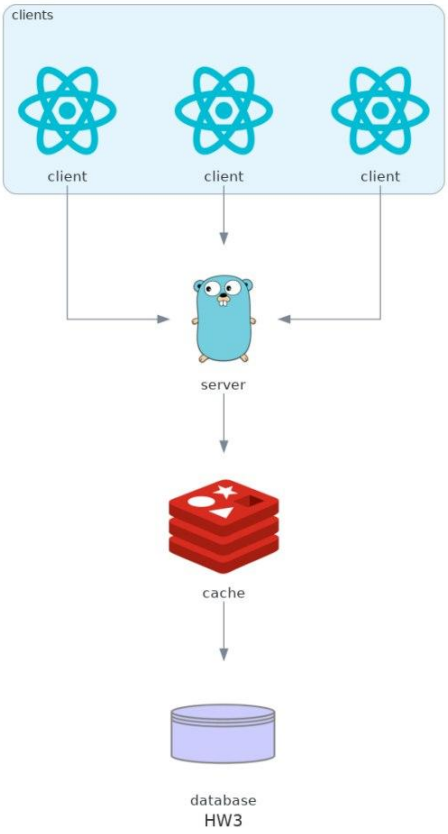
تمرین ۳ ام برنامه‌نویسی وب

استاد درس: امید جعفری‌نژاد
طراحی و ویرایش: مصطفی قدیمی، فرزین نصیری
تاریخ تحویل: ۱۹ ام دی‌ماه ۱۴۰۰

مقدمه

در این تمرین شما یک وب‌اپلیکیشن فول‌استک(بک‌اند_فرانت‌اند_دیتابیس_کش) را پیاده‌سازی می‌کنید. کلاینت، یک وب‌اپلیکیشن ساده برای نگهداری یادداشت‌های هر کاربر است. کاربران می‌توانند با نام کاربری و رمز عبور خود وارد برنامه شوند، طبیعتا درخواست‌های کاربران توسط بک‌اند مدیریت می‌شود. بک‌اند این سیستم با یک RDBMS در ارتباط است اما برای افزایش سرعت سیستم در درخواست‌های از نوع خواندن از یک cache استفاده می شود، هرچند برای پایگاه‌داده می‌توانید از دیتابیس‌های آماده استفاده کنید اما cache را باید به عنوان یک برنامه جداگانه پیاده کنید. تصویر زیر یک شمای کلی از ساختار سیستم به شما میدهد.



شکل ۱: معماری تمرین

در ادامه جزییات هر بخش این سیستم به همراه معیارهای نمره‌دهی و بخش‌های امتیازی آمده است. در انتهای این مستند نیز تعداد لینک مفید و منابع جهت مطالعه بیش‌تر گذاشته شده است.

توضیحات

۱.۰ کلاینت وب

برای پیاده‌سازی کلاینت وب شما تنها مجاز به استفاده از فریم‌ورک React هستید. برای طراحی اجزا و component های صفحات با توجه به تمرین‌های گذشته، می‌توانید از یکی ابزارهای آماده مانند SemanticUI Bootstrap، CSS، Tailwind و... به انتخاب خودتان استفاده نمایید. کلاینت وب باید بتواند با بک‌اند صحبت کند و درخواست‌ها را با پاسخ‌ها یا خطاها و پیغام‌های مناسب جواب دهد که در بخش authorization و authentication به آن بیش‌تر پرداخته شده است.

endpoint	method HTTP	description
/notes/new	POST	ایجاد یادداشت جدید
/notes/<note-id>	GET	گرفتن یادداشت خاص
/notes/<note-id>	PUT	آپدیت یادداشت خاص
/notes/<note-id>	DELETE	حذف یادداشت خاص

۲.۰ بک‌اند و سرویس‌های زیرساخت

در این بخش به جزئیات پیاده‌سازی و طراحی بک‌اند و دو سرویس زیرساخت برای ذخیره‌سازی داده‌ها می‌پردازیم. شما میتوانید از Go یا JS برای پیاده‌سازی این بخش‌ها استفاده کنید و به طور کلی مجاز هستید از فریم‌ورک‌ها و کتابخانه‌های موجود استفاده کنید (مگر اینکه در آن بخش به طور خاص اشاره شده باشد که چنین حقی ندارید). لیستی از وب فریم‌ورک‌های پیشنهادی در انتهای این مستند آمده است اما در استفاده از آن‌ها (و یا فریم‌ورک انتخابی خود) به این نکته توجه کنید که به ازای هر پیچیدگی که از نرم‌افزار شما کم می‌کنند، پیچیدگی دیگری (در قالب یاد گرفتن و کارکردن با آن فریم‌ورک و محدودیت‌های آن) به کار شما اضافه می‌کنند. در سطح این تمرین خود زبان Go (و کتابخانه استاندارد آن) و یا ابزارهایی مثل Node و Express پاسخگوی بسیاری از چالش‌ها هستند.

۱.۲.۰ سرویس Cache

Cache شما یک نرم‌افزار جداگانه نوشته شده با زبان Go یا JS است که انتظار می‌رود هاییAPI جهت استفاده هر نرم‌افزار دیگری ارائه دهد. این نرم‌افزار باید به صورت جدا و standalone روی پورتی که از قبل مشخص شده‌ای بالا بیاید (مثلا ۸۰۸۰). همچنین این کش فقط key-value ذخیره می‌کند و از روش LRU (اطلاعات بیشتر) جهت مدیریت این هاkey-value استفاده می شود و value هایی که بیشترین استفاده را دارند بیش‌ترین استفاده را دارند در دسترس‌تر هستند و اگر cache پر شود، داده با کم‌ترین استفاده پاک می‌شود.

نکات زیر را در پیاده‌سازی لحاظ کنید:

۱.پورتی که cache روی آن بالا می‌آید به همراه حداکثر حافظه کش (تعداد عناصر قابل ذخیره) باید از کامندلاین یا کانفیگ فایل گرفته شوند و نباید در برنامه hard code شده باشند.

۲ـ چه key ها و چه value هایی فقط میتوانند از نوع int۶۴ یا string باشند. همچنین کاربر نهایی نباید درگیر مشخص کردن نوع این متغیرها باشند و خود cache باید آن را مدیریت کند.

۳ـ با توجه به اینکه LRU cache نیاز به LinkedList دارد (که در کتابخانه استاندارد Go وجود ندارد) شما آزاد هستید تا از کتابخانه‌های موجود استفاده کنید.

۴ـ حداکثر اندازه key ها ۶۴ کاراکتر و حداکثر اندازه value نیز ۲۰۴۸ کاراکتر است

۵ـ اگر miss cache در درخواست‌های Get اتفاق بیفتد، این خطا باید کاربر برنامه اطلاع داده شود.

۶ـ تمام خطاهای ممکن باید به درستی مدیریت شوند و به کاربر به شکل درستی انتقال داده شوند(برنامه نباید هنگام خطا گرفتن دچار panic شود و پایین میاید)

دیگر جزئیات پیاده سازی ساختمان‌داده مورد نیاز کش و API ها و دیگر موارد به عهده خودتان است اما از منظر یک کاربر خارجی کش شما باید API های زیر را داشته باشد:

RPC	description
GetKey	key a of value the gets
SetKey	key existing or new a for value a set
Clear	values and keys all clears

همچنین انتظار میرود کش شما در مقابل دسترسی‌های چندگانه ایمن باشد(Thread Safety)، شما باید با استفاده از سمافورهای باینری(mutex) و یا چنل‌ها (در صورت استفاده از go) ساختمان داده‌ای طراحی کنید که دچار condition race و inconstancy نمیشود. روش ارتباط بین بک‌اند و کش از طریق gRPC است(توضیحات بیشتر جلوتر بیان شده است) البته شما میتوانید برای این بخش از http نیز استفاده کنید، اما در اینصورت حداکثر ۵۰ درصد از نمره این بخش را خواهید گرفت.

در نهایت دقت کنید که شما در کد سرویس کش مجاز به استفاده از کتابخانه‌ها و فریمورک‌ها موجود نیستید(بجز در بخش مربوط به api endpoint ها و بالا آوردن سرور برای گرفتن و پاسخ به درخواست‌ها).

۲.۲.۰ دیتابیس

در این بخش کافیست از PostgreSQL به عنوان یک RDBMS استفاده کنید. پس از نصب این برنامه روی کامپیوتر خودتان(چه با Docker چه به صورت عادی) میتوانید از یک کاربر و دیتابیس در آن ایجاد کنید و در نرم‌افزار خود استفاده کنید. میتوانید از این جهت نصب و راه‌اندازی اولیه کمک بگیرید.

سعی کنید شِمایِ دیتابیس را در کد بک‌اند خود پیاده‌سازی کنید. برای اینکار میتوانید از ORM هایی مثل gorm در Go و یا ORM های موجود برای JS استفاده کنید. روش کار به این صورت است که شما Data Model را با زبان برنامه‌نویسی خود مشخص میکنید(جدول‌ها و شمای دیتابیس توسط کد مشخص میشود) و ORM کد شما را تبدیل به SQL کرده و دیتابیس را migrate میکند. جهت Query زدن روی دیتابیس نیز عمدتا نیازی به SQL ندارید و با استفاده از توابعی که ORM در اختیارتان میگذارد میتوانید کار خود را انجام دهید. لازم به ذکر است که انتظار طراحی شمای پیچیده و جداول زیادی نداریم و با توجه به سادگی اپلیکیشن سعی کنید به ساده‌ترین راه‌حل‌ها و ساختارها تکیه کنید.

۳.۲.۰ بک‌اند

بک‌اند بخشی از سیستم است که تمام ۳ بخش قبلی را به هم وصل میکند. این بخش handler های مورد نیاز برای کلاینت وب را فراهم میکند و ویژگی‌های authentication and authorization را پیاده‌سازی کرده است. طبیعتا انتظار داریم این ویژگی‌ها به درستی کار کنند.

برای پیاده‌سازی بخش authorization و authentication از JWT استفاده کنید. (اطلاعات بیشتر) استفاده از JWT یک روش بسیار متداول است که به توسعه‌دهندگان و کاربران این امکان را می‌دهد تا با استفاده از یک روش بسیار فشرده، بتوانند اطلاعات خود را از طریق object ها با فرمت JSON به صورت امن جابه‌جا کنند. این توکن‌ها در header هر درخواستی‌ای (request) قرار گرفته و در سمت بک‌اند verify می‌شود که آیا توکن فرستاده شده معتبر است و یا این‌که ساختگی است. همان‌طور که اشاره شد، این توکن‌ها حاوی اطلاعاتی هستند که با استفاده از الگوریتم‌های رمزنگاری معروف مثل RSA و دیگر الگوریتم‌ها encrypt شده‌اند، بنابراین قابلیت authorization را هم می‌دهد. در صورتی که یک کاربر ثبت‌نام و لاگین نکرده بود و قصد گرفتن داده‌ی (یادداشت) خاص و یا فرستادن اطلاعات جدید (یادداشت جدید) را داشت باید این پیغامی با مفهوم این‌که «شما باید ابتدا ثبت‌نام و لاگین کنید» به او نمایش دهد و چنین دسترسی‌هایی به او داده نشود. هر کاربر فقط می‌تواند اطلاعات مربوط به یادداشت‌های خود را مشاهده کند و نمی‌تواند یادداشت‌های مربوط به دیگران را بخواند. بنابراین اگر کاربر A قصد مشاهده‌ی یادداشت‌های کاربر B را داشت باید به او پیغامی تحت عنوان این‌که «شما اجازه‌ی دسترسی به این اطلاعات را ندارید» نمایش داده شود.

نکته مهم: کاربر ادمین باید دسترسی (اضافه کردن، گرفتن، آپدیت و حذف) به همه‌ی یادداشت‌ها را داشته باشد.

به جز دو ویژگی بالا، بک‌اند شما باید توانایی Rate Limiting یا throttling داشته باشد، یعنی هر کاربر در بازه زمانی ۱ دقیقه مجاز به فقط n ریکوئست(n از کامندلاین یا کانفیگ گرفته میشود) است و در غیر اینصورت باید خطای مناسبی دریافت کند و بعد از مدتی دوباره تلاش کند(این خطا باید از سمت کلاینت هم به درستی مدیریت شود) به طور خاص limiting rate را میتوانید توسط این برنامه به صورت اتوماتیک تست کنید.

در اینجا مجاز به استفاده از هیچ کتاب‌خانه‌ای نیستید. بک‌اند باید بتواند ارتباط خود را با دیتابیس و cache خود مدیریت کند و از آن‌ها درجای مناسب استفاده کنید(مثلا هنگام Get کردن یک یادداشت، اگر miss cache شد آن یادداشت را از دیتابیس بگیرد و در cache هم اضافه کند). با توجه به اینکه چگونگی استفاده از cache و دیتابیس در پاسخ به ریکوئست کاربران در در سرعت و کارایی سیستم شما بسیار موثر است، در اینجا نحوه برخورد با هر ریکوئست تشریح نشده است و بر عهده خودتان آن است که به بهترین شکل ممکن آن‌ها را پیاده کنید. در اینجا مجاز به استفاده از هیچ کتابخانه یا ابزار کمکی در کد نیستید.

همچنین دقت کنید که اطلاعات کانکشن به cache و دیتابیس(و هر نوع کانفیگ دیگری) نیز باید به وسیله کانفیگ فایل یا از کامندلاین و یا به وسیله متغیر محیطی به برنامه داده شود.

قبلا بیان شده بود که ارتباط بین بک‌اند و سرویس cache از نوع gRPC، بنابراین از این یا این میتوانید استفاده کنید و پیشنیازهای اساسی را نصب کنید. سپس فایل‌های proto مورد نیاز هر سرویس را نوشته و به همراه بقیه سورس کدها قرار دهید. در اینجا دقت کنید کانکشن بین کلاینت و بک‌اند از نوع http است و نیازی به ssl نیست. راهنمایی‌های بیشتر به عنوان منابع مفید در انتهای مستند آمده است.

۳.۰ نکات تکمیلی

نکات تکمیلی: در تمام کدهایی که مینوسید انتظار داریم از اصول پایه و اساسی مهندسی نرم‌افزار استفاده کنید(کد کثیف نزنید!). استفاده از اصولی مثل S.O.L.I.D یا The Twelve-Factor App پیشنهاد میشود. معیار نمره‌دهی در اینجا این است که اساسی ترین ویژگی‌های طراحی که از یک نرم‌افزار خوب انتظار میرود را برآورده کرده باشید(توابع خیلی بزرگ نباشند، ماژول‌ها مینیمال و خوب طراحی شده باشند، کد خوانایی بالایی داشته باشد، از کامنت به شکل مناسبی استفاده شده است، عدم وجود وابستگی(dependency) مستقیم از کلاینت به سرویس‌ها داخلی مثل cache و دیتابیس ...) با این حال درستی کارکردهای اساسی کل سیستم عمده نمره‌ی شما را شکل خواهد داد بنابراین سعی کنید در ابتدا نرم‌افزاری بنویسید که به درستی کار میکند و در ادامه به جزییات دیگر بپردازید. همچنین در صورتی که در استفاده کردن از ابزارهای کمکی، کتابخانه‌ها یا فریمورک‌ها دچار ابهام یا مشکل شدید میتوانید مشکل خود را با طراحان تمرین در میان بگذارید.

۴.۰ امتیازی

عدد جلوی هر گزینه وزن آن را مشخص می‌کند. شما می‌توانید حداکثر به جرم ۱۰ از آیتم‌ها را پیاده کنید و حداکثر ۱۰ درصد نمره کل تمرین را دریافت کنید و پیاده‌سازی گزینه‌های بیشتر نمره‌ای نخواهد داشت.

- خلاقیت در UI وب‌اپلیکیشن[۱]
- استفاده از روش singleflight در پیاده سازی endpoint ها[۲]
- خلاقیت در طراحی بک‌اند یا سرویس‌ها[۱]
- پیاده سازی cache توزیع شده و یا مجموعه‌ای از سرویس‌های cache جهت fault-tolerance یا کاهش زمان پاسخ‌دهی یا ... [۴]
- اضافه کردن احرازهویت و یا ارتباط امن(ssl/tls) در سرویس cache (به همراه پیاده‌سازی‌های لازم در بک‌اند) [۲]
- اضافه کردن قابلیت load-balancing (می‌توانید از nginx یا traefik یا enovy و یا ابزار مشابه دیگر استفاده کنید)[۳]

۵.۰ منابع مفید

- go best practices
- gRPC
- Go and gRPC
- go books
- S.O.L.I.D
- Go buffalo
- Go Gin
- Sequelize
- Parse
- typeorm
- The Twelve-Factor App

زمان‌بندی‌ها و نحوه تحویل

تمرین جزییات دارد و از موکول کردن آن به روزهای آخر اکیدا پرهیز کنید. پاسخ خود را در سازمان درس بارگذاری کنید و در README حتما نام و شماره دانشجویی افراد گروه را بنویسید. تاریخ آخرین کامیت شما معیار محاسبه تاخیر خواهد بود.

سلامت باشید