MANUAL TÉCNICO



PROYECTO:

APLICACIÓN MÓVIL PARA EL SEGUIMIENTO MUNDIAL DE CASOS DE COVID-19 CON DATOS EN TIEMPO REAL

INTEGRANTES:

* PILCO QUISPE, Mireya Flavia
* SALAMANCA CONTRERAS, Fiorella Rosmery
* ZAVALA VENEGAS, Luis Ángel

TACNA - PERÚ

2020

# HISTORIAL DE VERSIONES

|  |  |  |  |
| --- | --- | --- | --- |
| ***Fecha*** | ***Versión*** | ***Descripción*** | ***Autor*** |
| 24/06/2020 | 1.0 | Creación del documento | MP, FS, LZ |

**ÍNDICE**

[HISTORIAL DE VERSIONES 1](#_Toc44416623)

[MANUAL TÉCNICO 3](#_Toc44416624)

[2. JUSTIFICACIÓN 4](#_Toc44416625)

[3. OBJETIVO 4](#_Toc44416626)

[4. ANÁLISIS 5](#_Toc44416627)

[5. DISEÑO 8](#_Toc44416628)

[6. PROGRAMAS 14](#_Toc44416629)

[Repository 18](#_Toc44416630)

[Branch 18](#_Toc44416631)

[Pull Request 18](#_Toc44416632)

[7. PRUEBAS 26](#_Toc44416633)

# MANUAL TÉCNICO

1. **DELIMITACIÓN**

La Aplicación Móvil para el Seguimiento Mundial de Casos de Covid-19 con Datos en Tiempo Real, las siguientes convenciones:

* Mostrar los datos estadísticos en tiempo real de los casos positivos, fallecimiento y recuperados a nivel mundial.
* Mostrar los datos estadísticos en tiempo real de los casos positivos, casos nuevos por día, número de fallecidos, casos nuevos de fallecidos por día, recuperados, casos activos y total de casos críticos por país.
* Mostrar información vital sobre el Covid-19 de fuentes importantes, revelando la población vulnerable, síntomas, precauciones, medidas preventivas sobre el Covid-19 y población vulnerable.
* Mostrar un guía de emergencia para el usuario, se podrá visualizar información de la propagación del virus, formas de transmisión, prevención y números de emergencia por localidad a nivel Perú.
* Realizar un autodiagnóstico referencial para detectar un posible caso infecto del Covid-19, mostrando el resultado del diagnóstico.
* Reportar posibles casos nuevos y donaciones por localidades del Perú.
* Gestionar los casos reportados y donaciones registradas mediante la aplicación, de manera que se pueda llevar una estadística de su uso.
* Autenticar usuario para brindar el acceso solo al administrador de la aplicación.

La Aplicación Móvil para el Seguimiento Mundial de Casos de Covid-19 con Datos en Tiempo Real, no contempla las siguientes convenciones:

* Mostrar el seguimiento de infectados alrededor del usuario de la aplicación móvil.
* Mostrar información y seguimientos de las áreas consideradas como foco de contagio en la localidad del usuario de la aplicación.
* La aplicación está desarrollada solo para el sistema operativo Android
* La aplicación móvil no mostrará el seguimiento de las donaciones o reporte de casos fuera de contexto de la aplicación, solo a nivel nacional.
* La aplicación móvil no se debería considerar como una fuente verídica en el caso del autodiagnóstico, está sujeto a observaciones.

## JUSTIFICACIÓN

La Aplicación Móvil para el Seguimiento Mundial de Casos de Covid-19 con Datos en Tiempo Real se realiza debido a la demanda de desarrollar una solución tecnológica que permita conocer sobre las estadísticas de los casos de infectados e información general del Covid-19 y los muchos casos de necesidad de pedir donaciones frente a esta crisis, siendo una forma tecnológica e informativa al alcance de los usuarios. Los teléfonos inteligentes están al alcance de la mayoría de las personas de una sociedad, por ello se pretende realizar una aplicación móvil que, de manera útil, permita a los usuarios acceder a esta información y hacer el seguimiento de los casos confirmados, fallecimientos y recuperados de los casos de Covid-19 en el mundo e información generada por cada país; mostrando información sobre la pandemia de fuentes de gran relevancia, mostrando una guía de emergencia, realización de autodiagnósticos, y a su vez permitiendo el reporte de casos y donaciones por localidades del Perú.

## OBJETIVO

La Aplicación Móvil para el Seguimiento Mundial de Casos de Covid-19 con Datos en Tiempo Real tiene como objetivo:

* Realizar una herramienta para el seguimiento de los casos confirmados, fallecimientos y recuperados de los casos de Covid-19 en el mundo, e información genera por cada país; mostrando información sobre la pandemia de fuentes de gran relevancia, mostrando una guía de emergencia, realización de autodiagnósticos, y a su vez permitiendo el reporte de casos y donaciones por localidades del Perú.

## ANÁLISIS

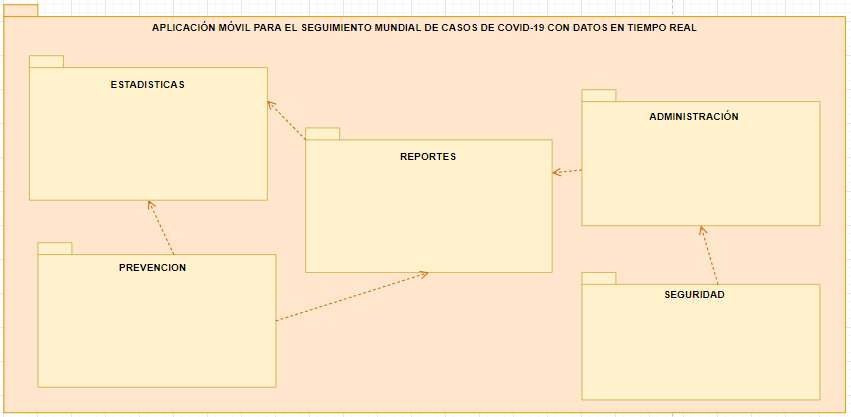
La Aplicación Móvil para el Seguimiento Mundial de Casos de Covid-19 con Datos en Tiempo Real contempla los siguientes requerimientos:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MÓDULO** | **CÓDIGO** | **REQUERIMIENTO** | **DESCRIPCIÓN** | **PRIORIDAD** |
| **ESTADÍSTICAS** | RF-01 | Visualizar estadísticas mundiales | La aplicación mostrará las estadísticas generales mundiales del Covid-19.  Se podrá visualizar el número de contagiados, el número de muertos y el número de recuperados. | ALTO |
| RF-02 | Visualizar estadísticas generales por país | La aplicación mostrará las estadísticas generales de casos por país del Covid-19.  Al seleccionar un país, se podrá visualizar, el total de casos, casos nuevos por día, el número de muertos, casos nuevos de muertos por día, total de recuperados, total casos activos y total de casos críticos. | ALTO |
| **PREVENCIÓN** | RF-03 | Visualizar información del Covid-19 | La aplicación mostrará la información del Covid-19.  Se visualizará información general y síntomas. | MEDIO |
| RF-04 | Visualizar Precauciones del Covid-19 | La aplicación mostrará las precauciones del Covid-19.  Se visualizará las precauciones, medidas preventivas sobre el Covid-19 y población vulnerable., | MEDIO |
| RF-05 | Visualizar guía de emergencia | La aplicación mostrará una guía de emergencia frente al contagio del Covid-19.  Se visualizará información de la propagación del virus, formas de transmisión, prevención y números de emergencia. | MEDIO |
| RF-06 | Realizar un autodiagnóstico | La aplicación permitirá realizar un cuestionario para el autodiagnóstico de un posible caso infectado del Covid-19, permitiendo evaluar y conocer el resultado de un diagnóstico referencial. | MEDIO |
| **REPORTES** | RF-07 | Reportar casos por localidades | El usuario podrá reportar un caso de una persona o de sí mismo.  Se visualizará un formulario que permita obtener los datos del posible infectado, para reportar un posible caso nuevo. | ALTO |
| RF-08 | Reportar donaciones por localidades | El usuario podrá aportar una donación o solicitar una donación de alimentos, productos sanitarios, medicinas o fondos.  La aplicación mostrará un formulario que permita obtener los datos necesarios para la donación. | ALTO |
| **ADMINISTRACIÓN** | RF-09 | Gestión de Casos Reportados | El administrador podrá gestionar los casos que han sido reportados.  Se visualizará una lista de los casos reportados y gestionar su estado (Atendido, En Proceso y No Atendido). | MEDIO |
| RF-10 | Gestión de Donaciones | El administrador podrá gestionar las donaciones que han sido registradas.  Se visualizará una lista de las donaciones registradas y gestionar su estado (Recogido, No Recogido y En Proceso). | MEDIO |
| **SEGURIDAD** | RF-11 | Autenticar Usuario | El administrador podrá iniciar sesión para acceder a la aplicación y realizar la gestión respectiva.  Se requerirá de datos como Usuario y Contraseña. | BAJO |

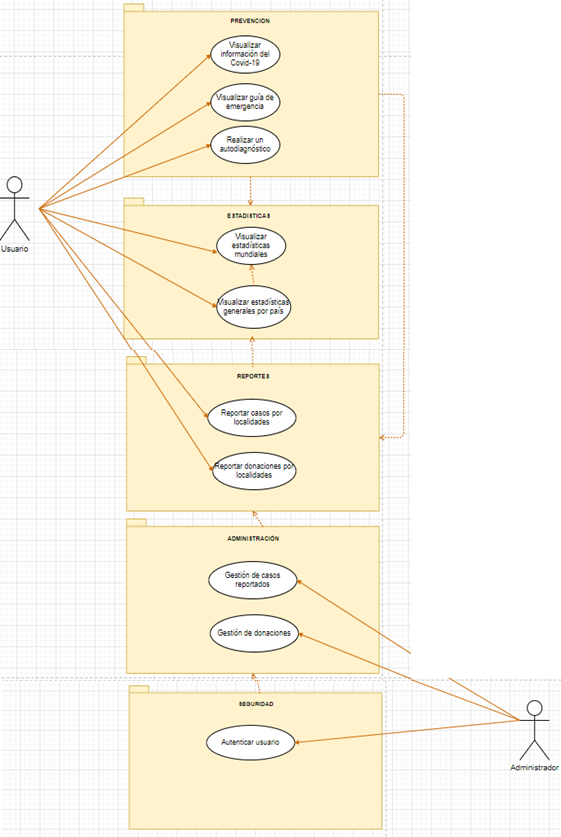
## DISEÑO

La Aplicación Móvil para el Seguimiento Mundial de Casos de Covid-19 con Datos en Tiempo Real contempla lo siguiente:

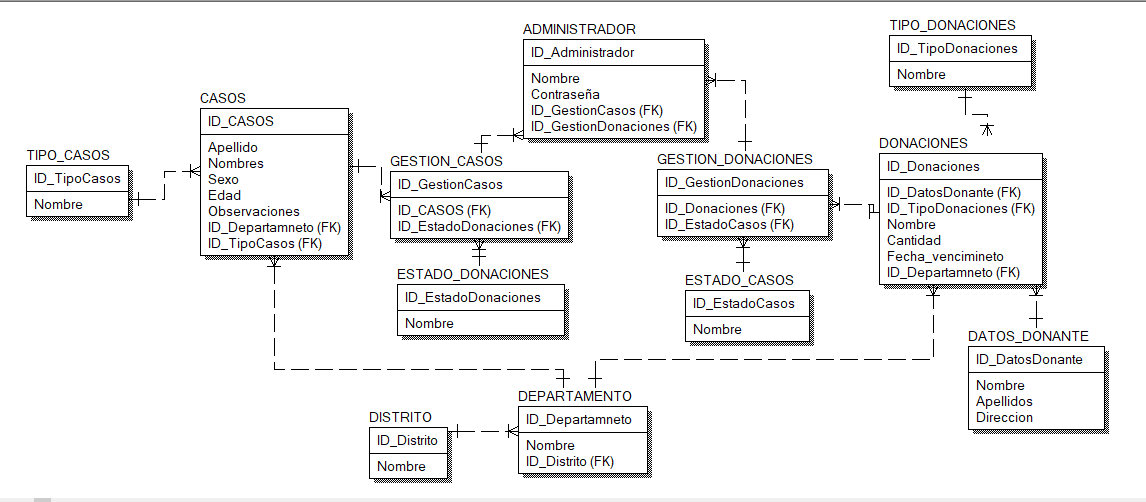
* Diagrama de Paquetes



* Diagrama de Casos de Uso

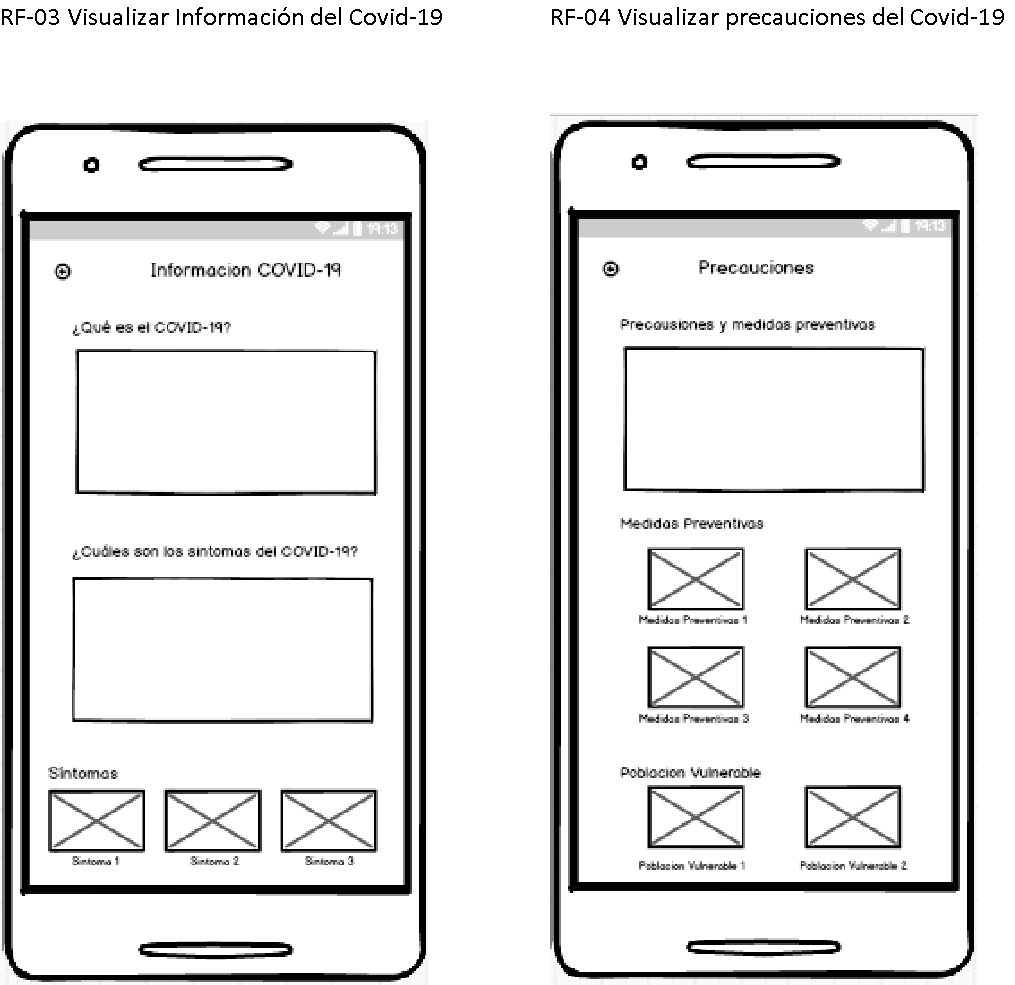


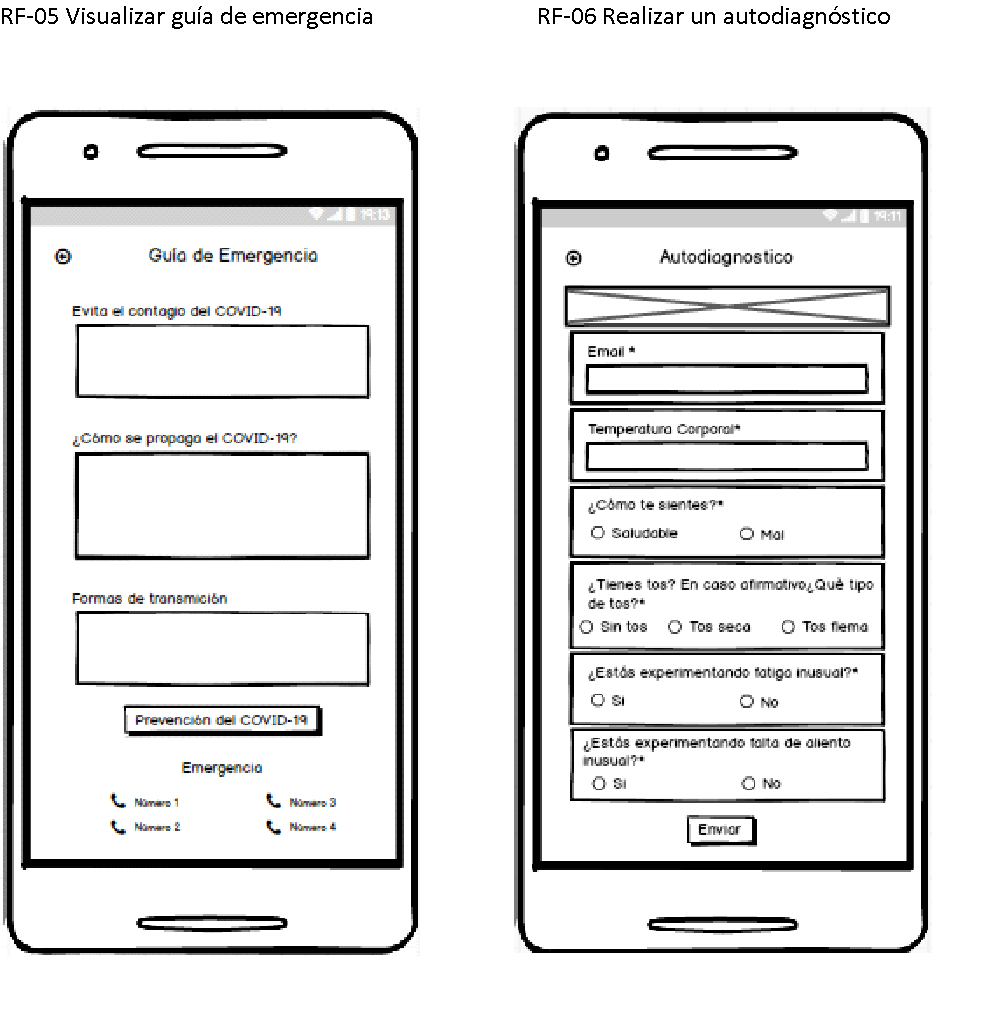
* Diagrama Entidad-Relación

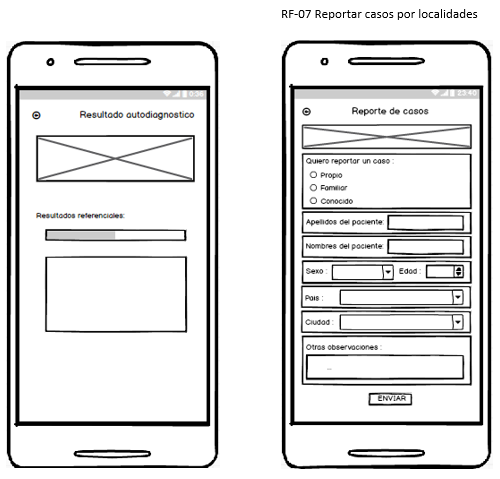


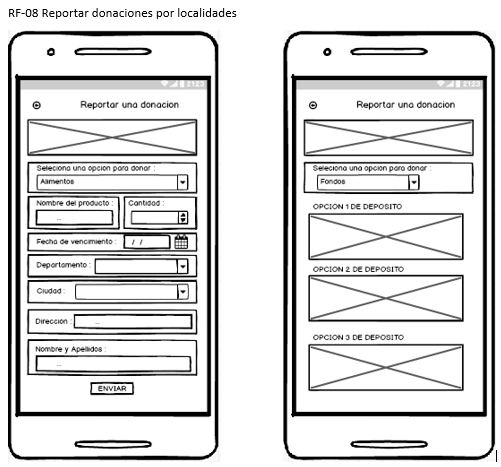
* Prototipos del aplicativo

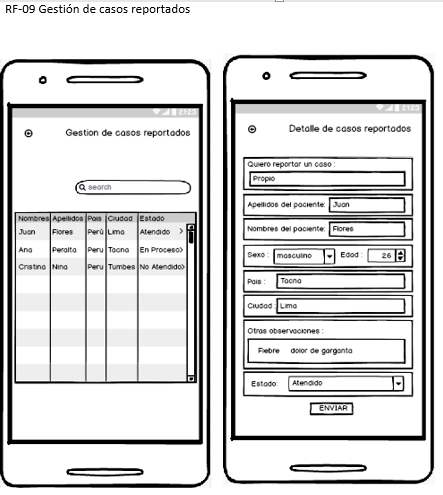




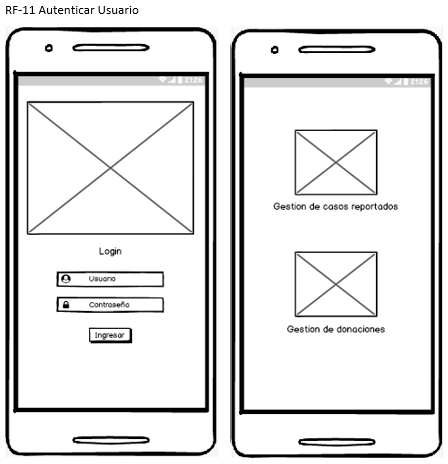












## PROGRAMAS

La Aplicación Móvil para el Seguimiento Mundial de Casos de Covid-19 con Datos en Tiempo Real contempla lo siguiente:

* **Balsamiq:**

Balsamiq es una aplicación/servicio pues no sólo cuenta con una aplicación nativa para OS X (también Windows y Linux) sino también con una versión web, de modo que podemos trabajar desde cualquier lugar. Y su finalidad no es otra que ayudar al desarrollo de aplicaciones con una herramienta que facilita la creación de esquemas.

****

* **Android Studio**

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de apps para Android, basado en IntelliJ IDEA. Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece incluso más funciones que aumentan tu productividad cuando desarrollas apps para Android, como las siguientes:

* Un sistema de compilación flexible basado en Gradle
* Un emulador rápido y cargado de funciones
* Un entorno unificado donde puedes desarrollar para todos los dispositivos Android
* Aplicación de cambios para insertar cambios de códigos y recursos a la aplicación en ejecución sin reiniciar la aplicación
* Integración con GitHub y plantillas de código para ayudarte a compilar funciones de apps comunes y también importar código de muestra
* Variedad de marcos de trabajo y herramientas de prueba
* Herramientas de Lint para identificar problemas de rendimiento, usabilidad y compatibilidad de la versión, entre otros
* Compatibilidad con C++ y NDK
* Compatibilidad integrada con Google Cloud Platform que facilita la integración con Google Cloud Messaging y App Engine.

****

* **Estructura de un proyecto**

Cada proyecto de Android Studio incluye uno o más módulos con archivos de código fuente y archivos de recursos. Entre los tipos de módulos se incluyen los siguientes:

* Módulos de apps para Android
* Módulos de biblioteca
* Módulos de Google App Engine

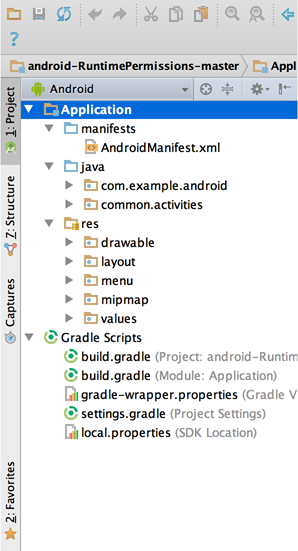
De manera predeterminada, Android Studio muestra los archivos de tu proyecto en la vista de proyecto de Android, como se ve en la Figura 1. Esta vista está organizada en módulos para que puedas acceder rápidamente a los archivos fuente clave de tu proyecto.

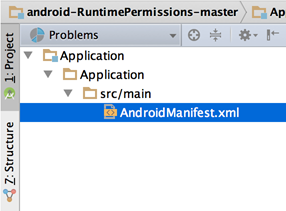
Puedes ver todos los archivos de compilación en el nivel superior de Secuencias de comando de Gradle y cada módulo de app contiene las siguientes carpetas:

* **manifests**: Contiene el archivo AndroidManifest.xml.
* **java:** Contiene los archivos de código fuente Java, incluido el código de prueba de JUnit.
* **res:** Contiene todos los recursos sin código, como diseños XML, strings de IU e imágenes de mapa de bits.

La estructura del proyecto de Android en el disco difiere de esta representación plana. Para ver la estructura de archivos real del proyecto, selecciona Project en el menú desplegable Project (en la figura 1, se muestra como Android).

También puedes personalizar la vista de los archivos del proyecto para concentrarte en aspectos específicos del desarrollo de tu app. Por ejemplo, si seleccionas la vista Problems de tu proyecto, se mostrarán vínculos a los archivos fuente que contengan errores conocidos de codificación y sintaxis, como una etiqueta de cierre que falte para un elemento XML en un archivo de diseño.

****

****

* **Github:**

GitHub es un sistema de gestión de proyectos y control de versiones de código, así como una plataforma de red social diseñada para desarrolladores. GitHub se usa en general, permite trabajar en colaboración con otras personas de todo el mundo, planificar proyectos y realizar un seguimiento del trabajo.

Git es un sistema de control de versiones desarrollado por Linus Torvalds (el hombre que creó Linux).



**Repository**

Un repositorio o “repo” es un directorio donde se almacenan los archivos de tu proyecto. Puede estar ubicado en el almacenamiento de GitHub o en un repositorio local en tu computadora. Puedes almacenar archivos de código, imágenes, audios o todo lo relacionado con el proyecto en el repositorio.

**Branch**

Branch, que se traduce como rama, es una copia de tu repositorio. Puedes utilizar la rama cuando quieras hacer un desarrollo de forma aislada.

Trabajar en una rama no afectará el repositorio central u otras ramas. Si has completado el trabajo, puedes combinar tu rama con otras ramas y con el repositorio central mediante una pull request.

**Pull Request**

Pull request significa que les informas a los demás que has enviado al repositorio principal el cambio que hiciste en una rama. Los colaboradores del repositorio pueden aceptar o rechazar una *pull request*. Una vez abierta, puedes discutir y revisar tu trabajo con los colaboradores.

* **The NovelCOVID API:**

Es una API que nos permite obtener datos en tiempo real de los infectados del Covid-19. Su servicio es de uso gratuito y tiene un puntaje de rendimiento de información de 100/100 páginas. Obtiene datos de los informes de la Universidad Johns Hopkins, el New York Times, Worldometers y Apple para brindar una visión integral de los datos. Lo datos se actualizan cada 10 minutos. Las respuestas se presentan en formato JSON, lo que permite una integración fácil y confiable a su servicio.



* **Coronavirus Tracker:**

Coronavirus Tracker es una API, que permite conocer las estadísticas globales de infectados del Covid-19, de manera gratuita, todas las estadísticas de país, estadísticas de país, línea de tiempo completa, línea de tiempo de país. Puede usar AJAX para llamar a la API gratuita de thevirustracker.com y recibirá datos a cambio. Si se utiliza jQuery, puede usar la función $ .ajax () en el código.

****

* **Codificación de la Clase: class\_CovidPais**

*/\**

*\* @(#)class\_CovidPais.java 1.1 25/06/20*

*\**

*\* UPT*

*\* Construccion de Software II.*

*\*/*

*/\*\**

*\**

*\* @author Luis Zavala*

*\* @version 1.1, 25/06/20*

*\* @since 1.0*

*\*/*

package com.milifi.myappcovid19.ui.country;

import android.os.Parcel;

import android.os.Parcelable;

public class class\_CovidPais implements Parcelable {

String mCovidPais, mCasosHoy, mMuertes, mMuertesHoy, mRecuperados, mActivos, mCriticos, mBanderas;

int mCasos;

public class\_CovidPais(String mCovidPais, int mCasos, String mTodayCases, String mDeaths, String mTodayDeaths, String mRecovered, String mActive, String mCritical, String mFlags) {

this.mCovidPais = mCovidPais;

this.mCasos = mCasos;

this.mCasosHoy = mTodayCases;

this.mMuertes = mDeaths;

this.mMuertesHoy = mTodayDeaths;

this.mRecuperados = mRecovered;

this.mActivos = mActive;

this.mCriticos = mCritical;

this.mBanderas = mFlags;

}

public String getmCovidPais() {

return mCovidPais;

}

public int getmCasos() {

return mCasos;

}

public String getmHoyCasos() {

return mCasosHoy;

}

public String getmMuertes() {

return mMuertes;

}

public String getmHoyMuertes() {

return mMuertesHoy;

}

public String getmRecuperados() {

return mRecuperados;

}

public String getmActivos() {

return mActivos;

}

public String getmCriticos() {

return mCriticos;

}

public String getmBanderas() {

return mBanderas;

}

@Override

public int describeContents() {

return 0;

}

@Override

public void writeToParcel(Parcel dest, int flags) {

dest.writeString(this.mCovidPais);

dest.writeInt(this.mCasos);

dest.writeString(this.mCasosHoy);

dest.writeString(this.mMuertes);

dest.writeString(this.mMuertesHoy);

dest.writeString(this.mRecuperados);

dest.writeString(this.mActivos);

dest.writeString(this.mCriticos);

dest.writeString(this.mBanderas);

}

protected class\_CovidPais(Parcel in) {

this.mCovidPais = in.readString();

this.mCasos = in.readInt();

this.mCasosHoy = in.readString();

this.mMuertes = in.readString();

this.mMuertesHoy = in.readString();

this.mRecuperados = in.readString();

this.mActivos = in.readString();

this.mCriticos = in.readString();

this.mBanderas = in.readString();

}

public static final Creator<class\_CovidPais> *CREATOR* = new Creator<class\_CovidPais>() {

@Override

public class\_CovidPais createFromParcel(Parcel source) {

return new class\_CovidPais(source);

}

@Override

public class\_CovidPais[] newArray(int size) {

return new class\_CovidPais[size];

}

};

}

* **Codificación de la Clase : class\_FragmentoPais**

*/\**

*\* @(#)class\_FragmentoPais.java 1.1 25/06/20*

*\**

*\* UPT*

*\* Construccion de Software II.*

*\*/*

*/\*\**

*\**

*\* @author Luis Zavala*

*\* @version 1.1, 25/06/20*

*\* @since 1.0*

*\*/*

package com.milifi.myappcovid19.ui.country;

import android.content.Intent;

import android.os.Bundle;

import android.util.Log;

import android.view.LayoutInflater;

import android.view.Menu;

import android.view.MenuInflater;

import android.view.MenuItem;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ProgressBar;

import android.widget.Toast;

import androidx.annotation.NonNull;

import androidx.appcompat.widget.SearchView;

import androidx.core.content.ContextCompat;

import androidx.fragment.app.Fragment;

import androidx.recyclerview.widget.DividerItemDecoration;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import com.milifi.myappcovid19.R;

import com.android.volley.Request;

import com.android.volley.Response;

import com.android.volley.VolleyError;

import com.android.volley.toolbox.StringRequest;

import com.android.volley.toolbox.Volley;

import org.json.JSONArray;

import org.json.JSONException;

import org.json.JSONObject;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

public class class\_FragmentoPais extends Fragment {

RecyclerView rvCovidPais;

ProgressBar progressBar;

class\_CovidPaisAdaptador covidPaisAdaptador;

private static final String *TAG* = class\_FragmentoPais.class.getSimpleName();

List<class\_CovidPais> covidPaises;

public View onCreateView(@NonNull LayoutInflater inflater,

ViewGroup container, Bundle savedInstanceState) {

View root = inflater.inflate(R.layout.*fragment\_country*, container, false);

setHasOptionsMenu(true);

*// Llamando vista*

rvCovidPais = root.findViewById(R.id.*rvCovidCountry*);

progressBar = root.findViewById(R.id.*progress\_circular\_country*);

rvCovidPais.setLayoutManager(new LinearLayoutManager(getActivity()));

DividerItemDecoration dividerItemDecoration = new DividerItemDecoration(rvCovidPais.getContext(), DividerItemDecoration.*VERTICAL*);

dividerItemDecoration.setDrawable(ContextCompat.*getDrawable*(getContext(), R.drawable.*line\_divider*));

rvCovidPais.addItemDecoration(dividerItemDecoration);

*//Llamando lista*

covidPaises = new ArrayList<>();

*// Llamando metodo voley*

getDataFromServerSortTotalCases();

return root;

}

private void showRecyclerView() {

covidPaisAdaptador = new class\_CovidPaisAdaptador(covidPaises, getActivity());

rvCovidPais.setAdapter(covidPaisAdaptador);

ItemClickSupport.*addTo*(rvCovidPais).setOnItemClickListener(new ItemClickSupport.OnItemClickListener() {

@Override

public void onItemClicked(RecyclerView recyclerView, int position, View v) {

mostrarCovidPaisSeleccionado(covidPaises.get(position));

}

});

}

private void mostrarCovidPaisSeleccionado(class\_CovidPais classCovidPais) {

Intent covidCovidCountryDetail = new Intent(getActivity(), class\_CovidPaisDetalle.class);

covidCovidCountryDetail.putExtra("EXTRA\_COVID", classCovidPais);

startActivity(covidCovidCountryDetail);

}

private void getDataFromServerSortTotalCases() {

String url = "https://corona.lmao.ninja/v2/countries";

StringRequest stringRequest = new StringRequest(Request.Method.*GET*, url, new Response.Listener<String>() {

@Override

public void onResponse(String response) {

progressBar.setVisibility(View.*GONE*);

if (response != null) {

Log.*e*(*TAG*, "onResponse: " + response);

try {

JSONArray jsonArray = new JSONArray(response);

for (int i = 0; i < jsonArray.length(); i++) {

JSONObject data = jsonArray.getJSONObject(i);

*// Extract JSONObject inside JSONObject*

JSONObject countryInfo = data.getJSONObject("countryInfo");

covidPaises.add(new class\_CovidPais(

data.getString("country"), data.getInt("cases"),

data.getString("todayCases"), data.getString("deaths"),

data.getString("todayDeaths"), data.getString("recovered"),

data.getString("active"), data.getString("critical"),

countryInfo.getString("flag")

));

}

*// ordenando descendentemente*

Collections.*sort*(covidPaises, new Comparator<class\_CovidPais>() {

@Override

public int compare(class\_CovidPais o1, class\_CovidPais o2) {

if (o1.getmCasos() > o2.getmCasos()) {

return -1;

} else {

return 1;

}

}

});

*// Action Bar Title*

getActivity().setTitle(jsonArray.length() + " Paises");

showRecyclerView();

} catch (JSONException e) {

e.printStackTrace();

}

}

}

},

new Response.ErrorListener() {

@Override

public void onErrorResponse(VolleyError error) {

progressBar.setVisibility(View.GONE);

Log.e(TAG, "onResponse: " + error);

}

});

Volley.newRequestQueue(getActivity()).add(stringRequest);

}

private void getDataFromServerSortAlphabet() {

String url = "https://corona.lmao.ninja/v2/countries";

StringRequest stringRequest = new StringRequest(Request.Method.GET, url, new Response.Listener<String>() {

@Override

public void onResponse(String response) {

progressBar.setVisibility(View.GONE);

if (response != null) {

Log.e(TAG, "onResponse: " + response);

try {

JSONArray jsonArray = new JSONArray(response);

for (int i = 0; i < jsonArray.length(); i++) {

JSONObject data = jsonArray.getJSONObject(i);

*// Extraer JSONObject dentro JSONObject*

JSONObject countryInfo = data.getJSONObject("countryInfo");

covidPaises.add(new class\_CovidPais(

data.getString("country"), data.getInt("cases"),

data.getString("todayCases"), data.getString("deaths"),

data.getString("todayDeaths"), data.getString("recovered"),

data.getString("active"), data.getString("critical"),

countryInfo.getString("flag")

));

}

getActivity().setTitle(jsonArray.length() + " Paises");

showRecyclerView();

} catch (JSONException e) {

e.printStackTrace();

}

}

}

},

new Response.ErrorListener() {

@Override

public void onErrorResponse(VolleyError error) {

progressBar.setVisibility(View.GONE);

Log.e(TAG, "onResponse: " + error);

}

});

Volley.newRequestQueue(getActivity()).add(stringRequest);

}

@Override

public void onCreateOptionsMenu(@NonNull Menu menu, @NonNull MenuInflater inflater) {

inflater.inflate(R.menu.country\_menu, menu);

MenuItem searchItem = menu.findItem(R.id.action\_search);

SearchView searchView = new SearchView(getActivity());

searchView.setQueryHint("Buscar...");

searchView.setMaxWidth(Integer.MAX\_VALUE);

searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {

@Override

public boolean onQueryTextSubmit(String query) {

return false;

}

@Override

public boolean onQueryTextChange(String newText) {

if (covidPaisAdaptador != null) {

covidPaisAdaptador.getFilter().filter(newText);

}

return true;

}

});

searchItem.setActionView(searchView);

super.onCreateOptionsMenu(menu, inflater);

}

@Override

public boolean onOptionsItemSelected(@NonNull MenuItem item) {

switch (item.getItemId()) {

case R.id.action\_sort\_alpha:

Toast.makeText(getContext(), "Ordenado Alfabeticamente", Toast.LENGTH\_SHORT).show();

covidPaises.clear();

progressBar.setVisibility(View.VISIBLE);

getDataFromServerSortAlphabet();

return true;

case R.id.action\_sort\_cases:

Toast.makeText(getContext(), "Ordenado por Total de Casos", Toast.LENGTH\_SHORT).show();

covidPaises.clear();

progressBar.setVisibility(View.VISIBLE);

getDataFromServerSortTotalCases();

return true;

}

return super.onOptionsItemSelected(item);

}

}

## PRUEBAS

La Aplicación Móvil para el Seguimiento Mundial de Casos de Covid-19 con Datos en Tiempo Real contempla lo siguiente:

* Acta de Pruebas del Requerimiento: RF-01 Visualizar estadísticas mundiales.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| PROYECTO/REQUERIMIENTO PUNTUAL | | | | | | |
| APLICACIÓN MÓVIL PARA EL SEGUIMIENTO MUNDIAL DE CASOS DE COVID-19 CON DATOS EN TIEMPO REAL / RF-01 Visualizar estadísticas mundiales | | | | | | |
|
| EQUIPO DE PRUEBAS | | | | | | |
| Nombre de Usuario | | | **Cargo** | | | |
| *Apellidos y Nombres* | | | *Cargo Funcional* | | | |
| Tipo de Usuario | [ ] Usuario final | [X] Usuario de proceso | | | [ ] Usuario de TI | |
| Agente Calidad |  | | | | | |
| FECHA INICIO | 30/06/2020 0:00:00 | | **FECHA FIN** | 02/07/2020 0:00:00 | | |
| 1. OBJETIVOS DE LAS PRUEBAS | | | | | | |
| Comprobar la efectividad de la herramienta usada (API) mediante el resultado obtenido de las estadísticas mundiales de casos contagiados, el número de muertos y el número de recuperados del Covid-19. | | | | | | |
|
| 2. OPCIONES PROBADAS | | | | | | |
| * Probar la conexión con la herramienta utilizada (API) * Comprobar los datos estadisticos son correctos (casos, recuperados y muertes) * Comprobar la funcionalidad del requerimiento funcional con las acciones. | | | | | | |
|
| 3. CONCUSIONES | | | | | | |
| Opciones | | | | | | **Marcar(X)** |
| Dejo Constancia de Conformidad con el Sistema (Opciones / Modificaciones) probado(as) y Solicito su pase a producción. | | | | | | [ ] |
|
| Manifiesto mi inconformidad con el Sistema (Opciones / Modificaciones) probado(as) y solicito se levanten las observaciones planteadas en este documento. | | | | | | [ ] |
|
| FIRMAS |  | | | | | |
| *Firma del usuario con sello* | | | | | | |
|
|
|
|

* Acta de Pruebas del Requerimiento: RF-02 Visualizar estadísticas generales por país

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| PROYECTO/REQUERIMIENTO PUNTUAL | | | | | | |
| APLICACIÓN MÓVIL PARA EL SEGUIMIENTO MUNDIAL DE CASOS DE COVID-19 CON DATOS EN TIEMPO REAL / RF-02 Visualizar estadísticas generales por país | | | | | | |
|
| EQUIPO DE PRUEBAS | | | | | | |
| Nombre de Usuario | | | **Cargo** | | | |
| *Apellidos y Nombres* | | | *Cargo Funcional* | | | |
| Tipo de Usuario | [ ] Usuario final | [X] Usuario de proceso | | | [ ] Usuario de TI | |
| Agente Calidad |  | | | | | |
| FECHA INICIO | 30/06/2020 0:00:00 | | **FECHA FIN** | 02/07/2020 0:00:00 | | |
| 1. OBJETIVOS DE LAS PRUEBAS | | | | | | |
| Comprobar la efectividad de la herramienta usada (API) mediante el resultado obtenido de las estadísticas generales por país de casos de Covid-19, el total de casos, casos nuevos por día, el número de muertos, casos nuevos de muertos por día, total de recuperados, total casos activos y total de casos críticos. | | | | | | |
|
| 2. OPCIONES PROBADAS | | | | | | |
| * Probar la conexión con la herramienta utilizada (API) * Comprobar los datos estadisticos son correctos (total de casos, casos nuevos por día, número de muertos, casos nuevos de muertos por día, total de recuperados, total de recuperados, total de casos activos y total de casos critticos) * Comprobar la funcionalidad del requerimiento funcional con las acciones. | | | | | | |
|
| 3. CONCUSIONES | | | | | | |
| Opciones | | | | | | **Marcar(X)** |
| Dejo Constancia de Conformidad con el Sistema (Opciones / Modificaciones) probado(as) y Solicito su pase a producción. | | | | | | [ ] |
|
| Manifiesto mi inconformidad con el Sistema (Opciones / Modificaciones) probado(as) y solicito se levanten las observaciones planteadas en este documento. | | | | | | [ ] |
|
| FIRMAS |  | | | | | |
| *Firma del usuario con sello* | | | | | | |
|
|
|
|