بسمه تعالی

دانشگاه صنعتی شریف

دانشکده‌ی مهندسی کامپیوتر

سند معماری

سامانه‌ی تیکت مستر

پروژه درس ایجاد چابک نرم‌افزار

اعضای تیم:

ساعی سعادت، بردیا رضایی کلانتری، محمد صادقی، مهدی عباس‌تبار

استاد درس:

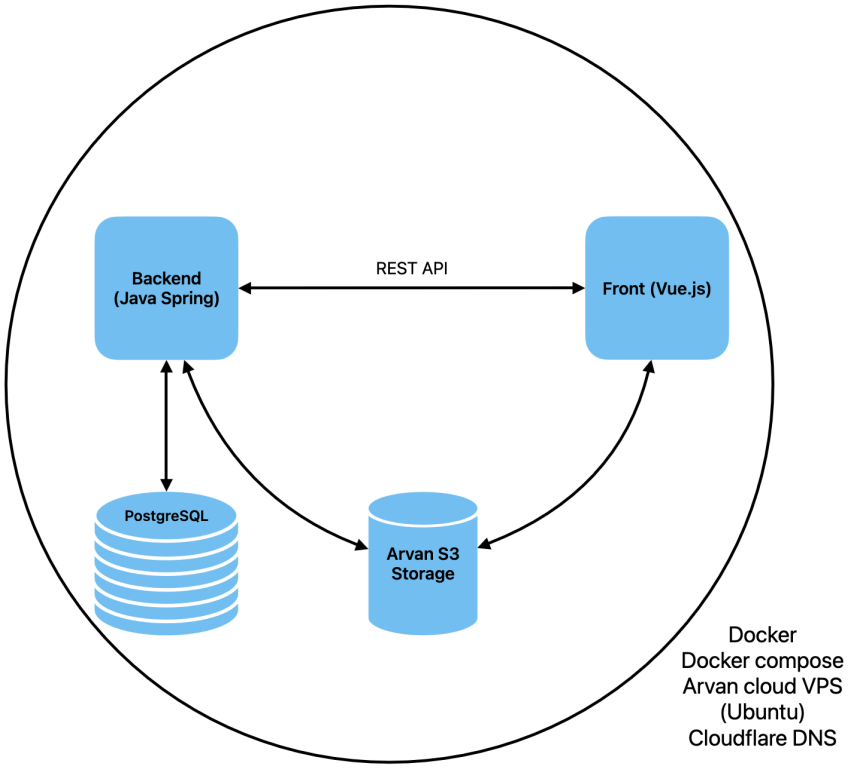
دکتر رامان رامسین

پاییز ۱۴۰۲

فهرست مطالب

ابزار مورد استفاده

- برای پیاده‌سازی سمت back-end، از زبان Java و فریم‌ورک Spring Boot استفاده می‌شود. فریم‌ورک Spring Boot این امکان را می‌دهد که یک REST API را بدون نیاز به پیکربندی اضافه‌ای پیاده‌سازی کرد. همچنین به دلیل سادگی سامانه معماری این قسمت به صورت یکپارچه می‌باشد. شایان ذکر است که Spring Boot امکانات امنیتی مانند جلوگیری از SQL Injection را نیز به همراه خود دارد.
- پیاده‌سازی سمت front-end، با استفاده از زبان Javascript و فریم‌ورک Vue.Js انجام می‌شود.
- قسمت‌های back-end و front-end با استفاده از REST API با یکدیگر ارتباط برقرار می‌کنند.
- از PostgreSQL به عنوان پایگاه داده و از Arvan S3 Storage برای ذخیره‌ی تصاویر استفاده می‌شود.
- تمامی قسمت‌های ذکر شده روی سرور مجازی ابری آروان و با استفاده از Docker راه‌اندازی می‌شوند.
- به عنوان Load Balancer از نرم‌افزار Nginx روی سرور استفاده می‌شود.
- از سرویس Cloudflare برای سرویس‌های Name Service استفاده می‌شود.
- از Github Actions برای خط لوله‌ی CI/CD استفاده می‌شود. ایمیج‌ها روی سرورهای گیت‌هاب ساخته شده، روی Docker Hub ارسال می‌شوند و سپس روی سرورهای خودمان به صورت self-hosted دانلود و اجرا می‌شوند.



معماری سطح دسترسی کاربران

کاربر عادی می‌تواند در صفحه مخصوص به محصول، برای آن محصول تیکت ثبت کند و تیکت‌های ثبت‌شده خود برای آن محصول را مشاهده کند. همچنین در صفحه دیگری می‌تواند تمام تیکت‌های ثبت شده خود را مشاهده کند. همچنین مدیر در صفحه مخصوص مدیریت می‌تواند تیکت‌های ثبت شده را مشاهده و مدیریت کند.

نقش‌های کاربران در زیر تعریف شده اند:

- **User:** کاربر عادی سامانه است.
- **Staff:** می‌تواند تیکت‌های مربوط به محصول را مدیریت کند.
- **Admin:** دسترسی کامل به محصول دارد. می‌تواند نقش کاربران در محصول را تغییر دهد.

برای پیاده‌سازی چنین ساختاری، یک راه حل این است که برای هر کاربر و هر محصول یک نقش تعریف شود که سطح دسترسی آن کاربر برای آن محصول را مشخص می‌کند.

معماری دیگری که می‌توانستیم برای سطح دسترسی کاربران استفاده کنیم، تعریف یک نقش برای کاربر برای فقط یک محصول است. این معماری این محدودیت را برای کاربر ایجاد می‌کند که می‌تواند مدیر فقط یک محصول باشد و برای مدیریت چند محصول لازم است چندین اکانت بسازد. از طرف دیگر این معماری از معماری قبلی ساده‌تر است. در نهایت با در نظر گرفتن برتری‌ها و ضعف‌های این دو معماری نسبت به یکدیگر، معماری دوم را به دلیل سادگی انتخاب کردیم.

برای این که بررسی سطح دسترسی کاربر قبل از انجام عملیات‌های مختلف ممکن باشد، سرویس‌های مربوط به مشاهده و مدیریت تیکت‌های یک محصول بر اساس محصول جدا می‌شوند (برای استفاده از سرویس باید محصول مشخص باشد). به این ترتیب قبل از انجام عملیات، بررسی می‌شود که آیا کاربر به آن محصول دسترسی لازم را دارد یا خیر و سپس عملیات مورد نظر انجام می‌شود.

معماری احراز هویت کاربران

پسورد به صورت رمزنگاری شده و یوزرنیم کاربران در پایگاه داده ذخیره می‌شوند. برای لاگین، در صورت تایید یوزرنیم و پسورد کاربر، یک session برای کاربر ایجاد می‌شود و توکن مربوط به آن به عنوان cookie برای کاربر ذخیره می‌شود. با هر request کاربر، زمان انقضای session به ۶ ساعت بعد تغییر می‌کند.

معماری تیکت‌ها

در تیکت‌ها باید امکان پرسش و پاسخ وجود داشته باشد. برای این که این امکان فراهم شود، در صفحه مربوط به تیکت یک قسمت برای چت بین کاربر و مدیر پیاده‌سازی می‌شود. راه حل دیگر برای ایجاد امکان پرسش و پاسخ، ارسال درخواست است. به این صورت که مدیر ابتدا از کاربر مسئله‌ای را می‌پرسد، سپس کاربر به این پرسش پاسخ می‌دهد و تنها بعد از پاسخ هر طرف، طرف دیگر می‌تواند یک درخواست ارسال کند. راه حل دوم ساده‌تر است اما برای هم مدیر و هم کاربر محدودیت ایجاد می‌کند در نتیجه راه حل اول انتخاب شد.

تیکت‌ها می‌توانند گونه‌های مختلفی داشته باشند و هر کدام از این گونه‌ها مشخصات بیشتر یا کمتری نسبت به سایر گونه‌ها دارند. برای پیاده‌سازی این قسمت، می‌توان تمام مشخصات مربوط به گونه‌های مختلف را در یک موجودیت نگه داشت و با تعریف یک ویژگی گونه برای این موجودیت، گونه‌های مختلف را از هم جدا کرد. پیاده‌سازی دیگر این است که یک موجودیت تیکت به عنوان پدر تعریف می‌شود و سایر گونه‌های تیکت به عنوان فرزند این موجودیت تعریف می‌شوند. با این کار می‌توان مشخصات خاص مربوط به یک گونه از تیکت را برای آن اضافه کرد. در پیاده‌سازی اول، برخلاف پیاده‌سازی دوم، لازم است همه مشخصات برای همه گونه‌ها تعریف شوند. همچنین پیاده‌سازی دوم انعطاف‌پذیری بیشتری دارد و تعریف رفتارهای مختلف برای گونه‌های مختلف آسان‌تر است. به همین دلایل پیاده‌سازی دوم انتخاب شد.

معماری پایگاه داده

برای انتخاب پایگاه داده می‌توان از پایگاه داده‌های Relational مانند PostgreSQL و یا پایگاه‌داده‌های Non-Relational (یا NoSQL) مانند MangoDB استفاده کرد. اما به توجه به ساختار داده‌های پروژه و همچنین آشنایی بیشتر اعضای تیم با پایگاه داده‌ی PostgreSQL تصمیم بر استفاده از PostgreSQL شد.

معماری ذخیره‌ی تصاویر

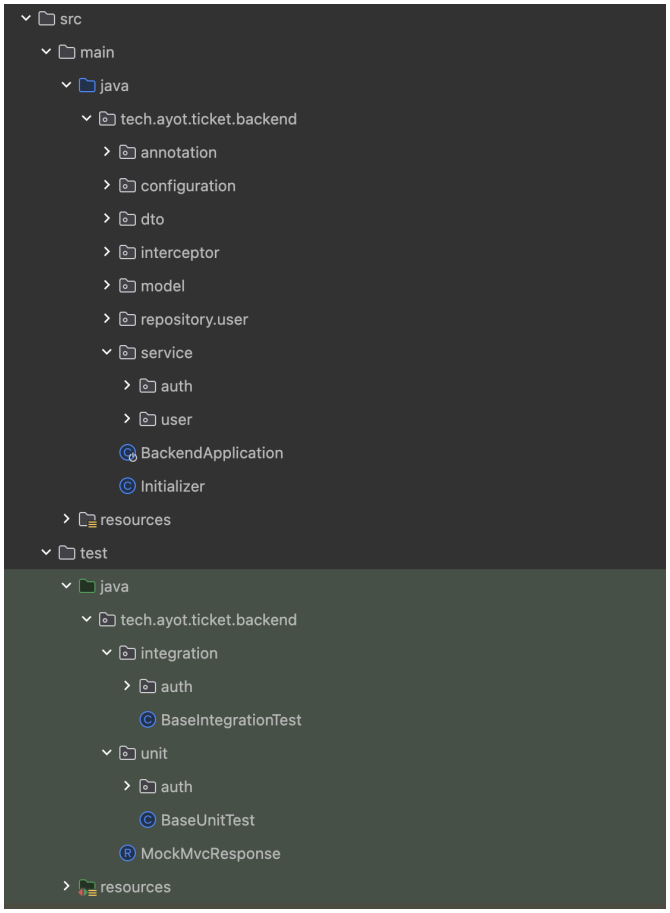
برای ذخیره‌ی تصاویر کاربران و کسب و کارها، می‌توانستیم از پایگاه داده‌ی postgreSQL استفاده کنیم که باعث سنگین شدن پایگاه داده، کند شدن API ها و همچنین مقیاس‌پذیری پایین سیستم می‌شد. در نتیجه از راه حل دوم که استفاده از سیستم S3 Storage ابر آروان (مطابق با AWS S3) بود استفاده کردیم که در این روش، بخش بک‌اند درگیر ارسال و دریافت تصاویر نمی‌شود و تنها UUID مربوط به آن‌ها را در پایگاه داده نگهداری

می‌کند. بخش فرانت‌اند با استفاده از این UUID ها به صورت مستقیم از سرویس S3 تصاویر را بارگذاری می‌کند.

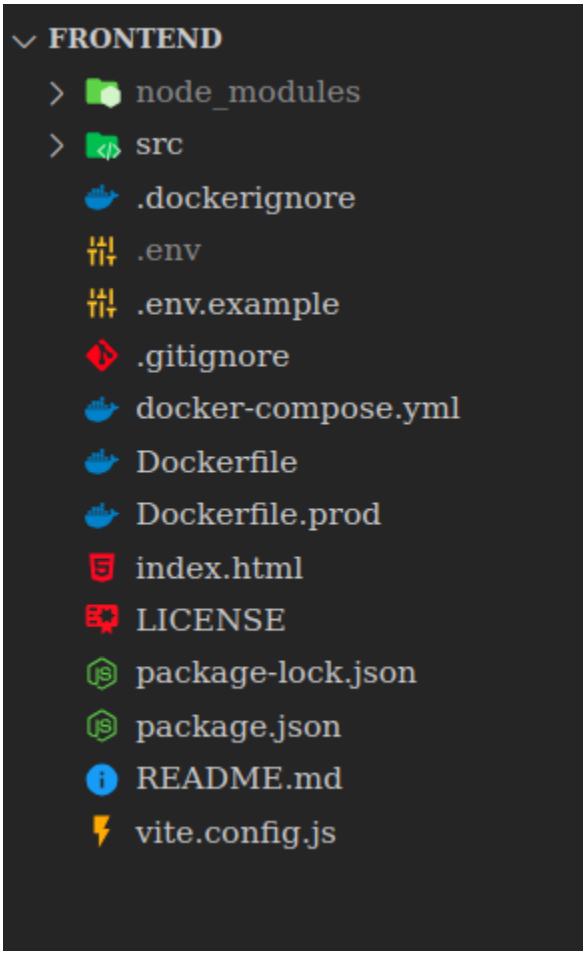
ساختار پروژه - بخش بک‌اند

پکیج‌های اصلی، repository ،model و service می‌باشند که هر کدام شامل زیرپکیج‌هایی می‌شوند. پکیج model شامل موجودیت‌های تعریف شده است. پکیج repository شامل سرویس‌هایی برای برقرار ارتباط با پایگاه داده می‌باشد و در پکیج service، سرویس‌هایی که تعریف شده اند شامل REST API و سایر سرویس‌ها مانند Authorization قرار دارند.

تست‌ها به دو بخش unit و integration تقسیم شده‌اند و هر کدام از این بخش‌ها به زیرپکیج‌هایی تقسیم می‌شوند.

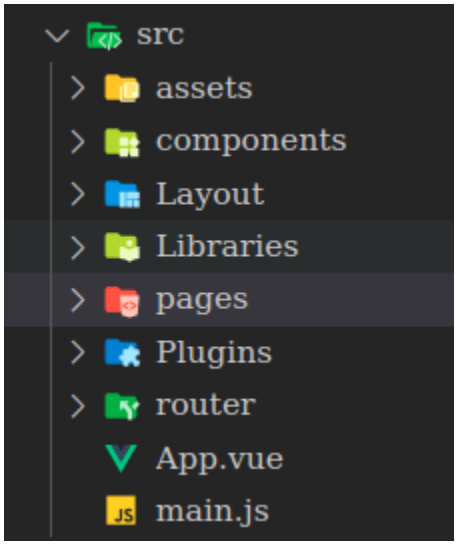


ساختار پروژه - بخش فرانت‌اند

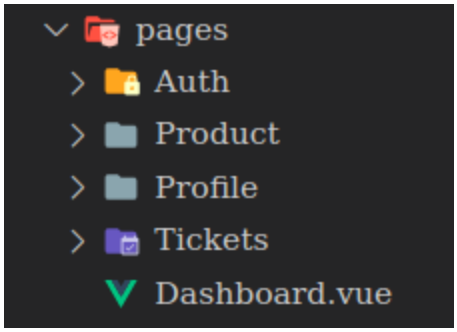


در بخش Front-end، اعضای کلیدی پروژه در پوشه src و فایل package.json هستند و هر چیز دیگری، ابزار کمکی محسوب میشود. لازم است تا در چارچوب های برنامه نویسی front-end از bundler استفاده شود تا کدها بهینه و به زبان مرورگر ترجمه شوند. ترجیح ما در این پروژه vite است به علت سرعت بالای پردازش در آن. در فایل package.json نیازمندی ها و dependency های پروژه

آورده شده است که با دستور npm install تمام آن کتابخانه ها نصب شده و در پوشه node_modules قرار میگیرند. همچنین برای راحتی بیشتر کار و ایزوله سازی محیط های اجرا از Docker استفاده میشود.



در پوشه src که کد های مربوط به برنامه در آن است، فایل App.vue نقطه آغاز کار است و با استفاده از vue-router به صفحات و مسیر های دیگر تغییر پیدا میکند. شکل کلی وب سایت شامل Header, Footer, Body, Navbar در پوشه Layout و کامپوننت های قابل استفاده در شرایط مختلف در components هستند. پوشه های plugins و libraries پوشه هایی برای تنظیم کردن (configuration) کتابخانه های مختلف هستند به طوری که میتوان آنها را جوری تنظیم کرد تا در تمام پروژه مورد استفاده قرار گیرند.



و اما در پوشه pages که صفحات اصلی سایت قرار دارد، پوشه های جدایی داریم که به طور مشخصی، کاربر و محصول و تیکت را از هم جدا کرده است. همچنین در پوشه Auth نیز فرآیند های مربوط به ثبت نام و ورود و نکات امنیتی انجام میشوند.

ساختار پروژه - مدیریت نسخه

برای مدیریت نسخه، نگهداری code base و همچنین دیپلوی روی سرور، از ابزار Git و پلتفرم GitHub استفاده می‌شود. پروژه شامل سه ریپازیتوری TicketMaster-Frontend، TicketMaster و TicketMaster-Backend می‌شود که بخش Frontend و Backend داخل ریپازیتوری ریشه‌ای TicketMaster به عنوان Submodule قرار داده شده‌اند.

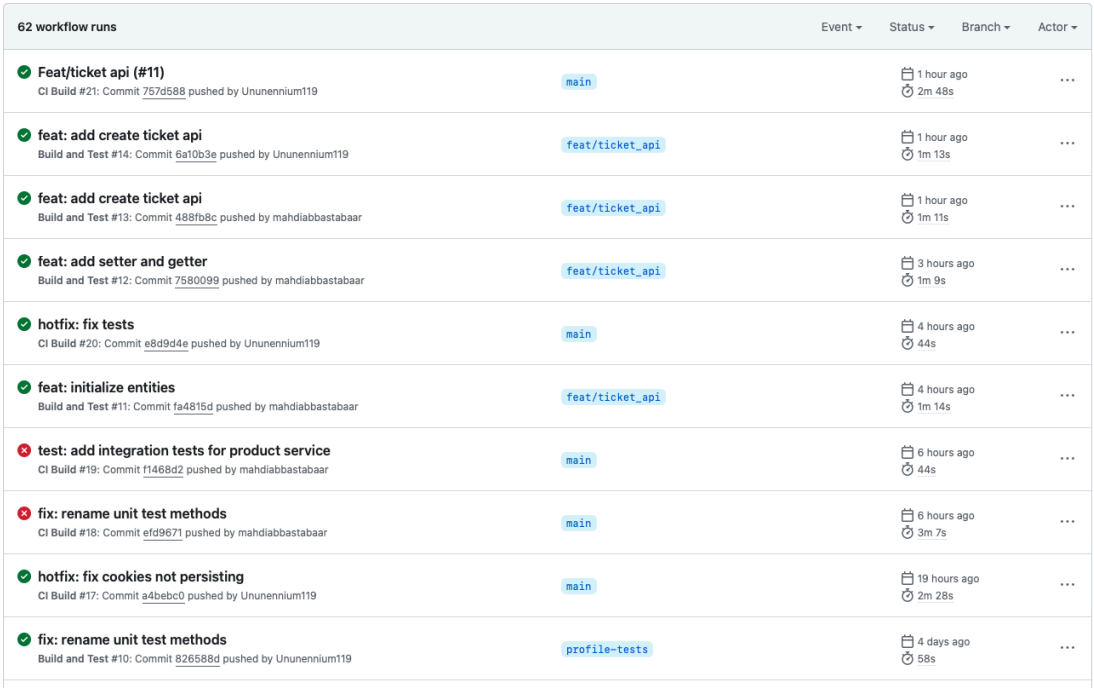
خط لوله‌ی CI/CD با استفاده از Github Actions

در فاز دوم پروژه که به صورت CD Lean جلو رفته است، برای deploy های اتوماتیک و سریع از خط لوله‌ی CI/CD استفاده شده. زیرساخت استفاده شده برای این امر، Github Actions است. برای هر دو قسمت فرانت‌اند و بک‌اند، به ازای هر commit بر روی شاخه‌ی main، پروژه بر روی سرورهای گیت‌هاب (تحت لینوکس) build شده و تست‌ها اجرا می‌شوند. در صورت پاس شدن تمامی تست‌ها، ایمیج داکر روی همان سرورها و با استفاده از مقادیر secret ذخیره شده روی قسمت secrets

گیت‌هاب (مانند پسورد دیتابیس و ...)، ساخته شده و بر روی Docker Hub ارسال می‌شود. در مرحله‌ی بعدی خط لوله که به صورت self hosted بر روی سرور اجرا می‌شود، ایمیج ساخته شده از روی Docker Hub دانلود شده و سپس با استفاده از Docker Compose بر روی سرور اصلی اجرا می‌شود.

در پروژه‌ی بک‌اند، در صورت ارسال کامیت بر روی شاخه‌های غیر از main، صرفا پروژه را روی سرورهای گیت‌هاب build می‌کند و تست‌ها را اجرا می‌کند ولی چیزی بر روی سرور اصلی دیپلوی نمی‌شود.

لازم به ذکر است که این فرآیند در ایران به علت تحریم Docker Hub با مشکلاتی مواجه می‌شد ولی با استفاده از سرورهای گیت‌هاب و سرویس «شکن» این مشکل برطرف شد. استفاده از مخازن ابر آروان برای ایمیج‌های داکر گزینه ی مناسبی نیست چرا که به صورت مکرر به روز نمی‌شوند و نسخه‌ی قدیمی بر روی سرور دیپلوی می‌گردد.



صفحه‌ی Actions برای قسمت بک‌اند پروژه