GIEMA-IoT: Herramienta móvil para la gestión de información generada por estaciones meteorológicas autónomas en un modelo IoT

ANEXO - D

Pérez Martínez Jhon Alexander

Universidad de San Buenaventura, Sede Bogotá.
Facultad de Ingeniería.
Programas de Ingeniería de Sistemas
Bogotá, Colombia
2020

Tabla de contenido

4. Fase de desarrollo

4.1 Configuración del Entorno de Desarrollo Y Versiones.

Para el desarrollo de la aplicación se realizaron algunas configuraciones en los entornos de desarrollo que se usaron para poder realizar de manera óptima la construcción del aplicativo móvil, como primera medida se configuro el entorno de desarrollo AndroidStudio en su versión:

Android Studio 3.5.1
Build #AI-191.8026.42.35.5900203, built on September 25, 2019
JRE: 1.8.0_202-release-1483-b03 amd64
JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o
Windows 10 10.0

Cuando el entorno de desarrollo se encuentre listo para poder construir un aplicativo saldrá

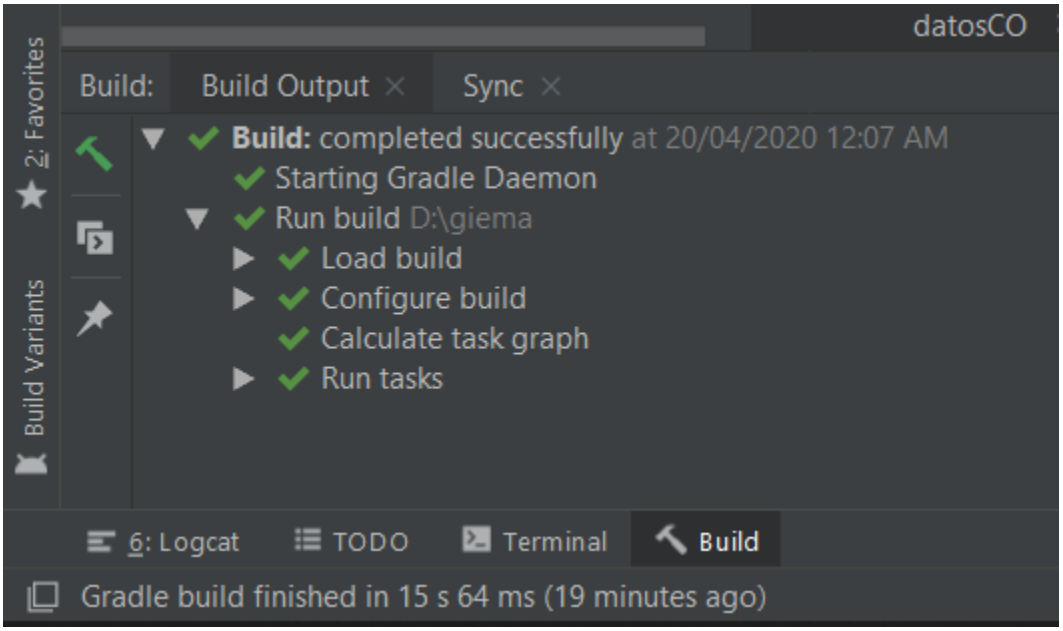

Figura 1 Entorno de desarrollo

Después de verificar que se todo se encuentre en orden se realizaran las siguientes configuraciones para que la aplicación pida los permisos necesarios para ejecutar la aplicación de manera exitosa, en este caso serán los encargados de pedir acceso a las conexiones de internet del dispositivo y permisos completos de red, para que la aplicación pueda conectarse a internet de manera automática.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

Figura 2 Permisos en la aplicación

Para el siguiente paso se importaron las librerías necesarias para que el proyecto funcione de manera óptima en este caso se usó las librerías como Volley que es la encargada de administrar las redes de la aplicación, la librería jplot encargada de la parte gráfica y ayudas visuales de la aplicación, la librería Loopj que se encarga de realizar peticiones HTTP de manera asíncrona, y la librería MPandroidchart que es la encargada de realizar la graficas en la aplicación, entre otras.

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.0.2'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.0'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.1.1'
    implementation 'com.github.PhilJay:MPAndroidChart:v3.0.3'
    implementation 'com.loopj.android:android-async-http:1.4.9'
    implementation 'com.android.support:design:29.1.1'
    implementation 'com.android.volley:volley:1.1.0'
    implementation project(path: ':jplot1.2')
    implementation 'com.loopj.android:android-async-http:1.4.9'
}
```

Figura 3 Librerías y complementos

Una vez importada las librerías necesarias y aplicando los cambios, se deberá refrescar la estructura de desarrollo y a continuación de deberá importar los componentes para que la librería MPandroidchart funcione de manera correcta con la siguiente configuración.

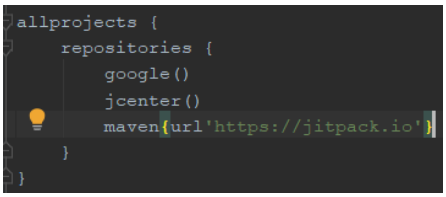

Figura 4 Repositorios

Realizando esto el entorno de desarrollo está listo para ser usado.

4.2. Desarrollo del WebService

En la actualidad podemos encontrar gran variedad de formas arquitectónicas para el desarrollo del servicio de los diferentes sistemas que se encuentran hoy en día en el mundo, después del análisis realizado se esta arquitectura ya que se acopla perfectamente a lo que se busca para el presente proyecto ya que tiene las características para manipular los datos de las diferentes variables correctamente y conectarlos a una API de una manera muy dinámica sus características se centran permitir crear, leer, actualizar y borrar información de manera ágil usando métodos como POST , GET, para las operaciones anteriores necesitan una URL y métodos HTTP para accederlas, regresan los datos en formato JSON y pueden retornar códigos de respuesta como 200,201,404 etc. Lo que convierte a esta arquitectura en un aliado ideal para el desarrollo del presente proyecto

El desarrollo del WebService será realizado de manera manual en PHP 7.0 debido a que por temas de permisos donde se alojará parte de los archivos del proyecto del WebService en el Servidor de la Universidad San Buenaventura no se puede usar un Framework para realización de este proceso ya que se deben instalar ciertos componentes para que este funcione y no se cuenta con los permisos, supliendo este requerimiento se hace una análisis y se determina que PHP en su línea 7.0 en adelante puede ofrecernos las herramientas en cuanto a seguridad y viabilidad para el desarrollo del WebService.

Tomando los principios de la arquitectura RESTful se logra crear un ambiente propicio para la manipulación de datos sin saturar el servidor y adicionalmente bajando la carga en la aplicación ya que el webService al almacenar todos los métodos que manipulan los datos, lo demás son peticiones

al mismo WebService para poder ejecutar la acción requerida dentro de los márgenes de la arquitectura ya montada, se desarrolló el webService usando los diferente métodos GET , POST, creando una clase controller la cual contiene todos estos métodos y pueden usarse solo llenando el método requerido y llamándolo a la aplicación como se ve en la figura 7.

```php
if ($_SERVER['REQUEST_METHOD'] == 'GET')
{
    if (isset($_GET['username']))
    {
      //Mostrar un post
      $sql = $dbConn->prepare("SELECT * FROM usuario where username=:username");
      $sql->bindValue(':username', $_GET['username']);
      $sql->execute();
      header("HTTP/1.1 200 OK");

      echo json_encode(  $sql->fetch(PDO::FETCH_ASSOC)  );
      exit();
```

Figura 5 Método GET - Controller

La arquitectura del WEB Servicie notifica al servidor el tipo de método http que va usar para realizar la acción cuando esta es validada , toma la variable que va ser en este caso tomada para una consulta por el método GET , donde una variable toma la consulta no sin antes llamar una variable que se encarga de almacenar los datos de conexión, cuando realiza la consulta valida que el proceso sea exitoso, si el proceso tiene éxito el servidor devolverá un estado HTTP 200 que significa que el proceso fue exitoso y procede a volver la respuesta de la consulta en un formato JSON, más adelante se explicara más para que sirve este formato.

Esta arquitectura se aplica a todas las tablas del modelo de datos, el WebService junto con el modelo de datos nos darán un gran manejo para optimizar recursos y manejar datos de manera más eficiente y segura ya, que todas las tablas tienen la misma estructura el manejo se vuelve más fácil a medida que se le van agregando más métodos y acciones que tiene que realizar el sistema IoT en su totalidad, esta estructura es el WebService como tal aplicado a las 7 tablas con los diferentes métodos,

dándole a la arquitectura gran ventaja en cuanto a cómo manipula los datos que envía la estación meteorológica .

4.3 Desarrollo de la aplicación.

El desarrollo de aplicaciones móviles y su arquitectura es muy variada, en la actualidad existen ciertas arquitecturas que dominan su preferencia en uso, debido a que cuentan con una estructura más amigable de usar y cuidan el concepto de buenas prácticas en la programación, para el presente proyecto se determinó que esta arquitectura es la más adecuada debido a que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos donde el modelo accede a la capa de datos , el controlador recibe los eventos de la entrada y la vista recibe los datos del modelo y los muestra al usuario, es la más indicada, trabajando junto con el IDE de desarrollo que se seleccionó para la creación de la aplicación ,Android Studio se adapta esta arquitectura de desarrollo para poder seguir con las buenas prácticas de la programación, siguiendo los lineamientos dados por el IDE de desarrollo se puede concluir que es la mejor opción para interactuar con los datos debido a que el IDE Android Studio 3.5.1 trabaja con Layouts para el apartado grafico que se codifican en XML y se trabaja su backen en JavaClass ofrece una herramienta completa para el desarrollo de la aplicación móvil con buenas prácticas.

Se programó el aplicativo separando cada sección en clases , debido a que en Android se maneja algo que se llama Activity que son las pantallas que deben ser programadas individualmente para su funcionamiento y navegación se trató de aprovechar al máximo la arquitectura propuesta en el WebService para que todos métodos y acciones quedaran dentro de los controller y los demás métodos que se crearon adicionalmente para ejecutar acción como validación registro y consulta, para el desarrollo de la aplicación se usó el concepto de llamar todo desde el webService para aliviar carga tanto en el servidor como en el aplicativo donde se evaluó y se desarrolló sobre un sistema Android.5.5 en adelante, es decir que los celulares con ese sistema operativo Android soportaran la carga de la aplicación, dicho esto la aplicación funciona por medio de clases que será la pantalla

individual ligada a un Activity que es el apartado gráfico , dentro de cada clase se desarrolla el backen usando métodos para consumir el servicio desde el webservice , solicitamos el tipo de servicio que deseamos realizar y donde por ejemplo creando una clase stringRequest llamamos un método POST a una URL , este proceso es como ordenarle al servidor enviare una petición POST y el servicio que se solicita está en esa dirección lo que iniciara un proceso donde hay una respuesta y se crear un objeto JSON donde se almacena la cadena JSON que devuelve el servidor para almacenar los datos , ya en este punto los datos se encuentran en nuestra cadena JSON en nuestra aplicación solo por el momento de realizar la petición , ya después de finalizado este proceso se rompe lo que no le provoca carga a la aplicación ni al servidor, si la aplicación tiene éxito en el proceso devuelve un estado success 1, si se llega a fallar envía un error con el motivo todo esto se puede evidenciar.

```
private void llenarDatos() {

    String url = "http://software.ingusb.com/giema/giematest/datosGraficaTemperatura.php?fecha="+etFecha.getText().toString();
    client.get(url, new AsyncHttpResponseHandler() {

        //Creamos un metodo paramtrizando el response
        @Override
        public void onSuccess(int statusCode, Header[] headers, byte[] responseBody) {
            if (statusCode == 200) {

                //Cargamos el metodo que contiene la logica que adminsitra los datos meterologicos
                cargarDatos(new String(responseBody));
            }
        }

        @Override
        public void onFailure(int statusCode, Header[] headers, byte[] responseBody, Throwable error) {

        }
    });

}
```

Figura 6 Ejemplo de petición

Para la visualización de los datos se tuvo especial cuidado para tratar de forma individual cada sensor en su petición donde hay una clase para cada uno y poder visualizar los datos, pero hay una javaClass independiente que realiza la llamada del objeto para ser filtrado por la consulta SQL en el webService , recordando que la estructura de datos la tabla mensaje contiene los datos que envía la estación meteorológica de los diferentes sensores , es un constante flujo de información en un punto, pero en su almacenamiento está debidamente filtrado lo que realiza una petición que se mantiene en la aplicación solo durante su consulta como se evidencia..

```
V/AsyncHttpRH: Progress 38 from 38 (100%)
I/tagconvertstr: [[{"nombreNodo":"U. San Buenaventura"}]]
V/AsyncHttpRH: Progress 699 from 699 (100%)
I/tagconvertstr: [[{"valor":"11"},{"valor":"11"},{"valor":"8"},{"valor":"9"},{"valor":"11"},{"valor":"9"},{"valor":"8"},{"valor":"11"},{"valor":"9"},{"v
D/OpenGLRenderer: endAllStagingAnimators on 0xb3d70e00 (RippleDrawable) with handle 0xb3fddac0
D/OpenGLRenderer: endAllStagingAnimators on 0xa0cd5b80 (RippleDrawable) with handle 0xb3f18660
V/AsyncHttpRH: Progress 361 from 361 (100%)
I/tagconvertstr: [[{"valor":"30"},{"valor":"30"},{"valor":"30"},{"valor":"29"},{"valor":"29"},{"valor":"30"},{"valor":"29"},{"valor":"29"},{"valor":"28"
D/OpenGLRenderer: endAllStagingAnimators on 0xb3d71180 (RippleDrawable) with handle 0xb3f490b0
D/OpenGLRenderer: endAllStagingAnimators on 0xa0e86000 (RippleDrawable) with handle 0xaec29ba0
V/AsyncHttpRH: Progress 361 from 361 (100%)
I/tagconvertstr: [[{"valor":"14"},{"valor":"15"},{"valor":"11"},{"valor":"13"},{"valor":"13"},{"valor":"17"},{"valor":"16"},{"valor":"14"},{"valor":"17"
D/OpenGLRenderer: endAllStagingAnimators on 0xb3d71500 (RippleDrawable) with handle 0xb3f49f90
V/AsyncHttpRH: Progress 337 from 337 (100%)
I/tagconvertstr: [[{"valor":"0"},{"valor":"0"},{"valor":"1"},{"valor":"1"},{"valor":"1"},{"valor":"1"},{"valor":"1"},{"valor":"0"},{"valor":"0"},{"valor
D/OpenGLRenderer: endAllStagingAnimators on 0xb3d71880 (RippleDrawable) with handle 0xaec29ba0
```

Figura 7 Log de datos

Y muestra estos datos solo durante la petición y de consulta, ya sea para su visualización o para ser graficados como se muestra en figura 8.

Figura 8 Visualización de datos en a la aplicación.

4.4. Repositorio del código fuente

Una vez terminado el desarrollo se tuvo como resultado los siguientes componentes, por el lado de la aplicación GIEMA, que se encuentra disponible en el link a continuación: http://software.ingusb.com/giema/descargaGiema.php, se podrá descargar la aplicación haciendo clic en “Descargar App”.

Por su parte todo el proyecto GIEMA está disponible en el repositorio de github donde se encontrar el webservice y el modelo de datos, junto a la documentación de despliegue tanto del modelo de datos en el servidor como el uso de la app en el siguiente link: a continuación: https://github.com/aledoc72/giema.

Con esto concluye el segmento de desarrollo, ahora se procederá a evaluar el protocolo de pruebas realizado para certificar la funcionalidad del proyecto cumpliendo con las metas pactadas.