materia: PROGRAMACIÓN DE SISTEMAS TELEMÁTICOS

Carrera de ingenieria en mecatronica

fiec-espol



**Reporte final**

**Mascarillas inteligentes - Aplicación Móvil.**

**Grupo #7**

Banchón Valdiviezo Melissa

Farinango Salazar Pedro

Lazo Ramírez Brando

Medina Chalen Lissette

Quinteros Moran Cristhian

1. **Resumen Ejecutivo**

El presente proyecto surge a partir de la situación actual que está atravesando el mundo, esto es, una emergencia sanitaria que impone a cada una de las personas llevar una mascarilla como medida de prevención y mitigación frente al covid-19. Una mascarilla inteligente es un producto basado en tecnología, innovación y confiabilidad en cuanto a la crisis mencionada, por lo que el proyecto consiste en un sistema dentro de una mascarilla capaz de detectar la proximidad de una persona, de modo que se cierre la compuerta de la mascarilla cuando tenga una persona cercana y se abra de no ser así, estableciendo si esta se encuentra protegida o no, y además, mostrando en tiempo real los datos adquiridos y un registro de las personas con las que tuvo contacto en una aplicación móvil denominada Intelimasc.

La mascarilla inteligente tiene como grupo objetivo a toda la población, debido a que es una necesidad primordial que ofrece salud, comodidad y seguridad involucrando tecnología e innovación para brindar una respuesta más eficaz contra la pandemia. Esto supone una ventaja frente a las mascarillas existentes en el mercado, puesto que representa un aporte en la medida de que es un elemento de bioseguridad, el cual determina si una persona está protegida o no de acuerdo con la distancia en que se encuentra respecto a otra persona, generando una alerta y llevando un registro de todos los datos en una aplicación.

Según el alcance del área de mercado actualmente y el grupo objetivo, se espera que la mascarilla inteligente tenga gran aceptación por parte de toda la población y, a su vez, estaría expuesta a modificaciones por expertos que inspeccionen su ajuste a la normativa internacional y hagan verificaciones en cuanto a riesgos de fuga y capacidad de filtrado.

1. **Descripción del problema**

Actualmente el mundo está atravesando por una pandemia provocada por un virus llamado Covid-19, por ende, es indispensable el uso de mascarillas para prevenir contagios y evitar el aumento de personas contagiadas. Esto ha generado inconformidades por cierta parte de la población, ya que a algunas personas les incomoda llevar una mascarilla por un tiempo prolongado, debido a que su uso puede ser incómodo en ocasiones ‘innecesarias’ y esta no permite respirar de forma adecuada o produce empañamiento en los lentes (en caso de que el usuario utilice), entre otros.

Aparte de esto, es importante que las personas que se encuentren fuera de sus establecimientos posean un tipo de registro de cuantos contactos tuvieron con personas para que puedan prevenir de forma más acertada algún tipo de contagio.

1. **Objetivos específicos**

* Implementar un sistema en una mascarilla que sea permita la detección de proximidad de una persona y realice acciones dependiendo de la distancia en que se encuentre.
* Establecer conexión con una base de datos entre un módulo wifi y una aplicación móvil para la transmisión de información de un componente presente en la mascarilla a una interfaz de usuario.
* Desarrollar una aplicación móvil que tenga conexión con una base de datos para la presentación del estado en el que se encuentra un usuario con respecto a los contactos que tuvo en un tiempo definido.
* Establecer conexión entre sensores de proximidad y una base de datos para la verificación de si el usuario se encuentra protegido o no.

1. **¿Cómo funciona la solución?**

Para implementar la solución se hará uso de un módulo wifi nodeMCU ESP8226 y otros módulos de Arduino, el módulo nodeMCU tomará datos constantemente a través de un módulo de proximidad y enviará datos hacia una aplicación móvil cuando el estado del usuario cambie, es decir, si su estado cambia de seguro a inseguro o viceversa.

El módulo nodeMCU detectará a través de un módulo de proximidad si el usuario se encuentra seguro; esto es, si la mínima distancia entre él y otra persona es de 2m, de este modo, accionará el motor Server que abrirá una compuerta que estará en la mitad de la mascarilla, y para el caso en que se encuentre inseguro, es decir, si la distancia es inferior a 2 m, se cerrará la compuerta para que las personas estén protegidas.

Cuando la aplicación móvil reciba un dato, este actualizará la información en tiempo real, y mostrará si el usuario está protegido o no, aparte llevará un registro del número de personas con las que tuvo contacto, el cual podrá ser visible en la aplicación.

1. **¿Qué se va a construir para resolver el problema?**

Se tiene pensado en aplicar la metodología de Design thinking la cual busca generar ideas innovadores para brindar una solución a las necesidades del usuario de manera original y eficaz, en este caso particular, para el modelado y dimensiones de la mascarilla, se debe diseñar de tal manera que se adapte a la forma de la cara de la persona, es decir, se tendrán diferentes medidas de estandarización para los diferentes casos (hombre, mujeres, niño, personas delgadas o con sobrepeso) y de la misma manera que los elementos electrónicos que estén en la mascarilla no sean una molestia para dicho usuario. Por ende, para facilitar la construcción se propone utilizar inventor o fusión 360 de Autodesk los cuales son softwares en 3D que permitirán analizar cada detalle de la mascarilla que se va a realizar.

Los elementos que estarán presentes en la mascarilla serán: sensor de proximidad, servo motor, fuente de alimentación de 9v, módulo NodeMCU y su adaptador USB.

Con los objetos anteriormente mencionados se tiene pensado crear una compuerta en la parte central de la mascarilla, la cual estará conectada con el eje del servo motor para que esta se abra si el usuario se encuentra seguro, y se cierre de no ser así. Por último, se realizarán modificaciones como colocar relieves que encapsulen los demás elementos electrónicos, con la finalidad de que estos no sean del todo visibles hacia el usuario y los espectadores, así teniendo un aspecto más amigable para el público en general.

1. **Recursos de hardware y software**

**Recursos de hardware**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre | Función | Descripción | Especificaciones | Imagen |
| Cables | Conexión entre los distintos componentes del sistema. | Permite el correcto funcionamiento del sistema completo. | N/A |  |
| Módulo   NodeMCU | Conexión Wi-Fi | Es una plataforma IoT de código abierto de bajo costo. Incluye el firmware que se ejecuta en el SoC Wi-Fi ESP8266. | Alimentación:  7 - 12V  Voltaje de operación:  3.3 V  Memoria Flash:  4 MB  Memoria SRAM:  64 KB  Tamaño:  47×21 mm |  |
| Servomotor | Control preciso en términos de posición angular, aceleración y velocidad para el movimiento de la compuerta. | Utiliza un motor normal y lo combina con un sensor para la retroalimentación de posición. | Voltaje de operación:  4 - 7.2 V  Velocidad de giro:  A 4.8 V → 0.12 seg/60º  Tamaño:  22×11.5×27 mm |  |
| Sensor de distancia ultrasónico  [HC-SR04] | Medición de distancias | Envía un pulso de alta frecuencia no audible por el ser humano, rebota en objetos cercanos y se refleja hacia el sensor que tiene integrado un micrófono capaz de percibir esta frecuencia. La distancia es medida a partir del tiempo en que demora en llegar al objeto y reflejarse en el sensor entre cada pulso, basándose en la velocidad del sonido. | Alimentación:  5V – 15 mA  Rango de trabajo:  2cm a 4 m  Frecuencia de pulso:  40 kHz  Apertura del pulso: 15º  Señal de disparo:  10µs  Tamaño:  40×20×15 mm |  |
| Batería | Fuente de alimentación | N/A | Voltaje entregado:  9V  Tamaño:  48.5×26.5×17.5 mm |  |
| Adaptador de batería | Conexión entre la fuente de alimentación y el módulo NodeMCU. | Permite portabilidad del producto. | N/A |  |

Tabla 1Recurso de hardware que se utilizara

**Recursos de software**

Para este apartado, los recursos de software que se tienen son los siguientes:

* Android Studio. - en este programa se realizará la aplicación móvil donde se registrará el usuario y se mostraran los datos de distancia, y si está seguro o no.
* Arduino IDE. - en este programa se desarrollará las funciones que cumplirán los componentes como el servomotor y el sensor ultrasónico HC SR04.
* PHPmyAdmin. - se manejará la administración de MySQL donde se creará y se almacenará la base de datos, con sus correspondientes tablas descritas en el modelo entidad-relación: usuario, mascarilla, datos y aplicación.

1. **Implementación del proyecto**

El problema se dividió en varias partes para su resolución, empezando desde el respectivo diagrama entidad-relación de los usuarios con las mascarillas. Para dicho diagrama se tomó en cuenta la tabla Usuario con los datos personales de cada usuario, un conteo de las personas que han tenido contacto con la misma y estos al ser guardados en la base de datos contarán con una ID única la cual permite relacionarse con la tabla mascarilla. Esta tabla Mascarilla tiene un único ID, el id del usuario al que pertenece dicha mascarilla, y otros datos que posteriormente se mostrarán en una aplicación. La tabla mascarilla se relacionará a la Tabla datos, la cual es de vital importancia ya que tendrá los datos que le dirán al usuario si está seguro o no.

Posterior a esto se construyó el respectivo diagrama del circuito electrónico, este tendrá un sensor ultrasónico el cuál va a medir la distancia de las personas respecto al usuario con la mascarilla puesta, este dato será importante, ya que determina si cada usuario está seguro o no, dependiendo de la distancia en mención. Este sistema electrónico también contará con un servomotor, el cual solo tiene 180° de giro, lo suficiente para abrir y cerrar la compuerta de la mascarilla ubicada entre la nariz y la boca de cada usuario. El abrir y cerrar depende del dato obtenido por el sensor ultrasónico, si está seguro el usuario se abre la compuerta, es decir, el servomotor se activa haciendo que el eje conectado a la compuerta de un giro de 180°, cuando el usuario no está seguro el eje vuelve a su posición inicial, es decir a 0° cerrándose la compuerta.

Para la aplicación se hizo uso del software Android Studio con lenguaje de programación JAVA. La aplicación cuenta con un inicio de sesión y en caso de que un usuario no se haya registrado, se registra dentro de la misma aplicación. La ventana de inicio de la aplicación va a mostrar el nombre del usuario, el número de personas con las que ha tenido contacto y las mascarillas registradas de cada usuario, a su vez, la aplicación contará con un submenú que mostrará las mascarillas y el botón para la administración de estas, es decir, agregar, eliminar o restaurar mascarillas.

Para finalizar, la construcción de la mascarilla se realizará de tal forma que se ajuste al rostro del usuario y se sujete bien en la parte trasera de la cabeza y así tener una mayor estabilidad.

1. **Diagramas del diseño del proyecto**

* **Diagrama esquemático**

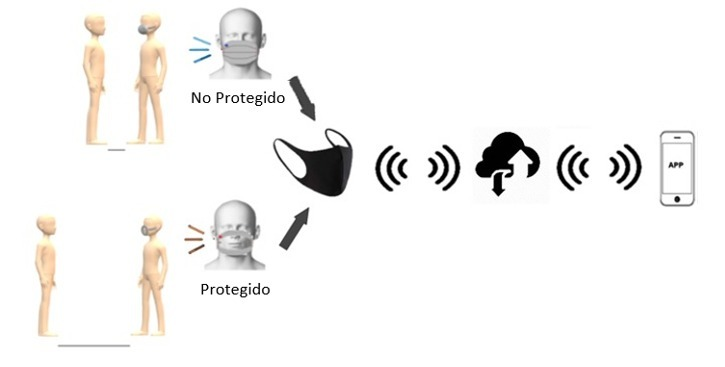


Ilustración 1. Diagrama de diseño del proyecto

* **Modelo entidad-relación**

**A screenshot of a cell phone

Description automatically generated**

Ilustración 2. Modelo entidad-relación

* **Diagrama de circuito**

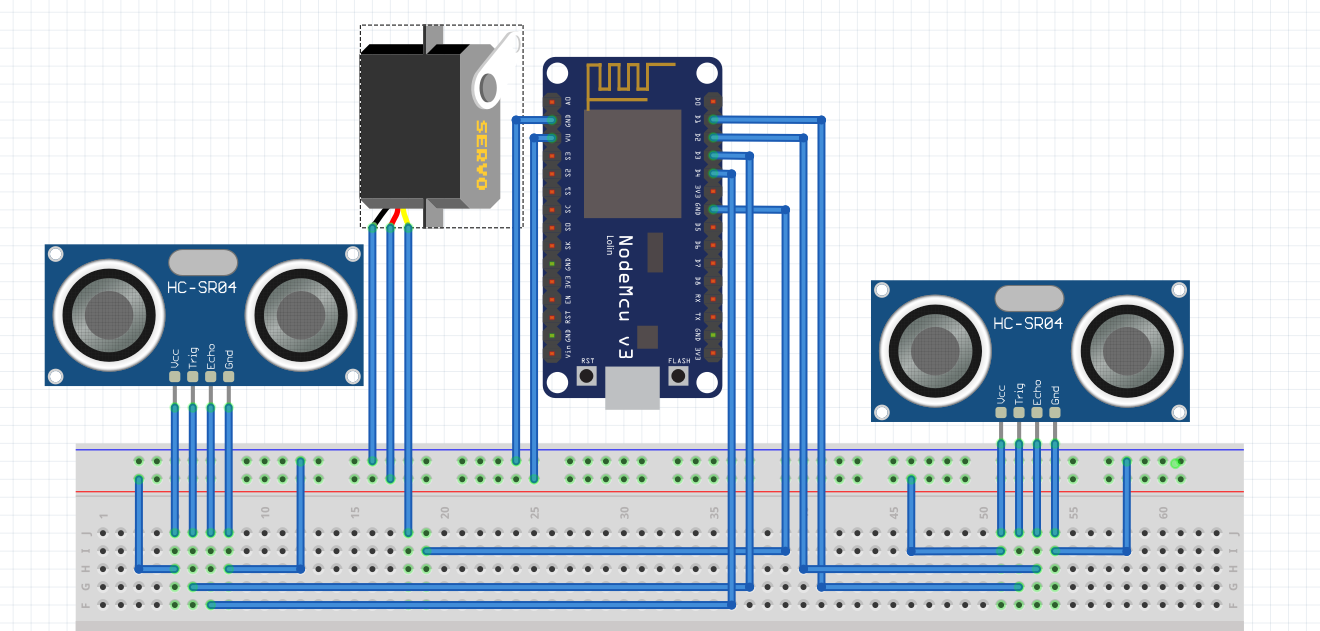
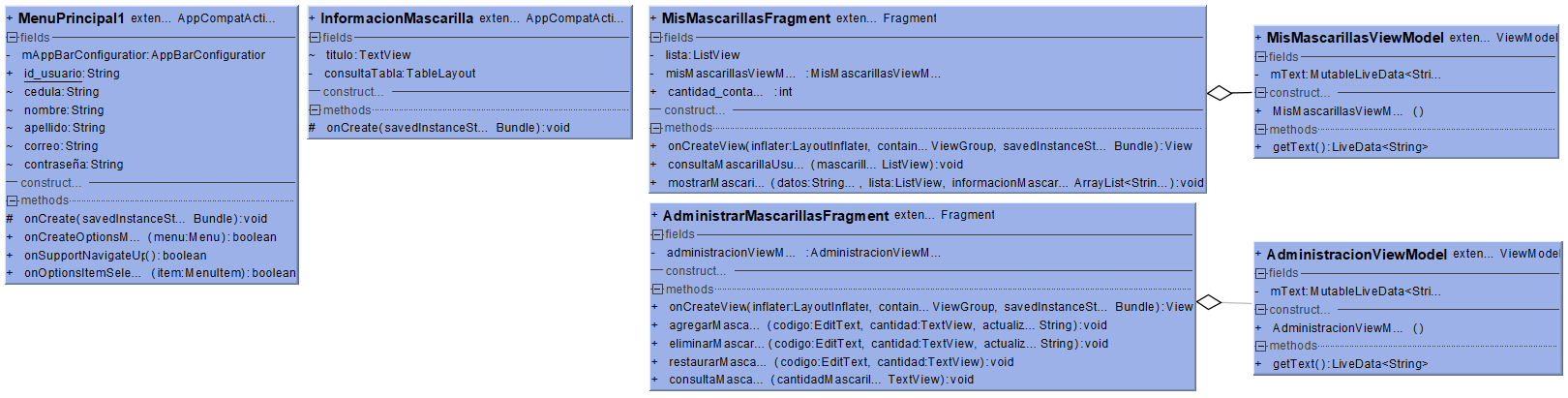
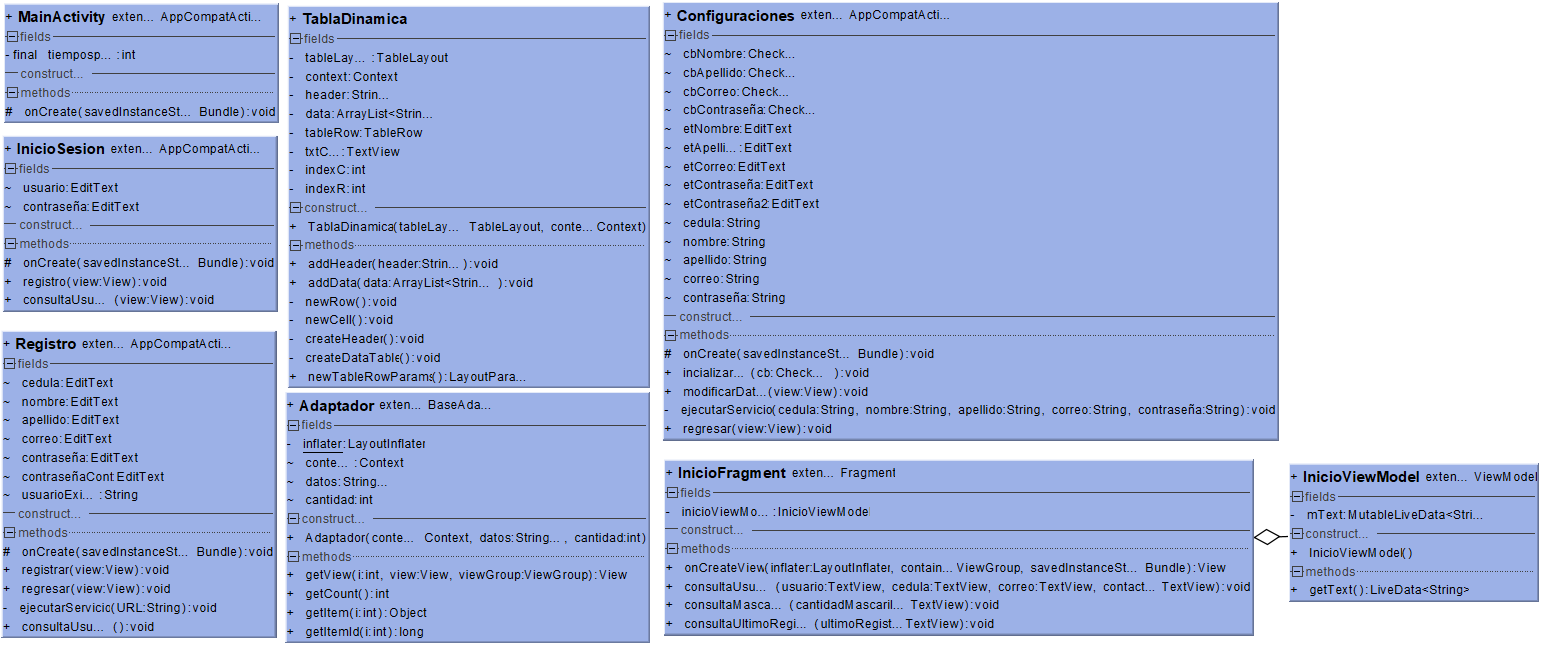
****

Ilustración 3. Diagrama del circuito

* **Diagrama de casos UML**

Ilustración 4. Diagrama de casos UML.

* **Diagrama de despliegue**

Mascarilla Inteligente

Base de datos relacional

000webhost

HTTP

GET/POST

HTTP

POST

Aplicación Móvil

1. **Descripción de campos, tipos de datos creados en la base de datos, y código SQL**

**PHP**

* busqueda\_usuario.php [Registro – Inicio Sesión]

Obtiene el Id de usuario, cédula, nombre, apellido, correo, contraseña y el conteo de contactos con personas del usuario cuyo número de cédula se encuentre en la base de datos. Sirve para comprobar las credenciales en el inicio de sesión y verificar la existencia de un usuario en el registro.

* consulta\_datos\_mascarilla.php [Inicio Fragment – Administrar Mascarillas Fragment ]

Obtiene el id y código de la mascarilla junto con el id de usuario correspondiente al usuario que ha iniciado sesión. Sirve para mostrar el número de mascarillas que posee el usuario y para verificar que la mascarilla que se desea eliminar, pertenezca al usuario.

* consulta\_datos\_usuario.php [Inicio Fragment]

Obtiene el Id de usuario, cédula, nombre, apellido, correo, contraseña y el conteo de contactos con personas del usuario que ha iniciado sesión. Sirve para mostrar los datos del usuario.

* consulta\_mascarillas\_totales.php [Administrar Mascarillas Fragment]

Obtiene el id y código de la mascarilla junto con el id de usuario correspondiente al código de mascarilla ingresado por el usuario. Sirve para verificar si la mascarilla ingresada existe o no en la base de datos.

* ingreso\_usuario.php [Registro]

Registra el usuario en la base de datos con los datos obtenidos en el formulario.

* fecha\_ultimo\_registro.php [Inicio Fragment]

Sirve para obtener la fecha del último registro de contacto con personas que hubo, se realizan varios inner join de tablas para relacionar finalmente el id del usuario con el dato requerido.

* prueba\_busqueda\_mascarilla.php [Mis Mascarillas Fragment]

Sirve para mostrar las mascarillas que tiene el usuario y los datos que ha recolectado (Fecha, distancia y seguridad).

* ingreso\_mascarilla.php [Administrar Mascarillas Fragment]

Registra en la base de datos la mascarilla ingresada por el usuario que ha iniciado sesión.

* eliminar\_mascarilla.php [Administrar Mascarillas Fragment]

Elimina la mascarilla ingresada por el usuario junto con sus datos de contactos en la base de datos.

* editar\_usuario.php [Configuraciones]

Sirve para que los cambios realizados por el usuario se actualicen en la base de datos.

* esp-data.php [Arduino]

Muestra los datos obtenidos por el sensor de proximidad.

* post-esp-data.php [Arduino]

Ingresa en la base de datos los datos obtenidos por el sensor de proximidad.

**MYSQL**

Tabla Usuario

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Tipo de dato | Tipo de campo | Característica | Función |
| Id\_usuario | Int | Primary\_key | Auto\_incremento | Identifica el dato a nivel de la base de datos |
| Cedula | varchar | Obligatorio | Unico | Identifica al usuario al nivel de aplicación |
| Nombre | varchar | Obligatorio |  | Dato general |
| Apellido | varchar | Obligatorio |  | Dato general |
| Correo | varchar | Obligatorio |  | Dato general |
| Contraseña | varchar | Obligatorio |  | Credencial que permite el inicio de sesión |
| Conteo\_personas | Int | Null |  | Conteo general por todas las mascarillas. |

Tabla Mascarilla

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Tipo de dato | Tipo de campo | Característica | Función |
| Id\_mascarilla | Int | Primary\_key | Auto\_incremento | Identifica el dato a nivel de la base de datos |
| Codigo\_mascarilla | Int | Foreign Key | Único | Identifica la mascarilla al nivel de aplicación |
| Id\_usuario | Int | Foreign Key |  | Relaciona la mascarilla con su usuario. |

Tabla Fecha

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Tipo de dato | Tipo de campo | Característica | Función |
| Id\_fecha | Int | Primary\_key | Auto\_incremento | Identifica el dato a nivel de la base de datos |
| fecha | date | Foreign Key | Único | Contiene la fecha de registro de los datos |

Tabla Dato

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Tipo de dato | Tipo de campo | Característica | Función |
| Id\_dato | Int | Primary\_key | Auto\_incremento | Identifica el dato a nivel de la base de datos |
| Id\_mascarilla | Int | Foreign Key |  | Identifica el dato a nivel de la base de datos |
| Id\_fecha | Int | Foreign Key |  | Identifica el dato a nivel de la base de datos |
| distancia | Int | Obligatorio |  | Distancia del objeto que se acercó el sensor |
| seguro | tinyint | Obligatorio |  | Permite saber si el usuario estuvo a una distancia segura |

Tabla DatoSensor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Tipo de dato | Tipo de campo | Característica | Función |
| codigo\_mascarilla | Int |  |  | Obtiene los datos del sensor y los envía a sus respectivas tablas. |

**Triggers**

**Tabla datoSensor**

AgregarDato: Agrega la información obtenida por el sensor en la tabla Dato.

CREATE TRIGGER `agregarDato` AFTER INSERT ON `DatoSensor` FOR EACH ROW [INSERT](https://databases-auth.000webhost.com/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/insert.html) INTO Dato(id\_mascarilla, id\_fecha, distancia, seguro) [VALUES](https://databases-auth.000webhost.com/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/miscellaneous-functions.html#function_values) (([SELECT](https://databases-auth.000webhost.com/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) id\_mascarilla FROM Mascarilla AS M WHERE M.codigo\_mascarilla= new.codigo\_mascarilla), ([SELECT](https://databases-auth.000webhost.com/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) id\_fecha FROM Fecha AS R WHERE R.fecha= new.fecha), new.distancia, new.seguro)

AgregarFecha: Agrega la fecha obtenida por el sensor en la tabla Dato.

CREATE TRIGGER `agregarDato` AFTER INSERT ON `DatoSensor` FOR EACH ROW [INSERT](https://databases-auth.000webhost.com/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/insert.html) INTO Dato(id\_mascarilla, id\_fecha, distancia, seguro) [VALUES](https://databases-auth.000webhost.com/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/miscellaneous-functions.html#function_values) (([SELECT](https://databases-auth.000webhost.com/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) id\_mascarilla FROM Mascarilla AS M WHERE M.codigo\_mascarilla= new.codigo\_mascarilla), ([SELECT](https://databases-auth.000webhost.com/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) id\_fecha FROM Fecha AS R WHERE R.fecha= new.fecha), new.distancia, new.seguro)

**Tabla Dato**

Contador: Suma la cantidad de contactos en la tabla Usuario si en el nuevo valor ingresado en la tabla Dato, ‘seguro’, es 0.

CREATE TRIGGER `contador` AFTER INSERT ON `Dato` FOR EACH ROW [UPDATE](https://databases-auth.000webhost.com/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/update.html) Usuario [SET](https://databases-auth.000webhost.com/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/set.html) conteo\_personas = [IF](https://databases-auth.000webhost.com/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/control-flow-functions.html#function_if)(NEW.seguro = 0, conteo\_personas+1, conteo\_personas) WHERE id\_usuario = ([select](https://databases-auth.000webhost.com/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) id\_usuario from Mascarilla where id\_mascarilla = NEW.id\_mascarilla)

**Tabla Mascarilla**

actualizarDatos: Resta la cantidad de contactos en la tabla Usuario dependiendo del número de contactos de una mascarilla antes de su eliminación.

CREATE TRIGGER `actualizarDatos` BEFORE DELETE ON `Mascarilla` FOR EACH ROW [UPDATE](https://databases-auth.000webhost.com/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/update.html) Usuario [SET](https://databases-auth.000webhost.com/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/set.html) conteo\_personas = conteo\_personas - ([SELECT](https://databases-auth.000webhost.com/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) [COUNT](https://databases-auth.000webhost.com/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/group-by-functions.html#function_count)(\*) from ([SELECT](https://databases-auth.000webhost.com/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) \* from ([SELECT](https://databases-auth.000webhost.com/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) codigo\_mascarilla,id\_mascarilla as id\_m from Mascarilla) as M INNER JOIN ([SELECT](https://databases-auth.000webhost.com/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) \* FROM Dato) as D on M.id\_m = D.id\_mascarilla) as T WHERE T.codigo\_mascarilla = old.codigo\_mascarilla [and](https://databases-auth.000webhost.com/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/logical-operators.html#operator_and) T.seguro = 0) WHERE id\_usuario = old.id\_usuario

inicializarMascarilla: Ingresa un dato nulo en la tabla Dato al momento de crear una mascarilla.

CREATE TRIGGER `inicializarMascarilla` AFTER INSERT ON `Mascarilla` FOR EACH ROW [INSERT](https://databases-auth.000webhost.com/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/insert.html) INTO Dato [VALUES](https://databases-auth.000webhost.com/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/miscellaneous-functions.html#function_values)(null,NEW.id\_mascarilla,0,0,1)

1. **Código fuente**

**Android**

**-----------------------------MainActivity1-------------------------------------**

Este es el principal contenedor de los fragment

package com.example.proyectopst;  
  
import android.content.Intent;  
import android.os.Bundle;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.TextView;  
  
import com.google.android.material.navigation.NavigationView;  
  
import androidx.navigation.NavController;  
import androidx.navigation.Navigation;  
import androidx.navigation.ui.AppBarConfiguration;  
import androidx.navigation.ui.NavigationUI;  
import androidx.drawerlayout.widget.DrawerLayout;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.widget.Toolbar;  
  
public class MenuPrincipal1 extends AppCompatActivity {  
 private AppBarConfiguration mAppBarConfiguration;  
 public static String *id\_usuario*;  
 String cedula,nombre,apellido,correo, pass;  
  
 */\*\*  
 \* Se obtienen los datos de inicio de sesión mediante un bundle y se setean en el nav\_view  
 \** ***@param*** *savedInstanceState  
 \*/* @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_menu\_principal1*);  
 Toolbar toolbar = findViewById(R.id.*toolbar*);  
 setSupportActionBar(toolbar);  
  
 Bundle obtenerDatos = getIntent().getExtras();  
 *id\_usuario* = obtenerDatos.getString("id\_usuario");  
 cedula = obtenerDatos.getString("cedula");  
 nombre = obtenerDatos.getString("nombre");  
 apellido = obtenerDatos.getString("apellido");  
 correo = obtenerDatos.getString("correo");  
 pass = obtenerDatos.getString("contraseña");  
  
 NavigationView navigationView = findViewById(R.id.*nav\_view*);  
 DrawerLayout drawer = findViewById(R.id.*drawer\_layout*);  
 View headerView = navigationView.getHeaderView(0);  
 TextView navUsername = (TextView) headerView.findViewById(R.id.*textViewUsuarioNav*);  
 TextView navMail = (TextView) headerView.findViewById(R.id.*textViewCorreoNav*);  
  
 navUsername.setText(obtenerDatos.getString("nombre")+" "+obtenerDatos.getString("apellido"));  
 navMail.setText(obtenerDatos.getString("correo"));  
 // Passing each menu ID as a set of Ids because each  
 // menu should be considered as top level destinations.  
 mAppBarConfiguration = new AppBarConfiguration.Builder(  
 R.id.*nav\_home*, R.id.*nav\_gallery*, R.id.*nav\_slideshow*)  
 .setDrawerLayout(drawer)  
 .build();  
 NavController navController = Navigation.*findNavController*(this, R.id.*nav\_host\_fragment*);  
 NavigationUI.*setupActionBarWithNavController*(this, navController, mAppBarConfiguration);  
 NavigationUI.*setupWithNavController*(navigationView, navController);  
 }  
  
 */\*\*  
 \* Añade los items al action bar si este está presente.  
 \** ***@param*** *menu  
 \** ***@return*** *true  
 \*/* @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
 // Inflate the menu; this adds items to the action bar if it is present.  
 getMenuInflater().inflate(R.menu.*menu\_principal1*, menu);  
 return true;  
 }  
  
 */\*\*  
 \*  
 \** ***@return*** *\*/* @Override  
 public boolean onSupportNavigateUp() {  
 NavController navController = Navigation.*findNavController*(this, R.id.*nav\_host\_fragment*);  
 return NavigationUI.*navigateUp*(navController, mAppBarConfiguration)  
 || super.onSupportNavigateUp();  
 }  
  
 */\*\*  
 \* Acciones disponibles en el menú:  
 \* CONFIGURACIÓN  
 \* Acción del botón que inicializa el activity de configuraciones.  
 \* Envía los datos actuales del usuario por medio de un bundle.  
 \* CERRAR SESIÓN  
 \* Acción del botón que inicializa el activity de inicio de sesión.  
 \* Finaliza el activity actual.  
 \** ***@param*** *item  
 \** ***@return*** *\*/* public boolean onOptionsItemSelected(MenuItem item) {  
 switch (item.getItemId()) {  
 case R.id.*action\_settings*:  
 Intent conf = new Intent(getApplicationContext(),Configuraciones.class);  
 conf.putExtra("cedula",cedula);  
 conf.putExtra("nombre",nombre);  
 conf.putExtra("apellido",apellido);  
 conf.putExtra("correo",correo);  
 conf.putExtra("contraseña", pass);  
 startActivity(conf);  
 return true;  
 case R.id.*action\_logout*:  
 Intent ini = new Intent(getApplicationContext(),InicioSesion.class);  
 startActivity(ini);  
 finish();  
 return true;  
 default:  
 // If we got here, the user's action was not recognized.  
 // Invoke the superclass to handle it.  
 return super.onOptionsItemSelected(item);  
 }  
 }  
}

-----------------------------InicioFragment-------------------------------------

Captura de pantalla de un celular

Descripción generada automáticamentepackage com.example.proyectopst.ui.Inicio;  
  
import android.os.Bundle;  
import android.util.Log;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ImageView;  
import android.widget.TextView;  
  
import androidx.annotation.Nullable;  
import androidx.fragment.app.Fragment;  
import androidx.lifecycle.Observer;  
import androidx.lifecycle.ViewModelProviders;  
  
import com.android.volley.RequestQueue;  
import com.android.volley.Response;  
import com.android.volley.VolleyError;  
import com.android.volley.toolbox.JsonArrayRequest;  
import com.android.volley.toolbox.Volley;  
import com.example.proyectopst.MenuPrincipal1;  
import com.example.proyectopst.R;  
  
import org.json.JSONArray;  
import org.json.JSONException;  
import org.json.JSONObject;  
  
public class InicioFragment extends Fragment {  
 private InicioViewModel inicioViewModel;  
  
 @Override  
 public View onCreateView(LayoutInflater inflater,  
 ViewGroup container, Bundle savedInstanceState) {  
 inicioViewModel = ViewModelProviders.*of*(this).get(InicioViewModel.class);  
 View root = inflater.inflate(R.layout.*fragment\_inicio*, container, false);  
 final TextView textView = root.findViewById(R.id.*text\_home*);  
 final TextView usuario = root.findViewById(R.id.*textViewUsuario*);  
 final TextView cedula = root.findViewById(R.id.*textViewCedula*);  
 final TextView correo = root.findViewById(R.id.*textViewCorreo*);  
 final TextView contactos = root.findViewById(R.id.*textViewCantidadContactos*);  
 final TextView cantidadMascarillas = root.findViewById(R.id.*textViewCantidadMascRegist*);  
 final TextView ultimoRegistro = root.findViewById(R.id.*textViewMostrarUltimoRegistro*);  
 final ImageView touchscreen = root.findViewById(R.id.*touchscreen*);  
 consultaUltimoRegistro(ultimoRegistro);  
 consultaUsuario(usuario,cedula,correo,contactos);  
 consultaMascarilla(cantidadMascarillas);  
 touchscreen.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 consultaUltimoRegistro(ultimoRegistro);  
 consultaUsuario(usuario,cedula,correo,contactos);  
 consultaMascarilla(cantidadMascarillas);  
 }  
 });  
 inicioViewModel.getText().observe(getViewLifecycleOwner(), new Observer<String>() {  
 @Override  
 public void onChanged(@Nullable String s) {  
 textView.setText(s);  
 }  
 });  
 return root;  
 }  
  
 */\*\*  
 \* Obtiene de la base de datos la información del usuario que ha iniciado sesión.  
 \** ***@param*** *usuario TextView para mostrar el nombre del usuario.  
 \** ***@param*** *cedula TextView para mostrar la cédula del usuario.  
 \** ***@param*** *correo TextView para mostrar el correo del usuario.  
 \** ***@param*** *contactos TextView para mostrar la cantidad de acercamientos o contactos con otra persona que ha tenido el usuario.  
 \*/* public void consultaUsuario(final TextView usuario, final TextView cedula, final TextView correo, final TextView contactos){  
 JsonArrayRequest jsonArrayRequest = new JsonArrayRequest("https://undried-modes.000webhostapp.com/consulta\_datos\_usuario.php?id\_usuario="+MenuPrincipal1.*id\_usuario*,  
 new Response.Listener<JSONArray>() {  
 @Override  
 public void onResponse(JSONArray response) {  
 JSONObject jsonObject = null;  
 for (int i = 0; i < response.length(); i++) {  
 try {  
 jsonObject = response.getJSONObject(i);  
 usuario.setText(jsonObject.get("nombre").toString()+" "+jsonObject.get("apellido").toString());  
 cedula.setText(jsonObject.get("cedula").toString());  
 correo.setText(jsonObject.get("correo").toString());  
 contactos.setText(jsonObject.get("conteo\_personas").toString());  
 } catch (JSONException je) {  
 Log.*e*("ERROR\_JSON",je.getMessage());  
 }  
 }  
 }  
 }, new Response.ErrorListener() {  
 @Override  
 public void onErrorResponse(VolleyError error) {  
 Log.*e*("ERROR\_CONEXION",error.getMessage());  
 }  
 });  
 RequestQueue requestQueue = Volley.*newRequestQueue*(getContext());  
 requestQueue.add(jsonArrayRequest);  
 }  
  
 */\*\*  
 \* Obtiene de la base de datos la cantidad de mascarillas que tiene el usuario.  
 \** ***@param*** *cantidadMascarillas TextView para mostrar la cantidad de mascarillas del usuario.  
 \*/* public void consultaMascarilla(final TextView cantidadMascarillas){  
 JsonArrayRequest jsonArrayRequest = new JsonArrayRequest("https://undried-modes.000webhostapp.com/consulta\_datos\_mascarilla.php?id\_usuario="+MenuPrincipal1.*id\_usuario*,  
 new Response.Listener<JSONArray>() {  
 @Override  
 public void onResponse(JSONArray response) {  
 if(Integer.*toString*(response.length())!=null){  
 cantidadMascarillas.setText(Integer.*toString*(response.length()));  
 }else{  
 cantidadMascarillas.setText("0");  
 }  
 }  
 }, new Response.ErrorListener() {  
 @Override  
 public void onErrorResponse(VolleyError error) {  
 Log.*e*("ERROR\_CONEXION",error.getMessage());  
 }  
 });  
 RequestQueue requestQueue = Volley.*newRequestQueue*(getContext());  
 requestQueue.add(jsonArrayRequest);  
 }  
  
 */\*\*  
 \* Obtiene de la base de datos la fecha del último registro de información de alguna de las mascarillas del usuario.  
 \** ***@param*** *ultimoRegistro TextView para mostrar la fecha del último contacto que ha tenido el usuario.  
 \*/* public void consultaUltimoRegistro(final TextView ultimoRegistro){  
 JsonArrayRequest jsonArrayRequest = new JsonArrayRequest("https://undried-modes.000webhostapp.com/fecha\_ultimo\_registro.php?id\_usuario="+MenuPrincipal1.*id\_usuario*,  
 new Response.Listener<JSONArray>() {  
 @Override  
 public void onResponse(JSONArray response) {  
 JSONObject jsonObject = null;  
 if(response.length()!=0){  
 for (int i = 0; i < 1; i++) {  
 try{  
 jsonObject = response.getJSONObject(i);  
 if(!jsonObject.get("fecha").toString().equals("0000-00-00")){  
 ultimoRegistro.setText(jsonObject.get("fecha").toString());  
 }  
 }catch (JSONException je) {  
 Log.*e*("ERROR\_JSON\_FECHA",je.getMessage());  
 }  
 }  
 }  
 }  
 }, new Response.ErrorListener() {  
 @Override  
 public void onErrorResponse(VolleyError error) {  
 Log.*e*("ERROR\_CONEXION\_FECHA",error.getMessage());  
 }  
 });  
 RequestQueue requestQueue = Volley.*newRequestQueue*(getContext());  
 requestQueue.add(jsonArrayRequest);  
 }  
}

**Captura de pantalla de un celular

Descripción generada automáticamente-----------------------------AdministrarMascarillasFragment-------------------------------------**

package com.example.proyectopst.ui.AdministrarMascarillas;  
  
import android.os.Bundle;  
import android.util.Log;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.AdapterView;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.annotation.NonNull;  
import androidx.annotation.Nullable;  
import androidx.fragment.app.Fragment;  
import androidx.lifecycle.Observer;  
import androidx.lifecycle.ViewModelProviders;  
  
import com.android.volley.AuthFailureError;  
import com.android.volley.Request;  
import com.android.volley.RequestQueue;  
import com.android.volley.Response;  
import com.android.volley.VolleyError;  
import com.android.volley.toolbox.JsonArrayRequest;  
import com.android.volley.toolbox.StringRequest;  
import com.android.volley.toolbox.Volley;  
import com.example.proyectopst.MenuPrincipal1;  
import com.example.proyectopst.R;  
  
import org.json.JSONArray;  
import org.json.JSONException;  
import org.json.JSONObject;  
  
import java.util.HashMap;  
import java.util.Map;  
  
public class AdministrarMascarillasFragment extends Fragment {  
  
 private AdministracionViewModel administracionViewModel;  
  
 public View onCreateView(@NonNull LayoutInflater inflater,  
 ViewGroup container, Bundle savedInstanceState) {  
 administracionViewModel =  
 ViewModelProviders.*of*(this).get(AdministracionViewModel.class);  
 View root = inflater.inflate(R.layout.*fragment\_administrar\_mascarillas*, container, false);  
 final TextView textView = root.findViewById(R.id.*text\_slideshow*);  
 final EditText codigo = root.findViewById(R.id.*editTextCodigo*);  
 final TextView numMascarillas = root.findViewById(R.id.*textViewNumMascarillas*);  
 consultaMascarilla(numMascarillas);  
 final Button agregar = root.findViewById(R.id.*buttonAgregarMasc*);  
 agregar.setOnClickListener(new AdapterView.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 agregarMascarilla(codigo,numMascarillas,"No");  
 }  
 });  
 administracionViewModel.getText().observe(getViewLifecycleOwner(), new Observer<String>() {  
 @Override  
 public void onChanged(@Nullable String s) {  
 textView.setText(s);  
 }  
 });  
  
  
 final Button eliminar = root.findViewById(R.id.*buttonEliminarMasc*);  
 eliminar.setOnClickListener(new AdapterView.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 eliminarRestaurarMascarilla(codigo,numMascarillas,"No");  
 }  
 });  
 administracionViewModel.getText().observe(getViewLifecycleOwner(), new Observer<String>() {  
 @Override  
 public void onChanged(@Nullable String s) {  
 textView.setText(s);  
 }  
 });  
  
  
 final Button restaurar = root.findViewById(R.id.*buttonRestablecerMasc*);  
 restaurar.setOnClickListener(new AdapterView.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 restaurarMascarilla(codigo,numMascarillas);  
 }  
 });  
 administracionViewModel.getText().observe(getViewLifecycleOwner(), new Observer<String>() {  
 @Override  
 public void onChanged(@Nullable String s) {  
 textView.setText(s);  
 }  
 });  
 return root;  
 }  
  
 */\*\*  
 \* Agrega una mascarilla a la base de datos.  
 \* Verifica si el código ingresado ya se encuentra registrado.  
 \** ***@param*** *codigo EditText donde el usuario ingresa el código de la mascarilla.  
 \** ***@param*** *cantidad TextView donde se setea la cantidad de mascarillas que tiene actualmente el usuario después de agregar una mascarilla.  
 \** ***@param*** *actualizar String que indica si la función es utilizada para solo agregar o resetear la mascarilla.  
 \*/* public void agregarMascarilla(final EditText codigo, final TextView cantidad, final String actualizar){  
 if(codigo.getText().toString().length()<3){  
 Toast.*makeText*(getContext(), "El código no es válido.", Toast.*LENGTH\_SHORT*).show();  
 }else{  
 Log.*i*("VERIFICACION\_1",codigo.getText().toString());  
 JsonArrayRequest jsonArrayRequestVerif = new JsonArrayRequest("https://undried-modes.000webhostapp.com/consulta\_mascarillas\_totales.php?codigo\_mascarilla="+codigo.getText(),  
 new Response.Listener<JSONArray>() {  
 @Override  
 public void onResponse(JSONArray response) {  
 JSONObject jsonObject = null;  
 for (int i = 0; i < response.length(); i++) {  
 try {  
 jsonObject = response.getJSONObject(i);  
 if(jsonObject.get("codigo\_mascarilla").toString().equals(codigo.getText().toString())){  
 Toast.*makeText*(getContext(), "Mascarilla ya registrada.", Toast.*LENGTH\_SHORT*).show();  
 codigo.setText("");  
 }  
 } catch (JSONException je) {  
 Log.*e*("ERROR\_JSON",je.getMessage());  
 }  
 }  
 }  
 }, new Response.ErrorListener() {  
 @Override  
 public void onErrorResponse(VolleyError error) {  
 Log.*e*("ERROR\_CONEXIONDB",error.getMessage());  
  
 StringRequest strRq = new StringRequest(Request.Method.*POST*,  
 "https://undried-modes.000webhostapp.com/ingreso\_mascarilla.php",  
 new Response.Listener<String>() {  
 @Override  
 public void onResponse(String response) {  
 if(actualizar.equals("No")){  
 Toast.*makeText*(getContext(), "Registro exitoso.", Toast.*LENGTH\_SHORT*).show();  
 cantidad.setText(Integer.*toString*(Integer.*parseInt*(cantidad.getText().toString())+1));  
 codigo.setText("");  
 }  
 }  
 }, new Response.ErrorListener() {  
 @Override  
 public void onErrorResponse(VolleyError error) {  
 Log.*e*("ERROR\_CONEXION\_AGREGAR",error.getMessage());  
 Toast.*makeText*(getContext(),"Ha ocurrido un error", Toast.*LENGTH\_SHORT*).show();  
 codigo.setText("");  
 }  
 }){  
 @Override  
 protected Map<String, String> getParams() throws AuthFailureError {  
 Map<String, String> parametros = new HashMap<String, String>();  
 parametros.put("id\_usuario",MenuPrincipal1.*id\_usuario*);  
 Log.*i*("VERIFICACION\_ANTES",codigo.getText().toString());  
 parametros.put("codigo\_mascarilla",codigo.getText().toString());  
 return parametros;  
 }  
 };  
 RequestQueue requestQueue = Volley.*newRequestQueue*(getContext());  
 requestQueue.add(strRq);  
 }  
 });  
 RequestQueue requestQueueVerif = Volley.*newRequestQueue*(getContext());  
 requestQueueVerif.add(jsonArrayRequestVerif);  
 }  
 }  
  
 */\*\*  
 Elimina una mascarilla a la base de datos.  
 \* Verifica si el código ingresado ya se encuentra registrado.  
 \* Utiliza el método agregarMascarilla para volverla a ingresar en la base de datos.  
 \** ***@param*** *codigo EditText donde el usuario ingresa el código de la mascarilla.  
 \** ***@param*** *cantidad TextView donde se setea la cantidad de mascarillas que tiene actualmente el usuario después de borrar una mascarilla.  
 \** ***@param*** *actualizar String que indica si la función es utilizada solo para eliminar o restaurar las estadísticas de la mascarilla.  
 \*/* public void eliminarRestaurarMascarilla(final EditText codigo, final TextView cantidad, final String actualizar){  
 if(codigo.getText().toString().length()<3){  
 Toast.*makeText*(getContext(), "El código no es válido.", Toast.*LENGTH\_SHORT*).show();  
 }else{  
 JsonArrayRequest jsonArrayRequestVerif = new JsonArrayRequest("https://undried-modes.000webhostapp.com/consulta\_datos\_mascarilla.php?id\_usuario="+ MenuPrincipal1.*id\_usuario*,  
 new Response.Listener<JSONArray>() {  
 @Override  
 public void onResponse(JSONArray response) {  
 JSONObject jsonObject = null;  
 int contador =0;  
 for (int i = 0; i < response.length(); i++) {  
 try {  
 jsonObject = response.getJSONObject(i);  
 Log.*i*("ID\_MASCBD",jsonObject.get("codigo\_mascarilla").toString());  
 Log.*i*("ID\_MASCBD2","Hay "+codigo.getText().toString());  
 if(jsonObject.get("codigo\_mascarilla").toString().equals(codigo.getText().toString())){  
 contador++;  
 Log.*i*("MASCARILLA\_IGUAL",jsonObject.get("codigo\_mascarilla").toString());  
  
 /\*----------------------------------------ELIMINAR DATOS-----------------------------------------\*/  
 StringRequest strRq = new StringRequest(Request.Method.*POST*,  
 "https://undried-modes.000webhostapp.com/eliminar\_mascarilla.php",  
 new Response.Listener<String>() {  
 @Override  
 public void onResponse(String response) {  
 if(actualizar.equals("No")){  
 Toast.*makeText*(getContext(), "Mascarilla eliminada exitosamente.", Toast.*LENGTH\_SHORT*).show();  
 cantidad.setText(Integer.*toString*(Integer.*parseInt*(cantidad.getText().toString())-1));  
 }/\*else{  
 agregarMascarilla(codigo,cantidad,"Si");  
 Toast.makeText(getContext(), "Mascarilla restaurada.", Toast.LENGTH\_SHORT).show();  
 }\*/  
 //codigo.setText("");  
 }  
 }, new Response.ErrorListener() {  
 @Override  
 public void onErrorResponse(VolleyError error) {  
 Toast.*makeText*(getContext(),"No se pudo eliminar la mascarilla.", Toast.*LENGTH\_SHORT*).show();  
 }  
 }){  
 @Override  
 protected Map<String, String> getParams() throws AuthFailureError {  
 Map<String, String> parametros = new HashMap<String, String>();  
 parametros.put("codigo\_mascarilla",codigo.getText().toString());  
 return parametros;  
 }  
 };  
 RequestQueue requestQueue = Volley.*newRequestQueue*(getContext());  
 requestQueue.add(strRq);  
 }  
 } catch (JSONException je) {  
 Log.*e*("ERROR\_JSON",je.getMessage());  
 }  
 }  
 if(contador==0){  
 Log.*e*("CONT","CONTADOR VACIO");  
 Toast.*makeText*(getContext(), "Mascarilla no encontrada.", Toast.*LENGTH\_SHORT*).show();  
 codigo.setText("");  
 }  
 }  
 }, new Response.ErrorListener() {  
 @Override  
 public void onErrorResponse(VolleyError error) {  
 Log.*e*("ERROR\_CONEXION",error.getMessage());  
 }  
 });  
 RequestQueue requestQueueVerif = Volley.*newRequestQueue*(getContext());  
 requestQueueVerif.add(jsonArrayRequestVerif);  
 if(actualizar.equals("Si")){  
 agregarMascarilla(codigo,cantidad,"Si");  
 Toast.*makeText*(getContext(), "Mascarilla restaurada.", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 }  
  
 */\*\*  
 \* Elimina y vuelve a agregar la mascarilla para eliminar cualquier registro de esta.  
 \* Utiliza el método eliminarRestaurarMascarilla.  
 \** ***@param*** *codigo EditText donde el usuario ingresa el código de la mascarilla.  
 \** ***@param*** *cantidad TextView donde se setea la cantidad de mascarillas que tiene actualmente el usuario.  
 \*/* public void restaurarMascarilla(final EditText codigo,final TextView cantidad){  
 eliminarRestaurarMascarilla(codigo,cantidad,"Si");  
 }  
  
 */\*\*  
 \* Obtiene de la base de datos la cantidad de mascarillas que posee el usuario.  
 \** ***@param*** *cantidadMascarillas TextView donde se setea la cantidad de mascarillas que tiene actualmente el usuario.  
 \*/* public void consultaMascarilla(final TextView cantidadMascarillas){  
 JsonArrayRequest jsonArrayRequest = new JsonArrayRequest("https://undried-modes.000webhostapp.com/consulta\_datos\_mascarilla.php?id\_usuario="+ MenuPrincipal1.*id\_usuario*,  
 new Response.Listener<JSONArray>() {  
 @Override  
 public void onResponse(JSONArray response) {  
 if(Integer.*toString*(response.length())!=null){  
 cantidadMascarillas.setText(Integer.*toString*(response.length()));  
 }else{  
 cantidadMascarillas.setText("0");  
 }  
 }  
 }, new Response.ErrorListener() {  
 @Override  
 public void onErrorResponse(VolleyError error) {  
 Log.*e*("ERROR\_CONEXION",error.getMessage());  
 }  
 });  
 RequestQueue requestQueue = Volley.*newRequestQueue*(getContext());  
 requestQueue.add(jsonArrayRequest);  
 }  
}

-----------------------------MisMascarillasFragment-------------------------------------

Consta de 3 java

**MisMascarillasFragment**

package com.example.proyectopst.ui.MisMascarillas;  
  
import android.content.Intent;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.AdapterView;  
import android.widget.ListView;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
import androidx.annotation.Nullable;  
import androidx.fragment.app.Fragment;  
import androidx.lifecycle.Observer;  
import androidx.lifecycle.ViewModelProviders;  
  
import com.android.volley.RequestQueue;  
import com.android.volley.Response;  
import com.android.volley.VolleyError;  
import com.android.volley.toolbox.JsonArrayRequest;  
import com.android.volley.toolbox.Volley;  
import com.example.proyectopst.MenuPrincipal1;  
import com.example.proyectopst.R;  
  
import org.json.JSONArray;  
import org.json.JSONException;  
import org.json.JSONObject;  
  
import java.util.ArrayList;  
import java.util.Hashtable;  
import java.util.Map;  
  
public class MisMascarillasFragment extends Fragment {  
 private ListView lista;  
 private MisMascarillasViewModel misMascarillasViewModel;  
 public int cantidad\_contactos = 0;  
  
 public View onCreateView(@NonNull LayoutInflater inflater,  
 ViewGroup container, Bundle savedInstanceState) {  
 misMascarillasViewModel =  
 ViewModelProviders.*of*(this).get(MisMascarillasViewModel.class);  
 View root = inflater.inflate(R.layout.*fragment\_mis\_mascarillas*, container, false);  
 final TextView textView = root.findViewById(R.id.*text\_gallery*);  
 final ListView lista = root.findViewById(R.id.*listViewContenedorMascarillas*);  
 consultaMascarillaUsuario(lista);  
 misMascarillasViewModel.getText().observe(getViewLifecycleOwner(), new Observer<String>() {  
 @Override  
 public void onChanged(@Nullable String s) {  
 textView.setText(s);  
 }  
 });  
 return root;  
 }  
  
 */\*\*  
 \* Recibe de la base de datos, la información de todas las mascarillas pertenecientes a un usuario para mostrarlas junto con su información correspondiente.  
 \** ***@param*** *mascarillas ListView donde se mostrarán las mascarillas.  
 \*/* public void consultaMascarillaUsuario(final ListView mascarillas){  
 JsonArrayRequest jsonArrayRequest = new JsonArrayRequest("https://undried-modes.000webhostapp.com/prueba\_busqueda\_mascarilla.php?id\_usuario="+MenuPrincipal1.*id\_usuario*,  
 new Response.Listener<JSONArray>() {  
 @Override  
 public void onResponse(JSONArray response) {  
 JSONObject jsonObject = null;  
 if(response.length()!=0){  
 String[][] datos\_adaptador = new String[response.length()][4];  
 ArrayList<String> datosCompletos = new ArrayList<>();  
 for (int i = 0; i < response.length(); i++) {  
 try {  
 jsonObject = response.getJSONObject(i);  
 datos\_adaptador[i][0]=jsonObject.get("codigo\_mascarilla").toString();  
 datos\_adaptador[i][1]=jsonObject.get("seguro").toString();  
 datosCompletos.add(jsonObject.get("codigo\_mascarilla").toString());  
 datosCompletos.add(jsonObject.get("fecha").toString());  
 datosCompletos.add(jsonObject.get("distancia").toString());  
 if(jsonObject.get("seguro").toString().equals("0")){  
 datosCompletos.add("No");  
 }else{  
 datosCompletos.add("Sí");  
 }  
 } catch (JSONException je) {  
 Log.*e*("ERROR\_JSON",je.getMessage());  
 }  
 }  
 Hashtable<String, Integer> personasyEdades = new Hashtable<String, Integer>();  
 for (int x=0; x < datos\_adaptador.length; x++){  
 int pruebaValor = 0;  
 for (Map.Entry<String, Integer> entry : personasyEdades.entrySet()) {  
 if(entry.getKey().equals(datos\_adaptador[x][0])){  
 pruebaValor = 1;  
 }  
 }  
 for (Map.Entry<String, Integer> entry : personasyEdades.entrySet()) {  
 }  
 if(pruebaValor==1){  
 if(datos\_adaptador[x][1].equals("0")){  
 int numero = personasyEdades.get(datos\_adaptador[x][0]) + 1;  
 personasyEdades.put(datos\_adaptador[x][0], numero);  
 }  
 }  
 else{  
 if(datos\_adaptador[x][1].equals("0")){  
 personasyEdades.put(datos\_adaptador[x][0], 1);  
 }else{  
 personasyEdades.put(datos\_adaptador[x][0], 0);  
 }  
 }  
 }  
 String [][] listaEnviar= new String[personasyEdades.size()][4];  
 int contador = 0;  
 for (Map.Entry<String, Integer> entry : personasyEdades.entrySet()) {  
 listaEnviar[contador][0] = Integer.*toString*(contador+1);  
 listaEnviar[contador][1] = entry.getKey();  
 listaEnviar[contador][2] = Integer.*toString*(entry.getValue());  
 int numero = entry.getValue();  
 numero /= 1.5;  
 if(numero>=100){  
 numero = 100;  
 }  
 listaEnviar[contador][3] = Integer.*toString*(numero);  
 contador++;  
 }  
 mostrarMascarillas(listaEnviar,mascarillas,datosCompletos);  
 }  
 }  
 }, new Response.ErrorListener() {  
 @Override  
 public void onErrorResponse(VolleyError error) {  
 Log.*e*("ERROR\_CONEXION","consultaMU -- "+error.getMessage());  
 String[][] datos\_adaptador = new String[1][4];  
 datos\_adaptador[0] = new String[]{"0", "0", "0", "0"};  
 ArrayList<String> datos = new ArrayList<>();  
 datos.add("0");  
 datos.add("0000-00-00");  
 datos.add("0");  
 datos.add("0");  
 mostrarMascarillas(datos\_adaptador,mascarillas,datos);  
 }  
 });  
 RequestQueue requestQueue = Volley.*newRequestQueue*(getContext());  
 requestQueue.add(jsonArrayRequest);  
 }  
  
 */\*\*  
 \* Ingresa los datos de las mascarillas en su respectivo ListView.  
 \* Determina qué información se va a mostrar cuando se presione sobre una de las mascarillas.  
 \** ***@param*** *datos  
 \** ***@param*** *lista  
 \** ***@param*** *informacionMascarilla  
 \*/* public void mostrarMascarillas(final String[][] datos, ListView lista,final ArrayList<String> informacionMascarilla){  
 lista.setAdapter(new Adaptador(getContext(), datos,datos.length));  
 lista.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
 @Override  
 public void onItemClick(AdapterView<?> adapterView, View view, int posicion, long id) {  
 Intent detalles = new Intent(view.getContext(),InformacionMascarilla.class);  
 detalles.putExtra("Codigo",datos[posicion][1]);  
 detalles.putExtra("Info",informacionMascarilla);  
 startActivity(detalles);  
 }  
 });  
 }  
}

**Adaptador** (Para mostrar cada mascarilla)

Captura de pantalla de un celular

Descripción generada automáticamentepackage com.example.proyectopst.ui.MisMascarillas;  
  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.BaseAdapter;  
import android.widget.TextView;  
  
import com.example.proyectopst.R;  
  
public class Adaptador extends BaseAdapter {  
 private static LayoutInflater *inflater* = null;  
  
 Context contexto;  
 String[][] datos;  
 int cantidad;  
  
 */\*\*  
 \* Constructor del adaptador que ingresa los datos de la mascarilla en el layout elemento\_lista.  
 \** ***@param*** *contexto Contexto del activity.  
 \** ***@param*** *datos Información que contendrán los elementos de la lista.  
 \** ***@param*** *cantidad Cantidad de datos que se mostrarán.  
 \*/* public Adaptador(Context contexto, String[][] datos, int cantidad) {  
 this.contexto = contexto;  
 this.datos = datos;  
 this.cantidad = cantidad;  
 *inflater* = (LayoutInflater) contexto.getSystemService(contexto.*LAYOUT\_INFLATER\_SERVICE*);  
 }  
  
 */\*\*  
 \* Ingresa los datos obtenidos en su respectivo contenedor.  
 \** ***@param*** *i  
 \** ***@param*** *view  
 \** ***@param*** *viewGroup  
 \** ***@return*** *\*/* @Override  
 public View getView(int i, View view, ViewGroup viewGroup) {  
 final View vista = *inflater*.inflate(R.layout.*elemento\_lista*, null);  
 TextView Num = (TextView) vista.findViewById(R.id.*textViewNumMascarilla*);  
 TextView Cod = (TextView) vista.findViewById(R.id.*textViewCodigoMascarilla*);  
 TextView Con = (TextView) vista.findViewById(R.id.*textViewCantidadContactos*);  
 TextView Uso = (TextView) vista.findViewById(R.id.*textViewUsoMascarilla*);  
 if(datos[i][1].equals("0")){  
 Cod.setText("Agregue una mascarilla.");  
 }else{  
 Cod.setText("Código: "+datos[i][1]);  
 }  
 Num.setText("Mascarilla N°"+datos[i][0]);  
 Con.setText("Cantidad de contactos: "+datos[i][2]);  
 Uso.setText("Uso: "+datos[i][3]+"%");  
 return vista;  
 }  
  
 */\*\*  
 \*  
 \** ***@return*** *\*/* @Override  
 public int getCount() {  
 return cantidad;  
 }  
  
 */\*\*  
 \*  
 \** ***@param*** *i  
 \** ***@return*** *\*/* @Override  
 public Object getItem(int i) {  
 return null;  
 }  
  
 */\*\*  
 \*  
 \** ***@param*** *i  
 \** ***@return*** *\*/* @Override  
 public long getItemId(int i) {  
 return 0;  
 }  
  
}

**InformaciónMascarilla** (Para mostrar la tabla con fechas y si es seguro o no)

Captura de pantalla de un celular

Descripción generada automáticamentepackage com.example.proyectopst.ui.MisMascarillas;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.Intent;  
import android.os.Bundle;  
import android.util.DisplayMetrics;  
import android.widget.TableLayout;  
import android.widget.TextView;  
  
import com.example.proyectopst.R;  
import com.example.proyectopst.TablaDinamica;  
  
import java.util.ArrayList;  
  
public class InformacionMascarilla extends AppCompatActivity {  
 TextView titulo;  
 private TableLayout consultaTabla;  
  
 */\*\*  
 \* Trata la información recibida en un bundle para mostrarla en una ventana emergente.  
 \** ***@param*** *savedInstanceState  
 \*/* @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_informacion\_mascarilla*);  
 titulo = (TextView)findViewById(R.id.*textViewCodigoMascDetalles*);  
 consultaTabla = (TableLayout) findViewById(R.id.*tablaDatos*);  
 Intent datos = getIntent();  
 Bundle b = datos.getExtras();  
  
 DisplayMetrics medidaVentana = new DisplayMetrics();  
 getWindowManager().getDefaultDisplay().getMetrics(medidaVentana);  
 int ancho = medidaVentana.widthPixels;  
 int alto = medidaVentana.heightPixels;  
 getWindow().setLayout((int)(ancho\*0.8),(int)(alto\*0.5));  
  
 titulo.setText("Mascarilla: "+b.getString("Codigo"));  
  
 TablaDinamica tablaD = new TablaDinamica(consultaTabla,getApplicationContext());  
 String[] header = new String[]{"Fecha", "Distancia [cm]", "Seguro"};  
 tablaD.addHeader(header);  
 ArrayList<String> datosCompletos = b.getStringArrayList("Info");  
 ArrayList<String[]> contenido = new ArrayList<String[]>();  
 for(int i=0; i<datosCompletos.size()-3;i++){  
 if(datosCompletos.get(i).equals(b.getString("Codigo"))){  
 if(!datosCompletos.get(i+1).equals("0000-00-00")){  
 String[] cont = new String[3];  
 cont[0] = datosCompletos.get(i+1);  
 cont[1] = datosCompletos.get(i+2);  
 cont[2] = datosCompletos.get(i+3);  
 contenido.add(cont);  
 }  
 }  
 }  
 if(contenido.size()==0){  
 contenido.add(new String[]{"No hay datos para mostrar."});  
 }  
 tablaD.addData(contenido);  
 }  
}

**TablaDinamica** (Para construir la tabla y darle formato)

package com.example.proyectopst;  
  
import android.content.Context;  
import android.graphics.Color;  
import android.view.Gravity;  
import android.widget.TableLayout;  
import android.widget.TableRow;  
import android.widget.TextView;  
  
import java.util.ArrayList;  
  
public class TablaDinamica {  
 private TableLayout tableLayout;  
 private Context context;  
 private String[] header;  
 private ArrayList<String[]> data;  
 private TableRow tableRow;  
 private TextView txtCell;  
 private int indexC;  
 private int indexR;  
  
 */\*\*  
 \* Constructor de la clase.  
 \** ***@param*** *tableLayout donde va a estar contenida la tabla.  
 \** ***@param*** *context del activity.  
 \*/* public TablaDinamica(TableLayout tableLayout, Context context){  
 this.tableLayout = tableLayout;  
 this.context = context;  
 }  
  
 */\*\*  
 \* Crea la cabecera de la tabla.  
 \** ***@param*** *header  
 \*/* public void addHeader(String[] header){  
 this.header = header;  
 createHeader();  
 }  
  
 */\*\*  
 \* Añade los datos ingresados en el ArrayList.  
 \** ***@param*** *data  
 \*/* public void addData(ArrayList<String[]> data){  
 this.data = data;  
 createDataTable();  
 }  
  
 */\*\*  
 \* Crea una nueva fila.  
 \*/* private void newRow(){  
 tableRow = new TableRow(context);  
 }  
  
 */\*\*  
 \* Crea un nuevo contenedor tipo textView para la celda.  
 \*/* private void newCell(){  
 txtCell = new TextView(context);  
 txtCell.setGravity(Gravity.*CENTER*);  
 txtCell.setTextSize(12);  
 }  
  
 */\*\*  
 \* Ingresa los datos de la cabecera en la tabla.  
 \*/* private void createHeader(){  
 indexC = 0;  
 newRow();  
 for(String datos: header){  
 newCell();  
 txtCell.setText(datos);  
 txtCell.setBackgroundColor(Color.*parseColor*("#7992CF"));  
 tableRow.addView(txtCell,newTableRowParams());  
 }  
 tableLayout.addView(tableRow);  
 }  
  
 */\*\*  
 \* Ingresa los datos del contenido en la tabla.  
 \*/* private void createDataTable(){  
 for(String[] datos2: data) {  
 newRow();  
 for (String datos3: datos2) {  
 newCell();  
 txtCell.setText(datos3);  
 txtCell.setBackgroundColor(Color.*parseColor*("#C3D1F3"));  
 tableRow.addView(txtCell, newTableRowParams());  
 }  
 tableLayout.addView(tableRow);  
 }  
 }  
  
 */\*\*  
 \* Setea los márgenes de la tabla.  
 \** ***@return*** *parámetros de la tabla.  
 \*/* public TableRow.LayoutParams newTableRowParams(){  
 TableRow.LayoutParams params = new TableRow.LayoutParams();  
 params.setMargins(1,1,1,1);  
 params.weight = 1;  
 return params;  
 }  
}

**Arduino**

1. #ifdef ESP32
2. #include <WiFi.h>
3. #include <HTTPClient.h>
4. #else
5. #include <ESP8266WebServer.h>
6. #include <ESP8266WiFi.h>
7. #include <ESP8266HTTPClient.h>
8. #include <WiFiClient.h>
9. #endif
11. #include <Wire.h>
12. #include <Servo.h>
14. Servo myServo;
15. static const int pinServo= 15; //D8 pin del ESP8226
16. const char\* ssid = "Nombre de la red";
17. const char\* password = "contraseña";
19. #define TRIGGER 16  //D0 pin del ESP8226
20. #define ECHO    5   //D1 pin del ESP8226
22. int distanciamax= 200; // distancia máxima de 2 metros con respecto a una persona
23. long duracion=0;
24. int16\_t distancia= 0;
25. uint8\_t contador = 0;
27. const char\* serverName = "http://ejemplo.com/ejemplo.php"; //link para hacer la conexión al Servidor
28. String apiKeyValue = "tPmAT5Ab3j7F9";
30. int mascarilla= 1234; //codigo de la máscarilla
31. String sensorName = "Ultrasonico";
32. String msj="";
33. boolean isSeguro=true;
34. boolean isSeguro2=true;
35. int comienzo=0;

38. void setup(void){
40. Serial.begin(115200);
41. myServo.attach(pinServo);
42. myServo.write(180); // la compuerta se cierra apenas se inicia el sistema
43. delay(100);
44. pinMode(TRIGGER, OUTPUT);
45. pinMode(ECHO, INPUT);                 // Se establecen los pines de salida o entrada
46. pinMode(BUILTIN\_LED, OUTPUT);
48. WiFi.begin(ssid, password);   // Se realiza la conexión wifi
49. Serial.println("Connecting");
50. while(WiFi.status() != WL\_CONNECTED) {
51. delay(500);
52. Serial.print(".");
53. }                                                                     // Se asegura que la conexión wifi sea exitosa
54. Serial.println("");
55. Serial.print("Connected to WiFi network with IP Address: ");
56. Serial.println(WiFi.localIP());
57. }
59. void loop() {
60. digitalWrite(TRIGGER, LOW);
61. delayMicroseconds(2);
62. digitalWrite(TRIGGER, HIGH);
63. delayMicroseconds(10);                      //Se realiza la conversión de la distancia a cm
64. digitalWrite(TRIGGER, LOW);
65. duracion = pulseIn(ECHO, HIGH);
66. distancia = (duracion/2) / 29.1;
68. if (distancia < 0) distancia = 0;
70. if (distancia < distanciamax) {
71. isSeguro=false;
72. }
73. else{
74. isSeguro=true;
75. }
77. if (comienzo==0){
78. isSeguro2=not isSeguro;
79. }
80. if (isSeguro!=isSeguro2) {           //Se envía datos cuando el estado de la persona cambie
81. if (distancia < distanciamax){
82. myServo.write(180);
83. }
84. if (distancia >= distanciamax) {
85. myServo.write(-180);
86. }
87. msj = String(distancia);
88. delay(500);
89. if(WiFi.status()== WL\_CONNECTED){
91. HTTPClient http;
92. http.begin(serverName); // Se hace la conexión con el servidor
94. http.addHeader("Content-Type", "application/x-www-form-urlencoded"); // Tipo de dato que se enviarán
96. String httpRequestData = "api\_key=" + apiKeyValue +
97. "&codigo\_mascarilla="+ String(mascarilla) + "&distancia=" + msj +  //Se crea un String que tendrá toda la informacion a enviar
98. "&seguro=" + String(isSeguro)+ "";
99. Serial.print("httpRequestData: ");
100. Serial.println(httpRequestData);
101. int httpResponseCode = http.POST(httpRequestData);  // se crea un int que nos arrojará un número de como fue la petición
103. if (httpResponseCode>0) { // Lo que se espera es un número de 200, la cual indica que está todo "ok".
104. Serial.print("HTTP Response code: ");
105. Serial.println(httpResponseCode);
106. }
107. else {
108. Serial.print("Error code: ");
109. Serial.println(httpResponseCode);
110. }
112. http.end(); //Se termina la conexión con el servidor
113. }
114. else {
115. Serial.println("WiFi Disconnected");
116. }
118. delay(30000);                 // Se espera 30 segundos por cada envío de dato
120. }
121. Serial.println(isSeguro!= isSeguro2);
122. Serial.println(distancia);
123. isSeguro2 = isSeguro;
124. comienzo++;
125. }

**Código PHP para obtener la información que envía el módulo wifi.**

1. **<?php**
2. $servername = "localhost";
3. $username = "Nombre de usuario";
4. $password = "contraseña";
5. $dbname = "Nombre de la base de datos";
7. $api\_key\_value = "tPmAT5Ab3j7F9";
9. if ($\_SERVER["REQUEST\_METHOD"] == "POST") {
10. //Obtener la clave enviada por el modulo wifi
11. $api\_key = test\_input($\_POST["api\_key"]);
12. if($api\_key == $api\_key\_value) { // si son iguales entonces se lee los datos enviados
13. $distancia = test\_input($\_POST["distancia"]);
14. $codigo\_mascarilla = test\_input($\_POST["codigo\_mascarilla"]);   //Se almacenan los datos en variables
15. $seguro = test\_input($\_POST["seguro"]);
16. // Crear la conexión
17. $conn = **new** mysqli($servername, $username, $password, $dbname);
18. if ($conn->connect\_error) {
19. [die](http://www.php.net/die)("Connection failed: " . $conn->connect\_error);
20. }
22. // Se crea una variable de como se enviarán los datos para el query
23. $sql = "INSERT INTO DatoSensor (codigo\_mascarilla, distancia, seguro, fecha)
24. VALUES ('".$codigo\_mascarilla."', '".$distancia."' , '".$seguro."', CURRENT\_DATE )";
26. // Se insertan los datos en una tabla general que tendrá los triggers para que se llenen las tablas
27. // del diagrama entidad relación
29. if ($conn->query($sql) === **TRUE**) {
30. echo "New record created successfully";
31. }
32. else {
33. echo "Error: " . $sql . "<br>" . $conn->error;
34. }
36. $conn->close();
37. }
38. else {
39. echo "Wrong API Key provided.";
40. }
42. }
43. else {
44. echo "No data posted with HTTP POST.";
45. }
47. **function** test\_input($data) {
48. $data = [trim](http://www.php.net/trim)($data);
49. $data = [stripslashes](http://www.php.net/stripslashes)($data);
50. $data = [htmlspecialchars](http://www.php.net/htmlspecialchars)($data);
51. return $data;
52. }
54. **?>**
55. **Análisis de presupuesto**

* Módulo WIFI $ 12.00
* Sensor ultrasónico $ 3.00
* ServoMotor $ 3.00
* Mascarilla $ 3.00
* Cables (paquete) $ 1

Total $22

Los precios mencionados anteriormente son en base a la tienda electrónica Castro, pero en realidad sólo se gastó en la mascarilla y en el módulo WIFI, ya que en anteriores proyectos de otras materias se usó el sensor ultrasónico, servomotor y los cables, casi todos los componentes se reutilizaron para este proyecto, por lo que sólo se gastó $15.

1. **Conclusiones**

-Se logró la obtención de datos específicos sobre la distancia de las personas respecto a cada usuario mediante la mascarilla, estableciendo si este se encontraba protegido o no.

- Se estableció una conexión inalámbrica entre los dispositivos que involucran el desarrollo de la mascarilla inteligente, estos son la mascarilla, base de datos y una aplicación móvil para llevar un registro de la información.

-El proyecto pemitió desarrollar una visión de cómo aplicar la programación y electrónica para la resolución de problemas de salud, involucrando ciencia y tecnología de manera remota y con componentes de bajo costo.

# Bibliografía

# Dominguez, R. [Skrium]. (2018, febrero 13). IOT NodeMCU ESP8266 - Sensor Ultrasonic WebSocket WebServer [Archivo de video]. Recuperado de <https://www.youtube.com/watch?v=N88l7FsYi50>

* *Reddit*. (26 de 5 de 2020). Obtenido de https://www.reddit.com/r/3Dmodeling/comments/ggbt0b/3d\_model\_of\_face\_mask\_designed\_using\_autodesk/

# Santos, N. (2017, Mayo 19). ESP32: HTTP GET Requests. Obtenido de: <https://techtutorialsx.com/2017/05/19/esp32-http-get-requests/>