

Ciclo formativo:	Desarrollo de Aplicaciones Multiplataforma
Curso académico:	2020/2021
Nombre:	Cristian
Apellidos	Rodriguez Abadia
Título del trabajo	Tres en raya online wars KOTLIN:Una comparación práctica

## Agradecimientos

Me gustaría agradecer el proyecto especialmente a mi madre y mi hermano por su apoyo incondicional y la ayuda prestada en todo momento.

A las diferentes comunidades de desarrollo de telegram especialmente a offtopicssince2017, Kotlin ES, programadores y desarrolladores android por orientarme en algunos aspectos.

A los profesores y profesoras del IES Virgen de la Paloma por su ayuda y consideración teniendo en cuenta la situación actual han sido muy importantes.

Al Cisco por darme mi primer acercamiento a la programación.

# Índice

<b>Agradecimientos</b>	<b>2</b>
<b>Índice</b>	<b>3</b>
<b>Introducción</b>	<b>5</b>
<b>Desarrollo del proyecto</b>	<b>7</b>
Objetivos	7
Objetivos generales	7
Objetivos específicos	7
Marco Teórico	8
Introducción	8
Introducción a Kotlin	8
Introducción a programación en Kotlin	10
Variables	10
Estructuras de control	12
if	12
When	13
Bucles	13
While	13
For	14
Funciones	15
Programación Orientada a Objetos (POO)	16
Clases	16
Data Class	17
Puntos de discusión Java contra kotlin	18
Introducción a Java	18
Laboral	19
Seguridad	19
Velocidad	20
Comunidad	20
Comparación con el anterior proyecto	21
Programación general	21
Programación Funcional	26
Programación Orientada a Objetos	27
Programación en Android	31
Implementación con bibliotecas externas	33
Conclusión	38
Desarrollo de la prueba de concepto	39
Requisitos	39
Funcionales	39
Diagrama de casos de uso	39
No funcionales	41

Base de datos	42
Introducción	42
Introducción a firebase	42
Esquema	43
Implementación	44
Firebase	44
Models	44
Clases Provider	46
Control referencial	49
Imágenes de las colecciones en firebase	51
Implementación de las notificaciones administrativas enviadas desde firebase	54
Shapreferens	56
Implementación	56
Create/Update	56
Select	57
Diseño	59
Implementaciones reseñables	59
Requisitos por activity	65
Comparación con el anterior proyecto	79
Metodología	85
Ciclo de vida	85
Lista de materiales y equipos	86
Resultados	86
<b>Conclusión</b>	<b>87</b>
<b>Líneas futuras</b>	<b>88</b>
<b>Referencias</b>	<b>89</b>
<b>Anexos</b>	<b>90</b>
Anexo 1 Comparación con otras tecnologías	90
Kotlin vs Python	91
Kotlin vs Flutter	92
Kotlin vs Javascript	93
Kotlin vs React Native	94
Kotlin vs Ionic	95

## Introducción

Es muy fácil recurrir a los otros. Es muy difícil depender de la obra de uno mismo, por esto he decidido crear mi propio proyecto para comprobar si realmente java es la mejor opción para desarrollar, después de investigar a fondo todas las opciones se ha elegido Kotlin. Ninguna tecnología es eterna, todas antes o después son sucedidas, por esto es categórico estar en constante proceso de aprendizaje y por ello la elección de una nueva tecnología en lugar de repetir lo mismo en java.

En el anterior proyecto se cometieron fallos de UX/UI, abstracción y estructura en general cosa que ha sido el principal criterio a la hora de elegir una tecnología que permitiera solucionarlas y que no consumiera demasiado tiempo esto además de realmente aportar un nuevo discurso al desarrollo, es categórico escuchar el discurso de aquellas tecnologías predecesoras para tener un discurso propio pero al ya conocer java y más lenguajes muy ligados a C este discurso ya es sobradamente conocido, por lo que se ha priorizado toda tecnología que creará una diferencia significativa frente a java y las sintaxis derivadas de C. A través de mi propio trabajo y de la readaptación de mi anterior proyecto de android demuestro las diferencias que crea el uso de diferentes tecnologías para acabar con el pensamiento general de “Todas las tecnologías son iguales”.

He realizado una readaptación del concepto tres en raya porque al ser el mismo concepto desarrollado de formas diferentes podemos tener puntos de comparación claros y empíricos.

“Tres en raya” se eligió al permitir un producto mínimo viable de rápida implementación al ser un juego muy simple. El juego tan solo ofrece  $9!(5!^4!)$  combinaciones, 126 sin tener en cuenta simetría por lo que es fácil de implementar. Al tener arrays se vuelve mucho más fácil de desarrollar puesto que gracias al paradigma funcional realizar operaciones sobre las mismas es muy simple e intuitivo esto sumado a la programación funcional de Kotlin mucho más madura que la de java hace que podamos probar más cosas del lenguaje para tener una opinión mucho más formada. El juego no depende en ningún momento de gráficos ni de librerías externas por lo que es un juego desarrollado al 100% con lógica de programación, en la versión online tan solo escuchamos los cambios en la base de datos pero al ser un juego en el que obligatoriamente un jugador tiene que esperar a otro no necesitamos concurrencia ni nada más complicado que implementar el escuchador de firebase. Y en cuanto a las IAs del modo offline podemos dar distintos enfoques y estrategias sin complicarnos demasiado. Si el primer movimiento es la celda central restamos al oponente de las 8 soluciones 5 en base a esta simpleza los bots pueden tomar enfoques y estrategias diferentes y ganar, perder o empatar de distintas maneras aunque la parte de ganar se le pondrá lo más difícil posible al usuario.

El proyecto está dividido en dos, el marco teórico en el cual se compararán directamente las tecnologías y la prueba de concepto en la que demuestro como el desarrollo en kotlin nos permite crear proyectos con mayor funcionalidad en menor tiempo gracias a su pragmatismo y sus notables ventajas frente a java así como sus pocas desventajas.

La utilización de Kotlin ha permitido integrar nuevas funcionalidades como crear partidas personalizadas, escanear y generar códigos QR, tener una administración de perfiles de usuario más madura y completa y un chat en su mínima expresión funcional que se ha desarrollado hasta dónde ha permitido el tiempo al ser un proyecto por sí mismo.

Más allá de las funcionalidades a nivel de desarrollo se ha reorganizado el código agrupando las clases por grupos funcionales, una mayor abstracción separando el manejo de firebase de la UI y al abstractar las IAs y procesos de validación de ganador de la partida se ha resumido el código y se ha hecho más reutilizable.

Por último y no menos importante reiterar que es una readaptación en lugar de una traducción o ampliación al haber sido una reestructuración completa y el código original no ha sido traducido de java a Kotlin ha sido replanteado respetando los conceptos, filosofía, funcionalidades y demás características que hacen de kotlin un lenguaje único e individual respecto a las otras opciones.

# Desarrollo del proyecto

## Objetivos

### Objetivos generales

1. Aprender algo nuevo que represente una mejora significativa frente a java.
2. Aprender kotlin.
3. Readaptar el proyecto previo a kotlin.
4. Mejorar la app preexistente en UX/UI.
5. Mejorar implementaciones de librerías externas.
6. Incrementar las funciones de la app.
7. Mejorar la estructura.
8. Crear algo nuevo y diferente.

### Objetivos específicos

1. Determinar las ventajas y desventajas de kotlin frente a java en la experiencia de desarrollo
2. Dar la posibilidad al usuario de tener un perfil personalizable
3. Proveer de diferentes formas de acceso al perfil
4. Permitir a los usuarios jugar entre ellos
5. Permitir diferentes accesos a la partida online
6. Proveer de partidas online
7. Permitir la comunicación entre usuarios en todo momento en tiempo real mediante chat
8. Proveer al usuario de una forma de uso de la app si no tiene conexión a internet
9. Crear competitividad entre jugadores con un ranking

## Introducción

En esta parte del proyecto se tratarán las diferentes ventajas y desventajas de kotlin frente a otras tecnologías tanto en el marco de desarrollo nativo en Android como en propósito general especialmente centrándonos en java y kotlin.

### Introducción a Kotlin

Kotlin es un lenguaje desarrollado por Jetbrains en 2016 de alto nivel de tipado estático que permite la inferencia de tipos, es multiparadigma, de propósito general y compila tanto en JVM por lo que es 100% interoperable con Java todas sus bibliotecas y frameworks, es 100% interoperable con Objective-C, también compila a nativo y en Javascript.

Kotlin tiene su mayor uso en el desarrollo nativo para android debido principalmente a que google lo reconoció como lenguaje oficial para android, solo esto es un argumento a tener en cuenta para aprender kotlin en nuestro grado.

Aunque la mayoría de la gente en general piensa que Kotlin únicamente tiene utilidad en Android, tiene más utilidades:

**En back-end** podemos usar Kotlin tanto con spring al ser 100% interoperable con java como con Kton framework back-end desarrollado por jetbrains pensado en Kotlin con el que se mantienen en mayor medida las intenciones de Kotlin y por supuesto node js con Kotlin/JS.

**El front-end** web también puede ser desarrollado con Kotlin/JS tanto con los frameworks de javascript como con los propios desarrollados para kotlin/JS: KVision, fritz2 y Doodle.

**En escritorio** podríamos usar JavaFx sin problema o TornadoFx que está expresamente desarrollado para respetar las diferencias de Kotlin frente a java.

**Desarrollo multiplataforma**, al usar KMM podríamos desarrollar para los y android reutilizando la mayoría de código salvo el propio de cada plataforma.

Por otra parte enfatiza el pragmatismo sobre la conversación. Kotlin trabaja a un nivel más alto de abstracción y proporciona funciones de lenguaje prácticas que no están disponibles en Java. Una de las características más importantes de Kotlin es su interoperabilidad al 100% con las bibliotecas y bases de código Java existentes.

El pragmatismo de Kotlin nos permite agilizar el desarrollo y simplificar el desarrollo que de ello será más legible, fácil de entender y sencillo de testear.

Con la inferencia de tipos sumada al tipado estático y al crear la diferencia entre mutable e inmutable nuestro código será más pragmático en diferentes aspectos como menos verbosidad en la declaración de las mismas y tener menos código repetitivo.

La diferencia entre mutable e inmutable nos obligará a plantearnos si vamos a tener variables u objetos que posteriormente queramos cambiar de estado, esto también dará mayor seguridad al código y mejor experiencia de testeo.

La seguridad en Kotlin es lo más importante por lo que es Null Safety por defecto lo que nos obligará a controlar en todo momento que un valor no sea nulo, en caso de serlo podremos decidir cómo actuar y aunque no acepta valores nulos por defecto kotlin permitirá saltarselo siempre y cuando indiquemos.

La sintaxis está bastante alejada de “C” por lo que tendremos nuevas ventajas lógicas recurrentes en los lenguajes modernos como que los “if” podrán retornar valores, estructura “When” una evolución significativa frente al switch/case y mas cosas que se detallan más adelante y que mejoran el desarrollo.

En esta parte vamos a ver una pequeña introducción a la programación en kotlin. Aunque en la comparación con java la veremos con mayor profundidad para tener una idea de cómo se trabaja con Kotlin es importante ver su sintaxis en alguna profundidad.

```
fun main() {
    println("Hola mundo !!!!")
}
```

Variables

Ejemplo	Código
<b>Inferencia de tipos</b> En este ejemplo podemos observar como solo declaramos las variables sin necesidad de poner tipos	<pre>val num = 3 val num2 = 3 var sum = num + num2 print(sum)</pre>
<b>Variable null mutable String</b> El propio entorno avisara de que esto no se debe hacer al ejecutarlo fallará	<pre>var nombre:String nombre = null print(nombre)</pre> <p>Null can not be a value of a non-null type String</p>
<b>Variable null inmutable</b> El propio entorno avisara de que esto no se debe hacer al ejecutarlo fallará	<pre>val nombre:String = null print(nombre)</pre> <p>Null can not be a value of a non-null type String</p>
<b>Variable nullable mutable string</b> Declaramos que la variable puede ser nula en caso de no especificar el tipo se usará Nothing? por defecto	<pre>var nombre:String? = null print("nombre = \$nombre") &gt; Task :EjemKt.main() nombre = null</pre>
<b>Variable post iniciada</b> En algún momento necesitaremos variables globales por lo que tendremos que especificarlo	<pre>lateinit var nombre:String  fun imprimir() {     nombre = "Cristian"     print(nombre) } &gt; Task :EjemKt.main() Cristian</pre>
<b>Desactivar comprobación de nulos</b> Esto no es buena idea en general pero en programación es normal que en ciertos momentos aceptemos el riesgo	<pre>val nombre:String? = null val longitud = nombre!! .length print(longitud) Execution failed for task ':EjemKt.main()'. &gt; Process 'Command 'C:/Program Files/Java/jdk1.8.0_281/bin/java.exe'' finished with non-zero exit value 1  * Try: Run with --stacktrace option to get the stack</pre>

	trace. Run with --info or --debug option to get more log output. Run with --scan to get full insights.
<b>Operador Elvis</b> Este operador desactiva la comprobación de nulos pero da un valor en caso de ser nulo para evitar la comprobación. He puesto 30 para que quede claro que el valor es asignado manualmente y no por defecto	<pre>val nombre:String? = null val longitud = nombre?.length ?: 30 print(longitud) 30</pre>
<b>Let</b> Bloque de código que se ejecutará en caso de que la variable no sea null	<pre>val nombre:String? = "Cristian"  nombre.let {     println("Nombre valido \$it") } Cristian</pre>
<b>NullCheck</b> Aunque no es una propiedad como tal del lenguaje los idles de jetbrains nos obligan a ponerlos en ciertas ocasiones por seguridad	<pre>var nombre:String? = "Cristian" if (nombre != null) {     println(nombre) } Cristian</pre>

if

La declaración if sin ningún cambio significativo

```
fun main() {
    val num = 3
    if (num > 0) {
        println("Mayor que 0")
    }
    else if (num < 0) {
        println("Menor de 0")
    }
    else{
        println("Cero")
    }
}
```

Mayor que 0

If como expresión, esto implica que al usarla como expresión podrá asignar valor a una variable.

Desaparece el ternario y podremos tener más de dos resultados.

```
fun main() {
    val num = 5
    val num2 = 4
    val result =
        if (num > num2) {
            num
        }
        else if (num < num2) {
            num2
        }
        else{
            "iguales"
        }
    println("result = $result")
}
result = 5
```

```
fun main() {
    val mes = 5
    val trimestre = when(mes) {
        in 1..3 -> "Primer"
        in 4..6 -> "Segundo"
        in 7..9 -> "Tercer"
        in 10..12 -> "Cuarto"
        else -> "Fuera de rango"
    }
    println("El mes: ${mes} pertenece al ${trimestre} trimestre")
}
```

El mes: 5 pertenece al Segundo trimestre

## Bucles

### *While*

Este bucle se ejecuta hasta que se cumple una condición, ninguna diferencia reseñable.

```
fun main() {
    var cont = 0
    while (cont < 10) {
        print("$cont, ")
        cont++
    }
}
```

0, 1, 2, 3, 4, 5, 6, 7, 8, 9,

A diferencia de los lenguajes C funciona mediante rangos

<b>Números del 1 al 10</b>	<pre>fun main() {     for(i in 1..10) {         print("\$i, ")     } }</pre> <p>1, 2, 3, 4, 5, 6, 7, 8, 9, 10,</p>
<b>Números del 1 al 10 menos el 10</b>	<pre>fun main() {     for(i in 1 until 10) {         print("\$i, ")     } }</pre> <p>1, 2, 3, 4, 5, 6, 7, 8, 9,</p>
<b>Números de 10 al 1</b>	<pre>fun main() {     for(i in 10 downTo 1) {         print("\$i, ")     } }</pre> <p>10, 9, 8, 7, 6, 5, 4, 3, 2, 1,</p>
<b>Números impares del 1 al 10</b>	<pre>fun main() {     for(i in 1..10 step 2) {         print("\$i, ")     } }</pre> <p>1, 3, 5, 7, 9,</p>
<b>Recorrer array</b>	<pre>fun main() {     var numeros = arrayOf(1,2,3,4,5,6,7,8)     for (i in 0 until numeros.size) {         print("\${numeros[i]}, ")     } }</pre> <p>1, 2, 3, 4, 5, 6, 7, 8,</p>
<b>ForEach</b>	<pre>fun main() {     var numeros = arrayOf(1,2,3,4,5,6,7,8)     for (i in numeros) {         print("\$i, ")     } }</pre> <p>1, 2, 3, 4, 5, 6, 7, 8,</p>

<b>Funciones void</b> Por defecto retornara Unit	<pre>fun main() {     imprimir() } fun imprimir() {     print("Hola desde imprimir") }</pre> Hola desde imprimir
<b>Funciones con return y parámetros</b>	<pre>fun main() {     print("\${sumar(2, 2) }") } fun sumar(x:Int, y:Int):Int {     return x + y }</pre> 4
<b>Funciones lambda</b>	<pre>fun main() {     val suma:(Int, Int) -&gt; Int = {a, b-&gt; a+b}     var result = suma(2, 2)     print(result) } </pre> 4

Kotlin proporciona una Programación orientada a objetos pragmática y concisa.  
 Los modificadores de acceso son private, protected, internal y public (acceso por defecto).

## Clases

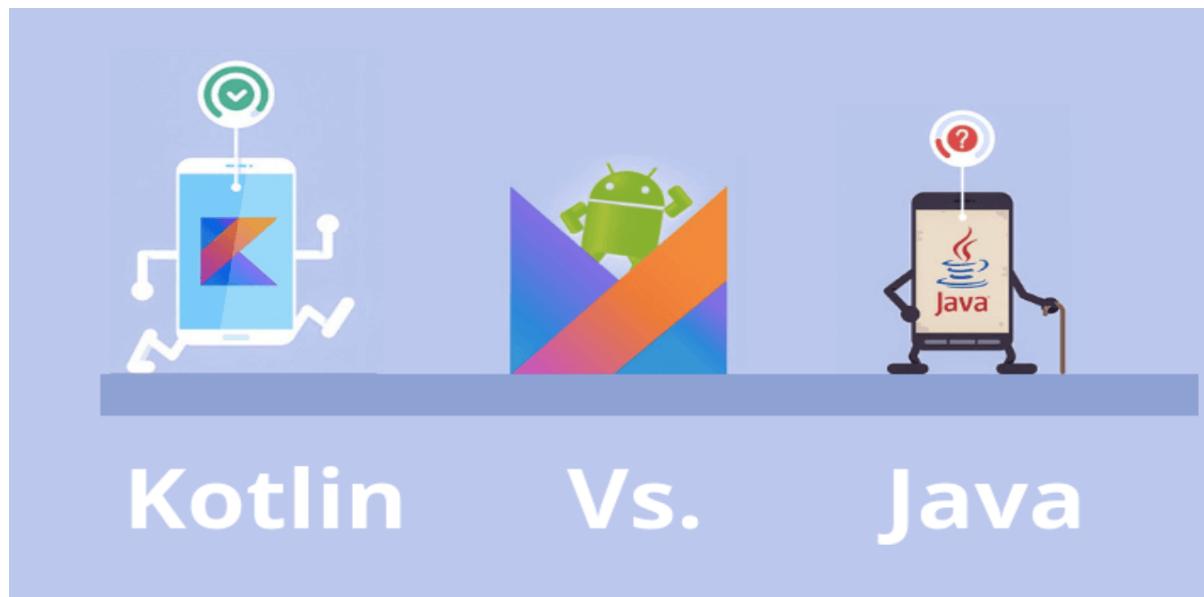
Class	<pre>class Robot {     //Variables de clase     var codigo:Int = 0     var marca:String = "marca"     var nombre:String = "nombre"     //Constructor con todos los valores     constructor(codigo:Int,marca:String, nombre:String) {         this.codigo = codigo         this.marca = marca         this.nombre = nombre     }     //Constructor sin argumento     constructor()     //Metodo     fun hablar():String{         return "Hola soy \${this.codigo} \${this.marca} \${this.nombre}"     } }</pre>
Main	<pre>fun main() {     val robotDefault = Robot()     val robot = Robot(2222,"xiaomi","MIROBOT")     println("robot habla \${ robot.hablar() }")     println("robotDefault habla \${ robotDefault.hablar() } ")</pre>
Salida	robot habla Hola soy 2222 xiaomi MIROBOT robotDefault habla Hola soy 0 marca nombre

Una de las principales aportaciones de Kotlin son las Data Class, estas clases se utilizan cuando tenemos clases cuya única intención es guardar datos.

El compilador deriva automáticamente los siguientes miembros de todas las propiedades declaradas en el constructor principal:

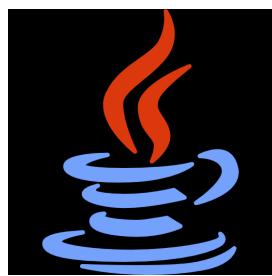
- equals()/ hashCode() pair
- toString()
- componentN()
- copy()

<b>Data Class</b>	<pre>data class Persona(val nombre:String, val edad:Int, val dni:String)</pre>
<b>Main</b>	<pre>fun main(){     val cristian = Persona("Cristian",21,"33333A")     println("toString = \${cristian.toString()}")     println("component1 \${cristian.component1()}")     val oldCristian = cristian.copy(edad = 66)     println("oldCristian     \${oldCristian.toString()}") }</pre>
<b>Salida</b>	<pre>toString = Persona(nombre=Cristian, edad=21, dni=33333A) component1 Cristian oldCristian Persona(nombre=Cristian, edad=66, dni=33333A)</pre>



Después de tanta investigación y tecnologías descartadas elegí Kotlin, en esta parte más incisiva se va ha mostrar en que aspectos Kotlin puede suponer una ventaja o desventaja frente a java.

## Introducción a Java



Java es un lenguaje de alto nivel de propósito general y orientado a objetos desarrollado por Sun Microsystems y continuado por Oracle que actualmente está presente principalmente en aplicaciones multiplataformas y backend.

Java SE 15, fue lanzada el 15 de septiembre de 2020, es la versión más reciente del lenguaje aunque en android estamos limitados a java 8 y las características de esta versión solo las podemos utilizar en proyectos desarrollados con la API 25 o superior.

Un punto de vista categórico a la hora de valorar el ocupar nuestro tiempo a una nueva tecnología es el ámbito laboral.

Para este apartado se han realizado cinco búsquedas en infojobs todas ellas con el criterio “En toda España” para tener una visión global y muestra siguientes datos:

Criterio	Java	Kotlin	Android	Java + Android	Kotlin + Android
N ofertas	1336	68	214	71	42

Java gana, cosa de esperar, es una tecnología más antigua y con más presencia a nivel

empresarial por lo que aprender java como lenguaje en general nos da un margen laboral extremadamente superior a kotlin. Sin embargo la diferencia en el mundo android es de 29 ofertas lo que demuestra que de enfocarse en android kotlin es un lenguaje a tener en cuenta pero como lenguaje fuera de android no tiene apenas presencia y no termina desligarse porque en la mayoría de ofertas se exige tanto java como kotlin por lo que el número real de 42 podría bajar a menos de la mitad.

Como podemos observar la diferencia salarial que perciben los desarrolladores es de 4.000 al año, 333 euros al mes en favor de Kotlin.

## Seguridad

Como he explicado en la introducción a Kotlin, Kotlin al ser Null Safety por defecto es más seguro al tener más controladas las NullPointerException.

Este enfoque hace más seguro el desarrollo al tener al programador más controlado por el propio entorno de desarrollo y obligarnos al controlar en todo momento si un objeto es null o puede ser susceptible a serlo, esto crea mayor conciencia sobre la dependencia que tenemos a la hora de que la aplicación no se cierre en caso de que un objeto no serializer/deserialize correctamente y podamos dar alternativas como avisos sin parar la ejecución.

Más allá del Null Safety el pragmatismo de Kotlin hará que escribamos menos código no significativo por lo que tendremos mayor conciencia de lo que hemos escrito, el compilador detecta mejor los errores que java, al ser más declarativo que java tenemos más claro lo que hace en lugar de como lo hace por lo que en caso de ser mantenido por otra persona o abandonar el proyecto durante cierto tiempo tendremos menos problema al descubrir/leer lo que hace.

## Velocidad

Java al tener un nivel menor de abstracción es más rápido pero esto no será tan significativo como con Python o Ionic al ser apenas tiempo y no generar un consumo excesivo del hardware.

Tanto Java como Kotlin son lenguajes interpretados por lo que siempre serán más lentos que los lenguajes de bajo nivel como c/c++ pero tienen un tiempo de desarrollo mayor.

Kotlin permite escribir aplicaciones más seguras y más funcionales en menor tiempo sacrificando velocidad de ejecución que realmente solo serán pequeños retardos no significativos para el usuario.

## Comunidad

Ambos están mantenidos por grandes empresas que invierten muchísimo tiempo y dinero en que los lenguajes mejoren.

De hecho el 19 de mayo de 2021 Jetbrains sacó la nueve versión y realizó cambios significativos tales como nuevas versiones cada seis meses de forma estricta para que una funcionalidad nueva no retrase al resto de tal forma que cada seis meses en lenguaje aportada cosas nuevas y corregirá los posibles fallos de una forma más dinámica para el desarrollador.

En mayor problema que nos encontraremos en Kotlin es que en general está encerrado en Android cuando se puede utilizar en todos los ámbitos propios del alto nivel.

La ficción de que kotlin únicamente es util en android se crea desde el aprendizaje del mismo por parte de ciertos cursos mal estructurados, el curso de OpenWebinars enseña kotlin como lenguaje de programación de forma estricta en android studio dando a entender que es necesario crear obligatoriamente una App para poder usarlo.

En la investigación laboral se encontraron tres ofertas backend en kotlin híbrido con java en Spring las empresas son: Telefónica, Milanuncios y Wallapop, tres empresas con un trabajo de red enorme que encuentran ventajas clave en el desarrollo con Kotlin.

Java sin embargo se ha quedado en versiones anteriores a la 8 por la mayoría de programadores, empresas y estudios por lo que se pierde todo lo que aportó java 8 una actualización categórica que aportó programación funcional entre otras cosas.

Las nuevas aportaciones de java en general tienen poca intención de aprendizaje por parte de los desarrolladores en general.

Por lo que podemos concluir que en ambas opciones encontraremos comunidades grandes y estables pero cada una con sus deficiencias, Kotlin sobre centrada en Android y la de java ignorando las aportaciones por parte de Oracle al lenguaje.

La comunidad en general respalda a Kotlin.

Podemos ver como Kotlin es el

cuarto lenguaje más amado por los desarrolladores y java es el decimoséptimo estando trece puestos por debajo de kotlin.

Empezamos con la parte más práctica de este marco teórico en la que comparare mis dos proyecto en general veremos en Kotlin una sintaxis menos amigable con C pero más pragmática por lo que si se está muy acostumbrado a C y derivados de su sintaxis puede parecer extraña pero no por ello difícil de comprender.

Kotlin en ningún momento pretende ahorrar líneas, intenta hacernos más conscientes del código que escribimos.

### *Programación general*

La estructura if puede ser utilizada como expresión por lo que puede retornar valores esto será interesante de utilizar en métodos con retornos de datos condicionales al no tener que escribir “return dato” en cada declaración.

Ejemplo	
java	kotlin
<pre>@RequiresApi(api = Build.VERSION_CODES.N) public int ganador() {     this.aux = [ this.convertirDatos();     if(this.vertical(1) == 1) {         return 1;     }     else if(this.vertical(2) == 2) {         return 2;     }     else if(this.horizontal(1) == 1) {         return 1;     }     else if(this.horizontal(2) == 2) {         return 2;     }     else if(         (this.diagonalFacil(1) == 1) {             return 1;         }         else         if(this.diagonalFacil(2) == 2) {             return 2;         }         else         if(this.diagonalDificil(1) == 1) {             return 1;         }         else         if(this.diagonalDificil(2) == 2) {             return 2;         }     } }</pre>	<pre>fun determinarGanador(): Int {     convertirDatos()     return if(         verificarVertical(1) === 1) {             1         } else if(         verificarVertical(2) === 2) {             2         } else if(         verificarHorizontal(1) === 1) {             1         } else if(         verificarHorizontal(2) === 2) {             2         } else if(         verificarDiagonal1(1) === 1) {             1         } else if(         verificarDiagonal1(2) === 2) {             2         } else if(         verificarDiagonal2(1) === 1) {             1         } else if(         verificarDiagonal2(2) === 2) {             2         } else if (tablero.filter {             x -&gt; x != 0 }.count() == 9) {             3         } else {             0         }     }</pre>

```

    else
if(this.partida.getTablero().st
ream().filter(x-> x!=0).count()
== 9) {
    return 3;
}
else {
    return 0;
}
}

```

También tendremos la estructura “when”, esta estructura también permite retornar valores, es una evolución de switch/case al ser más flexible y aceptar rangos y condiciones más complejas que un único valor.

### Ejemplo (apuntes)

java

```

public class Ejem {
    public static void
main(String[] args) {
    int num = 4;
    String result;
    switch (num) {
        case 1:
            result = "Lunes";
            break;
        case 2:
            result = "Martes";
            break;
        case 3:
            result =
"Miercoles";
            break;
        case 4:
            result = "Jueves";
            break;
        case 5:
            result =
"Viernes";
            break;
        case 6:
            result = "Sabado";
            break;
        case 7:
            result =
"Domingo";
            break;
        default:
            result = "Valor no
valido";
    }
    System.out.println("Has introducido " + result);
}

```

kotlin

```

fun main() {
    val num = 4
    var resultado = when (num) {
        1-> "Lunes"
        2-> "Martes"
        3-> "Miercoles"
        4-> "Jueves"
        5-> "Viernes"
        6-> "Sabado"
        7-> "Domingo"
        else -> "Valor no
valido"
    }
    println("Has elegido
\$resultado")
}

```

}

## Ejemplo

java

```
if (puntos >= 0) {
    if (puntos >=0 && puntos <=
5) [ ] nivel = "PRINCIPIANTE";
    else if (puntos >=6 && puntos
<= 20) [ ] nivel = "EXPERIMENTADO";
    else if (puntos >=21 &&
puntos <= 50) [ ] nivel = "EXPERTO";
    else [ ] nivel = "EXPERIMENTADO";
}
else [ ] nivel = "NOVATO";
```

kotlin

```
if (puntos >= 0) {
    when(puntos) {
        in 0..5 -> nivel =
"PRINCIPIANTE"
        in 6..20-> nivel =
"EXPERIMENTADO"
        in 21..50-> nivel
="EXPERTO"
        else-> nivel = "GRAN
EXPERTO"
    }
} [ ]
else{
    nivel = "NOVATO"
}
```

El bucle for ha sido mejorado y evolucionado de una forma más compacta.

## Ejemplo

java

```
for (int i = 0; i
<this.partida.getTablero().size
() ; i++) { [ ]

if(this.partida.getTablero().ge
t(i) == 0){ [ ]

this.casillas.get(i).setImageRe
source(R.drawable.casilla);
} [ ]
else
if(this.partida.getTablero().ge
t(i) ==
1){this.casillas.get(i).setImag
eResource(R.drawable.skull);
} [ ]
else{ [ ]

this.casillas.get(i).setImageRe
source(R.drawable.fantasma);
} [ ]
}
```

kotlin

```
for(i in 0 until
partida.tablero.size){
    when(partida.tablero[i]){
        0 -> [ ]
casillas[i].setImageResource(ic
asilla)
        1 -> [ ]
casillas[i].setImageResource(R.
drawable.crossbone)
        2 -> [ ]
casillas[i].setImageResource(R.
drawable.fantasma)
    }
}
```

También tendremos métodos que realmente no serán significativos a la hora de programar pero si sumara legibilidad.

Ejemplo	
java	kotlin
<pre>if (vCorreo.isEmpty() &amp;&amp; !mather.find()) { [REDACTED]     this.email.setError("CORREO NO VALIDO"); }</pre>	<pre>if ((temail.text.isNotEmpty() &amp;&amp; [REDACTED] validarEmail(temail.text.toString()))){ [REDACTED] temail.error = "EMAIL NO VALIDO" }</pre>

Por último todas las colecciones se manejan con la misma sintaxis a diferencia de Java que maneja de forma diferente Arrays y ArrayList

Ejemplo	
java	kotlin
<pre>@RequiresApi(api = Build.VERSION_CODES.N) public void initCasillas(){     this.casillas = new ArrayList&lt;&gt;();  this.casillas.add((ImageView) findViewById(R.id.imageView0));  this.casillas.add((ImageView) findViewById(R.id.imageView1));  this.casillas.add((ImageView) findViewById(R.id.imageView2));  this.casillas.add((ImageView) findViewById(R.id.imageView3));  this.casillas.add((ImageView) findViewById(R.id.imageView4));  this.casillas.add((ImageView) findViewById(R.id.imageView5));  this.casillas.add((ImageView) findViewById(R.id.imageView6));  this.casillas.add((ImageView) findViewById(R.id.imageView7));</pre>	<pre>private fun initCasillas() {     casillas= arrayOf(ocasilla0, ocasilla1, ocasilla2, ocasilla3, ocasilla4, ocasilla5, ocasilla6, ocasilla7, ocasilla8) [REDACTED]     casillas.forEach { x-&gt; x.visibility=View.VISIBLE }     casillas.forEach { x -&gt; x.setImageResource(R.drawable.c asilla) } }  for(i in 0 until partida.tablero.size){     when(partida.tablero[i]){         0 -&gt; [REDACTED] casillas[i].setImageResource(ic asilla)         1 -&gt; [REDACTED] casillas[i].setImageResource(R. drawable.crossbone)         2 -&gt; [REDACTED] casillas[i].setImageResource(R. drawable.fantasma)     } }</pre>

```
this.casillas.add((ImageView)
findViewById(R.id.imageView8));  
  
this.casillas.forEach(x->x.setVisibility(View.VISIBLE));  
  
this.casillas.forEach(x->x.setImageResource(R.drawable.casilla));  
}  
}  
for (int i = 0; i <this.partida.getTablero().size(); i++) {  
  
if(this.partida.getTablero().get(i) == 0){  
  
this.casillas.get(i).setImageResource(R.drawable.casilla);  
}  
else{  
if(this.partida.getTablero().get(i) == 1){  
  
this.casillas.get(i).setImageResource(R.drawable.skull);  
}  
else{  
  
this.casillas.get(i).setImageResource(R.drawable.fantasma);  
}  
}  
}  
}
```

El paradigma funcional no llegó a java hasta su versión 8 por lo que en android solo podemos utilizarla a partir del API 25 lo que nos limita bastante a la hora de desarrollar código.

La principal función de la programación funcional en el proyecto será filtrar datos

Filter	
Esta función sumada a la función count contara cuantos elementos coinciden con el parámetro facilitado	
No funcional	
<pre>private int filtro(int filtro){     int count = 0;     for (int i = 0; i &lt; lista.length ; i++) {         if(lista[i] == filtro) {             count++;         }     }     return count; }</pre>	
java	kotlin
Arrays.stream(tabaux).filter(au x -> aux[0]==).count() == 3	tableroDiagonal1.filter { x -> x == 0 }.count() == 3

Al juntar la programación general y funcional de kotlin podemos simplificar funciones previas

java	kotlin
<pre>@RequiresApi(api = Build.VERSION_CODES.N) public int horizontal(int jugador) {     for (int i = 0; i &lt;this.aux.length ; i++) {  if( Arrays.stream(aux[i]).filter( aux -&gt; aux == jugador).count() == 3) {             return jugador;         }     }     return 0; }</pre>	<pre>fun verificarHorizontal(jugador: Int): Int {     tableroHorizontal.forEach { x -&gt; if (x.filter { y -&gt; y == jugador }.count() == 3) return jugador }     return 0 }</pre>

Data class mejor opción para Models que Clases.

Ejemplo	
java	kotlin
<pre>package com.example.prueb2;  public class UserJava {      private String nick;     private String nivel;     private String imageURI;     private int puntos;     private int victorias;     private int derrotas;     private boolean estado;     public UserJava(){ }     public UserJava(String nick, String nivel, String imageURI, int puntos, int victorias, int derrotas, boolean estado) {         this.nick = nick;         this.nivel = nivel;         this.imageURI = imageURI;         this.puntos = puntos;         this.victorias = victorias;         this.derrotas = derrotas;         this.estado = estado;     }     private void cambiarRango(){         if (puntos &gt;= 0){             if (puntos &gt;=0 &amp;&amp; puntos &lt;= 5)                 nivel = "PRINCIPIANTE";             else if (puntos &gt;=6 &amp;&amp; puntos &lt;= 20)                 nivel = "EXPERIMENTADO";             else if (puntos &gt;=21 &amp;&amp; puntos &lt;= 50)                 nivel = "EXPERTO";             else                 nivel = "EXPERIMENTADO";         }         else             nivel = "NOVATO";     } }</pre>	<pre>package com.example.proyectofinal.model s  data class User(     var id:String="",     var nick:String = "",     var nivel:String = "",     var imageURI:String = "",     val puntos:Int = 0     ,     val victorias:Int = 0,     val derrotas:Int = 0,     val estado:Boolean = false)  constructor():this("", "", "", "", 0, 0, 0, false)     init{         cambiarRango()     }     fun cambiarRango(){         if (puntos &gt;= 0){             when(puntos){                 in 0..5 -&gt; nivel                 = "PRINCIPIANTE"                 in 6..20-&gt; nivel                 = "EXPERIMENTADO"                 in 21..50-&gt; nivel = "EXPERTO"                 else-&gt; nivel = "GRAN EXPERTO"             }         }         else{             nivel = "NOVATO"         }     } }</pre>

```
public String getNick() {
    return nick;
}

public void setNick(String
nick) {
    this.nick = nick;
}

public String getNivel() {
    return nivel;
}

public void setNivel(String
nivel) {
    this.nivel = nivel;
}

public String getImageURI()
{
    return imageURI;
}

public void
setImageURI(String imageURI) {
    this.imageURI =
imageURI;
}

public int getPuntos() {
    return puntos;
}

public void setPuntos(int
puntos) {
    this.puntos = puntos;
}

public int getVictorias() {
    return victorias;
}

public void setVictorias(int
victorias) {
    this.victorias =
victorias;
}

public int getDerrotas() {
    return derrotas;
}

public void setDerrotas(int
derrotas) {
    this.derrotas =
derrotas;
}
```

```
}

    public boolean isEstado() {
        return estado;
    }

    public void
setEstado(boolean estado) {
        this.estado = estado;
    }

    @Override
    public String toString() {
        return "UserJava{" +
            "nick='"
+ nick +
            "', nivel='"
+ nivel +
            "', imageURI='"
+ imageURI +
            "', puntos='"
+ puntos +
            "', victorias='"
+ victorias +
            "', derrotas='"
+ derrotas +
            "', estado='"
+ estado +
            "'}";
    }
}
```

105 líneas

24 líneas

Ejemplo	
java	kotlin
<pre>import java.util.ArrayList;  public class PartidaUtil {     private ArrayList&lt;Integer&gt; tablero; [REDACTED]     private ArrayList&lt;Integer&gt; tableroHorizontal; [REDACTED]     private ArrayList&lt;Integer&gt; tableroVertical; [REDACTED]     private ArrayList&lt;Integer&gt; tableroDiagonal1; [REDACTED]     private ArrayList&lt;Integer&gt; tableroDiagonal2; [REDACTED]     public [REDACTED] PartidaUtil(ArrayList&lt;Integer&gt; tablero) { [REDACTED]         this.tablero = tablero;         convertirDatos()     } [REDACTED]     private void convertirDatos(){ [REDACTED]         this.tableroHorizontal = convertirDatosHorizontales();         this.tableroVertical = convertirDatosVerticales(); [REDACTED]         this.tableroDiagonal1 = convertirDatosDiagonal1(); [REDACTED]         this.tableroDiagonal2 = convertirDatosDiagonal1();     } }</pre>	<pre>class PartidaUtil(tablero:IntArray) {      private var tablero: IntArray = tablero [REDACTED]     private lateinit var tableroVertical: Array&lt;IntArray&gt;     private lateinit var tableroHorizontal: Array&lt;IntArray&gt; [REDACTED]     private lateinit var tablerodiagonal1: IntArray     private lateinit var tablerodiagonal2: IntArray      init {         convertirDatos()     }      fun convertirDatos() {         tableroHorizontal = convertirDatosHorizontales()         tableroVertical = convertirDatosVerticales()         tablerodiagonal1 = convertirDatosDiagonal1()         tablerodiagonal2 = convertirDatosDiagonal2()     } }</pre>

Java 15 incluyó record data que son parecidas a las dataclass pero en android no se pueden usar porque android está limitado a java 8.

Por otra parte para simplificar las clases de solo datos sería interesante probar lombok aunque la versión de java lo impide y una parte de la comunidad no es partidaria de lombok al tocar el bytecode y en caso de fallar el debug no es preciso.

A la hora de acceder/manipular elementos de nuestro XML kotlin provee una serie de librerías para no tener que instanciar los uno por uno lo que genera minimo 2 líneas de código por cada elemento lo que si bien no es demasiado trabajo genera mucho código innecesario.

En el proyecto he utilizado kotlinc.android.synthetic, su uso es extremadamente simple, importas el activity o fragmentos que necesitas y el solo te los servira directamente para que puedas utilizarlos.

Ejemplo	
java	kotlin
<pre>EditText email,contra,nick,rcontra; TextView error; Button blogin; Switch condiciones; ProgressBar progressBar; protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity_registro);     //iniciamos las variables     con los componentes     correspondientes del activity     this.nick = [REDACTED]     findViewById(R.id.nick);     this.email = [REDACTED]     findViewById(R.id.email);     this.contra = [REDACTED]     findViewById(R.id.contra);     this.rcontra = [REDACTED]     findViewById(R.id.rcontra);     this.blogin = [REDACTED]     findViewById(R.id.blogin);      this.progressBar =     findViewById(R.id.progressBar);</pre>	<pre>import kotlinc.android.synthetic.main. activity_other_user_info.*  otherNick.text = "Nick:\t\t\${userAux.nick}" mynivel.text = [REDACTED] "Nivel:\t\t\${userAux.nivel}" myPuntos.text = [REDACTED] "Puntos:\t\t\${userAux.puntos}" myVictorias.text = [REDACTED] "Victorias:\t\t\${userAux.victorias}" myderrotas.text="Derrotas:\t\t\${userAux.derrotas}"</pre>

Por otra parte la captura de eventos es mucho más clara en kotlin

Ejemplo	
java	kotlin
<pre>this.nick.setOnClickListener(ne w View.OnClickListener() {     @Override     public void onClick(View v)     {         System.out.println("Hola desde botón");     } });</pre>	<pre>tinicsesion.setOnClickListener{     startActivity(Intent(this@Regis terActivity, LoginActivity::clas s.java)) }</pre>

El cambio de activity es mucho más claro

Ejemplo	
java	kotlin
<pre>Intent i = new Intent(EndGameActivity.this, EncontrarPartidaActivity.class) ; startActivity(i);</pre>	<pre>startActivity(Intent(this@EndGa meActivity, MainOnlineActivity:: class.java))</pre>

La principal biblioteca externa que vamos a estar manejado es Firebase esta si estará comparada con el proyecto, aquellas que no están en el anterior se compara con la documentación oficial.

Guardar datos en firebase

#### Ejemplo

```
db.collection("cities").document("LA")
    .set(city)
    .addOnSuccessListener(new OnSuccessListener<Void>() {
        @Override
        public void onSuccess(Void aVoid) {
            Log.d(TAG, "DocumentSnapshot successfully written!");
        }
    })
    .addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            Log.w(TAG, "Error writing document", e);
        }
    });
});
```

#### Documentación oficial

#### Ejemplo

```
this.almacenamiento.collection("users").document(id).get().addOnSuccessListener(documentSnapshot -> {
    if (!documentSnapshot.exists()) {
        completarNick.setVisibility(View.VISIBLE);
        User user1 = new User(snick,0,0);
        almacenamiento.collection("users")
            .document(id)
            .set(user1)
            .addOnCompleteListener(task -> {
                mostrarToast("Registro Correcto");
                finish();
                cambiarDePantalla();
            });
    } else{
        cambiarDePantalla();
    }
});
```

#### Ejemplo de la versión resumida con Lambdas en java

Con este ejemplo se pretende mostrar que muchas veces en las documentaciones se usa java de una mucho más antigua y verbosa de lo que en realidad es hoy en dia.  
Sin embargo se usó en el proyecto android la versión no resumida por el tiempo en Kotlin esto ha sido solucionado.

## Ejemplo

java

```

private void registrar(FirebaseUser user, String nick) {
    User user1 = new User(nick, 0, 0);
    this.db.collection("users")
        .document(user.getUid())
            .set(user1)
        .addOnCompleteListener(new OnCompleteListener() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                mostrarToast("Registro Correcto");
                finish();
                cambiarDePantalla();
            }
        });
}

```

kotlin

```

authProvider.register(temail.text.toString(), tcontra.text.toString())
    ?.addOnSuccessListener {
        Toast.makeText(this, "CORREO REGISTRADO", Toast.LENGTH_LONG).show()
        var usuario = AuthProvider().getUid()?.let {
            it1 -> User(
                it1, nick.text.toString(),
                "PRINCIPIENTE", imageURI, 0,
                0, 0, false)
        }
        if (usuario != null) {
            this.createUser(usuario)
        }
    }
}

```

## Ejemplo

java

```

private void recuperarUsuario() {
    this.db.collection("users")
        .document(this.user.getUid())
            .get()
        .addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
            @Override
            public void onSuccess(DocumentSnapshot documentSnapshot) {
                useraux = documentSnapshot.toObject(User.class);
            }
        });
    result.setText(resultado);
}

```

kotlin

```

authProvider.getUid()?.let {
    userProvider.read(it).addOnSuccessListener {
        var user:User = it.toObject(User::class.java)!!
        Log.e("IDE", extras.getString("RESULTADO").toString())
        if (user != null) {
            when(extras.getString("RESULTADO").toString()) {
                "GANADOR" -> {
                    generarConfetti()
                    user.victorias += 1
                    user.puntos += 3
                }
            }
        }
    }
}

```

```

if(resultado.equals("Ganador"))
{
    useraux.setPuntos(useraux.getPuntos() + 3);

    puntos.setText("+3");
}
else
if(resultado.equals("Perdedor"))
{
    animation.setVisibility(View.GONE);
    animation2.setVisibility(View.VISIBLE);

    useraux.setPuntos(useraux.getPuntos() - 3);

    puntos.setText("-3");
}
else{
    animation.setAnimation("empate.json");

    puntos.setText("0");
}

useraux.setPartidas(useraux.getPartidas() + 1);

nombre.setText(useraux.getNick());
total.setText(useraux.toString());
actualizarUser(useraux);
}
}

```

```

aWinner.setVisibility =
View.VISIBLE

alooser.setVisibility = View.GONE

aempata.setVisibility = View.GONE

user.cambiarRango()

ovictorias.text = "VICTORIAS
${user.victorias}\t(+1)"

oderrtas.text = "DERROTAS
${user.derrotas}"

oempates.text = "EMPATES
${user.empates}"

opuntos.text = "PUNTOS
${user.puntos}\t(+3)"
}
"PERDEDOR" ->{

user.derrotas += 1

user.puntos -= 3

aWinner.setVisibility = View.GONE

alooser.setVisibility =
View.VISIBLE

aempata.setVisibility = View.GONE

ovictorias.text = "VICTORIAS
${user.victorias}"

oderrtas.text = "DERROTAS
${user.derrotas}\t(+1)"

oempates.text = "EMPATES
${user.empates}"

opuntos.text = "PUNTOS
${user.puntos}\t(-3)"
}
"EMPATE" ->{

user.empates += 1

aWinner.setVisibility = View.GONE

alooser.setVisibility = View.GONE

aempata.setVisibility =
View.VISIBLE

```

```
ovictorias.text = "VICTORIAS  
${user.victorias}"  
  
oderrotas.text = "DERROTAS  
${user.derrotas}"  
  
oempates.text = "EMPATES  
${user.empates}\t(+1)"  
  
opuntos.text = "PUNTOS  
${user.puntos}"  
}  
}  
resultado.text =  
extras.getString("RESULTADO")  
  
user.cambiarRango()  
onivel.text =  
"${user.nivel}"  
onick.text =  
user.nick.toString()  
  
Glide.with(this).load(user.image  
URI).into(oimagenperfil)  
  
userProvider.create(user.id,user  
)?.addOnSuccessListener { }  
}  
}  
}
```

En estos ejemplos podemos ver como la implementación de firebase no cambia demasiado pero si añade mucha seguridad al poder comprobar en todo momento que los valores no son null y así evitar problemas tanto en el Front-end como en el back-end.

El código ha quedado más largo en Kotlin a ser más completo puesto que en el anterior solo se imprimía el método `toString` puesto que tener un `TextView` para cada campo habría supuesto mucho código no significativo y repetitivo.

Ejemplo	
java	kotlin
<pre> new IntentIntegrator(this).initiate Scan(); @Override protected void onActivityResult(int requestCode, int resultCode, Intent data) {     IntentResult result = IntentIntegrator.parseActivityResult( requestCode, resultCode, data);     if(result != null) {         if(result.getContents() == null) {  Toast.makeText(this, "Cancelled", Toast.LENGTH_LONG).show();     } else {  Toast.makeText(this, "Scanned: " + result.getContents(), Toast.LENGTH_LONG).show();     } } else {  super.onActivityResult(requestCode, resultCode, data); } </pre>	<pre> bscanner.setOnClickListener{     IntentIntegrator(this)         .initiateScan() }  override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {  super.onActivityResult(requestCode, resultCode, data)     val result = IntentIntegrator.parseActivityResult( requestCode, resultCode, da ta)     if (result != null) {  buscarPartida(result.contents)     }     else {         tscanner.text = "Codigo no valido"     } } </pre>
Implementación oficial	Implementación en el proyecto

Como podemos observar todo lo que sea nuestro propio código será más simple pero la implementación de bibliotecas no tiene porque cambiar demasiado esto es positivo al garantizar la interoperabilidad entre java y Kotlin.

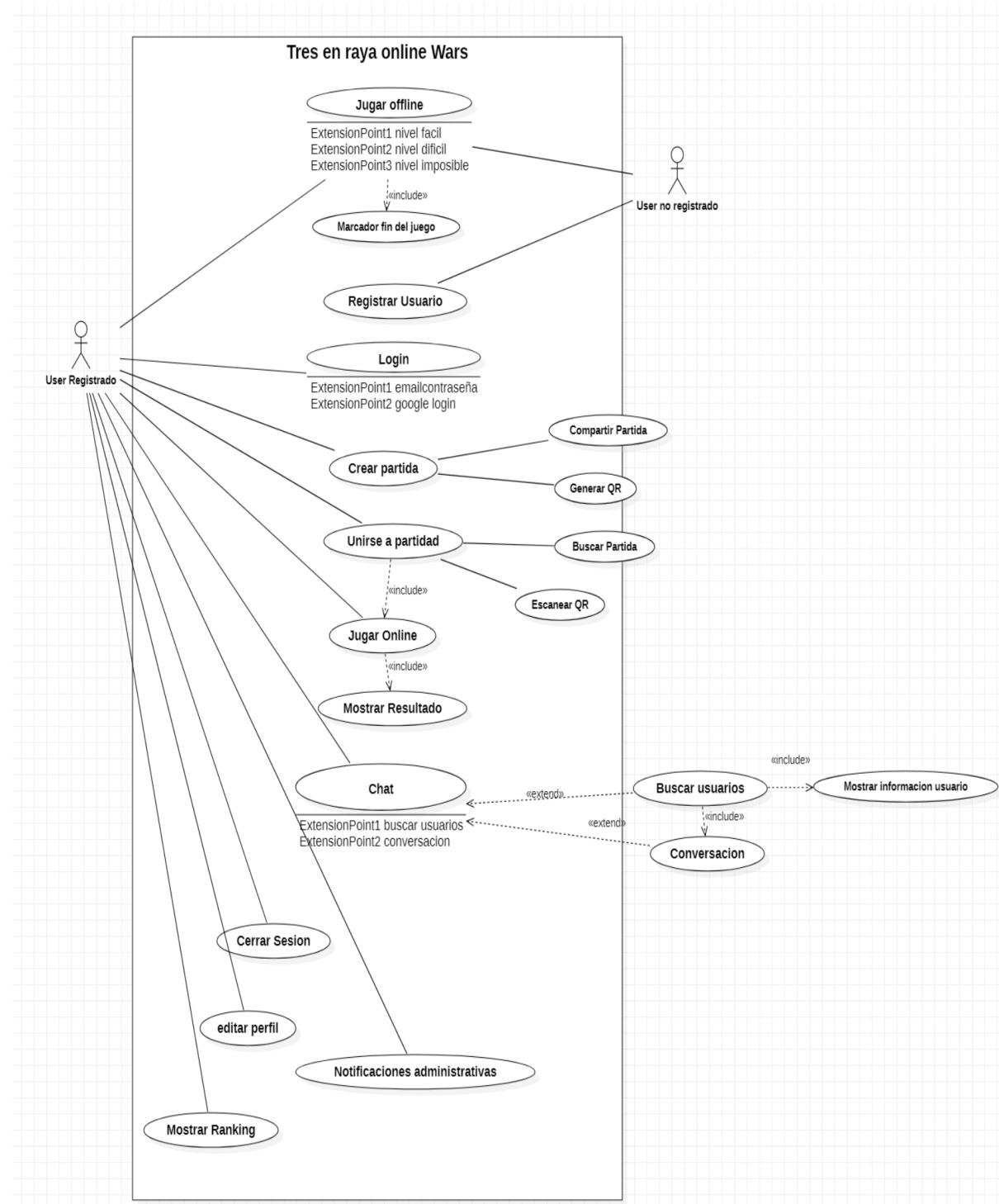
En conclusión Kotlin se presenta como un lenguaje moderno y pragmático que ha aprendido de los errores de sus predecesores y ha conseguido ser un lenguaje a tener en cuenta y la primera o segunda mejor opción para desarrollar en android tan solo teniendo como competencia a java al cual con el tiempo podría poner contra las cuerdas en cuanto se cree una mayor cultura de uso por parte de los desarrolladores y las empresas al tener notables mejoras y la única desventaja significativa, la velocidad con el tiempo está siendo corregida. En el resto de ámbitos a parte de android, Kotlin demuestra ser una herramienta diseñada para ser usada junto a otras tecnologías sin perder su individualidad gracias a su gran interoperabilidad o por si misma pudiendo solucionar deficiencias de las otras tecnologías tanto en desarrollo multiplataforma con KMM, como escritorio, back-end y front-end web al eliminar las principales desventajas de cada tecnología en cada ámbito y gracias a su tipado estático con inferencia de tipos, seguridad y pragmatismo le convierte en la mejor opción para desarrollar proyectos grandes en menor tiempo, con mayor seguridad y sin hacer sacrificios de rendimiento significativos pero esto solo será posible cuando deje de estar encarcelado en Android.

# Desarrollo de la prueba de concepto

## Requisitos

### Funcionales

#### Diagrama de casos de uso



1. Gestión de usuarios
  - a. Guardar datos de usuarios.
  - b. Permitir inicio de sesión a usuarios registrados.
  - c. Persistir la sesión del usuario.
  - d. Editar un perfil ya creado.
  - e. El usuario podrá iniciar sesión sin contraseña.
  - f. El usuario podrá cambiar su contraseña en caso de olvidarla.
  - g. El usuario podrá cerrar sesión.
2. Chat
  - a. Enviar mensajes.
  - b. Recibir mensajes.
  - c. Seleccionar a los usuarios con los que pretende interactuar.
3. Partida offline
  - a. Jugar en diferentes niveles de dificultad.
  - b. Mostrar al usuario el número de partidas ganadas, perdidas y empatadas.
  - c. Abandonar partida
4. Partida online
  - a. Se permitirá a los usuarios crear sus propias partidas .
  - b. Se permitirá a los usuarios generar códigos QR con la información de la partida.
  - c. Abandonar partida.
  - d. Compartir el id de la partida por redes sociales.
  - e. Verificar parte introducida manualmente.
  - f. De no existir partida con oponente aleatorio se creará.
  - g. El usuario podrá escanear el qr de la partida.
  - h. Se permitirá al usuario buscar partidas creadas.
  - i. Se mostrará el resultado de las partidas.
  - j. Se mostrarán todos los resultados de los usuarios ordenados(Ranking).

1. Generales

- a. El lenguaje de programación será Kotlin. Justificación de elección Anexo 1
- b. El Back-end será en firebase.
- c. La aplicación debe verse correctamente en todas las resoluciones.
- d. La aplicación debe ser lo más retrocompatible posible.
- e. Teniendo en cuenta las bibliotecas externas el API de desarrollo será 25.
- f. Los colores de fondos e iconos deben tener contraste para su lectura

2. Chat

- a. Se deben diferenciar los mensajes del emisor y receptor.
- b. Se mostrará la fecha de llegada de cada mensaje.
- c. Se mostrará la información del usuario receptor.
- d. No debe haber latencia entre envío y recepción de mensajes.
- e. Al enviar un mensaje se cambiará el foco de la pantalla.

3. Partida Offline

- a. Se ralentizará los cambios producidos por la máquina para mayor comodidad del usuario.
- b. Guardar los resultados de las partidas en la memoria interna del móvil.
- c. Se indica el nivel de dificultad de la partida.

4. Partida Online

- a. No debe haber latencia entre jugadores.
- b. Se debe indicar de que jugador es el turno.
- c. Se creará una pequeña latencia para que el jugador pueda ver la solución.

## Introducción

En esta parte del proyecto se describirá la base de datos del proyecto, para la parte online se ha utilizado firebase y para la parte offline al ser solo nueve datos numéricos se ha utilizado sharedpreferences. Las sharedpreferences permite guardar datos de tipo clave valor y se usará para guardar las victorias, derrotas y empates del usuario en los tres niveles, además de guardar dos booleanos para que solo vea una vez la retroactivity y una splashactivity resumida.

## Introducción a firebase

Firebase es una plataforma para el desarrollo de aplicaciones web y aplicaciones móviles lanzada en 2011 y adquirida por Google en 2014.

Es una plataforma ubicada en la nube, integrada con Google Cloud Platform, que usa un conjunto de herramientas para la creación y sincronización de proyectos. Utiliza un esquema de base de datos no relacional, documental y sin grafos.

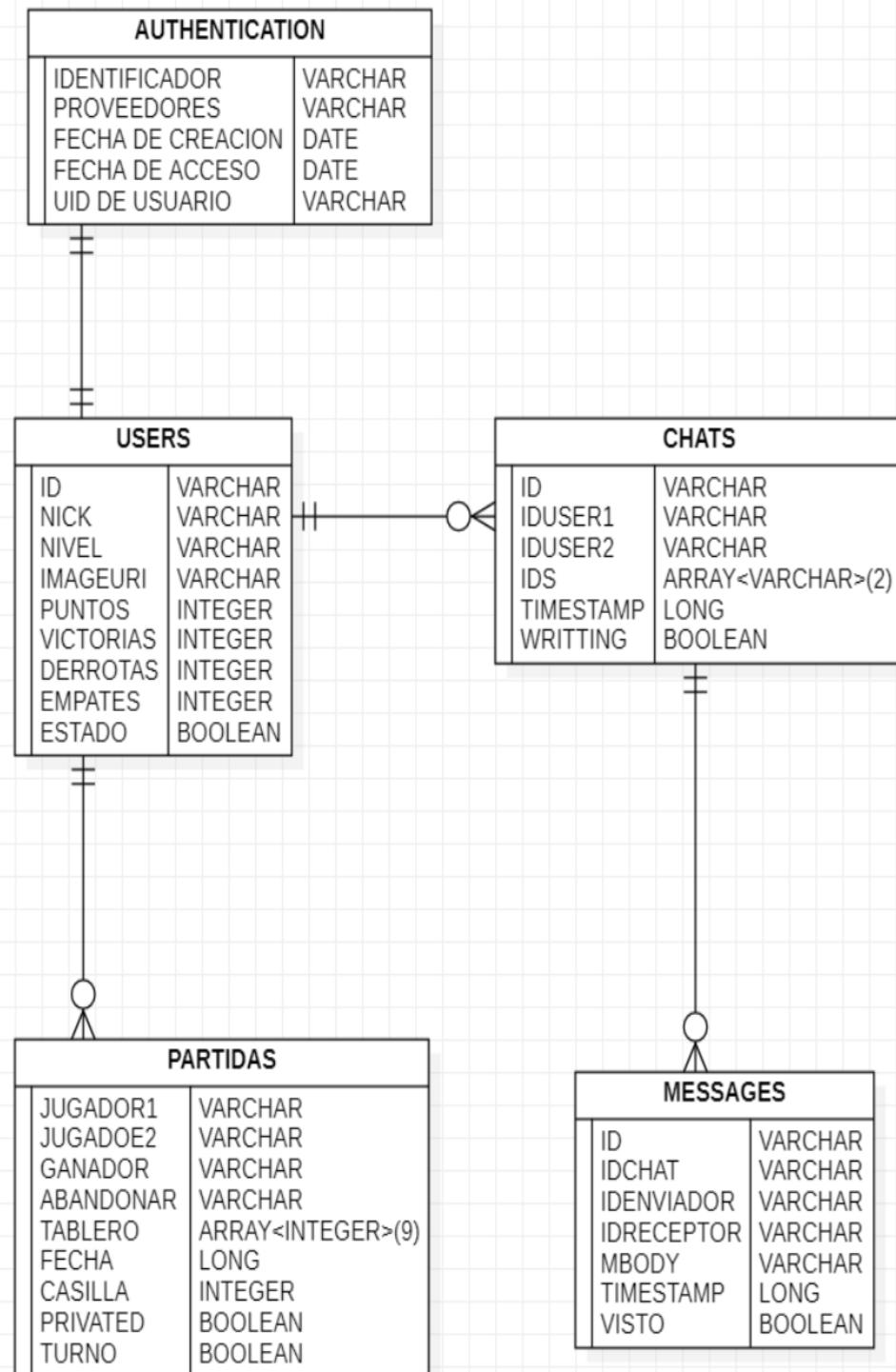
### Principales ventajas:

- Crear proyectos sin crear Back-end lo que ahorra tiempo y dinero.
- Es no relacional lo que hace un diseño menos pesado
- Un conjunto enorme de herramientas tanto en Back-end como en Front-end
- Compatibilidad con prácticamente todas las tecnologías Front-end
- Reactividad de datos
- Gran soporte y comunidad

### Desventajas:

- No es gratis, llegada a cierto tráfico de datos empieza a cobrar.
- El enfoque no relacional puede no ser el adecuado para el proyecto.
- No todos las funciones son gratuitas al principio.
- Dependemos completamente del proveedor.
- No es código abierto.

En conclusión, firebase es la tecnología perfecta para desarrollar de forma rápida y completa aplicaciones con interconexión de usuarios y ofrece un gran catálogo de soluciones pero tener un proyecto grande puede ser mejor idea crear nuestro propio Back-end.



## Firebase

A continuación se presentan las clases usadas como Models y las Clases empleadas para manejar las operaciones con firebase de forma independiente de UI.

Este sería el esquema aproximado de la BBDD que estoy utilizando.

Firebase es una base de datos en realtime, nosql , documental y sin grafos por lo que es difícil de documentar al no haber demasiados ejemplos.

La entidad Authentication aunque está creada por firebase he preferido mostrarla para explicar cómo se relacionan los datos.

### Models

#### Chat

```
data class Chat(val id:String?="",val isWritting:Boolean?=false,val timestamp:Long?=0,  
               var idUser1:String?="",var idUser2:String?="",val ids>List<String>?):Serializable{  
    constructor():this("",false,0,"","","")  
    emptyList()  
}
```

#### Message

```
data class Message(var id:String="", var idEnviador:String="",  
                   var idRecibidor:String="",
                           var idChat:String="", var timestamp:Long=0, var isVisto:Boolean = false,var mBody:String=""){
    constructor():this("", "", "", "", 0, false, "")  
}
```

#### Partida

```
data class Partida(var jugador1:String = "", var jugador2:String = "", var tablero:MutableList<Int>,
                    var ganador:String = "", var turno:Boolean = true, var fecha: Long = 0,
                    var abandonar:String="",var casilla:Int = 0,var isPrivated:Boolean = false):Serializable{
    constructor():this("", "",  
        MutableList(9){0}, "",true,0,"",0,false)  
}
```

```
User
```

```
data class User(     var id:String="", var nick:String = "", var nivel:String = "", var imageURI:String = "", var puntos:Int = 0, var victorias:Int = 0, var derrotas:Int = 0, var empates:Int=0, var estado:Boolean = false){     constructor():this("", "", "", "", 0, 0, 0, 0, false)}
```

```
    fun cambiarRango(){         if (puntos >= 0){             when(puntos){                 in 0..5 -> nivel = "PRINCIPIANTE"                 in 6..20-> nivel = "EXPERIMENTADO"                 in 21..50-> nivel ="EXPERTO"                 else-> nivel = "GRAN EXPERTO"             }         }         else{             nivel ="NOVATO"         }     }
```

```
}
```

```
AuthProvider

class AuthProvider {
    var mAuth: FirebaseAuth
    init {
        mAuth = FirebaseAuth.getInstance()
        mAuth.languageCode = "es"
    }
    fun register(email: String?, password: String?):
Task<AuthResult?>? {
        return mAuth!!.createUserWithEmailAndPassword(email,
password)
    }

    fun login(email: String?, password: String?):
Task<AuthResult?>? {
        return mAuth!!.signInWithEmailAndPassword(email, password)
    }

    fun getEmail(): String? {
        return if (mAuth!!.currentUser != null) {
            mAuth!!.currentUser.email
        } else {
            null
        }
    }

    fun getUserId(): String? {
        return if (mAuth!!.currentUser != null) {
            mAuth!!.currentUser.uid
        } else {
            null
        }
    }

    fun getUserSession(): FirebaseUser? {
        return if (mAuth!!.currentUser != null) {
            mAuth!!.currentUser
        } else {
            null
        }
    }

    fun logout() {
        if (mAuth != null) {
            mAuth!!.signOut()
        }
    }

    fun signGoogle(credential: AuthCredential): Task<AuthResult> {
        return mAuth.signInWithCredential(credential)
    }
}
```

**ChatProvider**

```
class ChatProvider {
    var mCollection: CollectionReference? = null
    init {
        mCollection =
            FirebaseFirestore.getInstance().collection("Chats")
    }
    fun create(chat: Chat) {
        mCollection!!.document(chat.idUser1 +
            chat.idUser2).set(chat)
    }
    fun getAll(idUser: String): Query {
        return mCollection!!.whereArrayContains("ids", idUser)
    }
    fun getChatByUser1AndUser2(idUser1: String, idUser2: String): Query? {
        val ids: ArrayList<String> = ArrayList()
        ids.add(idUser1 + idUser2)
        ids.add(idUser2 + idUser1)
        return mCollection!!.whereIn("id", ids)
    }
}
```

**MessageProvider**

```
class MessageProvider {
    var mCollection: CollectionReference? = null
    init {
        mCollection =
            FirebaseFirestore.getInstance().collection("Messages")
    }
    fun create(message: Message): Task<Void?>? {
        val document = mCollection!!.document()
        message.id = document.id
        return document.set(message)
    }
    fun getMessageByChat(idChat: String?): Query? {
        return mCollection!!.whereEqualTo("idChat",
            idChat).orderBy("timestamp", Query.Direction.ASCENDING)
    }
    fun getMessagesByChatAndSender(idChat: String?, idSender: String?): Query? {
        return mCollection!!.whereEqualTo("idChat",
            idChat).whereEqualTo("idSender", idSender).whereEqualTo("viewed",
            false)
    }
}
```

**PartidaProvider**

```

class PartidaProvider {
    private var mCollection: CollectionReference
    init { }
        mCollection =
    FirebaseFirestore.getInstance().collection("partidas")
    }
    fun savePartida(partida:Partida): Task<DocumentReference> {
        return mCollection.add(partida)
    }
    fun updatePartida(id: String,partida: Partida): Task<Void> {
        return mCollection.document(id).set(partida)
    }
    fun getAllWhereJugador2Empty(): Task<QuerySnapshot> {
        return
    mCollection.whereEqualTo("jugador2", "").whereEqualTo("privated", f
    alse).get()
    }
    fun getPartida(id:String): DocumentReference {
        return mCollection.document(id)
    }
    fun deletePartida(id:String): Task<Void> {
        return mCollection.document(id).delete()
    }
}

```

**UserProvider**

```

class UserProvider {
    private var mCollection: CollectionReference
    init { }
        mCollection =
    FirebaseFirestore.getInstance().collection("Users")
    }
    fun create(id:String,user: User): Task<Void?>? {
        return mCollection.document(id).set(user)
    }
    fun read(id: String): Task<DocumentSnapshot> {
        return mCollection.document(id).get()
    }
    fun getCollectionReference():CollectionReference{
        return mCollection
    }
}

```

Al no tener integridad referencial debo hacerla desde el front-end.

### RegisterActivity

```
private fun crearUser(usuario: User){  
    authProvider.getUid()?.let { it1 ->  
        userProvider.create(it1, usuario)  
            ?.addOnSuccessListener {  
                Toast.makeText(this, "Usuario Registrado",  
                    Toast.LENGTH_LONG).show()  
                startActivity(Intent(this@RegisterActivity, MainOnlineActivity::cl  
ass.java)) }  
            }?.addOnFailureListener{  
                Toast.makeText(this, "Usuario No Registrado",  
                    Toast.LENGTH_LONG).show()  
            }  
    }  
}
```

### ChatActivity

El id de cada documento es la combinación del UID TOKEN del usuario creador y el id del usuario con el que desee hablar.

```
fun crearChat(){  
    Toast.makeText(this, "Chat creado", Toast.LENGTH_LONG).show()  
    val ids = listOf(idUser1, idUser2)  
    var chat =  
        ChatidUser1+idUser2, false, Date().time, idUser1, idUser2, ids)  
        chatid = idUser1+idUser2  
        ChatProvider().create(chat)  
    }
```

### ChatActivity

Desde front-end se controla que no se generen duplicidades

```
fun checkIfChatExist() {
    ChatProvider().getChatByUser1AndUser2(idUser1, idUser2)?.get()?.addOnSuccessListener { it->
        if (it.isEmpty()) {
            Toast.makeText(this, "Creando chat...", Toast.LENGTH_LONG).show()
            crearChat()
        } else {
            chatId = it.documents[0].id
            Toast.makeText(this, "Mostrando chat", Toast.LENGTH_LONG).show()
        }
    }?.addOnFailureListener {
        Toast.makeText(this, "Error al crear el chat", Toast.LENGTH_LONG).show()
    }
}
```

### LoginActivity

Evitar que se registre el usuario cada vez que inicie sesión con google

```
private fun verificarUser() {
    AuthProvider().getUid()?.let { it1 ->
        UserProvider().read(it1).addOnSuccessListener {
            if (it.exists()){
                startActivity(Intent(this@LoginActivity, MainOnlineActivity::class.java))
            } else{
                startActivity(Intent(this@LoginActivity, CompletarPerfilActivity::class.java))
            }
        }
    }
}
```

```

{
  "Chats": {
    "K54tRlrezoTUi0roqgsWC1G3fyh10Nb3F1K": {
      "id": "K54tRlrezoTUi0roqgsWC1G3fyh10Nb3F1KNe9eCif0W7xXkvyJVJTD3",
      "idUser1": "K54tRlrezoTUi0roqgsWC1G3fyh1",
      "idUser2": "0Nb3F1KNe9eCif0W7xXkvyJVJTD3",
      "ids": [
        0: "K54tRlrezoTUi0roqgsWC1G3fyh1",
        1: "0Nb3F1KNe9eCif0W7xXkvyJVJTD3"
      ],
      "timestamp": 1621546737971,
      "writing": false
    }
  }
}
  
```

Identificador	Proveedores	Fecha de creación	Fecha de acceso	UID de usuario
spider@gmail.com		19 may. 2021	19 may. 2021	0Nb3F1KNe9eCif0W7xXkvyJVJTD3
proyectorofirebase2021@gm...		23 may. 2021	23 may. 2021	A1DbSCimREaKKw0l7ebjJbO57ty2
lacuna@coil.com		22 may. 2021	30 may. 2021	BbKwR5EInuPwyjtL8WZt0DmJH123
homer@gmail.com		19 may. 2021	19 may. 2021	CX56h93CGufjXQv0wDXTc39yuDI2
kwkw@gmai.com		13 may. 2021	13 may. 2021	CYvi5GFbovNKv5FaolshXnxQgEv1
demod3108@gmail.com		23 may. 2021	23 may. 2021	H0hJFMwTiqOTvZorwWKT1ZrDa...
zuberoa@gmail.com		20 may. 2021	20 may. 2021	K54tRlrezoTUi0roqgsWC1G3fyh1
john@gmail.co		15 may. 2021	15 may. 2021	OeJ01BblXnZzs6nohxluTOjfMyH2
blackmetal@metal.com		19 may. 2021	19 may. 2021	RP0FJJWqZHhM333Gwr45Zm86K...
xiaomi@gmail.com		18 may. 2021	18 may. 2021	V7Qds8xVlmdtUMeg0LzaxWBN5b...
redson@gmail.com		19 may. 2021	19 may. 2021	VpEF04Kh5UfWb4mJv3Taed2vsw...
juan@doe.com		13 may. 2021	22 may. 2021	ZZOyGR5M2QXlbKQ6hOqBiTfm3c...

projectofinal-adb58 > Messages > B6eFcwpEHLdIT0clhhE4

+ Iniciar colección	+ Agregar documento	+ Iniciar colección
Chats	+ Agregar campo	
Messages >		
Users		
partidas		
	D9YVybjRFF2fH16nDTcI In7CYsafx7QjbCwGySQM JLuwL026xGxEUEP5gAS7 NTD08D3VE26chMNPaVA P5NaBweupoCkk6oBcZj7 Ri1VSpT0cSOEaMV78C5D SkPYsqSwr8vPYfxVoWVC 13YxKe8UKpSxdkKne6Ig u6FsLTdByKvhEztJA5iz	

projectofinal-adb58 > Users > 0Nb3FikNe9eCif0W7xXkvyJVJTD3

+ Iniciar colección	+ Agregar documento	+ Iniciar colección
Chats	+ Agregar campo	
Messages		
Users >		
partidas	derrotas: 90 estado: false id: "0Nb3FikNe9eCif0W7xXkvyJVJTD3" imageURI: "https://firebasestorage.googleapis.com/v0/b/projectofinal-adb58.appspot.com/o/users%2Fspider?alt=media&token=0967a585-3e6c-40d8-b52b-b3c40e7ee4ef" nick: "spider" nivel: "PRINCIPIANTE" puntos: 790 victorias: 90	

Ubicación de Cloud Firestore: nam5 (us-central)

The screenshot displays the MongoDB Compass interface. On the left, the sidebar shows the database 'proyectofinal-adb58' and the collection 'partidas' (which is selected). The main area shows a list of documents in the 'partidas' collection, with one document expanded to show its fields. The expanded document includes fields such as 'abandonar', 'casilla', 'fecha', 'ganador', 'jugador1', 'jugador2', 'privated', and 'tablero'. The 'tablero' field is an array containing 8 elements, each with a value (0, 1, 2, 3, 4, 5, 6, 7) and a count (0). The document ID shown is 09mSnwl28wbckkmNMW2t.

```

package com.example.proyectofinal.services

import android.app.Notification
import android.app.NotificationChannel
import android.app.NotificationManager
import android.app.PendingIntent
import android.content.Context
import android.content.Intent
import android.media.RingtoneManager
import android.net.Uri
import android.os.Build
import android.util.Log
import androidx.core.app.NotificationCompat
import com.example.proyectofinal.MainActivity
import com.example.proyectofinal.R
import com.google.firebase.messaging.FirebaseMessagingService
import com.google.firebase.messaging.RemoteMessage
class MyFirebaseMessagingClient: FirebaseMessagingService() {
    private val TAG = "FireBaseMessagingService"
    val NOTIFICATION_ID = 100
    val NOTIFICATION_CHANNEL_ID :String =
"com.example.proyectofinal"
    override fun onMessageReceived(remoteMessage: RemoteMessage) {
        super.onMessageReceived(remoteMessage)
        Log.e("message", "Message Received ...")
        if (remoteMessage.data.size > 0) {
            val title = remoteMessage.data["title"]
            val body = remoteMessage.data["body"]
            showNotification(applicationContext, title, body)
        } else {
            val title = remoteMessage.notification!!.title
            val body = remoteMessage.notification!!.body
            showNotification(applicationContext, title, body)
        }
    }
    override fun onNewToken(p0: String) {
        if (p0 != null) {
            super.onNewToken(p0)
        }
        Log.e("token", "New Token")
    }
    fun showNotification(
        context: Context,
        title: String?,
        message: String?
    ) {
        val ii: Intent
        ii = Intent(context, MainActivity::class.java)
        ii.data = Uri.parse("custom://" +
System.currentTimeMillis())
        ii.action = "actionstring" + System.currentTimeMillis()
        ii.flags = Intent.FLAG_ACTIVITY_SINGLE_TOP or
Intent.FLAG_ACTIVITY_CLEAR_TOP
        val pi =
    
```

```

        PendingIntent.getActivity(context, 0, ii,
PendingIntent.FLAG_UPDATE_CURRENT)
    val notification: Notification
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        //Log.e("Notification", "Created in up to orio OS
device");
        notification = NotificationCompat.Builder(context,
NOTIFICATION_CHANNEL_ID)
            .setOngoing(true)
            .setSmallIcon(getNotificationIcon())
            .setContentText(message)
            .setAutoCancel(true)
            .setContentIntent(pi)
            .setPriority(NotificationCompat.PRIORITY_HIGH)
            .setCategory(Notification.CATEGORY_SERVICE)
            .setWhen(System.currentTimeMillis())
.setSound(RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION))
            .setContentTitle(title).build()
    val notificationManager = context.getSystemService(
        Context.NOTIFICATION_SERVICE
    ) as NotificationManager
    val notificationChannel = NotificationChannel(
        NOTIFICATION_CHANNEL_ID,
        title,
        NotificationManager.IMPORTANCE_DEFAULT
    )
    notificationManager.createNotificationChannel(notificationChannel)
)
    notificationManager.notify(NOTIFICATION_ID,
notification)
} else {
    notification = NotificationCompat.Builder(context)
        .setSmallIcon(getNotificationIcon())
        .setAutoCancel(true)
        .setContentText(message)
        .setContentIntent(pi)
.setSound(RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION))
        .setContentTitle(title).build()
    val notificationManager = context.getSystemService(
        Context.NOTIFICATION_SERVICE
    ) as NotificationManager
    notificationManager.notify(NOTIFICATION_ID,
notification)
}
}
private fun getNotificationIcon(): Int {
    val useWhiteIcon =
        Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP
    return if (useWhiteIcon) R.mipmap.ic_launcher else
R.mipmap.ic_launcher
}
}

```

Aunque no es una base de datos me ha parecido relevante mostrar cómo se manejan en Android esta forma de persistencia más ligera y más indicada cuando solo queremos guardar ciertos valores de forma persistente sin la complejidad añadida de una base de datos local real.

### Implementación

```
lateinit var preferences:SharedPreferences
lateinit var editor:SharedPreferences.Editor
@SuppressLint("CommitPrefEdits")
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_partida_offline)
    preferences = this?.getPreferences(Context.MODE_PRIVATE)
    editor = preferences.edit()
}
```

### Create/Update

En caso de no existir el valor se creará, en caso de existir se actualizará.

El dato que se actualizará se guardará previamente en una variable si no existe será cero por lo que al sumarle uno se guardará como uno.

```
@SuppressLint(" ResourceType")
private fun guardarDatos() {
    when(campeon) {
        "HUMAN" -> {
            when(nivel) {
                1 -> {
                    var aux = preferences.getInt("victorias1", 0)
                    editor.putInt("victorias1", aux+1)
                    editor.commit()
                }
                2 -> {
                    var aux = preferences.getInt("victorias2", 0)
                    editor.putInt("victorias2", aux+1)
                    editor.commit()
                }
                3 -> {
                    var aux = preferences.getInt("victorias3", 0)
                    editor.putInt("victorias3", aux+1)
                    editor.commit()
                }
            }
        }
        "MACHINE" -> {
            when(nivel) {
                1 -> {
                    var aux = preferences.getInt("derrotas1", 0)
                    editor.putInt("derrotas1", aux+1)
                }
            }
        }
    }
}
```

```
        editor.commit()
    }
2->{
    var aux = preferences.getInt("derrotas2", 0)
    editor.putInt("derrotas2", aux+1)
    editor.commit()
}
3->{
    var aux = preferences.getInt("derrotas3", 0)
    editor.putInt("derrotas3", aux+1)
    editor.commit()
}
}
}

"EMPATE"->{
    when(nivel){
        1->{
            var aux = preferences.getInt("empates1", 0)
            editor.putInt("empates1", aux+1)
            editor.commit()
        }
        2->{
            var aux = preferences.getInt("empates2", 0)
            editor.putInt("empates2", aux+1)
            editor.commit()
        }
        3->{
            var aux = preferences.getInt("empates3", 0)
            editor.putInt("empates3", aux+1)
            editor.commit()
        }
    }
}
```

## Select

```
when (nivel) {
    1->{ [REDACTED]
        dialog.showVictoriasP.text =
preferences.getInt("victorias1", 0).toString()
        dialog.showDerrotasaP.text = [REDACTED]
preferences.getInt("derrotas1", 0).toString()
        dialog.showEmpatesP.text =
preferences.getInt("empates1", 0).toString()
    } [REDACTED]
    2->{ [REDACTED]
        dialog.showVictoriasP.text =
preferences.getInt("victorias2", 0).toString()
        dialog.showDerrotasaP.text = [REDACTED]
preferences.getInt("derrotas2", 0).toString()
    }
}
```

```
        dialog.showEmpatesP.text =  
preferences.getInt("empates2", 0).toString()  
    }  
    3->{  
        dialog.showVictoriasP.text =  
preferences.getInt("victorias3", 0).toString()  
        dialog.showDerrotasaP.text =  
preferences.getInt("derrotas3", 0).toString()  
        dialog.showEmpatesP.text =  
preferences.getInt("empates3", 0).toString()  
    }  
}
```

### Implementaciones reseñables

Nombre	Lottie
Gradle	<code>implementation "com.airbnb.android:lottie:3.6.0"</code>
Descripción	Biblioteca más utilizada para mostrar animaciones en android
Xml	<pre>&lt;com.airbnb.lottie.LottieAnimationView     android:id="@+id/aWinner"     android:layout_width="0dp"     android:layout_height="169dp"     android:visibility="gone"     android:layout_alignParentBottom="true"     app:layout_constraintBottom_toBottomOf="parent"     app:layout_constraintEnd_toEndOf="parent"     app:layout_constraintHorizontal_bias="0.0"     app:layout_constraintStart_toStartOf="parent"     app:layout_constraintTop_toTopOf="parent"     app:layout_constraintVertical_bias="0.028"     app:lottie_autoPlay="true"     app:lottie_loop="true"     app:lottie_rawRes="@raw/onlinewinner" /&gt;</pre>
Muestra	 <p>The screenshot shows a mobile application interface. At the top, there is a trophy icon on a podium with three colored steps (orange, pink, blue). Below this, there is a navigation bar with icons for user profile, medal, handshake, and thumbs down. The main content area displays two player profiles. The first player, 'spider', is listed as 'GRAN EXPERTO' with stats: 790, 90, 0, 90. The second player, 'Nick230', is also listed as 'GRAN EXPERTO' with stats: 491, 5, 1, 8. At the bottom of the screen, there is a navigation bar with icons for profile, settings, ranking, and search.</p>

Nombre	CircleImageView
Gradle	<code>implementation 'de.hdodenhof:circleimageview:3.1.0'</code>
Xml	<pre>&lt;de.hdodenhof.circleimageview.CircleImageView     android:id="@+id/ifoto"     android:layout_width="110dp"     android:layout_height="110dp"     android:layout_alignParentEnd="true"      android:layout_alignParentBottom="true"     android:layout_marginEnd="141dp"     android:layout_marginBottom="410dp"     android:src="@drawable/shootinggrande" /&gt;</pre>
Muestra	 <p>The screenshot displays a user profile screen. At the top, there is a circular image of an angel with blue wings holding a candle. Below the image, the text "Nick: angeomon" and "Nivel: NOVATO" is shown. Further down, the stats "Puntos: -21", "Victorias: 8", and "Derrotas: 15" are listed. At the bottom left, a button says "Cargando usuario". The bottom of the screen features standard Android navigation icons: back, home, and recent apps.</p>

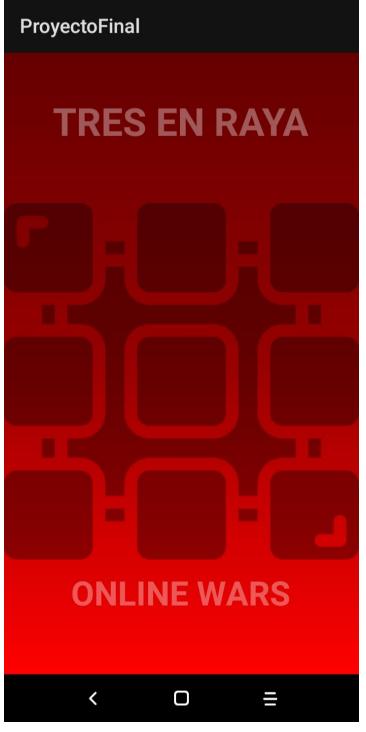
Nombre	Animación desarrollada por mi
Kotlin	<pre>var animation = AnimationUtils.loadAnimation(this,R.anim.animation1) textView.startAnimation(animation) textView2.startAnimation(animation) imageView.startAnimation(animation)</pre>
Xml	<pre>&lt;?xml version="1.0" encoding="utf-8"?&gt; &lt;alpha     xmlns:android="http://schemas.android.com/apk/res/android"     android:fromAlpha= "0.0"     android:toAlpha = "1.0"     android:duration = "3000"&gt; &lt;/alpha&gt;</pre>

Nombre	Gradient
Xml	<pre>&lt;?xml version="1.0" encoding="utf-8"?&gt; &lt;shape     xmlns:android="http://schemas.android.com/apk/res/android"&gt;     &lt;gradient         android:angle="90"         android:startColor="#FF0000"         android:centerColor="#980000"         android:endColor="#650000"     /&gt; &lt;/shape&gt;</pre>
Muestra	

Nombre	QR autogenerado
Gradle	<pre>implementation 'com.github.kenglxn.QRGen:android:2.3.0'</pre>
XML	<pre>&lt;ImageView     android:id="@+id/mostrarQR"     android:layout_width="150dp"     android:layout_height="150dp"     android:layout_marginTop="60dp"     android:src="@drawable/arcade"     app:layout_constraintEnd_toEndOf="parent"     app:layout_constraintHorizontal_bias="0.498"     app:layout_constraintStart_toStartOf="parent"      app:layout_constraintTop_toBottomOf="@+id/textView3"     &gt;</pre>
Kotlin	<pre>dialog.mostrarQR.setImageBitmap(QRCode.from(partida id).withSize(100,100).bitmap())</pre>
Muestra	 <p>The screenshot shows a red-themed game interface. At the top, it says "PARTIDA CREADA". In the center is a QR code. Below the QR code is a green circular button with a white arrow pointing left. Below the QR code, the text "ESPERANDO ..." is displayed above a progress bar. At the bottom, there is a button labeled "CANCELAR PARTIDA". The bottom of the screen features a black navigation bar with three icons: a left arrow, a square, and a menu icon.</p>

Nombre	Elegir imagen de galería/Sacar foto
Gradle	<pre>implementation 'com.opensoq.supernova:gligar:1.1.0'</pre>
Kotlin	<pre>    icamara.setOnClickListener{         GligarPicker().limit(this.limite).disableCamera(false)         .cameraDirect(false).requestCode(PICKER_REQUEST_CODE)     }     .withActivity(this).show() }  override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {     super.onActivityResult(requestCode, resultCode, data)     if (resultCode != RESULT_OK) {         return     }     when (requestCode) {         PICKER_REQUEST_CODE -&gt; {             val imagesList = data?.extras?.getStringArray(GligarPicker.IMAGES_RESULT)             if (!imagesList.isNullOrEmpty()) {                 ifoto.setImageURI(imagesList[0].toUri())             }         }     } }</pre>
Muestra	

Nombre	Konfeti
Gradle	<code>implementation 'nl.dionsegijn:konfetti:1.2.2'</code>
XML	<code>&lt;nl.dionsegijn.konfetti.KonfettiView     android:layout_width="match_parent"         android:layout_height="match_parent"         android:id="@+id/confetti"     /&gt;</code>
Kotlin	<code>private fun generarConfetti() {     confetti.build()      .addColors(Color.BLACK, Color.BLUE, Color.YELLOW)         .setDirection(0.0, 359.0)         .setSpeed(1f, 1f)         .setFadeOutEnabled(true)         .setTimeToLive(2000L)         .addShapes(Shape.Circle, Shape.Square)         .addSizes(Size(12))         .setPosition(-50f, confetti.width + 50f, -50f, -50f)         .streamFor(300, 5000L) }</code>
Muestra	<p>ProyectoFinal</p>  <p>The screenshot shows a red-themed victory screen. At the top, it says 'GANADOR'. Below that is a circular profile picture of a person. To the right of the profile picture, the player's name 'Nick230' is displayed, followed by the title 'GRAN EXPERTO'. Below this, statistics are shown: 'VICTORIAS 33 (+1)', 'DERROTAS 22', 'EMPATES 3', and 'PUNTOS 533 (+3)'. At the bottom, there is a button labeled 'VOLVER A MENU'.</p>

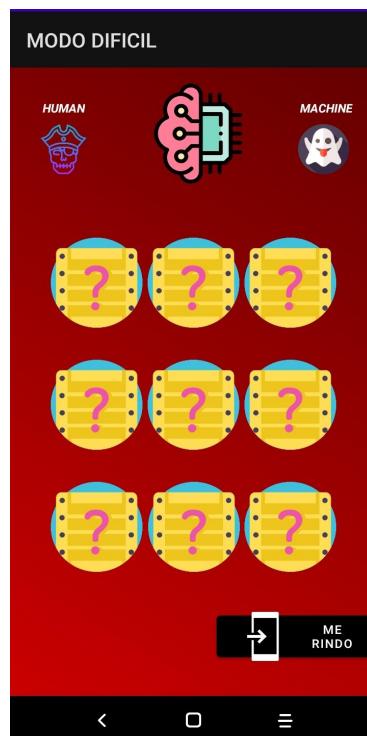
Activity	Funciones
MainActivity(Actuará de SplashActivity)	Mostrará una animación al inicio de la app
	
IntroActivity	Mostrará una descripción de la app en diversas activities
	

SelectorActivity	Dispondrá de 2 botones: 1. Enviar a OfflineActivity 2. Enviar a LoginActivity
	
Offline	
OfflineActivity	<p>Mostrará 4 botones</p> <ol style="list-style-type: none"> <li>1. Una flecha para volver al MainActivity</li> <li>2. Nivel fácil (enviará 1 al PartidaActivity)</li> <li>3. Nivel difícil(enviará 2 al PartidaActivity)</li> <li>4. Nivel imposible(enviará 3 al PartidaActivity)</li> </ol>



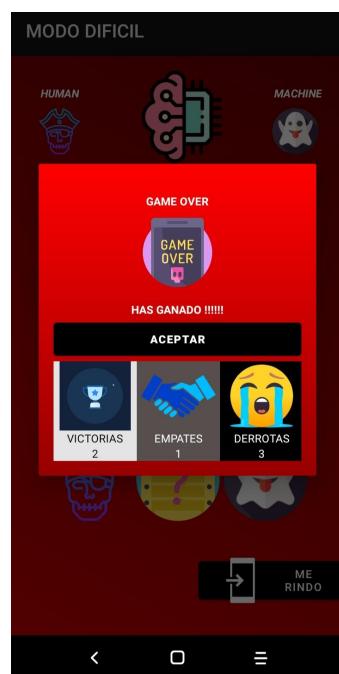
PartidaActivity

Recibirá el parámetro de la OfflineActivity en función del parámetro una IA jugará contra el jugador para que la partida sea más agradable para el jugador ha sido animada



AlertOfflineActivity

Al acabar la partida en la PartidaActivity se superpondrán mostrando el resultado y los datos almacenados en sharedpreferences



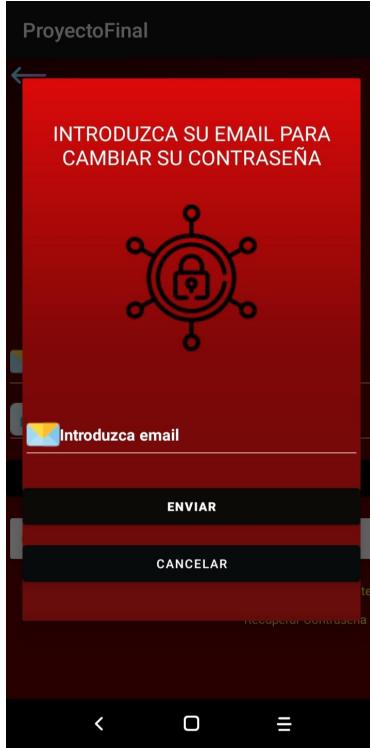
## Online

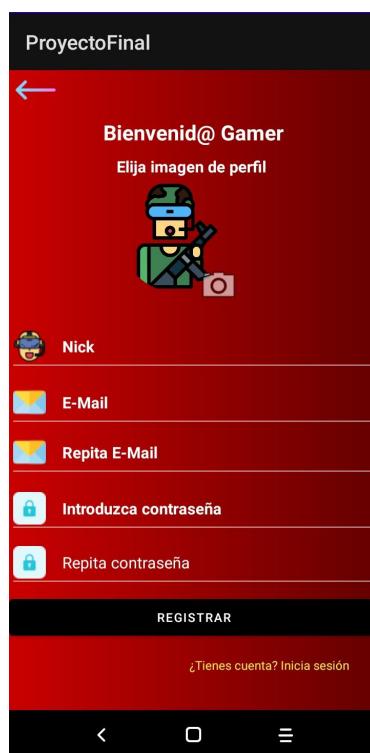
LoginActivity

En caso de que el usuario haya logueado previamente esta pantalla no se verá. En caso opuesto tendrá 4 funciones

1. Envía a RegisterActivity
2. Inicia sesión con email + contraseña
3. Inicia sesión con google sin contraseña
4. Envía a OfflineActivity
5. Mostrar dialog para cambiar contraseña



alertpassword	Recibira un email de ser valido enviara un email con el cambio de contraseña.
	
RegisterActivity	<ol style="list-style-type: none"><li>1. Formulario que recoge la información del usuario</li><li>2. Valida los datos a través de expresiones regulares</li><li>3. La imagen que se muestra es la imagen de perfil por defecto de pulsar en ella o en la camara se abrirá un activity para que el usuario pueda elegir de galería o sacar una foto</li><li>4. Volver a LoginActivity</li></ol>



### MainOnlineAcitivy

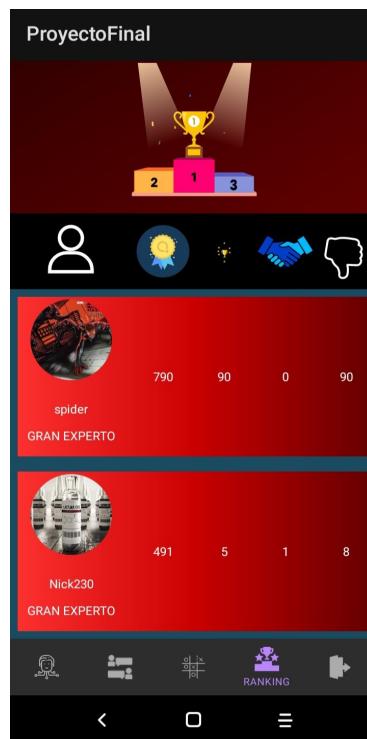
Dividida en 5 fragments con 5 funciones distintas

La activity solo se encarga de cargar los fragments y recoger los onActivityResult al no poderlo hacer en los fragments



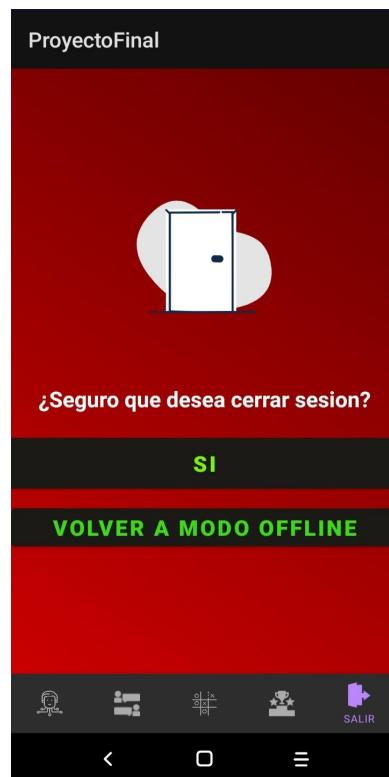
RankingFragment

Muestra un firestorecycler con las información de los usuarios ordenada por puntuación



LogOutFragment

1. Envía al modo offline sin cerrar sesión
2. Envía al modo offline y cierra la sesión



PerfilFragment

1. Muestra la información del usuario
2. Permite editar el nombre y la foto del usuario



#### GameBlankFragment

1. Buscar partida por id después de recibir el id de la partida de ser correcto envia a PartidaOnlineActivity
2. Enviar a CrearPartidaActivity
3. Envía a ScannerActivity
4. Enviar a Oponente Aleatorio

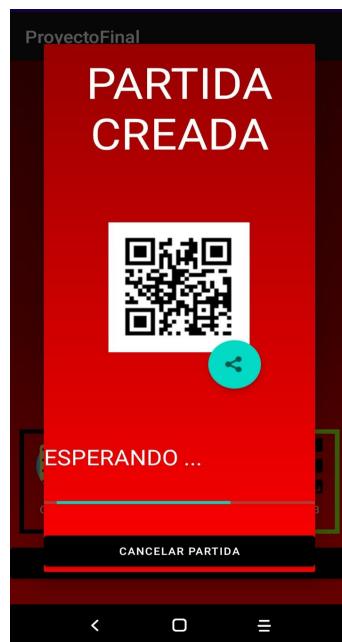


OponenteAleatorioActivity

1. Creará la partida
2. En caso de que el otro usuario se conecte enviará a PartidaOnlineActivity



CrearPartidaActivtiy	1. Permitirá elegir el turno del jugador 2. Permitirá elegir la casilla de la partida 3. Mostrará MostrarAlertInfo
	
MostrarInfoAlert	1. Mostrará QR con el ID de la partida 2. Proporciona un botón para compartir por redes sociales el id de la partida 3. En caso de querer cancelar la partida la borrara 4. En caso de unirse un jugador a la partida enviará a PartidaOnlineActivity



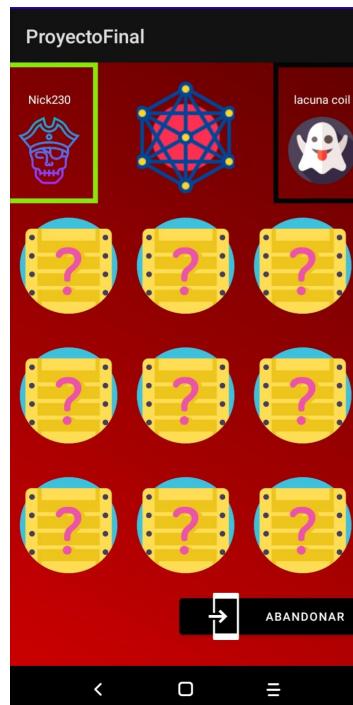
ScannerActivity

1. Tendrá un botón para escanear el código QR
2. Informará al usuario mediante animaciones del resultado
3. En caso del id ser valido enviara a GameOnlineActivity



PartidaOnlineActivity

1. Provee de una partida interactiva para los usuarios que firebase actualiza
2. Botón para abandonar la partida
3. En caso de cerrar la app la partida se dará por abandonada
4. Cuando acabe la partida enviara ResultOnlineActivity



## ResultOnlineActivity

1. Muestra el resultado de la partida
2. Actualiza al usuario en función del resultado



## ChatFragment

1. Muestra un listado con los chats y permite el acceso a cada chat
2. Envía a ChatActivity



ContactActivity

Mostrará un listado de los usuarios con su foto y nombre de pulsar en alguno enviará a OtherUserActivity

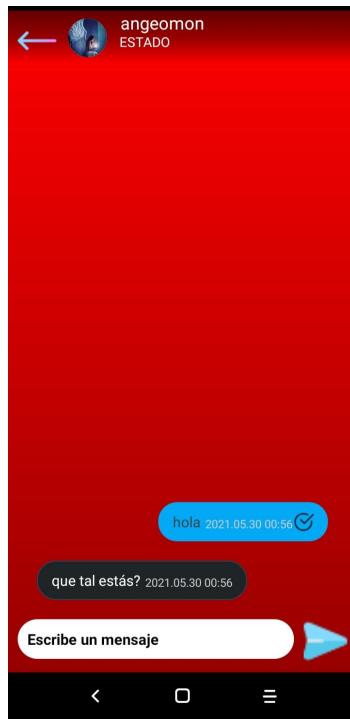


OtherUserActivity

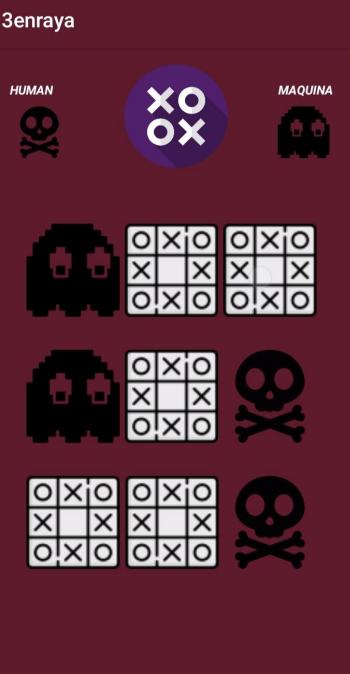
1. Mostrará la información del usuario
2. Tendrá un botón que creará el chat y nos enviará a ChatActivity

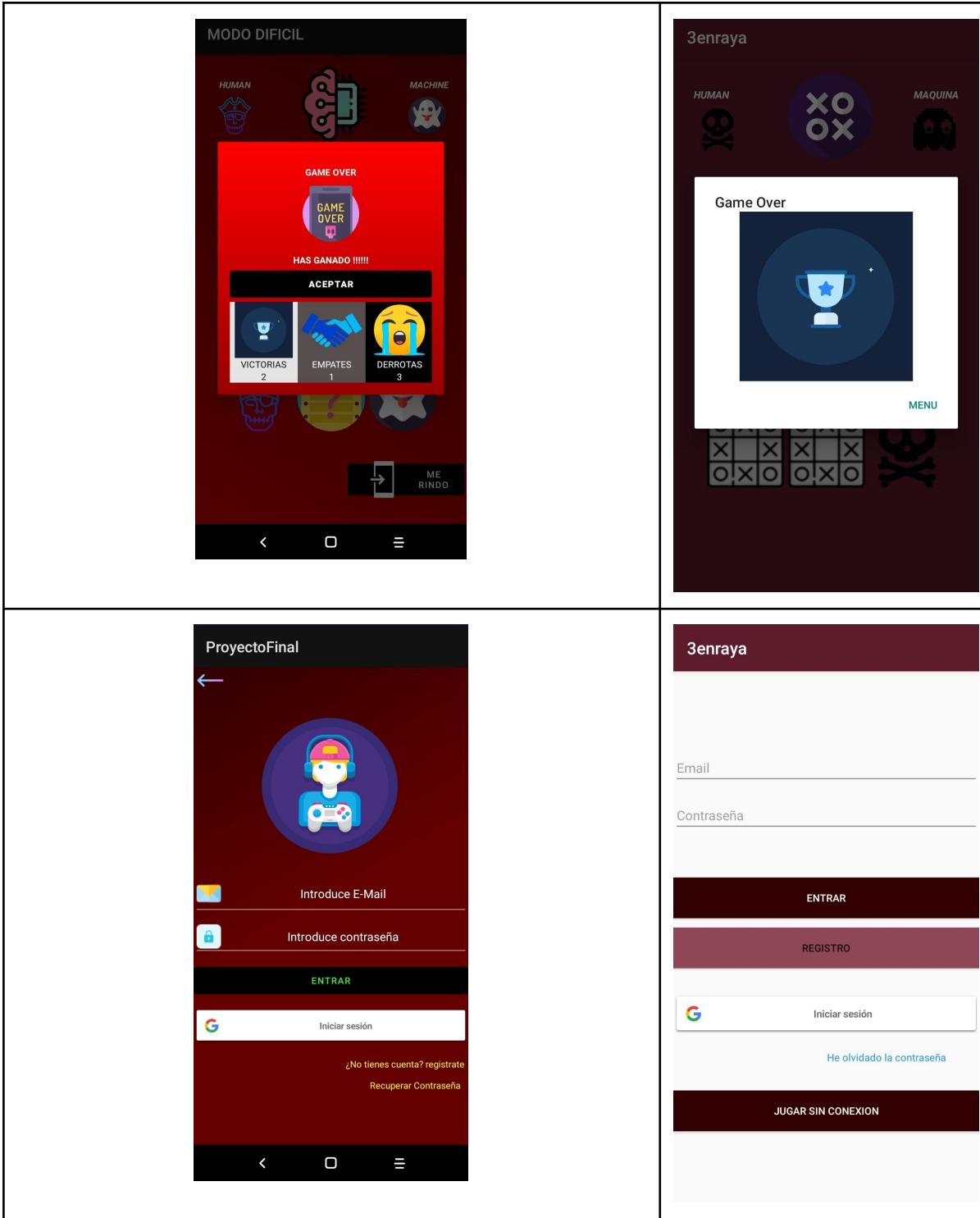


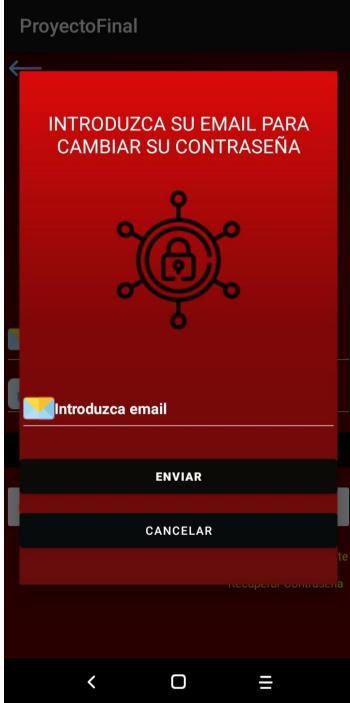
ChatActivity	<ol style="list-style-type: none"><li>1. Mostrar mensajes</li><li>2. Enviar mensajes</li><li>3. Mostrará en el toolbar la información del usuario</li><li>4. En caso de entrar desde OtherUserActivity se cerrará automáticamente por razones de seguridad e integridad de datos</li></ol>
--------------	--

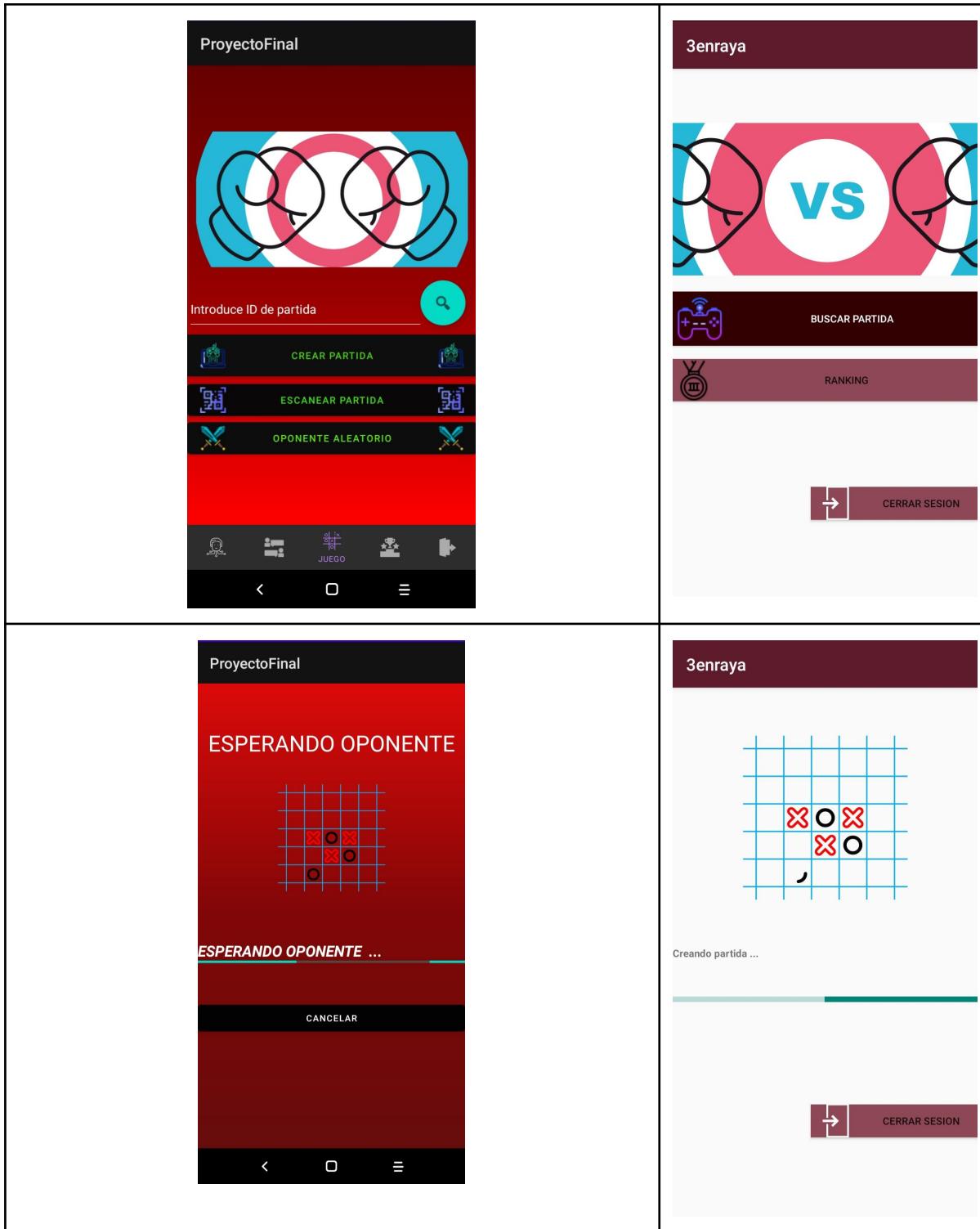


Comparación con el anterior proyecto

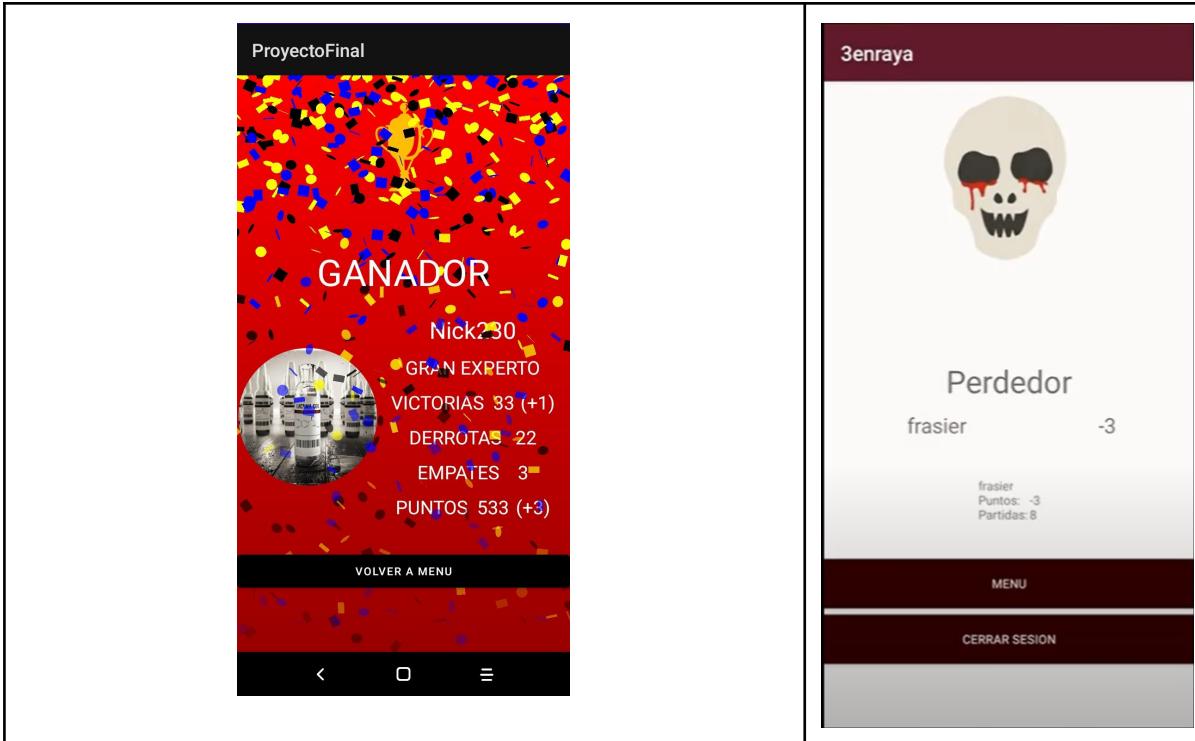
Nuevo	Anterior
 <p>ProyectoFinal</p> <p><b>TRES EN RAYA</b></p> <p>JUGAR SIN CONEXION</p> <p>JUGAR ONLINE</p>	 <p>ProyectoFinal</p> <p>HUMAN VS MACHINE</p> <p>BUENA SUERTE</p> <p>LA NECESITARAS</p> <p>FACIL</p> <p>DIFICIL</p> <p>IMPOSIBLE</p>
 <p>MODO DIFÍCIL</p> <p>HUMAN MACHINE</p> <p>?</p> <p>?</p> <p>?</p> <p>?</p> <p>?</p> <p>?</p> <p>?</p> <p>?</p> <p>ME RINDO</p>	 <p>3enraya</p> <p>HUMAN MAQUINA</p> <p>X O X X X X O X O</p> <p>X O X X X X O X O</p> <p>O X O X O X X X X X X X O X O O X O</p>





<p>ProyectoFinal</p> <p>spider GRAN EXPERTO</p> <p>Nick230 GRAN EXPERTO</p> <p>Puntuación: 790, 90, 0, 90</p> <p>Partidas: 491, 5, 1, 8</p>	<p><b>3enraya</b></p> <p>RANKING DE JUGADORES</p> <p>Puntos: 6666660 Partida: 0</p> <p>Puntos: 3000 Partida: 0</p> <p>jugador1</p>
<p>ProyectoFinal</p> <p>redson</p> <p>Nick230</p> <p>ABANDONAR</p>	<p><b>3enraya</b></p> <p>Jussi Adler-Olsen</p> <p>frasier</p> <p>XOXO</p> <p>Skull icons, Tic-Tac-Toe board, and a ME RINDO button.</p>



## Metodología

La metodología a seguir en el proyecto ha sido Scrum, al ser una metodología ágil que evita retraso y sobrecostes, adopta una estrategia de desarrollo incremental y mantiene la calidad del proyecto.

### Ciclo de vida

- **Fase 0:Introducción:** **Plantear el proyecto**, investigar las alternativas a kotlin y decidir el enfoque del proyecto.
- **Fase 1:Aprender Kotlin:** Después de elegir y tener seguro que voy a utilizar Kotlin a través de diferentes cursos, documentaciones y blogs tengo una idea general del lenguaje y cómo utilizarlo.
- **Fase 2: Base de datos:** Al ser no relacionar se desarrolló primero en E/R al tener más nivel sobre el mismo y se adaptó a no relacional.
- **Fase 3: Diseño UX/UI:** Como esta parte requiere poca lógica de programación y es bastante pesada se desarrollaron primero los diseños de todas las actividades y se creó la navegación para evitar problemas del anterior proyecto.
- **Fase 4: Diseño de la estructura del proyecto:** Con la navegación hecha se tenía claro que hacer con cada botón, imagen y demás por lo que se abstrajo todo el manejo de firebase y la mayoría de la programación repetitiva.
- **Fase 5: Desarrollo modo offline:** Al no depender de firebase se desarrolló primero, se desarrolló la persistencia y se implementan las funcionalidades respetando kotlin.
- **Fase 6 : Desarrollo de la administración de usuarios:** Antes de poder jugar online o chatear se necesitan usuarios en la base de datos por lo que se desarrolló el login, registro , editor de perfiles y los controles relaciones desde front-end al no tenerlos en firebase.
- **Fase 7: Desarrollo de Chat:** Al ser la parte más nueva se implementó primero permitiendo después poder ir más rápido con la partida online. Se probaron diferentes cosas y al final se dejó en su mínima expresión funcional al ser demasiado extenso.
- **Fase 8: Desarrollo de partida online:** Con todo ya listo se implementó la parte final del proyecto, se implementaron las bibliotecas necesarias, se crearon las activities para escanear y general el QR, el activity para esperar al oponente aleatorio, la partida como tal y la activity para mostrar el resultado de la partida.
- **Fase 9: Testing:** La aplicación se probó en diferentes móviles para asegurar su responsividad, asegurar la interconexión de usuarios y demás.No se utilizaron emuladores porque necesitaba usar la cámara y las notificaciones administrativas en emuladores dan muchos problemas.

Equipo	Precio
Ordenador con al menos 8gb de RAM	500 euros
Dispositivos móviles en los que realizar la aplicación	100*2 = 200 euros
Cuenta en google para usar firebase	Gratis
Android Studio	Gratis
Animaciones Lottie	Gratis
Soporte móvil vertical	11.99

Tiempo estimado 600 horas = 75 días \* 31.66(salario mínimo diario) = 2374.5 euros

Coste Total	3086,49 euros
-------------	---------------

## Resultados

En general los objetivos del proyecto se han conseguido, he conseguido una visión más global del ecosistema que rodea el desarrollo en android, he aprendido Kotlin y he hecho lo que quería para el proyecto.

Gracias al pragmatismo de Kotlin y la reestructuración he podido añadir muchas nuevas funciones añadiendo apenas 900 líneas de código extra.

La versión anterior constaba de 2399 líneas de código Java mientras que la actual consta 3206 líneas de código Kotlin, se esperaban más líneas.

## Conclusión

Las conclusiones son claras, todo aquello que esté desarrollado en Kotlin será más seguro y claro que algo desarrollado en java.

Las nuevas tecnologías de desarrollo presentan notables ventajas frente a sus predecesoras al cambiar el enfoque y centrarse en su propio discurso en lugar de repetir lo peor de cada tecnología pero con mejoras superficiales centrándose más que nada en la sintaxis.

Los desarrolladores en general ante esto responden de forma positiva en lugar de con odio como suele pasar en otros sectores cuando se presentan nuevas ideas y conceptos.

El reimaginar un proyecto previo aunque podría haber resultado cansado, repetitivo y aburrido, no lo ha sido, en absoluto puesto que Kotlin ha permitido no caer en repetir exactamente lo mismo, ha permitido crear un proyecto individual que aporta un nuevo enfoque.

Todos los objetivos iniciales han sido completados con éxito e incluso han surgido nuevos objetivos que han logrado redibujar la idea original.

La versión actual del proyecto más cercana a una red social orientada en la privacidad y seguridad de los usuarios en lugar de ser un juego online que no era algo muy complejo demuestra lo que se quería demostrar, que una tecnología de desarrollo cambiará nuestra forma de desarrollar tanto para bien como para mal.

Kotlin como se expresa en el marco teórico tiene notables ventajas frente al primer problema de java la falta de interés de la propia comunidad en la evolución del lenguaje y como por parte de la mayoría de programadores es una herramienta que usan pero sin el mayor interés en la misma.

Como se muestra en el marco teórico y en el proyecto original yo sí tengo interés en la evolución de java y se utilizó hasta donde se pudo, use java 8 todo lo que pude pero java está en su versión 15 y sigue aportando pero en android no tiene ningún tipo de aplicación. Por otra parte, en la investigación se muestra como Kotlin al no enfocarse en competir con otras tecnologías o quitar cuotas de mercado se convierte en una ayuda para la comunidad de desarrolladores en lugar de una tecnología que aprender para engordar currículos o completar stacks de proyectos.

Más allá de Kotlin en el proyecto se ha creado una estructura más madura que permitirá el poder completarlo de querer hacerlo y un diseño más personal y completo que es más agradable en UX/UI.

## Líneas futuras

El desarrollo es infinito, por ello se han seleccionado las líneas futuras más interesantes:

- Este proyecto aporta una visión global del lenguaje de programación Kotlin además de las tecnologías que permiten desarrollar para android por lo que:
  - No tengo una visión realmente completa de Kotlin pero si una global por lo que seguiré aprendiendo Kotlin.
  - Crear un hábito de desarrollo en Kotlin más allá de Android.
  - Aprender algunas de las tecnologías descartadas como React Native o Flutter.
- Mejorar el chat, el chat al ser un proyecto por sí mismo se ha dejado en la expresión mínima funcional actual idealmente se desarrollara :
  - Notificaciones.
  - Estado del usuario.
  - Mejorar la presentación de mensajes.
  - Crear grupos.
  - Enviar multimedia.
- En la parte Offline sería interesante:
  - Utilizar más propiedades de la plataforma android como bluetooth.
  - Dar la opción de guardar las puntuaciones en firebase.
  - Poder resetear las puntuaciones
- Mejorar la partida online.
- Sustituir firebase por un backend propio puesto que de crecer firebase no sería la mejor opción.
- Monetizar la app, añadir anuncios.
- Sacar versión para iOS con KMM.

# Referencias

## Referente a Kotlin

- <https://kotlinlang.org/>
- <https://kotlinlang.org/lp/mobile/>
- <https://kotlinlang.org/lp/server-side/>
- <https://kotlinlang.org/docs/js-overview.html>
- <https://kotlinlang.org/docs/android-overview.html>
- <https://www.youtube.com/channel/UCV31octs5hft6bZmokUgQIA>
- <https://openwebinars.net/cursos/kotlin-para-android/>
- <https://codigofacilito.com/cursos/android-kotlin>
- <https://developer.android.com/training/data-storage/shared-preferences>
- <https://developer.android.com/reference/com/google/android/material/bottomnavigation/BottomNavigationView>
- <https://github.com/airbnb/lottie-web>
- <https://github.com/Sarim-Ahmed93/QR-Code-Generator-Android-Kotlin>
- <https://github.com/journeyapps/zxing-android-embedded>
- <https://github.com/OpenSooq/Gligar>

## Referente a Python

- <https://kivy.org/#home>
- <https://www.python.org/doc/>
- <https://www.genbeta.com/actualidad/creador-phyton-reconoce-que-se-ha-quedado-fuera-moviles-navegadores-sea-usado-backend-estos-servicios>

## Referente a Flutter

- <https://flutter.dev/docs>

## Referente a React native

- <https://reactnative.dev/>
- <https://www.paradigmadigital.com/dev/desarrollando-aplicaciones-moviles-nativas-con-react-native/>

## Referente a Ionic

- <https://ionicframework.com/>

## Referente a firebase

- <https://firebase.google.com/docs/android/setup?hl=es>
- <https://firebase.google.com/docs/cloud-messaging/android/client?hl=es>
- <https://firebase.google.com/docs/storage/android/start?hl=es>
- <https://firebase.google.com/docs/storage/android/upload-files?hl=es>
- <https://firebase.google.com/docs/auth/android/start?hl=es>
- <https://firebase.google.com/docs/firestore?hl=es>
- <https://firebase.google.com/docs/auth/android/firebaseui?hl=es>

## Referente a Ofertas de trabajo

- <https://www.infojobs.net/>

## Estadísticas

- <https://mapadeempleo.fundaciontelefonica.com/>
- <https://insights.stackoverflow.com/survey/2020#technology-most-loved-dreaded-and-wanted-languages-loved>
- <https://insights.stackoverflow.com/survey/2020#technology-what-languages-are-associated-with-the-highest-salaries-worldwide-global>

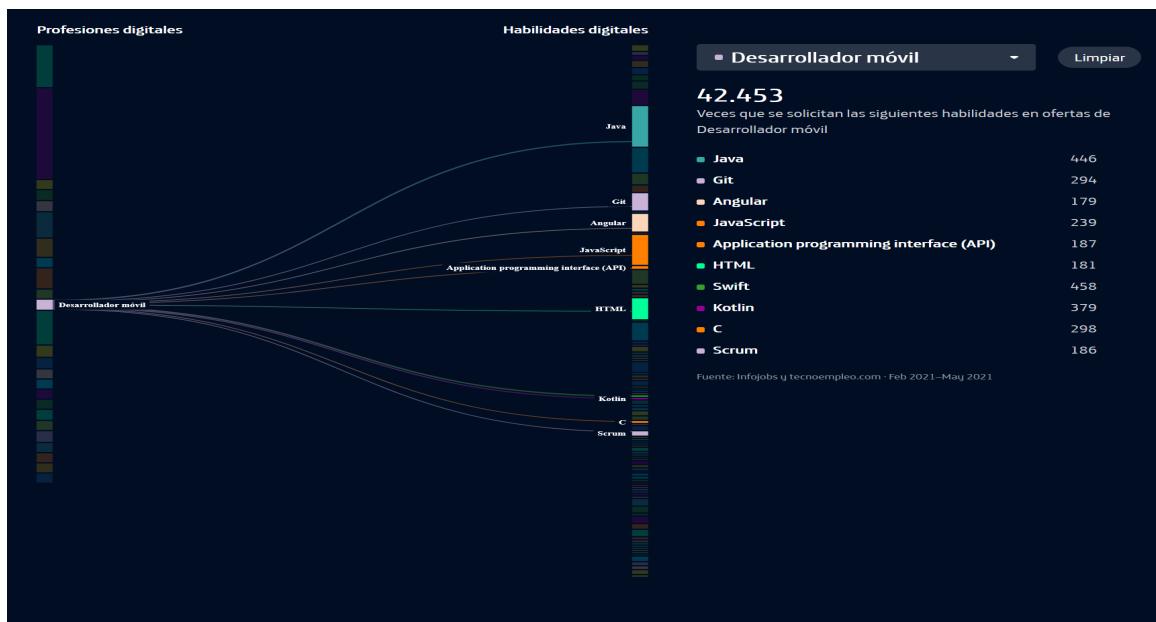
## Anexos

### Anexo 1 Comparación con otras tecnologías

En este apartado se comparan otras tecnologías que se tuvieron en cuenta y las razones por las que no se utilizaron. Se compara directamente kotlin al ser la primera opción del proyecto para sustituir a java.

La mayoría de estas comparativas se han desarrollado de forma teórica a través de artículos, críticas y pequeñas pruebas prácticas pero en menor medida por falta de tiempo.

Así se relacionan las tecnologías con el desarrollo móvil según el mapa de empleabilidad de telefónica.



Python fue desarrollado en 1991 por Guido van Rossum, es interpretado, multiparadigma, fuertemente tipado y con tipado dinámico está presente principalmente en back-end y ciencia de datos.



Aunque python es una tecnología en la que tengo más nivel puesto que llevo más tiempo programando en él y estoy certificado en él por el CISCO se ha descartado por distintos motivos. Para desarrollar android se usaría Kivy una biblioteca de código abierto de Python que se utiliza para crear aplicaciones en Windows, Linux, MacOS, Android e iOS

### Ventajas:

- Es un lenguaje que llevo estudiando 3 años y en el que estoy certificado.
- La app sería 100% multiplataforma sin repetir código.
- La sintaxis permite un desarrollo más rápido y dinámico que Kotlin.
- Gran comunidad.
- El lenguaje más querido por los programadores en general.

### Desventajas:

- No cumple el primer objetivo “Aprender algo nuevo”
- Según su propio creador admite que nunca estuvo destinado a desarrollo de interfaces grafica
- Kivy es una biblioteca muy inmadura y mal documentada.
- CPython compensa la falta de velocidad de Python creando un consumo excesivo de memoria y batería.
- Es monoproceso
- El tipado dinámico suele dar problemas, la mayoría de la comunidad utiliza un módulo para usar python con tipado estático por lo que el tipado estático en proyectos grandes demuestra ser a evitar.
- Python en Android a nivel laboral no tiene presencia.

**En conclusión, python** en plataformas móviles deja claro ser una tecnología completamente a evitar que tan solo tendría utilidad en móviles de alta y ni siquiera así la app podría resultar comercial al agotar rápidamente los recursos del dispositivo. Y Kotlin aunque menos aceptado fuera de android puede ofrecer hacer proyectos más amplios sin cambiar de tecnología base.



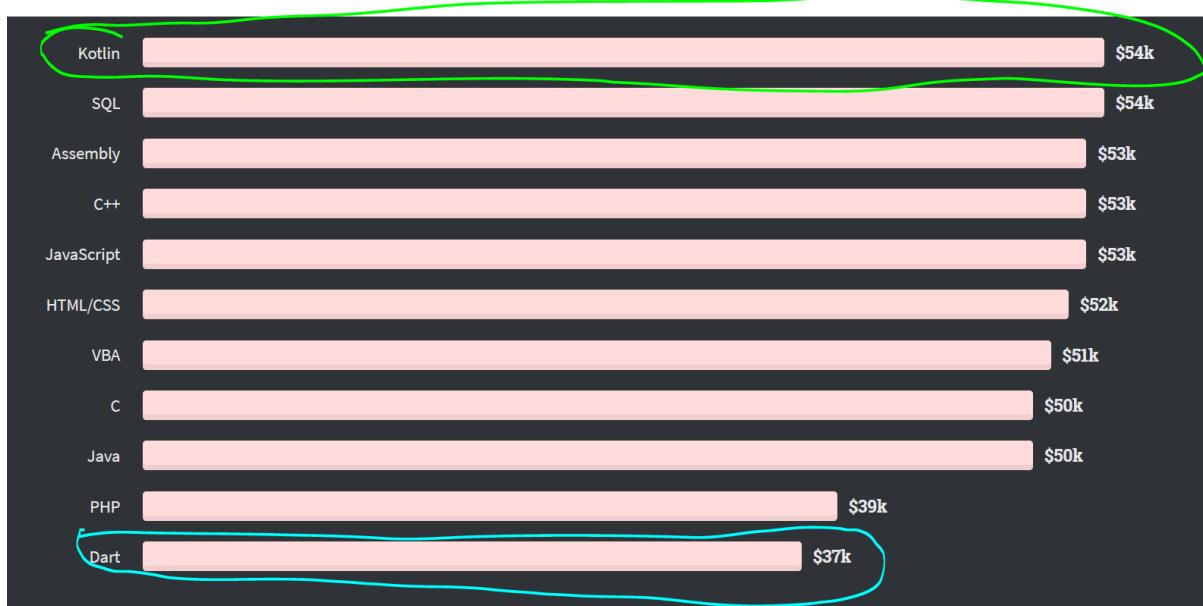
Dart fue lanzado por Google en 2017 bajo el estándar ECMA. Flutter es un conjunto de herramientas de interfaz de usuario multiplataforma que nos permite crear aplicaciones rápidas, enfocado en el diseño que compila de forma nativa en Android, IOs, Web y Escritorio utilizando el lenguaje de programación dart.

#### Ventajas:

- Hot Reload, permite ver los cambios según los realizamos sin recomilar en kotlin, en las pruebas gastamos mucho tiempo recopilando cada vez que cambiamos algo.
- Posibilidad de desarrollar en todas las plataformas al igual que Kotlin.
- Se centra mayormente en UX/UI lo que permitirá mejorar significativamente la UX/UI del proyecto anterior el cual es uno de los objetivos principales.
- Gran comunidad.

#### Desventajas:

- Dart como lenguaje que no da nada nuevo.
- Una tecnología muy inmadura.
- Mayor tamaño de aplicación.
- Kotlin no requiere bibliotecas y herramientas de terceros para tender un puente sobre una conexión con el entorno nativo mientras que Flutter tiene una dependencia absoluta.
- Esta asociado a ser el lenguaje con el que menos dinero se gana 37 mil al año frente a 54 mil al año de Kotlin.



**En conclusión** esta tecnología fue un duro rival frente a kotlin pero la sintaxis de Dart frente a Javascript prácticamente no aporta nada por lo que realmente inconscientemente no aprendería Dart traduciría de Javascript al no tener funcionalidades ni filosofías categóricas a diferencia de Kotlin.

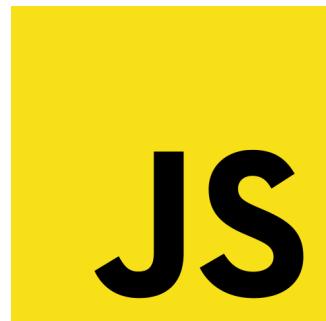
JavaScript fue desarrollado originalmente por Brendan Eich de Netscape con el nombre de Mocha, el cual fue renombrado posteriormente a LiveScript, para finalmente quedar como JavaScript en 1995. Es un lenguaje interpretado, débilmente tipado, multiparadigma y posiblemente el lenguaje más utilizado dependiendo de cómo contemos.

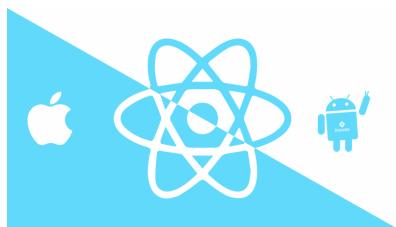
Nació originalmente para Front-end web y posteriormente dio el salto a Back-end, escritorio y plataformas móviles. Es principalmente característicos por tener muchísimos frameworks y bibliotecas como AngularJS, VueJS, ReactJS, GatsbyJS ...

También es interoperable con Typescript desarrollado por Microsoft.

Al ser un lenguaje tan exageradamente popular lo normal era que saltara a plataformas y así lo hizo, tiene dos vertientes, el desarrollo nativo y el desarrollo híbrido, a continuación veremos los dos mejores ejemplos de cada uno.

En cualquiera de los dos casos el resultado será 100% compatible con IOs y Android sin necesidad de realizar ningún cambio siendo esta la principal ventaja frente a Kotlin KMM. El desarrollo nativo nos permite que la aplicación se ejecute directamente en el dispositivo mientras que la App desarrollada desarrollo híbrido se ejecutará a través de un WebView lo que ralentiza de forma significativa la aplicación.





Facebook lanzó React Native basado en React, su propia biblioteca de Javascript en 2015 y lo ha mantenido desde entonces. En 2018, React Native tuvo el segundo número más alto de contribuyentes para cualquier repositorio en GitHub. En la actualidad, React Native cuenta con el respaldo de contribuciones de personas y empresas de todo el mundo.

- **Ventajas**

- Económicamente es más rentable para las empresas al solo desarrollar una vez
- La app funcionará tanto en Android como en IOS sin ningún cambio.
- La mayoría del código de la App móvil es reutilizable en Web.
- Tiene una comunidad muy fuerte.
- El proceso de desarrollo es rápido.

- **Desventajas**

- Sigue siendo una tecnología inmadura actualmente se encuentra en la versión 0.64
- Las bibliotecas de terceros pueden presentar diferencias de rendimiento en las diferentes plataformas por lo que crea más trabajo en la empresa, esto KMM lo soluciona al obligarnos a escribir la parte específica de cada plataforma.
- El equipo de desarrollo está obligado a conocer Android, IOS y React para solucionar las posibles incidencias
- Al igual que en Python el tipado dinámico puede crear problemas en tiempo de ejecución.
- Actualmente tiene menos trabajo que kotlin. Kotlin tiene 68 y React Native 37.
- Suele dar lugar a sueldos más bajos.

**En conclusión** react native no se a utilizado para el proyecto debido a ser una tecnología inmadura y tampoco sacaría el 100% de provecho al no tener dispositivos Apple para probarlo. Pero es una muy buena opción puesto que aprender React es algo a tener en cuenta al tener una gran presencia laboral y ser una librería que también se está utilizando en backend.



Ionic es un SDK de código abierto completo para el desarrollo de aplicaciones móviles híbridas creado por Max Lynch, Ben Sperry y Adam Bradley de Drifty Co. en 2013. La versión original se lanzó en 2013 y se construyó sobre AngularJS y Apache Cordova

- **Ventajas**

- Económicamente más rentable.
- Más opciones de desarrollo al poder desarrollar en Angular, React o Vue.
- Una única tecnología para ambas plataformas.
- Una comunidad muy amplia.
- Gran variedad de plugins para desarrollar.
- Una tecnología más madura que React Native.
- No tiene diferencias notables en el ámbito laboral respecto a Kotlin.

- **Desventajas**

- Una curva de aprendizaje mayor de no conocer Angular o React o Vue.
- En general tendrá un rendimiento más lento que el desarrollo nativo.
- Dependencia de los plugins, en caso de no existir lo tendremos que crear de cero.
- No aprovecha tan bien las funciones específicas de cada plataforma.
- Desarrollo mucho más pesado, tendremos que escribir de forma separada HTML + CSS + Javascript y de la forma concreta del framework que hayamos escogido.
- Suele dar lugar a sueldos más bajos.

**En conclusión** Ionic al tener una curva de aprendizaje mayor al no conocer por mi parte ni Angular ni Vue ni React y el desarrollo habría sido mucho más largo y pesado por lo que el proyecto podría no haber llegado a lo que ahora.