

به نام خدا

پروژه دوره گردانی

موضوع : سیستم جامع اطلاعات استانی

ارئه شده به

گروه کامپیوتر

دانشکده شهید شمسى پور

به راهنمایی

استاد

محمد مهدى آهنگرى

توسط دانشجو

محمد رسول طارمى

به شماره دانشجوئى

۹۶۱۱۱۰۳۳۱۵۴۰۳۶

تاریخ دفاع

۹۸/۰۶/۳۱

در شروع این مستند باید این را ذکر کنم که این پروژه با استفاده از زبان برنامه نویسی سطح بالای پایتون (python) نگارش شده.

این زبان به دلیل سادگی، رایگان بودن، و سرعت بالا در توسعه به یکی از مهم ترین زبان های فعلی صنعت برنامه نویسی تبدیل شده که من هم تصمیم گرفتم از قافله بروز بودن در دنیای کامپیوتر عقب نباشم و شروع به یادگیری این زبان کردن.

قصد ندارم در این بخش زبان پایتون را توضیح بدهم زیرا در بخش پیش گفتار به طور مفصل به این قضیه پرداخته ام. در اینجا قصد دارم از کتابخانه پایتون یعنی Django که به صورت لفظی آن را جنگو صدا میزنند توضیحاتی بدهم.

جنگو یکی از چندین هزار کتابخانه های زبان پایتون است که با کمک آن میتوان برنامه نویسی server-side انجام داد.

کتابخانه هایی که مثل جنگو میتوانند سمت سرور کد بزنند عبارت اند از

- Django ✓
- Flask ✓
- Pyramid ✓
- TurboGears ✓
- Web2Py ✓
- Bottle ✓
- CherryPy ✓
- Sanic ✓
- Tornado ✓
- Dash ✓

در اینجا صرفا از کتاب خانه جنگو صحبت میشه ولی این با این معنا نیست که کتاب خانه های دیگه ضعیف یا غیر کاربردی هستند.

صرفاً برای این جنگو انتخاب شده است که ترند فعلی بازار قرار دارد و بیشترین کورس های آموزشی، بزرگ ترین جامعه افرادی که در حل مشکلات میتوانند یاری رسان باشند، و بیشتری تقاضای بازار را داراست.

پروژه سیستم جامع اطلاعات استانی چیست؟

سیستمی طراحی شده که در این سیستم از طریق یکپارچه سازی و جامع سازی مکان های گردش گری ایران از طریق هدف مند سازی صنعت توریسم به شکوفایی و شناخت هر چه بیشتر مکان های گردش گری ایران کمک شود.

در این سیستم شما میتوانید بر اساس سلیقه یا استان مورد نظر خودتان منطقه را پیدا و مقاداری اطلاعات در مورد آن کسب کنید تا اطلاعات کامل تر به سفر بروید.

شما میتوانید با انتخاب استان مورد نظر در صفحه ی اول و فیلتر کردن نوع مکان گردش گری که میتواند مذهبی، تاریخی، گردشگری، طبیعی یا انواع و اقسام مکان ها که خودتان میتوانید به انها اضافه کنید مکان مورد نظر را پیدا کرده یا با مکان های دیگر مقایسه کنید

هدف از ساخت این پروژه جمع آوری و یکپارچه سازی تمام مکان های تاریخی، تفریحی، گردش گری، توریستی، مذهبی و... به طوری که هر شخص به تواند با در نظر گرفتن استان و نوع مکانی مورد نظر به راحتی به تمامی اطلاعات آن مکان دسترسی داشته باشد.

اطلاعات نظر نظیر

- ✓ چندین عکس از اون مکان
- ✓ مکان و آدرس دقیق آن محل روی نقشه
- ✓ چندین خط توضیح کارشناسانه
- ✓ و ما بقی اطلاعاتی که ممکن است یک گردشگر لازمش بشود

در این پروژه سعی شده از بروز ترین زبان های موجود برنامه نویسی که همان پایتون است استفاده شود. که هم سرعت قابل قبولی دارد هم جامعه برنامه نویسان بزرگی دارد.

اما فقط پایتون در این پروژه استفاده نشده ، کتاب خانه های بزرگی نظیر Django نیز در آن به کار رفته که جز بزرگ ترین و قوی کتابخانه های موجود است.

در سیستم پایگاه داده ای نیز از پایگاه داده sqlite استفاده شده، که از سرعت و امنیت قابل قبولی بهره مند است.

دلیل استفاده از این پایگاه داده این است که

(۱) پیشنهاد خود جنگو است زیرا برای پروژه های کوچک یا متوسط بسیار کارا و پر سرعت است.

(۲) نیاز به تنظیمات خاصی ندارد.

(۳) در تمامی سیستم عامل ها به راحتی قابل نصب و اجرا است.

در بخش ظاهر یا interface پروژه سعی شده از بروز و رایج ترین زبان ها و کتابخانه های موجود استفاده بشه تا در کنار دسترسی به بزرگ ترین پایگاه اطلاعاتی استانی ظاهری کاربر پسند و قابل استفاده داشته باشد.

از جمله زبان های استفاده شده عبارت اند از:

Html5 ✓

Css3 ✓

javascript ✓

البته به همینجا ختم نمیشه برای هر چه شکل تر و کاربر پسندتر شدن برنامه از کتابخانه های قدرت مندی چون

Bootstrap ✓

UIKit ✓

Jquery ✓

نیز استفاده شده تا در جهت یاری رساندن به user experience در بخش user interface قدرت مند باشد.

- چکیده
 - پروژه سیستم جامع اطلاعات استانی چیست؟
- پیشگفتار
- شرح کامل و مفصل
 - زبان برنامه نویسی (پایتون چیست؟)
 - کتابخانه
 - جنگو چیست؟
- ساختمان داده
 - پایگاه داده در سیستم
 - طراحی جداول
 - طراحی جدول استان ها
 - طراحی جدول موضوع ها
 - طراحی جدول عکس ها برای محل
 - طراحی جدول محل ها
 - طراحی جدول رابط
- دستورات
 - تمپلیت ها
 - خواندن فایل به صورت ایستا
 - متد ها
 - مسیریابی
 - تنظیمات پنجره مدیریت
 - نمایش اطلاعات در صفحه
- چگونگی نصب
 - حداقل سیستم
 - نرم افزار و کتابخانه های ضروری

○ نرم افزار های غیر ضروری ولی مهم

• راهنمای کاربردی

• پیشنهاد و نتیجه گیری

• ضمائم

• منابع و مآخذ

قبل از شروع درباره خود پروژه تصمیم دارم کمی درباره ی جنگو و برنامه نویسی جنگو صحبت بکنم. همان طور که توضیح داده شده جنگو یکی از بزرگترین کتابخانه های زبان قدرت مند پایتون است که در کمک رسانی هر چی بیشتر برنامه نویس در طراحی هر چه آسان تر یک وبسایت روز به روز در حال قوی تر شدن است.

اما نقش برنامه نویس کوچکی چون من در این طراحی و توسعه چیست؟

اگر با CMS های طراحی وبسایت آشنا باشید حتما با وردپرس آشنایی دارید. وردپرس یک رابط بین کاربر و کد ها است که با استفاده از چند کلیک میتوانید یک وبسایت را کاملا بالا بیاورید.

اما جنگو وردپرس نیست، بلکه شباهات بسیار زیادی به آن دارد. این شباهت ها به این معنا نیست که شخص هیچ گونه کدی نمیزد، بلکه جنگو بسیاری از کارهای روتینی که هر برنامه نویس مجبور است در هر پروژه تکرار کند به طوری که آن کد ها هیچ فرقی با هم ندارند را جمع آوری کرده و از قبل در قالب یک پروژه نیمه آماده در اختیار ما قرار میدهد

این کار های روتین که لازم نیست همه آن را فرا بگیرند یا در هر پروژه آن را از اول شروع به نوشتن بکنند عبارت اند از

web sockets : برای این که یه پروژه تحت وب اجرا شود برنامه نویس حتما باید وب سوکت را بلد باشد تا با استفاده از پورت های شبکه خط انتقال دیتا را کنترل کند.

templates : برای کوتاه کردن و خوانا کردن هر چه بیشتر کد ها برنامه نویس مجاب است این بخش را خودش پیاده سازی کند که کاری بسیار سخت و دشوار است.

Data Base : برای ساخت هر گونه وب سایتی حداقل به چند جدول یا رابط نیاز منید که این ها هم باید توسط کاربر طراحی شود.

و...

تمامی این مواردی بخشی از قدرت مندی جنگو است به نحوی که تمام آن ها توسط برنامه های بسیار خلاق در سرار دنیا به بهترین نحو انجام شده و ما کفایت از آنها استفاده کنیم.

با توجه به توضیحات بالا ممکن است در کد ها قطعه کدی باشد که دقیقا ندانم چه کار خاصی را انجام میدهند ولی به طور کلی میدانم چه نوع فعالیتی انجام میدهد.

در اینجا توضیح مختصری از پایتون گفته میشود و در ادامه با تعریف کامل تری از پایتون آشنا میشویم.

یک زبان برنامه نویسی همه منظوره، سطح بالا، شیءگرا، اسکریپتی و متن باز است که توسط خیدو فان روسوم در سال ۱۹۹۱ در کشور هلند طراحی شد. فلسفه ایجاد آن تأکید بر دو هدف اصلی خوانایی بالای برنامه های نوشته شده و کوتاهی و بازدهی نسبی بالای آن است. کلمات کلیدی و اصلی این زبان به صورت حداقلی تهیه شده اند و در مقابل کتابخانه هایی که در اختیار کاربر است بسیار وسیع هستند.

بر خلاف برخی زبان های برنامه نویسی رایج دیگر که بلاک های کد در آکولاد تعریف می شوند (به ویژه زبان هایی که از گرامر زبان سی پیروی می کنند) در زبان پایتون از نویسه فاصله و جلو بردن متن برنامه برای مشخص کردن بلاک های کد استفاده می شود. به این معنی که تعدادی یکسان از نویسه فاصله در ابتدای سطرهای هر بلاک قرار می گیرند، و این تعداد در بلاک های کد درونی تر افزایش می یابد. بدین ترتیب بلاک های کد به صورت خودکار ظاهری مرتب دارند.

پایتون مدل های مختلف برنامه نویسی (از جمله شیءگرا و برنامه نویسی دستوری و تابع محور) را پشتیبانی می کند و برای مشخص کردن نوع متغیرها از یک سامانه پویا استفاده می کند.

این زبان از زبان های برنامه نویسی مفسر بوده و به صورت کامل یک زبان شیءگرا است که در ویژگی ها با زبان های تفسیری پرل، روبی، اسکیم، اسمال تاکو تی سی ال مشابهت دارد و از مدیریت خودکار حافظه استفاده می کند.

پایتون پروژه ای آزاد و متن باز توسعه یافته است و توسط بنیاد نرم افزار پایتون مدیریت می گردد.

پایتون به شکل گسترده‌ای به کار گرفته شده و پشتیبانی می‌شود

زبان پایتون دو ویژگی یک زبان خوب یعنی محبوبیت و کاربرد گسترده را توأمان با یکدیگر در اختیار دارد. کافی است به آمارهای منتشر شده از سوی منابع معتبری همچون **Tiobe** و پروژه‌هایی که روی سایت گیت‌هاب قرار گرفته و با پایتون نوشته شده‌اند نگاهی داشته باشید تا متوجه شوید این زبان تا چه اندازه نزد طراحان محبوب است.

برنامه‌های نوشته شده با زبان پایتون روی سیستم عامل‌ها و سکوها‌ی اصلی و سیستم عامل‌های خاص‌تر به خوبی اجرا می‌شوند. بخش اعظمی از کتابخانه‌های بزرگ و سرویس‌های مبتنی بر **API** به اشکال مختلفی پیوندهای مرتبط با زبان پایتون را در خود جای داده‌اند، به طوری که به زبان پایتون اجازه داده‌اند از طریق واسط‌ها با این سرویس‌ها ارتباط برقرار یا به طور مستقیم از کتابخانه‌ها استفاده کند. در حالی که پایتون را در گروه سریع‌ترین زبان‌های برنامه‌نویسی نمی‌توان قرار داد و شاید کند بودن نقطه ضعف اصلی این زبان به شمار می‌رود، اما در مقابل تطبیق‌پذیری بسیار بالایی دارد.

پایتون زبانی نیست که برای انجام کارهای عادی و پیش پا افتاده مورد استفاده قرار گیرد. از این زبان به منظور ساخت برنامه‌های کاملاً حرفه‌ای با کیفیت بالا، برنامه‌های مستقل و سرویس‌های وب می‌توان استفاده کرد. اسکرپت‌هایی که با این زبان نوشته می‌شوند، به سادگی قادرند فرآیندهای بزرگی را مدیریت و خودکارسازی کنند.

از پایتون در ارتباط با برنامه‌نویسی‌های عادی و رایج نیز می‌توان استفاده کرد

طراحان و توسعه‌دهندگان نرم‌افزار این توانایی را دارند تا هر دو گروه برنامه‌های کنسولی و گرافیکی را با پایتون ایجاد و آن‌ها را به شکل خوداجرا مستقر کنند. پایتون به طور ذاتی این توانایی را ندارد تا یک فایل باینری مستقل را از یک اسکرپت ایجاد کند. اما پکیج‌های ثالثی شبیه به **cx_Freeze** یا **PyInstaller** این کاستی پایتون را جبران کرده‌اند.

زبان پایتون در ارتباط با یادگیری ماشینی و علم داده‌ها نیز به کار گرفته می‌شود

در چند سال اخیر فرآیند تجزیه و تحلیل داده‌های مرتبط با فناوری اطلاعات بیش از اندازه پیچیده شده است، به همین دلیل زبان پایتون و در تعقیب آن زبان آر به ستارگان یک‌ه‌تاز این میدان تبدیل شده‌اند. با توجه به محبوبیت بیش از اندازه زبان پایتون امروزه شاهدیم که طیف گسترده‌ای از کتابخانه‌های مورد استفاده در یادگیری ماشینی و علم داده‌ها یک واسطه یا به عبارت دقیق‌تر رابط‌های ویژه زبان پایتون را ارائه کرده‌اند.

پایتون در ارتباط با وب سرویس‌ها و توابع RESTful نیز به کار گرفته می‌شود

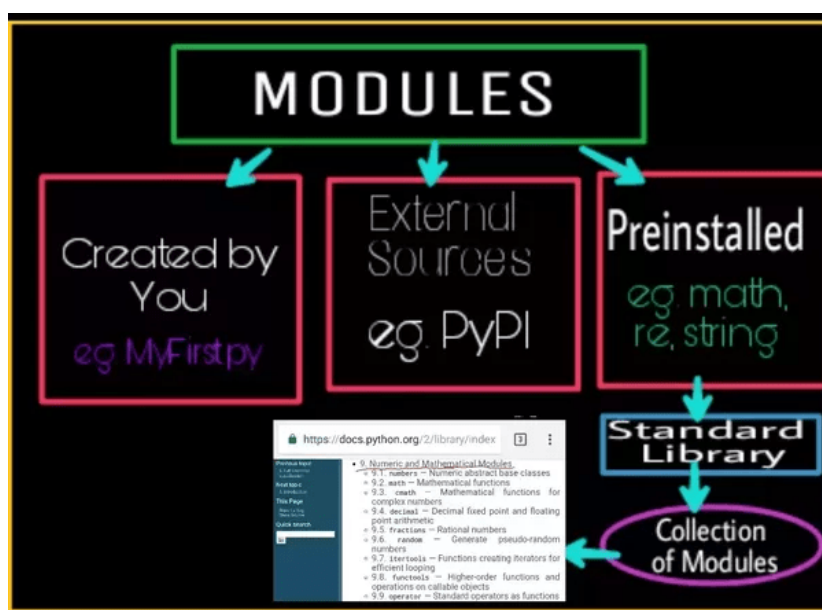
کتابخانه‌های محلی زبان پایتون به همراه چهارچوب‌های وب بخش ثالث سریع‌ترین و راحت‌ترین راهکار را در اختیار طراحان وب قرار داده‌اند تا بدون دردسر خاصی بتوانند توابع REST چندخطی یا یک سایت داده‌محور مملو از اطلاعات را با استفاده از زبان پایتون طراحی کنند (REST. سرنام Representational State Transfer) یک معماری وب سرویس است که از پروتکل HTTP برای انتقال اطلاعات میان کلاینت و سرور استفاده می‌کند. جدیدترین نگارش‌های عرضه شده از زبان پایتون به شکل قدرتمندی از عملیات غیرهم‌زمان پشتیبانی می‌کند. پشتیبانی از عملیات غیرهم‌زمان به معنای آن است که سایت‌ها قادرند ده‌ها هزار درخواست در هر ثانیه را از طریق کتابخانه‌های درستی که مورد استفاده قرار می‌دهند مدیریت کنند.



کتابخانه چیست؟

شاید مفهوم کتابخانه و معنی آن کمی پیچیده به نظر برسد. ولی در حقیقت کتابخانه مجموعه ای از ماژول هاست. کتابخانه یا شامل ماژول هایی است که یا زبان C و یا با زبان Python نوشته شده اند. پس بهتر است ابتدا به تعریف ماژول بپردازیم.

به مجموعه ای از قطعه کدهای استاندارد یا بسته های مستقل که می توانند برای ساخت یک ساختار پیچیده تر مورد استفاده قرار گیرند ماژول می گویند. به زبان ساده تر، ماژول مجموعه ای از خطوط کد است که برای یک هدف خاص استفاده می شود و می تواند در برنامه های متعددی از آن استفاده شود که این کار باعث پرهیز از تکرار می شود که در برنامه نویسی بسیار مهم است.



حال که با مفهوم کتابخانه ها آشنا شدیم بهتر از با ۱۰ کتاب خانه مهم و ضروری در پایتون نیز آشنا بشوید.

۱: Requests معروفترین کتابخانه در زمینه http که توسط kenneth reitz نوشته شده است. یادگیری این کتابخانه به همه علاقه مندان پایتون توصیه می گردد.

۲: Scrapy اگر شما مشغول برنامه نویسی وب هستید استفاده از این کتابخانه را فراموش نکنید. مطمئناً بعد از استفاده از آن نمی توانید از کتابخانه دیگری استفاده کنید.

۳: wxPython ابزاری برای کار با gui است. یکی از جایگزین های بسیار مناسب tkinter.

۴: Pillow یک کتابخانه کمکی بر روی PIL (Python Imaging Library) که بسیار User Friendly تر از PIL است. برای برنامه نویسانی که با تصاویر کار می کنند شدیداً توصیه می شود.

۵: SQLAlchemy کتابخانه ای برای کار با دیتابیس های گوناگون. خیلی ها عاق این کتابخانه هستند و البته مخالفان زیادی هم دارد. انتخاب با خود شما! (من شخصا از این کتابخانه استفاده کرده ام و البته راضی هستم).

۶: BeautifulSoup این کتابخانه در زمینه html و xml کاربرد دارد خصوصاً برای افراد مبتدی. که البته کمی کند است.

۷: Twisted مهمترین ابزار برای هر برنامه نویس برنامه های حوزه شبکه. شامل api بسیار زیبایی است

۸: NumPy ارائه دهنده ویژگی های پیشرفته ریاضی در پایتون که بناید فراموشش کنید.

۹: SciPy کتابخانه ای از الگوریتم ها و ابزارهای پیشرفته ریاضی برای پایتون است که بسیاری از دانشمندان را مجبور به مهاجرات از زبان روبي به پایتون کرده است.

۱۰: matplotlib یک کتابخانه ترسیم عددی. برای دانشمند حوزه اطلاعات و یا هرگونه تجزیه و تحلیل داده ها بسیار مفید است.

جنگو (Django) یک فریم ورک سطح بالا، رایگان و Open Source برای ساخت Web Application ها می باشد که در پایتون نوشته شده است. با استفاده از فریم ورک جنگو می توانید وب اپلیکیشن های خود را آسان و سریع تر توسعه دهید. هدف از ساخت این فریم ورک اتصال اجزای مشابه سایت است که به دیتابیس نیاز دارند مانند: ثبت نام، ورود و خروج از سیستم، پنل مدیریت، فرم ها، آپلود فایل ها و غیره. با استفاده از فریمورک جنگو نیازی به نوشتن کد های اضافی ندارید.

فریمورک جنگو بسیاری از ویژگی های پایتون را به ارث برده است. این فریم ورک قادر به ساخت وب سایت های پیچیده و حرفه ای در سریع ترین زمان و با امنیت بالا می باشد.

فریمورک جنگو از ساختار Model-View-Controller تبعیت می کند به همین دلیل کد های مربوط به بخش های کنترلی (Controller)، بخش داده ها (Model) و بخش مربوط به رابط کاربری (View) از هم جدا هستند

The image shows the Django logo, which consists of the word "django" in a white, lowercase, sans-serif font. The logo is centered on a dark green rectangular background.

تعریف دقیقی از جنگو

جنگو یکی از بهترین فریم ورک‌های پایتون است که امکان ساخت سایت‌های پیچیده را در زمانی کوتاه فراهم خواهد کرد. با استفاده از ماژول‌هایی که در این فریم ورک وجود دارد می‌توانید بدون انجام کدنویسی اختصاصی صفر تا صد، قابلیت‌هایی که در جنگو وجود دارد را به راحتی فعال کرده و پیاده سازی کنید. در ادامه به بررسی برخی مزایای استفاده از جنگو می‌پردازیم.

۱: افزایش سرعت برنامه نویسی

از آنجایی که جنگو یک فریم ورک است و پکیج‌های متعددی نیز در آن وجود دارد که روز به روز نیز این پکیج‌ها در حال بیشتر شدن هستند، استفاده از آن می‌تواند سرعت طراحی و پیاده سازی سایت را چندین برابر کند. با توجه به اینکه زمان گذاشتن روی پروژه باعث بالا رفتن هزینه‌ها خواهد شد، استفاده از django می‌تواند، از هزینه‌های زیاد و همچنین هدر رفت زمان جلوگیری کند.

۲: دسترسی کافی به پکیج‌های مختلف

django مدام در حال ارائه پکیج‌های مختلف برای استفاده در فریم ورک خود است که هر یک از این پکیج‌ها برای پیاده سازی یک قابلیت خاص تعریف شده‌اند. اگر در کتابخانه جنگو جستجو کنید به بیش از ۴,۰۰۰ پکیج دست پیدا خواهید کرد که می‌توانید از آنها استفاده کرده و با سرعت بیشتری کار روی پروژه را پیش ببرید.

۳: استفاده در سایت‌های بزرگ

شاید جالب باشد بدانید که فریم ورک جنگو در سایت‌هایی مثل اینستاگرام، بخش‌هایی از فیسبوک و پینترست استفاده شده است. توسعه روی جنگو به شدت در حال بیشتر شدن است و به لطف همین به‌روزرسانی مداوم می‌توانید برای ساخت سایت‌هایی بزرگ از آن استفاده کنید.

۴: تمرکز روی سئو و بهینه سازی django

با توجه به رویکردی که برنامه نویس سایت و سئو کار سایت دارند و کار هر دو جدا از هم است اما مکمل هم هستند، جنگو به شکلی ساخته شده است که ضمن کدنویسی استاندارد موارد مربوط به سئو را نیز

رعایت کند. ساختار URL یکی از این نمونه‌ها است. آدرس صفحات در django به شکلی است که برای کاربر انسانی قابل درک باشد، بنابراین برای موتورهای جستجو نیز بهینه بوده و سایت می‌تواند سئو مناسبی داشته باشد.

همچنین یکی دیگر از مقوله‌های سئو که داشتن سرعت بالا می‌باشد در این فریم ورک رعایت شده است. البته داشتن سرعت بالای سایت صرفاً به فریم ورک بر نمی‌گردد و باید در تمامی بخش‌های یک سایت موارد مربوط به سرعت بالا و کاهش مدت زمان لود صفحات را رعایت کرد.

۵: کارایی بالا و امکان توسعه

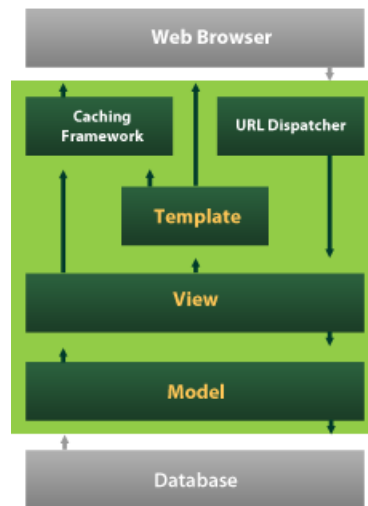
از میان فریم ورک‌های متعددی که امروزه استفاده می‌شود، جنگو برای شروع می‌تواند جزو اولین لیست‌هایی باشد که از آن استفاده کنید. طبیعی است که توسعه سایت به مرور برای شما به یک نیاز تبدیل خواهد شد. بنابراین component های مختلف جنگو این امکان را خواهند داد که به راحتی آن را توسعه دهید. بیش از ۴۰۰۰ component تا به امروز برای جنگو ساخته شده است که با قرار دادن در کنار فریم ورک می‌توانید از آنها برای توسعه سایت استفاده کنید. این مقدار مشخصاً همه نیازهای شما را برطرف خواهد کرد.

۶: داشتن امنیت بالا

در مقایسه php و python یکی از موضوعاتی که باعث زیر سوال بردن امنیت خواهد شد این است که اگر زمانی در کد برنامه نویسی php خطایی رخ دهد، مسیر خطا با جزییات کامل آن نمایش داده می‌شود. این مسئله باعث شناسایی دایرکتوری‌ها و نوع کدنویسی در برخی از موارد خواهد شد. اما در جنگو چنین حالتی وجود ندارد و اگر خطایی وجود داشته باشد، به صورت یک پیغام از طریق تمپلیت‌ها برای مرورگر ارسال خواهد شد.

۷: پشتیبانی از معماری MVC

در معماری MVC اولین واژه نشانگر کامپوننت MODEL است که مربوط به قسمت اطلاعات و داده‌ها یا همان پایگاه داده‌ها می‌باشد V. نمایانگر کامپوننت VEIW است که در واقع وظیفه رندر کردن قسمت مدل برای کاربر را برعهده دارد یا به زبان ساده و کلی وظیفه نمایش به کاربر را انجام می‌دهد C. نشانگر کامپوننت CONTROLLER است، قسمتی که درخواست‌ها را از کاربر دریافت می‌کند و با تعامل بین دو بخش دیگر مدل و ویو پاسخ مناسب به درخواست را ارسال می‌کند.



در تکنولوژی MVC که جنگو از آن استفاده می کند، ظاهر نمایشی (تگ های HTML یا template سایت در فایلی جدا ذخیره می شود. بخش کنترلی نیز به عنوان یک ماژول پایتون ایجاد و ذخیره خواهد شد. بنابراین برنامه نویس با بخش کنترلی و طراح با بخش html سروکار خواهند داشت که این مسئله باعث آمیختگی ظاهر نمایشی با کدنویسی خواهد شد.

۸: قرارگیری اطلاعات پایگاه داده در مسیری درست

نوشتن اطلاعات دیتابیس مربوط به اتصال پایگاه داده در داخل کد اصلی برنامه کار درستی به نظر نمی رسد. چرا که این اطلاعات تنظیمات اصلی و حساس یک سایت هستند که در صورت خوانده شدن این فایل از هر طریقی باعث ایجاد مشکلات امنیتی خواهد شد. برای همین جنگو یک محل جدا و واحد برای ذخیره اطلاعات دیتابیس و سایر تنظیمات را دارد که به راحتی می توان تغییراتی مثل انتخاب نوع پایگاه داده را در آن انجام داد.

در صورتی که در بیشتر نرم افزارهای توسعه داده شده توسط PHP این اطلاعات دقیقا در ساختار برنامه نویسی قرار دارند.

۹: پشتیبانی و در دسترس بودن

به دلیل استفاده زیادی که این روزها از جنگو می شود، نگرانی بابت رفع مشکلات خود نخواهید داشت. انجمن های پشتیبانی و فروم های مختلفی هستند که می توانید به کمک آنها مشکلات خود را برطرف کنید. علاوه بر این به دلیل استفاده بالایی که جنگو به خود گرفته است، مقالات آموزشی و ویدئوهای فراوانی در سطحی بسیار وسیع وجود دارد که می توانید از آنها برای پیاده سازی و توسعه سایت خود کمک بگیرید.

یکی از ابزارهای ذخیره و بازیابی اطلاعات، SQLite نام دارد. این نرم افزار مشهورترین سیستم ذخیره فایلی اطلاعات به شمار می رود. شهرت SQLite به دلیل پشتیبانی از انواع مختلف سیستم عامل ها از جمله ویندوز، لینوکس، آندروئید و مک او اس و همچنین رایگان و قدرتمند بودن آن است.

SQLite با زبان C برنامه نویسی شده است و به طور پیوسته در حال بهبود و توسعه است. به همین دلیل سرعت و کارایی بسیار بالایی دارد. در نگارش های جدید که در آینده منتشر خواهند شد، بهینه سازی های گسترده ای روی این سیستم به انجام رسیده است که سرعت عملکرد آن را بیش از پیش افزایش داده است.

پایگاه داده در سیستم

ساختمان داده یا به طور دقیق تر تعریف ساختمان داده در جنگو با دیگر زبان ها و کتابخانه ها فرق های بزرگ و اساسی دارد حداقل تا آنجایی که من میدانم.

به این شکل **code first** است ، به این طریق که شما تمام جداول و **relation** های مورد نیاز خود را نوشته و جنگو است که جداول را میسازد. البته این طریق برای استفاده از دیتابیس **sqlite** است، ولی شما مثل تمام زبان های دیگر مختارید در انتخاب دیتابیس.

بعد از طراحی نمونه اولیه از **structure** دیتا بیس وقت آن است که تبدیل به جداول شوند

برای پروژه سیستم اطلاعات جامع استانی بعد از انجام مراحل بررسی موجودیت ها تصمیم بر آن شد که شامل ۴ جدول اصلی که شامل

✓ استان ها

✓ محل ها

✓ عکس ها

✓ و موضوع ها

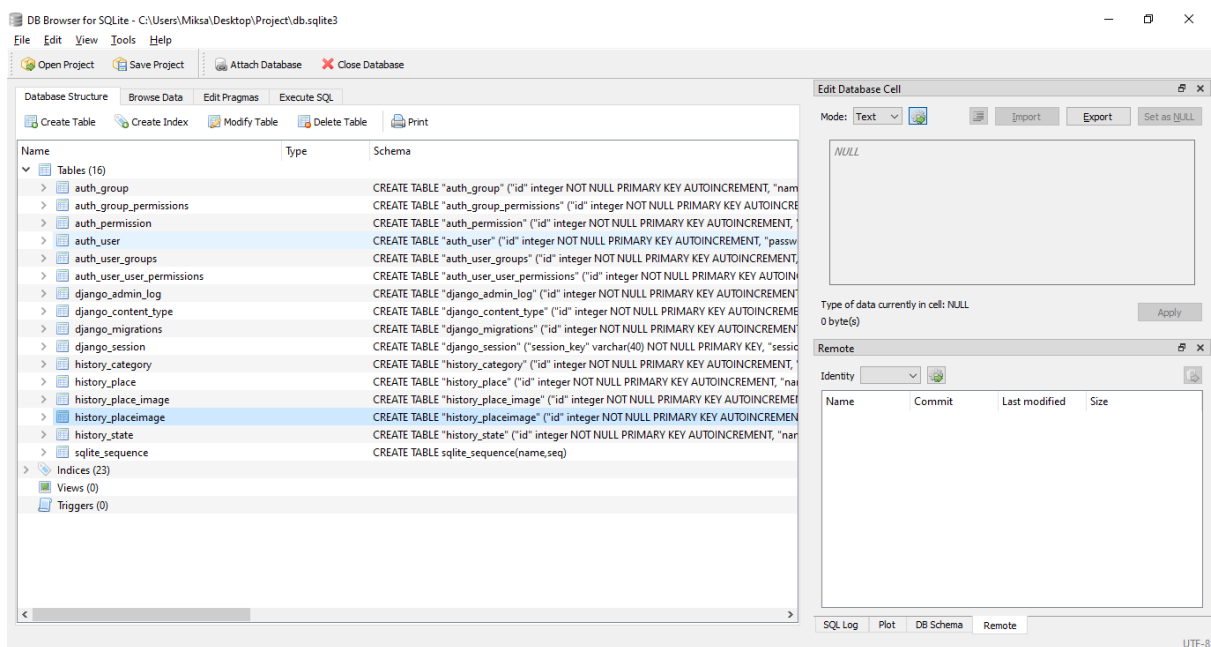
و یک جدول رابط که رابط بین عکس ها و محل ها بود ساخته شود

به بررسی دقیق تر روی تک تک موجودیت ها باید صفات آنها نیز درست میشد تا در آینده به مشکلی
برنخوریم. البته جدوالی جنگو میسازد هم ۵ تا جدولی که ما میخوایم نمیسازد

جنگو بالغ بر ۱۱ جدول دیگر به صورت اتوماتیک میسازد. این جداول شامل جداول

- ✓ کاربران
- ✓ گروه ها
- ✓ دسترسی ها
- ✓ لاگ ها
- ✓ سیشن ها
- ✓ و...

برای همین موارد است که جنگو در ساخت یک پروژه بسیار سریع و کارآمد است. به طوری که برنامه
نویس نیازی به طراحی دیتا بیس برای کاربران یا ... نیست و فقط کافیسیت از آنها استفاده کند یا در مسائل
تخصصی تر آنها را ویرایش کند، چون جنگو تمام آن چیزی که ممکن است برنامه نویس به آن برخورد برا
پیش بینی و لحاظ کرده.



در تصویر بالا لیستی از تمام جداول ساخته شده توسط جنگو را میبینیم که شامل جداول طراحی شده برای
پروژه نیز است.

همان طور که گفته شد طراحی جداول در جنگو متاول به بقیه زبان ها یا کتاب خانه ها نیست، به این صورت که با زدن کل های پایتون سیستم برای شما جدول درست میکند نه دستورات T-SQL

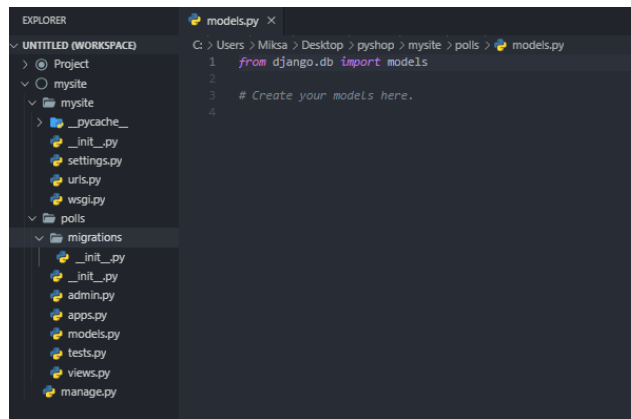
خب پس پیدا کردن موجودیت ها و صفات آنها شروع به کد زدن میکنیم اما کجا باید کد زد؟؟؟

وقتی یک پروژه با جنگو شروع میکنید، جنگو در دایرکتوری تعدادی فایل میسازد که فایل در جای خود کار خاصی میکند که بعدا تک تک آن هارا توضیح خواهیم داد

در تصویر بالا تعداد فایل و دایرکتوری که جنگو به صورت اتوماتیک میسازد را میبینیم ولی فعلا در این بخش قصد توضیح همه این عناوین رو نداریم

خوب همان طور که گفته شد قصد بر این بود که جداول دیتابیس را طراحی کنیم، فایلی که با آن کار داریم در دایرکتوری یکی از **app** های است که ساختیم به نام **models.py**

در بخش تعریف جنگو نیز گفته شده بود که جنگو مبتنی بری (MVC (Model-view-controller است که به این معناست که کد های هر بخش از یکی دیگر جدا قرار دارند



تصویر مربوط به صفحه models.py بعد از شروع پروژه است

ساخت جدول استان ها

خب همان طور که گفته شد برای درک بهتر موجودیت ها باید سعی در طراحی هر چی بهتر صفات بشود به طوری که جامعیت اطلاعاتی داشته باشیم و از افزونگی پرهیز کند.

با همه این تفاسیر برای موجودیت استان در سیستم اطلاعات جامع نیاز مند چند صفت است با استفاده از ارتباطات سعی در جامعیت اطلاعات میکنیم که در اینجا آن صفات را بررسی میکنیم.

Id: این صفت برای یکایی هر استان طراحی شده به طوری که به صورت یکتا در سیستم ذخیره شود.

این صفت کلید اصلی P.K این جدول به شمار می آید. و درموجودیت های بعدی از آن به عنوان کلید خارجی یعنی F.K استفاده میشود.

این صفت بصورت خودکار برای همه جداول به صورت پیش فرض توسط جنگو گذاشته شده بنابراین نیازی به تغییر یا تعریف موردی در سیستم یا کد نیست.

Name: این نیز برای قرار گرفتن نام استان طراحی شده

این صفت باید نام پارسی استان هارا در خود ذخیره کند پس رد موقع تعریف باید ویژگی به این صفت داده شود که قابلیت ذخیره سازی به کلمات و حروف پارسی را دارا باشد.همچنین این صفت نیاز مند

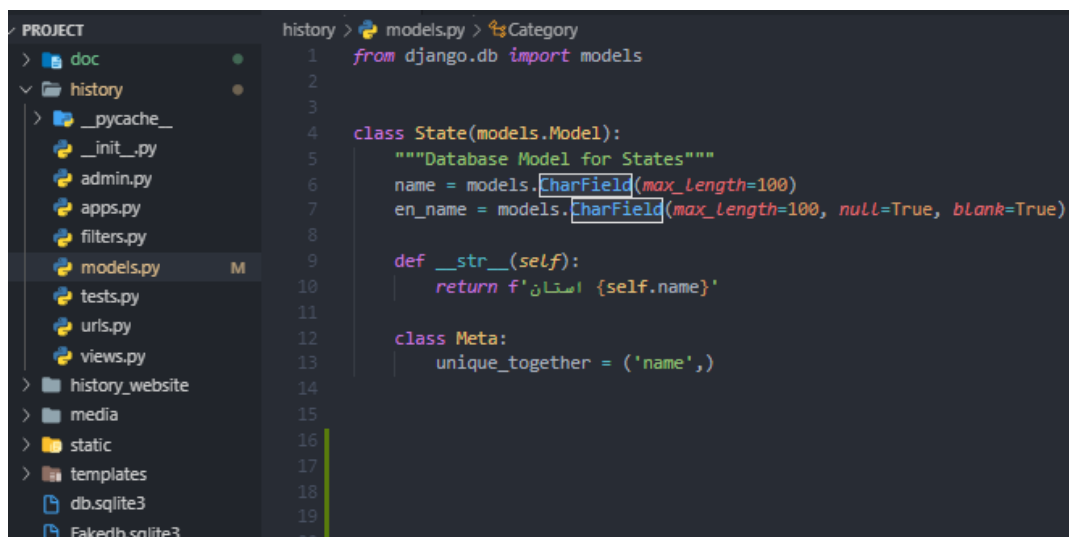
ذخیره سازی به صورت رشته های بلند نیست، زیرا فقط نام یک استان در آن ذخیره میشود مانند: گیلان، کهگیلویه و بویراحمد و...

En-name: روشی که بنده در فیلتر کردن اطلاعات داشتیم از طریق **url** سایت است و در این بخش نمیتوان تصمیم گرفتیم نام استان به صورت انگلیسی باشد.

البته قادر بودم در این بخش از **id** نیز استفاده کنم ولی برای درک بهتر کاربر از برنامه و قوی تر شدن **UX** و همچنین خوانا تر شدن **url** سایت تصمیم بر این شد که از **en-name** استفاده شود.

این صفت نیز شباهت های با صفت **name** دارا است و میتوان از همان ویژگی ها استفاده کرد.

خوب در این مرحله شروع به کد زدن در بخش **models.py** میکنیم.



```
PROJECT history > models.py > Category
1 from django.db import models
2
3
4 class State(models.Model):
5     """Database Model for States"""
6     name = models.CharField(max_length=100)
7     en_name = models.CharField(max_length=100, null=True, blank=True)
8
9
10 def __str__(self):
11     return f'استان {self.name}'
12
13 class Meta:
14     unique_together = ('name',)
15
16
17
18
19
20
```

باری شروع با کلاسی نوشته شود که میخوايد نام جدولمان باشد. بنابراین کلاس در اینجا همان جدول که میخوايد در دیتابیس ساخته شود و مغیر های این جداول همان صفات جداول هستند و **magic function** های نیز نوع عملکرد جداول.

خب میخوايم همان طور که گفته شد باید صفت **name** طراحی شود، با فراخوانی کتابخانه **models** میتوان ویژگی صفت خود را انتخاب کنید، در اینجا چون نیاز به ذخیره سازی حروف پارسی بود از متد **CharField** استفاده شده تا تمام کارکترها بدون مشکل ذخیره شوند.

در این صفت نیاز مند ذخیره سازی رشته های بلند نبوده پس تصمیم گرفتم طول کارکتر ها قابل ذخیره شدن را به ۱۰۰ کارکتر محدود کنم.

نوبت صفت `en-name` است، این صفت همانند صفت `name` دارای ویژگی های مشترک است از جمله طول یا متد ذخیره سازی تنها فرقی که با `name` داراست این است که صفت میتواند خالی باشد، که در این صورت سیستم به جای استفاده از `en-name` در `url` از همان `id` که بصورت خودکار پر میشود استفاده میکند. بنابراین ویژگی هایی `null` و `blank` را مساوی با `true` قرار دادم.

حال به `magic function` های میرسیم، قبل از شروع این `function` ها مجبورم توضیحات مختصری درباره `magic function` های بدهم تا با اصل قضیه آشنا شوید.

مجیک فانکشن ها، فانکشن های از پیش تعریف شده ی خود و پایون است که کار های بخصوصی انجام میدهند. این نوشتن کار ها ممکن است برای برنامه نویس کمی طولانی و بدون نتیجه باشد پس در `base` زبان پایتون این `function` ها تعریف شده.

در اینجا قصد در توضیح همه فانکشن ها ندارم ولی آنهایی که استفاده شده اند را تا جایی که بلد باشم توضیح میدهم. در ادامه لیستی از تمام مجیک فانکشن های مهم پایون میگذارم.




`__str__` ✓
`__init__` ✓
`__repr__` ✓
`__unicode__` ✓
`__format__` ✓
`__hash__` ✓
`__dir__` ✓

فانکشن `__str__` : این فانکشن به این صورت عمل میکند که هر جا نمونه ای این یک کلاس ساخته شد مقادر بازگشتی همان یک متغیر به صورت رشته ای یک رشته قابل تغییر و تعریف باشد در جدول استان ها تصمیم گرفتم رشته ای به شکل زیر باز گرداند

استان + `name`

به عنوان مثال : اگر در سیستم تهران ذخیره شده باشد هر جا که از `state` استفاده شود، مقدار "استان تهران" باز میگردد.

کلاس meta : این کلاس هم مانند همان magic fuction ها عمل میکند که در اینجا صفت name از طریق این کلاس یکتا کردم.

history_state		CREATE TABLE "history_state" ("id" integer NOT NULL PRI
 id	integer	"id" integer NOT NULL PRIMARY KEY AUTOINCREMENT
 name	varchar(100)	"name" varchar(100) NOT NULL
 en_name	varchar(100)	"en_name" varchar(100)

در تصویر بالا جدول ساخته شده توسط جنگو را در دیتابیس با فیلد های طراحی شده میبینیم




نکته: اسم جدول در بالا گفته شده state است ولی در دیتابیس میبینیم ذخیره شده history_state این به این دلیل است که هر پروژه جنگو ممکن است شامل چندین app باشد که هر app به معنا سیستم های مختلف کل پروژه است، بنابراین ممکن است هر سیستم یا app نام جداول یکسانی داشته باشد از این طریق یکتایی جداول در دیتابیس که مخزنی برای نگه داری تمامی جداول در تمامی app ها است حفظ میشود

توجه: از اینجا به بعد در مورد ساخت جداول تعریف مختصر تری داده میشود، زیرا به نظرم در بخش طراحی جدول استان ها توضیحات کاملی بیان شد ولی اگر طول توضیح با ویژگی جدیدی آشنا شویم به طور مفصل توضیح خواهم داد.

ساخت جدول موضوع ها

برای کامل کردن هر چه بهتر اطلاعات ما از مکان های ثبت شده ، نیازمند این بودیم که بدانیم هر محلی که در سیستم ذخیره میشود جزو کدام دسته از مکان های گردش گری است. این یعنی کاربری که از سیستم استفاده میکند با دستی باز واطلاعات کامل بتواند محل سفر خود را انتخاب کند.

برای این موضوع جدولی طراحی شده که این جدول نیز مانند جدول استان ها همان کار یکسان را میکند. یعنی اطلاعاتی که در داخل این جدول ذخیره میشود به تنهایی قابل استفاده نیست، پس نیاز مند صفات زیادی نیست.

history_category		CREATE TABLE "history_category" ("id" integer NOT NULL PRI
 id	integer	"id" integer NOT NULL PRIMARY KEY AUTOINCREMENT
 name	varchar(40)	"name" varchar(40) NOT NULL
 en_name	varchar(40)	"en_name" varchar(40) NOT NULL

خوب همان جور که میبینید این جدول نیز شباهات فراوانی با جدول استان ها دارد. اسم این جدول به اسم category نام گذاری شده و دارای ۳ صفت است.

Id : این صفت برای یکایی هر موضوع طراحی شده به طوری که به صورت یکتا در سیستم ذخیره شود.

این صفت کلید اصلی P.K این جدول به شمار می آید. و درموجودیت های بعدی از آن به عنوان کلید خارجی یعنی F.K استفاده میشود.

این صفت بصورت خودکار برای همه جداول به صورت پیش فرض توسط جنگو گذاشته شده بنابر این نیازی به تغییر یا تعریف موردی در سیستم یا کد نیست.

Name : این نیز برای قرار گرفتن نام موضوع طراحی شده

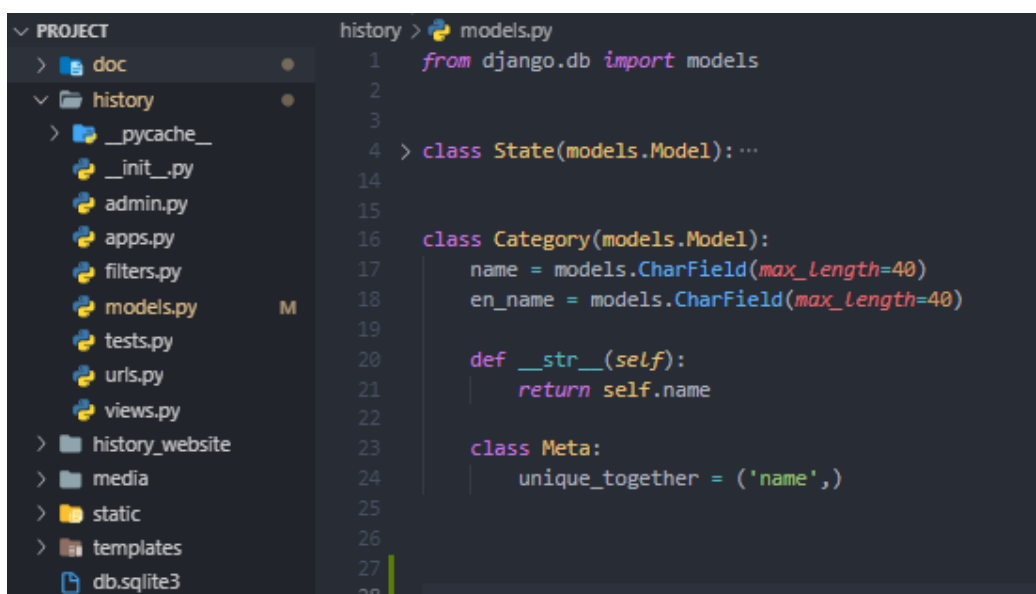
این صفت باید نام پارسی استان هارا در خود ذخیره کند پس رد موقع تعریف باید ویژگی به این صفت داده شود که قابلیت ذخیره سازی به کلمات و حروف پارسی را دارا باشد. همچنین این صفت نیاز مند ذخیره سازی به صورت رشته های بلند نیست، زیرا فقط نام یک موضوع در آن ذخیره میشود مانند: تاریخی، طبیعی و...

این صفت رشته های طولانی از کارکتر هارا نیاز ندارد ذخیره کند پس نیاز ندارد حافظه ای بزرگی را اشغال کند.

در این سیستم ما موضوع هارا به چهار گروه اصلی مذهبی، طبیعی، تاریخی، و تفریحی تقسیم کردیم ولی این به معنای آن نیست که این سیستم فقط به درد توریست میخورد ما با اضافه کردن گروه هایی مثل مکان های سیاسی، دولتی، روستوران ها یا... میتوانیم به قدرت مند کردن هرچه بیشتر این سیستم کمک کنیم.

En-name : این صفت نیز مانند صفت en-name در جدول استان ها به همین منظور طراحی شده که در بخش UX قوی تر نمایان شود.

این صفت در ویژگی طول نیز از صفت name تابعیت میکند و فرقی با آن ندارد.



```
history > models.py
1  from django.db import models
2
3
4  > class State(models.Model): ...
14
15
16  class Category(models.Model):
17      name = models.CharField(max_length=40)
18      en_name = models.CharField(max_length=40)
19
20      def __str__(self):
21          return self.name
22
23      class Meta:
24          unique_together = ('name',)
25
26
27
28
```

خوب به models.py میریم و صفات گفته شده را مینویسیم.

اسم این جدول category نام گذاری شده است دارای دو صفت name و en-name است. از کتابخانه models متد CharField را فراخوانی میکنیم و مقدار ۴۰ کارکتر مجاز را برایش تعریف میکنیم.

مانند کلاس state از مجیک فانکشن __str__ برای بازگرداندن یه رشته استفاده میکنیم

و از کلاس از پیش تعیین شده ی Meta به عنوان یکتا کردن صفت name استفاده میکنیم.

ساخت جدول عکس های محل

همان طور که میدانید هر مکان در صفحه مختص به آن شامل چندین عکس از آن مکان است که این یعنی ماهیت ارتباطی آن به هر مکان یک ارتباط یک به چند یا 1:n است.

برای این منظور مجبور با ساخت یک جدول جدا برای نگه داری تک تک این عکسا شدم.

این جدول مانند دو جدول یکسان است یعنی اطلاعاتی که در آن قرار میگیرد یک آدرس عکس است و یک نام که برای راحت تر خواندن عکس در بخش وارد کردن محل جدید استفاده میشود.

اسم این جدول را به نام **PleceImage** نام گذاری کردم که بعدا با جدول **Place_Image** دچار مشکل نشود. زیرا جدول **Place_name** یک جدول ارتباطی بین جدول **place** و **PleceImage** است و ماهیت **1:n** را برقرار میکند

همان طور که میدانید هر مکان در صفحه مختص به آن شامل چندین عکس از آن مکان است که این یعنی ماهیت ارتباطی آن به هر مکان یک ارتباط یک به چند یا **1:n** است.




برای این منظور مجبور با ساخت یک جدول جدا برای نگه داری تک تک این عکسا شدم، بنابراین جدولی ساختم که این جدول شامل صفتی به عنوان کلید خارجی یا **F.K** است که در داخل این صفت کلید اصلی موجودیت محل در آن قرار دارد. از این طریق توانستم ارتباط یک به چند را برقرار بکنم.

فعلا جدول **PleceImage** رو توضیح بدم و بعد توضیح کامل در مورد جدول **Plece** که مهمترین جدول ماست ، جدول **Place_Image** را توضیح خواهم داد که یک جدول رابط بین دو جدول است.

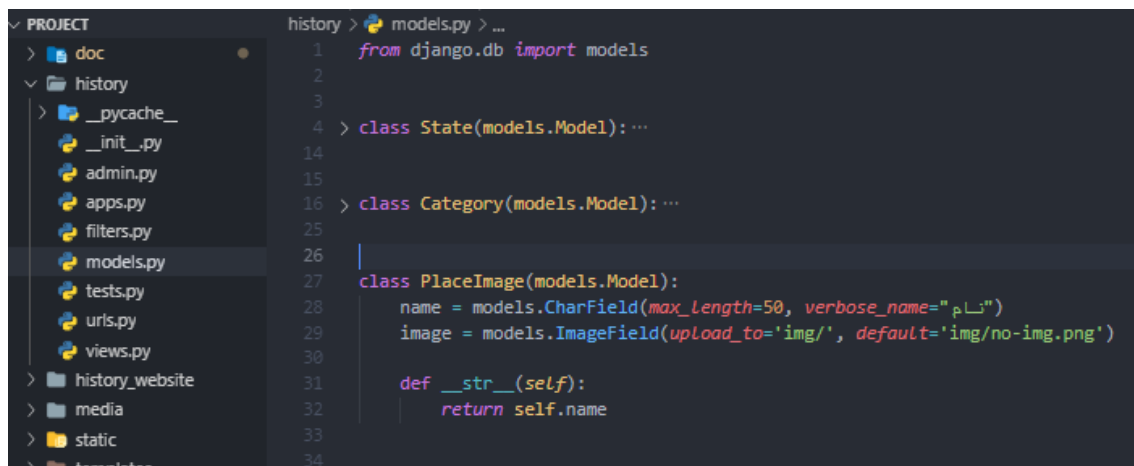
همان طور که گفته شد این جدول پیچیدگی خاصی ندارد و تنها نکته آن این است که شما با استفاده از پنل ادمین هر چه تعداد عکس بخواهید میتوان اضافه کنید و با قرار دادن یک نام به عکس آنرا در موقعی که میخواهید محل جدید اضافه کنید شناسایی کنید و چندین عکس را برگزینید.

این جدول شامل صفت **name** است که همانطور گفته شد برای شناسایی آن استفاده میشود. این صفت مانند تمام صفات نام که در جداول قبلی بود نیازمند ذخیره طولانی رشته ها نیست

صفت بعدی این جدول به نام **Image** نام دارد که آدرس ذخیره شده ی عکس در پروژه را داراست، این صفت باید قابلیت ذخیره سازی رشته های بلند را دارا باشد ، چون ممکن است نام عکس به اندازه قابل توجهی طولانی باشد.

history_placeimage		CREATE TABLE "history_placeimage" ("id" integer NOT NULL PRI
 id	integer	"id" integer NOT NULL PRIMARY KEY AUTOINCREMENT
 name	varchar(50)	"name" varchar(50) NOT NULL
 image	varchar(100)	"image" varchar(100) NOT NULL

در بالا تصویر ساخته شده جدول در دیتابیس توسط جنگو را مبینم و همان طور که گفته شد دارای خصوصیت **id** است که ما آن را تعریف نکردیم.



```
1 from django.db import models
2
3
4 > class State(models.Model): ...
14
15
16 > class Category(models.Model): ...
25
26
27 class PlaceImage(models.Model):
28     name = models.CharField(max_length=50, verbose_name="نام")
29     image = models.ImageField(upload_to='img/', default='img/no-img.png')
30
31     def __str__(self):
32         return self.name
33
34
```

و اما بخش کد نویسی آن کلاسی ساخته میشود به نام **PlaceImage** که نام جدول ما است و صفت **name** که یکی از متغیرها ما است.

تنها چیزی که در اینجا در جداول قبلی فرق میکند مقدار **verbose_name** است که عملکرد یکسانی با **magic function** ها دارد بهطوری که هر جا **PlaceImage.name** صدا زده شود مقدار کارکتری **name** بازگردانده میشود که این مورد فقط در پنل ادمین وجود دارد ، برای خوانا تر شدن هرچه بهتر پنل مدیریت.

صفت **image** همان طور که گفته شده بود در آن آدرس مهم ذخیره ی عکس است. از کتابخانه **models** متد **ImageField** را فرا خوانی میکنیم که مخصوص ذخیره سازی عکس است، مقداری که میگیرد ۱ : **upload_to** است که آدرس دقیق دایرکتوری که میخوايد در آن ذخیره شود، دقت فرمایید مسیری که عکس ها در آن ذخیره میشود پوشه **static** در شاخه اصلی پروژه است و من با قرار داد پوشه **img** در این دایرکتوری محل ذخیره سازی را انتخاب کردم.

۲: **default** که در صورت انتخاب نکردن یک عکس توسط کاربر به صورت خودکار یک عکس از پیش تعیین شده جای آن را میگیرد.

و در انتها مجیک فانکشن **__str__** که قبلا توضیحات کاملی از آن بیان شد.

ساخت جدول محل ها

همان طور که قبلا گفته شد مهمترین جدول در این سیستم جدول محل ها است. اطلاعاتی که در این جدول ذخیره میشود مجموعه ای اطلاعاتی که در ۴ جدول دیگر ذخیره میشود است، که شامل استان ، نوع و عکس های آن است.

این جدول شامل صفات زیادی است که در زیر آنها را بررسی میکنیم

صفت **name** : باید نام محل مورد نظر در آن قرار بگیرد و قابلیت نگه داری رشته ها با طول متوسط را داشته.

صفت **description** : این صفت باید توضیحاتی در مورد مهم در خود نگه دارد این صفت باید دارا قابلیت نگه داری رشته ها به صورت نامحدود باشد.

صفت **url** : این صفت باید نگهدارند یک لینک باشد این لینک در پروژه همان صفحه گوگل مپ است که مختصات دقیق آن مکان را در سیستم نگه داری میکند.

صفت **state** : این صفت یک کلید خارجی یا به اصطلاح F.K است. در اینجا ما نگه دارند کلید اصلی یا P.K جدول **state** ها هستیم که از این طریق میتوان از هر مکان پی به استان آن مکان برد. و در صورت انتخاب استان به تمام مکان هایی که در استان ثبت شده برسیم.

صفت **category** : این صفت مانده صفت **state** یک F.K است که نگه دارند P.K جدول **category** است. از این طریق میتوان محل ها را بر اساس نوع آنها طبقه بندی کرد، یعنی کاربر فقط محل های تاریخی را ببیند و میتوان در صورت داشتن یک محل به نوع آن محل که چه نوع است پی برد.

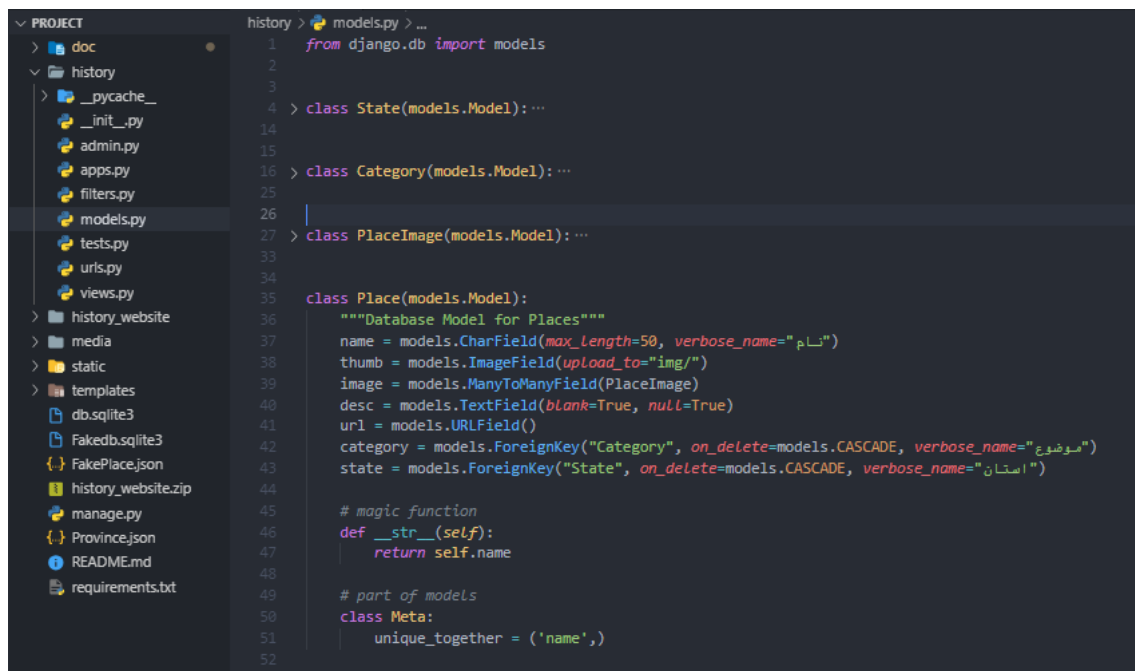
صفت **thumbnail** : صفتی است که در آن یک عکس قرار میگیرد، همان عکس که در صفحه لیست محل ها بر روی اتم آن میبینیم. اینکار برای این انجام شد که یک عکس شاخص بر روی تمام اتم ها قرار گرفته باشد.

صفت **image** : این صفت همان طور که گفته شده بود برای اتصال چندی عکس به یک محل درست شده، البته هنوز به بررسی جدول **Piece_image** که جدول رابط است نرسیدیم و در جای خود توضیح داده خواهد شد.

history_place		CREATE TABLE "history_place" ("id" integer NOT NULL PRI
id	integer	"id" integer NOT NULL PRIMARY KEY AUTOINCREMENT
name	varchar(50)	"name" varchar(50) NOT NULL
thumb	varchar(100)	"thumb" varchar(100) NOT NULL
desc	text	"desc" text
url	varchar(200)	"url" varchar(200) NOT NULL
category_id	integer	"category_id" integer NOT NULL
state_id	integer	"state_id" integer NOT NULL

در تصویر بالا جدو ساخته شده توسط جنگو در دیتا بیس را میبینیم.

و اما میرسیم به بخش کد نویسی در فایل models.py



```

1  from django.db import models
2
3
4  > class State(models.Model): ...
14
15
16 > class Category(models.Model): ...
25
26
27 > class PlaceImage(models.Model): ...
33
34
35 class Place(models.Model):
36     """Database Model for Places"""
37     name = models.CharField(max_length=50, verbose_name="نام")
38     thumb = models.ImageField(upload_to='img/')
39     image = models.ManyToManyField(PlaceImage)
40     desc = models.TextField(blank=True, null=True)
41     url = models.URLField()
42     category = models.ForeignKey("Category", on_delete=models.CASCADE, verbose_name="موضوع")
43     state = models.ForeignKey("State", on_delete=models.CASCADE, verbose_name="استان")
44
45     # magic function
46     def __str__(self):
47         return self.name
48
49     # part of models
50     class Meta:
51         unique_together = ('name',)
52

```

همان طور که در دتصور میبینید کلاس ساخته شده به نام place که نمایان گر اسم جدول است.

صفت thumbnail که در اینجا به اختصار thumb تعریف شده و از متد ImageField برای ذخیره

کردن عکس استفاده شد

صفت description که به اختصار desc در اینجا تعریف شده باید مقدار نامحدودی از رشته را در خود

ذخیره کند برای این کار از متد های کتابخانه models متد TextField انتخاب شد که این قابلیت را

داراست. همچنین برای این ویژگی null پذیر تعریف شده تا کاربر بتواند مقایر خالی در سیستم ذخیره کند.

صفت url که همان طور گفته شد باید یک لینک در خود ذخیره کند و با استفاده از متد URLField اینکار

را امکان پذیر کردم.

خوب می‌رسیم به صفت **state** این صفت همان طور که گفته شد باید یک **P.K** از یک جدول دیگر در خود ذخیره کند بنابراین در بین متد ها، از متد **ForeignKey** استفاده کردم، این متد در **parameters** های ورودی خود نام یک جدول را می‌گیرد که در اینجا جدول **state** داده شده است.

دیگر **parameters** های این متد ورودی **on_delete** است که به این معناست اگر خواسته شد محلی پاک شود سیستم **state** مربوط به آن محل را پاک نکند.

صفت **category** هم دقیقاً شبیه به صفت **state** است و هیچ گونه تفاوتی با آن ندارد، تنها فرقی که با **state** داراست این است که در بخش انتخاب یک جدول برای **F.K** باید آن را به جدول **category** متصل کرد تا ارتباط به درستی انجام شود.




دقت فرمایید در دیتا بیس فیلدی به نام **Image** وجود ندارد دلیل این را در بخش بعد توضیح خواهد داد.

ساخت جدول رابط

این جدول رابط بین جدول **Placeimage** و **Place** قرار دارد تا بتواند از این طریق ارتباط را برقرار کند، دقت داشته باشید ساخت این جدول توسط ما انجام نمیشد و خود جنگو این جدول را می‌سازد پس دنبال کد آن در فایل **Models.py** نگردید.

همان طور که در بخش قبل گفته شده بود فیلد **Image** جدول **Place** وجود ندارد، میدانید که در آنجا از ارتباط **n:m** استفاده شده و جنگو با استفاده از همان صفت و متد **ManyToManyField** جدول رابط را به نام **Place_Image** ساخته.

طرز کار این جدول به این شکل است که کلید اصلی جدول **Place** را فیلد **Place_id** خود قرار میدهد و در فیلد بعدی که **PlaceImage_id** است کلید اصلی عکس های ذخیره شده در جدول **PlaceImage** را قرار میدهد و با استفاده از کلید اصلی خود جدول، تمام عکس هایی که به نام آن محل ثبت شده را بارگزاری میکند.

history_place_image			CREATE TABLE "history_place_image" ("id" integer NOT NULL PRI
	id	integer	"id" integer NOT NULL PRIMARY KEY AUTOINCREMENT
	place_id	integer	"place_id" integer NOT NULL
	placeimage_id	integer	"placeimage_id" integer NOT NULL

دستورات و کدهای اجرایی

به مهم ترین بخش این مستند میرسیم، قسمت دستورات و کدهای که باعث اجرا برنامه میشه، توجه داشته باشید که در این قسمت بنده همه ی کدهای رو نردم یعنی پس از شروع پروژه با استفاده از جنگو، خود جنگو اکثر تنظیمات هارا از قبل انجام میده و برنامه نویس فقط کافیس رو پروژه ی خود تمرکز کند.

در اینجا قصد نداریم تمام تنظیمات یا کدهای زده شده را توضیح دهیم والا برای این کار اصلا مدیا نوشته مناسب نیست و حتما باید در فیلم و به صورت عملی آن را نشان داد دوما برای این کار فک کنم ۱۰-۱۵ ساعت باید ویدیو آموزشی ساخت تا تمام کارهایی که انجام شده را توضیح و دلیل آن را بگوییم. اگر علاقه مند به نحوه کار جنگو هستید در بخش [منابع و مآخذ](#) میتوانید آنها را دنبال کنید.

تمپلیت ها

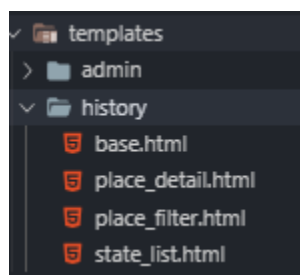
یکی از مهم ترین بخش های سایت **templates** است، زیرا تمام آن چیزی که کاربر میبینید یعنی کدهای HTML و CSS در اینجا قرار دارد.

خب همان طور که میدانید پروژه دارای چند صفحه است که و هر صفحه با صفحات دیگر در بخش هایی مثل **header** ، **head** ، **footer** و... مشترک است و بخش **content** یا محتوا هست که با یکدیگر متمایز هستند، حال فرض کنید ی تکه از قطعه کد در بخش مشترک را تغییر یا حذف کنیم؟

خب اگر با **template** ها آشنا نباشید طبیعتا در هر صفحه شروع به پیدا کردن آن کد میکنید و آن را تغییر میدهید. شاید برای ۱ یا ۲ صفحه کاری چندان سخت نباشد ولی اگر تعداد صفحات شما به ۱۰-۱۵ یا حتی بیشتر برسد میخوايد چه کاری بکنید؟

اینجاست که تمپلیت ها به داد ما میرسند، شما میتوانید به نگه داشت یک قطعات مشترک کد در یک صفحه آنرا به یک صفحه مرجع تبدیل کنید و تنها با گذاشتن پرچم در این صفحه content هایی که هر کدام با یک دیگر متمایز هستند دار راهنمایی کنید که در کجا بنشینند

البته این همه مزیت templates نیست، یکی دیگر از مزیت های templates های کمتر شدن کد ها و در نتیج سریع تر شدن سرعت بارگزاری پروژه است. در ضمن از این طریق کد ها هم دارای تمیزی قابل توجهی میشود و شما دچار مشکلات ساده و پیش پا افتاده برنامه نویسی نخواید شد.

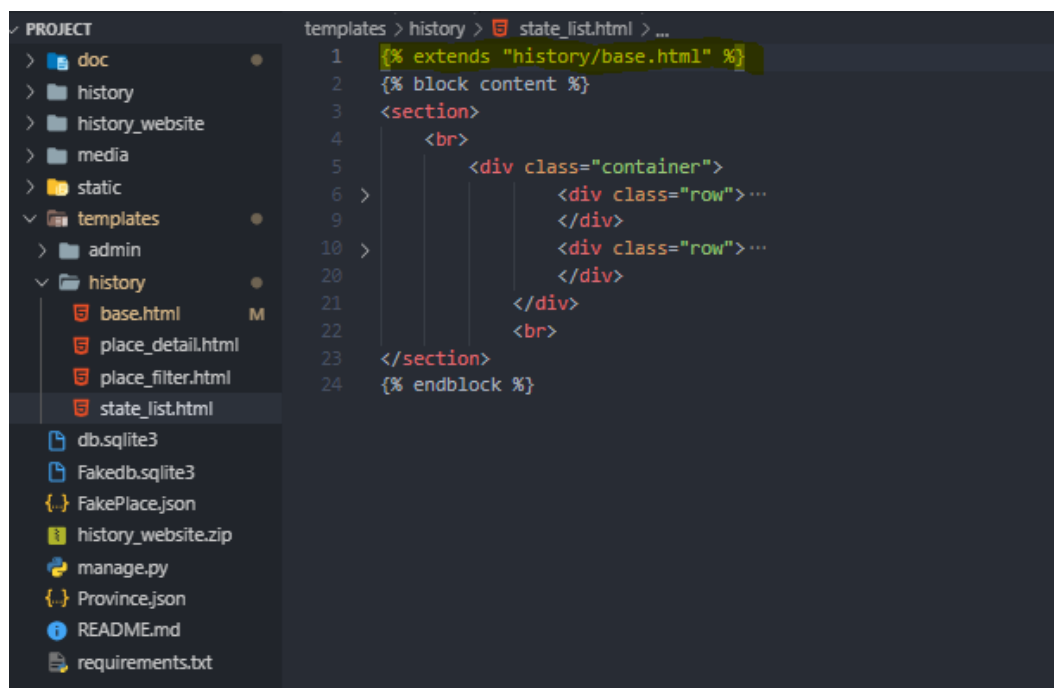


همان طور که در تصویر میبینید در این پروژه چهار صفحه طراحی شده که یکی از آنها به نام base.html صفحه مرجع است و سه صفحه دیگر یعنی place_detail.html ، place_filter.html و state_list.html که صفحات content هستند در صفحه مرجع جایگذاری میشوند.

```
PROJECT templates > history > base.html > html > body > script
1 {% load static %}
2 <!DOCTYPE html>
3 <html lang="en">
4
5 <head>...
27 </head>
28
29 <body>
30
31 <!-- Page Preloader -->
32 <div id="preloader">...
33 </div>
34
35 <!-- Header section -->
36 <header class="header-section">...
37 </header>
38
39 <!-- Header section end -->
40
41 <!-- Page info -->
42 <div class="page-info-section set-bg" data-setbg="{% static 'index/img/page-bg/2.png' %}">...
43 </div>
44 <!-- Page info end -->
45
46 <!-- search section -->
47 <section class="search-section ss-other-page">...
48 </section>
49
50 <!-- search section end -->
51
52 <!-- footer section -->
53 <footer class="footer-section pb-0">...
54 </footer>
55 <!-- footer section end -->
56
57 <script src="https://cdnjs.cloudflare.com/ajax/libs/uikit/3.1.6/js/uikit.min.js"
58 integrity="sha256-v789mr/z8bgR53mfydCI78CSAF+9+nRqu+JRfs1UPg0=" crossorigin="anonymous"></scr
59 <script src="https://cdnjs.cloudflare.com/ajax/libs/uikit/3.1.6/js/uikit-icons.min.js"
60 integrity="sha256-1+AmZG1Fz41J+gms80qC7fas1J0berZDhjEsmDmQy8s=" crossorigin="anonymous"></scr
61
62 db.sqlite3
63 Fakedb.sqlite3
64 FakePlace.json
65 history_website.zip
66 manage.py
67 Province.json
68 README.md
69 requirements.txt
```

خب همان طور که میبینید در این فایل **base.html** تمام کد های **HTML** و تمام کتابخانه های استفاده شده قرار دارد این همان صفحه مرجعی است که گفته شده بود.

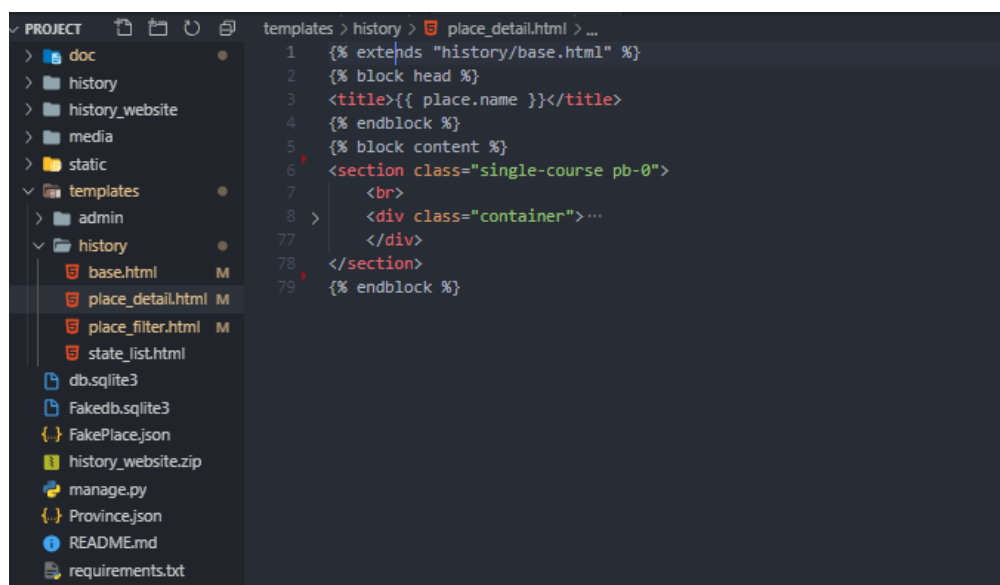
اگر به وسط صفحه دقت کنید بخشی را زرد رنگ کردم این همان پرچمی است که توضیح داده ام



```
1 {% extends "history/base.html" %}
2 {% block content %}
3 <section>
4   <br>
5   <div class="container">
6     <div class="row">...
9   </div>
10    <div class="row">...
20  </div>
21 </div>
22 <br>
23 </section>
24 {% endblock %}
```

خب یکی از صفحات **content** است که میبینید، همان طور که با رنگ زرد مشخص کردم این صفحه به صفحه مرجع **base.html** وصل میشود و به طوری کلی نمایش میدهد

اگر دقت کنید این صفحه هیچ یک از کد های اساسی برای بارگزاری یک صفحه ی استاندارد را ندارد یعنی بدون تگ های مهم **<html>** ، **<body>** ، **<head>** و...



```
1 {% extends "history/base.html" %}
2 {% block head %}
3 <title>{{ place.name }}</title>
4 {% endblock %}
5 {% block content %}
6 <section class="single-course pb-0">
7   <br>
8   <div class="container">...
77 </div>
78 </section>
79 {% endblock %}
```

The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like 'doc', 'history', 'history_website', 'media', 'static', 'templates', 'admin', and 'history'. The 'history' folder is expanded, showing files like 'base.html', 'place_detail.html', 'place_filter.html', and 'state_list.html'. The 'place_filter.html' file is selected, and its content is displayed in the code editor. The code is a Jinja2 template that extends 'history/base.html' and includes a static file. It defines a section class 'course-section' and a container class 'container'.

```

1 {% extends "history/base.html" %}
2 {% load static %}
3 {% block head %}
4 <title>سرچ</title>
5 {% endblock %}
6 {% block content %}
7 <section class="course-section">
8     <br>
9     <div class="container">
10         <div class="course-warp">...
11     </div>
12 </section>
13 {% endblock %}

```

دو تصویر بالا دو فایل دیگر content هستند که در فایل مرجع جای میگیرند.

خواندن فایل ها به صورت ایستا

The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like 'doc', 'history', 'history_website', 'media', 'static', 'templates', 'admin', and 'index'. The 'index' folder is expanded, showing files like 'css', 'icon-fonts', 'img', 'js', 'style.css', and 'templates'. The 'templates' folder is selected, showing files like 'base.html', 'index.html', and 'state_list.html'. The 'base.html' file is selected, and its content is displayed in the code editor. The code is a Jinja2 template that includes a static file and defines a section class 'course-section'.

```

1 {% load static %}
2 <!DOCTYPE html>
3 <html lang="en">
4
5 <head>
6     {% block head %}
7     <title>kia - Iranoni</title>
8     {% endblock %}
9     <meta charset="UTF-8">
10    <meta name="description" content="WebUni Education Template">
11    <meta name="keywords" content="webuni, education, creative, html">
12    <meta name="viewport" content="width=device-width, initial-scale=1.0">
13    <!-- Favicon -->
14    <link href="{% static 'index/img/favicon.ico'" rel="shortcut icon" />
15    <!-- Stylesheets -->
16    <link rel="stylesheet" href="{% static 'index/css/bootstrap.min.css'" />
17    <link rel="stylesheet" href="{% static 'index/css/font-awesome.min.css'" />
18    <link rel="stylesheet" href="{% static 'index/css/owl.carousel.css'" />
19    <link rel="stylesheet" href="{% static 'index/css/style.css'" />
20    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css">
21    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
22    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/js/bootstrap.min.js"></script>
23    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/uikit/3.0.3/css/uikit.min.css">
24 </head>
25
26 <body>...
27 </body>
28 </html>

```

یکی دیگر از مطالب مهمی که باید بیان میشد لود کردن فایل ها به صورت ایستا است به نحوی که با تغییر مسیر پروژه فایل های پروژه بهم نخورد ، این کار را از طریق خط هایی که زرد رنگ شده نشان داده ام. با این روش در صورت نوشتن کلمه static قبل آدرس سیستم پوینتر خواندن فایل را به پوشه static در شاخه اصلی پروژه می شناسد.

نوبت به مسیریابی در سایت میرسد. اهمیت این بخش بسیار بالاست زیرا این بخش اگر به خوبی عملگردد خود را نشان ندهد پروژه دچار مشکلات فراوانی میشود. از جمله این مشکلات عدم دسترسی به محل یا استان مورد نظر و... برخی مشکلات دیگه

اما اهمیت این بخش به این قسمت منتهی نمیشود و زیرا در این بخش به به خوبی سیستم MVC جنگو آشنا میشویم (قبلا از MVC توضیحاتی داده شده بود اگر آشنایی ندارید پیشنهاد میشود به بخش [تعریف دقیق تر از جنگو](#) بروید).

همان قبلا گفته شده بود در اینجا قصد ندارم تمام کتابخانه هایی که استفاده کردم رو توضیح بدهم زیرا این قسمت در تمام پروژه ها جنگو وجود دارد، یعنی مهم نیست سطح و وسعت پروژه شما در چه حدی باشد، همین که در آن از مسیریابی و متد های مربوطه استفاده کرده باشید یعنی آشنایی کامل با نحوه اضافه کردن کتاب خانه ها دارید.

بریم سر اصل مطلب همان طور که در قسمت های قبل دیدین این پروژه شامل ۳ صفحه میشود که این یعنی کل پروژه در همین سه صفحه خلاصه میشود ، با همین منطق طبیعتا کاربر ۳ صفحه قابل جا به جایی در آن ها را دارد ، که این یعنی ۳ url باید ساخته شود و هر url نیز باید متصل به یک متد مشخص باشد تا بتواند از این طریق بتوانیم صفحات HTML را با دیتا های خودمان پر کنیم یا...

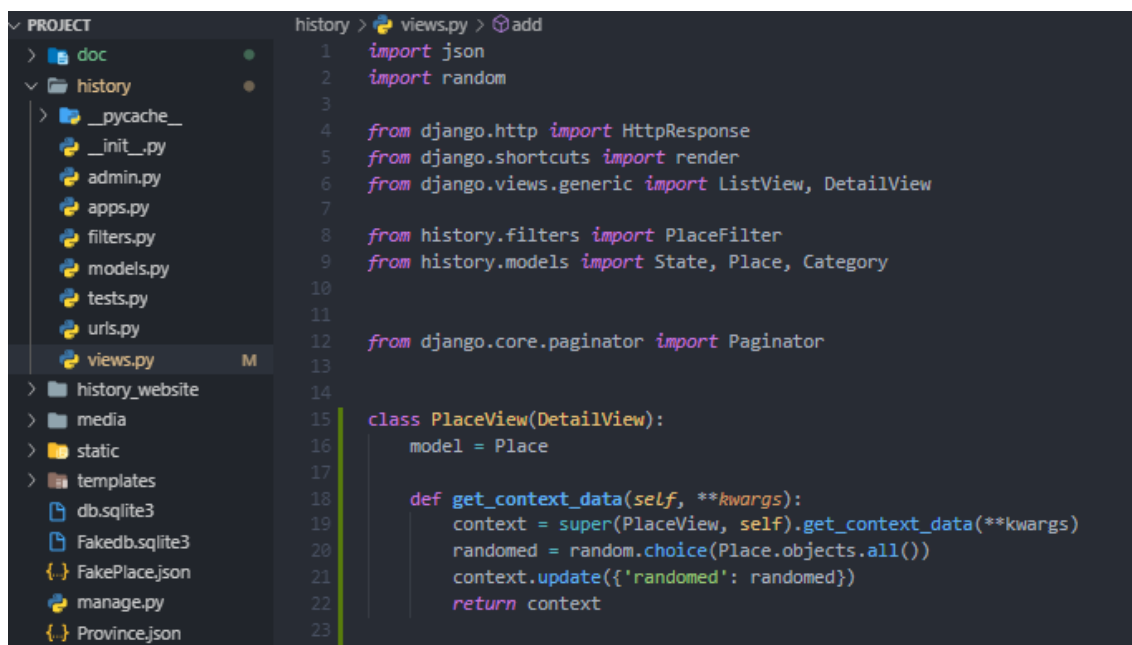
قبلا دیدم که صفحه ای به نام place_detail.html را داریم، ماهیت صفحه همان طور که از اسمش پیداست نمایش دهنده جزئیات یک محل است مانند گالری تصاویر، توضیحات و... ولی جنگو یا مروگر ما از کجا باید بفهمد که کدام مکان را برای ما به نمایش بگذارد؟

جواب این سوال در مبانی علم شبکه و طراحی صفحات وب است، به این صورت که ما میتوانیم با استفاده از url سایت به سیستم بگوییم چه چیزی میخوایم، برای توضیح بهتر ترجیح میدهم با یک مثال این موضوع را شرح دهم.

فرض کنید که میخوایم جزئیات محل شماره ۱۶ را در صفحه ببینیم برای این کار با استفاده url به سیستم بگوییم خب به این صورت به سیستم میگوییم

/place/16

این عدد ۱۶ یا هر عددی که بعد از آن قرار میگیرد همان id در جدول place است، از این طریق میتوانیم مشخص کنیم چه چیزی را میخواهیم.



```
1 import json
2 import random
3
4 from django.http import HttpResponse
5 from django.shortcuts import render
6 from django.views.generic import ListView, DetailView
7
8 from history.filters import PlaceFilter
9 from history.models import State, Place, Category
10
11
12 from django.core.paginator import Paginator
13
14
15 class PlaceView(DetailView):
16     model = Place
17
18     def get_context_data(self, **kwargs):
19         context = super(PlaceView, self).get_context_data(**kwargs)
20         randomized = random.choice(Place.objects.all())
21         context.update({'randomed': randomized})
22         return context
23
```

همان طور که در تصویر میبینید کلاس PlaceView برای این طراحی شده که بتواند با توجه به آن url که توضیح داده شد محل مورد نظر را نمایش دهد.

البته فایل views.py به تنهایی کار نمیکند به این صورت که در تعریف دقیقی از ادرس url نشده که این در بخش بعد یعنی تشریح فایل urls.py میگنجد.

همان طور که میبینید در بخش import های این فایل میبینیم که models.py و تمام آن جداول که توضیح داده شد در این صفحه اضافه شده است، پس از این طریق میتون از object های آن کلاس استفاده کرد.

نکته دیگری که در این صفحه قابل توجه است این است که از کتابخانه رندوم استفاده شده، دلیل این برای بخش "نمیدونم کجا برم؟؟؟" در صفحه اصلی است با استفاده در اختیار داشتن تمام object ها models محل ها به صورت رندوم یک مورد را انتخاب میکند و در اخر تمام این اطلاعات return میشود.

مورد بعدی که یکی از متد های **views** است، متد **state_list** است این متد در صفحا اصلی به کار میرود در همان قسمتی که لیست کامل تمام استان ها را به نمایش میگذارد.

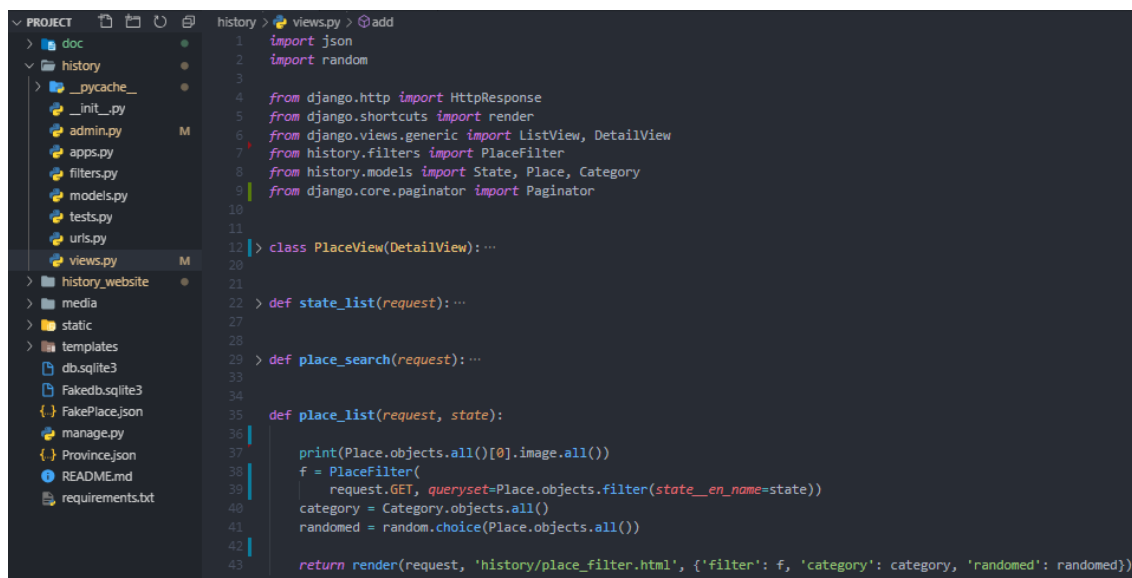
توجه فرمایید که **state_list** هم مانند **Place** عمل میکند یعنی برای به نمایش گذاشتن تمام محل هایی که در استان مورد نظرم آن ها را از طریق **url** فیلتر میکند

/state/Tehran

با این تفاوت که در **place** برای پیدا کردن از **id** آن مکان استفاده میشد ولی در **state** از **en_name** که در قسمت **models** ها تعریف کردیم

متد بعدی که **place_search** است این متد برای این طراحی شده که بتوانیم در بخش سرچ سایت از آن استفاده کنیم و محل مورد نظر خودمان را بین تمام محل ها پیدا کنیم.

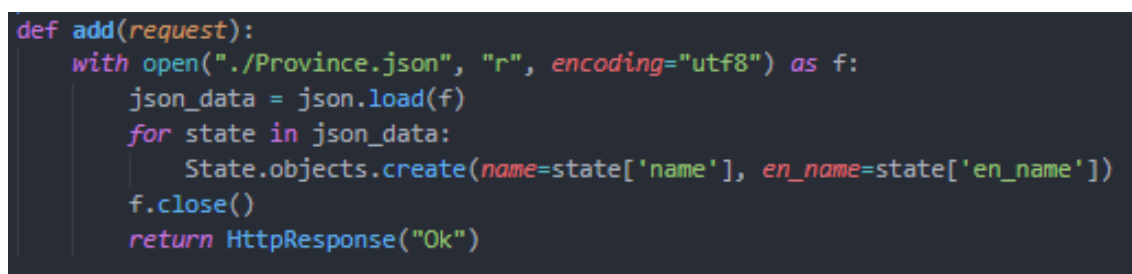
مقداری که این متد باز میگرداند مساوی با تمام محل ها است ولی بعدا این این object ها توسط form که داخل صفحه است محدود و فیلتر میشود، این را در بخش کد های اجرایی به طور کامل توضیح خواهیم داد.



```
1 import json
2 import random
3
4 from django.http import HttpResponse
5 from django.shortcuts import render
6 from django.views.generic import ListView, DetailView
7 from history.filters import PlaceFilter
8 from history.models import State, Place, Category
9 from django.core.paginator import Paginator
10
11
12 class PlaceView(DetailView):...
13
14
15
16
17
18
19
20
21
22 > def state_list(request):...
23
24
25
26
27
28
29 > def place_search(request):...
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

متد بعدی، متد `place_list` است این متد با استفاده از url سایت که هرچی باشد یعنی `/state/Tabriz` یا... محل ها را بر اساس آن `state` فیلتر میکند.

موقعی که شما در صفحه ای اصلی بر روی یکی از `state` ها کلیک میکنید اگر به url خود دقت کنید تغییر میکند و این زمان است که این متد شروع به کار میکند و تمام محل هایی که از آن استان انتخاب کرده اید را به شما نشان میدهد.



```
def add(request):
    with open("./Province.json", "r", encoding="utf8") as f:
        json_data = json.load(f)
        for state in json_data:
            State.objects.create(name=state['name'], en_name=state['en_name'])
        f.close()
    return HttpResponse("Ok")
```

و اما متد اخر که متد بسیار ساده ای است، البته این را باید متذکر بشوم که این متد در روند اجرایی پروژه هیچ گونه دخالتی ندارد و فقط برای راحتی حال شما در اضافه کردن تمام استان های ایران نوشته شده است. همان طور که از ظاهر این متد پیداست تمام object های فایل `Province.json` را میخواند و در `models State` مینویسد.

همان طور که گفته شد مسیر یابی یکی از مهم ترین بخش های این پروژه است، حال به دور از توضیحات اضافی شروع به تفسیر کد ها میکنیم.

خب کد های این بخش در فایل `urls.py` قرار دارد، ولی همان جوری که فایل `views.py` کاربردی خاصی به جز تعریف کار ها کار خاص دیگری انجام نمیدهد و مکمل این آن `urls.py` است که میتواند متد ها را قابل دسترسی کند.

همان طور که گفته شد میبینیم در قسمت `import` ها تمام متد های `views` اضافه شده

خب در `urlpatterns` تمام `path` های که ممکن است کاربر به آن ها مراجعه کند را نسبت به متد ها نوشته ام.

`url` اول همان صفحه اصلی سایت است که متدی در آن اجرا میشود، این متد `state_list` است، همان متدی که لیست کامل استان هارا باز میگرداند.

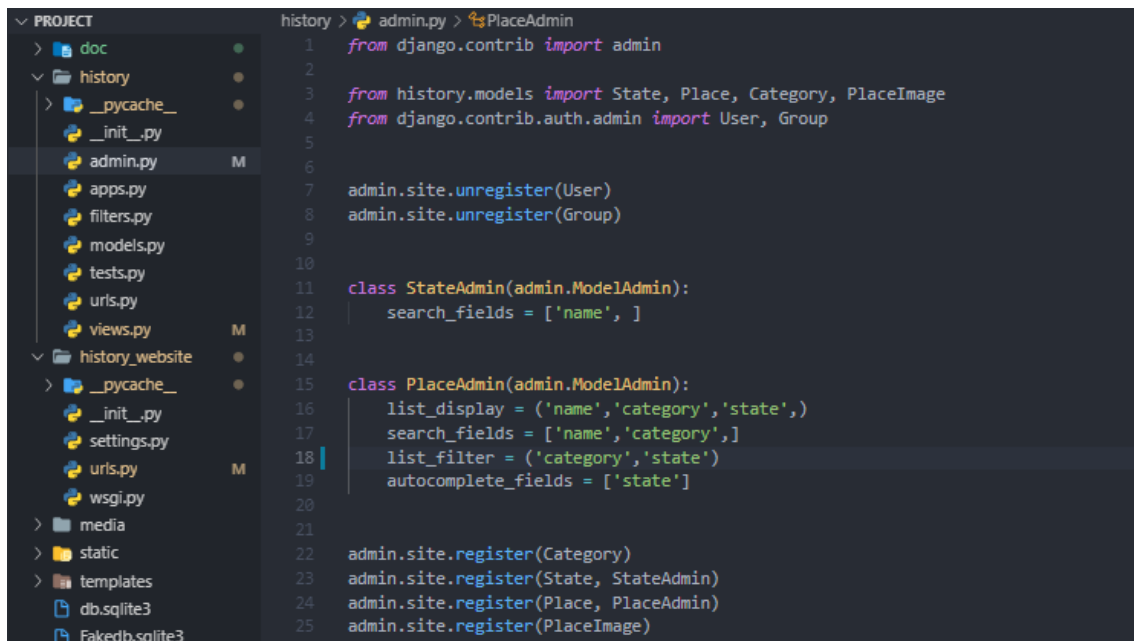
`url` دوم `/search` است که متد آن `place_search` است همان متدی که در سرچ یک `place` به کار میرود.

`url` بعدی `/state` است فرق این `path` با قبلی این است که این آدرس به تنهایی کار نمیکند و باید در ادامه ی آن یک رشته نوشت که هان `en_name` جدول `state` ها از و در اینجا استفاده میشود.

`url` اخر هم `palce` است دقیقا همانند `/state` کار میکند با این تفاوت که در مقابل آن باید `place id` را وارد کنید.

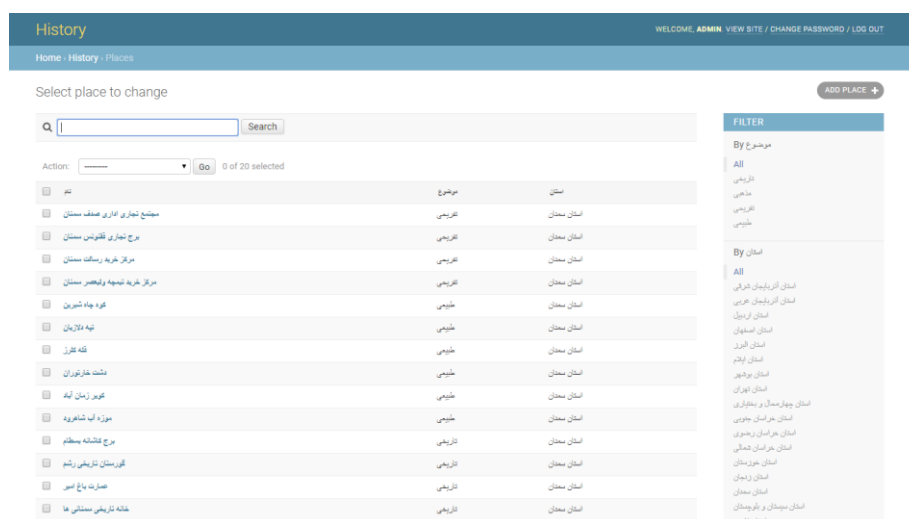
تنظیمات پنجره مدیریت

تنظیمات پنجره مدیریت یکی از بخش هایی است که فاصله کاربر را با محیط کد نویسی کمتر میکند و سطح دسترسی و مدیریت سایت را ارتقا میدهد، لازم به ذکر است که این بخش کد نویسی الگوریتمی ندارد و فقط pattern هایی که خود جنگو معرفی کرده را استفاده میکنیم.



```
1 from django.contrib import admin
2
3 from history.models import State, Place, Category, PlaceImage
4 from django.contrib.auth.admin import User, Group
5
6 admin.site.unregister(User)
7 admin.site.unregister(Group)
8
9
10
11 class StateAdmin(admin.ModelAdmin):
12     search_fields = ['name', ]
13
14
15 class PlaceAdmin(admin.ModelAdmin):
16     list_display = ('name', 'category', 'state',)
17     search_fields = ['name', 'category',]
18     list_filter = ('category', 'state')
19     autocomplete_fields = ['state']
20
21
22 admin.site.register(Category)
23 admin.site.register(State, StateAdmin)
24 admin.site.register(Place, PlaceAdmin)
25 admin.site.register(PlaceImage)
```

قصد ندارم خط به خط توضیح بدهم که هر کد دقیقا چه کاری انجام میدهد، و فکر کنم کد ها به اندازه ای تمیز و قابل فهم است که بتوانید درک کنید. این اطلاعات شامل این است که مثلا در جدول نمایش اطلاعات محل ها چه اطلاعاتی از آن قابل رویت باشد یا در باکس سرچ فیلد، چه مواردی سرچ شود.



نمایش اطلاعات در صفحه

اطلاعات همانطور که میدانید از فایل `models.py` به `views.py` میرسد، در اینجا کاربر `path` در فایل `urls.py` تعریف شده متد های که تعریف شده صدا میزنند، ایم متد های به فایل `html` ای که به آن ها تعریف شده میروند و در آنجا رندر میشوند. بعد این فایل رندر شده که همراه با کد های `html` است به صفحه مرجع خود میروند که در اینجا `base.html` که است. این از طریق `templates` ها امکان پذیر شده است. این چرخه گردش اطلاعات از دیتا بیس تا کد های `html` است که کاربر در صفحه مرورگر خود میبیند، در اینجا قصد داریم روش رندر شدن اطلاعات در مرحله `views to html` را توضیح دهیم.

```
templates > history > state_list.html > section
1  {% extends "history/base.html" %}
2  {% block content %}
3  <section>
4      <br>
5      <div class="container">
6          <div class="row">
7              <div name="categorys" id="categorys">
8              </div>
9          </div>
10         <div class="row">
11             {% for state in state_list %}
12                 <div class="col-3 state uk-margin-small-bottom">
13                     <a href="state/{{ state.en_name }}">
14                         <button class="btn btn-default" type="button">{{ state.name }}</button>
15                     </a>
16                 </div>
17                 {% empty %}
18                     <div class="col-2">
19                         هیچ استانی موجود نیست
20                     </div>
21                 {% endfor %}
22             </div>
23         </div>
24     </section>
25 {% endblock %}
```

همان طور که میبینید در صفحه `state_list` در وسط کد ها یک حلقه `for` قرار تا پس از گرفتن لیست تمام استان ها در متغیر `state_list` آنها را تک به تک با استفاده در `template` که قبلا طراحی شده بارگزاری میکند. بقیه صفحات هم از همین طریق یعنی با استفاده از حلقه ی `for` تمام اطلاعات خود را بارگزاری میکنند و دارا نکته خاص یا جدید نیستند که شایان ذکر باشد.

بقیه تغییرات انجام شده به قدری کوچک و ناچیز هستند که ارزش بازگو کردن در اینجا را به آنها نمیبینم. فقط یک تصور میماند که گفتم شاید مهم باشد، تنظیمات مرب.ط به زبان و ساعت پنل مدیریت و آدرس دایرکتوری `static`، تصور تمام این موارد را در پایین میگذارم

```
# language settings
LANGUAGE_CODE = 'en'
TIME_ZONE = 'Asia/Tehran'
USE_I18N = True
USE_L10N = True
USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.2/howto/static-files/
STATIC_URL = '/static/'
STATIC_ROOT = ''
STATICFILES_DIRS = (
    os.path.join(BASE_DIR, 'static/'),
)
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media/')
GOOGLE_MAPS_API_KEY = 'AIzaSyB5y1TcoQpby0BNZNC4LwqiaGhm0Bg'
```

حداقل سیستم مورد نیاز برای نصب و راه اندازی

- ✓ Processors: Intel Atom® processor or Intel® Core™ i3 processor
- ✓ Disk space: 1 GB
- ✓ Operating systems: Windows* 7 or later, macOS, and Linux

نرم افزار و کتاب خانه های ضروری

- ✓ Python* versions: 2.7.X, 3.6.X
- ✓ astroid* versions: 2.2.5
- ✓ autopep8* versions: 1.4.4
- ✓ Django* versions: 2.2.3
- ✓ django-filter* versions: 2.2.0
- ✓ django-widget-tweaks* versions: 1.4.5
- ✓ isort* versions: 4.3.21
- ✓ lazy-object-proxy* versions: 1.4.1
- ✓ mccabe* versions: 0.6.1
- ✓ Pillow* versions: 6.1.0
- ✓ pycodestyle* versions: 2.5.0
- ✓ pylint* versions: 2.3.1
- ✓ pytz* versions: 2019.1
- ✓ six* versions: 1.12.0
- ✓ sqlparse* versions: 0.3.0
- ✓ typed-ast* versions: 1.4.0
- ✓ wrapt* versions: 1.11.2

* بعدا به طور مفصل طریقه نصب این کتابخانه ها توضیح داده خواهید شد.

نرم افزار های غیر ضروری ولی مهم در اجرای هر چه بهتر پروژه

- ✓ Google Chrome* version 76.0.3809.100 (Official Build)
- ✓ FireFox* version 68.0.2
- ✓ PyCharm 2019.2 (Community Edition) Runtime version: 11.0.3+12-b304.10 amd64
- ✓ VScode* version: 1.37.0 (user setup)
- ✓ Atom* version 1.39.1

* تمامی برنامه های ذکر شده رایگان هستند و با یک سرچ ساده میتوان آن هارا نصب کنید

چگونگی نصب و راه اندازی

توجه: قبل از انجام مراحل نصب حتما به مواردی که در بخش [راهنمای کاربردی](#) ذکر شده بروید تا دچار مشکلاتی نشوید.

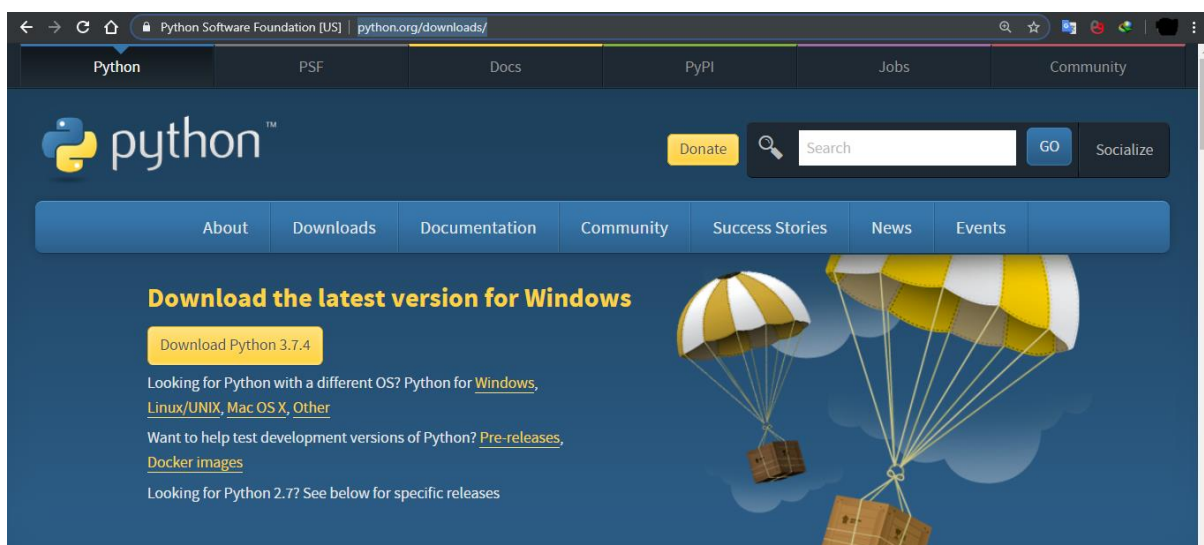
برای نصب و راه اندازی پروژه به سریع ترین شکل ممکن پیشنهاد میشود فایل [README.md](#) را مطالعه فرمایید.

ولی اگه در هنگام نصب به مشکلی برخوردید میتوانین از راهنمایی هایی که در داخل این بخش خواهد شد استفاده نمایید.

توجه: اگر از macOS یا هر توضیح linux استفاده میکنید نیاز به طی کردن مراحل ۱ و ۲ نیست، پایتون بصورت دیفالت در سیستم عامل شما قرار دارد.

مرحله ۱: یکی از مهم ترین برنامه هایی که باید نصب بشه کامپایلر python است که وظیفه ای آن خواندن کد ها و اجرای آن ها است

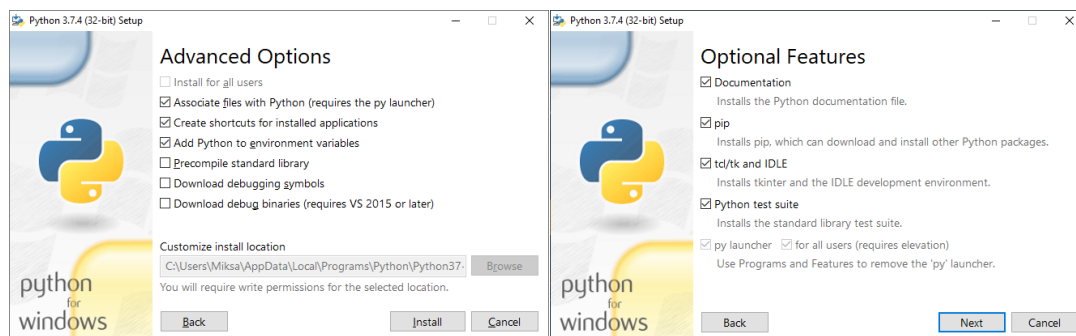
برای نصب python میتوانید به مراجعه به سایت www.python.org در نوار Downloads و با کلیک بر روی دکمه Download Python 3.7.X شروع به دانلود و نصب آن بکنید.



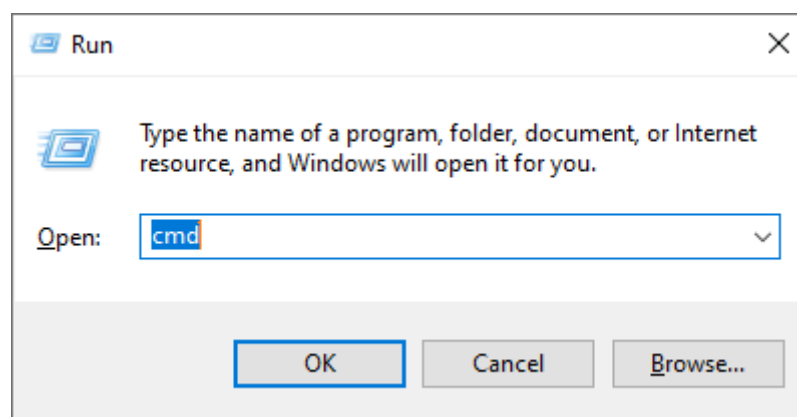
توجه: در زمان نگارش این پروژه از پایتون ورژن ۳.۷.۴ استفاده شده است. لذا اگر دچار مشکل شدید پیشنهاد میشود این نسخه را نصب یا **virtual environment** کنید

مرحله ۲: این مرحله بسیار آسان است و چند کلیک بر روی **next** مراحل نصب تمام میشود

فقط باید به این نکته توجه کرد که در مراحل نصب، تیک **install pip** و **add Python to environment variable** رو بزنید تا در مراحل بعدی به مشکل بر نخورید.



مرحله ۳: با فشار داد کلید های **Windows+R** پنجره **run** بازه میشود. کافیت بنویسید **cmd** تا **Command Prompt** باز شود.



مرحله ۴: در **command Prompt** بنویسید

```
pip install -r requirements.txt
```

```
C:\Windows\system32\cmd.exe
C:\Users\Miksa\Desktop\Project>pip install -r requirements.txt
```

توجه داشته باید که این کد و کدهای دیگر را باید در شاخه **root** پروژه اجرا نمایید!

منتظر شوید تا **pip** تمام [کتابخانه های ضروری](#) شما را نصب کند

توجه : این مرحله نیازمند اتصال به اینترنت است.

مرحله ۵ : حال وقت آن رسیده که تنظیمات پایگاه را انجام دهید

در همان **Command Prompt** دو دستور زیر را به ترتیب وارد کنید تا پایگاه داده و تمامی جداول آن ساخته شود

```
Python manage.py makemigrations
```

```
Python manage.py migrate
```

```
C:\Windows\system32\cmd.exe
[01/Sep/2019 04:55:49] "GET /static/admin/fonts/Roboto-Bold-webFont.woff HTTP/1.1" 200 86184
[01/Sep/2019 04:55:56] "GET /admin/history/place/ HTTP/1.1" 200 12376
[01/Sep/2019 04:55:56] "GET /admin/jsi18n/ HTTP/1.1" 200 3223
[01/Sep/2019 04:55:56] "GET /static/admin/js/jquery.init.js HTTP/1.1" 200 363
[01/Sep/2019 04:55:56] "GET /static/admin/css/changelists.css HTTP/1.1" 200 6170
[01/Sep/2019 04:55:56] "GET /static/admin/js/actions.js HTTP/1.1" 200 6766
[01/Sep/2019 04:55:56] "GET /static/admin/js/admin/RelatedObjectLookups.js HTTP/1.1" 200 6918
[01/Sep/2019 04:55:56] "GET /static/admin/js/prepopulate.js HTTP/1.1" 200 1530
[01/Sep/2019 04:55:56] "GET /static/admin/js/core.js HTTP/1.1" 200 7099
[01/Sep/2019 04:55:56] "GET /static/admin/js/urlify.js HTTP/1.1" 200 8941
[01/Sep/2019 04:55:56] "GET /static/admin/img/search.svg HTTP/1.1" 200 458
[01/Sep/2019 04:55:56] "GET /static/admin/js/vendor/xregexp/xregexp.js HTTP/1.1" 200 128820
[01/Sep/2019 04:55:56] "GET /static/admin/js/vendor/jquery/jquery.js HTTP/1.1" 200 271817
[01/Sep/2019 04:55:56] "GET /static/admin/img/tooltag-add.svg HTTP/1.1" 200 331
[01/Sep/2019 04:56:32] "GET / HTTP/1.1" 200 14829
[01/Sep/2019 04:56:32] "GET /static/index/css/style.css HTTP/1.1" 304 0
[01/Sep/2019 04:57:46] "GET / HTTP/1.1" 200 14813
[01/Sep/2019 04:57:46] "GET /static/index/img/favicon.ico HTTP/1.1" 200 8186
[01/Sep/2019 05:01:43] "GET /admin/ HTTP/1.1" 200 7221
[01/Sep/2019 05:01:46] "GET /admin/history/state/ HTTP/1.1" 200 11883
[01/Sep/2019 05:01:46] "GET /admin/jsi18n/ HTTP/1.1" 200 3223
[01/Sep/2019 05:18:28] "GET /admin/ HTTP/1.1" 200 7221

C:\Users\Miksa\Desktop\Project>py manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  No migrations to apply.

C:\Users\Miksa\Desktop\Project>
```


مرحله ۶: وقت آن است که یک ادمین سایت معرفی کنید تا بتوانید سایت را مدیریت کنید

با استفاده از دستور زیر در Command Prompt کار را شروع کنید

```
python manage.py createsuperuser
```

در اینجا سیستم از نام کاربری، ایمیل، و رمز معتبر درخواست میکند

توجه: در صورت وارد نکردن یک رمز قوی ممکنه است به شما خطاری بدهد، که میتوان آن را به زدن دکمه Y رد کنید.

```
C:\Windows\system32\cmd.exe - python manage.py createsuperuser
C:\Users\Wksa\Desktop\Project>python manage.py createsuperuser
Username (leave blank to use 'mksa'): admin
Email address: mrtaremi78@gmail.com
Password:
Password (again):
```

مرحله ۷: حال همه چیز آماده است. پروژه را Run کنید. با استفاده از دستور

```
python manage.py runserver
```

```
C:\Windows\system32\cmd.exe - python manage.py runserver
C:\Users\Wksa\Desktop\Project>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
August 28, 2019 - 06:13:41
Django version 2.2.3, using settings 'history_website.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

توجه: نباید این پنجره Command Prompt را تا موقعی که میخوايد پروژه run باشد ببندید. در صورت اتمام کار کافست در این پنجره کلید ترکیبی Ctrl+C را وارد کنید تا سرور قطع شود.

برای ورد به صفحه اصلی پروژه کافست در مرورگر خود آدرس <http://127.0.0.1:8000> را وارد

کنید تا به صفحه اصلی منتقل شوید

مرحله ۸: در انتها برای راحتی شما کافیسیت کانفیگ های اولیه را در مورد استان ها است را با استفاده از ورود به سایت <http://localhost:8000/add> انجام دهید.

توجه: این مرحله اجباری نیست و میتوانید خودتان آن را دستی انجام دهید.

راهنمای کاربردی

در این بخش راهنمایی هایی که در تجربه هر چه بهتر از پروژه میتوانید داشته باشید را بازگو میکنم.

(۱) حتما از در موقع نصب و اجرا از وصل بودن به اینترنت مطمئن باشید. زیرا در موقع نصب چندین پکیج (که لیست آنها را گفته بودم) شروع به دانلود شدن میکنند.

و در زمان اجرا از لینک هایی (CDN) استفاده شده که از طریق اینترنت خوانده میشود.

(۲) ممکن است برای دانلود بعضی اپدیت ها و پکیج ها نیاز به فیلتر شکن باشد.

حد امکان فیلتر شکن خود را روشن کنید.

(۳) تمام مراحل نصب قبلا طی شده است و تمام دیتا های تا حدی که در توانم بود وارد دایتایس شده اند

بنابراین نیاز به طی کردن مراحل که در [راهنمای نصب](#) گفته شده نیست.

فقط برای اجرا کافیسیت [مرحله ۷](#) را اجرا کنید و نیازی به طی کردن مراحل دیگر نیست.

(۴) اگر در هر صورت خواستید دیتابیس را پاک کرده و یک بار دیگر تمام دیتا هارا وارد کنید. کافیست فایل `db.sqlite3` را که در شاخه اصلی پروژه است را پاک کنید. و برای راه اندازی مجدد مراحلی که در [راهنمای نصب](#) گفته شده راه طی کنید.

(۵) برای سهولت در تست پروژه سعی شده یک دیتابیس به نام `Fakedb.sqlite3` نوشته شده که شامل چند صد دیتای فیک است. (مسیر این دیتابیس در شاخه اصلی پروژه قرار دارد)

برای تست این دیتابیس مراحل زیر را دنبال کنید

- (۱) اگر پروژه `Run` است آنرا متوقف کنید
- (۲) دیتابیس اصلی را پاک یا در صورت نیاز مجدد تغییر مسیر دهید
- (۳) دیتابیس `Fakedb.sqlite3` را به `db.sqlite3` تغییر نام دهید.
- (۴) پروژه را دوباره `Run` کنید.

توجه : این دیتابیس با دیتا های فیک ساخته شده در سایت <https://mockaroo.com/> است. بنابراین ممکن است در بخش لینک ها که برای صفحه ی گوگل لینک نا مناسبی باشد. بنده هیچ گونه مسئولیتی در قبال لینک ها ندارم.

در قسمت آخر این مستند باید چند نکته رو شایان ذکر بشم.

نکته اول این است که پروژه به صورت responsive طراحی شده یعنی شما میتونید در تمام پلتفرم ها از جمله موبایل، تبلت، تلویزیون های هوشمند و... استفاده کنید.

پیشنهاد میشه قبل از شروع به استفاده از پروژه حتما قسمت راهنمای کاربردی را بخوانید تا دچار مشکل نشوید.

نکته آخر اینکه شاید این پروژه به اندازه کافی بزرگ یا تخصصی نباشد، اما پروژه جوری کد زده شده که پتاسیل افزایش امکانات و توانایی هارو داراست، تمام این پتاسیل از کتابخانه قدرمت جنگو داراست که به هر چه راحت تر شدن کد نویسی کمک میکند.

موفق و پیروز باشید، امیدوارم تجربه ی خوبی از پروژه داشته باشید.

محمدرسول طارمی

شهریور ۹۸

<https://github.com/miksart/project>

https://www.amazon.com/gp/product/1491962291/ref=as_li_tl?ie=UTF8&tag=codewithmosh-20&camp=1789&creative=9325&linkCode=as2&creativeASIN=1491962291&linkId=c9cb035eec225b25d53274111951d95c

https://github.com/CoreyMSchafer/code_snippets

https://www.amazon.com/gp/product/1593276036/ref=as_li_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=1593276036&linkCode=as2&tag=codewithmosh-20&linkId=f61bef4447a86881181a90444ca06ed0

https://www.amazon.com/gp/product/1593275994/ref=as_li_tl?ie=UTF8&tag=codewithmosh-20&camp=1789&creative=9325&linkCode=as2&creativeASIN=1593275994&linkId=e55ad39c6469889a09dd16f4a6e4cbef

https://www.amazon.com/gp/product/B077Z55G3B/ref=as_li_tl?ie=UTF8&tag=codewithmosh-20&camp=1789&creative=9325&linkCode=as2&creativeASIN=B077Z55G3B&linkId=b26f8a05bec004e37060386fbc60f9e3

https://www.amazon.com/gp/product/1549617214/ref=as_li_tl?ie=UTF8&tag=codewithmosh-20&camp=1789&creative=9325&linkCode=as2&creativeASIN=1549617214&linkId=58b1ecf739a460e84d8dceb8df156ec0

https://www.amazon.com/Learning-Python-5th-Mark-Lutz/dp/1449355730/ref=sr_1_1?keywords=python&qid=1566940298&s=gateway&sr=8-1

https://www.amazon.com/Django-Beginners-Build-websites-Python/dp/1983172669/ref=sr_1_1?crid=34X0ZCT4ZX4QF&keywords=python+django&qid=1566940344&s=gateway&sprefix=python+djan%2Caps%2C322&sr=8-1

https://www.amazon.com/Django-Web-Development-Cookbook-practical/dp/1788837681/ref=sr_1_2?crid=34X0ZCT4ZX4QF&keywords=python+django&qid=1566940344&s=gateway&sprefix=python+djan%2Caps%2C322&sr=8-2

https://www.amazon.com/Test-Driven-Development-Python-Selenium-JavaScript/dp/1491958707/ref=sr_1_4?crid=34X0ZCT4ZX4QF&keywords=python+django&qid=1566940344&s=gateway&sprefix=python+djan%2Caps%2C322&sr=8-4

<https://www.python.org/>

<https://www.djangoproject.com/>

<https://www.youtube.com/>

https://www.youtube.com/channel/UCCezIgC97PvUuR4_gbFUs5g

<https://www.youtube.com/channel/UCWv7vMbMWH4-V0ZXdmDpPBA>

<https://www.w3schools.com>

<https://github.com/>

<https://.quora.com>

<https://medium.com/>

<https://adamj.eu/>

<https://wsvincent.com/>

<https://stackoverflow.com/>

<https://ultimatedjango.com/>

<https://programmingwithmosh.com/>

<https://codewithmosh.lpages.co/python-cheat-sheet/>

[https://en.wikipedia.org/wiki/Django \(web framework\)](https://en.wikipedia.org/wiki/Django_(web_framework))

<https://faradars.org/courses/fvpht9611-django-web-based-framework-using-python>

<https://hamrahyad.com/courses/%D8%A2%D9%85%D9%88%D8%B2%D8%B4-%D8%AC%D9%86%DA%AF%D9%88/>

<http://arzandehnia.com/>

<https://amoozesh.org/django/>

<https://zarinserver.com/blog/what-is-django/>

<https://git.ir/django/>

<https://learnfiles.com/course/%D8%A2%D9%85%D9%88%D8%B2%D8%B4-%D8%AC%D9%86%DA%AF%D9%88/>

<https://software.intel.com>

<https://hackernoon.com/top-10-python-web-frameworks-to-learn-in-2018-b2ebab969d1a>

<https://www.shabakeh-mag.com>

<http://python.coderz.ir/>

با تشکر از راهنمایی که دوست خوبم علیرضا ربیعی در ساخت این پروژه به من داشت.