



تمرین ۳ ام برنامه نویسی وب

استاد درس: امید جعفری نژاد

طراحی و ویرایش: مصطفی قدیمی، فرزین نصیری

تاریخ تحویل: زمان دقیق تحویل

مقدمه

در این تمرین شما یک وب اپلیکیشن به همراه نرم افزار سمت سرور (بک اند) و cache و دیتابیس را پیاده سازی کنید. کلاینت وب یک وب اپلیکیشن ساده برای نگهداری یادداشت های هر کاربر است. کاربران می توانند با نام کاربری و رمز عبور خود وارد برنامه شوند، طبیعتاً درخواست های کاربران توسط بک اند مدیریت می شود. بک اند این سیستم با یک RDMS در ارتباط است اما برای افزایش سرعت سیستم در درخواست های از نوع خواندن از یک cache استفاده می شود، هر چند برای پایگاه داده می توانید از دیتابیس های آماده استفاده کنید اما cache را باید به عنوان یک برنامه جداگانه پیاده کنید. تصویر زیر یک شمای کلی از ساختار سیستم به شما می دهد.

در ادامه جزئیات هر بخش این سیستم به همراه معیارهای نمره دهی و بخش های امتیازی آمده است. در انتهای این مستند نیز تعداد لینک مفید و منابع جهت مطالعه بیش تر گذاشته شده است.

توضیحات

۱.۰ کلاینت وب

برای پیاده سازی کلاینت وب شما تنها مجاز به استفاده از فریم ورک React هستید. برای طراحی اجزا و های component صفحات با توجه به تمرین های گذشته، می توانید از یکی ابزارهای آماده مانند SemanticUI Bootstrap، CSS، Tailwind و... به انتخاب خودتان استفاده نمایید. کلاینت وب باید بتواند با بک اند صحبت کند و درخواست ها را با پاسخ ها یا خطاها و پیغام های مناسب جواب دهد که در بخش authorization و authentication به آن بیش تر 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

یکی از مهم ترین بخش های این قسمت ارتباط بین بک اند و کلاینت وب هست. برای پیاده سازی این بخش معمولاً دو راهکار وجود دارد:

۱. طراحی صفحات و پر کردن آن با اطلاعات آزمایشی.
۲. پیاده سازی صفحات و قسمت بک اند به صورت هم روند و iterative و incremental. با توجه به اهمیت این ارتباط برای بار اول پیشنهاد می شود از روش دوم استفاده کنید.

۲.۰ بک اند و سرویس های زیر ساخت

در این بخش به جزئیات پیاده سازی و طراحی سرور بک اند و دو سرویس زیر ساخت برای ذخیره سازی داده ها می پردازیم. شما میتوانید از Go یا JS برای پیاده سازی این بخش ها استفاده کنید و به طور کلی مجاز هستید از فریم ورک ها و کتابخانه های موجود استفاده کنید (مگر اینکه در آن بخش به طور خاص اشاره شده باشد که چنین حقی ندارید). لیستی از وب فریم ورک های پیشنهادی در انتهای این مستند آمده است اما در استفاده از آن ها (و یا فریم ورک انتخابی خود) به این نکته توجه کنید که به ازای هر پیچیدگی که از نرم افزار شما کم می کنند، پیچیدگی دیگری (در قالب یاد گرفتن و کار کردن با آن فریم ورک و محدودیت های آن) به کار شما اضافه می کنند. در سطح این تمرین خود زبان Go (و کتابخانه استاندارد آن) و یا ابزارهایی مثل Node و Express پاسخگوی بسیاری از چالش ها هستند.

۱.۲.۰ سرویس Cache

Cache شما یک نرم افزار جداگانه نوشته شده با زبان Go یا JS است که انتظار می رود APIهایی جهت استفاده هر نرم افزار دیگری ارائه دهد. این نرم افزار باید به صورت جدا و standalone روی پورتهای که از قبل مشخص شده ای بالا بیاید (مثلا ۸۰۸۰). همچنین این کش فقط key-value ذخیره می کند و از روش LRU (اطلاعات بیشتر) جهت مدیریت این key-value استفاده می کند (value) که بیشترین استفاده را دارند در دسترس تر هستند و اگر cache پر شود، داده با کمترین استفاده پاک می شود).

نکات زیر را در پیاده سازی لحاظ کنید:

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

۲. چه های key و چه های value 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.

۳. با توجه به اینکه cach LRU نیاز به List Linked دارد (که در کتابخانه استاندارد Go وجود ندارد) شما آزاد هستید تا از کتابخانه های موجود استفاده کنید.

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

۵. اگر miss cach در درخواست های Get اتفاق بیفتد، این خطا باید کاربر برنامه اطلاع داده شود.

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

دیگر جزئیات پیاده سازی ساختمان داده 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 می رود کش شما در مقابل دسترسی های چندگانه ایمن باشد (Thread Safety)، شما باید با استفاده از سمافورهای باینری (mutex) و یا چنل ها (در صورت استفاده از go ساختمان داده ای طراحی کنید که دچار condition race و inconstancy نمیشود. روش ارتباط بین بک اند 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 کتابخانه ها و فریمورک ها موجود نیستید (بجز در بخش مربوط به api endpoint ها و بالا آوردن سرور برای گرفتن و پاسخ به درخواست ها).

۲.۲.۰ دیتابیس

در این بخش کافیسیت 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 نرم افزار خود استفاده کنید. میتوانید از این جهت نصب و راه اندازی اولیه کمک بگیرید.

سعی کنید شما دیتابیس را در کد بک اند خود پیاده سازی کنید. برای اینکار میتوانید از هابی ORM مثل gorm در Go و یا های ORM موجود برای JS استفاده کنید. روش کار به این صورت است که شما Model Date را با زمان برنامه نویسی خود مشخص میکنید (جدول ها 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 عمدتاً نیازی به SQL ندارید و با استفاده از توابعی که ORM در اختیاران 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 ساده ترین راه حل ها و ساختارها تکیه کنید.

۳.۲.۰ بک اند

بک اند بخشی از سیستم است که تمام ۳ بخش قبلی را به هم وصل میکند. این بخش های handler مورد نیاز برای کلاینت وب را فراهم میکند و ویژگی های authentication and authorization را پیاده سازی کرده است. طبیعتاً انتظار داریم این ویژگی ها به درستی کار کنند.

برای پیاده‌سازی بخش authentication و authorization 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 های object با فرمت JSON به صورت امن جابه‌جا کنند. این توکن‌ها در header هر درخواستی (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 limiting rate 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 صورت پیاده‌سازی flight single نیز این ابزار می‌تواند کمک کننده باشد) در اینجا مجاز به استفاده از هیچ کتابخانه یا ابزار کمکی در کد نیستید. بک‌اند باید بتواند ارتباط خود را با دیتابیس و cache خود مدیریت کند و از آن‌ها در جای مناسب استفاده کند (مثلاً 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.

قبلاً 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 ننید!). استفاده از اصولی مثل S.O.L.I.D یا App Twelve-Factor پیشنهاد می‌شود. معیار نمره‌دهی در اینجا این است که اساسی‌ترین 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 به سرویس‌ها خارجی 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 آیت‌ها را پیاده کنید و حداکثر ۱۰ درصد نمره کل تمرین را دریافت کنید و پیاده‌سازی گزینه‌های بیشتر نمره‌ای نخواهد داشت.

- استفاده از فریمورک‌هایی مثل react در سمت کلاینت [۳]

- خلاقیت در UI وب اپلیکیشن [۱]

- استفاده از روش singleflight در پیاده‌سازی کش [۲]

- خلاقیت در طراحی بک‌اند یا سرویس‌ها [۱]

- پیاده‌سازی cache توزیع شده و یا مجموعه‌ای از سرویس‌های cache جهت fault-tolerance یا افزایش زمان پاسخ‌دهی یا ... [۴]

- اضافه کردن احراز هویت و یا ارتباط امن (ssl/tls) در سرویس cache (به همراه پیاده سازی های لازم در بک اند) [۲]
- اضافه کردن قابلیت load-balancing (می توانید از nginx یا traefik یا envoy و یا ابزار مشابه دیگر استفاده کنید) [۳]
- استفاده از دیتابیس های NoSQL به جای دیتابیس رابطه ای [۲]

۵.۰ منابع مفید

- [practices best go](#)
- [Communication Microservice for Toolkit Modern A gRPC: to ntro](#)
- [Services Type-Safe and Efficient Developing Go: and gRPC](#)
- [books go](#)
- [S.O.L.I.D](#)
- [buffalo Go](#)
- [Gin Go](#)
- [Sequelize](#)
- [Parse](#)
- [typeorm](#)
- [App Twelve-Factor The](#)

زمان بندی ها و نحوه تحویل

ارایه توضیحات راجع به نحوه تحویل تمرین، محل بارگذاری و ددلاین ارسال.

سلامت باشید