Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Tecnologías Low-Code y No-Code: Un caso práctico para estudiar su potencial y limitaciones

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Diego Roig Hervás

Tutor: Patricio Letelier Torres

Curso 2020-2021

Resumen

Las tecnologías Low-Code y No-Code constituyen una de las propuestas más interesantes en el ámbito del desarrollo de software. Se han hecho populares por la promesa de que, con ellas, podrías realizar un proyecto software sin necesidad de conocimiento técnico y en un tiempo reducido. No obstante, estas tecnologías son unas grandes desconocidas; no se sabe con exactitud hasta dónde pueden llegar y si todo aquello que prometen se cumple.

Este Trabajo de Fin de Grado (TFG) tiene como principal objetivo el analizar dichas tecnologías Low-Code y No-Code con el fin de descubrir sus ventajas y desventajas. Para ello se llevan a cabo unos desafíos en los que se intenta desarrollar una serie de funcionalidades. Estas funcionalidades se obtienen con la ayuda de dichas tecnologías y sin ellas; para así poder cerciorarnos del estado en el que se encuentran las tecnologías estudiadas en comparación con el desarrollo tradicional. Estas pruebas se hicieron con el apoyo de la herramienta Low-Code desarrollada por Microsoft, Power Apps.

Palabras clave: Low-Code, No-Code, desarrollo de software, conocimiento técnico, Microsoft, Power Apps.

Abstract

Low-Code and No-Code technologies are two of the most interesting proposals in software development. They have become popular because, with them, you would be able to develop any software project without technical knowledge and in a short period of time. However, some aspects of these technologies are unknown; it is not possible to know how far they can go and if everything they promise is fulfilled.

The main objective of this Final Degree Project is to analyze the Low-Code and No-Code technologies to discover their advantages and disadvantages. To achieve this objective some challenges were made to reach some milestones. The milestones were made with and without these technologies; with that activity we can see the state of the studied technologies compared to their competitors. These tests were done with the support of the Low-Code tool developed by Microsoft, Power Apps.

Keywords: Low-Code, No-Code, software development, technical knowledge, Microsoft, Power Apps.

Índice de contenido

1. Introducción 9
   1.1 Motivación 9
   1.2 Objetivos 10
   1.3 Estructura de la memoria 10
2. Tecnologías No-Code y Low-Code 13
   2.1 Diferencias entre No-Code y Low-Code 13
   2.2 Mitificación de las tecnologías 14
   2.3 Entrevista con un experto 15
   2.4 Diferencias y semejanzas con el desarrollo dirigido por modelos (MDD) 17
3. Estado del arte 19
   3.1 Estudios sobre la materia 19
   3.2 Crítica al estado del arte y propuesta 20
   3.3 Conclusiones 22
4. Herramientas Low-Code y No-Code 24
   4.1 Herramientas Low-Code 24
   4.2 Herramientas No-Code 28
   4.3 Herramienta del caso de estudio 30
5. Caso de estudio 32
   5.1 Contexto 32
   5.2 Twins y Adventour usando Power Apps 34
   5.3 Diseño y navegación 37
   5.4 Lista de componentes a partir de una fuente de datos 43
   5.5 Conexión a base de datos y la persistencia 50
   5.6 Motor de búsqueda y filtros 56
   5.7 Multimedia 62
   5.8 Integración de un mapa de Google y creación de rutas 68
   5.9 Extensiones disponibles 74
   5.10 Conclusiones del caso de estudio 76
6. Conclusiones y trabajo futuro 78
   6.1 Conclusiones 78
   6.2 Trabajo futuro 80
7. Bibliografía 82

Índice de figuras

Figura 1: Interés a lo largo del tiempo basado en las búsquedas de Low-Code y MDD.. 17
Figura 2: Cantidad de usuarios de las tecnologías Low-Code previstos para los próximos años [20]. 22
Figura 3: Cuadrante clasificatorio de las principales tecnologías Low-Code [21]. 24
Figura 4: Filtrado de las aplicaciones para Power Apps en AppSource. 25
Figura 5: Cursos sobre Power Apps organizados por niveles de dificultad. 26
Figura 6: Entorno inicial de Power Apps. 34
Figura 7: Apartado de vista de árbol. 34
Figura 8: Apartado de las fuentes de datos. 35
Figura 9: Apartado multimedia. 35
Figura 10: Apartado de herramientas avanzadas. 35
Figura 11: Interfaz de las propiedades de un objeto. 36
Figura 12: Propiedad OnSelect. 36
Figura 13: Interfaz de origen del menú de Twins. 37
Figura 14: Réplica de la interfaz del menú de Twins. 37
Figura 15: Elección de la temática de la aplicación. 38
Figura 16: Carga de las imágenes necesarias para elaborar la réplica. 38
Figura 17: Referenciación de la imagen en el objeto. 39
Figura 18: Uso de la función Navigate en la propiedad OnSelect. 39
Figura 19: Uso de un listener para la navegación entre interfaces en Android Studio. . 39
Figura 20: Interfaz de origen del login de Adventour. 40
Figura 21: Réplica de la interfaz del login de Adventour. 40
Figura 22: Extracción de los colores de la interfaz de Adventour con Adobe Color. 40
Figura 23: Creación de una variable global con los valores RGBA de los colores del login. 41
Figura 24: Uso de la función onPressed para navegar entre interfaces en Flutter. 42
Figura 25: Opciones que proporciona Power Apps para comenzar a crear una aplicación. 43
Figura 26: Ejemplo del tipo de etiquetas del que se van a extraer los datos. 44
Figura 27: Fuente de datos en Excel. 45
Figura 28: Conexión con OneDrive con la que elegir la tabla que funciona como fuente de datos. 45
Figura 29: Resultado de la generación automática. 46
Figura 30: Contenido de uno de los campos generados automáticamente. 46
Figura 31: Apartado de los detalles de uno de los elementos de la lista. 46
Figura 32: Lista de productos personalizada. 47
Figura 33: Disposición de los detalles de cada uno de los productos. 47
Figura 34: Asociación de fuente de datos a la galería. 48
Figura 35: Lista de los lugares de interés de la ruta "Barcelona city". 49
Figura 36: Apartado de conexiones en la pantalla principal. 51
Figura 37: Uso del conector OneDrive para acceder a la tabla Prueba. 51
Figura 38: Muestra práctica de la diferencia entre a lo que hace referencia el formulario y cada una de las tarjetas. 53
Figura 39: Propiedades de una tarjeta del formulario. 53

Figura 40: Función Search en la documentación de Microsoft [33]. 56
Figura 41: Interfaz elaborada para realizar las pruebas del motor de búqueda. 57
Figura 42: Ejemplo de aplicación de la función Search. 57
Figura 43: Prueba de funcionamiento de la funcion Search de la figura 41.....57
Figura 44: Fuente de datos con la comuna nueva creada. 58
Figura 45: Interfaz del motor de búsqueda con el nuevo icono de filtrado. 59
Figura 46: Interfaz para el filtrado de los productos. 59
Figura 47: Ejemplo del filtrado de producto. 59
Figura 48: Función If en la documentación de Microsoft [34]. 60
Figura 49: Función Filter en la documentación de Microsoft [33]. 60
Figura 50: Ejemplo del uso simultáneo de la función If y Filter. 60
Figura 51: Elementos multimedia disponibles. 62
Figura 52: Referenciación de la canción en la propiedad multimedia. 63
Figura 53: Captura de pantalla del apartado "Crear baraja". 64
Figura 54: Interfaz de la réplica del apartado "Crear baraja". 64
Figura 55: Código para la inclusión de un vídeo en Android Studio. 66
Figura 56: Inclusión de vídeo mediante una URL. 67
Figura 57: URL de la petición ocultando la API key. 69
Figura 58: Resultado de la petición detallada. 69
Figura 59: Inclusión de marcadores en la petición. 70
Figura 60: Resultado de la petición con marcadores. 70
Figura 61: Función Launch en la documentación de Microsoft [39]. 71
Figura 62: Ejemplo de uso de la función Launch. 71
Figura 63: Ruta resultante de la petición realizada. 72

Índice de tablas

Tabla 1: Resumen de las conclusiones de los desafíos. 76

1. Introducción

1.1 Motivación

Actualmente existen tecnologías que prometen que no es necesaria la obtención de conocimientos avanzados en ingeniería del *software* para desarrollar una aplicación. Personas sin estudios relacionados con la informática podrían generar una aplicación haciendo uso de tecnologías Low-Code [1] y No-Code [2].

Low-Code es una tecnología que proporciona un entorno para desarrollar y desplegar una aplicación escribiendo muy poco código. En el caso de No-Code, es una tecnología la cual proporciona un entorno similar a Low-Code, no obstante, no se escribiría código para el desarrollo de la aplicación.

En el caso de estas dos tecnologías, constituyen un movimiento bastante significativo; pueden cambiar la forma en la que se entiende el desarrollo, pudiendo modificar las dinámicas que se llevan a cabo hasta ahora. Podría llegar a complementar o incluso sustituir las prácticas de programación que se siguen en la actualidad para el desarrollo de proyectos software.

Además, al no necesitar elaborar código para desarrollar aplicaciones, se podrían potenciar conceptos como la democratización del software [3]; haría mucho más fácil que cualquier persona aprendiera a desarrollar soluciones sin prácticamente conocimiento previo.

Dicha democratización constituye un concepto relevante; a lo largo de la historia de la informática se han dado muchos casos en los que, para avanzar y realizar proyectos de mayor complejidad, se apostaba por la abstracción y programación a más alto nivel. Muy poca gente sigue programando en lenguaje ensamblador. De esta misma manera nos preguntamos, ¿por qué no podría haber llegado el momento de programar a un más alto nivel, dejando de lado aspectos más repetitivos y, por tanto, automatizables?

El carácter democratizador también hace que se aliente a los profesionales de las empresas a participar de la innovación y generación de ideas. Creando así una estructura descentralizada, la cual hace que cada uno de los empleados tenga capacidad para crear su propia aplicación [4]. De hecho, actualmente casi el 60% de las aplicaciones personalizadas se crean por personas que no pertenecen al departamento de TI. De estas personas, el 30% tiene habilidades de desarrollo limitadas o nulas [5].

Las tecnologías expuestas generan un gran interés en la sociedad actual, se trata de un tema muy candente. Despiertan especial interés debido a que, en ocasiones, el aprendizaje de los lenguajes de programación es una dificultad añadida a la hora de comenzar un proyecto de desarrollo. Realizar una formación con el fin de dominar un lenguaje de programación es una traba que mucha gente no está dispuesta a asumir.

Sin embargo, el aprender a utilizar una tecnología Low-Code o No-Code sería mucho más sencillo; el 70% de los usuarios de herramientas Low-Code aprendieron a utilizarlas en un mes o menos. Las facilidades que aportaría el uso de una tecnología de este tipo se verían también reflejadas en los resultados obtenidos por parte de los equipos TI que las utilizan. Las unidades de negocio se sienten un 21% más satisfechas con los plazos de entrega que cuando no utilizan tecnologías Low-Code [6]

Cada vez más empresas, especialmente startups, construyen sus sistemas con código escrito por terceros [7]. La pandemia ha hecho que se adopten este tipo de plataformas y herramientas, debido a que esta situación ha hecho que aumente la presión sobre los equipos TI. La falta de tiempo, que deriva de dicha presión, hace que se promueva el uso de tecnologías que permitan mejorar en términos temporales. Por ello, Capgemini identifica las tecnologías Low-Code y No-Code como una de las 20 principales tendencias tecnológicas a seguir durante este año [8].

Por último, cabe añadir que existe un hecho especialmente motivante relacionado con una experiencia personal. En bachiller científico, con 17 años, el profesor de informática presentó las funcionalidades de AutoCAD. Con dicho software de diseño asistido se podía automatizar gran parte del trabajo que se veía en la asignatura de dibujo técnico; en cuestión de segundos se podían realizar trazas que, sin el uso del software, conllevarían una gran cantidad de tiempo. Desde aquel momento comencé a interesarme por aquel tipo de herramientas. Existen maneras de realizar de forma automática aquellos procesos mecánicos y repetitivos que, en muchas ocasiones, resultan un mero trámite para la obtención de requisitos de complejidad superior.

1.2 Objetivos

El objetivo de este TFG es estudiar y analizar las ventajas y desventajas de las tecnologías Low-Code y No-Code a través de su aplicación en un caso práctico. En el caso de estudio se desarrollarán diferentes funcionalidades con y sin dichas tecnologías. Esto permitirá realizar comparaciones entre las dos alternativas expuestas, detectando la potencialidad y las limitaciones de estas nuevas tecnologías.

1.3 Estructura de la memoria

La estructura de la memoria es la siguiente:

En el capítulo 1 se introduce la memoria, de manera que se expone la motivación del trabajo y los objetivos.

El capítulo 2 profundiza en la descripción de las tecnologías Low-Code y No-Code, enfatizando en sus diferencias y exponiendo las diferencias y semejanzas que se encuentran con el desarrollo dirigido por modelos (MDD) [9]. Además, en esta parte se incluye una breve entrevista con un especialista en el área de generación automática de código. Vicente Pelechano es profesor del departamento de Sistemas Informáticos y

Computación en la Escuela Técnica Superior de Ingeniería Informática (Universidad Politécnica de Valencia).

En el capítulo 3 se expone el estado del arte asociado a las tecnologías Low-Code y No-Code. En el que se proporciona una síntesis crítica sobre los límites, ventajas y desventajas de las tecnologías Low-Code y No-Code. Para ello se parte de una serie de estudios y artículos de este ámbito.

En el capítulo 4 se presentan las principales herramientas Low-Code y No-Code. Profundizando en aquellas que se consideraron como las mejores opciones del mercado.

En el capítulo 5 se desarrolla el caso de estudio. Se comienza con una explicación del contexto y con un tutorial de la herramienta que se utilizó. Posterior a ello, se exponen y resuelven una serie de desafíos con la ayuda de la herramienta Low-Code y sin la ayuda de ella. Una vez completados todos los desafíos, se presentan las conclusiones.

En el capítulo 6 se exponen las conclusiones resultantes del trabajo desarrollado y el trabajo futuro.

2. Tecnologías No-Code y Low-Code

2.1 Diferencias entre No-Code y Low-Code

Siguiendo con el estudio de las tecnologías Low-Code y No-Code, resultaría relevante saber la diferencia que hay entre ambas. Dichas tecnologías distan en dos aspectos: el público al que van dirigidas y el grado de apertura de la tecnología [10].

En el caso del Low-Code, son tecnologías dirigidas a desarrolladores profesionales; estas tecnologías requieren de ciertos conocimientos técnicos. Es necesario tener ciertos conocimientos básicos de programación; muchas de las tecnologías Low-Code permiten incluir funciones que se obtienen mediante la programación de un fragmento de código [11].

Para los desarrolladores profesionales resultaría un apoyo a la hora de desarrollar, ya que no se ven obstaculizados por la codificación repetitiva o trabajo duplicado. Estas tecnologías les permitiría diseñar aplicaciones rápidamente arrastrando y soltando elementos y con un mínimo de codificación manual.

Las tecnologías No-Code están dirigidas a un número de usuarios más amplio. A diferencia de las Low-Code, están enfocadas a los trabajadores que no tengan conocimientos técnicos relacionados con la informática [12]. El desarrollo de una aplicación usando esta tecnología suele conllevar un proceso visual paso a paso, el cual permite crear aplicaciones de menor complejidad y funcionalidades más limitadas.

Por otro lado, en cuanto al grado de apertura de las tecnologías, encontramos grandes diferencias. Low-Code permite realizar cambios en su funcionamiento, cambiando o agregando código; afectando así al comportamiento de la aplicación. Esto es ventajoso debido a que hace aumentar la personalización, haciendo que pueda utilizarse en más casos de uso. Sin embargo, esto limitaría la compatibilidad con versiones previas.

Esto quiere decir que con cualquier actualización de la herramienta Low-Code, el usuario de la herramienta debería dedicar tiempo en comprobar si su aplicación funciona. Esto se debe a que, si en esa versión se modifica una funcionalidad que previamente has personalizado, podría generar problemas; ya que ahora funciona de manera diferente.

En cambio, las herramientas No-Code no tienen ese problema. De manera que, de recibir una actualización, los usuarios de la herramienta no deberían preocuparse de ningún cambio; se trata de un sistema cerrado. Esto constituiría una gran ventaja, en las herramientas No-Code; es menos probable que una actualización haga que deje de funcionar tu aplicación [10].

2.2 Mitificación de las tecnologías

Al igual que cualquier tecnología que emerge y que, por sus características, genera muchas expectativas, desde los medios de comunicación se produce un proceso de mitificación. Dicha mitificación consiste en ensalzar en exceso los atributos de la tecnología, haciendo así que se proporcione una idea distorsionada de ellas. La propuesta de las tecnologías Low-Code y No-Code es especialmente llamativa; sobre todo para aquellos que no poseen formación técnica en el ámbito del desarrollo del software.

De hecho, con el motivo de incrementar las visitas a los artículos, hay diferentes webs que venden estas tecnologías como una idea revolucionaria y que va a cambiar todo el contexto de la informática. Pero ¿realmente es esto así?

En el caso del No-Code, no pretende posicionarse como una solución para cada una de las problemáticas que van apareciendo, sino que ataca a nichos muy específicos [3]. En este caso, el nicho de las webs es idóneo; No-Code obtiene un potencial muy alto en aplicaciones con funcionalidades sencillas. Además, hay una gran cantidad de herramientas gratuitas, de manera que, junto a la desaparición de la necesidad de conocimientos técnicos, haría que sea accesible para un público muy amplio.

Por otro lado, en el ámbito de las aplicaciones de funcionalidades más complejas nos encontraríamos un panorama bastante diferente. Dichas aplicaciones conllevan una mayor cantidad de algoritmos y otros elementos no extremadamente compatibles con el Low-Code y No-Code. Aun así, ¿podría dar un buen rendimiento a pesar de las dificultades? Es una de las dudas que impulsan el caso de estudio de este trabajo.

De ninguna manera se podría afirmar las utilidades de estas tecnologías para todos los casos de uso. Cada situación debería estar sujeta a un previo estudio para determinar si es realmente útil utilizarlas en tu proyecto.

2.3 Entrevista con un experto

Aprovechando la ventaja que nos proporciona el ser alumnos de la Universidad Politécnica de Valencia, se decidió contactar con el Centro de Investigación en Métodos de Producción de Software (también conocido como PROS Research Center[1]).

PROS es un centro, creado en 2008, que centra su trabajo en la creación de estrategias, métodos y procesos que abordan el proceso de producción del software. En especial, nos llamaría la atención la línea de investigación del desarrollo dirigido por modelos y generación automática de código.

Uno de sus miembros es el Dr. Vicente Pelechano, con el cual se realizó una entrevista sobre el tema principal de este trabajo, las tecnologías Low-Code y No-Code. Para ello se plantearon una serie de preguntas.

1 - ¿Crees que las tecnologías Low-Code y No-Code podrían sustituir en algún momento al desarrollo de software tal y como lo conocemos?

La historia nos hace pensar que no. Sus predecesoras, por ejemplo, el desarrollo de software dirigido por modelos, generaron grandes expectativas, se usaron y aplicaron en entornos industriales (y se siguen usando) pero no acabaron con el desarrollo de software clásico**.**

¿O más bien complementarlo?

Exactamente, pueden complementarlo, ayudan claramente a ser más productivos, pero es difícil conseguir un reemplazo completo.

2 - ¿Cuáles consideras que son los límites de estas tecnologías? ¿hay hitos que no se pueden obtener o que son excesivamente complejos para desarrollarlos con ellas?

Hay muchos límites:

1) Implementación de NFR (requisitos no funcionales): usabilidad, seguridad, mantenibilidad, eficiencia, etc.

2) Evolución y mantenimiento del software

3) Personalización o adaptación del software al contexto

4) Desarrollo de grandes proyectos software (escalabilidad)

5) Software con mucho comportamiento

6) Integración de inteligencia en los sistemas actuales

7) Interoperabilidad e integración con otros sistemas (IoT por ejemplo).

[1] PROS Research Center – Página oficial: http://www.pros.webs.upv.es/

3 - Uno de los aspectos más críticos y costosos del proceso de producción del software es el mantenimiento. Parecen aplicaciones simples de crear, pero ¿cómo de difícil se plantea el reto de hacer que el rendimiento de dicha aplicación no se vea disminuido con el paso del tiempo?

El software actual está continuamente adaptándose y evolucionando, este es un problema difícil de gestionar en este tipo de soluciones (Low-Code). Mantener y/o actualizar este tipo de aplicaciones generadas es complejo porque muchas veces el propio producto no soporta fácilmente la evolución, o el mismo producto se ha vuelto obsoleto y hay que modificar la aplicación generada de forma manual.

4 - Se está convirtiendo en tendencia la preocupación por la seguridad y fiabilidad de las aplicaciones, lo cual es lógico. Lo que me lleva a la duda de; ¿a nivel de desarrollador, se carece de control total de la aplicación? Esto haría que estos aspectos no se vieran bien tratados.

Sí, exactamente, como he comentado anteriormente, hay requisitos no funcionales, como la seguridad y la fiabilidad que no están bien soportados por este tipo de herramientas/soluciones. Si este tipo de soluciones no te ofrecen un abanico amplio de opciones de configuración para poder producir software seguro y fiable, finalmente va a ser el experto desarrollador el que deba modificar la aplicación generada para incorporar aspectos de seguridad y fiabilidad no soportados. En este contexto el abanico de soluciones que pueden desarrollarse es limitado.

5 - Desde un punto de vista personal, ¿prevés que el uso de este tipo de herramientas se extienda o, por el contrario, entiendes que se trata de una moda puntual?

Como ingeniero de software e investigador en esta área me interesa que su uso se extienda, pero la experiencia o la historia reciente, sumada al conocimiento del tipo de aplicaciones a desarrollar ahora y en el futuro (con más inteligencia, más seguridad, más fiabilidad, más interoperabilidad, más usabilidad, distintos mecanismos de interacción, nuevos dispositivos, y nuevos requisitos o experiencia de usuario), puede que limite el potencial de este tipo de herramientas.

2.4 Diferencias y semejanzas con el desarrollo dirigido por modelos (MDD)

Una vez presentadas las tecnologías Low-Code, se puede apreciar que se asemejan a lo que se conoce como el desarrollo dirigido por modelos (MDD) [9], pero ¿en qué se diferencian?

Cuando se habla de MDD, se entiende que es una técnica de desarrollo donde el artefacto predominante a la hora de crear un proyecto son los modelos, a partir de los cuales se genera código. Se trabaja con modelos de notación UML o cercana a ella.

En cambio, Low-Code no trabaja de la misma manera que MDD, presenta entornos más centrados en el desarrollo a partir de elementos de la interfaz. No usa modelos como artefacto principal, basa el desarrollo en el uso de elementos gráficos y su parametrización.

Low-Code es una versión más restrictiva del MDD; una herramienta Low-Code sirve para un dominio de negocios mucho más concreto y que utiliza una serie de lenguajes predefinidos. En el caso del MDD, tanto la definición del lenguaje como el dominio se podrían llegar a especificar durante el proceso.

De esta manera, si Low-Code no innova en el aspecto técnico, ¿por qué es tan popular? A pesar de poder adaptarse a un menor número de casos de uso, las tecnologías Low-Code llegan a un público mucho más amplio. Low-Code apuesta por el diseño y usabilidad de sus herramientas, mientras que MDD aporta infraestructuras muchos más potentes, pero más difíciles de utilizar [13].

Figura 1: Interés a lo largo del tiempo basado en las búsquedas de Low-Code y MDD.

Durante la entrevista a Vicente Pelechano también se trató esta temática. Se concluyó que el Low-Code, al ser un concepto más candente en la actualidad (figura 1), es un término que subsume al MDD. Pero que, realmente, no se puede decir con seguridad que se traten de exactamente lo mismo.

Por otro lado, comentó lo siguiente: *"Yo diría que MDD es algo más amplio, y que* Low-Code *se basa en los principios de MDD, si algunos lo ven igual no es un gran problema, pero piensa que será el mismo perro con distinto collar, pero el perro (MDD) ya existía desde hace tiempo"*

3. Estado del arte

3.1 Estudios sobre la materia

Tanto las tecnologías Low-Code como las No-Code generan mucho interés en la actualidad por el cambio de paradigma que suponen para el desarrollo del software. El interés es compartido por una gran cantidad de espectadores; llama la atención la posibilidad de desarrollar sus aplicaciones de manera más sencilla. Muchos de ellos incluso tienen intereses económicos en este tipo de herramientas; podrían aumentar la productividad y calidad en el desarrollo de sus proyectos.

El compromiso con estas tecnologías se puede ver reflejado en la inversión que realizan las grandes empresas a la hora de adentrarse en este campo. Dicha inversión es patente en el desarrollo de herramientas de este tipo por parte de las grandes empresas [14]; un caso claro sería Power Apps[2] por parte de Microsoft o la adquisición de AppSheet[3] por parte de Google.

Las ventajas de estas tecnologías han sido tan populares que se prevé una inversión de 45 mil millones de dólares ($45.5 billion) para el año 2025 en las tecnologías Low-Code. Es importante anotar que en el año 2020 hubo una inversión de 13 mil millones de dólares ($13.2 billion), lo cual supone una tasa de crecimiento anual del 28.1% [15].

Otros de los grandes espectadores de estas herramientas son las entidades educativas, las cuales comienzan a desarrollar diferentes estudios e iniciativas que hacen que este ámbito se vea potenciado [16]. El estudio de las tecnologías hace que su evolución sea más sencilla.

Por otro lado, en el contexto actual existe la necesidad de transformación digital de los negocios. En esta transformación las tecnologías Low-Code pueden tomar un papel importante; por ejemplo, pueden llegar a permitir realizar una transformación doble vía. La transformación doble vía se basa en realizar una transición al mundo digital donde los esfuerzos de transformación se ejecutan en paralelo con los ciclos rápidos de innovación.

La posibilidad de utilizar Low-Code para ámbitos tan relevantes podría significar que la herramienta tiene un gran potencial, además de la capacidad de ser un complemento para proyectos trascendentes. Este hecho genera buenas sensaciones, ya que podría constituir un nuevo nicho de actuación de las tecnologías que estamos analizando.

Cabe añadir que, un estudio realizado por Harvard revela que el 52% de los encuestados en una consulta realizada piensan que unos de los principales beneficios de las tecnologías Low-Code (en la transformación digital) es la capacidad de hacer que los gerentes y altos cargos se involucren más en el proceso de innovación. No obstante, el 38% de los encuestados creen que las tecnologías Low-Code no serían lo

[2] Power Apps – Página oficial: https://powerapps.microsoft.com/es-es/
[3] AppSheet – Página oficial: https://www.appsheet.com/

suficientemente robustas para su negocio. Además de que el 29% piensa que dichas plataformas no estarían a la altura de lo que se promete.

Todo ello haría que solo el 14% de los encuestados se decidiera a comprar la tecnología para dicha transformación, la gran mayoría centraría sus inversiones en otras tecnologías [17]. Esto deja claro una de las preocupaciones que generan dichas herramientas: ¿generan la suficiente confianza como para implantarlas en tu negocio o necesitan de mayor inversión para convertirse en propuestas serias?

Teniendo en cuenta el conjunto de las expectativas que generan estas tecnologías y su potencial, surge otra cuestión: ¿es la nueva revolución del mercado, una forma más de enfocar el desarrollo de software o una mala idea? Todo parece indicar a la segunda de las opciones.

3.2 Crítica al estado del arte y propuesta

Como se ha podido apreciar en el anterior punto expuesto, se reflejan las grandes expectativas y el interés creciente en la sociedad, no obstante, existen algunos conceptos relacionados con este tipo de herramientas que quedan un poco desatendidos. Esta desatención es totalmente lógica porque, al tratarse de herramientas sin un amplio recorrido, aún no están asentadas de la misma forma que otras tecnologías. Este hecho deriva en que haya algunas preguntas y desafíos relevantes sin respuesta. Aquellos aspectos que se echan de menos son los posibles usos y límites de las diferentes herramientas actuales. No queda claro qué podemos obtener con estas herramientas y cuándo utilizarlas.

Son una amplia gama de preguntas que se responden mediante el estudio de carácter empírico y, en la gran mayoría de las veces, a nivel de usuario. Lamentablemente este tipo de tecnologías, aunque creciendo en importancia, no tienen una amplia base de desarrolladores/usuarios que las exploten diariamente.

En respuesta a esta falta de información se ha planteado este trabajo. Tras la realización pruebas y recogida de información sobre este tipo de tecnologías, se dieron a conocer conclusiones mediante un caso de estudio basado en retos de desarrollo de funcionalidades. Resultó ser una manera adecuada de resolver alguna de las dudas que surgían tras la primera toma de contacto con estas tecnologías.

Con el fin de conseguir los objetivos de este trabajo resulta relevante conocer en profundidad las tecnologías Low-Code y No-Code; por lo que informarse sobre sus puntos fuertes y débiles es importante.

- **Ventajas y desventajas de las tecnologías Low-Code**

Una de las ventajas más reconocidas de este tipo de herramientas es la velocidad con la que se puede obtener un producto. La parte negativa de esto es que la rapidez con la que se realiza todo puede mermar el pensamiento estratégico; el usuario entiende que va a ser una solución rápida y podría llevar a ignorar algunos conceptos.

Otras de las ventajas que destacan son la capacidad de empoderamiento, la simplicidad o la consistencia. Siendo el empoderamiento la capacidad que se les da a los empleados de desarrollar aplicaciones, sin ser necesario que pertenezcan al departamento tecnológico o que tengan conocimientos técnicos previos. Lo cual puede promover el ingenio de los empleados de la línea de negocios. La simplicidad hace referencia a que las herramientas de este tipo están ideadas para realizar un desarrollo más fácil para el usuario, preocupándose únicamente por el negocio a desarrollar, sin dejarse llevar por aspectos técnicos. Por último, cuando se habla de consistencia, se hace referencia a la capacidad de dichas herramientas para proporcionar fácilmente soluciones a los desafíos estándar que suelen aparecer; haciendo que se proporcionen herramientas para realizar sin dificultad aquello que suele aparecer con más frecuencia, automatizan aquello que se repite.

Si se habla de las desventajas, el realizar las aplicaciones de la misma manera y procedimiento que tus competidores puede condenar tu aplicación a ser una más, sin capacidad de destacar entre la multitud de soluciones propuestas por otros rivales.

Otra desventaja podría ser la unión del destino de una aplicación con la empresa de la herramienta usada; cualquier cambio o contratiempo que suceda en la empresa que proporciona el servicio puede verse reflejado en tu proyecto.

Además de promover confusiones que surgen de la no información de la herramienta en términos de los errores que sucedan. Estas herramientas apuestan por el hermetismo, se puede ahorrar mucho tiempo si todo va bien, pero si algo fuera mal no proporcionaría la suficiente información para controlar y conocer el error [18].

- **Ventajas y desventajas de las tecnologías No-Code**

En cuanto a las ventajas, destacan el ahorro en términos de coste, el aumento de la productividad o la facilidad a la hora de cambiar el comportamiento de la aplicación. Se pueden realizar proyectos con bajo coste temporal, los cual puedes modificar y cambiar su forma de actuar de manera muy rápida.

Si nos centramos en las desventajas, llama especialmente la atención la rigidez de las plantillas en las que construir las aplicaciones, los problemas de seguridad o el hecho de no ser propietario del código fuente. Este tipo de herramientas permiten construir aplicaciones siguiendo una forma de actuar muy determinada. Por lo que esto, junto a no ser propietario del código, genera sensación de falta de control sobre la aplicación.

Por otro lado, como ya se ha mencionado, pueden generarse problemas de seguridad; estos surgirían en el caso de que la herramienta No-Code fuera vulnerada. Esto tiene relación con lo que se ha comentado anteriormente en las plataformas Low-Code, hay una dependencia muy grande con la empresa que proporciona el servicio; cualquier vulnerabilidad generada en ella, podría reflejarse en la aplicación creada. Por ello, sería esencial la elección de una herramienta en la que la empresa desarrolladora genere una confianza óptima; hecho que se verá reflejado en la elección de la herramienta a usar en el caso de estudio [19].

Las conclusiones a las que se llegan en los artículos disponibles son poco detalladas, aunque acertadas. Se trata de una contextualización para todos los públicos; una divulgación de estas tecnologías, sin entrar en aspectos técnicos.

3.3 Conclusiones

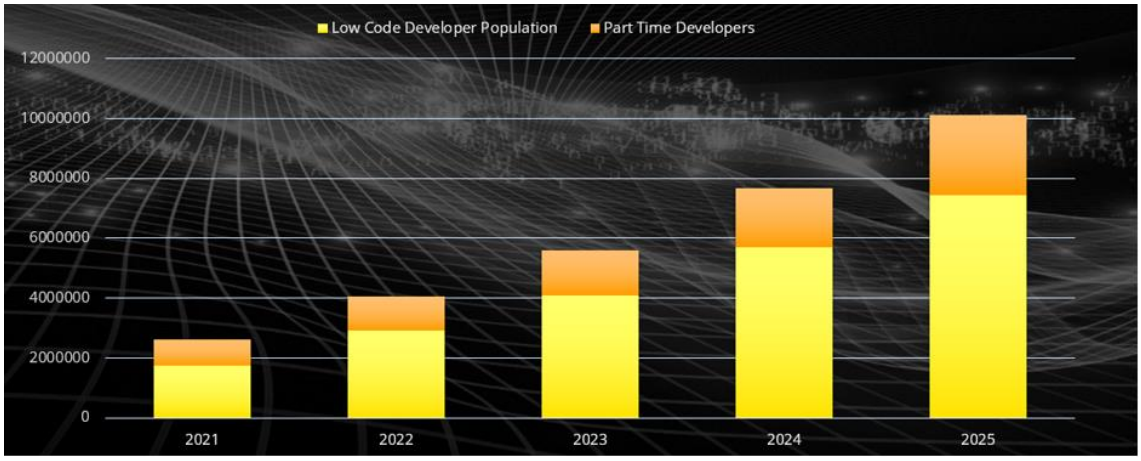


Figura 2: Cantidad de usuarios de las tecnologías Low-Code previstos para los próximos años [20].

En la actualidad, tanto las tecnologías Low-Code como las No-Code exponen una gran cantidad de puntos a mejorar. No obstante, como se puede ver en la figura 2, desde IDC pronostican que el número de usuarios de las tecnologías Low-Code y No-Code van a aumentar en los próximos años; siendo muchos de ellos personas que no tienen conocimiento técnico previo.

La posible inclusión de usuarios finales en el proceso de producción del software es un aspecto muy positivo. Muchos de los errores que se producen por parte del equipo de desarrollo suelen por la ambigüedad de los requisitos; por lo que involucrar a usuarios finales podría ayudar a acercarnos al mundo real y, por tanto, evitar fallos.

De esta manera, y pese a las desventajas de estas tecnologías, se augura un crecimiento tanto en inversión como en masa de desarrolladores [20].

4. Herramientas Low-Code y No-Code

4.1 Herramientas Low-Code

Para el estudio de las herramientas Low-Code se hizo uso de una gran cantidad de estudios y artículos. Todo ello, junto a la experiencia personal en cada una de las herramientas que se iban estudiando, hicieron que se pudieran escoger las más llamativas. Se priorizaron aspectos como la facilidad de aprendizaje de la herramienta, el entorno de trabajo que proporciona, la capacidad para formarse en la herramienta o el precio de esta.

Con el motivo de realizar una selección fundada, se decidió probar prácticamente todas las tecnologías que recomendaban; además de realizar un estudio sobre sus principales puntos fuertes y cursar formaciones que proporcionaban desde las mismas entidades.

Figura 3: Cuadrante clasificatorio de las principales tecnologías Low-Code [21].

En la figura 3 se muestra el "***Magic Cuadrant***" desarrollado por Gartner. En él se puede ver el estado de las principales herramientas del mercado. Al ver estos resultados, generó especial interés estudiar y desarrollar diferentes proyectos con las herramientas etiquetadas como líderes.

Haciendo uso de su versión de prueba se pudo tener una impresión general del conjunto de estas. Se optó por realizar pruebas sin abonar dinero ya que se prefirió probar una cantidad muy grande de herramientas.

Resultaron muy interesantes Mendix[4], Appian[5], Salesforce[6], etc. No obstante, aquellas que resultaron más llamativas y que, por tanto, se analizarán con más detenimiento fueron OutSystems[7] y Power Apps [21].

[4] Mendix – Página oficial: https://www.mendix.com/
[5] Appian – Página oficial: https://appian.com/
[6] Salesforce – Página oficial: https://www.salesforce.com/es/
[7] OutSystems – Página oficial: https://www.outsystems.com/

- **Power Apps**

Se trata de una herramienta Low-Code desarrollada por Microsoft. Se podría definir como un conjunto de servicios que proporcionan un marco de desarrollo de aplicaciones, con el fin de generar soluciones personalizadas para una empresa. Está pensada para realizar aplicaciones para tu propio negocio a partir de datos, no está enfocada para hacer aplicaciones a terceros [22].

Haciendo referencia al entorno, es muy similar a un marco ofimático, debido a que presenta un entorno que resulta muy familiar a aplicaciones como PowerPoint[8]. Dicho entorno, permiten diseñar y desarrollar todos los aspectos de la aplicación de manera sencilla e intuitiva. Además, el uso de la herramienta no conlleva ninguna descarga de aplicación, de manera que se podría utilizar en cualquier lugar siempre y cuando se disponga de un ordenador conectado a internet.

En cuanto al modo de ejecución de las aplicaciones, es bastante particular en relación con las otras herramientas. Como principal aspecto distintivo, vemos que se podría probar el comportamiento de la aplicación sin compilar ni realizar ningún paso previo, desde la misma interfaz del diseño.

La segunda opción de ejecución que ofrecen es la descarga de una aplicación en un teléfono móvil. En ella se almacenan las diferentes aplicaciones desarrolladas expuestas en una lista. Por lo que, para probarla en el mismo dispositivo, simplemente habría que seleccionar la aplicación deseada y comenzaría a ejecutarse, sin ningún protocolo adicional.

Existe una tercera forma de ejecución, Power Apps ofrece la posibilidad de ejecutar la aplicación apretando un botón. Una vez presionado, se comenzaría a ejecutar dicha aplicación, sin necesidad de salir del entorno de desarrollo.

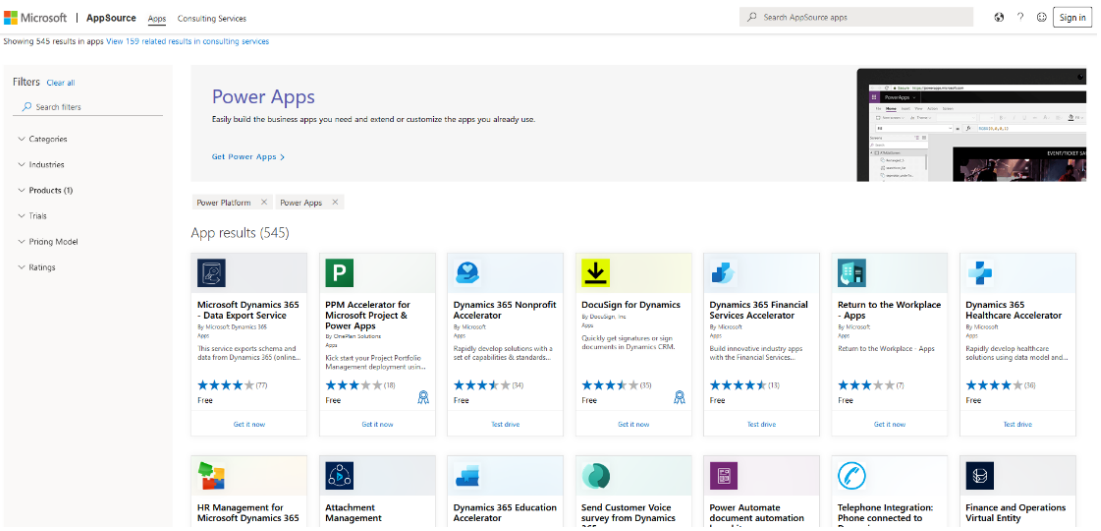


Figura 4: Filtrado de las aplicaciones para Power Apps en AppSource.

Si hablamos del *marketplace* y los *plugins* de la herramienta, basta con realizar la búsqueda que se realiza en la figura 4 para poder apreciar las más de 500 opciones de

[8] PowerPoint – Página oficial: https://www.microsoft.com/es-es/microsoft-365/powerpoint

personalización y extensión ofrecidas. Cabe resaltar la gran cantidad de opciones gratuitas que se proporcionan.

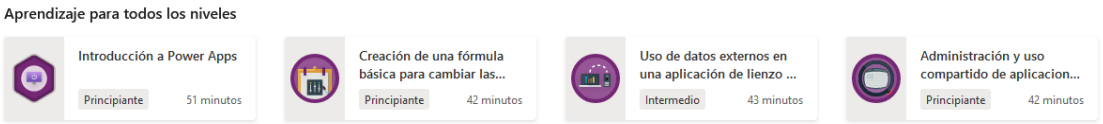


Figura 5: Cursos sobre Power Apps organizados por niveles de dificultad.

El nivel presentado por esta herramienta en referencia a los cursos y material de formación es bastante bueno. En la pantalla principal de la herramienta proporcionan una muestra de algunos cursos, como se puede ver en la figura 5.

Existen diversos cursos breves y detallados de diferentes niveles. Además, proporcionan una gran cantidad de documentación. En ella se explican conceptos esenciales a la hora de generar un proyecto, desde cómo crear aplicaciones hasta la descripción de las diferentes funciones predeterminadas.

Desde Microsoft presentan dos planes de pago. Un primer plan en el que se paga mensualmente una cuota de alrededor de 9 euros por el desarrollo de una sola aplicación en dicha plataforma; es un coste por aplicación, por ejemplo, de querer dos aplicaciones, se tendría que pagar 18 euros mensuales. Por otro lado, se puede pagar una cuota de alrededor de 35 euros al mes, si se desea desarrollar un número ilimitado de aplicaciones.

- **OutSystems**

En segundo lugar, OutSystems es una plataforma Low-Code que proporciona las herramientas y soluciones para poder desarrollar, administrar e implementar aplicaciones de carácter empresarial [23].

Respecto al entorno y modo de trabajo, en este caso sería necesario la descarga de una aplicación en la que podremos comenzar a trabajar.

Si nos centramos en el protocolo a seguir cuando se desea simular una ejecución bastaría con llevar a cabo una breve serie de pasos. Antes de ejecutar el prototipo realizado, se ha de comprobar si la aplicación es válida.

Una vez comprobado, simplemente se podría lanzar la aplicación en un navegador, sin necesidad de simuladores, ni un gasto de RAM excesivo. Esta sería muy similar a la tercera opción de ejecución que nos proporciona Power Apps, pero ejecutando la prueba fuera del entorno de desarrollo.

En cuanto al *marketplace* y los *plugins*, OutSystems también proporciona servicio en esta línea. Existe un *marketplace* denominado Forge, el cual proporciona componentes extra para móviles y aplicaciones web.

Las diferentes opciones que proporciona Forge ayudaron bastante durante el periodo de prueba de la tecnología, haciendo que se ahorrara una gran cantidad de tiempo

debido a que, funcionalidades que se querían incluir, ya estaban desarrolladas y disponibles.

Si el interés es exprimir al máximo el potencial de OutSystems, se da la posibilidad de acceder a diferentes cursos de formación. Existe un espacio en el que proporcionan cursos formativos, junto a una gran cantidad de información y documentación. Dichos cursos se componen, mayoritariamente, de videos formativos y de un posterior cuestionario.

En cuanto al precio, encontramos una distinción bastante notable; la versión de prueba es gratis sin límite de tiempo. Si bien es cierto que es una versión no completa en funcionalidad, igualmente permite probar en gran medida las posibilidades de OutSystems.

Para disfrutar de ediciones más completas, sería necesario pagar una cuota mensual que puede ir desde los 3500 hasta los 8500 euros mensuales. O, incluso, permite la configuración de una edición personalizada y, por tanto, de precio personalizado.

Por otro lado, comenzando con la exposición de razones de elección de dicha tecnología como buena opción, se mencionarán aquellos aspectos llamativos y de interés que se han ido encontrando. Como primer aspecto, es importante resaltar el entorno de la herramienta. En él, por ejemplo, se pueden consultar de manera gráfica los diferentes componentes de una solución e incluso acceder al esquema de la base de datos.

Además, un aspecto especialmente llamativo, es la posibilidad de trabajar de forma colaborativa. Un ejemplo podría ser que, si dos compañeros modifican el mismo elemento simultáneamente, en el momento de subir las versiones al repositorio se entrará en estado de colisión. Lo interesante es que, en respuesta a esta problemática, permite realizar acciones de comparación y de combinación de cambios entrantes. Lo cual nos recuerda bastante al uso de un sistema de control de versiones.

Otro aspecto relevante es la facilidad de tratamiento de datos. A la hora de incluir una capa de persistencia, se puede incluir una base de datos de manera muy simple. Tal como, podría importarse de un Excel[9]; generando una base de datos bien estructurada con una sola entrada y de forma automática. Además, también puedes consultar la estructura generada sin necesidad de salir de la aplicación y de forma bastante detallada. Es importante debido a la importancia que toman los datos dentro de una aplicación, siendo, en ocasiones, el eje principal de esta.

En tercer lugar, es bastante útil el hecho de poder generar aplicaciones IOS y Android sin variar el procedimiento. Es decir, una vez planteada la aplicación, podrías tener el resultado implementado en dos plataformas diferentes, sin conllevar un gasto de tiempo adicional.

Se trata de una de las aplicaciones más completas que se han probado; ya que permite, con previa formación (que la puede proporcionar la misma compañía), elaborar soluciones muy potentes. No obstante, esta capacidad de realizar grandes proyectos, también se ve reflejada en una mayor dificultad a la hora de aprender a usarla.

[9] Excel – Página oficial: https://www.microsoft.com/es-es/microsoft-365/excel

Proporcionan funciones muy completas, sin embargo, es necesaria una cantidad de tiempo considerable para dominarla.

- **Conclusiones**

En cuanto a OutSystems, sus principales puntos fuertes son la experiencia de usuario y su arquitectura. Contiene una gran variedad de plantillas y patrones en la interfaz de usuario, además de proporcionar componentes de creación de aplicaciones personalizables. En adición, también permite crear patrones de interfaz de usuario propios y utilizarlos. Su arquitectura es beneficiosa debido a que permite a las aplicaciones trabajar con varios servicios y fuentes de datos.

Por otro lado, como principales puntos débiles tenemos la estrategia de producto y el desarrollo de aplicaciones por parte de usuarios no expertos. Outsystems tiende a favorecer a los desarrolladores profesionales, no facilitan el desarrollo por parte de personas sin conocimientos previos sobre aspectos relacionados con la ingeniería del software.

Si hablamos de Power Apps, cabe resaltar la estrategia de producto, la innovación y la capacidad de integrar servicios. Se trata de una herramienta que simplifica mucho el diseño, ofreciendo un amplio conjunto de APIs y *endpoints*. En términos de innovación también se encuentra bastante avanzada, ofrece a los desarrolladores el poder consumir modelos de IA sin la necesidad de entrenar ningún modelo.

También existen desventajas, como podrían ser la capacidad de respuesta del mercado y los precios. En un inicio, Microsoft incluyó en las ofertas más populares de las licencias la capacidad de usar Power Apps. Tiempo después se cambiaron esas condiciones, interrumpiendo el servicio de la herramienta para muchos de los clientes.

Además de otros detalles como que el uso de flujos de procesos empresariales está restringido. Se pueden utilizar, pero solo para las aplicaciones basadas en modelos [21].

4.2 Herramientas No-Code

Siguiendo con las dinámicas expuestas a la hora de analizar las herramientas Low-Code, se mostrarán las principales herramientas del mercado teniendo en cuenta tanto la experiencia personal como el conjunto de artículos y estudios de las herramientas No-Code. Los criterios de elección de las herramientas son los mismos que los expuestos para las herramientas Low-Code.

Resultado de las pruebas, parecieron interesantes Appsheet, Airtable[10], IFTTT[11], Bubble[12], etc. No obstante, aquella que resultó más llamativa fue Webflow[13] [24].

[10] Airtable – Página oficial: https://www.airtable.com/
[11] IFTTT – Página oficial: https://ifttt.com/
[12] Bubble – Página oficial: https://bubble.io/
[13] Webflow – Página oficial: https://webflow.com/

- **Webflow**

Webflow es una herramienta de desarrollo No-Code que ofrece un entorno visual de desarrollo web y de gestión de contenido. Todo esto permite, como usuario, construir y lanzar sitios web [25].

Si hablamos del entorno y el modo de trabajo, para comenzar a utilizar la herramienta no hacen falta descargas ni aplicaciones adicionales. Bastaría con registrarse en la misma web y acceder al sitio que habilitan. En dicho sitio, se tiene a disposición un repositorio de proyectos en el que almacenar las diferentes páginas web que se crean.

Por otro lado, para realizar una ejecución de la solución únicamente se tendría que lanzar el proyecto mediante un botón habilitado en la misma interfaz y, siempre y cuando no haya un error bloqueante, se realizará una simulación.

Si centramos la redacción en el *marketplace* y los *plugins*, nos encontramos con ciertas peculiaridades, en comparación con las otras tecnologías analizadas. Disponemos de un *marketplace* que se basa en una tienda en la que podemos adquirir plantillas para la página web. Dichas plantillas son esquemas que se pueden tomar como base del proyecto.

Haciendo referencia a la capacidad de formarse en esta tecnología, existirían también diferentes caminos para la obtención de dicho conocimiento. En un primer lugar, al iniciar el reto de realizar una web ofrecen una guía inicial, paso a paso, hasta conseguir un resultado consistente; combina la realización de pasos sencillos con la consecución de un resultado satisfactorio.

Además de la guía mencionada, también se ofertan cursos que, a diferencia de otros competidores, resultan bastante amenos y fáciles de seguir. Esto se debe a que combinan un tono humorístico con contenido adecuado y una producción de vídeo de calidad.

Otros de los aspectos en los que también dista de otras tecnologías son la forma de pago y el coste de utilización de la tecnología. Se trata de una filosofía diferente, el comenzar a crear soluciones es totalmente gratuito, sin tiempo de prueba ni introducir una tarjeta de crédito.

Habría que pagar una cuota mensual únicamente cuando se quiera lanzar dicha web al mercado; proporcionando servicios como un dominio personal. Lo que hará variar el precio serán aspectos como, por ejemplo, el número de visitas, a más visitas más precio.

El aspecto que más llama la atención es la relación que tiene la herramienta con el código. Permite exportar el equivalente a lo que se ha realizado visualmente en código. Haciendo así que, aparte de crear webs, también pueda servir como complemento en la construcción de un proyecto.

En adición a esto, permite realizar una vista previa y a tiempo real del código a la vez que se realizan cambios. Esto haría sentir un mayor control sobre lo que se está desarrollando; viendo el impacto de los cambios ejercidos, a bajo nivel.

Se ha de destacar también la buena implementación del CMS en la herramienta. CMS es un sistema de gestión que permite crear y administrar contenidos. Esto resulta útil para crear estructuras, cuyo contenido es personalizable.

Por último, cabe resaltar la buena aceptación que dispone Webflow entre sus usuarios. Esto lo podemos ver patente en hechos como que en grandes empresas (como Rakuten) están apostando por esta tecnología en sustitución a la ya conocida WordPress[14].

4.3 Herramienta del caso de estudio

Para abordar el caso de estudio hubiera resultado idóneo desarrollar las diferentes funcionalidades planteadas con las herramientas más llamativas del mercado, no obstante, en este trabajo existen una serie de limitaciones de alcance.

Por ello, se optó por centrar el estudio únicamente en Power Apps. El desarrollo de las funcionalidades del caso de estudio con otras herramientas tendrá que ser planteado para trabajos futuros.

[14] WordPress – Página oficial: https://wordpress.com/es/

5. Caso de estudio

5.1 Contexto

Para el caso de estudio se ha planteado utilizar Power Apps. Usándola se puede demostrar el potencial de esta tecnología, intentando averiguar si es posible desarrollar algunas funcionalidades con su ayuda. Para demostrar que una tecnología es válida e interesante en un contexto determinado, es esencial saber su estado en relación con las otras opciones que tenemos en el mercado.

Por este motivo se plantea la realización de desafíos de desarrollo, los cuales se basan en realizar una serie de funcionalidades típicas de una aplicación. Estos desafíos serán desarrollados utilizando tecnologías tradicionales y Power Apps.

Las tecnologías tradicionales son aquellas que, para desarrollar una aplicación, exigen elaborar código. Como exponentes de aplicaciones elaboradas con este tipo de tecnologías disponemos de Adventour y Twins, las cuales fueron proyectos realizados durante el grado. De estos desarrollos existe un histórico almacenado en la herramienta de gestión ágil del trabajo, Worki[15]. Este histórico contiene aspectos como los tiempos empleados en cada una de las partes implementadas, priorización de los aspectos más importantes, etc.

Twins es una aplicación que plantea un juego de memoria bastante popular, el cual se basa en la realización de parejas. La dinámica del juego es recordar la posición de las cartas en un tablero para, a lo largo de la partida, realizar parejas de cartas iguales; una vez se aciertan todas las parejas de un tablero, el juego termina y se asocia una puntuación y logros.

Como es evidente, la aplicación recoge más aspectos como podrían ser la implementación de un menú, un configurador de baraja, un modo multijugador, un ranking de puntuaciones, etc. Es importante añadir que en Twins se utilizó el lenguaje de programación Java y Android Studio[16] como entorno de desarrollo.

Por otro lado, **Adventour** es una aplicación turística. Tiene como principal objetivo planear rutas en la ciudad que se desee, seleccionando los diferentes puntos de interés por los que pasar y la duración deseada de la ruta. La creación y trazado de rutas está soportado con ayuda de la API de Google Maps. Es relevante matizar que en Adventour se utilizó Dart como lenguaje de programación y Visual Studio[17] como entorno de desarrollo. Además, se utilizó el kit de desarrollo software Flutter[18].

[15] Worki – Página oficial: https://cliente.tuneupprocess.com/web/#/login
[16] Android Studio – Página oficial: https://developer.android.com/studio
[17] Visual Studio – Página oficial: https://visualstudio.microsoft.com/es/
[18] Flutter – Página oficial: https://flutter.dev/

Al igual que la aplicación anterior, se recogen más aspectos en la aplicación a parte de la creación de rutas. Se incluyen funcionalidades como un sistema de usuarios con su correspondiente login, historial de las rutas, perfiles de usuario, sistema de logros y reconocimientos, etc.

Los desafíos que se proponen se basan en funcionalidades que se encuentran en las dos aplicaciones mencionadas. Por lo que el propósito del caso de estudio es la realización de dichos desafíos en Power Apps, almacenando información relacionada con qué se ha hecho, cómo se ha hecho y cuánto ha costado. Esto nos concedería la oportunidad de ver, en diferentes términos, las desemejanzas entre ambas propuestas.

5.2 Twins y Adventour usando Power Apps

Con el fin de desarrollar las funcionalidades de Adventour y Twins con Power Apps, se van a realizar algunas aclaraciones básicas en relación con la herramienta Power Apps y su entorno. Todo aquello que exponemos es en base a la versión 3.21051 de la herramienta.

Si nos centramos, en primer lugar, en su entorno, al entrar en la herramienta nos encontramos un aspecto como el que se muestra en la figura 6.

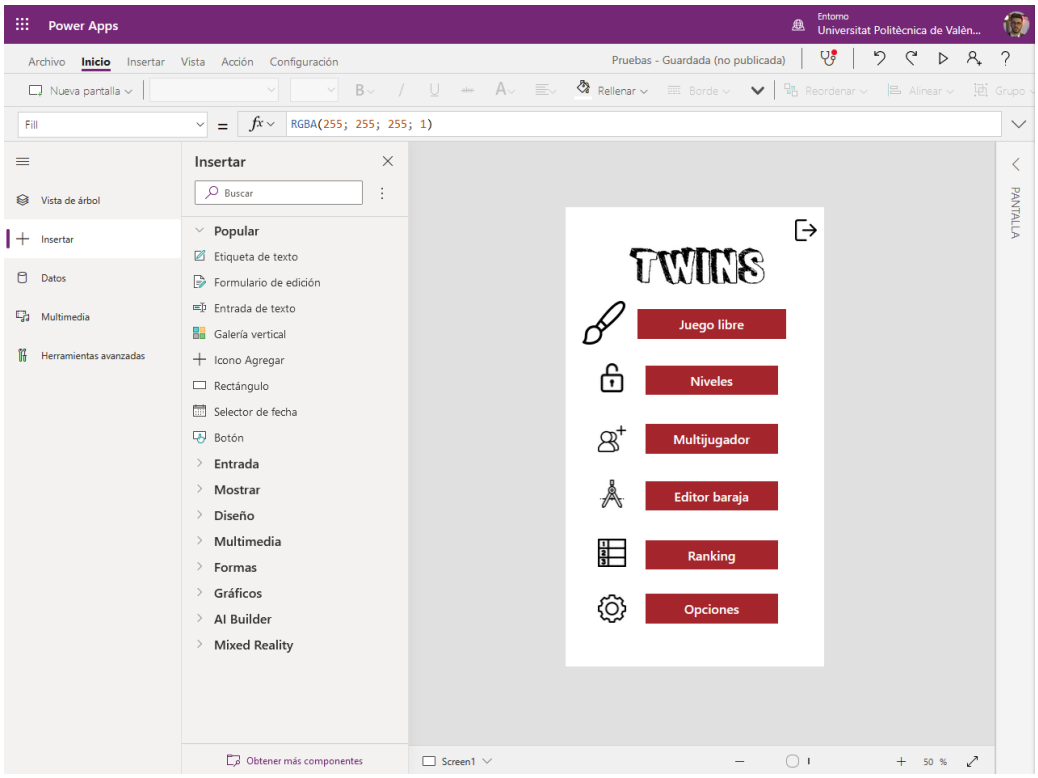


Figura 6: Entorno inicial de Power Apps.

Como se puede ver, es un entorno caracterizado por la clasificación de las diferentes funciones en distintos apartados.

En la figura 7 se muestra el primer apartado denominado **Vista de árbol**; este nos mostraría de manera esquemática las diferentes pantallas de las que disponemos. En cada una de las pantallas se puede consultar los distintos componentes que se encuentran en su interior. En este apartado podríamos crear nuevas pantallas, modificar los nombres de tanto pantallas como componentes, ordenar los componentes o pantallas en base a cierta jerarquía, entre otras acciones.

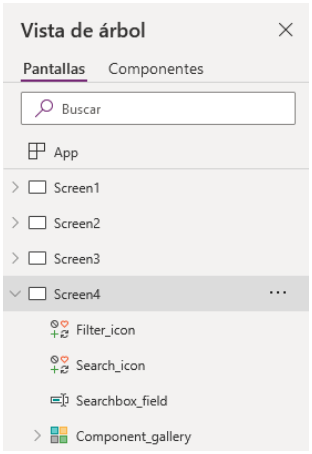


Figura 7: Apartado de vista de árbol.

Como se puede ver en la figura 6, tenemos un apartado llamado **Insertar**. En él se dispone de una serie de componentes que podríamos incluir. Desde este mismo apartado se pueden añadir componentes a la pantalla, arrastrándolos hacia el lugar donde los queremos incluir.

En el apartado **Datos**, mostrado en la figura 8, se podrían gestionar las diferentes fuentes de datos que podemos utilizar durante el desarrollo de la aplicación. Esas fuentes se pueden obtener mediante las diferentes conexiones que realizamos con los servicios de alojamiento de archivos.

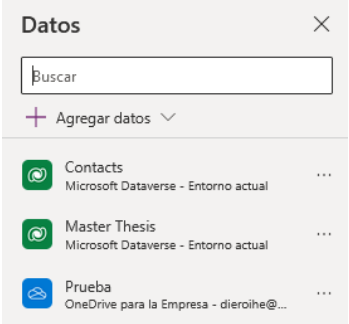


Figura 8: Apartado de las fuentes de datos.

Figura 9: Apartado multimedia.

La figura 9 ilustra el apartado **Multimedia**. En él se gestiona el uso de elementos multimedia en nuestra aplicación. Desde esta interfaz se podrían subir los diferentes objetos multimedia a utilizar (imágenes, audio, vídeos...), además de poder ver todos esos elementos mediante una lista; clasificados por el tipo de elementos que son.

En el último apartado, denominado **Herramientas avanzadas**, podríamos hacer uso de las funciones más novedosas y vanguardistas de la aplicación. Es por ello por lo que algunas de ellas se encuentran en fase experimental. No constituye un punto fundamental de la herramienta, se puede desarrollar una aplicación completa sin usar ninguna función de este apartado.

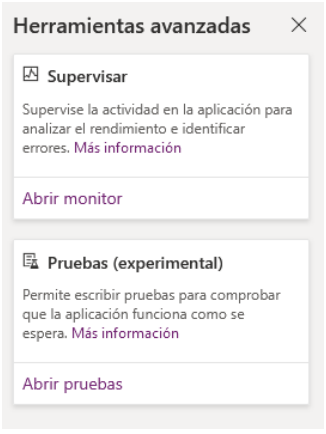


Figura 10: Apartado de herramientas avanzadas.

Una vez que se insertan los elementos en pantalla, tendríamos la posibilidad de modificar sus propiedades. Este tipo de acciones se pueden realizar desde la interfaz que nos proporcionan en el momento que seleccionamos un objeto. En la figura 11 se puede ver una interfaz con cada uno de los factores que se pueden modificar de dicho objeto; la personalización de objetos se realiza mediante la parametrización de las variables que te muestran.

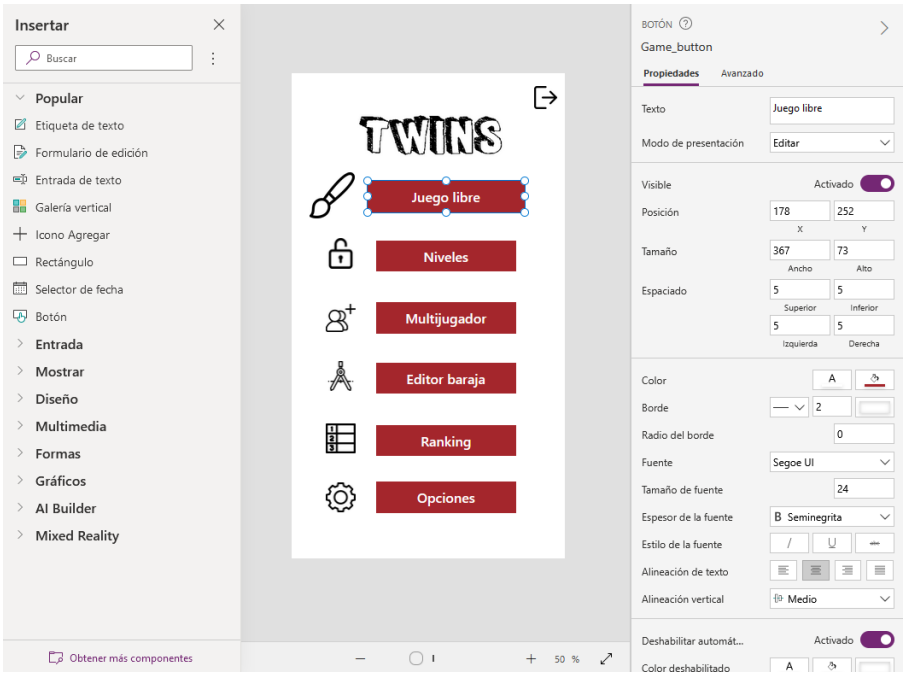


Figura 11: Interfaz de las propiedades de un objeto.

Otras de las propiedades de los objetos que podemos modificar las podemos ver en la barra superior. En ella, disponemos de otros factores que se pueden modificar expuestos en una lista desplegable.

La figura 12 muestra la propiedad *OnSelect* (en el momento que se pulse el elemento de la interfaz, se ejecutará aquello que se encuentre en el valor de dicha propiedad). El valor de la propiedad se puede especificar a su derecha. Actualmente se encuentra con valor "false", esto haría referencia a que no se debe realizar nada si se pulsa en dicho elemento. De querer incluir alguna funcionalidad, deberíamos añadir una fórmula en dicho campo.

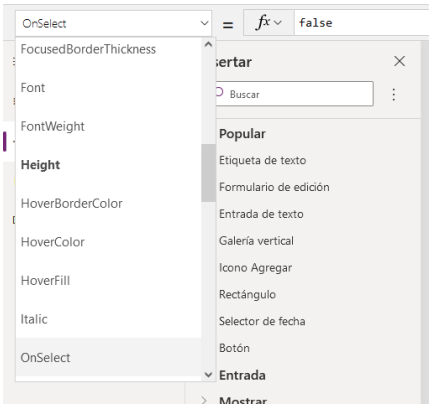


Figura 12: Propiedad OnSelect.

La fórmula que podemos incluir, la cual se calculará para la propiedad en cuestión, puede estar compuesta por valores, operadores y funciones. Las funciones son muy utilizadas: toman una serie de parámetros, con los cuales realizan una operación y devuelven un valor. Disponemos de una amplia variedad de funciones, todas ellas se pueden consultar desde la documentación oficial de Power Apps [26].

5.3 Diseño y navegación

El diseño de interfaces y la navegación entre ellas constituyen uno de los grandes puntos a la hora de realizar una aplicación; todo lo que se refiere a la estética conlleva un peso especial para el usuario final.

Ante la gran cantidad de oferta que disponemos en el mercado hay que cuidar cada detalle, en especial aquellos que son de **primer impacto**; "lo que nos entra por los ojos". Lo estético constituye, en especial para los usuarios menos interesados por los aspectos técnicos, uno de los aspectos más relevantes a la hora de consumir cierto producto. Se trata de un aspecto distintivo, por ello se ha planteado este desafío.

Para realizar una comparación correcta, se ha apostado por hablar desde la experiencia. Muchas veces nos guiamos por las especificaciones de ciertos productos y sus atributos, pero el aspecto diferencial se encuentra en la experiencia de usuario.

De esta manera se presentan dos pruebas. En primer lugar, se realizará el menú que se planteó en la aplicación de Twins. Y, por otro lado, se elaborará la pantalla de login que implementamos en la aplicación de Adventour.

- **Menú de la aplicación Twins**

Este primer reto se basaba en replicar la interfaz de origen. En base a la interfaz de Twins se trató realizar una interfaz de complejidad similar, es decir, con los mismos componentes y configuraciones.

Figura 13: Interfaz de origen del menú de Twins.

Figura 14: Réplica de la interfaz del menú de Twins.

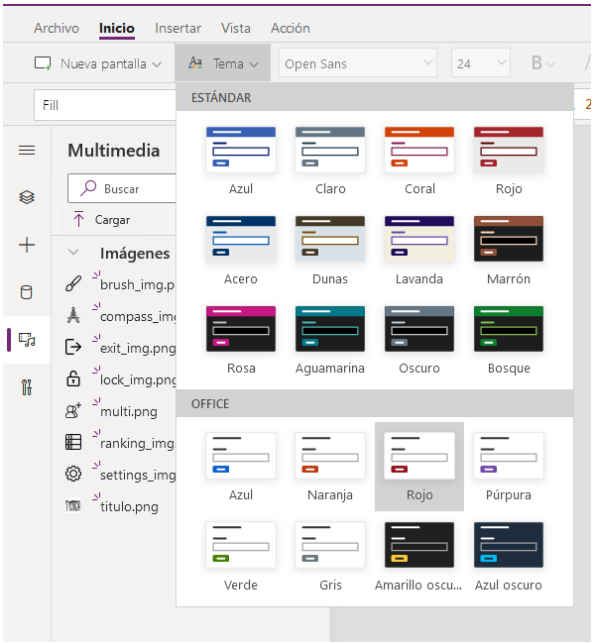


Figura 15: Elección de la temática de la aplicación.

Tras la experiencia, se percibe que la opción de Power Apps proporciona más facilidades para personalizar la aplicación. En cuestión de dos clics podríamos elegir el tema de la aplicación, este contiene un formato y presentación de los diferentes componentes de la interfaz.

Se trata de una solución rápida para elegir el tema de la aplicación. Como se ve en la figura 15, disponemos de una gran cantidad de opciones predeterminadas. Estas proporcionan resultados bastante convincentes si los comparamos con el gasto de tiempo que conllevan. En la réplica se utilizó la temática roja.

La forma de personalización que se siguió en Twins consumía bastante más tiempo; se editaban los diferentes objetos en los XML. No obstante, a nivel de desarrollador, da la sensación control y de un alto grado de personalización.

Es importante decir que ese grado de personalización no es inalcanzable en Power Apps, simplemente proporcionan formas más rápidas de personalización. Para este caso, basta con usar una de estas configuraciones predeterminadas para obtener un resultado similar.

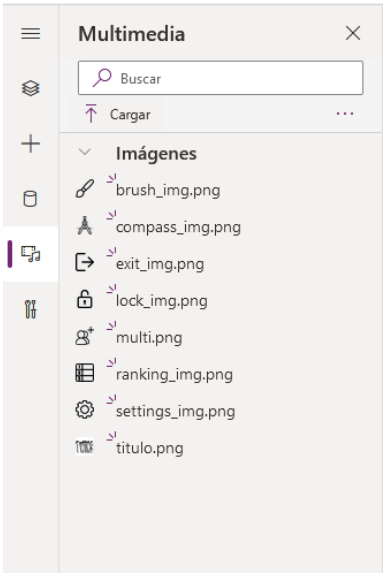


Figura 16: Carga de las imágenes necesarias para elaborar la réplica.

La inclusión de imágenes se hace de manera similar en ambas aplicaciones. En Android Studio se dispone de carpeta de recursos en la que ir almacenando y gestionando tus activos, entre los que se incluyen las fotos. Una vez incluidos en dicha carpeta se tendría que hacer referencia a esa imagen vía código o con el editor gráfico.

En Power Apps también es necesario subir las imágenes a un apartado multimedia, de manera que se obtiene un resultado como el de la figura 16.

Pulsando el botón Cargar, se puede incluir las imágenes a dicho menú para su posterior uso.

Para hacer uso de los elementos multimedia, bastaría con referenciar esa misma imagen en la propiedad *Image* del objeto en el que deseamos incluirla (figura 17).

Figura 17: Referenciación de la imagen en el objeto.

Por otro lado, cabe añadir que Android Studio dispone de un editor gráfico en el que incluir los objetos de forma sencilla. Sin embargo, durante el diseño genera una mayor cantidad de problemas. Con su uso surgen una gran cantidad de errores y, sobre todo, *warnings* en el uso de este editor. Muchos de estos problemas están relacionados con la falta de dependencias entre objetos, es decir, si falta situar un objeto en base a otro surge un fallo. En otras palabras, para situar un objeto en pantalla hace falta relacionarlo con otro; lo cual no aumenta la complejidad, pero es un protocolo repetitivo.

Por su parte, Power Apps proporciona un entorno mucho más sencillo; siguiendo una filosofía de "*drag and drop*" [27]. Dicho de otra forma, reduce la complejidad, haciendo que la duración del diseño también disminuya.

De esta manera, se dedicó menos tiempo a la elaboración de la réplica. Para el planteamiento inicial con Android Studio se registraron 47 minutos en Worki. Por otro lado, en Power Apps el diseño dura 25 minutos.

Por último, la navegación entre interfaces se realiza de manera más simple en Power Apps. Haríamos uso de la función *Navigate* en la propiedad *OnSelect* (figura 18). Solo sería necesario proporcionarle, como parámetro, el nombre de la interfaz a la que se desea transitar.

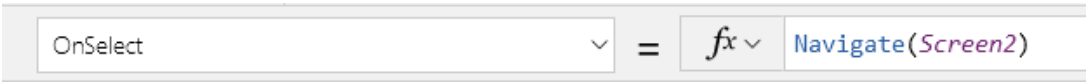


Figura 18: Uso de la función Navigate en la propiedad OnSelect.

Mientras que en Android Studio es necesario implementar *listeners* (figura 19). *Navigate* es una sentencia sencilla de una línea y un *listener* conlleva protocolo en su interior. Esto se traduce en un tiempo mucho menor para la plataforma Low-Code.

```
buttonRanking.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent i = new Intent(MainMenuActivity.this, RankingActivity.class);
        startActivity(i);
    }
});
```

Figura 19: Uso de un listener para la navegación entre interfaces en Android Studio.

- **Login de la aplicación Adventour**

Siguiendo con la misma filosofía que el ejemplo anterior, en este caso replicamos la pantalla del *login* de la aplicación de Adventour.

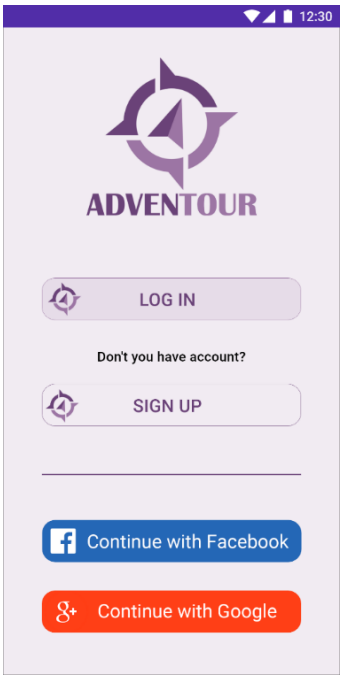


Figura 20: Interfaz de origen del login de Adventour.

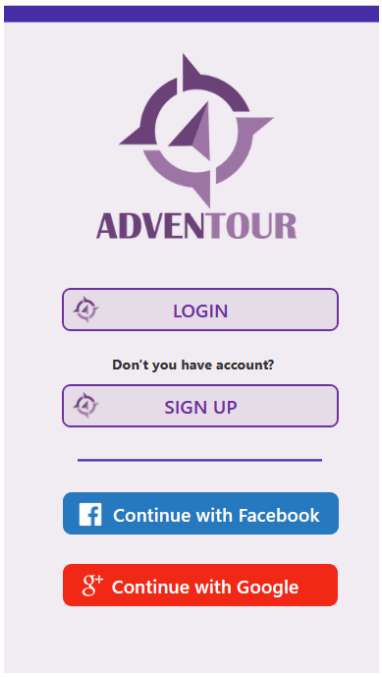


Figura 21: Réplica de la interfaz del login de Adventour.

En este caso, para la elaboración de la estética, se ha procedido de forma diferente; no bastaba con la configuración predeterminada que proporcionan, sino que requería de una solución más compleja.

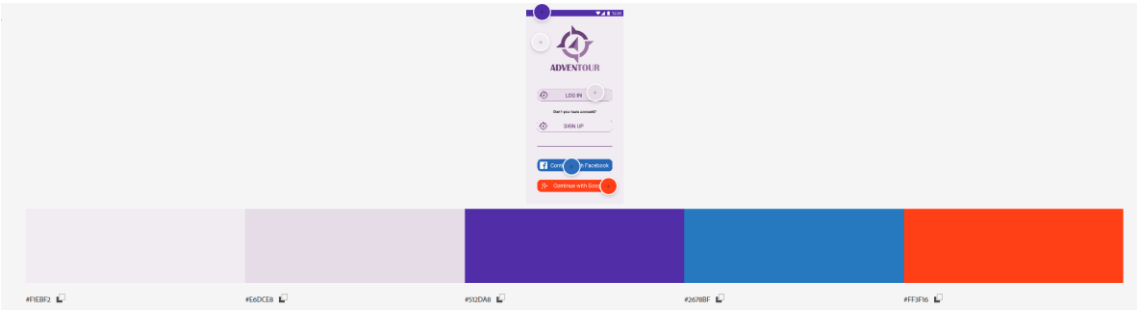


Figura 22: Extracción de los colores de la interfaz de Adventour con Adobe Color.

Como se muestra en la figura 22, haciendo uso de la web de Adobe Color[19] (aplicación que permite obtener los colores exactos y sus especificaciones en base a cierta muestra) se incluyó una muestra de la aplicación Adventour para obtener los colores principales. De esta forma, como vemos en la imagen adjunta, se puede obtener los colores de la barra superior, del fondo y de los botones.

[19] Adobe Color – Página oficial: https://color.adobe.com/create/color-wheel

Posteriormente, se añadieron esos colores en el método OnStart de la aplicación (figura 23); el cual, como su nombre indica, se ejecuta al iniciar la aplicación. Se hizo uso de una variable global que almacenaba dicho tema. Esta expresión guarda en la variable global "*Colors*" los diferentes valores RGBA de los colores bajo los nombres matizados.

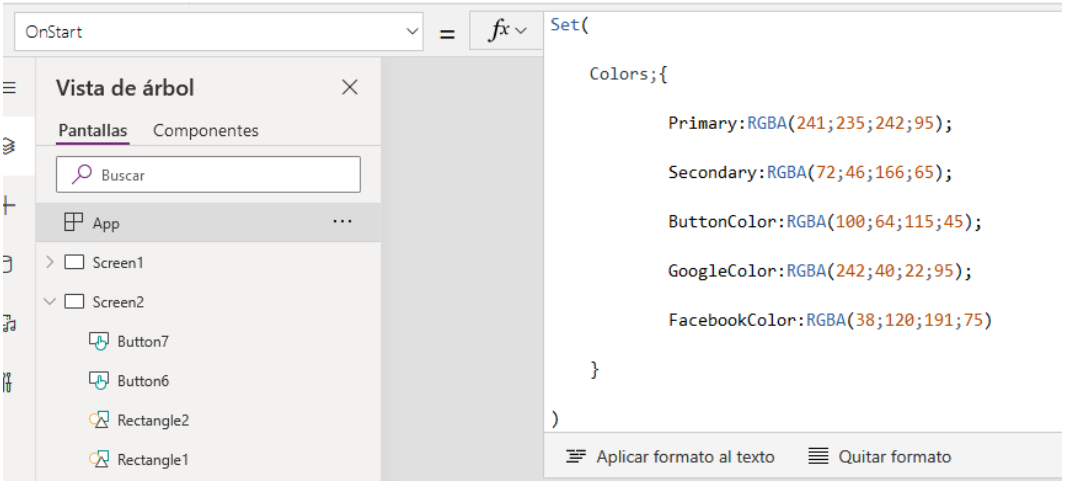


Figura 23: Creación de una variable global con los valores RGBA de los colores del login.

Quedaría referenciar en la propiedad Fill de cada objeto el color al que queremos acceder.

Ejemplo: *"Colors.ButtonColor"* para acceder al color asociado a los botones.

Por otro lado, en Adventour se siguió una dinámica similar. Se creó una variable global denominada "*Theme*", la cual almacena el conjunto de colores que componen la aplicación.

La forma de referenciarlo es también parecida a la de Power Apps, se configuran haciendo referencia al tema.

Ejemplo: *"color: Theme.of(context).primaryColor"* para establecer el color denominado como primario.

En este caso, la personalización resultó prácticamente igual (en términos de complejidad) para ambas opciones mostradas. En las dos se plantea realizar un tema, el cual almacena los diferentes valores de los colores principales para, posteriormente, referenciarlos en las propiedades de los objetos correspondientes.

En segundo lugar, es importante decir que la comparativa en la incorporación de contenido multimedia se encuentra en un estado similar al del anterior apartado. Ambas contienen una carpeta en la que almacenan los activos multimedia de la aplicación, y se añaden mediante referenciación.

Como ya se ha mencionado con anterioridad, en Adventour se hizo uso de Flutter. Flutter utiliza un sistema de jerarquía basado en el anidado de widgets; siendo un widget cada uno de los componentes de dicha interfaz (contenedores, botones, filas, columnas, texto...).

Esto se traduce en que hay un alto grado de personalización, pero con la problemática de que se generan una gran cantidad de líneas de código, con su consecuente gasto alto de tiempo.

Para hacernos una idea del gasto de líneas de código que resulta: en el Login de usuario se utilizaron 120 líneas de código para el diseño de la interfaz. Lo cual conllevó cerca de 2 horas para que quedara a gusto del grupo. Mientras que, por otro lado, el diseño utilizando Power Apps ha supuesto apenas 45 minutos.

Por último, en cuanto a la navegación, faltaría analizar cómo de complicado se plantea el realizarlo en Flutter. Como se ve en la figura 24, para transitar entre interfaces habría que incluir en la función *onPressed* el método *Navigator* que, proporcionándole dos parámetros, realizaría dicha navegación. La complejidad de la navegación en Flutter es bastante similar a la que se plantea en Twins.

```
onPressed: () {
  Navigator.push(
    context,
    MaterialPageRoute(builder: (context) => SecondRoute()),
  );
}
```

Figura 24: Uso de la función onPressed para navegar entre interfaces en Flutter.

Como conclusión podemos destacar la facilidad con la que se diseñan las interfaces en Power Apps. Al realizar un desarrollo tradicional/manual nos encontramos en situaciones de producción de código muy repetitivo, donde se siguen procedimientos muy marcados, los cuales podrían ser automatizados.

5.4 Lista de componentes a partir de una fuente de datos

Uno de los puntos más destacados del uso de Power Apps es la buena relación que tiene con las fuentes de datos, ya que se trata de una herramienta que puede generar aplicaciones predeterminadas a partir de ellos.

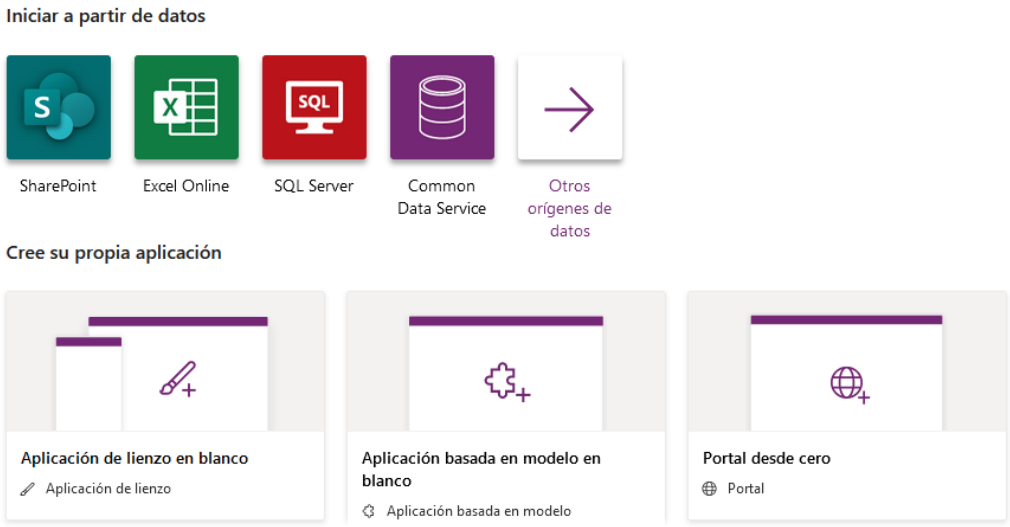


Figura 25: Opciones que proporciona Power Apps para comenzar a crear una aplicación.

En la figura 25 se muestran las diferentes opciones que proporciona la herramienta para comenzar a crear una aplicación. En la primera, crear una aplicación de lienzo en blanco, se comienza a crear la aplicación desde cero, sin ningún tipo de plantilla ni guía marcada. Al igual que pasa en la creación de un portal desde cero; se plantea el crear un portal web sin seguir ninguna estructura determinada. Por otro lado, podemos crear una aplicación basada en modelo en blanco, que se trata de un enfoque del desarrollo de aplicaciones centrado en los componentes.

Sin embargo, vemos que hay otro apartado que permite crear una aplicación base **a partir de una fuente de datos**. Con esta opción se genera una lista de componentes en la que, seleccionando uno de ellos, se puede acceder a ver los detalles de cada uno.

Esto puede resultar bastante interesante y nos puede hacer valorar seriamente la posibilidad de generar una aplicación de este modo y tomarlo como la base inicial de nuestro proyecto. Así pues, generaríamos la aplicación, e iríamos personalizándola; incluyendo interfaces, modificando la estructura dada, añadiendo funcionalidad u objetos, etc.

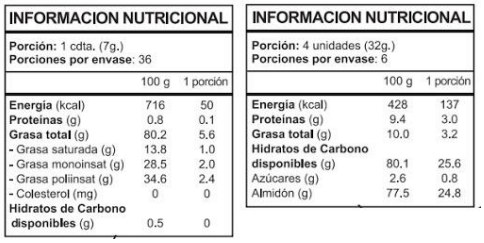
Por lo que tendríamos dos vías de consecución del objetivo. La primera sería realizar la generación automática a partir de datos, y la segunda sería la realización manual. Cada una tiene sus ventajas e inconvenientes, las cuales vamos a analizar a continuación. Además de compararlo, finalmente, con lo que costaría implementarlo sin la ayuda de la tecnología Low-Code.

Cabe añadir que Power Apps admite el uso de una gran cantidad fuentes de datos, como podemos ver en la imagen adjunta. Una de las más llamativas (y la que vamos a usar en este apartado) es la generación mediante Excel. No obstante, en otros

apartados del caso de estudio, veremos y comprobaremos cómo es la inclusión de fuentes de datos más complejas.

Para acercarnos en mayor medida a un uso cotidiano de las tecnologías, en este apartado se ha desarrollado una solución muy recurrente en las grandes empresas dedicadas al sector de la alimentación.

Se ha planteado de manera que, al final de la construcción de la aplicación, vamos a obtener una lista de alimentos. Cada uno de los alimentos tendrá un apartado de detalles con unas especificaciones más detalladas, como podría ser el valor energético, las proteínas, los hidratos de carbono, las grasas, etc.



INFORMACION NUTRICIONAL

Porción: 1 cdta. (7g.)
Porciones por envase: 36

	100 g	1 porción
Energía (kcal)	716	50
Proteínas (g)	0.8	0.1
Grasa total (g)	80.2	5.6
- Grasa saturada (g)	13.8	1.0
- Grasa monoinsat (g)	28.5	2.0
- Grasa poliinsat (g)	34.6	2.4
- Colesterol (mg)	0	0
Hidratos de Carbono disponibles (g)	0.5	0

INFORMACION NUTRICIONAL

Porción: 4 unidades (32g.)
Porciones por envase: 6

	100 g	1 porción
Energía (kcal)	428	137
Proteínas (g)	9.4	3.0
Grasa total (g)	10.0	3.2
Hidratos de Carbono disponibles (g)	80.1	25.6
Azúcares (g)	2.6	0.8
Almidón (g)	77.5	24.8

Figura 26: Ejemplo del tipo de etiquetas del que se van a extraer los datos.

Los principales elementos que van a incluirse en los detalles de los componentes son los que aparecen en las etiquetas de información nutricional (figura 26). Las cuales se encuentran en la inmensa mayoría de los productos.

Además, es importante añadir, que los datos que conforman la solución tienen como principal fuente una cadena de supermercados muy conocida en España, siguiendo con el objetivo de realizar un modelo lo más cercano a la realidad posible.

- **Uso de Excel como fuente de datos**

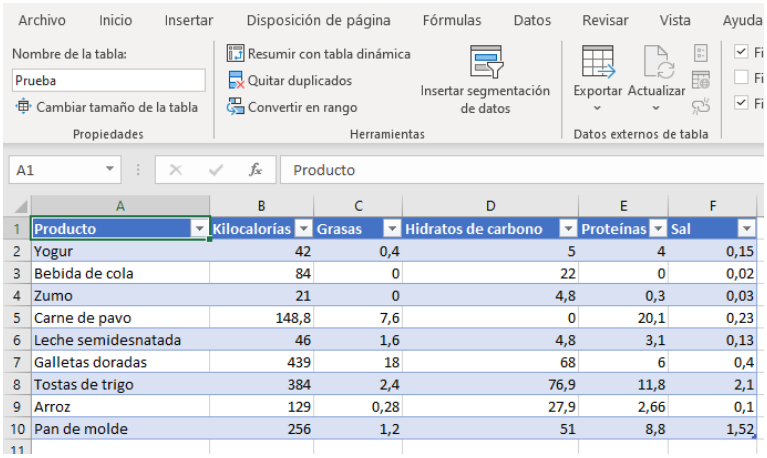
Para utilizar Excel como fuente de datos es necesaria, en primer lugar, la creación de un documento de extensión xlsx. Posteriormente, dicho documento debe seguir cierta estructura y cumplir ciertos requisitos.

El primer requisito que nos encontramos es la inclusión de una tabla en la hoja de Excel. Esta será la que se encargue de almacenar tanto los productos como sus características. Esta tabla se actualizará (añadirá o eliminará registros) de manera dinámica, en respuesta a las operaciones CRUD que se realicen en la aplicación.

Como segundo requisito, es necesario que dicho documento se encuentre en línea, es decir, almacenado en algún espacio en la nube; en este caso utilizamos OneDrive[20]. Es importante añadir que, dicho OneDrive, debe estar asociado a una cuenta Microsoft de empresa. De forma que, una vez se sube el archivo a OneDrive, simplemente debemos hacer una conexión con nuestra cuenta y seleccionar el archivo que hemos almacenado en ella [28].

[20] OneDrive – Página oficial: https://www.microsoft.com/es-es/microsoft-365/onedrive/online-cloud-storage

Para mostrar la apariencia que toma la base de datos en este tipo de entornos, en la figura 27 se puede ver la base de datos planteada en un inicio y la que es la fuente a partir de la cual se crea la aplicación.



	A	B	C	D	E	F
1	Producto	Kilocalorías	Grasas	Hidratos de carbono	Proteínas	Sal
2	Yogur	42	0,4	5	4	0,15
3	Bebida de cola	84	0	22	0	0,02
4	Zumo	21	0	4,8	0,3	0,03
5	Carne de pavo	148,8	7,6	0	20,1	0,23
6	Leche semidesnatada	46	1,6	4,8	3,1	0,13
7	Galletas doradas	439	18	68	6	0,4
8	Tostas de trigo	384	2,4	76,9	11,8	2,1
9	Arroz	129	0,28	27,9	2,66	0,1
10	Pan de molde	256	1,2	51	8,8	1,52

Figura 27: Fuente de datos en Excel.

Se pueden apreciar los diferentes registros que tenemos dentro de la tabla. Además, nos cercioramos de otros aspectos como podría ser el nombre de la tabla, que será útil en apartados posteriores.

- **Generación automática**

Para poder generarlo habría que seleccionar la opción de iniciar a partir de datos con Excel Online. Una vez seleccionada, da a elegir entre las diferentes conexiones que existen. En este caso, mostraría la ya realizada con OneDrive y los diferentes archivos que almacenamos en este espacio (figura 28).

Figura 28: Conexión con OneDrive con la que elegir la tabla que funciona como fuente de datos.

Por lo que únicamente tenemos que seleccionar el Excel que nos interesa tener como fuente de datos; posteriormente nos aparecerían las diferentes tablas contenidas en dicho archivo. Para este caso en particular, y como vemos en la figura 28, solo tenemos una, denominada “Prueba”. Una vez seleccionada, comenzaría la construcción de la aplicación, la cual se demora menos de 30 segundos.

El resultado de dicha generación es el que se puede ver en la figura 29. Se trata de una aplicación muy básica, pero que tiene como principales ventajas la rapidez de creación y la simple personalización que se puede ejercer.

Figura 29: Resultado de la generación automática.

En la primera pantalla vemos que se ha creado una lista con los diferentes elementos de la tabla. Nos damos cuenta de que, debajo de cada uno de los elementos, se han incluido dos números.

Estos corresponden a los valores de los hidratos de carbono y las kilocalorías, ambos son columnas de la tabla. De no convencernos esa disposición, se pueden eliminar esos campos o cambiarlos por otros elementos de la columna editando su valor en el campo Text.

Figura 30: Contenido de uno de los campos generados automáticamente.

De querer modificar el valor, deberíamos hacer referencia a otra columna de la tabla de la misma manera que se realiza con las kilocalorías en la figura 30. Además, también se puede modificar la forma en la que se muestran cada uno de los elementos de la lista: tanto los objetos que contienen como su disposición. Bastaría con modificar el primero de los elementos de la lista y el resto adoptarían la misma configuración automáticamente.

En cuanto al apartado de detalles, se podría acceder seleccionando uno de los productos de la lista generada. En primera instancia, se generaría una interfaz como la que se muestra en la figura 31. En ella, y al igual que anteriormente, se podrían personalizar cada uno de los elementos.

En este caso, una de las funciones que resultó especialmente útil fue la posibilidad de reorganizar la sucesión de elementos. Ya que, queremos que se muestren el conjunto de las columnas de la tabla como elementos de los detalles, pero el orden de apariencia es muy relevante. Ajustarse a "lo que el usuario espera ver" es una de las prácticas más seguidas en el diseño de interfaces; adoptar un orden lógico de los elementos es esencial.

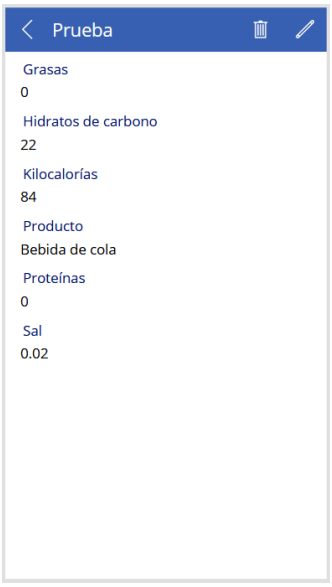


Figura 31: Apartado de los detalles de uno de los elementos de la lista.

Con un trabajo de 20 minutos posterior a la generación automática de la aplicación, conseguimos personalizarla. Como resultado se obtuvieron las dos interfaces que se exponen en las figuras 32 y 33.

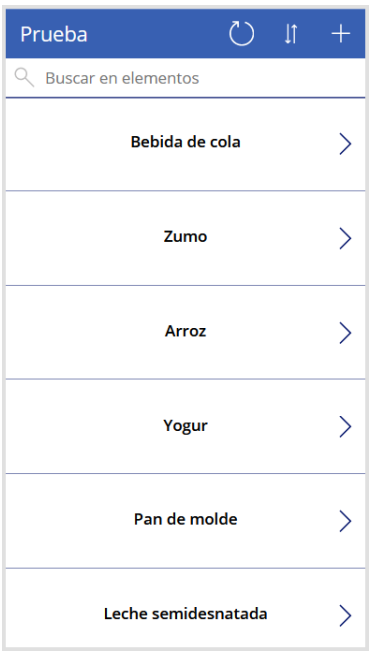


Figura 32: Lista de productos personalizada.

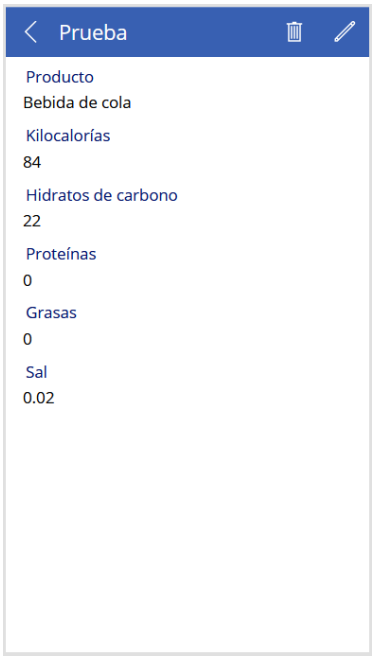


Figura 33: Disposición de los detalles de cada uno de los productos.

Un pequeño matiz, antes de concluir la realización de dicho apartado, es que esta generación instantánea no solo nos proporciona una lista con sus componentes, sino que nos proporcionan más funcionalidades interesantes.

Podemos cerciorarnos que no se genera una simple lista con sus respectivos detalles; permite la actualización (editar, eliminar y añadir) de los valores de dicha tabla. Es decir, permite hacer cambios que más tarde se verían reflejados en la base de datos.

El segundo punto que más llama la atención es la incorporación de un motor de búsqueda. Esto nos resulta realmente útil para encontrar elementos en almacenes de datos grandes, además de que la realización de dicha funcionalidad en otros paradigmas puede resultar bastante compleja.

- **Generación manual**

Evidentemente, cuando se habla de generar la misma funcionalidad de manera manual, se da por hecho de que es una opción mucho más ineficiente en términos temporales. La generación manual conlleva un aumento de carga en la construcción de interfaces y en la inclusión de la funcionalidad correspondiente.

Un ejemplo de uno de los procedimientos que se tienen que llevar a cabo es la implementación de una lista con los diferentes productos.

Para ello, incluiríamos una galería en la que deberíamos especificar la entrada de datos correspondiente (figura 34). En este caso la tabla almacenada en el archivo Excel de OneDrive. Posteriormente, únicamente quedaría personalizar dicha salida de datos.

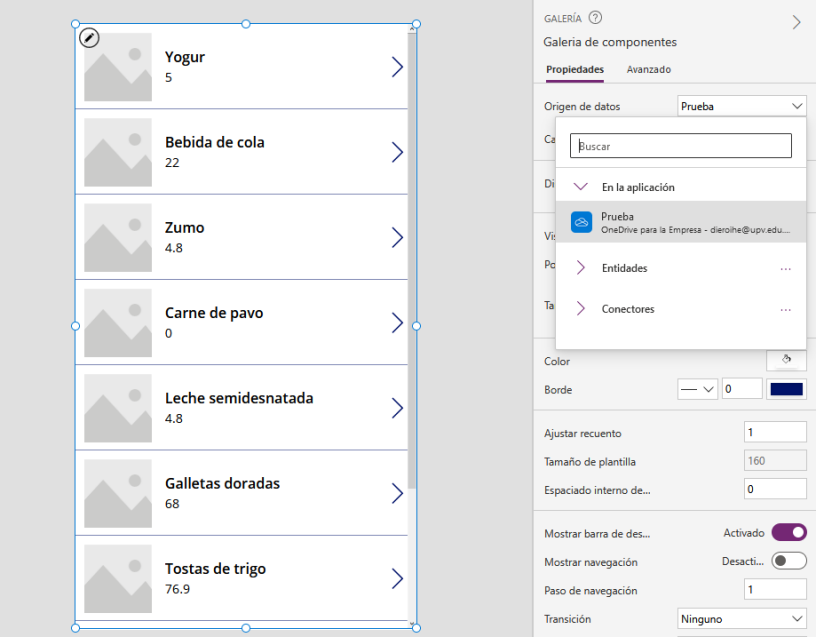


Figura 34: Asociación de fuente de datos a la galería.

Para la realización de la interfaz de los detalles se siguió un proceso similar, por lo que resultaría redundante volver a explicarlo. Básicamente se ha incluido un formulario con diferentes campos y se ha enlazado con la base de datos correspondiente. Haciendo así que simplemente quedara el enlazar cada uno de los campos al resultado de la columna de la tabla indicada.

Cabía esperar que, de poder automatizar un proceso determinado, este sería de carácter simple y repetitivo. Sin embargo, también es simple cerciorarse de que conllevan un gasto de tiempo superior, en comparación con la opción anterior.

El sumatorio de los tiempos resulta bastante más elevado que en la generación automática, esta llevó un total de 1 hora y 15 minutos. No es un tiempo especialmente elevado, pero la generación automática conllevó apenas un instante y una posterior personalización de 20 minutos para obtener exactamente el mismo resultado.

Como se puede apreciar, aquello que hemos desarrollado no tiene relación directa ni con Adventour ni con Twins. Esto se debe a que las listas que se realizan en estas aplicaciones tienen una mayor complejidad, ya que incorporan elementos externos que complicarían el realizar una réplica de la interfaz; sin aportar al desafío de realizar una lista. De esta manera, se optó por realizar una lista simple para ver de manera más sencilla cómo de complicado se plantea su elaboración.

- **Elaboración de una lista en Adventour**

Para comparar las soluciones realizadas anteriormente con las realizadas con las tecnologías tradicionales, centramos la comparación con Adventour. Esto se debe a que, a lo largo de su desarrollo, se han tenido que plantear diferentes listas muy similares a las que nos hemos propuesto impulsar en el actual desafío.

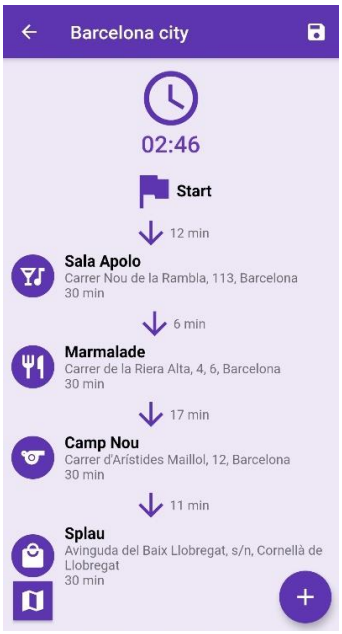


Figura 35: Lista de los lugares de interés de la ruta "Barcelona city".

Un ejemplo de aparición de una lista con interfaz de detalles es la que podemos ver en la figura 35. Se trata de una lista que concatena los diferentes lugares de interés de una ruta almacenada bajo el nombre “Barcelona city”.

Cada uno de los elementos de dicha lista muestra un lugar de Barcelona, como es evidente. Además, mediante la pulsación en un paraje podríamos acceder a otra pantalla, la cual almacena diferentes aspectos relevantes de dicho paraje de la ciudad.

Como ya se mencionó en el apartado de diseño de interfaces, el uso de la combinación de Flutter y Dart es más complejo que en Power Apps; de manera que la construcción de las interfaces necesarias conlleva un mayor gasto de recursos temporales.

Por otra parte, se programó también una pequeña reunión con el grupo de desarrollo de la aplicación Adventour; para compartir ideas con el fin de matizar la estimación adecuada en base a nuestras capacidades. No bastaría con obtener el tiempo registrado en Worki, ya que este tiempo está sujeto a una funcionalidad que no solo aúna la realización de la lista y la interfaz de detalles; la elaboración de dicha lista estaba subordinada a la realización de otros aspectos. Las listas planteadas en la aplicación eran componentes más completos, con más funcionalidades que no tienen relación con el desafío actual.

La realización de dicho desafío con las tecnologías mencionadas se movería entre las 5 y 6 horas de trabajo. Si lo comparamos con la inmediatez de la fórmula de generación automática y el buen tiempo registrado por la forma manual, se puede concluir que la opción propuesta por la tecnología Low-Code sería óptima en términos temporales.

5.5 Conexión a base de datos y la persistencia

Uno de los puntos más importantes en la realización de un proyecto software es el tratamiento que se le proporciona a los datos; dedicando bastantes recursos en idear cómo persistirlos y gestionarlos. Por ello, en este apartado se analizará la relación que tiene dicha herramienta con la capa de persistencia.

La capa de persistencia es aquella que se encarga de preservar los datos de la aplicación desarrollada, para la realización de operaciones sobre estos (leer, escribir y borrar) [29].

Los datos, en este caso, son representaciones o información concreta, de un elemento o hecho determinado, que son registrados en la aplicación. Los cuales nos permitirían describir ciertos sucesos y entidades. Con estos, tras un procesamiento sofisticado y significativo, podríamos obtener información; la cual tendría la capacidad de ser entendida e interpretada por un receptor determinado [30].

Se trata de un recurso esencial, por lo que el tratamiento de los datos es una parte fundamental del proceso de desarrollo de la aplicación; de forma que, de no evaluar este aspecto en Power Apps, estaríamos incurriendo en un error.

Para gestionar este aspecto, en las aplicaciones desarrolladas con las tecnologías tradicionales, en el grupo de desarrollo se planteó el uso de Firebase[21].

Firebase es una tecnología, ubicada en la nube, que se utiliza principalmente para la creación o sincronización de diferentes proyectos. Se obtienen una gran cantidad de ventajas al utilizarla, una de ellas podría ser la capacidad de sincronizar los datos sin tener que desarrollar una lógica compleja. Además de ahorrarte aspectos como el administrar las conexiones.

Uno de los productos de esta plataforma es **Firebase Realtime Database**, el cual utilizamos. Este nos permitió sincronizar y almacenar nuestros datos en una base de datos alojada en la nube.

El potencial de la herramienta reside en que estos datos se actualizan en tiempo real con todos los clientes que lo utilizan. Haciendo, incluso, que dichos datos se encuentren disponibles cuando la aplicación se encuentra sin conexión [31]. La solidez como herramienta se ve reflejada en el uso de esta misma por parte de grandes compañías; como podrían ser Youtube, Atlassian, Twitch o Alibaba.

La principal desventaja es la necesidad de pago, no obstante, en nuestro caso no afectaba. Esto se debe a que, para los proyectos de un tamaño relativamente pequeño, con la versión gratuita se puede desempeñar las funcionalidades de forma adecuada. De hecho, en ambos proyectos, no hizo falta realizar ningún desembolso. El estado de la capa de persistencia en Twins y Adventour es correcto, no generó ningún problema.

[21] Firebase – Página oficial: https://firebase.google.com/?hl=es

Como presentamos en apartados anteriores, Power Apps proporciona diferentes formas de comenzar una aplicación a partir de los datos. Esta constituye una forma de conectar nuestra aplicación a una base de datos. En el caso anterior se realizó eligiendo como almacén de datos un Excel, pero se podría realizar también mediante SharePoint[22], SQL Server[23], Microsoft Dataverse[24], entre otros.

Se siguen las mismas dinámicas expuestas en el apartado previo, sin embargo, hay otra forma de enlazar una aplicación a una base de datos. En el caso de que se haya optado, por ejemplo, por realizar una aplicación desde cero (Aplicación de lienzo en blanco), el procedimiento para conectarse a una base de datos sería diferente.

En primer lugar, se tendría que agregar una conexión en la pantalla de inicio de Power Apps (figura 36). En el caso de, por ejemplo, querer enlazarlo con un Excel de manera manual; únicamente tendrías que subir el archivo a la nube, por ejemplo, a OneDrive. Y, a continuación, realizar una conexión con dicho OneDrive.

Hay una gran cantidad de posibles conexiones que se pueden realizar. Aunque, es importante añadir, que algunas de ellas solo están disponibles para la versión Premium de la herramienta.

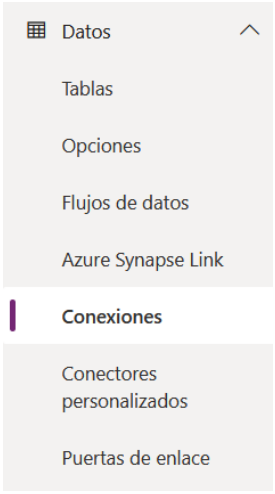


Figura 36: Apartado de conexiones en la pantalla principal.

Una vez realizada, quedaría acceder a los datos que nos puede proporcionar dicha conexión. Esto se realiza una vez dentro de la aplicación que se está desarrollando.

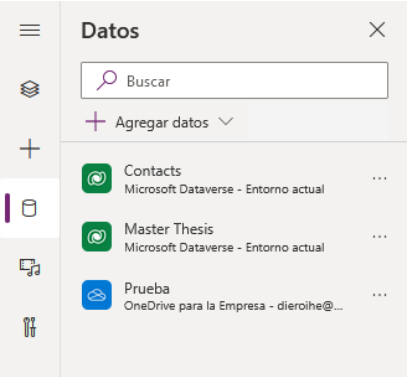


Figura 37: Uso del conector OneDrive para acceder a la tabla Prueba.

Para ello, en el apartado de datos, hay que seleccionar la función "Agregar datos". Una vez seleccionada, se elegiría el conector del que se quiere extraer datos y se accederá a él.

Llegados a este punto, queda elegir qué es aquello a lo que queremos acceder dentro del conector. En el caso del Excel, seleccionaríamos el archivo en cuestión y la tabla a la que queremos acceder. Como se puede ver en la figura 37, habríamos hecho uso del conector OneDrive y seleccionado la tabla denominada "Prueba".

A partir de ese momento, ya se podría referenciar los datos en cuestión, al incluirse en nuestros datos la tabla contenedora de estos.

[22] SharePoint – Página oficial: https://www.microsoft.com/es-es/microsoft-365/sharepoint/collaboration
[23] SQL Server – Página oficial: https://www.microsoft.com/es-es/sql-server
[24] Microsoft Dataverse – Página oficial: https://powerplatform.microsoft.com/es-es/dataverse/

Usar Excel es totalmente funcional, ahora bien, si se quiere generar una aplicación sustancialmente más compleja y con necesidades más específicas, es recomendable utilizar otras tecnologías de almacenamiento de datos más sofisticadas.

En este caso, se apostó por realizar dos pruebas, compatibles entre ellas. Se utilizó Azure SQL DB[25] para guardar los datos de la aplicación de prueba y Azure Blob Storage[26] para guardar las imágenes necesarias. En un principio se planteó utilizar Firebase, como en los proyectos anteriores, sin embargo, no se pudo implementar; parece no ser una de las conexiones que Power Apps permite realizar.

Para utilizar Azure SQL DB es necesario, en primer lugar, crear una base de datos SQL en Azure[27]. Siendo Azure un servicio de computación en la nube que nos proporciona Microsoft. Este facilita el uso de su centro de datos para lanzar aplicaciones y servicios; cumpliendo todas las condiciones de resguardo y mantenimiento [32].

De esta manera, tras realizar un registro en la página de Azure, crearíamos la base de datos SQL que necesitamos para persistir nuestros datos; siguiendo el procedimiento que se muestra en el formulario de creación. Asumiendo también el coste de usar dichos servidores, simbólico en comparación con el coste que supone desplegarlo por tu cuenta. Cuesta 3 euros mensuales utilizando 2 GB con 5 DTU (siendo las DTU una medida de rendimiento de la base de datos; en este caso conllevaría, entre otras cosas, unas 5 transacciones por segundo y 300 sesiones como máximo).

Posterior a la creación, y siguiendo la dinámica que también seguimos con el Excel, crearíamos una tabla, con la cual se interactuará en un futuro desde Power Apps. Para la creación de dicha tabla se utiliza una operación CREATE TABLE.

Una vez se tenga todo correctamente configurado, habría que seguir pasos similares a los ya seguidos con anterioridad (crear conexión y agregar datos). El único aspecto distintivo a la hora de crear la conexión es que esta vez se seleccionaría la opción “SQL Server”. De manera que sería necesario introducir ciertos datos representativos de nuestro servidor, como podrían ser el nombre del servidor, el nombre de usuario, la contraseña y el nombre de la base de datos SQL.

Para almacenar diferentes datos en la base de datos, desde Power Apps se tiene que hacer uso de un formulario de edición. Dicho formulario, estaría compuesto fundamentalmente por tarjetas. A la hora de crear elementos en la base de datos, el formulario representaría el registro nuevo que estamos creando y cada tarjeta un único campo de dicho registro.

[25] Azure SQL DB – Página oficial: https://azure.microsoft.com/es-es/products/azure-sql/database/
[26] Azure Blob Storage – Página oficial: https://azure.microsoft.com/es-es/services/storage/blobs/
[27] Azure – Página oficial: https://azure.microsoft.com/es-es/

	A	B	C	D	E	F	G	H
1	Producto	Tipo	Kilocalorías	Grasas	Hidratos de car	Proteínas	Sal	
2	Yogur	Comida	42	0,4	5	4	0,15	
3	Bebida de cola	Bebida	84	0	22	0	0,02	
4	Zumo	Bebida	21	0	4,8	0,3	0,03	
5	Carne de pavo	Comida	148,8	7,6	0	20,1	0,23	
6	Leche semidesnatada	Bebida	46	1,6	4,8	3,1	0,13	
7	Galletas doradas	Comida	439	18	68	6	0,4	
8	Tostas de trigo	Comida	384	2,4	76,9	11,8	2,1	
9	Arroz	Comida	129	0,28	27,9	2,66	0,1	
10	Pan de molde	Comida	256	1,2	51	8,8	1,52	
11								
12								

Figura 38: Muestra práctica de la diferencia entre a lo que hace referencia el formulario y cada una de las tarjetas.

Haciendo referencia a la figura 38, en el caso de crear el registro correspondiente al "Pan de molde" en nuestro Excel, el formulario de edición haría referencia a la fila en su totalidad (rectángulo rojo) y cada tarjeta a cada una de las columnas que las compone (por ejemplo, el rectángulo verde; haciendo referencia a la tarjeta de los Hidratos de carbono)

Es importante matizar que, para completar el proceso de almacenamiento, queda notificar en la propiedad *Update* de cada tarjeta lo que queremos guardar. De manera que, por ejemplo, podemos matizar en esa propiedad que guarde la propiedad Text de un elemento; guardando así dicho texto en el campo de la base de datos correspondiente a la tarjeta seleccionada.

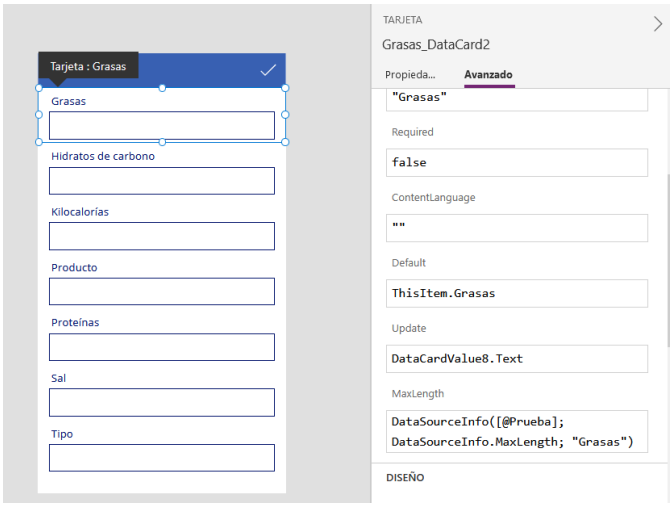


Figura 39: Propiedades de una tarjeta del formulario.

Si nos centramos en la figura 39, podemos ver las diferentes propiedades de la tarjeta que hace referencia a la columna Grasas.

Vemos como en su propiedad *Update* hace referencia al texto que hay en el campo situado en el interior de la tarjeta (DataCardValue8). Esto haría que se almacene el texto matizado en la columna Grasas.

Como es evidente, esto debería estar especificado en cada una de las tarjetas del formulario de edición. Es una manera de comunicar qué se desea guardar en cada uno de los campos del registro.

En el caso de Azure Blob Storage, habría que llevar a cabo una serie de pasos similares a los ya seguidos previamente. Adicionalmente, en la cuenta de Azure anteriormente creada, tendrías que generar una cuenta de almacenamiento. Para cumplimentar dicha creación se demandan datos similares a los que se pedían para la creación anterior del servidor Azure (nombre, ubicación de la instancia, tipo de cuenta, rendimiento...).

De esta manera, ya estaríamos gestionando desde la misma cuenta de Azure el servidor SQL, la base de datos SQL y la cuenta de almacenamiento (que corresponde al espacio reservado para el almacenamiento de las imágenes). Resulta relevante añadir que, en el

interior de dicha cuenta de almacenamiento, se creó una carpeta contenedora. Esa carpeta será la encargada de persistir y almacenar las diferentes imágenes que se vayan enviando.

¿Cómo se accedería esta vez a los datos? Como era de esperar, se siguen unas dinámicas que ya nos resultan familiares. Se tendría que agregar como origen de datos un conector de Azure Blob Storage dentro de la aplicación. No sería necesario volver a incluir la conexión debido a que utilizamos la misma cuenta de Azure con la que hemos hecho uso el servidor, de manera que ya estaría enlazado.

Y, ¿cómo podríamos almacenar dichas imágenes desde Power Apps? Mediante el uso de la propiedad *OnSuccess* del formulario (esta se ejecutará si, a la hora de enviar el formulario de edición, no sucede ningún problema). En dicha propiedad deberíamos incluir la función *CreateFile*:

AzureBlobStorage.CreateFile*(folderPath; name; file)*

El atributo *folderPath* haría referencia a la carpeta en la que quieres almacenar el archivo (el nombre de la carpeta contenedora que hemos creado dentro de la cuenta de almacenamiento). Por otro lado, el atributo *name* es el nombre que se le quiere dar al archivo a persistir. Y, por último, el atributo *file* hace referencia al archivo que quieres almacenar en dicha carpeta y con ese nombre.

De esta forma tan sencilla, y con tan solo una línea de código, podríamos indicar la ubicación y el nombre del archivo que queremos almacenar en Azure. En el caso de que quisiéramos acceder al contenido de las imágenes de la cuenta de almacenamiento desde Power Apps, habría que usar la función GetFile:

AzureBlobStorage.GetFileContent(id)

Dicha expresión, mediante el identificador con el que se almacena la foto, obtendría el archivo en cuestión.

Antes de concluir con las ventajas y desventajas, se planteará una pregunta que surgió a lo largo de la realización del desafío. Power Apps permite poblar la base de datos desde la misma aplicación, pero ¿sería posible crear una base de datos desde cero, solo con el uso de la herramienta?

En respuesta a la pregunta, hasta donde he podido saber, no se podría crear una base de datos desde la misma herramienta. Se podría crear una base de datos vacía usando una tecnología externa y, mediante el uso de las funciones ***AddColumn()*** *y* ***RemoveColumn()***, formar la tabla que deseemos. Con esta tabla, ya formada, podríamos seguir la dinámica de añadir registros que hemos expuesto a lo largo del apartado.

Lo que sí se podría realizar es crear una base de datos con tecnología que nos ofrece Power Apps. No podría crearse una base de datos desde la misma herramienta en la que se desarrolla la aplicación, pero sí con tecnología que también nos proporciona la solución de Microsoft. Registrándote con tus credenciales en el *PowerApps Admin center*, seleccionando la opción de "*New Environment*" y siguiendo unos sencillos pasos, podrías crear una base de datos.

Ahora sí, en cuanto a las ventajas y desventajas del uso de Power Apps, hemos encontrado que nos proporciona una solución bastante competitiva. Hay que tener en cuenta, y recordar, que es una herramienta que centra su eje principal en los datos y la relación con ellos. Pudiendo crear aplicaciones funcionales a partir de ellos con escasos minutos de trabajo.

Por lo que este punto positivo nos generaría una gran cantidad de facilidades y ahorro de tiempo a la hora de tratar el hito de la persistencia en nuestra aplicación. Por otro lado, generaría menor confianza en su uso que las opciones “más tradicionales”.

¿Por qué? Es habitual que, en la mayoría de los campos, la opción más técnica sea aquella que proporciona una mayor sensación de control en su uso. De esta manera, el uso de Firebase, aunque más largo y complejo resultó en sus inicios, proporcionaba sensación de total control sobre los datos y transparencia en las operaciones que se realizaba. En el caso de esta tecnología Low-Code, la facilidad y abstracción de conceptos complejos va en detrimento de la transparencia en sus acciones. Haciendo que la duda de saber qué está pasando “por detrás” de las operaciones simples sea recurrente.

Además, es importante recordar que el número de conexiones a fuentes de datos que se pueden realizar desde Power Apps, aunque extenso, es más limitado que con las tecnologías tradicionales. Esto se puede ver patente en que no se haya podido realizar una conexión con Firebase, teniendo así que buscar otra opción compatible entre las que se nos ofrecen.

Para concluir, se ha de matizar que el hecho de que las pruebas en la persistencia hayan supuesto un coste no se toma como ventaja ni desventaja. El coste viene derivado del uso de un servidor Azure; de querer usarlo con tecnologías tradicionales también sería necesario sufragarlo. Además, el coste es simbólico en comparación a la gran funcionalidad y apoyo que proporciona.

5.6 Motor de búsqueda y filtros

Un aspecto que supuso mucha problemática a la hora de desarrollar la aplicación de Adventour fue la implantación de un motor de búsqueda. Dicho motor se encargaba de buscar diferentes lugares por su nombre, dentro de los diferentes lugares de interés que nos proporcionaba la API de Google Maps.

De manera que, con el fin de evaluar dichas diferencias, se ha planteado el desarrollo de un motor de búsqueda. El objetivo es que nos permita encontrar diferentes elementos de una base de datos, en base a ciertos parámetros o palabras clave que introduzcamos.

Además, como complemento a este motor de búsqueda, se añadirá la opción de filtrar dichas búsquedas. Este fue un aspecto que, en su día, se descartó del último sprint de Adventour. Pese a esto, nos hubiera gustado incluirlo; no fue posible debido a que nos cercioramos de que iba a requerir de una cantidad de tiempo de la que no disponíamos.

- **Motor de búsqueda**

Para la realización del motor de búsqueda, es esencial tener claro el uso de la función ***Search***. Dicha función nos permite filtrar los elementos de una galería en base a un parámetro proporcionado. Accediendo a la documentación que nos proporciona Microsoft, nos encontramos con la expresión de la figura 40.

Search(*Table, SearchString, Column1* [, *Column2, ...*])

Figura 40: Función Search en la documentación de Microsoft [33].

Se le han de proporcionar tres parámetros. En primer lugar, habría que indicar el lugar donde se va a realizar dicha búsqueda, en otras palabras, la tabla contenedora de los datos.

En segundo lugar, la cadena que se va a buscar. Es decir, se va a filtrar en función a lo que se escriba en el segundo parámetro. Resultaría mucho más interesante si, en vez de buscar una cadena predeterminada, buscáramos en base a lo que escribamos en un campo. De manera que haríamos que la aplicación disponga de una barra de búsqueda. La dinámica que se debería seguir consta de obtener el valor de lo escrito en la barra de búsqueda y proporcionarlo como segundo parámetro de la función *Search*.

Por último, el tercer parámetro hace referencia a la columna o columnas en las que se va a buscar. Esto es un aspecto muy interesante, debido a que no solo tendríamos la posibilidad de buscar por el nombre (lo más común), sino que también se puede hacer una búsqueda que tenga en cuenta diferentes aspectos simultáneamente. Es decir, buscaría la cadena del segundo parámetro en todas las columnas que le indiquemos en el tercero [33].

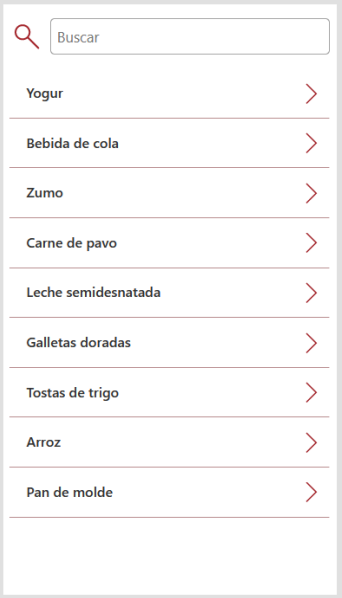


Figura 41: Interfaz elaborada para realizar las pruebas del motor de búqueda.

Para realizar las diferentes pruebas del motor de búsqueda, se ha dotado a la aplicación de una interfaz en la que se incluyeron diferentes elementos (figura 41).

Primeramente, una entrada de texto para introducir las palabras clave. Conjuntamente, es posible cerciorarse de la existencia de un icono de búsqueda al costado de dicha entrada; utilizado para hacer saber al usuario que utiliza la aplicación que se trata de una barra de búsqueda.

También se añadió una galería, la cual se encarga de almacenar los diferentes elementos de la base de datos de prueba. Dicha fuente de datos ya fue presentada en apartados anteriores. Cabe decir que ha sido utilizada, en este caso, para satisfacer el fin de tener elementos que nos resulten familiares para facilitar la explicación.

Para completar una búsqueda se debería introducir la sentencia *Search* en la propiedad ***Items*** de la galería (figura 42). Dicha función tiene como fuente de datos la base de datos Prueba, el cual filtra por el nombre de los productos (los elementos de la columna Producto) en base a lo introducido en la barra de búsqueda (SearchBox_field)

Figura 42: Ejemplo de aplicación de la función Search.

Con el fin de demostrar su funcionamiento, se realizaron diferentes pruebas en ejecución. Una de ellas es la que se muestra subsiguientemente en la figura 43. En esta realizamos una búsqueda, la cual obtiene por resultado todos aquellos componentes de la lista que contengan (en su nombre) la cadena “pa”.

Figura 43: Prueba de funcionamiento de la funcion Search de la figura 41.

Se podrían realizar búsquedas en base a diferentes columnas; teniendo en cuenta el valor de las grasas, proteínas, sal, kilocalorías, etc. Haciendo así que se puedan realizar búsquedas muy útiles con sentencias muy escuetas.

- **Filtrado de búsquedas**

Cuando solo se plantea una búsqueda por palabras clave se quedan muchos detalles sin especificar; esto hace que los resultados no sean tan exactos. Es decir, dicha búsqueda permite mostrar aquellos productos que coincidan con el valor introducido, pero de ninguna manera permitiría escoger solo aquellos en los que la coincidencia se de en varios aspectos. O, entre otras cosas, tampoco podría comprobar si determinado valor se encuentra en un intervalo de valores determinados. Todo lo mencionado anteriormente se podría conseguir añadiendo filtros.

Un buen ejemplo de utilización de los filtros lo encontramos en el siguiente caso de uso: las personas que desean realizar dietas para adelgazar buscan realizar un déficit calórico, de manera que se realice un mayor gasto de calorías en comparación con la ingesta. Por ello, una función interesante podría ser filtrar los productos en base a las kilocalorías. De forma que el usuario solamente visualizara, en los resultados de la búsqueda, aquellos productos que no excedan una cantidad determinada de calorías.

Es importante matizar que, para este ejemplo, se ha optado por añadir a nuestro Excel una columna nueva (figura 44). Esta columna se denomina “Tipo” y puede tomar dos valores diferentes: “Bebida” si se trata de un producto de alimentación en estado líquido o “Comida” si este no se presenta en estado líquido. A priori puede resultar redundante, sin embargo, resulta útil para mostrar el funcionamiento de los filtros.

A20

	A	B	C	D	E	F	G
1	Producto	Tipo	Kilocalorías	Grasas	Hidratos de carbono	Proteínas	Sal
2	Yogur	Comida	42	0,4	5	4	0,15
3	Bebida de cola	Bebida	84	0	22	0	0,02
4	Zumo	Bebida	21	0	4,8	0,3	0,03
5	Carne de pavo	Comida	148,8	7,6	0	20,1	0,23
6	Leche semidesnatada	Bebida	46	1,6	4,8	3,1	0,13
7	Galletas doradas	Comida	439	18	68	6	0,4
8	Tostas de trigo	Comida	384	2,4	76,9	11,8	2,1
9	Arroz	Comida	129	0,28	27,9	2,66	0,1
10	Pan de molde	Comida	256	1,2	51	8,8	1,52
11							

Figura 44: Fuente de datos con la comuna nueva creada.

Con el fin de abordar la implantación de un par de filtros, se planteó una nueva interfaz (figura 45) y se añadió en la anterior presentada un icono que hace de acceso directo a esta (figura 46).

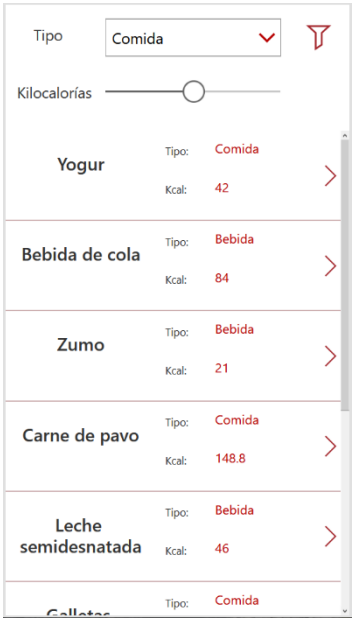


Figura 46: Interfaz para el filtrado de los productos.

Figura 45: Interfaz del motor de búsqueda con el nuevo icono de filtrado.

Como se puede apreciar la nueva interfaz (figura 45) consiste en una lista y dos filtros; una lista desplegable y un control deslizante. Dicha lista tiene unas particularidades, en comparación con las planteadas anteriormente, ya que muestra los valores por lo que vamos a filtrar para comprobar que se realiza todo correctamente.

En cuanto a los filtros, la lista desplegable almacena los valores Comida y Bebida. Dependiendo de cuál se elija, aparecerán solo las comidas o solo las bebidas. En el control deslizante matizamos las calorías máximas (que no se desean superar) para los productos mostrados. De forma que si se establece un valor de 230 (por ejemplo), aparecerán todos los productos que tengan 230 o menos kilocalorías.

Si nos centramos en el funcionamiento de dichos filtros, podemos apoyarnos en la figura 47. Como se ve el campo Tipo toma el valor Comida y, por otro lado, situamos el valor del campo de las kilocalorías en 260. De manera que en la lista aparecen todos aquellos productos que son comida y que tienen 260 kilocalorías o menos, como podemos comprobar en sus especificaciones.

Figura 47: Ejemplo del filtrado de producto.

Sin entrar en muchos detalles, ya que podría resultar repetitivo, podemos explicar la forma de proceder a la hora de filtrar componentes con el fin de demostrar la facilidad de uso. Al igual que realizábamos con la búsqueda, tendríamos que introducir sentencias en la propiedad *Items* de la galería.

Hacemos uso de una sentencia ***If*** en la que incluiremos diferentes funciones ***Filter***.

If(*Condition, ThenResult* [, *DefaultResult*])
If(*Condition1, ThenResult1* [, *Condition2, ThenResult2, ...* [, *DefaultResult*]])

Figura 48: Función If en la documentación de Microsoft [34].

La función *If* (figura 48), se compone esencialmente por 3 parámetros. El primero que podemos observar es la variable *Condition*; esta se compondría de diferentes fórmulas, las cuales se evalúan a true o false, dependiendo si la condición se cumple o no.

En cuanto al segundo parámetro, *ThenResult* es aquello que se devuelve cuando una condición es evaluada como verdadera. Y, por último, *DefaultResult* es un valor opcional que se devuelve si ninguna condición se evalúa a true [34].

Filter(*Table, Formula1* [, *Formula2, ...*])

Figura 49: Función Filter en la documentación de Microsoft [33].

Por otro lado, el *Filter* (figura 49) es una función cuya utilidad es buscar registros que satisfacen ciertas fórmulas, en una tabla determinada. Se compone de dos parámetros; *Table* correspondería a la tabla en la que vamos a ejercer la búsqueda y las diferentes fórmulas que encadenamos a continuación harían referencia a aquello que comprobamos en cada registro. Es decir, de cumplirse el conjunto de las fórmulas en dicho registro, lo mostraría [33].

En la propiedad *Items* de la galería, incluimos una sentencia *If* que recogía el conjunto de los diferentes casos que se podrían dar en la utilización de dichos elementos gráficos; tanto la lista desplegable como el control deslizante. Para realizar una mejor explicación, se apoyará con un ejemplo.

En la figura 50 se puede ver una de las condiciones del *If* junto a su *ThenResult*. Esta condición restringe el valor de ambos elementos gráficos. En primer lugar, que la selección de la lista desplegable no esté en blanco, es decir, que se haya seleccionado una de las opciones. Y, en segundo lugar, que el valor del control deslizante sea mayor a cero.

```
!IsBlank(Dropdown_type.Selected) && Slider_kcal.Value > 0;
Filter(
    Prueba;
    Tipo = Dropdown_type.Selected.Result;
    Value(Kilocalorías) < Slider_kcal.Value
);
```

Figura 50: Ejemplo del uso simultáneo de la función If y Filter.

Si se cumple dicha condición, se filtra en la tabla Prueba y devuelve todos aquellos registros cuyo tipo sea igual al seleccionado en la lista desplegable y cuyo valor de las kilocalorías sea menor al indicado en el control deslizante.

Es simple, aunque se generaron diferentes problemas de incompatibilidad de tipos en una evaluación. La herramienta comentaba que, en la sentencia que comparábamos las kilocalorías con el valor del control deslizante, estábamos comparando un valor numérico con otro que no lo era.

Esto podría venir derivado de la falta de control que a veces tenemos en dicha herramienta; en ningún momento he especificado el tipo de dicha columna (Kilocalorías), se ha establecido predeterminado. No obstante, pudimos solventarlo con el uso de la función *Value*, que convierte una cadena de texto en un valor numérico.

En cuanto a la aplicación de la misma funcionalidad en Adventour, el proceso de obtención fue diferente. En términos más objetivos, para la elaboración de un motor de búsqueda en la aplicación destinada a la elaboración de rutas turísticas, nos costó cerca de 15 horas de desarrollo realizarlo. Valor muy lejano al que obtenemos con Power Apps que, por el mismo trabajo, gastamos cerca de 1 hora de trabajo.

Por último, hablando del filtrado, en esta aplicación se planteó una serie de filtros que acompañaran al motor de búsqueda. Sin embargo, esto no se pudo realizar por falta de tiempo. Lo que hace pensar que, de haber ahorrado tiempo en el desarrollo del motor de búsqueda, se podría haber incluido.

5.7 Multimedia

La inclusión de elementos multimedia es un ámbito esencial en muchas herramientas actuales del mercado. Incluir este tipo de elementos puede llegar a ayudar a la hora de mejorar muchos aspectos como el mantener la atención y el interés del usuario. Incluso para mejorar la retención y el entretenimiento de las personas que utilizan un servicio.

Hay una gran cantidad de vías a la hora de comunicar aquello que queremos hacer llegar a quien nos está prestando atención en ese momento. El multimedia nos permite establecer una comunicación efectiva y entendible entre interlocutores sin generar ningún tipo de interpretaciones fallidas, lo cual es un aspecto increíblemente atractivo [35].

En este caso, se plantea el desafío de **solucionar** algunos de los **problemas**, del apartado multimedia, que aparecieron mientras se desarrollaba Twins. Además de documentar aquellos descubrimientos de funciones multimedia interesantes que ofrece Power Apps.

En esta herramienta, encontramos aspectos negativos, pero también aspectos muy positivos. Uno de los aspectos positivos es la gran variedad de opciones multimedia que existe, como podemos ver en la figura 51.

Podemos disfrutar de facilidades a la hora de incluir aspectos muy utilizados como son las imágenes, los videos, los audios y un largo etcétera. Pero también se puede incluir aspectos multimedia más vanguardistas o inusuales, como podrían ser la inclusión de figuras en 3D o un escáner de código de barras

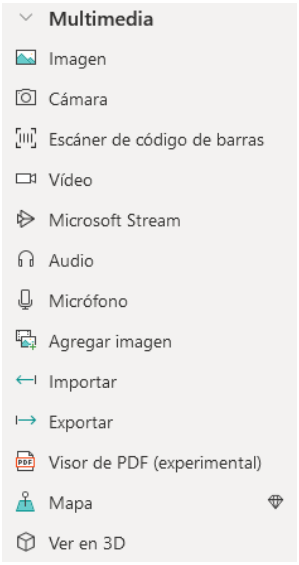


Figura 51: Elementos multimedia disponibles.

Centrándonos en el primer punto a tratar (los problemas que nos surgieron en Twins), tendremos en cuenta cada una de las problemáticas por separado; explicando, entre otras cosas, qué desarrollamos y cuáles fueron las dificultades.

El primer problema encontrado fue la inclusión de audio en la aplicación. Al tratarse de un juego, se requería añadir música de fondo. Se desarrolló dicho aspecto introduciendo un patrón Fachada, el cual nos permitía gestionar todo lo relacionado con la música desde una única clase.

A pesar de ser una buena práctica y de habernos simplificado su uso, nos surgieron algunos problemas relacionados con inconsistencias: a veces sonaba la música cuando no debía o no sonaba cuando debía. Esto fue solventado, no obstante, conllevó un gasto de tiempo adicional. Dicho problema era causado por el poco dominio que se tenía de las librerías utilizadas en la consecución del hito.

Si hablamos en términos cuantitativos, en un principio se estimó que el desarrollo de esto no superaría el par de horas. Sin embargo, se alargó hasta las 5 horas, por los problemas ya explicados.

Como posible solución, propuesta por Power Apps, tenemos el uso del elemento **Audio**. Este nos proporciona un reproductor de audios, que nos podría ayudar a incluir una canción de fondo para nuestra aplicación.

Una vez añadido en pantalla, se debería incluir la canción que deseamos que se reproduzca en el campo Multimedia, dentro de sus propiedades (figura 52). Dicha canción debe incluirse en la carpeta de recursos, como se hizo en apartados anteriores con las fotografías.

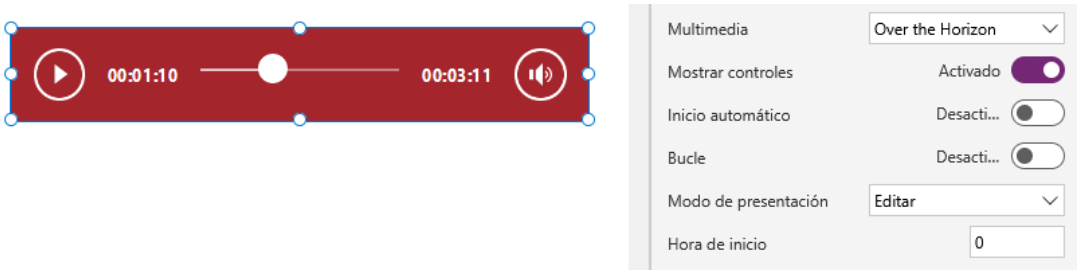


Figura 52: Referenciación de la canción en la propiedad multimedia.

No obstante, si se selecciona una canción que almacenada en el ordenador vía el explorador de Windows que nos proporcionan al pulsar en el campo Multimedia, automáticamente se incluiría en nuestra carpeta de recursos, ahorrándonos así un paso.

Dicho reproductor permite reproducir la canción pulsando el *play*, adelantar o retrasar el momento de la canción e, incluso, subir y bajar el volumen de la pista seleccionada. Proporcionando así control total, y en ejecución, de lo que está sonando.

El problema que surge es que este elemento no nos solucionaría la problemática de Twins por completo. Si bien no se genera ninguna de las inconsistencias que nombrábamos antes, tampoco nos proporciona exactamente lo que se propone en el desafío.

La idea es establecer una canción como música de fondo constante para la aplicación, desafortunadamente esto no es posible en una aplicación desarrollada en Power Apps. Para que suene continuamente deberías pulsar al *play* y, además, el reproductor ocuparía espacio en la pantalla.

Investigando se ha descubierto que, al cambiar de interfaz, la música dejaría de sonar. Por lo que el reto de tener una música de fondo en toda la aplicación se vería bastante complicado con esta solución. Habría que pulsar el botón cada vez que cambiara de interfaz.

Cabe añadir que no se ha encontrado ninguna otra opción de realizar este hito. Por lo que la inclusión de este tipo de funcionalidades sería un lastre bastante grande en la producción de aplicaciones de tipo lúdico.

En segundo lugar, otro problema que surgió fue el añadir imágenes desde una fuente externa. En nuestra aplicación, quisimos realizar un editor de barajas; en este se configuraba el mazo de cartas con el que se deseaba jugar las diferentes partidas.

En el desarrollo de Twins se realizó una interfaz en la que se tenían que incluir diferentes aspectos distintivos del mazo a crear; como podrían ser el nombre de la baraja o las diferentes cartas que se deseaban incluir en dicho mazo. La apariencia es la que se muestra en la figura 53, imagen que constituye una captura de pantalla de la aplicación en ejecución.

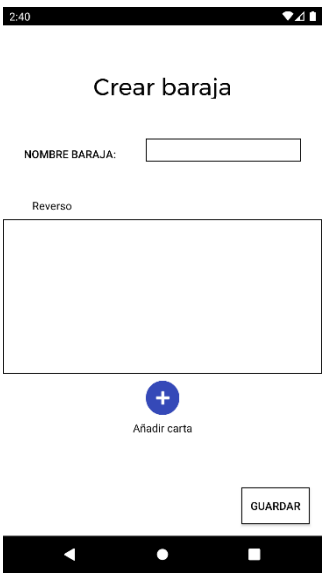


Figura 53: Captura de pantalla del apartado "Crear baraja".

La problemática surgía en el momento de desarrollar la funcionalidad de añadir cartas al mazo. La idea era que, al pulsar el botón, se abriera un explorador de archivos y poder navegar hasta la imagen que querías incluir como carta.

Esta misma carta debía incluirse en el recuadro del centro de la pantalla, poniéndose en cola detrás de la última incluida. Y esto se repetía continuamente, incluyendo cada una de las cartas que se deseaban usar en la partida.

El problema es que no se conseguía integrar bien el uso del navegador de archivos. Al incluir dicha funcionalidad se podía seleccionar la carta, pero esta no se guardaba en la aplicación ni se mostraba en el recuadro que le correspondía.

Para la consecución de dicho hito en Power Apps, podríamos hacer uso del elemento **Agregar imagen** que nos proporcionan. Este provee parte de la funcionalidad que se desea obtener.

Es una imagen en la que, al pulsar en ella, se abre un navegador de Windows en el que puedes elegir el archivo que se va a mostrar en la imagen. Esto nos facilitaría, adaptándonos a nuestro caso de estudio, el poder incluir las imágenes que queremos como cartas.

Se planteó el uso de una interfaz como la que se puede ver en la figura 54. Tratando de imitar la solución de Twins, pero adaptándolo a las necesidades y limitaciones de Power Apps.

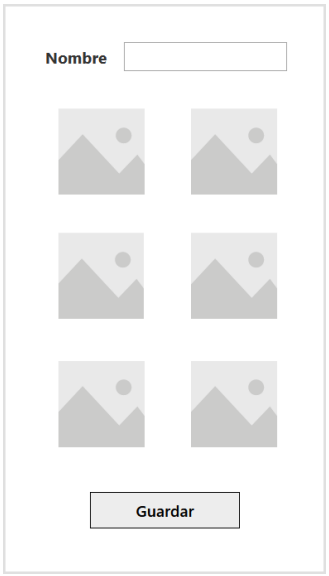


Figura 54: Interfaz de la réplica del apartado "Crear baraja".

En ella destacan los 6 elementos de tipo Agregar imagen en el centro de la pantalla. Estos servirían para almacenar cada una de las cartas que se van a utilizar. De manera que, pulsando en cada una de ellas añadiríamos una foto que se encuentre en nuestro ordenador y, al finalizar, guardaríamos la baraja formada por todas ellas.

Con la pulsación del botón “Guardar” se pretende almacenar dicha baraja para su posterior uso. Esta última funcionalidad también se debería desarrollar para este desafío, ya que el

cómo se comportan los elementos multimedia al intentar homogeneizarlos en la aplicación es muy importante. Un elemento no es bueno solo por la funcionalidad que proporciona, sino también por la facilidad que muestra al interrelacionarse con otras funcionalidades totalmente diferentes.

Implantar las 6 imágenes no conlleva mucho esfuerzo, ya que la funcionalidad viene de serie. Por lo que nuestro principal objetivo, en este caso, es personalizarlas. En cambio, desarrollar la funcionalidad de guardado tiene una complejidad de carácter un poco más elevado; al tener que compatibilizar elementos de tipo multimedia con conceptos relacionados con la persistencia.

Para el guardado de las imágenes, seguimos la misma dinámica expuesta en el apartado relacionado con la persistencia en Power Apps. Usando Azure Blob Storage se puede persistir las fotografías en una cuenta de almacenamiento configurada en nuestra cuenta de Azure.

Una vez expuestos y resueltos los principales puntos críticos que encontramos en Twins, la última parte de este apartado será utilizada para la descripción de algunos de los aspectos que llaman la atención por su utilidad.

Tras la prueba de todos los elementos multimedia, el primer elemento que llama la atención es la cámara. Dicho elemento, nos permite ver aquello que se está viendo por nuestra webcam en ese mismo instante de tiempo. Es decir, nos muestra una retransmisión en vivo de lo que está enfocando dicha cámara. Esto nos permite, por ejemplo, hacer una foto y subirla a un repositorio.

Además, es especialmente llamativo porque la forma de implementar dicha funcionalidad en las tecnologías tradicionales no es tan clara y accesible. De hecho, a priori no tendría las nociones necesarias para abordar dicha funcionalidad en esas tecnologías, mientras que en Power Apps simplemente tendrías que arrastrar el elemento a la pantalla.

Otro elemento llamativo es el visor de PDF. Si bien se encuentra en una fase experimental, igualmente nos permite visualizar el documento PDF que le maticemos. Una posible aplicación podría ser en una aplicación de gestión de documentos. Se podría plantear un repositorio de documentos, almacenados en forma de lista. Y que, al acceder a cada elemento de la lista, podamos disfrutar de una vista previa del documento.

Por último, es interesante comentar el buen papel que desempeña el escáner de código de barras. Este elemento permite incluir un escáner nativo en nuestro dispositivo móvil (nunca vía web), en el que podríamos detectar todo tipo de códigos de barra o QR.

Puede resultar bastante útil en la monitorización de personal de una empresa, por ejemplo. Como es una herramienta para realizar aplicaciones internas, podríamos crear una aplicación para los trabajadores la cual, entre otras cosas, vaya dotada de dicho escáner. En muchas empresas existe la posibilidad de escanear un código QR cada vez que entras o sales de la oficina, con el fin de saber las horas de trabajo del conjunto de los empleados.

La realización de hitos multimedia se ve mucho más simplificada utilizando la ya mencionada plataforma de Low-Code. Se puede justificar dicho efecto por el ahorro experimentado con la supresión de diferentes protocolos densos, a los que sí nos sometíamos en otro tipo de tecnologías. Entendiendo como protocolo la realización de un conjunto de acciones que hay que repetir cada vez que se ha de aplicar una funcionalidad.

Dicho protocolo no suele ser problemático, al constituirse normalmente de acciones de carácter mecánico y de complejidad baja. No obstante, el que sean pasos de poca dificultad no implica que no se gaste un tiempo, valioso en muchas ocasiones, en realizarlos.

Para apoyar el contenido teórico expuesto en las conclusiones, sería interesante introducir una ejemplificación práctica de lo expuesto. Esta se basaría en el desarrollo hermético de la misma funcionalidad; como sería la inclusión de un vídeo.

Resulta interesante matizar que la inclusión de vídeos resultó bastante útil a la hora de introducir los tráileres de las aplicaciones en las mismas. Si lo deseáramos incluir en Twins, dicha aportación estaría sujeta a una serie de cambios en diferentes archivos de nuestro proyecto. Empezando por incluir dicho vídeo en la carpeta de recursos (comúnmente denominada *assets*).

A continuación, se debería incluir un objeto de tipo ***VideoView***, el cual tomaría el papel de contenedor en el que se almacena dicho vídeo. La inclusión genera código en el archivo XML correspondiente a la interfaz en la que lo hayamos añadido, como es de esperar. Lo cual hace que podamos configurar a nuestro gusto aspectos relacionados con el widget.

Una vez añadido, habría que agregar el vídeo al *VideoView*, haciendo referencia al archivo subido a los recursos. Además de incluir un controlador multimedia (*MediaController*), que nos permitiría concluir la inclusión del vídeo. El equivalente en código lo podríamos ver en la figura 55.

```
//Enlace del vídeo al objeto VideoView
VideoView videoView = (VideoView) findViewById("REFERENCIA AL OBJETO VIDEOVIEW");
videoView.setVideoPath("REFERENCIA AL VIDEO DE LA CARPETA ASSETS");

//Creación de MediaController y su posterior enlace con el objeto VideoView
MediaController mediaController = new MediaController(this);
mediaController.setAnchorView(videoView);
videoView.setMediaController(mediaController);
videoView.start();
```

Figura 55: Código para la inclusión de un vídeo en Android Studio.

De esta manera, se puede comprobar experimentalmente que no se trata de un proceso de complejidad alta, pero sí de acciones de naturaleza repetitiva, casi maquinal.

Por otro lado, de querer replicar la misma funcionalidad, se podría representar de manera muy simple en Power Apps. Habría que añadir un elemento de tipo **Vídeo** y, en su campo Media, incluir una referencia al vídeo. Se puede realizar una referencia a la carpeta de recursos multimedia de nuestro proyecto o, incluso, introducir una URL de un vídeo de internet. Para reforzar esta idea, podemos fijarnos en la figura 56, en la que se incluye un video en la aplicación haciendo referencia a una dirección de Youtube.

Figura 56: Inclusión de vídeo mediante una URL.

Como se ha podido comprobar, en este caso, la inclusión de este tipo de tecnologías nos ayudaría a reducir complejidades y protocolos. Sin embargo, también es importante mencionar algunas limitaciones conocidas que nos podemos encontrar en la implementación de este tipo de hitos. Si bien, nos ayudaría a ahorrar bastante tiempo, hemos encontrado (y documentado) grandes limitaciones a lo largo de este desafío. La más llamativa es la dificultad de alcanzar funcionalidades multimedia personalizadas.

Es una herramienta potencialmente buena en este campo, pero, puede generar malos resultados si quieres realizar comportamientos o incluir elementos muy específicos. Como se ha demostrado empíricamente, se comporta muy bien en casos muy puntuales (inclusión de vídeo), no obstante, la capacidad de adaptación al caso de uso es muy limitada.

Hemos enfrentado casos en los que se pueden realizar soluciones parecidas a las que nos proponemos, pero no exactamente lo que se quiere. Además, es importante matizar que, estas conclusiones se han alcanzado tras un periodo intenso de formación; y se ha encontrado, tanto en foros como en documentación oficial, que muchas cosas no eran posibles.

Otras restricciones son aquellas que están impuestas sobre los archivos multimedia que podemos añadir. En cuanto al tamaño total de los archivos, no podrían superar los 200 MB, siendo 64 MB el límite individual de cada uno. Haciendo referencia a la compatibilidad, solo habría unos tipos de archivos aceptados, como pueden ser jpg, gif, png, mp3, entre otros.

De esta manera, quedaría a criterio del impulsor de la aplicación el evaluar si en su caso particular es rentable el uso de esta herramienta. Teniendo en cuenta, sobre todo, las limitaciones.

Se podría concluir con que esta rama de la herramienta presenta posibilidades con bastante potencial en funcionalidades muy particulares. Lo cual no excluye que, en un futuro, este ámbito se vea reforzado.

5.8 Integración de un mapa de Google y creación de rutas

En este apartado, se tratará uno de los puntos más importantes de la aplicación de Adventour, el eje central de esta; al cual le proporcionamos la gran parte de los recursos dedicados a la elaboración de la aplicación. Este hito se basa en la inclusión de un mapa y la elaboración de rutas.

Como primer reto, se planteó la problemática de incluir un mapa en la misma aplicación. Queriendo obtener una imagen satelital de una zona determinada, teniendo así una primera toma de contacto con el desafío.

En dicha herramienta, para realizar vistas de mapas, geolocalizar puntos y situarlos en el mismo mapa, supone una gran ayuda el uso de la **API de Google Maps**. Nos aprovechamos, en este caso, de la abstracción que nos proporcionan las APIs, al igual que planteamos en la solución de origen implantada en Adventour.

Para acceder a los servicios que nos proporciona Google, sería necesario realizar una petición al *endpoint* que es facilitado por la misma empresa para acceder a la funcionalidad ("https://maps.googleapis.com/maps/api/staticmap"). Recibiendo como respuesta un mapa, en forma de imagen.

Para realizar una petición al servicio es necesario parametrizar diferentes factores (empiezan a partir del símbolo "?", separados usando un *ampersand* y al final de la URL proporcionada). Con la idea de contextualizar, se definirán los más utilizados.

El primero que se nos solicita es el denominado centro, básicamente es el punto central que deseas que adopte la pantalla. Por otro lado, también hay que matizar el *zoom*; es decir, qué nivel de acercamiento o alejamiento deseas que tome el mapa. Además de puntualizar el *size*, en otras palabras, los píxeles que quieres que tenga de resolución dicho mapa.

Todo esto acompañado de la configuración del tipo de mapa que se desea que adopte (hay muchos donde elegir; vista normal, en blanco y negro, modo oscuro…) y de la *API Key*. Este último punto es importante ya que, como gran parte de los servicios, conllevan un coste determinado [36]. Para esta API, el servicio de Google te proporciona cierto crédito mensual.; de no llegar a gastar más de 200 dólares al mes (en el caso de la API de *Maps*, *Routes* y *Places*), el servicio sería gratis. A partir de ese punto, comenzarían a cobrar dichas prestaciones. Para dicho cobro disponemos de la *API Key*, es una clave distintiva que sirve de identificador; es usada para asignar todos los cargos que son producto del uso del servicio. Dicha *API Key* se debe solicitar en la plataforma para aplicaciones de desarrollo web Google Cloud Platform[28].

En la web de Google Cloud Platform es necesario registrarse, aceptar los términos y, además, incluir una tarjeta de crédito; en la cual se cargarán los diferentes pagos. Esta misma plataforma nos permite gestionar, desde una consola habilitada para ello, todos los aspectos relacionados con las APIs y su uso.

[28] Google Cloud Platform – Página oficial: https://cloud.google.com/

En un principio puede generar cierto reparo o rechazo el cobro de ciertos servicios, no obstante, el cobro de los servicios está impuesto de tal manera que no te supone ningún esfuerzo. Por ejemplo, en el caso de la API que utilizamos (***Maps Static API***), cobrarían 2 dólares por cada 1000 peticiones si se realizan menos de 100,000 peticiones al mes, o 1.60 dólares si se realizan más [37]. Todo esto, teniendo en cuenta que tenemos 200 dólares mensuales de uso gratuito, haría que en las aplicaciones de pequeño tamaño se genere un coste simbólico o, directamente, no se genere coste.

Los parámetros ya expuestos no son necesarios en su totalidad, es decir, hay algunos que son obligatorios y otros que son opcionales. Suelen ser todos opcionales, las excepciones son *size* y la *API key*. También son necesarios el centro y el *zoom* cuando no haya marcadores presentes, de lo contrario son opcionales.

Para incluir el mapa basta con añadir a la interfaz un objeto de tipo **Imagen** y, en la propiedad ***Image***, introducir la petición URL que se ha explicado anteriormente.

Como apunte, haría falta recordar que el uso del *ampersand* no es exclusivo para la concatenación de parámetros en la petición, también se puede utilizar para concatenar variables a un String. Como en este caso se hace con la *API key*; al ser un aspecto sensible de la aplicación se ha decidido no mostrarla en la captura. En su defecto se ha almacenado en una variable global.

```
"https://maps.googleapis.com/maps/api/staticmap?center=39.482369,-0.343578&zoom=12&size=600x400&maptype=roadmap&key="& _apiKey
```

Figura 57: URL de la petición ocultando la API key.

En la figura 57 se puede ver fácilmente cómo se realiza una petición. Se escribe la URL del *endpoint* al que queremos acceder y posteriormente se agrega un "?" seguido de los diferentes parámetros de la búsqueda. En este caso, especificamos un centro (cuyas coordenadas son cercanas a Valencia), el *zoom* que se requiere, la resolución de la fotografía (600x400) y el tipo de mapa (*roadmap*). Lo que se comentaba de la *API key* se puede ver al final de la petición; se concatena al String una variable global que contiene nuestra clave.

Para la consulta de ejemplo que se ha planteado se devolvería una respuesta como la que podemos ver en la figura 58, en la que se pueden ver reflejados de manera gráfica los diferentes parámetros matizados.

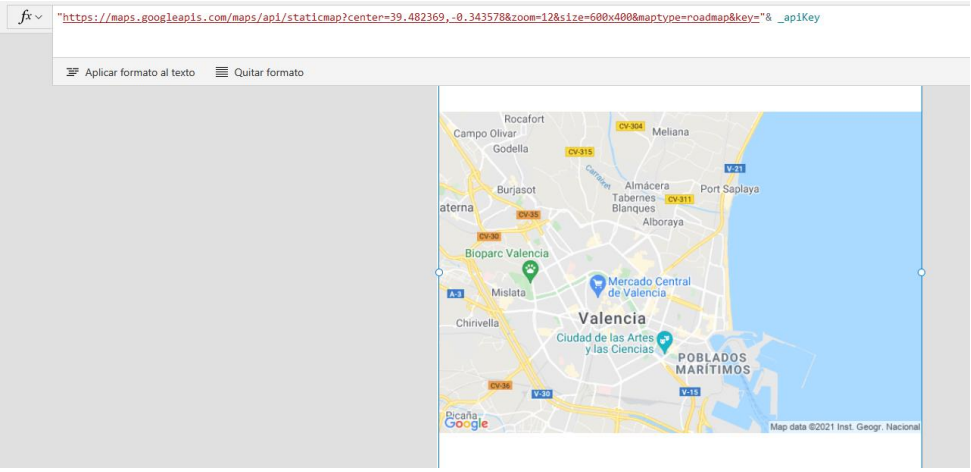


Figura 58: Resultado de la petición detallada.

Para incluir un marcador se puede añadir una cadena al final de la URL, como un parámetro más de la petición de servicio HTTP (figura 59). Sin embargo, dista un poco de los otros parámetros; en este habría que matizar otros aspectos como el color del marcador, el texto mostrado (*label*) y la longitud y latitud de dicho señalador. Se podría decir que es un parámetro al que hay que proporcionarle más parámetros.

```
"https://maps.googleapis.com/maps/api/staticmap?center=39.482369,-0.343578&zoom=12&size=600x400&
maptype=roadmap&key="& _apiKey & "&markers=color:blue|label:H|39.482369,-0.343578"
```

Figura 59: Inclusión de marcadores en la petición.

Concatenaríamos los marcadores justo después de la *API Key*. Como es posible ver, la introducción de parámetros dentro del parámetro *markers* se realiza de forma distinta a la que llevábamos a cabo anteriormente. En vez del "=" para matizar el valor del parámetro utilizamos un ":" y en vez de "&" para separar cada uno de los parámetros utilizaríamos "|" [38].

Incluyendo en el marcador expuesto, obtendríamos la salida que se muestra en la figura 60. En la que es posible ver un marcador de color azul ("color:blue"), con un texto como el que se expone en la petición ("label:H") y en las coordenadas que se matizaban.

Figura 60: Resultado de la petición con marcadores.

Como datos curiosos de los marcadores, se puede decir que no tienen por qué ser únicos, podríamos incluir múltiples marcadores. Además de poder especificar valores dinámicos, haciendo que, por ejemplo, el color de los marcadores varíe conforme a ciertos estímulos.

En cuanto a la comparativa con Adventour, es importante decir que, los pasos para el uso de la API son los mismos. Además, las dinámicas que se siguen son las mismas; se realiza en ambos casos una petición a la URL habilitada y, en respuesta a dicha petición HTTP, se obtiene una imagen (de tipo GIF, JPEG o PNG).

La comparativa no nos proporciona conclusiones muy distintivas; la dificultad es similar para ambas aplicaciones. Esto nos hace pensar, por tanto, que los desarrolladores de la herramienta Low-Code han realizado un buen trabajo; haciendo que la capacidad de integrar APIs sea similar a la de otras tecnologías más veteranas.

Por otro lado, nos quedaría comprobar el potencial de la herramienta para la elaboración de rutas. Las rutas constituyen una parte realmente importante en la aplicación que se desarrolló, Adventour. Se trata de un punto muy importante ya que es la principal funcionalidad de la aplicación, donde reside la lógica más amplia y se gastó

un tiempo sustancial en realizarlo. Se dedicaron alrededor de 13 horas de desarrollo, más las posteriores mejoras que se realizaron.

Buscando realizar una misma funcionalidad, o similar, para Power Apps se encontró una solución óptima. Esta se basa en hacer uso de la función ***Launch*** (figura 61); esta nos permitiría, en nuestro caso, iniciar una página web.

Para ello, en pantalla se ha habilitado un botón, en el cual incluiremos en su propiedad *onSelect* (se ejecuta cuando se pulsa el botón) dicha función *Launch*. De esta manera, al pulsarlo, nos dirigiría a la web que hayamos matizado en la función.

Launch(*Address* [, *ParameterName1*, *ParameterValue1*, ...])

Figura 61: Función Launch en la documentación de Microsoft [39].

La función *Launch* tiene tres parámetros de entrada, con los que configurar aquello a donde nos dirige; dos de ellos son opcionales y, el restante, es obligatorio. El parámetro obligatorio sería la dirección; la cual constituye la URL de la web.

Los parámetros opcionales son los parámetros y el destino; que resultan ser los valores que se desea pasar a la web destino y la pestaña del navegador en la que se inicia la web, respectivamente. Pero, en el desafío actual, solo haremos uso del parámetro dirección [39].

La dirección que vamos a acceder será “https://www.google.com/maps/dir/”; dicha URL nos lleva directamente a Google Maps, en el apartado de crear una ruta. Lo que nos queda para poder realizarla correctamente sería proporcionarle los valores de la latitud y longitud del lugar de partida y del de destino, separados por un “/”.

De esta manera, se realizaron una gran cantidad de pruebas, con el fin de comprobar su correcto funcionamiento. Una de ellas fue la que se muestra en la figura 62, dicha ruta está configurada para guiarte desde la Escuela Técnica Superior de Ingeniería Informática a la Ciudad de las Artes y las Ciencias.

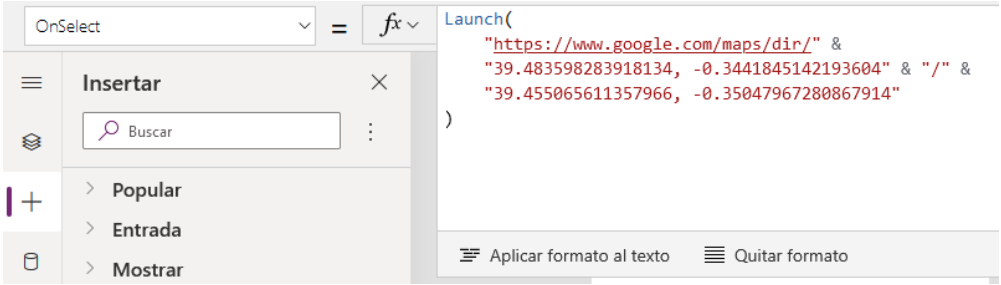


Figura 62: Ejemplo de uso de la función Launch.

Como se puede apreciar, se introdujo la URL concatenando las latitudes y longitudes correspondientes a los sitios mencionados. En la figura 63 se puede ver la respuesta a dicha consulta.

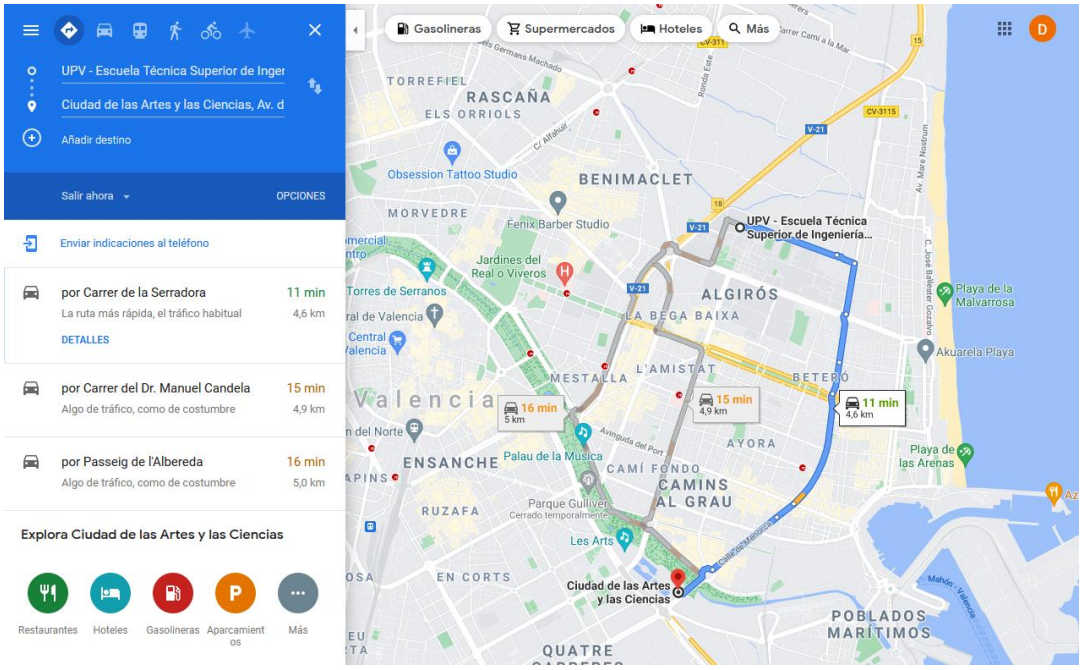


Figura 63: Ruta resultante de la petición realizada.

Esta sería la dinámica con la que se realizaría la funcionalidad de crear una ruta en Power Apps, haciendo uso de aquello que nos proporciona ya realizado Google Maps. Aunque, esto supondría un lastre en otros aspectos.

Cuando se utiliza una solución externa, la capacidad de personalización de lo desarrollado disminuye. Por ejemplo, añadir funcionalidad más allá de lo que proporcionan resulta muy complicado. Es decir, supone el riesgo de tomar una posición muy dependiente de las capacidades de interrelación que tenga dicha solución externa. Por lo que, entre otras cosas, la escalabilidad se podría ver afectada.

Esta solución podría tomarse como un aspecto positivo para Power Apps, porque proporciona mucha facilidad para realizar la funcionalidad de la manera más optima en términos de tiempo. Cabe añadir que en Adventour no tuvimos en cuenta esta opción debido a que, quizás, no se expone de una manera tan clara como se proporciona en la herramienta Low-Code (el uso de una conocida función con un solo parámetro).

Esta pequeña dificultad encontrada, a la hora de buscar diferentes soluciones en Adventour, nos hizo que en dicho proyecto no encontráramos una solución tan sencilla en primera instancia, de modo que nos encaminamos hacia una más compleja. No pudiendo, por ejemplo, contrastar cuál de las dos opciones que se presentan nos hubiera venido mejor.

Esta sencillez en el uso que se ha encontrado, al buscar una solución más atractiva, puede venir dada por el conocimiento previo de Google Maps; la capacidad de encontrar soluciones cuando se domina una materia es mayor. Esta previa experiencia

habría hecho saber cómo movernos entre los foros en los que se tratan temas de este tipo y, también, por la documentación que se proporciona desde Google.

Hablando de aquello por lo que se apostó en Adventour: se realizó una compleja solución basada en realizar un algoritmo que, con la ayuda de la API, nos proporcionara de manera visual una ruta desde el punto inicial hasta el punto final que se notifican.

Se podría decir que la comparación no es muy concluyente si dejáramos las conclusiones en este punto. Por ello, se encontró que se puede dar también la solución propuesta para Power Apps en el entorno de Adventour.

La duda que nos surge ahora es, ¿podría realizarse, el complejo algoritmo que planteamos, en Power Apps? Actualmente, con la formación y contextualización obtenida, la respuesta es no. Constituye un algoritmo demasiado complejo para reproducirlo en esta herramienta.

Se considera óptima la solución que hemos encontrado recientemente debido a que se puede realizar en cuestión de 10 minutos, lo cual supone un ahorro de tiempo grande. Pero, es importante recordar, que en Adventour se realizó una herramienta que creaba las rutas sin necesidad de hacer uso de aplicaciones externas. Por lo que todas las desventajas que surgen del uso de este tipo de aplicaciones se evitan.

A partir de ahí queda como deber de cada uno el concluir si la necesidad de utilizar una aplicación de un tercero se ve justificada con el notable ahorro de tiempo que esto implica.

Como conclusión general, podemos decir que se obtiene un rendimiento similar en ambos paradigmas. Eso sí, con las tecnologías tradicionales se puede realizar tanto la solución más sencilla como la compleja. Mientras que en Power Apps solo se ha podido realizar la opción que se ha expuesto.

Esto se debe a la mala relación que tienen las herramientas Low-Code con la introducción de algoritmos complejos. Haciendo de esto una desventaja preocupante si se deseara escalar la aplicación u obtener funcionalidad de un nivel de complejidad más alto.

Pero, lo que sí que es importante decir, es que resulta sorprendente la buena relación que expone la herramienta Low-Code con el uso de APIs, dejando un rendimiento muy bueno en términos de compatibilidad.

5.9 Extensiones disponibles

En prácticamente todos los proyectos se plantea el uso de extensiones. Las cuales nos permiten construir de forma más sencilla una aplicación; extendiendo o customizando las soluciones que se tienen desplegadas.

En esencia, cuando se habla de extensión, se hace referencia a todas aquellas tecnologías que se utilizan en adicción a la herramienta que hemos planteado utilizar en un principio. En nuestro caso serían el conjunto de tecnologías que complementan las funcionalidades que nos ofrecen desde Power Apps. Este planteamiento reconocería los *plugins*, *add-ons* o, incluso, los *frameworks* como extensiones. Entendiendo por *framework* al conjunto estandarizado de prácticas que sirven como referencia para enfrentar una serie de problemáticas específicas.

Como ya se matizó anteriormente en el capítulo de las tecnologías, Power Apps nos proporciona, mediante AppSource, una amplia gama de extensiones que nos podrían facilitar en gran medida el trabajo. Se trata de más de 500 soluciones a nuestro servicio que nos harían ahorrar en el recurso más esencial, el tiempo.

Además, es bastante llamativo la gran cantidad de aplicaciones gratuitas o de prueba gratuita, haciendo así que la funcionalidad que se proporciona con este tipo de extensiones pueda ser disfrutada por un amplio número de usuarios.

Es importante matizar que, como hemos demostrado en el desafío de integrar Google Maps, también se podría hacer uso de APIs. De esta manera, la compatibilidad con agentes externos se ve ampliamente mejorada, haciendo así que la interactuación con elementos externos sea cada vez más cercana a las soluciones que nos proporcionan las herramientas tradicionales.

No obstante, aquello que más sorprendió en este campo, fue el planteamiento de Microsoft con los *frameworks*. Durante el primer semestre del 2021, se encontró una noticia que hablaba sobre un *framework* que se había desarrollado para Power Apps; este se denomina **Power Apps Component Framework**[29].

Power Apps Component Framework permitiría al conjunto de desarrolladores de aplicaciones generar componentes que nos proporcionarían una experiencia de usuario ampliamente mejorada. Este *framework* es especialmente útil para el trabajo con datos en vista o formularios [40].

Es interesante añadir que, dicho *framework*, a fecha de hoy se encuentra en una versión preliminar pública para las aplicaciones de lienzo. Para las aplicaciones basadas en modelo está disponible.

Sin embargo, basándonos en la experiencia de múltiples usuarios, a pesar de estar en una versión correspondiente a una etapa temprana del desarrollo, genera una gran cantidad de ventajas. Alguna de ellas son la optimización del rendimiento, la reutilización de componentes o el acceso a una gran cantidad de APIs.

[29] Power Apps Component Framework – Página oficial: https://docs.microsoft.com/es-es/powerapps/developer/component-framework/overview

Esta apuesta no solo conlleva los beneficios evidentes que se pueden esperar de la inclusión de una funcionalidad, sino que también constituye una declaración de intenciones. Es sintomático el hecho de que se realicen inversiones en ampliaciones del servicio; enfocadas principalmente a obtener una mayor completitud de la herramienta. Con ello puede darse a entender que la apuesta que se realizan desde las grandes instituciones hacia este tipo de tecnologías es bastante seria. Esto hace que se conserve la aparente facilidad que se muestra a la hora de crear aplicaciones, en conjunto con el potencial que puede proporcionar el uso de componentes más avanzados, como podría ser el uso de *frameworks*.

Se sigue una filosofía muy clara de simplificar y abstraer aquello que resulta complejo o inexistente a oídos de personas no cercanas al mundo informático. Pudiendo servir, en ocasiones, como puente entre "el mundo real" y el tecnológico.

En cuanto a la comparación con las herramientas tradicionales es evidente que la experiencia es un grado. Con esto quiero decir que, una vez estudias las herramientas Low-Code, es bastante patente el reducido tiempo que llevan en el mercado.

Se pueden apreciar una gran cantidad de buenas prácticas y de apuestas fuertes, que nos pueden hacer especular con buenos augurios para un futuro cercano. Pero no deja de ser un ámbito que se encuentra por madurar. Esto se puede ver en diferentes aspectos, uno claro podría ser la posesión de un *framework* que aún se encuentra en fase preliminar; constituyen ideas muy correctas pero que necesitan rodaje. Mientras que el uso de extensiones en el otro tipo de herramientas está más que asentado, evolucionando a pasos agigantados.

No obstante, ¿podría ser esta la realidad de las herramientas Low-Code en un periodo de tiempo no muy amplio? Podría llegar a ese punto, sin embargo, debería tomar como elemento propulsor el conocimiento de los errores que se han realizado en este campo, pudiendo así realizar un planteamiento más positivo.

Un ejemplo podría ser la problemática que surge de la creación de una gran cantidad de *frameworks*, ya que el aprendizaje de dichas tecnologías es limitado. Es imposible ser experto de todos y cada uno de ellos. Por ello, expertos informáticos auguran una centralización de los *frameworks*, haciendo que solo aquellos con más potencial, se desarrollen y popularicen.

Es imposible saber de todo y esto puede ser uno de los detonantes de la falta de perfiles informáticos en el panorama actual; ¿quién cumple la totalidad de requisitos en una oferta de trabajo? Quizás si se centralizase sería más sencillo para todos. De esto se tendría que aprovechar Power Apps, seguir apostando por la sencillez, ajustándose al panorama actual.

5.10 Conclusiones del caso de estudio

A continuación, en la tabla 1, se presentan las diferentes conclusiones extraídas de cada uno de los desafíos que se han planteado a lo largo del caso de estudio:

Desafío	Conclusiones
Diseño y navegación	Power Apps proporciona una mayor facilidad a la hora de diseñar interfaces y navegar entre ellas; evitan situaciones de producción de código repetitivo.
Lista de componentes a partir de una base de datos	El realizar una lista de componentes en Power Apps conlleva un coste temporal menor. La relación de la herramienta con las fuentes de datos es muy buena.
Conexión a base de datos y la persistencia	Power Apps proporciona muchas facilidades para conectar la base de datos. No obstante, da una menor sensación de control en su uso. Además de que el número de conexiones que se pueden realizar es más limitado que en las tecnologías tradicionales.
Motor de búsqueda y filtros	Power Apps te proporciona la capacidad de realizar un motor de búsqueda, con sus respectivos filtros, de manera rápida y eficaz; obteniendo mejores resultados que en las tecnologías tradicionales en términos temporales.
Multimedia	Power Apps nos permite reducir complejidades y protocolos, ahorrándonos tiempo. Sin embargo, existen limitaciones a la hora de incluir funcionalidades multimedia personalizadas; cuando se quiere reproducir un comportamiento muy específico, en ocasiones genera problemas.
Integración de un mapa de Google y creación de rutas	Obtenemos las mismas facilidades en Power Apps que en las tecnologías tradicionales a la hora de incluir un mapa. En la implementación de la creación de rutas nos vemos limitados en Power Apps por su mala relación con la incorporación de algoritmos complejos.
Extensiones disponibles	Las extensiones disponibles para Power Apps no son tan variadas ni están tan asentadas como las de las tecnologías tradicionales. A pesar de esto, se están tomando buenas dinámicas; haciendo que se augure un futuro prometedor en este ámbito de la herramienta.

Tabla 1: Resumen de las conclusiones de los desafíos.

6. Conclusiones y trabajo futuro

6.1 Conclusiones

Los objetivos propuestos al inicio del trabajo se han cumplido. Se han experimentado en primera persona las ventajas y desventajas que se obtienen al incluir una tecnología Low-Code en el ámbito del desarrollo de la aplicación.

Además de ayudarnos a reducir el coste temporal en el desarrollo de una aplicación, Power Apps nos ha echado una mano a la hora de llevar a cabo ideas que, sin su ayuda, no se habrían podido desarrollar. Como ejemplo de esto, cabe resaltar las funcionalidades que nos ofrecen en el apartado multimedia. Esto se debe a que nos proporciona objetos predeterminados que no habría podido incluir en el proyecto por falta de conocimiento técnico. Esto se puede ver, por ejemplo, en la posibilidad de incluir un escáner de barras.

Además, nos facilita otras muchas labores, en las que destacan sobre todo las tareas de diseño de la aplicación. Como se pudo ver en el apartado correspondiente a dicho desafío, permite generar las interfaces con unas dinámicas bastante simples.

Resaltar también la buena relación con el uso de datos externos y la posibilidad de incluir APIs. Todo ello hace que se pueda considerar una opción interesante en el desarrollo de futuros proyectos.

No obstante, no todo son aspectos positivos, también nos hemos encontrado con puntos negativos que nos hacen plantearnos el uso de las herramientas estudiadas. El abstraer aspectos técnicos para proporcionar una mayor sensación de facilidad al usuario de la herramienta puede generar problemas de control. Desde el punto de vista del desarrollador se siente una menor capacidad de control en diferentes aspectos: el mantenimiento o la seguridad de la aplicación. En adición, tampoco se puede controlar la vertiente relacionada con la eficiencia de la solución. Es decir, no se tiene la capacidad de controlar puntos como el espacio que ocupa la aplicación o la eficiencia del código que la compone.

Tanto el mantenimiento como la seguridad y la eficiencia dependerían única y exclusivamente de la empresa que provee el servicio. De manera que se generan una gran cantidad de dudas al respecto, por la falta de claridad que se muestra en estos aspectos.

Por último, como otras desventajas encontradas a lo largo del caso de estudio, hay que destacar la rigidez de la herramienta Low-Code y la falta de compatibilidad con otras herramientas. En cuanto al primer concepto, nos hemos encontrado con que existen algunos hitos difíciles de conseguir por la falta de flexibilidad de la herramienta. Cabe destacar la mala relación que tienen con la introducción de algoritmos complejos y con todo aquello que no entre en las funcionalidades que proporcionan. Ofrecen una serie de funcionalidades limitadas que no dejan cabida a la innovación; se plantea fácil

utilizar objetos predeterminados, pero la creación de nuevos objetos personalizados no es posible.

En cuanto a la compatibilidad, considero que debería potenciarse el compenetrar las herramientas Low-Code y No-Code con las herramientas tradicionales. Quizás las herramientas estudiadas no podrían sustituir a las tradicionales, pero sí complementarlas.

Teniendo en cuenta las ventajas y desventajas que hemos extraído del caso de estudio y del análisis de este tipo de herramientas se plantean unas cuestiones: ¿dónde sería adecuado utilizar este tipo de tecnologías? ¿podrían llegar a sustituir a las tecnologías tradicionales?

El sustituir por completo a sus predecesoras se plantea como un reto bastante complicado porque, en mi opinión, tanto Low-Code como No-Code no cubren todavía todas las necesidades a la hora de desarrollar por completo un proyecto software.

No obstante, el potencial se podría encontrar en la capacidad de complementación que tienen estas tecnologías; el poder complementarse con tecnologías tradicionales para reducir tiempos y recursos en algunas fases de la obtención del producto. De hecho, en el periodo de prueba de los diferentes exponentes que hay en el mercado, se ha encontrado que hay algunas aplicaciones que permiten extraer el código, haciendo que se pueda obtener el trabajo realizado hasta el momento.

Además, también dependería del nicho en el que se quiera impulsar la aplicación, uno de los mejores resultados se obtuvo fue desarrollando una página web. Debido a que Low-Code y, en especial, No-Code no son de carácter “horizontal”; no se pueden aplicar en todos los dominios de aplicación.

Este tipo de tecnologías funcionan muy bien en proyectos de funcionalidades sencillas y repetitivas, como pueden ser los proyectos desarrollados con HTML y CSS; partes fundamentales en la generación de sitios web. La idea es sencilla, si en la programación web hay patrones que se repiten constantemente, ¿por qué no automatizarlos?

Otro caso en el que podría ser útil la aplicación de tecnologías Low-Code y No-Code es en la elaboración de aplicaciones internas, aquellas que elabora una empresa para ella misma. Esto se debe a que aspectos como el mantenimiento, la eficiencia o la necesidad de incluir software con mucho comportamiento, son menos cruciales que en aplicaciones que realizas a terceros.

Para concluir, en lo personal me encuentro muy satisfecho con la elaboración de este trabajo, debido a que me ha permitido estudiar y aprender sobre este tipo de tecnologías, las cuales constituyen un nuevo paradigma en el desarrollo de software. En adición, también he podido estudiar el uso de las principales herramientas Low-Code y No-Code que se encuentran en el mercado. Esto hará que, a la hora de abordar un proyecto personal, el uso de este tipo de herramientas pueda ser considerado como una de las opciones a utilizar.

Además, todo el estudio que se ha realizado hace que se vean reforzados conceptos relacionados con asignaturas que se llevaron a cabo a lo largo de la carrera. Las

asignaturas que están más relacionadas con este trabajo son Interfaces persona computador y Desarrollo de software dirigido por modelos.

6.2 Trabajo futuro

Si nos enfocamos en el futuro del trabajo que se ha planteado, tenemos una gran cantidad de frentes abiertos. Todos ellos relacionados con la continuación del estudio de las tecnologías Low-Code y No-Code.

Con el fin de continuar investigándolas, se plantearía seguir de cerca todos los avances de dichas tecnologías y, en específico, prestar especial atención a las nuevas versiones de Power Apps.

En particular, se debería observar detenidamente todos aquellos conceptos que se han etiquetado como puntos negativos a lo largo de las conclusiones del trabajo; para así cerciorarnos de si este tipo de tecnologías han avanzado en aquello que peor desempeño ejercen.

Por otro lado, en un futuro más cercano, se podría realizar un caso de estudio más extenso. En dicho caso de estudio se incluirían una mayor cantidad de desafíos, los cuales se tratarían de manera más profunda y detallada (al disponer de mayor tiempo y una formación previa en este tipo de tecnologías).

7. Bibliografía

[1] Alores. (2020, 24 enero). ¿Qué es una plataforma low-code? Velneo. https://velneo.es/que-es-plataforma-lowcode/ Consultado el 2 de febrero de 2021.

[2] Gonzalo, F. (2021, 19 abril). ¿Qué es Nocode? Desarrollar webs y apps sin código. No coders. https://www.nocoders.academy/blog/que-es-nocode Consultado el 2 de febrero de 2021.

[3] El movimiento no-code. La democratización del software. (2020, 19 junio). ABANCA innova. http://abancainnova.com/es/opinion/el-movimiento-no-code-la-democratizacion-del-software/ Consultado el 10 de febrero de 2021.

[4] Mayer, H. M. (2020, 6 agosto). High Ambitions For Low Code. Forbes. https://www.forbes.com/sites/hannahmayer/2020/08/06/high-ambitions-for-low-code/?sh=24f2014a6605 Consultado el 15 de febrero de 2021.

[5] Rayome, A. D. (2017, 30 agosto). Report: 60% of apps are built outside of IT, and that's a good thing. TechRepublic. https://www.techrepublic.com/article/report-60-of-apps-are-built-outside-of-it-and-thats-a-good-thing/ Consultado el 27 de febrero de 2021.

[6] Dilmegani, W. B. C., & Dilmegani, C. (2021, 1 agosto). 42 Low Code/ No code Statistics in 2021. AIMultiple. https://research.aimultiple.com/low-code-statistics/ Consultado el 01 de marzo de 2021.

[7] Dans, E. (2020, 9 agosto). Could The No Code Movement Put Programmers Out Of A Job? Forbes. https://www.forbes.com/sites/enriquedans/2020/08/09/could-the-no-code-movement-put-programmers-out-of-ajob/?sh=739efd322345 Consultado el 15 de febrero de 2021.

[8] McKendrick, J. (2020, 18 abril). With so much remote work, the time is ripe for low-code and no-code software development. ZDNet. https://www.zdnet.com/article/with-so-much-remote-work-the-time-is-ripe-for-low-code-and-no-code-software-development/ Consultado el 02 de marzo de 2021.

[9] Pons, C., Giandini, R. S., & Pérez, G. (2010). Desarrollo de software dirigido por modelos. Editorial de la Universidad de la Plata. Consultado el 25 de marzo de 2021.

[10] Meijers, J. (2018, 2 mayo). The difference between low-code and no-code platforms. Triggre. https://triggre.com/the-difference-between-low-code-and-no-code-platforms/ Consultado el 10 de febrero de 2021.

[11] Alores. (2021, 16 febrero). ¿Por qué el movimiento no-code / low-code beneficia a los programadores profesionales? Velneo. https://velneo.es/por-que-el-movimiento-no-code-low-code-beneficia-a-los-programadores-profesionales/ Consultado el 21 de febrero de 2021.

[12] Alexander, F. (2021, 8 enero). Low-Code and No-Code: What's the Difference and When to Use What? OutSystems. https://www.outsystems.com/blog/posts/low-code-vs-no-code/ Consultado el 1 de marzo de 2021.

[13] Cabot, J. (2020, 22 noviembre). ¿Qué es esto del low-code? ¿Dejaremos de programar? Ingeniería de Software. https://ingenieriadesoftware.es/que-es-low-code-no-code-modelos-generacion/ Consultado el 2 de abril de 2021.

[14] Cabot, J. (2021, 3 agosto). Las 5 grandes tecnológicas apuestan por el modelado y la generación de código. Ingeniería de Software. https://ingenieriadesoftware.es/grandes-tecnologicas-apuestan-modelado-low-code-generacion-codigo/ Consultado el 20 de marzo de 2021.

[15] Low-Code Development Platform Market Worth $45.5 Billion by 2025 - Exclusive Report by MarketsandMarketsTM. (2020, 28 abril). Cision. https://www.prnewswire.com/news-releases/low-code-development-platform-market-worth-45-5-billion-by-2025--exclusive-report-by-marketsandmarkets-301048322.html Consultado el 21 de abril de 2021.

[16] Bubble at the Harvard Innovation Lab - Bubble Blog. (2020, 3 septiembre). Bubble. https://bubble.io/blog/bubble-at-the-harvard-innovation-lab/ Consultado el 25 de abril de 2021.

[17] Harvard. (2020, julio). Drive Business Success with a Dual-Track Approach to Transformation. https://hbr.org/sponsored/2020/07/drive-business-success-with-a-dual-track-approach-to-transformation Consultado el 11 de mayo de 2021.

[18] Wayner, P. (2020, 25 agosto). Pros and cons of low-code platforms. CIO. https://www.cio.com/article/3572394/pros-and-cons-of-low-code-platforms.html Consultado el 13 de abril de 2021.

[19] England, S. (2020, 16 enero). What is no code? The pros and cons of no code for software development. Codebots. https://codebots.com/low-code/what-is-no-code-the-pros-and-cons-of-no-code-for-software-development Consultado el 14 de abril de 2021.

[20] Weston, C., & Dowd, M. (2021, 21 julio). Low-Code, No-Code — Are they Right for You? IDC Europe Blog. https://blog-idceurope.com/low-code-no-code-are-they-right-for-you/ Consultado el 15 de mayo de 2021.

[21] Vicent, P., Natis, Y., Iijima, N., Wong, J., Ray, S., Jain, A., & Leow, A. (2020). Magic Quadrant for Enterprise Low-Code Application Platforms. Gartner. Published.

[22] K. (2021, 25 mayo). ¿Qué es Power Apps? - Power Apps. Microsoft Docs. https://docs.microsoft.com/es-es/powerapps/powerapps-overview Consultado el 15 de febrero de 2021.

[23] Peña, A. (2021, 8 abril). Qué es Outsystems y para qué se utiliza. Incentro. https://www.incentro.com/es-es/blog/stories/que-es-outsystems-para-que-sirve/ Consultado el 16 de febrero de 2021.

[24] Rodríguez, T. (2020, 31 mayo). Programar sin escribir código es posible: así funciona la nueva era del No Code. Xataka. https://www.xataka.com/otros/programar-escribir-codigo-posible-asi-funciona-no-code Consultado el 20 de febrero de 2021.

[25] Galván, C. (2020, 11 abril). ¿Qué es Webflow y para qué sirve? -. Desafío hosting. https://desafiohosting.com/que-es-webflow/ Consultado el 25 de febrero de 2021.

[26] G. (2021, 23 marzo). Referencia de fórmulas para Power Apps - Power Apps. Microsoft Docs. https://docs.microsoft.com/es-es/powerapps/maker/canvas-apps/formula-reference Consultado el 1 de junio de 2021.

[27] ¿Qué es el Drag & Drop y para qué sirve? - Neo Wiki. (2021, 11 mayo). NeoAttack. https://neoattack.com/neowiki/drag-drop/ Consultado el 2 de mayo de 2021.

[28] Meganathan, A. (2020, 17 junio). How To Add Excel Data Source In Microsoft PowerApps. C#corner. https://www.c-sharpcorner.com/article/how-to-add-excel-data-source-in-microsoft-powerapps/ Consultado el 1 de mayo de 2021.

[29] Colaboradores de Wikipedia. (2020, 17 junio). Persistencia (informática). Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Persistencia_(inform%C3%A1tica) Consultado el 3 de mayo de 2021.

[30] D'Ambrosio, S. (s. f.). El Concepto de Datos - Monografias.com. Monografias. https://www.monografias.com/trabajos14/datos/datos.shtml Consultado el 3 de mayo de 2021.

[31] López, S. (2020, 17 mayo). Firebase: qué es, para qué sirve, funcionalidades y ventajas. DIGITAL55. https://www.digital55.com/desarrollo-tecnologia/que-es-firebase-funcionalidades-ventajas-conclusiones/ Consultado el 4 de mayo de 2021.

[32] ¿Qué es Microsoft Azure? ¿Cómo funciona? (2021, 13 enero). Tecon. https://www.tecon.es/que-es-microsoft-azure-como-funciona/ Consultado el 6 de mayo de 2021.

[33] G. (2017, 5 febrero). Funciones Filter, Search y LookUp en Power Apps - Power Apps. Microsoft Docs. https://docs.microsoft.com/es-es/powerapps/maker/canvas-apps/functions/function-filter-lookup Consultado el 25 de mayo de 2021.

[34] G. (2017, 24 abril). Funciones If y Switch de Power Apps - Power Apps. Microsoft Docs. https://docs.microsoft.com/es-es/powerapps/maker/canvas-apps/functions/function-if Consultado el 26 de mayo de 2021.

[35] Conejo, E. (2019, 14 febrero). LA IMPORTANCIA DE LA MULTIMEDIA. La trinchera divergente. https://blog.junglacode.org/time-machine/lecturas/la-importancia-de-la-multimedia/ Consultado el 2 de junio de 2021.

[36] Get Started, Maps Static API. (s. f.). Google Developers. Recuperado 5 de agosto de 2021, de https://developers.google.com/maps/documentation/maps-static/start Consultado el 5 de junio de 2021.

[37] Maps Static API Usage and Billing. (s. f.). Google Developers. Recuperado 5 de agosto de 2021, de https://developers.google.com/maps/documentation/maps-static/usage-and-billing Consultado el 5 de junio de 2021.

[38] Using a custom image marker on google static map? (2015, 13 febrero). Stack Overflow. https://stackoverflow.com/questions/28492733/using-a-custom-image-marker-on-google-static-map Consultado el 10 de junio de 2021.

[39] G. (2021, 30 junio). Funciones Launch y Param en Power Apps - Power Apps. Microsoft Docs. https://docs.microsoft.com/es-es/powerapps/maker/canvas-apps/functions/function-param Consultado el 10 de junio de 2021.

[40] N. (2021, 8 junio). Descripción general de Power Apps component framework en Microsoft Dataverse - Power Apps. Microsoft Docs. https://docs.microsoft.com/es-es/powerapps/developer/component-framework/overview Consultado el 20 de junio de 2021.