SLF-MANAGEMENT

PROYECTO DE DESARROLLO DE APLICACIONES MULTIPLATAFORMA

CURSO 2019/20

JUAN CARLOS ALARCÓN PEDRAZA

CENEC MALAGA

ÍNDICE

1.- Descripción

Esta app es un software de sistema de gestión de empresa dirigida a eventos, aunque es válida para cualquier tipo de empresa similar.

Englobando su funcionalidad, permite llevar un registro de logística, eventos y personal de la empresa.

Dada a su Base de Datos totalmente enlazada, el administrador puede saber en todo momento lo que está sucediendo con su empresa.
Al tener todo lo necesario en una misma aplicación móvil, hace de la administración y organización de los eventos sea mucho más fluida.

Está aplicación debería ir acompañada de un programa para ordenador.
Esta serviría para poder usar al 100% las posibilidades que te ofrece la app.

2.- Análisis del Mercado Actual

Público objetivo

La aplicación que estoy desarrollando (*SLF-MANAGEMENT*), es un app de Sistema de Gestión de Empresas dirigida a las empresas dedicadas a eventos.

Esta permite controlar *logística*, *personal* y *eventos*. Dentro de estos hay un seguimiento:

> ***-Logística***: Saber si está activo (en el caso de almacenar material antiguo), control a tiempo real de quien está usando tal material, comentarios sobre el mismo (ej: los empleados podrán comentar cualquier tipo de avería), registro de eventos en los cuales ha estado para llevar un seguimiento.
>
> ***-Eventos***: Tener en solo una lista todos los eventos disponibles, de forma que es mucho más sencillo controlar cada trabajo, dentro de cada evento se podrá ver toda su información, tales como: información del cliente, la contratación, personal adjudicado, información general del evento, material asignado y fichas técnicas.
>
> ***-Personal***: Control de actividad, disponibilidad, pagos pendientes, eventos realizados.

Aun estando orientada a este tipo de empresa, realmente es útil para la gran mayoría de oficios.

En el caso de que una empresa ofrezca una oferta de anhelar esta app con distintos cambios personalizados a la misma, se podría llegar a hacer.

Método de venta

La distribución de la aplicación será individualmente a las empresas, ofreciendo personalmente la venta de la misma, también colgaré un vídeo promocional en redes sociales y webs.

Posibilidades de venta

Actualmente la aplicación se está desarrollando para una empresa, es decir en el momento de finalización, se venderá a dicha empresa.

No obstante, conozco varias empresas similares las cuales estarían interesadas en disponer de este software

Aplicaciones similares

Tras investigar por varias tiendas online de aplicaciones, no he encontrado ninguna app que coincida al 100% con la idea que quiero transmitir, ya que es para solucionar unos problemas más específicos.

Aunque hay algunas que tratan de gestión de empresas, basadas más en ventas y facturación, incluyendo puntos que contiene mi aplicación.

Mi Negocio: Esta app incluye muchas opciones de ventas, compras, facturación, etc… si es muy parecida en el aspecto de que también almacena inventario, servicios y agenda.

eStock: Con esta aplicación se solucionaría uno de los problemas que mi idea también implementa, con la diferencia que esta se cierra solo en almacenamiento.

Agenda de eventos: Objetivamente no se asemeja a mi aplicación, aunque es la más cercana a la sección de administración de eventos, teniendo *SLF-MANAGEMENT* más opciones para los eventos.

3.- Propuesta Innovadora

La idea de este proyecto nace de la necesidad que vi como trabajador en una empresa de eventos, al tener mucho contacto con los dueños y ser encargado de muchos de sus sectores, conozco los problemas internos que tiene la empresa. Tras darle unas vueltas al concepto de incluir las aplicaciones, programas y herramientas que usan para la administración de la empresa, surgió la primera versión de este proyecto, la cual realicé como trabajo final en el primer año del grado Desarrollo Multiplataformas en el módulo de *Programación*.
Al ofrecerme este *Proyecto de fin de grado,* me pareció buena idea llevar esta idea al siguiente nivel, añadiendo una base de datos más completa, solucionando así todos los inconvenientes dados.

4.- Estudio de Viabilidad

En estos instantes hay 2 empresas interesadas en el proyecto y dispuestos a financiarlo con el propósito de finalmente comprarlo para siempre.

4.1.- DESCRIPCIÓN DEL MODELO DE NEGOCIO

Mi intención es vender el software a un precio fijo, sin anuncios ni micropagos integrados.
Aunque si el comprador le interesaría hacer cambios dentro de la propia aplicación para personalizarla a su empresa, habrían dos métodos.

Suscripción mensual: El cliente tendría soporte técnico de la aplicación el tiempo que haya contratado, esta opción sería la mas economica.

Cambios puntuales: En el momento que surja una necesidad de la app, se debería abonar una cantidad relevante al cambio en cuestión, esta opción sería efectiva en el caso de querer cambios mínimos.

4.2.- EMBUDO DE MARKETING

La app tendrá un vídeo presentación el cual se colgará en las redes sociales más utilizadas por este tipo de empresas, **Instagram** y **Facebook**. A las cuales se les pagaría por patrocinar el vídeo a personas relacionadas a el público objetivo.

El logotipo no es fijo, dependerá de la empresa la cual quiera usarla, ya que es una aplicación privada que solo los dueños podrán usar.

Dependiendo de la época del año, se irán haciendo promociones, ya que con cómo cada sector, tiene sus épocas de alto furor.
Ej. Añadir al precio original 3 meses de mantenimiento gratis.

La aplicación es fija, no debería de tener actualizaciones generales, sí se actualiza para corregir errores e implementar nuevas funcionalidades básicas.

La técnica para atraer a más personas es que la aplicación siempre será personalizada por el cliente con las opciones, colores, logotipos y funcionalidades adecuadas a sus necesidades.

4.3.- ESTIMACIÓN DE LA POBLACIÓN OBJETIVO

La intención que tengo con *SLF-MANAGEMENT* es mejorarla hasta tal punto de poder venderla a empresas que conozco que están interesadas en este software.

Una vez finalizada, se subirá a la **Play Store** como una versión de prueba, con esto se consigue que los interesados en obtenerla puedan experimentar la funcionalidad que ofrece y una vez satisfechos con el resultado, se podrán poner contacto conmigo para negociar lo cambios y suscripción.

Añadir publicidad a este tipo de app sería un error, ya que molestaría al usuario y afectaría en su decisión de seguir usando el software por la pesadez que provoca. Aunque no estaría cerrado a poner un cuadro de anuncio dentro de la app, de forma que no moleste la fluidez de utilización que tiene la misma, dentro de la versión de prueba. Así, adquiriendo la suscripción se eliminarían por completo la publicidad. Actualmente la app está en fase BETA, así que actualmente es sin ánimo de lucro si alguien quiere utilizar esta versión, ya que la funcionalidad no promete al 100% su rendimiento.

Una vez finalizada con todas sus implementaciones al completo, si estaría dentro del mercado para poder generar ingresos con este proyecto.

4.4.- CÁLCULO DE LA MUESTRA POBLACIONAL NECESARIA PARA LA ENCUESTA

Siguiendo la siguiente fórmula, he calculado la muestra poblacional.

$$n = \frac{N \times Z_a^{\,2} \times p \times q}{d^2 \times (N-1) + Z_a^{\,2} \times p \times q}$$

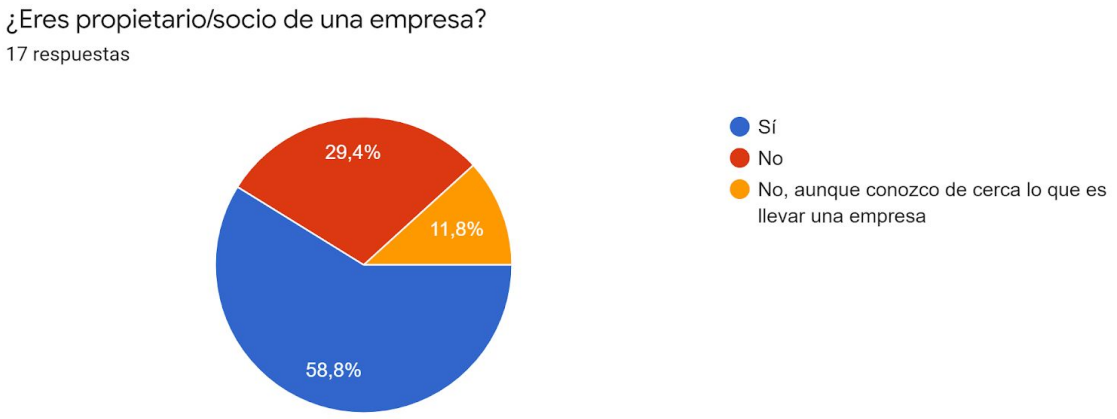
N: Tamaño máximo de la muestra, calculado en el punto anterior.
Z: nivel de confianza. Usa al menos 1,96 y como máximo 3.
P: Probabilidades de respuestas favorables. Normalmente 0.5
Q: Probabilidades de respuestas en contra. Normalmente 0.5
d: Precisión (Error máximo admisible)
n: Nuestro resultado, el número de personas que deberíamos entrevistar

N = 5.989,4
Z = 2
p = 0,5
q = 0,5
d = 0,1
n = 89,35719862478

4.5.- ENCUESTA Y RESULTADOS

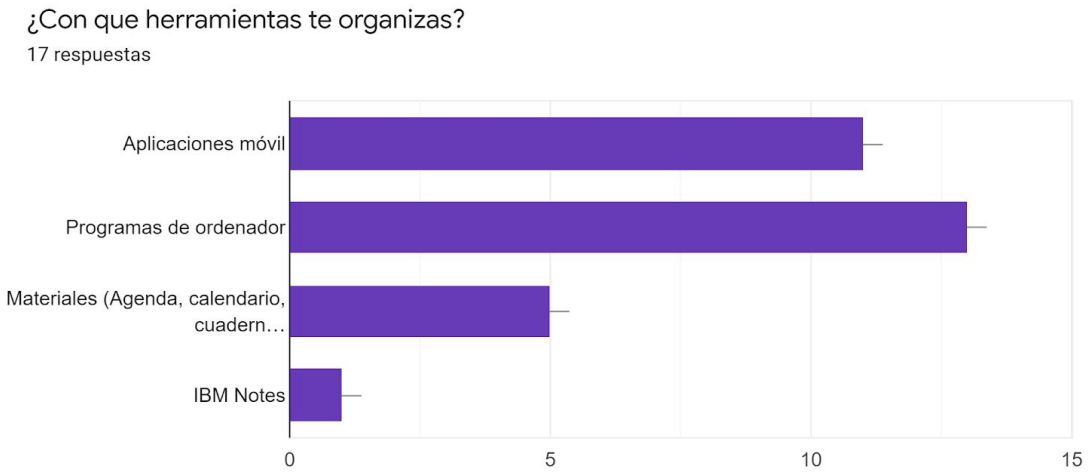
Encuesta: https://forms.gle/PgJFSHtMLuEkAqDp9

El público objetivo de la encuesta eran trabajadores y emprendedores y esta es la media de las personas que la han realizado.

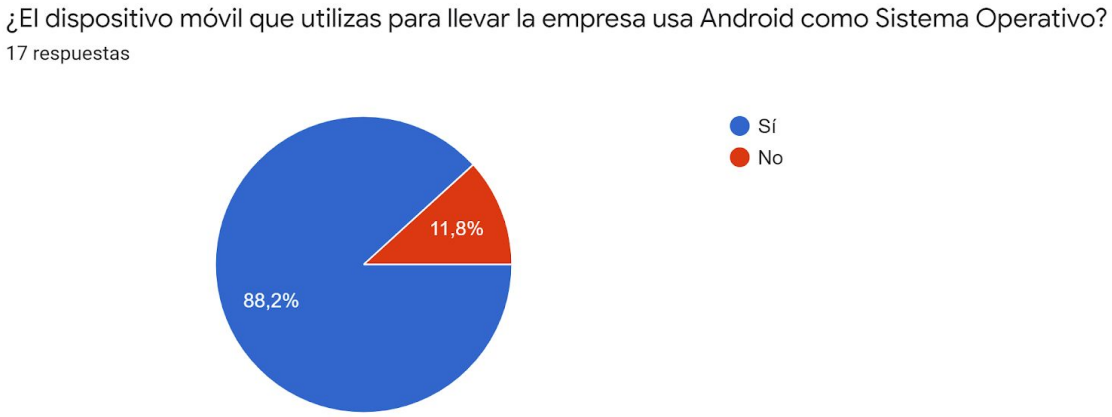


Como podemos comprobar la gran mayoría del personal ha sufrido problemas de todo tipo dentro de la empresa, problemas los cuales se podrían solucionar mi proyecto.

Los sistemas de organización digital lideran el ranking de gestión empresarial.



Los dispositivos móviles con el Sistema Operativo Android son los más usados para llevar la administración empresa.

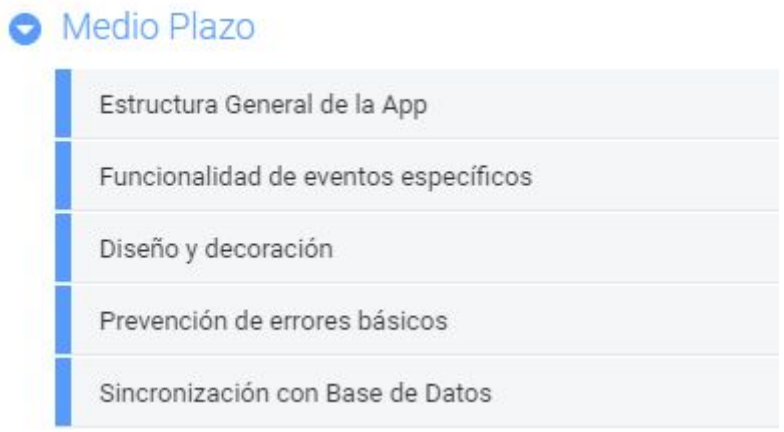


Los problemas más repetidos en la encuesta:

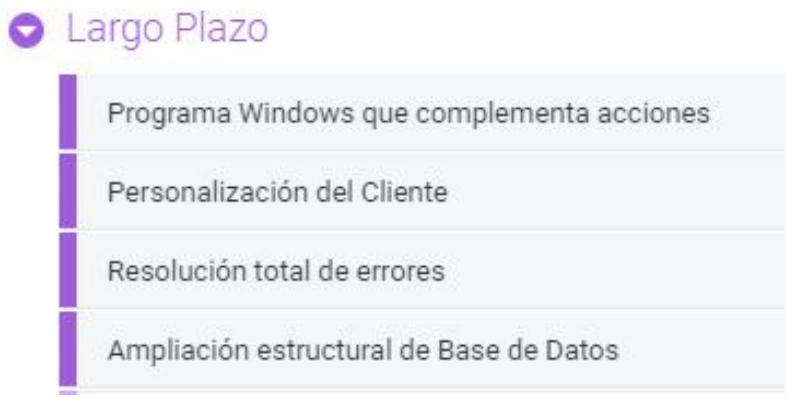
- •Dificultad con la organización de stock.
- •Roturas y pérdidas de material sin conocer el culpable.
- •Desconocimiento de la localización de el material.
- •Falta de organización en los eventos con el personal.
- •Errores en la sincronización de calendario.
- •Control de pagos, horas extras, coste unitario por empleado

Haciendo un uso correcto de la aplicación, se podrían solucionar todos los problemas dados por los emprendedores y trabajadores que realizaron la encuesta.

4.6.- PLANIFICACIÓN DEL TIEMPO



Para el día de presentación la aplicación estará en fase beta, es decir que cumple su función pero no promete su completa funcionalidad y rendimiento.



Una vez realizado unos cambios e implementar nuevas tecnologías, la app ya sería funcional en todos sus aspectos, dispuesta a cumplir su función sin riesgo de grandes errores causados por la propia configuración.

4.7.- ESTIMACIÓN DE COSTES

Las horas estimadas que se han invertido en el proyecto son un total de **130 horas**. Siendo un programador en su periodo de aprendizaje he decidido cobrar **10€/hora**. El resultado total sería de: **1.300€**

El vídeo promocional para dar a conocer la aplicación lo realizaría yo, con lo cual **evitaría el coste**.
Dicho vídeo será publicado en diferentes redes sociales (*Instagram* y *Facebook*, ya que son las más usadas por el público objetivo), la inversión en publicidad da un total de: **200€**

4.8.- ESTIMACIÓN DE INGRESOS

Como ya se ha descrito en puntos anteriores, la app funciona por **suscripción anual.**
Es decir que los ingresos serán variables dependiendo de la duración de los clientes con la suscripción activada.

Si multiplicamos el público objetivo por el precio de la aplicación, podremos estimar los ingresos:

89,35719862478 * 150 = 13.403€ (anual)

El siguiente paso es restarle a este último valor el beneficio neto:

13.403 - 200 = 13.203€ (anual)

Tras estos cálculos, se aproximan los ingresos a conseguir con la aplicación: **13.203€** (anual)

4.9.- CONCLUSIONES

Sacar a la venta esta aplicación **sí sería rentable**, ya que no conlleva gastos más que la publicidad.

El concepto es algo revolucionario dentro de la industria ya que no hay ninguna aplicación móvil que lleve un registro total de la empresa, siendo esta personalizada a la estructura de cada empresa.

Una vez acomodados a la facilidad y funcionalidad que ofrece el proyecto, los clientes querrán seguir con su suscripción anual, ya que es algo imprescindible dentro de la empresa.

5.- Arquitectura de la Aplicación

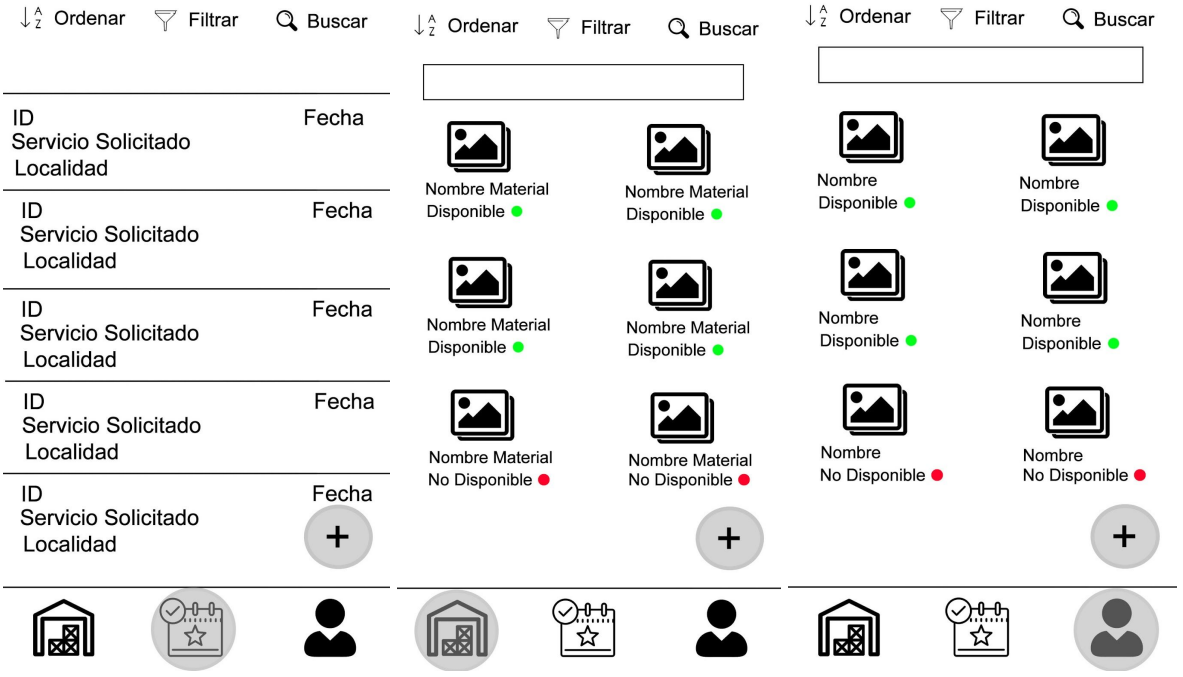
5.1.- RECURSOS Y TECNOLOGÍAS EMPLEADAS

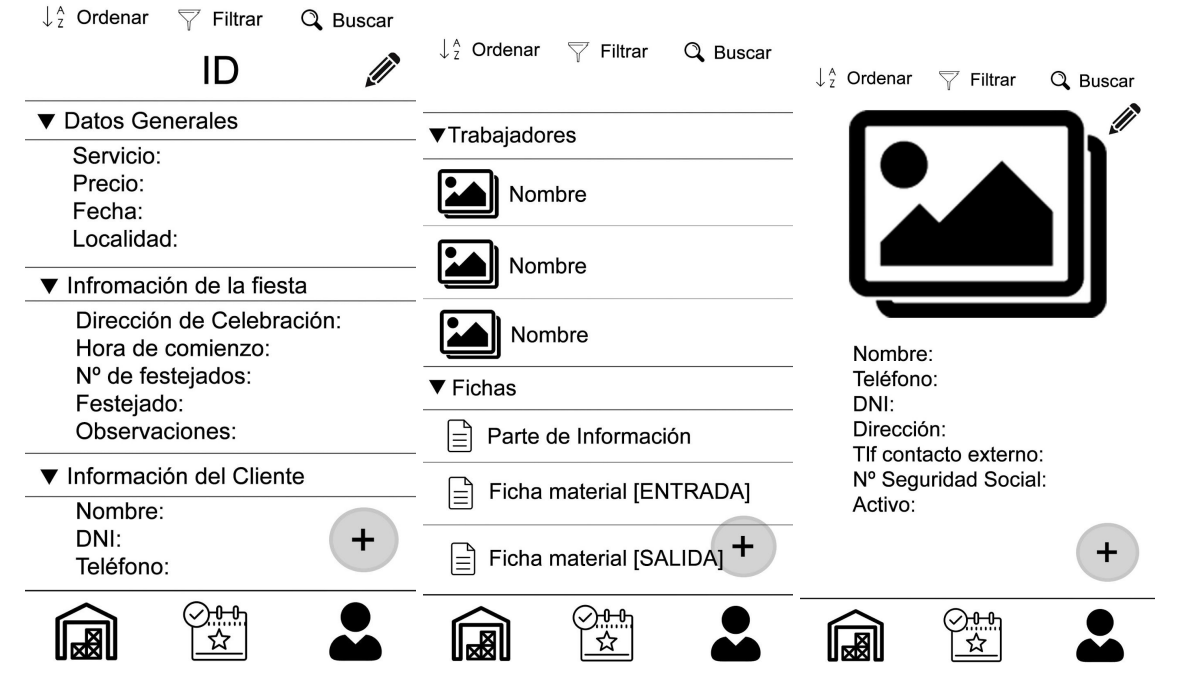
Para poder sacar el máximo partido a la aplicación se requiere de un ordenador situado en central logística de la empresa.

También hace uso de *FireBase* como herramienta de uso de **Base de Datos.**
(Esta podría ser sustituida por el servidor propio de la empresa en cuestión por preferencia del cliente)

5.2.- DISEÑO INICIAL

A continuación adjunto los diseños preliminares de diseño para la aplicación respecto a la **estructura**:





Para la **base de datos** hice una tabla escrita, sin relaciones ni herencias (características las cuales sí se desarrollaron a lo largo del trayecto)

Materiales

-ID

-Nombre

-Tipo

-Disponibilidad

-Imagen

Trabajadores

-DNI

-Nombre

-Teléfono

-Dirección

-Teléfono de Contacto externo

-Nº Seguridad Social

-Imagen

-Disponibilidad

Eventos

-ID

-Servicio Solicitado

-Observaciones

-Precio

-Nombre Cliente

-DNI Cliente

-Tlf Cliente

-Localidad

-Fecha y Hora

-Dirección Celebración

-Número de Invitados

-Nombre festejado

-Trabajadores

-Listado de Material (entrada y salida)

5.3.- DISEÑO FINAL

Diagrama de clases de la aplicación en estos momentos:

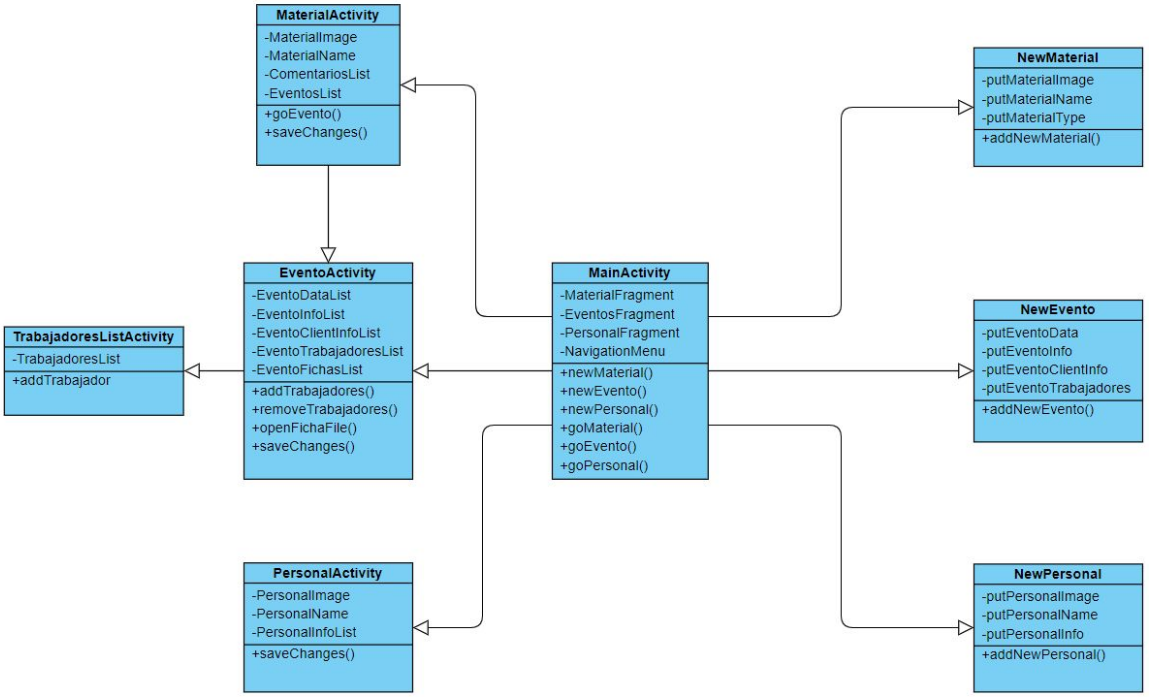
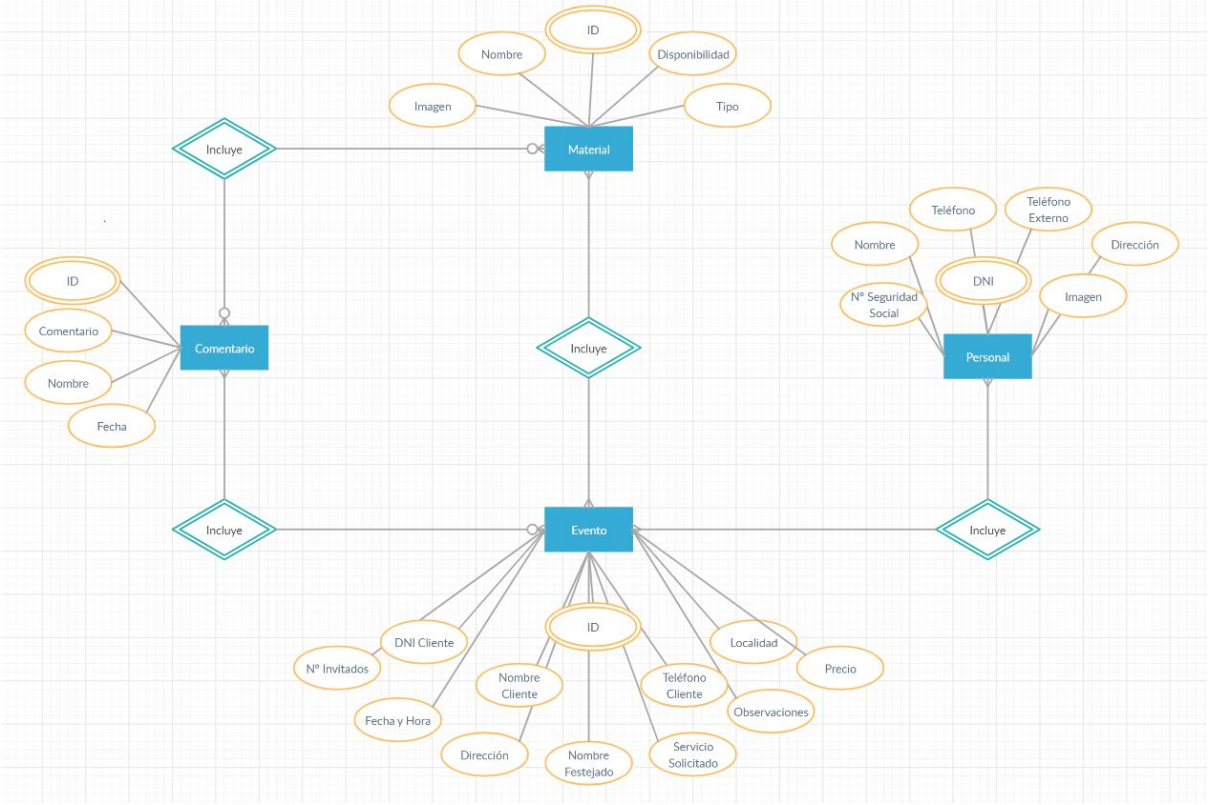


Diagrama Entidad-Relación de la aplicación en estos momentos:



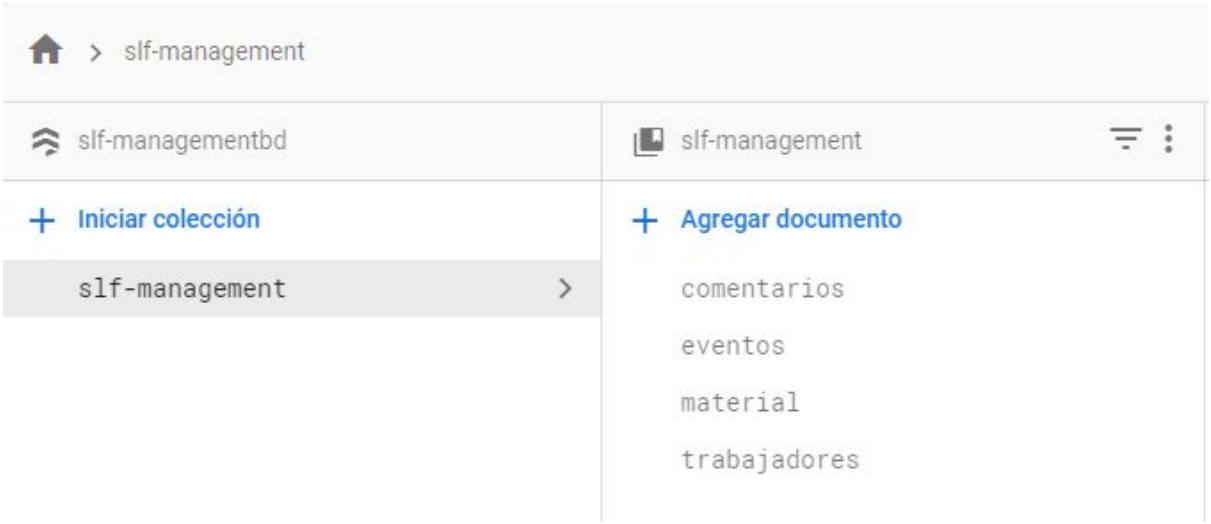
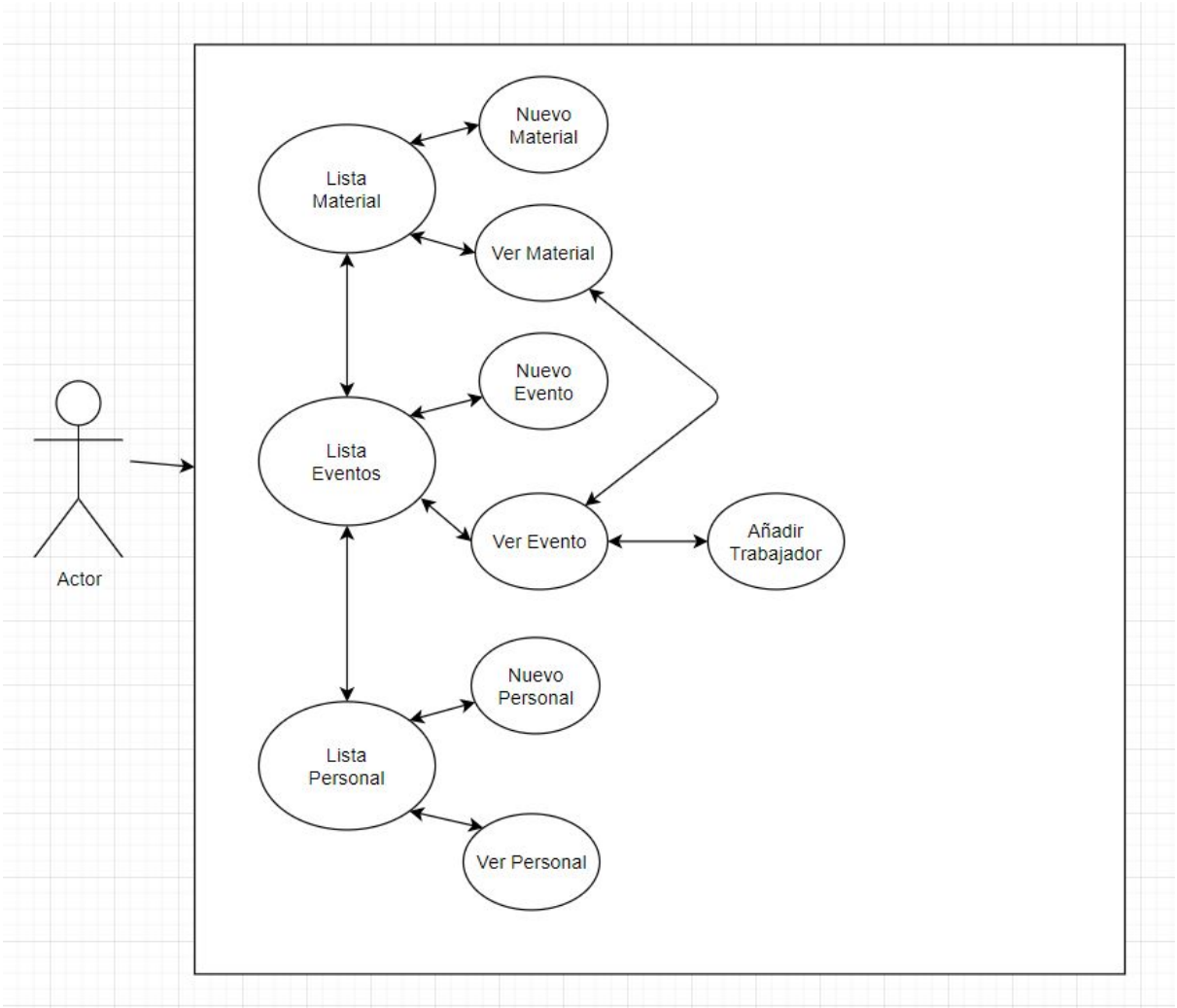


Diagrama de casos de uso de la aplicación en estos momentos:



6.- Futuras Mejoras y Conclusiones

Actualmente la versión de la aplicación se considera **beta**, ya que no garantiza una funcionalidad plena.

Los cambios y mejoras que se implementarán en un futuro son:

-Prevención de errores: Dar solución a todos los fallos/bugs posibles que pueda ocasionar el uso de la aplicación.

-Ampliación de Base de Datos: Añadir nuevos elementos para aprovechar las funcionalidades.

-Programa en Windows: Para poder sacar un mayor partido a la aplicación, requiere de un programa Windows para los trabajadores en la central logística de la empresa.

-Versión Usuario: *SLF-MANAGEMENT* está ambientada en el uso de los encargados/propietarios de la empresa, así que una versión para los trabajadores que permite enviar notificaciones con los datos necesarios sería una inversión.

-Mejoras de Diseño: El diseño de elementos, textos y listas. Opino que podría mejorarse en el futuro.

Mi intención con este proyecto es aprender nuevas tecnologías gracias a los conocimientos adquiridos en clase tras este grado. Esta idea me ha hecho invertir muchas horas en pequeños fragmentos de la aplicación dado que desconocía su funcionamiento.
Aunque haya costado mucho esfuerzo implementar pequeñas partes de la aplicación, me llevo lo aprendido y sé que en próximos proyectos ya conoceré más herramientas de uso.

Aun entregando la aplicación en esta versión temprana, mi intención es seguir mejorandola durante el año por diversos motivos.
Me gustaría sentirme orgulloso del trabajo realizado y poder mostrarlo a las empresas como carta de presentación a la hora de una entrevista de trabajo.
La aplicación está orientada en la empresa en la cual trabajo actualmente y me interesaría que llegasen a usarla, les sea útil y funcione como tengo esperado.
Dada la situación actual de el COVID-19, se nos ha ofrecido una segunda oportunidad de entrega en Diciembre (6 meses después de la primera entrega), y me gustaría exponer los cambios al centro.

7.- Pruebas y Resultados

A lo largo del desarrollo de la aplicación, me he topado con diversos contratiempos debidos a la necesidad de aprender en el transcurso de la programación.
A continuación voy a citar algunos de ellos:

Listas Expandibles

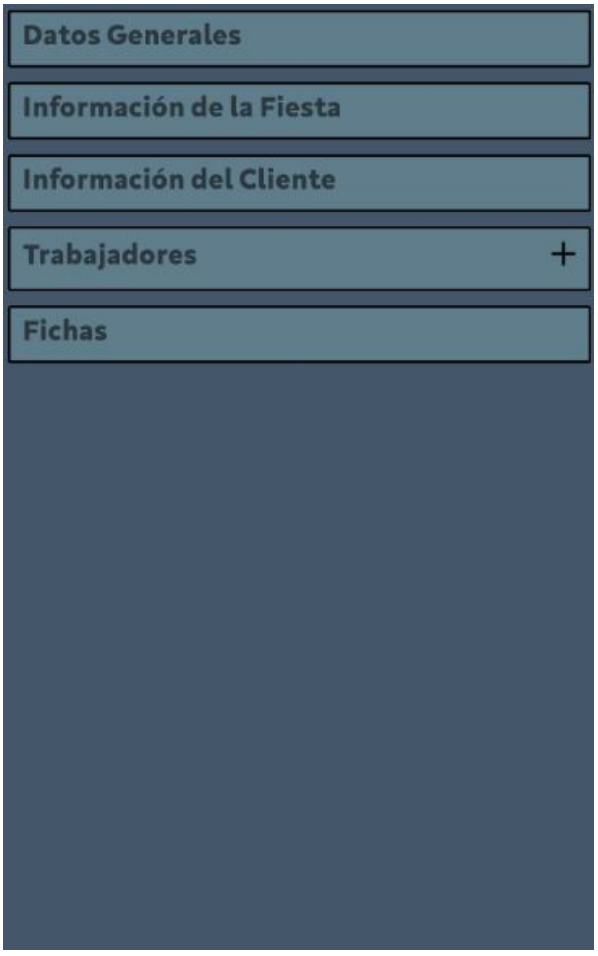
Según el funcionamiento de la aplicación, pensé que le podría venir muy bien listas expandibles para un mejor manejo y fluidez.

Al ser una tecnología que nunca había tocado, recurrí a investigar por foros y tutoriales, los cuales mencionan una implementación un tanto anticuada que estuve probando durante dos semanas.
La herramienta está muy desactualizada la cual me limitaba muchas de las utilidades que requería mi proyecto.

Trás varios intentos y horas de investigación di con una alternativa más compleja, funcional y muy útil hoy día.
Ya implementado se ve tal que así:

Expandida

Contraída



Este ha sido con diferencia el error que más tiempo me llevó resolver.

Imágenes

Para almacenar materiales y personal, una imagen del mismo es necesaria para distinguir fácilmente entre los demás ítems.

La complicación viene a la hora de tomar y guardar las imágenes en Base de Datos ya que uso un soporte que no conocía.

A día de hoy, tras investigar, buscar información y estudiar el error, conozco la solución para acabar con el problema, actualmente sigue permaneciendo en el proyecto, aunque fuera de plazo lo añadiré.

Como podemos comprobar, existe la posibilidad de imagen pero muestra una genérica por el momento:

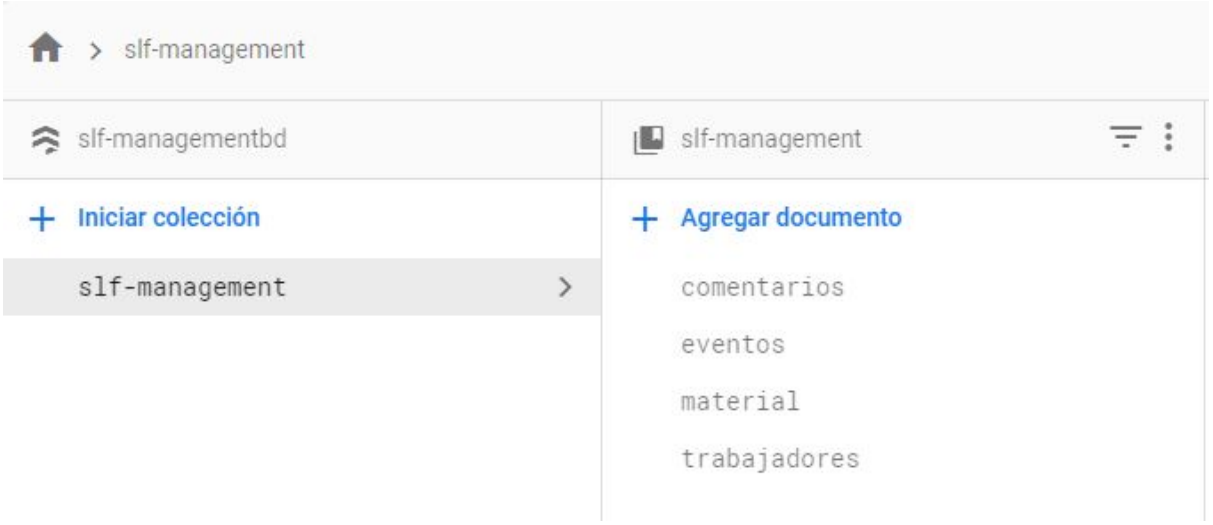
Base de Datos

SLF-MANAGEMENT no deja de ser un software ambientado en la gestión de una base de datos con una interfaz personalizada, es decir que la base de datos es una parte muy importante en el proyecto.

Una vez terminada la estructura general de la aplicación, decidí crear la base de datos según el diagrama creado previamente con las necesidades consultadas por el público objetivo.
En este aspecto no hubo ningún problema, en cuanto lo sincronicé con el código todo empezó a causar problemas, provocando empezar la distribución de base de datos desde el principio.

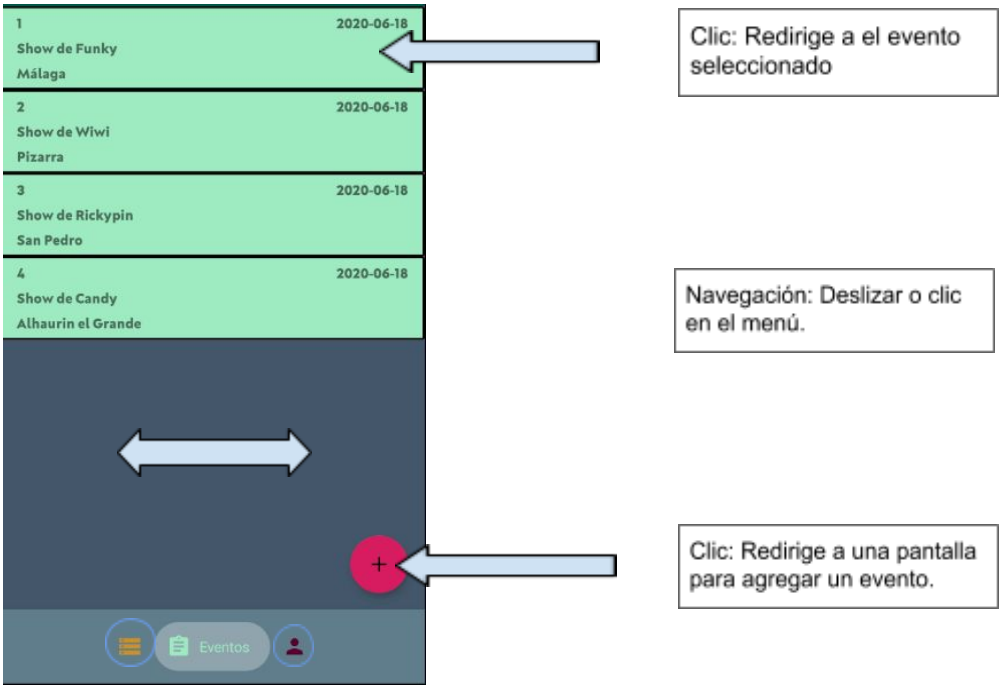
Ya descubiertos los fallos cometidos, implementé una parte en código.
Desgraciadamente coincidió con una fecha cercana a la entrega del proyecto, es decir que no están todos los datos sincronizados con base de datos, es cuestión de tiempo el poder integrar todas las funciones correspondientes al código.

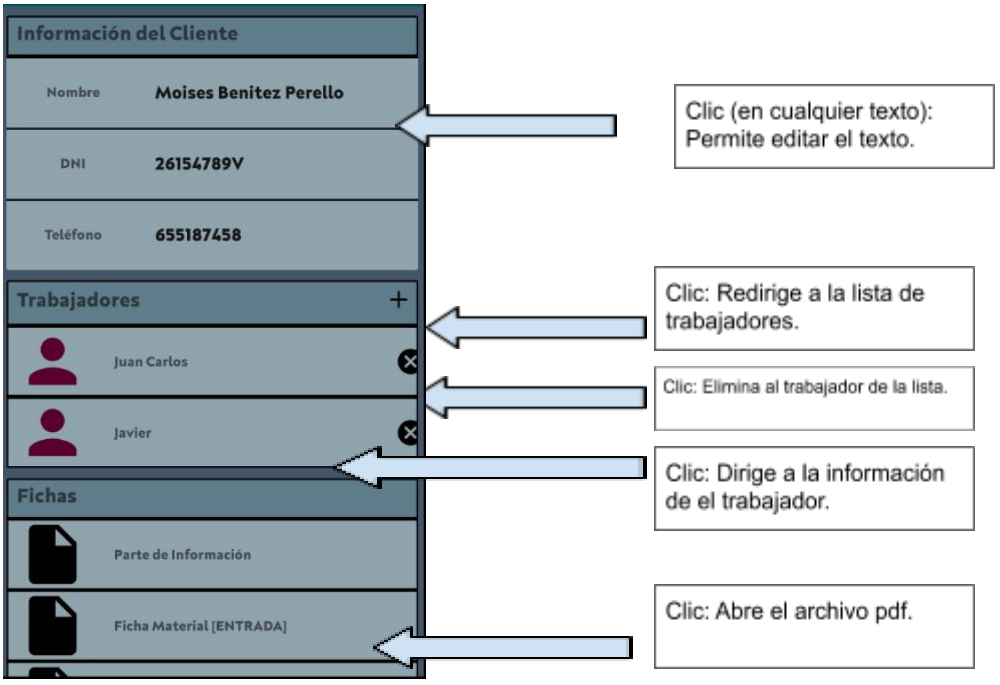
En la siguiente imagen podemos comprobar que realmente la Base de Datos está creada con los correspondientes objetos.



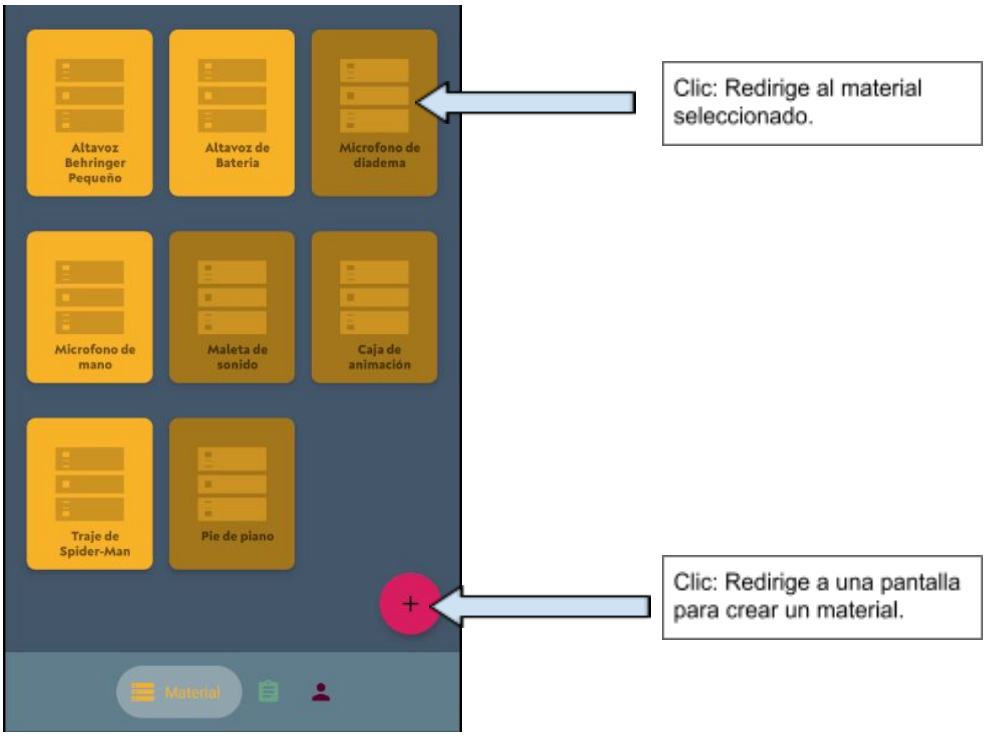
8.- Manual Usuario

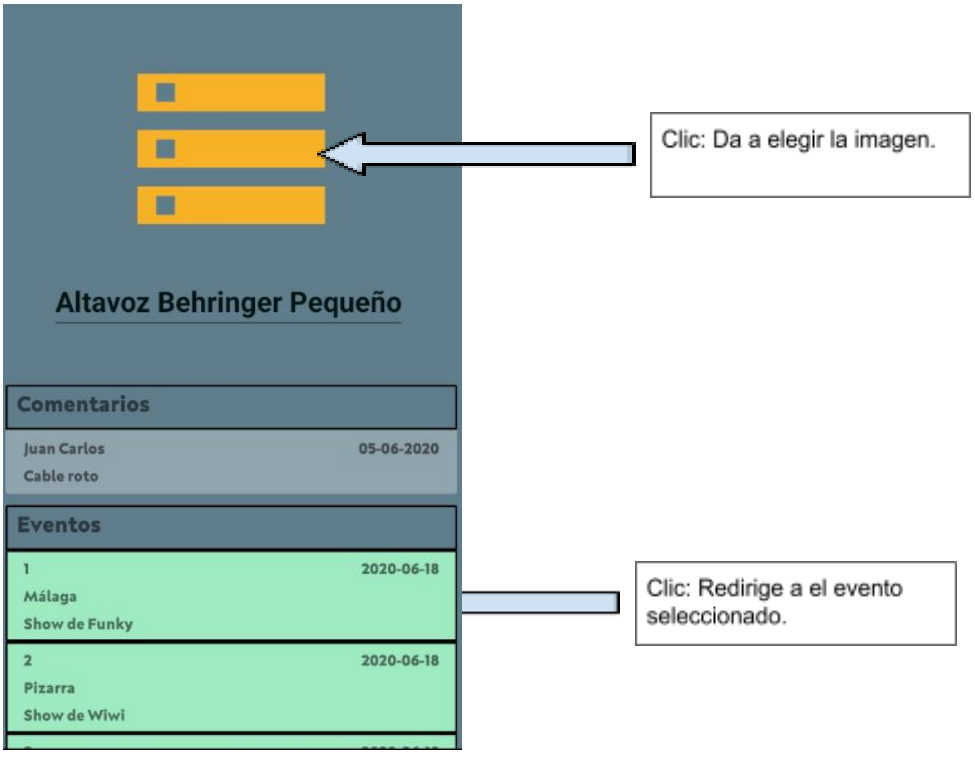
Al entrar en la aplicación, instantáneamente aparecerá el listado de *eventos* el cuál muestra los eventos con la información necesaria, en esta pantalla podremos hacer las siguientes acciones:

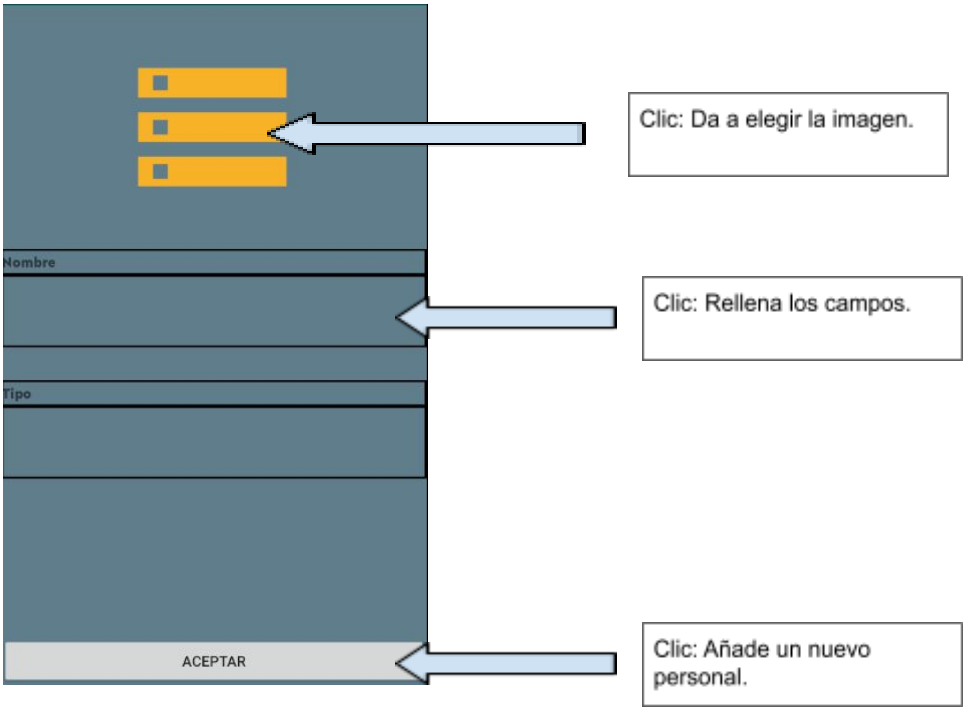


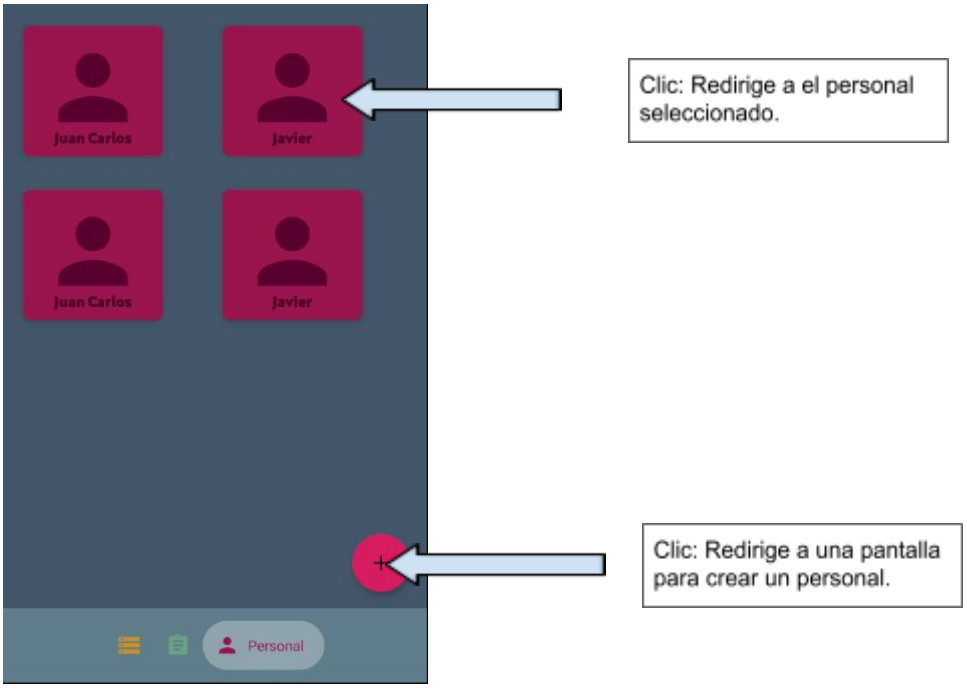


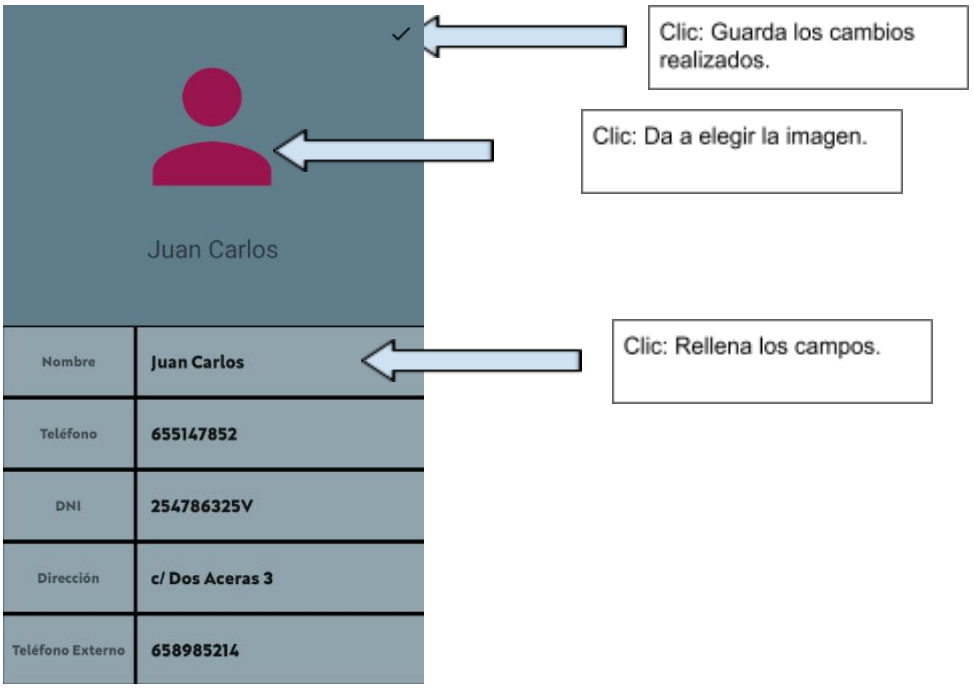

Juan Carlos
Javier
Ricardo
Paz
Lorena
Maria
Clic: Añade trabajador a la lista trabajadores.
Dirección de Celebración
Hora de comienzo
Nº de Festejados
Festejado
Observaciones
Nombre del Cliente
ACEPTAR
Clic: Rellena los campos.
Clic: Añade un nuevo personal.

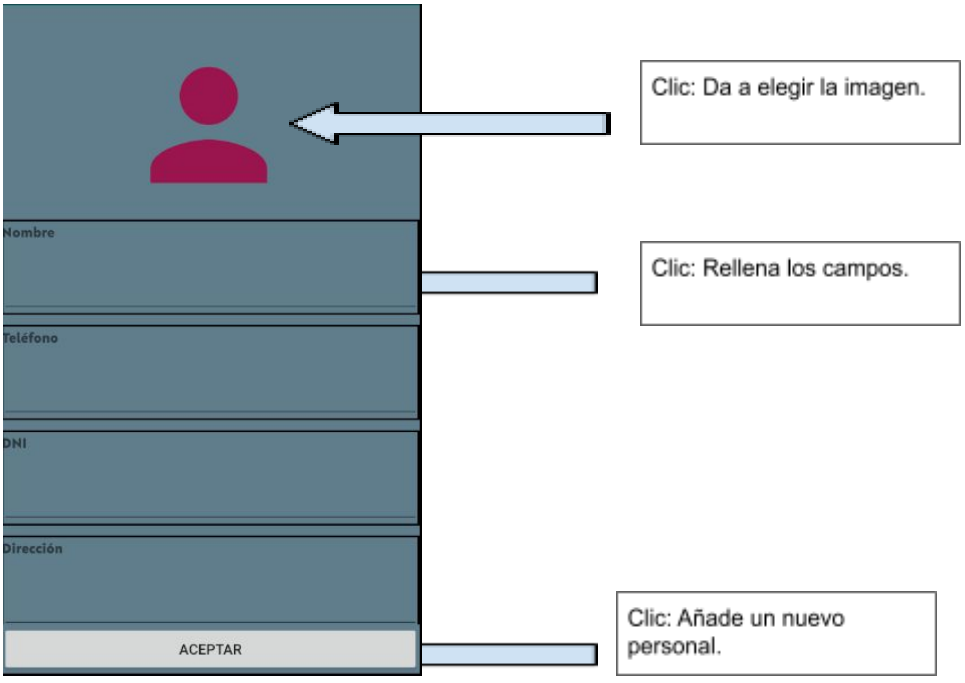

Altavoz Behringer Pequeño
Altavoz de Batería
Micrófono de diadema
Micrófono de mano
Maleta de sonido
Caja de animación
Traje de Spider-Man
Pie de piano
+
Material
Clic: Redirige al material seleccionado.
Clic: Redirige a una pantalla para crear un material.


Altavoz Behringer Pequeño
Comentarios
Juan Carlos 05-06-2020
Cable roto
Eventos
1 2020-06-18
Málaga
Show de Funky
2 2020-06-18
Pizarra
Show de Wiwi
Clic: Da a elegir la imagen.
Clic: Redirige a el evento seleccionado.


Clic: Da a elegir la imagen.
Nombre
Clic: Rellena los campos.
Tipo
ACEPTAR
Clic: Añade un nuevo personal.


Clic: Redirige a el personal seleccionado.
Juan Carlos
Javier
Juan Carlos
Javier
Clic: Redirige a una pantalla para crear un personal.
Personal


Clic: Guarda los cambios realizados.
Clic: Da a elegir la imagen.
Juan Carlos
Clic: Rellena los campos.
Nombre
Juan Carlos
Teléfono
655147852
DNI
254786325V
Dirección
c/ Dos Aceras 3
Teléfono Externo
658985214


Clic: Da a elegir la imagen.
Clic: Rellena los campos.
Nombre
Teléfono
DNI
Dirección
ACEPTAR
Clic: Añade un nuevo personal.

9.- Bibliografía

Cómo hacer un estudio de viabilidad. *uncomo.com* [online],
2020. *Play.google.com* [online],
Google Forms: Sign-in, 2020. *Docs.google.com* [online],
Google Ads - Crea anuncios online fácilmente y consigue más clientes, 2020. *Ads.google.com* [online],
Diagram Maker | Software de Diagramas en Línea, 2020. *Creately.com* [online],
Visual Paradigm Online - Suite of Powerful Tools, 2020. *Online.visual-paradigm.com* [online],
Flowchart Maker & Online Diagram Software, 2020. *App.diagrams.net* [online],
Cloud Firestore | Firebase, 2020. *Firebase* [online],
Gestión de proyectos | monday.com, 2020. *monday.com* [online],
RÁEZ, MÓNICA, 2020, ¿Qué es el embudo de Marketing? - MARATUM. *MARATUM* [online]. 2020. [Accessed 18 June 2020]. Available from: https://maratum.com/que-es-el-embudo-de-marketing/

10 - Anexo I.- Principales clases del código

MainActivity

```
class MainActivity : AppCompatActivity() {

    private val viewPagerFragment by lazy { findViewById<ViewPager>(R.id.view_pager)}
    private val navigation by lazy { findViewById<BubbleNavigationLinearView>
(R.id.bottom_navigation_view_linear)}
    private var positionFragment = 1

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        /**
         * DECLARACION DE VARIABLES
         */
        val fragmentAdapter = FragmentAdapter(supportFragmentManager)
        val materialTab = MaterialFragment()
        val eventosTab = EventosFragment()

        /**
         * AÑADIENDO FRAGMENTS
         */
        val personalTab = PersonalFragment()
        fragmentAdapter.addFragments(materialTab)
        fragmentAdapter.addFragments(eventosTab)
        fragmentAdapter.addFragments(personalTab)

        viewPagerFragment.adapter = fragmentAdapter
        viewPagerFragment.addOnPageChangeListener(object :
ViewPager.OnPageChangeListener {
```

```
            override fun onPageScrollStateChanged(state: Int) {
            }

            override fun onPageScrolled(position: Int, positionOffset: Float, positionOffsetPixels:
Int) {
            }

            override fun onPageSelected(position: Int) {
                navigation.setCurrentActiveItem(position)
                positionFragment = position
            }
        })
        viewPagerFragment.setCurrentItem(1)

        navigation.setNavigationChangeListener(object : BubbleNavigationChangeListener {
            override fun onNavigationChanged(view: View?, position: Int) {
                viewPagerFragment.setCurrentItem(position, true)
                positionFragment = position
            }
        })

        /**
         * BOTON FLOTANTE
         */
        /**
         * DEPENDIENDO DE EN QUE FRAGMENT SE SITUE, ENVIARÁ A UN ACTIVITY
DIFERENTE
         */
        val fab: View = findViewById(R.id.fabAñadir)
        fab.setOnClickListener { view ->
            when (positionFragment) {
                0 -> {
                    val intent = Intent(this, NuevoMaterialActivity::class.java)
                    startActivity(intent)
                }
                1 -> {
                    val intent = Intent(this, NuevoEventoActivity::class.java)
                    startActivity(intent)
                }
                2 -> {
                    val intent = Intent(this, NuevoPersonalActivity::class.java)
                    startActivity(intent)
                }
            }
        }
```

```
        when(positionFragment){
            0 -> {

            }
        }
    }

}
```

Activitys

class EventoActivity : AppCompatActivity() {

```
    private var mostrarDatosGenerales = true
    private var mostrarInformacionDeLaFiesta = true
    private var mostrarInformacionDelCliente = true
    private var mostrarTrabajadores = true
    private var mostrarFichas = true

    @RequiresApi(Build.VERSION_CODES.O)
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_evento)

        /**
         * LISTA DATOS GENERALES
         */
        /**
         * CREACIÓN DE LISTA E ITEMS (CAMBIAR POR BASE DE DATOS)
         */
        val listaDatosGenereales: ArrayList<ListaItem> = arrayListOf()
        val item = ListaItem("Servicio", "Show de Funky")
        val item2 = ListaItem("Precio", "150€")
        val item3 = ListaItem("Fecha", LocalDate.now().toString())
        val item4 = ListaItem("Localidad", "Málaga")

        /**
         * INSERTANDO ITEMS EN LISTA
         */
        listaDatosGenereales.add(item)
        listaDatosGenereales.add(item2)
        listaDatosGenereales.add(item3)
        listaDatosGenereales.add(item4)

        /**
```

```
        * ESTABLECIENDO TITULO
        */
        val textoDatosGenerales =
listaRecyclerDatosGenerales.findViewById<TextView>(R.id.tituloMenu)
        textoDatosGenerales.text = "Datos Generales"

        /**
        * ESTABLECIENDO TITULO
        */
        val adapterDatosGenerales =
SectionRecyclerViewListaItemAdapter(listaDatosGenereales)
        val recyclerMenuDatosGenerales =
listaRecyclerDatosGenerales.findViewById<RecyclerView>(R.id.recyclerMenu)
        val layoutManagerDatosGenerales = LinearLayoutManager(this@EventoActivity,
LinearLayoutManager.VERTICAL, false)
        recyclerMenuDatosGenerales.layoutManager = layoutManagerDatosGenerales
        recyclerMenuDatosGenerales.adapter = adapterDatosGenerales
        recyclerMenuDatosGenerales.visibility = View.VISIBLE

        /**
        * FUNCIÓN ONCLIC PARA OCULTAR/MOSTRAR LISTA
        */
        listaRecyclerDatosGenerales.setOnClickListener(object : View.OnClickListener {
            override fun onClick(v: View?) {
                if (!mostrarDatosGenerales) {
                    recyclerMenuDatosGenerales.visibility = View.VISIBLE
                    mostrarDatosGenerales = true
                } else {
                    recyclerMenuDatosGenerales.visibility = View.GONE
                    mostrarDatosGenerales = false
                }
            }
        })

        /**
        * LISTA INFORMACION DE LA FIESTA
        */
        /**
        * CREACIÓN DE LISTA E ITEMS (CAMBIAR POR BASE DE DATOS)
        */
        val listaInformacionDeLaFiesta: ArrayList<ListaItem> = arrayListOf()
        val informacion1 = ListaItem("Dirección de Celebración", "c/Gondola 2")
        val informacion2 = ListaItem("Hora de comienzo", "17:30")
        val informacion3 = ListaItem("Nº de Festejados", "20")
        val informacion4 = ListaItem("Festejado", "Jaimito")
        val informacion5 = ListaItem("Observaciones", "Es alergico al azúcar")
```

```
        /**
         * INSERTANDO ITEMS EN LISTA
         */
        listaInformacionDeLaFiesta.add(informacion1)
        listaInformacionDeLaFiesta.add(informacion2)
        listaInformacionDeLaFiesta.add(informacion3)
        listaInformacionDeLaFiesta.add(informacion4)
        listaInformacionDeLaFiesta.add(informacion5)

        /**
         * ESTABLECIENDO TITULO
         */
        val textoInformacionDeLaFiesta =
listaRecyclerInformacionDeLaFiesta.findViewById<TextView>(R.id.tituloMenu)
        textoInformacionDeLaFiesta.text = "Información de la Fiesta"

        /**
         * SINCRONIZACIÓN CON LOS ADAPTERS
         */
        val adapterInformacionDeLaFiesta =
SectionRecyclerViewListaItemAdapter(listaInformacionDeLaFiesta)
        val recyclerMenuInformacionDeLaFiesta =
listaRecyclerInformacionDeLaFiesta.findViewById<RecyclerView>(R.id.recyclerMenu)
        val layoutManagerInformacionDeLaFiesta =
LinearLayoutManager(this@EventoActivity, LinearLayoutManager.VERTICAL, false)
        recyclerMenuInformacionDeLaFiesta.layoutManager =
layoutManagerInformacionDeLaFiesta
        recyclerMenuInformacionDeLaFiesta.adapter = adapterInformacionDeLaFiesta
        recyclerMenuInformacionDeLaFiesta.visibility = View.VISIBLE

        /**
         * FUNCIÓN ONCLIC PARA OCULTAR/MOSTRAR LISTA
         */
        listaRecyclerInformacionDeLaFiesta.setOnClickListener(object : View.OnClickListener {
            override fun onClick(v: View?) {
                if (!mostrarInformacionDeLaFiesta) {
                    recyclerMenuInformacionDeLaFiesta.visibility = View.VISIBLE
                    mostrarInformacionDeLaFiesta = true
                } else {
                    recyclerMenuInformacionDeLaFiesta.visibility = View.GONE
                    mostrarInformacionDeLaFiesta = false
                }
            }
        })
```

```
        /**
         * INFORMACIÓN DEL CLIENTE
         */
        /**
         * CREACIÓN DE LISTA E ITEMS (CAMBIAR POR BASE DE DATOS)
         */
        val listaInformacionDelCliente: ArrayList<ListaItem> = arrayListOf()
        val informacionCliente1 = ListaItem("Nombre", "Moises Benitez Perello")
        val informacionCliente2 = ListaItem("DNI", "26154789V")
        val informacionCliente3 = ListaItem("Teléfono", "655187458")

        /**
         * INSERTANDO ITEMS EN LISTA
         */
        listaInformacionDelCliente.add(informacionCliente1)
        listaInformacionDelCliente.add(informacionCliente2)
        listaInformacionDelCliente.add(informacionCliente3)

        /**
         * ESTABLECIENDO TITULO
         */
        val textoInformacionDelCliente =
listaRecyclerInformacionDelCliente.findViewById<TextView>(R.id.tituloMenu)
        textoInformacionDelCliente.text = "Información del Cliente"

        /**
         * SINCRONIZACIÓN CON LOS ADAPTERS
         */
        val adapterInformacionDelCliente =
SectionRecyclerViewListaItemAdapter(listaInformacionDelCliente)
        val recyclerMenuInformacionDelCliente =
listaRecyclerInformacionDelCliente.findViewById<RecyclerView>(R.id.recyclerMenu)
        val layoutManagerInformacionDelCliente = LinearLayoutManager(this@EventoActivity,
LinearLayoutManager.VERTICAL, false)
        recyclerMenuInformacionDelCliente.layoutManager =
layoutManagerInformacionDelCliente
        recyclerMenuInformacionDelCliente.adapter = adapterInformacionDelCliente
        recyclerMenuInformacionDelCliente.visibility = View.VISIBLE

        /**
         * FUNCIÓN ONCLIC PARA OCULTAR/MOSTRAR LISTA
         */
        listaRecyclerInformacionDelCliente.setOnClickListener(object : View.OnClickListener {
            override fun onClick(v: View?) {
                if (!mostrarInformacionDelCliente) {
                    recyclerMenuInformacionDelCliente.visibility = View.VISIBLE
```

```
                mostrarInformacionDelCliente = true
            } else {
                recyclerMenuInformacionDelCliente.visibility = View.GONE
                mostrarInformacionDelCliente = false
            }
        }
    })
    /**
     * LISTA TRABAJADORES
     */
    /**
     * CREACIÓN DE LISTA E ITEMS (CAMBIAR POR BASE DE DATOS)
     */
    val listaTrabajadores: ArrayList<TrabajadorItem> = arrayListOf()
    val trabajador1 = TrabajadorItem(R.drawable.personal_icon, "Juan Carlos")
    val trabajador2 = TrabajadorItem(R.drawable.personal_icon, "Javier")

    /**
     * INSERTANDO ITEMS EN LISTA
     */
    listaTrabajadores.add(trabajador1)
    listaTrabajadores.add(trabajador2)

    /**
     * ESTABLECIENDO TITULO
     */
    val textoTrabajadores =
listaRecyclerTrabajadores.findViewById<TextView>(R.id.tituloHeaderTrabajadores)
    textoTrabajadores.text = "Trabajadores"

    /**
     * SINCRONIZACIÓN CON LOS ADAPTERS
     */
    val adapterTrabajadores = SectionRecyclerViewTrabajadorAdapter(listaTrabajadores)
    val recyclerMenuTrabajadores =
listaRecyclerTrabajadores.findViewById<RecyclerView>(R.id.recyclerHeaderTrabajadores)
    val layoutManagerTrabajadores = LinearLayoutManager(this@EventoActivity,
LinearLayoutManager.VERTICAL, false)
    recyclerMenuTrabajadores.layoutManager = layoutManagerTrabajadores
    recyclerMenuTrabajadores.adapter = adapterTrabajadores
    recyclerMenuTrabajadores.visibility = View.VISIBLE

    /**
     * FUNCIÓN ONCLIC PARA OCULTAR/MOSTRAR LISTA
     */
    listaRecyclerTrabajadores.setOnClickListener(object : View.OnClickListener {
```

```
            override fun onClick(v: View?) {
                if (!mostrarTrabajadores) {
                    recyclerMenuTrabajadores.visibility = View.VISIBLE
                    mostrarTrabajadores = true
                } else {
                    recyclerMenuTrabajadores.visibility = View.GONE
                    mostrarTrabajadores = false
                }
            }
        })
        /**
         * ADAPTER TRABAJADORES
         */
        adapterTrabajadores.setSectionRecyclerViewListener(object :
SectionRecyclerViewTrabajadorAdapter.SectionRecyclerViewListener {
            override fun onItemClick(itemPosition: Int) {
            }
            /**
             * FUNCION ONCLICK PARA ELIMINAR ITEMS DE LA LISTA
             */
            override fun onDeleteClick(itemPosition: Int) {
                listaTrabajadores.removeAt(itemPosition)
                adapterTrabajadores.setListaTrabajadores(listaTrabajadores)
            }
        })

        /**
         * LISTA FICHAS
         */
        /**
         * CREACIÓN DE LISTA E ITEMS (CAMBIAR POR BASE DE DATOS)
         */
        val listaFichas: ArrayList<FichaItem> = arrayListOf()
        val ficha1 = FichaItem(R.drawable.ic_insert_drive_file_black_24dp, "Parte de
Información")
        val ficha2 = FichaItem(R.drawable.ic_insert_drive_file_black_24dp, "Ficha Material
[ENTRADA]")
        val ficha3 = FichaItem(R.drawable.ic_insert_drive_file_black_24dp, "Ficha Material
[SALIDA]")

        /**
         * INSERTANDO ITEMS EN LISTA
         */
        listaFichas.add(ficha1)
        listaFichas.add(ficha2)
        listaFichas.add(ficha3)
```

```
        /**
         * ESTABLECIENDO TITULO
         */
        val textoFichas = listaRecyclerFichas.findViewById<TextView>(R.id.tituloMenu)
        textoFichas.text = "Fichas"

        /**
         * SINCRONIZACIÓN CON LOS ADAPTERS
         */
        val adapterFichas = SectionRecyclerViewFichaAdapter(listaFichas)
        val recyclerMenuFichas =
listaRecyclerFichas.findViewById<RecyclerView>(R.id.recyclerMenu)
        val layoutManagerFichas = LinearLayoutManager(this@EventoActivity,
LinearLayoutManager.VERTICAL, false)
        recyclerMenuFichas.layoutManager = layoutManagerFichas
        recyclerMenuFichas.adapter = adapterFichas
        recyclerMenuFichas.visibility = View.VISIBLE

        /**
         * FUNCIÓN ONCLIC PARA OCULTAR/MOSTRAR LISTA
         */
        listaRecyclerFichas.setOnClickListener(object : View.OnClickListener {
            override fun onClick(v: View?) {
                if (!mostrarFichas) {
                    recyclerMenuFichas.visibility = View.VISIBLE
                    mostrarFichas = true
                } else {
                    recyclerMenuFichas.visibility = View.GONE
                    mostrarFichas = false
                }
            }
        })

        /**
         * FUNCION ONCLIC PARA AÑADIR UN ITEM A LISTA TRABAJADOR
         */
        val buttonNuevoTrabajador = findViewById<ImageView>(R.id.buttonNuevoTrabajador)

        buttonNuevoTrabajador.setOnClickListener {
            val intent = Intent(this, ListaPersonalActivity::class.java)
            startActivity(intent)
        }
    }
}
```

```
class ListaPersonalActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_lista_personal)

        /**
         * CREACIÓN DE LISTA E ITEMS (CAMBIAR POR BASE DE DATOS)
         */
        var listaTrabajadores:ArrayList<FichaItem> = arrayListOf()

        val trabajador1 = FichaItem(R.drawable.personal_icon, "Juan Carlos")
        val trabajador2 = FichaItem(R.drawable.personal_icon, "Javier")
        val trabajador3 = FichaItem(R.drawable.personal_icon, "Ricardo")
        val trabajador4 = FichaItem(R.drawable.personal_icon, "Paz")
        val trabajador5 = FichaItem(R.drawable.personal_icon, "Lorena")
        val trabajador6 = FichaItem(R.drawable.personal_icon, "Maria")

        /**
         * INSERTANDO ITEMS EN LISTA
         */
        listaTrabajadores.add(trabajador1)
        listaTrabajadores.add(trabajador2)
        listaTrabajadores.add(trabajador3)
        listaTrabajadores.add(trabajador4)
        listaTrabajadores.add(trabajador5)
        listaTrabajadores.add(trabajador6)

        /**
         * SINCRONIZACIÓN CON LOS ADAPTERS
         */
        val adapterTrabajador = TrabajadoresAdapter(listaTrabajadores)
        val recyclerTrabajador =
recyclerListaTrabajadores.findViewById<RecyclerView>(R.id.recyclerListaTrabajadores)
        val layoutManagerTrabajador = LinearLayoutManager(this,
LinearLayoutManager.VERTICAL, false)
        recyclerTrabajador.layoutManager = layoutManagerTrabajador
        recyclerTrabajador.adapter = adapterTrabajador

        /**
         * FUNCION ONCLIC QUE AÑADE ITEM EN LISTA TRABAJADOR Y DEVUELVE A
LA PANTALLA ANTERIOR
         */
        adapterTrabajador.setSectionRecyclerViewListener(object
:TrabajadoresAdapter.SectionRecyclerViewListener{
            override fun onItemClick(itemPosition: Int) {
```

```
            //INSERT EN BASE DE DATOS
            adapterTrabajador.setlistaTrabajador(listaTrabajadores)
            Toast.makeText(this@ListaPersonalActivity, "Trabajador añadido",
Toast.LENGTH_SHORT).show()
            finish()
        }
    })
  }
}
```

class MaterialActivity : AppCompatActivity() {

```
  private val nombreMaterialActivity by lazy {
findViewById<TextView>(R.id.nombreMaterialActivity) }
  private val imagenMaterialActivity by lazy {
findViewById<ImageView>(R.id.imagenMaterialActivity) }
  private var mostrarComentarios = true
  private var mostrarEventos = true
  private val db by lazy { FirebaseFirestore.getInstance()}
  private var listaComentarios = ListaComentarios(arrayListOf())
  private var adapterComentarios =
SectionRecyclerViewComentariosAdapter(listaComentarios.listaComentario)

  @RequiresApi(Build.VERSION_CODES.O)
  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_material)
    intent.extras?.let {
      val nombreMaterial = it.getString("nombreMaterial")
      val imagenMaterial = it.getInt("imagenMaterial")
      nombreMaterialActivity.text = nombreMaterial
      imagenMaterialActivity.setImageResource(imagenMaterial)
    }

    title = "Material"

    /**
     * LISTA COMENTARIOS
     */
    /**
     * RECOGE DE BASE DE DATOS LOS COMENTARIOS
     */

db.collection("slf-management").document("comentarios").get().addOnCompleteListener{
task ->
      if (task.isSuccessful){
```

```
                listaComentarios = task.result!!.toObject(ListaComentarios::class.java)!!
                adapterComentarios.setListaComentario(listaComentarios.listaComentario)
            }else{
                Toast.makeText(this@MaterialActivity, task.exception.toString(),
Toast.LENGTH_LONG).show()

            }
        }

        /**
         * SINCRONIZACIÓN CON LOS ADAPTERS
         */
        val recyclerMenuComentarios =
listaComentario.findViewById<RecyclerView>(R.id.recyclerMenu)
        val layoutManagerComentarios = LinearLayoutManager(this@MaterialActivity,
LinearLayoutManager.VERTICAL, false)
        recyclerMenuComentarios.layoutManager = layoutManagerComentarios
        recyclerMenuComentarios.adapter = adapterComentarios
        recyclerMenuComentarios.visibility = View.VISIBLE
        listaComentario.setOnClickListener(object : View.OnClickListener{

            /**
             * FUNCIÓN ONCLIC PARA OCULTAR/MOSTRAR LISTA
             */
            override fun onClick(v: View?) {

                if (!mostrarComentarios) {
                    recyclerMenuComentarios.visibility = View.VISIBLE
                    mostrarComentarios = true
                }
                else {
                    recyclerMenuComentarios.visibility = View.GONE
                    mostrarComentarios = false
                }
            }
        })

        /**
         * ESTABLECIENDO TITULO
         */
        val textoComentario = listaComentario.findViewById<TextView>(R.id.tituloMenu)
        textoComentario.text = "Comentarios"

        /**
         * LISTA EVENTOS
         */
```

```
        /**
         * CREACIÓN DE LISTA E ITEMS (CAMBIAR POR BASE DE DATOS)
         */
        val listaEventos : ArrayList<Evento> = arrayListOf()
        val evento1 = Evento(1, "Show de Funky", "Málaga", LocalDate.now())
        val evento2 = Evento(2, "Show de Wiwi", "Pizarra", LocalDate.now())
        val evento3 = Evento(3, "Show de Rickypin", "San Pedro", LocalDate.now())
        val evento4 = Evento(4, "Show de Candy", "Alhaurin el Grande", LocalDate.now())

        /**
         * INSERTANDO ITEMS EN LISTA
         */
        listaEventos.add(evento1)
        listaEventos.add(evento2)
        listaEventos.add(evento3)
        listaEventos.add(evento4)

        /**
         * SINCRONIZACIÓN CON LOS ADAPTERS
         */
        val adapter = SectionRecyclerViewEventosAdapter(listaEventos)
        val recyclerMenu = listaEvento.findViewById<RecyclerView>(R.id.recyclerMenu)
        val layoutManager = LinearLayoutManager(this@MaterialActivity,
LinearLayoutManager.VERTICAL, false)
        recyclerMenu.layoutManager = layoutManager
        recyclerMenu.adapter = adapter
        recyclerMenu.visibility = View.VISIBLE

        /**
         * FUNCIÓN ONCLIC PARA OCULTAR/MOSTRAR LISTA
         */
        listaEvento.setOnClickListener(object : View.OnClickListener{
            override fun onClick(v: View?) {

                if (!mostrarEventos) {
                    recyclerMenu.visibility = View.VISIBLE
                    mostrarEventos = true
                }
                else {
                    recyclerMenu.visibility = View.GONE
                    mostrarEventos = false
                }
            }
        })
        /**
         * ESTABLECIENDO TITULO
```

```
        */
        val textoEvento = listaEvento.findViewById<TextView>(R.id.tituloMenu)
        textoEvento.text = "Eventos"

        /**
         * FUNCIÓN ONCLICK QUE REDIRIGE AL ACTIVITYEVENTO
         */
        adapter.setSectionRecyclerViewListener(object
:SectionRecyclerViewEventosAdapter.SectionRecyclerViewListener{
            override fun onItemClick(itemPosition: Int) {
                val intent = Intent(this@MaterialActivity, EventoActivity::class.java)
                startActivity(intent)
            }
        })

        /**
         * IMAGEN
         */
        val imagen = findViewById<ImageView>(R.id.imagenMaterialActivity)
        imagen.setOnClickListener{
            Toast.makeText(this, "Hacer foto", Toast.LENGTH_SHORT).show()
        }
    }

}
```

```
class NuevoEventoActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_nuevo_evento)
        /**
         * CREACIÓN DE LISTA E ITEMS (CAMBIAR POR BASE DE DATOS)
         */
        var listaElementos: ArrayList<NuevoElementoItem> = arrayListOf()

        val elemento1 = NuevoElementoItem("Dirección de Celebración", "")
        val elemento2 = NuevoElementoItem("Hora de comienzo", "")
        val elemento3 = NuevoElementoItem("Nº de Festejados", "")
        val elemento4 = NuevoElementoItem("Festejado", "")
        val elemento5 = NuevoElementoItem("Observaciones", "")
        val elemento6 = NuevoElementoItem("Nombre del Cliente", "")
        val elemento7 = NuevoElementoItem("DNI del Cliente", "")
        val elemento8 = NuevoElementoItem("Teléfono del Cliente", "")

        /**
         * INSERTANDO ITEMS EN LISTA
         */
        listaElementos.add(elemento1)
        listaElementos.add(elemento2)
        listaElementos.add(elemento3)
        listaElementos.add(elemento4)
        listaElementos.add(elemento5)
        listaElementos.add(elemento6)
        listaElementos.add(elemento7)
        listaElementos.add(elemento8)

        /**
         * SINCRONIZACIÓN CON LOS ADAPTERS
         */
        val adapter = NuevoElementoItemAdapter(listaElementos)
        val recycler =
recyclerNuevoEvento.findViewById<androidx.recyclerview.widget.RecyclerView>(com.exam
ple.slf_management.R.id.recyclerNuevoEvento)
        val layoutManager= LinearLayoutManager(this@NuevoEventoActivity,
LinearLayoutManager.VERTICAL, false)
        recycler.layoutManager = layoutManager
        recycler.adapter = adapter

        /**
         * BOTON
         */
```

```
        val buttonAceptar = findViewById<Button>(R.id.buttonAceptarNuevoEvento)
        buttonAceptar.setOnClickListener{
            Toast.makeText(this, "Evento añadido", Toast.LENGTH_SHORT).show()
            finish()
        }
    }
}



class NuevoMaterialActivity : AppCompatActivity() {
    private val db by lazy { FirebaseFirestore.getInstance()}
    private var listaMateriales = ListaMateriales(arrayListOf())

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_nuevo_material)


        val nombreEditText = findViewById<EditText>(R.id.nombreNuevoMaterial)
        val tipoEditText = findViewById<EditText>(R.id.tipoNuevoMaterial)

        val buttonAceptar = findViewById<Button>(R.id.buttonAceptarNuevoMaterial)

        buttonAceptar.setOnClickListener{

            db.collection("slf-management").document("material").get().addOnCompleteListener
{
                task ->
                if (task.isSuccessful){
                    listaMateriales = task.result!!.toObject(ListaMateriales::class.java)!!

                    var materialItem = MaterialItem(listaMateriales.listaMaterial.size+1,
nombreEditText.text.toString(), tipoEditText.text.toString(), true, 0, arrayListOf())
                    listaMateriales.listaMaterial.add(materialItem)
                    val map = hashMapOf<String, Any>("listaMaterial" to listaMateriales)

db.collection("slf-management").document("material").set(listaMateriales).addOnCompleteLi
stener{task ->
                        if (task.isSuccessful){
                            Toast.makeText(this, "Material añadido con exito",
Toast.LENGTH_SHORT).show()
                            finish()
                        }else{
                            Toast.makeText(this, "Error", Toast.LENGTH_SHORT).show()
                        }
```

```
                }

            }else{
                Toast.makeText(this@NuevoMaterialActivity, task.exception.toString(),
Toast.LENGTH_LONG).show()
            }
        }

    }

    /**
     * IMAGEN
     */
    val imagen = findViewById<ImageView>(R.id.imagenNuevoMaterial)
    imagen.setOnClickListener{
        Toast.makeText(this, "Hacer foto", Toast.LENGTH_SHORT).show()
    }

  }

}


class NuevoPersonalActivity : AppCompatActivity() {

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_nuevo_personal)

    /**
     * CREACIÓN DE LISTA E ITEMS (CAMBIAR POR BASE DE DATOS)
     */
    var listaElementos: ArrayList<NuevoElementoItem> = arrayListOf()

    val elemento1 = NuevoElementoItem("Nombre", "")
    val elemento2 = NuevoElementoItem("Teléfono", "")
    val elemento3 = NuevoElementoItem("DNI", "")
    val elemento4 = NuevoElementoItem("Dirección", "")
    val elemento5 = NuevoElementoItem("Tlf. contacto externo", "")
    val elemento6 = NuevoElementoItem("Nº Seguridad Social", "")
    val elemento7 = NuevoElementoItem("Activo", "")

    /**
     * INSERTANDO ITEMS EN LISTA
     */
    listaElementos.add(elemento1)
```

```
        listaElementos.add(elemento2)
        listaElementos.add(elemento3)
        listaElementos.add(elemento4)
        listaElementos.add(elemento5)
        listaElementos.add(elemento6)
        listaElementos.add(elemento7)

        /**
         * SINCRONIZACIÓN CON LOS ADAPTERS
         */
        val adapter = NuevoElementoItemAdapter(listaElementos)
        val recycler =
recyclerNuevoPersonal.findViewById<androidx.recyclerview.widget.RecyclerView>(com.exa
mple.slf_management.R.id.recyclerNuevoPersonal)
        val layoutManager= LinearLayoutManager(this@NuevoPersonalActivity,
LinearLayoutManager.VERTICAL, false)
        recycler.layoutManager = layoutManager
        recycler.adapter = adapter

        /**
         * BOTON
         */
        val buttonAceptar = findViewById<Button>(R.id.buttonAceptarNuevoPersonal)
        buttonAceptar.setOnClickListener{
            Toast.makeText(this, "Personal añadido", Toast.LENGTH_SHORT).show()
            finish()
        }
        /**
         * IMAGEN
         */
        val imagen = findViewById<ImageView>(R.id.imagenNuevoPersonal)
        imagen.setOnClickListener{
            Toast.makeText(this, "Hacer foto", Toast.LENGTH_SHORT).show()
        }
    }
}
```

```
class PersonalActivity : AppCompatActivity() {

    private val nombrePersonalActivity by lazy {
findViewById<TextView>(R.id.nombrePersonalActivity) }
    private val imagenPersonalActivity by lazy {
findViewById<ImageView>(R.id.imagenPersonalActivity) }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_personal)
        setSupportActionBar(findViewById(R.id.toolbarPersonal))

        intent.extras?.let {
            val nombrePersonal = it.getString("nombrePersonal")
            val imagenPersonal = it.getInt("imagenPersonal")
            nombrePersonalActivity.text = nombrePersonal
            imagenPersonalActivity.setImageResource(imagenPersonal)
        }
        /**
         * INSERTANDO TEXTO
         */
        val nombrePersonal = findViewById<EditText>(R.id.nombrePersonal)
        val telefonoPersonal = findViewById<EditText>(R.id.telefonoPersonal)
        val dniPersonal = findViewById<EditText>(R.id.dniPersonal)
        val direccionPersonal = findViewById<EditText>(R.id.direccionPersonal)
        val telefonoExternoPersonal = findViewById<EditText>(R.id.telefonoExternoPersonal)
        val numeroSSPersonal = findViewById<EditText>(R.id.numeroSSPersonal)

        nombrePersonal.setText("Juan Carlos")
        telefonoPersonal.setText("655147852")
        dniPersonal.setText("254786325V")
        direccionPersonal.setText("c/ Dos Aceras 3")
        telefonoExternoPersonal.setText("658985214")
        numeroSSPersonal.setText("985147652")

        /**
         * IMAGEN
         */
        val imagen = findViewById<ImageView>(R.id.imagenPersonalActivity)
        imagen.setOnClickListener{
            Toast.makeText(this, "Hacer foto", Toast.LENGTH_SHORT).show()
        }
    }

    /**
     * INSERTANDO EL BOTON EDITAR
```

```
     */
    override fun onCreateOptionsMenu(menu: Menu?): Boolean {
        menuInflater.inflate(R.menu.edit_toolbar, menu)
        return true
    }

    /**
     * FUNCION ONCLICK DEL BOTON EDITAR
     */
    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        Toast.makeText(this, "Cambios Guardados", Toast.LENGTH_SHORT).show()
        when(item.itemId){
            R.id.editProfile -> {}
        }
        return true
    }

}
```

Fragments

```
/**
 * A simple [Fragment] subclass.
 */
class EventosFragment : Fragment() {
    private lateinit var recyclerView: RecyclerView
    private var gridLayoutManager: GridLayoutManager? = null


    @RequiresApi(Build.VERSION_CODES.O)
    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
                              savedInstanceState: Bundle?): View? {

        val view = inflater.inflate(R.layout.fragment_eventos, container, false)

        recyclerView = view!!.findViewById(R.id.recyclerEventos) as RecyclerView

        gridLayoutManager = GridLayoutManager(context, 1)
        recyclerView?.layoutManager = gridLayoutManager
        recyclerView?.setHasFixedSize(true)

        /**
         * CREACIÓN DE LISTA E ITEMS (CAMBIAR POR BASE DE DATOS)
         */
        var listaEventos:ArrayList<EventoItem> = ArrayList()

        val evento1 = EventoItem(1, "Show de Funky", "Málaga", LocalDate.now())
        val evento2 = EventoItem(2, "Show de Wiwi", "Pizarra", LocalDate.now())
        val evento3 = EventoItem(3, "Show de Rickypin", "San Pedro", LocalDate.now())
        val evento4 = EventoItem(4, "Show de Candy", "Alhaurin el Grande", LocalDate.now())

        /**
         * INSERTANDO ITEMS EN LISTA
         */
        listaEventos.add(evento1)
        listaEventos.add(evento2)
        listaEventos.add(evento3)
        listaEventos.add(evento4)

        /**
         * SINCRONIZACIÓN CON EL ADAPTER
         */
        val adapter = EventosAdapter(inflater.context, listaEventos)
```

```
        /**
         * FUNCION ONCLICK QUE DIRIGE AL ACTIVITY EVENTO
         */
        adapter.setEventosListener(object : EventosAdapter.EventosListener {
            override fun onClick(position: Int) {
                val intent = Intent(context, EventoActivity::class.java)
                val bundle = Bundle()
                bundle.putInt("idEvento", listaEventos[position].idEvento)
                bundle.putString("servicioSolicitado", listaEventos[position].servicioSolicitado)
                bundle.putString("localidad", listaEventos[position].localidad)
                bundle.putString("fecha", listaEventos[position].fecha.toString())

                intent.putExtras(bundle)
                startActivity(intent)
            }

        })

        recyclerView.adapter = adapter

        return view
    }

}



/**
 * A simple [Fragment] subclass.
 */
class MaterialFragment : Fragment() {
    private lateinit var recyclerView: RecyclerView
    private var gridLayoutManager: GridLayoutManager? = null


    @RequiresApi(Build.VERSION_CODES.O)
    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
                              savedInstanceState: Bundle?): View? {

        val view = inflater.inflate(R.layout.fragment_material, container, false)

        recyclerView = view!!.findViewById(R.id.recyclerMaterial) as RecyclerView

        gridLayoutManager = GridLayoutManager(context, 3)
        recyclerView?.layoutManager = gridLayoutManager
        recyclerView?.setHasFixedSize(true)
```

```
        /**
         * CREACIÓN DE LISTA E ITEMS (CAMBIAR POR BASE DE DATOS)
         */
        var listaComentarios:ArrayList<ComentarioItem> = ArrayList()

        var listaMaterial:ArrayList<MaterialItem> = ArrayList()

        val material = MaterialItem(1,"Altavoz Behringer Pequeño","Sonido", true,
R.drawable.material_icon, listaComentarios)
        val material2 = MaterialItem(2,"Altavoz de Bateria","Sonido", true,
R.drawable.material_icon, listaComentarios)
        val material3 = MaterialItem(3,"Microfono de diadema","Sonido", false,
R.drawable.material_icon, listaComentarios)
        val material4 = MaterialItem(4,"Microfono de mano","Sonido", true,
R.drawable.material_icon, listaComentarios)
        val material5 = MaterialItem(5,"Maleta de sonido","Sonido", false,
R.drawable.material_icon, listaComentarios)
        val material6 = MaterialItem(6,"Caja de animación","Animacion", false,
R.drawable.material_icon, listaComentarios)
        val material7 = MaterialItem(7,"Traje de Spider-Man","Vestuario", true,
R.drawable.material_icon, listaComentarios)
        val material8 = MaterialItem(8,"Pie de piano","Montaje", false,
R.drawable.material_icon, listaComentarios)

        /**
         * INSERTANDO ITEMS EN LISTA
         */
        listaMaterial.add(material)
        listaMaterial.add(material2)
        listaMaterial.add(material3)
        listaMaterial.add(material4)
        listaMaterial.add(material5)
        listaMaterial.add(material6)
        listaMaterial.add(material7)
        listaMaterial.add(material8)

        /**
         * SINCRONIZACIÓN CON EL ADAPTER
         */
        val adapter = MaterialAdapter(inflater.context, listaMaterial)

        /**
         * FUNCION ONCLICK QUE DIRIGE AL ACTIVITY MATERIAL
         */
        adapter.setMaterialListener(object : MaterialAdapter.MaterialListener {
            override fun onClick(position: Int) {
```

```
            val intent = Intent(context, MaterialActivity::class.java)
            val bundle = Bundle()
            bundle.putString("nombreMaterial", listaMaterial[position].nombreMaterial)
            bundle.putInt("imagenMaterial", listaMaterial[position].imagenMaterial)
            intent.putExtras(bundle)
            startActivity(intent)
        }

    })

    recyclerView.adapter = adapter

    return view
  }

}



/**
 * A simple [Fragment] subclass.
 */
class PersonalFragment : Fragment() {
  private lateinit var recyclerView: RecyclerView
  private var gridLayoutManager: GridLayoutManager? = null


  @RequiresApi(Build.VERSION_CODES.O)
  override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
                            savedInstanceState: Bundle?): View? {

    val view = inflater.inflate(R.layout.fragment_personal, container, false)

    recyclerView = view!!.findViewById(R.id.recyclerPersonal) as RecyclerView

    gridLayoutManager = GridLayoutManager(context, 2)
    recyclerView?.layoutManager = gridLayoutManager
    recyclerView?.setHasFixedSize(true)

    /**
     * CREACIÓN DE LISTA E ITEMS (CAMBIAR POR BASE DE DATOS)
     */
    var listaPersonal:ArrayList<PersonalItem> = ArrayList()

    val personal = PersonalItem("Juan Carlos", R.drawable.personal_icon)
    val personal2 = PersonalItem("Javier", R.drawable.personal_icon)
```

```
        val personal3 = PersonalItem("Juan Carlos", R.drawable.personal_icon)
        val personal4 = PersonalItem("Javier", R.drawable.personal_icon)

        /**
         * INSERTANDO ITEMS EN LISTA
         */
        listaPersonal.add(personal)
        listaPersonal.add(personal2)
        listaPersonal.add(personal3)
        listaPersonal.add(personal4)

        /**
         * SINCRONIZACIÓN CON EL ADAPTER
         */
        val adapter = PersonalAdapter(inflater.context, listaPersonal)

        /**
         * FUNCION ONCLICK QUE DIRIGE A ACTIVIRY PERSONAL
         */
        adapter.setPersonalListener(object : PersonalAdapter.PersonalListener {
            override fun onClick(position: Int) {
                val intent = Intent(context, PersonalActivity::class.java)
                val bundle = Bundle()
                bundle.putString("nombrePersonal", listaPersonal[position].nombrePersonal)
                bundle.putInt("imagenPersonal", listaPersonal[position].imagenPersonal)
                intent.putExtras(bundle)
                startActivity(intent)
            }

        })

        recyclerView.adapter = adapter

        return view
    }

}
```

Items

```
data class ComentarioItem(val nombreComentario:String, val comentario:String, val fecha:
String) {
}

class EventoItem(val idEvento:Int, val servicioSolicitado:String, val localidad:String, val
fecha: LocalDate) {
}

class FichaItem(val imagenFicha:Int, val nombreFicha: String) {
}

class ListaComentarios(var listaComentario:ArrayList<Comentario>) {
    constructor():this(arrayListOf())
}

class ListaItem(val tituloItemLista:String, val informacionItemLista: String) {
}

class ListaMateriales(var listaMaterial:ArrayList<MaterialItem>) {
    constructor():this(arrayListOf())
}

class MaterialItem(val idMaterial:Int, val nombreMaterial:String,val tipoMaterial:String, val
disponibilidadMaterial:Boolean, val imagenMaterial: Int, val comentariosMaterial:
ArrayList<ComentarioItem>) {
    constructor():this(0, "", "", true, 0, arrayListOf())
}

class NuevoElementoItem(val tituloNuevaInformacion:String, val nuevaInformacion: String) {
}

class PersonalItem(val nombrePersonal:String, val imagenPersonal: Int) {
}


class TrabajadorItem(val imagenTrabajador:Int, val nombreTrabajador: String) {
}
```

```
data class MenuComentario (
    var menu : ArrayList<Section>
)

data class Section (
    var nombre : String,
    var comentarios : ArrayList<Comentario>,
    var isExpanded : Boolean
)

data class Comentario (
    var nombre : String,
    var comentario : String,
    var fecha : String
){
    constructor():this("","","")
}

data class MenuEvento (
        var menu : ArrayList<SectionEvento>
)

data class SectionEvento (
        var nombre : String,
        var eventos : ArrayList<Evento>,
        var isExpanded : Boolean
)

data class Evento (
        var id : Int,
        var localidad : String,
        var servicioSolicitado : String,
        var fecha : LocalDate
)
```

11 - Anexo 2: Opinión personal sobre la FCT

Por las medidas de precaución dadas para prevenir el COVID-19, no he podido realizar mi FCT en Marzo de 2020, en consecuencia las podré realizar en Septiembre de 2020.

Una vez realizadas, rellenaré este apartado.

Elaborado por: Juan Carlos Alarcón Pedraza

Fecha de entrega: 26/06/2020

Email: alarconpedrazajuancarlos@gmail.com