRELOAD = False
SESSION_COOKIE_SECURE = False
CSRF_COOKIE_SECURE = False
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

How to ‘design’ tests?

1. **Arrange**
   - Create your data; eg. Fill your db
2. **Act**
   - Perform your action; eg. When testing update of a model, execute that update method
3. **Assert**
   - Check if your result matches your expected result

GitHub IEEE Student Branch KU Leuven Campus Brugge

Types of TestCases

- **SimpleTestCase**:
  - No db access
- **TestCase**
  - Rolls back db changes using transactions
- **TransactionTestCase**
  - Rolls back db changes by flushing all tables
- **LiveServerTestCase**
  - Based on TransactionTestCase, also launches live server thread (for eg. Selenium)
- Want to **test business logic**? Use **SimpleTestCase**
- Want to **test functionality that needs a db**? Use **TestCase**
- TransactionTestCase and LiveServerTestCase are slower than the others

GitHub IEEE Student Branch KU Leuven Campus Brugge

Types of setup methods

- **setUpClass**
  - Runs once at start of case class
- **setUp**
  - Runs at start of each test
- **setUpTestData**
  - Like setUpClass but rolls back transactions done on this data during yesys

GitHub IEEE Student Branch KU Leuven Campus Brugge

Pages app

1. Execute: **docker-compose exec web python manage.py startapp pages**
2. Configure templates
3. Add **templates/pages/home.html**
4. Update **pages/views.py**
5. Add **pages/urls.py**
6. Update **config/urls.py**
7. Update **pages/tests.py**

GitHub IEEE Student Branch KU Leuven Campus Brugge

pages/tests.py (1)

```
import pytest
from django.test import SimpleTestCase
from django.urls import reverse

@pytest.mark.django_db
class HomePageTests(SimpleTestCase):
    def setUp(self):
        # Arrange
        url = reverse("home")
        # Act
        self.response = self.client.get(url)
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

pages/tests.py (2)

```
def test_url_exists_at_correct_location(self):
    # self.assertEqual(self.response.status_code, 200)
    # option above is used frequently in Django, better to use option below when using pytest
    # Assert
    assert self.response.status_code == 200

def test_homepage_template(self):
    # self.assertTemplateUsed(self.response, "pages/home.html")
    # same remark as before
    assert "pages/home.html" in [x.name for x in self.response.templates]
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

Pages app

1. Execute: **docker-compose exec web python manage.py startapp pages**
2. Configure templates
3. Add **templates/pages/home.html**
4. Update **pages/views.py**
5. Add **pages/urls.py**
6. Update **config/urls.py**
7. Update **pages/tests.py**
8. Update **templates/_base.html**

GitHub IEEE Student Branch KU Leuven Campus Brugge

templates/_base.html

```
<li><a href="{% url 'home' %}">Home</a></li>
<li><a href="">Music</a></li>
<li><a href="">My music</a></li>
<li><a href="">Log out</a></li>
<li><a href="">Log in</a></li>
<li><a href="">Sign up</a></li>
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

Git checkpoint

1. Commit changes
2. Create PR
3. Tests will fail:
   1. Edit run on 'Run Tests' in action from **python manage.py test** to **pytest**
   2. Update branch in PR
4. Useful **recommend**:
   1. Pip install **black** and **isort** & freeze
   2. Run before each PR (can also be added to GitHub action):
      - **black .**
      - **isort .**

GitHub IEEE Student Branch KU Leuven Campus Brugge

Questions?

GitHub IEEE Student Branch KU Leuven Campus Brugge

4. Static assets

54

Static assets

- Static assets?
  - CSS
  - JavaScript
  - Images
- Local: auto served

55

GitHub

IEEE Student Branch
KU Leuven Campus Brugge

55

Bootstrap

- In templates you can freestyle a bit, I will provide the bare minimum
- Definitely try using bootstrap for these: https://getbootstrap.com/
- Also use crispy-forms: https://django-crispy-forms.readthedocs.io/

GitHub IEEE Student Branch KU Leuven Campus Brugge

Git checkpoint

1. Create new feature branch

GitHub IEEE Student Branch KU Leuven Campus Brugge

Image

1. Update **config/settings/base.py**
2. Add image to **static/images**
3. Add it to **home.html** using **{% static 'images/<NAME>' %}**
    1. Don't forget to add **{% load static %}**

GitHub IEEE Student Branch KU Leuven Campus Brugge

config/settings/base.py

```
STATIC_URL = '/static/'
STATICFILES_DIRS = [BASE_DIR / "static"]
STATIC_ROOT = BASE_DIR / "staticfiles"
STATICFILES_STORAGE =
"django.contrib.staticfiles.storage.StaticFilesStorage"
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

Production development

1. Execute: **docker-compose exec web python manage.py collectstatic**

GitHub IEEE Student Branch KU Leuven Campus Brugge

Git checkpoint

1. Commit changes
2. Create PR

GitHub IEEE Student Branch KU Leuven Campus Brugge

Questions?

GitHub

IEEE Student Branch
KU Leuven Campus Brugge

69

5. Common app

70

Git checkpoint

1. Create new feature branch

GitHub IEEE Student Branch KU Leuven Campus Brugge

Common app

1. Execute: **docker-compose exec web python manage.py startapp common**
2. Update **common/models.py**

GitHub IEEE Student Branch KU Leuven Campus Brugge

common/models.py

```
from django.db import models

class TimeStampedModel(models.Model):
    created = models.DateTimeField(auto_now_add=True)
    modified = models.DateTimeField(auto_now=True)

    class Meta:
        abstract = True
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

Git checkpoint

1. Commit changes
2. Create PR

GitHub IEEE Student Branch KU Leuven Campus Brugge

Questions?

GitHub

IEEE Student Branch
KU Leuven Campus Brugge

75

6. Albums app

76

Git checkpoint

1. Create new feature branch

GitHub IEEE Student Branch KU Leuven Campus Brugge

Albums app

1. Pip install **pillow** & freeze
2. Execute: **docker-compose exec web python manage.py startapp albums**
3. Update **albums/models.py**

GitHub IEEE Student Branch KU Leuven Campus Brugge

albums/models.py (1)

```
import uuid
from django.db import models
from common.models import TimeStampedModel
from django.urls import reverse

class Album(TimeStampedModel):
    id = models.UUIDField(
        primary_key=True,
        default=uuid.uuid4,
        editable=False
    )
    title = models.CharField(max_length=200)
    cover = models.ImageField(upload_to="covers/", blank=True)
    price = models.DecimalField(max_digits=6, decimal_places=2)
    record_label = models.CharField(max_length=200)
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

albums/models.py (2)

```
    def artist(self):
        return self.artist_set.all()[0]

    def artists(self):
        return self.artist_set.all()

    def __str__(self):
        return self.title

    def get_absolute_url(self):
        return reverse("album_detail", args=[str(self.id)])
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

albums/models.py (3)

```
class Artist(TimeStampedModel):
    name = models.CharField(max_length=200)
    albums = models.ManyToManyField(Album)

    def __str__(self):
        return self.name
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

Albums app

1. Execute: **docker-compose exec web python manage.py startapp albums**
2. Update **albums/models.py**
3. Update **albums/admin.py**

GitHub IEEE Student Branch KU Leuven Campus Brugge

albums/admin.py

```
from django.contrib import admin
from .models import Album, Artist

class AlbumAdmin(admin.ModelAdmin):
    list_display = ("title", "cover", "price", "record_label", "artists")

class ArtistAdmin(admin.ModelAdmin):
    list_display = ("name",)

admin.site.register(Album, AlbumAdmin)
admin.site.register(Artist, ArtistAdmin)
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

Albums app

1. Execute: **docker-compose exec web python manage.py startapp albums**
2. Update **albums/models.py**
3. Update **albums/admin.py**
4. Update **albums/views.py**

GitHub IEEE Student Branch KU Leuven Campus Brugge

albums/views.py (1)

```
from django.views.generic import ListView, DetailView
from django.db.models import Q

from .models import Album


class AlbumListView(ListView):
    paginate_by = 10
    model = Album
    context_object_name = "album_list"
    template_name = "albums/album_list.html"
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

albums/views.py (2)

```
    def get_queryset(self):
        query = self.request.GET.get("q")

        if query is None:
            return self.model.objects.all()

        return self.model.objects.filter(
            Q(title__icontains=query) | Q(artist__name__icontains=query)
        )

class AlbumDetailView(DetailView):
    model = Album
    context_object_name = "album"
    template_name = "albums/album_detail.html"
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

Albums app

1. Execute: **docker-compose exec web python manage.py startapp albums**
2. Update **albums/models.py**
3. Update **albums/admin.py**
4. Update **albums/views.py**
5. Create **templates/albums/album_list.html**
6. Create **templates/albums/album_detail.html**

GitHub IEEE Student Branch KU Leuven Campus Brugge

templates/albums/album_list.html (1)

```
{% extends "_base.html" %}

{% block title %}Albums{% endblock title %}

{% block content %}
    <form action="{% url 'album_list' %}" method="get">
        <input type="search" name="q" placeholder="Search">
        <button type="submit">Search</button>
    </form>

    {% for album in album_list %}
        <div>
            <h2><a href="{{ album.get_absolute_url }}">
                {{ album.title }} – {{ album.artist.name }}
            </a></h2>
        </div>
    {% endfor %}
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

templates/albums/album_list.html (2)

```
    <div class="pagination">
        <span class="step-links">
            {% if page_obj.has_previous %}
                <a href="?page=1">&laquo; first</a>
                <a href="?page={{ page_obj.previous_page_number }}">previous</a>
            {% endif %}

            <span class="current">
                Page {{ page_obj.number }} of {{ page_obj.paginator.num_pages }}.
            </span>

            {% if page_obj.has_next %}
                <a href="?page={{ page_obj.next_page_number }}">next</a>
                <a href="?page={{ page_obj.paginator.num_pages }}">last &raquo;</a>
            {% endif %}
        </span>
    </div>
{% endblock content %}
```


89

templates/albums/album_detail.html (1)

```
{% extends "_base.html" %}

{% block title %}{{ album.title }}{% endblock title %}

{% block content %}
    <div>
        {% if album.cover %}
            <img src="{{ album.cover.url }}" alt="{{ album.title }}">
        {% endif %}
        <h2>{{ album.title }}</h2>
```


90

templates/albums/album_detail.html (2)

```
        <p>Artist(s):</p>
        <ul>
            {% for artist in album.artists %}
                <li>{{ artist.name }}</li>
            {% endfor %}
        </ul>
        <p>Record label: {{ album.record_label }}</p>
        <p>Price: €{{ album.price }}</p>
    </div>
{% endblock content %}
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

Albums app

1. Execute: **docker-compose exec web python manage.py startapp albums**
2. Update **albums/models.py**
3. Update **albums/admin.py**
4. Update **albums/views.py**
5. Create **templates/albums/album_list.html**
6. Create **templates/albums/album_detail.html**
7. Add **albums/urls.py**

GitHub IEEE Student Branch KU Leuven Campus Brugge

albums/urls.py

```
from django.urls import path

from .views import AlbumListView, AlbumDetailView

urlpatterns = [
    path("", AlbumListView.as_view(), name="album_list"),
    path("<uuid:pk>/", AlbumDetailView.as_view(), name="album_detail")
]
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

Albums app

1. Execute: **docker-compose exec web python manage.py startapp albums**
2. Update **albums/models.py**
3. Update **albums/admin.py**
4. Update **albums/views.py**
5. Create **templates/albums/album_list.html**
6. Create **templates/albums/album_detail.html**
7. Update **albums/urls.py**
8. Update **config/settings/base.py**

GitHub IEEE Student Branch KU Leuven Campus Brugge

config/settings/base.py

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'pages.apps.PagesConfig',
    'albums.apps.AlbumsConfig'
]
…
MEDIA_URL = "/media/"
MEDIA_ROOT = BASE_DIR / "media"
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

95

Albums app

1. Execute: **docker-compose exec web python manage.py startapp albums**
2. Update **albums/models.py**
3. Update **albums/admin.py**
4. Update **albums/views.py**
5. Create **templates/albums/album_list.html**
6. Create **templates/albums/album_detail.html**
7. Update **albums/urls.py**
8. Update **config/settings.py**
9. Update **config/urls.py**

GitHub IEEE Student Branch KU Leuven Campus Brugge

96

config/urls.py

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('pages.urls')),
    path('albums/', include('albums.urls'))
] + static(
    settings.MEDIA_URL, document_root=settings.MEDIA_ROOT
)
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

Albums app

1. Execute: **docker-compose exec web python manage.py startapp albums**
2. Update **albums/models.py**
3. Update **albums/admin.py**
4. Update **albums/views.py**
5. Create **templates/albums/album_list.html**
6. Create **templates/albums/album_detail.html**
7. Update **albums/urls.py**
8. Update **config/settings.py**
9. Update **config/urls.py**
10. Execute: **docker-compose exec web python manage.py makemigrations**

GitHub IEEE Student Branch KU Leuven Campus Brugge

Examine migrations: albums/migrations/0001_initial.py (1)

```
from django.db import migrations, models
import uuid


class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

Examine migrations: albums/migrations/0001_initial.py (2)

```
operations = [
    migrations.CreateModel(
        name='Album',
        fields=[
            ('created', models.DateTimeField(auto_now_add=True)),
            ('modified', models.DateTimeField(auto_now=True)),
            ('id', models.UUIDField(default=uuid.uuid4, editable=False, primary_key=True, serialize=False)),
            ('title', models.CharField(max_length=200)),
            ('cover', models.ImageField(blank=True, upload_to='covers/')),
            ('price', models.DecimalField(decimal_places=2, max_digits=6)),
            ('record_label', models.CharField(max_length=200)),
        ],
        options={
            'abstract': False,
        },
    ),
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

Examine migrations: albums/migrations/0001_initial.py (3)

```
        migrations.CreateModel(
            name='Artist',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('created', models.DateTimeField(auto_now_add=True)),
                ('modified', models.DateTimeField(auto_now=True)),
                ('name', models.CharField(max_length=200)),
                ('albums', models.ManyToManyField(to='albums.album')),
            ],
            options={
                'abstract': False,
            },
        ),
    ]
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

Migrations in production

1. Create your migrations in dev & run these on your PC
    1. Good practice to examine your migrations
2. No issues? Commit migrations
3. Backup production db
4. Execute your migrations
5. Check if everything is ok

GitHub IEEE Student Branch KU Leuven Campus Brugge

Albums app

1. Execute: **docker-compose exec web python manage.py startapp albums**
2. Update **albums/models.py**
3. Update **albums/admin.py**
4. Update **albums/views.py**
5. Create **templates/albums/album_list.html**
6. Create **templates/albums/album_detail.html**
7. Update **albums/urls.py**
8. Update **config/settings.py**
9. Update **config/urls.py**
10. Execute: **docker-compose exec web python manage.py makemigrations**
11. Execute: **docker-compose exec web python manage.py migrate**
12. Update **albums/tests.py**

GitHub  IEEE Student Branch KU Leuven Campus Brugge

albums/tests.py (1)

```
import pytest
from django.test import TestCase
from django.urls import reverse

from .models import Album, Artist


@pytest.mark.django_db
class AlbumTests(TestCase):
```


GitHub  IEEE Student Branch KU Leuven Campus Brugge

albums/tests.py (2)

```
@classmethod
    def setUpTestData(cls):
        cls.artist = Artist.objects.create(
            name = "Beartooth"
        )

        cls.album = Album.objects.create(
            title = "Below",
            price = "15.99",
            record_label = "Red Bull Records"
        )
        cls.artist.albums.add(cls.album)
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

albums/tests.py (3)

```
    def test_album_listing(self):
        assert f"{self.album.title}" == "Below"
        assert f"{self.album.artists()[0].name}" == "Beartooth"
        assert f"{self.album.price}" == "15.99"
        assert f"{self.album.record_label}" == "Red Bull Records"

    def test_album_list_view(self):
        response = self.client.get(reverse("album_list"))
        assert response.status_code == 200
        assert "Beartooth" in response.rendered_content
        assert "albums/album_list.html" in [x.name for x in response.templates]
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

albums/tests.py (4)

```
def test_album_detail_view(self):
    response = self.client.get(self.album.get_absolute_url())
    assert response.status_code == 200
    assert "Beartooth" in response.rendered_content
    assert "albums/album_detail.html" in [x.name for x in response.templates]
```


GitHub IEEE Student Branch KU Leuven Campus Brugge

Albums app

1. Execute: **docker-compose exec web python manage.py startapp albums**
2. Update **albums/models.py**
3. Update **albums/admin.py**
4. Update **albums/views.py**
5. Create **templates/albums/album_list.html**
6. Create **templates/albums/album_detail.html**
7. Update **albums/urls.py**
8. Update **config/settings.py**
9. Update **config/urls.py**
10. Execute: **docker-compose exec web python manage.py makemigrations**
11. Execute: **docker-compose exec web python manage.py migrate**
12. Update **albums/tests.py**
13. Update **templates/_base.html**

GitHub IEEE Student Branch KU Leuven Campus Brugge

templates/pages/_base.html

```
<li><a href="{% url 'home' %}">Home</a></li>
<li><a href="{% url 'album_list' %}">Music</a></li>
<li><a href="">My music</a></li>
<li><a href="">Log out</a></li>
<li><a href="">Log in</a></li>
<li><a href="">Sign up</a></li>
```


Git checkpoint

1. Commit changes
2. Create PR

Questions?

GitHub

IEEE Student Branch
KU Leuven Campus Brugge