

وزارت علوم تحقیقات و فناوری

دانشگاه صنعتی سیرجان

دانشكده مهندسی برق و کامپیوتر

پایان‌نامه دوره‌ی كارشناسی مهندسی کامپیوتر

[نام پایان‌نامه]

[نام دانشجو]

استاد راهنما:

[نام استاد راهنما]

استاد مشاور:

[نام استاد یا اساتید مشاور]

ماه سال

[شهریور 95]



وزارت علوم تحقیقات و فناوری

دانشگاه صنعتی سیرجان

دانشكده مهندسی برق و کامپیوتر

اين پايان نامه به عنوان يکي از شرايط احراز درجه کارشناسي به

**بخش برق و کامپیوتر**

**دانشگاه صنعتی سیرجان**

تسليم شده است و هيچگونه مدرکي به عنوان فراغت از تحصيل دوره‌ي مزبور شناخته نمي‌شود.

|  |  |  |
| --- | --- | --- |
| عنوان | نام و نام‌خانوادگی | امضاء |
| دانشجو |  |  |
| استاد راهنما |  |  |
| داور اول |  |  |
| داور دوم |  |  |

|  |
| --- |
| **حق چاپ محفوظ و مخصوص به مولف است.** |

###### تقدیم

###### تشكر و قدردانی

###### چكیده

امروزه نرم‌افزارهای وب دارای اهمیت ویژه‌ای هستند، یکی از فریم‌ورک‌های مدرن و قوی که البته کار توسعه دهنده را تای جای ممکن ساده کرده است، جنگو می‌باشد که فریم‌ورک زبان برنامه‌نویسی پایتون برای ایجاد نرم‌افزارهای وب است. همچنین از HTML، CSS و JavaScript برای طراح سایت استفاده شده است. در نتیجه یک سایت با ویژگی نمایش کتاب‌ها، نویسنده‌ها، سبد خرید و ... تولید می‌شود.

**کلید واژه:** جنگو، نرم‌افزار وب، پایتون

###### فهرست مطالب

عنوان صفحه

فهرست شكل‌‌ها

عنوان صفحه

[شکل ‏4‑1: ارتباط بخش‌های مختلف کد در جنگو 9](#_Toc106897525)

[شکل ‏4‑2: صفحه اولیه یک پروژه جنگو 13](#_Toc106897526)

[شکل ‏5‑1: لیست کتاب‌ها در سایت ادمین 19](file:///M:\Workspace\DjangoProjects\bookShop\document\bookShopDocument.docx#_Toc106897527)

# مقدمه

## پیشگفتار

اینترنت و فضای وب روز به روز در حال رشد است و تعداد کاربرهای آن روندی افزایشی دارد. امروزه کارهای بسیاری را می‌توانیم از طریق نرم‌افزارهای متصل به اینترنت انجام دهیم و وجود آن‌ها زندگی ما را بسیار ساده‌تر و بهینه‌تر کرده است. با شیوع ویروس کرونا به صورت جهانی در اوایل سال ۲۰۱۹ میلادی، نیاز به خدمات تحت وب بیش از پیش شد و نقش اینترنت را در زندگی ما بسیار جدی‌تر از قبل کرد.

## بیان نیاز موضوع

امروزه بسیاری از مشاغل و خدمات، نسخه‌‌های مجازی خود را دارند که از نسخه‌های غیرمجازی، دارای کاربران بیشتری بوده و آینده درخشان‌تری برای آن‌ها احساس می‌شود. به عنوان نمونه می‌توان به فروشگاه‌ها اشاره کرد که نسخه‌های مجازی آن‌ها می‌توانند تعداد بسیار بیشتری از محصولات را در لحظه جلوی چشمان ما بیاورند، یا سیستم بانکداری که دیگر محدود به چندین ساعت از روزهای غیر تعطیل هفته نمی‌شود. همچنین عدم نیاز به حضور در محل از مزایای دیگر این نوع خدمات است که کمک شایانی به کم شدن آلودگی‌های مربوط به حمل و نقل نیز می‌شود.

بنابراین داشتن یک شعبه مجازی که حداقل به عنوان یک مکمل عمل کند، برای بسیاری از شغل‌ها و خدمات، جز ضروریات به حساب می‌آید.

## تعریف مسئله و اهمیت آن

کتاب‌ها هر چند به اندازه سایر محتوای تصویری و صوتی دارای محبوبیت نیستند، اما محتوای آموزشی و سرگرم کننده بسیاری را به شیوه خودشان ارائه می‌کنند. نرم‌افزار‌های فروش یا مطالعه کتاب به صورت آنلاین جایگاهی خاصی در میان کاربرها دارند، زیرا تهیه کردن کتاب‌ها را بسیار ساده‌تر می‌کنند و انتخاب‌های بسیار بیشتری را در اختیار ما قرار می‌دهند.

## نتیجه‌گیری

پس برای این پروژه تصمیم گرفته شد که یک سایت فروش کتاب طراحی شود. در فصل‌های بعد، موارد تخصصی‌تر مورد بررسی قرار می‌گیرند، از جمله انتخاب تکنولوژي مناسب، شیوه پیاده‌سازی پروژه و موارد دیگر.

# انتخاب بستر مناسب

## مقدمه

در این فصل، بسترهای مختلف برای انجام دادن این پروژه مورد بررسی قرار خواهد گرفت. مزایا و معایب هر کدام نام برده می‌شود و در نهایت بستر مناسب را انتخاب شده و دلیل آن شرح داده می‌شود.

## بسترهای مختلف

نرم‌افزارها از نظر ذات، به دسته‌های مختلفی تبدیل می‌شوند. حوزه فعالیت این پروژه در دسته نرم‌افزارهای دسکتاپ[[1]](#footnote-2)، نرم‌افزارهای موبایل[[2]](#footnote-3) و نرم‌افزارهای تحت‌وب[[3]](#footnote-4) قرار می‌گیرد. زیرا می‌خواهیم این پروژه بر روی دستگاه‌های مختلف در دسترس باشد. نمونه‌های مشابه نیز دارای چنین وضعیتی هستند، برخی صرفا دارای یک نسخه وب هستند، و برخی برای سکوهای[[4]](#footnote-5) پر کاربر، نسخه‌های متفاوتی ارائه می‌دهند.

نرم‌افزارهای تحت‌وب تنها به یک مرورگر[[5]](#footnote-6) برای اجرا شدن نیاز دارند و تفاوتی نمی‌کند که این مرورگر روی چه سکویی باشد. برخلاف آن‌ها، نرم‌افزارهای دسکتاپ و موبایل که جز نرم‌افزاری‌های بومی[[6]](#footnote-7) به شمار می‌روند، برای یک سکوی خاص توسعه داده شده و تولید می‌شوند.

### مقایسه نرم‌افزارهای بومی و تحت‌وب

برخلاف نرم‌افزارهای بومی، نرم‌افزارهای تحت‌وب برای اجرا شدن نیاز به اینترنت دارند. اما با توجه به ماهیت پروژه این مورد نیازی به بررسی ندارد، زیرا کاربر در هر صورت باید به اینترنت دسترسی داشته باشد تا بتواند نرم‌افزار را مشاهده کرده و به کتاب‌ها دسترسی پیدا کند.

دیگر تفاوت مهم، عدم نیاز نرم‌افزارهای تحت‌وب به نصب کردن روی سیستم شخصی کاربر و آپدیت کردن آن است. کافی‌ست کاربر وارد سایت شده و همواره از جدیدترین نسخه نرم‌افزار استفاده می‌کند.

نرم‌افزارهای بومی به دلیل توانایی آن‌ها در بهره‌وری بهتر از منابع سیستم، دارای کارایی[[7]](#footnote-8) بیشتری هستند، اما از آن‌جایی که این پروژه نیاز به پردازش‌های سنگین ندارد، نرم‌افزارهای تحت‌وب هم‌چنان می‌توانند نیازهای این پروژه را برطرف کنند.

## نتیجه‌گیری

با توجه به تشریح بسترهای مختلف و در نظر گرفتن نیازهای این پروژه، بهترین گزینه توسعه دادن یک نرم‌افزار تحت‌وب می‌باشد. از طرفی نرم‌افزارهای تحت‌وب گزینه بسیار مناسبی برای شروع هستند، زیرا با توسعه دادن تنها یک نرم‌افزار می‌توان به کاربرهای سکوهای مختلف دسترسی داشت.

حال که بستر مناسب انتخاب شد، در فصل بعد ابزارهای مختلف برای توسعه یک نرم‌افزار تحت‌وب را بررسی کرده و گزینه مناسب برای این پروژه انتخاب می‌شود.

# فریم‌ورک‌های[[8]](#footnote-9) توسعه نرم‌افزار تحت‌وب

## مقدمه

در این فصل، ابتدا به صورت دقیق‌تر با نرم‌افزار تحت‌وب آشنا شده، سپس به بررسی فریم‌ورک‌‌‌های کاندید پرداخته می‌شود و در نهایت گزینه مناسب انتخاب خواهد شد.

## فریم‌ورک‌های وب

یک روش استاندارد برای ساخت و استقرار[[9]](#footnote-10) نرم‌افزارهای تحت‌وب بر روی شبکه جهانی وب[[10]](#footnote-11) ارائه می‌کنند. برخی از آن‌ها قابلیت‌های مختلفی را در اخیتار توسعه‌دهنده و کاربر قرار می‌دهند که به چند مورد از رایج‌ترین‌ها در ادامه پرداخته شده است:

* برقراری ارتباط با پایگاه‌های داده[[11]](#footnote-12) مختلف
* فراهم کردن گزینه‌های امنیتی از جمله سیستم احراز هویت و مجوز[[12]](#footnote-13)
* ارائه قابلیت کش، که در آن داده‌های صفحات وب درون سیستم کاربر ذخیره می‌شوند تا دفعه‌های بعد با سرعت بهتری بارگیری شوند
* نگاشت[[13]](#footnote-14) URLها، که در آن آدرس‌هایی برای صفحات سایت در نظر گرفته می‌شوند و فریم‌ورک مسئولیت بررسی کردن آدرس ورودی کاربر را برعهده می‌گیرد تا بتواند صفحه مورد نظر را فراخوانی کند

## بررسی فریم‌ورک‌های مختلف

وقتی پای انتخاب فریم‌ورک مناسب برای یک پروژه می‌رسد، گزینه‌های زیادی وجود دارند و انتخاب کردن چندان ساده نمی‌باشد، اما می‌توان با توجه به نیازهای هر پروژه، لیست را تا حدی محدود کرد. با توجه به اینکه پروژه مورد بحث در دسته نرم‌افزارهای عمومی قرار می‌گیرد و تمرکز بر روی یک یا چند ویژگی خاص برای مثال سرعت یا سادگی در استقرار نسیت، پس فریم‌ورکی که بتواند نیازهای عمومی را برطرف کرده و در بخش‌های مختلف سطح خوبی داشته باشد، می‌تواند بهترین گزینه برای این در نظر گرفته شود. از طرفی با توجه به تک‌نفره بودن پروژه، انتخاب یک فریم‌ورک که حداکثر کار ممکن را انجام دهد و کار را برای توسعه‌دهنده به ساده‌ترین حالت ممکن در بیاورد، دارای اهمیت زیادی است.

برای انتخاب فریم‌ورک، ابتدا باید زبان‌برنامه‌نویسی را انتخاب کرد و سپس از بین فریم‌ورک‌های آن زبان، یک گزینه را تعیین کرد. زبان‌های PHP، JavaScript و البته Python از موارد توصیه شده در این زمینه هستند.

بر کسی پوشیده نیست که پایتون دارای نحو[[14]](#footnote-15) بسیار ساده‌ای بوده و دارای آموزش‌های بسیاری است. این‌ها موارد آشنایی برای فریم‌ورک اصلی آن، یعنی جنگو[[15]](#footnote-16) نیز می‌باشند. در برابر آن، پی‌اچ‌پی و جاوااسکریپت هر چند دارای نحو خوبی هستند اما پایتون همچنان برتری را در این زمینه دارا است. از طرفی امروزه پایتون به عنوان یک زبان همه کاره در بخش‌های مختلف مورد استفاده قرار می‌گیرد و بسیاری از قبل با آن آشنایی دارند.

در ارتباط به فریم‌ورک‌های اصلی یا بیشتر توصیه شده پی‌اچ‌پی می‌توان به Laravel و برای جاوااسکریپت به NodeJS و VueJS اشاره کرد. وقتی به بررسی دقیق این فریم‌ورک‌ها با هم پرداخته می‌شود، می‌توان شاهد بود که رقابت بسیار نزدیکی با یکدیگر دارند. اما به صورت خلاصه می‌توان دریافت که جنگو به ساده‌ترین شکل ممکن بیشترین ویژگی‌های مورد نیاز برای این پروژه را عرضه می‌کند. دارای امنیت بیشتری نسبت به رقیب‌های خودش است، از لحاظ بهره‌وری بهتر از لاراول و کمی ضعیف‌تر از نودجی‌اس می‌باشد، دارای جامعه[[16]](#footnote-17) بسیار خوبی است که در صورت برخورد با خطاها می‌تواند بسیار کارآمد باشد، همچنین قابلیت مقیاس پذیری[[17]](#footnote-18) بالایی دارد و می‌تواند حجم عظیمی از ترافیک را به خوبی کنترل کند و بهینه‌سازی‌های خودکاری انجام می‌دهد که کار توسعه‌دهنده را بسیار راحت می‌کند.

## نتیجه‌گیری

همان‌طور که در بخش قبل بیان شد، جنگو که یکی از فریم‌ورک‌های پایتون به شمار می‌رود، بعنوان یک فریم‌ورک همه‌کاره و با قابلیت‌های مناسب برای این پروژه، بهترین گزینه به شمار می‌رود. این فریم‌ورک به سرعت درحال پیشرفت است و همکاری خوبی با سایر تکنولوژی‌ها و ابزارها دارد، همچنین دارای یک وب‌ سرور سبک و سریع است که اجرا کردن پروژه‌ها را بسیار ساده می‌کند.

در این فصل بررسی کردیم که چرا جنگو گزینه مناسبی برای این پروژه‌ است، در فصل بعد با شیوه کار و بخش‌های مختلف آن آشنا می‌شویم.

# کار با جنگو

## مقدمه

در این فصل، ابتدا به تعریف دقیق‌تری از جنگو پرداخته شده و سپس یک نمای کلی از آنچه که این فریم‌ورک وب را خاص می‌کند، بررسی مختصری از ویژگی‌های اصلی، برخی ویژگی‌های پیشرفته و اجزای اصلی یک نرم‌افزار جنگو به شما ارائه داده می‌شود.

## جنگو چیست؟

جنگو یک فریم‌ورک وب سطح بالای پایتون است که توسعه سریع نرم‌افزارهایی کامل، همه کاره[[18]](#footnote-19)، امن، مقیاس پذیر، قابل نگهداری[[19]](#footnote-20) و قابل حمل[[20]](#footnote-21) را امکان‌پذیر می‌کند. جنگو که توسط توسعه دهندگان باتجربه ساخته شده است، بسیاری از دردسرهای توسعه وب را برطرف می‌کند، بنابراین می‌توانید بدون نیاز به اختراع مجدد چرخ، روی توسعه نرم‌افزار خود تمرکز کنید. جنگو رایگان و منبع باز[[21]](#footnote-22) است، دارای یک جامعه پر رونق و فعال، اسناد[[22]](#footnote-23) عالی، و گزینه‌های بسیاری برای پشتیبانی رایگان و پولی است.

جنگو از سال ۲۰۰۵ میلادی عرضه شد، نقطه عطف[[23]](#footnote-24) آن در سال ۲۰۰۸ میلادی بود و از آن زمان تا کنون که نسخه ۴ آن در سال ۲۰۲۲ میلادی عرضه شد، پیشرفت‌های زیادی داشته است.

## میزان محبوبیت جنگو چقدر است؟

مقیاس دقیقی وجود ندارد که بتوان براساس آن محبوبیت یک فریم‌ورک وب را تعیین کرد، اما می‌توان با نگاه کردن به ستاره‌های دریافت شده توسط هر پروژه در گیت‌هاب[[24]](#footnote-25)، نظرسنجی سالیانه StackOverflow و موارد دیگر، محبوبیت این فریم‌ورک را حدس زد. اما سوال مهم‌تر این است که آیا جنگو به قدر کافی محبوب هست که بخواهیم از آن استفاده کنیم و با مشکلات محبوب نبودن یک تکنولوژی از جمله نبودن منابع و جامعه قوی روبه‌رو نشویم؟

بر اساس تعداد سایت‌های پرمخاطب که از جنگو استفاده می‌کنند، تعداد افرادی که در پایگاه کد[[25]](#footnote-26) مشارکت می‌کنند، و موارد دیگر، بله، جنگو یک فریم‌ورک محبوب است! از جمله این سایت‌ها می‌توان به Instagram، Mozilla، Pinterest و National Geographic اشاره کرد.

## روال کار و بخش‌های کد

در جنگو، یک نرم‌افزار وب منتظر درخواست‌های[[26]](#footnote-27) HTTP از مرورگر وب (یا کلاینت[[27]](#footnote-28) دیگر) است. هنگامی که درخواستی دریافت می‌شود، نرم‌افزار بر اساس URL و احتمالاً اطلاعات موجود در داده‌های POST یا داده‌های GET مربوط به درخواست دریافت شده، موارد مورد نیاز را بررسی می‌کند. بسته به آنچه مورد نیاز است، ممکن است اطلاعات را از پایگاه داده بخواند یا در آن بنویسد یا سایر وظایف مورد نیاز برای برآورده کردن نیاز درخواست دریافت شده را انجام دهد. سپس برنامه پاسخی را به مرورگر وب برمی‌گرداند، و اغلب به صورت پویا یک صفحه HTML برای مرورگر ایجاد می‌کند تا با قرار دادن داده‌های به دست آورده در نگهدارنده‌های مکان[[28]](#footnote-29) درون قالب HTML نمایش داده شود.

جنگو برای کدهای مربوط هر یک از این بخش‌ها یک فایل مجزا در نظر مي‌گیرد. در ادامه هر یک از این موارد شرح داده شده است.



شکل ‏4‑1: ارتباط بخش‌های مختلف کد در جنگو

* **URL:** این فایل دارای مجموعه‌ای از الگوها است، هر یک از این الگوها نشان‌دهنده یک آدرس خاص مي‌باشد. هر یک از این الگوها، در صورت وارد شدن توسط کاربر درون نوار آدرس مرورگر، یک View را فراخوانی می‌کنند. البته این تنها حالت ممکن نیست، برای مثال، گاهی اوقات برخی از این الگوها، یک مجموعه جدیدی از الگوها را بارگذاری می‌کنند.
* **View:** توابع یا کلاس‌هایی هستند که درخواست HTTP را گرفته و یک پاسخ HTTP را برمی‌گرداند. از طریق View می‌توان به پایگاه‌داده و برخی بخش‌های دیگر دسترسی داشت، می‌توان درون آن‌ها به زبان پایتون محاسبات انجام داد، همچنین می‌توان متغیرهای مورد نیاز برای بارگذاری شدن به قالب‌ مخصوص آن صفحه ارسال کرد.
* **Model:** مدل‌ها اشیاء پایتون هستند که ساختار داده‌های یک نرم‌افزار را تعریف می‌کنند و مکانیسم‌هایی را برای مدیریت (افزودن، اصلاح، حذف) و پرس‌و‌جو[[29]](#footnote-30) رکوردها در پایگاه داده ارائه می‌کنند. مدل‌ها به صورت یک جدول درون پایگاه داده ذخیره می‌شوند.
* **Template:** قالب یک فایل متنی است که ساختار یا طرح‌بندی یک فایل (مانند صفحه HTML) را با متغیرهایی که برای نمایش محتوای واقعی استفاده می‌شود، تعریف می‌کند. یک view می‌تواند به صورت پویا یک صفحه HTML را با استفاده از یک قالب HTML ایجاد کند و آن را با داده‌های محاسبه شده توسط View یا داده‌های یک مدل پر کند. یک قالب می‌تواند برای تعریف ساختار هر نوع فایلی استفاده شود. لازم نیست حتما HTML باشد!

## نصب جنگو و راه‌اندازی پروژه

با توجه به اینکه حالا می‌دانید جنگو چیست و برای چه چیزی کاربرد دارد، مراحل نصب و راه‌اندازی اولیه یک پروژه به صورت کلی مورد بررسی قرار می‌گیرد. جنگو قابل نصب بر روی ویندوز، مک و لینوکس می‌باشد، اما در این قسمت، فقط مراحل مربوط سیستم‌عامل ویندوز پوشش داده می‌شود.

### نصب پایتون

همان‌طور که در بخش‌های قبل بیان شد، جنگو یکی از فریم‌ورک‌های زبان برنامه‌نویسی پایتون برای توسعه نرم‌افزارهای تحت وب می‌باشد، بنابراین برای کار کردن با آن نیاز به نصب داشتن پایتون و مدیر بسته آن یعنی pip است. برای این پروژه به نسخه سه پایتون و pip نیاز است. برای نصب پایتون می‌توان از سایت رسمی آن[[30]](#footnote-31)، آخرین نسخه از سری نسخه‌های سه را متناسب با سیستم‌عامل خود دانلود کرد.

### راه‌اندازی محیط مجازی[[31]](#footnote-32)

محیط مجازی به فضایی ایزوله گفته می‌شود که می‌‌توان در آن برنامه‌ها و پکیج‌ها را نصب کرد. این برنامه‌ها به صورت سراسری در سیستم‌عامل قابل دسترس نخواهند بود، که اجازه می‌دهد تا چندین نسخه از یک برنامه را به صورت همزمان نصب داشته باشید. برای دسترسی به این برنامه‌ها، کافیست وارد محیط مجازی مربوط به آن شوید.

برای نصب جنگو از محیط مجازی استفاده می‌شود تا در صورتی که چندین پروژه با نسخه‌های مختلف جنگو درون سیستم باشد، یا ممکن باشد بعداً پروژه‌های دیگری با نسخه‌های دیگر ایجاد شوند، مشکلی به وجود نیاید.

برای این کار از پکیج virtualenvwrapper-win روی ویندوز استفاده می‌شود که با وارد کردن دستور زیر در cmd قابل نصب است.

pip3 install virtualenvwrapper-win

سپس برای ایجاد یک محیط مجازی از دستور زیر استفاده می‌شود. دقت کنید که تمام بخش‌های درون <…> باید توسط خود شما تغییر کند. متن درون آن توصیف کننده مقداری است که باید وارد شود.

mkvirtualenv <my\_django\_environment>

چهار دستور مهم برای کار کردن با محیط مجازی وجود دارد:

* workon <name\_of\_environment>: فعال کردن محیط مجازی
* deactivate: جهت خروج از محیط مجازی
* workon: نمایش تمام محیط‌های مجازی قابل استفاده
* rmvirtualenv <name\_of\_environment>: حذف کردن محیط مجازی نام برده

### نصب جنگو

حال برای نصب جنگو باید محیط مجازی را فعال کرد و سپس نسخه مد نظر برای جنگو را نصب کرد که در این پروژه از آخرین نسخه عرضه شده یعنی نسخه چهار استفاده شده.

pip3 install django~=4.0

و درنهایت می‌توان با دستور زیر از نصب شدن جنگو مطمئن شد:

py -m django --version

### راه‌اندازی و اجرای پروژه

برای راه‌اندازی یک سایت با جنگو باید این مراحل را طی کرد؛ ایجاد یک پروژه و ایجاد یک نرم‌افزار[[32]](#footnote-33). پروژه، به نوعی محیط توسعه دادن به حساب می‌رود، که برای مثال حاوی مجموعه‌ای از پیکربندی‌ها[[33]](#footnote-34) و نرم‌افزارها برای یک سایت است. نرم‌افزار که معمولا از آن تحت عنوان اَپ یا app یاد می‌شود، یک نرم‌افزار وب است که کار خاصی انجام می‌دهد، برای مثال سیستم یک وبلاگ، یک دیتابیس یا نرم‌افزاری که در این مقاله مورد بحث قرار خواهد گرفت. یک پروژه می‌تواند بیش از یک نرم‌افزار داشته باشد، و یک نرم‌افزار می‌تواند درون چند پروژه مورد استفاده قرار بگیرد.

برای ایجاد پروژه می‌توان ابتدا با استفاده از cmd وارد مسیری شد که قرار است پروژه آن‌جا ایجاد شود، سپس از دستور زیر برای ایجاد آن استفاده کرد:

django-admin startproject <project\_name>

با این کار یک پوشه[[34]](#footnote-35) با نام انتخاب شده برای پروژه ایجاد می‌شود. درون آن، یک سری دایرکتوری[[35]](#footnote-36) و فایل وجود دارد که ساختار قرار گرفتن آن‌ها به صورت زیر است. فاصله از ابتدای خط، نشان‌دهند جایگاه هر فایل یا دایرکتوری درون دایرکتوری‌ها دیگر است:

project\_name/

manage.py

db.sqlite3

project\_name/

\_\_init\_\_.py

settings.py

urls.py

asgi.py

wsgi.py

چند تا از بخش‌های مهم آن در ادامه تعریف می‌شوند:

* دایرکتوری project\_name بیرونی: دایرکتوری ریشه پروژه است. نام آن برای جنگو اهمیتی نداشته و می‌توان آن را در هر زمان دلخواه تغییر داد.
* دایرکتوری project\_name درونی: پکیج اصلی پایتون برای پروژه است. از نام آن برای وارد کردن[[36]](#footnote-37) فایل‌ها در جاهای مختلف استفده می‌شود بنابراین تغییر دادن آن کار توصیه شده‌ای نیست.
* manage.py: یک ابزار خط فرمانی[[37]](#footnote-38) است که به شما اجازه می‌دهد به روش‌های مختلفی با پروژه ارتباط برقرار کنید.
* db.sqlite3: پایگاه داده پیش‌فرض جنگو می‌باشد. پایگاه داده‌ای سبک و مناسب برای پروژه‌های کوچک.
* mysite/settings.py: تمام پیکربندی‌های مربوط به پروژه در این فایل ذخیره می‌شود. برای مثال،‌ منطقه زمانی[[38]](#footnote-39)، محل ذخیره فایل‌های استاتیک[[39]](#footnote-40) و موارد دیگر.
* mysite/urls.py: مانند فهرست محتوای می‌ماند، جایی که آدرس صفحه‌های مختلف سایت در آن نوشته شده ‌است.

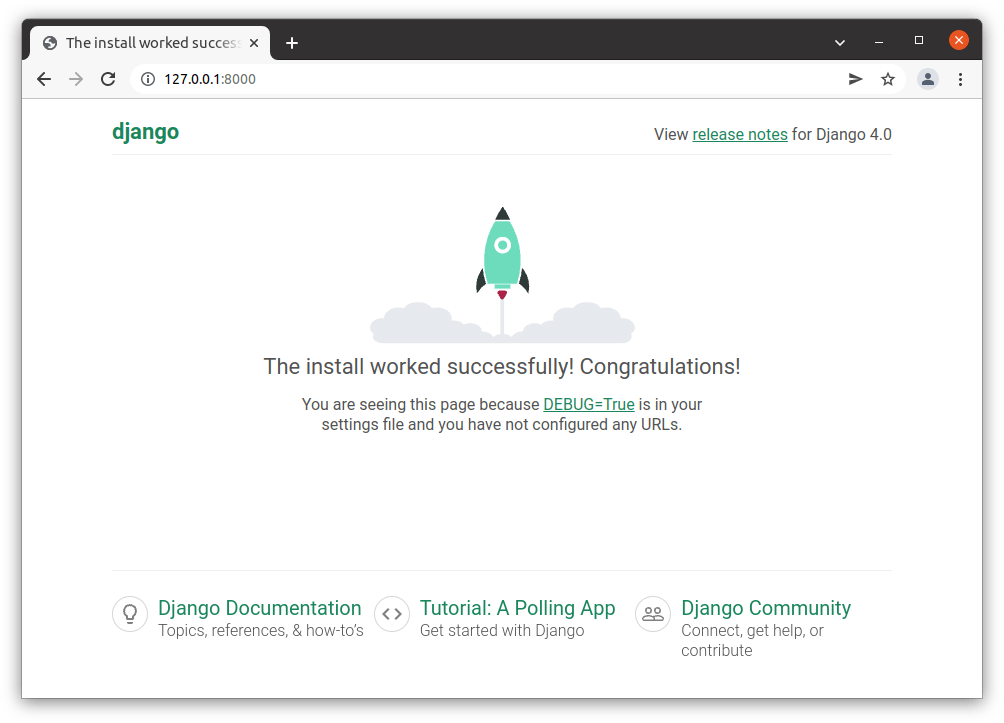
حال که با بخش‌های مختلف پروژه آشنا شدید، می توان سرور پروژه را با دستور زیر اجرا کرد. مطمئن شوید که درون دایرکتوری project\_name بیرونی قرار داشته باشید، جایی که فایل manage.py قرار دارد.

py manage.py runserver

و سپس با وارد کردن آدرس زیر که آدرس میزبان محلی با پورت ۸۰۰۰ است در مرورگر می‌توان صفحه طراحی شده توسط جنگو را دید که نشان‌دهنده درست بودن پروژه تا اینجای کار است.

http://127.0.0.1:8000/

صفحه مشاهده شده باید مشابه تصویر زیر باشد:

شکل ‏4‑2: صفحه اولیه یک پروژه جنگو

## نتیجه‌گیری

در بالا بخش‌هایی از جنگو که در بیشتر مواقع برای پیاده‌سازی یک ویژگی جدید مورد استفاده قرار می‌گیرند بیان شد، اما این همه بخش‌ها و فایل‌ها نیست، برای مثال فرم‌ها[[40]](#footnote-41) که برای دریافت کردن داده از کاربر مورد استفاده قرار می‌گیرد و سایت ادمین که برای مشاهده مدل‌ها و رکوردها استفاده می‌شود، از بخش‌های مهم به شمار می‌روند.

همچنین دیدید که یک پروژه و نرم‌افزار در جنگو به چه شکل ایجاد شده و چگونه داده‌ها در پایگاه داده ذخیره می‌شوند.

در فصل‌ بعد به ایجاد کردن پروژه و نرم‌افزار پرداخته می‌شود و حین پیاده‌سازی ویژگی‌های مختلف نرم‌افزار، از بخش‌هایی که در این فصل دید استفاده خواهد شد و می‌توانید نگاه نزدیک‌تری به آن‌ها بیاندازید.

# پیاده‌سازی پروژه

## مقدمه

این فصل با ایجاد کردن پروژه و نرم‌افزار شروع می‌شود، بعد با بخش‌های مختلف سایت تعریف شده و نحوه پیاده‌سازی آن‌ها توضیح داده می‌شود. بدون شک یک سایت از بخش‌های زیادی تشکیل شده و ریزه کاری‌های خود را دارد، بنابراین در این فصل با مهم‌ترین بخش‌ها آشنا خواهید شد.

## ایجاد کردن محیط توسعه

با توجه به مطالب بیان شده در بخش راه‌اندازی و اجرای پروژه در فصل چهار، یاد گرفتید که چگونه کارهای اولیه را انجام دهید، نامی که برای این پروژه در نظر گرفته شده است، bookShop مي‌باشد. این پروژه دارای یک نرم‌افزار است که نام آن را برای سادگی کار، همان app می‌گذاریم. اپ را با دستور زیر می‌توان ایجاد کرد:

py manage.py startapp app

اکنون ساختار دایرکتوری‌های پروژه به صورت زیر است:

bookShop/

manage.py

bookShop /

app/

migrations/

\_\_init\_\_.py

admin.py

apps.py

models.py

tests.py

views.py

از جمله بخش‌های جدید می‌توان به موارد زیر اشاره کرد: دایرکتوری migrations اشاره کرد، که در آن فایل مربوط به تغییراتی که در پایگاه‌داده اعمال کرده‌ایم قرار می‌گیرد. فایل

* دایرکتوری migrations: محل ذخیره فایل‌های مربوط به تغییرات اعمال شده در پایگاه ‌داده
* admin.py: تنظمیات و پیکربندی مربوط به سایت ادمین
* models.py: جایی که مدل‌ها در آن تعریف می‌شوند
* views.py: جایی که ویو‌ها نوشته می‌شوند

## ثبت کردن اپ در پروژه

اکنون که برنامه ایجاد شده است، باید آن را در پروژه ثبت کنیم تا در هنگام اجرای هر ابزاری (مثلاً اضافه کردن مدل‌ها به پایگاه داده) در دسترس باشد. اپ‌ها با افزودن آن‌ها به لیست INSTALLED\_APPS در تنظیمات پروژه ثبت می‌شوند.

برای این کار فایل bookshop/settings.py را باز کرده، لسیت INSTALLED\_APP را یافته و نام اپ را درون آن قرار دهید. وضعیت این لیست حالا باید شبیه به حالت زیر باشد:

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

. . .

app,

]

## تنظیم URLها

پس از ایجاد اپ، باید آدرس فایل urls.py آن را درون فایل bookShop/urls.py قرار داد تا بتوان به آدرس‌های درون آن نیز دسترسی داشت. برای این کار ابتدا باید ماژول include را اضافه کرد و سپس با استفاده از آن، آدرس‌ها را وارد نمود:

from django.urls import include

urlpatterns = [

. . .

path('app/', include('app.urls')),

]

## ایجاد کردن مدل‌ها

همان‌طور که گفته شد، مدل‌ها از نوع کلاس هستند و مانند جدول‌های پایگاه داده عمل می‌کنند. هر صفت[[41]](#footnote-42) از این کلاس حکم ک ستون از جدول را دارد و هر شیء ایجاد شده از این کلاس، یک رکورد یا ردیف در پایگاه داده به شما می‌رود.

در این پروژه شش مدل نیاز است:

* **Book:** مدل مربوط به هر ک از کتاب‌های موجود در سایت است. دارای صفت‌هایی از جمله عنوان کتاب، قیمت و تصویر جلد می‌باشد، صفت نویسنده را دارد که یک کلید خارجی از جدول Author به حساب می‌آید و همچنین صفت ژانر یا دسته را دارد که یک ارتباط چند به چند به جدول Genre است،‌ زیرا یک کتاب می‌تواند چند ژانر داشته و یک ژانر نیز می‌تواند مطعلق به چندین کتاب باشد.
* **Author:** مدل نویسنده دارای صفت‌های نام، تاریخ تولد و فوت، زندگینامه‌ای مختصر و تصویری از شخص می‌باشد. هر شیء از این مدل می‌تواند به عنوان صفت author درون مدل Book به کار رود.
* **Genre:** مدل ژانر فقط دارای صفت نام می‌باشد که بیان‌گر نام آن ژانر است. هر شیء از این مدل می‌تواند به عنوان صفت genre درون مدل Book به کار رود.
* **ShoppingCart:** این مدلدارای صفت کاربر که کلید خارجی به جدول User می‌باشد که به صورت خودکار توسط جنگو ایجاد شده است و همچنن صفت کتاب‌ها که مجموعه از کل کتاب‌های موجود در هر سبد خرید می‌باشد، این صفت از نوع ارتباط چند به چند است. هر کاربر یک شیء در این مدل دارد که کتاب‌هایش در صفت کتاب‌ها ذخیره می‌شوند.
* **Comment:** این مدل دارای صفت‌های امتیاز، توصیه کردن ، متن پیام و تاریخ می‌باشد. همچنین دو صفت به صورت کلید خارجی یکی برای کاربر و دیگری برای کتاب دارد. بنابراین هر شیء از این مدل یک نظر[[42]](#footnote-43) ثبت شده توسط کاربر مشخص برای کتاب مشخصی است. این مدل همچنین دارای دو صفت از نوع لیست می‌باشد که درون یکی، امتیازهای قابل قبول برای صفت امتیاز و درون دیگری حالت‌های مختلف توصیه کردن برای صفت مربوطه است.
* **Contact:** این مدل مربوط به فرم تماس با ما می‌باشد که در آن اطلاعات تماس گیرنده و پیام آن ذخیره می‌شود.

دو متد[[43]](#footnote-44) رایج برای مدل‌ها در زیر تعریف شده:

* **\_\_str\_\_:** وقتی شما یک شیء از هر مدلی را بخواهید چاپ[[44]](#footnote-45) کنید، پایتون به صورت خودکار نوع و یک سری اطلاعات دیگر از آن را برمی‌گرداند، اما کاربردی است تا در چنین شرایطی ک متن دلخواه برگشت داده شود تا اطلاعات بهتری نسبت به همان شیء به توسعه دهنده دهد. این کار در زمان عیب‌یابی[[45]](#footnote-46) کاربرد زیادی دارد. این متد همین کار را می‌کند و می‌تواند یک متن، مقدار یکی از صفت‌های شیء یا موارد این چنینی را برگرداند.
* **get\_absolute\_url:** فرض کنید لیستی از اشیاء یک کلاس را خروجی گرفته‌اید و حالا می‌خواهید با کلیک روی هر کدام، وارد همان شیء به خصوص شوید، این متد در این جا کاربرد دارد، به این صورت که id آن شیء را به URL مربوطه می‌دهد تا صفحه مخصوص همان شیء نشان داده شود.

پس از تعریف مدل‌ها، باید آن‌ها را درون پایگاه داده ذخیره کرد. برای این کار، ابتدا دستور زیر را اجرا کرده تا تغییرات را پیدا کرده و همچنین به دنبال خطاهای مربوط به پایگاه داده بگردد:

py manage.py makemigrations

سپس با دستور زیر، تغییرات را در پایگاه داده ثبت می‌کنیم:

py manage.py migrate

## سایت ادمین

نرم‌افزار ادمین جنگو می‌تواند از مدل‌های شما برای ایجاد خودکار یک سایت استفاده کند که می‌توانید از آن برای ایجاد، مشاهده، به‌روزرسانی و حذف رکوردها استفاده کنید. این سایت می‌تواند زمان زیادی را در حین توسعه برای شما ذخیره کند و آزمایش مدل‌ها را بسیار آسان می‌کند.

سایت ادمین می‌تواند تا حد خوبی شخصی‌سازی شده و ویژگی‌هایی را به آن اضافه نمود. اما همان‌طور که از اسمش پیداست، باید فقط برای ادمین‌ها و افرادی داخلی سایت در دسترس باشد. این سایت به شما اجازه گروه‌ بندی و ایجاد دسترسی‌های[[46]](#footnote-47) خاص برای کاربران مختلف را می‌دهد. البته با توجه به عدم نیاز این پروژه به این ویژگی‌ها، این قسمت‌ها مورد بحث قرار نمی‌گیرند.

برای استفاده از این سایت کافیست مدل‌های مد نظر را در فایل app/amin.py ثبت کنیم. ساده‌ترین حالت ممکن به صورت زیر است که مدل Genre را در سایت ادمین ثبت می‌کند. البته باید مدل مد نظر را ابتدا وارد کرد.

admin.site.register(Genre)

اما می توان با تعریف یک کلاس مجزا در همین فایل برای هر یک از مدل‌ها، ویژگی‌های خاصی برای هر مدل درون سایت ادمین اضافه کرد. برای مثال کد زیر برای مدل کتاب دو صفت را برای هر رکورد انتخاب می‌کند تا در لیست کتاب‌ها نشان داده شود و همچنین امکان جستجو کردن براساس عنوان هر کتاب را اضافه می‌کند.

class BookAdmin(admin.ModelAdmin):

list\_display = ('title', 'author')

search\_fields = ('title',)

admin.site.register(Book, BookAdmin)

برای وارد شدن به سایت ادمین نیاز به اکانت superuser دارید که با استفاده از دستور زیر و پر کردن اطلاعات خواسته شده، می‌توان یک اکانت ایجاد کرد. البته یک اکانت با نام کاربری admin و رمز عبور admin از قبل برای این پروژه ساخته شده است.

python3 manage.py createsuperuser

حال اگر به آدرس سایت ادمین[[47]](#footnote-48) بروید و پس از وارد شده به سایت، گزینه Books را از منوی APP انتخاب کنید، با صفحه‌ای مشابه تصور زیر مواجه می‌شوید:

## پیاده‌سازی ویژگی‌ها

شکل ‏5‑1: لیست کتاب‌ها در سایت ادمین

در این بخش ابتدا ویژگی‌های مختلف سایت تعریف شده و سپس نحوه پیاده‌سازی آن‌ها ارائه می‌شود. برای هر یک از این ویژگی‌ها نیاز به یک URL و یک View داریم. برخی از آن‌ها نیاز به Form نیز دارند.

### لیست کتاب‌ها

در این صفحه لیستی از تمام کتاب‌های موجود در سایت نمایش داده می‌شود.

برای نوشتن آدرس هر یک از این ویژگی‌ها باید یک تابع path برای آن‌ها بنویسید. این تابع معمولا دارای سه ورودی است، که به ترتیب، خود آدرس، ویوی مورد نظر برای اجرا و نام آن آدرس است. از این نام در قالب‌های سایت و ویوها استفاده می‌شود تا صفحه مربوط به آن آدرس خاص را بارگیری کند. وقتی مقدار ورودی اول یک رشته خالی در نظر گرفته می‌شود، یعنی همان آدرس اصلی فایل. برای مثال در اینجا می‌شود http://127.0.0.1:8000/app.

path('', views.BookListView.as\_view(), name='books')

از آن‌جایی که نشان دادن یک لیست از تمام اشیاء موجود در یک مدل کار بسیار رایجی است، جنگو این ویژگی را بصورت یک کلاس generic.ListView در آورده که هر ویویی که از نوع کلاس باشد و نه تابع، می‌تواند با به ارث بردن آن، لیستی از اشیاء یک مدل را برگرداند. تنها کافیست صفت model را برابر با نام مدل مد نظر قرار داد. با اضافه شدن این کلاس به عنوان والد یک ویو، کارهای دیگری هم به صورت خودکار انجام می‌شود که وقتی یک ویو از نوع تابع را بررسی کردیم به صورت واضح به چشم می‌آیند. بنابراین ویو مربوط به لیست کتاب‌ها به صورت زیر است:

class BookListView(generic.ListView):

model = Book

### جزئیات کتاب‌ها

آدرس این صفحه به صورت زیر تعرف شده. نکته قابل توجه وجود <int:pk> در قسمت آدرس است، زمانی که می‌خواهید یک قسمت متغیر در آدرس وجود داشته باشد، به این صورت نوع آن متغیر و نامی که برایش در نظر گرفته شده را باید نوشت. پس int نوع متغیر و pk نام آن است. برای این مسیر، قسمت متغیر با id کتابی که قرار است جزئیاتش نشان داده شود، پر می‌شود. این کار به کمک متد get\_absolute\_url انجام می‌شود.

path('book/<int:pk>/', views.bookDetail, name='book-detail')

ویوی این صفحه، ۴ کار کلی را باید انجام دهد. اول از همه چون حالت generic برای آن وجود ندارد، تعریف آن به صورت تابعی خواهد بود که به توسعه‌دهنده کنترل بیشتری می‌دهد:

بخش اول این ویو، گرفتن شیء مربوط به کتاب مورد نظر از پایگاه داده و همچنین لیستی از کتاب‌های مشابه که در همان ژانر هستند را به دست آورده و آن‌ها را به قالب ارسال می‌کند.

بخش دوم، به دست آوردن یک سری آمار از نظراتی که برای همان کتاب نوشته شده است، از جمله امتیاز کتاب می‌باشد.

بخش سوم، فرم ارسال کامنت می‌باشد، برای این کار باید یک فرم در فایل app/forms.py ایجاد شود تا فیلدهای[[48]](#footnote-49) مورد نیاز در آن تعریف شود. این فیلدها ممکن است نیاز به اعتبارسنجی[[49]](#footnote-50) داشته باشند که باز هم در همین فایل انجام می‌شوند.

بخش چهارم، نشان دادن کلیه نظرات است. برای این کار کافیست یک پرس‌وجو نوشت که لیستی از تمام نظرات مربوط به همان کتاب را برگرداند و سپس آن مجموعه اشیاء را به قالب بدهد.

### لیست نویسنده‌ها

در این صفحه لیستی از تمام نویسنده‌هایی که در سایت کتابی برای آن‌ها ثبت شده، نمایش داده می‌شود. آدرس آن نکته جدیدی ندارد. ویو هم به صورت generic.ListView نوشته می‌شود.

### جزئیات نویسنده‌ها

در ویو، باید دو کار انجام شود. اول باید نویسنده مورد نظر را از پایگاه داده گرفت و کار دوم، نوشتن یک پرس‌و‌جو برای دریافت کردن لیستی از کتاب‌های همان نویسنده است.

### لیست ژانرها

در این صفحه لیستی از تمام ژانرها نمایش داده می‌شود. ویو هم به صورت generic.ListView نوشته می‌شود.

### جزئیات ژانرها

خود مدل Genre، چیز خاصی برای نشان دادن ندارد، بنابراین برای صفحه جزئیات آن، لیستی از تمام کتاب‌های موجود در آن ژانر نمایش داده می‌شود.

### نماش سبد خرید

ویو مربوط به این قسمت، showShoppingCart است. در این ویو ابتدا باید بررس شود که کاربر در سات وارد شده است یا نه، اگر وارد نشده باشد به صفحه کتاب‌ها منتقل می‌شود. اما اگر وارد شده باشد یک پرس‌وجو روی مدل ShoppingCart نوشته که براساس شیء مربوط به کاربری که الان وارد سایت شده است، سبد خرید او را دریافت شود. همین‌طور تعداد کتاب‌های موجود در سبد خرید را گرفته و هزینه کل کتاب‌های موجود در لیست محاسبه می‌شود.

### حذف از سبد خرید

ویو مربوط به این قسمت، removeFromShoppingCart است. در این ویو ابتدا کتابی که قرار است حذف شود را از پایگاه داده گرفته و سپس از متد remove روی سبد خرید کاربر استفاده می‌شود تا کتاب مورد نظر حذف شود.

### اضافه کردن به سبد خرید

ویو مربوط به این قسمت، addToShoppingCart است. ابتدا کتاب مورد نظر را از پایگاه داده گرفته و سپس بررسی می‌شود که آیا کاربر سبد خرید دارد یا نه. اگر نداشت، ابتدا برای او یک شیء از کلاس shoppingCart ایجاد می‌شود و بعد کتاب درخواست شده به سبد خریدش اضافه می‌شود.

### خرید موفقیت آمیز

ویو successfulPurchase زمانی اجرا می‌شود که پرداخت با موفقیت انجام شده باشد. تنها کاری که در این ویو باید انجام شود، پاک کردن سبد خرید کاربر و نمایش یک متن به کاربر است.

### تماس با ما

صفحه تماس با ما یک سری راه‌های ارتباطی را به کاربر نمایش می‌دهد و همچنین به او اجازه می‌دهد که تا در صورت نیاز از طریق یک فرم، پیام خود را ارسال کند.

### وارد شدن[[50]](#footnote-51) و ثبت‌نام[[51]](#footnote-52)

جنگو به صورت پیش‌فرض قابلیت وارد شدن را فراهم کرده و فقط کافیست یک قالب برای آن طراحی شود. اما برای ثبت‌نام کردن نیاز به نوشتن ویو نیز می‌باشد. ویوی آن از نوع کلاس است و Signup نام دارد. حالت‌های generic شامل مواردی بیشتری از ListView می‌شود، از جمله CreateView که در این ویو استفاده شده. برای این ویو کافیست سه صفت تعریف شود:

* form\_class: مشخص می‌کند که باید از چه فرمی استفاده شود. که اینجا از UserCreationForm استفاده می‌شود.
* success\_url: باید تایین کرد که در صورت موفقیت آمیز بودن ثبت‌نام، کاربر به چه آدرس منتقل بشود.
* template\_name: یک قالب برای این ویو مشخص شود تا جنگو بتواند آن را برای ثبت‌نام بارگذاری کن.

### جست‌وجو

برای قسمت جست‌وجوی کتاب، از یک generic.ListView استفاده می‌شود. ویوی مربوط به آن SearchResultsView است. صفت model را برابر با Book گذاشته و برای قسمت پرس‌وجو، از متد get\_queryset استفاده می‌شود تا پرس‌وجوی پیش‌فرض کلاس‌های generic.ListView را بازنویسی کرد. در این متد، عبارت جست‌وجو شده توسط کاربر را گرفته و در بین عنوان‌ کتاب‌ها جست‌وجو می‌شود.

## نتیجه‌گیری

در این فصل با قسمت‌های مختلف سایت و شیوه پیاده‌سازی آن‌ها به صورت مختصر آشنا شدید. در فصل بعد به سیستم قالب‌های سایت و تکنولوژی‌های استفاده شده در آن پرداخته می‌شود.

# سیستم قالب‌ها

## مقدمه

در این فصل، سیستم قالب‌های جنگو بررسی می‌شود و همین‌طور به صورت مختصر با برخی از تکنولوژي‌ها استفاده شده در طراحی سایت آشنا می‌شوید.

## ساختار نوشتن قالب‌ها

در هر سایت‌، بخش‌هایی وجود دارد که در صفحات مختلف تکرار می‌شوند. یک فایل html پایه در نظر گرفته می‌شود که در آن هر بخش به صورت یک بلاک[[52]](#footnote-53) در می‌آید. برای مثال بلاک sidenav که منوی سایت در آن قرار می‌گیرد، بلاک title که مخصوص عنوان هر فصله است و بلاک content که محتوای اصلی سایت در آن قسمت مي‌گیرد.

سپس با افزودن عبارت زیر در هر یک از قالب‌ها می‌توان قالب جدید را براساس قالب پایه توسعه داد.

{% extends "base.html" %}

البته می‌توان هر بلاک را در قالب جدید بازنویسی کرد تا مخصوص خود آن صفحه شود. برای مثال می‌توان محتوای بلاک title را در فایل base.html به صورت زیر نوشت:

{% block title %}<title>Book Shop</title>{% endblock %}

سپس برای صفحه لست کتاب‌ها آن را به صورت زیر بازنویسی کرد:

{% block title %}<title>List of all Books</title>{% endblock %}

## تکنولوژی‌های استفاده شده

برای قسمت فرانت‌اند[[53]](#footnote-54) یا همان قسمت ظاهری سایت، از تکنولوژي‌های زیر استفاده شده است:

* HTML: برای ساختن بخش‌های مختلف سایت استفاده می‌شود.
* CSS: برای طراحی کردن بخش‌های ساخته شده استفاده می‌شود. یکی از ابزارهای مهمی که در این قسمت به کار رفته، CSS Grid می‌باشد که با آن می‌توان چیدمان[[54]](#footnote-55) سایت را طراحی کرد.
* JavaScript: برای انجام کارهایی از جمله باز و بسته کردن منو که نمی‌توان با HTML یا CSS آن را انجام داد.

## نتیجه‌گیری

در این فصل نشان داده شد که چطور ویژگی‌های توضیح داده شده در فصل پنج را به شکل صفحات وب در آورده به کاربر نشان داد.

# نتیجه‌گیری

در این مقاله، با نحوه ساخت یک سایت یا نرم‌افزار وب با استفاده از جنگو آشنا شده و بخش‌های مختلف کار را دیدید.

# واژه نامه انگلیسی به فارسی

|  |  |
| --- | --- |
| Application (app) | اپ |
| authentication and authorization system | سیستم احراز هویت و مجوز |
| block | بلاک |
| client | کلاینت |
| codebase | پایگاه کد |
| command-line | خط فرمان |
| comment | نظر |
| community | جامعه |
| configuration | پیکربندی |
| database | پایگاه داده |
| debugging | عیب‌یابی |
| deploy | استقرار |
| desktop application | نرم‌افزارهای دسکتاپ |
| directory | دایرکتوری |
| django | جنگو |
| documentation | اسناد |
| fields | فیلدها |
| folder | پوشه |
| form | فرم |
| framework | فریم‌ورک |
| front-end | فرانت‌اند |
| github | گیت‌هاب |
| import | وارد کردن |
| layout | چیدمان |
| login | ورود |
| maintainable | قابل نگهداری |
| mapping | نگاشت کردن |
| method | متد |
| milestone | نقطه عطف |
| mobile application | نرم‌افزارهای موبایل |
| native application | نرم‌افزارهای بومی |
| open source | متن باز |
| performance | کارایی |
| permissions | دسترسی‌ها |
| placeholder | نگهدارنده مکان |
| platform | سکو |
| portable | قابل حمل |
| print | چاپ |
| property | صفت |
| query | پرس‌وجو |
| register/signup | ثبت‌نام |
| request | درخواست |
| scalability | مقیاس‌پذیری |
| static | استاتیک |
| syntax | نحو |
| time zone | منطقه زمانی |
| validations | اعتبار سنجی |
| versatile | همه کاره |
| virtual environment | محیط مجازی |
| web application | نرم‌افزارهای وب |
| web browser | مرورگر وب |
| World Wide Web (WWW) | شبکه جهانی وب |

1. desktop application [↑](#footnote-ref-2)
2. mobile application [↑](#footnote-ref-3)
3. web application [↑](#footnote-ref-4)
4. platform [↑](#footnote-ref-5)
5. web browser [↑](#footnote-ref-6)
6. native application [↑](#footnote-ref-7)
7. performance [↑](#footnote-ref-8)
8. framework [↑](#footnote-ref-9)
9. deploy [↑](#footnote-ref-10)
10. World Wide Web (WWW) [↑](#footnote-ref-11)
11. database [↑](#footnote-ref-12)
12. authentication and authorization system [↑](#footnote-ref-13)
13. mapping [↑](#footnote-ref-14)
14. syntax [↑](#footnote-ref-15)
15. django [↑](#footnote-ref-16)
16. community [↑](#footnote-ref-17)
17. scalability [↑](#footnote-ref-18)
18. versatile [↑](#footnote-ref-19)
19. maintainable [↑](#footnote-ref-20)
20. portable [↑](#footnote-ref-21)
21. open source [↑](#footnote-ref-22)
22. documentation [↑](#footnote-ref-23)
23. milestone [↑](#footnote-ref-24)
24. github [↑](#footnote-ref-25)
25. codebase [↑](#footnote-ref-26)
26. request [↑](#footnote-ref-27)
27. client [↑](#footnote-ref-28)
28. placeholder [↑](#footnote-ref-29)
29. query [↑](#footnote-ref-30)
30. <https://www.python.org> [↑](#footnote-ref-31)
31. virtual environment [↑](#footnote-ref-32)
32. Application (app) [↑](#footnote-ref-33)
33. configuration [↑](#footnote-ref-34)
34. folder [↑](#footnote-ref-35)
35. directory [↑](#footnote-ref-36)
36. import [↑](#footnote-ref-37)
37. command-line [↑](#footnote-ref-38)
38. time zone [↑](#footnote-ref-39)
39. static [↑](#footnote-ref-40)
40. form [↑](#footnote-ref-41)
41. property [↑](#footnote-ref-42)
42. comment [↑](#footnote-ref-43)
43. method [↑](#footnote-ref-44)
44. print [↑](#footnote-ref-45)
45. debugging [↑](#footnote-ref-46)
46. permissions [↑](#footnote-ref-47)
47. <http://127.0.0.1:8000/admin> [↑](#footnote-ref-48)
48. fields [↑](#footnote-ref-49)
49. validations [↑](#footnote-ref-50)
50. login [↑](#footnote-ref-51)
51. register/signup [↑](#footnote-ref-52)
52. block [↑](#footnote-ref-53)
53. front-end [↑](#footnote-ref-54)
54. layout [↑](#footnote-ref-55)