**Tutorial 02 to do in class – Remember to upload the repo link to Teams**

1. **Configuring our database**

* Open an Anaconda Prompt with your environment activated and run the following command:

|  |
| --- |
| * **Execute in Terminal** |

conda install factory\_boy

* In your environment, open VS Code and run the following commands in the Terminal:

|  |
| --- |
| * **Execute in Terminal** |

python manage.py makemigrations

python manage.py migrate

* The following file should appear in your project folder:

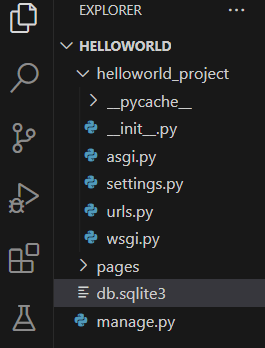


Figure 10-2. Database creation.

**B. A basic product model**

**Model**

* In your *pages* app got to *models.py* and add the following code:.

|  |
| --- |
| **Add Code** |

class Product(models.Model):

name = models.CharField(max\_length=255)

price = models.IntegerField()

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

updated\_at = models.DateTimeField(auto\_now=True)

**Factories**

* Go to *pages* and create a file *factories.py*, with the following content:

|  |
| --- |
| **Add Entire Code** |

import factory

from .models import Product

class ProductFactory(factory.django.DjangoModelFactory):

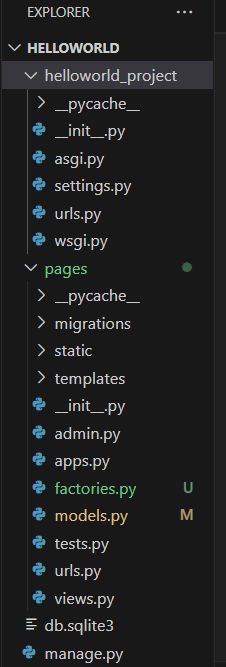
class Meta:

model = Product

name = factory.Faker('company')

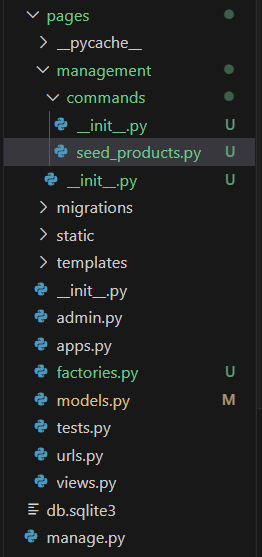
price = factory.Faker('random\_int', min=200, max=9000)

* Your project tree should look like this:



**DatabaseSeeder**

* In *page* add a *management* directory. Create a file named *\_\_init\_\_.py* inside (leave it empty), then create a *commands* directory inside. In /*management/commands* create two files named *\_\_init\_\_.py* and a *seed\_products.py.* Your app tree should look like this:



* Go to *pages/management/commands/seed\_products.py* and add the following code:

|  |
| --- |
| **Add Entire Code** |

from django.core.management.base import BaseCommand

from pages.factories import ProductFactory

class Command(BaseCommand):

help = 'Seed the database with products'

def handle(self, \*args, \*\*kwargs):

ProductFactory.create\_batch(8)

self.stdout.write(self.style.SUCCESS('Successfully seeded products'))

* In the Terminal, go to the project directory, and execute the following:

|  |
| --- |
| **Execute in Terminal** |

python manage.py makemigrations

python manage.py migrate

python manage.py seed\_products

* You should eight new products added into the database.



**C. A basic listing products from database**

**View/Controller**

* In *pages/views.py*, make the following changes in **bold**.

|  |
| --- |
| **Modify Bold Code** |

from django.shortcuts import render, redirect, **get\_object\_or\_404**

from django.views.generic import TemplateView, **ListView**

from django.views import View

from django.http import HttpResponseRedirect

from django.urls import reverse

from django import forms

from django.core.exceptions import ValidationError

**from .models import Product**

**~~class Product:~~**

**~~products = [~~**

**~~{"id":"1", "name":"TV", "description":"Best TV", "price":50},~~**

**~~{"id":"2", "name":"iPhone", "description":"Best iPhone", "price":150},~~**

**~~{"id":"3", "name":"Chromecast", "description":"Best Chromecast", "price":80},~~**

**~~{"id":"4", "name":"Glasses", "description":"Best Glasses", "price":30}~~**

**~~]~~**

…

class ProductIndexView(View):

template\_name = 'products/index.html'

def get(self, request):

viewData = {}

viewData["title"] = "Products - Online Store"

viewData["subtitle"] = "List of products"

viewData["products"] = **Product.objects.all()**

return render(request, self.template\_name, viewData)

class ProductShowView(View):

template\_name = 'products/show.html'

def get(self, request, id):

# Check if product id is valid

try:

product\_id = int(id)

if product\_id < 1:

raise ValueError("Product id must be 1 or greater")

**product = get\_object\_or\_404(Product, pk=product\_id)**

except (ValueError, IndexError):

# If the product id is not valid, redirect to the home page

return HttpResponseRedirect(reverse('home'))

viewData = {}

**product = get\_object\_or\_404(Product, pk=product\_id)**

viewData["title"] = **product.name** + " - Online Store"

viewData["subtitle"] =  **product.name** + " - Product information"

viewData["product"] = product

return render(request, self.template\_name, viewData)

…

**class ProductListView(ListView):**

**model = Product**

**template\_name = 'product\_list.html'**

**context\_object\_name = 'products' # This will allow you to loop through 'products' in your template**

**def get\_context\_data(self, \*\*kwargs):**

**context = super().get\_context\_data(\*\*kwargs)**

**context['title'] = 'Products - Online Store'**

**context['subtitle'] = 'List of products'**

**return context**

…

**Template/view**

* In *templates/products/show.html*, make the following changes in **bold**.

|  |
| --- |
| **Modify Bold Code** |

{% extends 'pages/base.html' %}

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

{% block header\_title %} {{subtitle}} {% endblock %}

{% block content %}

<div class="card mb-3">

<div class="row g-0">

<div class="col-md-4">

<img src="https://static.djangoproject.com/img/logos/django-logo-positive.svg" class="img-fluid rounded-start">

</div>

<div class="col-md-8">

<div class="card-body">

<h5 class="card-title">

{{product.name}}

</h5>

**~~<p class="card-text">{{product.description}}</p>~~**

**{% if product.price > 2000 %}**

**<p class="card-text" style="color: red;">{{product.price}}</p>**

**{% else %}**

**<p class="card-text">{{product.price}}</p>**

**{% endif %}**

</div>

</div>

</div>

</div>

{% endblock %}

**Execution**

|  |
| --- |
| **Execute in Terminal** |

python manage.py runserver

**Go to the (“/products”) route and you should see the application running with information retrieved from the database.**

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**D. A basic save product**

**View/Controller**

* In *pages/views.py*, make the following changes in **bold**.

|  |
| --- |
| **Modify Bold Code** |

…

class ProductForm(**forms.ModelForm**):

**~~name = forms.CharField(required=True)~~**

**~~price = forms.FloatField(required=True)~~**

**class Meta:**

**model = Product**

**fields = ['name', 'price']**

def clean\_price(self):

price = self.cleaned\_data.get('price')

if price is not None and price <= 0:

raise ValidationError('Price must be greater than zero.')

return price

class ProductCreateView(View):

template\_name = 'products/create.html'

def get(self, request):

form = ProductForm()

viewData = {}

viewData["title"] = "Create product"

viewData["form"] = form

return render(request, self.template\_name, viewData)

def post(self, request):

form = ProductForm(request.POST)

if form.is\_valid():

**form.save()**

return redirect('product-created')

else:

viewData = {}

viewData["title"] = "Create product"

viewData["form"] = form

return render(request, self.template\_name, viewData)

**Execution**

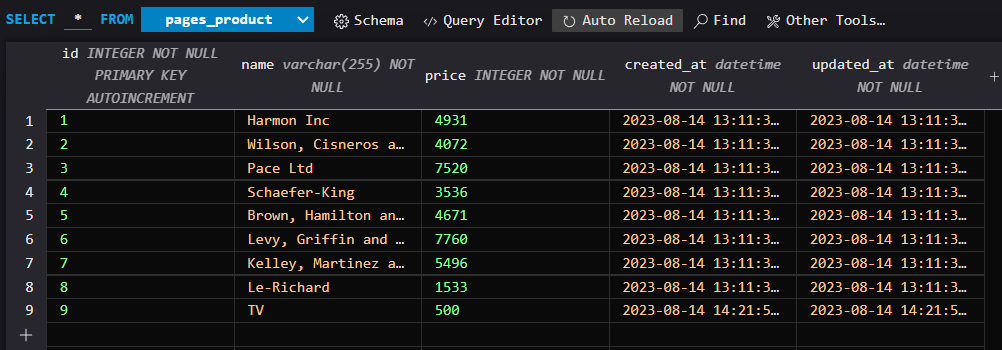
|  |
| --- |
| **Execute in Terminal** |

python manage.py runserver

**Go to the (“/products/create”) route and insert a new product. It should be added into the database.**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente



**E. A basic comment model and comment relationship**

**Model**

* Go to *pages/models.py* and add the following code:

|  |
| --- |
| **Add Entire Code** |

class Comment(models.Model):

product = models.ForeignKey(Product, on\_delete=models.CASCADE)

description = models.TextField()

**Executing the migrations**

* To run the migrations, in the Terminal:

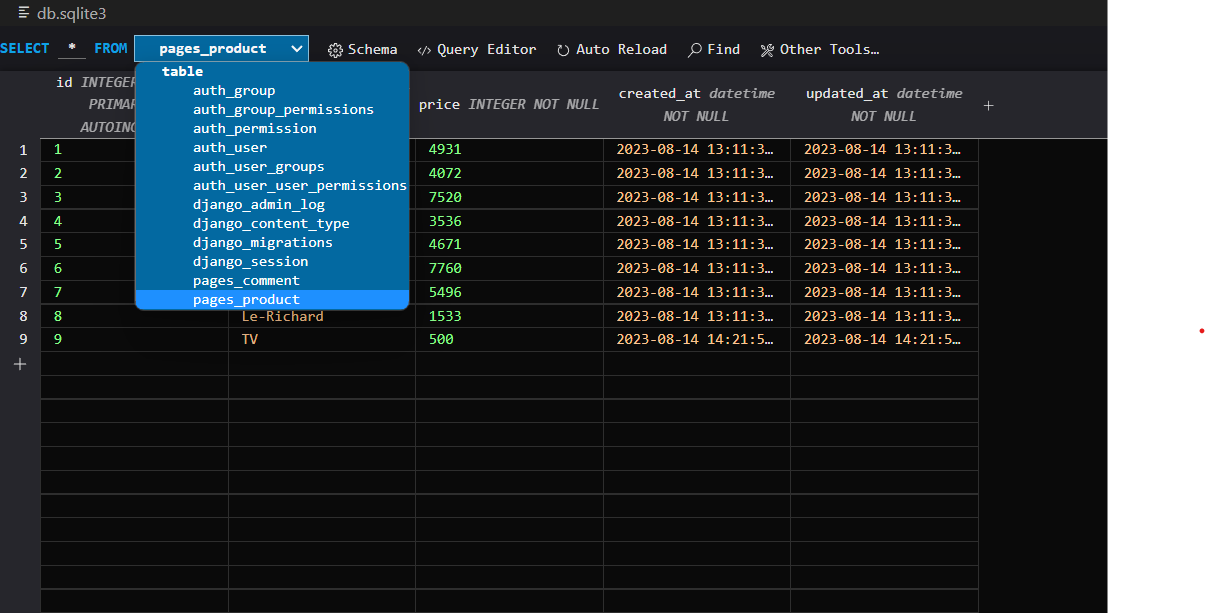
|  |
| --- |
| **Execute in Terminal** |

python manage.py makemigrations

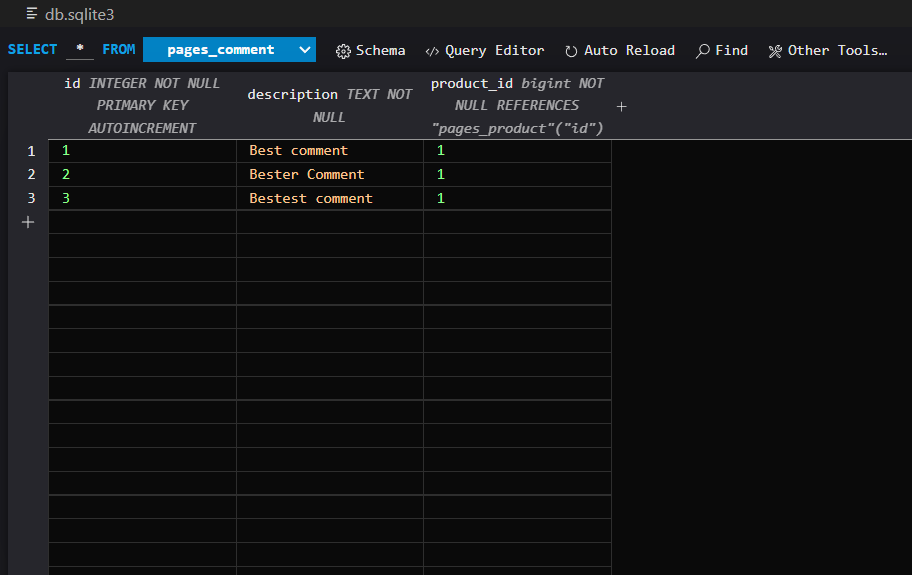
python manage.py migrate

**Comments rows**

* Open your project’s *db.sqlite3* database and find the pages\_comments table:



* Click on a new row add a comment with comment id, a description and link it to product id 1. Do this process three time and it should look like this in the pages\_comment table:



**Product show view**

* Go to *pages/templates/products/show.html* and make the following changes to the Template:

|  |
| --- |
| **Modify Bold Code** |

{% extends 'pages/base.html' %}

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

{% block header\_title %} {{subtitle}} {% endblock %}

{% block content %}

<div class="card mb-3">

<div class="row g-0">

<div class="col-md-4">

<img src="https://static.djangoproject.com/img/logos/django-logo-positive.svg" class="img-fluid rounded-start">

</div>

<div class="col-md-8">

<div class="card-body">

<h5 class="card-title">

{{product.name}}

</h5>

{% if product.price > 2000 %}

<p class="card-text" style="color: red;">{{product.price}}</p>

{% else %}

<p class="card-text">{{product.price}}</p>

{% endif %}

**{% for comment in product.comment\_set.all %}**

**- {{ comment.description }}<br />**

**{% endfor %}**

</div>

</div>

</div>

</div>

{% endblock %}

**Execution**

|  |
| --- |
| **Execute in Terminal** |

python manage.py runserver

**Go to the (“/products/1”) route. You will see the product with id 1 with its comments.**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Congratulations, you have created an app which uses Django Models (including relationships), Fakers, Migrations, Seeds, and many more.**

**Bonus: try one of the Coding Standard Libraries in your code to clean it.**