ANEXO V: Manual técnico

Plataforma de monitorización remota para la atención domiciliaria

Trabajo de Fin de Grado

Ingeniería Informática

Julio de 2023

Autor

Germán Francés Tostado

Tutor/a

David Cruz García

Gabriel Villarrubia González

Tabla de contenidos

Índice de figuras

1.Introducción

Este anexo explicará la aplicación desde el punto de vista técnico para ofrecer ayuda al programador mediante descripciones del código.

Se podrá dividir en dos secciones, por un lado la plataforma web (con el frontend y el backend tal y como se vió en el anexo anterior) y por otro lado la aplicación de la tablet médica.

- ➔ **Plataforma web**: Hará referencia a los servidores desarrollados para la aplicación, separando el frontend del backend en dos directorios.
  - ◆ **Frontend**: Servidor que proporcionará la web con una interfaz gráfica a los usuarios, de manera que podrán interactuar con ella para identificarse, ver datos etc.
  - ◆ **Backend**: Servidor que proporcionará la API con la que interactuará el resto del programa. Procesará todas las peticiones recibidas y gestionará los datos del sistema.
- ➔ **Tablet**: Hará referencia a la aplicación android desarrollada exclusivamente para la tablet, se comunicará con la plataforma web mediante peticiones a la API REST.

2. Estructura del sistema

En este apartado especificará dónde estará alojada cada parte del código para su posterior localización.

2.1 Frontend

El código fuente del servidor frontend estará ubicado dentro de la carpeta “codigo-fuente/web-medica-frontend”.

Este servidor ha sido creado utilizando NodeJS y React.

Dentro del directorio tendremos como carpeta principal el directorio **src/**:

- **src/** : Contendrá todo el código necesario para las vistas y diferentes assets. Se compondrá de diferentes fuentes.
    - **_nav.js**: Barra de navegación lateral básica para todos los usuarios
    - **_navAdmin.js**: Barra de navegación lateral para los usuarios con rol Admin.
    - **_navMedic.js**: Barra de navegación lateral para los usuarios con rol Médico.
    - **App.js**: Cargará toda la página y el BrowserRouter.
    - **config.js**: Fichero de configuración para establecer la conexión al servidor Backend.
    - **routes.js**: Fichero que contiene todas las rutas a las vistas.
    - **assets/**: Directorio con imágenes y logos que se utilizan durante la aplicación
    - **components/**: Guardará los componentes y cabeceras de la plantilla utilizada para la aplicación.
    - **layout/**: Tendrá el fichero con la vista por defecto (DefaultLayout)
    - **services/**: Diferentes fuentes que harán de middleware para la conexión con los microservicios del backend.
    - **views/**: Directorio con todas las páginas accesibles del sistema, donde cada una tendrá su fuente con la vista correspondiente. Las vistas han sido descrita en la fase de diseño del Anexo IV.

A parte de src/, tendremos también el directorio **public/** , que será usada por React para poder operar, en adición de guardar el favicon de la aplicación.

2.2 Backend

El código fuente del servidor backend estará ubicado dentro de la carpeta "codigo-fuente/web-medica-backend".

Este servidor ha sido creado utilizando NodeJS y Express.

- **app.js** y **bin/www**: Ficheros de arranque del servidor Express que cargarán los routers, los modelos de la base de datos, el puerto de arranque del server…

- **data/**: Guardará los archivos .data de cada usuario, con sus últimas mediciones, aparte de un fichero general que guarda las últimas mediciones del sistema.

- **medicalHistories/**: Directorio donde se guardan los historiales médicos en pdf.

- **middleware/**: Tendrá los archivos middleware encargados de verificar el acceso a recursos y validaciones previas a las peticiones.

- **models/** : Tendrá los modelos de las tablas de la base de datos requeridas por la biblioteca Sequelize

- **routes/**: Contiene los diferentes routers que ofrecen las rutas API. Los fuentes están documentados automáticamente mediante la herramienta Swagger. Esta documentación se encuentra en **web-medica-backend/api.pdf**

- **services/**: Contiene los microservicios ofrecidos por la aplicación. Las rutas API de estos ficheros se encontrarán también en **web-medica-backend/api.pdf**

- **config.js**: Fichero de configuración que almacenará los datos de conexión a la base de datos y el secreto del token de autenticación JWT.

- **swagger.js**: Fichero que genera documentación automática de la aplicación Express.

2.3 Tablet

El código fuente de la aplicación exclusiva de la tablet se ha basado en un SDK de la empresa Contec Medical Systems Co, para el dispositivo HMS7500 Multi-parameter Vital Signs Monitor, denominado "Tablet" para la simplificación de la documentación.

Mediante el SDK de Contec, se abrirán los puertos serie de la Tablet, a la cual estarán conectados los sensores, que cuando se activen, enviarán las lecturas mediante este puerto serie.

Estas lecturas se recogerán en la aplicación en el fuente **WaveActivity.java** cada vez que haya una actualización de un dato, se invocará el método callXMsg(), donde la X será la constante vital medida.

Por ejemplo veamos el caso del sensor de presión arterial:

```java
@Override
public void callNBPRESULT1Msg(short sbp, short dbp, short abp) {
    nbp_result[0] = sbp;
    nbp_result[1] = dbp;
    nbp_result[2] = abp;
    mHandler.obtainMessage(MESSAGE_UPDATE_NBPRESULT, nbp_result)
            .sendToTarget();
    HttpClient httpClient = new DefaultHttpClient();
    try {
        HttpPost request = new HttpPost( uri: "http://192.168.0.164:4000/data?id="+DNI);
        JSONObject json = new JSONObject();
        json.put( name: "sbp",sbp);
        json.put( name: "dbp",dbp);
        json.put( name: "abp",abp);
        StringEntity params = new StringEntity(json.toString());
        request.addHeader( name: "Content-type",  value: "application/json");
        request.setEntity(params);
        HttpResponse response = httpClient.execute(request);
    } catch (Exception ex) {
    } finally {
        httpClient.getConnectionManager().shutdown();
    }
}
```

Figura 1: Fragmento ejecutado al leer presión arterial

Observamos que se enviará una petición POST a la API backend de la plataforma web, más concretamente al endpoint **/data**. Esta petición tendrá adjuntada en el body mediante un json los datos medidos, mientras que por argumento se pasará el DNI del usuario logueado en la aplicación.

Para la interfaz de login, tendremos el fuente **LoginActivity.java** que enviará los datos introducidos a la API backend para ello (**/auth/signin**). En lo relevante a la interfaz, está desarrollada en xml en los ficheros **/res/layout/activity_login.xml** para el login, y **/res/layout/wave.xml** para la página principal.

Cómo último apunte, queda destacar que para la conexión con la sala de consultas para videollamadas, se hará uso del SDK de Jitsi.meet, que lanzará una actividad **JitsiMeetActivity** cuando se pulse el botón de la sala de consultas, accediendo así a la sala de videollamadas de manera nativa en la aplicación.

3.Generación de documentación API

Para la documentación de la API del server backend, se ha usado Swagger, generando así, un fichero **web-medica-backend/swagger_output.json** con las especificaciones de cada endpoint.

Puesto que es un JSON, se ha hecho uso de la herramienta **swagger2pdf** para convertirlo a archivo PDF más legible, el cual está situado en **web-medica-backend/api.pdf**

4. Referencias

- Generación de documentos usando swagger
  - https://swagger.io/
- Herramienta Swagger2pdf
  - http://rajeevdotnet.blogspot.com/2019/02/export-swagger-api-document-to-pdf.html
- Página oficial de CONTECMED
  - https://www.contecmed.com/sy