TRABAJO FIN DE GRADO

Gamificación en el aprendizaje de Programación en C a través de una aplicación Android

TRABAJO FIN DE GRADO PARA LA OBTENCIÓN DEL TÍTULO DE **GRADUADO EN INGENIERÍA QUÏMICA**

SEPTIEMBRE 2025

Óscar Alejandro Vílchez Cano

DIRECTOR DEL TRABAJO FIN DE GRADO:
Ascensión López Vargas

INDUSTRIALES
ETSII | UPM

POLITÉCNICA

"Ser ingeniero es como ser un niño con juguetes, simplemente los juguetes son más caros y las reglas de uso más estrictas (aunque es igual de divertido)."

Anónimo

AGRADECIMIENTOS

Esta es probablemente la página que más me va a costar escribir.

Sin duda las personas más importantes han sido mis padres, Óscar y Antonia, quienes apostaron por mí para que pudiera estudiar en Madrid, confiando siempre en mí durante la carrera a pesar de que muchas veces no lo puse fácil con momentos negativos. Fue en esos momentos en los que, sin esa educación que me forjaron en la que adquirí una fortaleza mental enorme, hubiera existido el riesgo de que yo no acabara como ingeniero.

Agradecer también a mi hermana Marina, la MHM, que sin ella yo no sería nada, ella es siempre mi gran apoyo, mi gran motor y es la persona que más fe me tiene.

Sin duda también han sido un gran apoyo a aquellos que llamo yo mi arma secreta, que son mis amigos de siempre, con los cuales uno nunca puede estar triste, porque tenerlos a ellos como amigos ya es un éxito. Dubi, Joaquín, Miguel, Tárraga, Barreda, Andrés, Gil… Esto va por vosotros.

En la universidad también he hecho una serie de amigos que ya se convirtieron en gente de absoluta confianza y, son personas sin las cuales esta carrera hubiera sido una auténtica tortura.

Al acabar el primer año empecé a trabajar los fines de semana en un McDonald's. Al final he acabado trabajando en dos restaurantes casi 3 años y, sinceramente, me he enamorado de un trabajo que podría parecer uno de esos llamados 'trabajo basura', pero que su gente los convierte en algo mágico. Aquí podría mencionar a tanta gente que me parecería injusto mencionar a alguien sin mencionar al resto (mínima mención a la señorita que me ayudó con la estética de la aplicación), he aprendido tantas cosas, me he sentido tan querido… Gracias y ahora que estoy de vuelta… La que se viene :)

Y, por último, las personas que han hecho posible que esté realizando un TFG soñado, de una temática con la que tuve un flechazo desde la primera clase, gracias Asu, gracias Javi, gracias por confiar en que podía sacar esto adelante.

RESUMEN EJECUTIVO

En este Trabajo Fin de Grado se presenta el diseño y desarrollo de una aplicación móvil para el sistema operativo Android orientada al aprendizaje de la programación en lenguaje C mediante técnicas de gamificación. La aplicación se concibe como un recurso didáctico complementario, busca facilitar la comprensión de los fundamentos de programación, así como ayudar a aumentar la creatividad, siendo esta entendida en términos de programación. El público objetivo de la aplicación son aquellas personas sin conocimientos de programación que buscan aprender a programar en C, estando principalmente enfocada en los alumnos de la ETSII UPM.

La idea surgió a partir de un hecho muy simple, a la gran mayoría de alumnos de primer año que se introducen en la programación sufren para aprender, al ser un reto totalmente distinto a los que se han enfrentado con anterioridad. A través de un enfoque interactivo y progresivo en el que el usuario empieza desde el elemento más básico hasta otros mucho más complejos, la aplicación divide los contenidos en bloques temáticos que contienen dentro de sí varios subapartados, cada uno con explicaciones breves, ejemplos prácticos y algunos minijuegos que permiten poner a prueba los conocimientos adquiridos. Además, se ha implementado una sección de desafíos independientes, con problemas orientados al razonamiento lógico y la consolidación de habilidades adquiridas, siendo estos programas distintos, cuyo objetivo es que el usuario entienda que la programación es aplicable a una gran cantidad de ámbitos de la vida.

Durante la fase inicial, se llevó a cabo un pequeño análisis de las herramientas ya existentes orientadas al aprendizaje de la programación, identificando sus carencias, así como la escasa integración de la gamificación o incluso la falta de libertad de navegación entre niveles. Frente a ello, se definieron una serie de requisitos específicos que guiaron el diseño de la aplicación:

a) Navegación libre: el usuario puede acceder a cualquier pantalla sin necesidad de completar los anteriores, adaptándose así a distintos ritmos de aprendizaje, pudiendo en cualquier momento acceder a cualquier subapartado.

b) Sistema de progreso visual: cada bloque cuenta con un botón para marcar el contenido como “aprendido”, lo cual se refleja en un icono de la pantalla inicial de la aplicación, ofreciendo un seguimiento visual y motivador al usuario.

c) Desafíos opcionales: se incluye una sección de retos independientes del modo de aprendizaje progresivo, pensados para reforzar el pensamiento computacional y el dominio

del lenguaje C.

d) Interfaz intuitiva y accesible: diseñada para facilitar la navegación en dispositivos móviles, siguiendo las guías de estilo de Android y principios de usabilidad.

e) Compatibilidad: se ha comprobado que la aplicación es ejecutable en un amplio rango de versiones de Android, maximizando así su alcance potencial, intentando que esté disponible para el mayor número de usuarios posible.

La aplicación ha sido desarrollada en *Android Studio*, utilizando *Java* y *XML* para el desarrollo del entorno y C para los contenidos de programación. El sistema de aprendizaje se basa en actividades prácticas que el usuario completa manualmente en su dispositivo en el que él suele programar el código que puede observar en la aplicación, centrando el foco por tanto en la comprensión de los programas y no en la automatización de correcciones, es decir, se busca un formar al usuario con una base de conocimientos antes de que él se introduzca al mundo de aprender a partir de prueba-error.

Como parte del proceso final, se han realizado pruebas funcionales y de usabilidad con usuarios reales, usuarios con conocimientos en programación, así como otros usuarios que no poseen ningún conocimiento (público objetivo), analizando sus comentarios y detectando posibles mejoras futuras. Estas pruebas han permitido validar la utilidad de la aplicación como herramienta de refuerzo en el aprendizaje inicial de programación.

Finalmente, en las conclusiones se exponen los principales logros obtenidos, se plantean las mejoras a futuro basadas en los comentarios de los usuarios experimentales, además de reflexionar sobre el valor que aporta la gamificación como recurso pedagógico en carreras técnicas.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

1. Introducción

La introducción de este trabajo busca contextualizar el proyecto y presentar los elementos clave que lo sostienen. En ella se expone el marco general en el que surge la propuesta, la justificación que da sentido a su desarrollo y los objetivos que guiarán cada una de sus fases. De este modo, se establece una base sólida que permite comprender tanto la relevancia académica como la pertinencia práctica de *Sí C puede*.

1.1. Marco

Aprender nunca ha sido un proceso neutral. Desde tiempos remotos, el ser humano ha buscado maneras de transmitir lo que sabe, de forma cada vez más eficaz, accesible y adaptada a su contexto. Hoy, ese contexto está marcado por lo digital. Las pantallas, los móviles y las aplicaciones son herramientas que han irrumpido con fuerza en nuestro día a día, convirtiéndose en entornos habituales donde trabajamos, socializamos y, cada vez más, aprendemos.

Pero esta transformación no se ha visto muchas veces reflejada de la misma manera en las clases. La enseñanza de la programación, por ejemplo, sigue planteando grandes retos, sobre todo cuando se trata de enseñar desde cero a un alumno, ya que se requieren una lógica rigurosa y un alto nivel de atención al detalle totalmente novedosos para aquel que busca aprender. Para muchos estudiantes, enfrentarse a este tipo de lenguajes como puede ser el C, puede resultar un reto muy difícil de superar, sobre todo en esos casos mencionados anteriormente en los que no poseen experiencia previa con el pensamiento computacional.

Ante este problema, cada vez se exploran más métodos alternativos e innovadores que no solo buscan enseñar, sino también motivar a partir de disfrutar durante el aprendizaje. En los últimos años, uno de los enfoques que más fuerza ha cobrado es la gamificación: introducir elementos propios del juego en contextos educativos, con el objetivo de hacer más atractivo y dinámico el aprendizaje. No se trata de “jugar por jugar”, sino de aprovechar mecánicas como la progresión por niveles, la retroalimentación inmediata o los desafíos voluntarios para que el alumno se implique más activamente en su propio proceso formativo [1].

Este tipo de enfoques encaja especialmente bien con las plataformas móviles. Los dispositivos *Android*, por ejemplo, ofrecen un entorno ideal para desarrollar aplicaciones

educativas accesibles y personalizables. Gracias a su amplia penetración y versatilidad, permiten crear experiencias de aprendizaje adaptadas a distintos perfiles de estudiante, rompiendo con la idea tradicional de que aprender a programar solo puede hacerse frente a un ordenador y con un manual en la mano [2].

En este Trabajo de Fin de Grado se recoge el desarrollo de una aplicación que une precisamente estos dos elementos: el aprendizaje de programación en C y la gamificación como estrategia pedagógica. Se ha diseñado pensando en estudiantes que se inician en el lenguaje, especialmente en titulaciones como Ingeniería Química, donde el código no es el centro del plan de estudios, pero cada vez se valora más como herramienta transversal. A través de un sistema de bloques temáticos, ejercicios prácticos y retos independientes, esta aplicación busca no solo enseñar, sino también generar una experiencia motivadora y útil, desde el propio móvil del estudiante.

1.2. Motivación

Aprender a programar puede ser una experiencia reveladora… o un auténtico quebradero de cabeza. Para muchos estudiantes que se enfrentan por primera vez al lenguaje C, la sensación es más bien lo segundo. No es para menos: hablamos de un lenguaje que exige precisión, lógica y una comprensión profunda de cómo funciona un ordenador por dentro. A menudo, lo que debería ser una puerta de entrada a nuevas posibilidades, se convierte en una barrera que frustra, desanima o simplemente genera rechazo.

Y, sin embargo, programar es cada vez más necesario. En titulaciones como la de Ingeniería Química, donde no siempre se pone el foco en la informática, tener nociones básicas de programación puede marcar la diferencia. Desde el análisis de datos experimentales hasta la simulación de procesos, saber escribir unas líneas de código abre caminos que hace unos años eran impensables para un perfil no informático [3].

Aquí es donde aparece la motivación de este trabajo. Si el problema está en cómo se enseña —y no tanto en lo que se enseña—, ¿por qué no plantear una alternativa que se acerque más a la realidad de los estudiantes? Una herramienta que no dependa del papel ni de largas sesiones frente al ordenador, sino que funcione en algo tan cotidiano como el teléfono móvil, que permita aprender a tu ritmo, sin presión, que te de información y que tú la digieras a tú manera, que convierta el error en parte del juego y no en un motivo de frustración [4].

De ahí nace la idea de esta aplicación: una aplicación para *Android* que enseña a

programar en C mediante pequeños bloques temáticos y retos interactivos, usando dinámicas propias del juego para mantener la motivación. Nada de conexión permanente, nada de interfaces recargadas. Solo una estructura clara, accesible, pensada desde la experiencia de quienes están aprendiendo. Y sí, también pensada para poder usarse sin conexión, porque no todo el mundo tiene datos móviles o buena cobertura todo el tiempo.

Y, ¿cuál es el nombre de la aplicación? La aplicación se llama 'Sí C Puede', el porqué está muy claro, es un juego de palabras en el que se busca motivar al usuario, haciéndole ver en todo momento que sí es posible aprender a programar. En la *figura 1* se puede observar el logo de la aplicación, que tiene ya integrados los colores verde y amarillo que predominan en la aplicación.

Figura 1 Logo App - Sí C Puede

Además, hay un factor más. Digitalizar estos contenidos no es solo práctico, también es más sostenible. Al evitar el uso de papel para ejercicios básicos o prácticas repetitivas, se contribuye, aunque sea en pequeño, a una forma de aprendizaje más responsable con el entorno.

Este proyecto, en el fondo, es una respuesta a una sensación compartida: que aprender C puede ser mucho más llevadero si se adapta a cómo pensamos, cómo aprendemos y cómo vivimos hoy en día.

1.3. Objetivos

El desarrollo de la aplicación *Sí C puede* parte de una idea muy simple: demostrar que aprender a programar en C puede ser más accesible, más intuitivo y, por qué no, incluso más entretenido. El objetivo principal de este trabajo está muy claro, diseñar y programar una aplicación Android completamente funcional que sirva como herramienta educativa para

aquellos que se están iniciando en este lenguaje.

Para que sea posible, se plantean varios objetivos específicos que han ido guiando cada una de las fases del proyecto:

Diseñar una experiencia de usuario clara y fluida, con una navegación sencilla entre bloques de contenido, ejemplos y desafíos. Se ha puesto especial atención en que cualquier estudiante, incluso sin familiaridad previa con *Android*, pueda utilizar la app sin complicaciones. La interfaz es limpia, directa y acompañada de ayudas contextuales que explican cada función en los primeros usos.

Garantizar la compatibilidad con una amplia variedad de dispositivos *Android*, desde las versiones más recientes hasta modelos más antiguos. Esto ha requerido realizar pruebas continuas en distintos terminales, anticipando así los posibles errores y ajustando el comportamiento de la aplicación para asegurar que se mantenga estable y funcional para el usuario en todos los casos.

Ofrecer un entorno de práctica estructurado y motivador, mediante la implementación de un sistema de bloques temáticos y una sección de desafíos. La aplicación no pretende reemplazar al aula, sino complementar el aprendizaje con un enfoque más flexible y adaptado al ritmo del estudiante.

Permitir el seguimiento del progreso, incorporando herramientas visuales que refuercen la sensación de avance. Cada bloque puede marcarse como “aprendido”, y esta acción se refleja de forma inmediata en una barra de progreso general, una forma simple pero eficaz de mantener al usuario motivado.

En conjunto, estos objetivos buscan algo más que un simple desarrollo técnico. El propósito es construir una herramienta educativa que se sienta cercana, útil y bien pensada; una aplicación que cumpla con lo que su nombre promete: que aprender C, realmente, sí se puede.

1.4. Estructura de la memoria

Este informe se divide en nueve partes, estructuradas para mostrar el camino completo del proyecto *Sí C puede*, desde su idea teórica hasta su camino a futuro. Cada parte está hecha para enseñar no sólo lo que se hizo, sino las razones y el modo detrás de cada decisión que se tomó.

El Capítulo 2 muestra el estado del arte. Arranca con una mirada general al sistema *Android* y

cómo ha ido cambiando, mostrando su forma, versiones y modo de creación en Android Studio. También se tocan las bases educativas que apoyan la idea, incluyendo el papel de los juegos en el aprendizaje, el análisis de aquello que ya existe y la elección del lugar ideal para dar a conocer estas aplicaciones.

El Capítulo 3 explica el funcionamiento de la aplicación. Se presenta la estructura de la aplicación, y las decisiones que se fueron tomando sobre la apariencia y la usabilidad. Aquí se muestra todo aquello a lo que tiene acceso el usuario, así como cuál es el plan para que este aprenda a programar.

El Capítulo 4 se centra en cómo se creó la aplicación, centrándose en el código empleado en *Android Studio*. Aquí se detalla la vista de la aplicación, cómo se adaptan las versiones, la forma del código y el modelo de clases. También se explican otros factores como el montaje, cómo crece o cómo se adapta a varios dispositivos y vistas, siempre desde un modo práctico que ayuda a ponerlo en *Android*.

El Capítulo 5 examina los resultados obtenidos. Se juzga si se han cumplido las metas, se muestran los resultados técnicos y se piensa sobre el impacto educativo de la aplicación. Aparte, se estudia el recorrido del usuario y se enfrenta la solución diseñada con otras opciones a mano en el mercado.

El Capítulo 6 compendia una pequeña conclusión en la que se definen las limitaciones actuales y se define el rango educativo de la aplicación.

El Capítulo 7 está dedicado a los caminos futuros del desarrollo. Se plantean probables optimizaciones que tenga la capacidad de mejorar la experiencia del usuario.

El Capítulo 8 muestra la organización y el cálculo de gastos estimado. Se introduce un diagrama de Gantt, introduciendo en el apartado de gastos el cómputo de las horas empleadas y el material utilizado.

Para finalizar, el Capítulo 9 toca varios apartados, siendo estos los social y educativo, el económico y el medioambiental, finalizando con un análisis de los Objetivos de Desarrollo Sostenible (ODS).

2. Estado del Arte

El estado del arte recoge los antecedentes más relevantes que sirven de base para el desarrollo de *Sí C puede*. Este apartado revisa el contexto tecnológico y educativo en el que se enmarca la aplicación, atendiendo especialmente al papel de los sistemas operativos móviles y a las herramientas que facilitan la enseñanza de la programación. En primer lugar, se analiza *Android*, como plataforma de referencia en la que se ha construido el proyecto.

2.1. El sistema operativo Android

Android se ha consolidado como el sistema operativo móvil más extendido a nivel mundial, presente en millones de dispositivos de diferentes gamas y fabricantes. Su carácter abierto y su ecosistema de aplicaciones lo han convertido en un estándar tecnológico clave para la innovación educativa y el desarrollo de proyectos académicos como *Sí C puede*. Para entender mejor su impacto actual, es necesario repasar los orígenes de la compañía *Android Inc.* y los primeros pasos que llevaron a su integración en *Google*.

2.1.1 Inicios de la compañía Android Inc.

Cuando pensamos en *Android* hoy, lo primero que nos viene a la cabeza es un sistema operativo que está en infinidad de dispositivos, ya sean teléfonos, tabletas, televisores, relojes... y lo usamos casi sin prácticamente darnos cuenta. Pero lo curioso es que *Android* no nació con la ambición de dominar el mundo móvil, de hecho, ni siquiera nació para teléfonos.

Todo empezó en 2003, en un pequeño entorno de desarrollo en Palo Alto, California, donde cuatro ingenieros —Andy Rubin, Rich Miner, Nick Sears y Chris White— decidieron fundar una *startup* llamada Android Inc. con una idea: querían crear un sistema operativo para cámaras digitales que tanto triunfaban en aquellos años. Querían que estas pudieran adaptarse automáticamente a las preferencias del usuario y a su ubicación. Un concepto interesante, pero que pronto se toparía con una realidad: el mercado era demasiado pequeño porque para sostener algo tan ambicioso se requería de un mercado aún mayor [5].

Así que dieron un giro. Decidieron aplicar ese mismo enfoque, pero al mundo de los teléfonos móviles. En aquella época, los dominaban *Symbian* y *Windows Mobile*, y los smartphones todavía eran algo tosco, con interfaces poco intuitivas y muchas limitaciones. Android apostó entonces por construir un sistema basado en Linux, que fuera más flexible y abierto, algo que permitiera a los fabricantes adaptarlo según sus necesidades.

Fue justo en ese momento, en 2005, cuando *Google* se fijó en ellos [6]. Vio potencial en lo que estaban haciendo y no tardó en adquirir la empresa. La idea original se mantuvo, pero fue tomando forma dentro de un ecosistema más grande. El primer prototipo que desarrollaron funcionaba en un dispositivo con teclado físico, muy al estilo *Blackberry*. Pero entonces ocurrió algo que lo cambió todo: en 2007, *Apple*, liderado por Steve Jobs, presentó el *iPhone*. Una pantalla táctil sin botones físicos y una nueva forma de interactuar con el teléfono. Fue un gran golpe sobre la mesa.

En respuesta, Google y su equipo reformularon su estrategia. Ese mismo año se formó la *Open Handset Alliance*, una alianza de grandes nombres como *HTC, Samsung, Sony, Qualcomm o T-Mobi*le. Su apuesta: un sistema operativo abierto, gratuito y adaptable a cualquier fabricante. Mientras *Apple* decidía apostar por el control total, Android ofrecía esa libertad que muchos anhelan.

En octubre de 2008, salió al mercado el primer dispositivo con *Android*: el *HTC Dream*, también conocido como G1. Tenía teclado físico, una interfaz bastante básica y no levantó grandes pasiones... pero fue el comienzo de algo enorme. Ese modelo abrió el camino a un sistema que, años después, acabaría triunfando llegando a los bolsillos de miles de millones de personas.

2.1.2 Características del sistema operativo Android

Hoy en día, *Android* sobresale entre los sistemas móviles por ser tan abierto, adaptable y estar en muchos aparatos distintos. Aunque lo usamos a diario de forma fácil, lo que pasa por dentro es muy técnico y siempre está cambiando.

Desde el principio, *Android* se hizo con el corazón de *Linux*, dándole fuerza y la opción de estar en varias plataformas. Esto, junto con licencias libres como Apache 2.0, deja que los creadores vean, cambien y usen el código como quieran [7]. Esta libertad ha creado una comunidad activa y muchos aparatos diferentes.

Antes, *Android* cambiaba mucho su forma de funcionar. Al principio usaba *Dalvik*, un programa que convertía el código de Java en instrucciones nativas cuando se usaba la aplicación. Esto daba flexibilidad, pero hacía que los aparatos lentos no funcionaran tan bien [8].

Desde la versión 4.4, se añadió ART *(Android Runtime),* que lo cambió todo. Con ART, el código de Java se convierte antes de instalar la app, así se carga más rápido y se usa mejor el aparato [9]. Esto hizo que usar el móvil fuera mejor y que todo el sistema fuera más eficiente.

Android es más que solo Java, es como una mezcla de varios lenguajes. Tiene muchas bibliotecas escritas en C y C++, importantes para trabajos del sistema, también en *SQLite*, que organiza las bases de datos del teléfono; *OpenGL ES*, que dibuja gráficos en 2D y 3D; o *WebKit*, el cerebro del navegador. Estas bibliotecas ayudan a las aplicaciones a realizar contenidos muy visuales sin la necesidad de tener una gran cantidad de código.

Para que realizar aplicaciones sea sencillo, Android tiene herramientas especiales que cambian con el tiempo. Antes, se usaba Eclipse con el plugin ADT. Pero ahora, *Android Studio* es el rey, es una plataforma basada en *IntelliJ IDEA* con todo lo necesario para escribir código, probar aplicaciones en un teléfono emulado, arreglar errores y ver qué tan bien funciona la aplicación realizada [10].

Con esta mezcla de libertad, poder y buenas herramientas, *Android* es perfecto para expertos y principiantes en la creación de aplicaciones. Su forma de ser permite hacer cosas muy diferentes, lo que es genial para proyectos como este.

2.1.3 Arquitectura de Android

Android no es un sistema operativo uniforme, sino que está dividido por ‘pisos’, como bien se puede observar en la *figura 2*, cada cual con su rol dentro del sistema. Es esto lo que permite que cada nivel trabaje por su lado, pero a la vez en equipo, funcionando para que todo sea estable y a la vez puedas sea posible el desarrollo.

Lo que más se ve son las aplicaciones. Es lo que aquello que uno usa sin pensar, como el correo, los contactos, el navegador o el calendario. Estas aplicaciones están escritas, la gran mayoría, en Java (o *Kotlin*, si es moderno) y son la cara más amable entre el sistema y tú, el usuario final.

Debajo está el *framework* de aplicaciones, un sitio que proporciona a los creadores de aplicaciones una gran variedad de herramientas y trucos para hacerlas o cambiarlas. Lo bueno es que puedes usar las herramientas una y otra vez: notificaciones, ubicación o ventanas ya están hechas, así que no hay que empezar desde el principio.

Luego hay una gran cantidad de librerías nativas, hechas en C y C++, que hacen que el sistema funcione. También están las bases de datos (como *SQLite*), los gráficos en 2D y 3D (con *OpenGL ES*), la web (con *WebKit*) y la multimedia. Estas librerías se mezclan con el *framework* para que las aplicaciones sean utilizadas sin tocar el código nativo.

El *Android Runtime* (ART) por su parte es el encargado de inicializar estas aplicaciones. Esta parte sustituyó a la antigua máquina Dalvik. Lo hace transformando el

código Java a un formato.dex (*Dalvik Executable*), que luego se convierte en órdenes que el procesador entiende. Cada aplicación se inicializa aparte, en su propio espacio ART, lo que ayuda a usar mejor la memoria y a estar más seguro.

En lo más profundo de todo está el núcleo Linux, que es como un puente entre las partes físicas del aparato y las demás capas. Se encarga de cuidar la memoria, controlar lo que hacen las aplicaciones, la seguridad, la conexión a internet y hablar con las partes físicas. Aunque la persona que usa el teléfono no toca esta parte, sin ella nada más podría funcionar.

Esta forma de organizar todo en capas ayuda a que Android funcione bien en aparatos muy diferentes, y deja que los programadores de aplicaciones se concentren en crear novedades sin tener que pensar en todo aquello de lo que se encarga el sistema.

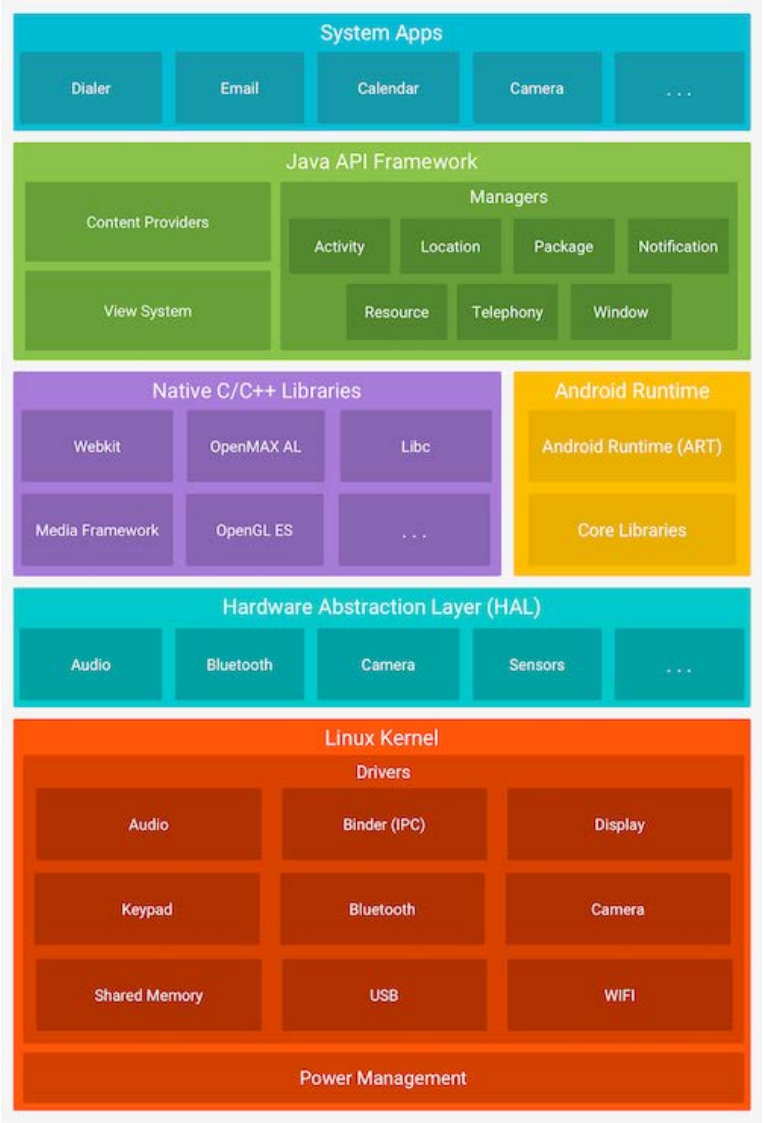

Figura 2 Arquitectura interna del sistema operativo Android – (Cyber0asis

2.1.4 Versiones de Android

Un factor clave del poderío de Android como sistema operativo ha sido su promesa de seguir funcionando con las versiones anteriores. No como en Apple, Android siempre ha tratado de que las versiones nuevas no colapsen con las antiguas. Esto quiere decir que cada plataforma nueva guarda las funciones antiguas, aunque algunas ya no se usen tanto con el tiempo. Así, las aplicaciones hechas para versiones antiguas siguen funcionando sin necesidad de cambios, lo que ayuda a que todo esté más estable.

Cada versión de Android se conoce de tres formas: con su número de versión, su nivel de API y su nombre característico. Estas tres cosas existen juntas para hablar de la misma plataforma, pero cada una tiene un uso diferente al hacer aplicaciones y al tratar elementos técnicos.

La versión se refiere al número que muestra un cambio grande en el sistema. Cada vez que Android lanza una versión nueva, le añade nuevas funciones o realiza cambios internos importantes.

El nivel de API (Interfaz de Programación de Aplicaciones) dice de forma única qué funciones se pueden usar en esa versión del sistema. Este nivel numérico le indica al sistema qué puede o no hacer una aplicación, según la versión para la que se hizo. Incluye clases, permisos, atributos XML y atributos que los programadores pueden utilizar al crear sus aplicaciones.

El nombre de venta, en cambio, es como todos llaman a cada edición. Por años, Android usó nombres de dulces en orden de letras para sus versiones, como KitKat (4.4), Lollipop (5.0), Marshmallow (6.0) u Oreo (8.0 y 8.1). Aunque ya no se hace desde Android 10, aún se usa dentro en secreto [11].

Aunque *Android* es el sistema más usado en el mundo, la llegada de las nuevas versiones a los usuarios no es al instante. Esto pasa por la gran variedad de marcas y modelos que usan Android, lo que quiere decir que las mejoras dependen de cada marca y aparato. Por eso, las versiones antiguas siguen en muchos móviles por años, aunque haya versiones más nuevas.

Según los datos oficiales de reparto de versiones, en mayo de 2024, se observa que *Android* aún tiene una división grande. Por ejemplo, *Android* 10 (API 29) y *Android* 11 (API 30) siguen en muchos aparatos, con partes del 81,2% y 67,6% (esto indica el porcentaje de uso de esa versión o una superior) cada uno. Hasta versiones como *Android* 9 (Pie) tienen una

parte casi del 90% (89,6%) si se ven en el total. Pero, las versiones nuevas —como *Android* 13 (API 33) y *Android* 14 (API 34), también *Android* "U"— aún no se usan mucho, con partes del 33,9% y 13,0%, cada uno [12].

La *figura 3* muestra este reparto, con varias de las mejoras y cambios que trae *Android* U; se ven, por ejemplo, mejoras en lo privado, manejo de trabajos en segundo plano y nuevos permisos para la seguridad de las personas que lo usan.

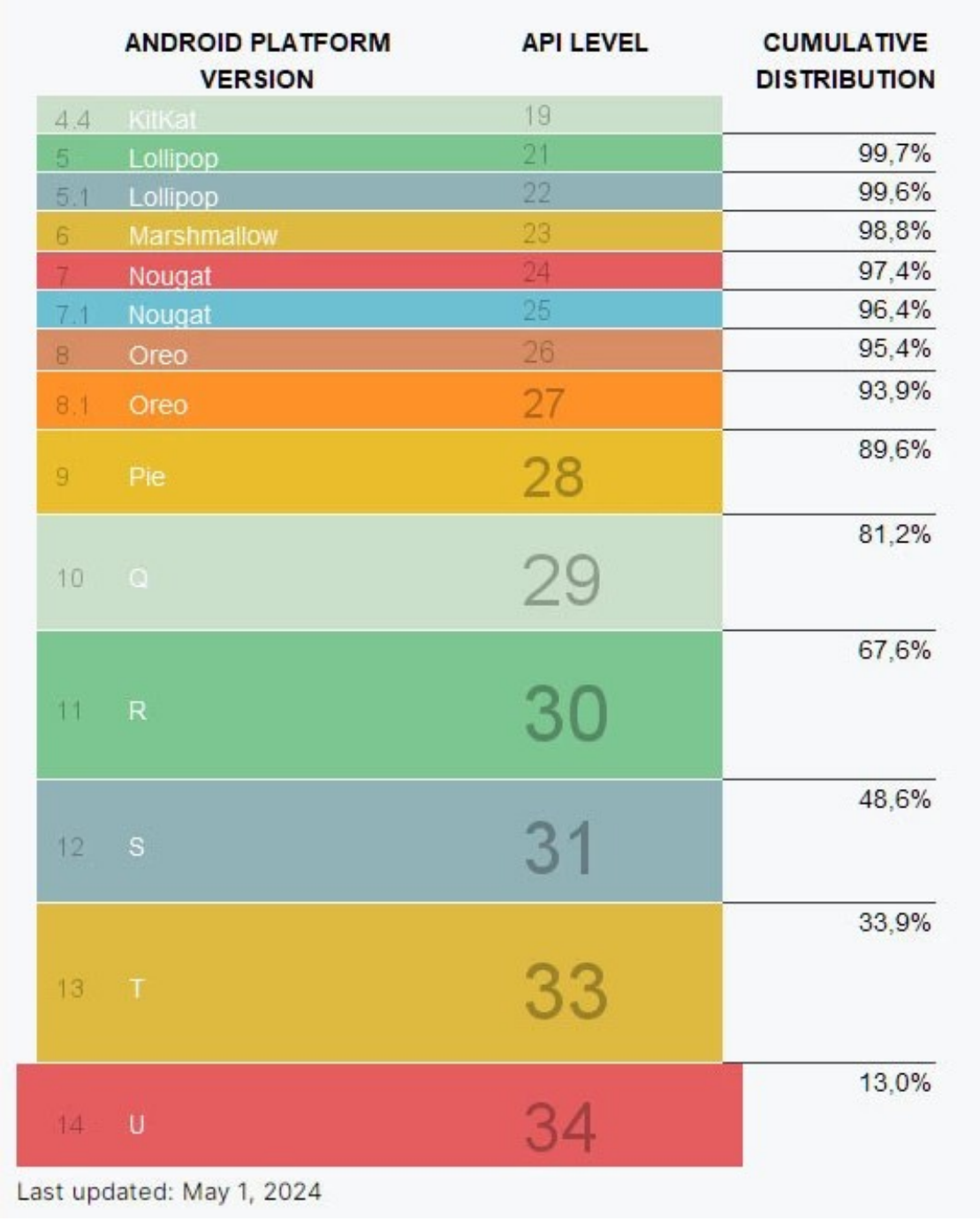

Figura 3 Distribución de versiones Android en Mayo 2024 – (Xataka Android)

Estos datos no solo muestran cómo el sistema va cambiando, sino también supone un reto clave para los creadores: hacer aplicaciones que funciones bien en muchos dispositivos y versiones, sin tener que dejar de lado a las funciones más nuevas. Aspectos fundamentales de una aplicación

2.2. Aspectos fundamentales de Sí C Puede

En este bloque se tratarán todas las herramientas que proporciona Android Studio a los creadores para la programación de sus aplicaciones. Además, se verá cómo se ha ido aplicando todo en Sí C Puede.

Cuando creamos aplicaciones para *Android,* el sistema operativo nos da herramientas esenciales para construir interfaces donde el usuario interactúa, manejar funciones sin que la aplicación esté a la vista, comunicar diferentes partes de la aplicación, o compartir información con otras aplicaciones. Estas herramientas, al estar en *Android Studio*, nos dan una forma ordenada de organizar el código y decidir cómo actúa cada parte de la aplicación.

Aquí te contamos sobre los componentes principales que forman la estructura lógica de una aplicación *Android*:

a) Actividades (*Activities*): las actividades son las pantallas que ves y usas. Cada actividad es como una ventana única, y normalmente una aplicación tiene varias, que se conectan con los *Intents*. Por dentro, toda actividad viene de una clase llamada *Activity*, y puede tener elementos visuales, botones, textos, formularios, etc.

Algo importante de las actividades es que poseen un ciclo de vida, no están siempre activas. Pasan por estados como creada, iniciada, pausada, detenida o destruida, según lo que haga el usuario y el sistema. Entender este ciclo *(figura 4)* ayuda a guardar bien los datos y evitar errores al moverte por la aplicación [13].

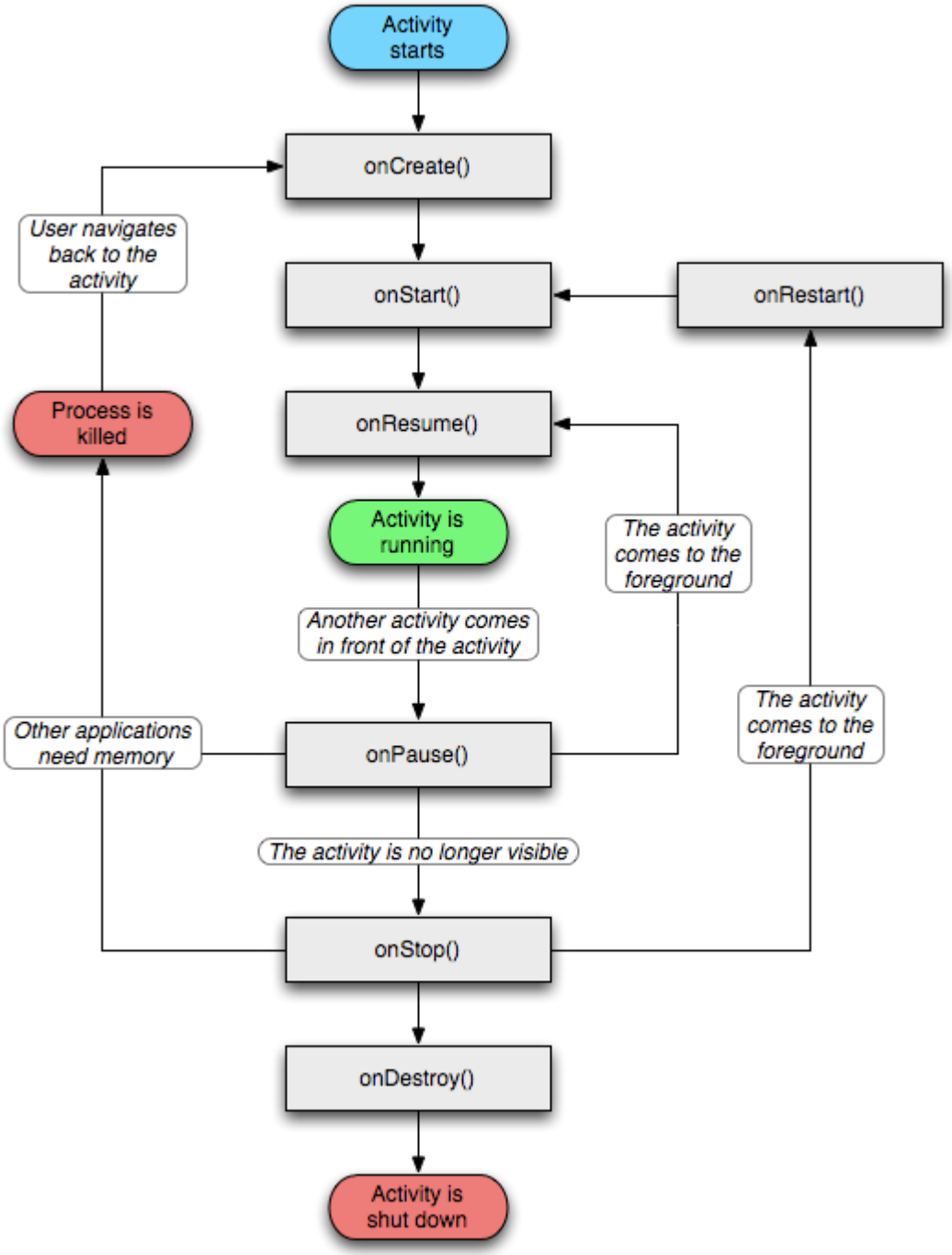

Figura 4 Ciclo de vida de una actividad en Android – (uqbar-wiki)

b) Fragmentos (*Fragments*): los fragmentos sirven para dividir una actividad en partes más pequeñas y que se pueden usar varias veces. Cada fragmento tiene su propia lógica y diseño, y puedes juntarlos en una misma pantalla. Esto ayuda mucho a crear interfaces que cambian según el tamaño de la pantalla o la forma en que se ve [14].

c) Servicios (*Services*): los servicios son componentes hechos para realizar tareas sin que tener que estar usando la aplicación. Por ejemplo, reproducir música, sincronizar datos o

ver tu ubicación. Los servicios pueden ser locales (en la misma aplicación) o remotos (en otras aplicaciones) [15].

d) *Intents* (Solicitudes): los *intents* son como mensajes que dicen que quieres hacer algo, como abrir una actividad, empezar un servicio o hablar con otra parte del sistema. Pueden ser directos (sabes a dónde vas) o indirectos (buscas algo que pueda hacer esa acción). Son como el sistema de mensajes interno de *Android* [16].

e) Receptores de difusión *(Broadcast Receivers*): estos componentes te permiten reaccionar a cosas que pasan en el sistema o en otras aplicaciones. Por ejemplo, si cambia la conexión a internet, si recibes una llamada o si termina de cargar el móvil. Los receptores no tienen pantalla y suelen ser rápidos, solo reciben y manejan lo que pasó [17].

f) Proveedores de contenido (*Content Providers*): en *Android*, la forma en que las aplicaciones comparten información de manera segura se conoce como proveedor de contenido. Imagina que una aplicación necesita usar tus contactos o compartir una foto con otra aplicación; esto es posible gracias a estos proveedores. El sistema establece reglas muy claras sobre quién puede acceder a qué, protegiendo así la información personal del usuario [18].

Cada pieza de estas aplicaciones tiene su propio trabajo, pero se juntan para crear algo adaptable y bien organizado. Esta forma de trabajar es lo que hace posible que aplicaciones como Sí C puede unan cómo se ven, cómo funcionan, cómo guardan datos y cómo se conectan, todo en un solo lugar robusto. En el caso de Sí C puede lo más empleado son las *Activities*, ya que, al requerirse de exponer una gran cantidad de información, se ha decidido separarla en una gran cantidad de pantallas.

2.3. Estructura de un proyecto en Android Studio

Android Studio, siendo el *IDE* oficial para crear aplicaciones *Android*, organiza los proyectos mediante una estructura que separa la lógica del programa, los recursos visuales y la configuración técnica. Esta organización no solo facilita el mantenimiento del código, sino que también permite que la aplicación crezca de forma ordenada conforme se amplían sus funcionalidades.

En la raíz del proyecto se encuentran varios archivos de configuración fundamentales, como *build.gradle* (a nivel de proyecto), que define los ajustes globales, y *settings.gradle*, que indica qué módulos forman parte del proyecto. También aparecen carpetas como *.idea* (relativa a la configuración del *IDE*) y *gradle* (asociada a las herramientas de compilación),

generadas automáticamente por *Android Studio*.

Dentro del módulo principal, que habitualmente recibe el nombre *app*, se encuentra la estructura central de la aplicación:

- La carpeta *src/main/java* contiene el código en *Java* o *Kotlin*. En ella se definen las clases correspondientes a actividades, fragmentos, adaptadores, servicios y otras partes funcionales de la aplicación.
- Por su parte, la carpeta *src/main/res* almacena los recursos visuales: diseños en *XML*, imágenes (*drawable*), textos (*strings.xml*), estilos (*styles.xml*) y menús. Esta separación permite una gestión más eficiente y facilita la internacionalización de la aplicación.
- El archivo *AndroidManifest.xml*, ubicado también dentro de *main*, es clave para la configuración. En él se declaran todas las actividades, permisos, servicios y filtros de intención necesarios para que la aplicación funcione correctamente.
- Adicionalmente, los archivos *build.gradle* del módulo (*app*) permiten ajustar dependencias, versiones del *SDK*, configuración de firma y otros parámetros que influyen directamente en la ejecución de la aplicación.

Esta organización estructurada se observa en la *figura 5*, y permite que distintos desarrolladores trabajen simultáneamente en áreas independientes del proyecto y que las pruebas o ajustes puedan realizarse sin comprometer el funcionamiento global de la aplicación.

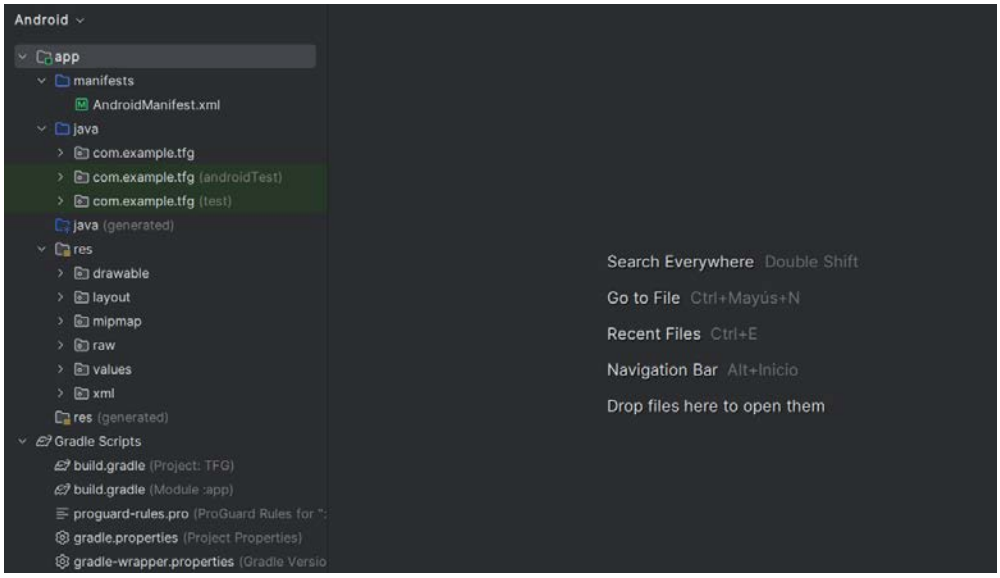

Figura 5 Estructura de carpetas y archivos en un proyecto estándar de Android Studio

Además, *Android Studio* permite cambiar entre distintas vistas del proyecto: *Android*, *Project*, *Packages* o *Project Files*. La vista *Android* resulta especialmente intuitiva para quienes se inician, ya que agrupa los archivos por función en lugar de por su ubicación física, lo cual facilita el trabajo en proyectos con múltiples recursos o módulos.

En el caso de la aplicación *Sí C puede*, esta disposición ha sido clave para mantener una organización clara entre los bloques de formación, los desafíos de programación y la gestión del progreso del usuario. Actividades, fragmentos, diseños y datos se han estructurado de acuerdo con su funcionalidad, respetando esta lógica para facilitar tanto el desarrollo inicial como su futura ampliación.

2.4. Técnicas de aprendizaje gamificado

En los años recientes, la ludificación se ha afianzado como una táctica efectiva para impulsar el compromiso y el desempeño en los espacios formativos. Más allá de ser una simple moda, se trata de una metodología que traslada aspectos propios del juego, como superar desafíos, ganar recompensas o subir de nivel, a escenarios no lúdicos, como la formación reglada o el desarrollo laboral [19].

El propósito esencial de esta práctica es espolear la motivación interna del alumnado, haciendo que el camino del aprendizaje resulte más atractivo, profundo y dinámico. En lugar de seguir la estructura lineal tradicional, las herramientas ludificadas proponen una senda en la que el estudiante se convierte en protagonista activo de su propio avance. Este enfoque ha demostrado ser especialmente útil en ámbitos como la enseñanza de idiomas, la programación o las ciencias experimentales, donde la práctica continua y la retroalimentación inmediata resultan determinantes [20].

Entre las estrategias más comunes de ludificación aplicadas al aprendizaje, destacan las siguientes:

- Sistemas de niveles o avance: permiten trocear el contenido en fases superables, de complejidad creciente. Al completarlas, el estudiante percibe una sensación de logro que refuerza su compromiso.

- Puntuaciones y premios: ayudan a visualizar el avance y recompensan el esfuerzo. Pueden adoptar la forma de puntos, medallas, insignias virtuales (*badges*) o acceso a contenido exclusivo.

- Retos y juegos cortos: introducen tareas concretas que el estudiante debe resolver

aplicando lo aprendido. Favorecen el aprendizaje activo y la fijación de contenidos de forma contextualizada.

- Feedback al instante: proporciona información inmediata sobre el resultado de cada acción, lo cual permite corregir errores de forma autónoma y continuar avanzando sin interrupciones.

- Narrativa envolvente: algunos entornos ludificados sitúan el aprendizaje dentro de un relato o contexto ficticio que da sentido a los desafíos propuestos, generando una mayor conexión emocional.

La combinación de estas técnicas (todas recogidas en la *figura 6*) no solo incrementa la motivación, sino que también favorece el desarrollo de la autonomía, la constancia y la tolerancia a la frustración. Estas son habilidades particularmente relevantes en el ámbito de la programación. Diversos estudios en pedagogía digital indican que la ludificación bien diseñada puede elevar de forma significativa la tasa de finalización de tareas y el tiempo de dedicación dentro de las plataformas educativas [21].

Figura 6 Principales tácticas de ludificación aplicadas al aprendizaje

En el caso específico de *Sí C puede*, la ludificación no se emplea como un añadido superficial, sino como un pilar estructural. La aplicación incluye niveles de dificultad gradual, desafíos independientes y un mecanismo visual de seguimiento del avance, con el objetivo de transformar el aprendizaje del lenguaje C en una experiencia participativa, motivadora y adaptada a los intereses del alumnado actual.

2.5. Plataformas educativas digitales con gamificación

La irrupción de las tecnologías digitales en la enseñanza ha modificado radicalmente el modo de impartir clases y adquirir conocimientos. En este nuevo escenario, han surgido varias plataformas diseñadas para el estudio en línea, muchas de ellas integrando estrategias para enriquecer la experiencia del usuario y fomentar un involucramiento dinámico.

Plataformas como *Kahoot* [22], *Duolingo* [23] o *CodeCombat* [24] (*figura 7*) se han destacado por su destreza para insertar componentes recreativos sin comprometer la formalidad del material. Todas comparten una perspectiva donde el usuario progresa a través de etapas, obtiene premios virtuales, rivaliza contra sí mismo o con otros, y recibe retroalimentación instantánea después de cada movimiento. Tales procedimientos han demostrado ser eficientes para sostener el interés y afianzar el saber por medio de la reiteración, especialmente en situaciones remotas [25].

Figura 7 Logos de las distintas aplicaciones de gamificación

Una de las plataformas más reconocidas en el entorno escolar y universitario es *Kahoot*, que facilita la creación de exámenes interactivos con una estructura de juego competitivo entre los alumnos. Su sencillez de uso y su aspecto divertido, dado que con frecuencia se emplea en tiempo real en clase, la han convertido en un instrumento habitual tanto en clases presenciales como en línea.

En el terreno del aprendizaje de idiomas, *Duolingo* ha conseguido establecerse como una de las aplicaciones más populares a nivel global gracias a su esquema de lecciones

breves, oportunidades restringidas, conteos acumulativos, distinciones (*badges*) y desafíos semanales. Su táctica se fundamenta en la reiteración gradual y en el estímulo constructivo, generando un hábito diario en los usuarios.

En cambio, sitios como *CodeCombat* o *Grasshopper* se dedican a mostrar cómo programar. Ahí, los usuarios solucionan problemas con una cierta rareza en mundos alternativos, pudiendo adquirir ideas técnicas el usuario al hacer tareas. El problema en este tipo de páginas reside en que no se enseña la programación desde una base y carece de una meta de aprendizaje.

En la *tabla 1*, podemos observar las principales características de las plataformas mencionadas, todas estas plataformas tienen algo en común: han hecho que aprender se parezca más a jugar, se parezca a divertirse. Esto es muy útil para los que crecieron con videojuegos, redes sociales y cosas que se tocan, donde las imágenes y saber al instante si lo haces bien son importantes.

Tabla 1 Comparativa entre plataformas educativas con dinámicas gamificadas

Plataforma	Público objetivo	Contenido	Mecánicas de juego	Retroalimentación
Kahoot	Escolar/universitario	Cuestionarios	Puntos, ránking, tiempo límite	Inmediata y competitiva
Duolingo	General	Idiomas	Puntos, vidas, insignias	Instantánea y motivadora
CodeCombat	Estudiantes de programación	Programación	Narrativa, niveles, retos	Contextual y narrativa

La aplicación *Sí C Puede* va por el mismo camino, pero enseñando el lenguaje C de programación, con niveles sueltos, retos aparte y todo en tu teléfono móvil y sin la necesidad de conexión a Internet. No busca ser mejor que las otras, sino usar lo bueno que tienen para ponerlo en un lugar de estudio más técnico y específico.

2.6. Comparativa entre aplicaciones educativas tradicionales y gamificadas

La evolución de las aplicaciones destinadas al aprendizaje ha generado dos enfoques claramente diferenciados: aquellas que siguen un modelo más clásico y aquellas que incorporan dinámicas propias del juego para captar la atención. Ambas comparten un mismo objetivo —facilitar el aprendizaje—, pero lo hacen desde perspectivas distintas, ofreciendo experiencias que difieren notablemente en su forma y en su impacto sobre el usuario.

Las aplicaciones tradicionales suelen presentar los contenidos de manera secuencial: bloques teóricos seguidos de ejercicios prácticos. La interacción es limitada y la

retroalimentación suele proporcionarse únicamente al final de cada unidad o evaluación. El progreso se mide mediante calificaciones, y la interfaz suele estar más enfocada en la funcionalidad que en el diseño visual o la experiencia del usuario.

En contraste, las aplicaciones gamificadas convierten el aprendizaje en un proceso más participativo. Los contenidos se organizan en forma de niveles, misiones o retos, acompañados de elementos visuales estimulantes que invitan a interactuar. Este enfoque, además de fomentar la permanencia en la plataforma, potencia la motivación intrínseca y el compromiso con el proceso formativo.

En la *tabla 2* encontramos un breve resumen comparativo entre las aplicaciones tradicionales y las gamificadas.

Tabla 2 Comparativa entre aplicaciones educativas tradicionales y gamificadas

Aspecto	Aplicaciones tradicionales	Aplicaciones gamificadas
Organización del contenido	Secuencial: teoría seguida de ejercicios	Por niveles, retos o misiones
Tipo de *feedback*	Al final de cada bloque o test	Inmediato, tras cada acción
Motivación	Extrínseca: basada en calificaciones	Intrínseca: basada en logros y recompensas
Participación del usuario	Limitada, centrada en lectura y respuesta	Activa, interactiva y constante
Diseño de la interfaz	Funcional, poco visual	Visual, dinámica y estimulante

En este sentido, la aplicación *Sí C puede* adoptar decididamente el enfoque gamificado. El lenguaje C, por su complejidad inicial y su lógica estructurada, puede resultar intimidante para quienes se inician. Por ello, la introducción de niveles de dificultad, retos individuales, recompensas simbólicas y seguimiento visual del progreso permite generar un entorno más accesible y estimulante, sin sacrificar el rigor técnico del contenido.

2.7. Plataformas para la distribución de aplicaciones

Después de crear una *aplicación* para *Android*, lo más natural es publicarla para que la gente pueda instalarla en sus móviles. Para esto, hay varias tiendas digitales donde puedes subirla, darle visibilidad y actualizarla. Estas tiendas no son solo un escaparate, sino que también te dan datos de uso, opciones para actualizar la aplicación automáticamente y un sistema de opiniones de los usuarios.

La más famosa y usada en todo el mundo es la *Google Play Store*. Es la tienda oficial para *Android* y te permite llegar a una gran cantidad de gente. Ahí puedes lanzar versiones *beta*, revisar la seguridad y ganar dinero con anuncios o compras dentro de la aplicación [26].

Eso sí, para publicar en *Google Play* tienes que pagar una cuota inicial y cumplir con varias normas técnicas, legales y de privacidad estrictas.

Otra opción interesante es *Uptodown*, una tienda de aplicaciones independiente que está en España. A diferencia de *Google Play*, no te cobra ni requiere cuenta de desarrollador, pudiéndose descargar los archivos *APK* directamente sin registrarte. Su política más abierta la ha hecho popular en países donde es difícil acceder a *Google* o donde no hay buena conexión a *internet*. Además, tiene muchos usuarios y también proporciona estadísticas de descargas, visibilidad en otros países y versiones para varias plataformas [27].

También hay otras tiendas, como la *Amazon Appstore*, que se centran en aparatos de *Amazon* (como las *tablets Fire*), o *F-Droid*, que se centra en *software* libre y de código abierto. Pero estas últimas no llegan a tanta gente y suelen ser para un público más específico.

En la *tabla 3* vemos una comparativa entre las tiendas, a través de la cual se puede determinar cuál es la mejor opción dependiendo del estado de desarrollo de la aplicación.

Tabla 3 Comparativa entre tiendas para publicar aplicaciones móviles

Plataforma	Coste de publicación	Requisitos técnicos	Visibilidad	Monetización	Descarga directa
Google Play Store	Pago único	Altos	Muy alta	Sí	No
Uptodown	Gratuito	Moderados	Alta	Limitada	Sí
Amazon Appstore	Gratuito	Moderados	Media	Sí	No
F-Droid	Gratuito	Código Abierto	Baja	No	Sí

Para la aplicación *Sí C puede*, una opción realista es la de publicarla en *Uptodown* porque es fácil, gratis y rápido para que la aprueben. Esto es ideal para una aplicación educativa que no busca ganar dinero al momento, sino que se use en universidades o cursos. Aunque en un primer momento el objetivo es que la aplicación sea testada por los alumnos de la ETSII que van a cursar la asignatura ‘Fundamentos de Programación’. No se descarta entrar a la *Google Play Store* si la aplicación evoluciona.

3. Diseño y funcionalidad de la aplicación

Cualquier proyecto tecnológico, antes de empezar a codificar, necesita un tiempo para pensar y planificar. En este apartado se explicarán las decisiones clave que moldearon *Sí C puede*, tanto en su faceta didáctica como en la técnica. Profundizaremos en los principios de

enseñanza que guían este proyecto, las funcionalidades que queríamos que tuviera, y cómo organizamos la experiencia para quienes lo usen.

También mostraremos cómo logramos que el contenido avance poco a poco, la razón de ser de los retos, cómo se lleva el control de lo que aprende cada persona, y lo que se hizo para que la aplicación sea fácil de usar y accesible para todos. Todo esto construye una base firme que ayuda a aprender el lenguaje C en un ambiente digital que engancha y se ajusta a lo que necesita la gente.

3.1. Enfoque pedagógico aplicado

La concepción de *Sí C puede* se apoya en una estrategia educativa activa, centrada en el estudio por cuenta propia y que se basa en la motivación positiva. En vez de adherirse a una fórmula rígida o directa, la aplicación permite a cada persona desarrollarse a su propio paso, indagando en los temas según sus inclinaciones o requerimientos personales. Dicha adaptabilidad impulsa un compromiso superior con el desarrollo y honra la variedad de tiempos y métodos de estudio presentes en el grupo.

Un punto esencial del esquema es la división del material en dos partes que se complementan entre sí. Por un lado, está el sistema de enseñanza gradual, que organiza los asuntos desde lo más elemental hasta ideas de mayor dificultad. Esta progresión sencilla facilita la creación de saberes poco a poco y hace posible fortalecer lo aprendido antes de seguir adelante. Por otro lado, el sistema de retos sueltos ofrece desafíos prácticos que pueden solucionarse cuando se quiera, sin tener que mantener un orden marcado. Esta unión ayuda tanto a la asimilación teórica como al uso real del lenguaje C.

El empleo de tácticas que se inspiran en la dinámica de juegos potencia esta perspectiva. Detalles como los niveles o los botones de "señalar como aprendido" que dejan que el usuario observe su desarrollo, generan que el usuario se sienta animado a continuar. Por medio de estas vías se alienta la perseverancia, se refuerza el proceso de toma de decisiones y se activa el deseo de mejora personal, aptitudes muy útiles en la instrucción de lenguajes de programación.

Toda la estructura didáctica se ideó desde el punto de vista de una herramienta de ayuda, que valga como añadido a la enseñanza formal y no como un reemplazo de esta. *Sí C puede* no trata de forzar un solo camino, sino de dar un espacio cercano, visual e incitante donde ensayar, repasar y consolidar las bases del lenguaje C con más independencia sea posible.

3.2. Casos de uso y requisitos del sistema

El esquema práctico de *Sí C puede* considera un abanico de situaciones que revelan cómo el usuario interactúa con el programa en las diversas fases del trayecto educativo. Estas situaciones sirven para imaginar las acciones cruciales y prever las exigencias técnicas que se deben satisfacer para certificar un desempeño constante.

Entre las situaciones más destacadas se hallan:

- Indagación abierta del material: el usuario puede entrar a cualquier peldaño del bloque educativo cuando lo desee, sin depender de una secuencia fijada. Esta elección obedece al propósito de brindar versatilidad y ajuste a variados estilos y velocidades de aprendizaje.

- Ejecución de retos: cada prueba separada propone una tarea específica vinculada con la programación en C. El usuario ingresa el código apropiado y recibe una respuesta gráfica sobre el resultado, sea este acertado o erróneo.

- Control del avance: la aplicación da la opción de señalar los niveles como "aprendidos" y, a la vez, expone un icono que al ser pulsado plasma visualmente el progreso del usuario dentro del itinerario. Esta característica funciona como estímulo positivo y orientación interna.

Partiendo de estas situaciones, se establecieron las necesidades prácticas y accesorias que debía llenar el sistema. Entre las más sustanciales, se localizan:

- Acceso rápido y sin registro obligatorio.
- Interfaz comprensible para usuarios sin conocimientos técnicos previos.
- Compatibilidad con versiones recientes de *Android*.
- Estructura modular que permita añadir nuevos niveles o desafíos sin modificar la base del código.
- Estabilidad en la navegación entre pantallas.
- Almacenamiento interno para conservar el progreso sin conexión a *internet*.

- Ausencia de publicidad o elementos que interrumpan la experiencia educativa.

Todos estos requisitos han sido fundamentales para tomar las mejores decisiones durante el desarrollo de la aplicación, asegurando que la experiencia de uso sea funcional, coherente y centrada en el aprendizaje. Es en la *figura 8* donde podemos ver muy *grosso modo* lo que es la aplicación.

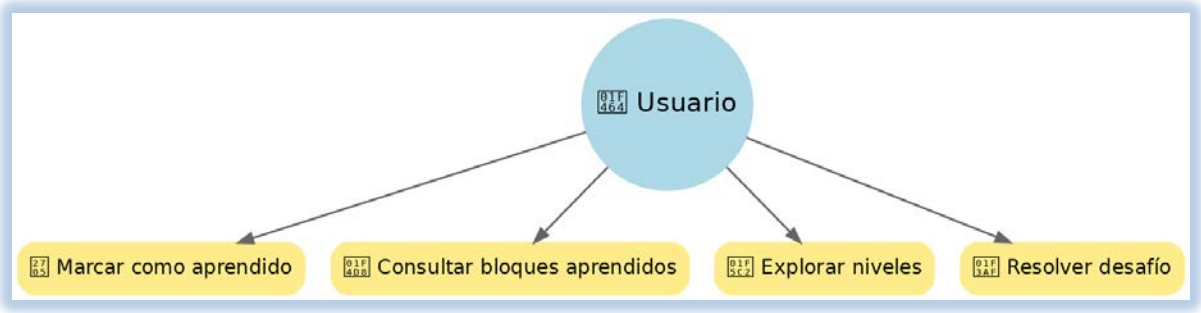

Figura 8 Diagrama simplificado de los principales casos de uso de la aplicación Sí C puede

3.3. Diseño de la interfaz y experiencia del usuario

En *Sí C puede*, una prioridad ha sido diseñar una experiencia de usuario intuitiva, ágil y atractiva, que impulse el aprendizaje sin trabas técnicas ni cognitivas. Para conseguir esto, se eligió una interfaz sencilla, con un diseño visual ordenado y elementos fáciles de entender para cualquier persona, incluso si nunca han usado aplicaciones educativas.

Moverse por la aplicación es fácil gracias a tres secciones clave: el modo de aprendizaje paso a paso, la sección de retos y el menú de progreso, al que se accede desde la pantalla principal. Esta organización por partes ayuda al usuario a ubicarse sin problemas y a elegir cómo quiere seguir aprendiendo en cada momento.

Cada nivel del modo paso a paso tiene un título que explica de qué va, un breve resumen del contenido y un botón para señalar que se ha aprendido. Una vez marcado, el nivel se guarda y aparece en el menú de progreso, donde se juntan todos los bloques completados hasta ahora. En vez de usar una barra de progreso común, se optó por un sistema más sutil y que depende del usuario.

En cuanto a los retos, cada uno se muestra como una tarjeta aparte, con una instrucción clara y un espacio para la explicación y la posterior puesta en práctica del código. La idea es animar a ser independientes y a experimentar, sin obligar a seguir un camino fijo ni un orden preestablecido.

Se ha cuidado mucho la elección de colores, los espacios y el tipo de letra, buscando

siempre un equilibrio entre lo bonito y lo práctico. Todos los textos están escritos de forma que sean fáciles de entender, sin jerga innecesaria, y los botones tienen nombres claros para que no haya dudas.

En la *figura 9* podemos observar tanto la pantalla principal, como las otras dos pantallas que podemos abrir a partir de esta.

Figura 9 Las 3 pantallas principales de Sí C puede

La meta es crear una experiencia a medida, donde el usuario controle su aprendizaje y pueda avanzar cuando quiera, en un entorno visualmente agradable y sin elementos que distraigan.

3.4. Navegación entre pantallas

El esquema de navegación dentro de *Sí C puede* se ideó con un enfoque de sencillez práctica: la idea es que cualquier persona pueda moverse entre las diversas áreas sin necesidad de un tutorial ni ser un experto. Esta elección busca lograr que la experiencia al usarla sea ágil, fácil de entender y sin complicaciones innecesarias.

En la página de inicio, se ofrecen dos botones (*figura 10*) bien definidos que guían a las dos formas de usar la aplicación: el modo de aprendizaje gradual, donde se encuentran los niveles ordenados por temas, y el modo de retos, que plantea desafíos concretos para aplicar lo que se ha aprendido. Se puede entrar a cualquiera de los dos sin importar el orden y según lo que prefiera el usuario.

Figura 10 Botones Aprendizaje Progresivo – Desafíos – Sí C puede

En cada pantalla, se puede volver al inicio tocando un icono discreto que indica el regreso y que está abajo, así como hay otro botón que permite pasar al siguiente subapartado de aprendizaje *(figura 11).* Gracias a esta función, el usuario siempre tiene el control de por dónde va, evitando así que se sienta atascado o desorientado mientras usa la aplicación.

Figura 11 Botones Volver al menú – Siguiente - Sí C puede

En el modo gradual, cada tema incluye su propio botón *(figura 12)* para indicar que ya se dominó, lo cual hace que se muestre en la lista desplegable de progreso, a la cual se entra desde la esquina superior derecha de la página principal. Este sistema reemplaza la típica barra de progreso y ofrece una forma más personalizada y sutil de ver lo que se ha logrado.

Figura 12 Botón Marcar como aprendido - Sí C puede

En el modo de retos, los desafíos se muestran como botones dispuestos en una pantalla *(figura 13),* con nombres claros e iconos pequeños que ayudan a saber de qué se trata cada uno rápidamente. La navegación en esta sección también es directa: se escoge un desafío y se entra de una vez a la interfaz para resolverlo.

Figura 13 Disposición de los desafíos por pantalla - Sí C puede

3.5. Funcionalidades de aprendizaje progresivo

El bloque de aprendizaje progresivo en *Sí C puede* está concebido como una guía estructurada y gradual para introducir al usuario en los fundamentos del lenguaje C. La organización del contenido se realiza en bloques temáticos, que agrupan niveles por orden lógico y pedagógico. Esta distribución permite avanzar desde los conceptos más básicos hasta estructuras de mayor complejidad, respetando el ritmo de aprendizaje de cada persona.

Cada nivel está compuesto por:

- Un título que identifica de forma clara el contenido del nivel.
- Un texto explicativo breve y comprensible, que introduce el concepto con ejemplos sencillos.
- Un botón de acción que permite al usuario marcar el nivel como aprendido una vez completado (cada bloque).

Una vez pulsado el botón *Marcar como aprendido*, ese nivel se incorpora automáticamente al desplegable de progreso (*figura 14*), accesible desde la pantalla principal (*figura 15).* Este sistema actúa como un registro del camino recorrido, permitiendo consultar en cualquier momento qué niveles se han completado sin imponer un sistema de evaluación ni puntuación que pueda desmotivar al usuario.

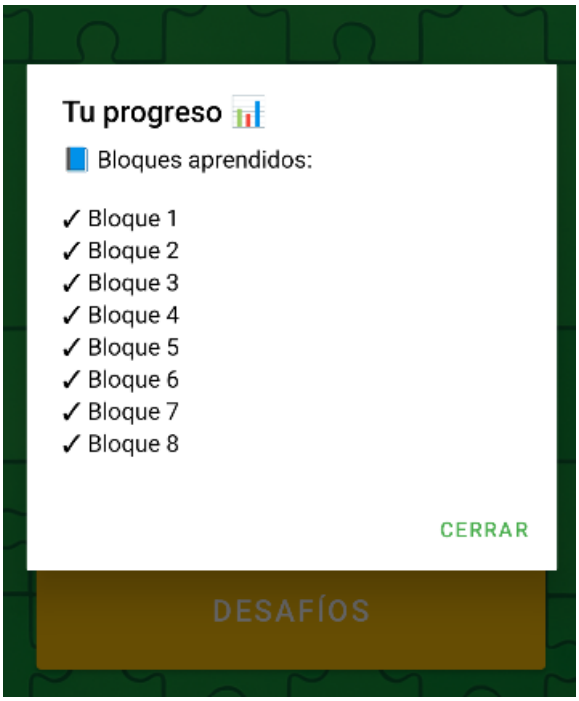

Figura 14 Desplegable con los bloques 'Marcados como aprendidos' - Sí C puede

Figura 15 Icono para la aparición del desplegable - Sí C puede

Esta lógica no solo facilita la organización del contenido, sino que también fomenta la autonomía y la autorregulación del aprendizaje. El usuario elige cuándo y cómo avanzar, y puede volver a consultar cualquier nivel anterior sin restricciones.

El formato visual emplea colores diferenciados (amarillo y rosa) para que el usuario sepa el momento en el que entra a un nuevo bloque (*figura 16*), además de una tipografía clara, con márgenes amplios que facilitan la lectura en dispositivos móviles. Todo el diseño responde a una premisa fundamental: que el contenido sea accesible, útil y motivador para estudiantes con distintos niveles de experiencia en programación.

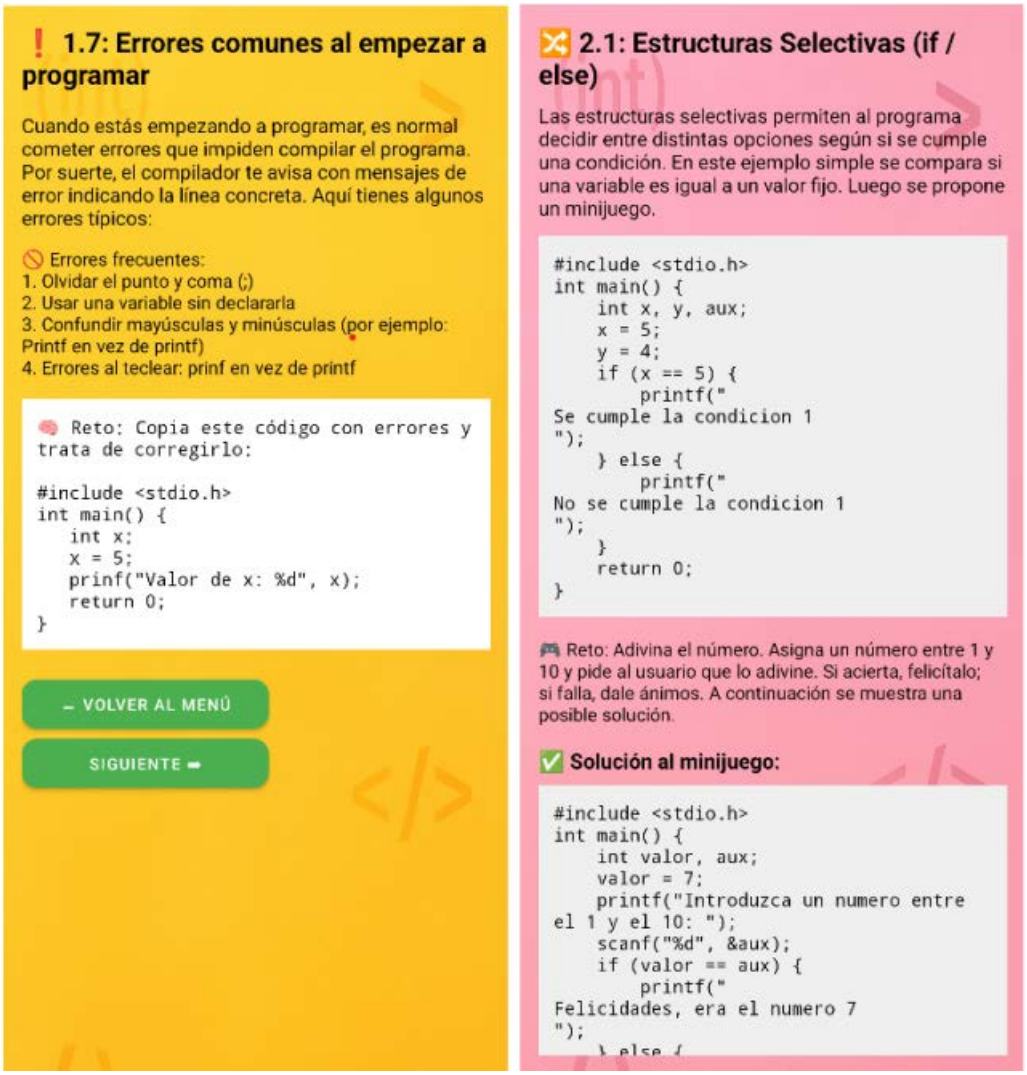

Figura 16 Último subapartado de un bloque y primero del siguiente bloque - Cambio de color - Sí C puede

3.6. Funcionalidades de desafíos de programación

El apartado de desafíos de *Sí C puede* está diseñado como un espacio complementario al aprendizaje progresivo, un espacio donde el usuario puede aplicar los conocimientos adquiridos en situaciones prácticas, contextualizadas y donde él puede apreciar el gran potencial que tiene la programación en cualquier ámbito. A diferencia del bloque teórico, aquí no se sigue un orden predeterminado: los desafíos pueden resolverse de forma independiente, favoreciendo la exploración libre y la resolución activa de problemas.

Cada desafío aparece como un botón visualmente destacado, acompañado de un icono representativo y un título claro. Al pulsar sobre cualquiera de ellos, el usuario accede a una nueva pantalla donde se detalla la consigna del reto, orientada a estimular el razonamiento lógico y la escritura de código funcional en lenguaje C.

Los diez desafíos actuales abarcan una variedad de contenidos y niveles de dificultad:

- Desafío 1: Puntuación en F1 *(figura 17)*.

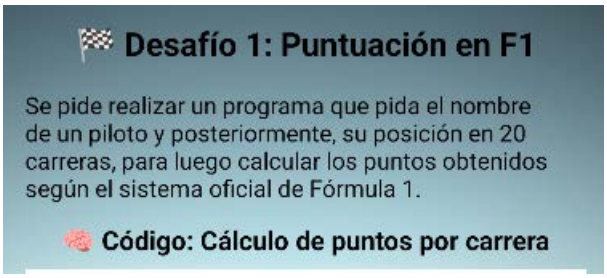

Figura 17 Desafío 1 - Puntuación en F1 - Sí C puede

- Desafío 2: Encuesta de popularidad *(figura 18)*.

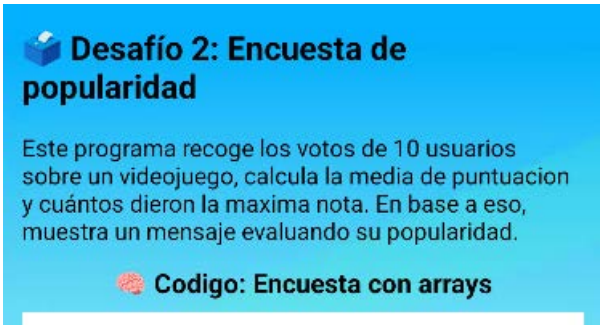

Figura 18 Desafío 2 - Encuesta de popularidad - Sí C puede

- Desafío 3: Simulador de dados *(figura 19)*.

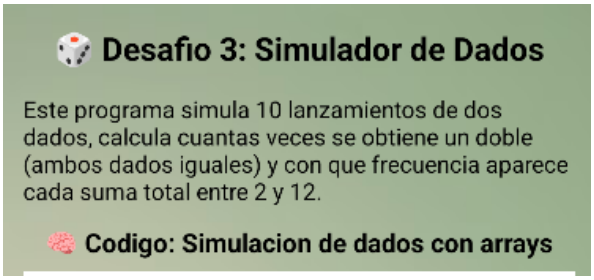

Figura 19 Desafío 3 - Simulador de dados - Sí C puede

- Desafío 4: Generador de contraseñas seguras *(figura 20).*

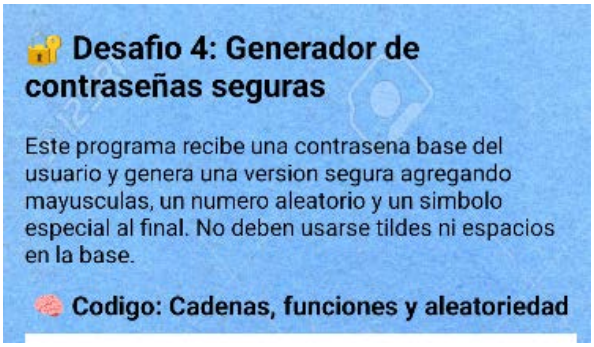

Figura 20 Desafío 4 - Generador de contraseñas seguras - Sí C puede

- Desafío 5: Calculadora polivalente *(figura 21)*.

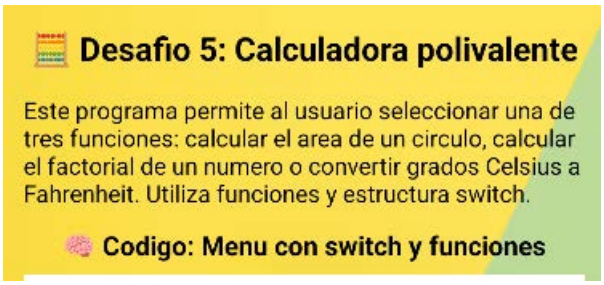

Figura 21 Desafío 5 - Calculadora polivalente - Sí C puede

- Desafío 6: Adivina el número *(figura 22).*

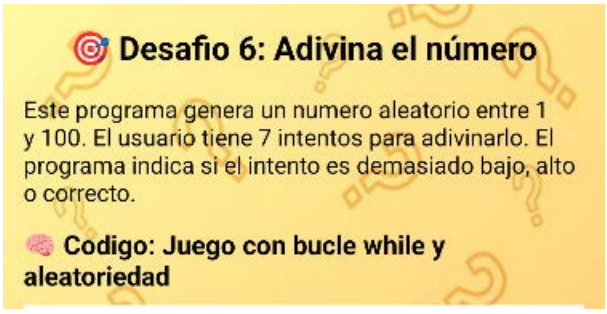

Figura 22 Desafío 6 - Adivina el número - Sí C puede

- Desafío 7: Validador de contraseñas *(figura 23)*.

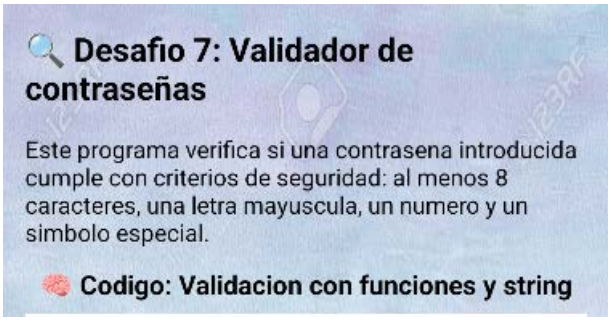

Figura 23 Desafío 7 - Validador de contraseñas - Sí C puede

- Desafío 8: Conversor de unidades *(figura 24)*.

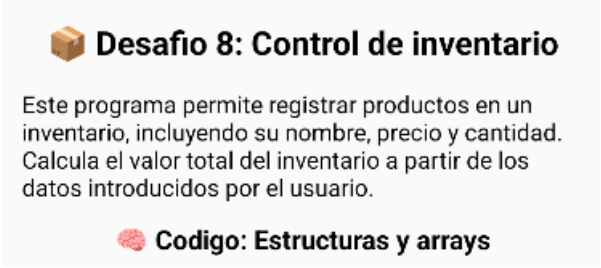

Figura 24 Desafío 8 - Conversor de unidades - Sí C puede

- Desafío 9: Estadísticas de temperaturas (figura 25).

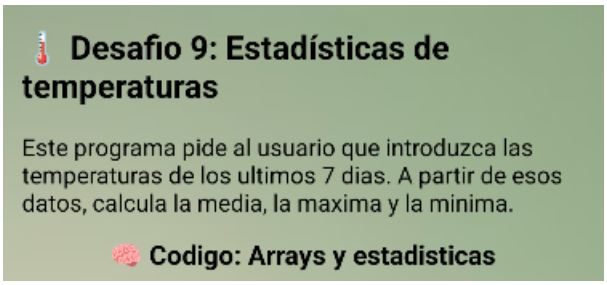

Figura 25 Desafío 9 - Estadísticas de temperaturas - Sí C puede

- Desafío 10: Analizador de texto *(figura 26)*.

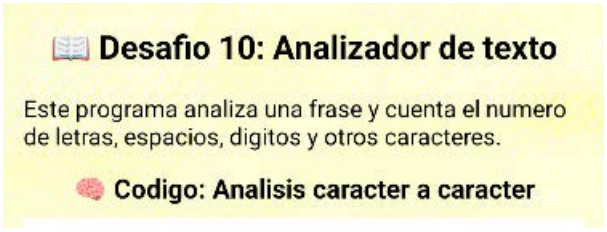

Figura 26 Desafío 10 - Analizador de texto - Sí C puede

Estos desafíos han sido planteados para cubrir distintos aspectos del lenguaje C, desde operaciones elementales hasta el manejo de estructuras más complejas, siempre desde un enfoque lúdico y motivador.

Al no estar condicionados por un orden rígido, los desafíos permiten al usuario experimentar, cometer errores y volver a intentar sin penalizaciones, lo que refuerza el aprendizaje a través de la práctica y fomenta la perseverancia en la resolución de problemas.

3.7. Gestión del progreso del usuario

Este modelo que ya se ha expuesto en la memoria, facilita la revisión de contenidos, ya que permite identificar rápidamente qué temas se han trabajado y cuáles podrían retomarse. Así, el propio usuario puede construir su itinerario de estudio en función de sus necesidades, sus dudas o el tiempo disponible.

Cabe destacar que esta gestión del progreso se realiza de forma local, sin necesidad de conexión a internet, registro previo ni recopilación de datos personales. Esta decisión se alinea con el carácter educativo, lúdico y respetuoso con el usuario de la aplicación.

3.8. Consideraciones de accesibilidad y usabilidad

Desde que se ideó, *Sí C puede* se pensó como una herramienta didáctica que fuera fácil de usar y pensada para todos. Para lograrlo, se integraron criterios de accesibilidad y uso sencillo que permiten que la use mucha gente, con conocimientos de tecnología variados y necesidades particulares.

Visualmente, se eligió una interfaz despejada, con un diseño nítido y elementos bien ubicados. Los bordes anchos, la letra fácil de leer y el buen contraste entre el texto y el fondo ayudan a que la lectura sea agradable, incluso en pantallas pequeñas. Se evitó usar colores fuertes o movimientos que puedan molestar o distraer durante el tiempo de uso.

Para moverse por la aplicación, todos los botones se distinguen bien con nombres claros y están donde uno se los puede esperar. El menú principal facilita el acceso a las dos funciones principales, aprender poco a poco y los retos. El icono del progreso siempre está a la vista en la pantalla principal, sin estorbar a la interacción principal.

Algo que se aprecia mucho en cuanto a la accesibilidad es que no pide nada extra: la aplicación no te pide registrarte, no necesita internet y guarda tus avances directo en el dispositivo, lo que ayuda a usarla en escuelas con pocos recursos o mala conexión.

Finalmente, es importante decir que se ha cuidado la carga mental. Cada pantalla solo muestra lo necesario, sin llenar al usuario con opciones o texto de más. Esta estructura simple ayuda a concentrarse en el contenido y reduce las probabilidades de equivocarse, lo que hace que la aplicación sea muy útil para estudiantes que recién empiezan a programar y es muy posible que, si de primeras no les gusta, generen un rechazo natural que luego les puede dificultar más el aprendizaje.

4. Desarrollo de la aplicación

Una vez que de definen los propósitos educativos y las funcionalidades más importantes de *Sí C puede*, este apartado profundiza en los detalles técnicos que han viabilizado la materialización del proyecto. Aquí, se desglosan las resoluciones tomadas a lo largo del curso del desarrollo, la arquitectura interna del código, el esquema de clases utilizado y los componentes visuales que configuran el carácter distintivo de la aplicación.

Se elabora todo el proyecto utilizando en *Android Studio*, utilizando tecnologías que se son compatibles con los dispositivos de hoy y asegurando un desempeño ágil en pantallas de

diversos tamaños. Cada parte fue concebida para simplificar la comprensión del lenguaje C en un ambiente que sea a la vez fácil de usar, práctico y que sea llamativo a la vista.

4.1. Nombre, logo y paleta de colores

La imagen que busca proyectar Sí C va mucho más allá de lo estético, se busca crear un espacio agradable y fácil de interiorizar que ayude a aprender y anime al usuario. Cada diseño y forma se pensó para que todo se vea claro, cercano y tenga sentido con lo que se explica.

El logo de la aplicación *(figura 1),* juega con las letras para que leamos “sí se puede” y a la vez pensemos en el lenguaje de programación C. Este truco visual no solo muestra el contenido temático de la aplicación, sino que nos recuerda que todos podemos aprender a programar.

Para los colores, elegimos tonos alegres y fáciles de distinguir: verde, amarillo, rosa, naranja y rojo *(figura 27)*. Cada uno representa una parte diferente de la aplicación, para que sea más fácil encontrar lo que buscas y la experiencia sea divertida, pero sin cansar la vista. Además, usar los colores así ayuda a concentrarse y a separar los temas.

Figura 27 Paleta cromática de Sí C puede

La letra que se usa es sencilla *(figuras 17 – 26),* facilitando la lectura en cualquier pantalla. También se evita poner sobrecargas de información, buscándose practicidad y estética al mismo tiempo.

4.2. Organización del código fuente y estructura interna

Para darle forma a *Sí C puede*, como se ha mencionado ya infinidad de veces, se utiliza *Android Studio* como nuestro espacio de trabajo principal. Para ello se comienza con la creación de un nuevo proyecto, específicamente con la funcionalidad de *‘Empty Views Activity’ (figura 28).*

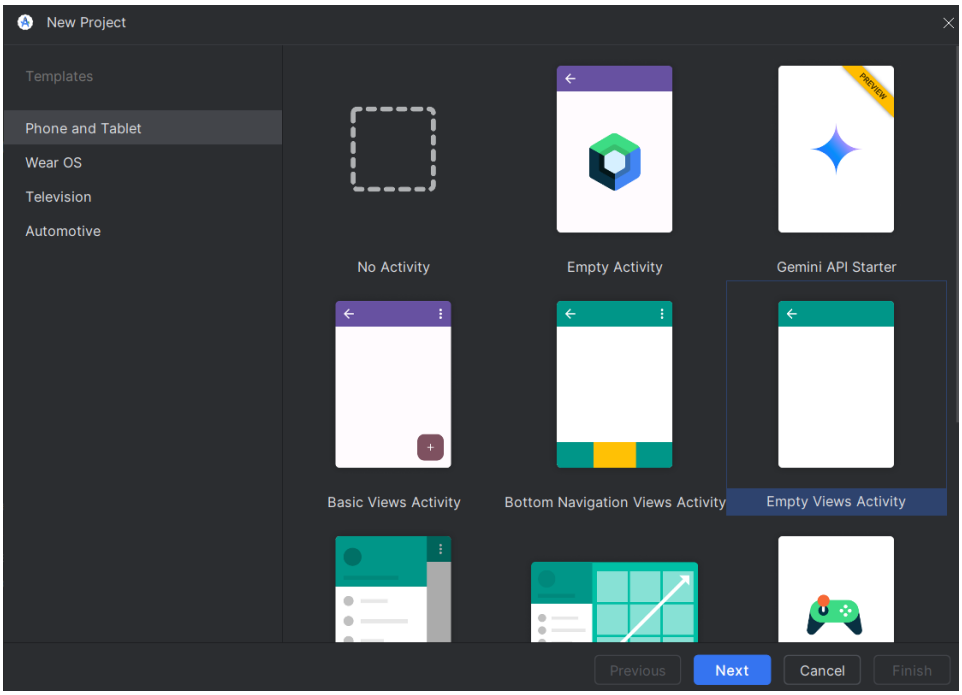

Figura 28 Pantalla Android Studio - Nuevo Proyecto

Desde el comienzo, se opta por una configuración común basada en módulos. El principal, llamado *app*, guarda lo esencial del proyecto. Dentro de él, están carpetas importantes como *java (figura 29)*, donde se introduce el código fuente; *res (figura 30)*, para todos los detalles gráficos y de la interfaz, es decir, es aquí donde se programan los *layouts (figura 31)* de cada una de las pantallas que forman la aplicación, además de configurar por ejemplo la música de fondo que suena en la aplicación a través de la carpeta *raw (figura 32)*; y el archivo *AndroidManifest.xml (figura 33)*, que se encarga de definir las actividades, los permisos y las configuraciones clave para que la aplicación funcione bien.

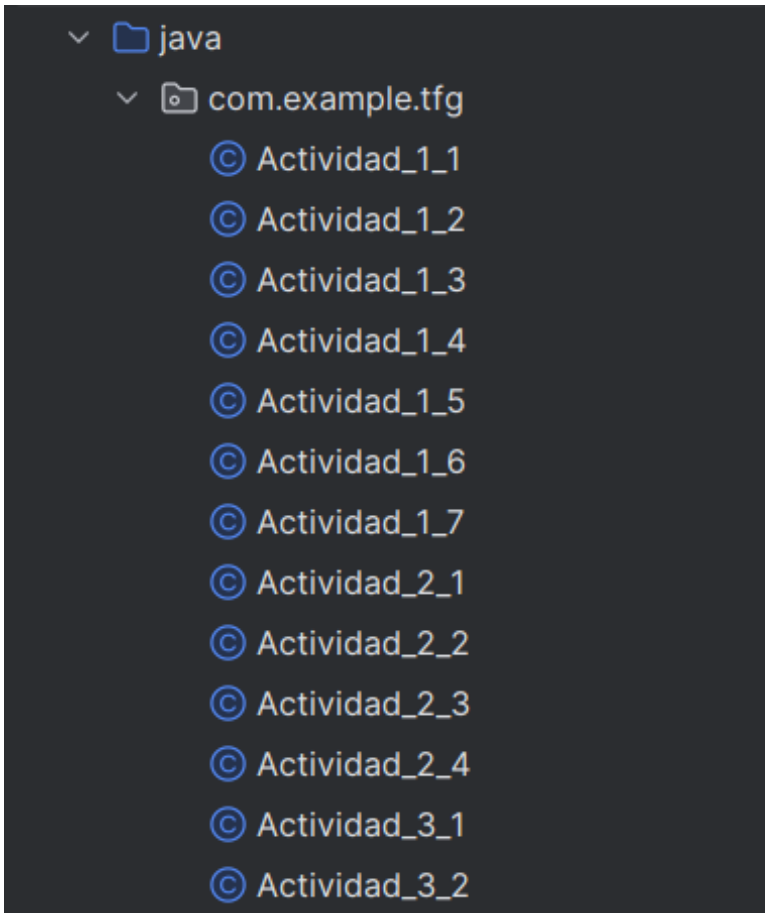

Figura 29 Actividades que componen la aplicación Sí C puede en la carpeta java – Android Studio

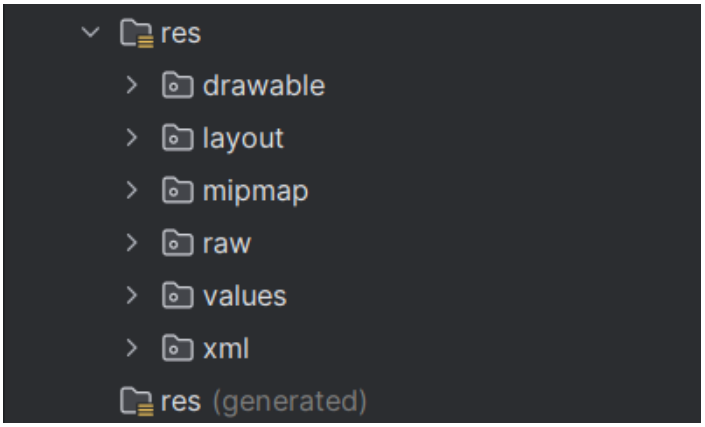

Figura 30 Carpeta res – Android Studio

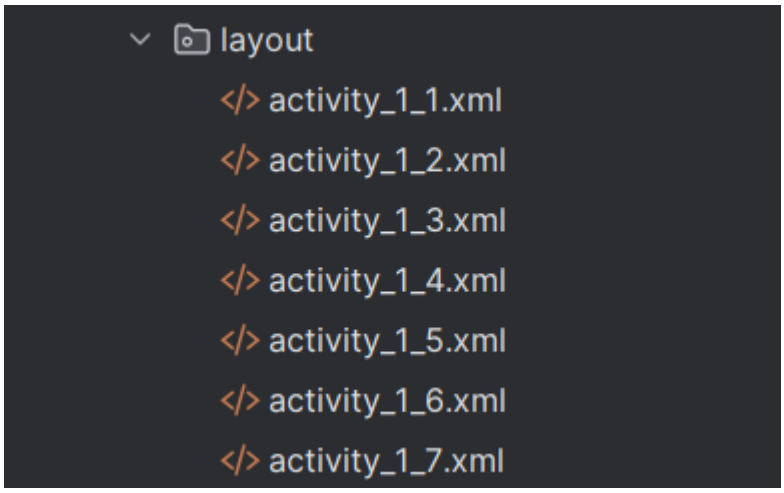

Figura 31 Carpeta layout con varios xml que componen la aplicación – Android Studio

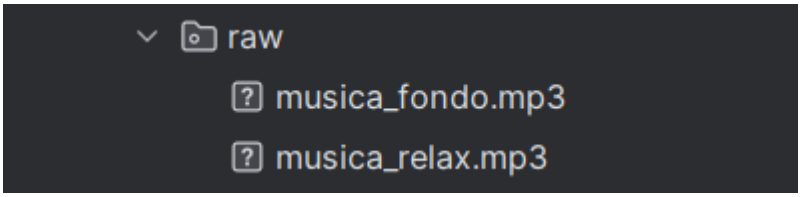

Figura 32 Carpeta raw con archivos .mp3 (música de fondo) – Android Studio

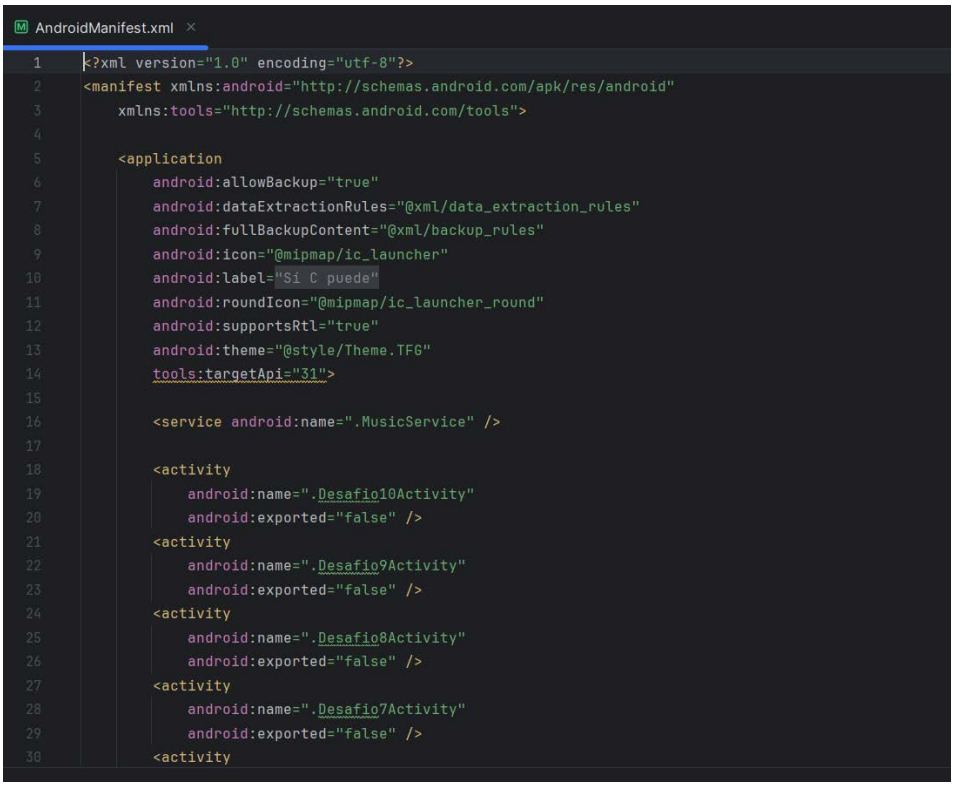

Figura 33 Parte del código AndroidManifest.xml – Android Studio

Se ajusta la aplicación para que funcione bien con versiones de *Android* a partir de la 8.0 (*API* nivel 26). Se elige esto en la búsqueda del punto medio entre la cantidad de dispositivos que cubre, la estabilidad del sistema y el acceso a las funciones más nuevas. De

esta manera, se asegura que la *aplicación* funcione en muchos dispositivos sin afectar su rendimiento.

El archivo *build.gradle (figura 34)* del módulo principal tiene como función manejar las dependencias, fijar las versiones del *SDK* y definir cómo se compila el código. Gracias a esta estructura, pudimos mantener el proyecto ordenado, evitar errores y facilitar las actualizaciones en el futuro.

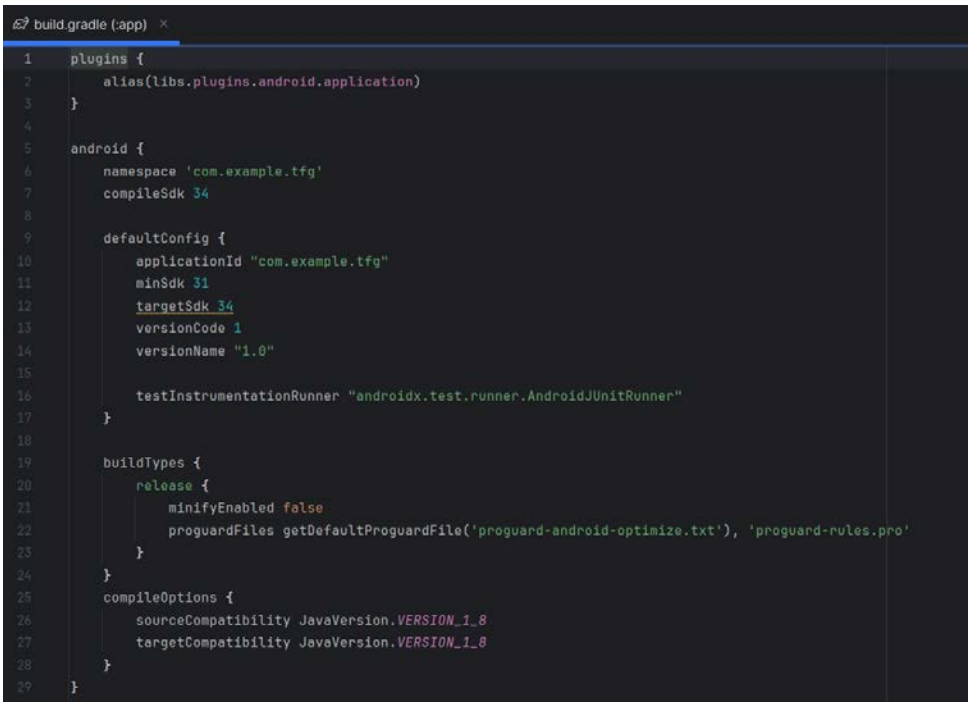

Figura 34 Archivo build.gradle – Android Studio

Aunque no se usa un sistema externo para controlar las versiones, como *GitHub* o *GitLab*, sí se trabaja con varias copias locales del proyecto, que se guardan antes de hacer cambios importantes. Esta manera de trabajar permite probar novedades sin dañar la versión que ya funcionaba bien, y recuperar versiones anteriores si algo salía mal.

Dado que las pantallas están programadas todas con el mismo patrón, se va a analizar una al azar activity_5_5.xml, para observar cuál es el código que se ha empleado para la programación de las pantallas.

Observando la *figura 35*, en la línea 1 de código ya se observa como cada pantalla está configurada como un *scrollview*, es decir, el usuario deslizará con su dedo para bajar dentro de la pantalla y poder observar toda la información que contenga esta. En la línea 5 se puede observar que cada pantalla tiene una imagen como fondo de la misma. En la línea 10 se discierne que la orientación de la pantalla será vertical. Y, ¿qué signifa el código de las líneas 2, 3, 8, y 9? Tanto *match_parent* como *wrap_content* són códigos para que la pantalla

se adapte al dispositivo en cuestión.

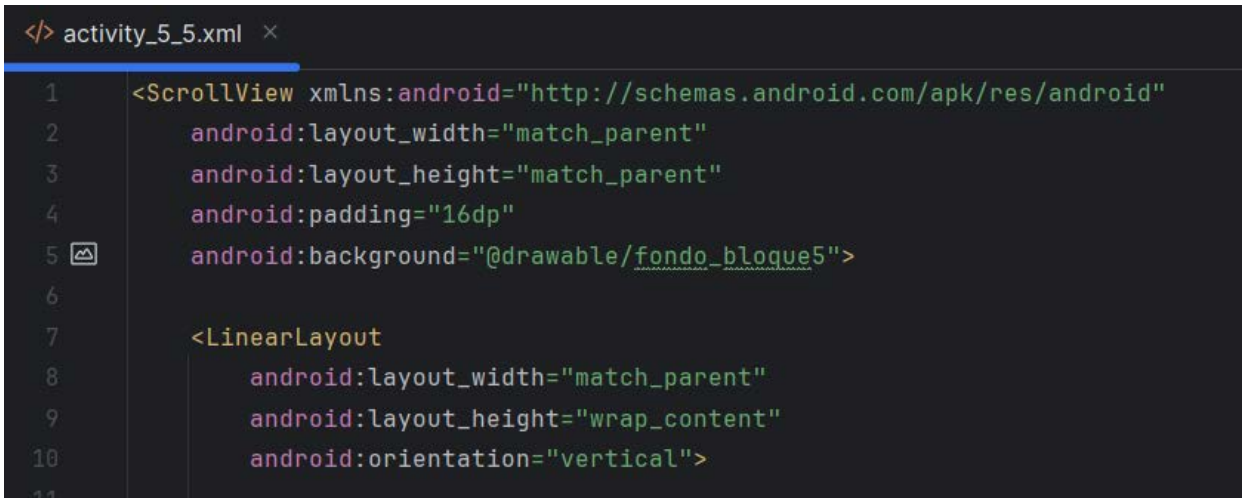

Figura 35 activity_5_5.xml - Primera parte – Android Studio

En la *figura 36* se puede apreciar que, tanto el título de la pantalla, como la explicación del temario se realizan a partir de un *TextView*. El título se encuentra en negrita gracias al código de la línea 17 "*bold*" además de tener un tamaño de letra un poco más grande con el código de la línea 18 "22sp". Estos 'sp' indican tamaño de letra, a mayor número acompañando a este código, mayor será el tamaño de letra. La letra de ambos apartados se pone de color negro gracias al código de la línea 19, siendo #000000, el código que representa el color negro (#FFFFFF por ejemplo, representa el color blanco).

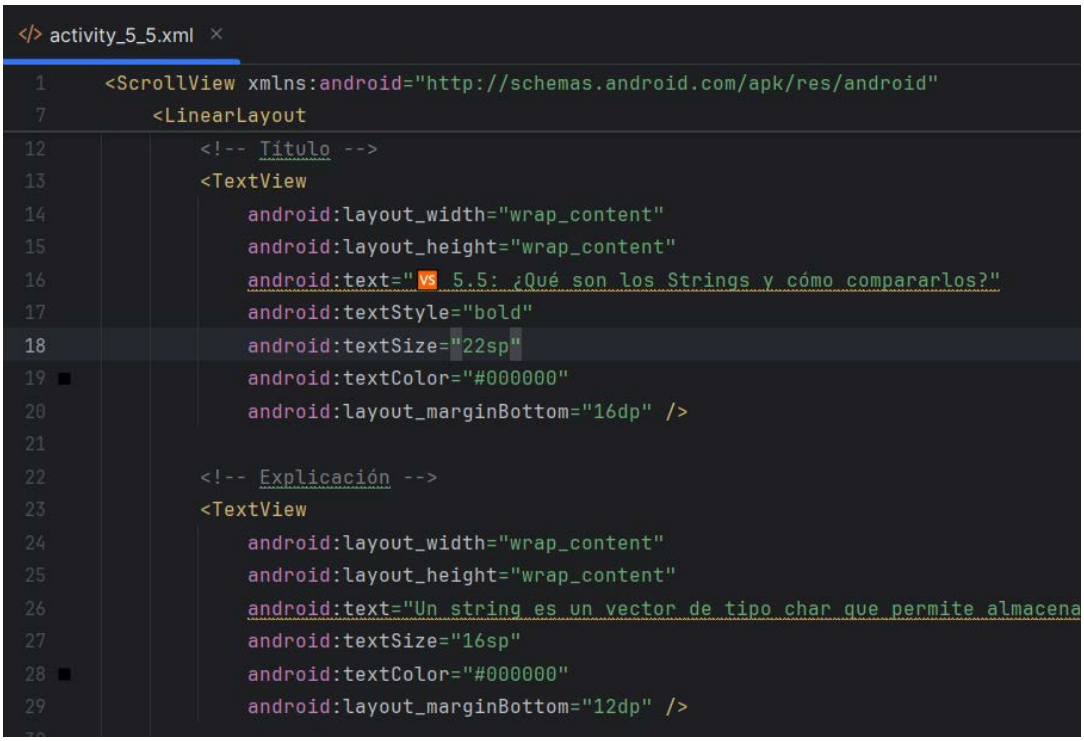

Figura 36 activity_5_5.xml - Segunda parte – Android Studio

Siguiendo con la *figura 37*, aquí se puede diferenciar la parte de la pantalla en la que se introduce el código, para darle un aspecto diferente (*figura 38*) al código respecto al resto de texto (también se emplea un *TextView),* en la línea 68 se utiliza el comando "*monospace*". El gran reto en términos de programar llega en que para que se observen de manera correcta por pantalla los símbolos especiales, muy utilizados en lenguaje C, es necesario escaparlos tal que la *tabla 4*:

Tabla 4 Escapes XML

Símbolo	Significado	Escapado en XML
"	Comillas dobles	&quot;
'	Comillas simples	&apos;
&	Ampersand (y comercial)	&amp;
<	Menor que	&lt;
>	Mayor que	&gt;

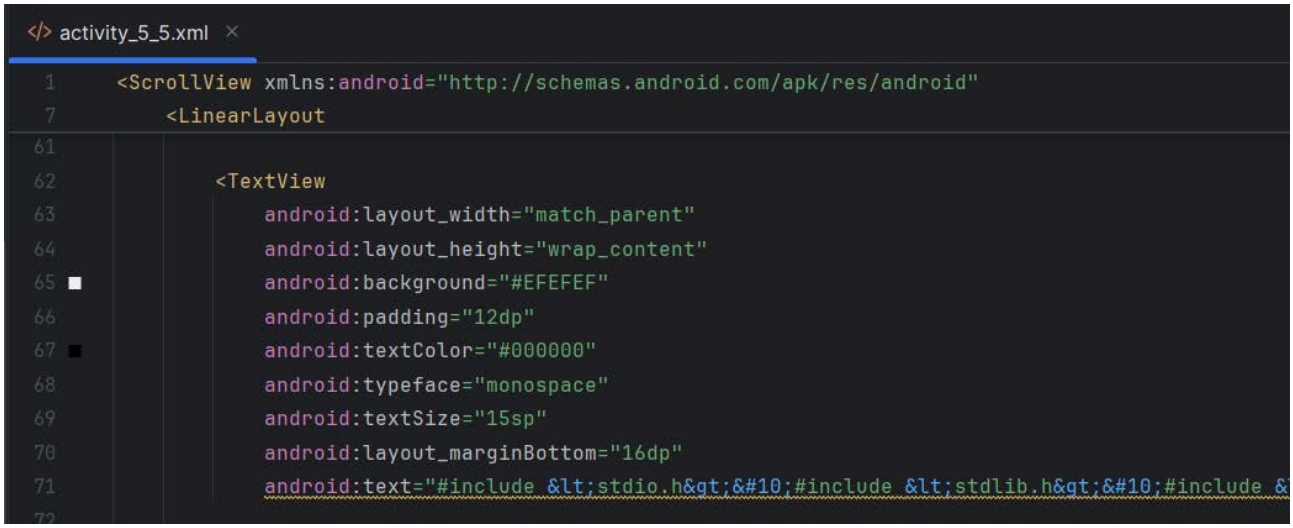

Figura 37 activity_5_5.xml - Tercera parte – Android Studio

Ejemplo 1: Lectura segura de una frase (string)

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(){
    char vector[13];
    printf("Introduce una palabra: ");
    scanf("%12s", vector);
    printf("Has introducido: %s
", vector);
    return 0;
}
```

Figura 38 Ejemplo de cómo se ve por pantalla el código en C – Sí C puede

Por último, en la pantalla representada en la *figura 39*, se observa cómo han sido configurados los botones de 'Siguiente' y 'Volver al Menú' *(figura 11)*. Destacar que es con las líneas de código 76 y 77 con las que se le da al botón un tamaño determinado, tanto de ancho (76) como de alto (77).

activity_5_5.xml

```
1   <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
7       <LinearLayout
73          <!-- Botón Volver -->
74          <Button
75              android:id="@+id/btnVolver"
76              android:layout_width="200dp"
77              android:layout_height="40dp"
78              android:background="@drawable/estilo_boton_bloque"
79              android:text="⬅ Volver al menú"
80              android:textSize="13sp"
81              android:textColor="#FFFFFF"
82              android:layout_marginBottom="8dp" />
83
84          <!-- Botón Siguiente -->
85          <Button
86              android:id="@+id/btnSiguiente"
87              android:layout_width="200dp"
88              android:layout_height="40dp"
89              android:background="@drawable/estilo_boton_bloque"
90              android:text="Siguiente ➡"
91              android:textSize="13sp"
92              android:textColor="#FFFFFF"
93              android:layout_marginBottom="24dp" />
94      </LinearLayout>
95  </ScrollView>
```

Figura 39 activity_5_5.xml - Cuarta parte – Android Studio

En resumen, todas las pantallas tienen una estructura muy similar, teniendo estas la configuración técnica y una organización del entorno de desarrollo, que ayudan a mantener un ritmo de trabajo constante, flexible y que se adapta a lo que realmente necesita una aplicación educativa ligera y útil para el usuario.

4.3. Modelo de datos y estructura de clases

La arquitectura interna de *Sí C puede* se ha diseñado con una lógica sencilla y modular, lo que ha permitido mantener el código organizado y facilitar tanto la escalabilidad como el mantenimiento futuro. Aunque no se han utilizado bases de datos externas ni estructuras complejas, se ha definido un modelo de clases funcional capaz de gestionar de forma eficiente

la navegación, la lógica de cada modo de uso y el progreso del usuario.

La estructura principal del código se organiza en torno a tres componentes clave:

1. Activities: cada pantalla de la aplicación cuenta con su propia *Activity*, lo que permite encapsular la lógica y el diseño de forma independiente. Es en estos archivos.java donde se configura todo el fuero interno de la aplicación, consiguiendo conectar todos los botones con aquello que se desea (*figura 40* como ejemplo). Entre ellas destacan:

    - *MainActivity*: punto de entrada y selector de modo.
    - *AprendizajeProgresivoActivity*: gestiona el listado de niveles y el marcado como aprendido.
    - *DesafiosActivity*: muestra los retos disponibles.

En la *figura 40* observamos cómo funciona el funcionamiento de los botones, este sería el ejemplo en el que el botón de la pantalla principal de aprendizaje progresivo te lleva a su pantalla correspondiente.

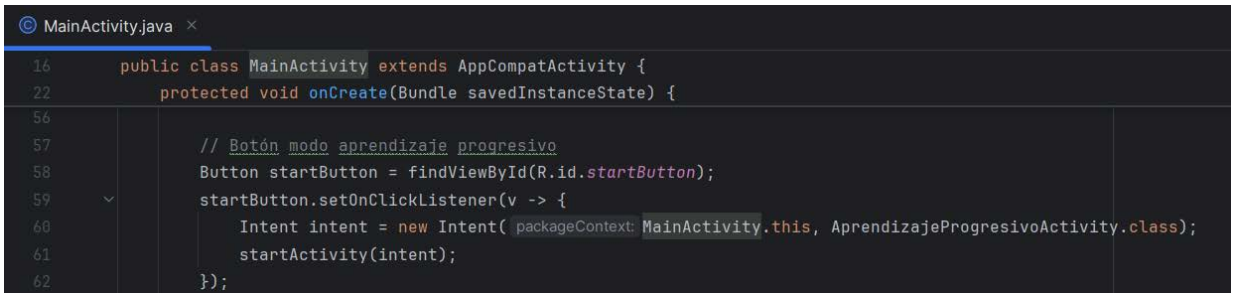

Figura 40 MainActivity.java - Conexión de botones – Android Studio

Otra parte de código interesante es el que se observa en la *figura 41*, donde vemos el código para que al tocar el icono de música que hay en la pantalla principal, la música se encienda o apague, dependiendo del estado en el que esté.

```java
MainActivity.java
16      public class MainActivity extends AppCompatActivity {
22          protected void onCreate(Bundle savedInstanceState) {
32
33              // Iniciar música
34              mediaPlayer = MediaPlayer.create( context: this, R.raw.musica_relax);
35              mediaPlayer.setLooping(true); // para que suene siempre
36              mediaPlayer.start();
37
38              // Botón para pausar/reanudar
39              ImageButton musicButton = findViewById(R.id.musicButton);
40              musicButton.setImageResource(R.drawable.ic_music_on); // Por defecto
41
42              musicButton.setOnClickListener(v -> {
43                  if (isPlaying) {
44                      mediaPlayer.pause();
45                      isPlaying = false;
46                      musicButton.setImageResource(R.drawable.ic_music_off); // icono tachado
47                  } else {
48                      mediaPlayer.start();
49                      isPlaying = true;
50                      musicButton.setImageResource(R.drawable.ic_music_on); // icono normal
51                  }
52              });
```

Figura 41 MainActivity.java - Configuración de la música – Android Studio

2. Clases auxiliares: se han definido clases de apoyo para tareas recurrentes, como la gestión del estado de los bloques completados o la carga dinámica del contenido de los desafíos. Una de las más relevantes es la clase *(figura 42)* que almacena de forma local los niveles marcados como aprendidos y permite consultarlos desde el desplegable.

```java
MainActivity.java
16      public class MainActivity extends AppCompatActivity {
22          protected void onCreate(Bundle savedInstanceState) {
70
71              // Botón de progreso con alerta
72              ImageButton btnProgreso = findViewById(R.id.btnProgreso);
73              btnProgreso.setOnClickListener(v -> {
74                  StringBuilder mensaje = new StringBuilder("📘 Bloques aprendidos:\n\n");
75                  for (int i = 1; i <= 8; i++) {
76                      boolean aprendido = prefs.getBoolean( s: "bloque" + i + "_aprendido",  b: false);
77                      if (aprendido) {
78                          mensaje.append("✓ Bloque ").append(i).append("\n");
79                      }
80                  }
81
82                  if (mensaje.toString().equals("📘 Bloques aprendidos:\n\n")) {
83                      mensaje.append("¡Aún no has marcado ningún bloque como aprendido!");
84                  }
85
86                  new androidx.appcompat.app.AlertDialog.Builder( context: MainActivity.this)
87                          .setTitle("Tu progreso 📊")
88                          .setMessage(mensaje.toString())
89                          .setPositiveButton( text: "Cerrar",  listener: null)
90                          .show();
91              });
```

Figura 42 MainActivity.java - Configuración aprendizaje bloques – Android Studio

3. *Archivos de recursos*: explicados en el 4.3 a fondo, si bien no son clases en sí, los archivos XML asociados a cada *Activity* forman parte esencial del modelo de datos visual. En ellos se definen los elementos gráficos (botones, listas, textos) y su comportamiento básico.

El enfoque adoptado busca mantener una separación clara entre lógica y presentación, facilitando la lectura del código y la incorporación de mejoras futuras. Aunque el almacenamiento se realiza en variables locales y archivos internos del sistema, el modelo actual permite extender fácilmente la aplicación a un sistema de persistencia más avanzado si fuera necesario.

4.4. Pruebas funcionales en distintos dispositivos

Una vez implementadas las funcionalidades principales de *Sí C puede*, se han realizado pruebas funcionales en varios dispositivos, en este caso los dispositivos de los miembros de mi familia, con el objetivo de comprobar su correcto comportamiento en diferentes entornos y configuraciones. Estas pruebas han permitido validar la estabilidad de la aplicación, además de detectar posibles inconsistencias relacionadas con la interfaz, la navegación o la compatibilidad entre versiones de *Android*.

Los dispositivos utilizados durante la fase de pruebas incluyeron modelos con características variadas en cuanto a resolución de pantalla, versión del sistema operativo y capacidad de procesamiento. Entre ellos se encuentran:

- Xiaomi Redmi Note 10 Pro (Android 12)
- Samsung Galaxy A32 (Android 13)
- Xiaomi Mi 10 Lite (Android 11)
- Emulador Pixel 4a (Android 10 y Android 14)

En la *figura 43* se puede observar cómo hay pequeñas diferencias, por ejemplo, en los márgenes, pero todas las funcionalidades de la aplicación se siguen manteniendo activas, siendo estas:

- Acceso correcto a los modos de ‘aprendizaje progresivo’ y ‘desafíos’ desde el menú principal.
- Visualización completa de los bloques temáticos y retos en pantallas de diferentes

tamaños.

- Funcionamiento estable del botón ‘Marcar como aprendido’ y registro adecuado en el desplegable de progreso.
- Navegación fluida entre pantallas y respuesta inmediata de los botones.
- Correcta adaptación del contenido textual y gráfico según la densidad de píxeles y orientación del dispositivo.

Figura 43 Pantalla principal de la aplicación en 3 dispositivos distintos – Sí C puede

El resultado general de las pruebas fue positivo. La aplicación se comportó de forma estable y consistente en todos los dispositivos, sin cierres inesperados ni problemas críticos. En casos puntuales, se ajustaron márgenes y tamaños de texto para mejorar la visualización en dispositivos con pantallas más pequeñas, pero no ha sido necesario modificar la lógica central del programa.

Estas pruebas han contribuido a reforzar la robustez de la aplicación y garantizaron una experiencia homogénea independientemente del dispositivo utilizado, lo cual resulta fundamental para su implementación en entornos educativos reales.

4.5. Firma, empaquetado y publicación

Una vez finalizada la implementación de *Sí C puede* y verificadas sus funcionalidades mediante pruebas funcionales, el siguiente paso fue preparar la aplicación para su distribución. Este proceso incluye tres fases clave: la firma digital del archivo ejecutable, el empaquetado del proyecto en formato APK y la elección de una plataforma de publicación accesible y adecuada al tipo de app**.**

4.5.1. Firma de la aplicación

Para poder instalar una aplicación Android en cualquier dispositivo fuera del entorno de desarrollo, es necesario firmarla digitalmente. Esta firma garantiza que el archivo no ha sido modificado desde su creación y que proviene de una fuente reconocible. En este caso, se generó un archivo *keystore (figura 44)* a través de *Android Studio*, configurando una clave privada específica para la versión de producción.

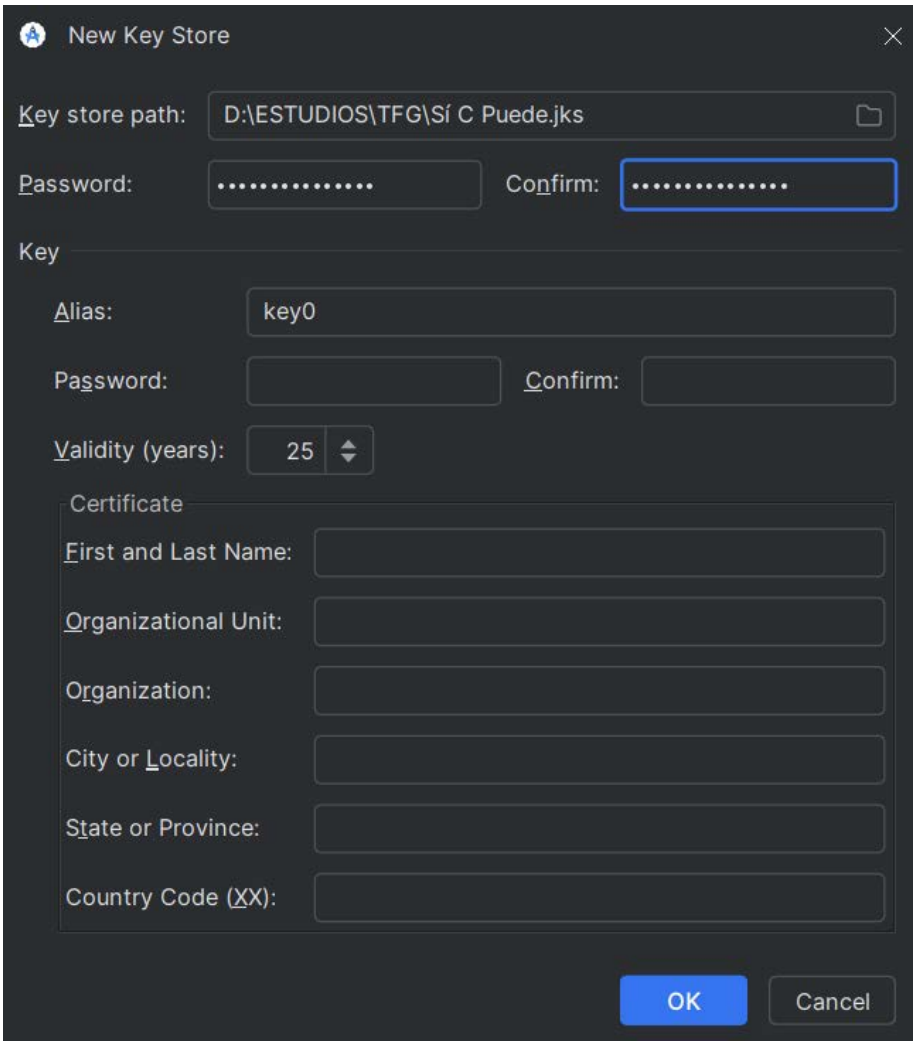

Figura 44 Pantalla para la creación de un archivo keystore – Android Studio

La firma se ha integrado automáticamente durante el proceso de generación del APK, tal y como exige el sistema operativo Android para permitir la instalación en terminales físicos. Esta práctica es esencial no solo para la distribución, sino también para futuras actualizaciones, que solo se validan si provienen del mismo certificado.

4.5.2. Empaquetado en formato APK

Tras firmar la aplicación, se ha procedido al empaquetado en formato *APK* (*Android Package*), que es el tipo de archivo utilizado por los dispositivos Android para instalar aplicaciones. El paquete incluye todos los recursos, clases compiladas y archivos de configuración necesarios para su ejecución.

El proceso se he realizado desde la propia herramienta *Android Studio*, seleccionando la opción *Generate Signed Bundle / APK*. Se ha elegido la variante de *release* para asegurar un rendimiento óptimo y eliminar elementos de depuración. La versión final del APK *(figura 45)* fue probada manualmente en los dispositivos previamente evaluados, verificando su instalación y correcto funcionamiento.

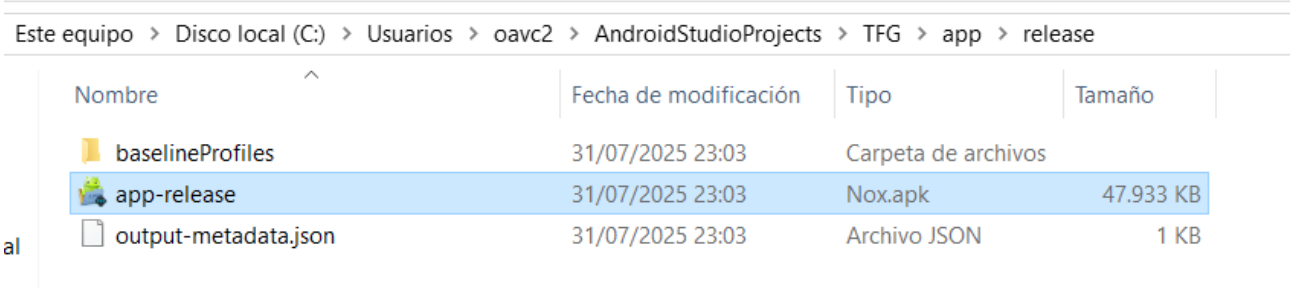

Figura 45 APK final en el dispositivo

4.5.3. Publicación y distribución

Aunque inicialmente se valoró publicar la aplicación en Google Play, como ya se había comentado, se opta en un principio por una alternativa más ágil y abierta: *Uptodown (figura 46)*, una tienda de aplicaciones independiente con sede en España. Esta plataforma permite subir aplicaciones sin coste, sin necesidad de cuenta de desarrollador y con la posibilidad de descarga directa de archivos APK, lo que resulta especialmente útil en contextos educativos o sin conexión estable a internet.

Además, *Uptodown* ofrece visibilidad internacional y estadísticas de descarga, sin imponer los estrictos requisitos de publicación de otras plataformas. Esta opción resulta ideal para una aplicación educativa como *Sí C puede*, que no incluye compras, anuncios ni funcionalidades comerciales.

Figura 46 Logo uptodown

4.6. Escalabilidad y modularidad

Uno de los principios clave que ha guiado el desarrollo de *Sí C puede* ha sido la búsqueda de una estructura que permita crecer en un futura a la aplicación sin tener que comprometer la estabilidad del sistema. Por ello, desde las primeras fases de diseño, se ha ido optando por una arquitectura modular y escalable**,** en la que cada componente de la aplicación está separado según su función y puede evolucionar de forma independiente.

En términos de organización interna, cada bloque funcional ya sea un nivel del modo progresivo, un desafío o una pantalla específica, se construyó como una entidad aislada. Esto significa que añadir un nuevo nivel de aprendizaje o incorporar un nuevo reto no requiere modificar el núcleo del proyecto, sino simplemente extender las clases y los archivos XML correspondientes. Esta estrategia de diseño evita conflictos y facilita el mantenimiento a largo plazo.

Además, se han seguido buenas prácticas en la organización del código, separando la lógica de negocio de la presentación visual. Las actividades principales (*Activities*) se encargan únicamente de gestionar la interacción con el usuario, mientras que la carga de datos y el control del progreso se realizan desde clases auxiliares. Esto permite que futuras funcionalidades, como podrían ser estadísticas, sistemas de ayuda o persistencia remota, puedan integrarse sin alterar el comportamiento de las pantallas ya existentes.

Gracias a esta arquitectura, *Sí C puede* no es solo una aplicación cerrada, sino una base desde la que construir nuevas funcionalidades educativas.

4.7. Adaptabilidad a distintas resoluciones

Una de las decisiones más importantes que se tomaron al desarrollar una aplicación móvil es la de pensar en cómo se va a comportar en pantallas muy distintas entre sí. Hoy en día, hay dispositivos con resoluciones enormes, otros con pantallas más pequeñas, densidades muy variadas, e incluso formatos alargados o con recortes. Ignorar eso supone

un gran riesgo, ya que lo que funciona bien en un móvil puede que no se vea correctamente en otro.

Para evitar ese problema, en *Sí C puede* se ha cuidado desde el principio la adaptabilidad. Toda la interfaz se ha construido utilizando unidades relativas como *dp* (píxeles independientes de la densidad) y *sp* (píxeles escalables para texto). Esto asegura que los márgenes, tamaños y tipografías se ajusten automáticamente al dispositivo donde se está ejecutando, sin que el usuario tenga que hacer nada fuera de lo normal. En el apartado 4.4. ya se trató el tema del código y las determinaciones que se tomaron.

También se han utilizado contenedores flexibles en el diseño de cada pantalla, como *LinearLayout, ConstraintLayout* y *ScrollView,* que ayudan a que los elementos fluyan de forma ordenada incluso cuando el espacio disponible varía mucho. Gracias a eso, las pantallas de la aplicación se ven limpias y equilibradas tanto en móviles básicos como en terminales más potentes con pantallas más alargadas.

Durante las pruebas funcionales realizadas con distintos dispositivos (ver apartado 4.5), se comprobaron las diferentes combinaciones de tamaño y resolución. Desde pantallas con 720p hasta modelos que tienen la capacidad de superar los 2400x1080, todas las vistas mostrando un comportamiento estable. En algunos casos sí que fue necesario hacer pequeños ajustes en los márgenes o en el tamaño del texto, pero siempre sin romper la estructura general ni afectar la usabilidad, como ya se trató en dicho apartado.

Esta flexibilidad no es simplemente una cuestión estética, sino que es una forma concreta de hacer que la aplicación llegue a un público de mayor tamaño, especialmente en aquellos entornos educativos donde los recursos son limitados y no todos los usuarios tienen acceso al mismo tipo de terminal, pudiendo a partir del desarrollo de la aplicación homogeneizar las experiencias de los usuarios.

5. Análisis de resultados

Una vez que la aplicación está terminada y se ha comprobado que es funcional, es el momento de evaluar todo el global del proyecto. En este apartado, se junta todo lo aprendido mientras se ha testado la aplicación, además de lo que visto cuando *Sí C puede* se utilizó en diferentes dispositivos.

Lo más importante es determinar si la aplicación cumple con los objetivos marcados. La evaluación se centra en determinar si la aplicación no colapsa, si funciona de forma fluida

al moverse por todas sus interfaces, si es fácil entenderla y si el sistema para ver observar el progreso también funciona.

También, es clave discernir si la aplicación vale para personas en distintos puntos en cuanto aprendizaje de programación en C.

Así que, este apartado es un enlace entre cómo la idea inicial y el resultado final, definiéndose todo lo alcanzado, así como los siguientes retos por alcanzar.

5.1. Evaluación de objetivos alcanzados

Al inicio del proyecto, se fijaron varios objetivos clave que guiarían el desarrollo de *Sí C puede*. Estos objetivos abarcaban tanto aspectos técnicos como la experiencia del usuario, la facilidad para aprender y la accesibilidad del material. Este segmento presenta un análisis detallado de qué tanto se han alcanzado estas metas.

La meta primordial era tener una aplicación Android plenamente operativa que brindara una navegación ágil, un diseño nítido y una lógica de uso fácil de entender. En este sentido, la aplicación ha mostrado un desempeño sólido en los diversos dispositivos donde se ha evaluado (ver capítulo 4.5), sin bloqueos repentinos ni errores graves. Se puede acceder a todas las pantallas desde el menú principal, y el cambio entre niveles o retos se efectúa de manera sencilla y sin apenas demora.

También se pretendía que la aplicación fuera fácil de entender para usuarios con diferentes niveles de manejo de dispositivos Android, algo que se ha logrado al usar botones que se explican por sí solos y un orden visual bien definido. Siempre se ha dado prioridad a que el diseño no solo sea llamativo, sino también práctico y directo.

Otro de los objetivos principales era potenciar la motivación del alumno por medio de la gamificación, sin recurrir a adornos innecesarios. El sistema de registro del progreso, junto con los retos temáticos y la muestra visual del aprendizaje, ha permitido brindar una experiencia activa y flexible, sin causar tensión en el usuario con un sistema de puntuación. El objetivo no es enseñar rápido, el objetivo es enseñar.

Finalmente, se planteó la necesidad de que la aplicación fuera liviana (en términos de almacenaje en el dispositivo), accesible y fácil de distribuir, en especial en entornos educativos donde los recursos tecnológicos pueden ser escasos. Gracias a su tamaño reducido, su funcionamiento sin conexión y la opción de distribuirla a través de plataformas abiertas como *Uptodown*, este objetivo también se puede considerar logrado.

Figura 47 Resumen visual de los objetivos alcanzados

En resumen *(figura 47)*, la aplicación ha respondido de forma eficiente a los desafíos planteados en su fase inicial. Aunque aún hay espacio para mejorar y evolucionar, la versión actual de *Sí C puede* cumple con creces su propósito educativo, demostrando que es factible aprender a programar en C de manera organizada, autónoma e inspiradora.

5.2. Resultados de las pruebas técnicas

Las pruebas funcionales realizadas durante el desarrollo de *Sí C puede* no solo tenían como objetivo comprobar si cada apartado “se abría” o respondía correctamente, sino también observar el comportamiento de la aplicación en situaciones reales de uso, con diferentes dispositivos, resoluciones y versiones de *Android*. Este apartado recoge los principales resultados obtenidos, agrupados por tipo de prueba y centrados en aspectos clave como la estabilidad, la compatibilidad y la fluidez de navegación.

En conjunto, los resultados técnicos *(tabla 5)* validan la estabilidad, compatibilidad y eficiencia de *Sí C puede*. Aunque la aplicación es sencilla en cuanto a estructura, su diseño consciente y su atención a los detalles han permitido alcanzar un nivel de madurez técnica muy sólido, acorde con los objetivos planteados desde el inicio del proyecto.

Tabla 5 Principales resultados obtenidos a través de las pruebas

ESTABILIDAD	Sin cierres ni errores	Comportamiento estable
COMPATIBIBLIDAD	Android 10 a 14	Totalmente funcional
ADAPTACIÓN VISUAL	Correcta	Ajustes mínimos para baja resolución
TIEMPO DE CARGA	< 1 segundo	Respuesta inmediata al interactuar
CONSUMO	Bajo	No requiere conexión a Internet

5.2.1. Estabilidad de la aplicación

En todas las sesiones de prueba, la aplicación mostró un comportamiento estable, sin cierres inesperados ni bloqueos. Las actividades se abren de forma fluida y la navegación entre pantallas mantiene un rendimiento constante, incluso en dispositivos de gama media y baja. No se detectaron fugas de memoria ni errores críticos durante la ejecución, lo que confirma que la arquitectura modular adoptada ha favorecido la robustez del sistema.

5.2.2. Compatibilidad entre versiones de Android

En dispositivos con resoluciones altas, la aplicación mantuvo su estética limpia, sin elementos desalineados o superpuestos. En terminales con menor densidad de píxeles, el texto continuó siendo legible y los botones permanecieron accesibles, aunque se realizaron ajustes menores en márgenes y *paddings* tras las primeras pruebas. El uso de unidades relativas y contenedores flexibles resultó clave para esta adaptabilidad (ver capítulo 4.8).

5.2.3. Rendimiento visual y adaptación

Todas las pantallas cargan en menos de un segundo, y la respuesta a las interacciones del usuario (tocar botones, marcar como aprendido, cambiar de modo) es inmediata. No se observaron lags ni retardos que afectaran a la experiencia de uso. La aplicación, además, no requiere conexión a internet ni consume recursos en segundo plano, lo que mejora su eficiencia energética.

5.3. Comparación con otras soluciones del mercado

En el capítulo dos, ya se exploraron varias plataformas de aprendizaje que usan la gamificación, como *Duolingo*, *CodeCombat*, *Kahoot* o *Grasshopper (figura 7)*, todas pensadas para ser divertidas y educativas. Este apartado vuelve a ese punto de partida para comparar directamente a *Sí C puede* con algunas de ellas, viendo qué ofrece esta aplicación en el mundo actual de herramientas digitales para aprender a programar.

Para empezar, a diferencia de plataformas como *CodeCombat* o *Grasshopper*, que suelen usar *JavaScript* o *Python*, *Sí C puede* se centra solo en el lenguaje C (lenguaje impartido en el plan de estudios de la ETSII), que es estructurado y difícil para los que empiezan, pero muy apreciado en la formación técnica. Este enfoque le da un toque más especializado y lo coloca en un lugar poco atendido por otras aplicaciones de tipo lúdico.

Otro punto importante es que *Sí C puede* no necesita internet ni depende de servidores de fuera para funcionar. A diferencia de *Duolingo* o *Kahoot*, que usan una plataforma centralizada, esta *aplicación* se instala en el dispositivo, lo que la hace muy útil en sitios educativos con mala conexión o en sesiones de estudio individuales fuera de clase.

Sobre la experiencia del usuario, *Sí C puede* tiene cosas en común con las otras herramientas, como niveles de avance, retos individuales y ver cómo se progresa, pero evita la competición directa. Esta decisión va con un enfoque más personal, donde el progreso es una motivación y no algo que presiona.

En resumen, aunque no busca competir con las plataformas grandes, *Sí C puede* tiene su propio espacio gracias a su sencillez, especialización y enfoque local. Lo valioso no es que sea más potente o global, sino que se adapta mejor a situaciones formativas específicas, donde el apoyo al aprendizaje y la flexibilidad son más importantes que lo visual o la popularidad.

6. Conclusiones

Este apartado engloba los conocimientos más cruciales adquiridos al llevar a cabo *Sí C puede*, abarcando tanto los aspectos técnicos como los enfoques pedagógicos y metodológicos. Tras examinar el panorama, idear y crear la aplicación, y ponerla a prueba en situaciones reales, se obtiene una perspectiva clara del efecto y los desafíos que el proyecto ha supuesto.

En las siguientes líneas, se resumen los principales logros alcanzados, se señalan las carencias detectadas y se detallan las contribuciones que el proyecto brinda tanto en el plano académico como en el técnico.

6.1. Síntesis de los resultados

Tras el desarrollo de *Sí C puede*, los resultados que hemos visto nos dejan decir que cumplimos de sobra los objetivos que marcamos en el capítulo 1.3. La aplicación va bien en

distintos dispositivos *Android*, algo que vimos en el capítulo *5.2*, y cumple con lo que esperábamos en cuanto a funciones y a lo que necesitábamos para enseñar, según el análisis que hicimos antes (capítulo 2).

La forma en que se diseñó la *aplicación*, con partes separadas y una estructura por bloques (capítulos 3 y 4), hizo que sea fácil de entender, que sea visual y que cualquiera pueda utilizarla y comprenderla. Al separar claramente los modos de *Aprendizaje progresivo* y *Desafíos* (mira la *figura 9*), cada usuario decide qué camino seguir, aprendiendo a su propio ritmo y como más le guste. Esto va de la mano con lo tratado sobre aprender de una manera lúdica en el capítulo 2.4, donde se dijo que es clave que cada uno pueda elegir y ver sobre la marcha si va bien su progreso o requiere de más trabajo.

Además, la manera en que uno se mueve por la aplicación (capítulo 3.4) y la opción de marcar qué bloques ya aprendió (*figura 11*) hacen que no tengas que seguir un camino fijo, sino que tú decides cómo avanzar. Esto te anima más, como explicamos al hablar de cómo los juegos ayudan a aprender cosas técnicas.

En lo técnico, usamos recursos como separar bien lo que se ve de lo que hace la aplicación, hacer que se vea bien en distintas pantallas (capítulo 4.8) y que funcione en versiones de *Android* desde la *API 26* (capítulo 4.2). Esto hizo que la aplicación sea rápida, se adapte bien y gaste pocos recursos. Y encima, no se requiere internet para usarla, que es un factor diferencial respecto a otras aplicaciones (capítulo 5.3).

Para cerrar, el proyecto *Sí C puede* no solo alcanzó el objetivo inicial, sino que ahora es una opción más en el mundo de las *aplicaciones* para aprender, porque es fácil de usar y piensa en el usuario. Su diseño simplista, cómo está concebida y cómo te enseña responden a todas aquellas necesidades que se definían al principio, comprobándose con las pruebas que realizadas al analizar los resultados.

6.2. Limitaciones encontradas

Aunque *Sí C puede* ha alcanzado los objetivos propuestos, el desarrollo y evaluación del proyecto ha permitido identificar una serie de limitaciones que es importante reconocer. Estas no restan valor al producto final, pero sí ayudan a trazar con mayor precisión las futuras líneas de mejora.

La primera limitación viene dada por el enfoque estrictamente local del almacenamiento de progreso. Si bien esta decisión permite trabajar sin conexión y respetar la privacidad del usuario, también restringe las posibilidades de sincronización entre dispositivos

o de recuperación del avance en caso de desinstalación. En un entorno educativo más amplio, donde se utilicen diferentes terminales o se trabaje en conjunto con plataformas académicas, esta limitación puede suponer un obstáculo.

Otro aspecto que considerar es que, al no integrar un compilador dentro de la aplicación, el aprendizaje se basa únicamente en la lectura y comprensión de código, sin permitir ejecutar o testear directamente los programas. Esta elección fue consciente, como se explicó en capítulos anteriores, para priorizar la comprensión teórica antes de lanzarse al ensayo-error. Sin embargo, puede hacer que algunos usuarios echen en falta una experiencia más interactiva o práctica, especialmente quienes estén acostumbrados a entornos como *CodeCombat* o *Grasshopper*.

También se detectaron pequeñas diferencias en la visualización de la interfaz según el dispositivo utilizado (ver capítulo 4.5). Aunque no afectaron al funcionamiento general, sí implicaron ajustes adicionales en márgenes, tipografías y distribución de botones. Esto evidencia la necesidad de seguir afinando la adaptabilidad de la aplicación para ofrecer una experiencia verdaderamente homogénea en cualquier terminal Android.

Desde el punto de vista técnico, otro límite importante es la ausencia de un sistema de análisis estadístico que permita recoger datos de uso reales. Dado que la aplicación no recopila información del usuario, por respeto a su privacidad, no se puede saber con precisión qué niveles se visitan más, dónde abandonan los usuarios o qué retos presentan mayores dificultades. Esta información sería muy útil para optimizar la experiencia de aprendizaje, revisar niveles y subir la aplicación al siguiente nivel.

Por último, no se debe obviar el hecho de que la aplicación ha sido testeada con un número limitado de personas, muchas de ellas cercanas a mi entorno. La gran prueba de fuego sería que los nuevos alumnos que vayan a cursar la asignatura de *Fundamentos de Programación* testearan la aplicación, en contextos reales de clase, para obtener un diagnóstico más completo de su impacto educativo.

En conjunto, estas limitaciones no suponen fallos estructurales del proyecto, pero sí abren la puerta a futuras versiones más robustas, interactivas y adaptadas a distintos entornos de uso.

6.3. Aportaciones del proyecto

Para empezar, el proyecto brinda una opción educativa hecha a medida para aprender el lenguaje C, un área donde hay muchos recursos clásicos, pero faltan herramientas con

juegos adaptadas a móviles. A diferencia de otros lenguajes más comunes en el mundo de los juegos como *Python* o *JavaScript*, C necesita una manera de aprender más ordenada y paso a paso, algo que la aplicación soluciona separando la teoría de los retos prácticos.

Además, la aplicación añade un modelo de diseño que se puede adaptar y dividir en partes, fácil de copiar en otros temarios. Como se explica en el capítulo *4*, cada parte puede funcionar por sí sola, lo que hace más fácil usarla de nuevo o hacerla más grande. Este diseño no solo permite poner al día los temas sin tocar el resto de la aplicación, sino que también da pie a conectar con compiladores, sistemas de evaluación que se hacen solos u otras tecnologías que ayudan a aprender.

Visto desde la forma de enseñar, la aplicación mete una forma de aprender que puede o no puede ser lineal, en la que el estudiante puede elegir su camino, avanzar cuando quiera y marcar las partes como aprendidas sin tener que seguir un orden fijo. Esta idea, que se basa en las ideas de los juegos para aprender que se ven en el capítulo *2*, ayuda a que uno sea más independiente, se evalúe a sí mismo y aprenda a su manera.

También es importante que *Sí C puede* funcione sin necesidad de estar conectado a *internet*, lo que la hace muy útil en sitios donde no hay buena conexión.

Por último, el proyecto ha demostrado ser una gran oportunidad para juntar conocimientos de distintas áreas, combinando el diseño de interfaces, la estructura de datos, el desarrollo en *Android Studio* y las bases de la enseñanza.

7. Líneas futuras de desarrollo

A pesar de que *Sí C puede* cumplió con las metas fijadas en este proyecto, siempre hay margen para mejorar en cualquier iniciativa tecnológica, así como para crecer y ajustarse a diferentes ambientes de enseñanza. En este apartado, se juntan varias ideas clave para el desarrollo a futuro, identificadas gracias a lo aprendido y a una evaluación detallada de los hallazgos.

Estas sugerencias no solo harían que la aplicación fuera más útil y llegara a más gente, sino que también atenderían a las necesidades de capacitación, de las instituciones y de la tecnología que van surgiendo. Unas tienen que ver con detalles técnicos que no se han implementado en esta primera versión, mientras que otras buscan una mejor integración en la enseñanza o un aumento en la cantidad de usuarios.

7.1. Funcionalidades pendientes de implementar

Durante el desarrollo de *Sí C puede*, se priorizó un diseño funcional, ligero y centrado en lo esencial para garantizar una experiencia de aprendizaje clara y directa. No obstante, quedaron identificadas varias funcionalidades que podrían enriquecer considerablemente la aplicación en futuras versiones. Para desarrollar este apartado (ver capítulo 6.2) el foco se centra en las limitaciones actuales que tiene la aplicación.

Una de las más destacadas es la posibilidad de incluir un sistema de ejecución de código dentro de la propia aplicación. Esta funcionalidad, aunque compleja, permitiría que el usuario no solo lea y analice ejemplos de programación, sino que también pueda escribir, compilar y ejecutar fragmentos de código en lenguaje C, todo desde el mismo entorno.

También se contempla la integración de notificaciones internas, que puedan recordar al usuario continuar con el aprendizaje o indicarle nuevos contenidos o mejoras de la aplicación disponibles. Esto contribuiría a generar un hábito y facilitaría el retorno periódico a la aplicación, un factor clave en procesos de aprendizaje autónomo.

Por otro lado, aunque el seguimiento del progreso ya está implementado mediante un sistema de bloques marcados como aprendidos (ver figura *11*), se valora la incorporación de informes visuales de avance, como estadísticas personales. Esta función podría mejorar la percepción del progreso real y aumentar la motivación.

Finalmente, se detectó la necesidad de implementar un sistema que permita guardar una copia de seguridad del avance, lo cual facilitaría la reinstalación de la aplicación incluso en otro dispositivo sin perder el progreso acumulado, algo especialmente útil en entornos educativos donde los dispositivos se renuevan con frecuencia. Tal y como se encuentra la aplicación, esta sería la facultad menos necesaria, aunque ante un posible desarrollo de la aplicación podría suponer un gran avance.

7.2. Ampliación a otros lenguajes de programación

Si bien *Sí C puede* se creó pensando justo en hacer más fácil aprender C, su diseño por partes y lo bien pensada que está para enseñar, hacen que uno se imagine que tranquilamente se podría usar para otros lenguajes.

Python y *JavaScript* son los lenguajes más evidentes, ya que las empresas los solicitan a sus empleados. Para ambos lenguajes se podría usar el mismo sistema de juego con explicaciones y retos para que los estudiantes entiendan bien lo básico del código.

Como la aplicación está hecha por módulos, no es necesario hacerla de nuevo desde cero, sino que se pueden crear versiones parecidas que tengan el mismo sistema para moverse, ver el progreso y hacer seguimiento. Solo habría que cambiar lo que dice el texto, los ejercicios y los ejemplos de código, adaptándolos a cómo se escribe y piensa en el lenguaje nuevo.

Además, añadir nuevos lenguajes haría que los usuarios puedan seguir aprendiendo más. Por ejemplo, después de tener una buena base en C, el estudiante podría usar otra versión de la aplicación enfocada en programación con objetos, creación de páginas web o programas automáticos. Así, lo que aprenden no se queda solo en un lenguaje, sino que se convierte en un camino de aprendizaje más completo y con sentido.

Esta mejora también permitiría que la aplicación la usara más gente en el mundo de la educación, desde escuelas de formación profesional hasta carreras universitarias técnicas e incluso cursos para personas mayores.

7.3. Integración con entornos visuales de aprendizaje (LMS)

Una de las líneas con más potencial para la evolución de *Sí C puede* es su posible integración con plataformas de gestión del aprendizaje, conocidas como *Learning Management Systems* (*LMS*). Herramientas como *Moodle*, *Google Classroom* o *Canvas* son ampliamente utilizadas en el ámbito educativo para organizar cursos, distribuir contenidos, evaluar y hacer seguimiento del alumnado, con todo esto en cuenta, salta a la vista que una integración así daría un gran impulso a la aplicación.

Vincular *Sí C puede* a uno de estos entornos permitiría ampliar notablemente su alcance y funcionalidad, ya que facilitaría la sincronización del progreso del usuario con el perfil del estudiante en el aula virtual, así como el seguimiento docente de los avances realizados, todo desde un entorno conocido por profesorado y alumnado.

Esta integración podría realizarse mediante la creación de módulos específicos para LMS que se conectaran con la aplicación, permitiendo, por ejemplo:

- Registrar automáticamente los bloques aprendidos y los desafíos completados.
- Generar informes de actividad y progreso exportables a la plataforma.
- Proporcionar enlaces directos a desafíos específicos desde el propio curso virtual.
- Activar funcionalidades de gamificación compartida, como rankings internos donde el

usuario pueda ver el progreso de los demás usuarios o logros por clase o grupos de alumnos.

Además, al incorporar compatibilidad con estos entornos, la aplicación ganaría en valor institucional, ya que se adaptaría fácilmente a los sistemas ya implantados en escuelas, universidades o centros de formación. Esta sinergia podría facilitar su adopción oficial por parte de entidades educativas, algo que reforzaría la estabilidad y sostenibilidad del proyecto.

A nivel técnico, esta integración requeriría definir protocolos de comunicación entre la aplicación móvil y el LMS correspondiente, ya sea a través de *APIs*, archivos exportables o sincronización por usuario. Aunque implica cierta complejidad, es una vía viable y con alto potencial educativo.

7.4. Posibilidades de implementación institucional

Como se comentó en el apartado anterior, el potencial para ser adoptado dentro de entornos educativos formales, tanto en institutos de educación secundaria como en ciclos formativos o grados universitarios relacionados con la programación y la informática es lo que más destaca de *Sí C puede.*

Gracias a su enfoque autónomo, que no depende de conexión a internet ni de servidores externos, permite que el alumnado pueda usar la herramienta directamente desde sus propios dispositivos, sin necesidad de una red institucional compleja ni de licencias de software muy caras.

Además, la estructura modular de la aplicación facilita su adaptación curricular. Cada bloque temático puede ser asignado como parte del temario oficial, permitiendo al profesorado decidir qué secciones trabajar en clase, cuáles dejar como refuerzo y qué desafíos pueden utilizarse como ejercicios prácticos o incluso como pruebas formativas.

Por otro lado, si se implementan las mejoras descritas en los apartados anteriores como la sincronización con *LMS* o los informes de progreso, la aplicación podría convertirse en un recurso central dentro de programas de formación básica en programación. Esto permitiría a las instituciones evaluar resultados, detectar dificultades comunes y ofrecer apoyo más personalizado, todo a través de una herramienta que combina accesibilidad, pedagogía activa y tecnología móvil.

En este sentido, *Sí C puede* no se concibe solo como una aplicación de apoyo individual, sino también como un puente viable hacia la integración de la gamificación en

políticas educativas concretas, alineándose con las tendencias actuales en innovación docente.

Actualmente está concebida para su implantación el campus de la UPM ‘ETSII’, aunque gracias a su estructura y a la independencia de cada pantalla, no supondría una gran complejidad adaptar la aplicación a otras guías de aprendizaje.

8. Planificación temporal y presupuesto

El desarrollo de *Sí C puede* ha requerido una organización clara tanto en los tiempos para su creación como en recursos necesitados. En este apartado se presenta la planificación temporal, la estimación de horas de trabajo y la valoración económica de los recursos necesarios para llevar a cabo el proyecto a buen puerto.

8.1. Diagrama de Gantt

El desarrollo de la aplicación se estructuró en cinco bloques, definiendo plazos de entrega semanales, planteando este proyecto como un proyecto realizable en un plazo de 12 semanas de trabajo y dedicación completa (ver *figura 48*):

1. Investigación y diseño (1 semana): Revisión de gamificación aplicada a aprendizaje de C, definición de itinerarios y retos, y mapa de contenidos.

2. Diseño funcional y UX/UI (1 semanas): flujos de pantallas, *wireframes* y guías de estilo accesibles (contrastes, tamaños, toques por pantalla).

3. Desarrollo (4 semanas): implementación de forma independiente en Android Studio: módulos de lecciones, sistema de retos/puntos, persistencia local y analítica básica.

4. Pruebas y mejoras (1 semana): pruebas del funcionamiento de la aplicación, corrección de errores y optimización en los distintos dispositivos probados.

5. Documentación y memoria (5 semanas): redacción académica conteniendo toda la información sobre *Sí C puede*.

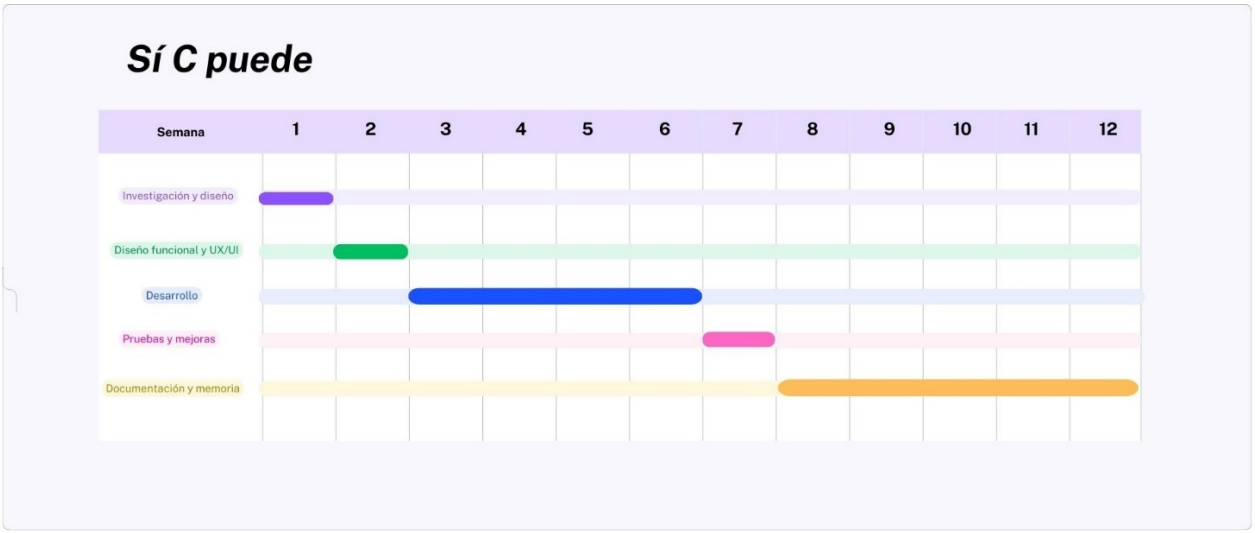

Figura 48 Diagrama de Gantt - Sí C puede

8.2. Estimación de horas de trabajo

Se completaron 400 horas reales de un programador junior de aplicaciones, en este apartado se presenta la división de las horas (*figura 49*). La distribución corresponde a las tareas necesarias para una aplicación educativa funcional y presentable en un TFG:

- Investigación y diseño: 40 h
- Diseño funcional y UX/UI: 40 h
- Desarrollo de la aplicación: 120 h
- Pruebas (unitarias/funcionales) y correcciones: 40 h
- Documentación técnica y memoria del TFG: 160 h

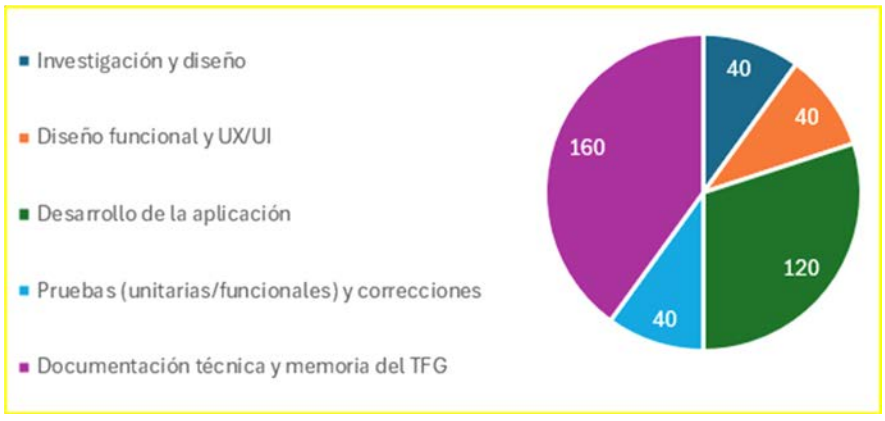

Figura 49 División del trabajo (horas) para la gestación de Sí C puede

8.3. Presupuesto y valoración de recursos

8.3.1. Costes operativos directos

a) Electricidad (desarrollo en portátil)

- Consumo medio razonable de un portátil en uso de desarrollo: ~60 W (intervalo habitual 30–140 W). Para 400 h el consumo sería de 24kWh [28]
- Precio medio doméstico orientativo 2024 en España (impuestos incluidos): 0,244 €/kWh Coste estimado: 24 × 0,244 = 5,86 € [29]

b) Conectividad fija (fibra en el hogar)

- El gasto medio en paquete sin televisión (fibra + móvil) rondó 42 - 43 €/mes en 2024 según datos del Panel de Hogares CNMC publicados por medios. Para un desarrollo de 3 meses, se toma 42,1 €/mes: 126,3 €. Si se le imputa un 20 % de uso atribuible al proyecto (el resto es uso doméstico), el coste asignado sería ≈ 25,3 € [30]

c) Amortización del equipo

- Portátil de HP de 500€. La tabla de amortización simplificada de la AEAT para “equipos para tratamiento de la información y programas informáticos” fija coeficiente lineal máximo 26 % (vida útil 10 años máx.). Amortización anual: 130 €; para 3 meses de proyecto: 32,5 € [31]

d) Publicación en *UptoDown*

- Tasa única de alta en *UptoDown*. Coste: 0 €.

e) Software y repositorios

- Android Studio y librerías empleadas: gratuitas (licencias libres). Coste: 0 €.

f) Otros (desplazamientos, impresión, materiales)

- No aplican (proyecto 100 % digital). Coste: 0 €.

8.3.2. Valorización del trabajo

Buscando el equivalente a la contratación de un programador Android junior en Madrid, el salario de referencia ronda los 23000 €/anuales. Esto supondría un salario 11 €/h, por lo que, para un total de 400 horas implicaría un coste en personal de 4400 € [32]

8.3.3. Resumen de costes

En la *tabla 6* se puede observar tanto el desglose de las costas como el total. Teniendo en cuenta, el potencial del proyecto se determina que es un dinero muy asumible (ver *apartado 9.2*).

Tabla 6 Desglose costes - Sí C puede

Tarea	Coste (€)
Electricidad	5,86
Conectividad	25,30
Equipo	32,50
Programador Junior	4400,00
COSTE TOTAL	4463,66

9. Evaluación de impactos y aspectos transversales

La evaluación de impactos y aspectos transversales permite observar el proyecto con una mirada amplia, más allá de lo técnico, y valorar sus implicaciones sociales, económicas, ambientales y ético-legales, así como su contribución a la Agenda 2030. Este capítulo organiza el análisis en cinco ejes, social y educativo; económico; medioambiental; legal y ético; y alineamiento con los ODS, con el fin de identificar beneficios, riesgos y medidas de mejora de forma coherente con los marcos normativos y de sostenibilidad vigentes [33] [34]. Dependiendo del impacto, así como del grupo de interés al que este pertenezca, la importancia será una u otra *(figura 50).*

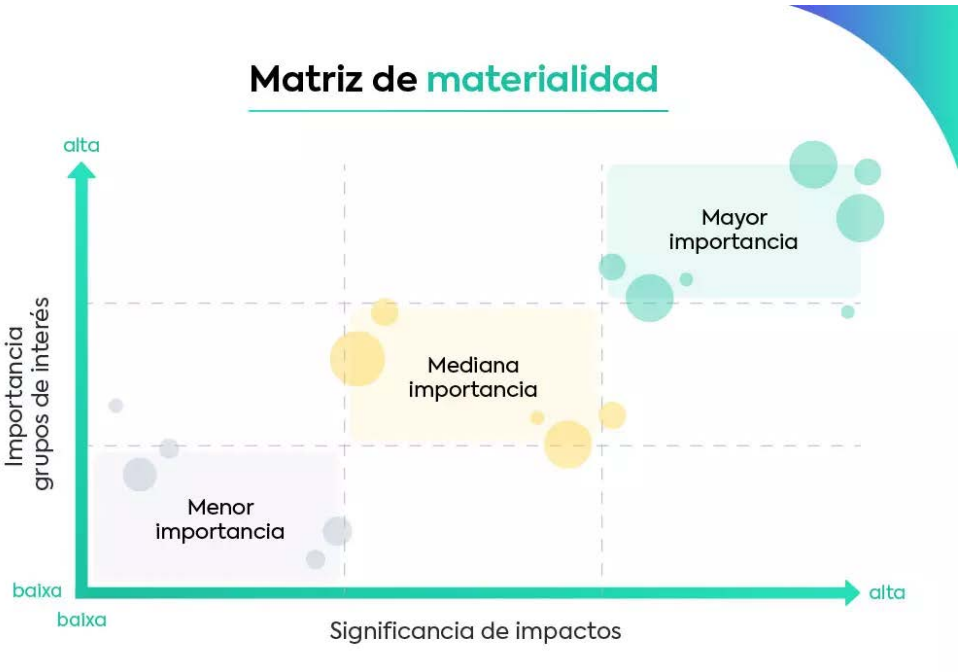

Figura 50 Matriz de materialidad (AtlasGov)

Metodológicamente, se combinan referencias consolidadas: la Evaluación de Impacto Social (EIS) *(figura 51)*, con buenas prácticas internacionalmente aceptadas y materiales en castellano que orientan sobre participación, identificación de afectados, trazabilidad y gestión de efectos [35], combinada con la Evaluación de Impacto Ambiental (EIA), integrada en el ordenamiento español por la Ley 21/2013 para planes, programas y proyectos, con especial atención a prevención, alternativas y participación pública. Por su parte, el enfoque de ciclo de vida permite estimar impactos a lo largo de toda la cadena y, cuando procede, se complementa con costes y dimensiones sociales [36].

Figura 51 EIS – Funcionamiento (CIIJA)

Finalmente, la contribución del proyecto a los Objetivos de Desarrollo Sostenible se valorará como hilo conductor para priorizar impactos materiales, definir indicadores y proponer medidas de mitigación o maximización del valor público, en línea con los documentos oficiales de la Agenda 2030 y guías operativas de Naciones Unidas [37].

9.1. Impacto social y educativo

La huella social y formativa de una iniciativa evidencia su habilidad para propiciar avances constructivos y duraderos tanto en las poblaciones como en los contextos de enseñanza. Este estudio pondera cómo las acciones propuestas pueden afectar la integración social, la equidad en las oportunidades, la edificación de habilidades y el fortalecimiento de las conexiones colaborativas. La perspectiva no se restringe a las consecuencias inmediatas, sino que sopesa también las repercusiones a medio y a largo plazo, prestando particular atención a los grupos directamente favorecidos y a los agentes involucrados de manera indirecta [38].

Para detectar y jerarquizar estos efectos, una de las herramientas más empleadas es la Matriz de Materialidad (*figura 50*), que consiente representar en un esquema bidimensional la trascendencia que cada elemento social o formativo posee para los grupos de interés en contraste con su relevancia para la organización. Dicho análisis simplifica la adopción de decisiones y la provisión estratégica de recursos hacia las áreas de mayor capacidad transformadora. Modelos más sofisticados, como el mostrado en la *figura 52*, integran categorías operativas, por ejemplo, “Involucrar”, “Colaborar” o “Monitorear” que aconsejan sobre la contestación conveniente ante cada clase de impacto [39].

Figura 52 Matriz de Materialidad con categorías operativas

La iniciativa, además, se enriquece de metodologías participativas, muy importantes para certificar que las opiniones y carencias de la comunidad estén representadas en el diagnóstico y en las resoluciones resultantes. La guía de participación pública argentina ilustra cómo añadir mecanismos inclusivos, que abarcan desde consultas abiertas hasta comités asesores locales, asegurando que la estimación no sea un trámite unilateral, sino una conversación constante [40].

En el espacio formativo, la Teoría del Cambio (*figura 53)* se muestra como un marco provechoso para planear y examinar intervenciones, dibujando una conexión clara entre las labores efectuadas, los resultados inmediatos y los efectos aguardados. Esta visión permite establecer indicadores concretos, calibrar el progreso y adecuar las medidas conforme a la prueba obtenida, vigorizando de esta manera la eficiencia y la pertinencia de las tácticas formativas [41].

DIAGRAMA DE LA TEORÍA DE CAMBIO SOCIAL DE EDUCO

VISIÓN	**Un mundo donde niñas, niños y adolescentes disfrutan plenamente de sus derechos y de una vida digna.**		
IMPACTO SOCIAL	Niñas, niños y adolescentes disfrutan plenamente de su bienestar, sus derechos y de las oportunidades para desarrollar sus capacidades en un planeta compartido.		
RESULTADOS DE IMPACTO SOCIAL	**RIS 1.** Niñas, niños y adolescentes desarrollan plenamente su personalidad y sus capacidades.	**RIS 2.** Niñas, niños y adolescentes viven libres de violencia en entornos seguros y de buen trato.	**RIS 3.** Niñas, niños y adolescentes impulsan cambios sociales en su entorno social y medioambiental.
INDICADORES GLOBALES DE LOS RESULTADOS DE IMPACTO SOCIAL	**RIS 1** **1.1.** Niñas, niños y adolescentes desarrollan su proyecto de vida y su capacidad de aspirar, cumpliendo así sus expectativas vitales. **1.2.** Niñas, niños y adolescentes disfrutan de una educación pertinente, equitativa, segura e inclusiva. **1.3.** Niñas, niños y adolescentes disfrutan de una educación crítica, colaborativa y creativa, orientada a la transformación social. El RIS 1 contribuye a alcanzar las siguientes Metas de los ODS: 4.1. / 4.2. / 4.3. / 4.4. / 4.5. / 4.6. / 4.7. / 4.A.	**RIS 2** **2.1.** Niñas, niños y adolescentes disfrutan de relaciones positivas y empáticas, basadas en el respeto con sus figuras de apego, referentes adultos y sus pares. **2.2.** Niñas, niños y adolescentes confían plenamente en las personas de su entorno y se sienten seguros y protegidos en su comunidad y entorno escolar. **2.3.** Niñas, niños y adolescentes disfrutan de sistemas de protección de la infancia efectivos que previenen y erradican cualquier forma de violencia y les protegen de la vulnerabilidad extrema, crisis y desastres. El RIS 2 contribuye a alcanzar las siguientes Metas de los ODS: 1.5. / 4.A. / 5.2. / 5.3. / 5.6. / 8.7. / 13.1. / 13.3. / 16.2. / 16.9.	**RIS 3** **3.1.** Niñas, niños y adolescentes se organizan y actúan como agentes de cambio social. **3.2.** Niñas, niños y adolescentes impulsan en sus comunidades y entorno la toma de consciencia alrededor del desarrollo sostenible. **3.3.** Niñas, niños y adolescentes promueven en sus comunidades y entornos sociales cambios en valores, creencias y normas sociales. El RIS 3 contribuye a alcanzar las siguientes Metas de los ODS: 5.5. / 5.C. / 11.7. / 12.8. / 13.2. / 13.3. / 16.7.

Figura 53 Diagrama de Teoría del Cambio aplicado a impacto social y educativo

La aplicación *Sí C puede* no solo se centra en la enseñanza de la programación en C, sino que también busca facilitar el acceso a habilidades digitales esenciales, disminuyendo las limitaciones económicas, geográficas y técnicas. Gracias a su formato de juego, impulsa la motivación y el aprendizaje independiente, lo cual promueve la integración de alumnos con distintos niveles de experiencia y situaciones socioeducativas. Al adaptarse sin problemas a

ambientes universitarios y de autoaprendizaje, la aplicación tiene un efecto inmediato en las oportunidades laborales y en la aptitud de los usuarios para ajustarse a un entorno laboral que depende cada vez más de las tecnologías de la información.

9.2. Impacto económico

La creación e introducción de *Sí C puede* se centra en el impacto que podría tener sobre los alumnos, los cuales podrían usar esta herramienta de manera gratuita, ahorrando una gran cantidad de dinero evitando el pago de caras academias o demás cursos cerrados. Este ahorro se convierte en mayor facilidad para acceder a la formación, lo que ayuda a estudiantes a aprender a programar en C y, por tanto, a tener más oportunidades de trabajo en el futuro. El beneficio económico que no se ve está en que mejora las opciones de encontrar empleo y de ganar más, lo que a la larga beneficia a toda la sociedad [42].

En las universidades y centros de estudio, la herramienta se puede usar en los planes de estudio sin tener que gastar más en equipos o licencias, lo que ayuda a usar mejor el dinero de las facultades. Esto no solo baja los gastos, sino que también ayuda a probar nuevas formas de enseñar sin añadir costes [43].

Si miramos al sector, el proyecto ayuda a fortalecer el mundo del software libre y la enseñanza digital. Usar tecnologías abiertas significa que otros (escuelas, empresas de formación o gente que aprende por su cuenta) pueden usar y cambiar la herramienta, lo que genera negocio alrededor de la creación de materiales, clases o títulos. Esta conexión con el mundo de la tecnología puede llevar a trabajos en conjunto o patrocinios, que ayuden a que el proyecto siga adelante económicamente.

Todo esto hace que la iniciativa sea muy valiosa económicamente, no solo por lo que se ahorra, sino porque puede ayudar a que crezca la economía del conocimiento y a que todos tengan las mismas oportunidades de aprender habilidades digitales importantes.

En un escenario hipotético basado en la realidad del campus, si en una clase de 60 estudiantes al menos 10 utilizaran *Sí C puede* como alternativa a un curso externo valorado en 150 €, el ahorro directo sería de 1 500 € por clase. Considerando que el campus puede incluso tener hasta ocho clases anuales que imparten la asignatura, el ahorro total anual ascendería a 12 000 €. Esta cifra refleja únicamente el beneficio económico directo para el estudiante, sin contabilizar el valor añadido derivado de la reutilización de la herramienta en siguientes años, la reducción de costes para las instituciones educativas o el incremento en la empleabilidad gracias al acceso gratuito a competencias digitales.

9.3. Impacto medioambiental

Si bien *Sí C puede* es una herramienta digital, no un objeto tangible, su uso ayuda a disminuir ciertos efectos negativos en el medio ambiente que se aprecian en las formas de enseñar de siempre. Cambiar los libros y apuntes en papel por materiales digitales reduce la cantidad de papel que se necesita y, por lo tanto, alivia la presión sobre los bosques y la industria que imprime y reparte los materiales, que gasta mucha agua, luz y químicos.

Que se pueda usar la aplicación cuando sea y donde sea, sin tener que viajar, también baja las emisiones que produce el transporte de los estudiantes a clases extras. Si, por ejemplo, 50 estudiantes no tuvieran que viajar 10 km cada semana para ir a un curso, se ahorrarían más de 3 000 km de viaje en un semestre, lo que evitaría que se emitieran unos 450 kg de CO_2 [44].

Además, como está hecha para ser un programa ligero, Sí C puede utilizar menos capacidad de los ordenadores y necesita menos espacio para guardar datos que otras aplicaciones más grandes, lo que significa que gasta menos energía en los servidores y en los aparatos que usan las personas. Esto es importante si pensamos que, en todo el mundo, la tecnología es responsable de entre el 2 % y el 4 % de todos los gases que causan el efecto invernadero [45].

En resumen, la aplicación ayuda al medio ambiente gracias a que disminuye tanto el uso de materiales, como la contaminación que produce ir a clases presenciales adicionales, impulsando una forma de aprender digital más efectiva y que cuida el planeta de forma indirecta.

9.4. Análisis de aspectos legales y éticos

La aplicación *Sí C puede* ha de respetar varios aspectos legales y morales que deben asegurarse desde su inicio y a lo largo de su existencia. Como un medio didáctico que mezcla datos de uso, como avance en las fases, tiempo en línea o logros de las tareas, se ubica dentro de la Ley General de Protección de Datos (RGPD) de la Unión Europea y de la Ley Orgánica 3/2018 de Protección de Datos Personales y garantía de los derechos digitales [46] [47]. Es por ello por lo que es importante cumplir con los derechos RGPD *(figura 54).*

Figura 54 Esquema de derechos del usuario bajo el RGPD (CoREGISTROS)

En el tema moral, el proyecto debe mirar por todos los estudiantes. Es por ello por lo que el programa está configurado para ser usado por gente con diferentes talentos. Para ello, es importante la claridad en el uso de elementos de código y textos. Al hacerse bajo un estilo de software libre, es clave respetar los permisos de otros (como *GNU GPL, MIT o Creative Commons) (figura 55).* Este hecho no solo evita problemas legales, sino que apoya la honradez de estudio y de trabajo del plan.

Para decirlo de otra forma, ver los asuntos legales y morales bien hechos no solo asegura que sigamos las reglas, sino que también hace que la gente confíe más, algo muy importante para que Sí C puede se use y dure en el tiempo.

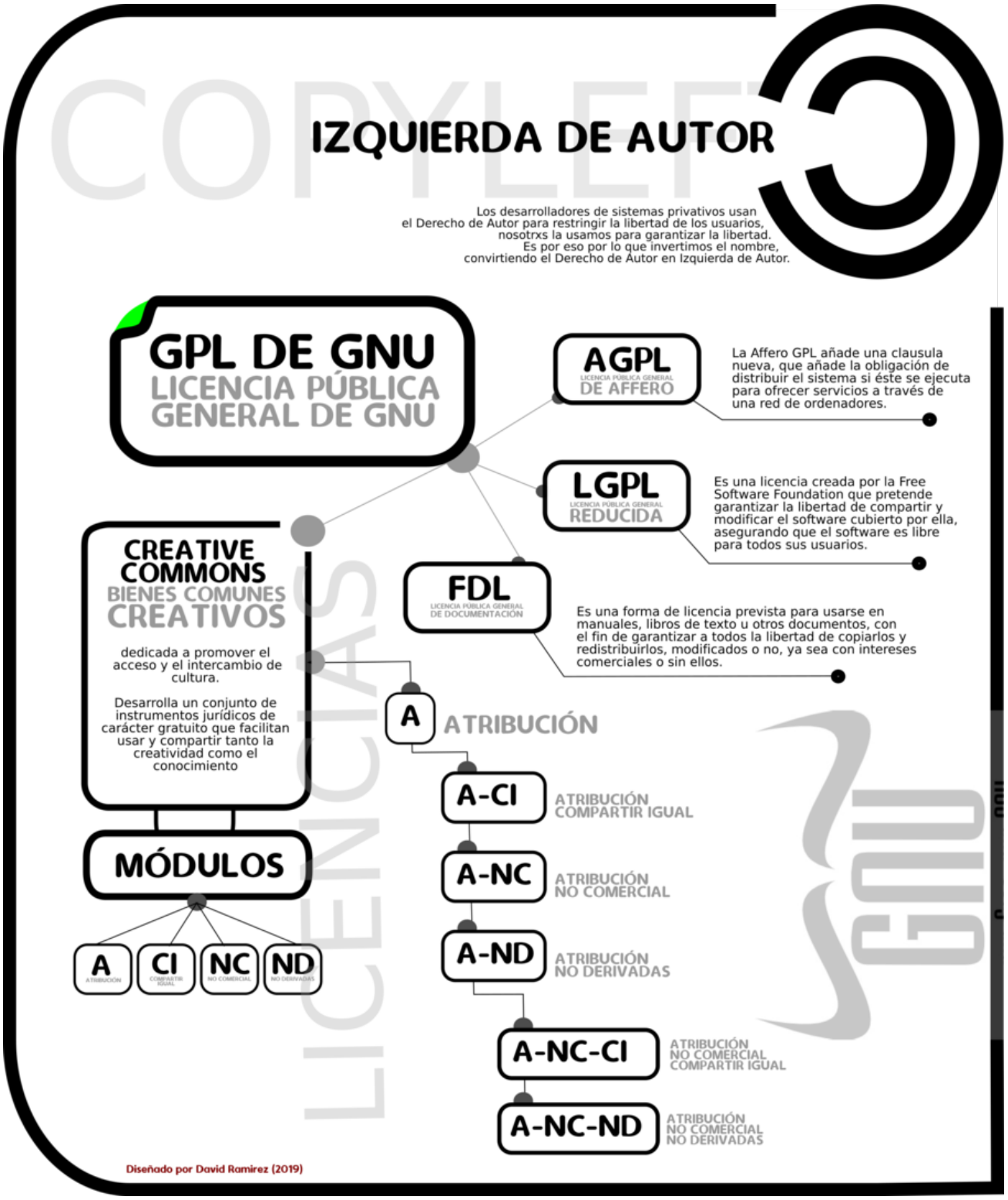

Figura 55 Tipos de licencias de software libre y Creative Commons (David Ramírez)

9.5. Contribución a los Objetivos de Desarrollo Sostenible (ODS)

Aunque su objetivo principal es enseñar programación en C, Sí C puede está en sintonía con la educación y el desarrollo que promueve la Agenda 2030. Su diseño sencillo, el fácil acceso y la forma en la que se utiliza la convierten en un recurso que contribuye a varios Objetivos de Desarrollo Sostenible.

ODS 4 (figura 56) - Educación de calidad: El hecho de que la aplicación sea gratuita o de bajo coste permite que cualquier estudiante aprenda una habilidad muy valiosa como es la programación, sin importar su situación económica o el lugar donde viva, simplemente necesita un dispositivo Android, sin siquiera necesitar conexión a internet. Esto encaja con la idea de una educación para todos, en la que la tecnología derriba barreras y abre oportunidades. Además, al incorporar dinámicas de juego, aprender resulta más ameno y divertido, facilitando la retención de los contenidos.

Figura 56 ODS 4 Educación de calidad

ODS 5 (figura 57) - Igualdad de género: En el campo de la programación, la presencia femenina todavía es reducida, siendo la actividad considerada como una de ‘chicos’. Ofrecer un entorno de aprendizaje accesible, seguro y equitativo contribuye a animar a más mujeres a formarse en áreas de ciencia y tecnología. Sí C puede crea un espacio donde el género no condiciona las oportunidades, y todos pueden desarrollar sus capacidades en igualdad de condiciones.

Figura 57 ODS 5 - Igualdad de género

ODS 8 (figura 58) - Trabajo decente y crecimiento económico: Aprender a programar va más allá de adquirir una destreza técnica: es abrir la puerta a empleos de calidad en un sector que demanda profesionales cualificados. La aplicación puede ser un puente para que más personas accedan a oportunidades laborales bien remuneradas, impulsando a su vez el crecimiento de la economía digital.

Figura 58 ODS 8 - Trabajo decente y crecimiento económico

ODS 9 (figura 59) - Industria, innovación e infraestructura: El proyecto aprovecha infraestructuras digitales ya existentes para generar innovación educativa. No se trata de crear más dispositivos, sino de darles un nuevo uso: convertir teléfonos y conexiones a internet en aulas abiertas y accesibles desde cualquier lugar.

Figura 59 ODS 9 - Industria, innovación e infraestructura

ODS 10 (figura 60) - Reducción de las desigualdades: Cuando la posibilidad de aprender algo importante deja de depender del dinero o de la cercanía a un centro especializado, las diferencias entre personas disminuyen. Sí C puede contribuye a nivelar el terreno, ofreciendo el mismo contenido y calidad a cualquiera que disponga de un dispositivo y conexión a internet.

Figura 60 ODS 10 - Reducción de las desigualdades

En definitiva, esta herramienta no solo enseña a programar, sino que impulsa un modelo de aprendizaje que abre puertas, elimina barreras y contribuye a un futuro más inclusivo, equitativo y sostenible.

10. Bibliografía

[1]
Vasco G. Guía para la aplicación conjunta de los Análisis de Ciclo de Vida (LCA) y Coste del Ciclo de Vida (LCC) 2016.

[2]
Tributaria A. Tabla de amortización simplificada 2025.

[3]
Consejo PE y. Reglamento (UE) 2016/679 relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos. 2016.

[4]
Backes M, Bugiel S, Schranz O, Von Styp-Rekowsky P, Weisgerber S. ARTist: The Android Runtime Instrumentation and Security Toolkit, IEEE; 2017, p. 481–95. https://doi.org/10.1109/EuroSP.2017.43.

[5]
Smyth. An Overview of Gradle in Android Studio. United States: PAYLOAD MEDIA, INC; 2025.

[6]
Kvam R. Evaluación del impacto social: Integrando los aspectos sociales en los proyectos de desarrollo. 2018. https://doi.org/10.18235/0012592.

[7]
Correa Nuñez SM. Metodología de gobierno y gestión en el sistema universitario de España para garantizar el resultado esperado en la transformación digital 2024:11–5. https://doi.org/10.61752/acd.i12.180.

[8]
Glassdoor. Sueldos para el puesto de Desarrollador Android Junior en España 2025.

[9]
Nombres internos, etiquetas y números de compilación 2024.

[10]
Hamari J, Koivisto J, Sarsa H. Does Gamification Work? -- A Literature Review of Empirical Studies on Gamification, IEEE; 2014, p. 3025–34. https://doi.org/10.1109/HICSS.2014.377.

[11]
Hamari J, Koivisto J, Sarsa H. Does Gamification Work? -- A Literature Review of Empirical Studies on Gamification, IEEE; 2014, p. 3025–34. https://doi.org/10.1109/HICSS.2014.377.

[12]
Developers G. Realiza lanzamientos con confianza n.d.

[13]

Dominguez A, Saenz-de-Navarrete J, de-Marcos L, Fernandez-Sanz L, Pages C, Martinez-Herraiz J-J. Gamifying learning experiences: Practical implications and outcomes 2013;63:380–92. https://doi.org/10.1016/j.compedu.2012.12.020.

[14]
García Ureta AM. Comentarios sobre la Ley 21/2013, de evaluación ambiental 2014:317–71.

[15]
ONU. Manual básico sobre la Agenda 2030 para el personal de las Naciones Unidas 2020.

[16]
Developers A. Descripción general de las transmisiones 2025. https://developer.android.com/develop/background-work/background-tasks/broadcasts?hl=es-419 (accedido 20 de Julio, 2025).

[17]
Developers A. Ciclo de vida de la actividad 2025. https://developer.android.com/guide/components/activities/activity-lifecycle?hl=es-419 (accedido 20 de Julio, 2025).

[18]
España G de. Factores de emisión para la estimación de CO_2 en transporte 2021.

[19]
España G de. Ley Orgánica 3/2018 de Protección de Datos Personales y garantía de los derechos digitales. Boletín Oficial del Estado. 2018.

[20]
Developers A. Descripción general de los servicios 2025. https://developer.android.com/develop/background-work/services?hl=es-419 (accedido 24 de Julio, 2025).

[21]
CodeCombat. CodeCombat: Learn to Code by Playing a Game 2025.

[22]
Ramírez I. Android 13 sigue en cabeza y Android 14 en quinto puesto: así quedan las versiones de Android más usadas en 2024 2024.

[23]
Deterding S, Dixon D, Khaled R, Nacke L. From game design elements to gamefulness, New York, NY, USA: ACM; 2011, p. 9–15. https://doi.org/10.1145/2181037.2181040.

[24]
Kahoot. Kahoot! Learning platform 2025.

[25]
Developers U. Publish Your App Uptodown Developers Console n.d.

[26]
Developers A. Intents y filtros de intents 2025.

https://developer.android.com/guide/components/intents-filters?hl=es-419 (Accedido 26 de Julio, 2025).

[27]
Duolingo. Duolingo: language lessons for everyone 2025.

[28]
Argentina G de. Guía para la participación pública en evaluaciones de impacto social. Ministerio de Ambiente y Desarrollo Sostenible 2019.

[29]
Malmodin J, Lundén D. The Energy and Carbon Footprint of the Global ICT and E&M Sectors 2010–2015 2018;10:3027-. https://doi.org/10.3390/su10093027.

[30]
Repsol. ¿Cuál es el consumo de un ordenador? 2023.

[31]
Jayathilake L, Jayawardana Y. Apache Software Foundation (ASF) and The Apache Way. figshare; 2025. https://doi.org/10.6084/m9.figshare.28629356.

[32]
Education in the knowledge society 2015.

[33]
Express EP. La factura telefónica baja mientras sube el gasto en ‘streaming’ 2024.

[34]
Kapp KM. The gamification of learning and instruction. San Francisco, Calif: Pfeiffer; 2012.

[35]
Developers A. Proveedores de contenido 2025. https://developer.android.com/guide/topics/providers/content-providers?hl=es-419 (Accedido 29 de Julio, 2025).

[36]
ResponsaBle.net. Análisis de materialidad: Más allá de un requisito para reportar 2022.

[37]
OECD Skills Outlook 2021. Paris: OECD Publishing; 2021. https://doi.org/10.1787/0ae365b4-en.

[38]
Transformar nuestro mundo : la Agenda 2030 para el Desarrollo Sostenible 2015. https://data.europeana.eu/item/403/https___hispana_mcu_es_lod_oai_bibliotecadigital_aecid_es_12211_ent0.

[39]
Developers A. Fragmentos 2025.

[40]
Tam D. Android originally designed as smart-camera system 2013.

[41]
Callaham J. Google made its best acquisition nearly 17 years ago: Can you guess what it was? 2022.

[42]
Educo.org. IMPACTO, EVALUACIÓN Y APRENDIZAJE 2021.

[43]
Overview - Developers: Why Does Android Run Time Score Over Dalvik? 2015.

[44]
Bankinter. Comparativa del precio de la luz en Europa: ¿Dónde se paga más electricidad? 2023.

[45]
Uruguay NU-. MANUAL BÁSICO SOBRE LA AGENDA 2030 PARA EL DESARROLLO SOSTENIBLE 2020.

[46]
UNESCO. Digital technology and the futures of education – towards 'non-stupid' optimism 2021.

[47]
Hubwieser P, Giannakos MN, Berges M, Brinda T, Diethelm I, Magenheim J, et al. A Global Snapshot of Computer Science Education in K-12 Schools, New York, NY, USA: ACM; 2015, p. 65–83. https://doi.org/10.1145/2858796.2858799.

POLITÉCNICA

Escuela Técnica Superior de Ingenieros Industriales
Universidad Politécnica de Madrid

José Gutiérrez Abascal, 2. 28006 Madrid
Tel.: 91 336 3060
info.industriales@upm.es

www.industriales.upm.es