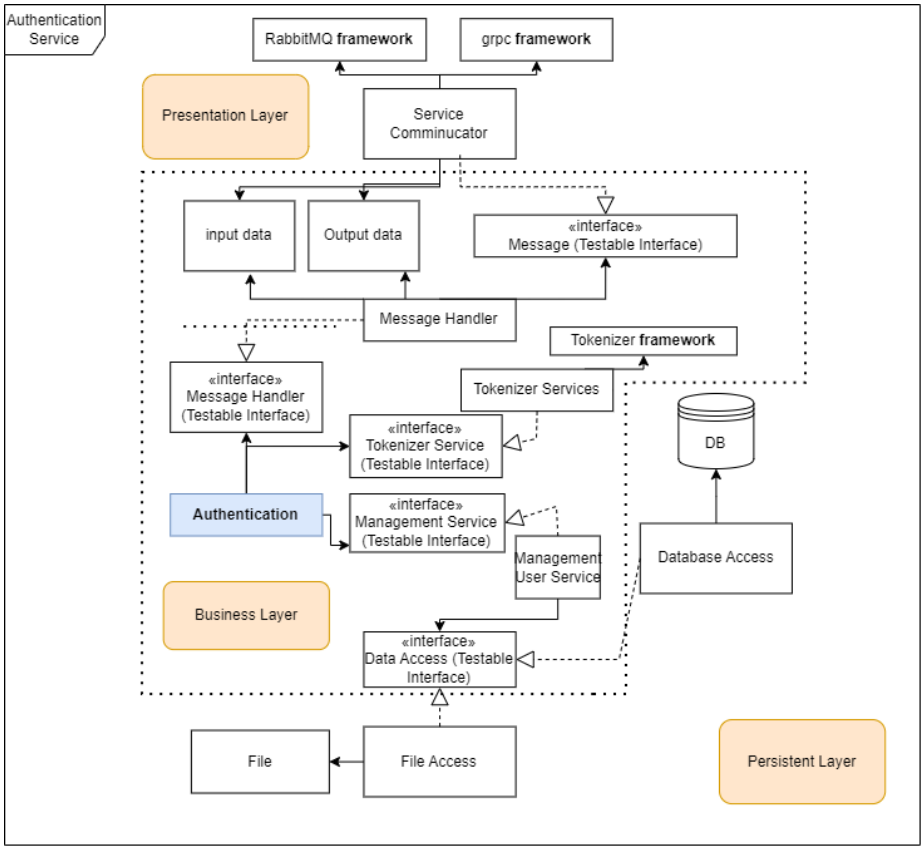
معماری نرم‌افزار - سرویس Authentication

سید حسین زراعتکار و نیما گمرکیان
دانشگاه خواجه نصیر الدین طوسی

۱۴ آذر ۱۴۰۴

۱.۰ میکروسرویس Authentication - Module View

در این بخش به بررسی معماری میکروسرویس Authentication خواهیم پرداخت. این میکروسرویس وظیفه احراز هویت کاربران و ادمین‌ها را به عهده دارد. پس از احراز هویت به هر کاربر یک توکن داده خواهد شد که دارای زمان انقضا هست. هر سرویس دیگر متناسب با توکن‌های کاربران له آنها سرویس می‌دهد. در ادامه ماژول‌ها و وابستگی‌شان را در قالب دیاگرام در شکل ۱ ارائه کردیم.



شکل ۱: دیاگرام وابستگی ماژول‌ها

این ماژول از طریق دو framework با نام‌های grpc و RabitMQ با دیگر سرویس‌ها و bff تعامل می‌کند. برای تعاملات بین سرویس به دلیل سرعت بالاتر از grpc و برای تعاملات با bff از RabitMQ استفاده کردیم. برای پیاده سازی متدها و مدیریت پیام‌های این دو سرویس یک ماژول پیاده سازی کردیم که بتواند کارهایی که فریموورک قرار است برای ما انجام دهد را مدیریت کند که این ماژول باید interface مربوط به message interface را پیاده سازی کند تا business logic ما بتواند بدون وابستگی به ماژول low level با بقیه ارتباط برقرار کند. در قسمت business logic ما یک ماژول به منظور هندل کردن رخدادهای دریافتی یا ارسال پیام‌ها پیاده سازی شده است به نام Message Handler این ماژول نیز به علت سطح پایین بودن باید برای مورد استفاده قرار گرفتن توسط Authentication که بالاترین سطح را دارد با استفاده از معکوس سازی وابستگی و استفاده از interface پیاده سازی شود. از طرفی دیگر ما برای افزودن کاربران(ادمین و کاربرمعمولی) ماژولی به نام Man-agement User Service داریم که عملیات های verification, registration, update را انجام می‌دهد. این ماژول نیز به دلیل نیاز به استفاده از دیتابیس با interface و معکوس سازی وابستگی با دیتابیس مورد نظر کار می‌کند. در آخر ماژول Tokenizer service با استفاده از کتابخانه‌ها و framework ها توکن‌ها را ایجاد کرده و آپدیت و انقضای توکن‌ها از این طریق مدیریت می‌شوند. برای انقضای توکن در واسط مربوط به Tokenizer ما یک متد برای ایونت انقضا داشته و در طرف مازول mangement یک هندلر تابع و در authentication این‌ها را

مدیریت می‌کنیم.

۱. یکی از نکات مهم این است که در سرویس‌های دیگر برای تعاملات باید از معماری تراکنش saga استفاده کنیم تا قبل از اعمال تراکنش احراز هویت بر اساس توکن کاربر در این سرویس بررسی شود.

۲. نکته دیگر این است که برای تست پذیری از قابلیت ذخیره سازی دادگان در فایل نیز انجام می‌شود و البته نیاز به رعایت نکاتی برای امنیت و کارایی برنامه در هنگام پیاده سازی دارد. (Testability: Abstract Data source)

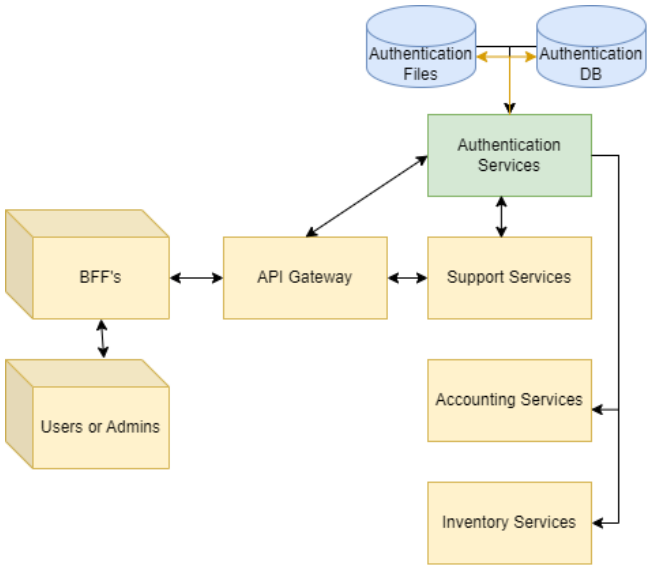
۳. همچنین به منظور تست پذیری بهتر زمان پیاده سازی واسط‌ها باید متدهایی به منظرو تست آن بخش پیاده سازی شود. (Testability: Specialize Interface)

۴. به دلیل استفاده از معماری میکروسرویس ما قابلیت تست پذیری بیشتری داشته‌ایم زیرا سرویس‌ها مجزا شده اند و separation of concern را رعایت کرده و وابستگی درو ماژولی افزایش و وابستگی بین ماژولی کاهش یافته است. (Testability: Limit structural complexity)

۵. بر اساس فریمورک‌هایی مانند jwt نیاز به احراز هویت توکن برای هر تراکنش نیست یعنی با این سرویس نیاز نیست که احراز هویت توکن را هربار چک کنیم ولی نیاز به verify داریم. بر اساس public key که به سرویس‌ها ارسال شده احراز هویت هر تراکنش انجام می‌شود.

۲.۰ میکروسرویس Authentication - Component & Connector View (C&C)

در این قسمت به بررسی نمای دیگر از معماری سرویس Authentication به نام Connection And Component View می‌پردازیم. نمای کلی ارتباطات این سرویس با سرویس‌های دیگر و BFF در شکل ۲ ارائه شده است. بر



شکل ۲: نمای C&C

اساس نیاز موجود هر کاربر با هر سطح دسترسی یا درخواست احراز هویت اولیه می‌دهد که در آن صورت مسیر از کاربر به BFF، API Gateway و در آخر به سرویس احراز هویت میرسد و اما در صورت درخواست سرویس دیگر مرحله آخر به سرویس مورد نظر رسیده و سرویس مورد نظر با کلید عمومی دریافت شده از سرویس Authentication میتواند تراکنش را احراز هویت کند. ضمننا، پس از تاریخ انقضای هر توکن، توکن نامعتبر در هر سیستم می‌باشد.