Auth & Mocks

What’s an auth?!🤔

Motivation

What lead to our current situation and need for deep infra changes

Motivations

- **Decoupling** The code shouldn’t known about Auth0
- **Independence** Avoid vendor lock-in
- **Flexibility** Allow changing Auth provider "on the fly"
- **Design Ownership** We should decide how our data is built, not a third party provider
- **Testing *(future)*** We should be able to test our code without needing to mock the Auth0 SDK

Implementation (Frontend)

Today, our code imports directly from the `auth0` library. This means that our code is tightly coupled to Auth0.

Making changes to the auth system is difficult and risky. What if we want to change to a different provider? What if we want to change the way we handle tokens?

```
import { useUser } from '@/common/hooks/use-user';

///  ... more code ...

function NavBar() {
  const { user } = useUser();

  return (
  <div>
    <Profile user={user} /> // This component takes a user and displays their name
  </div>
  )
}
```

Implementation (Frontend)

Now that we saw how our components can stay unaware of the provenance of the user object, let's look at the implementation of `useUser` and its `Context` provider.

```
const userFetcher = async (url: string) => {
  const response = await fetch(url);
  return response.json();
};

async function fetchAuth0User(): Promise<IUser> {
  const user = await userFetcher('/api/auth/me');
  return user as IUser;
}

function fetchMockUser(): IUser {
  return {
      name: 'Test User',
      email: 'test@test.com',
      org_id: 'test_org_id',
    };
}
```

Implementation (Communication Frontend-Backend)

```
// Since our data is currently using Auth0 properties,
// we don't have a choice but to follow the same structure for now
const createMockJWT = () ⇒ {
  const payload = {
    name: 'Local User',
    email: 'local@test.com',
    org_id: 'org_local',
    ['https://www.apexsec.ai/user']: {
      email: 'local@test.com',
    },
    ['https://www.apexsec.ai/roles']: ['apex_admin', 'apex_user'],
  };

  return jwt.sign(payload, 'dev-secret', { expiresIn: '1h' });
};

export const getMockAccessToken = () ⇒ ({
  accessToken: createMockJWT(),
});
```

Implementation (Backend)

The backend is also affected by the changes. We dynamically initialize a "simpler" JWT strategy when we use mocks.

```
export class MockJwtStrategy extends PassportStrategy(Strategy, 'jwt') {
  constructor() {
    super({
      jwtFromRequest: ExtractJwt.fromAuthHeaderAsBearerToken(),
      secretOrKey: 'dev-secret',
      ignoreExpiration: true, // For mock purposes, ignore token expiration
    });
  }
}
```

Open Issues

Next steps and potential improvements

Open Issues

- **Duplication** Chat & Console have common components that are duplicated
- **Data Design** Our data has been designed by Auth0 format for now. How to improve this and make it provider agnostic?

Thank you✨