### داکیومنت پروژه رستوران

حامد میرزایی

۹۹۱۱۰۴۱۰۵۶

جنگو یک فریمورک بر پایه زبان پایتون است که برای طراحی وبسایت استفاده میشود و یکی از محبوب ترین فریمورک ها در حوزه طراحی وب محسوب میشود.

فریمورک قالبی را برای برنامه نویس فراهم میکند که کد نویسی خود را سریعتر انجام دهد و از ابزار ها و قابلیت های فراهم شده در فریمورک استفاده کند و کد نویسی اضافی نداشته باشد.

مزایای جنگو:

- در جنگو با استفاده از Orm کد های پایتون تبدیل به کد های Sql میشود و نیازی به کد نویسی پایگاه داده نیست.

- دارای authentication پایه است.

- درای پنل ادمین پایه است.

- پسورد ها با استفاده از الگوریتم PBKDF2 رمزگذاری میشوند و به همین دلیل بسیار امن است.

- جنگو اپن سورس میباشد.

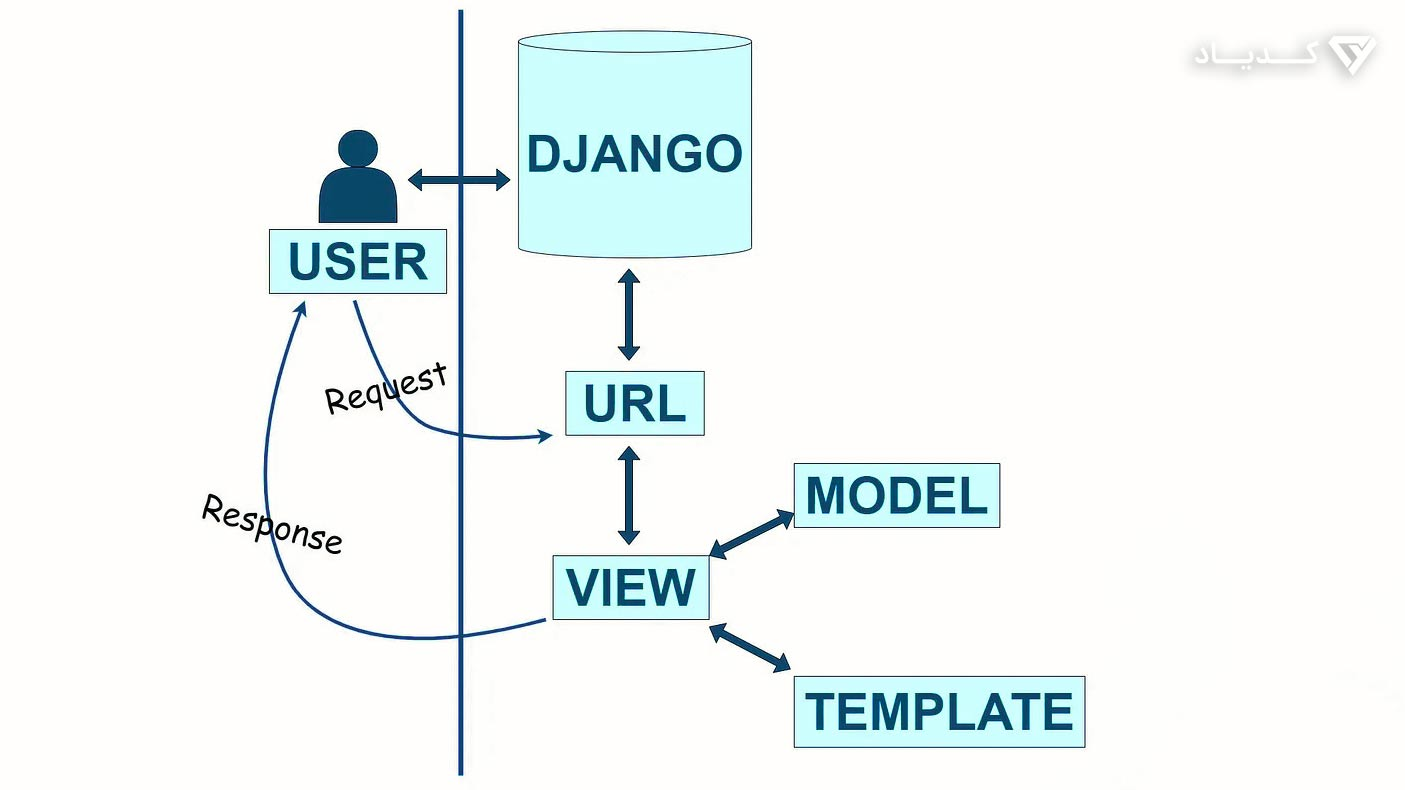
## **ساختار جنگو**

هر صفحه ای که بر روی وب ایجاد می شود، بر مبنای سه جز اصلی منطق ورودی، منطق تجاری، و منطق رابط کاربری (UI) شکل می گیرد، که هر کدام وظیفه خاص خود را دارند. منطق ورودی را می توان مجموعه ای در نظر گرفت که داده یا ورودی را دریافت می کند تا با فرمت دلخواه به پایگاه داده ارسال کند. منطق تجاری وظیفه اعتبار سنجی داده های ورودی را دارد تا به صورت Html یا فرمتی مشابه، کنترل شوند. منطق رابط کاربری نیز همانطور که انتظار می رود فرانت اند را پشتیبانی می کند. جنگو در هسته خود از معماری MVT پیروی می کند که سبکی تازه از الگوی کلاسیک MVC است که در ادامه به آنها می پردازیم.

## **معماری MVT در جنگو**

الگوی معماری MVT در جنگو، از سه مولفه مدل (Model)، نما (View) و قالب (Template) شکل گرفته است. الگوی MVT با وجود دشواری بیشتری که در یادگیری و نحوه استفاده در طراحی جنگو دارد، در مقایسه با الگوی MVC از کاربرد بیشتری برخوردار است. زیرا جنگو بطور پیش فرض به عنوان یک کنترل کننده عمل می کند و قادر است با واکشی داده، درخواستی را از برنامه دریافت کند و پاسخی را به آن بازگرداند. به بیانی ساده تر، جنگو مسئولیت هایی که بخش کنترل کننده الگوی MVC انجام می دهد را می تواند با منطق خودش انجام دهد. بنابراین، معماری MVT کارایی بیشتری نسبت به MVC دارد.

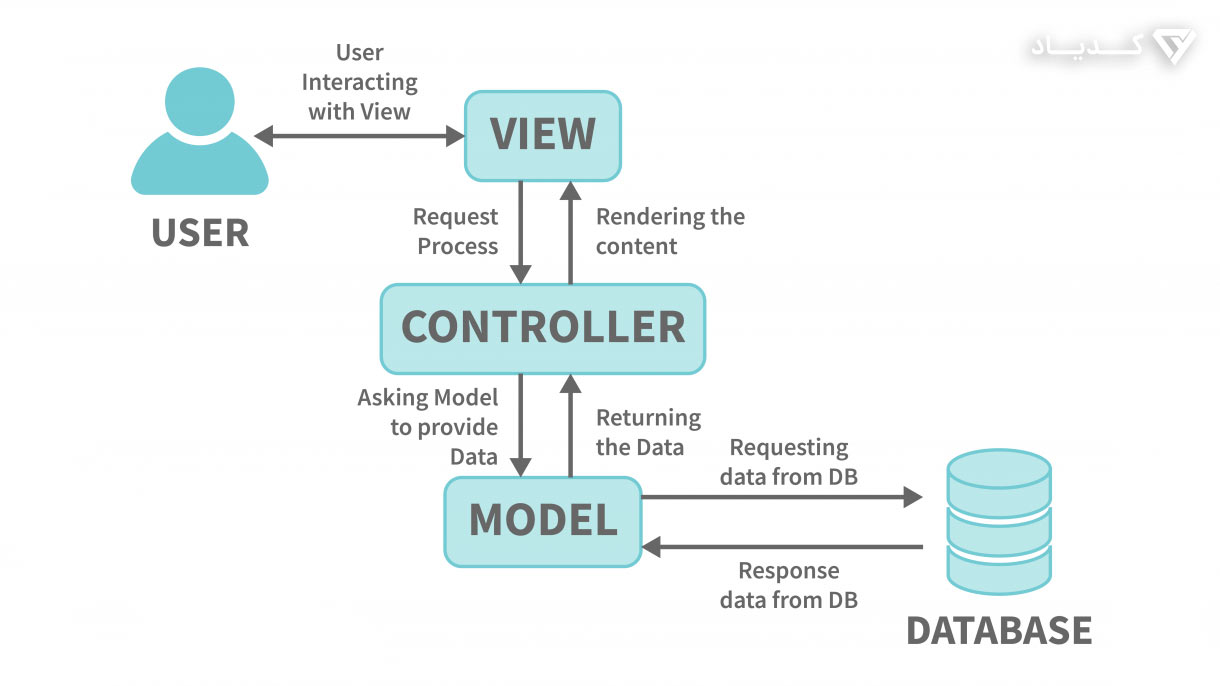
به طور کلی عملکرد معماری MVT در جنگو به این گونه است که در هنگام جستجوی عبارتی در صفحات وب توسط کاربر، یک درخواست URL به جنگو ارسال می شود و سپس جنگو منبع آن را بازیابی می کند. اگر URL به یک "نما" مرتبط باشد، جنگو آن "نما" را فرا می خواند. سپس "نما" و "مدل" با یکدیگر در تعامل قرار می گیرند تا داده ی مناسب از پایگاه داده بازیابی شود. در نهایت این داده ها به وسیله "الگو" به کاربر ارسال می شوند. اما الگوی معماری MVC در جنگو چگونه عمل می کند؟



## **معماری MVC در جنگو**

الگو MVC تشکیل شده از سه مولفه مدل (Model)، نما (View) و کنترل کننده (Controller) است که در توسعه برنامه های کاربردی بزرگتر عملکرد بهتری دارد. همچنین MVC می تواند پیچیدگی های طراحی وب یا اپلیکیشن را با حفظ ساختار کد کاهش دهد تا در صورت نیاز، اصلاح آن ساده تر شود. همچنین این الگو می تواند ویژگی های ASP.NET را با جنگو ترکیب کند.

هنگامی که یک کاربر در موتورهای جستجو به دنبال چیزی می گردد، یک درخواست HTTP به نما می رسد. نما اطلاعات مورد نیاز را از کاربر دریافت می کند، سپس داده های مدل را ارائه می دهد. لازم به ذکر است که تمامی این مراحل توسط کنترل کننده هدایت می شود. اما این مولفه ها دقیقا چه هستند و کاربردشان چیست؟



1. مدل (Model)

مدل مولفه ای است که بر مبنای منطق تجاری، ساختار داده را در جنگو تعریف می کند. همچنین وظیفه نگهداری، به روز رسانی و ذخیره داده ها در وب یا اپلیکیشن را نیز بر عهده دارد. و به عنوان واسطه بین وب و پایگاه داده هم عمل می کند. اما مدل، نحوه ارائه داده به کاربر را نمی داند.

همانطور که می دانید، عملکرد مدل در هر دو الگوی MVC و MVT مشابه است.

1. نما (View)

نما یک تابع کنترل کننده است که درخواست های HTTP را با واکشی داده ها از مدل پردازش می کند، تا روی سرور دیده شود و به URL ارسال کند. به بیانی ساده تر، نما درخواستی را از وب دریافت می کند و پاسخی را بر مبنای منطق رابط کاربری خود به وب باز می گرداند. پاسخ در نماهای جنگو می تواند هر چیزی باشد که یک مرورگر وب می تواند نشان دهد، مانند خطای 404.

نما در MVC و MVT تا حدی متفاوت عمل می کند. به این صورت که در معماری MVC، نما لایه ای است که اطلاعاتی را از کاربر دریافت می کند و سپس تصمیم می گیرد که کدام داده از میان داده هایی که مدل جمع آوری کرده است را به کاربر ارسال کند.

اما در معماری MVT برای نما تعریف شده است که کدام داده را به کاربر نشان دهد. ولی اینکه داده ها چگونه نمایش داده شوند تعریف نشده است. در عوض چگونگی ارائه داده ها به الگو واگذار شده است.

1. کنترل کننده (Controller)

هنگامی که درخواستی به نما می رسد، روند پردازش آن درخواست توسط کنترل کننده کنترل می شود. کنترل کننده میان دو مولفه مدل و نما، هماهنگی و تعامل ایجاد می کند.  وظیفه مولفه کنترل کننده، مدیریت و انتخاب نما بر اساس تعاملی است که با کاربر دارد. به این صورت که داده ها را از رشته کوئری به مدل منتقل می کند، تا مدل داده ها را در پایگاه داده ذخیره کند و سپس نما آن ها را نمایش دهد. حال هرگاه داده ها تغییر کنند، کنترل کننده باید نما را به روز کند. در واقع کنترل کننده، نمای مناسب را با توجه به درخواستی که کاربر دارد ایجاد می کند.

1. الگو (Template)

هر زمان که بخواهیم یک وب سایت با عملکردی پیچیده ایجاد کنیم، الگو یا قالب ها به ما کمک می کنند تا به روشی آسان به آن دست پیدا کنیم. الگو در معماری MVT جنگو، ساختار یا طرح رابط کاربری را تعریف می کند و می تواند داده ها را از نما دریافت کند و آنها را نمایش دهد.

الگو به طور مشابه وظیفه نما در معماری MVC را دارد به این صورت که نحوه ارائه داده ها را تعریف می کند. هنگامی که کاربر از یک برنامه تحت وب استفاده می کند، قالب یا همان الگو، تمام فایل های HTML و CSS را مدیریت می کند و محتوا را نشان خواهد داد.

در ابتدای کار نیاز است که یک virtual environment بسازیم. هدف اصلی محیط مجازی (Virtual environment) ایجاد یک محیط ایزوله شده و مجزا برای برنامه های پایتون است. این بدان معناست که هر پروژه‌ای می‌تواند صرف نظر از اینکه پروژه‌های دیگر به چه وابستگی‌هایی نیاز دارند، وابستگی‌های خاص خود را داشته باشد و پکیج های مد نظر خود را نصب و استفاده کند.

ایجاد محیط مجازی:

python3 -m venv env

فعالسازی:

source ./env/bin/activate

نصب آخرین نسخه جنگو:

pip install django

ایجاد پروژه جنگویی:

django-admin startproject config .

دایرکتوری config: این دایرکتوری شامل فایل های زیر است

- settings: تنظیمات پروژه در آن قرار میگیرد.

- urls: لینک های اصلی پروژه در آن قرار میگیرد.

- ASGI: برای اجرای پروژه به صورت همزمان (concurrent) و با پشتیبانی از ویژگی‌های مبتنی بر وقایع (event-driven) مانند WebSocket و HTTP/2 در جنگو.

- WSGI: برای اجرای پروژه به صورت سریال (serial) و با استفاده از واسط بین وب سرور و برنامه‌ی جنگو در معماری وب پایتون، که معمولاً برای پروژه‌های جنگو از آن استفاده می‌شود.

در هنگام ایجاد پروژه فایلی به اسم manage.py ایجاد میشود که با استفاده از آن میتوانیم دستورات داخلی جنگو را اجرا کنیم.

لیستی از دستورات:

با استفاده از دستور زیر میتوانیم پروژه را روی پورت 8000 اجرا کنیم:

python3 manage.py runserver

جمع آوری داده های استاتیک:

python3 manage.py collectstatic

ایجاد فایل migration:

pyhon3 manage,py makemigrations

اعمال تغییرات فایل های migration روی دیتابیس و جداول:

python3 manage.py migrate

ایجاد کاربر superuser:

python3 manage.py createsuperuser

ایجاد اپ:

python3 manage.pt startapp <app\_name>

فایل settings دارای تنظیمات مختلفی است که برخی از آنها در زیر توضیح داده شده اند:

- BASE\_DIR: ادرس دایرکتوری اصلی پروژه

-SECRET\_KEY: کلید اجرای پروژه

- DEBUG: تعیین اجرای پروژه به صورت development یا production که اگر مقدار آن True باشد، خطاها با جزئیات نمایش داده میشوند.

- ALLOWED\_HOSTS: آدرس هاست هایی که مجاز هستند.

- INSTALLED\_APPS: لیست اپ هایی که در پروژه استفاده میشود.

- MIDDLEWARE: در جنگو به قطعه‌های کد گفته می‌شود که بین درخواست و پاسخ در یک برنامه اجرا می‌شوند و امکان انجام عملیاتی مانند مدیریت و تغییر درخواست یا پاسخ را فراهم می‌کنند.

- ROOT\_URLCONF: آدرس لینک های اصلی پروژه

- TEMPLATES: یه سری تنظیمات مربوط به تمپلیت ها مانند تعیین context processor ها که مقادیری را در کل پروژه به صورت گلوبال داشته باشیم.

- WSGI\_APPLICATION: تعیین مسیری که سرور WSGI (Web Server Gateway Interface) باید برنامه‌ی جنگویی را برای اجرا انتخاب کند. این تنظیمات به سرورهای وب اطلاع می‌دهد که برنامه‌ی جنگو را از کدام فایل WSGI بارگیری کنند تا برنامه به درستی اجرا شود.

- DATABASES: تعیین دیتابیس های پروژه

- AUTH\_PASSWORD\_VALIDATORS: تعیین انواع مختلفی از روش های ولیدیشن پسوورد.

- LANGUAGE\_CODE: تعیین زبان مورد استفاده در پروژه

- TIME\_ZONE: تعیین محدوده زمانی

- USE\_I18N: فعال‌سازی پشتیبانی از بین‌المللی‌سازی (Internationalization) در جنگو، که این امکان را فراهم می‌کند تا متن‌ها و رابط کاربری در برنامه شما به زبان‌های مختلف ترجمه شود و بین‌المللی شود.

- USE\_L10\_N: فعال‌سازی پشتیبانی از محلی‌سازی و ترجمه‌ی متن به زبان‌های مختلف در جنگو.

- USE\_TZ: فعال‌سازی پشتیبانی از مناطق زمانی (Timezones) در جنگو، که این امکان را فراهم می‌کند که زمان‌های مختلف را با توجه به محلی‌سازی و تنظیمات مکانی مدیریت کنیم.

- STATIC\_URL: تعیین آدرس فایل های استاتیک

-STATIC\_ROOT: در جنگو، مسیری است که برنامه‌ی شما از طریق آن فایل‌های استاتیک (مانند فایل‌های CSS، JavaScript، تصاویر و ...) را برای استفاده در محیط تولیدی (production environment) ذخیره می‌کند. وقتی که شما از دستور collectstatic در جنگو استفاده می‌کنید، تمامی فایل‌های استاتیک موجود در برنامه‌ی شما از مسیرهای مختلف (مانند STATICFILES\_DIRS) جمع‌آوری شده و در مسیر مشخص‌شده توسط STATIC\_ROOT ذخیره می‌شوند. این مسیر باید قبل از استقرار برنامه در محیط تولیدی تنظیم شده باشد تا وب سرور بتواند به آن دسترسی داشته باشد و فایل‌های استاتیک را به درستی ارائه دهد.

- STATICFILES\_DIRS: در جنگو، فهرستی از مسیرهای فایل‌های استاتیک است که جنگو در آنها به دنبال فایل‌های استاتیک می‌گردد. این فایل‌ها از این مسیرها جمع‌آوری شده و پس از آن با استفاده از دستور collectstatic در مسیری که توسط STATIC\_ROOT مشخص شده، ذخیره می‌شوند.

- MEDIA\_URL: تعیین url اولیه فایل های اپلود شده توسط کاربران

- MEDIA\_ROOT: تعیین دایرکتوری اصلی فایل های اپلود شده توسط کاربران

- DEFAULT\_AUTO\_FIELD: تعیین نوع پیشفرض primary key های مدل ها.

سایر دایرکتوری ها و فایل ها:

- دایرکتوری templates: کد های html در آن قرار میگیرد.

- دایرکتوری static: کد های css, js, font و سایر موارد مربوط به فرانت اند در آن قرار میگیرد.

- فایل .gitignore:‌ برای ایگنور کردن فایل ها هنگام پوش کردن به ریپازیتوری موجود در گیتهاب.

- فایل db.sqlite: دیتابیس sqlite پروژه.

ساختار یک اپ جنگویی:

- models.py: محل قرارگیری مدل ها و ارتباط با دیتابیس

- admin.py: محل قرارگیری تنظیمات مربوط به پنل ادمین جنگو

- urls.py:‌ محل قرارگیری لینک های مربوط به اپ

- apps.py: محل قرارگیری برخی تنظیمات مربوط به اپ

- views.py: محل قرارگیری ویو های اپ

- tests.py: محل قرارگیری تست های اپ

- forms.py: محل قرارگیری فرم های اپ

- دایرکتوری migrations: محل قرارگیری فایل های migrations و تغییرات اعمال شده روی مدل ها(جداول دیتابیس)

انواع مختلف داده ای برای فیلد های مدل ها:

- CharField: داده ی کاراکتری با محدودیت طول کاراکتر

- ImageField: عکس

- FileField: فایل

- JsonField: ذخیره جیسون

- TextField: متن بدون محدودیت طول کاراکتر

- EmailField: ایمیل

- DateTimeField: زمان و تاریخ

…

فیلد ها دارای اپشن هایی هستند که در بعضی از آنها مشترک برای همه استفاده میشوند:

- max\_length: حداکثر طول مجاز

- upload\_to: محل ذخیره فایل و عکس

- unique: مشخص کردن یونیک بودن یا نبودن فیلد و دیتای آن

- auto\_now\_add: تعیین ذخیره خودکار زمان در هنگام ایجاد ابجکت

- auto\_now: تعیین ذخیره خودکار زمان در هنگام تغییر ابجکت

- blank: تعیین امکان خالی بودن مقدار فیلد در فرم های جنگویی

- null: تعیین امکان خالی بودن مقدار فیلد در دیتابیس و جداول

استفاده از html در پروژه: فایل های html در جنگو در پوشه ی templates قرار میگیرند. برای اینکه جنگو تشخیص دهد برای پیدا کردن html ها کجا را جستجو کند، باید در settings.py مقدارtemplates را به DIR در دیکشنری OPTIONS در داخل لیست TEMPLATES بدهیم.

استفاده از bootstrap برای استایل دهی فرانت اند:

با استفاده از cdn ها میتوانیم از بوت استرپ در پروژه استفاده کنیم.

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

استفاده از css و js و فونت ها و عکس ها در django: این موارد در فولدری به اسم static قرار میگیرند که میتوانیم آن را داخل config پروژه ایجاد کنیم.

هر کدام از موارد css, js, img میتوانند یک فولدر داشته باشند.

سپس نیاز است در داخل تنظیمات مواردی را تعیین کنیم:

STATIC\_URL = '/static/'

STATIC\_ROOT = BASE\_DIR / 'static'

STATICFILES\_DIRS = [

'config/static',

]

همچنین نیاز است که استایل ها را در html لینک دهی کنیم. اینکار را با استفاده از سینتکس زیر انجام میدهیم.

در ابتدای صفحه یک تمپلیت تگ را لود میکنیم، سپس فایل ها را لینک دهی میکنیم.

{% load static %}

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

برای اینکه سازماندهی مناسبی در کد هایمان داشته باشیم میتوانیم از قابلیت ایجاد اپ در جنگو استفاده کنیم. معمولا هر اپ تنها یک عملکرد خاص را داراست. برای مثال اپی با اسم cart صرفا کارهای مربوط به سبد خرید را انجام میدهد. یا اپی با نام employee صرفا مواردی که مربوط به کارمندان است را در خود نگه میدارد.

در پروژه ما اپ های زیر وجود دارند:

- account: برای مدیریت کاربران و ثبت نام و لاگین

- cart: برای مدیریت سبد خرید

- coupon: برای مدیریت تخفیف ها

- order: برای مدیریت سفارشات

- payment: برای مدیریت پرداخت ها

- restaurant: برای مدیریت رستوران

به ترتیب جزئیات تمام فایل های مربوط به هر اپ در پایین شرح داده میشود:

1. اپ account:

* models: شامل یک مدل User که از AbstractUser ارثبری کرده و برای گسترش اپ در اینده اینکار صورت گرفته است.
* admin: در این فایل مدل User به پنل ادمین اضافه شده و قابلیت افزودن/حذف/ویرایش و نمایش کاربران فعال شده است.
* apps: در این فایل صرفا یک سری تنظیمات اولیه مربوط به اپ قرار گرفته است.
* forms: فرم های مربوط به ویرایش پروفایل و ثبت نام در این فایل قرار گرفته است.
* mixins: در این فایل یک کلاس قرار گرفته که مسئول کنترل دسترسی برای کاربران است و اگر از آن در یک ویو ارثبری کنیم، صرفا کاربرانی که سوپریوزر هستند توانایی فراخوانی و دسترسی به محتوای صفحه آن را دارند.
* tests: در این فایل میتوان تست های اپ را نوشت و به صورت خودکار تست اپ را انجام داد.
* tokens: یک متد به اسم make\_hash\_value در یک کلاس قرار گرفته که مسئول ایجاد توکن برای بازگردانی رمز عبور به وسیله جیمیل است.
* urls: لینک های وصل شده به ویو ها در این فایل قرار میگیرند. در اینجا لینک های پروفایل کاربری و عملیات های ایجاد/حذف/ویرایش/لیست مدل های موجود در پنل ادمین اختصاصی قرار گرفته است.
* views:

- Home: نمایش صفحه اول پنل

- Profile: نمایش پروفایل کاربری

- Login: ورود کاربران

- PasswordChange: تغییر رمز عبور

- Register: ثبت نام کاربران

- activate: ارسال کد بازنشانی رمز عبور به ایمیل

- FoodList: لیست غذاها

- FoodCreate: ایجاد غذای جدید

- FoodUpdate: ویرایش غذا

- FoodDelete: حذف غذا

- CategoryList: لیست دسته بندی ها

- CategoryCreate: ایجاد دسته بندی جدید

- CategoryUpdate: ویرایش دسته بندی

- CategoryDelete: حذف دسته بندی

* یک ویو برای نمایش صفحه اول پنل به اسم Home، ویو Profile برای نمایش پروفایل کاربری،‌ ویو Login برای ورود کاربران،‌ ویو PasswordChange برای تغییر رمز عبور، ویو Register برای ثبت نام کاربران، ویو Aciivate

اپ employee: هر کارمند دارای فیلد های زیر است که در models.py تعیین شده اند و با استفاده ازorm تبدیل به ستون های جدول employee در دیتابیس میشنوند.

first\_name, last\_name, photo, desgnation, email\_address, phone\_number, created\_at, updated\_at

نام، نام خانوادگی، عکس، عنوان،‌ آدرس ایمیل، شماره تلفن، تاریخ ایجاد، تاریخ بروزرسانی

همانطوری که قبلا گفته شد، جنگو دارای پنل ادمین پایه است. با استفاده از اضافه کردن مدل ها به پنل ادمین میتوانیم تغییراتی روی مقادیر موجود ایجاد کنیم یا مقادیری را به دیتابیس وارد کنیم.

برای افزودن مدل ها به پنل ادمین در فایل admin.py میتوانیم به شکل زیر عمل کنیم:

admin.site.register(Employee)

برای اینکه فایل های اپلود شده توسط کاربر به درستی تشخیص داده شوند نیاز است که خط زیر را به ادامه urlpatterns اصلی پروژه اضافه کنیم:

urlpatterns = [

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

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

] + static(settings.MEDIA\_URL, ducoment\_root=settings.MEDIA\_ROOT)

متد str: برای نمایش درست ابجکت ها در پنل ادمین و هنگام چاپ آنها،‌ این متد را در model بازنویسی میکنیم.