I need you to add an authentication system to this project, for that reason I will need you to pay attention to the following instructions:

1. You should create a secure http way to consume any microservice from an allowed domain list, always trying to follow the OWASP mobile rules, excepting the cache rules, because we don’t need to storage the response of the services.
2. For any other microservices that aren’t authentication one you shouldn’t store cache.
3. The authentication service has the following structure:
   1. Domain: <http://10.14.11.200:7161/bowpi/>
   2. Microservice identifier: /micro-auth-service/
   3. Endpoint: /auth/login
   4. Method: POST
   5. Payload: { "username": USER\_EMAIL, "password": USER\_PWD, "application": "MOBILE", "isCheckVersion" : false }

**NOTE: For dev reason you must allow to consume HTTP request but in production you only should accept HTTPS requests.**

1. Most services response with ResponseWS interface, that has the following structure:

Export interface ResponseWS{

“code” : number,

“message” : string,

“success” : Boolean,

“data” : <T>

}

Always try to parse responses to ResponseWS interface and treats “data” as an object.

1. The authentication service in their data attribute response with a large string which is a JWT Token with their claims encrypted, so you must extract the “data” attribute from JWT Token and decrypt using a decrypting service which we will see later
2. Once JWT was parsed you can see a “requestId” attribute in the claims section, in this case for authentication purposes you must storage the “requestId” and use it as a main way to identify the user session using the auth storage method defined previously.

**IMPORTANT NOTE:** Since this is an offline-first app, the token **MUSTN’T EXPIRE** unless the user explicitly log out.

1. Once you decrypted the “data” claim, you should see a JSON User information like this interface:

export interface AuthTokenData {

*// JWT metadata*

  iss: string;

  aud: string;

  exp: number;

  iat: number;

  sub: string;

  jti: string;

*// Decrypted profile data*

  userId: string;                    *// officerCode o requestId*

  username: string;                  *// username del perfil*

  email: string;                     *// email del perfil*

  userProfile: {                     *// Estructura completa del perfil*

    username: string;

    email: string;

    names: string;

    lastNames: string;

    firstLogin: boolean;

    state: { id: number; value: string };

    phone: string;

    time: number;

    duration: number;

    agency: { id: number; value: string };

    region: { id: number; value: string };

    macroRegion: { id: number; value: string };

    employeePosition: { id: number; value: string };

    company: { id: number; name: string; type: string };

    permissions: string[];

    Groups: string[];

    hasSignature: boolean;

    officerCode: string;

    requestId: string;

    passwordExpirationDate?: string;

    passwordExpirationDays?: number;

  };

  permissions: string[];             *// permissions array*

  roles: string[];                   *// Groups array*

}

Please parse it and save as a global context to be easy recover user information when it’s necessary.

1. All Authentication requests to the server MUST have some mandatory headers which are the following:

export interface BowpiRequestHeaders extends Record<string, string> {

// ALWAYS

'Authorization': string; // 'Basic Ym93cGk6Qm93cGkyMDE3'

'Cache-Control': string; // 'no-cache'

'Pragma': string; // 'no-cache'

'OTPToken': string; // Generado dinámicamente

}

export type BowpiMandatoryHeaders = {

'X-Date': string; // Solo PUT/POST/PATCH

'X-Digest': string; // Solo PUT/POST/PATCH

'bowpi-auth-token'?: string; // This one only if existing session and isn’t /login

}

1. As you can see in the previous rule the BowpiMandatoryHeaders Type use X-Date and X-Digest those are generated by a custom service which you can see later.
2. To generate X-Digest and X-Date data you should create a service that MUST use the code can be found in the following files without modifying:
   1. BowpiOTPService.ts
   2. BowpiHMACService.ts
   3. BowpiCryptoService.ts
   4. BowpiAuthenticationInterceptor.ts
3. For the logout feature you should follow all the previous steps but the only thing which changes is the endpoint as follows:
   1. Domain: same as first
   2. Microservice: same as first
   3. Endpoint: /management/session/invalidate/request/{requestId}

Note: requestId is the main way to identify a user.

Note 2: This endpoint shouldn’t be waited for, just call it and forget about it.

1. This is the most important instruction, this is an application offline first, so before you perform an action you should check if the user has connection to the internet, if user has no network, you should perform offline following actions:

For login without internet:

* Tell the users that they haven’t internet access and this action can only be performed with internet access.

For login with internet access:

* Allow users to login and follow all the normal flows
* Save the token and let the user navigate through the app.
* If user has token in their phone skip login and navigate to tabs screen.

For logout without internet:

* WARN users to perform this action because they can’t log in again without internet
* If users confirm then log out.

For logout with internet access:

* Let the users log out normally and redirect to login screen.

1. Be clearly with logs while in dev, set all log you may need to debug.