UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

MÁSTER UNIVERSITARIO EN INGENIERÍA DEL SOFTWARE PARA LA WEB

Trabajo Fin de Máster

ANÁLISIS DEL LENGUAJE KOTLIN MEDIANTE EL DESARROLLO DE UNA APLICACIÓN WEB Y MÓVIL

Santiago Paúl Monge López

2021

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

MÁSTER UNIVERSITARIO EN INGENIERÍA DEL SOFTWARE PARA LA WEB

Trabajo Fin de Máster

Análisis del lenguaje Kotlin mediante el desarrollo de una aplicación web y móvil

Autor: Santiago Paúl Monge López

Tutor: José María Gutiérrez Martínez

TRIBUNAL:

Presidente:

Vocal 1º:

Vocal 2º:

FECHA:

Dedicado a Ricardo Alberto Monge López.

RESUMEN

Con el progreso del desarrollo de software nace la necesidad de tener un lenguaje de programación sencillo de usar y fácil de aprender, por esta razón JetBrains crea Kotlin, para obtener un lenguaje en el que los desarrolladores se sientan cómodos al usar, sin cumplir con lógicas de programación extensas y además pueda ser utilizado para crear programas multiplataforma.

El propósito de este proyecto es comprobar los beneficios y características que JetBrains y los programadores que lo utilizan dicen sobre él. Para cumplir con el objetivo del proyecto, se realiza una investigación de su historia, versiones, aplicaciones, compatibilidades y sintaxis, un estudio y aprendizaje del lenguaje y el análisis del desarrollo de una aplicación web y otra móvil para Android, realizadas solamente con Kotlin y librerías que ayudan con las conexiones a la base de datos y el web service.

Como resultado, se comprueba que Kotlin es un excelente lenguaje de programación y cumple con las características que mencionan las personas que lo utilizan, además de ser moderno, competente y multiplataforma, su aprendizaje no demanda mucho tiempo, por lo tanto, se recomienda utilizar en cualquier proyecto.

Palabras clave: Desarrollo web, desarrollo para Android, Kotlin

ABSTRACT

With the progress of software development comes the need for a user-friendly, easy to learn programming language, for this reason JetBrains creates Kotlin to obtain a comfortable language when the developers use it, without keep to extensive programming logic, furthermore, it can be used to develop cross-platform software.

The purpose of this project is to prove the benefits and features that JetBrains and the programmers say about it. In order to achieve with the project objective, an investigation takes place, about its history, versions, applications, compatibility and syntax, moreover, a study and learning of the language, finally, an analysis about the development of a web and Android application, they are developed only with Kotlin and libraries which help with connections to the database and the web service.

As a result, Kotlin is proved to be an excellent programming language and keeps with the features mentioned by the developers, furthermore, to be modern, competent and cross-platform, its learning does not demand much time, therefore Kotlin is recommended to use in any project.

Key words: Web development, Android development, Kotlin

ÍNDICE

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

La existencia de muchos lenguajes de programación ha provocado que el desarrollador de software deba aprender a usar varios de ellos, y adaptarse según la necesidad que requiera; a partir de este dilema se buscó un lenguaje de programación que sea moderno, sencillo de usar y que su aprendizaje no sea complicado.

Kotlin se encuentra dentro de estos parámetros, además de ser multiplataforma, su potencial es muy amplio y su uso empieza a crecer entre la comunidad de desarrolladores; por esta razón se eligió este lenguaje para conocer, profundizar y aprender más sobre él.

Con esta investigación se puede conocer más sobre las principales características y beneficios al utilizar Kotlin; de esta manera se observa si es un lenguaje que ayuda a cubrir los requerimientos del desarrollador, principalmente si vale la pena aprender sobre él y profundizar toda su capacidad que ofrece.

Ahora bien, como todo lenguaje de programación se debe conocer porque fue creado, en otras palabras, se realizó un breve estudio sobre la historia de Kotlin, además se aprendió los diferentes usos que se puede aplicar con este lenguaje, se descubrió que puede ser desarrollado para cualquier fin y que un programa puede ser escrito en menos líneas de código en comparación con otros lenguajes.

Por consiguiente, para demostrar que estas y otras características de Kotlin son ciertas, se desarrolló dos aplicaciones, estas dos están enfocadas al registro y visualización del horario laboral de los empleados de cualquier empresa. El primero es un sistema web, el cual cuenta con dos roles, la parte administrativa y la parte del empleado, en la parte administrativa se puede realizar las operaciones de creación, actualización, visualización y eliminación de los empleados, también se puede observar el historial de registros de cada uno y editarlos; para la parte del empleado solo se puede registrar su turno laboral y visualizar su historial de registros. El segundo es una aplicación móvil para dispositivos Android con el que solo el empleado podrá ingresar a este, y realizará las mismas actividades que puede hacer en el sistema web.

Con estas dos aplicaciones se pudo realizar un análisis de las capacidades y la eficacia de utilizar Kotlin en un proyecto; así mismo se descubrió que la curva de aprendizaje para entender este lenguaje no es muy larga; da la posibilidad de conocer mucho de el en un corto tiempo, además al aprender sobre él se notó porque es usado y amado por muchos desarrolladores en la actualidad.

Con respecto a las siguientes secciones, el proyecto incluye varios apartados como: los objetivos que describen la meta principal a realizar, el estado del arte que trata sobre la diferente información recopilada acerca de Kotlin, el desarrollo en donde se habla de las aplicaciones realizadas, su funcionamiento y las conclusiones que relata los resultados obtenidos con este proyecto.

2. OBJETIVOS

El principal objetivo de este proyecto es el análisis del lenguaje de programación Kotlin mediante el estudio y desarrollo de aplicaciones web y móvil, para obtener un enfoque de sus principales características.

Para ello, se plantea como objetivos secundarios los siguientes:

- Estudiar sobre las principales características y usos de Kotlin.
- Analizar las primordiales diferencias que existe entre Java y Kotlin.
- Adquirir conocimientos acerca del lenguaje de programación enfocado al desarrollo web y móvil.
- Comprobar las características y aspectos que posee Kotlin a través del desarrollo de dos aplicaciones.
- Mostrar el funcionamiento de los dos sistemas desarrollados y el resultado obtenido gracias al uso de Kotlin.

Una vez definido los objetivos que se alcanzara con el proyecto, se procede a realizar un estado del arte para conocer más sobre este lenguaje de programación.

3. ESTADO DEL ARTE

3.1. Kotlin

3.1.1. ¿Qué es Kotlin?

Kotlin es un lenguaje de programación moderno, multiplataforma y puede ser utilizado para varios fines, además su potencial es extraordinario, es por esta razón, que los programadores lo usan para el desarrollo móvil, tanto en Android como en iOS, y en la creación de aplicaciones web del lado del cliente y del servidor; también es conocido por su diseño práctico y su conciso sintaxis, sin olvidar que provee una gran oportunidad para la reutilización de código y poder compartir entre desarrolladores en el uso de múltiples proyectos; actualmente Kotlin es el cuarto lenguaje más apreciado por los desarrolladores y tiene el décimo quinto puesto entre los lenguajes más populares en el 2019 [1].

3.1.2. ¿Cuándo se crea?

La empresa JetBrains se dedica al desarrollo de software desde el 2000 y ha creado muchas herramientas que son usadas por grandes empresas [2], en el 2010 empezó con el desarrollo y diseño del lenguaje Kotlin, después de un año de arduo desempeño liderado por Andrey Breslav, en julio del 2011 esta empresa reveló el Proyecto Kotlin, y en febrero del 2012 se presentó como un lenguaje de código abierto para que cualquier desarrollador lo pueda utilizar con la licencia Apache2 [1] [3].

Kotlin fue llamado así por una isla que se encuentra en el mar Báltico, cerca de la ciudad de la empresa en San Petersburgo, JetBrains sigue la tradición de Java al nombrar un lenguaje de programación con el nombre de una isla [3] [4].

3.1.3. ¿Por qué se crea?

Se creó Kotlin por la necesidad de tener un lenguaje de programación moderno, capaz de ejecutarse en la Java Virtual Machine (JVM) y sea compatible con Java y sus herramientas existentes [5].

Con este objetivo en mente los desarrolladores encargados de este proyecto querían obtener las mejores características de otros lenguajes, como Java, C#, Groovy y Scala, por esta razón fue creado para que sea fácil de entender, escribir, y aún más seguro que otros lenguajes de programación; esos fueron sus principales objetivos ser conciso y seguro; el resultado de este proyecto fue un lenguaje de programación el cual, puede ser escrito de manera rápida y eficiente, sin tener que añadir código que solo sea para

cumplir una lógica de programación como es el caso de Java, con Kotlin se pudo solucionar este problema. [3] [4].

3.1.4. Versiones

La primera versión oficialmente estable de Kotlin fue la 1.0 y se lanzó el 15 de febrero del 2016, en el transcurso de ese año se lanzó versiones que corregían ciertos problemas y errores desde la 1.0.1 hasta la 1.0.5; a inicios del 2017 se lanzó la versión 1.1, y a finales de ese mismo año salió la versión 1.2; actualmente Kotlin se encuentra en la versión 1.4 [6].

Desde su primera versión hasta el último censo realizado por la empresa en el 2019, Kotlin es usado por 2.2 millones de usuarios y existe 154 millones de línea de código escritas en GitHub en lenguaje Kotlin, según el siguiente gráfico, cada año se duplica el número de usuarios que utilizan este lenguaje, por esta razón esperan que para este año exista más de 4 millones de usuarios que programen con Kotlin [1].

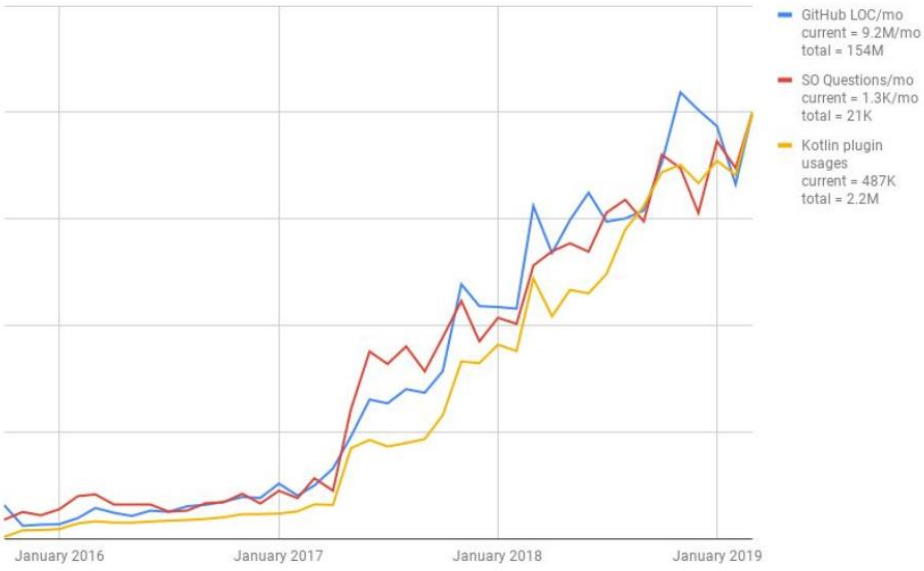

Figura 3.1: Crecimiento del uso de Kotlin *[1]*

3.2. Aplicaciones de Kotlin

Al utilizar Kotlin la JVM para ejecutar sus aplicaciones, este lenguaje de programación puede ser usado básicamente para cualquier cosa que se utilice Java; los principales usos de Kotlin son: el desarrollo de aplicaciones web en el lado del cliente y del servidor, y la más importante, en donde Kotlin tiene su mayor popularidad, el desarrollo de

aplicaciones de Android, además también puede ser usado para escribir aplicaciones nativas de macOS, Windows y Linux [7] [8].

3.2.1. Kotlin y Java Virtual Machine (JVM)

Java Virtual Machine es un software que ejecuta un conjunto de instrucciones, llamadas bytecode, lee un fragmento de estas instrucciones y llama al proceso específico del software o hardware que se asigna al bytecode en donde se ejecuta la JVM, por lo tanto, con el lenguaje Kotlin lo que hace es compilar su código fuente a bytcode de Java y ejecutar estas instrucciones en la JVM [9].

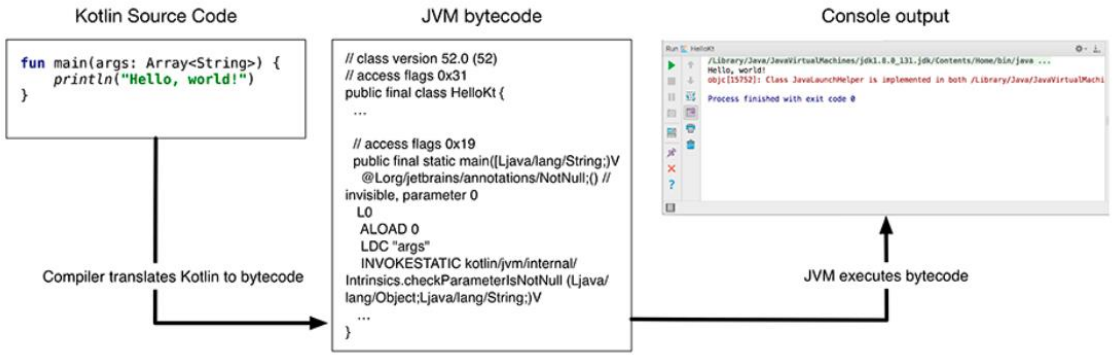

Figura 3.2: Compilación y ejecución de Kotlin *[9]*

3.2.2. Kotlin para la Web

Existen dos formas de trabajar con Kotlin para la realización de aplicaciones web, la primera es en el Front End, es posible porque se puede trabajar en conjunto con JavaScript, el segundo es en la parte del servidor.

- Kotlin en el lado del cliente

Es posible trabajar con JavaScript gracias al compilador de Kotlin, así como JVM puede compilar el código y transformarlo en bytecode, para poder ejecutarlo, se puede hacer lo mismo con JavaScript; cuando se crea un proyecto en Kotlin que se dirige a JavaScript, este código Kotlin se transpila en código JavaScript, dicho de otra manera todo el código se compila y se convierte en otro lenguaje, incluida las bibliotecas de Kotlin, de esta manera pareciera que todo el código está escrito solo en JavaScript, es por esta razón que JetBrains hace todo lo posible para garantizar que el compilador de Kotlin genere un código legible de JavaScript durante el proceso de transpilación [5].

Además de utilizar Kotlin con JavaScript también es posible usarlo para el desarrollo de HTML y CSS con unos DSLs de Kotlin, con el fin de realizar toda la parte del lado del cliente solo con Kotlin [8].

- Kotlin en el lado del servidor

En el lado del servidor el uso de Kotlin es muy amplio, además de poder crear el Backend de una aplicación web, también se puede usar para crear Backends de aplicaciones móviles que se comunican por Json, o en el desarrollo de microservicios que se comunican con otros microservicios; y como en todos estos años de desarrollo web se ha creado aplicaciones con Java, y muchas veces es necesario añadir nuevos procesos a estos sistemas, con Kotlin gracias a su interoperabilidad con Java se puede crear nuevos módulos o migrar un servicio entero a este lenguaje sin ningún problema, esta es una de las grandes ventajas que posee, además también se puede crear el Backend enteramente con Kotlin así se obtiene un código más compacto, confiable y fácil de mantener [10].

3.2.3. Kotlin en Android

En mayo del 2017 Kotlin tuvo oficialmente el apoyo de Google como lenguaje de desarrollo para Android, porque desde el 2015 tuvo una buena aceptación por parte de los desarrolladores móviles y su popularidad incrementó [5] [1]

El uso de Kotlin en Android proporcionó como resultado un lenguaje más moderno y da como resultado una experiencia más productiva y placentera, además de que las tareas comunes que realizaban los desarrolladores se podían realizar con menos líneas de código o en algunos casos sin necesidad de código, el compilador hace todo el trabajo, es por esta razón que empezó a hacerse popular y a más desarrolladores les gustó trabajar con Kotlin para Android [5] [10].

Más adelante se profundiza más sobre este tema de Kotlin y Android.

3.2.4. Kotlin Nativo

Jetbreans trabaja para que Kotlin sea un lenguaje de programación que pueda usarse en cualquier dispositivo, por eso se creó Kotlin Nativo; este permite a los desarrolladores compilar Kotlin en binario nativo sin la necesidad de tener una JVM presente y de esta manera se pueda ejecutar en Windows, macOS o Linux, además permite interactuar e integrarse con otros lenguajes como C++, Objective C o Swift, como lo hace con Java cuando se utiliza la JVM y así logra crear proyectos multiplataforma donde sus procesos comunes son desarrollados en Kotlin [5] [8].

Las ventajas de utilizar Kotlin nativo son las siguientes:

- Se ejecuta sin una JVM, gracias a la utilización de una máquina virtual de bajo nivel (LLVM).
- Al no haber la necesidad de una JVM se puede ejecutar en entornos como iOS.

- Compila el código en lenguaje nativo.
- Tiene la capacidad de producir un ejecutable autónomo.
- Es multiplataforma, puede ejecutarse en varios entornos.
- Tiene interoperabilidad con otros lenguajes de programación. [11]

3.2.5. Entorno de desarrollo para aplicaciones web y sistemas nativos

Al ser Kotlin un lenguaje de programación como cualquier otro, este puede ser escrito hasta en un editor de texto normal, pero es recomendable utilizar el entorno de desarrollo (IDE) creado por la misma empresa JetBrains, IntelliJ Idea, cuando se utiliza para el desarrollo de aplicaciones web y sistemas nativos [9].

IntelliJ facilita en la escritura de código y ayuda que este sea correcto en su sintaxis y semántica, además tiene características como sugerir código según se desarrolla y la finalización automática del mismo; se puede añadir puntos de interrupción de depuración y pasos de código en tiempo real cuando se ejecuta un programa; es capaz de formatear el código que tenga un mal espaciado y cuando se utiliza de una forma incorrecta la sangría; y la parte más importante, como JetBreains fue la empresa que creo Kotlin al igual que IntelliJ, la integración entre estos dos está bien diseñada, por consecuencia se obtiene una mejor experiencia a la hora de desarrollar y se trabaja con un entorno maduro, estable y ofrece varias herramientas para trabajar en conjunto con Kotlin [9].

3.3. Casos de éxito y apoyo de empresas

Kotlin es muy popular en la comunidad de Android, es por esta razón que la mayoría de sus proyectos populares desarrollados son para este sistema; las aplicaciones más conocidas programadas en este lenguaje son:

- Netflix
- Pinterest
- Twitter
- Airbnb
- NYTimes
- Reddit
- American Express [12]

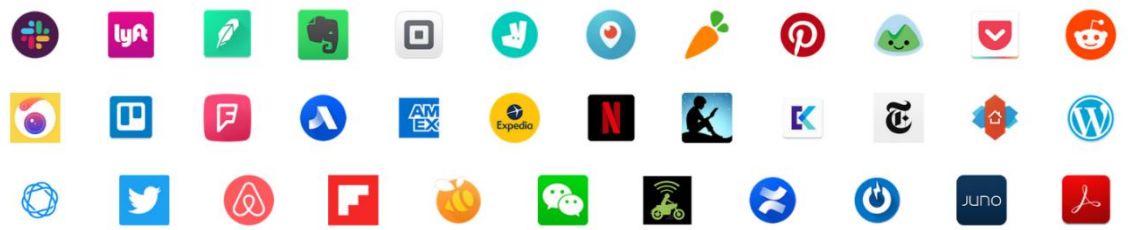

Figura 3.3: Aplicaciones más populares desarrolladas con Kotlin *[12]*

Los desarrolladores encargados de Zomato, una aplicación que es utilizada para encontrar alimentos de buena calidad de restaurantes, con la opción de acceder a sus menús, fotos e información de contacto del lugar, optó por cambiar su lenguaje de programación a Kotlin, con el fin de obtener un código más seguro y conciso, además redujeron el número de líneas de código cuando añadían nuevas funcionalidades en el sistema, con relación a Java, en algunos casos se reducía un 55%; esta empresa recomienda utilizar Kotlin [12] [13].

JetBrains posee algunos socios tecnológicos con el fin de colaborar, integrar y comercializar entre ellos, además de compartir objetivos comunes como hacer que los desarrolladores sean aún más productivos [14]; las mejores empresas que están asociadas con JetBrains son:

Figura 3.4: Socios tecnológicos de JetBrains *[14]*

3.4. Kotlin y Java

3.4.1. Java

Java es un lenguaje de programación y una plataforma de desarrollo (JVM) creado por Sun Microsystems originalmente, después de varios años de su creación Oracle adquiere la empresa Sun y adicionalmente Java; en sus inicios fue creado para ser más seguro que los lenguajes C y C++ [15].

Actualmente es el lenguaje de programación número uno para el desarrollo de las tendencias tecnológicas actuales, posee millones de desarrolladores que ejecutan alrededor de 45 mil millones la JVM; además es la primera elección como lenguaje de programación para el desarrollo en la nube, las empresas lo usan más que otro lenguaje,

todo esto se logró porque Java se encuentra en continua innovación por parte de sus propietarios [16].

Entre las principales características que admite Java como lenguaje de programación son clases, interfaces, herencia, objetos, polimorfismo, paquetes, excepciones, anotaciones, tipos anidados, importaciones estáticas, enumeraciones y muchas otras cosas más, que hacen de Java un lenguaje completo y utilizado alrededor del mundo para un sinfín de propósitos [15].

3.4.2. Kotlin vs Java

Kotlin es un lenguaje de programación bastante joven en comparación con Java, sin embargo, como se habló anteriormente su comunidad de desarrolladores está constante aumento [17], pero cuáles son sus diferencias con Java.

- Una de las primeras y más notables diferencias que existe entre estos dos lenguajes es la cantidad de líneas de código que se utiliza para desarrollar un programa, gracias a la consistencia que posee Kotlin, se puede tener más funcionalidades con menos código, y eso da como resultado una menor cantidad de errores que se puede cometer [17]. En el siguiente ejemplo se observa que tan sencillo es escribir una clase con Kotlin comparado con Java:

Java

```java
public final class Person
{
    private String name;
    private int age;
    private float height;

    public Person(String name, int age, float height)
    {
        this.name = name;
        this.age = age;
        this.height = height;
    }
    public Person(String name, int age)
    {
        this.name = name;
        this.age = age;
        this.height = 1.8f;
    }
    public String getName()
    {
        return name;
    }
    public void setName(String name)
    {
        this.name = name;
    }
    public int getAge()
    {
        return age;
    }
    public void setAge(int age)
    {
        this.age = age;
    }
    public float getHeight()
    {
        return height;
    }
    public void setHeight(float height)
    {
        this.height = height;
    }
}
```

Kotlin

```kotlin
data class Person(var name: String,
                  var age: Int,
                  var height: Float = 1.8f)
```

Figura 3.5: Diferencia de una clase entre Java y Kotlin *[17]*

• Para agregar una función adicional a una clase, por lo general en Java se crea una nueva clase que hereda las características de la anterior o a veces se utiliza un patrón de diseño; en Kotlin para solventar este problema existe una función llamada extensión, esta se define fuera de la clase, pero es miembro de ella, sin tener que modificar la clase original [18].

• NullPointerException ha sido un gran problema para muchos desarrolladores, en Java si se quiere acceder a una variable que sea null, dará como resultado un NullPointerException que interrumpe el flujo de código; para evitar este problema los desarrolladores de JetBrains crearon una característica especial llamada Null Safety [18], y es muy sencillo de usarlo, en el siguiente ejemplo se muestra como da este problema en Java y como evitarlo en Kotlin.

Java

```
Public static void main(String args[])
{
    String name= null;
    System.out.println(name.length());
}
```

Kotlin

```
Fun main(args: Array<String>)
{
    var name: String? = nullprintln(name?.length)
}
```

Figura 3.6: Solución al problema de NullPointerException *[18]*

El resultado que daría con Java sería un NullPointerException y la ejecución del código se detendría, pero con Kotlin daría como respuesta Null.

Kotlin hace posible esto con el uso del operador ?. y con ello realiza una llamada segura a la variable, si esta es null le da el valor de null y evita que la aplicación se bloquee automáticamente [17].

• Con Java en Android es muy frecuente utilizar el método findViewById() para localizar algún elemento de la interfaz, y para cada elemento hay que declarar una variable, lo que finalizaba con un trabajo tedioso de realizar, con Kotlin se evita todo esto porque ya no es necesario realizar estas declaraciones, ahora solo se accede a ellos con el identificador que se declara en el XML [17].

Java

```java
public class MainActivity extends AppCompatActivity
{
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button = (Button) findViewById(R.id.button);
        final TextView text = (TextView) findViewById(R.id.text);
        button.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                text.setText("You've clicked a button");
            }
        });
    }
}
```

Kotlin

```kotlin
class MainActivity : AppCompatActivity()
{
    override fun onCreate(savedInstanceState: Bundle?, persistentState: PersistableBundle?)
    {
        super.onCreate(savedInstanceState, persistentState)
        setContentView(R.layout.activity_main)

        button.setOnClickListener { text.text = "You've clicked a button" }
    }
}
```

Figura 3.7: Llamadas a elementos de interfaz en Java y Kotlin *[17]*

Ahora es más fácil llamar a un elemento que se encuentra en XML, sin la necesidad de declararlo, de esta manera se ocupa menos código en Kotlin con respecto a como se hacía las declaraciones por cada elemento en Java.

• Cuando se ejecuta en Java una aplicación debe cargar su contenido así este no sea aún requerido o necesario, lo que da como resultado una carga lenta; pero en Kotlin se corrige este problema gracias a una función llamada Lazy loading, con esta función se logra diferir la inicialización de los objetos hasta que se lo necesite, y de esta manera se disminuye el tiempo de carga de una aplicación; Java no posee esta función [18].

• Realizar constructores secundarios en Kotlin es mucho más fácil, se ahorra líneas de código con respecto a Java, si existiera una clase con bastantes campos y se necesita de dos constructores, con Java se duplicaría el código innecesariamente, pero con Kotlin basta con añadir los parámetros que se necesita en el segundo constructor [18], como se muestra en el siguiente ejemplo.

Java

```java
class Student
{
    String name;
    Int roll;
    Int marks;
    Int elective;

    Student (String name, Int roll, Int marks)
    {
        this.name = name;
        this.roll = roll;
        this.marks = marks;
        this.elective = 0;
    }
    Student (String name, Int roll, Int marks, Int elective)
    {
        this.name = name;
        this.roll = roll;
        this.marks = marks;
        this.elective = elective;
    }
}
```

Kotlin

```kotlin
class Student
{
    val name: String
    val roll: Int
    val marks: Int
    private var elective = false

    constructor(name: String, roll: Int, marks: Int)
    {
        this.name = name
        this.roll = roll
        this.marks = marks
    }
    constructor(name:   String,   roll:   Int,
            marks:   Int,   elective:   Boolean) : this(name, roll, marks)
    {
        this.elective = elective
    }
}
```

Figura 3.8: Constructores en Java vs Kotlin *[18]*

• Las colecciones en Kotlin son super sencillas de utilizar, ahorra como siempre espacio de código y se tiene un código más sencillo y eficiente, con respecto a Java que suele tornarse difícil, con este pequeño ejemplo muestra la diferencia entre estos dos, y si se tratase de un proyecto enorme, se redujera mucho las líneas de código utilizadas. [17].

Java

```java
ArrayList<Student> students = new ArrayList<Student>()
{
    {
        add(new Student("John", 0));
        add(new Student("Julia", 2));
        add(new Student("Matt", 1));
        add(new Student("Katie", 0));
        add(new Student("Dan", 0));
    }
};

ArrayList<Student> firstList = new ArrayList<>();
ArrayList<Student> secondList = new ArrayList<>();

for (Student student: students)
{
    boolean isFirstFilled = firstList.size() >= 3;
    boolean isSecondFilled = secondList.size() >= 2;

    if (isFirstFilled && isSecondFilled) break;
    int mark = student.getMark();
    if (mark == 0 && !isFirstFilled)
    {
        firstList.add(student);
    } else if (mark == 1 && !isSecondFilled)
    {
        secondList.add(student);
    }
}
```

Kotlin

```kotlin
var students = listOf(Student("John", 0), Student("Julia", 2), Student("Matt", 1),
        Student("Katie", 0), Student("Dan", 0))

var firstList = students.filter { it.mark == 0 }.take(3)
var secondList = students.filter { it.mark == 1 }.take(2)
```

Figura 3.9: Diferencia de colecciones entre Java y Kotlin *[17]*

3.4.3. Interoperabilidad entre Java y Kotlin

Una de las mejores características que posee Kotlin es la interoperabilidad con Java, esto significa que los archivos desarrollados en cualquiera de estos dos lenguajes de programación pueden coexistir en el mismo proyecto sin problema alguno, se puede llamar a métodos hechos en Java desde Kotlin y viceversa, esto quiere decir que Kotlin puede usar librerías ya existentes de Java; además si se quiere pasar un proyecto a Kotlin hay la posibilidad de realizar esta transición lentamente, tal vez no se obtenga un proyecto enteramente en Kotlin pero se tiene la oportunidad de que las nuevas funciones sean desarrolladas en este lenguaje [9].

3.4.4. Llamar a Java desde Kotlin y Kotlin desde Java

En ambos casos las llamadas que se realizan son muy sencillas gracias a su capacidad de integrarse entre los dos lenguajes, en cualquiera de los casos se debe realizar una instancia de la clase que se vaya a utilizar, y adicional llamar al método que se necesite, si es requerido se envía parámetros como en cualquier otro lenguaje [8].

Java

```java
public class JavaClass
{
    public int getZero()
    {
        return 0;
    }
}
```

Kotlin

```kotlin
val javaObject = JavaClass()

println(javaObject.zero) // 0
```

Figura 3.10: Llamar a Java desde Kotlin *[8]*

Kotlin

```kotlin
data class Counter(val value: Int)
{
    operator fun plus(other: Counter) = Counter(value + other.value)
}
```

Java

```java
public class UseCounter {
    public static void main(String[] args) {
        Counter counter = new Counter(1);

        System.out.println(counter.plus(counter)); //Counter(value=2)
    }
}
```

Figura 3.11: Llamar a Kotlin desde Java *[8]*

3.5. Sintaxis de Kotlin

Toda la sintaxis básica que se describe a continuación fue tomada del sitio oficial de Kotlin [19].

- Paquetes e importaciones

Deben ser escritas al inicio de un archivo.

```
package my.demo
import kotlin.text.*
// ...
```

• Función principal

Es llamada punto de entrada en Kotlin.

```
fun main()
{
    println("Hello world!")
}
```

• Variables

Variables cuyo valor solo puede ser definido una sola vez:

```
val a: Int = 1  // Asignación inmediata
val b = 2   // El tipo Int es inferido
val c: Int  // El tipo es definido pero el valor aun no asignado
c = 3       // Valor asignado
```

Variables cuyo valor puede ser cambiado tantas veces sea necesario:

```
var x = 5 // El tipo de la variable es inferido
x += 1
```

• Funciones

Como todo lenguaje de programación existen diferentes formas de utilizar las funciones:

Función con dos parámetros Int y con un retorno Int:

```
fun sum(a: Int, b: Int): Int
{
    return a + b
}

fun main()
{
    print("La suma de 3 y 5 es: ")
    println(sum(3, 5))
}
```

Función con un valor de retorno inferido:

```
fun sum(a: Int, b: Int) = a + b

fun main()
{
    println("Suma de 19 y 23 es ${sum(19, 23)}")
}
```

Función sin un valor de retorno (el tipo de valor Unit puede ser omitido):

```
fun printSum(a: Int, b: Int): Unit
{
    println("La suma de $a y $b es ${a + b}")
}

fun main()
{
    printSum(-1, 8)
}
```

• Comentarios

Como cualquier lenguaje de programación tiene comentarios de una sola línea o multilínea:

```
// Este es un comentario de una línea

/* Este es un bloque comentado
    en múltiples líneas*/
```

• Condicionales

La expresión if es usado como en cualquier otro lenguaje:

```
fun maxOf(a: Int, b: Int): Int
{
    if (a > b)
    {
        return a
    } else {
        return b
    }
}
```

• Valores anulables

Para permitir valores nulos en una variable se debe utilizar ? luego del tipo, por ejemplo, en este caso va a dar un error si se le asigna null a la variable.

```
var a: String = "abc"
a = null // Error de compilación
```

Pero si se le asigna ? al tipo de variable no va a existir ningún problema.

```
var b: String? = "abc" // Puede ser null
b = null // ok
print(b)
```

Sin embargo, si se quiere acceder a un método de la variable b que es nula, va a dar un error, por ejemplo:

```
val l = b.length //error: la variable 'b' no puede ser null
```

Para que no genere ningún problema se debe utilizar también ? en el momento que se llama a un método de la variable:

```
println(b?.length) // El resultado es null
```

A esto se le conoce como una llamada segura para cuando los valores que se buscan pueden ser nulos y no se desea que en el tiempo de ejecución genere algún error.

• Comprobaciones de tipo null

Comparar variables de tipo null es muy sencillo, se lo demuestra con este ejemplo:

```
fun printProduct(arg1: String, arg2: String)
{
    val x = parseInt(arg1)
    val y = parseInt(arg2)
    if (x != null && y != null)
    {
        println(x * y)
    }
    else
    {
        println("'$arg1' o '$arg2' no son números")
    }
}
```

De esta manera solo realiza la acción si ninguna de las dos variables es nula.

• Ciclo for

Se puede realizar el ciclo de las dos siguientes maneras, en el primero se va a realizar el ciclo desde uno hasta tres, en el segundo caso va a ir de seis a cero y se resta dos por cada iteración.

```
for (i in 1..3)
{
    println(i)
}
for (i in 6 downTo 0 step 2)
{
    println(i)
}
```

• Ciclos while y do while

Estos dos son similares a cualquier lenguaje de programación, como se puede observar en el siguiente ejemplo:

```
var i = 0
while (i < 5)
{
    println(i)
    i++
}

var j=0
do
{
    println(j)
    j++
} while (j<5)
```

• When

La función es similar a la expresión switch en otros lenguajes de programación.

```
fun describe(obj: Any): String =
    when (obj)
    {
```

```
        1           -> "Uno"
        "Hola"      -> "Mundo"
        is Long     -> "Long"
        !is String  -> "No es un string"
        else        -> "Desconocido"
    }

fun main()
{
    println(describe(1)) // Uno
    println(describe("Hola")) // Mundo
    println(describe(1000L)) // Long
    println(describe(2)) // No es un string
    println(describe("otro")) // Desconocido
}
```

Los valores que van en el lado izquierdo serían como los case para el switch, según la elección se muestra el resultado en la parte derecha.

• Rangos

Se utiliza para comprobar si una variable o valor se encuentra dentro de un rango específico.

```
if (i in 1..4)  // equivalente a 1 <= i && i <= 4
{
    print(i)
}
```

3.6. Kotlin y Android

Para empezar a hablar de Kotlin para el desarrollo de aplicaciones en Android, primero se debe conocer un poco sobre el sistema operativo para dispositivos móviles, Android.

3.6.1. Android

Antes de Android las empresas que se dedicaban a la fabricación de móviles debían obtener una licencia pagada para el sistema operativo (SO) o gastar más en la creación de su propio SO; en 2007 Google se alió con el sector de dispositivos celulares, y se creó la alianza Open Handset Alliance, además se anunció el sistema operativo Android para que esta nueva unión ayudara en el desarrollo del SO y fuese de código abierto, de esta manera cualquiera podría tener el código de Android para el desarrollo de aplicaciones

o mejorar el sistema operativo para crear uno diferente, esta es la filosofía de Android a diferencia con iOS de Apple [20] [21].

Android comenzó con la versión 1 y luego pasó a la 1.5 llamada Cupcake, desde ahí empezaron a llamar a cada versión con el nombre de un postre, las siguientes versiones fueron llamadas de esta manera: 1.6 Donut, 2.0 Eclair, 2.2 Froyo, 2.3 Gingerbread, 3.0 Honeycomb, 4.0 Ice Cream Sandwich, 4.1 Jelly Bean, 4.4 KitKat, 5.0 Lollipop, 6.0 Marshmallow, 7.0 Nougat, 8.0 Oreo, 9 Pie y actualmente se encuentra en la versión 10 [22] [23].

Actualmente las versiones que se encuentran en más dispositivos Android son Oreo, Nougat y Marshmallow, esto quiere decir que el mercado de Android para el desarrollo de aplicaciones debe estar más enfocadas a estas versiones, porque son las que más usuarios tienen; como muestra el siguiente gráfico de datos recogidos el 1 de junio del 2020 [23].

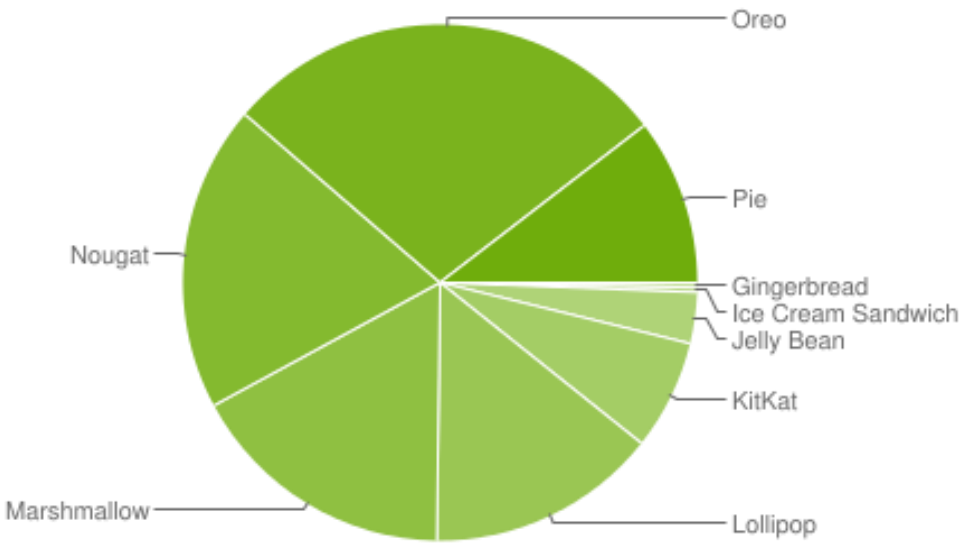

Figura 3.12: Cantidad de dispositivos que usan una determinada versión *[23]*

- Android Oreo

Esta versión mejora la experiencia de usuario, rediseña las notificaciones, con el fin de que sea más sencillo su administración y comportamiento; se añade la opción de tener insignias de notificaciones asociadas a una aplicación, estas están presentes en los íconos de la app; también permite posponer notificaciones a una fecha y hora que se desee; mejora la compatibilidad para la herramienta de múltiples pantallas; con lo que respecta a la accesibilidad, se añadió un botón en el área de navegación del sistema, para poder activar de forma rápida la funcionalidad de accesibilidad [24].

- Android Pie

Cuenta con varias mejoras en el sistema, una de ellas es el sistema de mensajes en el área de notificaciones, ahora es compatible visualizar imágenes, mejora la interfaz de esta herramienta, lo que da la posibilidad de ver el avatar de una persona; es compatible la utilización de varias cámaras en un dispositivo que posea dos o más cámaras ya sea frontales o posteriores; también se añade un botón en la barra del sistema para realizar una rotación de pantalla, para que no suceda de manera involuntaria y el usuario tenga el control de ello [25].

- Android 10

Para esta versión la tecnología para dispositivos móviles avanzó rápido, es por esta razón que se desarrolló Android 10 para tener la compatibilidad con dispositivos plegables y redes 5G; como en la versión Oreo se mejoró la herramienta de múltiples pantallas, se utiliza esta opción para los dispositivos plegables, así se optimiza el uso de varias tareas al mismo tiempo; con las redes 5G se obtuvo una velocidad de navegación mejor y con una menor latencia, gracias a esta compatibilidad se puede tener más beneficio de esta tecnología; además se añadió nuevas características que mejoran la experiencia de usuario, ahora existe la posibilidad de tener una respuesta inteligente a las notificaciones, el dispositivo sugiere acciones para el usuario de manera rápida sin acceder a la aplicación y solo a través de la notificación; se añade un tema oscuro para todo el sistema Android que ayuda al ahorro de energía y es genial para espacios con poca luz; también se puede realizar la navegación por el dispositivo a través de gestos, esto elimina la barra de navegación, lo que añade un espacio adicional a las aplicaciones [26].

3.6.2. Kotlin para Android

Luego del lanzamiento de Kotlin algunas compañías respaldaron el proyecto de Kotlin, entre ellas son Gradle que añadió soporte en la creación de scripts; Square que es la compañía encargada de la mayoría de las bibliotecas de Android, dijo que estas son bastantes compatibles con este lenguaje; y como se mencionó en puntos anteriores, Google oficialmente anunció el soporte de Kotlin para Android; esto significó que todas las herramientas que sean creadas para Android además de ser compatibles con Java también serían con Kotlin; por esta razón hubo un gran incremento en el número de compañías que empezaron a incluir el lenguaje en sus proyectos [27].

Para realizar una aplicación se necesita del kit de desarrollo de software de Android (SDK), este se encuentra escrito en Java, pero para que sea compatible con Kotlin, este lenguaje se fusiona con Java a través del SDK antes de que el entorno de desarrollo lo convierta en una aplicación ejecutable, de esta manera si se escribe el código en Java o

en Kotlin el resultado es el mismo, sin embargo como se ha mencionado anteriormente Kotlin tiene sus grandes ventajas para hacer de una opción viable para el desarrollo de aplicaciones para Android [28].

Muchos desarrolladores del Customer Advisory Board (CAB) y Google Developers Experts (GDE) disfrutan al utilizar Kotlin con Android y piden que este lenguaje tenga más soporte; además destacan que entre sus principales características son: la interoperabilidad, su código es más seguro, es expresivo y conciso; actualmente más del 55 aplicaciones de Google están desarrolladas con Kotlin, y cuando crean nuevas funcionalidades reducen un 33% de líneas de código y reduce un 30% el número de problemas causados por Null Pointer Exception; además como se observa en el siguiente gráfico Kotlin posee los mismos beneficios de trabajar con Java [29].

	Lenguaje Java	Kotlin
Compatibilidad con el SDK de la plataforma	Sí	Sí
Compatibilidad con Android Studio	Sí	Sí
Lint	Sí	Sí
Compatibilidad con documentos guiados	Sí	Sí
Compatibilidad con documentos de API	Sí	Sí
Compatibilidad con AndroidX	Sí	Sí
API específicas de AndroidX Kotlin (KTX, corrutinas, etc.)	NA	Sí
Capacitación en línea	Mejor esfuerzo	Sí
Ejemplos	Mejor esfuerzo	Sí
Jetpack Compose	No	Sí

Figura 3.13: Soporte de Java y Kotlin frente a Android *[29]*

Además de los beneficios que poseen ambos lenguajes, con Kotlin existe la posibilidad de obtener un mejor aprendizaje gracias a sus cursos en línea y sus claros ejemplos, que se pueden acceder a través de la página oficial de Android Developers. También ofrece la posibilidad de trabajar con el kit de desarrollo Jetpack Compose, este es un conjunto de herramientas que se utiliza en la creación de interfaces nativas para Android, lo que hace es simplificar y acelerar el desarrollo de la interfaz, estas herramientas se compilan a partir de Kotlin [30], aún no se encuentra completo su desarrollo, pero su uso ya se encuentra disponible.

3.6.3. Entorno de desarrollo para Android con Kotlin

Existen algunos entornos de desarrollo para crear aplicaciones de Android, pero el ideal es Android Studio, además de ser compatible con Kotlin, tiene la capacidad de crear

nuevas aplicaciones con este lenguaje, agregar archivos con Kotlin a una aplicación ya creada o convertir código Java en Kotlin gracias a su convertidor de código [31].

Android Studio está basado en el IDE IntelliJ Idea, y es el entorno de desarrollo oficial para realizar aplicaciones para Android, además con este IDE se obtiene una gran productividad en el desarrollo porque posee varias características importantes, como el emulador de un dispositivo móvil que está ya cargado con varias funciones y es rápido; su sistema de compilación es eficiente y flexible porque está basado en Gradle; se puede realizar cambios en el código de la aplicación mientras se encuentra en ejecución sin tener que reiniciarla; su entorno es unificado porque tiene la posibilidad de crear proyectos para cualquier dispositivo Android; entre otras características más, hacen de Android Studio el entorno de desarrollo ideal para realizar aplicaciones para el sistema operativo Android y aún más al ser compatible de trabajar con Kotlin [32].

Concluido el estado del arte y definido las principales características, beneficios y diferencias de Kotlin se realiza el desarrollo de las aplicaciones para comprobar lo escrito en este capítulo.

4. DESARROLLO

En este capítulo se planteó realizar dos aplicaciones una Web y otra móvil, con la finalidad de comprobar las características y capacidades que posee este lenguaje.

A continuación, se describe sobre los requisitos que poseen estas dos aplicaciones, su arquitectura, su modelo de datos, ciertas características que se aplicaron durante el desarrollo y el funcionamiento de ellas.

4.1. Requisitos de las aplicaciones

Como se escribió anteriormente, se desarrolló dos aplicaciones, la idea principal de estas es llevar un control de registro de turnos de los empleados dentro de una empresa; para ello existen dos roles, el administrador y el empleado.

En la primera aplicación que se realizó para el entorno web estos dos roles pueden iniciar sesión en él; y en la parte móvil solo el empleado puede hacer uso de él, con las mismas opciones que tiene en el sistema web.

En los siguientes puntos se demuestra los diferentes requisitos que poseen las aplicaciones, para ello se divide en el rol del administrador y el del empleado.

4.1.1. Rol de administrador

En el siguiente diagrama se muestra como este rol puede realizar varias operaciones.

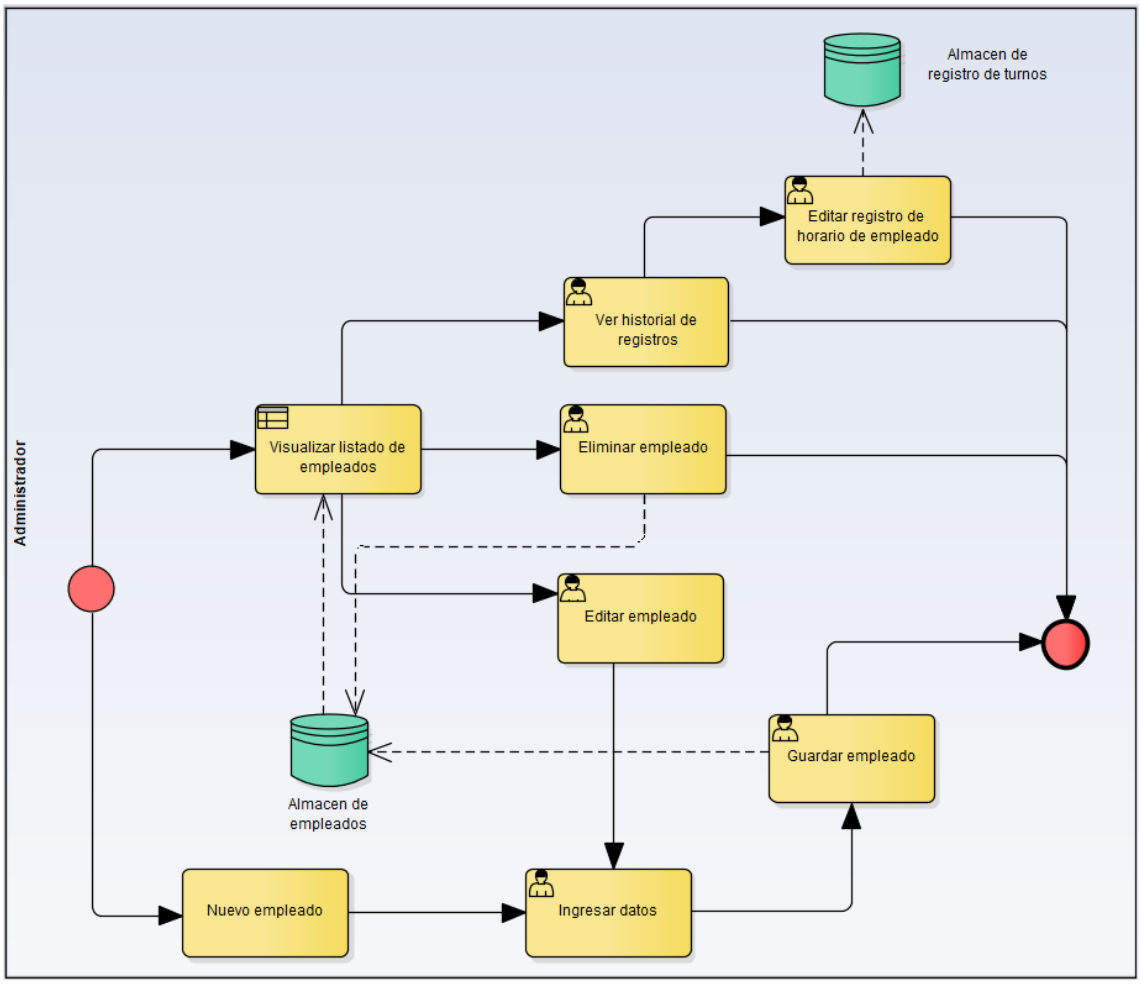

Figura 4.1: Diagrama de operaciones del administrador

Como se observa en el diagrama, el administrador tiene control de todos los empleados, él puede realizar todas las operaciones básicas que son: inserción, edición, eliminación y visualización, además de revisar el historial de registros para cada empleado y poder editarlo, todas estas tareas son conectadas con la base de datos.

Estas operaciones solo se pueden realizar en el sistema web, para ello el administrador deberá ingresar con sus credenciales y accederá a todo este apartado.

4.1.2. Rol de empleado

Las operaciones que realiza este rol son tanto para el sistema web y móvil, porque puede hacer las mismas acciones en los dos. Con este diagrama se observa sus tareas.

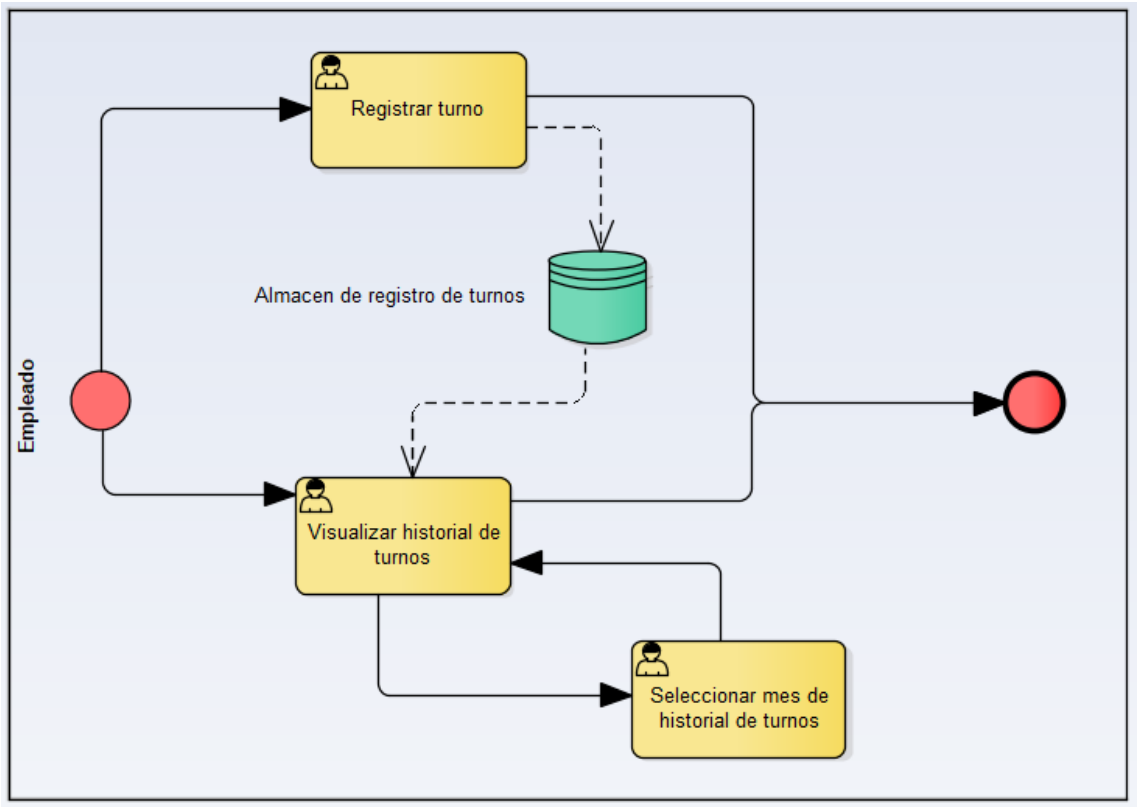

Figura 4.2: Diagrama de operaciones del empleado

El empleado básicamente realiza dos cosas, el registro de su turno que depende del horario laboral que exista en el sistema, y visualizar todos los registros que ha realizado durante el mes; también tiene la posibilidad de cambiar el mes para revisar historiales anteriores. Todos estos datos son almacenados en la base de datos y al igual que el administrador, primero debe ingresar con sus credenciales en cualquiera de las dos aplicaciones para acceder a realizar estas dos operaciones.

4.2. Modelo de datos

En los anteriores diagramas se visualizó que la información que se maneja es de los empleados y los registros de sus turnos, además de que las acciones se conectan a un base de datos que contiene información sobre estos dos. Por esta razón los dos elementos son los que tienen interacción constante con los dos sistemas desarrollados y los más importantes.

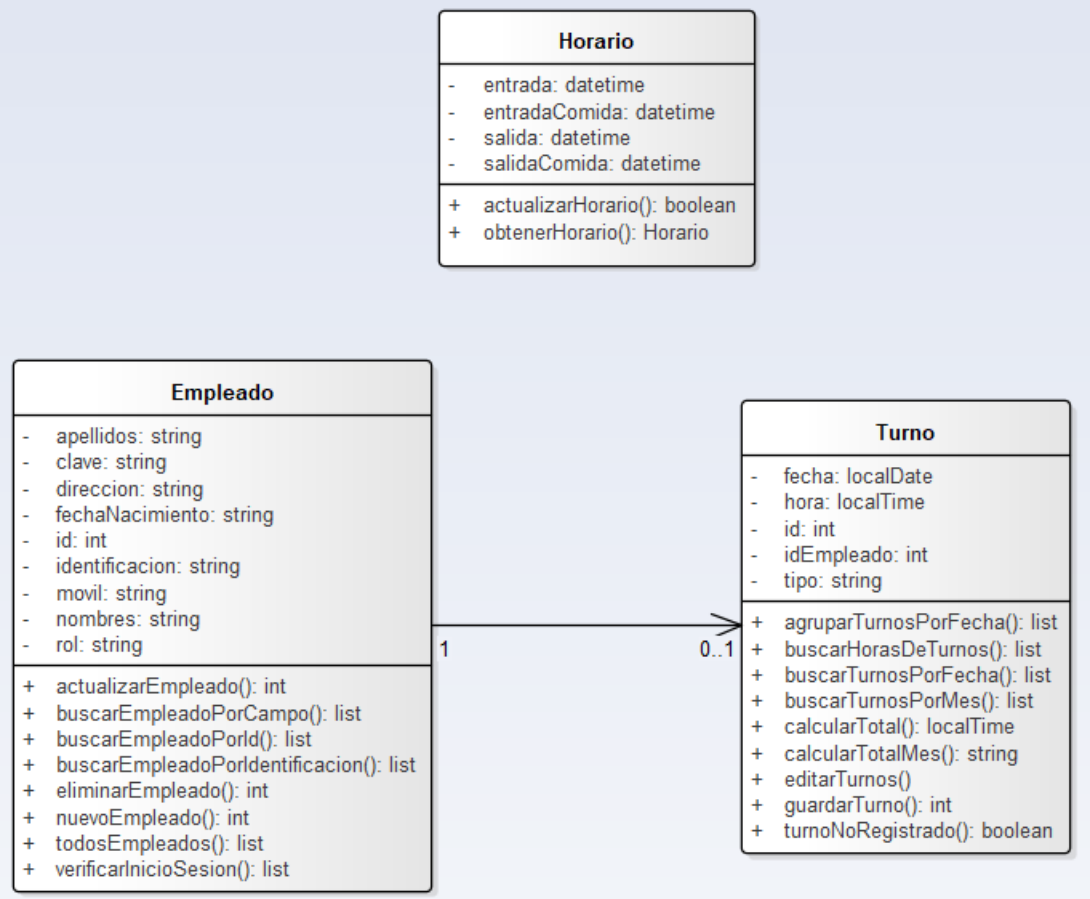

Figura 4.3: Diagrama de clases

Con este diagrama se puede apreciar los datos que posee cada clase y las diferentes operaciones que se manejaran con ellas, la clase Empleado está relacionada directamente con la clase Turno, puesto que un empleado puede tener cero o varios turnos. Además, se utilizó otra clase, el Horario, en él se almacena el horario laboral que posee la empresa para que los empleados puedan registrarse según estos datos almacenados y no lo hagan a cualquier hora del día.

La clase Empleado posee los atributos identificación y clave, con estos dos puede iniciar sesión en cualquiera de los dos sistemas, también posee un rol, este puede ser el rol de administrador y empleado, para el sistema solo existe empleado con el rol de administrador.

En la clase Turno se almacena el día, aquí llamado fecha en el que se registra el empleado, la hora, el idEmpleado para identificar a que empleado pertenece este turno y un tipo para saber si es el registro en la entrada, salida para la comida, entrada después de la comida o a la salida.

4.3. Arquitectura del sistema

Debido a que son dos tipos de sistemas que se utilizan en plataformas distintas, cada una posee su propia arquitectura, por consiguiente, se detalla las utilizadas para estos dos sistemas.

4.3.1. Arquitectura del sistema web

Para el sistema web se utilizó una arquitectura del tipo Modelo-Vista-Controlador (MVC), como se muestra en la imagen.

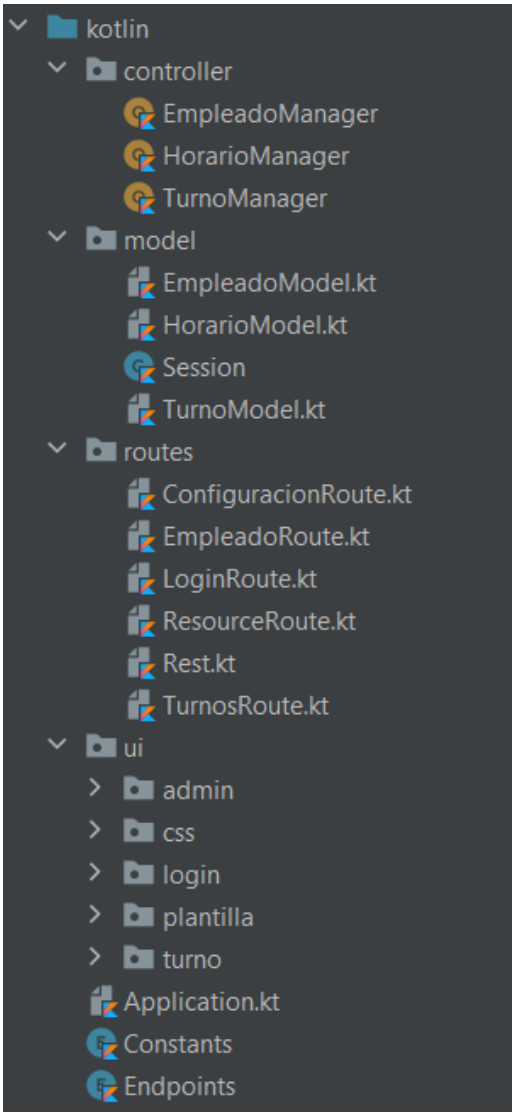

Figura 4.4: Arquitectura del sistema web

- Modelo

En el modelo se encuentra las clases definidas en el anterior diagrama, además de los datos de sesión para almacenar cuando un usuario ingresa en el sistema. Para definir en Kotlin una clase de datos se lo realiza de la siguiente manera:

```
data class EmpleadoModel(
    val id: Int?, val nombres: String, val apellidos: String, val
fechaNacimiento: String,val movil: String, val direccion: String, val
identificacion: String, val clave: String, val rol: String
)
```

Basta con ingresar todos los campos que contendrán esta clase, y sus respectivos tipos de datos; no hace falta generar las funciones get y set para cada uno, ni utilizar un constructor, si algún campo es opcional al momento de llamar a esta clase, solo se debe poner ese campo con ? para mencionar que será nulo y no sea necesario utilizarlo.

Esto es igual para todas las clases creadas en el modelo.

- Vista

La vista del sistema se encuentra dentro de la carpeta ui, ahí se desarrolló todas las páginas web, tanto para la parte del administrador como la del empleado.

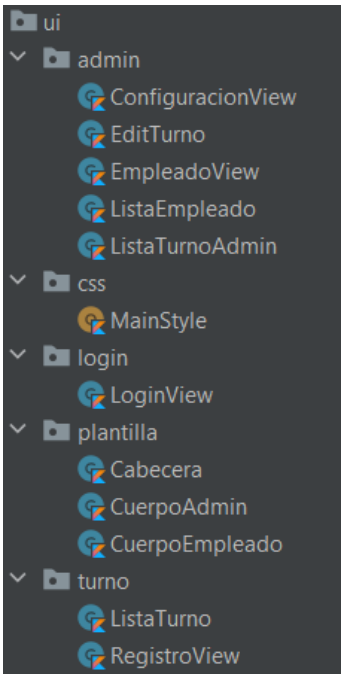

Figura 4.5: Archivos de la vista del sistema web

Para no repetir líneas de código, se utilizó tres plantillas, la cabecera que se encuentra en el LoginView y dentro de las otras dos plantillas, el CuerpoAdmin y CuerpoEmpleado son utilizadas para las páginas del administrador y empleado respectivamente. Todas estas son desarrolladas en lenguaje Kotlin sin utilizar HTML.

```
class LoginView : Template<HTML> {
    val cabecera: Cabecera = Cabecera()
    val alerta = Placeholder<FlowContent>()

    override fun HTML.apply() {
        insert(cabecera) {}
        head {
            link(rel = "stylesheet", href =
"/resources/LoginStyle.css") { }
        }
        body {
            attributes["style"] = """background-image:
URL('/resources/fondo.jpg');
                    background-size: cover;"""
            form(classes = "form-signin", method = FormMethod.post,
action = Endpoints.LOGIN.url) {
                div(classes = "card") {
                    div(classes = "card-body") {
                        h1(classes = "h3 mb-3") { +"Inicio de Sesión"
}
                        br {  }
                        div {
                            insert(alerta)
                        }
                        label(classes = "sr-only") {
                            attributes["for"] = "identificacion"
                            +"Identificación"
                        }
                        input(classes = "form-control", type =
InputType.text, name = "identificacion") {
                            attributes["id"] = "identificacion"
                            attributes["placeholder"] =
"Identificación"
                            attributes["required"] = "true"
                            attributes["autofocus"] = "true"
                        }
                        label(classes = "sr-only") {
                            attributes["for"] = "clave"
                            +"Contraseña"
                        }
                        input(classes = "form-control", type =
InputType.password, name = "clave") {
                            attributes["id"] = "clave"
                            attributes["placeholder"] = "Contraseña"
                            attributes["required"] = "true"
                            attributes["autofocus"] = "true"
                        }
                        br {  }
                        button(classes = "btn btn-lg btn-primary btn-
block", type = ButtonType.submit) {
                            +"Ingresar"
                        }
                    }
                }
            }
        }
    }
}
```

Como se observa se crea una instancia de la plantilla cabecera y se inserta dentro de la página desarrollada, el resto del documento es similar a programarlo en HTML, la estructura es igual, todo va en un orden jerárquico de etiquetas, y existe algunos atributos por defecto que se lo pueden ingresar directamente al iniciar una etiqueta, como el atributo classes, adicional si no existe este atributo se lo puede añadir dentro de la etiqueta con la palabra attributes y para ingresar texto solo se lo hace con el + y entre comillas lo que se desea mostrar en la página.

Del mismo modo se puede crear reglas CSS con Kotlin, el funcionamiento es similar a lo visto con HTML.

```
object MainStyle {
    fun style() :String {
        return CSSBuilder().apply {
            rule(".btn-search"){
                borderRadius = LinearDimension("0px .25rem .25rem
0px")
                border = "2px solid transparent"
            }
            rule(".icon-nav") {
                paddingRight = LinearDimension("10px")
            }
            rule(".list-btn") {
                borderBottomRightRadius = LinearDimension("0px")
                borderTopRightRadius = LinearDimension("0px")
            }
            rule(".edit-btn") {
                borderRadius = LinearDimension("0px")
            }
            rule(".delete-btn") {
                borderBottomLeftRadius = LinearDimension("0px")
                borderTopLeftRadius = LinearDimension("0px")
            }
            rule(".registrarse") {
                fontSize = LinearDimension("22px")
                padding = "16px"
            }
        }.toString()
    }
}
```

Para cada regla que se quiera crear, se debe usar la palabra rule y dentro escribir el selector al cual le afectará el estilo, dentro de él se ingresa los diferentes tipos de estilos que se requiera.

Se puede lograr esto gracias a los DSLs que posee Kotlin para utilizar HTML y CSS, de esta manera se desarrolla páginas web dinámicas con un solo lenguaje de programación y todo el resto se encarga el compilador.

- Controlador

En los controladores se encuentran todas las operaciones que se pueden ejecutar en el sistema, ahí se desarrolló las diferentes conexiones con la base de datos según la acción que se realice.

El siguiente código muestra una parte del controlador para el empleado, con la función de insertar uno nuevo.

```
object EmpleadoManager {

    fun nuevoEmpleado(empleado: EmpleadoModel): Int {
        var result = -1
        return if
(buscarEmpleadoPorIdentificacion(empleado.identificacion).isEmpty()) {
            transaction {
                result = EmpleadoTable.insert {
                    it[nombres] = empleado.nombres
                    it[apellidos] = empleado.apellidos
                    it[fechaNacimiento] = empleado.fechaNacimiento
                    it[movil] = empleado.movil
                    it[direccion] = empleado.direccion
                    it[identificacion] = empleado.identificacion
                    it[clave] = empleado.clave
                    it[rol] = empleado.rol
                }[EmpleadoTable.id]
            }
            result
        } else
            -2
    }
}
```

Con Kotlin no es necesario utilizar el patrón de diseño Singleton, porque da la opción de utilizar la palabra reservada object para una clase, de esta manera no hace falta instanciarla, basta con llamar con el nombre que se declara y se tiene disponible todas las funciones que posee.

- Rutas

Además de utilizar el Modelo, Vista y Controlador, se usó un Framework creado por JetBrains llamado Ktor, gracias a él se añadió las rutas al sistema web que ayudan a crear las direcciones para las diferentes páginas web, además de conectarse con los controladores cuando se realiza alguna acción dentro del sistema.

```
fun Route.login() {
    get(Endpoints.LOGIN.url) {
        call.respondHtmlTemplate(LoginView()) {}
    }
    post(Endpoints.LOGIN.url) {
        val parameters: Parameters = call.receiveParameters()
        val empleado =
EmpleadoManager.verificarInicioSesion(parameters["identificacion"]!!,
parameters["clave"]!!)
```

```
        if(empleado.isNotEmpty()) {
            call.sessions.set(Constants.COOKIE_NAME.value,
Session(empleado[0].id!!,empleado[0].identificacion, empleado[0].rol))
            if(empleado[0].rol == "admin")
                call.respondRedirect(Endpoints.LISTEMPLEADO.url)
            else
                call.respondRedirect(Endpoints.REGISTROTURNO.url)
        } else {
            call.respondHtmlTemplate(LoginView()) {
                alerta {
                    div(classes = "alert alert-dark") {
                        attributes["role"] = "alert"
                        +"¡Datos incorrectos!"
                    }
                }
            }
        }
    }
}
```

Como se muestra en el código, cuando se realiza una petición GET para el Login, este lo dirige a una página web que se encuentra en la vista, llamada LoginView; el Endpoints.Login.url solo es un String de una dirección declarado en otra clase.

Si existe una petición para la misma dirección, pero POST, recibe los parámetros enviados, realiza una consulta al controlador para verificar que los datos sean correctos, depende de la respuesta se dirige al sistema y almacena los datos en sesión o si no muestra un mensaje de error en la misma página.

Gracias a este Framework se pudo realizar las diferentes peticiones GET y POST, de esta manera se muestra correctamente las páginas web según lo que se requiere, además con este también se desarrolló el web service para que la aplicación móvil pueda conectarse con la base de datos y se mantenga la información sincronizada en estos dos sistemas.

4.3.2. Arquitectura del sistema móvil

Como el sistema móvil para Android posee diferentes componentes que un sistema web, se utilizó una arquitectura de tipo Modelo-Vista-Modelo de Vista (MVVM).

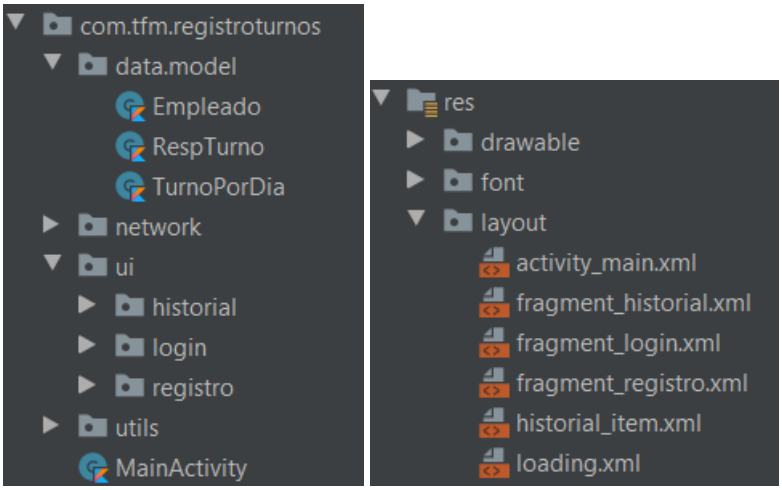

Figura 4.6: Arquitectura del sistema móvil

En la figura anterior se muestra los componentes, dentro de la carpeta model, como su nombre lo dice va los Modelos del sistema, en la carpeta ui se encuentra los Modelos de vista y otros componentes, y en la carpeta res, dentro de layout están todas las Vistas del sistema. También se encuentra la carpeta network, aquí se realizó la conexión con el webservice del sistema web, y las diferentes peticiones de tipo GET y POST para realizar las acciones dentro de la aplicación.

A diferencia del sistema web, no todos sus componentes están escritos en lenguaje Kotlin, puesto que, para realizar las vistas en Android, se debe utilizar lenguaje de etiquetado, es por esta razón que las extensiones de los archivos de la vista son XML.

- Modelo

El modelo es similar al sistema web, en él se específica los datos que poseen la clase, y como es escrito en el mismo lenguaje el resultado es igual.

```
@Parcelize
data class Empleado(
    var id: Int = -2,
    var nombres: String = "",
    var apellidos: String = "",
    var fechaNacimiento: String = "",
    var movil: String = "",
    var direccion: String = "",
    var identificacion: String = "",
    var clave: String = "",
    var rol: String = ""
) : Parcelable
```

Se utilizó la notación Parcelize para la librería que se usa para el webservice, que se hablará en puntos siguientes. Como se dijo el resultado es similar, es un data class con sus campos, lo único que se añadió son datos por defectos para que ninguno de ellos sea nulleable.

- Vista

Las vistas son creadas en lenguaje de etiquetado, son archivos XML, por esta razón no se explica mucho de él, puesto que, es similar a como se lo realiza en Java, a continuación, se muestra una parte de una vista.

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">
    <data>
        <variable
            name="loginViewModel"
            type="com.tfm.registroturnos.ui.login.LoginViewModel" />
    </data>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:id="@+id/container"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@drawable/fondo"
        tools:context=".ui.login.LoginFragment">


            <Button
    android:id="@+id/loginButton"
    android:layout_width="wrap_content"
    android:layout_height="51dp"
    android:text="@string/action_sign_in"
    android:textColor="#FFFFFF"
    android:textSize="14sp"
    android:onClick="@{() -> loginViewModel.onInicioSesion()}"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/passwordEditText" />

    </androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```

Aquí se puede apreciar un botón, que al seleccionarlo se dispara una acción, esta se encuentra en un View Model o modelo de vista, para que esto fuese posible, se debe insertar una etiqueta data, y dentro de ella una variable, describiendo su dirección y el nombre que se le asigna, gracias a esto se puede acceder a las funciones que posee el View Model dentro de una vista.

- Modelo de Vista

Con el ViewModel se puede mantener y administrar de una mejor manera los datos que están relacionados con la vista, dicho de otra forma, estos sobrevivirán a los cambios que pueden ocurrir durante un cambio en la configuración del sistema, porque si solo se utilizara fragmentos o actividades, en cualquier cambio se pueden destruir y los datos se pierden.

Por consiguiente, al utilizar los ViewModel se trabaja de manera eficiente con los datos utilizados en la interfaz, sin preocuparse de que estos se vayan a perder. El siguiente código es un ejemplo de cómo se lo utiliza.

```
class HistorialViewModel(val empleado: Empleado) : ViewModel() {
    private val _turnos = MutableLiveData<List<TurnoPorDia>>()
    val turnos: LiveData<List<TurnoPorDia>>
        get() = _turnos

    private val _mes = MutableLiveData<Int>()
    val mes: MutableLiveData<Int>
        get() = _mes

    fun getAllTurnos() {
        viewModelScope.launch {
            try{
                _turnos.value =
TurnoEmpleadoApi.retrofitService.getHistorialRegistros(empleado.id,
_mes.value!!)
            } catch (e: Exception) {
                _turnos.value = null
            }
        }
    }

    fun changeMes(value: Int) {
        _mes.value = value
    }
}
```

En este ViewModel se encuentran dos variables y dos funciones, estas variables son del tipo MutableLiveData, esto quiere decir que son LiveData los cuales se puede modificar sus valores, los LiveData son clases que utilizan el patrón de diseño observer que sirven para cuando ocurre algún cambio de datos, el primero es para almacenar todos los turnos diarios que existen en el mes, el segundo para guardar el mes elegido dentro de la aplicación.

La primera función sirve para obtener los datos del historial de registro del empleado del mes elegido y almacenarlos en la variable _turnos, y la segunda es para cambiar el valor de la variable _mes.

Estas variables y funciones son utilizadas en el Fragment, como muestra esta parte del código:

```
if(historialViewModel.mes.value == null){
    historialViewModel.changeMes(LocalDate.now().monthValue)
}

historialViewModel.mes.observe(viewLifecycleOwner, {
    historialViewModel.getAllTurnos()
})

binding.mesRegistro.onItemSelectedListener = object : OnItemSelectedListener{
    override fun onNothingSelected(parent: AdapterView<*>?) {}

    override fun onItemSelected(parent: AdapterView<*>?, view: View?,
position: Int, id: Long) {
        historialViewModel.changeMes(position+1)
    }
}
```

Primero se llama a la función changeMes del ViewModel, para asignar el mes actual, si no existe ningún mes asignado a él, segundo se añade un observable a la variable mes, de esta manera cuando exista algún cambio en esta se ejecutará la función getAllTurnos, y tercero cuando el usuario seleccione un mes diferente se realiza la misma acción que en el primer paso, pero ahora con el mes seleccionado; así, si se cambia el mes se dispara el observable y realiza la función getAllTurnos.

En síntesis, si se utilizará solo el Fragment, al momento de utilizar la aplicación y cambiar de mes para ver el historial de registro se mostrará todos los datos correctos, pero si se cambia la rotación de la pantalla estos datos se perderán y volverá a mostrar los registros del mes actual y no del que se seleccionó, porque el Fragment se vuelve a ejecutar y todo se muestra como en un inicio; en cambio con el ViewModel los datos se almacenan en él así exista cambios en el sistema de este tipo, de esta manera se asegura que se muestre todo correctamente.

4.4. Desarrollo de las aplicaciones

A pesar de que en las dos arquitecturas ya se habló de manera breve ciertas características de cómo se programó, se escribió este punto para describir más sobre el desarrollo que se realizó en los dos sistemas.

4.4.1. Desarrollo del sistema web

Se escribe de los siguientes temas para entender cómo funcionan y para qué han sido implementados en el sistema.

- Ktor

Es un Framework basado enteramente en Kotlin, facilita la creación de aplicaciones web con un mínimo esfuerzo y su principal objetivo es crear conexiones entre web services, aplicaciones web y móviles; al estar basado en Kotlin toma sus principales herramientas y funciones [5] [11].

Como se mencionó en la arquitectura del sistema, Ktor posee una herramienta llamada Routing, gracias a esta se puede estructurar y simplificar las diferentes solicitudes que existen dentro del sistema [33].

```
routing {
    get("/") {
        call.respondText("Hola Mundo")
    }
    get("/bye") {
        call.respondText("Adios Mundo!")
    }
}
```

Por ejemplo, en este código existen dos rutas, la primera se dirigirá cuando se acceda a la dirección del sistema, y existirá una respuesta solo de texto que dice “Hola Mundo”, en el segundo cuando se ingrese a la ruta “/bye” y responderá el segundo mensaje. De esta manera se puede realizar la estructura de las solicitudes del sistema, además de utilizar el método Get, se puede usar Post, Put y Delete; así también, las respuestas que se muestran al acceder a estas direcciones no solo deben ser texto, también se puede responder con contenido Html, o para el caso de crear un web service, se responde en formato Json, o si no solo llamar a una función interna en el sistema.

Si en las solicitudes se recibe parámetros ya sea por tipo Get o por Post, se puede acceder a ellos solo con su nombre, de la siguiente manera:

```
post(Endpoints.SEARCHEMPLEADO.url) {
    val empleado = parameters["busqueda"]
}
get(Endpoints.REMOVEEMPLEADO.url) {
        val identificacion = call.request.queryParameters["id"]
}
```

Con Ktor se pudo crear toda esta estructura del sistema de forma fácil y eficiente, con el uso del mismo lenguaje Kotlin; gracias a este Framework se desarrolla sistemas, módulos o web services de forma rápida sin tener que aprender mucho, basta con crear todo el enrutamiento y sus diferentes respuestas.

- Exposed

Es un Framework para realizar la conexión y diferentes acciones con una base de datos, se utilizó esta herramienta porque actúa de la misma manera sin importar el motor de base de datos, o si se reemplaza con algún otro motor, el cambio en el código es mínimo, además de que aprenderlo es muy sencillo y se adapta bastante bien a Kotlin.

Realizar la conexión es muy sencillo, en la función principal hay que utilizar este código:

```
Database.connect(
    "jdbc:mysql:///turnos_empleados?cloudSqlInstance=registro-
turnos:us-east1:uah-
tfm&socketFactory=com.google.cloud.sql.mysql.SocketFactory",
    user = "root",
    password = "*******",
    driver = "com.mysql.jdbc.Driver"
)
```

En donde se ingresa la dirección de la base de datos, seguido del usuario, la contraseña y el tipo de driver que se usa, en este caso se utilizó MySql, depende del motor de base de datos para realizar la conexión, para cada uno es diferente.

Una vez añadido esta línea de código, para este proyecto, las acciones que hacen peticiones a la base de datos se encuentran en los controladores, ahí se realizan las operaciones básicas de Crear, Leer, Actualizar y Borrar. Para realizar cualquier acción se debe hacerlo dentro de la función transaction:

```
transaction {

}
```

Para demostrar todas estas acciones se muestra pedazos de código del controlador del empleado:

-Creación:

```
transaction {
    result = EmpleadoTable.insert {
        it[nombres] = empleado.nombres
        it[apellidos] = empleado.apellidos
        it[fechaNacimiento] = empleado.fechaNacimiento
        it[movil] = empleado.movil
        it[direccion] = empleado.direccion
        it[identificacion] = empleado.identificacion
        it[clave] = empleado.clave
        it[rol] = empleado.rol
    }[EmpleadoTable.id]
}
```

Aquí se debe asignar los datos a cada campo del Empleado para poder ingresar uno nuevo, y al final se puede almacenar en una variable el id con el que se guardará.

-Actualización:

```
transaction {
    result = EmpleadoTable.update({ EmpleadoTable.identificacion eq
empleado.identificacion }) {
        it[nombres] = empleado.nombres
        it[apellidos] = empleado.apellidos
        it[fechaNacimiento] = empleado.fechaNacimiento
        it[movil] = empleado.movil
        it[direccion] = empleado.direccion
        it[identificacion] = empleado.identificacion
        it[clave] = empleado.clave
    }
}
```

El actualizar es similar al crear, igual se debe asignar datos a cada campo de la tabla, y al inicio, cuando se llama a la función, hay que ingresar la regla del Where, en este caso el campo identificación debe ser igual al de un empleado que se encuentre almacenado, aquí se utiliza la palabra eq para referirse a "igual a"; estas expresiones, así como los condicionales se encuentran en el repositorio de GitHub de esta librería:

https://github.com/JetBrains/Exposed/wiki/DSL#where-expression

-Borrado:

```
transaction {
    result = EmpleadoTable.deleteWhere { EmpleadoTable.identificacion
eq empleado.identificacion }
}
```

Para el borrar solo hay que ingresar las reglas del Where que se requiera, para este ejemplo solo se necesitó que la identificación coincida.

-Leer:

```
transaction {
    empleado =
EmpleadoTable.selectAll().orderBy(EmpleadoTable.apellidos to
SortOrder.ASC)
        .map { it.toEmpleadoModel() }
}
```

Esta acción se debe ingresar las reglas según lo que se quiera obtener de la base de datos, en este caso, no se insertó ninguna regla para el Where porque se desea tener todos los Empleados que existan, pero si se ordenó por el apellido de forma ascendente, el resultado se almacena en una lista del tipo Empleado.

Con Exposed todas estas acciones son fáciles de realizar y de forma sencilla, sin complicaciones, se mantiene el objetivo de Kotlin que es hacer mucho con pocas líneas de código y de manera eficiente, para ocupar estas funciones, basta con llamar al controlador y elegir la función que se desea realizar e ingresar los datos que se requiera.

- Interfaz

Para tener un mejor diseño de la interfaz de la aplicación web se utilizó Bootstrap 4.5.2, en específico el ejemplo que se tomó como referencia para la creación de este proyecto fue el Dashboard que ofrece Bootstrap:

https://getbootstrap.com/docs/4.5/examples/dashboard/

Este fue tomado como plantilla principal y a partir de él se realizó las modificaciones necesarias para que se adecúe a las necesidades del sistema. Los enlaces para insertar Bootstrap en la aplicación se encuentran en la plantilla Cabecera:

```
link(
    rel = "stylesheet",
    href =
"https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.
css"
) {
    this.integrity = "sha384-
JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"
    this.attributes["crossorigin"] = "anonymous"
}
link(
    rel = "stylesheet",
    href = "https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css"
)
script(src = "https://code.jquery.com/jquery-3.5.1.slim.min.js") {
    defer = true
    integrity = "sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
    this.attributes["crossorigin"] = "anonymous"
}
script(src =
"https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
) {
    defer = true
    integrity = "sha384-
9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN"
    this.attributes["crossorigin"] = "anonymous"
}
script(
    src =
"https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.j
s"
) {
    defer = true
    integrity = "sha384-
B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV"
    this.attributes["crossorigin"] = "anonymous"
}
```

Las capturas de pantalla de la interfaz se encuentran en el capítulo 5, en la parte que se describe el funcionamiento del sistema web, ahí se puede ver el resultado final de esta aplicación.

Además de utilizar Bootstrap, se usó el código creado por [34] para el reloj digital que se muestra al momento de registrar un turno:

Figura 4.7: Reloj digital

Este se actualiza automáticamente según la hora actual a través de código JavaScript, además se añadió un poco de código para que también actualice la hora y fecha de un formulario creado en el registro de turnos, para que todos esos datos se envíen y almacenen en la base de datos.

4.4.2. Desarrollo del sistema móvil

Así como para la aplicación web, también se detalla algunas características que se utilizaron en el desarrollo móvil.

- Corrutinas

Las corrutinas son patrones de diseño utilizadas en Android para cuando se quiere ejecutar de forma asíncrona un código, su uso es muy simple; además ayuda en la productividad porque mejora la administración de las tareas de larga duración sin afectar a los procesos principales [34].

Para utilizarlo es muy fácil, en este proyecto todas las corrutinas se encuentran en los ViewModel, como se muestra en el siguiente código:

```
fun getAllTurnos() {
    viewModelScope.launch {
        try{
            _turnos.value =
TurnoEmpleadoApi.retrofitService.getHistorialRegistros(empleado.id,
_mes.value!!)
        } catch (e: Exception) {
            _turnos.value = null
        }
    }
}
```

Está es una función para obtener todos los turnos del mes seleccionado, y para identificar que es una corrutina se puede ver que está dentro de este código:

```
viewModelScope.launch {
}
```

Todo lo que este dentro de esta función se realizará de manera asíncrona, así el resto de los procesos continuarán en ejecución sin tener que esperar a que termine lo que este dentro de la corrutina.

- Retrofit y Moshi

Retrofit es una librería que permite a una aplicación móvil realizar solicitudes a un web service de tipo Rest; como las respuestas son de tipo Json se necesitó de otra librería para poder convertir en objetos lo enviado por el web service, esta se llama Moshi.

Para que esto sea posible se utilizó el siguiente código:

```
private const val BASE_URL =
    "https://tfmregistroempleados.com/rest/"

private val moshi = Moshi.Builder()
    .add(KotlinJsonAdapterFactory())
    .build()

private val retrofit = Retrofit.Builder()
    .addConverterFactory(MoshiConverterFactory.create(moshi))
    .baseUrl(BASE_URL)
    .build()
```

Aquí se define la URL del web service con el que se realiza la conexión, luego se define dos variables, la primera para indicar que se utilizará el convertidor de Json, Moshi, y el segundo para realizar la construcción de Retrofit.

Una vez hecho esto se escribe las solicitudes que se va a realizar:

```
interface TurnoEmpleadoApiService {
    @FormUrlEncoded
    @POST("login")
    suspend fun getEmpleadoLogin(
        @Field("identificacion") identificacion: String,
        @Field("clave") clave: String
    ): Empleado

    @GET("historialregistros")
    suspend fun getHistorialRegistros(
        @Query("id") id: Int,
        @Query("m") mes: Int
    ): List<TurnoPorDia>

    @FormUrlEncoded
    @POST("guardarturno")
    suspend fun saveTurno(
        @Field("id") id: Int,
        @Field("fecha") fecha: String,
        @Field("hora") hora: String
    ): RespTurno
}
```

Para esta aplicación solo existen tres, el primero es una petición POST que realiza una consulta a ver si el usuario existe en la base de datos en el momento que quiere iniciar sesión, cuando es una petición POST se escribe la anotación @FormUrlEncoded si se envía algún parámetro, la respuesta de esta será de tipo Empleado.

El segundo es una petición GET para obtener todos los registros del mes del empleado que inició sesión, la respuesta será una lista de tipo TurnoPorDia, y como es una petición GET, basta con ingresar los parámetros que se enviarán.

Y el último es de tipo POST el cual guarda un turno, envía tres parámetros y la respuesta es de tipo RespTurno.

Para los parámetros usados en una petición GET se debe utilizar la palabra Query y para los de tipo POST se usa Field.

Ahora solo se debe crear un objeto para inicializar los servicios de Retrofit y al cual pueda ser llamado para acceder a las peticiones que han sido creadas.

```
object TurnoEmpleadoApi {
    val retrofitService: TurnoEmpleadoApiService by lazy {
        retrofit.create(TurnoEmpleadoApiService::class.java)
    }
}
```

Cuando se habló de los ViewModel se describió que en ellos debe ir los datos que están relacionados con la interfaz, es por esta razón que las llamadas a las peticiones se realizaron en los ViewModel, dentro de las corrutinas, como por ejemplo la siguiente función:

```
fun getAllTurnos() {
    viewModelScope.launch {
        try{
            _turnos.value =
TurnoEmpleadoApi.retrofitService.getHistorialRegistros(empleado.id,
_mes.value!!)
        } catch (e: Exception) {
            _turnos.value = null
        }
    }
}
```

Como puede ocurrir algún problema en el momento que se realiza la conexión con el web service, el código debe ir dentro de un try, si todo salió correcto, se guarda en una variable el resultado obtenido de la petición, para llamarlo basta con ingresar el nombre del objeto y de la variable creada en el objeto, de esta manera se puede acceder a las peticiones e ingresar los parámetros requeridos, si falla la petición, en el catch se ingresa el código necesario para demostrar que hubo un problema.

Una vez finalizada la parte del desarrollo y conocido las características y tecnologías que han sido utilizadas en las dos aplicaciones, en el siguiente capítulo se va a hablar sobre el funcionamiento de estos dos sistemas.

5. FUNCIONAMIENTO

Para terminar, se va a hablar sobre el funcionamiento de la aplicación web y móvil, con la finalidad de conocer el alcance que poseen estos sistemas con las herramientas que se desarrollaron y hablaron en el anterior capítulo.

5.1. Funcionamiento del sistema web

Para entrar al sistema web, se debe acceder a esta dirección https://tfmregistroempleados.com/login e ingresar las credenciales correspondientes, para entrar como administrador la identificación y contraseña son admin.

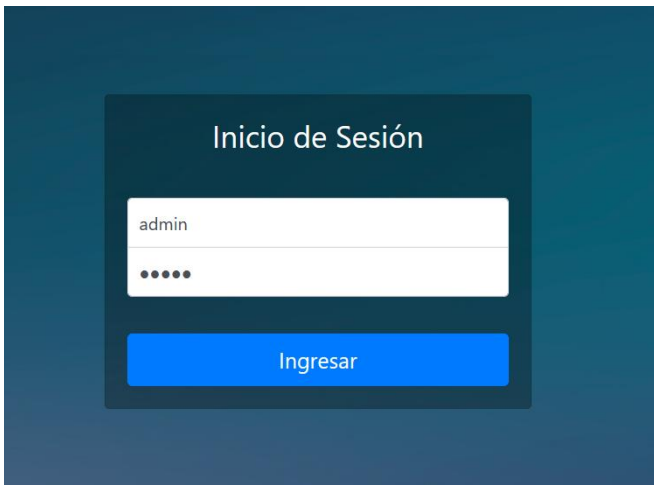

Figura 5.1: Página de inicio de sesión

Como administrador del sistema se puede realizar varias tareas dentro de la aplicación, todas ellas son detalladas en los siguientes puntos:

- Listado de empleados

Una vez ingresado al sistema, aparece la página con el listado de empleados que posee la empresa.

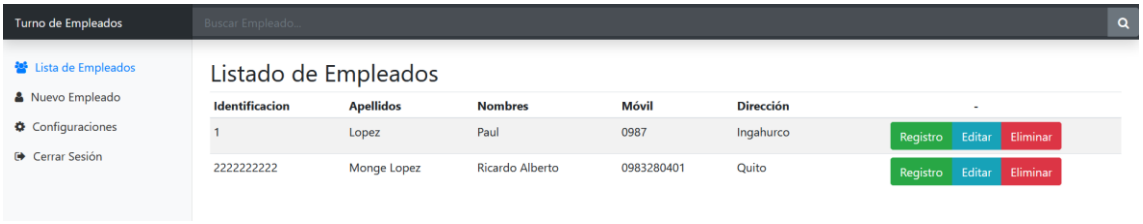

Figura 5.2: Página de lista de empleados

En la parte de la izquierda se ve las acciones generales que puede realizar el administrador, en la parte superior se realiza una búsqueda de los empleados, y dentro

del listado para cada empleado se puede realizar 3 acciones, ver el registro de turnos, editar los datos o eliminar.

- Nuevo empleado

Para ingresar un nuevo empleado hay que dar clic en la opción de la parte izquierda, una vez dentro se debe escribir todos los campos requeridos y guardar.

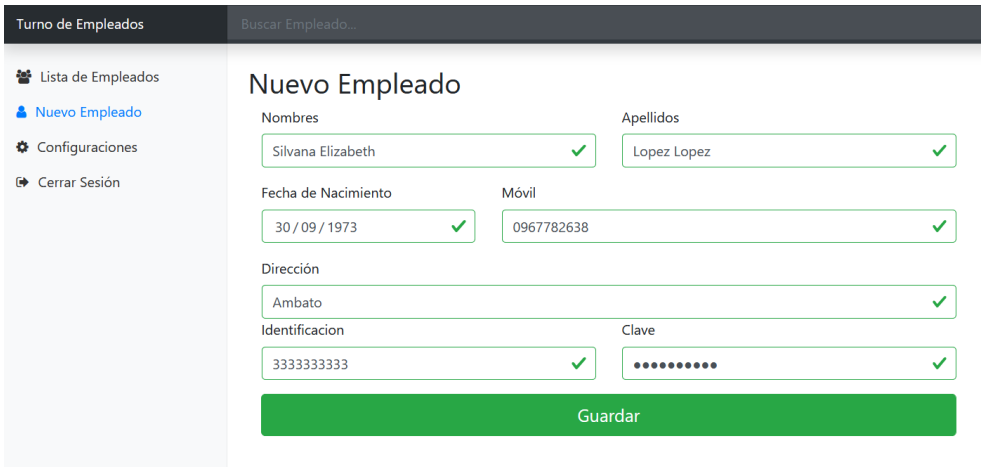

Figura 5.3: Página de nuevo empleado

Si todos los campos fueron llenados se guardará con éxito el nuevo empleado, y se podrá ver en la lista de empleados.

- Editar empleado

Para poder editar, en la lista se debe elegir un empleado y dar clic en su opción que dice editar, aquí al igual que en la página de nuevo empleado, se debe completar todos los campos y dar clic en guardar para actualizar los datos.

Figura 5.4: Página de editar empleado

- Eliminar empleado

En la lista de empleados, hay que elegir a quien se desea eliminar y dar clic en su respectivo botón que dice Eliminar, una vez hecho aparecerá este mensaje y desaparece el empleado eliminado de la base de datos.

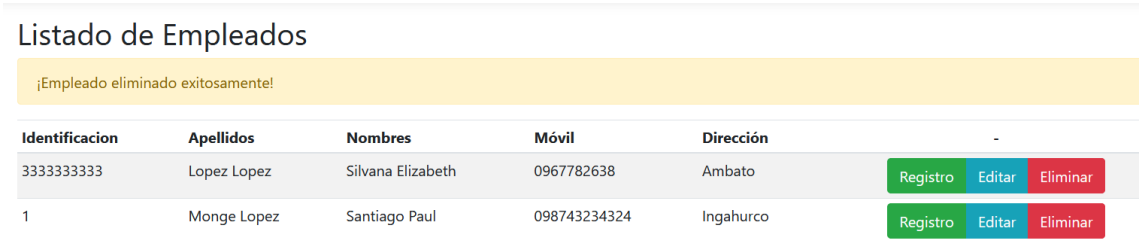

Figura 5.5: Eliminar empleado

- Buscar empleado

En la parte superior se encuentra la opción para buscar empleados, se puede buscar por cualquier campo que se muestra en la tabla del listado, solo hay que ingresar el dato y aplastar enter o dar clic en la lupa del lado izquierdo.

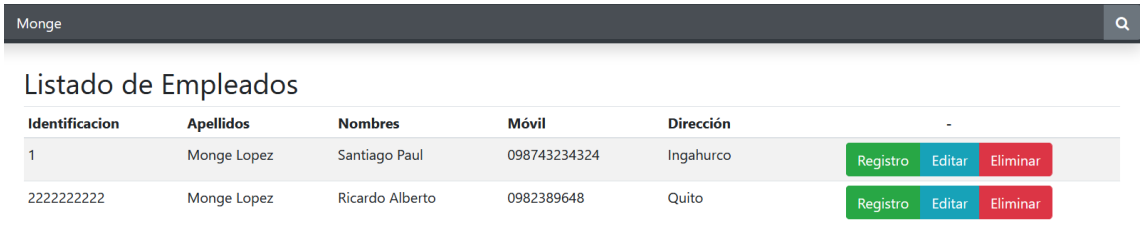

Figura 5.6: Buscar empleado

- Listado de registro de turnos de empleado

Cada empleado posee su registro de turnos, para poder verlo solo hay que elegir uno y dar clic en el botón Registro, al inicio se muestra el mes actual por defecto, si se quiere cambiar de mes, en la lista de meses que se encuentra al lado derecho se da clic y se selecciona el otro mes que se quiere ver.

Figura 5.7: Página de listado de turnos de empleado

Aquí se verá la fecha, la hora de entrada, de salida de la comida, entrada de la comida, salida y el total del día en horas, adicional hay un botón para editar cada día de registro.

- Editar turno de empleado

Para poder editar un turno se debe seleccionar el día que se quiera editar en el listado anterior, una vez dentro se ingresa todas las horas correctamente, hay que fijarse que tengan un orden lógico, no que la última hora, la de salida sea más temprano que la hora de entrada, si no se mostrará el siguiente error.

Figura 5.8: Error al ingresar mal las horas del turno

Si todo es correcto se guarda sin ningún problema todas las horas y se actualiza el registro de este día.

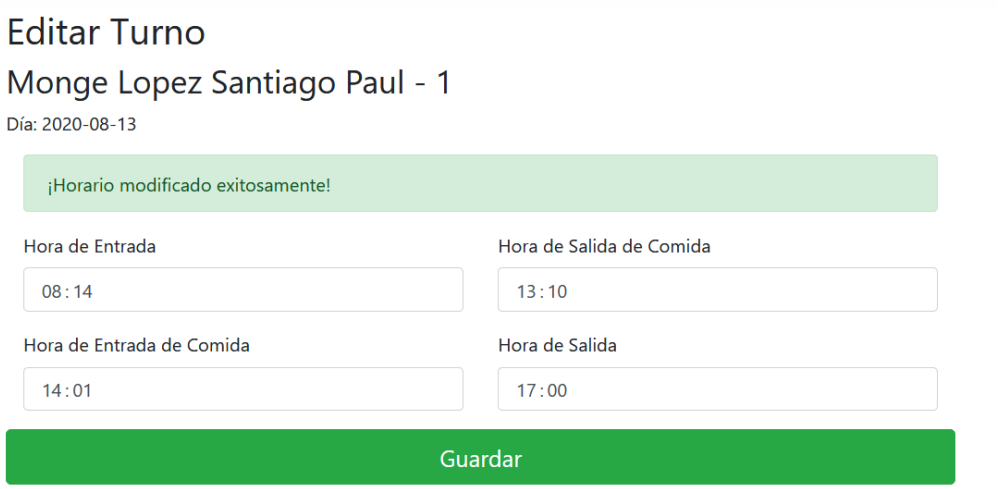

Figura 5.9: Editar turno de empleado

- Editar horario de registros

También existe la posibilidad de cambiar el horario el cual los empleados realizan el registro a diario, para ellos hay que dar clic en la opción Configuraciones del lado izquierdo de la página, ahí se dirige a las horas guardadas por defecto, al igual que en la actualización del turno de un empleado, para guardarlo correctamente debe cumplir con un orden adecuado.

Figura 5.10: Página de configuraciones de horario

Esas son todas las acciones que puede realizar el administrador en el sistema, ahora falta por ver lo que puede realizar un empleado, esto se describe en los siguientes puntos, para acceder se utiliza el mismo enlace y la identificación y contraseña es 1111111111.

- Registro de turnos

Luego de iniciar sesión, la página de registro de turnos aparece, se muestra la hora actual para que el empleado pueda registrarse, para ello solo se debe dar clic en el botón que dice Registrarse, si lo hace en las horas correspondientes se verá el siguiente mensaje:

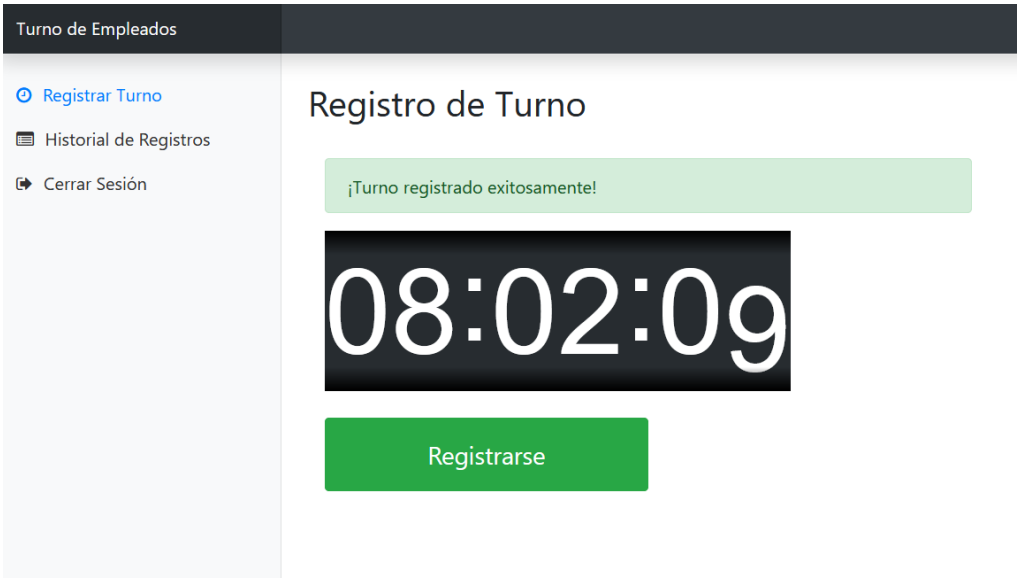

Figura 5.11: Página de registro de turno

Para que el registro sea exitoso, debe hacerlo solo 15 minutos antes y hasta 15 minutos después de cada horario del día, si lo hace fuera de horario se muestra este mensaje:

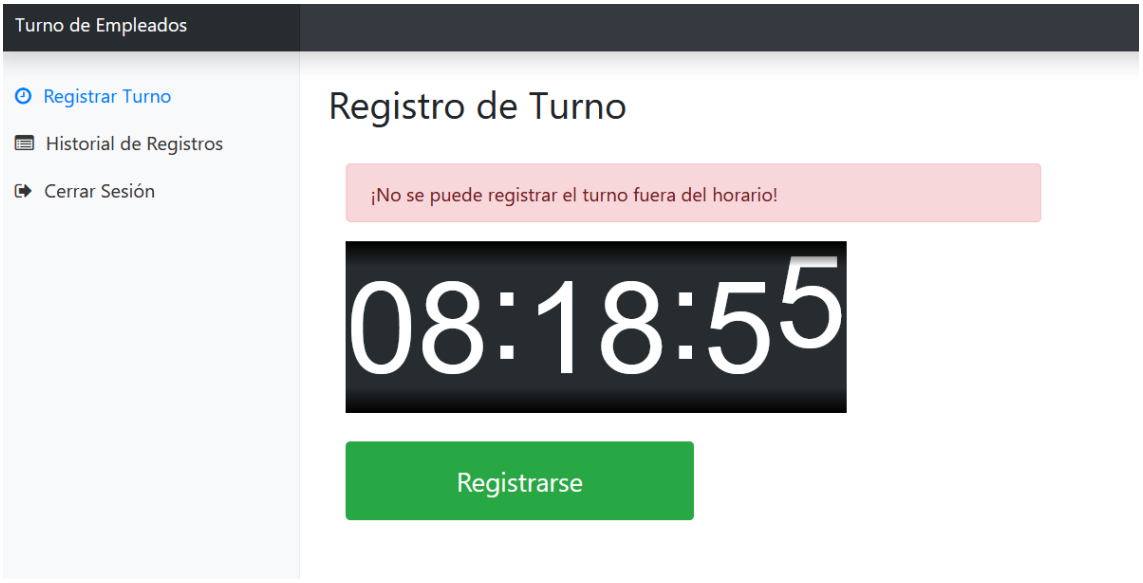

Figura 5.12: Registro de turno fuera de horario

También puede aparecer otro mensaje si ya se ha registrado con anterioridad al mismo horario, y el empleado lo quiere hacer de nuevo, ahí aparece este mensaje:

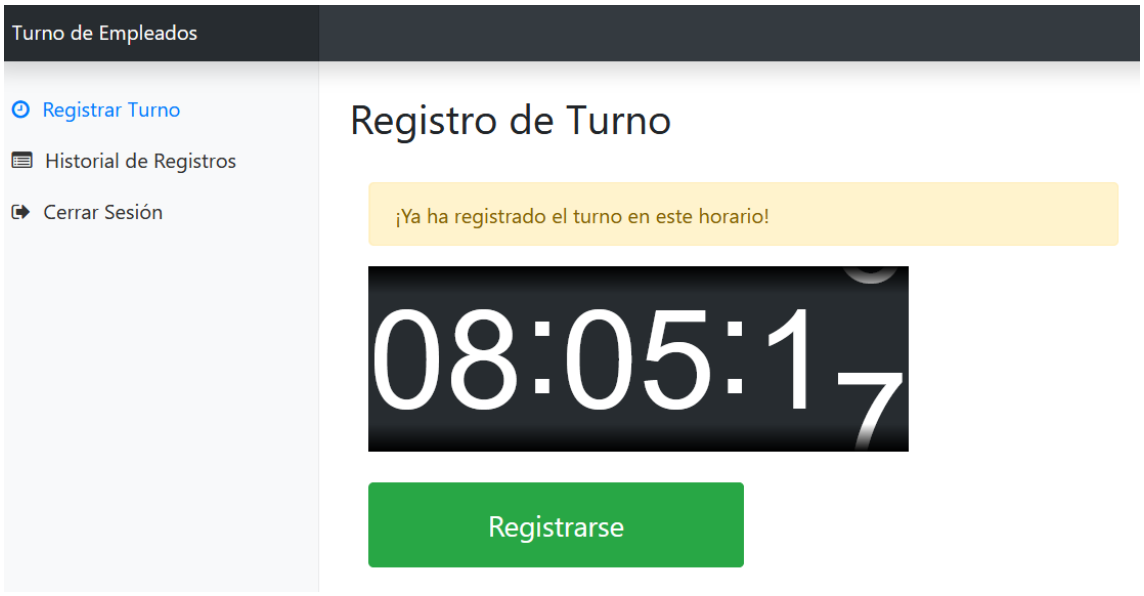

Figura 5.13: Registro de turno repetido

- Historial de registros

Al igual que en la parte del administrador, se verá los mismos datos del historial de registros, con la diferencia que aquí solo será del empleado que inicio sesión, así mismo puede seleccionar el mes que quiera ver o por defecto el actual.

Figura 5.14: Historial de registro personal

5.2. Funcionamiento del sistema móvil

Las funciones del sistema móvil son las mismas que la parte del empleado en el sistema web, por esta razón la aplicación solo se puede acceder a través de las credenciales de un empleado; en la pantalla de inicio de sesión se accede con 1111111111 para la identificación y la contraseña, luego se selecciona Iniciar Sesión para acceder al sistema.

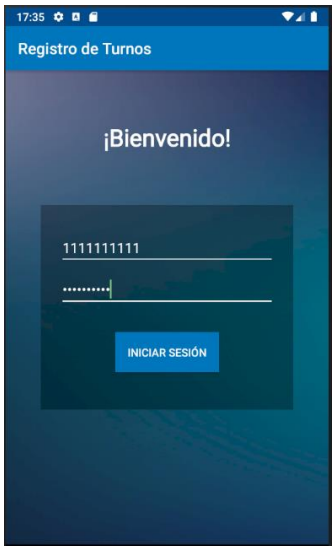

Figura 5.15: Aplicación móvil - Inicio de sesión

- Registro de turnos

La primera pantalla que aparece luego de iniciar sesión es la de registro de turnos, el empleado debe registrarse a la hora correcta según el horario laboral, los mensajes de error son similares a la aplicación web si se ingresa fuera de horario o se quiere registrar nuevamente en el mismo horario. Para registrarse basta con pulsar el botón que dice Registrar Turno.

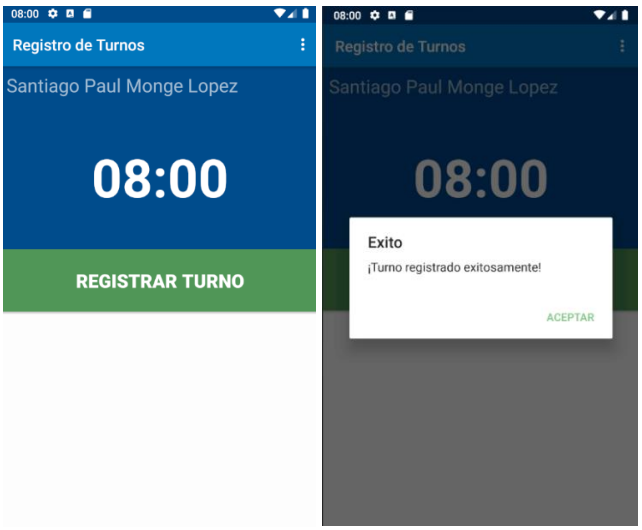

Figura 5.16: Aplicación móvil - Registro de turno

- Historial de registros

Para acceder al historial de registros hay que seleccionar los tres puntos que se encuentra en la parte superior derecha de la aplicación, luego la opción Historial; una vez dentro aparece los registros que se realizaron en el mes, así mismo se puede cambiar de mes para ver fechas anteriores.

Figura 5.17: Aplicación móvil - Historial de registros

Para regresar al registro hay que presionar en los tres puntos y seleccionar Registro, y si se quiere cerrar sesión se selecciona Cerrar Sesión o presionar el botón de atrás del móvil.

Figura 5.18: Aplicación móvil – Opciones

Una vez terminado con la explicación del funcionamiento, en el siguiente capítulo se describe las conclusiones obtenidas con el desarrollo de este proyecto.

6. CONCLUSIONES

Al terminar con el desarrollo de las aplicaciones web y móvil, ver su funcionamiento y haber analizado los aspectos que posee Kotlin para hacer de él un moderno, novedoso y apreciado lenguaje de programación, se confirmó lo que JetBrains y los desarrolladores que utilizan Kotlin dicen sobre él; por esta razón se dedujo que su principal ventaja es ser simple y eficiente, gracias a esto facilita el aprendizaje para los programadores y se pueden adaptar a él rápidamente, además existe varios cursos que se encuentra en su sitio web oficial para poder aprender sobre él.

Se dice que es un lenguaje simple porque a diferencia de otros, con él no hay que cumplir ciertas lógicas de programación que por lo general dan como resultado demasiadas líneas de código innecesarias y en muchas ocasiones un programa pequeño con pocas funciones se convierte en mucho código; con Kotlin su desarrollo es más fácil, solo se escribe lo necesario y el programa realiza las mismas acciones sin ninguna complicación, por eso se convierte en un lenguaje eficiente.

Su lógica de programación es amigable, desde crear una variable hasta realizar toda una función, un microservicio, un web services o cualquier cosa que demande más tiempo desarrollarlo, es muy sencillo de realizar, una vez que se aprende la sintaxis del lenguaje el resto se avanza de manera rápida y de esta forma Kotlin puede ser usado cuando es necesario lanzar un programa en el menor tiempo posible.

Además, como se está volviendo un lenguaje de programación apreciado por la comunidad de desarrolladores, se empieza a crear librerías que se adaptan a las necesidades de cada uno, como las que se utilizaron en el proyecto, a parte de ellas existen muchas otras más que ayudan en diferentes funciones.

Por otro lado, al tener oficialmente el apoyo de Google como lenguaje de desarrollo para Android, Kotlin sigue en constante crecimiento dentro de este entorno, uno de estos ejemplos es la creación de la librería Jetpack Compose que fue realizada en Kotlin y se compila igual a través de él; esto quiere decir que Google aprovecha de las ventajas que posee este lenguaje para crear nuevas herramientas y tuvo un buen recibimiento por parte de ellos.

Otra de sus ventajas es que se adapta a cualquier entorno, como pasó con este proyecto, se desarrolló dos aplicaciones una para web y otra para móvil, las dos cumplen casi con las mismas funciones y gracias a Kotlin se pudo desarrollar sin ninguna complicación y necesidad de utilizar otros lenguajes, basta con las librerías que ofrece y sus herramientas, además se pudo crear un web services para que la aplicación móvil se

pueda comunicar con la base de datos; como consecuencia, esto ayuda a que con el aprendizaje de un solo lenguaje de programación se pueda realizar aplicaciones completas que interactúen entre sí.

En conclusión, Kotlin es un buen lenguaje de programación, cumple con todas las expectativas que se quería comprobar, su sintaxis y lógica de programación lo convierten en un lenguaje competente y gracias a su capacidad de adaptarse a cualquier entorno, hace que valga la pena aprender a utilizarlo y crear aplicaciones con él. Es cierto que aún no es tan completo como lo es Java porque es un lenguaje nuevo en comparación con el resto, pero con la posibilidad de trabajar con Java y viceversa se puede crear grandes aplicaciones con él, sin embargo, al usar solo Kotlin ya se puede desarrollar un sinfín de programas y este proyecto es una prueba de eso.

7. BIBLIOGRAFÍA

[1] T. Grin, "Kotlin programming language," [Online]. Available: https://kotlinlang.org/assets/kotlin-media-kit.pdf. [Accessed 08 Junio 2020].

[2] "JetBrains," [Online]. Available: https://www.jetbrains.com/company. [Accessed 08 Junio 2020].

[3] J. D. Luján Castillo, "Aprende a programar con Kotlin," 20 Septiembre 2017.

[4] N. Smyth, Kotlin / Android Studio 3.0 Development Essentials - Android, Octava ed., 2017.

[5] N. Ebel, Mastering Kotlin, Packt Publishing, 2019.

[6] "Kotlin Blog," [Online]. Available: https://blog.jetbrains.com/kotlin. [Accessed 08 Junio 2020].

[7] M. Devcic, Kotlin Quick Start Guide, Packt Publishing, 2018.

[8] V. Subramaniam, Programming Kotlin, Pragmatic Bookshelf, 2019.

[9] J. Skeen and D. Greenhalgh, Kotlin Programming: The Big Nerd Ranch Guide, Primera ed., Big Nerd Ranch Guides, 2018.

[10] D. Jemerov and S. Isakova, Kotlin in Action, Manning Publications, 2017.

[11] A. Belagali, A. Chordiya and H. Trivedi, Kotlin Blueprints, Packt Publishing, 2017.

[12] "Android Developers," [Online]. Available: https://developer.android.com/kotlin. [Accessed 12 Junio 2020].

[13] A. Developers, "Android Developer Story: Zomato uses Kotlin to write safer," 03 Octubre 2018. [Online]. Available: https://youtu.be/ao9QvtpszOU. [Accessed 12 Junio 2020].

[14] "JetBrains," [Online]. Available: https://www.jetbrains.com/es-es/company/partners-technology/. [Accessed 12 Junio 2020].

[15] J. Friesen, Learn Java for Android Development, Tercera ed., Apress, 2014.

[16] "Oracle," [Online]. Available: https://www.oracle.com/java. [Accessed 15 Junio 2020].

[17] I. Galata, "Kotlin VS Java: Basic Syntax Differences," [Online]. Available: https://yalantis.com/blog/kotlin-vs-java-syntax. [Accessed 15 Junio 2020].

[18] S. Bose, A. Kundu, M. Mukherjee and M. Banerjee, "A comparative study: Java vs Kotlin programming in Android application development," *International Journal of Advanced Research in Computer Science,* vol. IX, no. 3, Mayo - Junio 2018.

[19] "Kotlin - Basic Syntax," [Online]. Available: https://kotlinlang.org/docs/reference/basic-syntax.html. [Accessed 19 Junio 2020].

[20] M. Karch, Android Tablets Made Simple, Apress, 2011.

[21] "Android," [Online]. Available: https://www.android.com/intl/es_es/everyone/enabling-opportunity. [Accessed 22 Junio 2020].

[22] A. Hoog, Android Forensics, Syngress, 2011.

[23] "Android Developers," [Online]. Available: https://developer.android.com/about/dashboards. [Accessed 23 Junio 2020].

[24] "Android Developers - Android Oreo," [Online]. Available: https://developer.android.com/about/versions/oreo/android-8.0. [Accessed 23 Junio 2020].

[25] "Android Developers - Android 9 Pie," [Online]. Available: https://developer.android.com/about/versions/pie/android-9.0. [Accessed 23 Junio 2020].

[26] "Android Developers - Android 10," [Online]. Available: https://developer.android.com/about/versions/10/highlights. [Accessed 23 Junio 2020].

[27] M. Moskala and I. Wojda, Android Development with Kotlin, Packt Publishing, 2017.

[28] J. Horton, Android Programming with Kotlin for Beginners, Packt Publishing, 2019.

[29] "Android Developers," [Online]. Available: https://developer.android.com/kotlin/first. [Accessed 25 Junio 2020].

[30] "Android Developers," [Online]. Available: https://developer.android.com/jetpack/compose. [Accessed 22 Julio 2020].

[31] "Android Developers," [Online]. Available: https://developer.android.com/kotlin/get-started. [Accessed 26 Junio 2020].

[32] "Android Developers," [Online]. Available: https://developer.android.com/studio/intro. [Accessed 26 Junio 2020].

[33] JetBrains, "Ktor 1.4.0," 18 Septiembre 2020. [Online]. Available: https://ktor.io/docs/quickstart-index.html. [Accessed 26 Noviembre 2020].

[34] F. D. Montis, "Codepen," 13 Septiembre 2017. [Online]. [Accessed 8 Diciembre 2020].

[35] "Android Developers," 01 Octubre 2020. [Online]. Available: https://developer.android.com/kotlin/coroutines. [Accessed 1 Diciembre 2020].