SGA-2022

David Soriano Enguidanos
Desarrollo de Aplicaciones Móviles (DAM)
2º Curso
Periodo: 2021 / 2022
IES SERRA PERENXISA
Tutor: Jose Zomeño

Contenido

Agradecimientos

Me gustaría agradecer a Jose Zomeño como mi Tutor Individual en el proyecto, por su compromiso, también me gustaría agradecer Carlos Tarazona, como tutor del Curso y por ultimo a Sergi como profesor de SGE.

Resumen

El proyecto realizado es una aplicación móvil que soluciona un problema de movilidad para cualquier empresa que tenga la necesidad de crear una serie de registros, de los movimientos que se han realizado entre almacenes.

Mediante la interfaz que se ha diseñado, los usuarios que tengan las credenciales autentificadas, podrán acceder a la aplicación y tendrán la posibilidad de empezar a crear los movimientos, añadir los artículos que se van a mover y a que almacenes los van a llevar.

Versión: **Español**

Resum

El projecte és una aplicació de mòbil que soluciona un problema de mobilitat per quansevol empresa que tinga la necessitat de crear una sèrie de registres, dels movimiments que s'han realitzat entre magatzems.

Mitjançant la interfície que s'ha dissenyat, els usuaris que tinguen les credencials autenticades, podran accedir a l'aplicació i tindran la possibilitat de escomençar a crear els movimimentes, afegir els articles que es mouran i al fet que magatzems els van a portar.

Versió: **Valencià**

Abstract

This Project its a mobil application that solves a mobility problema for any company, that needs to créate a series of records of the movements that have been made between warehouses.

With the inteface that has been designed, the users who have authenticated credentials, will be able to Access the application and will have the posibility start to créate the movements, add the ítems than will be move and which warehouse they will be taken.

Version: **English**

1. Introducción

SGA-2022, es un proyecto que está centrado en solucionar un problema de movilidad que cualquier empresa puede tener a la hora de generar unos registros, ya que, en muchas ocasiones, muchas empresas siguen utilizando papel y boli registrar los movimientos que se realizan en un almacén, esta forma de tener registradas estas acciones está bien, pero puede ser muy caótica cuando ya tenemos muchísimos registros.

Por ello para solucionar la acumulación de papeles, y poder así tener unos registros ordenados, limpios e indexados, se utiliza la aplicación “SGA-2022”.

De esta manera solucionamos dos problemas, el problema de movilidad para crear cada registro de los movimientos y la disminución del consumo de papel.

2. Estado del Arte

2.1. BACKEND

2.1.1. Como plataformas de desarrollo:

1- Android Studio

Android Studio, se utilizó como el IDE principal para el desarrollo de la aplicación, se eligió Android Studio ya que era un IDE muy potente a la hora de desarrollar aplicaciones móviles Android, además de ser muy versátil para Android, Android Studio nos permite crear interfaces para los usuarios de una manera fácil, rápida y sencilla.
Además, también tiene un sistema de emulación muy rápido y con la posibilidad de conectar nuestro propio dispositivo Android.

2- Visual Studio 2019

Por otra parte, también se ha utilizado “VS 2019”, ya que ha sido el IDE que ha dado vida no a la aplicación, sino a la “RestApi” que se ha diseñado para albergar cada una de las funcionalidades de la App.

3- SQL SERVER

“SQL Server”, ha sido la plataforma utilizada para albergar la base de datos de nuestra aplicación, se ha utilizado “SQL SERVER”, ya que es un motor muy desarrollado por Microsoft y tiene una gran potencia de escalabilidad. “Microsoft SQL SERVER 2019”

2.1.2. Como lenguajes de desarrollo:

1- Kotlin

Como bien se ha mencionado anteriormente, se ha utilizado “Android Studio” como IDE de desarrollo principal para la aplicación.
Pues “Kotlin”, ha sido el lenguaje que se ha utilizado al 100 % en la aplicación.
Este lenguaje es relativamente nuevo (2016), es un lenguaje orientado a objetos, el cual fue desarrollado por en base a Java y otros lenguajes.

2- C#

De la misma forma que “Kotlin”, “C#” o C Sharp, ha sido uno de los lenguajes utilizados en todo el proyecto, este lenguaje la igual que “Kotlin”, también es orientado a objetos y fue desarrollado en base a Java y otros.
“C#” ha sido el lenguaje que se ha utilizado para crear el “RestApi” en su totalidad.

2.1.3. Control de Versiones:

1- GitHub

Mediante el IDE “Android Studio”, se ha hecho uso del plugin de “GitHub”, él cual ha permitido controlar los cambios que se ha ido realizando a la aplicación en el transcurso de su desarrollo.

2.1.4. Control del “RestAPI”:

1- Postman

Para controlar el funcionamiento de la “RestAPI”, se utilizó “Postman”, como herramienta, para comprobar el funcionamiento de los métodos creados en el “RestApi”.

2.2. FRONTEND

Como bien se ha mencionado anteriormente, se ha utilizado “Kotlin”, este lenguaje es muy versátil a la hora de crear interfaces, ya que dispone de muchas funciones pre diseñada para que a la hora de crear “pantallas”, el desarrollador pueda centrarse más en la parte lógica que en el propio diseño.

Aún así, para el desarrollo de la aplicación se han utilizado varias “dependencias” adicionales a las de base:

- RecyclerView
- Shimmer

Además de estas librerías hay más, pero ya no pertenecen a la interfaz, sino que pertenecen a la lógica de la aplicación, como:

- Coroutines
- OkHttp
- JSON

A la hora de realizar el diseño, me he basado en alguna aplicación, como *“WareHouse (Smartbox)”* está aplicación está disponible en la “AppStore”*,* la cual tiene una interfaz muy simple y limpia.

3. Estudio de Viabilidad

3.1. METODO DAFO

DEBILIDADES	- Precisa de desarrollo constante. - Mucho personal de desarrollo tanto en la app como en el Api.
AMENAZAS	- Mucha Competencia.
FORTALEZAS	- Producto ajustado a las necesidades de la empresa. - Con gran capacidad de distribuirse a diferentes empresas cambiando solo la base de datos.
OPORTUNIDADES	- Con el móvil el usuario dispone de gran movilidad. - El cliente se adapta a la forma de trabajo y pide mejora del producto.

3.2. ESTUDIO DE MERCADO

A la hora de vender el producto, se han realizado las siguientes preguntas, para saber de que manera impactaría la puesta en venta del producto.

¿Es necesario nuestro producto?

- Depende de la empresa, nuestro producto sirve para tener almacenados y ordenados todos los registros, de la misma forma que se hacen en papel, pero también permite a los usuarios finales depender de un solo dispositivo sin nada más.

¿Tiene hueco en el mercado?

- Por supuesto, ya que, si es un buen producto, y está bien trabajado, y desempeña la función que el cliente está buscando siempre va a haber hueco.

¿A qué público objetivo interesa?

- El producto esta diseñado para grandes empresas con muchos almacenes, pero también para pequeñas empresas que tengan un almacén y que muevan productos constantemente.

3.3. RESCURSOS NECESARIOS

Para trabajar con la aplicación, se necesitan varios recursos que son imprescindibles para trabajar con la misma.

Para ello se ha realizado un estudio de los recursos necesarios tanto de hardware como de software:

3.3.1. RECURSOS HARDWARE NECESARIOS

RECURSO	COSTE*
Ordenado para el desarrollo Lenovo Legión	Desde 500€
Móvil Android	Desde 170€
Servidor	Desde 500€

*Los precios son los más básicos dependiendo de la potencia que queramos se destinará más presupuesto, pero los precios son aproximados a los del desarrollo.

3.3.2. RECURSOS SOFTWARE NECESARIOS

RECURSO	COSTE
Android Studio	0€
Visual Studio 2019	0€
SQL Server	0€
Postman	0€
GitHub	0€

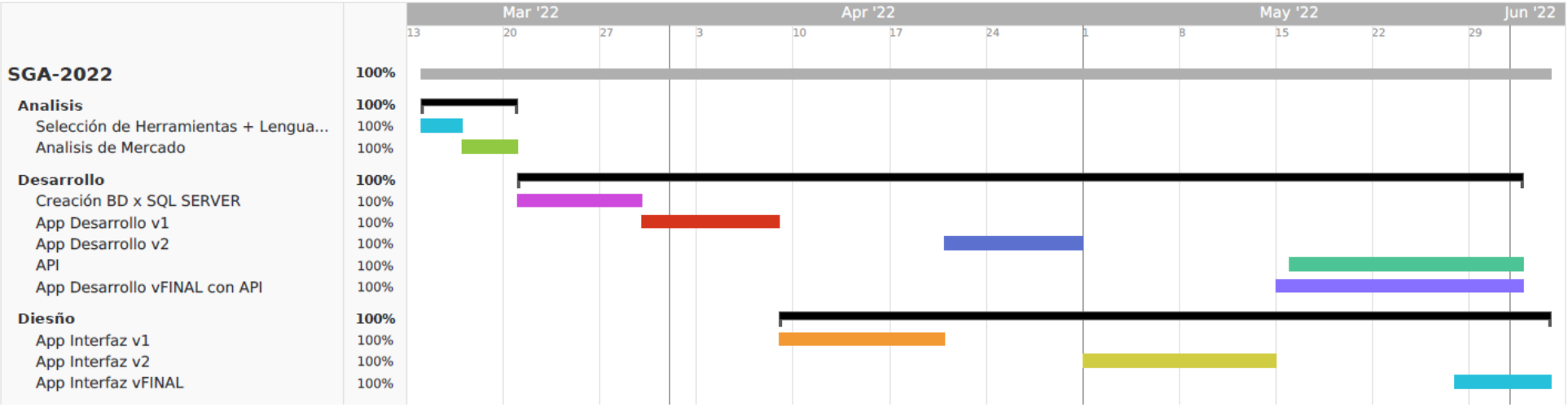
3.3.3. RECURSOS HUMANOS NECESARIOS

David Soriano Enguidanos(programador) Horas: ∞ Salario: ∞

3.4. VIABILIDAD TEMPORAL

Para realizar el proyecto se han empleado 3 meses desde marzo de 2022 hasta junio de 2022. En caso de que fuese una plantilla mayor, el programador podría distribuir los trabajos entre los demás programadores. Y el tiempo empleado se podría distribuir para además del trabajo principal de la aplicación, para otras funcionalidades adicionales.

3.5. PLANIFICACIÓN TEMPORAL



Para planificar el trabajo se ha utilizado “TeamGantt”, a las empresas nos permite crear mediante el diagrama de Gantt la estructura de como se ha distribuido el tiempo para crear el proyecto.

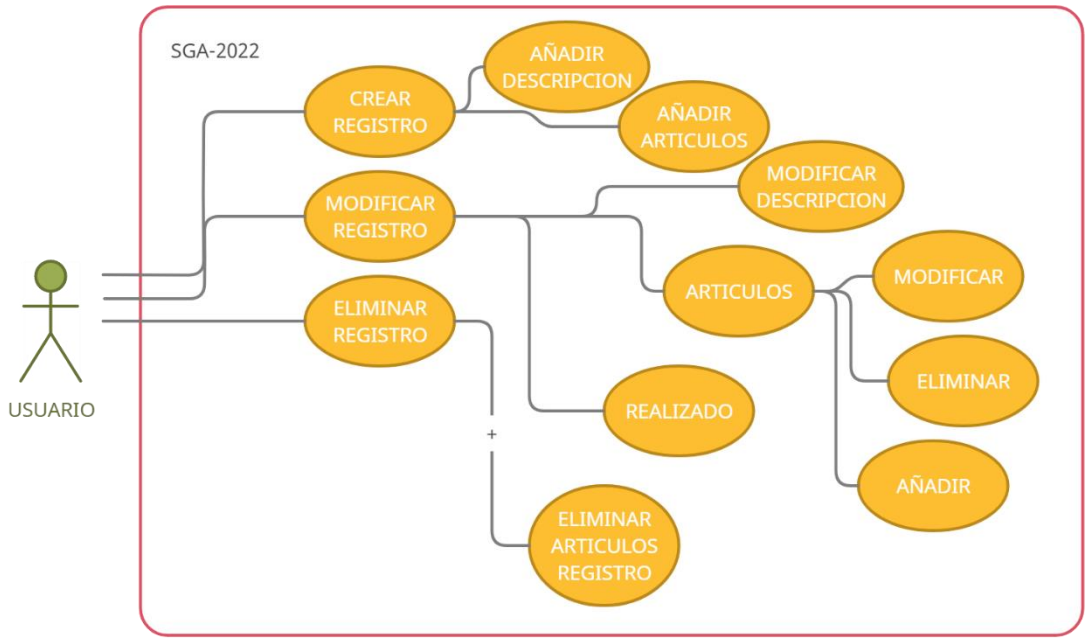
Dicho esto, la planificación se ha cumplido tal como se muestra en el esquema, exceptuando algunos días festivos locales.

4. Análisis de Requisitos

A la hora de presentar el proyecto al cliente se han de tener los siguientes requisitos en cuenta.

- Para el uso de la aplicación los usuarios deben estar dados de alta en la base de datos, esto se debe manejar desde el propio servidor.
- Los usuarios podrán crear, modificar e incluso eliminar los registros, todo esto se representará de cara a los usuarios mediante la interfaz intuitiva y amigable diseñada.
- En caso de que el cliente necesite una incorporación de alguna función que no requiera de mucho tiempo se realizará en la semana en que es comunicada y la posterior, en caso de que el programador esté disponible.
- También ha de tener en cuenta de que, si su empresa va a trabajar muchos usuarios y van a haber muchas peticiones al servidor, necesita disponer de un potente servidor.
- En la aplicación no se dispone de modo Administrador, ya que no está esta enfocado para ser administrado desde el móvil.

4.1. DIAGRAMA DE CASOS DE USO



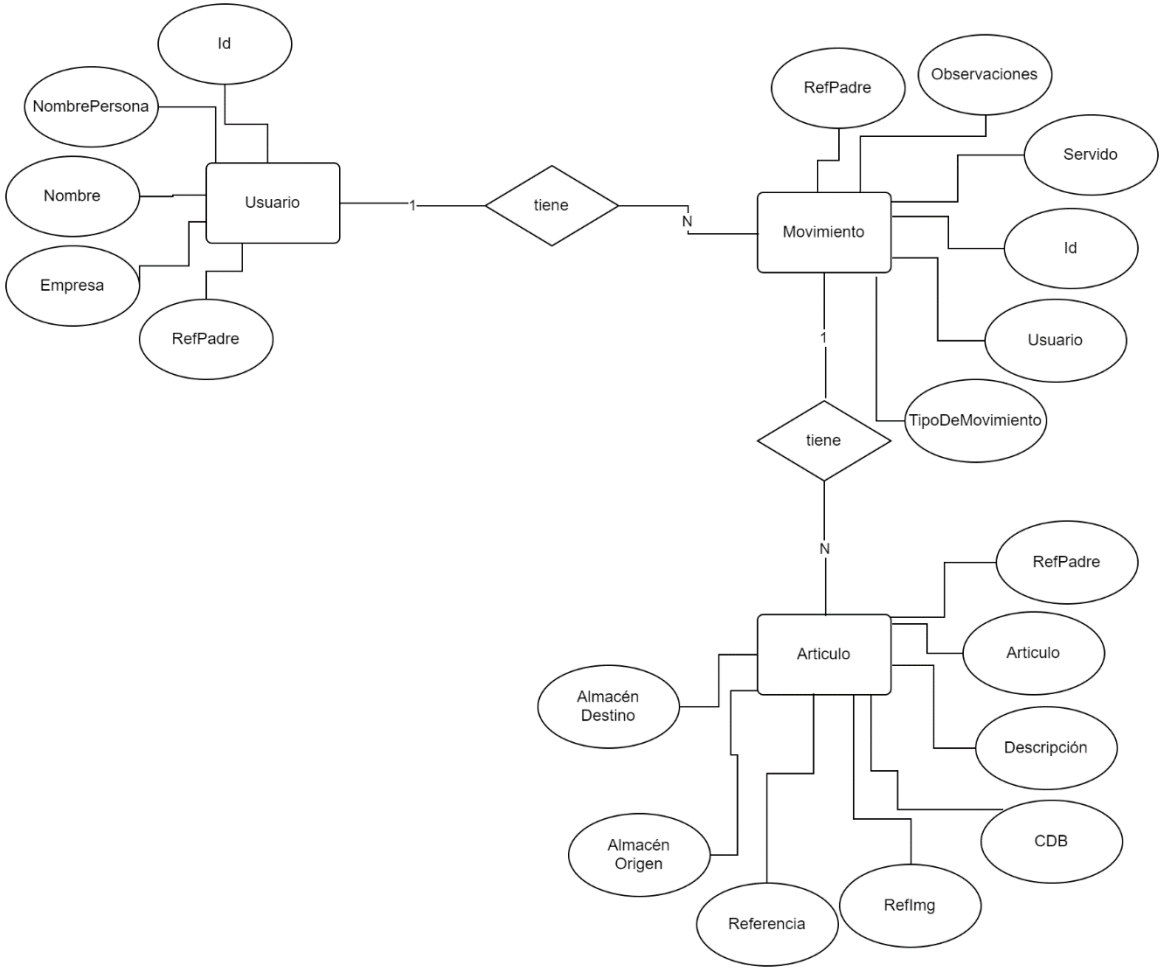
Como se ha comentado anteriormente, la aplicación no necesita de un modo Administrador, ya que todos los operarios/usuarios, tendrán todos los accesos, en el caso de que la empresa necesite un modo administrador, solo habría que configurarlo en la base de datos.

5. Diseño

5.1. BASE DE DATOS

Al principio del proyecto, se plantearon unas estructuras para las tablas, las cuales poco a poco fueron cambiando, ya que se iban adaptando a las necesidades que necesitaba el cliente.

5.1.1. DIAGRAMA ENTIDAD-RELACION

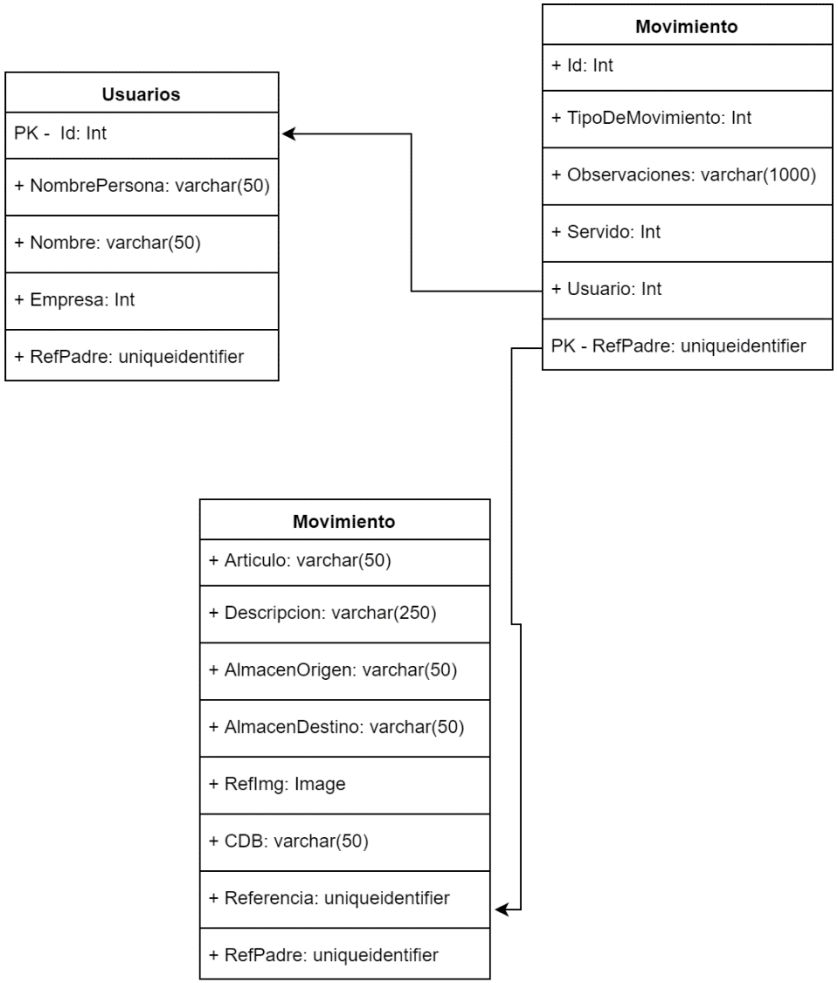


En este diagrama de entidad relacional, tenemos toda la estructura que se ha utilizado en el desarrollo, no obstante, para poder trabajar desde la aplicación, se ha hecho uso también de procedimientos almacenados y vistas, estos procedimientos almacenados, son llamados des la app pasando por la propia restapi, la cual almacena los valores pasados y este prc se ejecuta dentro de la base de datos.

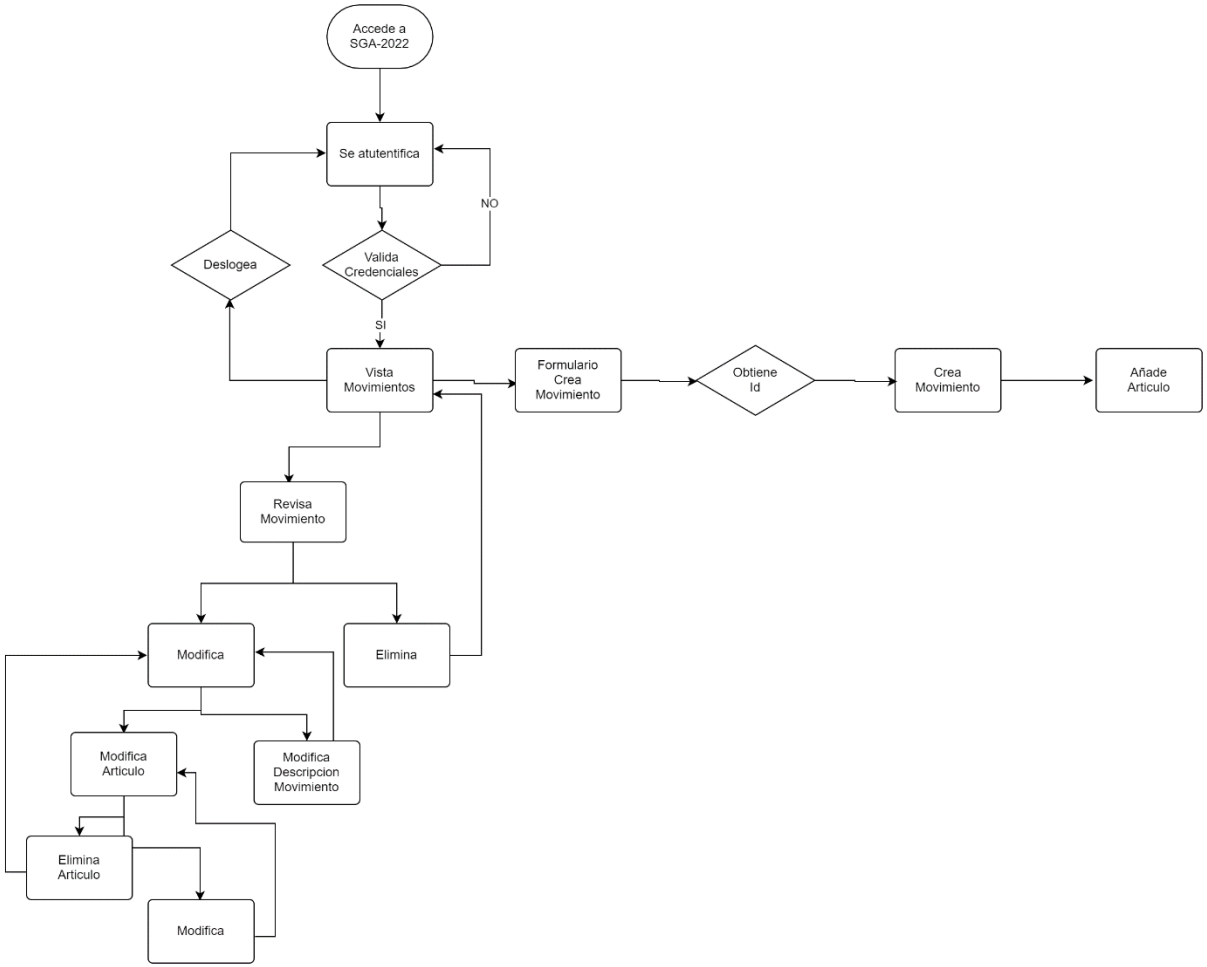
Como vemos en el diagrama, los usuarios pueden tener más de un movimiento, pero un movimiento no puede tener más de un usuario, así mismo los movimientos pueden tener muchos artículos, pero un artículo no puede tener más de un movimiento, pero esto no es del todo cierto.

Al crear un Movimiento, nosotros podemos añadir cualquier artículo, pero a cada articulo que añadamos el numero del articulo será el mismo pero la referencia del articulo cambiará.

5.1.2. DIAGRAMA DE CLASES



5.1.3. DIAGRAMA DE FLUJO

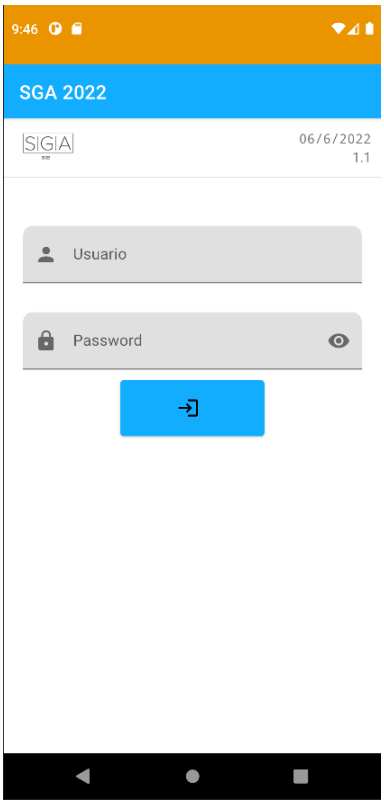


5.1.4. MOCKUPS

Esta es una de las partes en la cuales se ha dedicado más tiempo ya que es la que va interactuar con el usuario final, para ello se llevaron a cabo un total de 3 modificaciones a lo largo del desarrollo hasta llegar al actual.

En la aplicación se intentando minimizar los menús pequeños, ya que esta es una aplicación para personal de almacén y no pueden perder el tiempo en intentar darle a un pequeño botón, para ello se ha hecho algo medianamente grande y fácil de clicar.

Se ha creado una interfaz base para la aplicación con el nombre del proyecto y el nombre del usuario, en la parte superior, y en la parte inferior la versión de la aplicación y la fecha actual.



Pantalla de inicio de sesión, la cual crea una conexión con la base de datos, la aplicación recibe un token el cual permite al usuario acceder.

Esta pantalla muestra los registros/movimientos que están en la base de datos, mediante un reciclerview, además también tenemos un buscador y 2 botones uno para cerrar sesión y el otro para crear un nuevo movimiento.

Esta ventana es la vista del registro creado, como se puede observar, tenemos la información detallada del registro, además también nos muestra los artículos que tienes y toda la información de los mismos.
Como también se puede ver tiene un marcador, eso quiere decir que le registro no se puede modificar ni eliminar, ya que ha sido realizado, en caso de querer eliminarlo habría que hacerlo en la base de datos.

6. Codificación

6.1. TECNOLOGÍAS UTILIZADAS

Entramos en la fase de codificación del proyecto, la cual va a explicar el porqué de las tecnologías utilizadas etc.

Como bien se ha realizado anteriormente, la explicación de que es cada una en el apartado de “Diseño del Arte”, ahora se va a explicar la razón por lo que se han seleccionado.

Para realizar el proyecto se ha utilizado SQL Server como base de datos ya que es una herramienta muy desarrollada y versátil, con una interfaz fácil de usar y una programabilidad muy sencilla, es decir con un par de minutos nos será suficiente para familiarizarnos con ella.

A la hora de elegir el lenguaje de desarrollo se sabía que Kotlin era y es la mejor opción hasta el momento ya que a pesar de estar familiarizado con el lenguaje, ya que está basado en java en gran parte, además también ha sido utilizado por que el programador ya había hecho uso del mimo.

También se ha utilizado Android Studio como framework o IDE, ya que es muy potente y fácil de utilizar además también está más enfocado a Kotlin y Java.

6.2. DOCUMENTACIÓN INTERNA

Después de tener el proyecto documentado, se va a mostrar una de las funciones que realiza la aplicación cada vez que quiere realizar una petición a la base datos, para ello se procede a ver el código:

```
private var result: Boolean = false
fun loginApi(user:String,pass:String):Boolean{
    runBlocking {
        launch(Dispatchers.IO) {
            val client = OkHttpClient()
            val body = "grant_type=password&username=$user&password=$pass"
            val credentialsApi: RequestBody = RequestBody.create(MediaType.parse("text/plain"), body)
            val request = Request.Builder()
                .url("http://dsoriano.ddns.net/Login")
                .post(credentialsApi)
                .addHeader("Accept", "*/*")
                .addHeader("Cache-Control", "no-cache")
                .addHeader("Connection", "keep-alive")
                .build()
            try {
                val response = client.newCall(request).execute()
                if(response.isSuccessful){
                    val responseBody = response.body()!!.string().trimIndent()
                    val jsonObject = JSONTokener(responseBody).nextValue() as JSONObject
                    BasicData.token_access = jsonObject.getString("access_token")
                    result = true
                }
            }catch (e: Exception){
                result = false
                e.printStackTrace()
            }
        }
    }
```

```
    return if(result){
        result = false
        true
    }else{
        result = false
        false
    }

}
```

Como podemos observar, este trozo de código realiza una petición Http, esta petición se realiza al dominio.

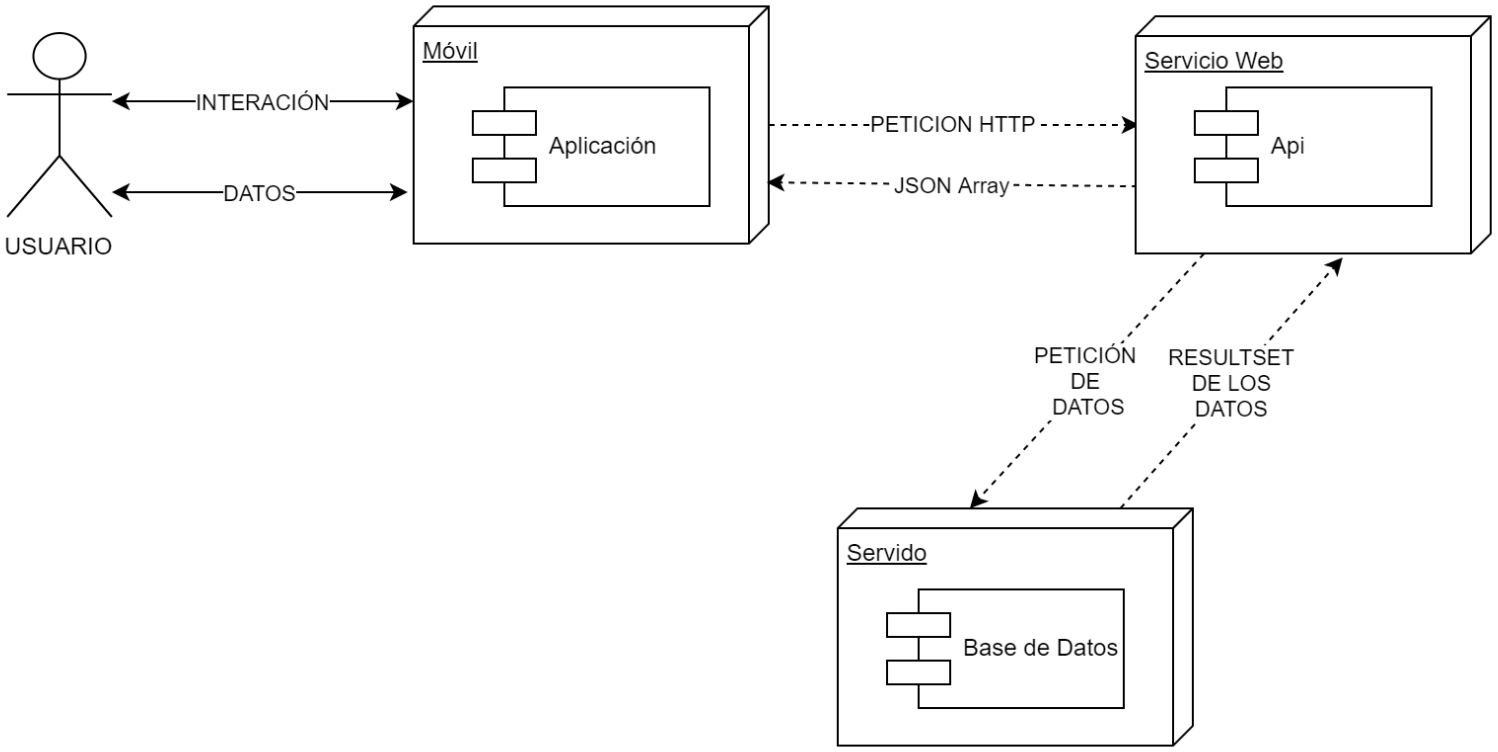
A esta petición debemos pasarle tanto el usuario como la contraseña, sin cifrar, en esta versión.

Si la conexión es fructífera, nos devolverá un token que almacenaremos en una clase objeto, donde se almacenan otras variables que se usan a lo largo de toda la aplicación.

7. Despliegue

7.1. DIAGRAMA DE DESPLIEGUE

A continuación, se muestra el manual de despliegue del proyecto:



Como podemos ver en el Diagrama de despliegue superior, el usuario realiza una interacción con el móvil el cual transforma dicha acción en una petición Http, la cual pasa por el servicio API, la cual realiza una sentencia select o ejecuta un procedimiento almacenado.

Una vez la API manda esa sentencia sobre la base de datos, esta, la realiza y dependiendo de la acción que haya hecho, devolverá información o simplemente que la ejecución se ha completado, poniéndonos en el lugar de que ha devuelto una información, esta se devuelve a la API, la cual trata la información formando de los datos obtenidos un JSON Array el cual es devuelto a la aplicación y esta es capaz de extraer la información del JSON y adaptarla a la interfaz del usuario, y por último el usuario puede leer la información.

8. Herramientas de apoyo

Para poder comprobar que todas las funciones dentro de la app funcionaban se ha utilizado la función debug de Android Studio, la cual funciona muy bien y te muestra todo pasa a paso y super detallado.

También para comprobar las peticiones al API, he utilizado como he mencionado anteriormente, Postman una herramienta diseñada para realizar peticiones http/s a direcciones web, además he combinado Postman con el debug de Visual Studio 2019, y este ha sido muy útil ya que podía ver qué información se trataba y como se convertía a JSON Array.

Además, también se ha hecho uso del transpilador de Android Studio ya que había algunas funciones que se han obtenido de internet y como estaban en java se ha hecho uso del transpilador.

Como también se comentó, se utilizó GitHub para controlar las versiones de la aplicación, esta herramienta se empezó a usar a mediados del desarrollo de la aplicación, se ha de argumentar de que, con esta herramienta, se puede subir a la nube el proyecto y bajar la versión que quieras desde otro ordenador.

9. Herramientas de apoyo

Para comprobar que la aplicación final y no tan final estuviese en perfectas condiciones en rendimiento, se realizaron múltiples pruebas las cuales se llevaron a cabo en diferentes móviles para comprobar el rendimiento en diferentes modelos tanto físicamente como en el propio emulador de Android Studio, es cierto que en el emulador depende más del hardware del equipo que se esté utilizando.

Al realizar estas pruebas se empezó a utilizar más las funciones asíncronas o async para que cargase en un hilo secundario los datos mientras que la principal carga la interfaz de esta manera todo carga perfecto y de una forma muy fluida.

10. Conclusiones

10.1. CONCLUSIONES SOBRE EL TRABAJO REALIZADO

El presente proyecto se puede categorizar en desarrollo de movilidad empresarial, ya que muchas empresas ya empiezan a utilizar dispositivos móviles, porqué realmente estos, pueden realizar las mismas funciones que un ordenador, por ello se ha creado esta aplicación ya que se cree que utilizar estos dispositivos ayudaría a cualquier empresa ya sea grande o pequeña a ahorrar dinero en comprar nuevo ordenadores y material de oficina.

Ya que en muchas ocasiones las empresas utilizan métodos más tradicionales para guardar toda su información.

En este proyecto no se han realizado ningún desarrollo “de empresa” ya que este proyecto ha sido diseñado y desarrollado en plazo de 3 meses aproximadamente por una sola persona.

También respecto a las fases que ha tenido el proyecto, el programador ha ido creciendo por que no contento con el diseño que tenía en mente se adentró en el mundo de la RESTAPI, el cual no tenía ni idea. ¿Qué por qué?

Bueno, era poco profesional crear una aplicación que se iba a realizar peticiones mediante TCP ip, con un driver jdts dentro de la aplicación.

Y por eso se decidió crear la API para tener una mayor seguridad a la hora de conectar con la base de datos.

10.2. CONCLUSIONES PERSONALES

En conclusión, este proyecto realizado, ha sido duro, pero al verlo finalizado y listo, me llena de alegría ver que he hecho un proyecto, el cual en mi criterio creo que se puede vender, si es cierto que faltan cosas por pulir y además también se pueden desarrollar nuevas funcionalidades que se podrían hacer en un futuro.

Por puesto, también me ha servido como para gustarme más el mundo de la programación se que no soy el mejor, que soy un desastre, pero he intentado centrarme en este proyecto durante los 3 meses y lo he conseguido cumpliendo mis objetivos tanto personales de realizar una aplicación completa para Android como para entregar el proyecto.

10.3. POSIBLES AMPLIACIONES Y MEJORAS

Como he comentado, se pueden crear nuevas mejoras a la aplicación como, por ejemplo:

- Nuevas funcionalidades que puedan ayudar a las empresas, como por ejemplo añadir el stock de cada producto.

- La creación de usuarios desde la pantalla de inicio.

- La opción de crear un modo Administrador.

- También se podría otra gestión como la de dar de alta clientes en la empresa

- Un realizador de pedidos, que le operario le notifique que tiene un nuevo pedido por hacer y que con un código QR o con un código de barras, el operario pueda ir añadiendo al pedido los productos que ya están en preparación.

- Algo muy importante sería controlar cuantos dispositivos quiere la empresa y que, con unas series de Licencias, que se cobrarían mensualmente o anualmente, el cliente tuviese además una asistencia completa de la aplicación.

- Otra mejora sería teniendo el feedback de un “comprador”, saber si el diseño de la interfaz es el adecuado o debería cambiar, ya que incluso a mí no terminad del todo.

10.4. PROBLEMAS ENCONTADOS

La realización de SGA-2022, ha sido dura ya que la falta de conocimientos a la hora de realizar una herramienta laboral y crear una API sin tener ningún conocimiento, ha sido difícil, pero no imposible.

11. Bibliografía

11.1. LIBROS, ARTICULOS Y APUNTES

Para el desarrollo del proyecto los apuntes empleados han sido los que he recibido durante el desarrollo del ciclo formativo.

También he visto algunos proyectos que se han realizado durante el desarrollo del ciclo formativo.

11.1. DIRECCIONES WEB

A continuación, se van a detallar las direcciones web que se han visitado a lo largo del desarrollo del proyecto.

https://www.youtube.com/c/AristiDevs - Canal de YouTube de un programador de Kotlin.

https://cursokotlin.com/ - Pagina web donde existen múltiples tutoriales, esta página pertenece al Youtuber del anterior link.

https://github.com/sdram58/apuntesPMDM - Repositorio de GitHub donde he consultado varios documentos de interés.

https://square.github.io/okhttp/ - Pagina web donde se encuentran tanto los métodos como la documentación para crear la conexión en este caso con el REST API.

https://mvnrepository.com/ - Pagina web donde he encontrado las versiones de algunas dependencias que he utilizado.

https://developer.android.com/kotlin/coroutines?gclid=CjwKCAjwy_aUBhACEiwA2IHHQNJlHPPQUSk5EGXuuW53gNeL4Sp-TKb55oIyUTVBtmHGQE2Vdk8avhoCiVQQAvD_BwE&gclsrc=aw.ds – Este enlace es de la propia página, la cual he consultado muchas veces hasta que me he quedado como se hace.

https://docs.microsoft.com/es-es/sql/relational-databases/database-engine-tutorials?view=sql-server-ver16 – Tutorial de como montar una base de datos en SQL SERVER.

https://docs.microsoft.com/es-es/aspnet/core/tutorials/first-web-api?view=aspnetcore-6.0&tabs=visual-studio – Como he creado mi "RestAPI" con ASP.NET Core.

https://programacion.net/articulo/acceso_directo_a_sql_server_desde_android_1155 - Pagina web de como hice la primera conexión a la base de datos cuando todavía no tenía el “RestAPI” creado.

https://www.youtube.com/watch?v=fFbkptOUunk&t=330s&ab_channel=DevExperto-AntonioLeiva – Video de YouTube de Corrutinas

https://www.youtube.com/watch?v=7ZtJLMFIk0s&ab_channel=Programaci%C3%B3nAndroidbyAristiDevs – Video de YouTube de “Shimmer” una librería que genera un efecto.