تمرین دوم برنامه‌نویسی وب

استاد درس: امید جعفری‌نژاد
طراحی و ویرایش: گروه دستیاران آموزشی
تاریخ تحویل: ۹ بهمن ۱۴۰۱

---

۱ مقدمه

در این تمرین قصد داریم تا بک‌اند وبسایت یک هواپیمایی را که در آن فروش بلیط انجام می‌شود پیاده‌سازی کنیم. در این وب‌سایت کاربر باید بتواند مبدا، مقصد، تاریخ سفر و تعداد مسافران را مشخص کند و پیشنهادهای مرتبط با جستوجوی خود را دریافت کند و سپس از بین پیشنهادهای موجود گزینه مورد نظر خود را انتخاب و خرید خود را نهایی کند. در حین نهایی سازی پرداخت، کاربر باید به درگاه بانکی برای پرداخت هدایت شود و پس از پرداخت به وب‌سایت اصلی برگردد. همچنین برای انجام این امور نیاز است تا کاربر بتواند حساب کاربری داشته باشد و احراز هویت انجام دهد.

در ادامه این مستند به بررسی دقیق نیازمندی‌ها و مواردی که باید پیاده‌سازی شوند می‌پردازیم. در این تمرین تلاش شده است تا تمام منابع مورد نیاز در اختیار شما قرار گیرد تا بتوانید تمرکز خود را بر روی مواد درسی حفظ کنید. لذا جداول و دیدهای[۱] مورد نیاز و همچنین مجموعه دادگان جعلی در اختیار شما قرار گرفته است که در ادامه به توضیح آنها نیز پرداخته خواهد شد.

۲ منابع مفید

پیش از شروع توصیه می‌شود که وب‌سایت چند مورد از شرکت‌های هواپیمایی معروف را برای آشنایی با کلیت کار بررسی کنید.

- هواپیمایی امارات
- هواپیمایی قطر

در این تمرین نیاز است که از پایگاه‌داده PostgreSQL استفاده کنید. لذا آشنایی اولیه با پایگاه‌داده‌های رابطه‌ای و همچنین نحوه کار با آن مورد نیاز است. توصیه می‌شود که پیش از شروع تمرین مستندات زیر را مطالعه شود.

- مقدمه‌ای بر PostgreSQL
- دوره یادگیری PostgreSQL

باید در جاهایی که نیاز است، از تکنیک‌های Caching استفاده کنید. برای آشنایی با Redis می‌توانید از مستندات زیر استفاده کنید.

- مقدمه‌ای بر Redis
- شروع کار با Redis

در این تمرین از معماری مایکروسرویس[۲] استفاده می‌کنیم که می‌توانید از طریق لینک‌های زیر مطالبی را در مورد آن مطالعه کنید.

- The What, Why, and How of a Microservices Architecture
- Microservices Architecture From A to Z

برای احراز هویت از روش Bearer Token استفاده می‌کنیم. در این روش در زمانی کاربر نام‌کاربری و رمز عبور خود را وارد می‌کند، یک رشته تصادفی تحت عنوان Token به کاربر برگردانده می‌شود که کاربر آن را همراه با درخواست‌هایش به سمت سرور ارسال می‌کند و سرور با تطبیق آن رشته با پایگاه داده یا رمزگشایی آن، کاربر را احراز هویت می‌کند. یک کتابخانه معروف در این حوزه JWT است. از منابع زیر می‌توانید برای آشنایی بیشتر با JWT استفاده کنید.

- Understanding the concept of JWT — JSON Web Tokens

---

[۱] View
[۲] Microservice Architecture

- How JSON Web Token(JWT) authentication works?

در این برنامه از پروتکل معروف HTTP برای پیاده‌سازی API استفاده می‌کنیم. اگر آشنایی کافی با این پروتکل ندارید، از منابع زیر می‌توانید برای آشنایی بیشتر استفاده کنید.

- What is HTTP?
- HTTP Tutorial

در نهایت در مورد مسائل امنیتی که جلوتر با آنها مواجه خواهد شد، توصیه می‌کنیم که موارد زیر را مطالعه بفرمایید.

- OAuth۲
- TLS

۳ شرح نیازمندی‌ها

در این تمرین از معماری مایکروسرویس استفاده می‌کنیم. برنامه ما از دو مایکروسرویس Auth و Ticket تشکیل شده است. از مزایای این معماری این است که پیاده‌سازی را می‌توان به صورت جداگانه برای سرویس‌های مختلف انجام داد و آنها را به صورت جداگانه مستقر[۳] کرد.

۱.۳ Auth Service

سرویس احراز هویت مسئول بخش ثبت نام، ورود و منطق‌های مربوط به کاربران وبسایت است. این سرویس از آنجایی که تقریبا در هر سیستمی از بخش‌های کلیدی است، نیاز است تا سرعت بالایی داشته باشد بنابرین باید با استفاده از زبان Go پیاده سازی شود. همچنین لازم است تا از تکنیک‌های Caching نیز برای بهبود کارایی این سیستم استفاده شود. جداول پایگاه‌داده این سیستم در PostgreSQL پیاده‌سازی شده است و در مخزن گیت‌هاب ضمیمه شده در اختیار شما قرار گرفته است که می‌توانید از آن استفاده کنید. هر چند شما مختارید تا از پایگاه داده مورد علاقه خود برای این منظور استفاده کنید.

۱.۱.۳ موجودیت‌ها

در این سیستم دو جدول اصلی داریم. مورد اول و اصلی‌تر آن جدول کاربران است که به صورت زیر تعریف می‌شود. هر رکورد این جدول، یک کاربر است. دقت بفرمایید که user_id به صورت خودکار به عنوان یک عدد صحیح به هر کاربر تخصیص داده می‌شود. همچنین هر کاربر باید یک ایمیل و شماره تلفن متمایز داشته باشد. جنسیت کاربر نیز باید با یک تک حرف F و M مشخص شود.

```
CREATE TABLE IF NOT EXISTS user_account
(
    user_id         SERIAL PRIMARY KEY,
    email           VARCHAR UNIQUE NOT NULL,
    phone_number    VARCHAR UNIQUE NOT NULL,
    gender          VARCHAR(1),
    first_name      VARCHAR,
    last_name       VARCHAR,
    password_hash VARCHAR
);
```

جدول دیگر در این پایگاه داده مربوط به Token های منقضی شده است. که در ادامه در مورد هر یک از آنها توضیح خواهیم داد.

```
CREATE TABLE IF NOT EXISTS unauthorized_token (
    user_id     INTEGER REFERENCES user_account ON DELETE CASCADE ON UPDATE CASCADE,
    token       VARCHAR,
    expiration TIMESTAMP
);
```

[۳] Deploy

۲.۱.۳ Endpoints

در این سیستم شما باید اندپوینت‌های زیر را پیاده‌سازی کنید.

۱. Sign up: این اندپوینت باید تمام اطلاعات کاربر را دریافت کند و پس از ساخت رکورد در جداول پایگاه داده، کاربر را وارد سیستم کند.

۲. Sign in: این اندپوینت باید با دریافت ایمیل یا شماره تلفن و رمز عبور کاربر، کاربر را وارد سیستم می‌کند. منظور از ورود به سیستم این است که یک Token به کاربر نسبت داده می‌شود.

۳. Sign out: این اندپوینت هیچ ورودی‌ای دریافت نمی‌کند. صرفا با فراخوانی این اندوینت کاربری که داخل سیستم است باید از سیستم‌خارج شود. در ادامه در بخش ملاحظات پیاده‌سازی در مورد سازوکار این فرآیند توضیحات تکمیلی خواهیم داد.

۴. User Info: این اندپوینت توسط سرویس‌های دیگر برای احراز هویت کاربران استفاده می‌شود. سیستم‌های دیگر می‌توانند توکن دریافتی را به این اندپوینت ارسال کنند و اطلاعات کاربر را در جواب دریافت کنند. توجه کنید که ممکن است توکن منقضی شده باشد و در این صورت باید پاسخ مناسب برگردانده شود.

۳.۱.۳ ملاحظات پیاده‌سازی

۱. دقت بفرمایید که در تمام پیاده‌سازی‌های فوق، اجرای Validation[۴] مناسب برای بررسی ورودی‌های دریافتی الزامی است.

۲. اندپوینت‌ها باید پاسخ خود را با کد وضعیت مناسب[۵] برگردانند.

۳. شما باید رمز عبور را به صورت Hash شده در پایگاه داده بنویسید.

۴. شما باید برای تولید Token از JWT استفاده کنید. با استفاده از JWT می‌توانید مواردی از این دست که این توکن مربوط به چه کسی است و تاریخ انقضای آن تا چه زمانی است را درون خود توکن رمزنگاری کنید و در زمانی که می‌خواهید صحت توکن را بررسی کنید، تنها کافی است که آن توکن را رمزگشایی کنید. نیازی به ثبت توکن در پایگاه داده نخواهد بود.

۵. زمانی که یک کاربر از سیستم خارج می‌شود،همچنان یک توکن در دست خود دارد. چون این توکن در پایگاه داده ثبت نشده است و خود آن درون خود تاریخ انقضا خود را دارد، همچنان معتبر خواهد بود. شما باید توکن‌هایی که کابر در زمان خروج از سیستم غیر معتبر می‌کند را در جدول unathorized_token ثبت کنید تا دیگر ورود با آنها به سیستم ممکن نباشد.

۶. هر بار که کاربر درخواست میزند، باید بررسی شود که آیا توکن وی در جدول توکن‌های منقضی شده در پایگاه داده قرار دارد یا خیر. برای کم شدن سربار این کار، از یک Cache استفاده میکنیم تا سربار زمانی این کار کم شود. می‌توانید برای این Caching از سیستم‌هایی مانند Redis استفاده کنید. دقت بفرمایید که منطق Caching شما باید کاملا سازگار با نیازهای سیستم باشد و هیچگاه جواب اشتباه برنگرداند. منظور از جواب اشتباه این است که مثلا زمانی که یک توکن منقضی شده آن را به عنوان توکن معتبر تشخیص دهد و یا برعکس.

۷. رمزنگاری کردن درخواست‌ها برای حفظ محرمانگی آنها در شبکه با استفاده از TLS یا SSL امتیاز مثبت اضافی به همراه خواهد داشت.

۸. در صورت ارتباط دیگر سرویس‌ها با این سرویس با gRPC پیاده‌سازی شود، امتیاز مثبت اضافی به همراه خواهد داشت.

۹. در صورتی که از سیستم OAuth۲ که دو توکن به کاربر اختصاص می‌دهد برای پیاده‌سازی توکن استفاده کنید. امتیاز مثبت اضافه به همراه خواهد داشت.

۴.۱.۳ OAuth2

این سرویس به صورت امتیازی می‌تواند پروتکل OAuth۲ را پیاده سازی کند. در این پروتکل هنگامی که کاربر نام‌کاربری و رمزعبور خود را می‌دهد به آن دو توکن به نام‌های AccessToken و RefreshToken داده می‌شود.

Access Token

[۴] به این معنی که صحت ورودی‌ها چک شود. به عنوان مثلا نشانی ایمیل باید فرمت خاصی داشته باشد و طول آن از مقدار خاصی بیشتر نشود.
[۵] Status Code

هر توکن مدت زمان معلومی معتبر است (مثلا ۳۰ دقیقه). کاربر برای احراز هویت‌اش را باید به وسیله AccessToken انجام بدهد. یعنی برای مثال کاربر لاگین می‌کند و AccessToken را می‌گیرد. حالا هنگامی که کاربر می‌خواهد درخواستی به سرویس‌های دیگر بزند باید این توکن را نیز همراه درخواست بفرستد و سرویس تیکت این توکن را به سرویس احرازهویت می‌دهد و سرویس احرازهویت بررسی می‌کند آیا توکن معتبر است یا خیر و در صورت نیاز می‌تواند اطلاعات کاربر را نیز بدهد.

علت وجود AccessToken این است که کاربر برای هر درخواست خود مجبور نباشد نام‌کاربری و رمزعبور را همراه درخواست بفرستد. این‌طوری احتمال leak شدن اطلاعات فرد کمتر می‌شود.

علت اینکه AccessToken تا مدت زمان مشخصی معتبر است نیز این است که اگر به هر دلیلی AccessToken دست فرد دیگری افتاد مدت زمان محدودی دسترسی به اکانت فرد داشته باشد.

Refresh Token

همانطور که گفته شد AccessToken تا مدت زمان محدودی معتبر است. اگر کاربر بخواهد دوباره یک AccessToken جدید بگیر باید با استفاده از RefreshToken این کار را انجام بدهد.

دقت کنید که خود RefreshToken نیز تا زمان مشخصی اعتبار دارد ولی هربار که کاربر درخواست توکن جدید کند مدت زمان باقی مانده‌ی RefreshToken تمدید می‌شود.

۲.۳ Ticket Service

این سیستم مسئولیت منطق‌های مربوط به بحث فروش بلیط را بر عهده خواهد داشت. پیاده‌سازی این سیستم را باید به وسیله NodeJS انجام دهید. همچنین مشابه سرویس قبل جداول پایگاه داده مربوط به این سیستم نیز در اختیار شما قرار گرفته است و می‌توانید از آنها استفاده کنید.

۱.۲.۳ موجودیت‌ها

در این سیستم ما مجموعه دادگاه جعلی در نظر گرفته‌ایم تا شما درگیر وارد کردن داده به سیستم نشود. این دادگان در قالب جداول CSV هستند که شما می‌توانید در جداول SQL آنها را وارد کنید. همچنین دیدهای مورد نیاز نیز طراحی شده‌اند تا درگیر هایQuery پیچیده در SQL نشوید و بتوانید با Query های ساده اطلاعات مورد نیاز را بازیابی کنید. در واقع با اینکه ساختار جداول و دیدهای این سرویس پیچیده است، ولی شما تنها با تعداد اندکی از آنها درگیر خواهید بود.

دید زیر تمام فرودگاه‌ها را در بر دارد. همچنین توجه کنید که ممکن است که یک شهر چند فرودگاه داشته باشد، گزینه ALL به این معنی است که تمام فرودگاه‌های آن شهر مورد قبول است. دقت بفرمایید که هر فرودگاه یک کد سه حرفی تحت عنوان کد یاتا دارد که یکتا است. به عنوان مثال کد فرودگاه بین‌المللی جان اف. کندی نیویورک JFK و کد فرودگاه بین‌المللی مهرآباد THR است.

```sql
CREATE VIEW origin_destination AS
SELECT country_name as county,
       city_name    as city,
       airport_name as airport,
       iata_code    as iata
FROM airport
UNION
SELECT country_name   as county,
       city_name      as city,
       'All airports' as airport,
       'ALL'          as iata
FROM city;
```

جدول زیر مربوط به خریدهای کاربر است. در این جدول همچنین باید اطلاعات تراکنش بانکی منجر به خرید نیز ثبت شود. در ادامه در مورد تراکنش بانکی صحبت خواهیم کرد.

```sql
CREATE TABLE IF NOT EXISTS purchase
(
    corresponding_user_id INTEGER,
    title                 VARCHAR,
    first_name            VARCHAR,
```

```
    last_name           VARCHAR,
    flight_serial       INTEGER REFERENCES flight ON DELETE RESTRICT ON UPDATE RESTRICT,
    offer_price         INTEGER,
    offer_class         VARCHAR
    transaction_id      INTEGER,
    transaction_result  INTEGER
);
```

دید زیر مربوط به تمام پروازهای موجود است. نیازی به فهم این دید نیست، آشنایی با ستون‌های آن برای شما کفایت می‌کند. به سادگی با دستورات ساده SELECT می‌توانید کوئری‌های مورد نیاز سیستم را از این دید دریافت کنید.

```
CREATE VIEW available_offers AS
SELECT flight.flight_id                                                    AS flight_id,
       flight.origin                                                       AS origin,
       flight.destination                                                  AS destination,
       flight.departure_utc::TIMESTAMP WITH TIME ZONE AT TIME ZONE
       (SELECT timezone_name
        FROM airport_timezone
        WHERE airport_timezone.iata_code = flight.origin)                  AS departure_local_time,
       (flight.departure_utc + flight.duration)::TIMESTAMP WITH TIME ZONE AT TIME ZONE
       (SELECT timezone_name
        FROM airport_timezone
        WHERE airport_timezone.iata_code = flight.destination)             AS arrival_local_time,
       flight.duration                                                     AS duration,
       flight.y_price                                                      AS y_price,
       flight.j_price                                                      AS j_price,
       flight.f_price                                                      AS f_price,
       ((SELECT y_class_capacity FROM aircraft_view WHERE aircraft_view.registration = flight.
    aircraft) -
        (SELECT COUNT(*)
         FROM purchase
         WHERE purchase.flight_serial = flight.flight_serial
           AND offer_class = 'Y'))                                         AS y_class_free_capacity,
       ((SELECT j_class_capacity FROM aircraft_view WHERE aircraft_view.registration = flight.
    aircraft) -
        (SELECT COUNT(*)
         FROM purchase
         WHERE purchase.flight_serial = flight.flight_serial
           AND offer_class = 'J'))
                                                                           AS j_class_free_capacity,
       ((SELECT f_class_capacity FROM aircraft_view WHERE aircraft_view.registration = flight.
    aircraft) -
        (SELECT COUNT(*)
         FROM purchase
         WHERE purchase.flight_serial = flight.flight_serial
           AND offer_class = 'F'))                                         AS f_class_free_capacity,
       aircraft_view.aircraft_type                                         AS equipment
FROM flight
         JOIN aircraft_view on aircraft_view.registration = flight.aircraft;
```

۲.۲.۳ اندپوینت‌ها

در این سیستم طراحی ساختار اندپوینت‌ها بر عهده خودتان است. شما باید این اندپوینت‌ها را به صورتی پیاده‌سازی کنید که پاسخ‌گو نیازهای سیستم باشد. الزامی است که در پیاده‌سازی اندپوینت‌ها از Method و StatusCode مناسب استفاده کنید. همچنین اعتبارسنجی داده‌ها برای

بررسی صحت آنها نیز الزامی خواهد بود. تعداد اندپوینت‌های مورد نیاز برای پیاده‌سازی کم است ولی ساختار آنها با توجه به صلاح دید خودتان می‌تواند متفاوت باشد.

۴ درگاه بانکی

در این تمرین شما باید اتصال به یک درگاه بانکی فرضی را پیاده‌سازی کنید. یک درگاه بانکی ساده توسط دستیاران آموزشی پیاده‌سازی شده است که شما می‌توانید از طریق مخزن ضمیمه شده به آن دسترسی داشته باشید. کار با بانک جعلی پیاده‌سازی شده بسیار ساده است. دستورالعمل راه‌اندازی بانک در README مخزن مربوطه نوشته شده است.

اگر در مرورگر خود مسیر را وارد نمایید. با یک UI ساده از این بانک مواجه خواهید شد که می‌توانید از طریق آن تمام تراکنش‌های ثبت شده در سیستم را مشاهده کنید و یا یک تراکنش جدید در سیستم بسازید. همچنین وضعیت تراکنش‌های قبلی را نیز از این طریق می‌توانید مشاهده کنید.

این سیستم بسیار ساده است. یک Endpoint در این سیستم وجود دارد که شما باید تراکنش را به وسیله آن در بانک تعریف کنید.

POST /transaction

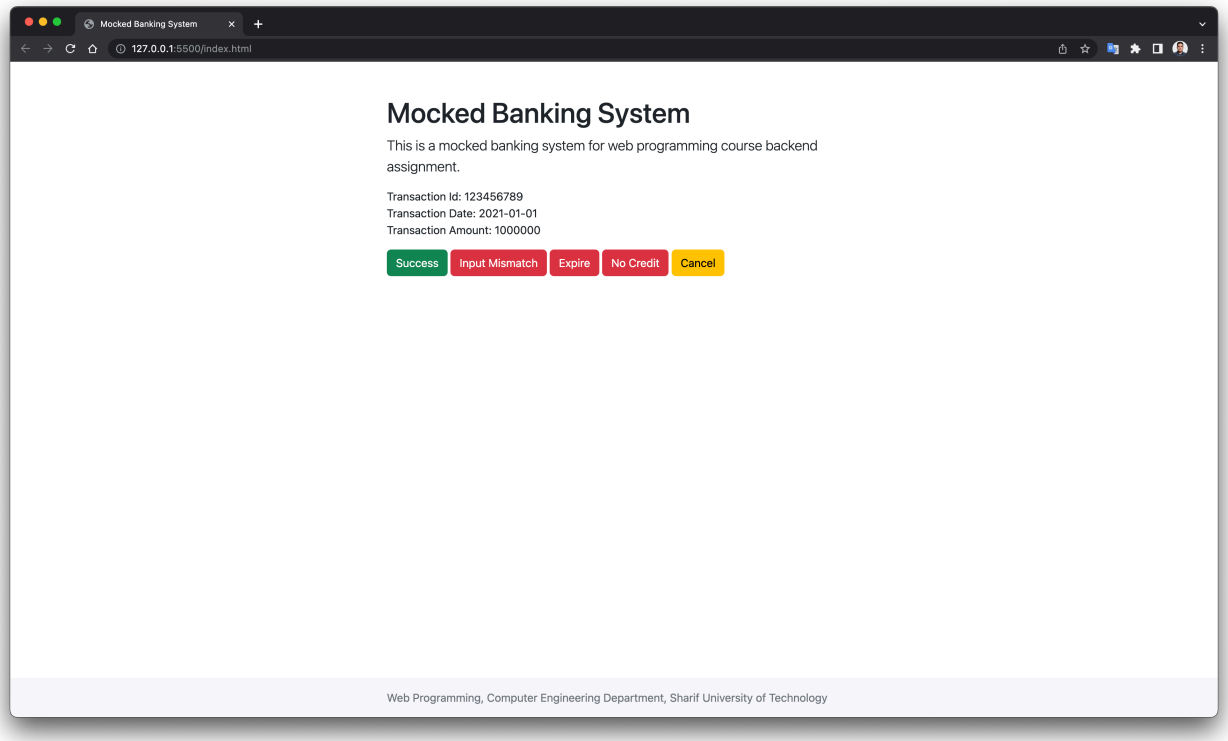
در تعریف تراکنش نیاز است که شما موارد زیر را در قالب FormData در بدنه درخواست به سیستم ارسال کنید.

کلید	عنوان	توضیح
amount	مبلغ تراکنش	
receipt_id	شناسه پذیرنده	یک عدد دلخواه را به عنوان شناسه وبسایت خودتان در نظر بگیرید.
callback	محل بازگشت	پس از اتمام تراکنش به این مکان redirect خواهید شد.

در پاسخ به این درخواست یک شناسه به شما برگردانده خواهد شد. شما در وبسایت خود باید کاربر متقاضی پرداخت را به مسیر

GET /payment/<transaction_id>

هدایت کنید. کاربر با یک چنین صفحه‌ای مواجه خواهد شد.



پس از انجام تراکنش کاربر به مسیر

GET /callback/<result>

هدایت خواهد شد. در متغیر result نتیجه تراکنش نوشته شده است. این نتیجه می‌تواند مقادیر ۱ الی ۵ را به خود بگیرد. هر یک از این مقادیر متناظر با یک حالت تراکنش است.

۱. Success

۲. Input Mismatch

۳. Expire

۴. No Credit

۵. Cancel

شما پس از هدایت شدن کاربر به وبسایت، باید پیغام متناسب با هر یک از نتایج به وی نمایش دهید.

۵ پرسش و پاسخ

در حین حل تمرین اگر به ابهامی برخورد کردید، می‌توانید در گروه درس آن را با دستیاران آموزشی مطرح کنید. دستیاران آموزشی پاسخگوی سوالات شما خواهند بود.

۶ ارتباط با تمرین قبلی

شما باید تمام Endpoint ها را به نحوی به تمرین قبلی وصل کنید تا یک وبسایت کامل بر روی سیستم لوکال خودتان داشته باشید.

۷ یک توصیه مهم

ابتدا سعی کنید Backend را پیاده‌سازی و تست کنید و پس از آن عملیات وصل کردن به Front را انجام دهید. برای تست کردن API های پیاده‌سازی شده می‌توانید از Postman استفاده کنید.

۸ نحوه تحویل

جزئیات تمرین‌های این درس معمولا باعث می‌شوند تا تمرین‌ها بیش از تخمین اولیه شما زمان لازم داشته باشند. توصیه می‌کنیم تا از موکول کردن تمرین به روزهای آخر پرهیز کنید.

یک مخزن گیت‌هاب در سازمان درس بسازید و تمام کدهای خود را در آن بارگذاری کنید. میزان مشارکت هر یک از اعضای تیم با توجه به این مخزن و commit های آن سنجیده خواهد شد.

در نهایت در یک ویدئو کنفرانس شما تمرین خود را ارائه می‌دهید و نمره‌دهی نهایی انجام می‌شود.

۹ مخزن‌های مربوطه

- مخزن بانک GitHub
- مخزن پایگاه داده و دادگان جعلی GitHub

سلامت باشید