Sistema de pantalla de desbloqueo para dispositivos Android

Basado en patrones y ritmos

Fernando Fernández Hidalgo

Máster en Ciberseguridad y Privacidad
Área de Privacidad

Tutor de TF
Joan Caparrós Ramírez
Profesor/a responsable de la asignatura
Andreu Pere Isern Deyà

01/2023

Universitat Oberta de Catalunya

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Sistema de pantalla de desbloqueo para dispositivos Android basado en patrones y ritmos*
Nombre del autor:	*Fernando Fernández Hidalgo*
Nombre del consultor/a:	*Joan Caparrós Ramírez*
Nombre del PRA:	*Andreu Pere Isern Deyà*
Fecha de entrega (mm/aaaa):	*01/2023*
Titulación o programa:	Máster en Ciberseguridad y Privacidad
Área del Trabajo Final:	*Area de Privacidad*
Idioma del trabajo:	*Castellano*
Palabras clave	*Desbloqueo, Android, Ritmos*

Resumen del Trabajo

El objetivo fundamental de este trabajo fin de máster es el de proponer un modelo para añadir una capa de seguridad adicional sobre el patrón de desbloqueo en los dispositivos móviles de la plataforma Android, de cara a incrementar la privacidad de los usuarios y reducir potenciales ataques de desbloqueo del terminal conociendo el patrón.

Para ello se plantea la implementación de una aplicación Android que permita fijar y validar patrones de bloqueo en una matriz de 3x3 añadiéndoles la dimensión del ritmo, que se entiende como el tiempo que pasa entre la selección de una celda y la anterior.

Tras la fase de desarrollo, se implementa una aplicación de ejemplo y una librería que incluye la lógica de negocio de la operativa rítmica. Se define inicialmente un umbral de tolerancia entre la selección de las celdas del patrón de un 20% que posteriormente se modifica al 40% tras una ronda de pruebas con usuarios reales. Este umbral permite obtener una tasa de éxito en las validaciones por parte de los usuarios del 93% y una tasa de éxito del 100% frente a ataques en los que un atacante es conocedor de la secuencia de celdas del patrón de bloqueo del usuario pero no del patrón rítmico.

Índice

1. Introducción 4
   1.1. Contexto y justificación del Trabajo 4
   1.2. Objetivos del Trabajo 6
   1.3. Metodología 6
   1.4. Tareas 7
   1.5. Planificación del Trabajo 9
   1.6. Breve sumario de estado del arte 11
   1.7. Riesgos y soluciones 12
   1.8. Costes asociados al material 12
2. Materiales y métodos 14
   2.1 Análisis de requisitos 14
   2.2 Casos de uso 20
   2.3 Estructuras de datos 22
   2.4 Arquitectura 23
   2.5 Algoritmo de validación rítmica 24
   2.6 Interfaz gráfica 25
3. Desarrollo de la aplicación 30
   3.1 Gestión y organización 30
   3.2 Estructura de carpetas 30
   3.3 Modelos 32
   3.4 Vistas 33
   3.5 Activities 35
   3.6 Fragments 35
   3.7 Adapters 36
   3.8 Bases de datos 36
   3.9 Diagrama de clases 36
   3.10 Librerías externas 38
   3.11 RhythmPattern en acción 38
   3.12 Compilación del proyecto 39
   3.13 Fase de pruebas 40
4. Resultados 46
5. Conclusiones y trabajos futuros 49
   5.1 Conclusiones 49
   5.2 Trabajos futuros 50
6. Glosario 53
7. Bibliografía 54
8. Anexos 55

Lista de figuras

Figura 1: Ejemplo de patrón de desbloqueo 5
Figura 2: Ejemplo de ataque por marcas 5
Figura 3: Diagrama de Gantt con planificación del trabajo 10
Figura 4: Estructura de una celda con ritmo 22
Figura 5: Estructura de un patrón 22
Figura 6: Pantallas de la aplicación 23
Figura 7: Diagrama de sistema 24
Figura 8: Validación del ritmo entre dos celdas 25
Figura 9: Aplicación – Pantalla principal 26
Figura 10: Aplicación – Matriz de patrón 26
Figura 11: Aplicación – Primer intento en asistente 27
Figura 12: Aplicación – Intento fallido en asistente 27
Figura 13: Aplicación – Introducción de patrón 28
Figura 14: Aplicación – Asistente completado 28
Figura 15: Aplicación – Desbloqueo exitoso 29
Figura 16: Aplicación – Desbloqueo fallido 29
Figura 17: Estructura del proyecto 31
Figura 18: Ficheros de la librería 32
Figura 19: Vista 'PatternLockView' 33
Figura 20: Layout de pantalla principal 34
Figura 21: Layout de pantalla de patrón 35
Figura 22: Almacenamiento a base de datos 36
Figura 23: Diagrama de clases de la librería 37
Figura 24: RhythmPattern en acción 38
Figura 25: Módulos en Android Studio 39
Figura 26: Construcción del proyecto 40
Figura 27: Ejecución sobre emulador 40
Figura 28: Ejemplo resultado plantilla pruebas 42
Figura 29: Perfiles usuario de prueba 43
Figura 30: Resultados ronda 1 fase de pruebas 43
Figura 31: Resultados ronda 2 fase de pruebas 44

1. Introducción

1.1. Contexto y justificación del Trabajo

Los dispositivos móviles se han convertido en herramientas indispensables en el día a dia de las personas. Actualmente son utilizados para trabajar desde ellos, mantener conversaciones privadas, tomar fotografías, albergar documentos y realizar muchas más actividades. El tipo de información almacenada en estos dispositivos se vuelve por lo tanto un recurso valioso a proteger por tratarse en gran medida de información sensible de los usuarios.

Los fabricantes de dispositivos móviles y desarrolladores de sistemas operativos son conscientes de la criticidad de preservar adecuadamente la privacidad de los usuarios e implementan continuamente mejoras en sus sistemas de acceso a las partes sensibles de los dispositivos. Un diseño habitual es el de disponer de una pantalla de bloqueo que presenta un desafío, mediante el cual se otorga o deniega el acceso a la información existente en el dispositivo.

Uno de estos desafíos es el característico bloqueo de pantalla basado en un número PIN o una contraseña, destinado a impedir que usuarios no autorizados accedan de manera física al contenido del dispositivo móvil. Sobre este sistema existen diferentes variantes basadas por ejemplo en la longitud del código o el uso de caracteres especiales.

Otra tecnología habitual hoy en día es la del uso de información biométrica del usuario, como por ejemplo la huella dactilar o la representación de la cara. Este tipo de identificación supone algunas limitaciones en diferente escenarios, como durante el uso de guantes o bufandas.

Aunque cada plataforma de dispositivos móviles utiliza diferentes sistemas de bloqueo de pantalla, uno de los más habituales el basado en un **patrón de desbloqueo**. Este sistema consiste en repetir una secuencia de pasos previmente definidos en una matriz de posiciones. En la siguiente imagen se puede observar el diseño habitual que suele tener:

Figura 1: Ejemplo de patrón de desbloqueo – Elaboración propia

Este sistema presenta un inconveniente similar al basado en PIN/contraseña, y es que si un atacante consigue observar el patrón que introduce el usuario legítimo, el atacante podría replicarlo en un acceso físico y acceder al contenido sensible del dispositivo.

En algunas ocasiones, hay formas muy sencillas de conocer el patrón introducido por el usuario, como puede ser simplemente observando las marcas físicas que se producen en la pantalla tras la introducción exitosa del código por parte del usuario legítimo y que son producidas por factores como el polvo, sudor, grasa corporal, etc. En la siguiente imagen se puede ver un ejemplo de marcas en el terminal que fue analizado por Adam J. Aviv y colaboradores en su paper "*Smudge Attacks on Smartphone Touch Screens*"[1]:

Figura 2: Ejemplo de ataque por marcas – Imagen obtenida del paper "*Smudge Attacks on Smartphone Touch Screens*" de Adam J. Aviv y colaboradores[1]

El objetivo principal de este TFM es el de añadir una capa de seguridad adicional al patrón de desbloqueo de los terminales. Esta capa adicional está basada en añadir una **dimensión rítmica**, mediante la cual, la introducción del patrón de desbloqueo solamente se considere exitosa si se ha introducido con el mismo ritmo (o similar con cierto grado de tolerancia) al definido previamente. De esta forma, aunque un atacante pudiera conocer la forma del patrón de desbloqueo de un usuario, este no podría ser capaz de repetirlo salvo en caso de disponer del ritmo de introducción.

El resultado que se pretende obtener con este TFM, es el de disponer de una versión de una pantalla de desbloqueo para dispositivos móviles con tecnología Android que implemente el objetivo anteriormente descrito.

1.2. Objetivos del Trabajo

Este TFM se plantea con el objetivo principal de implementar **un sistema de pantalla de desbloqueo para dispositivos móviles Android basado en patrón de desbloqueo con la dimensión adicional del ritmo**.

Como objetivos principales de este trabajo se encuentran:

- Desarrollar una librería propia para tecnología Android de validación de patrones rítmicos.

- Implementar una aplicación móvil en Android con las siguientes funcionalidades:
    - Desarrollar un tutorial guiado para fijar la definición del patrón de desbloqueo rítmico.
    - Construir un módulo de validación de patrones rítmicos.
    - Diseñar e implementar una pantalla de desbloqueo con resultado exitoso o fallido en base a la validación de patrón y rítmica.

Como objetivos secundarios de este trabajo se pretende:

- Que la implementación pueda ser ejecutada en un emulador Android de AOSP (Android Open Source Project) con versión 12.

- Que el código sea código libre y no haga uso de licencias comerciales.

Como caso de uso principal a resolver, un usuario que únicamente conozca la secuencia de movimientos del patrón de desbloqueo (es decir, sin conocer la parte rítmica) no ha de ser capaz de desbloquear el terminal.

1.3. Metodología

Para el desarrollo de este TFM se utilizará la metodología ágil *Kanban* con una serie de entregas definidas que coinciden con los hitos del calendario del trabajo. Por cada tarea se generará una tarjeta que se irá gestionando con la

herramienta *Trello* de Atlassian, a través de una cuenta gratuita. Los bugs y las tareas de bugfixing que aparezcan durante el desarrollo del proyecto también se gestionarán a través de la generación de sus tarjetas correspondientes.

Como IDE de desarrollo se utilizará *Android Studio* en su versión *Chipmunk.*

El control de versiones del código fuente se realizará con la herramienta *GIT*.

Respecto a la implementación gráfica de la matriz de posiciones de la pantalla de desbloqueo, existe un proyecto de software libre (versión Apache 2.0) llamado “pattern lock”[2], que implementa una pantalla de bloqueo sin control de ritmo. La idea para este TFM es utilizar algunos de los componentes de dicho proyecto para implementar la vista gráfica de la pantalla de desbloqueo.

El inicio del desarrollo para este TFM se realizará adaptando el código del proyecto “pattern lock” para ajustarlo a las necesidades del trabajo en lo que a interfaz gráfica se refiere. Posteriormente se implementará la base rítmica y finalmente se cerrará el caso de uso con una aplicación que ejemplifique todas las partes.

Para la implementación de la validación rítmica, se ha de desarrollar un modelo que permita la identificación de los patrones rítmicos con un nivel de tolerancia previamente definido. Esto se debe a que introducir **exactamente el mismo** ritmo en dos secuencias diferentes de pasos es altamente improbable, por lo que ha de existir un factor de desviación que haga que el desbloqueo sea válido siempre y cuando sea similar al patrón original. Para esta parte habrá de realizarse una fase de pruebas con diferentes usuarios, con el objetivo de definir un umbral de tolerancia lo más genérico posible.

En el siguiente punto de este documento se pueden ver las tareas a realizar de una forma más detallada.

1.4. Tareas

El desarrollo de este TFM consiste en la implementación de una aplicación Android que dé como resultado una pantalla de desbloqueo basada en un patrón de desbloqueo con una capa rítmica y que pueda ser mostrada en un emulador oficial de Android con la versión 12. Para este desarrollo se han definido las tareas de alto nivel que se describen a continuación:

1. Fase de investigación y diseño del trabajo.

Durante esta fase se estudiará la viabilidad del desarrollo así como el posible diseño de implementación del trabajo propuesto.

2. Fase de análisis del trabajo.

En esta fase se analizarán las opciones de implementación del TFM. Uno de los puntos fundamentales de esta parte será la de analizar una propuesta de

implementación del algoritmo de validación de la parte rítmica que sea eficiente para realizar una validación satisfactoria de los usuarios.

3. Preparación del entorno de desarrollo

La preparación del entorno de desarrollo se realizará preparando el proyecto en Android Studio del TFM. También se creará un repositorio para el código fuente y las tareas de desarrollo correspondientes en *Trello*. Se creará un emulador de un dispositivo Android 12 sobre el que deberá correr la aplicación desarrollada para este trabajo.

4. Estudio, adaptación y compilación del proyecto “pattern lock”

Se realizará la comprensión, adaptación e integración de este proyecto acorde a las necesidades de la idea propuesta para la validación del patrón basado en ritmos. Únicamente se considerará la utilización de la parte que implementa los aspectos gráficos del patrón de desbloqueo.

5. Implementación de la interfaz gráfica del TFM que constará de una aplicación móvil Android con:
    a. Un tutorial guiado para el usuario con el objetivo de definir el código de bloqueo basado en patrones rítmicos (ha de solicitarlo 3 veces, y generará un patrón basado en las 3 introducciones)
    b. Una pantalla de desbloqueo del terminal que muestre si el desbloqueo ha sido exitoso o fallido.

6. Una librería Android que permita la validación de patrones rítmicos.

Esta librería implementará el algoritmo de validación de patrones rítmicos que se utilizará para consultar el desbloqueo exitoso o fallido por parte del usuario.

7. Pruebas de usuario.

Se realizarán pruebas de usuario sobre el emulador con Android 12 con usuarios diferentes para validar que la implementación sea satisfactoria desde la perspectiva del caso de uso del usuario. Esta fase servirá para validar o corregir la implementación realizada hasta ese momento.

8. Resolución de errores.

Durante esta fase se hará bugfixing de los bugs que hayan salido durante la fase de pruebas.

9. Entrega de *release* final.

Se generará una compilación de release con los binarios resultantes de la implementación. También se generará un entregable con el código fuente de la implementación.

10. Generación de documentación.

La elaboración de la documentación será una tarea transversal a todo el desarrollo. El resultado final será la entrega de la memoria resultante.

11. Elaboración de video y material para la presentación.

Se realizará la edición de un video y una presentación para la defensa del TFM.

1.5. Planificación del Trabajo

La planificación del trabajo se ha elaborado teniendo presentes los hitos a entregar marcados por la propia estructura de la asignatura. Teniendo esto en cuenta, se han generado 4 tareas de alto nivel, nombradas con el nombre de las PECs y con las fechas de entrega asignadas según el plan docente.

Dentro de las 4 PECs se han planificado las tareas descritas en el punto anterior. Adicionalmente se ha añadido una tarea al término de las 4 PECs, que representa la elaboración del video de la presentación.

Para la planificación del trabajo se ha utilizado la herramienta gratuita “Online Gantt”[3]. A continuación se puede observar la planificación definida:

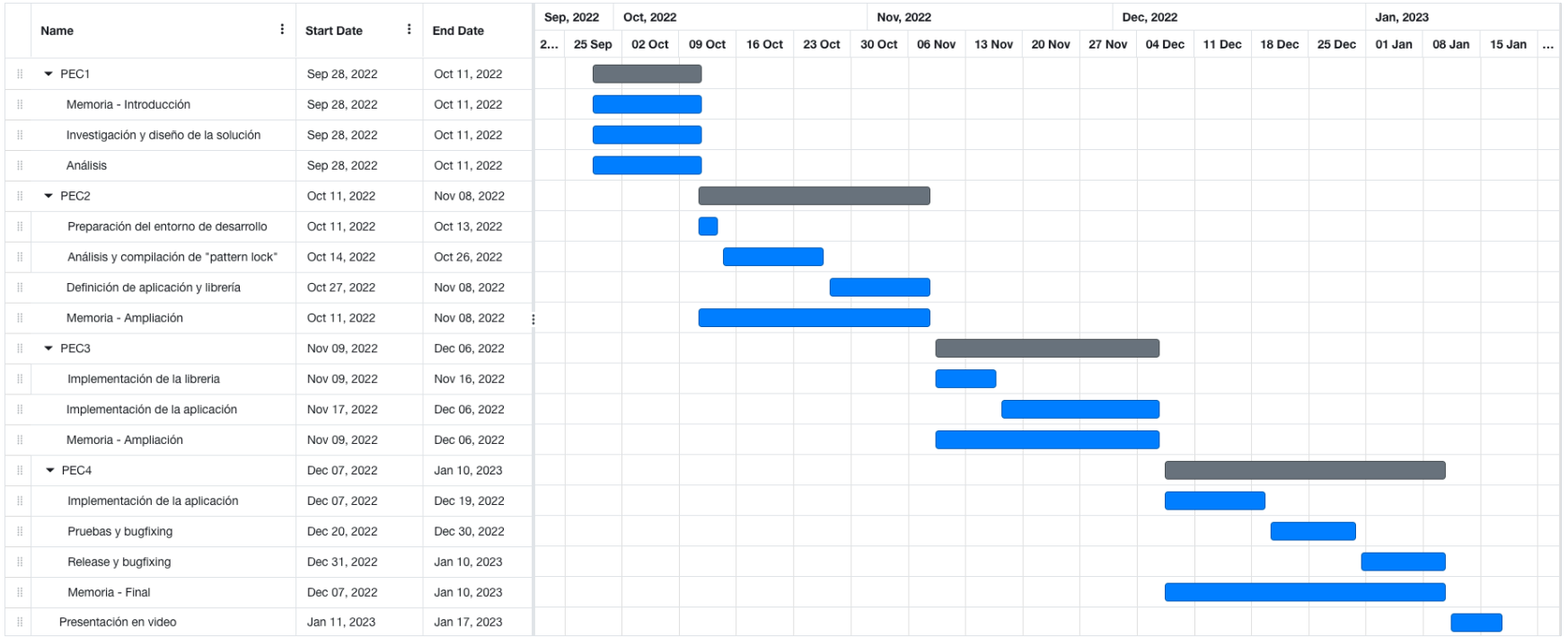

Figura 3: Diagrama de Gantt con planificación del trabajo – Elaboración propia

La elaboración de la documentación es una tarea transversal a la implementación del trabajo.

A continuación se puede ver una distribución en horas asociadas a cada tarea:

Tarea	Inicio	Fin	Horas
PEC1	Sep 28, 2022	Oct 11, 2022	50h
Memoria - Introducción	Sep 28, 2022	Oct 11, 2022	24h
Investigación y diseño de la solución	Sep 28, 2022	Oct 11, 2022	13h
Análisis	Sep 28, 2022	Oct 11, 2022	13h
PEC2	Oct 11, 2022	Nov 08, 2022	75h
Preparación del entorno de desarrollo	Oct 11, 2022	Oct 13, 2022	8h
Análisis y compilación de "pattern lock"	Oct 14, 2022	Oct 26, 2022	26h
Definición de aplicación y librería	Oct 27, 2022	Nov 08, 2022	26h
Memoria - Ampliación	Oct 11, 2022	Nov 08, 2022	15h
PEC3	Nov 09, 2022	Dec 06, 2022	75h
Implementación de la libreria	Nov 09, 2022	Nov 16, 2022	30h
Implementación de la aplicación	Nov 17, 2022	Dec 06, 2022	30h
Memoria - Ampliación	Nov 09, 2022	Dec 06, 2022	15h
PEC4	Dec 07, 2022	Jan 10, 2023	75h
Implementación de la aplicación	Dec 07, 2022	Dec 19, 2022	20h
Pruebas y bugfixing	Dec 20, 2022	Dec 30, 2022	30h
Release y bugfixing	Dec 31, 2022	Jan 10, 2023	10h
Memoria - Final	Dec 07, 2022	Jan 10, 2023	15h
Presentación en video	Jan 11, 2023	Jan 17, 2023	10h
			Total: 285h

El hecho de disponer de experiencia en la plataforma Android ha resultado en que las tareas puras de desarrollo estén cuantificadas con un nivel de incertidumbre bajo. La parte de pruebas y bugfixing dispone de una cuantía de 30h para intentar hacer un buen ajuste de cara a la eficiencia en la validación del patrón.

No se estima ningún periodo vacacional durante el transcurso del desarrollo de este TFM.

La implementación del proyecto se realizará utilizando un equipo portátil convencional y el software oficial de Android (el cual incluye emuladores de terminales móviles, que serán utilizados en este trabajo).

1.6. Breve sumario de estado del arte

Se pueden encontrar algunas aplicaciones que realizan el desbloqueo de pantalla basado únicamente en la ejecución de ritmos como la aplicación de Android "*TapTap Lock*"[4]. De todos modos, esta aplicación es muy limitada y

únicamente ofrece opciones del estilo de desbloquear la pantalla al tocar dos veces la pantalla (en cualquier punto) con un ritmo determinado.

Algunos terminales móviles como el LG G3 tuvieron un sistema de desbloqueo llamado "Knock Code"[5], consistente en tocar partes de la pantalla (generalmente subdividida en una matriz de 2x2) en un orden predeterminado. De esta forma, el usuario era capaz de desbloquear el dispositivo introduciendo un patrón previamente definido.

No se ha encontrado ninguna aplicación ni librería que implemente el desbloqueo de la pantalla basada en la introducción del patrón de desbloqueo siguiendo un patrón rítmico.

1.7. Riesgos y soluciones

A continuación se detallan los principales riesgos asociados a la elaboración de este trabajo y sus soluciones:

Riesgo 1	**Problemas en la compilación del proyecto "pattern lock"**
Solución	Solucionar los problemas de compilación, buscar un proyecto alternativo o implementar la parte gráfica íntegramente.

Riesgo 2	**Problemas en la integración de "pattern lock"**
Solución	Intentar solucionar los problemas de integración o buscar opción alternativa.

Riesgo 3	**Problemas de compatibilidad del desarrollo con el emulador de Android**
Solución	Utilizar código estándar y corregir las incompatibilidades que pudieran surgir.

Riesgo 4	**Dificultad para desbloquear el terminal basado en ritmos**
Solución	Mejorar el algoritmo de definición y comprobación de los ritmos.

Riesgo 5	**Pérdida de código fuente**
Solución	El código cuenta con un control de versiones y una copia periódica en un disco duro externo. Recuperar el código de dicho disco duro.

Riesgo 6	**Problemas con el material hardware de desarrollo**
Solución	Utilizar hardware equivalente.

1.8. Costes asociados al material

La implementación de este TFM tiene unos costes asociados al material muy bajos ya que todo el software que se utiliza dispone de licencias de software libre. Para el desarrollo de la solución se utilizará el siguiente material con su correspondiente valoración de costes:

Material	Coste
MacBook Pro 2021	2.249€
Licencia Android Studio	Gratuita
Licencia GIT	Gratuita
Licencia Trello	Gratuita
Licencia Microsoft Office	Gratuita
Licencia Online Gantt	Gratuita
Licencia Emulador Android AVD	Gratuita

El total asciende únicamente al coste del equipo portátil que es de **2.249€**.

2. Materiales y métodos

2.1 Análisis de requisitos

A continuación se detallan los requisitos funcionales para este proyecto. Cada requisito dispone de un código identificativo, un nombre y descripción del mismo así como posibles observaciones que se tengan que tener en cuenta en referencia al requisito.

Código	Nombre
2.1.1	Aplicación Android
Descripción	
Se generará una aplicación Android que permita introducir a un usuario un patrón de desbloqueo personal, basado en el orden de selección de celdas en una matriz de [3x3] así como del ritmo de selección de dichas celdas. Entenderemos como ritmo el tiempo que ha pasado entre la selección de una celda y la siguiente. La aplicación deberá ofrecer 2 acciones claramente diferenciadas: 1. Ofrecer un asistente para definir el patrón de desbloqueo. 2. Validar la correcta o fallida introducción de un patrón, una vez el patrón de usuario ha sido definido.	
Observaciones	
En los siguientes requerimientos funcionales se detalla cada una de estas funciones.	

Código	Nombre
2.1.2	Definición de patrón personal de bloqueo
Descripción	
El patrón que el usuario quiera definir se introducirá 3 veces, siguiendo un asistente con el objeto de:	

- Asegurar que el usuario introduce el patrón de celdas correctamente y no se equivoca
- Obtener un patrón de ritmo medio resultante de los 3 intentos

El usuario ha de obtener retroalimentación del número de intento en el que se encuentra dentro del asistente.

Las celdas del primer intento de introducción se considerarán válidas para la formación del patrón de bloqueo del usuario. El usuario ha de ser informado si se equivoca al introducir las celdas en cualquier intento posterior al primero durante el asistente.

Observaciones
Como es altamente improbable que el ritmo de introducción del patrón sea el mismo durante las 3 introducciones, se realiza 3 veces con el objeto de obtener una media rítmica resultante de los 3 intentos.

Código	Nombre
2.1.3	Generación de patrón personal de bloqueo

Descripción

Una vez se han realizado las 3 introducciones en el asistente, la aplicación ha de ser capaz de generar el modelo de un patrón válido para el usuario. Dicho patrón constará de 2 partes:

- Las celdas introducidas por el usuario. Esta parte hace referencia al orden de introducción de las celdas de la matriz de [3x3]. Han de ser las mismas en los 3 intentos.
- El ritmo con el que se han introducido dichas celdas.

El patrón introducido por el usuario ha de ser persistente al cierre de la aplicación así como al reinicio del dispositivo.

El usuario deberá ser capaz de crear un nuevo patrón de bloqueo.

Observaciones

La generación del patrón rítmico se detalla en el requisito funcional 1.4.

Código	Nombre
2.1.4	Modelado del ritmo del patrón personal de bloqueo

Descripción

Una vez se disponga del ritmo obtenido de las 3 introducciones, se realizará una media aritmética con los tiempos obtenidos entre la selección de las celdas. Por ejemplo, si el patrón de bloqueo consta de la selección de 2 celdas, y los tiempos entre dichas celdas, en los 3 intentos, son de 200ms, 400ms y 600ms respectivamente, el tiempo que se definirá para este patrón de 2 celdas es de 400ms.

Si por ejemplo el patrón introducido por el usuario estuviera formado por 5 celdas, se calcularían las medias entre las celdas 1 y 2, 2 y 3, 3 y 4 y 5. Una vez obtenida la media, se definirá un umbral que será la variación de tiempo tolerada en considerar éxito o fracaso la validación del ritmo entre 2 notas. Esto se debe a que como el ritmo entre 2 celdas se mide en milisegundos, es altamente improbable poder repetir la introducción de 2 celdas con los mismos milisegundos. Este umbral se definirá en función de las pruebas realizadas durante el desarrollo.

Código	Nombre
2.1.4	Validación de las celdas patrón personal de desbloqueo

Descripción

Se considerará que la validación de las celdas introducidas a la hora de desbloquear el terminal es exitosa, siempre y cuando el orden de las celdas introducidas sea exactamente el mismo que el orden de las celdas introducidas en la definición del patrón de bloqueo.

Código	Nombre

2.1.5	Validación del ritmo del patrón personal de desbloqueo
Descripción	
Se considerará que la validación del ritmo del patrón introducido por el usuario a la hora de desbloquear el terminal es exitosa, siempre y cuando el ritmo introducido esté dentro del umbral tolerable respecto al ritmo introducido en la fase de definición del patrón de bloqueo.	

Código	Nombre
2.1.6	Validación del patrón personal de bloqueo
Descripción	
La aplicación deberá informar del éxito o fracaso en la validación, sin decir si se debe al éxito/fracaso particular de las celdas o del ritmo.	
Observaciones	
La aplicación no informará si el fracaso es debido a que no se valide el ritmo o las celdas. El mensaje será genérico.	

Código	Nombre
2.1.7	Visualización de patrón introducido
Descripción	
La aplicación ha de permitir retroalimentar al usuario en la introducción del patrón. Para ello se mostrarán las celdas que se van introduciendo, marcándolas en color azul cuando han sido seleccionadas, y uniendo con una línea azul el siguiente movimiento que se pretende hacer, de manera similar a como lo realiza Android.	

Código	Nombre
2.1.8	Visualización del resultado del patrón introducido
Descripción	

La aplicación ha de mostrar el resultado exitoso o fallido de introducción del patrón de desbloqueo.

Si el desbloqueo es exitoso, se mostrará un aviso notificándolo.

Si el desbloqueo es fallido, se marcarán las celdas en rojo y se mostrará un aviso notificándolo.

A continuación se detallan los requisitos no funcionales considerados para este trabajo.

Código	Nombre
2.1.9	Aplicación Android

Descripción

Se deberá proveer una aplicación Android que se pueda abrir con Android Studio. La aplicación mostrará la funcionalidad anteriormente descrita.

La aplicación importará una librería Android desarrollada también para este trabajo y donde se encontrará la mayor parte de la implementación de validación.

La aplicación servirá como ejemplo para validar el funcionamiento de la propuesta de patrón de desbloqueo basado en patrones y ritmos.

Observaciones

La aplicación se proveerá como un proyecto de Android Studio.

Código	Nombre
2.1.10	Librería Android

Descripción

Se desarrollará una librería Android que contendrá la lógica referente a la validación de celdas y patrones rítmicos.

Esta librería deberá poder cargarse en Android Studio y estar vinculada a la aplicación anteriormente descrita.

Observaciones

Se proveerá un proyecto de Android Studio con la librería y la aplicación vinculadas.

Código	Nombre
2.1.11	Compatibilidad con Android 12
Descripción	
Tanto la aplicación Android como la librería deberán ser compatibles con Android en su versión 12.	

Código	Nombre
2.1.12	Compatibilidad con emulador AVD de Android
Descripción	
La aplicación y la librería deberán poder ejecutarse en un emulador AVD oficial de Android con la versión 12.	
Observaciones	
El emulador se generará directamente desde Android Studio con la versión de Android 12 sobre un modelo Google Pixel.	

Código	Nombre
2.1.13	Seguridad
Descripción	
Como el objeto de esta aplicación es puramente demostrativo respecto a la validación de patrones rítmicos, no se considerará ninguna medida de seguridad especial para la implementación de este proyecto.	
Observaciones	

Se utilizará el espacio de almacenamiento de la propia aplicación, únicamente accesible por la misma aplicación en condiciones normales para almacenar el patrón de bloqueo definido por el usuario.

Código	Nombre
2.1.14	Eficiencia
Descripción	
La introducción de los patrones ha de ser ágil, por lo tanto la retroalimentación al usuario referente a cualquier interacción con la interfaz gráfica deberá tener unos tiempos inferiores a 20ms para ser imperceptible por parte del usuario. La aplicación no deberá mostrar ANRs, bloquear el hilo principal ocupado o tener que reiniciarse para manejar cualquiera de las funcionalidades.	

Código	Nombre
2.1.15	Licencias de software
Descripción	
Todo el software implementado para este trabajo ha de ser software libre.	
Observaciones	
Las librerías utilizadas también serán software libre. No se importarán binarios; únicamente se importará código fuente.	

2.2 Casos de uso

Los casos de uso de este trabajo están enfocados a la demostración de la elección de un patrón de bloqueo basado en ritmos y celdas, así como en la validación exitosa o fallida del mismo. A continuación se detallan los casos de uso:

Código	Nombre
2.2.1	Elegir un patrón de bloqueo
Descripción	

El usuario abrirá la aplicación y podrá elegir definir un patrón de bloqueo.

El patrón de bloqueo se fijará tras la introducción del patrón 3 veces por parte del usuario.

El orden de las celdas ha de ser el mismo en las 3 introducciones.

El ritmo puede variar, ya que se calculará la media aritmética entre las introducciones.

La aplicación deberá informar al usuario que el patrón ha sido fijado con éxito. En caso de introducir erróneamente el orden de las celdas, la aplicación deberá informar al usuario que ha habido un error.

Observaciones
La introducción del patrón rítmico debería ser "similar" entre las 3 introducciones, ya que si el patrón cambia bastante, se verá reflejado a la hora de realizar la media aritmética.

Código	Nombre
2.2.2	Verificar con éxito el patrón de desbloqueo

Descripción
El usuario podrá abrir la aplicación y seleccionar la opción de realizar un desbloqueo. Se le mostrará la matriz de introducción del patrón y la aplicación estará esperando la introducción del patrón. La aplicación ha de informar si el patrón se ha introducido de forma exitosa.

Código	Nombre
2.2.3	Verificar sin éxito el patrón de desbloqueo

Descripción
Siguiendo el proceso del caso de uso anterior, la aplicación deberá informar al usuario cuando la introducción del patrón de desbloqueo no sea exitosa.

2.3 Estructuras de datos

La aplicación móvil deberá mantener una serie de estructuras destinadas al almacenamiento de los patrones definidos así como a los intentos de validación.

La estructura de datos elemental para el trabajo con los patrones será la de 'RhythmCell'. Esta estructura representa una celda del patrón, que constará de un identificador de posición y de una huella de tiempo, para saber el momento exacto en el que ha sido seleccionada. Esta huella de tiempo, será el parámetro destinado al cálculo del ritmo del patrón. Por lo tanto, esta estructura tendrá una representación similar a la siguiente:

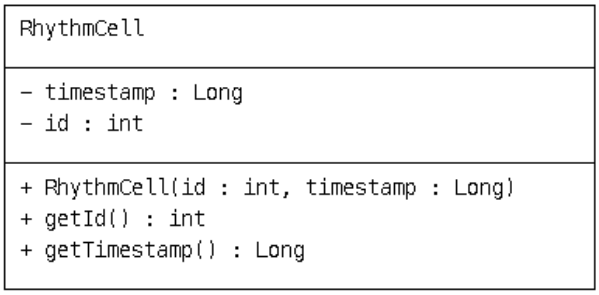

Figura 4: Estructura de una celda con ritmo – Elaboración propia

Un patrón estará formado por varias de las celdas anteriores. Por lo tanto se podría definir como una lista de celdas, todas ellas con un identificador de posición y una huella de tiempo. La estructura de un patrón, definido como 'RhythmPattern' será similar a la siguiente:

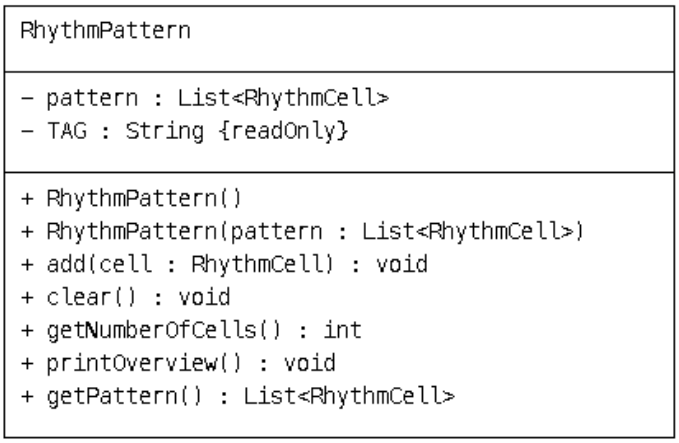

Figura 5: Estructura de un patrón – Elaboración propia

Dispondrá de un patrón, que estará formado por una lista de celdas con ritmo y un TAG. El TAG será utilizado solamente para efectos de log. Dentro de sus funciones, tendrá las opciones de instanciar el objeto, añadir nuevas celdas o borrar el patrón entre otras.

Tanto ‘RythmCell’ como ‘RythmPattern’ serán las estructuras básicas de la aplicación. Sobre estas se construirán los algoritmos y procesadores necesarios que realizarán la definición y validación de los patrones. Por ejemplo, el asistente de definición del patrón de bloqueo constará de 3 patrones.

Una vez el patrón ha sido definido por parte del usuario, este deberá almacenarse en el dispositivo para mantener persistencia tras el cierre de la aplicación. Para el desarrollo de este trabajo se va a utilizar el espacio de almacenamiento protegido de la propia sandbox[6] de la aplicación. Para ello, se van a utilizar las SharedPreferences[7] de Android como espacio de almacenamiento persistente al reinicio. En este espacio de almacenamiento se albergará el patrón definido por el usuario, y será el patrón contra el que se valide posteriormente el patrón introducido por el usuario a la hora de intentar realizar un desbloqueo.

2.4 Arquitectura

La interfaz de usuario de la aplicación constará principalmente de dos pantallas: la de inicio, desde la cual se podrá acceder tanto al asistente como a realizar un intento de validación del patrón, y una pantalla con la matriz de 9x9 destinada a la inserción de los patrones por parte del usuario.

La implementación de estas pantallas se realizará con el componente “Activity”[8] de Android destinado a tal efecto. A continuación se puede observar un diagrama con las interacciones entre las pantallas disponibles en la aplicación.

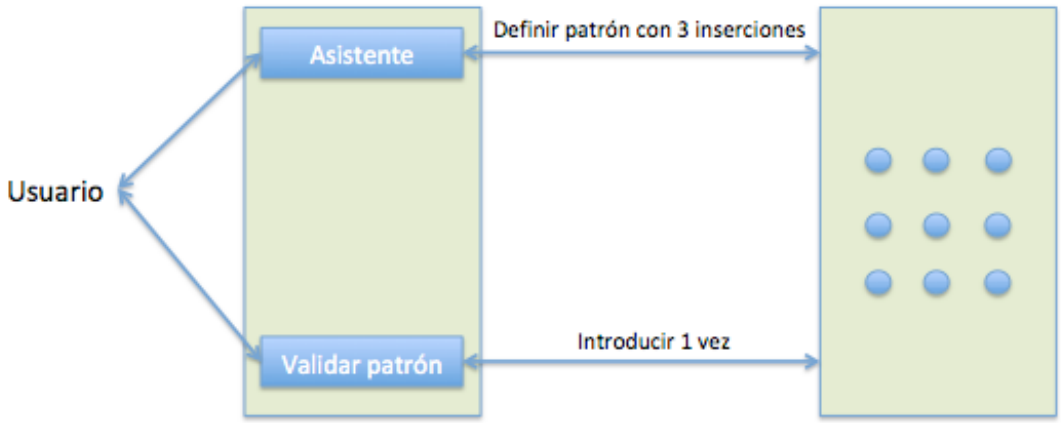

Figura 6: Pantallas de la aplicación – Elaboración propia

Como se puede observar en la figura 6, el proceso del asistente habitual constará de 3 inserciones y la introducción del patrón de 1. Independientemente de esto, si el usuario quiere salir de la pantalla del patrón, podrá utilizar los botones de navegación habituales del dispositivo Android, ya

que no se realizará ninguna modificación de estos y continuarán completamente operativos. Por lo tanto, si el usuario desde la pantalla de inserción del asistente quisiera salirse por ejemplo durante la segunda introducción, simplemente tendría que pulsar el botón “atrás” de su dispositivo.

El sistema estará compuesto por una aplicación y una librería, ambas desarrolladas para Android, con lenguajes Java y Kotlin[9]. La aplicación contendrá las dos “Activities” de Android que implementan los contenedores de interfaz gráfica para la pantalla principal y la de inserción de patrón.

La librería estará compuesta por el modelo de datos, la lógica de negocio y la implementación del almacenamiento del patrón definido por el usuario. Tanto la librería como la aplicación de Android estarán contenidas en un único proyecto de Android Studio y enlazadas entre sí. De este modo la modificación y compilación de la librería deberá verse inmediatamente reflejada en la aplicación de Android.

El almacenamiento del patrón de bloqueo definido por el usuario se realizará desde la librería, haciendo uso de las SharedPreferences de Android.

La funcionalidad principal de este trabajo se encontrará implementada en la librería. La aplicación de Android está enfocada únicamente a fines demostrativos del funcionamiento planteado. La librería podrá importarse en cualquier proyecto adicional de Android y podrá ser usada por terceras partes para realizar potenciales integraciones.
A continuación se puede observar un diagrama con los principales componentes del sistema:

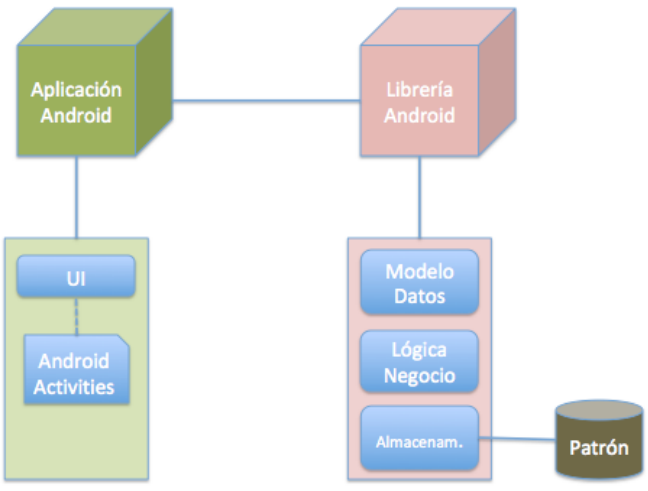

Figura 7: Diagrama de sistema – Elaboración propia

2.5 Algoritmo de validación rítmica

La validación rítmica es uno de los objetivos de este trabajo. Para ello se ha diseñado un algoritmo capaz de validar el ritmo de una forma eficiente. Cada

celda introducida por el usuario se realizará de manera lineal en el tiempo. La aplicación registrará el instante en el tiempo en el que se ha introducido cada celda.

Al dispone de las huellas de tiempo de cada una de las celdas se calculará el tiempo que ha pasado entre cada una y la siguiente. Esto formará el patrón rítmico. La sensibilidad con la que se registra cada huella de tiempo es de milisegundos.

Cada vez que se introduce un mismo patrón de celdas es altamente improbable que se introduzca exactamente el mismo patrón rítmico y esto se debe a la sensibilidad de milisegundos. Es por esta razón por la que se introducirá un umbral de corrección para paliar este efecto, lo que permitirá aceptar pequeñas variaciones como válidas, siempre y cuando se respete la forma del patrón. Para el inicio de este trabajo se ha definido un umbral del 0.2 sobre 1 entre pares de dos notas. Este umbral se ajustará durante la fase de pruebas de usuario.

La parte de validación del ritmo entre el patrón del usuario y el del asistente, tendrá una estructura similar al siguiente fragmento de pseudocódigo:

```
for(int i=0;i<userPattern.getNumberOfCells();i++){
    long userCellDelay =
userPattern.getPattern().get(i).getDelayWithPreviousCell();
    long wizarCellDelay =
wizardPattern.getPattern().get(i).getDelayWithPreviousCell();
    long variableDelay = wizarCellDelay/100*20;
    if(userCellDelay < (wizarCellDelay - variableDelay) ||
userCellDelay > (wizarCellDelay + variableDelay)) {
        //Diferente patrón de ritmo. Validación fallida
    }
}
```

Figura 8: Validación del ritmo entre dos celdas – Elaboración propia

Se considerará una validación exitosa del patrón en el caso en el que el ritmo calculado de todos los pares de celdas introducidos por el usuario para tratar de realizar un desbloqueo, se encuentre dentro del ritmo del patrón definido durante la fase del asistente.

2.6 Interfaz gráfica

La interfaz de este trabajo ha de permitir al usuario definir un patrón de bloqueo así como realizar un intento de desbloqueo. Por ello, la pantalla principal de la aplicación constará de dos botones, que darán acceso a las funcionalidades anteriormente descritas. Toda la interfaz gráfica estará implementada con componentes estándar de Android, con el objetivo de maximizar la compatibilidad entre diferentes dispositivos. A continuación se puede ver un ejemplo de la pantalla principal:

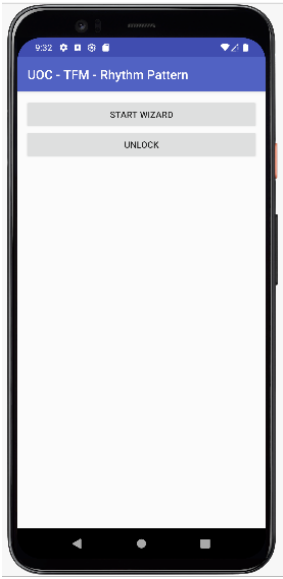

Figura 9: Aplicación – Pantalla principal – Elaboración propia

Como se puede observar en la figura 7, existen dos botones para dar acceso en primer lugar a comenzar el asistente y en segundo lugar a realizar un desbloqueo. Se ha elegido el idioma inglés para la aplicación con el objeto de ser mas facilmente identificable por parte de posibles futuros desarrolladores/integradores. El primer paso por parte del usuario será el de definir un patrón de desbloqueo. Una vez pulsada la opción de comenzar el asistente se le mostrará una pantalla como la siguiente:

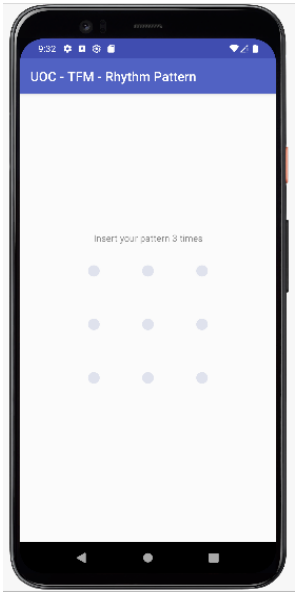

Figura 10: Aplicación – Matriz de patrón – Elaboración propia

En la figura 8 se puede ver la matriz de 9x9 del patrón que usará la aplicación. Esta matriz será la utilizada para definir los patrones tanto de celdas como rítmicos.

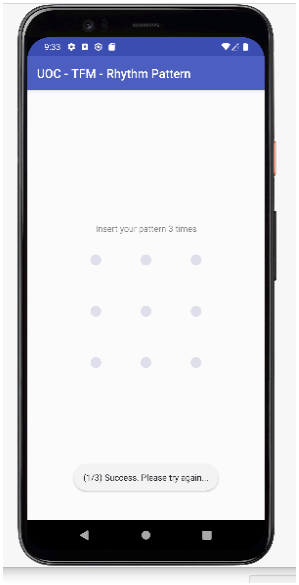

Figura 11: Aplicación – Primer intento en asistente – Elaboración propia

En la figura 9 se puede ver el feedback que se le dará al usuario a medida que va introduciendo diferentes intentos en el asistente para definir el patrón de desbloqueo. Aparecerá un mensaje indicando el intento por el que va y si se ha considerado válido o fallido. Si las posiciones de las celdas coinciden con la inserción anterior, o si la inserción es la primera de todas, se mostrará un aviso de éxito como el disponible en la figura 9.

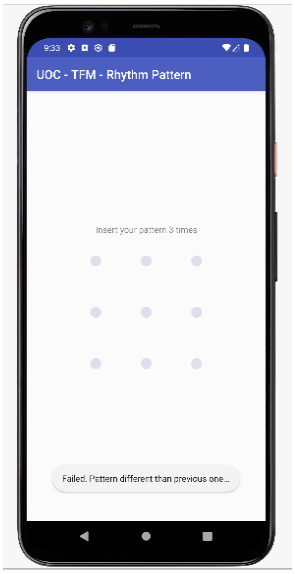

Figura 12: Aplicación – Intento fallido en asistente – Elaboración propia

En el caso de realizar una inserción fallido del patrón durante el asistente, se mostrará un mensaje similar al mostrado en la figura 10.

Existirá una retroalimentación para el usuario cuando interaccione con las posiciones de las celdas de la matriz de cara a ayudar al usuario a conocer el patrón que está introduciendo (de forma similar a como sucede en las pantallas de bloqueo habituales de los dispositivos Android). A continuación se puede observar cómo se representará la inserción de un patrón por parte del usuario:

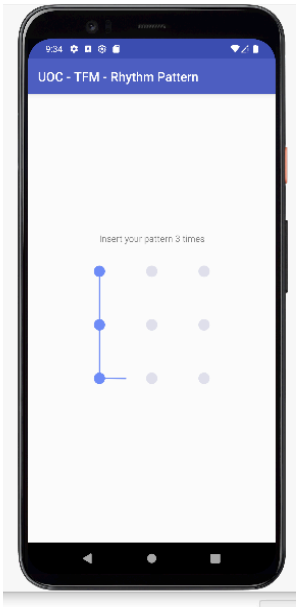

Figura 13: Aplicación – Introducción de patrón – Elaboración propia

Una vez el patrón ha sido completado con éxito durante el asistente, la pantalla del patrón se cerrará y se notificará al usuario que el asistente ha sido completado.

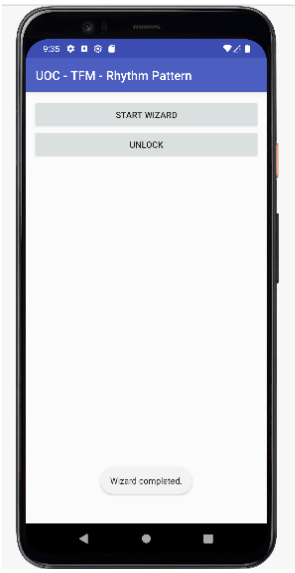

Figura 14: Aplicación – Asistente completado – Elaboración propia

Una vez el patrón de bloqueo ha sido definido, el usuario puede proceder con la opción de intentar desbloquear el dispositivo. En este caso, si introduce el patrón de desbloqueo con éxito, se le informará como se puede observar en la figura 13.

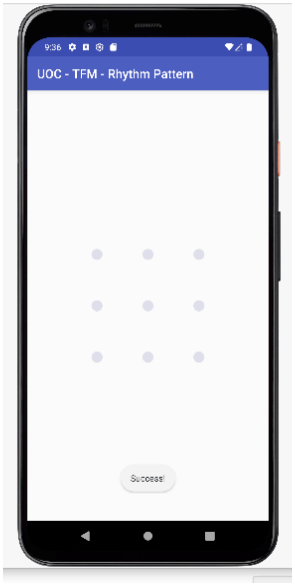

Figura 15: Aplicación – Desbloqueo exitoso – Elaboración propia

En caso de introducción de un patrón erróneo se notificará tal y como se muestra en la figura 14 a través de un mensaje de error y mostrando el patrón introducido por el usuario para intentar desbloquear el dispositivo en rojo.

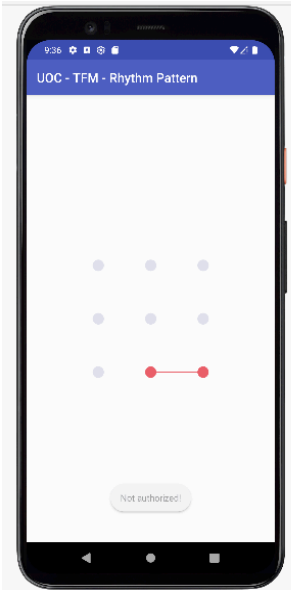

Figura 16: Aplicación – Desbloqueo fallido – Elaboración propia

3. Desarrollo de la aplicación

3.1 Gestión y organización

Este trabajo se ha estructurado en un proyecto de 'Android Studio' que contiene dos módulos: la librería de 'RhythmPattern' y una aplicación de ejemplo 'sample' que la integra.

El proyecto hace uso de la librería 'com.itsxtt.patternlock' sobre la cual se ha construido íntegramente la parte de validación rítmica. El desarrollo de este proyecto comenzó validando que fuera viable utilizar dicha librería para poder representar los aspectos gráficos del patrón así como su validación.

Posteriormente se añadió la capa de validación rítmica, y se enlazó con la interfaz gráfica disponible en la aplicación de ejemplo. También se hizo uso de las 'SharedPreferences' para almacenar información necesaria para la validación del patrón, concretamente el propio patrón de desbloqueo definido por el usuario en la fase del asistente.

Para la gestión del repositorio de código fuente se ha utilizado la herramienta Git.

Para realizar las pruebas se ha utilizado el emulador AVD disponible en Android Studio, lanzando un dispositivo emulado modelo "Google Pixel 4" con API nivel 31, que corresponde a la versión de Android 12.

3.2 Estructura de carpetas

Todo el código fuente del proyecto está incluido en un directorio raíz con el nombre "**RhythmPattern**". Este directorio incluye los dos módulos de desarrollo, correspondientes a la aplicación y a la librería. Ambos módulos de desarrollo tienen la estructura de un proyecto compatible con la herramienta Android Studio.

A continuación se puede ver dicha estructura descrita:

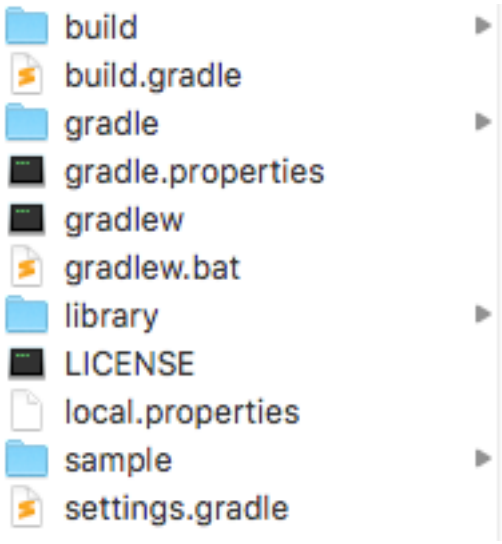

Figura 17: Estructura del proyecto – Elaboración propia

El código fuente principal del proyecto se encuentra contenido dentro del directorio src/main en sendos directorios “library” y “sample”.

El directorio que corresponde a la aplicación posee el nombre “**sample**”. Dentro de sample se encuentra el directorio “**src/main**” que incluye todo el código fuente de la parte de la aplicación. El directorio “main” posee a su vez dos directorios que son java y res. En el directorio “java” se pueden encontrar todos los ficheros que poseen código Java/Android/Kotlin, mientras que a su vez en “res” se encuentran los recursos de la aplicación como iconos, vistas gráficas o cadenas de texto.

En el caso del código fuente de la aplicación existen dos ficheros con extensión “kt” ya que son ficheros de Kotlin. El primer fichero existente es el “MainActivity” y corresponde al código fuente de la actividad principal de la aplicación. El segundo es “PatternLockActivity” y corresponde al código fuente de la actividad que muestra la celda de 3x3 destinada a la inserción de los patrones.

La estructura de la librería es similar al de la aplicación salvo porque apenas tiene recursos en el directorio “res” al tratarse de una librería. Por el contrario, en el directorio “src” se encuentra el código fuente. Dentro del código fuente existen dos paquetes de Android claramente diferenciados. Por un lado está “com.itsxtt.patternlock” que en su raíz tiene implementadas principalmente tres clases correspondientes a la entidad celda, la vista del patrón de desbloqueo y el estado. Por otro lado existe un directorio “rhythm” que da entidad al paquete “com.itsxtt.patternlock.rhythm”. A continuación se puede ver una imagen de ambas estructuras:

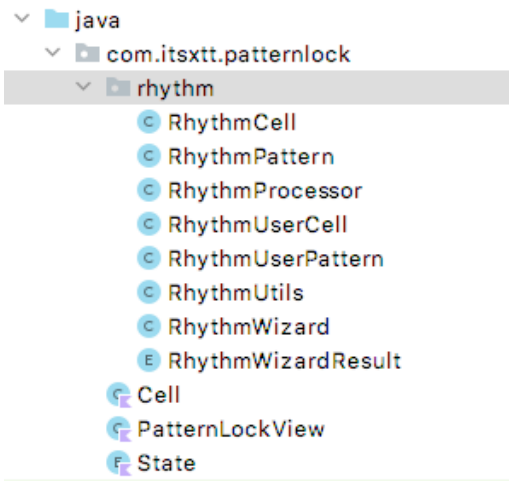

Figura 18: Ficheros de la librería – Elaboración propia

Los objetos que se incluyen en el directorio “rhythm” corresponden a entidades y estructuras relacionadas con la validación rítmica como patrones, procesadores de la validación rítmica o la implementación del asistente de definición del patrón.

3.3 Modelos

Las entidades del proyecto se encuentran principalmente en el módulo de la librería. En dicha librería podemos encontrar las siguientes entidades:

- **RhythmCell** – Representa a una celda del patrón de desbloqueo. Dicha celda tiene un identificador de posición y un timestamp, que corresponde al momento concreto en el que la celda ha sido seleccionada.

- **RhythmPattern** – Un patrón es una representación de varias celdas pulsadas de manera secuencial. Dicho patrón consta de una lista ordenada de RhythmCells.

- **RhythmProcessor** – Esta entidad es un procesador que maneja el estado de la aplicación, desde si se encuentra en estado de asistente o introducción de un patrón hasta realizar limpiezas de las estructuras.

- **RhythmUserCell** – Representa una celda del patrón introducida específicamente por parte de un usuario. Estas celdas tienen un delay como propiedad que representa la diferencia de tiempo en el que se pulsó en relación a la celda anterior. Esto se utiliza para calcular el patrón rítmico.

- **RhythmUserPattern** – Corresponde a una lista de RhythmUserCells del usuario.

- **RhythmUtils** – Hace referencia a un conjunto de utilidades para operar con estructuras rítmicas. Realiza desde validaciones de patrones hasta conversiones de patrones entre los mismos.

- **RhythmWizard** – Define la operativa durante un proceso de definición del patrópn de desbloqueo desde el asistente.

- **RhythmWizardResult** – Representa los posibles resultados que pueden estar sucediendo durante una fase de definición de un patrón de desbloqueo.

- **Cell** – Corresponde a la implementación de una celda gráfica del patrón de desbloqueo.

- **PatternLockView** – Representa la vista de la matriz de 3x3 utilizada para el patrón de desbloqueo.

- **State** – Ofrece los valores posibles de aspecto gráfico de una celda de la matriz de 3x3.

Por otro lado, la aplicación 'sample' únicamente dispone de las dos 'Activities' utilizadas para representar los aspectos gráficos de las dos pantallas de la aplicación.

3.4 Vistas

El proyecto cuenta con una serie de vistas destinadas principalmente a implementar la inserción y validación de los patrones. En la librería se encuentra la vista "PatternLockView" que corresponde a una vista de tipo GridLayout[10] que implementa todos los aspectos gráficos de la vista de la celda de 3x3. Esta vista se utiliza posteriormente tanto para la opción de definir el patrón de desbloqueo como para desbloquear dicho patrón.

La implementación gráfica de esta vista en Android se correspondería con la siguiente imagen:

Figura 19: Vista 'PatternLockView' – Elaboración propia

En el módulo de la aplicación existen dos vistas adicionales que corresponden a la pantalla de selección de opción por parte del usuario (definir patrón o

validarlo) y a la propia definición/introducción del patrón. Ambas vistas están implementadas a través de layouts en el proyecto de la aplicación. Los layouts son “activity_main.xml” que corresponde a la pantalla principal de la aplicación y “activity_pattern_default.xml” que corresponde a la vista que integra la celda de 3x3 de la librería y que se utiliza tanto para definir el patrón como para validarlo. A continuación se puede ver una muestra de la definición de la vista de la pantalla principal:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="12dp">

    <Button
        android:id="@+id/startWizardBtn"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Start Wizard" />

    <Button
        android:id="@+id/unlockBtn"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Unlock" />

</LinearLayout>
```

Figura 20: Layout de pantalla principal – Elaboración propia

Tal y como se puede observar en la figura anterior, la vista principal está compuesta por dos botones destinados a la definición del patrón y a la introducción del patrón para su desbloqueo.

Por otro lado, el layout que realiza la integración de la vista de la celda de 3x3 tiene el siguiente aspecto:

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center">

    <com.itsxtt.patternlock.PatternLockView
        android:id="@+id/defaultPatternLockView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="72dp"
        android:layout_marginRight="72dp">
    </com.itsxtt.patternlock.PatternLockView>

</LinearLayout>
```

Figura 21: Layout de pantalla de patrón – Elaboración propia

En la figura anterior se puede observar que el layout de la aplicación integra a su vez la vista de la celda definida en la librería. Esta estructura será utilizada para realizar todas las tareas en las que se haga uso de la introducción de un patrón.

3.5 Activities

La implementación de este trabajo cuenta solamente con dos componentes 'Activity' de Android que se encuentran en la aplicación 'sample'. Las actividades se denominan "MainActivity" y "PatternLockActivity". El objeto de estas actividades es contener la principal implementación gráfica de la aplicación.

Tal y como sus nombres indican, la actividad "MainActivity" es la actividad principal de la aplicación, desde la que se pueden seleccionar dos opciones: definir un patrón de bloqueo a través del asistente o realizar un intento de desbloqueo. Por otro lado, la activity "PatternLockActivity" es la encargada de mostrar la cuadricula de 3x3 desde la cual se realiza la validación o definición del patrón de bloqueo.

3.6 Fragments

De cara a posibilitar una sencilla integración de este proyecto por parte de un tercero, se ha enfocado el desarrollo en componentes basados únicamente en vistas y Activities que puedan ser integrados de forma sencilla en otros componentes como Fragments u otras Activities. Es por esta razón por la que este proyecto no incluye ningún Fragment.

3.7 Adapters

Este proyecto no hace uso de ningún Adapter[12] de Android ya que todo el manejo de la información lo realiza tanto en la memoria destinada al proceso de la aplicación como en el espacio de almacenamiento del dispositivo destinado a la propia aplicación.

3.8 Bases de datos

Todo el almacenamiento de este proyecto se realiza a través de las 'SharedPreferences' de Android, por lo que no se incluye ningún motor adicional de base de datos.

El único dato almacenado por la aplicación corresponde al patrón de bloqueo definido por el usuario a través del asistente. A continuación se puede ver el bloque de código que representa el almacenamiento a base de datos:

```
SharedPreferences mPrefs =
context.getSharedPreferences("educational_purposes",MODE_PRIVATE);
SharedPreferences.Editor prefsEditor = mPrefs.edit();
prefsEditor.putString("wizard_pattern",
RhythmUtils.fromPatternToString(getUserPattern()));
prefsEditor.commit();
```

Figura 22: Almacenamiento a base de datos – Elaboración propia

Como la aplicación 'sample' únicamente se ha implementado con fines demostrativos, los datos no se almacenan cifrados ni se hace uso de ningún otro espacio de almacenamiento con seguridad adicional como podría ser un chip criptográfico.

3.9 Diagrama de clases

A continuación se puede ver el diagrama de clases de la librería del proyecto:

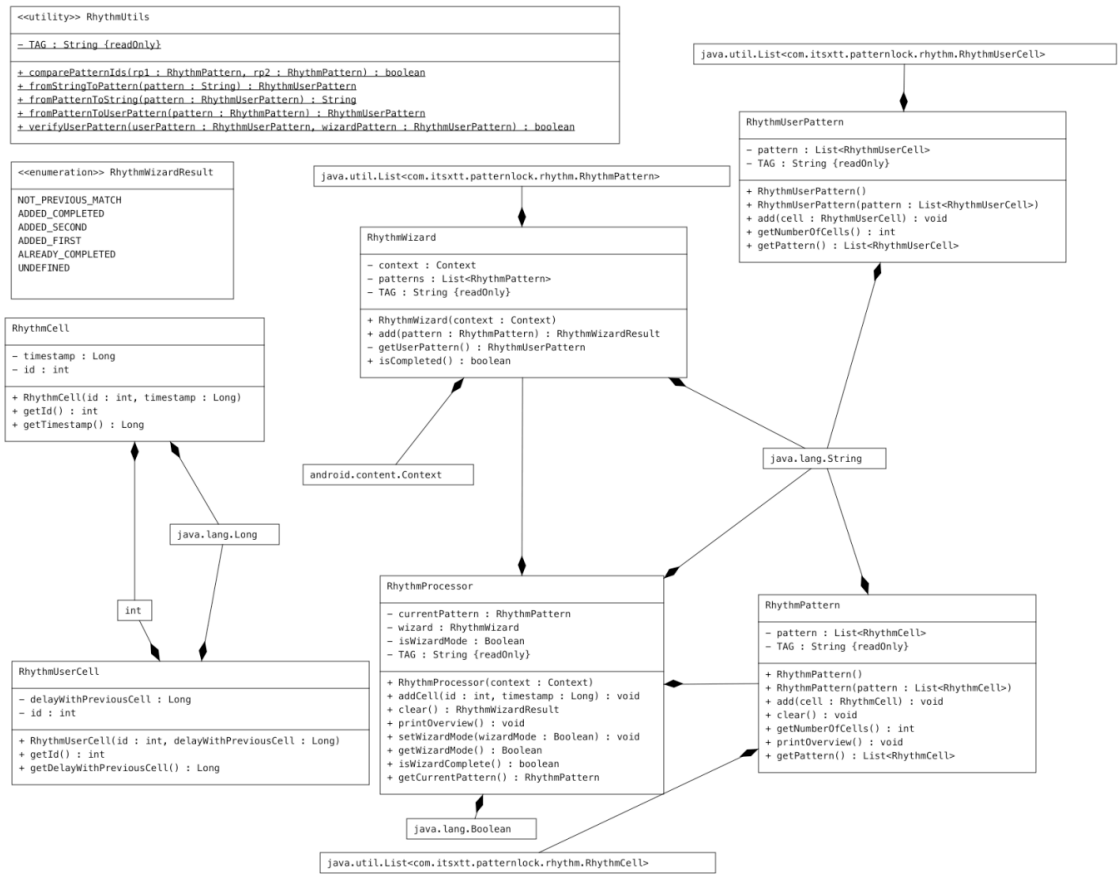

Figura 23: Diagrama de clases de la librería – Elaboración propia

Tal y como se puede observar en el diagrama de clases de la librería 'RhythmProcessor', ahí se encuentran contenidas todas las entidades que forman parte del patrón, así como de las estructuras necesarias para su procesamiento, como 'RhythmProcessor', utilizado para ir gestionando diferentes aspectos de la fase en la que se encuentra la definición o desbloqueo de patrón.

Los componentes existentes en la librería, son utilizados desde la aplicación 'sample'. La aplicación consta únicamente de dos 'Activities', desde los cuales se invoca a:

- **PatternLockView** – Para utilizar la vista de la matriz 3x3.
- **RhythmPattern** – Para disponer de la estructura de un patrón de ritmo.
- **RhythmUtils** – Para utilizar diferentes funcionalidades referentes al manejo de patrones de ritmo.
- **RhythmWizardResult** – Para obtener información durante el proceso de definición del patrón durante el asistente.

3.10 Librerías externas

La librería ‘RhythmPattern’ de este trabajo está construida sobre la librería ‘com.itsxtt.patternlock’ que se utilizó como base para la representación gráfica de la celda de 3x3. Sobre dicha librería se ha implementado la validación rítmica del patrón.

Adicionalmente, este proyecto hace uso de la librería ‘androidx.appcompat:appcompat’, utilizada para ampliar la retrocompatibilidad de la aplicación con diferentes versiones de Android. No se utiliza ninguna otra librería adicional.

3.11 RhythmPattern en acción

La implementación del proyecto se corresponde con los requisitos inicialmente definidos, por lo que tanto el aspecto de la interfaz gráfica como los casos de uso se encuentran alineados con la definición previa. Un aspecto positivo de la definición, es el hecho de realizar el diseño utilizando componentes específicos de Android, por lo que se minimiza el riesgo de diferencia entre el diseño y la implementación final.

A continuación se puede ver un uso real de la aplicación a través del emulador disponible en ‘Android Studio’. En este caso se está intentando realizar un desbloqueo de un patrón previamente definido:

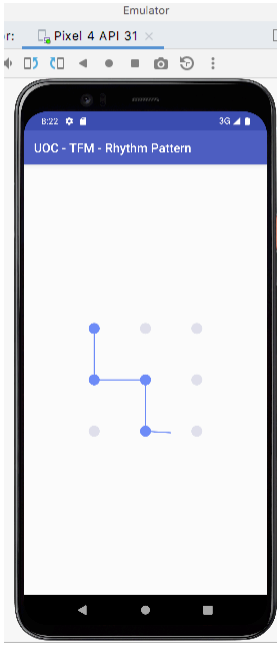

Figura 24: RhythmPattern en acción – Elaboración propia

El uso se puede realizar directamente a través del emulador o también desplegando la aplicación en un dispositivo físico.

Los botones de navegación de Android (‘atrás’, ‘minimizar’ o ‘poner en segundo plano’) no han sido redefinidos, por lo que su funcionamiento es exactamente el

estándar en un dispositivo Android. Desde la pantalla de la celda 3x3 se puede acceder a la pantalla principal pulsando el botón 'atrás' de cualquier dispositivo Android.

La aplicación muestra mensajes de éxito o fallo en el caso de validaciones exitosas o fallidas del patrón de bloqueo.

3.12 Compilación del proyecto

El directorio raíz del proyecto se puede abrir directamente desde Android Studio, el cual cargará los dos módulos (librería y aplicación) en la herramienta, tal y como se puede observar en la siguiente imagen:

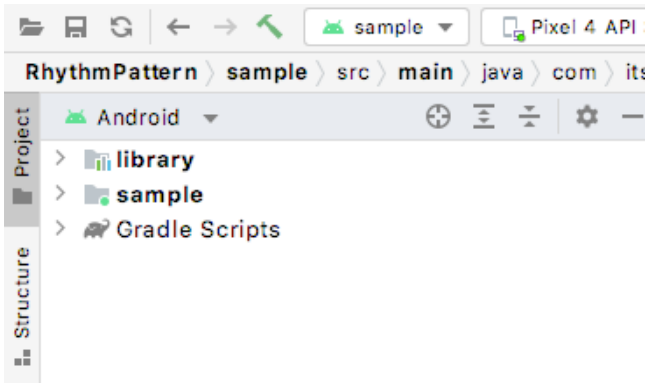

Figura 25: Módulos en Android Studio – Elaboración propia

El módulo ejecutable corresponde al "sample". Este módulo se puede desplegar directamente sobre un terminal físico o emulador Android, y es el que integra el funcionamiento del proyecto, incluyendo la librería. Además de estos módulos, existen una serie de scripts de gradle[13] necesarios para la correcta compilación y ejecución del proyecto.

Existe un principal binario resultante de la compilación del proyecto que es el fichero APK (Android Package Kit). Este es el archivo instalable en un dispositivo Android. Una vez compilado el proyecto, este fichero se genera en la siguiente ruta del proyecto:

```
RhythmPattern/sample/build/outputs/apk/debug/sample-debug.apk
```

El fichero APK anteriormente descrito incluye la librería. Este fichero es multidispositivo y está compilado en modo debug. También se podría compilar en modo release activando la ofuscación a través de la modificación de los scripts de gradle. De cara a la ofuscación, existen unos ficheros de proguard en el proyecto con extensión "pro" que están destinados a que el usuario que quiera compilar el proyecto pueda especificar sus reglas personalizadas de ofuscación del código.

Para construir el proyecto, únicamente será necesario tener seleccionado el módulo "sample" y ejecutar la opción "Build -> Make project" en Android Studio.

Una vez hecho esto, el proyecto se construirá y generará el APK resultante. A continuación se muestra una compilación del proyecto y la salida esperada:

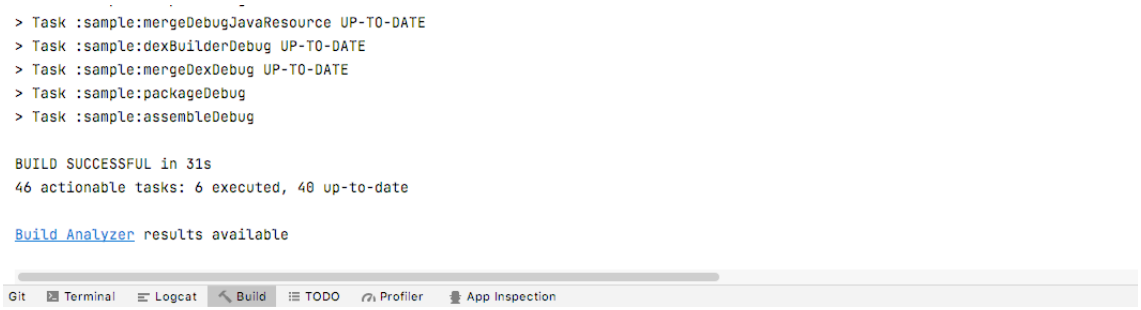

Figura 26: Construcción del proyecto – Elaboración propia

Como se puede observar en la imagen anterior, Android Studio informa de la compilación exitosa del proyecto. En este momento se genera el fichero instalable APK.

También existe la opción de desplegar el proyecto directamente sobre un emulador o un terminal físico Android. Para ello habría que seleccionar el dispositivo en el espacio de selección de dispositivos de Android Studio, y darle a “Run sample”. En la siguiente imagen se muestra un ejemplo de ejecución en un emulador:

Figura 27: Ejecución sobre emulador – Elaboración propia

Para lanzar el proyecto sobre un terminal físico Android, habría que conectar este por USB al equipo y habilitar las opciones de desarrollador así como la depuración USB, siguiendo las instrucciones específicas del fabricante de cada dispositivo. Una vez hecho esto, Android Studio reconoce el dispositivo y se podría lanzar la aplicación directamente sobre el mismo.

3.13 Fase de pruebas

Una vez implementado el algoritmo de validación rítmica así como las vistas de la interfaz gráfica, la aplicación se sometió a una fase de pruebas con usuarios independientes con diferentes perfiles. Para la fase de pruebas se definieron los siguientes escenarios:

Test Case	Nombre
1	Registro de un nuevo patrón de desbloqueo
Descripción	

El usuario definirá un nuevo patrón de desbloqueo que quedará fijado en la aplicación
Resultado esperado
El usuario ha de ser capaz de definir un nuevo patrón de desbloqueo de manera intuitiva y sin ninguna complicación

Test Case	Nombre
2	Desbloqueo exitoso del patrón
Descripción	
El usuario deberá realizar 3 intentos de desbloqueo del patrón previamente definido.	
Resultado esperado	
El usuario deberá ser capaz de desbloquear el patrón con una tasa de éxito en al menos 2 ocasiones.	

Test Case	Nombre
3	Intento de acceso por parte de otro usuario
Descripción	
Un tercer usuario deberá intentar desbloquear un patrón previamente definido por otro usuario conociendo únicamente el orden de celdas introducidas (sin conocer el patrón rítmico definido)	
Resultado esperado	
El tercer usuario no ha de ser capaz de desbloquear el patrón previamente definido por otro usuario.	

Adicionalmente se preparó una breve plantailla de resultados para los usuarios participantes en la fase de pruebas. Dicha plantilla siguió el siguiente formato:

Usuario:	1
Edad:	39
Perfil técnico	4
Definición del patrón de desbloqueo	OK
Intento desbloqueo 1	OK
Intento desbloqueo 2	FAIL
Intento desbloqueo 3	OK
Intento desbloqueo por un tercero	OK

Figura 28: Ejemplo resultado plantilla pruebas – Elaboración propia

Con esta plantilla se pretende verificar que los requerimientos de definición de patrón de bloqueo y la implementación del desbloqueo se satisfacen correctamente para un conjunto de usuarios de entre 12 y 70 años con diferentes perfiles de conocimientos técnicos.

El perfil técnico se mide en unidades de 1 a 5 en función de la percepción del propio usuario participante en la fase de pruebas en relación a sus conocimientos técnicos dentro del ámbito digital.

El valor 'OK' del apartado "Intento desbloqueo por un tercero" corresponde a que el usuario tercero no sea capaz de realizar el desbloqueo del patrón previamente definido por parte del usuario previo.
Se desea obtener una tasa de éxito superior al 90% en la validación del patrón de desbloqueo previamente definido. Esta tasa se medirá calculando el número de aciertos frente al número de fallos a la hora de validar el patrón previamente definido por un usuario.

Para la fase de pruebas se seleccionaron 10 usuarios con los siguientes perfiles:

Usuario	Edad	Perfil técnico
1	39	4
2	12	4
3	37	5
4	36	4
5	25	2
6	27	4
7	48	3
8	48	3

9	68	1
10	70	2

Figura 29: Perfiles usuario de prueba – Elaboración propia

Al comienzo de cada prueba, se le explicó a cada usuario de forma breve y simple en qué consiste la aplicación y el proceso de validación de patrón de desbloqueo basado en ritmo. Todos los usuarios comprendieron de forma sencilla el concepto del proyecto. A continuación se muestran los resultados obtenidos tras las pruebas de todos los usuarios:

	Definición patrón	Intentos	Desbloqueo Tercero
Usuario 1:	OK	OK FAIL OK	OK
Usuario 2:	OK	FAIL OK OK	OK
Usuario 3:	OK	FAIL OK OK	OK
Usuario 4:	OK	FAIL OK OK	OK
Usuario 5:	OK	FAIL FAIL OK	OK
Usuario 6:	OK	OK OK OK	OK
Usuario 7:	OK	FAIL OK OK	OK
Usuario 8:	OK	OK FAIL OK	OK
Usuario 9:	OK	FAIL FAIL OK	OK
Usuario 10:	OK	OK OK OK	OK

Figura 30: Resultados ronda 1 fase de pruebas – Elaboración propia

Tal y como se puede observar en la figura anterior, todos los usuarios fueron capaces de definir el patrón de bloqueo sin ninguna complicación y no se aprecia variación en los resultados en función de los perfiles de los usuarios.

Los resultados de la validación por parte de un tercero también fueron exitosos, ya que en ningún caso el tercero fue capaz de desbloquear el patrón del usuario sin conocer el patrón rítmico. Sin embargo, los resultados de validación del patrón de desbloqueo arrojaron una tasa de éxito de un 66.7%, una cifra inferior al valor superior del 85% deseado. Para ello se decidió modificar el umbral de tolerancia del algoritmo de validación rítmica, ampliándolo de un 0.2 sobre 1 a un 0.4 sobre 1. Se decidió volver a realizar una fase de pruebas con el umbral definido al valor de 0.4. A continuación se pueden ver los resultados de la segunda ronda de pruebas:

	Definición patrón	Intentos	Desbloqueo Tercero
Usuario 1:	OK	OK OK OK	OK
Usuario 2:	OK	OK OK OK	OK
Usuario 3:	OK	OK OK OK	OK
Usuario 4:	OK	OK OK OK	OK
Usuario 5:	OK	OK OK OK	OK
Usuario 6:	OK	OK OK FAIL	OK
Usuario 7:	OK	OK OK OK	OK
Usuario 8:	OK	OK OK FAIL	OK
Usuario 9:	OK	OK OK OK	OK
Usuario 10:	OK	OK FAIL OK	OK

Figura 31: Resultados ronda 2 fase de pruebas – Elaboración propia

En este caso la tasa de éxito se incrementó hasta el 93% mientras que el intento de validación por parte de un tercero siguió siendo de un 100% por lo que se definió el umbral al 0.4 sobre 1 para el algoritmo de validación rítmica.

4. Resultados

El planteamiento fundamental de este trabajo fin de máster es el de tratar de proponer un modelo de seguridad adicional para la privacidad de los usuarios de dispositivos móviles en cuando a las capacidades de desbloqueo del terminal a través de patrón físico se refiere.

La implementación del trabajo previsto ha dado como resultado una aplicación móvil para la plataforma Android de definición y validación de patrones de bloqueo basados en ritmo. La aplicación es compatible con la versión 12 de Android y se puede ejecutar con éxito en un emulador de Android AOSP con una configuración por defecto. Además, la aplicación incluye una librería donde se implementa la lógica correspondiente a la validación rítmica, tal y como se definió como objetivo al inicio de este proyecto.

Las herramientas utilizadas durante el transcurso del proyecto han sido las definidas al inicio del mismo, dando especial importancia al uso de la herramienta 'Android Studio' para la elaboración de la aplicación y la librería, así como a GIT para el control de versiones del código y Microsoft Office para la elaboración de la documentación y la recogida de los valores de prueba, a través de las herramientas Microsoft Word y Microsoft Excel.

La aplicación permite definir un patrón de bloqueo por parte de un usuario sobre una matriz de 3x3 a través de un asistente y guarda una huella temporal de cada pulsación sobre las celdas del patrón. De esta manera, se guarda tanto el patrón gráfico como el rítmico; así se consigue que el patrón esté compuesto por dos dimensiones, la secuencial (correspondiente al orden en el que se seleccionan las celdas) y la rítmica, que corresponde al tiempo que pasa entre cada pulsación.

Una de las funcionalidades de la aplicación es la de disponer de un asistente que permite realizar un tutorial guiado para fijar la definición del patrón de desbloqueo rítmico. Este asistente se implementó requiriendo la introducción del patrón a definir en 3 ocasiones, para así obtener un modelo con la media de las 3 introducciones y utilizarlo como patrón de referencia. La selección de las 3 inserciones ha arrojado resultados positivos, ya que el asistente se completa de manera sencilla y breve, en comparación con otros asistentes que suelen requerir mayores tiempos de registro como en el caso de los lectores de huellas, que suelen requerir un mayor número de inserciones. El modelado a partir de las 3 inserciones ha permitido generar un patrón base lo suficientemente robusto para que un tercero no conocedor del patrón rítmico no fuera capaz de desbloquear el terminal y en cambio el usuario que lo definió si sea capaz.

La aplicación considera válida la comparativa de patrones siempre y cuando se satisfagan las dimensiones de secuencia y ritmo. El usuario obtiene feedback visual por parte de la aplicación a través de mensajes de éxito o fallo a la hora de realizar un intento de desbloqueo del terminal. La aplicación de ejemplo se ha conseguido implementar ofreciendo feedback al usuario sobre sus

interacciones respecto al patrón. El feedback visual se provee tanto a la hora de realizar un intento de desbloqueo como durante el proceso de definición del patrón a través del asistente.

Se implementó un algoritmo de validación rítmica para la dimensión del ritmo que se incluyó en la librería del proyecto. El algoritmo almacena las diferencias de tiempo entre la selección de las celdas del patrón.

Para la parte rítmica, se definió un umbral de tolerancia entre la pulsación de cada nota. Esto se debe a que debido al nivel de detalle a la hora de obtener el timestamp de cada pulsación, hace falta contemplar cierta variación entre la introducción de patrones de manera secuencial por parte de un usuario, ya que es altamente improbable que un usuario introduzca el mismo patrón en términos de tiempo (tendría que pulsar cada celda exactamente con el mismo número de milisegundos entre cada una de ellas).

En la fase de diseño del proyecto, se definió un umbral de tolerancia del 20% entre la introducción de cada nota. Si bien el proceso de bloqueo y desbloqueo era mayoritariamente exitoso con este valor, había ocasiones en las que el patrón que un usuario creía haber introducido correctamente era validado incorrectamente, debido a que no lo había hecho dentro del umbral de tolerancia definido. Esta ronda de pruebas arrojó una tasa de éxito del 66.7%. Tras estos resultados, se modificó el umbral de tolerancia al 40% y se obtuvo una tasa de éxito del 93% manteniendo la tasa de protección ante un ataque por parte de un tercero que únicamente es conocedor de la secuencia de celdas en un 100%.

Cabe destacar que no existió ningún falso positivo durante la fase de pruebas de la aplicación, evaluando esa métrica en función del número de desbloqueo por parte de un tercero no conocedor del patrón rítmico.

Se ha conseguido encapsular toda la operativa relativa a la dimension rítmica íntegramente en la librería. Las estructuras y modelos también se encuentran en la misma. De esta forma se permite su sencilla integración por parte de un tercero importando únicamente dicha librería en su proyecto. La librería únicamente hace uso de clases estándar de Android/Java/Kotlin por lo que no hace falta ninguna librería adicional para su correcto funcionamiento.

Otro de los aspectos particulares del proyecto, es que dentro del código implementado no se encuentra ningún código que haga uso de licencias comerciales, por lo que podría ser utilizado por terceras partes para realizar posibles integraciones en productos de terceros, en línea con los objetivos globales del proyecto de tratar de mejorar la privacidad de los usuarios.

Teniendo en cuenta los objetivos del trabajo definidos en el apartado 1.3 y tras las pruebas realizadas con los usuarios y los resultados recientemente expuestos, se considera que se ha añadido la capa adicional de seguridad para la privacidad de los usuarios, ya que terceros conocedores de la secuencia del patrón de desbloqueo de otro usuario no han sido capaces de desbloquear el

patrón gracias a la validación rítmica existente en él, por lo que se considera que se han alcanzado los objetivos definidos al inicio del proyecto.

5. Conclusiones y trabajos futuros

Es esencial que la privacidad de los usuarios se respete especialmente cuando se trata de dispositivos móviles, ya que estos están continuamente con nosotros accediendo a una gran cantidad de información por lo que podrían recopilar y transmitir una gran cantidad de datos. A continuación se detallan las conclusiones y trabajos futuros derivados de este trabajo de cara a incrementar la privacidad de los usuarios en el ámbito de los dispositivos móviles.

5.1 Conclusiones

Un aspecto fundamental para los usuarios de dispositivos móviles, es que los propios usuarios comprendan que su privacidad es valiosa y tomen medidas para protegerla. Las medidas de protección que ofrecen los dispositivos móviles actuales, suelen venir definidas tanto por los desarrolladores del sistema operativo del dispositivo como por los fabricantes, ofreciendo continuamente mejoras en la seguridad para tratar de impedir el acceso a los datos sensibles de los usuarios. Adicionalmente, existe una amplia comunidad dedicada a desarrollar sistemas operativos alternativos para los dispositivos móviles, los cuales implementan medidas específicas de seguridad en muchos casos diferentes a las de los propios fabricantes.

El modelo planteado en este trabajo, podría ser utilizado por un tercero, ya sea desarrollador o fabricante para añadir una capa adicional de seguridad ante los ataques físicos de desbloqueo del patrón del dispositivo, ya que tal y como se ha expuesto en el apartado de resultados, se consigue una alta protección ante ataques por parte de un tercero conocedor de la secuencia del patrón de desbloqueo o capaz de obtenerlo mediante técnicas de visualización del polvo/grasa/etc. de la pantalla.

Los resultados obtenidos tras el desarrollo están en línea con lo esperado, si bien hubo que ajustar el umbral de tolerancia del patrón rítmico durante la fase de pruebas para obtener una mejor experiencia de usuario. Respecto al desarrollo tanto de la aplicación como de la librería no hubo contratiempos.

Se han alcanzado todos los objetivos definidos, y se han satisfecho todos los requisitos definidos durante la fase de análisis. Durante la fase de definición y análisis se realizó una evaluación muy detallada que permitió que no existieran desviaciones al respecto de los objetivos y requisitos de este trabajo.

La metodología seguida para este proyecto ha facilitado el trabajo. El hecho de utilizar una metodología ágil como Kanban, ha permitido eliminar gran parte de la burocracia de la gestión de un proyecto manteniendo el control sobre la planificación de las tareas y aportando una gran flexibilidad. El uso de la herramienta GIT ha sido de gran ayuda para el control de versiones tanto del código como de la documentación. La división del trabajo en cuatro grandes hitos correspondientes a las entregas intermedias del trabajo, han permitido mantener un horizonte claro respecto al desarrollo del propio trabajo. La planificación se ha respetado en términos generales, variando la carga

distribuida entre implementación del proyecto Android y la documentación entre la tercera entrega y la entrega final. La variación se enfocó de la siguiente manera:

- Durante la tercera entrega se destinó más tiempo del planificado a la implementación tanto de la aplicación como de la librería Android, y ese tiempo se obtuvo de las tareas de documentación.

- Por el contrario, durante la entrega final se destinó menos tiempo del planificado a la parte de la implementación de la aplicación y la librería, ya que se había incrementado la carga en la entrega anterior, por lo que se pudo destinar más tiempo a la documentación y a las pruebas, compensando así la entrega anterior.

Un aspecto muy positivo de la fase de definición de la interfaz gráfica, fue el hecho de utilizar componentes de Android para realizar dicha definición. De este modo se redujo incertidumbre entre el aspecto final de la aplicación y la definición, ya que es exactamente igual. Además, este enfoque permite trabajar en la fase de implementación de una forma mucho más sencilla, gracias a que el diseño ya dispone de una estructura de componentes de programación.

Respecto al algoritmo de validación del patrón rítmico, se optó por crear un algoritmo basado en el tiempo que pasa entre la pulsación de dos celdas manteniendo el umbral de tolerancia. Si bien el algoritmo podría haberse realizado de otras múltiples formas, como por ejemplo derivando líneas de tendencia respecto al patrón rítmico del usuario, la implementación realizada en este trabajo es eficaz resolviendo los objetivos planteados.

Durante el desarrollo de este trabajo no se introdujeron cambios significativos para garantizar el éxito del proyecto, más allá de la distribución de horas ya comentadas entre la tercera y la entrega final y el ajuste del umbral de tolerancia del algoritmo de validación rítmica.

5.2 Trabajos futuros

Este trabajo se ha enfocado en la validación de los patrones de bloqueo en dispositivos Android. La integración del modelo de validación rítmica planteado podría ser integrado en dispositivos Android por diferentes vías:

- Por medio del fabricante del dispositivo:
    - A través de la integración en la capa de personalización software del fabricante.

- Por medio de un desarrollador:
    - A través de la implementación de un software específico de personalización que se pueda descargar e incluir en el dispositivo del fabricante.

    - En un sistema operativo alternativo implementado para un dispositivo concreto.

- Por medio del desarrollador de AOSP

    - Integrado en la base para el resto de fabricantes

Existen varios trabajos futuros que podrían llevarse a cabo en base a la validación rítmica del patrón de desbloqueo. Entre ellos se encuentran:

- Realizar mejoras o nuevos planteamientos de algoritmos de validación rítmica

- Utilizar capas adicionales de seguridad como podría ser el movimiento del dispositivo móvil a la hora de introducir el patrón

- Crear patrones rítmicos sin necesidad de utilizar una matriz, por ejemplo realizando determinadas pulsaciones rítmicas sobre el dispositivo

- Integrar la validación rítmica con otros sistemas de autenticación como biometría

- Estudio de viabilidad de la portabilidad de la solución a otras plataformas como iOS de Apple.

Para la implementación del almacenamiento de la base de datos de este trabajo se ha utilizado el espacio de almacenamiento específico habilitado por la plataforma para la aplicación. Una posible mejora sería la de almacenar el patrón en un lugar seguro del dispositivo, como un chip criptográfico presente en muchos de los dispositivos móviles actuales. En caso de integración por parte de un tercero, sería necesario alinear la integración con el patrón acutalmente disponible en los terminales y las capas de seguridad de las que dispone.

En este proyecto no se ha contemplado la tarea de integración con el patrón actual de desbloqueo de un dispositivo móvil, este punto podría ser interesante en un futuro para explorar posibles despligues de la tecnología.

6. Glosario

Activity – Componente gráfico de Android que representa una pantalla
Adapter – Componente de Android destinado a proveer modelos de datos
Android – Sistema operativo para dispositivos móviles
Android Studio – Herramienta desarrollada por Google para desarrollo de aplicaciones Android
ANR – “Application Not Responding” error en Android
AOSP – Android Open Source Project
Apple – Fabricante estadounidense de dispositivos electrónicos
Fragment – Componente gráfico de Android destinado a representar fragmentos de vistas
GIT – Herramienta para el control de versiones de código
Gradle – Herramienta de automatización de construcciones
GridLayout – Sistema de representación gráfica basada en celdas en Android
iOS – Sistema operativo utilizado por Apple en sus smartphones
JAVA – Lenguaje de programación orientado a objetos
Kanban – Metodología ágil de desarrollo basada en tarjetas
Kotlin – Lenguaje de programación de tipado estático
Layout – Definición de parte de la estructura gráfica de una aplicación Android
Macbook – Equipo portátil desarrollado por la compañía estadounidense Apple
Microsoft Excel – Herramienta de hojas de cálculo
Microsoft Office – Set de herramientas de ofimática
Microsoft Word – Herramienta de procesamiento de textos
Sandbox – Espacio de almacenamiento destinado a una aplicación concreta en un dispositivo Android
SharedPreferences – Sistema de almacenamiento de información basado en ficheros xml en la sandbox de una aplicación Android
Smudge Attack – Ataque producido en base a identificación de manchas en pantallas

7. Bibliografía

[1] - Aviv, A. J. (2010). Smudge Attacks on Smartphone Touch Screens | USENIX. Recuperado 03 de octubre de 2022, de https://www.usenix.org/conference/woot10/smudge-attacks-smartphone-touch-screens
[2] - GitHub - itsxtt/pattern-lock: Awesome pattern lock view for android written in kotlin. #手□密□. (s. f.). GitHub. Recuperado 03 de octubre de 2022, de https://github.com/itsxtt/pattern-lock
[3] - Free Online Gantt Chart Software. (s. f.). Recuperado 04 de octubre de 2022, de https://www.onlinegantt.com/
[4] - TapTap Lock: Double-tap to loc - Apps on Google Play. (s. f.). Recuperado 05 de octubre de 2022, de https://play.google.com/store/apps/details?id=com.davidcstudio.mylock
[5] - Just Unboxing. (2014, 17 junio). Como funciona el Knock Code en el LG G3. YouTube. Recuperado 05 de octubre de 2022, de https://www.youtube.com/watch?v=X-FrAB218B8
[6] - Application Sandbox |. (s. f.). Android Open Source Project. Recuperado 29 de octubre de 2022, de https://source.android.com/docs/security/app-sandbox
[7] - SharedPreferences |. (s. f.). Android Developers. Recuperado 29 de octubre de 2022, de https://developer.android.com/reference/android/content/SharedPreferences
[8] - Activity |. (s. f.). Android Developers. Recuperado 30 de octubre de 2022, de https://developer.android.com/reference/android/app/Activity
[9] - Kotlin Programming Language. (s. f.). Kotlin. Recuperado 30 de octubre de 2022, de https://kotlinlang.org/
[10] - GridLayout |. (s. f.). Android Developers. Recuperado 11 de diciembre de 2022, de https://developer.android.com/reference/android/widget/GridLayout
[11] - Fragments |. (s. f.). Android Developers. Recuperado 12 de diciembre de 2022, de https://developer.android.com/guide/fragments
[12] - Adapter |. (s. f.). Android Developers. Recuperado 12 de diciembre de 2022, de https://developer.android.com/reference/android/widget/Adapter
[13] - Gradle Build Tool. (s. f.). Gradle. Recuperado 04 de diciembre de 2022 de https://gradle.org

8. Anexos

Este trabajo no contiene anexos.