EASYSPORT

Memoria de la aplicación

Información Relevante

Curso: 2019-2020
Tutora: Anahí Mula De La Banda
Ciclo: Desarrollo de Aplicaciones Multiplataforma

Oscar Garcia-Redondo Gonzalez

Índice

Abstract

Castellano

En este proyecto se busca mostrar todos los pasos en el desarrollo de una aplicación que facilite a la gente el hacer ejercicio cuando las circunstancias no son propicias. El proyecto no incluirá solo el desarrollo de la aplicación en sí, sino que también mostrará la organización necesaria para que la empresa pueda organizar los recursos humanos y mecánicos (tanto hardware como software) necesarios para el desarrollo, de una forma sencilla, clara y fácil de entender para todo aquel que desee leerlo

Inglés

In this project we show all the steps we need to do for an application's development, which will be able to make easier for people to do exercise when circumstances are not favourable. We are not just showing the development procces, but also all the preparations that the bussiness need, so it can preprare both human and mecanical resources (including hardware and software) needed for the development. We also include the costs that they had and show all in a easy and understable way to allow everyone who whish to read it to understand everything

Justificación del proyecto

La idea de esta aplicación surge a partir del enclaustramiento forzoso al que se somete la población española en estos días, la cual ha generado, en la mayoría de los casos, un aumento exponencial del tiempo libre disponible, como una gran reducción en el número de opciones disponibles a la hora de realizar actividades que permitan mantenerse activo y, siendo esta una de las mejores maneras (si no la mejor) de mantener la salud y el bienestar psicológico (por no mencionar también el físico), esta aplicación resultaría una gran herramienta para la población general.

Introducción

Desde principios de marzo del mes pasado la población española ha permanecido en una situación inédita en la historia de la democracia española, confinados y con su capacidad de movimiento restringida, tanto en horarios como en localizaciones. Esto ha provocado que se den situaciones que no se habían ni siquiera pensado, además de conflictos derivados de estas y que, en una u otra forma generan estrés en las personas,

así como muchas otras afecciones que, si bien no son generadas directamente, la situación actual actúa como un disparador y las hace aparecer.

Ante esta situación la actividad física se presenta como un alivio social y personal, ya que posee numerosos beneficios, tanto físicos como psicológicos, que pueden favorecer la salud y mejorar el estado de ánimo, que tanta falta hace en esta situación. Aquí se presentan algunos de ellos:

Beneficios biológicos

Mejora la forma y la resistencia física
Regula la presión arterial
Mejora la densidad ósea
Mejora la resistencia a la insulina
Ayuda a mantener el peso corporal
Aumenta el tono y la fuerza muscular
Mejora la flexibilidad y la movilidad de las articulaciones
Reduce la sensación de fatiga

Beneficios psicológicos

Aumenta la autoestima
Mejora la imagen personal
Reduce el aislamiento social
Rebaja la tensión y el estrés
Reduce los niveles de depresión
Ayuda a la relajación
Aumenta el estado de alerta
Disminuye el número de accidentes laborales
Menor grado de agresividad, ira, angustia…
Incrementa el bienestar general
Practicado en familia ayuda a estrechar los lazos familiares

Estos han sido los detonantes que han desencadenado que esta sea la idea que elija desarrollar. Así pues, resumiendo, mi aplicación busca acercar todos estos beneficios a la población, al darles una herramienta con ejercicios que pueden realizar, los cuales además no requieren de grandes inversiones (o directamente de ninguna) para poder realizarlos, lo que permite superar también otro factor limitante, pues el enclaustramiento y la parada de la actividad nacional ha generado en la población un descenso en sus capacidades económicas en el mejor de los de los casos, y en el peor de ellos, las han eliminado totalmente

Esta aplicación además poseerá otras funciones, las cuales son, por un lado, actuar como diario, permitiendo al usuario anotar cualquier cosa relevante, como por ejemplo los ejercicios que realizó o la dieta que lleva (elemento que, si bien no cubre esta aplicación, es un factor crucial para cuidar la salud y debe tenerse muy en cuenta). Estos registros podrán ser consultados más adelante a voluntad por el usuario.

La otra función de esta aplicación dará acceso al usuario a la página de Aemet de forma directa, donde podrá consultar el clima en su zona. Esta función se ha pensado para, en primer lugar, las personas que dispongan de patio privado y deseen realizar el ejercicio al aire libre, y, por otro, para cuando se vaya permitiendo salir a la población y esté disponible la opción de realizar deporte en la calle.

Cada una de las secciones de la aplicación están dedicadas a facilitar y acercar el deporte a la población. Como dice el dicho popular, *mens sana in corpore sano.*
Aquí se detallan de forma sintetizada los objetivos de la aplicación:

Objetivos

- Aportar ideas sobre ejercicios realizables cuando no es posible acceder a material específico y no se puede costear los gastos de ir al gimnasio
- Permitir al usuario llevar un registro de la actividad realizada y consultarla en cualquier momento, de forma sencilla
- Dar al usuario, mediante acceso a internet, la capacidad de comprobar el tiempo para evitar sorpresas climáticas

En el proyecto no solo se incluyen todos los datos que deben conocerse sobre el planteamiento y desarrollo de la aplicación, sino que también se detallan los programas y el hardware que el equipo necesitará para desarrollarla y los costes de los mismos, además del hardware necesario para poder utilizar la aplicación

Desarrollo

El marco empresarial

Esta aplicación no se encuentra destinada a una empresa, sino que se va destinada al público general. Por este motivo, no se puede definir una "empresa objetivo", pues, aunque se considere a la sociedad como una empresa, las variables que la componen, como pueden ser por ejemplo la edad de las personas, su condición económica o, incluso, su capacidad de acceso a la tecnología. Por todo esto se tratará de optimizar la aplicación lo máximo posible, de forma que sea lo más competitiva posible

Sobre la empresa

Para poder desarrollar cualquier aplicación necesitamos una serie de recursos, tanto materiales como humanos. Para ello, nuestra empresa formará un equipo de trabajo, que nos permita no solo diseñar, sino también desarrollar, la aplicación de la forma más eficiente posible. Este equipo será pequeño, pues se trata de una aplicación cuyo destino no es tanto el dinero como dar un servicio por la salud de la gente. Este equipo se compondrá de:

- **Jefe de proyecto**: Se encargará de diseñar la aplicación y coordinar al equipo. También escribirá el código responsable de la interfaz y se encargará del grueso de las pruebas de testeo de la aplicación
- **Desarrolladores**: Habrá 2, y cada uno de ellos tendrá asignadas las siguientes funciones:
  - Desarrollador 1: Se encargará de investigar, analizar y contrastar la información necesaria para poder realizar la aplicación de forma eficiente y que aporte los resultados esperados. También desarrollará el código correspondiente a los ejercicios de la aplicación, realizará las imágenes necesarias y realizará funciones de testeo
  - Desarrollador 2: Se encargará de preparar la base de datos, los ficheros de datos necesarios y todo el código correspondiente a ella. También realizará funciones en el testeo de la aplicación

Ambos desarrolladores se coordinarán en todo momento entre ellos y con el jefe de equipo

Utilizando este equipo de trabajo, se espera poder optimizar el tiempo para reducir el tiempo de desarrollo lo máximo posible para optimizar los costes, sin reducir con ello la estabilidad y las garantías de buen funcionamiento de la aplicación. Para desarrollar su trabajo, el equipo necesitaría, para cada uno de los miembros, un equipo de las siguientes características al menos:

Características de sistema:

- Procesador: AMD Ryzen 7 3700U 4 Núcleos,8 Subprocesos. Caché: 4MB Level 3, 2.3GHz
- Memoria RAM: 8GB (4GB en placa + 4GB) DDR4 2400 MHz
- Disco duro 512GB SSD M.2 PCIe Gen3x2NVMe
- Display 15.6" LED Retroiluminado / 60 Hz / NanoEdge(Borde Estrecho) / Ultra Slim / 200nits / FullHD/ Antirreflejos / NTSC 45%
- Tarjeta gráfica AMD Radeon RX Vega 10 Graphics
- Conectividad:
  - Wi-Fi 5 (802.11ac) 1x1
  - Bluetooth 4.1
- Cámara, Micrófono y Salida de entrada de auriculares y entrada de micrófono 35mm(combo)
- Batería 32Wh, 2 Celdas, Polímero de litio
- Conexiones:
  - 2 USB 2.0
  - 1 USB-CTM 3.1(GEN 1)
  - 1 HDMI
  - Lector de tarjetas
  - Bloqueo de seguridad Kengsinton
  - Entrada de corriente
- Dimensiones: 360mm(Ancho)x235mm(Profundidad)x22.9mm(Altura)
- Peso: 1.9Kg
- Sistema Operativo: Microsoft Windows 10 Pro 64 Bits OEM Versión PT (añadido aparte)

Aplicaciones instaladas:

- Edición de imágenes: GIMP
- IDE: Dado que la aplicación desarrollada será para Android, usaremos Android Studio
- Elaboración de la documentación: Microsoft Office, Foxit Reader

Además, para las labores de búsqueda de información y organización del equipo es necesario disponer de una conexión a internet

Para poder trabajar también es necesario un lugar en el que reunirse. Para reducir costes, esto puede hacerse en un local o, dado el tamaño del equipo, y para una reducción mayor de los costes, en las casas de los propios miembros del equipo. Por ello, las reuniones de coordinación se realizarán en el domicilio del jefe de proyecto o, en su defecto, las reuniones se realizarán mediante video llamada. Esto implica que, al menos de inicio, no será necesaria una red, ni para la empresa ni para la aplicación. Se plantearía su implantación, si la empresa escalara en el futuro al tamaño suficiente, mas ahora sería contraproducente debido al coste que tendría su implantación

Presupuesto para la puesta en marcha de la empresa

La inversión inicial necesaria para el desarrollo de la app sería la siguiente. Esta estimación no incluye los impuestos requeridos para legalizar la empresa como puede ser el alta de autónomos u otros impuestos que sea necesario pagar y que pueden llegar a variar según la legislación vigente. Sin embargo, también incluye el coste de la licencia de publicación de aplicaciones en Google Play, puesto que es importante el poseer una plataforma de fácil distribución y esta una de las mayores.

Así pues, el presupuesto se reduce a los costes que tendría la aplicación si hubiera que equipar de 0 a los empleados para que puedan realizar todos sus trabajos (con la excepción de la licencia de publicación, que pese a no ser material también se incluye en él). Se ha estimado un tiempo de trabajo en la aplicación, pruebas incluidas, de 1 mes, por lo que los gastos en personal reflejarán solo este periodo de tiempo:

	Personal	Equipamiento y licencias	Total
Salarios	2023.92€(x3)		6071,78€
Equipo		499€(x3)	1.497,00€
Sistema Operativo		150,51€(x3)	451,53€
Microsoft Office		266€(x3)	798€
GIMP		0€(x3)	0€
Android Studio		0€(x3)	0€
Tasa desarrollador Google Play		25$ (~20€)	20€
Coste total			8.838,31€

Tabla 1: Desglose de los costes de puesta en marcha de la actividad profesional, sin contar con los costes burocráticos de los permisos de la empresa. También se han excluido los impuestos que paga la empresa para el mantenimiento de sus trabajadores, más allá de los salarios

Se han elegido estos elementos por los siguientes motivos:

- El equipo ofrece equilibrio entre calidad y precio
- El sistema operativo es ampliamente extendido en uso y tiene buen soporte, así como el paquete de Office
- Respecto al IDE, se trata del estándar para desarrollar en Android
- GIMP se eligió dado que ofrece buenas capacidades para el trabajo, siendo además gratuito
- Dado el tamaño de la empresa, no se hace necesario el uso de un ERP, aunque de ser necesario si la empresa escala a un mayor tamaño, cuando necesite de un local de forma forzosa, podría emplearse ODOO. Esto es así porque se trata de una empresa que está empezando y solo puede encargarse de uno o como mucho 2 proyectos a la vez

Planteamiento de la aplicación

Idea base

La situación que ha pasado la población ha generado una necesidad de herramientas que les permitan superar esta situación. Han surgido muchas iniciativas que buscan entretener a la población de una u otra forma, pero… ¿Y si hubiera alguna manera de

no solo entretener a la población sino también ayudarla a mejorar su salud? Así fue cómo surgió la idea de realizar la aplicación EasySport, una aplicación que reúna por sí misma todas las herramientas necesarias para acercar el deporte a las personas y con esto mejorar la salud y el bienestar de la sociedad de forma fácil y económica.

Esto significa que la aplicación no está destinada a una empresa específica, sino a la sociedad en general, lo que genera la necesidad de construir la aplicación de forma que pueda ser accesible a la mayoría de la población, es decir, hay que optimizarla para que independientemente del acceso a la tecnología que tenga el usuario final, pueda utilizar la aplicación y recibir los beneficios que conllevará su uso. A continuación, se mostrarán los requisitos que debe tener el teléfono para asegurar su funcionamiento

Requisitos para la utilización de EasySport

Características de Hardware necesarias para su instalación y funcionamiento

- 2 GB de RAM
- 4 GB de almacenamiento
- Android 7
- Procesador: 8 Núcleos

Otras características necesarias para el uso de EasySport:

- Acceso a Internet (Necesario para poder comprobar el clima mediante la aplicación)

Diseño de la aplicación

Dada la idea base de la aplicación y su configuración, no existen distintos niveles de acceso y gestión para usuarios y administradores, puesto que la idea principal de la aplicación es que cada usuario la lleve en su móvil, de forma individual. Por esto se podría decir que cada usuario a su vez será su administrador.

A continuación, se detallarán los esquemas de la aplicación, tanto los de caso de uso como el diagrama de clases de la aplicación

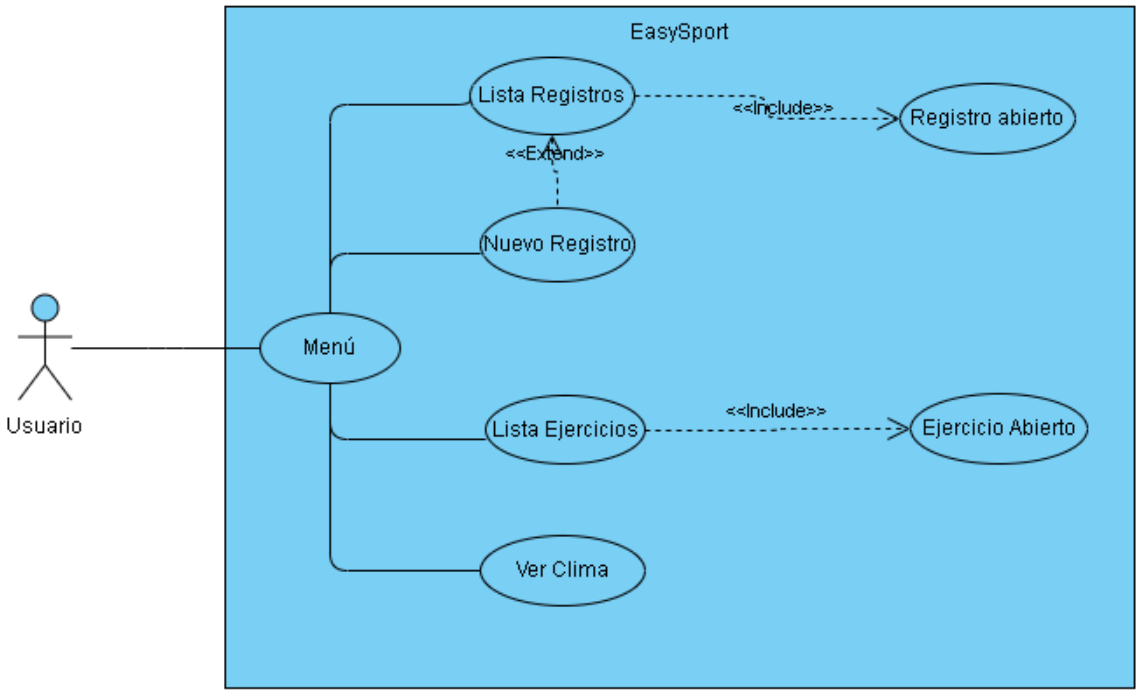


Diagrama 1: Diagrama de casos de uso de EasySport

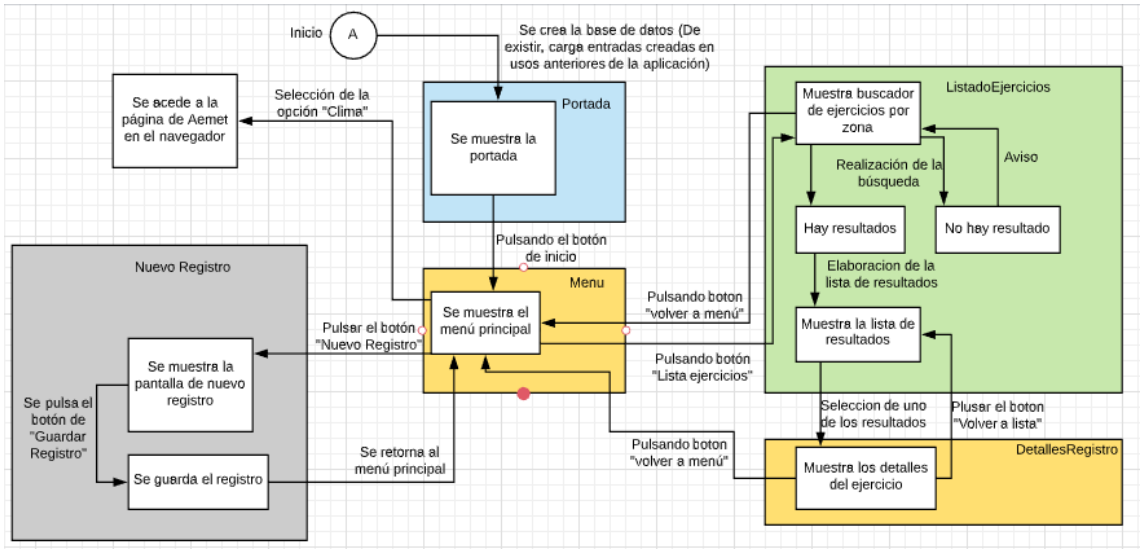


Diagrama 2: Diagrama de estados de EasySport

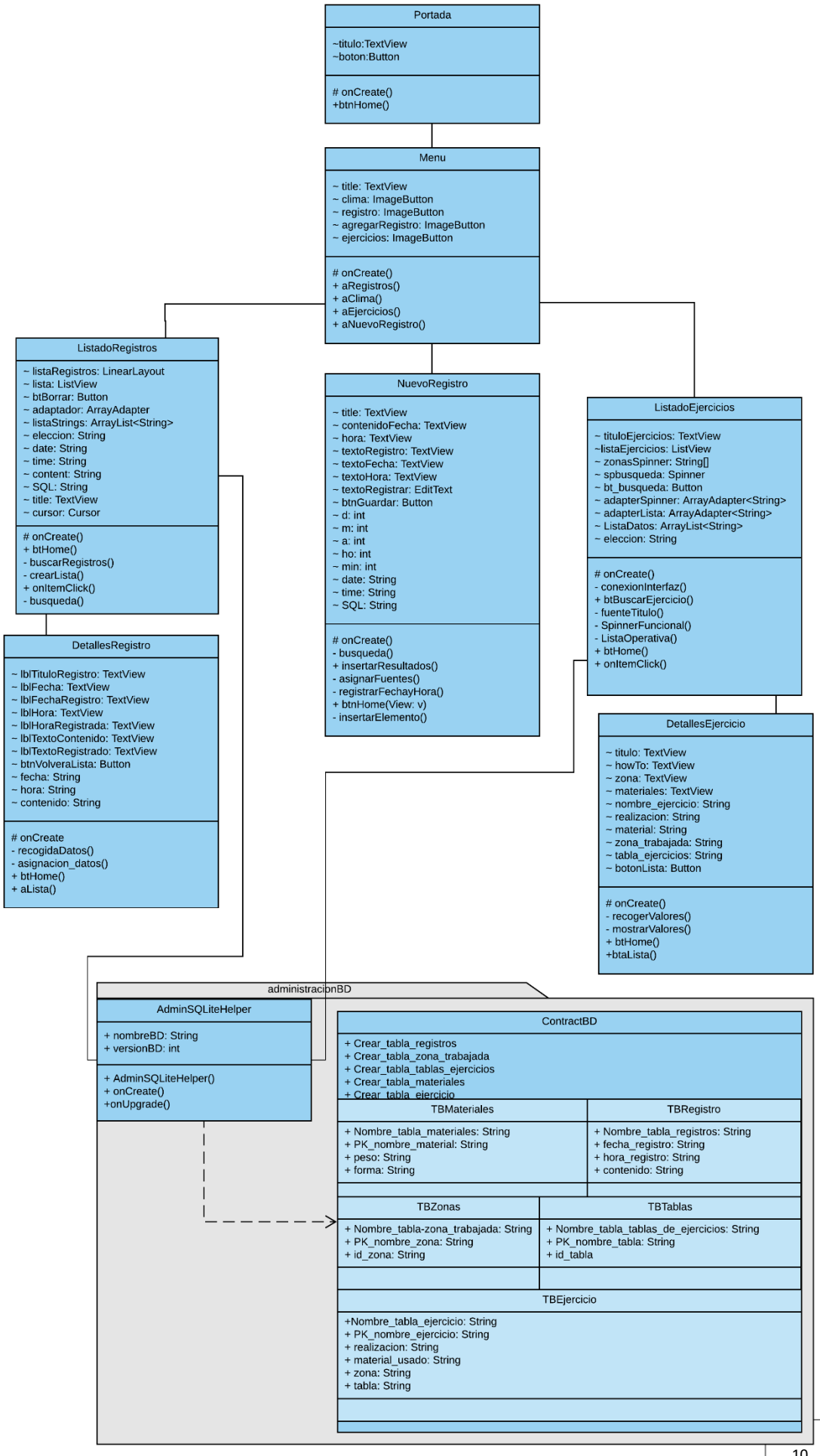

Portada
~titulo:TextView
~boton:Button
onCreate()
+btnHome()
Menu
~ title: TextView
~ clima: ImageButton
~ registro: ImageButton
~ agregarRegistro: ImageButton
~ ejercicios: ImageButton
# onCreate()
+ aRegistros()
+ aClima()
+ aEjercicios()
+ aNuevoRegistro()
ListadoRegistros
~ listaRegistros: LinearLayout
~ lista: ListView
~ btBorrar: Button
~ adaptador: ArrayAdapter
~ listaStrings: ArrayList<String>
~ eleccion: String
~ date: String
~ time: String
~ content: String
~ SQL: String
~ title: TextView
~ cursor: Cursor
# onCreate()
+ btHome()
- buscarRegistros()
- crearLista()
+ onItemClick()
- busqueda()
NuevoRegistro
~ title: TextView
~ contenidoFecha: TextView
~ hora: TextView
~ textoRegistro: TextView
~ textoFecha: TextView
~ textoHora: TextView
~ textoRegistrar: EditText
~ btnGuardar: Button
~ d: int
~ m: int
~ a: int
~ ho: int
~ min: int
~ date: String
~ time: String
~ SQL: String
# onCreate()
- busqueda()
+ insertarResultados()
- asignarFuentes()
- registrarFechayHora()
+ btnHome(View: v)
- insertarElemento()
ListadoEjercicios
~ tituloEjercicios: TextView
~listaEjercicios: ListView
~ zonasSpinner: String[]
~ spbusqueda: Spinner
~ bt_busqueda: Button
~ adapterSpinner: ArrayAdapter<String>
~ adapterLista: ArrayAdapter<String>
~ ListaDatos: ArrayList<String>
~ eleccion: String
# onCreate()
- conexionInterfaz()
+ btBuscarEjercicio()
- fuenteTitulo()
- SpinnerFuncional()
- ListaOperativa()
+ btHome()
+ onItemClick()
DetallesRegistro
~ lblTituloRegistro: TextView
~ lblFecha: TextView
~ lblFechaRegistro: TextView
~ lblHora: TextView
~ lblHoraRegistrada: TextView
~ lblTextoContenido: TextView
~ lblTextoRegistrado: TextView
~ btnVolveraLista: Button
~ fecha: String
~ hora: String
~ contenido: String
# onCreate
- recogidaDatos()
- asignacion_datos()
+ btHome()
+ aLista()
DetallesEjercicio
~ titulo: TextView
~ howTo: TextView
~ zona: TextView
~ materiales: TextView
~ nombre_ejercicio: String
~ realizacion: String
~ material: String
~ zona_trabajada: String
~ tabla_ejercicios: String
~ botonLista: Button
# onCreate()
- recogerValores()
- mostrarValores()
+ btHome()
+btaLista()
administracionBD
AdminSQLiteHelper
+ nombreBD: String
+ versionBD: int
+ AdminSQLiteHelper()
+ onCreate()
+onUpgrade()
ContractBD
+ Crear_tabla_registros
+ Crear_tabla_zona_trabajada
+ Crear_tabla_tablas_ejercicios
+ Crear_tabla_materiales
+ Crear_tabla_ejercicio
TBMateriales
+ Nombre_tabla_materiales: String
+ PK_nombre_material: String
+ peso: String
+ forma: String
TBRegistro
+ Nombre_tabla_registros: String
+ fecha_registro: String
+ hora_registro: String
+ contenido: String
TBZonas
+ Nombre_tabla-zona_trabajada: String
+ PK_nombre_zona: String
+ id_zona: String
TBTablas
+ Nombre_tabla_tablas_de_ejercicios: String
+ PK_nombre_tabla: String
+ id_tabla
TBEjercicio
+Nombre_tabla_ejercicio: String
+ PK_nombre_ejercicio: String
+ realizacion: String
+ material_usado: String
+ zona: String
+ tabla: String

Diagrama 3: Diagrama de clases de EasySport

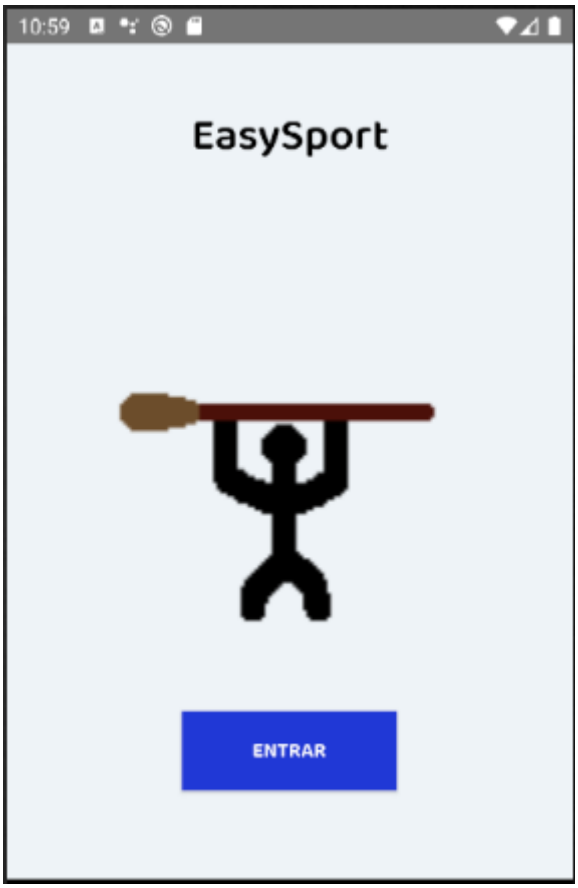
Sobre la interfaz de usuario

Para el aspecto visual de EasySport utilizaremos la librería gráfica que incorpora por defecto Android Studio, puesto que aporta un equilibrio entre facilidad de uso y opciones de manejo, junto a un buen acabado

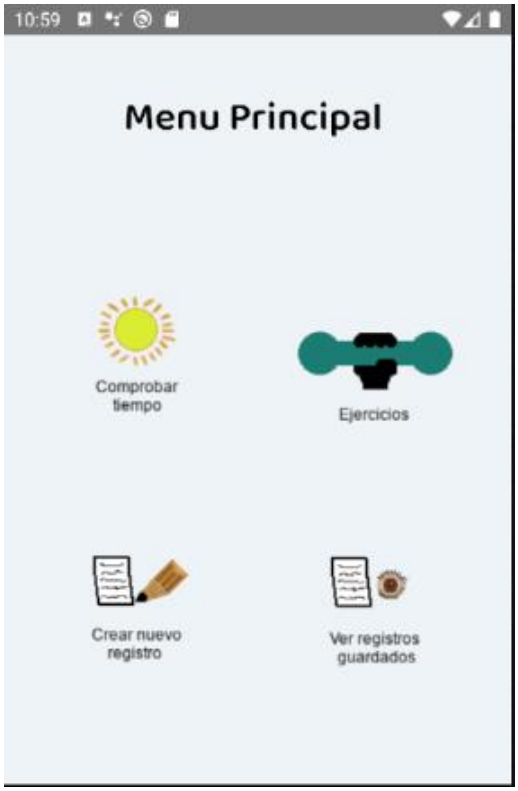
Detalles de la interfaz

A continuación, se mostrará la interfaz de usuario mediante capturas con una descripción. Pero antes, aclararemos un par de puntos que afecta a todas las pantallas:

- Se ha decidido eliminar la barra de acciones de la aplicación para así dejar una interfaz más limpia al usuario
- Se ha aplicado una fuente a los textos que los hace más claros y fáciles de leer, con el objetivo de mejorar la usabilidad de la aplicación.

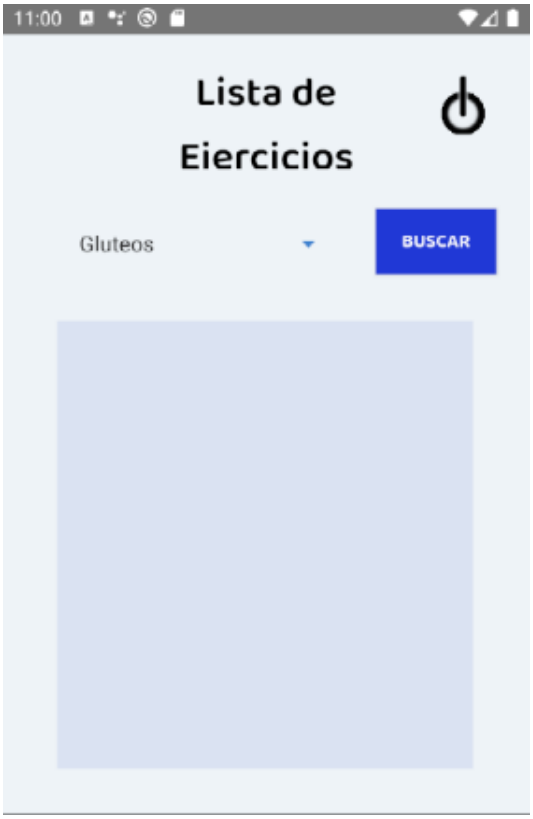


Pasemos ahora a los detalles de la interfaz Esta pantalla no tiene demasiado interés. Simplemente nos muestra el nombre de la aplicación, una imagen representativa de las intenciones que tiene la aplicación (En este caso, el facilitar a las personas hacer deporte sin importar su condición física y sin grandes inversiones económicas) y un botón para adentrarnos en la aplicación



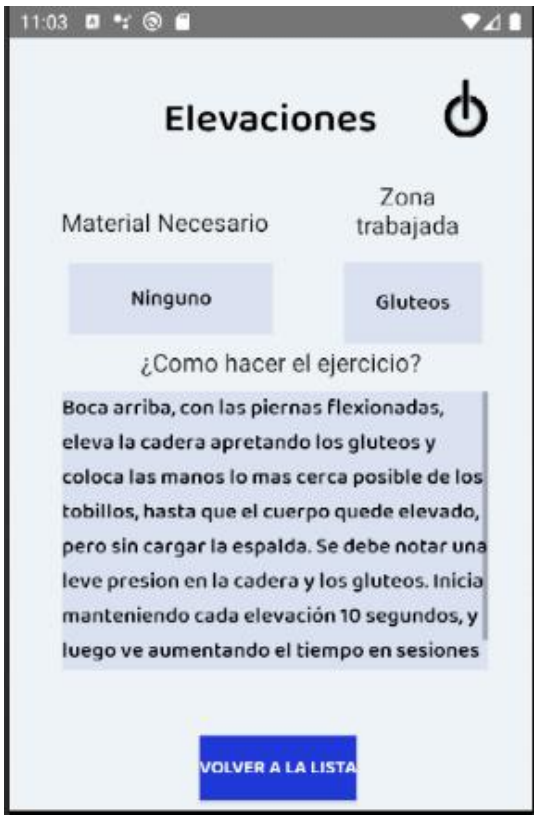
En esta pantalla encontramos acceso a todo lo que se puede hacer en la aplicación. Las opciones son las siguientes:

- **Comprobar tiempo**: Nos permite acceder a la página de Aemet para comprobar el clima y así poder evitar sorpresas al salir a practicar deporte al exterior
- **Ejercicios**: Impreescindible para poder hacer los ejercicios, es el poder saber cómo realizarlos correctamente. Para ello está esta sección
- **Crear nuevo registro**: Gracias a este apartado podremos crear registros de nuestras rutinas para poderlos examinar más adelante
- **Ver registros guardados**: Aquí podremos ver los registros de días pasados, para controlar nuestra actividad



En esta pantalla podremos elegir la zona que queremos trabajar de una lista para, mediante el botón “Buscar” podamos ver los ejercicios disponibles para esa parte del cuerpo

La zona sombreada muestra donde aparecerán datos que vayan a variar, como ejercicios, materiales, u otros datos que varíen entre apartados. En la versión final el color de fondo por el del resto de la pantalla. Se mostrará en cada sección sombreada su utilidad. Esto aplica para



Esta pantalla mostrará los detalles del ejercicio seleccionado, y un botón de acceso al menú principal. Estos detalles incluyen:

- **Material necesario para realizar el ejercicio**
- **Zona principal que trabaja el ejercicio:** Se mostrará solo la principal, ya que muchos ejercicios aunque van enfocados a una zona también pueden trabajar zonas anexas u otras zonas con menor intensidad
- **¿Cómo hacer el ejercicio?**: En esta parte podremos ver una descripción de los pasos necesarios para realizar el ejercicio de forma correcta, entre otros detalles de interés.

En esta sección podremos registrar los ejercicios realizados durante ese día. Junto a lo que queramos registrar se tomarán también la fecha y la hora del registro, para poder organizar después los registros al mostrarlos al usuario. También están presentes un botón para guardar el registro (Que además nos devolverá al menú principal) y el mismo botón presente en el resto de secciones, que cancelará el registro y nos devolverá al menú principal

En esta pantalla se podrán ver todos los registros almacenados por el usuario, listados de forma que sean fáciles de encontrar para el usuario. Solo con tocar el registro deseado pasaremos a ver los detalles de ese registro

En esta pantalla será donde veamos los detalles de cada registro que hayamos creado. Se mostrará la fecha de creación, la ultima hora en la que se modificó el registro y el propio contenido de este. También estarán presentes un botón para volver a la lista de registros y un botón para volver al menú principal

A la hora de diseñar la interfaz se han tenido en cuenta:

- **Simplicidad en el diseño**: Se ha buscado en todo momento presentar un diseño en el que no haya elementos sobrantes
- **Uniformidad**: Para lograrla se han emplazados los elementos clave siempre en el mismo lugar (Titulo de la sección, y botón de volver al menú principal, sobre todo), para permitir siempre al usuario saber de forma intuitiva donde está y como volver al menú principal. También el diseño general se ha unificado para para una cierta fluidez entre las distintas secciones y, así, dar una visión de conjunto a toda la aplicación
- **Formatos de fuente**: Según la importancia del texto mostrado, se han aplicado tamaños de texto distintos, para permitir al usuario conocer de forma inconsciente las partes más importantes y así localizarlas más fácilmente

Propuestas de mejora de la experiencia de usuario

Puestos a pensar en posibles mejoras para la experiencia de usuario, podrían realizarse algunas como las siguientes:

- **Respecto a los ejercicios:** Podrían implementarse opciones para buscar ejercicios de otras formas, como por ejemplo según el material necesario para hacerlo, o la tabla a la que pertenece el ejercicio, y no solo por zonas.

  También podrían añadirse videos a los detalles de ejercicios, sea de forma directa en la aplicación o mediante accesos a YouTube, por ejemplo
- **Respecto a los registros:** Se podrían implementar opciones diversas respecto a la gestión de los mismos, como podría ser visualizarlos en el orden deseado o la capacidad de borrar los registros deseados, sean cuales sean
- **Respecto al aspecto general:** El diseño de las imágenes podría profesionalizarse de forma que tengan un aspecto más pulido y profesional

Sobre la Base de Datos de EasySport

Para el correcto funcionamiento de la aplicación es necesario que incorpore una base de datos. Para implementarla utilizaremos SQLite, debido a su fácil manejo y a que no requiere de ningún proceso extra para su incorporación.

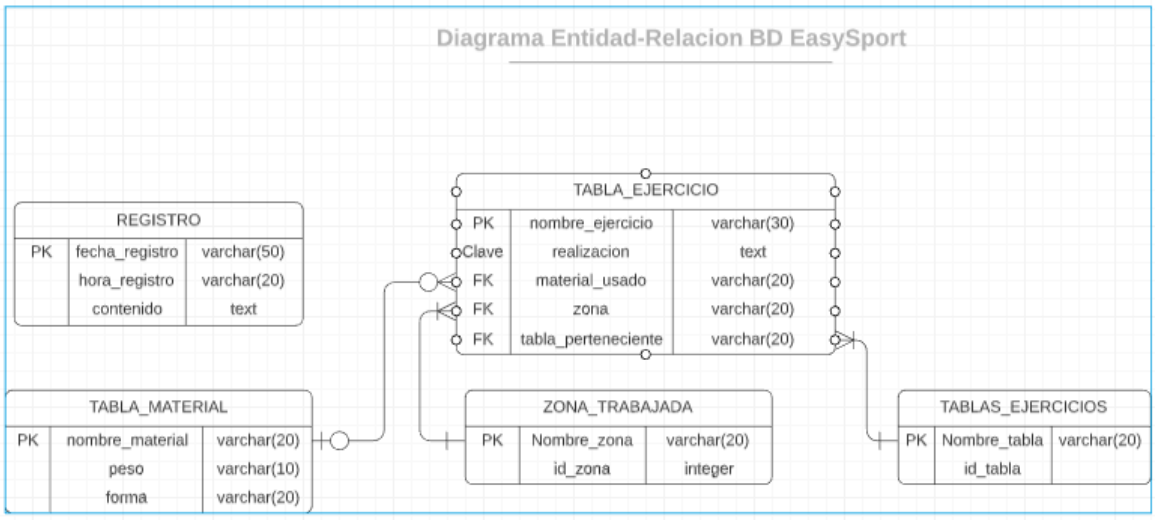


Diagrama 4:Diagrama Entidad-Relación de la Base de Datos de EasySport

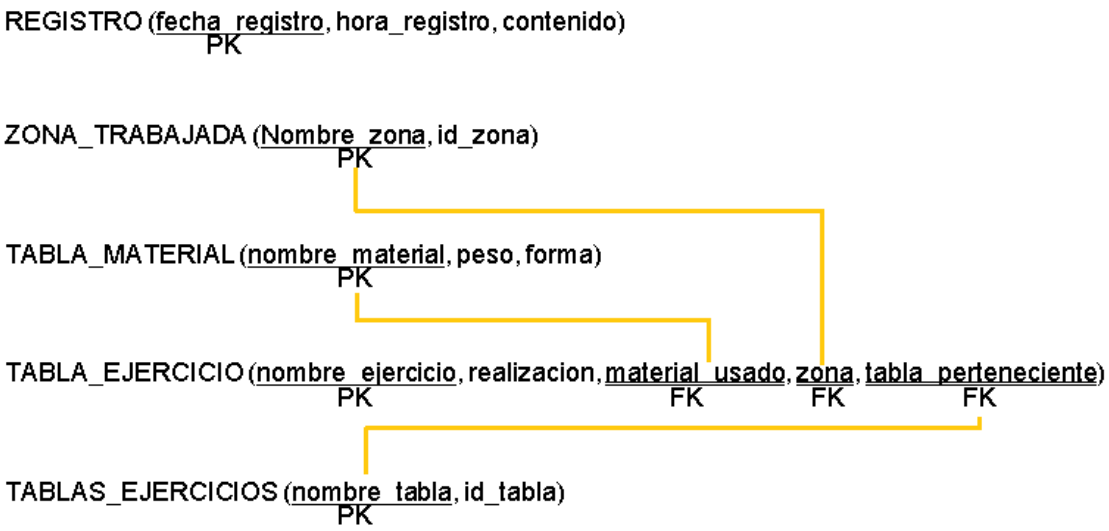


Ilustración 1: Transformación a modelo relacional

```
private ContractBD() {
}

//Tabla REGISTRO
public static class TBRegistro implements BaseColumns {
    public static final String Nombre_tabla_registros = "REGISTROS";
    public static final String fecha_registro = "fecha_registro";
    public static final String hora_registro = "hora_registro";
    public static final String contenido = "contenido";
}

//Tabla TABLA_MATERIAL
public static class TBMateriales implements BaseColumns {
    public static final String Nombre_tabla_materiales = "TABLA_MATERIAL";
    public static final String PK_nombre_material = "nombre_material";
    public static final String peso = "peso";
    public static final String forma = "forma";
}

//Tabla ZONA_TRABAJADA
public static class TBZonas implements BaseColumns {
    public static final String Nombre_tabla_zona_trabajada = "ZONA_TRABAJADA";
    public static final String PK_nombre_zona = "Nombre_zona";
    public static final String id_zona = "id_zona";

}
```

Ilustración 2: Estructura de la base de datos (Parte 1)

```
//TABLA TABLAS_EJERCICIOS
public static class TBTablas implements BaseColumns {
    public static final String Nombre_tabla_tablas_de_ejercicios = "TABLAS_EJERCICIOS";
    public static final String PK_nombre_tabla = "Nombre_tabla";
    public static final String id_tabla = "id_tabla";

}

//Tabla TABLA_EJERCICIO
public static class TBEjercicios implements BaseColumns {
    public static final String Nombre_tabla_ejercicio = "TABLA_EJERCICIO";
    public static final String PK_nombre_ejercicio = "nombre_ejercicio";
    public static final String realizacion = "realizacion";
    public static final String material_usado = "material_usado";
    public static final String zona = "zona";
    public static final String tabla = "tabla_perteneciente";
}
```

Ilustración 3: Estructura de la base de datos (Parte 2)

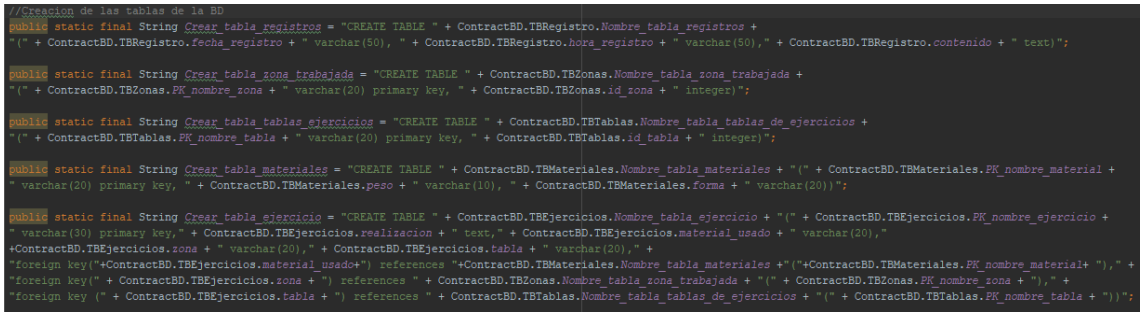
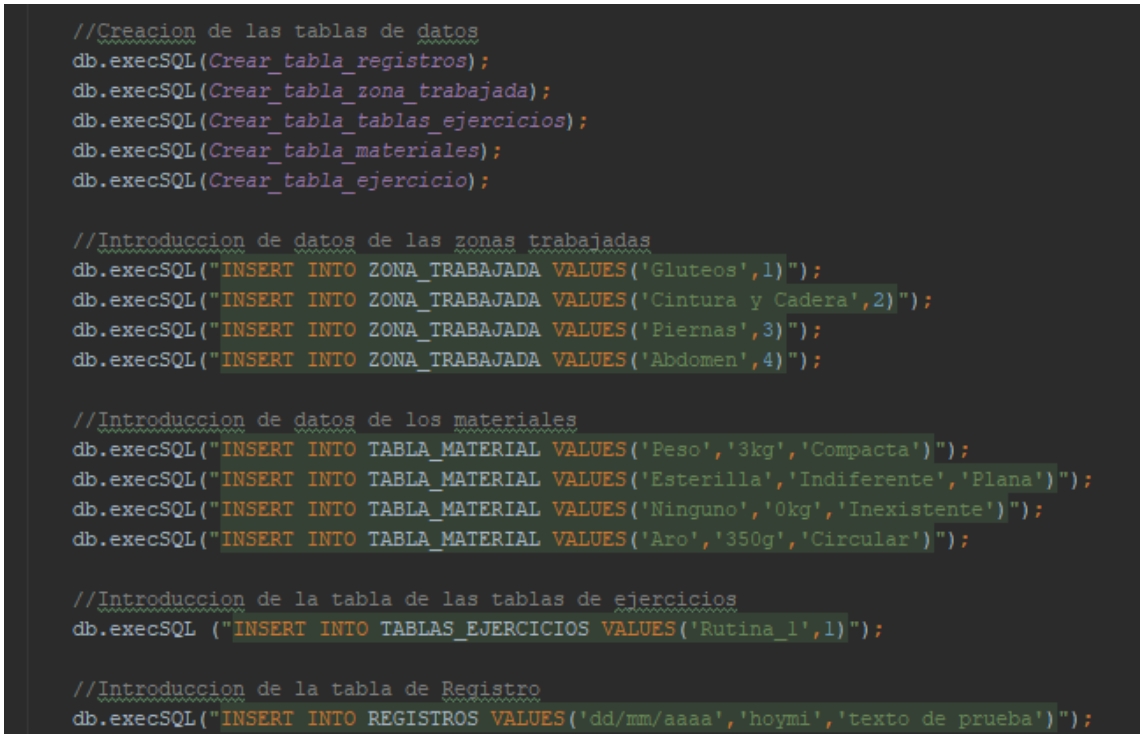


Ilustración 4: Estructura de creación de las tablas de la base de datos



```
//Creacion de las tablas de datos
db.execSQL(Crear_tabla_registros);
db.execSQL(Crear_tabla_zona_trabajada);
db.execSQL(Crear_tabla_tablas_ejercicios);
db.execSQL(Crear_tabla_materiales);
db.execSQL(Crear_tabla_ejercicio);

//Introduccion de datos de las zonas trabajadas
db.execSQL("INSERT INTO ZONA_TRABAJADA VALUES('Gluteos',1)");
db.execSQL("INSERT INTO ZONA_TRABAJADA VALUES('Cintura y Cadera',2)");
db.execSQL("INSERT INTO ZONA_TRABAJADA VALUES('Piernas',3)");
db.execSQL("INSERT INTO ZONA_TRABAJADA VALUES('Abdomen',4)");

//Introduccion de datos de los materiales
db.execSQL("INSERT INTO TABLA_MATERIAL VALUES('Peso','3kg','Compacta')");
db.execSQL("INSERT INTO TABLA_MATERIAL VALUES('Esterilla','Indiferente','Plana')");
db.execSQL("INSERT INTO TABLA_MATERIAL VALUES('Ninguno','0kg','Inexistente')");
db.execSQL("INSERT INTO TABLA_MATERIAL VALUES('Aro','350g','Circular')");

//Introduccion de la tabla de las tablas de ejercicios
db.execSQL ("INSERT INTO TABLAS_EJERCICIOS VALUES('Rutina_1',1)");

//Introduccion de la tabla de Registro
db.execSQL("INSERT INTO REGISTROS VALUES('dd/mm/aaaa','hoymi','texto de prueba')");
```

Ilustración 5: Creación de las tablas e inserción de los datos de base de la base de datos de la aplicación

Respecto a la presencia de varios usuarios en la aplicación

Puesto que la aplicación ha sido diseñada para que cada usuario lleve la aplicación en su teléfono, con una base de datos local, resultaría poco práctico diseñar un manejo para varios usuarios. Por este motivo, la base de datos no contiene varios usuarios.

Tecnologías Web Utilizadas

EasySport usará la conexión a internet del terminal para acceder a la página de Aemet y permitir al usuario buscar la ubicación donde desee comprobar el clima. De esta manera, se accede a un servicio de confianza y a la vez se aligera el peso de la aplicación y sus requerimientos

Planificación del trabajo

Nombre de la actividad	Inicio	Fin
Planteamiento de la aplicación	30/04/2020	07/05/2020
Desarrollo de los esquemas	08/05/2020	11/05/2020
Diseño de la interfaz	12/05/2020	17/05/2020
Concepcion de la base de datos	18/05/2020	21/05/2020
Estructuracion de la base de datos	22/05/2020	25/05/2020
Creación de las imágenes de la aplicación	26/05/2020	26/05/2020
Escritura del codigo de la aplicación	27/05/2020	30/05/2020
Pruebas unitarias de la aplicación	31/05/2020	01/06/2020
Pruebas de funcionamiento integrales	02/06/2020	02/06/2020
Escritura de los manuales necesarios	03/06/2020	03/06/2020

Tabla 2: Diagrama de Gantt con la planificación temporal del trabajo

Pruebas de funcionamiento

Se han realizado pruebas de funcionamiento sobre la aplicación tanto en emulador como en terminal físico, y han sido comprobados todos los elementos. En concreto, este es el desglose de las pruebas realizadas:

- **Funcionamiento de cada interfaz de la aplicación:** Esto incluye tanto la conexión entre interfaces como la aplicación de fuentes, fondos e imágenes de la aplicación
- **Respecto a la lista de ejercicios:** El Spinner funciona bien, se seleccionan correctamente todos los ejercicios de la base de datos que corresponden a cada sección, se seleccionan bien todos los ejercicios y, en los detalles de los mismos, todos los detalles se muestran de forma correcta
- **Respecto a la sección "ver clima":** El enlace funciona correctamente y lleva a donde debe llevar
- **Respecto a la creación de nuevos registros:** Se registran de forma correcta tanto la fecha como la hora. Respecto al texto que se puede introducir, se ha comprobado que acepte tanto mayúsculas como minúsculas, así como caracteres como la "ñ" o las tildes
- **Respecto a la lista de registros:** Se ha comprobado que la lista se genera de forma satisfactoria, incluyendo siempre los elementos creados por el usuario (Para eliminar los fallos en la creación de la lista se ha generado un registro "fantasma" en la base de datos que permite la creación de la tabla en el primer inicio de la aplicación)

- **Respecto a los detalles de cada registro:** Se ha comprobado que se recogen de forma correcta los datos del registro (fecha del registro, hora del registro y texto registrado) y que se ubican en la posición correspondiente

Medidas de seguridad

La empresa no almacenará datos sensibles en lo referente a la aplicación, pero estará pendiente del feedback aportado por los usuarios para en las actualizaciones poder solucionar los distintos bugs o fallos que hayan pasado inadvertidos en las pruebas de seguridad y que los usuarios encuentren y reporten

Documentación del código de la aplicación

Ver archivo “index.html” de la carpeta “JavaDoc proyecto” para ver la documentación completa del proyecto. A continuación, se mostrarán algunas capturas del código fuente:

```
public class Portada extends AppCompatActivity {

    TextView titulo;
    Button boton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_portada);

        //Relacion con la interfaz
        titulo = findViewById(R.id.lbltitulo);
        boton = findViewById(R.id.btEntrar);

        //Asignar fuente
        Typeface Titl = Typeface.createFromAsset(getAssets(), path: "baloo2semibold.ttf");
        titulo.setTypeface(Titl);
        boton.setTypeface(Titl);
    }

        //metodo del boton que vuelve al menu principal
    public void btnHome(View v){
        Intent aPrincipal = new Intent( packageContext: this, Menu.class );
        startActivity(aPrincipal);
    }
}
```

Ilustración 6: Clase java perteneciente a la portada de la aplicación

```java
public class Menu extends AppCompatActivity {

    TextView title;
    ImageButton clima,registro,agregarRegistro,ejercicios;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_menu);

        title = findViewById(R.id.tituloPrincipal);
        clima = findViewById(R.id.btVerClima);
        registro = findViewById(R.id.btRegistrosGuardados);
        agregarRegistro = findViewById(R.id.btCrearRegistro);
        ejercicios = findViewById(R.id.btEjercicios);

        Typeface titulo = Typeface.createFromAsset(getAssets(), path: "baloo2semibold.ttf");
        title.setTypeface(titulo);
    }

    //Metodo para acceder a la sección de los registros
    public void aRegistros (View v){
        Intent aListadoRegistros = new Intent( packageContext: this, ListadoRegistros.class );
        startActivity(aListadoRegistros);
    }
```

Ilustración 7: Clase .java del menú principal (parte 1)

```java
    //Metodo para acceder a internet y comprobar el clima
    public void aClima(View v){
        Uri pagina_aemet=Uri.parse("http://www.aemet.es/es/eltiempo/prediccion/municipios");
        Intent ver_tiempo_zona=new Intent(Intent.ACTION_VIEW,pagina_aemet);
        if(ver_tiempo_zona.resolveActivity(getPackageManager())!=null){
            startActivity(ver_tiempo_zona);}
    }

    //Metodo para acceder a la lista de ejercicios
    public void aEjercicios(View v){
        Intent aEjercicios = new Intent( packageContext: this, ListadoEjercicios.class );
        startActivity(aEjercicios);
    }

    //Metodo para acceder a la creación de un nuevo registro
    public void aNuevoRegistro(View v){
        Intent aNewRegistro = new Intent( packageContext: this, NuevoRegistro.class);
        startActivity(aNewRegistro);
    }
}
```

Ilustración 8: Clase .java del menú principal (parte 2))

```java
public class ListadoEjercicios extends AppCompatActivity implements AdapterView.OnItemClickListener {

    TextView tituloEjercicios;
    ListView listaEjercicios;
    String[] zonas_Spinner= {"Gluteos","Cintura y Cadera","Piernas","Abdomen"};
    Spinner spbusqueda;
    Button bt_busqueda;
    ArrayAdapter<String> adapterSpinner, adapterLista;
    ArrayList<String> ListaDatos = new ArrayList<>();
    String eleccion = "(fallo)";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_listado_ejercicios);

        //Conexion con la interfaz
        conexionInterfaz();

        //Metodos de la clase
        fuenteTitulo();
        SpinerFuncional();
        ListaOperativa();

        //Recogida de datos del elemento seleccionado de la lista
        listaEjercicios.setOnItemClickListener(this);
    }
```

Ilustración 9:Clase .java del listado de Ejercicios (parte 1)

```java
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

        eleccion =listaEjercicios.getItemAtPosition(position).toString() ;
        String SQL = "SELECT * FROM TABLA_EJERCICIO WHERE nombre_ejercicio ='" + eleccion + "'";

        AdminSQLiteOpenHelper admin = new AdminSQLiteOpenHelper( context: this, factory: null);
        SQLiteDatabase BD = admin.getReadableDatabase();
        Cursor cursor = BD.rawQuery(SQL, selectionArgs: null);
        cursor.moveToFirst();
        String nEjercicio =cursor.getString( columnIndex: 0) ;
        String realizacionEjercicio = cursor.getString( columnIndex: 1);
        String material_usado = cursor.getString( columnIndex: 2);
        String zona_afectada = cursor.getString( columnIndex: 3);
        String tabla_ejercicios = cursor.getString( columnIndex: 4);

        cursor.close();
        BD.close();

        Intent aDetalleEjercicio = new Intent( packageContext: this, DetallesEjercicio.class);

        aDetalleEjercicio.putExtra( name: "nombre_ejercicio", nEjercicio);
        aDetalleEjercicio.putExtra( name: "realizacion",realizacionEjercicio);
        aDetalleEjercicio.putExtra( name: "material",material_usado);
        aDetalleEjercicio.putExtra( name: "zona",zona_afectada);
        aDetalleEjercicio.putExtra( name: "tabla_ejercicios",tabla_ejercicios);

        startActivity(aDetalleEjercicio);
}
```

Ilustración 10: Clase .java del listado de ejercicios (parte 2)

```java
//Metodo de conexion a la interfaz
private void conexionInterfaz(){
    listaEjercicios = findViewById(R.id.lista);
    spbusqueda = findViewById(R.id.buscador);
    tituloEjercicios = findViewById(R.id.lblRegistroNuevo);
    bt_busqueda = findViewById(R.id.bt_buscar_ejercicios);
}
```

Ilustración 11: Clase .java del listado de ejercicios (parte3)

```java
//Buscamos los ejercicios correspondientes de la base de datos
public void btBuscarEjercicio(View v) {

        //Primero dejamos vacio el arrayList<>
        ListaDatos.clear();

        //Conexion con la BD
        AdminSQLiteOpenHelper conn = new AdminSQLiteOpenHelper( context: this, factory: null);
        SQLiteDatabase DB = conn.getReadableDatabase();

        //Strings necesarios
        String categoria = spbusqueda.getSelectedItem().toString();//Aqui recoge el valor elegido en el spinner
        String elemento;

        //Sentencia SQL
        String SQL ="SELECT "+ContractBD.TBEjercicios.PK_nombre_ejercicio+" FROM "+ContractBD.TBEjercicios.Nombre_tabla_ejercicio+
                " WHERE "+ContractBD.TBEjercicios.zona+"='"+categoria+"'";

        //Cursor para las busquedas
        Cursor conexion = DB.rawQuery(SQL, selectionArgs: null);

        //Cargar elementos en la lista
        if (conexion.isBeforeFirst()){
            while(conexion.moveToNext()) {
                elemento = conexion.getString( columnIndex: 0);
                //Añadir elementos a la lista
                ListaDatos.add(elemento);
            }
```

Ilustración 12: Clase .java del listado de ejercicios (parte 4)

```java
            //Generacion de la lista de elementos
            ListaOperativa();

            //El cierre de la BD
            conexion.close();
            DB.close();

        }else {
            Toast.makeText( context: this, text: "Busqueda vacia. Prueba con otro campo", Toast.LENGTH_SHORT).show();

            //Cierres
            conexion.close();
            DB.close();
    }

}

//Asignacion de fuente al titulo
private void fuenteTitulo(){
    Typeface titulo = Typeface.createFromAsset(getAssets(), path: "baloo2semibold.ttf");
    tituloEjercicios.setTypeface(titulo);
    bt_busqueda.setTypeface(titulo);
}

//Establecimiento del Spinner
private void SpinerFuncional(){

    adapterSpinner = new ArrayAdapter<>( context: this, android.R.layout.simple_expandable_list_item_1,zonas_Spinner);
    spbusqueda.setAdapter(adapterSpinner);
}
```

Ilustración 13: Clase .java del listado de ejercicios (parte 5)

```
    //Construccion de la lista
    private void ListaOperativa(){
        adapterLista = new ArrayAdapter<>( context: this,android.R.layout.simple_expandable_list_item_1,ListaDatos);
        listaEjercicios.setAdapter(adapterLista);
    }

    //Boton para volver al menu principal
    public void btHome(View v){
        Intent aMenu = new Intent( packageContext: this, Menu.class );
        startActivity(aMenu);
    }

}
```

Ilustración 14: Clase .java del listado de ejercicios (parte 6)

```
public class DetallesEjercicio extends AppCompatActivity {

    TextView titulo,howTo,zona,materiales;
    String nombre_ejercicio,realizacion,material,zona_trabajada,tabla_ejercicios;
    Button botonLista;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_detalles_ejercicio);

        //Asociacion entre interfaz y variables
        titulo = findViewById(R.id.lblTituloNRegistro);
        howTo = findViewById(R.id.lblTextoHTDit);
        zona = findViewById(R.id.lblTextoZona);
        materiales = findViewById(R.id.lbltextoMaterial);
        botonLista = findViewById(R.id.btLista);

        //Recogida y muestreo de los datos
        recogervalores();
        mostrarValores();

        //Asignación de fuentes
        Typeface fuente = Typeface.createFromAsset(getAssets(), path: "baloo2semibold.ttf");
        titulo.setTypeface(fuente);
        howTo.setTypeface(fuente);
        zona.setTypeface(fuente);
        materiales.setTypeface(fuente);
        botonLista.setTypeface(fuente);
```

Ilustración 15: Clase .java de los detalles del ejercicio (parte 1)

```
        //Scroll para la descripcion de los ejercicios
        howTo.setMovementMethod(new ScrollingMovementMethod());

    }

    //Este metodo asigna los detalles del ejercicio seleccionado a sus posiciones en la interfaz.
    private void mostrarValores() {

        howTo.setText(realizacion);
        titulo.setText(nombre_ejercicio);
        materiales.setText(material);
        zona.setText(zona_trabajada);
    }

    //Este metodo recoge los detalles del ejercicio desde la pantalla anterior.
    private void recogervalores() {

        Bundle valores = getIntent().getExtras();
        nombre_ejercicio = valores.getString( key: "nombre_ejercicio");
        realizacion = valores.getString( key: "realizacion");
        material = valores.getString( key: "material");
        zona_trabajada = valores.getString( key: "zona");
        tabla_ejercicios = valores.getString( key: "tabla_ejercicios");
    }

    //Boton para volver al menu principal
    public void btHome(View v){
        Intent aMenu = new Intent( packageContext: this, Menu.class );
        startActivity(aMenu);
    }
```

Ilustración 16: Clase .java de los detalles del ejercicio (parte 2)

```
    //Permite volver a la lista de ejercicios
    public void btaLista(View v){
        Intent aEjercicios = new Intent( packageContext: this, ListadoEjercicios.class );
        startActivity(aEjercicios);
    }
}
```

Ilustración 17: Clase .java de los detalles del ejercicio (parte 3)

```java
public class ListadoRegistros extends AppCompatActivity implements AdapterView.OnItemClickListener {


    LinearLayout listaRegistros;
    ListView lista;
    ArrayAdapter adaptador;
    ArrayList<String> listaStrings = new ArrayList<>();
    String eleccion,date,time,content,SQL;
    TextView title;
    Cursor cursor;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_listado_registros);

        listaRegistros = findViewById(R.id.listadoRegistros);
        lista = findViewById(R.id.lista_registros);
        title = findViewById(R.id.lblTituloRegistros);

        //Asignacion de fuentes
        Typeface titulo = Typeface.createFromAsset(getAssets(),  path: "baloo2semibold.ttf");
        title.setTypeface(titulo);

        //Buscamos los registros guardados
        crearLista();
        lista.setOnItemClickListener(this);
    }
```

Ilustración 18: Clase .java del listado de registros (parte 1)

```java
//Metodo para volver al menu principal
public void btHome(View v) {
    Intent aMenu = new Intent( packageContext: this, Menu.class);
    startActivity(aMenu);
}
//Este metodo busca los registros existentes para poder crear la lista
private void buscarRegistros() {
    listaStrings.clear();
    AdminSQLiteOpenHelper conn = new AdminSQLiteOpenHelper( context: this,  factory: null);
    SQLiteDatabase DB = conn.getReadableDatabase();
    String SQL = "SELECT " + ContractBD.TBRegistro.fecha_registro + "," + ContractBD.TBRegistro.hora_registro + " FROM "
            + ContractBD.TBRegistro.Nombre_tabla_registros;
    String agregado;
    Cursor cursor = DB.rawQuery(SQL,  selectionArgs: null);
    if(cursor.getCount()>=2) {
        if (cursor.moveToFirst()) {
            while (cursor.moveToNext()) {
                agregado = "Fecha: " + cursor.getString( columnIndex: 0) + ". Hora: " + cursor.getString( columnIndex: 1);
                listaStrings.add(agregado);
            }
            cursor.close();
            DB.close();
            Toast.makeText( context: this,  text: "Mostrando Lista. Encontrados " + listaStrings.size() + " elementos", Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText( context: this,  text: "No hay registros que mostrar", Toast.LENGTH_SHORT).show();
            cursor.close();
            DB.close();
        }
```

Ilustración 19: Clase .java del listado de registros (parte 2)

```
    }else {
        Toast.makeText( context: this,  text: "No hay elementos que mostrar", Toast.LENGTH_SHORT).show();
    }
}

//Metodo que crea la lista
private void crearLista() {
    buscarRegistros();
    try {
        if (!listaStrings.isEmpty()) {
            adaptador = new ArrayAdapter( context: this, android.R.layout.simple_expandable_list_item_1, listaStrings);
            lista.setAdapter(adaptador);
        }
    } catch (Exception e) {
        Toast.makeText( context: this,  text: "No se ha podido crear la lista", Toast.LENGTH_SHORT).show();
        e.printStackTrace();
    }
}
```

Ilustración 20: Clase .java del listado de registros (parte 3)

```
@Override
//En este metodo se nos permite elegir un elemento de la lista y ver sus detalles
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        //Recogemos el registro que vamos a buscar:
         eleccion = lista.getItemAtPosition(position).toString();

       String[] elementos;
    elementos = eleccion.split( regex: "\\.");
    String[] dato_fecha = elementos[0].split( regex: ":");
       String[] dato_hora = elementos[1].split( regex: ":");

        date = dato_fecha[1];
        time = dato_hora[1];

        busqueda();
}
```

Ilustración 21: Clase .java del listado de registros (parte 4)

```java
    private void busqueda() {
        //Conexion con BD
        AdminSQLiteOpenHelper conn = new AdminSQLiteOpenHelper( context: this,  factory: null);
        SQLiteDatabase BD = conn.getReadableDatabase();

        //Preparacion del cursor
             SQL = "SELECT " + fecha_registro + " ,"+hora_registro+" ," + contenido +
             " FROM " + Nombre_tabla_registros +
             " WHERE " + fecha_registro + " = '" + date.trim() + "' AND "+hora_registro+" ='"+time.trim()+"'" ;

        cursor = BD.rawQuery(SQL,  selectionArgs: null);
        //Busqueda en la BD
        //Hay registro
        if (cursor.moveToFirst()) {
            content = cursor.getString( columnIndex: 2);

            Intent datosRegistro = new Intent( packageContext: this, DetallesRegistro.class);
            datosRegistro.putExtra( name: "fecha", date);
            datosRegistro.putExtra( name: "hora", time);
            datosRegistro.putExtra( name: "contenido", content);

            cursor.close();
            BD.close();
            startActivity(datosRegistro);

        } else {
            BD.close();
        }
    }
}
```

Ilustración 22: Clase .java del listado de registros (parte 5)

```java
public class DetallesRegistro extends AppCompatActivity {

    TextView lblTituloRegistro, lblFecha, lblFechaRegistro, lblhora, lblHoraRegistrada,lblTextoContenido, lblTextoRegistrado;
    Button btnVolveraLista;
    String fecha, hora, contenido;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_detalles_registro);

        //Conexion interfaz-clase
        lblTituloRegistro = findViewById(R.id.lblTituloDetalleRegistro);
        lblFecha = findViewById(R.id.lblFecha);
        lblFechaRegistro = findViewById(R.id.lblFechaUltimaModificacion);
        lblhora = findViewById(R.id.lblHora);
        lblHoraRegistrada = findViewById(R.id.lblHoraUltimaModificacion);
        lblTextoContenido = findViewById(R.id.lblContentidoRegistro);
        lblTextoRegistrado = findViewById(R.id.lblDatosRegistrados);
        btnVolveraLista = findViewById(R.id.btVolver);
```

Ilustración 23: Clase .java del detalle del registro (parte 1)

```
        //Asignación de fuente
        Typeface cambio_fuente = Typeface.createFromAsset(getAssets(), path: "baloo2semibold.ttf");
        lblTituloRegistro.setTypeface(cambio_fuente);
        lblFecha.setTypeface(cambio_fuente);
        lblFechaRegistro.setTypeface(cambio_fuente);
        lblhora.setTypeface(cambio_fuente);
        lblHoraRegistrada.setTypeface(cambio_fuente);
        lblTextoContenido.setTypeface(cambio_fuente);
        lblTextoRegistrado.setTypeface(cambio_fuente);
        btnVolveraLista.setTypeface(cambio_fuente);

        recogidaDatos();
        asignacion_datos();
    }

    //Metodo que asigna los datos del elemento seleccionado en la pantalla anterior
    private void asignacion_datos() {
        lblFechaRegistro.setText(fecha);
        lblHoraRegistrada.setText(hora);
        lblTextoRegistrado.setText(contenido);
    }

    //Metodo que recoge los datos provenientes de la seleccion de la pantalla anterior
    private void recogidaDatos() {
        Bundle parametros = this.getIntent().getExtras();
        fecha=parametros.getString( key: "fecha");
        hora = parametros.getString( key: "hora");
        contenido = parametros.getString( key: "contenido");
    }
```

Ilustración 24: Clase .java de los detalles del registro (parte 2)

```
    //Metodo que permite volver al menu principal
    public void btHome (View v){
            Intent amenu = new Intent( packageContext: this, Menu.class );
            startActivity(amenu);
    }

    //Metodo que nos permite volver a la lista de Registros
    public void aLista (View v){
        Intent alista = new Intent ( packageContext: this, ListadoRegistros.class);
        startActivity(alista);
    }
}
```

Ilustración 25: Clase .java de los detalles del registro (parte 3)

```java
public class NuevoRegistro extends AppCompatActivity {

    TextView title, contenidoFecha, hora, textoRegistro, textoFecha,textoHora;
    EditText textoRegistrar;
    Button btnGuardar;
    int d,m,a,ho,min;
    String date, time;
    String SQL;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_nuevo_registro);

        //Conexion entre clase y elementos de interfaz
        title = findViewById(R.id.lblRegistroNuevo);
        textoFecha = findViewById(R.id.lblTextoFecha);
        contenidoFecha = findViewById(R.id.lblFecha);
        textoHora = findViewById(R.id.lblTextoHora);
        hora = findViewById(R.id.lblHora);
        textoRegistro = findViewById(R.id.lblTextoRegistro);
        textoRegistrar = findViewById(R.id.editTextoRegistro);
        btnGuardar = findViewById(R.id.btGuardar);

        //Metodos usados
        asignarFuentes();
        registrarFechayHora();
        busqueda();
    }
```

Ilustración 26: Clase .java de la pantalla de agregación de registros (parte 1)

```java
private void busqueda(){
    //Conexion con BD
    AdminSQLiteOpenHelper conn = new AdminSQLiteOpenHelper( context: this,  factory: null);
    SQLiteDatabase BD = conn.getReadableDatabase();

    //Preparacion del cursor
    SQL = "SELECT "+ fecha_registro+","+contenido+" FROM "+Nombre_tabla_registros+" WHERE " +fecha_registro+" = '"+date+"'";
    Cursor cursor = BD.rawQuery(SQL,  selectionArgs: null);
    //Busqueda en la BD
    //Hay registro
    if (cursor.moveToFirst()){
        BD.close();
        cursor.close();

        //No hay registro
    }else{
        cursor.close();
        BD.close();
    }
}
```

Ilustración 27: Clase .java de la pantalla de agregación de registros (parte 2)

```
// realiza la insercion
public void insertarResultados(View v) { insertarElemento(); }

//Asignacion de fuentes de la clase
//Asignación de la fuente
private void asignarFuentes() {
    Typeface font = Typeface.createFromAsset(getAssets(),  path: "baloo2semibold.ttf");
    textoFecha.setTypeface(font);
    contenidoFecha.setTypeface(font);
    textoHora.setTypeface(font);
    hora.setTypeface(font);
    textoRegistro.setTypeface(font);
    textoRegistrar.setTypeface(font);
    title.setTypeface(font);
    btnGuardar.setTypeface(font);
}
```

Ilustración 28: Clase .java de la pantalla de agregación de registros (parte 3)

```
//Registro de la fecha y la hora para el registro
private void registrarFechayHora(){
    Calendar momento_dia = Calendar.getInstance();
    d = momento_dia.get(Calendar.DAY_OF_MONTH);
    m = momento_dia.get (Calendar.MONTH)+1;
    a = momento_dia.get(Calendar.YEAR);
    //Asignacion de la fecha
    if (d<10 & m<10){date = "0"+d+" / 0"+m+" / "+a;}
    else if (d<10){date = "0"+d+" / "+m+" / "+a;}
    else if(m<10){date = ""+d+" / 0"+m+" / "+a;}
    else{date = d+" / "+m+" / "+a;}
    contenidoFecha.setText(date);

    ho = momento_dia.get(Calendar.HOUR_OF_DAY);
    min = momento_dia.get (Calendar.MINUTE);

    //Asignacion de la hora
    if (ho<10 & min<10){time = "0"+ho+"y0"+min;}
    else if (ho<10){time = "0"+ho+"y"+min;}
    else if (min<10){time = ho+"y0"+min;}
    else{time = ho+"y"+min;}
    hora.setText(time);
}
//Boton de acceso al menu principal
public void btnHome(View v){
    Intent aMenu = new Intent( packageContext: this, Menu.class );
    startActivity(aMenu);
}
```

Ilustración 29: Clase .java de la pantalla de agregación de registros (parte 4)

```
    private void insertarElemento(){
        AdminSQLiteOpenHelper admin = new AdminSQLiteOpenHelper( context: this,  factory: null);
        SQLiteDatabase bd = admin.getWritableDatabase();

        String hora_registrar = hora.getText().toString();
        String texto_a_insertar = textoRegistrar.getText().toString();

        ContentValues valores_a_insertar = new ContentValues();
        valores_a_insertar.put(fecha_registro, date);
        valores_a_insertar.put(hora_registro, hora_registrar);
        valores_a_insertar.put(contenido, texto_a_insertar);

        bd.insert(Nombre_tabla_registros,  nullColumnHack: null, valores_a_insertar);

        bd.close();

        Intent aMenu = new Intent( packageContext: this, Menu.class );
        startActivity(aMenu);
    }
}
```

Ilustración 30: Clase .java de la pantalla de agregación de registros (parte 5)

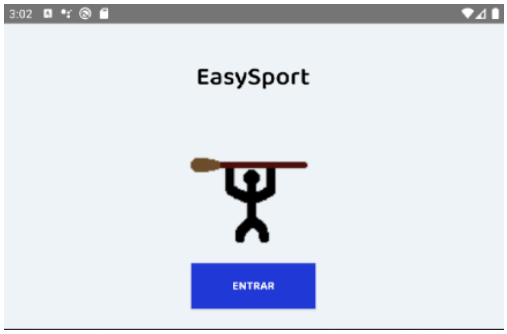
Respecto a las clases referentes a la base de datos, se encuentran capturadas en el apartado de la base de datos de la memoria

Manuales de la aplicación

Manual de Instalación de la aplicación

Para instalar la aplicación solo hay que descargarla y ejecutar el archivo .APK, por lo que no es necesario que haya un manual de instalación

Manual de usuario de la aplicación



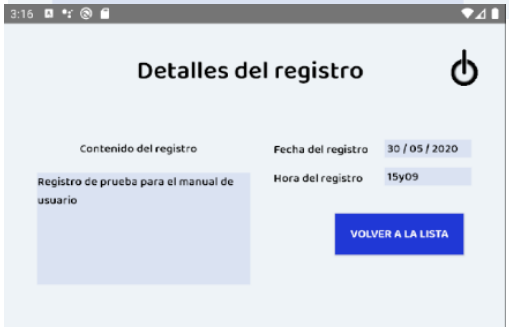
Para acceder al menú principal, pulsar el botón "Entrar" de la portada.

- Comprobar tiempo: Permite comprobar el clima
- Ejercicios: Permite acceder a los ejercicios almacenados en la base de datos
- Crear nuevo registro: Permite crear un nuevo registro con la información deseada
- Ver registros guardados: Muestra la lista de registros realizados por el usuario

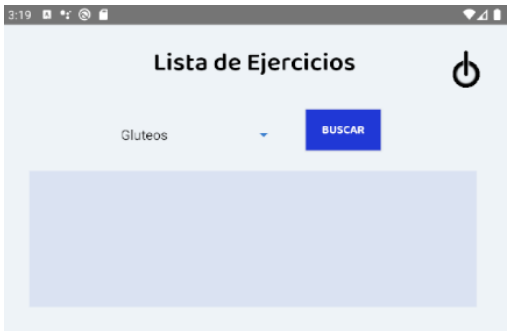
Para agregar un nuevo registro, simplemente escribir el texto deseado en la sección indicada, y luego pulsar en el botón “GUARDAR REGISTRO”. Si queremos volver al menú principal, solo hay que pulsar el icono de la esquina superior izquierda

Para seleccionar un registro de los que se hayan creado, solo hay que elegirlo de la lista.

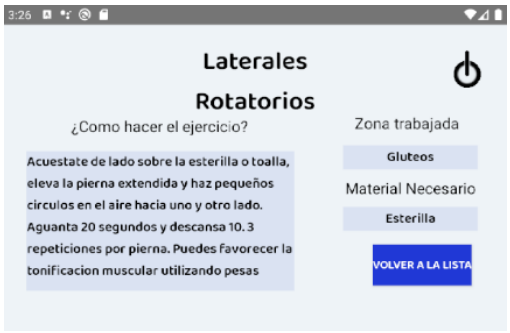
También podemos volver al menú principal con el icono de la esquina superior izquierda



aquí se muestran los detalles del registro seleccionado. Desde aquí se puede volver a la lista con el botón “VOLVER A LA LISTA”, o al menú con el icono de la esquina superior izquierda. Para volver al menú, seleccionamos el icono de la esquina superior izquierda



Aquí podremos elegir el ejercicio que deseemos realizar. Para ello, se elige la zona que vamos a trabajar del desplegable y luego pulsaremos el botón buscar. Esto nos mostrará la lista de ejercicios almacenados que trabajan dicha zona



En esta pantalla se muestra lo necesario para hacer el ejercicio. Podemos volver al menú principal mediante el icono de la esquina superior izquierda, o a la lista con el botón “VOLVER A LA LISTA”

Conclusiones

Durante la producción y el desarrollo de la aplicación EasySport he podido conocer en mis carnes la metodología de trabajo necesaria para este trabajo, desde las primeras fases de la creación de la aplicación hasta su finalización, y las metodologías y requerimientos referidos a la profesión, tanto a nivel técnico (conocimientos, imaginación en la fase creativa,

Bibliografía

Costes

- Salario desarrollador java: https://www.indeed.es/salaries/desarrollador-java-Salaries
- Equipos para la instalación https://www.pccomponentes.com/asus-vivobook-d509da-ej098-amd-ryzen-7-3700u-8gb-512gb-ssd-156
- Sistema operativo para los equipos: https://www.pccomponentes.com/microsoft-windows-10-pro-64bits-oem-version-pt
- Microsoft Office: https://www.pccomponentes.com/microsoft-office-hogar-y-empresas-2019-1-pc-mac
- Coste de la licencia para publicar aplicaciones en Google Play: https://cincodias.elpais.com/cincodias/2015/02/01/lifestyle/1422792260_243066.html

Beneficios del deporte

https://www.webconsultas.com/ejercicio-y-deporte/vida-activa/beneficios-del-ejercicio-fisico-869

Ejercicios

- Rutina 1: https://www.proximaati.com/bienestar/ejercicios/ejercicios-en-casa?gclid=CjwKCAjw4871BRAjEiwAbxXi2-l8V-QiviUduHx-DDGHbqyw9zRJcEme68eyeaiYyFRvPwenc-pqvBoCvB0QAvD_BwE

Diseño de diagramas

- Diagrama de casos de uso: https://online.visual-paradigm.com/

- Diagrama Entidad-Relación y diagrama de clases: https://www.lucidchart.com/

Referencias

Índice de ilustraciones

Índice de diagramas

Índice de tablas