UNIVERSIDAD CENTROAMERICANA
JOSÉ SIMEÓN CAÑAS

SISTEMA PARA LA RECOLECCIÓN DE DATOS DE DIFERENTES DISPOSITIVOS INSTALADOS EN LA UCA UTILIZANDO LA RED LORA

MANUAL TÉCNICO DEL SISTEMA

CONTENIDO

RESUMEN

Se presenta el siguiente manual técnico como una guía de apoyo con la información adecuada para realizar la instalación y mantenimiento del sistema desarrollado para la recolección y presentación de datos de diferentes dispositivos instalados en la universidad, utilizando la red LoRa.

El manual técnico presenta información acerca de los elementos utilizados para el desarrollo del sistema, el funcionamiento de estos y las especificaciones necesarias para facilitar la edición de este, de manera propicia.

OBJETIVOS

- Informar sobre los elementos técnicos utilizados para el desarrollo del sistema de recolección de datos.

- Ayudar a facilitar el mantenimiento, actualización e instalación de los recursos necesarios que requiere el sistema.

INTRODUCCION

HERRAMIENTAS TECNICAS

NODE JS

Node.js es una biblioteca y un entorno de ejecución de JavaScript multiplataforma de código abierto para ejecutar aplicaciones web fuera del navegador del cliente. Ryan Dahl lo desarrolló en 2009 y su última versión, la versión 15.14, se lanzó en abril de 2021. Los desarrolladores utilizan Node.js para crear aplicaciones web del lado del servidor y es perfecto para aplicaciones con uso intensivo de datos, ya que utiliza un evento asincrónico.

Node.js permite ejecutar JavaScript en el lado del servidor. Está construido sobre el motor JavaScript V8 de Chrome, que compila JavaScript en un código de máquina eficiente. Node.js opera en una arquitectura basada en eventos de un solo subproceso, utilizando un bucle de eventos para manejar múltiples operaciones simultáneas sin bloquear. (Taha Sufiyan, 2024).

En el sistema se utiliza para la creación y control de la API, que es la que consume los datos, los añade a la base de datos y posteriormente los muestra en la interfaz de usuario.

VUE JS

Vue es un framework progresivo para construir interfaces de usuario. A diferencia de otros frameworks monolíticos, Vue está diseñado desde cero para ser utilizado incrementalmente. La librería central está enfocada solo en la capa de visualización, y es fácil de utilizar e integrar con otras librerías o proyectos existentes. Por otro lado, Vue también es perfectamente capaz de impulsar sofisticadas *Single-Page Applications* cuando se utiliza en combinación con herramientas modernas y librerías de apoyo. (Vue, s. f.)

Utilizada en el sistema para darle vida a la aplicación de manera visual, siendo la principal herramienta para construir la interfaz de usuario de manera amigable.

DOCKER

El sistema de software de TI llamado "Docker" es la tecnología de organización en contenedores que posibilita la creación y el uso de los contenedores de Linux®.

El enfoque de Docker sobre la organización en contenedores se centra en la capacidad de separar una parte de la aplicación para actualizarla o repararla, sin necesidad de deshabilitarla por completo. Además de aprovechar este modelo basado en los microservicios, puede intercambiar procesos entre varias aplicaciones casi de la misma forma en que funciona la arquitectura orientada a los servicios (SOA).

Cada archivo de imagen Docker está compuesto por varias capas que conforman una sola imagen. Cuando un usuario especifica un comando, como *ejecutar* o *copiar*, la imagen cambia, y se crea una capa nueva. (RedHat, 2023)

Utilizado en el trabajo de graduación para el alojamiento del sistema de recolección de datos.

GITHUB

Github es un portal creado para alojar el código de las aplicaciones de cualquier desarrollador, y que fue comprada por Microsoft en junio del 2018. La plataforma está creada para que los desarrolladores suban el código de sus aplicaciones y herramientas, y que como usuario no sólo puedas descargarte la aplicación, sino también entrar a su perfil para leer sobre ella o colaborar con su desarrollo.

Como su nombre indica, la web utiliza el sistema de control de versiones Git diseñado por Linus Torvalds. Un sistema de gestión de versiones es ese con el que los desarrolladores pueden administrar su proyecto, ordenando el código de cada una de las nuevas versiones que sacan de sus aplicaciones para evitar confusiones. Así, al tener

copias de cada una de las versiones de su aplicación, no se perderán los estados anteriores cuando se va a actualizar. (Yubal, 2019)

Se han ocupado repositorios separados para el alojamiento y control de versiones de la aplicación, se cuenta con uno para el frontend uno para el backend y uno para la aplicación móvil, todos los repositorios son privados, con acceso a los miembros del equipo,

POSTGRESQL

PostgreSQL es una base de datos de código abierto que tiene una sólida reputación por su fiabilidad, flexibilidad y soporte de estándares técnicos abiertos. A diferencia de otros RDMBS (sistemas de gestión de bases de datos relacionales), PostgreSQL soporta tipos de datos relacionales y no relacionales. Esto la convierte en una de las bases de datos relacionales más compatibles, estables y maduras disponibles actualmente.

PostgreSQL soporta diferentes optimizaciones de rendimiento que normalmente solo se encuentran en la tecnología de base de datos patentada, como el soporte geoespacial y la concurrencia sin restricciones. Esto hace que PostgreSQL sea extremadamente eficiente cuando se ejecuta un análisis de datos extenso y profundo en múltiples tipos de datos.

Para el sistema de recolección de datos se ha utilizado una base de datos PostgreSQL para el almacenamiento de la información recolectada de los sensores mediante la red LoRa, separándola en tablas organizadas y relacionadas para su posterior consumo en la aplicación.

PARA APLICACIÓN MOVIL

KOTLIN

Kotlin es un lenguaje de programación de código abierto creado por JetBrains que se ha popularizado gracias a que se puede utilizar para programar aplicaciones Android.

Este lenguaje es de tipado estático, ya que se puede desarrollar sobre JVM o JavaScript; o desde hace unos meses, incluso sin necesidad de ninguna de ellas, ya que paralelamente se está desarrollando en nativo con LLVM. Gracias a eso, es totalmente interoperable con código Java, lo que permite migrar de una forma gradual nuestros proyectos.

Una de las características principales de Kotlin es que está diseñado para interoperar completamente con la sintaxis del lenguaje Java.

Kotlin se puede utilizar para cualquier tipo de desarrollo, desde la web del lado del servidor y del lado del cliente, hasta Android y iOS. Como el lenguaje se ejecuta en JVM, permite compartir código entre diferentes plataformas.

Utilizado por el equipo para el desarrollo de la aplicación móvil, abarcando el frontend y backend dentro de sí.

DIAGRAMAS

DIAGRAMA ENTIDAD - RELACION

Modelo entidad-relación creado para el almacenamiento de la data recolectada

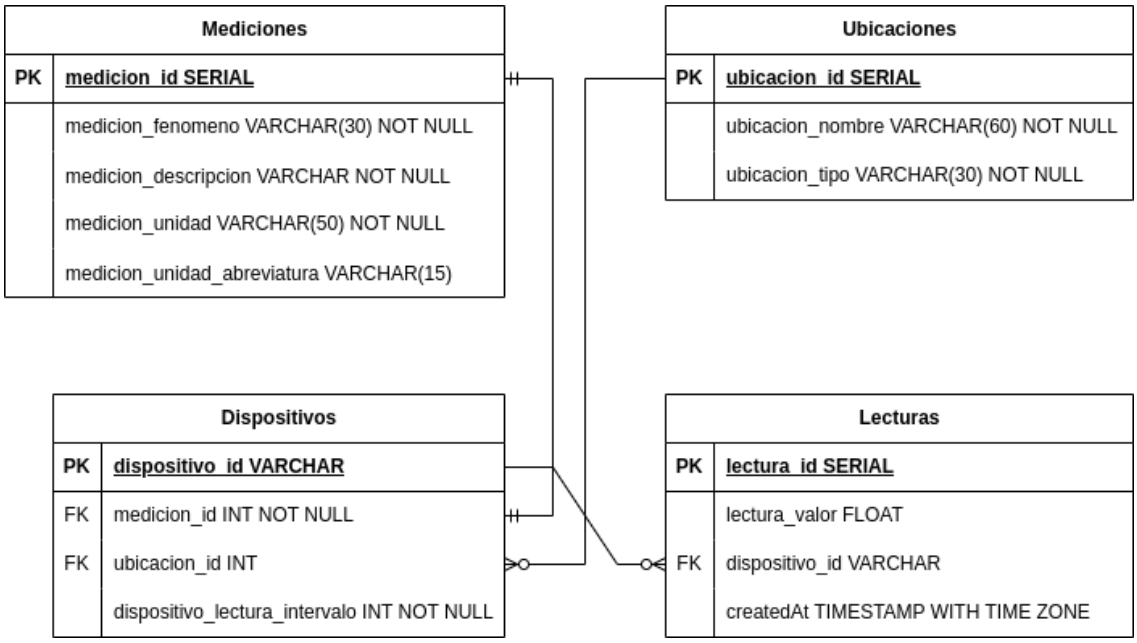


Diagrama 1: Diagrama entidad-relación.

Fuente: Elaboración propia

Detalle de las entidades utilizadas para la asignación y almacenamiento de datos en donde:

Mediciones: Almacena el detalle de las mediciones realizadas por los sensores.

Ubicaciones: Almacena el detalle geográfico de donde se realizó la medición.

Dispositivos: Almacena el detalle del dispositivo (sensor) que se encuentra generando datos.

Lecturas: Almacena el detalle de los valores obtenidos en las mediciones

DICCIONARIO DE DATOS

El diccionario de datos presenta el detalle de cada campo que conforma las entidades de la base de datos.

Campo	Tipo	Descripción
medicion_id	SERIAL	Llave primaria con la que se identifica dentro de la tabla.
medicion_fenomeno	VARCHAR	Nombre del fenómeno que se busca medir. Por ejemplo: “Radiación” en el caso de la medición de los rayos UV, o “Partículas en el aire” en el caso de la medición de partículas.
medicion_descripcion	VARCHAR	Complemento para el campo fenomeno_medicion para proporcionar mayor información acerca de la medición que se busca. Por ejemplo: “Intensidad de rayos UV a campo abierto” o “Densidad de partículas en una habitación cerrada”.
medicion_unidad	VARCHAR	Nombre de la unidad sobre la cual se mide la magnitud numérica. Por ejemplo: “Microvatios por centímetro cuadrado por nanómetro” o “Índice UV” para las mediciones de rayos UV, o “Partes por millón” para la concentración de partículas.
medicion_unidad_abreviatura	VARCHAR	Campo opcional en caso de que medicion_unidad cuente con una abreviatura. Por ejemplo: “µW/cm²nm” para “Microvatios por centímetro cuadrado por nanómetro”, o “ppm” para “Partes por millón”.

Tabla 1: Mediciones.

Fuente: Elaboración propia

Campo	Tipo	Descripción
dispositivo_id	VARCHAR	Llave primaria con la que se identifica dentro de la tabla, tomado del ID proporcionado en la configuración con el gateway LoRaWAN.
medicion_id	INT	Llave foránea que para referenciar el tipo de fenómeno que el dispositivo medirá, con cardinalidad 1:N.
ubicacion_id	INT	Llave foránea que para referenciar la ubicación donde se colocará el dispositivo, con cardinalidad 1:N.
dispositivo_lectura_intervalo	INT	Cantidad entera que indique con qué frecuencia (en segundos) se realiza una nueva medición.

Tabla 2: Dispositivos.

Fuente: Elaboración propia

Campo	Tipo	Descripción
ubicacion_id	SERIAL	Llave primaria con la que se identifica dentro de la tabla.
ubicacion_nombre	VARCHAR	Campo para especificar un lugar físico perteneciente a la universidad. Por ejemplo: “Campo de fútbol”, “Aula B-23”, “Tejado Edificio Jon de Cortina”.
tipo_ubicacion	VARCHAR	Campo para añadir información acerca del entorno de la ubicación. Por ejemplo: “Al aire libre”, “Techado”, “Cerrado”.

Tabla 3: Ubicaciones.

Fuente: Elaboración propia

Campo	Tipo	Descripción
lectura_id	SERIAL	Llave primaria con la que se identifica dentro de la tabla.
lectura_valor	FLOAT	Campo que contiene el valor recibido de un dispositivo.
dispositivo_id	VARCHAR	Llave foránea que relaciona el dispositivo al que pertenece la lectura registrada.
Campo	Tipo	Descripción
createdAt	TIMESTAMP WITH TIME ZONE	Campo que almacena la fecha y hora de creación de la lectura, con zona horaria de El Salvador y convertida a formato UTC.

Tabla 4: Lecturas.

Fuente: Elaboración propia

DIAGRAMA DE CASOS DE USO

Se ha diseñado el siguiente diagrama de casos de uso para representar el papel que desempeña cada actor relacionado al Sistema, siendo el actor principal el usuario final que consulta la información (entidades universitarias/estudiantes).

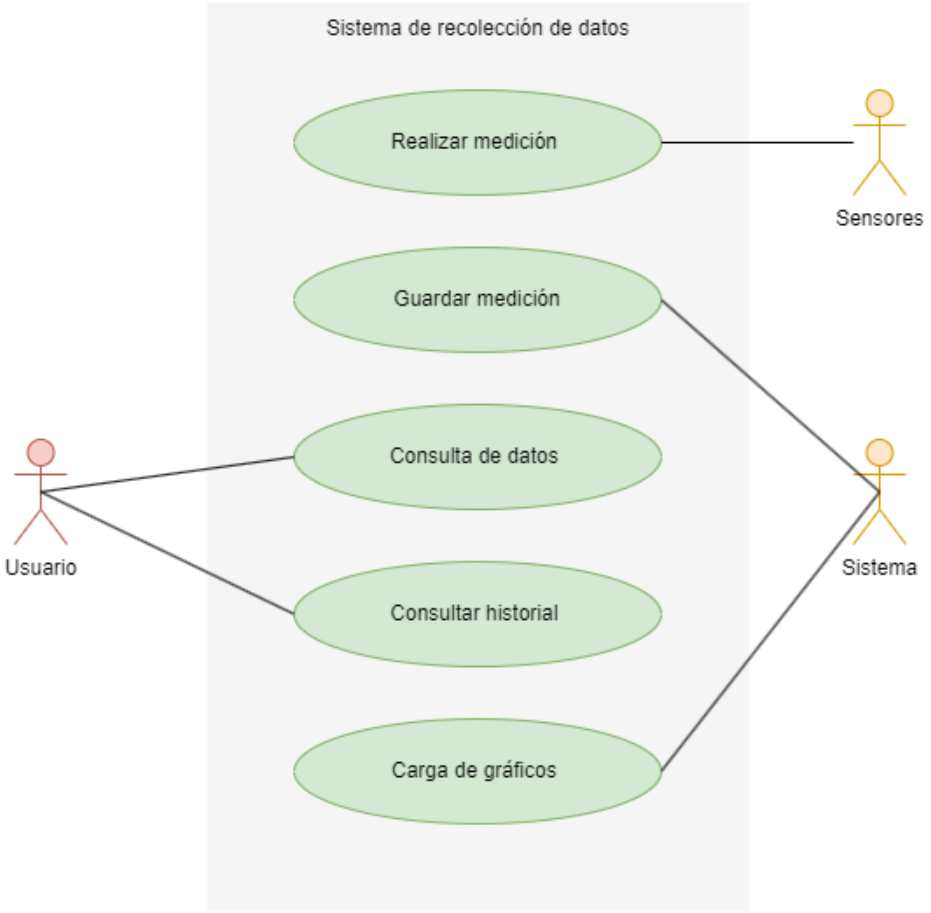


Diagrama 2: Diagrama de casos de uso.

Fuente: Elaboración propia

EJECUCIÓN Y MANTENIMIENTO

REQUERIMIENTOS

Para el acceso a la fuente del sistema (código) para realizar mantenimiento de este, tanto web como en versión móvil, es necesario considerar ciertos requerimientos o programas a utilizar para un mejor manejo de código.

Si lo que se desea es realizar modificaciones en ambiente de desarrollo (de manera local), se recomienda contar con los softwares siguientes para un cómodo manejo del sistema.

PROGRAMAS

- Visual Studio Code, editor de código fuente en su latest version el cual puede ser descargado de forma gratuita en el sitio https://code.visualstudio.com/ .

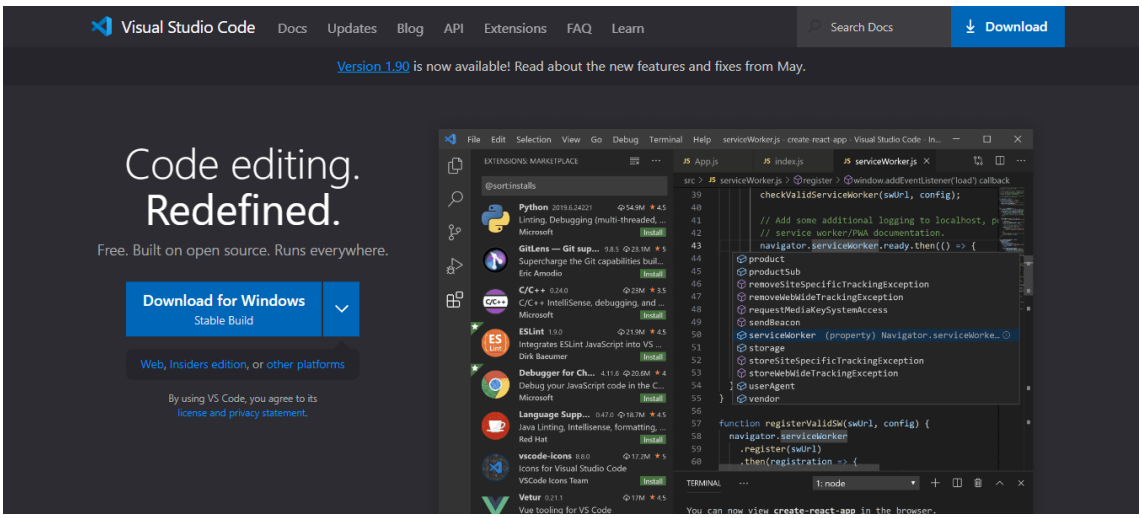


Imagen 1: Sitio web Visual Studio Code.

Fuente: Microsoft (2024)

- Docker, para el manejo de contenedores en donde se alojará la aplicación web para poder ser consultada, el cual puede ser descargado de manera gratuita en el sitio https://www.docker.com/.

Imagen 2: Sitio web Docker Inc

Fuente: Docker Inc (2024)

- GitHub, para el manejo de control de versiones, clonación de repositorios existentes y manipulación de código fuente.

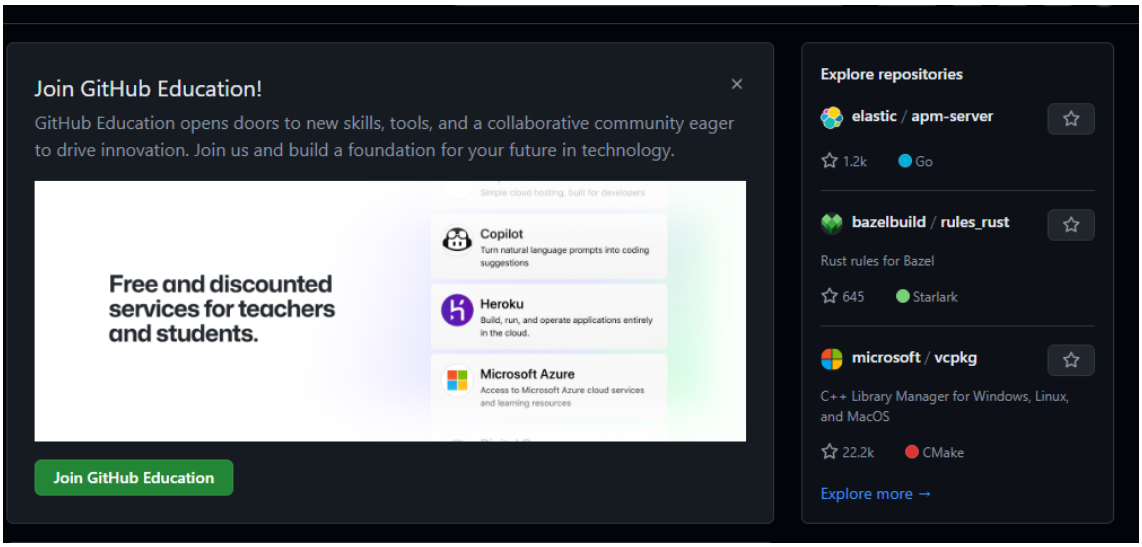


Imagen 3: Sitio web GitHub

Fuente: GitHub (2024)

Para la aplicación móvil:

- Android Studio, como un entorno de desarrollo de Android, en su versión más reciente, la cual puede ser descargada de forma gratuita en el sitio https://developer.android.com/studio?hl=es-419.

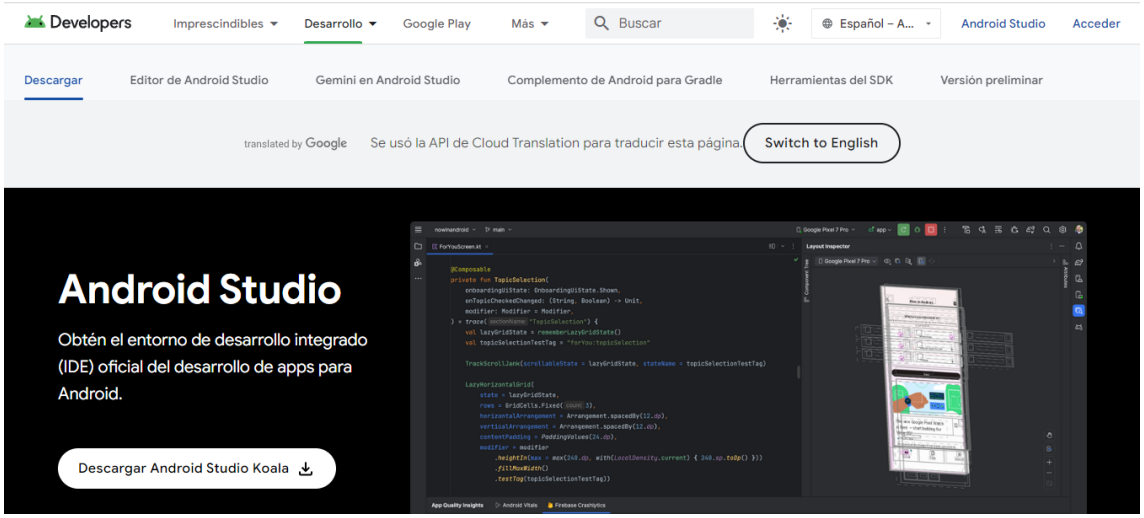


Imagen 4: Sitio web Android Studio

Fuente: Android Studio (2024)

Para la ejecución del código es necesario llevar a cabo la clonación de los repositorios tanto de la aplicación web como móvil, siendo Visual Studio Code el entorno de desarrollo para la aplicación web y Android Studio para la aplicación móvil.

EJECUCIÓN EN LOCAL PARA APLICACIÓN WEB

1. Levantar docker compose

   docker compose -f compose-dev.yml up –build -d

2. Cargar los datos en la base

   Ejecutar el contenedor del backend

   docker exec -it <nombre_contenedor> sh

   Poblar la base de datos

   node seed/seeder.js -i

   node utils/crearDatos.js -i

Nota: Esta acción se debe realizar únicamente una vez en el contenedor del backend.

Al ejecutar la aplicación web se recibe el siguiente objeto:

```
{
createdAt: fecha en formato UTC (string),
dispositivo_id: id del dispositivo (string),
dispositivo_lectura_intervalo: cantidad de segundos para una nueva medida (number),
lecturasRecientes: [
                        {
                            createdAt: {
                                    hora: hora de captura de la medida (string),
                                    fecha: fecha de captura de la medida (string)
                            },
                            dispositivo_id: id del dispositivo al que pertenece la lectura
                                    (string),
                            lectura_id: id de la lectura (number),
                            lectura_valor: valor registrado (number)
                            }
                    ],
medicion: {
        información sobre el fenómeno para el que está diseñado el dispositivo, como el
        siguiente objeto:
        medicion_descripcion: descripción del fenómeno que se mide (string),
        medicion_fenomeno: nombre del fenómeno que se mide (string),
        medicion_unidad: unidad de medida para el fenómeno (string),

        medicion_unidad_abreviatura: abreviatura para la unidad (string o null)
        },
medicion_id: id de la medicion asignada al dispositivo (number),
ubicacion: {
        información sobre la ubicación donde se coloque el dispositivo, como el
        siguiente objeto:
        ubicacion_nombre: nombre de la ubicación (string),
        ubicacion_tipo: tipo del entorno de la ubicación (string)
        }
ubicacion_id: id de la ubicación asignada al dispositivo (number)

}
```

EJECUCIÓN EN LOCAL APLICACIÓN MOVIL

Para ejecutar la aplicación móvil desde la plataforma de Android Studio, basta con hacer clic en el ícono “run”, ubicado en la parte superior central de la pantalla.

Esto realizará la ejecución del proyecto y la construcción de la aplicación para ser visualizada desde el emulador propio de la plataforma o de un celular externo conectado vía Wi-Fi o mediante un puerto USB.

MANTENIMIENTO DE BASE DE DATOS

- Para comprobar el comportamiento de los datos en el desarrollo local o en caso de iniciar nuevamente con una base de datos, se deberá ejecutar el comando para el llenado de la tabla con datos necesarios para la tabla de dispositivos.

  node seed/seeder.js -i

- En caso de no poder acceder a datos reales, es posible crear datos de prueba por medio de la siguiente función, que seguirá insertando datos hasta detenerla manualmente.

  node utils/crearDatosPrueba.js -i

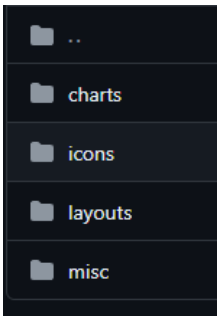
- La opción de borrar la base de datos debe ejecutarse **ÚNICAMENTE** en entorno local.

  node seed/seeder.js -d

- Evitar ejecutar los comandos de datos de prueba con la versión del sistema en producción, ya que llenara la tabla con datos falseados.

MANTENIMIENTO DE APLICACIÓN WEB

Para realizar mantenimiento de la interfaz, en el repositorio correspondiente al frontend de la aplicación se cuenta con la carpeta src, la cual aloja carpetas con el código fuente que controla la vista de la aplicación. Cada función utilizada se encuentra detallada con comentarios que permiten conocer el objetivo de estas.



Cada carpeta se divide en subcarpetas que contienen los componentes cargados en pantalla, funcionando de acuerdo al formato del framework utilizado (Vue.js). Para su manipulación, considerar el formato de creación de componentes, funciones y peticiones.

```
// Función para aleatorizar el tipo de gráfico para cada dispositivo
const crearGraficosAleatoriosNumeros = (cantidadDispositivos) => {
  let numerosAleatorios = [];

  // Se generan números aleatorios entre 0 y la cantidad de gráficos existentes
  for (let i = 0; i < cantidadDispositivos; i++) {
    numerosAleatorios.push(
      Math.floor(Math.random() * graficosDisponibles.value.length)
    );
  }
  return numerosAleatorios;
};
```

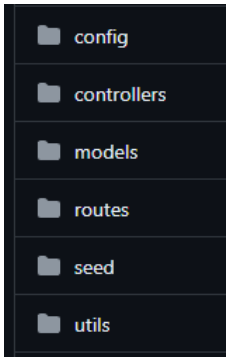
Formato para una función.

```
<div id="graficos-container">
  <div v-for="(dispositivo, index) in dispositivos" class="grafico-div">
    <BButton
      variant="outline-info"
      size="sm"
      :to="{
        name: 'grafico-detallado',
        params: { dispositivoId: dispositivo.dispositivo_id },
      }"
      >Más información</BButton
    >
    <component
      :is="graficosDisponibles[graficosAleatoriosNumeros[index]]"
      :dispositivo="dispositivo"
    />
```

Formato para componentes visuales

Si se desea añadir o modificar iconos y gráficos para el renderizado de datos, pueden ser añadidos en su carpeta asignada, siempre dentro de la carpeta source (src).

Por otro lado, para realizar mantenimiento de la API se deberá trabajar sobre el repositorio del backend, el cual cuenta en su carpeta principal con las subcarpetas que manejan el funcionamiento, consulta y envío de datos al frontend.



- Config: Para realizar la configuración de la conexión de la base de datos, manipular para la actualización de base de datos (base de datos nueva).

- Controllers: Para la creación de métodos o funciones de consumo y manipulación de datos, modificar para la adición de nuevas consultas de datos.

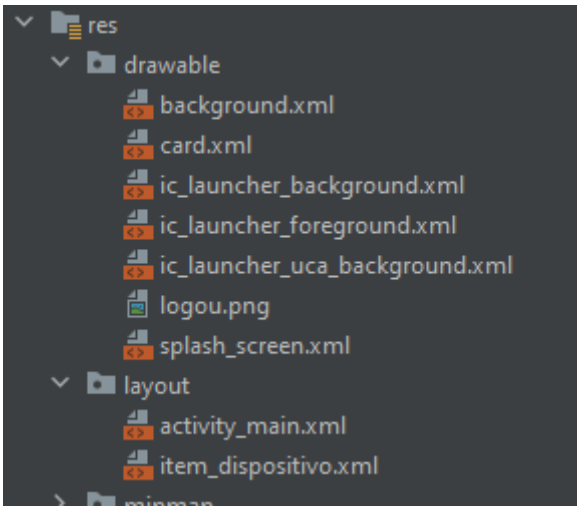
- Models: Para manejar los modelos e instancias, manipular para la adición o actualización de atributos de las tablas de base de datos.

- Routes: Definición de rutas de acceso desde el frontend con la API, manipular para agregar o modificar nuevas rutas de consulta.

- Seed y utils: Configuración de datos de prueba, manipular solo para ambiente local.

MANTENIMIENTO DE APLICACIÓN MOVIL

Para realizar mantenimiento de la parte visual (interfaz) se deben modificar los archivos siguientes:



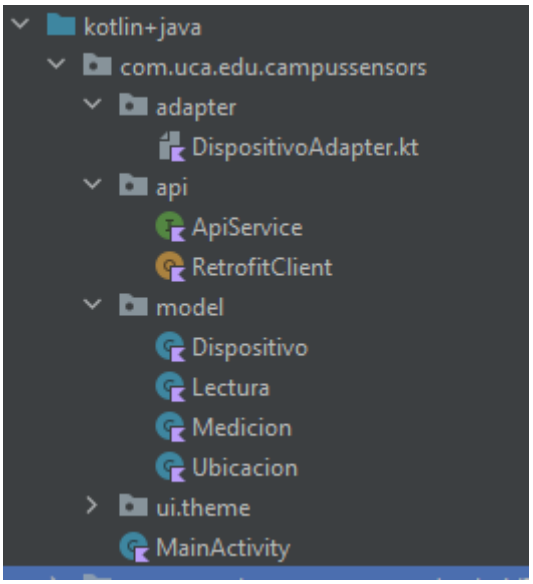
Los cuales proporcionan la estructura visual de la aplicación, siendo la carpeta "**drawable**" la encargada de manejar los componentes específicos de la aplicación y la carpeta "**layout**" la encargada de renderizar la vista general

Siguiendo la estructura básica de kotlin.

```
<RelativeLayout
    android:id="@+id/relativeLayout"
    android:layout_width="0dp"
    android:layout_height="70dp"
    android:background="@drawable/background"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/recyclerView">

    <ImageView
        android:id="@+id/image"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:src="@drawable/logou" />
</RelativeLayout>
```

Para realizar el mantenimiento del backend dentro de la aplicación móvil, los archivos enfocados a la funcionabilidad de la data dentro de la aplicación son los siguientes:



Siendo el detalle de estas carpetas el siguiente:

- Adapter: Manejo de la data obtenida y mostrada en la interfaz, manipular para la adición o sustracción de campos a mostrar.

- Api: Medio para el tráfico de datos desde la base hasta la vista principal, manipular para el cambio o adición de consulta y tipo de consulta a realizar.

- Model: Definición de tablas de la base de datos para el manejo adecuado de los datos obtenidos, manipular para la modificación de campos de la base de datos, adición de nuevas tablas, eliminación de estas o de campos actuales.

- Ui.theme: Manejo de renderización de la vista, manipular para cambiar la orden y visualización de los datos así como, manejar las funciones a ejecutarse.

REQUERIMIENTOS OPERATIVOS

Para un mejor funcionamiento de las herramientas anteriormente mencionadas se hace la recomendación de los requerimientos del sistema siguientes:

REQUISITOS MINIMOS DEL SISTEMA

- ✓ Sistema Operativo Microsoft® Windows® 8/10/11 de 64 bits
- ✓ Procesador Intel Core de segunda generación o posterior
- ✓ 8GB de Memoria RAM
- ✓ 8GB de Disco Duro
- ✓ Resolución de pantalla mínima de 1280 × 800

BIBLIOGRAFIA

Aplicaciones Empresariales (2022, 30 marzo) ¿Qué es Kotlin y para qué sirve? Recuperado el 30 de junio de 2024 de https://www.plainconcepts.com/es/kotlin-android/

Fernández. Y. (2019, 30 octubre) Qué es Github y qué es lo que le ofrece a los desarrolladores. Recuperado el 30 de junio de 2024 de https://www.xataka.com/basics/que-github-que-que-le-ofrece-a-desarrolladores

IBM (s. f.) ¿Qué es PostgreSQL? Recuperado el 30 de junio de 2024 de https://www.ibm.com/mx-es/topics/postgresql

RedHat (2023, 20 enero) ¿Qué es Docker y como funciona? Recuperado el 30 de junio de 2024 de https://www.redhat.com/es/topics/containers/what-is-docker

Sufiyan. T. (2024, 26 junio) What Is Node.js? A Complete Guide for Developers. Recuperado el 30 de junio de 2024 de https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs

Vue JS (s. f.) ¿Qué es Vue.js? Recuperado el 30 de junio de 2024 de https://es.vuejs.org/v2/guide/