ساختار پروژه

در این فاز از شما خواسته شده است طراحی میکروسرویسی پروژه‌ای که در فاز اول را بدست آوردید را با کمک داکر پیاده سازی کنید. ساختار کلی کد پروژه بدین شکل است که در پوشه اصلی یک فایل docker-compose وجود دارد که با کمک آن می‌توان کل پروژه را با یک دستور اجرا کرد.

بقیه میکروسرویس‌ها هرکدام در یک از پوشه جداگانه در کنار فایل docker-compose قرار دارند.

- root dir of your project:
  - docker-compose.yml
  - microservice1 dir
  - microservice2 dir
  - microservice3 dir

نیازمندی‌های تحویل پروژه

- برای نوشتن کد پروژه، اولویت با استفاده مجدد(reuse) از کدهای موجود در پروژه monolith است.
- کد شما می‌بایست از ساختاری که در مرحله دوم مستندات آن را ارسال کردید پیروی کند.

- برای تست API ها، در صورت استفاده از Postman، فایل خروجی کالکشن مورد نیاز است.
- برای استفاده از هرکدام از api های خارجی برنامه، احراز هویت با کمک توکن ضروری است.

نکات پیاده‌سازی داکر

- کد هریک از میکروسرویس‌ها ایزوله بوده و تنها از طریق شبکه به میکروسرویس‌های دیگر دسترسی دارد.
- هریک از میکروسرویس‌ها، در حداقل یک container داکر قرار می‌گیرند.
- استفاده از sqlite برای ذخیره داده‌ها مجاز نمی‌باشد. توصیه می‌شود از کانتینرهای postgres، mysql یا mongodb برای ذخیره داده استفاده کنید.

- با توجه به این که هریک از میکروسرویس‌ها یک دیتابیس مخصوص به خودش دارد، لازم است یک container جداگانه نیز برای دیتابیس آن در نظر گرفته شود.
- در مورد container های دیتابیس‌ها، نیاز داریم داده‌ها پس از هربار ریستارت شدن سیستم هاست یا کانتینر داکر، از بین نروند.
- در مورد container ها، نیاز داریم پس از هربار ریستارت شدن سیستم به صورت خودکار اجرا شوند.
- هیچ‌کدام از container ها، امکان استفاده از بیش از ۲۵۶ مگابایت رم و ۵۰ درصد یک هسته سی پی‌یو را ندارند.

فراخوانی API داخلی: gRPC

یک پروتکل برای فراخوانی یک سرویس از یک برنامه دیگر است، تقریبا مشابه REST API اما سریع‌تر، بهینه تر و ایمن تر.

این پروتکل مستقل از زبان برنامه نویسی است، بنابراین با کمک آن، دو میکروسرویس که با زبان‌های مختلف نوشته شده اند می‌توانند با یکدیگر ارتباط برقرار کنند.

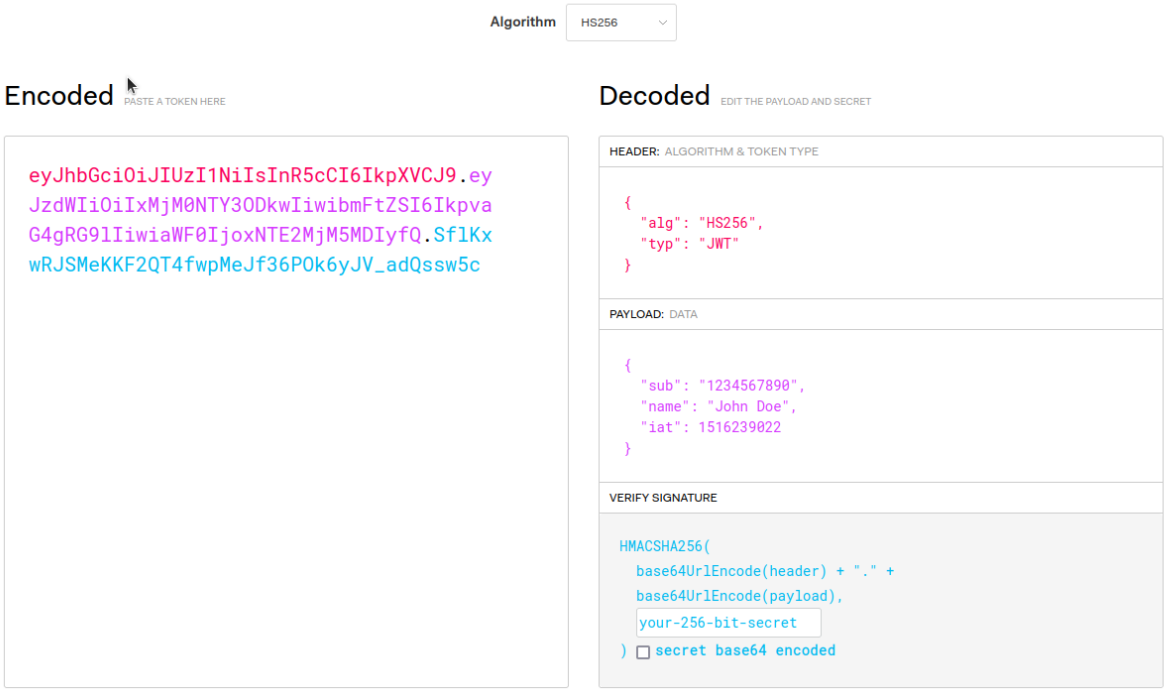
با توجه به این که طبق توضیحات فاز قبل برنامه شما حداقل یک internal call دارد، شما نیاز دارید با کمک این پروتکل این فراخوانی بین دو میکروسرویس مورد نظر را انجام دهید.

برای کار با grpc، ابتدا لازم است برای اطلاعاتی که قصد تبادل آنها را دارید یک شِما تعریف کنید. در این شما مشخص می‌شود این سرویس چه نوع اطلاعاتی را در اختیار سرویس گیرنده قرار می‌دهد.

احراز هویت: JWT

با توجه به این که در کد شما یک سرویس user management وجود دارد، و دسترسی به API های هر سه میکروسرویس باید مشروط به احراز هویت کاربر باشد، می‌توانید از JWT ها برای این کار استفاده کنید.

زمانی که کاربر، لاگین موفقی را به سیستم انجام می‌دهد، یک توکن توسط سرور صادر شده و با فرمت JWT به سمت کلاینت ارسال می‌شود. این توکن باید به صورت محلی در سمت کاربر(معمولا در مرورگر) ذخیره شود.

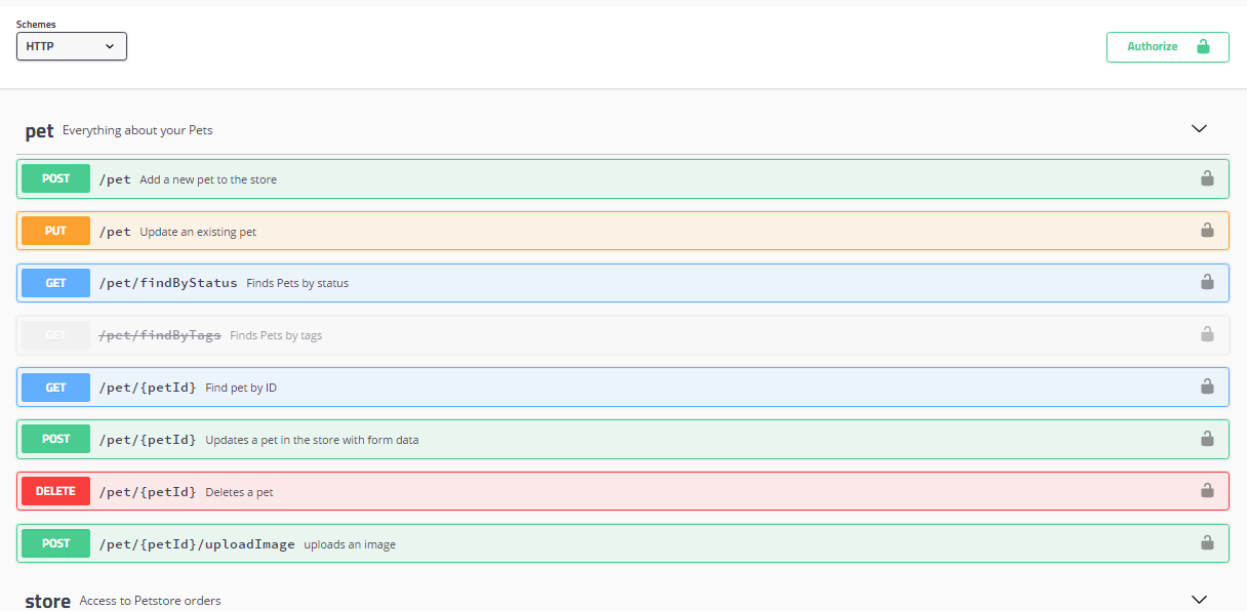


این روش اعتبارسنجی، بدون حالت (stateless) است، از این جهت که وضعیت کاربر هیچگاه در سمت سرور ذخیره نمی‌گردد. API سمت سرور ابتدا به دنبال هدر Authorization فوق در درخواست دریافتی می‌گردد، اگر یافت شد و اصالت آن تایید شد کاربر امکان دسترسی به منبع محافظت شده را پیدا می‌کند. نکته‌ی مهم اینجا است که چون این توکن‌ها قابلیت ذخیره سازی مقدار محدودی داده با عنوان payload را دارا هستند می‌توان تمام اطلاعات لازم جهت احراز هویت کاربر را در آن قرار داد و دیگر نیازی به رفت و برگشت به بانک اطلاعاتی یا ارتباط میکروسرویس‌های مختلف با سیستم user management جهت تایید این اطلاعات تصدیق شده نیست.

مستندسازی و تست: SwaggerUI

SwaggerUI یک ابزار برای ایجاد ایجاد مستندات برای API ها می‌باشد. با کمک SwaggerUI، می‌توانید علاوه بر ایجاد مستندات، به صورت تعاملی با endpoint ها کار کرده و آنها را تست کنید.

فریم ورک‌های محبوب اکثرا از SwaggerUI پشتیبانی می‌کنند، به عنوان مثال در فریم ورک Django می‌توانید با کمک کتابخانه‌های drf-spectacular یا drf-yasg، به صورت خودکار یک صفحه مشابه شکل زیر ایجاد کنید:



تست هرکدام از endpointها: