آزمایشگاه مهندسی نرم‌افزار

الهه خدایی ۹۶۱۱۰۲۰۴
علیرضا توکلی ۹۶۱۰۵۶۳۷

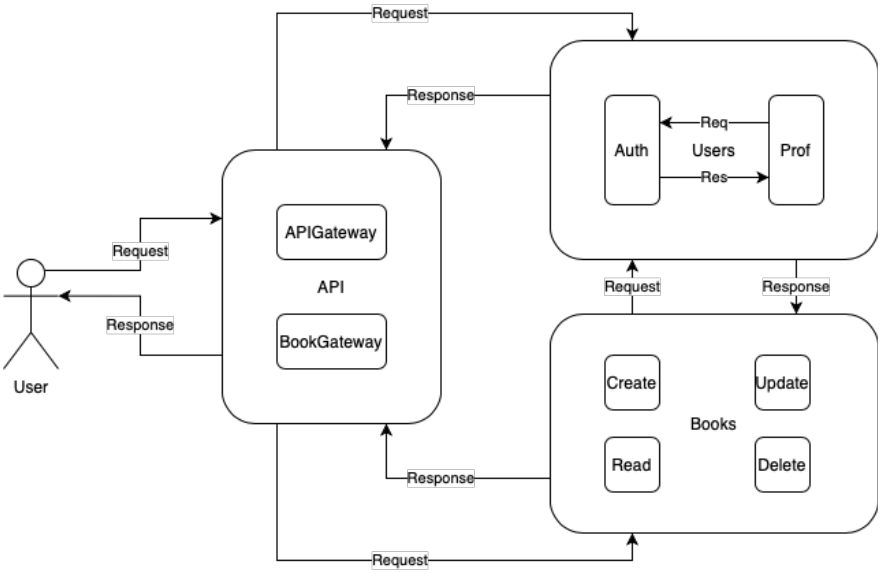
گزارش آزمایش پنجم

دستور کار.

در این آزمایش الگوی معماری Microservice کار شده و یک سامانه با استفاده از آن طراحی شده است. در این سامانه از زبان پایتون و جنگو استفاده کردیم. این پروژه در آدرس گیت زیر پیاده‌سازی شده است؛
https://github.com/sinatav/SElab5

معماری.

معماری سامانه را مطابق با اصول و قواعد توضیح داده شده در فیلم فیلم آموزشی پیاده می‌کنیم. معماری سامانه چیزی شبیه به شکل زیر خواهد بود؛



بخش‌های پروژه.

پروژه ۴.

در پروژه‌ی قبلی ما دو نوع کاربر ساختیم که شامل admin و client می‌شدند. این کاربران می‌توانستند در سامانه ثبت‌نام کنند، به صفحه خود وارد شوند، صفحه‌ی خود را ببینند و اطلاعات آن را به روزرسانی کنند. یک API برای آن ساخته شد که درخواست ها در آن مدیریت می‌شوند. و در این قسمت برنامه‌ریزی می‌کنیم که پس از گذشت چه زمانی درخواست‌ها اعتبار خود را از دست بدهند. برای این کار توکن جدا نوشته شد و شمارشگری برای اقدام‌های مناسب به آن اضافه شد.

```
initiate_values = {"register_count": 0, "login_counts": 0, "show_counts": 0, "update_counts": 0}
```

که برای مثال در ثبت‌نام اینگونه هندل می‌شود؛

```
if req_type == "register":
    try:
        return self.register(request.data)
    except:
        initiate_values['register_count'] += 1
        if self.is_request_timeout(initiate_values['register_count']):
            print("REQUEST_TIMEOUT")
```

در زمان مناسب ارتباط کاربر با سرور هم قطع می‌شود.

پروژه ۵.

در این آزمایش به قسمت‌های قبلی بخش‌هایی افزوده می‌شوند؛ برای مثال دایرکتوری Book که در آن مدل کتاب‌ها و API برای آن‌ها نوشته شده و اعمالی که می‌توان روی آن‌ها انجام شود نوشته شده است. این اعمال شامل موارد زیر می‌شوند؛

- ساخت کتاب جدید.

در این قسمت توکن متناسب با داده ست می‌شود و بررسی می‌شود که آیا کاربری که در حال انجام این فعالیت است از نوع admin است یا client. اگر admin نباشد اجازه‌ی دسترسی به آن داده نمی‌شود. در غیر این صورت زمان ساخت کتاب در سامانه ثبت شده و پارامترهای مختلف آن مقدار دهی می‌شوند و کتاب ذخیره می‌شود. مطابق زیر؛

```python
class Create(viewsets.ViewSet):
    def user_request_type(self, request):
        data = request.data
        token = data['token']
        admin = User.objects.get(token=token, isAdmin=True)
        if not admin:
            return Response("no user or not admin")
        admin.token_time = datetime.datetime.now()
        book = Book()
        book.title = data['title']
        book.author = data['author']
        book.category = data['category']
        book.save()
        return Response("book added")
```

- به روزرسانی کتاب

در این قسمت توکن متناسب با داده ست می‌شود و بررسی می‌شود که آیا کاربری که در حال انجام این فعالیت است از نوع admin است یا client. اگر admin نباشد اجازه‌ی دسترسی به آن داده نمی‌شود. در غیر این صورت زمان به روزرسانی کتاب در سامانه ثبت شده و کتاب درخواست شده جستجو می‌شود، در صورتی که کتاب با این مشخصات پیدا شود مطابق داده‌های داده شده، اطلاعات آن تغییر کرده و به روزرسانی می‌شود؛

```python
class Update(viewsets.ViewSet):
    def user_request_type(self, request):
        data = request.data
        token = data['token']
        admin = User.objects.get(token=token, isAdmin=True)
        if not admin:
            return Response("no user or not admin")
        admin.token_time = datetime.datetime.now()
        book_id = data['id']
        book = Book.objects.get(_id=book_id)
        if not book:
            return Response("no such book")
        else:
            book.title = data['title']
            book.author = data['author']
            book.category = data['category']
            book.save()
            return Response("book updated")
```

- خواندن کتاب

در این قسمت توکن متناسب با داده ست می‌شود و بررسی می‌شود که آیا کاربری که در حال انجام این فعالیت است از نوع admin است یا client. اگر admin نباشد اجازه‌ی دسترسی به آن داده نمی‌شود. در غیر این صورت زمان دسترسی ثبت می‌شود. سپس بررسی می‌کنیم که کتابی که درخواست داده پیدا می‌شود یا نه. اگر پیدا شد مشخصات کتاب را برمی‌گردانیم؛

```python
class Read(viewsets.ViewSet):
    def user_request_type(self, request):
        data = request.data
        token = data['token']
        admin = User.objects.get(token=token, isAdmin=True)
        if not admin:
            return Response("no user or not admin")
        admin.token_time = datetime.datetime.now()
        book_id = data['id']
        book = Book.objects.get(_id=book_id)
        if not book:
            return Response("no such book")
        else:
            return Response(
                "title : " + book.title + "\nauthor : " + book.author + "\ncategory : " + book.category)
```

- حذف کتاب

در این قسمت توکن متناسب با داده ست می‌شود و بررسی می‌شود که آیا کاربری که در حال انجام این فعالیت است از نوع admin است یا client. اگر admin نباشد اجازه‌ی دسترسی به آن داده نمی‌شود. در غیر این صورت زمان حذف کتاب در سامانه ثبت شده و کتاب درخواست شده جستجو می‌شود. در صورتی که کتاب با مشخصات داده شده پیدا شد این کتاب از سامانه حذف می‌گردد.

```python
class Delete(viewsets.ViewSet):
    def user_request_type(self, request):
        data = request.data
        token = data['token']
        admin = User.objects.get(token=token, isAdmin=True)
        if not admin:
            return Response("no user or not admin")
        admin.token_time = datetime.datetime.now()
        book_id = data['id']
        book = Book.objects.get(_id=book_id)
        if not book:
            return Response("not book")
        else:
            book.delete()
            return Response("deleted")
```

در تمام این قسمت‌ها فرض بر این است که یک آیدی یکتا به هر کتاب داده شده و از طریق دسترسی به آن می‌توانیم این اعمال را روی کتاب‌ها انجام دهیم.

- دیدن کتاب‌ها

این قسمت در SElab4.display.py پیاده‌سازی شده و در آن تمام کتاب‌ها به کاربر که نقش کلاینت داشته باشد نشان داده می‌شود. ابتدا چک می‌شود که کاربر مورد نظر وجود داشته باشد و سپس زمان دسترسی ثبت شده و کتاب ها نمایش داده می‌شوند. به صورت زیر؛

```python
class SeeBooks(viewsets.ViewSet):
    def booker(self):
        books = Book.objects.filter()
        result = []
        for b in books:
            result.append({
                "title": b.title,
                "author": b.author,
                "category": b.category
            })
        return result


    def user_request_type(self, request):
        data = request.data
        token = data['token']
        client = User.objects.get(token=token, isAdmin=False)
        if not client:
            return Response("no user or not client")

        client.token_generation_time = datetime.datetime.now()

        return Response(self.booker())
```

تضمین صحت.

برای این که مطمئن شویم که سامانه‌ی پیاده‌سازی شده کار می‌کند می‌توانیم قسمت‌های مختلف آن را بررسی کنیم. برای این کار لازم به ذکر است که باید توکن‌های خود را به سامانه بدهیم که شامل رشته‌های استرینگ رندوم خواهند بود. برای قسمت‌های مختلف داده‌ها به صورت زیر وارد می‌شوند؛

- ثبت‌نام

برای ثبت‌نام داده‌ها برابر با نوع درخواست، نام‌کاربری، رمز عبور، شماره‌ی موبایل، آدرس ایمیل خواهد بود. باید تعیین شود که این کاربر از نوع کلاینت است یا ادمین که این عمل توسط فیلد isAdmin مشخص می‌شود.

- ورود

نوع درخواست و نام کاربری و رمز عبور به عنوان داده‌های ورودی وارد سیستم می‌شوند و مانند قسمت قبل تعیین می‌شود که آیا این درخواست مربوط به ادمین است یا کلاینت.

- نشان دادن پروفایل

در این قسمت تنها نیاز است که نوع درخواست و توکنی که بصورت رندوم تولید شده پاس داده شود تا بتوانیم عملیات مورد نظر را انجام دهیم.

- به روزرسانی پروفایل

در این قسمت هم نوع درخواست و اطلاعات جدید مورد تغییر وارد شده و توسط سامانه تغییر می‌یابند. یک توکن رندوم هم مانند قسمت قبل تولید شده و پاس داده می‌شود.

- دیدن کتاب‌ها توسط کلاینت

در این قسمت توکن رندوم تولید شده پاس داده می‌شود و کتاب‌ها به کاربر نمایش داده می‌شوند.

- ساختن کتاب

برای این قسمت نوع درخواست، توکن رندوم ساخته شده، عنوان، نویسنده و دسته‌بندی کتاب وارد شده و کتاب جدید ساخته می‌شود.

- به روزرسانی کتاب

در این قسمت علاوه بر تمام موارد قسمت قبل، آیدی کتابی که می‌خواهیم آن را به روزرسانی کنیم هم باید وارد شود تا آن کتاب را بیابیم. این آیدی توسط دیتابیس بصورت خودکار ساخته می‌شود و برای همین نیازی نیست که هنگام ساختن یک کتاب آن را به عنوان داده‌ی ورودی وارد کنیم.

- خواندن کتاب

در این قسمت نوع درخواست و توکن رندوم به همراه آیدی کتاب مورد نظر وارد سامانه شده و اطلاعات مورد نیاز کتاب نمایش داده می‌شوند.

- حذف کتاب

در پایان، برای حذف کتاب مانند قسمت قبل داده‌های نوع درخواست، توکن رندوم و آیدی کتاب مورد نظر را وارد می‌کنیم تا این کتاب از سامانه حذف شود.

حالا چند مثال از کارکرد سامانه می‌زنیم تا متوجه صحت درستی آن بشویم. در اینجا دو کاربر داریم که به صورت زیر هستند؛

```
"username": "sina",
"password": "123",
"email": "stavakkoli1999@gmail.com",
"mobile": "09137262768",
"isAdmin": True
```

```
"username": "elahe",
"password": "321",
"email": "elahekhodaie@gmail.com",
"mobile": "09123456789",
"isAdmin": False
```

حالا با استفاده از این دو کاربر که یکی ادمین و دیگری کلاینت است اقدام به ارزیابی سامانه می‌کنیم؛
برای مثال ساختن یک کتاب اطلاعات زیر را وارد می‌کنیم

```
“type of action": "create",
"token" : “anyrandomtokenwilldo",
"title": "Odyssey",
"author": "Homer",
"category": "Epic"
```

در صورتی که اقدام به ساخت این کتاب با کاربر sina بکنیم، با پیام "book added" مواجه خواهیم شد و در صورتی که با کاربر elahe این کار را بکنیم با پیغام "no user or not admin" چون این کاربر از نوع کلاینت می‌باشد.

آدرس مخازن.

مخزن‌های انجام این آزمایش به صورت زیر می‌باشند؛ خود مخزن SElab5 شامل دو فایل است که در هر کدام یکی از موارد توضیح داده شده پیاده‌سازی شده‌اند.

https://github.com/elahekhodaie/MicroService-architecture
https://github.com/sinatav/SElab5