Qwerty

Ingeniería de Software

Miguel Yurivilca - Julio 2023

Frameworks

Problemática 😟

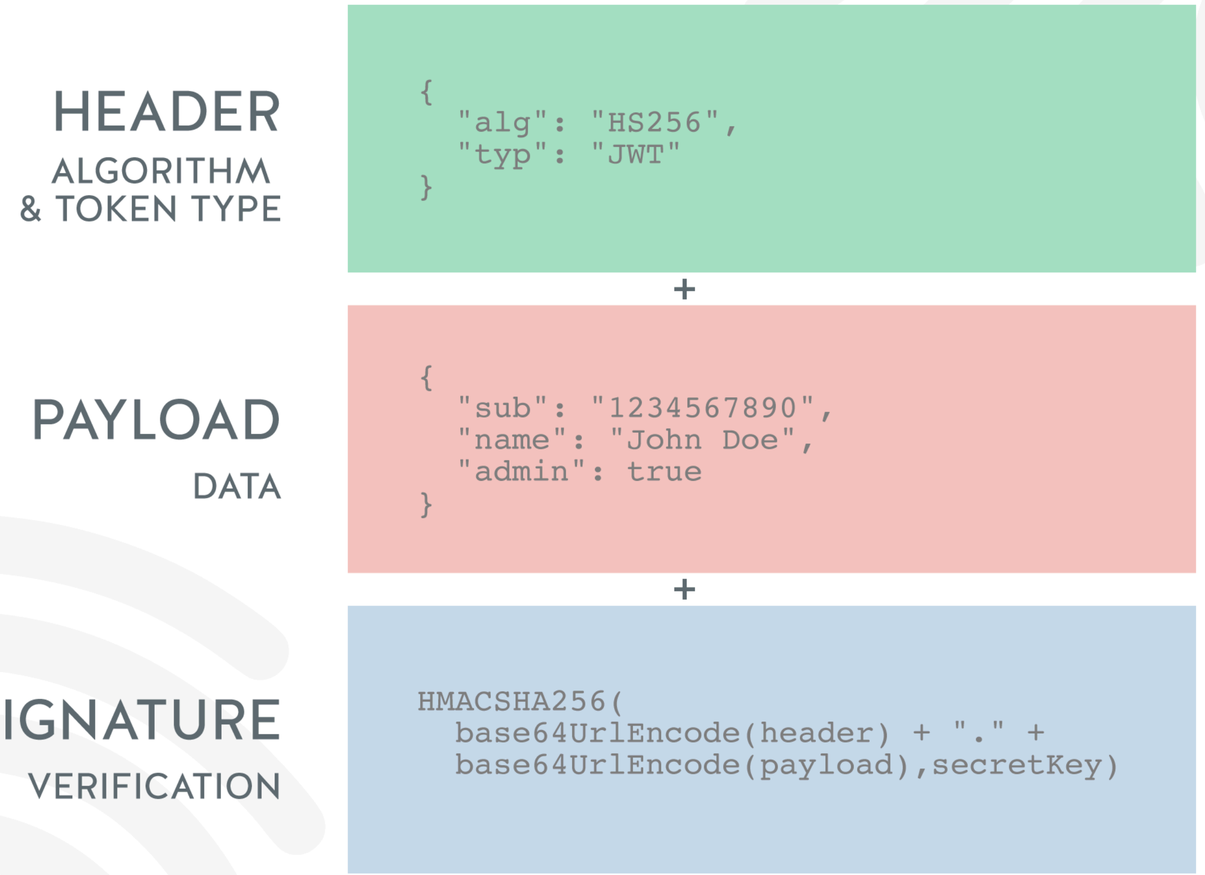
Problemática

- Aplicación sin seguridad
- Nuevos cambio pueden traer errores

Técnicas 

Técnicas

- JWT
- Testing & Code coverage
- ~~GraphQL~~ (hubieron problemas) 😢

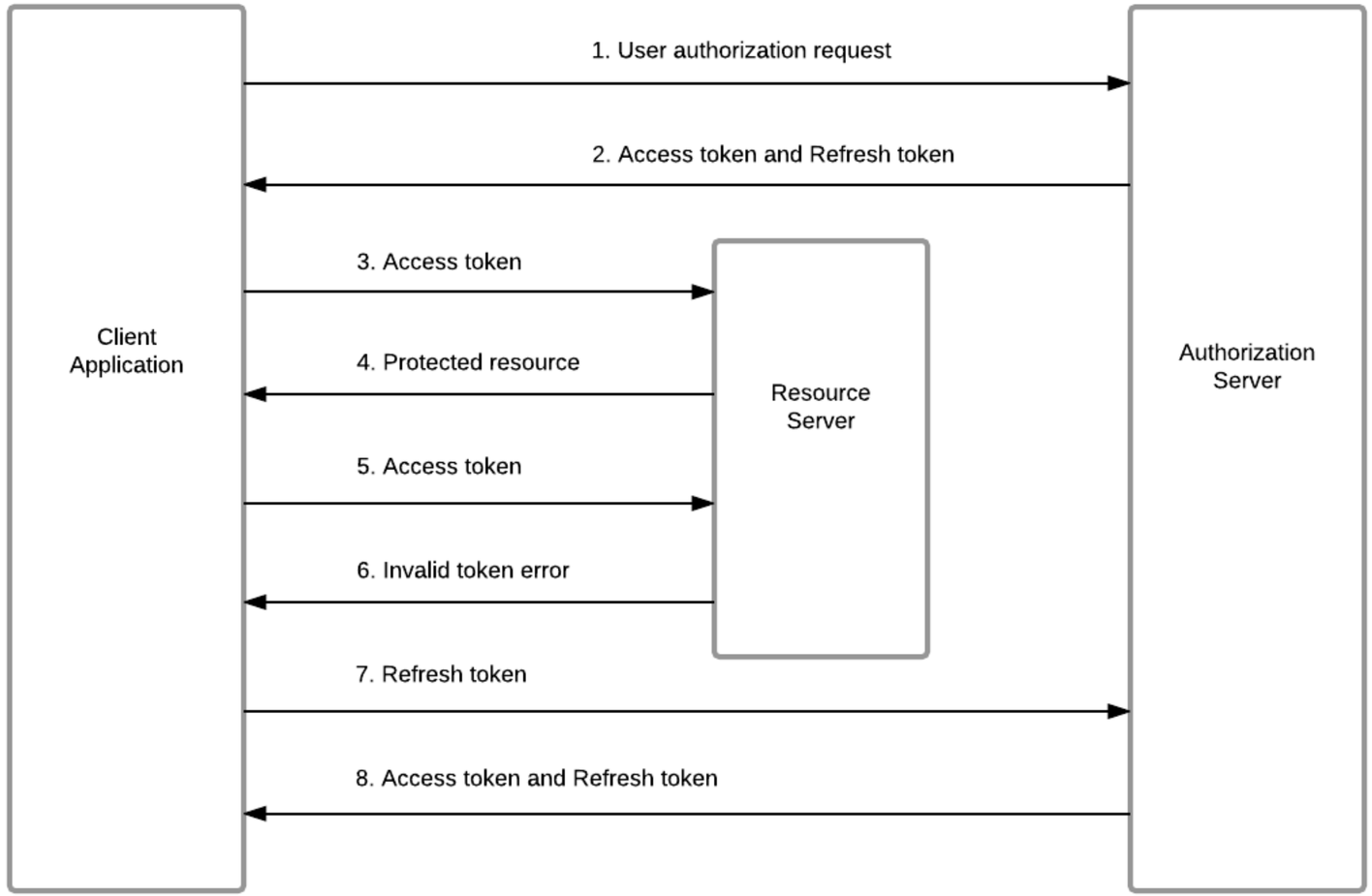


NORDICAPIS.COM

Solución 🥳

Solución

JWT



Solución

Unit testing & Code coverage

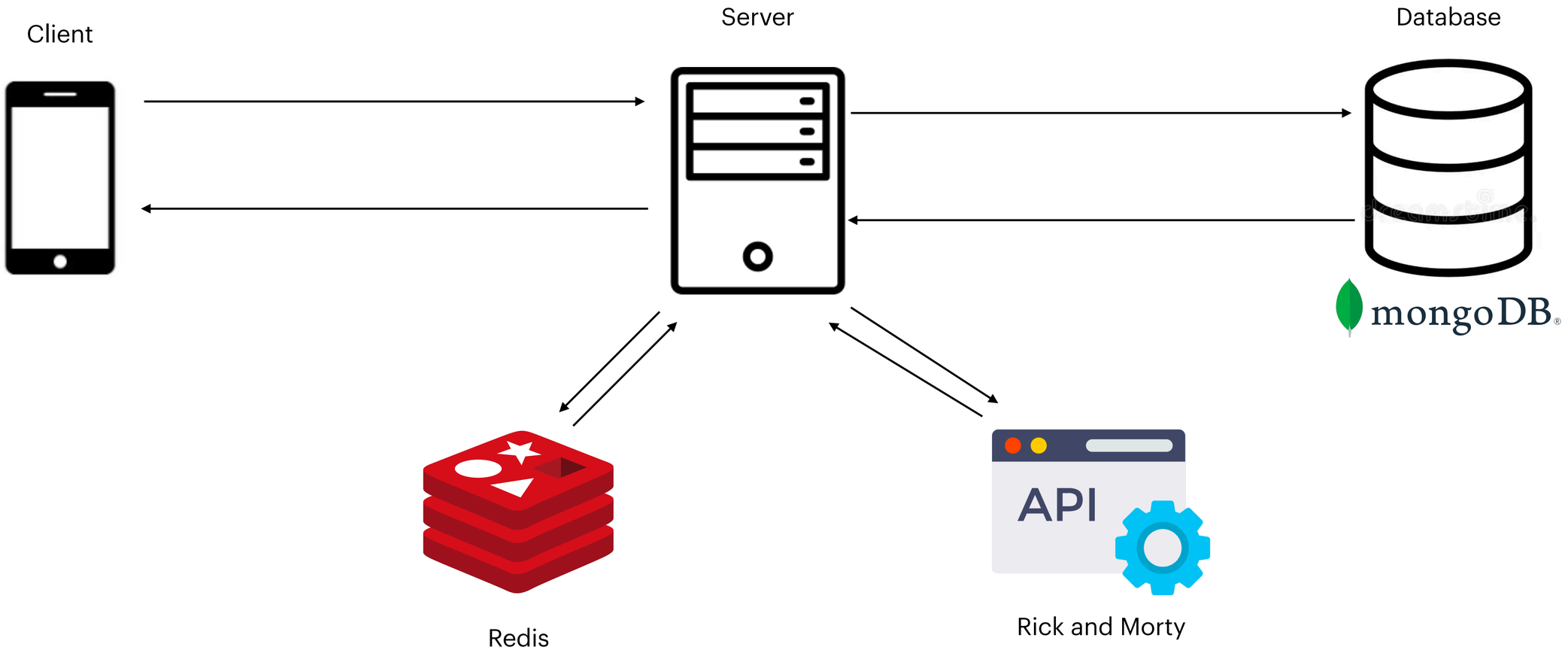
LCOV - code coverage report

				Coverage	Total	Hit
Current view:	top level - src/lib/src - secure_storage.dart (source / functions)		**Lines:**	100.0 %	10	10
Test:	lcov.info		**Functions:**	-	0	0
Test Date:	2023-07-11 17:42:00					

```
      Line data    Source code
    1             : import 'package:flutter_secure_storage/flutter_secure_storage.dart';
    2             : import 'package:storage/storage.dart';
    3             :
    4             : /// {@template secure_storage}
    5             : /// A Secure Storage client which implements the base [Storage] interface.
    6             : /// [SecureStorage] uses `FlutterSecureStorage` internally.
    7             : /// {@endtemplate}
    8             : class SecureStorage implements Storage {
    9             :   /// {@macro secure_storage}
   10           2 :   const SecureStorage({FlutterSecureStorage? secureStorage})
   11             :       : _secureStorage = secureStorage ?? const FlutterSecureStorage();
   12             :
   13             :   final FlutterSecureStorage _secureStorage;
   14             :
   15           1 :   @override
   16             :   Future<String?> read({required String key}) async {
   17             :     try {
   18           2 :       return await _secureStorage.read(key: key);
   19             :     } catch (error, stackTrace) {
   20           2 :       Error.throwWithStackTrace(StorageException(error), stackTrace);
   21             :     }
   22             :   }
   23             :
   24           1 :   @override
   25             :   Future<void> write({required String key, required String value}) async {
   26             :     try {
   27           2 :       await _secureStorage.write(key: key, value: value);
   28             :     } catch (error, stackTrace) {
   29           2 :       Error.throwWithStackTrace(StorageException(error), stackTrace);
   30             :     }
   31             :   }
   32             :
   33           1 :   @override
   34             :   Future<void> delete({required String key}) async {
   35             :     try {
   36           2 :       await _secureStorage.delete(key: key);
   37             :     } catch (error, stackTrace) {
   38           2 :       Error.throwWithStackTrace(StorageException(error), stackTrace);
   39             :     }
   40             :   }
   41             : }
```

Solución

Arquitectura



Demo 😎

Conclusiones 🤯

Conclusiones

- JWT nos brinda la seguridad de que los servicios que se ofrecen sea utilizado por aquellos que tengan la autorización de usarlo
- Unit testing nos ayuda a garantizar que las funcionalidades tengan un comportamiento adecuado
- Code coverage nos ayuda a ver que tanto de nuestro código esta protegido a posibles errores.

Bibliografía

- *Access Token Response - OAuth 2.0 Simplified*. (2021, December 16). OAuth 2.0 Simplified. https://www.oauth.com/oauth2-servers/access-tokens/access-token-response/
- *RFC 7519: JSON Web Token (JWT)*. (2015, May 19). IETF Datatracker. https://datatracker.ietf.org/doc/html/rfc7519
- Sevilla, M. (2022, April 12). Very good layered architecture in Flutter. *A Very Good Blog*. https://verygood.ventures/blog/very-good-flutter-architecture