



دانشگاه صنعتی خواجه نصیرالدین طوسی

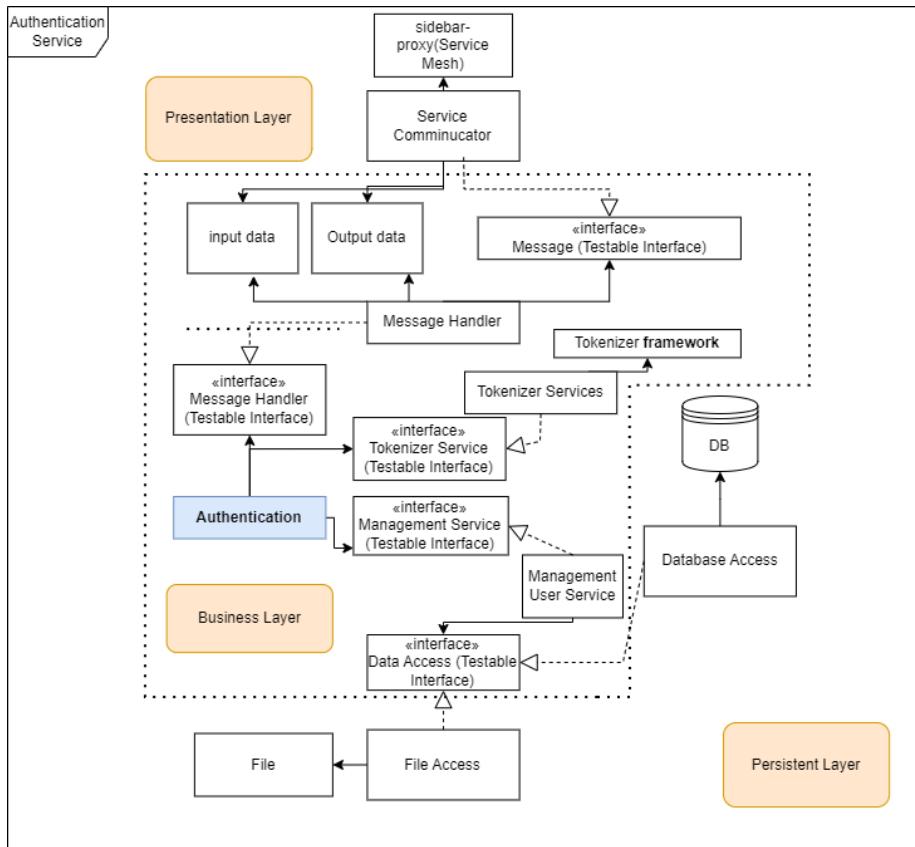
معماری نرم افزار - سرویس Authentication

سید حسین زراعتکار و نیما گمرکیان
دانشگاه خواجه نصیر الدین طوسی

۱۴۰۴ آذر ۲۱

Module View - Authentication ۱.۰ میکروسرویس

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



شکل ۱: دیاگرام وابستگی مازول‌ها

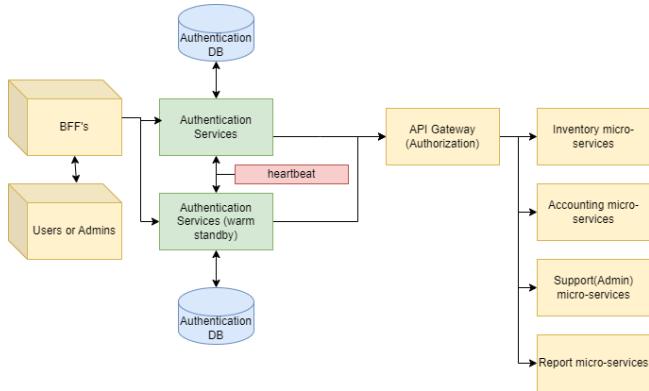
این مازول از طریق دو **bff** با نام‌های **grpc** و **RabbitMQ** تعامل می‌کند. این قسمت در مازولی مجزا به نام **sidebar-proxy** استفاده می‌شود و دلیل آن استفاده از **mesh service pattern** می‌باشد. این الگو سبب بهبود کارایی می‌شود و برای انجام کارهای شبکه و پیاده سازی موارد امنیتی وروودی و خروجی و رمزگذاری استفاده می‌شود. اگرچه با افزودن این لایه باعث مصرف بیشتر توان می‌شود و سربار دارد. برای تعاملات بین سرویس به دلیل سرعت بالاتر از **grpc** و برای تعاملات با **bff** از **RabbitMQ** استفاده کردیم. برای پیاده سازی متدها و مدیریت پیام‌های این دو سرویس یک مازول پیاده سازی کردیم که بتواند کارهایی که فریمورک قرار است برای ما انجام دهد را مدیریت کند که این مازول باید **interface** مربوط به **message interface** را پیاده سازی کند تا **business logic** ما بتواند بدون وابستگی به مازول **low level** ارتباط برقرار کند. در قسمت **Message Handler** این مازول نیز به علت سطح پایین بودن باید برای مورد استفاده پیاده سازی شده است به نام **Management User Service**. این مازول 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** **management User Service** که عملیات‌های **verification, registration, update** داریم را انجام می‌دهد. این مازول نیز به دلیل نیاز به استفاده از **database** و معکوس سازی وابستگی با **database** 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 واسطه مربوط به Tokenizer ما یک متدهای ایونت انقضا داشته و در طرف ماژول mangement یک هندرل تابع و در authentication 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 واسطه‌ها باید متدهایی به منظور 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 مازوی افزایش و وابستگی بین مازوی کاهش یافته است. **(Testability: Limit structural complexity)**
 ۵. بر اساس فرمول که این $jwtoken$ به اهراز هویت توکن برای هر تراکنش نیست یعنی با این سرویس نیاز نیست که اهراز هویت توکن را هر بار چک کنیم ولی نیاز به verify داریم. بر اساس public key که به سرویس‌ها ارسال شده اهراز هویت هر تراکنش انجام می‌شود.

۲۰۰ میکروسرویس - Authentication & Component & Connection View (C&C)

در این قسمت به بررسی نمای دیگر از معماری سرویس Authentication به نام Connection And Com-ponent View می‌پردازیم. نمای کلی ارتباطات این سرویس با سرویس‌های دیگر و BFF در شکل ۹۹ ارائه شده است.



شکل ۲:

بر اساس نیاز موجود هر کاربر با هر سطح دسترسی یا درخواست اهراز هویت اولیه می‌دهد که در آن صورت مسیر از کاربر به API Gateway و در آخر به سرویس اهراز 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 اهراز هویت کند. ضمناً، پس از تاریخ انقضای هر توکن، توکن نامعتبر در هر سیستم می‌باشد.

برای دسترسی پذیری سرویس Authentication از متد Heartbeat (warm standby) استفاده کردیم. به دلیل

تغییرات کم در کاربران چندوقت یکبار سرویس کمکی همگام سازی می‌شود. با خرابی یکی از سرویس‌ها دیگری سریعاً جایگزین می‌شود.