

**DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN**

**Carrera de Ingeniería en Sistemas e Informática**

**PROGRAMACIÓN MÓVIL**

Martínez Carlos

Monteros María Paula

NRC: 6112

FECHA: 10 - septiembre – 2020

SEMESTRE: May 2020 - Sep 2020

**PROYECTO MASCOTAS PERDIDAS**

**Contenido**

[**OBJETIVO 3**](#_Toc47388824)

[**MARCO TEÓRICO 3**](#_Toc47388825)

[**Firebase 3**](#_Toc47388826)

[**Patrón M-V-P (Modelo-Vista-Presentador) 4**](#_Toc47388827)

[**Diferencia entre MVC y MVP 5**](#_Toc47388828)

[**Capas del patrón MVP 5**](#_Toc47388829)

[**MVP en Android 5**](#_Toc47388830)

[**DESARROLLO 6**](#_Toc47388831)

[**CONCLUSIONES 36**](#_Toc47388832)

[**ANEXO N°1: Modelo 37**](#_Toc47388833)

[**ANEXO N°2: Requisitos 38**](#_Toc47388834)

[**ANEXO N°3: Unit Test 40**](#_Toc47388835)

[**ANEXO N°4: JavaDoc 42**](#_Toc47388836)

[**ANEXO N°5: Código 43**](#_Toc47388837)

OBJETIVO

* Desarrollar una aplicación para dispositivos móviles que permita publicar mascotas perdidas, reportar una mascota herida o publicar una adopción indicando el sector, una breve descripción y dar un número de contacto, mediante la herramienta de Android Studio, el lenguaje de programación java y su integración con Firebase para entender y aplicar la arquitectura de desarrollo MVP (Modelo Vista Presentador) y persistencia de datos en programación móvil.

MARCO TEÓRICO

## **Firebase**

Firebase es una plataforma para el desarrollo de aplicaciones web y aplicaciones móviles desarrollada por Google en 2014. Es una plataforma ubicada en la nube, integrada con Google Cloud Platform, que usa un conjunto de herramientas para la creación y sincronización de proyectos que serán dotados de alta calidad, haciendo posible el crecimiento del número de usuarios y dando resultado también a la obtención de una mayor monetización.

Ofrece ventajas como:

* Sincronizar fácilmente los datos de sus proyectos sin tener que administrar conexiones o escribir lógica de sincronización compleja.
* Usa un conjunto de herramientas multiplataforma: se integra fácilmente para plataformas web como en aplicaciones móviles. Es compatible con grandes plataformas, como IOS, Android, aplicaciones web, Unity y C++.
* Usa la infraestructura de Google y escala automáticamente para cualquier tipo de aplicación, desde las más pequeñas hasta las más potentes.
* Crea proyectos sin necesidad de un servidor: Las herramientas se incluyen en los SDK para los dispositivos móviles y web, por lo que no es necesario la creación de un servidor para el proyecto.

Firebase dota a sus usuarios de una gran documentación para crear aplicaciones usando esta plataforma. Aparte de esto, ofrece soporte gratuito mediante correo electrónico para todos sus usuarios, y además sus desarrolladores participan activamente en plataformas como Github y StackOverflow.

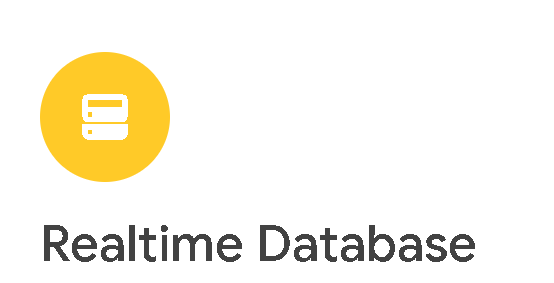


Entre sus productos más usados se encuentra:

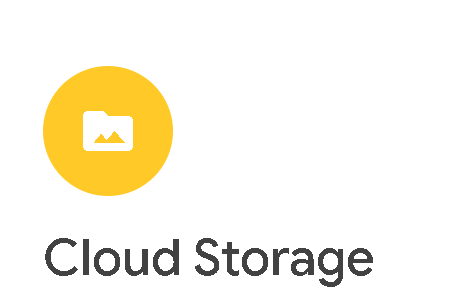
* **Authentication**: Administra tus usuarios de manera simple y segura. Firebase Auth ofrece varios métodos de autenticación (por ejemplo, mediante correo electrónico y contraseña) proveedores externos como Google o Facebook, o tu sistema de cuentas existente de manera directa. Crea tu propia interfaz o aprovecha nuestra IU de código abierto y totalmente personalizable.



* **RealTime Database:** Realtime Database es la base de datos original de Firebase. Esta eficiente solución de baja latencia para aplicaciones móviles exige que el estado de los clientes permanezca sincronizado en tiempo real. Recomendamos Cloud Firestore en lugar de Realtime Database para casi todos los desarrolladores que comienzan un proyecto nuevo.

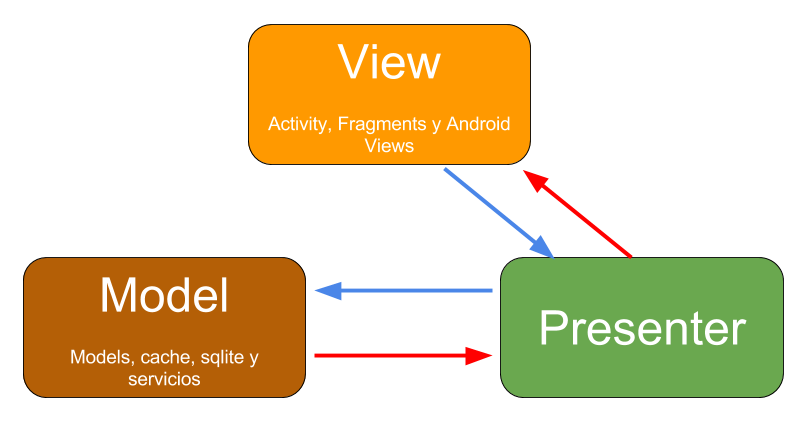


* **Cloud Storage:** Almacena y comparte contenido generado por usuarios, como imágenes, audio y video, con el potente, simple y rentable almacenamiento de objetos creado para la escala de Google. Los SDK de Firebase para Cloud Storage agregan la seguridad de Google a las operaciones de carga y descarga de archivos para tus apps de Firebase, sin importar la calidad de la red



## **Patrón M-V-P (Modelo-Vista-Presentador)**

Modelo–Vista–Presentador (MVP) es una derivación del patrón arquitectónico modelo–vista–controlador (MVC), y es utilizado mayoritariamente para construir interfaces de usuario. En MVP el presentador asume la funcionalidad del "medio-hombre". En MVP, toda lógica de presentación es colocada al presentador.



### **Diferencia entre MVC y MVP**

Ambos modelos parecen bastante similares, pero ofrecen importantes diferencias:

* En el MVC, el modelo notifica a la vista cualquier cambio que sufra el estado del modelo. La información puede pasarse en la propia notificación, o después de la notificación, la vista puede consultar el modelo directamente para obtener los datos actualizados. Por el contrario, en el MVP, la vista no sabe nada sobre el modelo y la función del presentador es la de mediar entre ambos, enlazando los datos con la vista.
* En el modelo MVC, la vista tiende a tener más lógica porque es responsable de manejar las notificaciones del modelo y de procesar los datos. En el modelo MVP, esa lógica se encuentra en el presentador, haciendo a la vista "estúpida".  Su única función es representar la información que el presentador le ha proporcionado.
* En MVC, el modelo tiene lógica extra para interactuar con la vista. En el MVP, esta lógica se encontraría en el presentador.

### **Capas del patrón MVP**

* **Modelo**: Esta capa gestiona los datos. Son las clases que denominaríamos de lógica de negocio.
* **Vista**: Se encarga de mostrar los datos. Aquí se encontrarían nuestros Fragmentos y Vistas.
* **Presentador**: Se sitúa entre el modelo y la vista, permitiendo conectar la interfaz gráfica con los datos.

### **MVP en Android**

Android no nos ofrece de forma nativa la posibilidad de desarrollar nuestras aplicaciones bajo el patrón MVP, de hecho, viola mucho de sus principios básicos. Aun así, podemos llevar a cabo alguna aproximación para este fin. Vamos a ver un posible ejemplo de implementación en el que el usuario dispone de un formulario dónde puede introducir contactos.

* El usuario introduce un contacto y pulsa el botón "añadir contacto".
* El Presentador crea el objeto Contacto con los datos introducidos por el usuario y se lo pasa al Modelo para que lo introduzca en la base de datos.
* El Modelo inserta el contacto en la base de datos.
* Si todo ha ido bien, el Presentador limpia el formulario y refresca la lista de contactos para que aparezca el que acaba de añadir. En caso de que se haya producido algún error, muestra una alerta con un mensaje de error.

La comunicación entre las capas se va a llevar a cabo mediante el uso de interfaces. Para este ejemplo necesitaremos cuatro interfaces.

* ***ProvidedPresenterOps***: Operaciones a las cuales tiene acceso la Vista y son implementadas por el Presentador. Permiten que la Vista se comunique con el Presentador.
* ***ProvidedModelOps***: Operaciones a las cuales tiene acceso el Presentador y son implementadas por el Modelo. Permiten que el Presentador se comunique con el Modelo.
* ***RequiredViewOps***: Operaciones a las cuales tiene acceso el Presentador y son implementadas por la Vista. Permiten que el Presentador se comunique con la Vista.
* ***RequiredPresenterOps***: Operaciones a las cuales tiene acceso el Modelo y son implementadas por el Presentador. Permiten que el Modelo se comunique con el Presentador.

Una vez tenemos definidas las capas de nuestro MVP, necesitamos instanciarlas e insertar las referencias necesarias, además de un mecanismo responsable de mantener el estado del Presentador y del Modelo durante el ciclo de vida de nuestro Fragmento o Actividad. [1]

DESARROLLO

|  |
| --- |
| **AdapterDatos.java** |
| /\*  \* ESPE - DCC - PROGRAMACIÓN MÓVIL  \* NRC: 6112  \*  \* Sistema: Mascotas Perdidas  \* Creado 01/09/2020  \*  \* Los contenidos de este archivo son propiedad privada y estan protegidos por  \* la licencia BSD  \*  \* Se puede utilizar, reproducir o copiar el contenido de este archivo.  \*/  package app.ejemplo.carlos.proyecto2;  import android.content.Context;  import android.content.Intent;  import android.graphics.drawable.Drawable;  import android.support.annotation.NonNull;  import android.support.annotation.Nullable;  import android.support.v7.widget.RecyclerView;  import android.view.LayoutInflater;  import android.view.View;  import android.view.ViewGroup;  import android.widget.ImageView;  import android.widget.LinearLayout;  import android.widget.TextView;  import com.bumptech.glide.Glide;  import com.bumptech.glide.load.DataSource;  import com.bumptech.glide.load.engine.GlideException;  import com.bumptech.glide.request.RequestListener;  import com.bumptech.glide.request.target.Target;  import java.util.ArrayList;  /\*\*  \* Clase que representa a un usuario del chat.  \*  \* @author Carlos Martínez  \* @author Paula Monteros  \*/  public class AdapterDatos extends RecyclerView.Adapter<AdapterDatos.ViewHolderDatos> {  ArrayList<Objeto> list = new ArrayList<>();  ArrayList<Objeto> listOb = new ArrayList<>();  Context context;  public AdapterDatos(ArrayList<Objeto> list, Context context1) {  this.list = list;  this.context=context1;  }  @NonNull  @Override  public AdapterDatos.ViewHolderDatos onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {  View view = LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.item\_list,viewGroup,false);  return new ViewHolderDatos(view);  }  @Override  public void onBindViewHolder(@NonNull final AdapterDatos.ViewHolderDatos viewHolderDatos, final int i) {  viewHolderDatos.nombre.setText(list.get(i).getNombre\_mascota());  viewHolderDatos.lugar.setText(list.get(i).getMapa());  viewHolderDatos.desc.setText(list.get(i).getDesc());  //viewHolderDatos.id.setText(list.get(i).getId());  Glide.with(context).load(list.get(i).getUrl()).listener(new RequestListener<Drawable>() {  @Override  public boolean onLoadFailed(@Nullable GlideException e, Object model, Target<Drawable> target, boolean isFirstResource) {  viewHolderDatos.imagen.setImageResource(R.drawable.ic\_error\_black\_24dp);  return false;  }  @Override  public boolean onResourceReady(Drawable resource, Object model, Target<Drawable> target, DataSource dataSource, boolean isFirstResource) {  viewHolderDatos.imagen.setVisibility(View.VISIBLE);  return false;  }  }).into(viewHolderDatos.imagen);  viewHolderDatos.parentLayout.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  Intent miIntent = new Intent(context,Descripcion.class);  miIntent.putExtra("idObjeto",list.get(i).getId\_mascota());  miIntent.addFlags(miIntent.FLAG\_ACTIVITY\_CLEAR\_TASK|miIntent.FLAG\_ACTIVITY\_NEW\_TASK);  context.startActivity(miIntent);  }  });  }  @Override  public int getItemCount() {  return list.size();  }  public class ViewHolderDatos extends RecyclerView.ViewHolder {  TextView nombre,id;  TextView desc, lugar;  LinearLayout parentLayout;  ImageView imagen;  public ViewHolderDatos(@NonNull View itemView) {  super(itemView);  nombre = itemView.findViewById(R.id.idNombre);  desc = itemView.findViewById(R.id.idInfo);  lugar = itemView.findViewById(R.id.idLugar);  // id=itemView.findViewById(R.id.idObj);  imagen=itemView.findViewById(R.id.imgImgen);  parentLayout=itemView.findViewById(R.id.parent\_layout);  }  }  public void updateList( ArrayList<Objeto> newList){  list = new ArrayList<>();  list.addAll(newList);  notifyDataSetChanged();  }  } |
| **IntroAdapter.java** |
| /\*  \* ESPE - DCC - PROGRAMACIÓN MÓVIL  \* NRC: 6112  \*  \* Sistema: Mascotas Perdidas  \* Creado 01/09/2020  \*  \* Los contenidos de este archivo son propiedad privada y estan protegidos por  \* la licencia BSD  \*  \* Se puede utilizar, reproducir o copiar el contenido de este archivo.  \*/  package app.ejemplo.carlos.proyecto2;  import android.support.v4.app.Fragment;  import android.support.v4.app.FragmentManager;  import android.support.v4.app.FragmentPagerAdapter;  import app.ejemplo.carlos.proyecto2.Intros.Intro1;  import app.ejemplo.carlos.proyecto2.Intros.Intro2;  import app.ejemplo.carlos.proyecto2.Intros.Intro3;  import app.ejemplo.carlos.proyecto2.Intros.Intro4;  /\*\*  \* Clase  \*  \* @author Carlos Martínez  \* @author Paula Monteros  \*/  public class IntroAdapter extends FragmentPagerAdapter {  public IntroAdapter(FragmentManager fm) {  super(fm);  }  @Override  public Fragment getItem(int position) {  switch (position){  case 0:  return new Intro1();  case 1:  return new Intro2();  case 2:  return new Intro3();  case 3:  return new Intro4();  default:  return null;  }  }  @Override  public int getCount() {  return 4;  }  } |
| **Mascota.java** |
| package app.ejemplo.carlos.proyecto2;  /\*  \* ESPE - DCC - PROGRAMACIÓN MÓVIL  \* NRC: 6112  \*  \* Sistema: Mascotas Perdidas  \* Creado 01/09/2020  \*  \* Los contenidos de este archivo son propiedad privada y estan protegidos por  \* la licencia BSD  \*  \* Se puede utilizar, reproducir o copiar el contenido de este archivo.  \*/  public class Mascota {  /\*\*  \* Clase que representa al objeto Mascota  \*  \* @author Carlos Martínez  \* @author Paula Monteros  \*/  private String nombre\_mascota;  private String mapa;  private String desc;  private String nomPersona;  private String numCelular;  private String email;  private String url;  private String id\_mascota;  private String tipo;  private String user\_id;  public Objeto() {  }  public String getId\_mascota() { return id\_mascota;  }  public void setId\_mascota(String id\_mascota) {this.id\_mascota = id\_mascota;  }  public String getNomPersona() {  return nomPersona;  }  public void setNomPersona(String nomPersona) {  this.nomPersona = nomPersona;  }  public String getNumCelular() {  return numCelular;  }  public void setNumCelular(String numCelular) {  this.numCelular = numCelular;  }  public String getEmail() {  return email;  }  public void setEmail(String email) {  this.email = email;  }  public String getNombre\_mascota() {  return nombre\_mascota;  }  public void setNombre\_mascota(String nombre\_mascota) {  this.nombre\_mascota = nombre\_mascota;  }  public String getMapa() {  return mapa;  }  public void setMapa(String mapa) {  this.mapa = mapa;  }  public String getDesc() {  return desc;  }  public void setDesc(String desc) {  this.desc = desc;  }  public String getUrl() { return url; }  public void setUrl(String url) { this.url = url; }  public String getTipo() { return tipo; }  public void setTipo(String tipo) { this.tipo = tipo; }  public String getUser\_id() {  return user\_id;  }  public void setUser\_id(String user\_id) {  this.user\_id = user\_id;  }  } |
| **Publicados.java** |
| package app.ejemplo.carlos.proyecto2;  /\*  \* ESPE - DCC - PROGRAMACIÓN MÓVIL  \* NRC: 6112  \*  \* Sistema: Mascotas Perdidas  \* Creado 01/09/2020  \*  \* Los contenidos de este archivo son propiedad privada y estan protegidos por  \* la licencia BSD  \*  \* Se puede utilizar, reproducir o copiar el contenido de este archivo.  \*/  import android.content.Context;  import android.content.Intent;  import android.support.annotation.NonNull;  import android.support.v7.app.ActionBar;  import android.support.v7.app.AppCompatActivity;  import android.os.Bundle;  import android.support.v7.widget.LinearLayoutManager;  import android.support.v7.widget.RecyclerView;  import android.view.animation.AnimationUtils;  import android.view.animation.LayoutAnimationController;  import android.widget.Toast;  import com.google.firebase.database.DataSnapshot;  import com.google.firebase.database.DatabaseError;  import com.google.firebase.database.DatabaseReference;  import com.google.firebase.database.FirebaseDatabase;  import com.google.firebase.database.ValueEventListener;  import android.view.MenuItem;  import java.util.ArrayList;  /\*\*  \* Clase que se muestra los objetos publicados  \*  \* @author Carlos Martínez  \* @author Paula Monteros  \*/  public class Publicados extends AppCompatActivity{  DatabaseReference databaseReference;  ArrayList<Objeto> list = new ArrayList<Objeto>();  RecyclerView recyclerView;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_publicados);  getSupportActionBar().setTitle("Publicados");  getSupportActionBar().setDisplayHomeAsUpEnabled(true);  recyclerView = (RecyclerView)findViewById(R.id.recycler);  recyclerView.setLayoutManager(new LinearLayoutManager(this, LinearLayoutManager.VERTICAL,false));  databaseReference = FirebaseDatabase.getInstance().getReference().child("perdidos");  databaseReference.addValueEventListener(new ValueEventListener() {  @Override  public void onDataChange(@NonNull DataSnapshot dataSnapshot) {  for (final DataSnapshot snapshot : dataSnapshot.getChildren()){  Objeto objeto = snapshot.getValue(Objeto.class);  list.add(objeto);  }  AdapterDatos adapterDatos = new AdapterDatos(list,getApplicationContext());  recyclerView.setAdapter(adapterDatos);  Context context= recyclerView.getContext();  LayoutAnimationController controller=null;  recyclerView.setAdapter(adapterDatos);  controller=AnimationUtils.loadLayoutAnimation(context,R.anim.layout\_fall);  recyclerView.setLayoutAnimation(controller);  recyclerView.getAdapter().notifyDataSetChanged();  recyclerView.scheduleLayoutAnimation();  }  @Override  public void onCancelled(@NonNull DatabaseError databaseError) {  Toast.makeText(getApplicationContext(),"Ups hubo un problema enseguida lo arreglamos",Toast.LENGTH\_SHORT).show();  }  });  setupActionBar();  }  @Override  public boolean onOptionsItemSelected(MenuItem item) {  switch (item.getItemId()) {  case android.R.id.home:  onBackPressed();  return true;  }  return super.onOptionsItemSelected(item);  }  @Override  public void onBackPressed() {  super.onBackPressed();  Intent miIntenr = new Intent(this,menumain.class);  miIntenr.setFlags(Intent.FLAG\_ACTIVITY\_CLEAR\_TASK|Intent.FLAG\_ACTIVITY\_NEW\_TASK);  startActivity(miIntenr);  finish();  }  private void setupActionBar(){  ActionBar actionBar = getSupportActionBar();  if(actionBar!=null){  actionBar.setDisplayShowHomeEnabled(true);  actionBar.setDisplayHomeAsUpEnabled(true);  actionBar.setTitle("Publicados");  }  }  } |

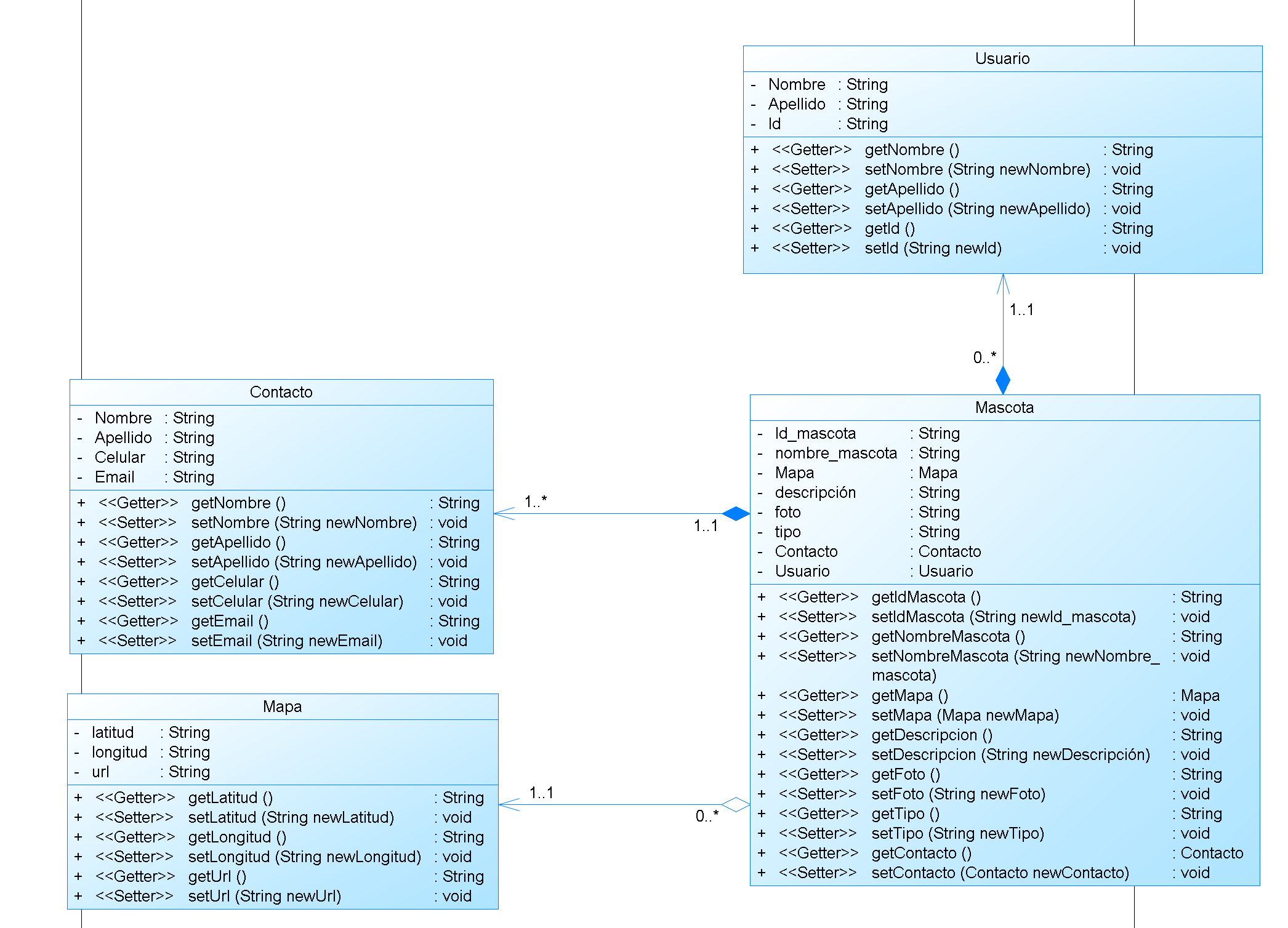
CONCLUSIONES

* Firebase ofrece un catalogo abierto de servicios que mejora y facilita el desarrollo de aplicaciones móviles mediante sus diferentes integraciones y bases de datos en tiempo real.
* Es importante tener en cuenta que Firebase tiene una cuota limite para los planes gratuitos, por lo que se debe tener en cuenta como van a ser usados sus diferentes servicios.
* Al usar el patrón MVP la capa de presentador es independiente de la tecnología de interfaz de usuario.
* En el presente proyecto se hizo uso del patrón MVC lo que nos permitió obtener mayores ventajas al momento de programar en comparación con la programación tradicional sin aplicar patrones de diseño. Nos brindó más rapidez y escalabilidad en cada una de las funciones de los botones.

|  |  |
| --- | --- |
| [1] | "DevelApps," [Online]. Available: http://www.develapps.com/es/noticias/modelo-vista-presentador-mvp-en-android. [Accessed 24 Junio 2020].  [2] «Firebase. » https://firebase.google.com/?hl=es(accedido ago. 03, 2020).  [3] «Pirmeros pasos en Firebasee», *Firebase Docs*. https://firebase.google.com/docs?hl=es (accedido ago. 03, 2020). |
|  |  |

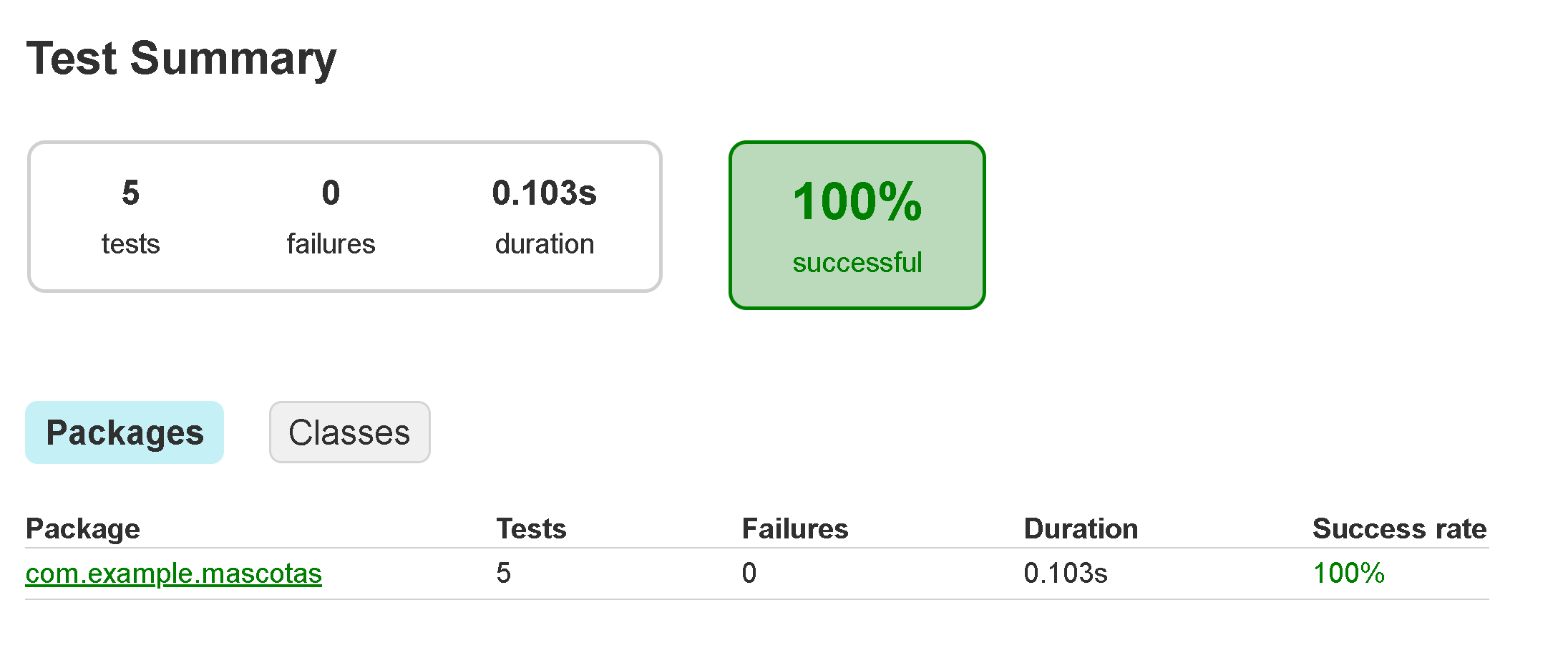
**BIBLIOGRAFÍA**

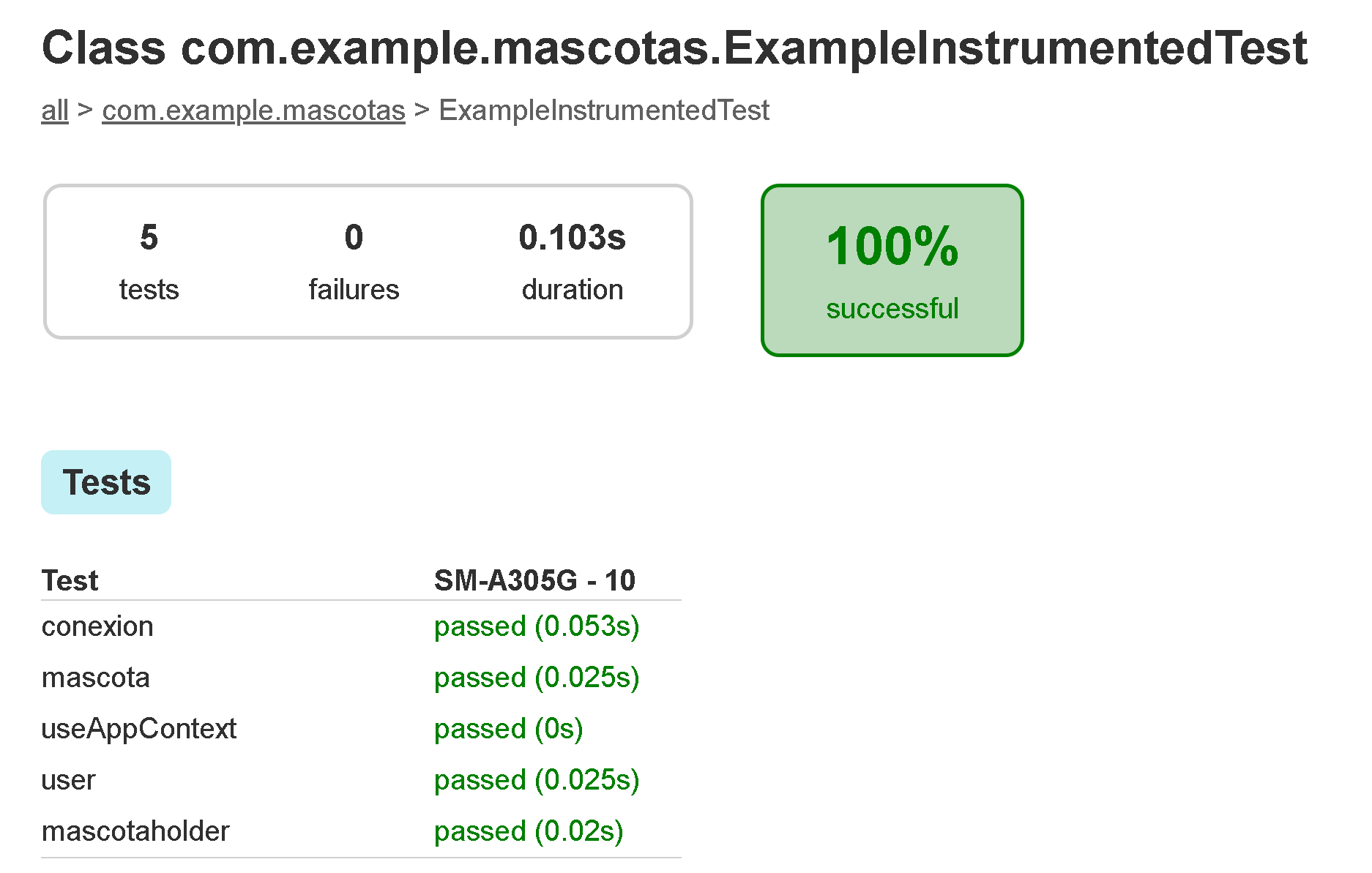
# **ANEXO N°1: Modelo**



**ANEXO N°2: Unit Test**

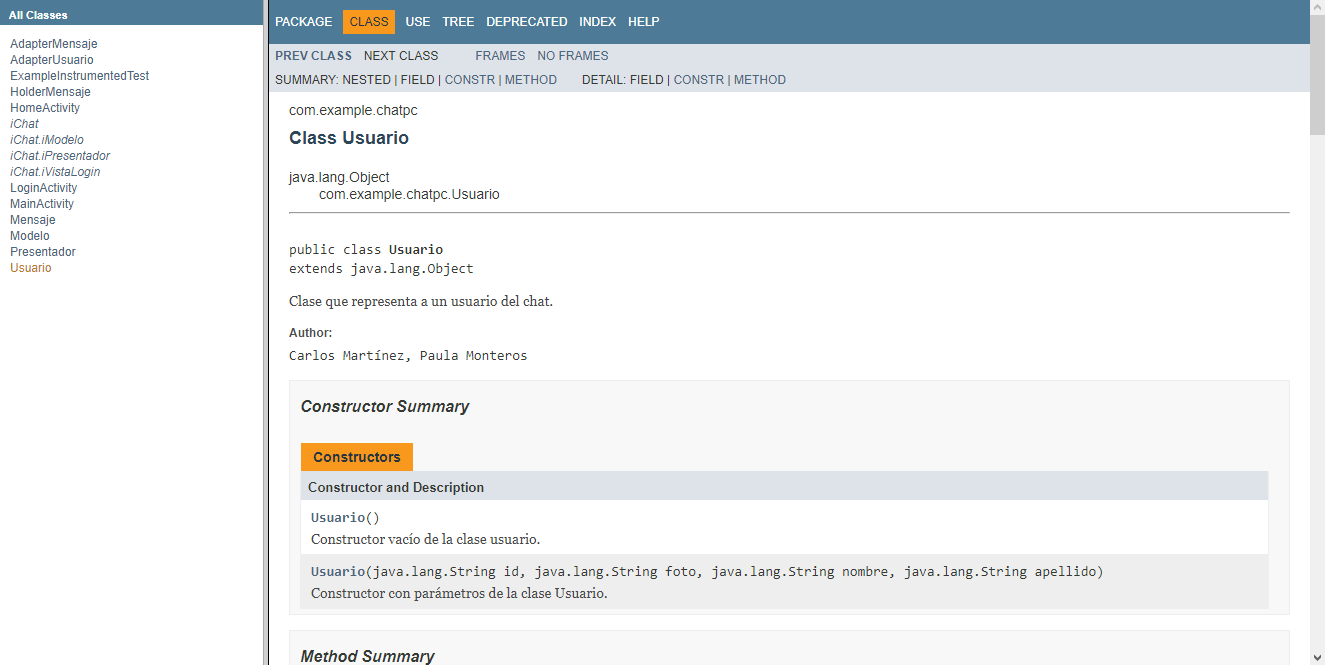
|  |
| --- |
| **ExampleInstrumentedTest.java** |
| /\*  \* ESPE - DCC - PROGRAMACIÓN MÓVIL  \* NRC: 6112  \*  \* Sistema: CHATP&C  \* Creado 02/08/2020  \*  \* Los contenidos de este archivo son propiedad privada y estan protegidos por  \* la licencia BSD  \*  \* Se puede utilizar, reproducir o copiar el contenido de este archivo.  \*/ package com.example.mascotas import android.content.Context;  import androidx.test.platform.app.InstrumentationRegistry; import androidx.test.ext.junit.runners.AndroidJUnit4;  import org.junit.Test; import org.junit.runner.RunWith;  import static org.junit.Assert.\*;  /\*\*  \* Clase para las pruebas unitarias del programa desarrollado.  \*  \* @author Carlos Martínez  \* @author Paula Monteros  \*/ @RunWith(AndroidJUnit4.class) public class ExampleInstrumentedTest {   /\*\*  \* Prueba unitaria para el contexto de la aplicación.  \*/  @Test  public void useAppContext() {  // Context of the app under test.  Context appContext = InstrumentationRegistry.getInstrumentation().getTargetContext();  assertEquals("com.example.chatpc", appContext.getPackageName());  }   /\*\*  \* Prueba unitaria para la creación de usuarios.  \*/  @Test  public void user(){  Usuario usuario=new Usuario();  assertNotNull(usuario);  }   /\*\*  \* Prueba unitaria para la creación de mascotas.  \*/  @Test  public void mensaje(){  Mensaje usuario=new Mensaje();  assertNotNull(usuario);  }   /\*\*  \* Prueba unitaria para la conexión con la base.  \*/  @Test  public void coneccion(){  Modelo modelo=new Modelo();  assertNotNull(modelo);  } } |

****

****

# **ANEXO N°4: JavaDoc**





# **ANEXO N°5: Código**

