**Creating a Django App**

Open Gitbash to source folder

To create virtual environment:

$ vagrant ssh

cd /vagrant/

python -m venv ~/env

source ~/env/bin/activate

In virtual environment type:

pip install -r requirements.txt

django-admin.py startproject profiles\_project .

python manage.py startapp profiles\_api

python manage.py runserver 0.0.0.0:8000

In local project Gitbash (not virtual environment):

git add .

git commit -am "Created django project and app"

$ git push origin

**Setting up the Database, Add a User Model Manager**

User Profile Model

Not using Django standard which doesn’t allow a login as an email

from django.db import models

from django.contrib.auth.models import AbstractBaseUser

from django.contrib.auth.models import PermissionsMixin

from django.contrib.auth.models import BaseUserManager

class UserProfileManager(BaseUserManager):

"""Manager for user profiles"""

def create\_user(self, email, name, password=None):

"""Create a new user profile"""

if not email:

raise ValueError('User must have an email address')

email = self.normalize\_email(email)

user = self.model(email=email, name=name)

user.set\_password(password)

user.save(using=self.\_db)

return user

def create\_superuser(self, email, name, password):

"""Create and save a new superuser with given details"""

user = self.create\_user(email, name, password)

user.is\_superuser=True

user.is\_staff = True

user.save(using=self.\_db)

return user

class UserProfile(AbstractBaseUser, PermissionsMixin):

"""Database model for users in the system"""

email = models.EmailField(max\_length=255, unique=True)

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

is\_active = models.BooleanField(default=True)

is\_staff = models.BooleanField(default=False)

objects = UserProfileManager()

USERNAME\_FIELD = 'email'

REQUIRED\_FIELDS = ['name']

def get\_full\_name(self):

"""Retrieve full name of user"""

return self.name

def get\_short\_name(self):

"""Retrieve short name of user"""

return self.name

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

"""Return string representation of our user"""

return self.email

**Set our custom user model**

In Oracle, go to ‘profile\_project’ folder and then ‘settings.py’ subfolder and add new line at bottom of standard script:

AUTH\_USER\_MODEL = 'profiles\_api.UserProfile'

**Create migrations and sync Database**

Open up Gitbash in local folder:

Type:

vagrant ssh

cd /vagrant

source ~/env/bin/activate

python manage.py makemigrations profiles\_api

then

python manage.py migrate

Then open a new Gitbash terminal and type:

git add .

git commit -am "Added custom user profile model, manager and migrations"

git push origin

**Set up Django admin**

**Creating a SuperUser**

Open Gitbash

Type:

vagrant ssh

cd /vagrant

source ~/env/bin/activate

python manage.py createsuperuser

enter email passwords etc

**Enable Django Admin**

In admin.py in Atom:

from profiles\_api import models

admin.site.register(models.UserProfile)

**Test Django Admin**

Go to Gitbash, get onto vagrant (env)

Type:

python manage.py runserver 0.0.0.0:8000

Go to chrome, type <http://127.0.0.1:8000/admin> and log in.

Open gitbash:

git add .

git commit -am "Enable Django admin for user profile model"

git push origin

**APIs**

**Create first API view**

Type in Atom view.py subfolder:

from rest\_framework.views import APIView

from rest\_framework.response import Response

class HelloApiView(APIView):

"""Test API View"""

def get(self, request, format=None):

"""Returns a list of APIView features"""

an\_apiview = [

'Uses HTTP methods as function (get, post, patch, put, delete)',

'Is similar to a traditional Djago view',

'Gives you the most control over your application logic',

'Is mapped manually to URLs',

]

return Response({'message': 'Hello!', 'an\_apiview': an\_apiview})

**Configure your API View**

Create a new file in the profiles\_api folder in Atom, called urls.py

Go to urls.py subfolder in profiles\_project folder

Add the following (highlighted):

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

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

path('api/', include('profiles\_api.urls'))

Go to the new urls.py file in the profiles\_api folder and type:

from django.urls import path

from profiles\_api import views

urlpatterns = [

path('hello-view/', views.HelloApiView,as\_view()),

]

**Testing our API View**

Go to chrome:

Type in 127.0.0.1.8000/api