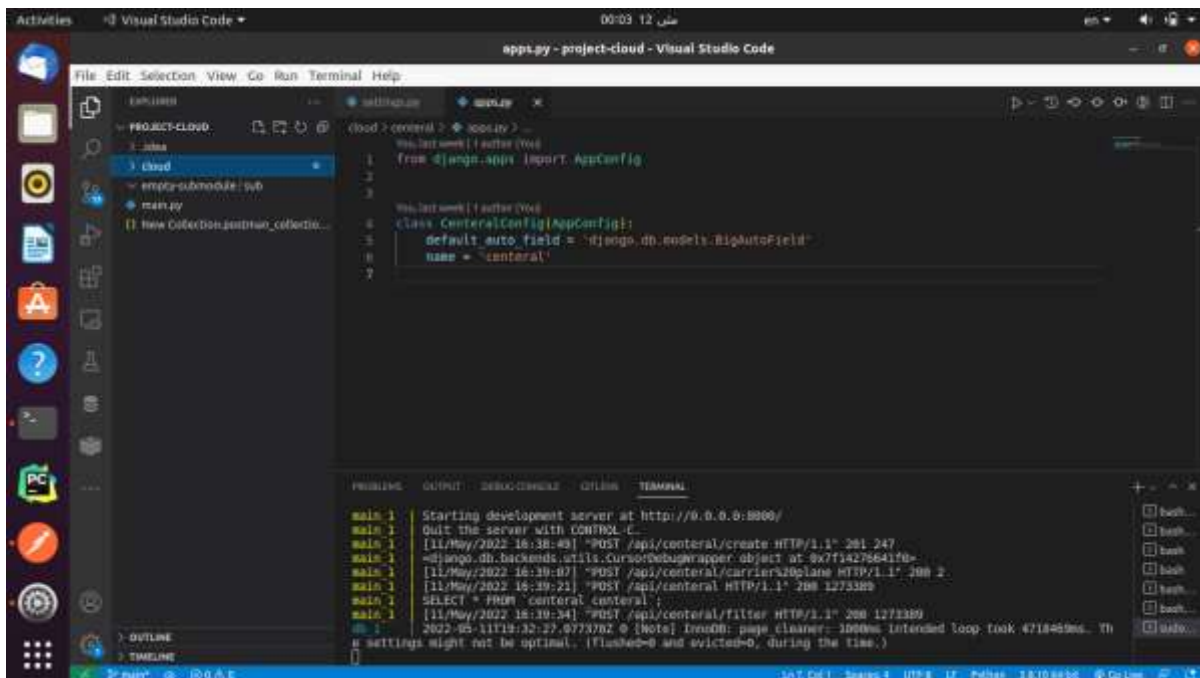


مراحل انجام پروژه فاز ۱ و ۲ (مهسا عبدالله زاده)

لازم به ذکر است پروژه اجرا می‌گردد

در مرحله اول باید یک پروژه تعریف کنیم که من برای کار با پروژه از VSC استفاده کردم که نرم افزارهایی مثل pycharm نیز برای کار خوب می باشد. بهتر است حتما در لینوکس کار کنید چرا که در ویندوز خطاهای زیادی خواهید داشت. بعد از نصب جنگو و داکر و پستمن که موارد مورد نیاز ما هست انجام پروژه را شروع می کنیم.

۱. برای شروع با استفاده و اجرا دستور `django-admin startproject cloud` و در انتها نوشتن اسم پروژه، پروژه خود را تعریف نمایید که من در اینجا نام پروژه را `cloud` گذاشتم



```
File Edit Selection View Go Run Terminal Help
PROJECT-CLOUD
  cloud
  empty-submodule.pyb
  main.py
  new Collection.justhno_collection...

cloud > central > app.py
1 from django.apps import AppConfig
2
3
4 class CentralConfig(AppConfig):
5     default_auto_field = 'django.db.models.BigAutoField'
6     name = 'central'
7

main:1 Starting development server at http://0.0.0.0:8000/
main:1 Quit the server with CONTROL-C.
main:1 [11/May/2022 16:38:49] "POST /api/central/create HTTP/1.1" 201 247
main:1 <django.db.backends.utils.CursorDebugWrapper object at 0x7f14278641f0>
main:1 [11/May/2022 16:39:07] "POST /api/central/carrier20plane HTTP/1.1" 200 2
main:1 [11/May/2022 16:39:21] "POST /api/central HTTP/1.1" 200 1273389
main:1 SELECT * FROM 'central' central;
main:1 [11/May/2022 16:39:34] "POST /api/central/filter HTTP/1.1" 200 1273389
main:1 2022-05-11T19:32:27.077370Z @ [Note] broodm: page cleaner: 1000ms, intended loop took 4718edms... Th
H settings might not be optimal. (Flushed=0 and evicted=0, during the time.)
```

۲. و بعد از رفتن به دایرکتوری `cloud` که مسیر اصلی پروژه هست با اعمال دستور `python3 manage.py runserver` سرور جنگو ران می شود و لینک مربوط را می دهد که وقتی بر روی آن کلیک کنید با صفحه ذیل مواجه می شوید



The install worked successfully! Congratulations!

You are seeing this page because `DEBUG=True` is in your settings file and you have not configured any URLs.



**Django Documentation**

Topics, references, & how-to's



**Tutorial: A Polling App**

Get started with Django



**Django Community**

Connect, get help, or contribute

از همین لحظه شروع به ایجاد فایل های لازم برای داکر می کنیم . سه پوشه به نام های `Dockerfile` و `docker-` `requirements.txt` و `compose.yml` ایجاد می کنیم .

این سه فایل باید در یک مسیر ایجاد شوند و در کنار یکدیگر باشند پس از ایجاد، اعمال دستورات لازم و و استاپ کردن سرور جنگو با اجرا کردن دستور `docker-compose up` داکر را اجرا می کنیم و بعد از اتمام وقتی سرور جنگو را رفرش کنید می بینید که `building` می گردد وقتی تکمیل شد می بینید که دیگه مطابق قبل لینکی می دهد که جنگو با داکر لینک شده است



Visual Studio Code interface showing the `requirements.txt` file in the `project-cloud` directory. The file lists dependencies for Django, Django REST Framework, and MySQL. The terminal shows the output of running `python manage.py runserver`, indicating the development server is starting at `http://0.0.0.0:8000/`.

۳. حال باید دیتا بیس `mysql` را بسازیم که در ادامه فایل `docker-compose.yml` دستورات لازم را می نویسیم و مشخص می کنیم که نام دیتابیس و ... چیست سپس پس از اجرا مجدد `docker-compose up` می بینیم که مواردی که مشخص کردیم ساخته شد و یک پوشه به نام `dbdate` در مسیر پوشه `cloud` ایجاد شد

Visual Studio Code interface showing the `docker-compose.yml` file in the `project-cloud` directory. The file defines a service named `db` using the `mysql:5.7.22` image, with environment variables for database name, user, password, and root password. The terminal shows the output of running `docker-compose up`, indicating the database service is starting.

```

# Application definition
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'corsheaders',
    'central',
    'authentication'
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
]

# Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
[11/May/2022 16:38:49] "POST /api/central/create HTTP/1.1" 201 247
=django.db.backends.utils.CursorDebugWrapper object at 0x7f14278641f0=
[11/May/2022 16:39:07] "POST /api/central/carrier2oplane HTTP/1.1" 200 2
[11/May/2022 16:39:21] "POST /api/central HTTP/1.1" 200 1273389
SELECT * FROM 'central.central';
[11/May/2022 16:39:34] "POST /api/central/filter HTTP/1.1" 200 1271389
2022-05-11T19:32:29.677370Z @ [Note] InnoDB: page cleaner: 1000ms. Internal loop took 4718499ms. Th
a settings might not be optimal. (Flushed=0 and evicted=0, during the time.)

```

۴. حال از طریق vsc از بخش database دیتابیس را می سازیم ، حال در ترمینال با اجرا دستور docker-compose exec main که نام سرویس هست وارد می شویم و حال با central manage.py python۳ اولین api خودمون رو با جنگو به نام central می سازیم حال می بینیم یک پوشه به نام central ایجاد شده است

۵. در این بخش حتما باید وارد setting پوشه اصلی شده و در بخش urls ها central را معرفی کرده و در بخش database نیز دیتابیس خودمان را معرفی کنیم ، حال وقتی دوباره docker-compose را بزنیم و وارد central شویم می بینیم که models اضافه شده است که در ان ما کلاس های مختلف را تعریف می کنیم و بعد با اجرا python makemigration مدل های تعریف شده به پوشه migration اضافه می شوند و بعد با python migrate manage.py همه migration ها را می شوند ، حال اگر دیتابیس ها را چک کنیم می بینیم که به طور کامل تمامی جدول هایی که در مدل اعلام کرده بودیم ساخته شد



The screenshot shows the Visual Studio Code interface with the file explorer on the left displaying the project structure. The main editor shows the content of `0001_initial.py`, which is a Django migration file. The terminal at the bottom shows the output of the development server, including the starting message and several HTTP requests and responses.

```

0001_initial.py
# Generated by Django 4.0.3 on 2022-05-02 07:10

from django.db import migrations, models

class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='airfield',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False,
                ('city_name', models.CharField(max_length=300))),
    ]

    main:1 Starting development server at http://0.0.0.0:8000/
    main:1 Quit the server with CONTROL-C.
    main:1 [11/May/2022 16:38:49] "POST /api/central/create HTTP/1.1" 201 247
    main:1 <django.db.backends.utils.CursorDebugWrapper object at 0x7f14278641f0>
    main:1 [11/May/2022 16:39:07] "POST /api/central/carrier320plane HTTP/1.1" 200 2
    main:1 [11/May/2022 16:39:21] "POST /api/central HTTP/1.1" 200 1273389
    main:1 SELECT * FROM 'central' central;
    main:1 [11/May/2022 16:39:34] "POST /api/central/filter HTTP/1.1" 200 1273389
    main:1 [11/May/2022 16:39:34] "POST /api/central/filter HTTP/1.1" 200 1273389
    main:1 2022-05-11T19:32:27.077370Z @ [Note] InnoDB: page cleaner: 1000ms. Intended loop took 471849ms. Th
    a settings might not be optimal. (Flushed=0 and evicted=0, during the time.)
  
```

۶. حال در `central` یک پوشه به نام `SERIALIZERS` ایجاد می کنیم و کلاس های سریالایزر را در آن قرار می دهیم  
 حال تنظیمات مربوط به `views.py` را انجام می دهیم و کارهایی که قرار است در `api` انجام شود را می نویسیم و سپس  
 با ساخت فایل `urls` و اضافه کردن `url` ها و مسیر های آن `view` ها متد مربوط به `function` ها را می نویسیم

The screenshot shows the Visual Studio Code interface with the file explorer on the left displaying the project structure. The main editor shows the content of `0001_initial.py`, which is a Django migration file. The terminal at the bottom shows the output of the development server, including the starting message and several HTTP requests and responses.

```

0001_initial.py
# Generated by Django 4.0.3 on 2022-05-02 07:10

from django.db import migrations, models

class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='airfield',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False,
                ('city_name', models.CharField(max_length=300))),
    ]

    main:1 Starting development server at http://0.0.0.0:8000/
    main:1 Quit the server with CONTROL-C.
    main:1 [11/May/2022 16:38:49] "POST /api/central/create HTTP/1.1" 201 247
    main:1 <django.db.backends.utils.CursorDebugWrapper object at 0x7f14278641f0>
    main:1 [11/May/2022 16:39:07] "POST /api/central/carrier320plane HTTP/1.1" 200 2
    main:1 [11/May/2022 16:39:21] "POST /api/central HTTP/1.1" 200 1273389
    main:1 SELECT * FROM 'central' central;
    main:1 [11/May/2022 16:39:34] "POST /api/central/filter HTTP/1.1" 200 1273389
    main:1 [11/May/2022 16:39:34] "POST /api/central/filter HTTP/1.1" 200 1273389
    main:1 2022-05-11T19:32:27.077370Z @ [Note] InnoDB: page cleaner: 1000ms. Intended loop took 471849ms. Th
    a settings might not be optimal. (Flushed=0 and evicted=0, during the time.)
  
```

Visual Studio Code - central\_central - project-cloud - Visual Studio Code

File Edit Selection View Go Run Terminal Help

central\_central.py

```
SELECT * FROM central_central LIMIT 100;
```

Input To Search Data

	id	report_id	city_origin	selected_origin	destination_city	destination_report	timestamp	price
1	1		Bahai	BER	Stockholm	ARN	2021-11-06 20:00:00.00	166
2	2		Tehran	THR	Stockholm	STO	2021-10-28 18:00:00.00	188
3	3		Tehran	THR	London	STN	2020-03-21 01:00:00.00	325
4	4		Manchester	MHT	London	LOW	2020-03-30 21:00:00.00	312
5	5		New York	NYCA	Munich	MUC	2020-01-13 20:00:00.00	334
6	6		ROME	ROME	New York	NYCA	2021-04-12 18:00:00.00	128

PROBLEMS OUTPUT DEBUG CONSOLE OUTPUTS TERMINAL

```
main:1 Starting development server at http://0.0.0.0:8080/
main:1 Quit the server with CTRL-C.
main:1 [11/May/2022 16:38:49] "POST /api/central/create HTTP/1.1" 201 247
main:1 <django.db.backends.utils.CursorDebugWrapper object at 0x7f14276641f8>
main:1 [11/May/2022 16:39:07] "POST /api/central/carrier2plane HTTP/1.1" 200 2
main:1 [11/May/2022 16:39:21] "POST /api/central HTTP/1.1" 200 1273389
main:1 SELECT * FROM central_central;
main:1 [11/May/2022 16:39:34] "POST /api/central/filter HTTP/1.1" 200 1273389
main:1 [2022-05-11T19:32:27.077376Z @ [Note] InnoDB: page cleaner: 1000ms intended loop took 4718469ms. The
a settings might not be optimal. (Flushed=0 and evicted=0, during the time.)
```

Visual Studio Code - serializers.py - project-cloud - Visual Studio Code

File Edit Selection View Go Run Terminal Help

serializers.py

```
from .models import central
from rest_framework import serializers

class CentralSerializer(serializers.ModelSerializer):
    class Meta:
        model = central
        fields = '__all__'
```

PROBLEMS OUTPUT DEBUG CONSOLE OUTPUTS TERMINAL

```
main:1 Starting development server at http://0.0.0.0:8080/
main:1 Quit the server with CTRL-C.
main:1 [11/May/2022 16:38:49] "POST /api/central/create HTTP/1.1" 201 247
main:1 <django.db.backends.utils.CursorDebugWrapper object at 0x7f14276641f8>
main:1 [11/May/2022 16:39:07] "POST /api/central/carrier2plane HTTP/1.1" 200 2
main:1 [11/May/2022 16:39:21] "POST /api/central HTTP/1.1" 200 1273389
main:1 SELECT * FROM central_central;
main:1 [11/May/2022 16:39:34] "POST /api/central/filter HTTP/1.1" 200 1273389
main:1 [2022-05-11T19:32:27.077376Z @ [Note] InnoDB: page cleaner: 1000ms intended loop took 4718469ms. The
a settings might not be optimal. (Flushed=0 and evicted=0, during the time.)
```

Visual Studio Code interface showing the file explorer on the left with a project structure including folders like 'cloud' and files like 'urls.py'. The main editor displays the content of 'urls.py' in the 'cloud' directory, which includes Django URL routing configurations. The terminal at the bottom shows the output of running the development server, including HTTP request logs for POST and GET requests to the API endpoints.

Visual Studio Code interface showing the file explorer on the left with a project structure including folders like 'cloud' and files like 'urls.py'. The main editor displays the content of 'urls.py' in the 'cloud' directory, which includes Django URL routing configurations. The terminal at the bottom shows the output of running the development server, including HTTP request logs for POST and GET requests to the API endpoints.





