### داکیومنت پروژه رستوران

حامد میرزایی

۹۹۱۱۰۴۱۰۵۶

موضوع پروژه:

موضوع پروژه یک رستوران است که مدیر رستوران توانایی وارد کردن دسته بندی ها، غذاها و مشاهده سفارشات و تیکت ها را دارد.

از طرفی کاربران توانایی لاگین، انتخاب چندین غذا و خرید آنلاین و همچنین مشاهده تاریخچه خرید های خود را دارد.

مقدمه ای بر جنگو:

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.

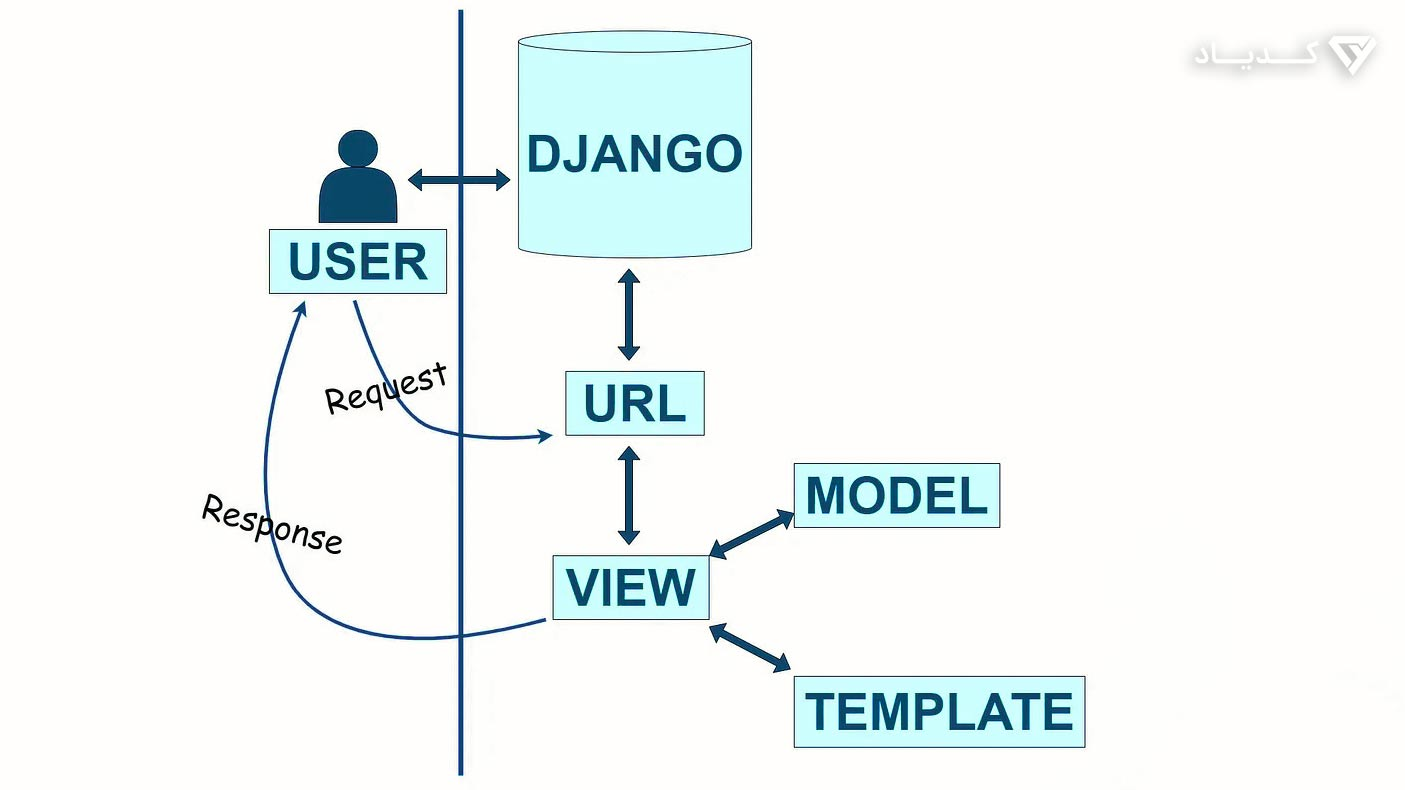
## ساختار جنگو:

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.

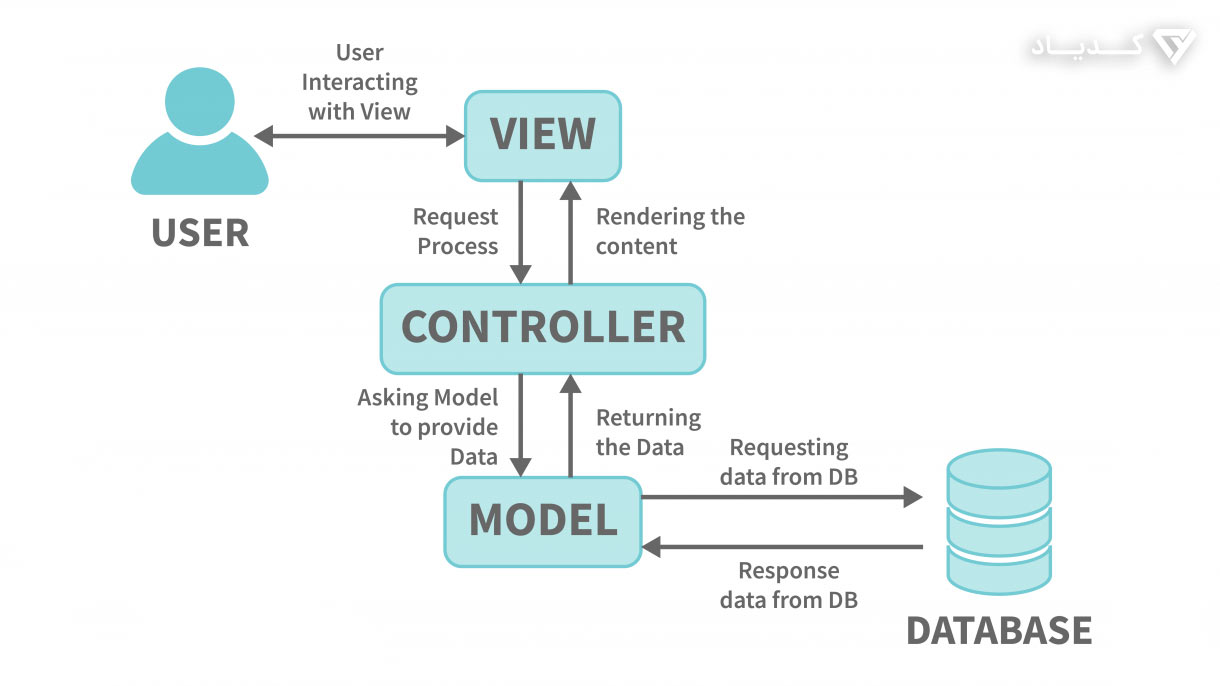
به طور کلی عملکرد معماری MVT در جنگو به این گونه است که در هنگام جستجوی عبارتی در صفحات وب توسط کاربر، یک درخواست URL به جنگو ارسال می شود و سپس جنگو منبع آن را بازیابی می کند. اگر URL به یک "نما" مرتبط باشد، جنگو آن "نما" را فرا می خواند. سپس "نما" و "مدل" با یکدیگر در تعامل قرار می گیرند تا داده ی 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.py 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
    - EmployeeList: لیست کارمندان
    - EmployeeCreate: ایجاد کارمند جدید
    - EmployeeUpdate: ویرایش کارمند
    - EmployeeDelete: حذف کارمند
    - AboutCreate: ایجاد اطلاعات صفحه درباره ما
    - AboutUpdate: ویرایش اطلاعات صفحه درباره ما
    - ContactList: لیست اطلاعات مربوط به پیام های مشتریان
    - ContactDetail: جزئیات پیام مشتری
    - ContactDelete: حذف پیام مشتری
    - AddressCreate: ایجاد اطلاعات مربوط به آدرس رستوران
    - AddressUpdate: ویرایش اطلاعات مربوط به آدرس رستوران
    - HistoryCreate: ایجاد اطلاعات مربوط به تاریخچه رستوران
    - HistoryUpdate: ویرایش اطلاعات مربوط به تاریخچه رستوران
    - OrderList: لیست سفارشات مشتریان
    - UserOrderList: لیست سفارشات هر مشتری(پنل کاربری)

1. اپ cart:

* cart: در این فایل یک کلاس به اسم Cart برای ذخیره کوکی سشن بیس اطلاعات سبد خرید کاربران قرار گرفته است. دارای متد های زیر است:
  + - \_\_init\_\_: ایجاد کارت یا دریافت کارت قبلی از سشن
    - \_\_iter\_\_: پیمایش روی ایتم های کارت و قابل پیمایش کردن کل کارت
    - \_\_len\_\_: نمایش تعداد ایتم های موجود در سبد خرید
    - add: اضافه کردن یک محصول به سبد خرید
    - save: بازنویسی متد save برای اپدیت کردن سبد خرید در سشن
    - remove: پاک کردن یک محصول از سبد خرید
    - decrement: کاهش یک واحدی موجودی یک محصول در سبد خرید
* clear: حذف کامل محصولات موجود در سبد خرید
* get\_total\_price: مجموع قیمت سبد خرید
* coupon: اعمال تخفیف در صورت وجود
* get\_discount: دریافت مقدار تخفیف
* get\_total\_price\_after\_discount:‌ دریافت قیمت بعد از تخفیف برای سبد خرید
  + - context\_processors:‌ اضافه کردن سبد خرید به کل پروژه برای دسترسی راحت در تمپلیت ها
    - urls: لینک های وصل شده به ویو ها در این فایل قرار میگیرند.
    - views:
  + cart\_add: اضافه کردن محصول به سبد خرید
  + item\_clear: حذف کردن محصول از سبد خرید
  + item\_incrmenet: اضافه کردن یک عدد موجودی به یک محصول در سبد خرید
  + item\_decrement: کم کردن یک عدد موجودی از یک محصول در سبد خرید
  + cart\_clear: پاک کردن کل سبد خرید
  + cart\_detail: نمایش جزئیات سبد خرید

1. اپ coupon:

* admin: اضافه کردن مدل Coupon به پنل ادمین پایه ی جنگو
* list\_display: لیست فیلدهای قابل نمایش در صفحه لیست تخفیفات
* list\_filter: لیست فیلدهای قابل فیلتر بر اساس آنها
* search\_fields: لیست فیلدهای قابل سرچ براساس آنها
* forms: فرم های مورد استفاده در ویو ها و رندر شده در تمپلیت ها که در اینجا به فرم CouponApplyForm برای اعمال کد تخفیف توسط کاربر نیاز داریم.
* models:
* Coupon: این مدل برای ذخیره اطلاعات کد های تخفیف ایجاد شده است و دارای فیلد های زیر است.
* فیلد code: برای کد تخفیف
* فیلد valid\_from: تاریخ شروع قابل استفاده بودن کد تخفیف
* فیلد valid\_to: تاریخ پایان قابل استفاده بودن کد تخفیف
* فیلد discount: مقدار تخفیف
* فیلد active: فعال بودن یا نبودن کد تخفیف
* متد str: نمایش مقدار code در هنگام صدا زدن ابجکتی از Coupon
* urls: لینک های وصل شده به ویو ها
* views:
* coupon\_apply: اعمال تخفیف با استفاده از کد تخفیف وارد شده توسط کاربر در صورت تایید شدن آن

1. اپ order:

* admin:
* OrderItemInline: اضافه کردن اینلاین ایتم ها به سفارشات برای اعمال تغییراتی مانند اضافه کردن/ویرایش/حذف یک ایتم از یک سفارش
* OrderAdmin: اضافه کردن سفارشات به پنل ادمین پایه
  + - inlines: تعیین مدل های اینلاین برای نمایش در پایین هر سفارش و دسترسی راحت تر به ابجکت های وصل شده.
* ‍‍forms:
* OrderCreateForm: ایجاد یک مدل فرم(فرمی که فیلد های آن بر پایه یک مدل ایجاد شده)، برای ایجاد یک سفارش توسط کاربر
* models:
* Order: این مدل برای ذخیره سفارشات ایجاد شده است.
  + - user: ذخیره کاربر ایجاد کننده ی سفارش
    - فیلد first\_name: ذخیره نام سفارش دهنده
    - فیلد last\_name: ذخیره نام خانوادگی سفارش دهنده
    - فیلد address: ذخیره آدرس سفارش دهنده
    - فیلد email: ذخیره آدرس ایمیل سفارش دهنده
    - فیلد city: ذخیره شهر محل زندگی سفارش دهنده
    - فیلد postal\_code: ذخیره کد پستی سفارش دهنده
    - فیلد created: ذخیره تاریخ ایجاد سفارش
    - فیلد updated: ذخیره تاریخ ویرایش سفارش
    - فیلد total\_paid: ذخیره مقدار پرداخت شده
    - فیلد additional\_information: ذخیره سایر اطلاعات
    - فیلد order\_key: ذخیره شماره سفارش
    - فیلد billing\_status: ذخیره وضعیت پرداخت سفارش
    - فیلد coupon: ذخیره کد تخفیف اعمال شده روی سفارش
    - فیلد discount: ذخیره مقدار تخفیف اعمال شده روی سفارش
    - کلاس Meta: تعیین مرتب سازی بر اساس تاریخ ایجاد سفارش به صورت نزولی
    - متد str: نمایش مقدار تاریخ ایجاد سفارش در هنگام صدا زدن ابجکتی از Order
    - متد get\_total\_cost: بازگرداندن قیمت کل سفارش
    - متد get\_discount: بازگرداندن مقدار تخفیف در صورت وجود کد تخفیف
    - متد get\_total\_cost\_after\_discount: برگرداندن قیمت کل سفارش بعد از اعمال تخفیف
    - متد get\_billing\_status: نمایش وضعیت پرداخت سفارش
* OrderItem: برای ذخیره ایتم های سفارشات
  + - فیلد order: سفارش مربوطه
    - فیلد product: محصول مربوطه
    - فیلد price: قیمت ایتم
    - فیلد quantity: تعداد ایتم
    - متد str: نمایش مقدار id یک ایتم در هنگام فراخوانی یک ابجکت از OrderItem
    - متد get\_cost: بازگرداندن مقدار کل یک ایتم(تعداد ایتم ضرب در قیمت ایتم)
* urls: لینک های وصل شده به ویو های موجود.
* views:
* order\_create: ایجاد یک سفارش برای کاربر
* user\_orders: نمایش کل سفارشات یک کاربر

1. اپ payment:

* views:
* PaymentView: دارای متد get برای نمایش یک صفحه پرداخت و متد post برای رفتن به درگاه پرداخت زرین پال و پرداخت مبلغ سفارش
* VerifyView: دارای یک متد get برای ریدایرکت شدن به آن پس از پرداخت مبلغ سفارش و تایید پرداخت موفق.
* zarinpal:
* zpal\_request\_handler: ارسال درخواست به درگاه پرداخت زرین پال برای پرداخت مبلغ سفارش
* zpal\_payment\_checker: بررسی پرداخت موفق بعد از ریدایرکت شدن از صفحه پرداخت

1. اپ restaurant:

* admins: اضافه کردن تمام مدل های ایجاد شده در این اپ به پنل ادمین پایه جنگو
* forms:
* ContactForm: برای ارسال پیام توسط کاربر به مدیران وبسایت(ارتباط با ما)
* models:
* SingletonBaseModel: برای استفاده از الگوی سینگلتون و ایجاد حداکثر یک ابجکت از یک مدل
  + - کلاس Meta: تعیین اینکه برای این مدل در دیتابیس جدولی ایجاد نمیشود.
    - متد save: تعیین ایدی 1 برای ابجکت.
    - متد delete: غیرفعالسازی امکان حذف ابجکت.
    - کلاس متد load: برگرداندن ابجکت در صورت وجود، در غیراین صورت ایجاد آن.
* AvailableManager :
  + - متد get\_queryset: فیلتر کردن محصولاتی که در فروشگاه موجود نیستند و نمایش ندادن آنها در کوئری ست.
* Category:
  + - فیلد name: نام دسته بندی
    - فیلد slug: نام انگلیسی دسته بندی که منحصر به فرد است.
    - متد str: نمایش نام دسته بندی در هنگام فراخوانی ابجکتی از Category
    - متد get\_absolute\_url: تعیین لینک نمایش صفحه جزئیات یک دسته بندی، که در واقع محصولات بر اساس این دسته بندی فیلتر میشوند.
* Food:
  + - فیلد name: نام غذا
    - فیلد slug: نام انگلیسی غذا که منحصر به فرد است.
    - فیلد Image: عکس غذا
    - فیلد description: توضیحات غذا
    - فیلد available: موجود بودن یا نبودن غذا
    - فیلد price: قیمت غذا
    - فیلد created: تاریخ ایجاد غذا
    - فیلد updated: تاریخ ویرایش غذا
    - منجر objects: منجر اصلی که کوئری را کامل اجرا میکند.
    - منجر objects\_available: منجر اختصاصی که صرفا ابجکت هایی که فیلد available آنها True باشد را برمیگرداند.
    - کلاس Meta: تعیین مرتب سازی بر اساس تاریخ ایجاد به صورت نزولی و تعیین اینکه slug بر اساس id غذا به صورت خودکار ساخته شود.
    - متد get\_absolute\_url: تعیین لینک نمایش صفحه جزئیات یک غذا.
  + Employee:
    - * فیلد first\_name: نام کارمند
      * فیلد last\_name: نام خانوادگی کارمند
      * فیلد image: عکس کارمند
      * فیلد position: سمت کارمند
      * فیلد description: توضیحات
      * فیلد Instagram: آدرس اینستاگرام کارمند
      * فیلد tweeter: آدرس توییتر(X) کارمند
      * فیلد facebook: آدرس فیسبوک کارمند
      * فیلد youtube: آدرس یوتیوب کارمند
      * متد str: نمایش اسم کامل کارمند هنگام فراخوانی ابجکتی از Employee
      * متد get\_full\_name: برگرداندن اسم کامل کارمند
* About: این کلاس از کلاس SingletonBaseModel ارثبری کرده و فقط یک ابجکت از آن ساخته میشود که برای وارد کردن متن صفحه ی درباره ی ما است.
  + - فیلد description: برای متن صفحه ی درباره ی ما
    - متد str: برای نمایش description هنگام فراخوانی ابجکت About
* History: این کلاس از کلاس SingletonBaseModel ارثبری کرده و فقط یک ابجکت از آن ساخته میشود که برای وارد کردن متن بخش تاریخچه رستوران است.
  + - فیلد description: برای متن بخش تاریخچه رستوران
    - متد str: برای نمایش description هنگام فراخوانی ابجکت History
* Address: این کلاس از کلاس SingletonBaseModel ارثبری کرده و فقط یک ابجکت از آن ساخته میشود که برای وارد کردن اطلاعات آدرس رستوران است.
  + - فیلد description: برای توضیحات
    - فیلد phone\_number: برای شماره تلفن رستوران
    - فیلد email: برای آدرس ایمیل رستوران
    - فیلد tweeter: برای آدرس توییتر رستوران
    - فیلد facebook: برای آدرس فیسبوک رستوران
    - فیلد Instagram: برای آدرس اینستاگرام رستوران
    - متد str: برای نمایش description هنگام فراخوانی ابجکت Address
* Contact: برای ذخیره اطلاعات مربوط به پیام های ارسال شده توسط کاربران در صفحه ارتباط با ما.
  + - فیلد name: برای نام کاربر
    - فیلد email: برای آدرس ایمیل کاربر
    - فیلد message: برای پیام کاربر
    - متد str: برای نمایش نام کاربر در هنگام فراخوانی ابجکتی از Contact
* urls: لینک های وصل شده به ویو های اپ
* views:
  + HomeView: نمایش صفحه اصلی سایت که غذاها و دسته بندی ها در آن قرار دارند و به وسیله ی get\_context\_data کوئری های مربوطه را به تمپلیت فرستادیم.
  + AboutView: نمایش صفحه درباره ما
  + ContactView: صفحه ارتباط با ما که فرم ContactForm در آن نمایش داده میشود.

و اما میرسیم به دایرکتوری static که دارای دایرکتوری های زیر است:

1. css: محل قرارگیری فایل های استایل دهی css
2. img: محل قرارگیری عکس ها
3. js: محل قرارگیری فایل های جاوااسکریپت
4. registration: محل قرارگیری فایل های استاتیک اپ account
5. webfonts: محل قرارگیری فونت ها

آخرین دایرکتوری templates است که در آن فایل های html قرار گرفته که در پایین جزئیات آنها آورده شده است:

* cart/cart\_detail.html: جزئیات سبد خرید کاربر در آن قرار گرفته است.
* order/order\_create.html: ایجاد یک سفارش توسط کاربر(وارد کردن آدرس و ...) در این صفحه انجام میگیرد.
* payment/process.html: پردازش پرداخت(قبل از ورود به درگاه بانک) در این صفحه اتفاق می افتد.
* payment/callback.html: بعد از بازگشت از درگاه پرداخت(پرداخت موفق یا ناموفق)،‌ به این صفحه منتقل میشویم.
* registration/activate\_Account.html: تمپلیتی برای ارسال پیام بازنشانی رمزعبور کاربر از طریق ایمیل.
* registration/base.html: تمپلیت پایه ی مورد استفاده در سایر فایل های html این دایرکتوری که شامل ساختار اولیه همه ی آنها است. با استفاده از extends از ان ارثبری میکنند.
* registration/home.html: صفحه اصلی پروفایل کاربری.
* registration/login.html: صفحه ورود کاربر.
* registration/password\_change\_form.html: صفحه نمایش فرم تغییر رمز عبور کاربر.
* registration/password\_change\_done.html: صفحه تایید تغییر رمز عبور کاربر.
* registration/password\_reset\_form.html: صفحه نمایش فرم بازنشانی رمز عبور کاربر با استفاده از ارسال کد به ایمیل.
* registration/password\_reset\_confirm.html: صفحه تایید ارسال ایمیل برای بازنشانی رمز عبور.
* registration/password\_reset\_done.html: صفحه اتمام و نمایش پیام موفقیت آمیز بودن ارسال ایمیل برای بازنشانی رمز عبور.
* registration/profile.html: نمایش پروفایل کاربری و جزئیات آن.
* registration/register.html: صفحه ثبت نام کاربران.
* registration/sidebar.html: صفحه ساید بار صفحاتی مانند پروفایل کاربری
* registration/models/about\_create\_update.html: ایجاد و ویرایش مدل About.
* registration/models/address\_create\_update.html: ایجاد و ویرایش مدل Address.
* registration/models/category\_create\_update.html: ایجاد و ویرایش یک دسته بندی غذایی.
* registration/models/category\_list.html: نمایش لیست دسته بندی ها.
* registration/models/contact\_confirm\_delete.html: صفحه تایید عملیات حذف یک ابجکت از تیکت(ارتباط با ما).
* registration/models/contact\_detail.html: صفحه جزئیات یک ابجکت از ارتباط با ما.
* registration/models/contact\_list.html: صفحه لیست ابجکت های ارتباط با ما.
* registration/models/employee\_confirm\_delete.html: صفحه تایید عملیات حذف یک ابجکت از کارمند.
* registration/models/employee\_create\_update.html: صفحه ایجاد و ویرایش یک ابجکت از کارمند.
* registration/models/employee\_list.html: صفحه لیست ابجکت های کارمند.
* registration/models/food\_confirm\_delete.html: صفحه تایید عملیات حذف یک ابجکت از غذا.
* registration/models/food\_create\_update.html: صفحه ایجاد و ویرایش یک ابجکت از غذا.
* registration/models/food\_list.html: صفحه نمایش لیست ابجکت های غذا.
* registration/models/history\_create\_update.html: صفحه ایجاد و ویرایش ابجکت تاریخچه رستوران.
* registration/models/order\_list.html: صفحه لیست سفارشات ایجاد شده توسط مشتریان.
* registration/models/user\_order\_list.html: صفحه سفارشات مختص به یک کاربر هنگامی که وارد پروفایل کاربری خود میشود.
* restaurant/about.html: صفحه نمایش درباره ی ما وبسایت.
* restaurant/base.html: صفحه پایه ی مورد استفاده در تمپلیت های about, contact, home که دارای فوتر و هدر وبسایت میباشد.
* restaurant/contact.html: صفحه نمایش فرم ارتباط با ما برای کاربران.
* restaurant/home.html: صفحه اصلی وبسایت که در آن غذاها و دسته بندی های آن نمایش داده میشوند.