Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

LiveDiagrams, el Open data en tu móvil
Una App para el futuro

LiveDiagrams, the Open Data on your phone
An App for the future

Liam Joseph O'kelly Herrero

La Laguna, 6 de Septiembre de 2020

Da. **María Elena Sánchez Nielsen**, con N.I.F. 42.848.599-J profesora Titular de Universidad adscrita al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutora

C E R T I F I C A (N)

Que la presente memoria titulada:

"LiveDiagrams, el Open data en tu móvil
Una App para el futuro"

ha sido realizada bajo su dirección por D. **Liam Joseph O'kelly Herrero**,

con N.I.F. 78.648.115-Y.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 6 de septiembre de 2020

Agradecimientos

En primer lugar, quiero dar las gracias a mi familia por todo el apoyo que me han mostrado en los buenos momentos y en los malos, que no han sido pocos. Por no dejar que me rinda, cuando quise, para que hoy pueda estar aquí.

En segundo lugar, quiero agradecer a la Universidad de La Laguna por estos últimos 5 años, en los que he progresado gracias al esfuerzo y dedicación no solo míos, sino de todo el profesorado.

Por último, quiero también hacer una mención especial a Elena Sánchez Nielsen, por su tutorización en este Trabajo Fin de Grado y su gran labor como profesora estos últimos años.

Licencia

Resumen

El objetivo de este trabajo ha sido la búsqueda de información y el desarrollo de una aplicación móvil que sea capaz de representar datos mediante gráficas. Durante su desarrollo, se han investigado numerosos tipos de diagramas, observando todos los tipos que existen, los datos que son capaces de representar y sobre todo en su accesibilidad para la gente.

Una de las características principales de esta aplicación que se ha desarrollado, es su capacidad para representar datos en tiempo real. Es decir, cada vez que se genera un diagrama, lo hace con los últimos datos de los que se dispone. Estos datos, se obtienen mediante portales de "Open Data" de sitios institucionales como pueden ser Ayuntamientos, Cabildos o Gobiernos autonómicos. Debido a la legislación actual, estos datos deben ser actualizados y publicados de forma periódica y totalmente gratuitos en los respectivos portales de cada institución.

El "Open Data" es un modo de trabajo que se está llevando a cabo en las instituciones públicas en España, de cara a fomentar la transparencia, y para suministrar valiosa información al público. El problema es saber interpretarla y traducirla a un diagrama visual, para que cualquier persona pueda entender dicha información. Con ese objetivo en mente, nace LiveDiagrams.

Palabras clave: Diagrama, OpenData, Portales, Transparencia.

Abstract

The objective of this work has been the search for information and the development of a mobile application that is capable of representing data through graphs. During its development, numerous types of diagrams have been investigated, observing all the types that exist, the data that they are capable of representing and, above all, their accessibility to people.

One of the main features of this application that has been developed is its ability to represent data in real time. In other words, every time a diagram is generated, it does so with the latest available data. These data are obtained through "Open Data" portals from institutional sites such as municipalities, councils or autonomous governments. Due to current legislation, these data must be updated and published periodically and totally free on the respective portals of each institution.

OpenData is a way of working that is being carried out in public institutions in Spain, in order to promote transparency, and to provide valuable information to the public. The problem is knowing how to interpret it and translate it into a visual diagram, so that anyone can understand this information. With that goal in mind, LiveDiagrams is born.

Keywords: Diagram, Open Data, Portals, Transparency.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1 Introducción

1.1 La tecnología y su evolución

En el mundo moderno, la tecnología juega un papel clave en la sociedad. La humanidad ha ido evolucionando hasta llegar a la actualidad, pasando por distintas fases de desarrollo tecnológico. Hace 2000 años, los humanos vivían en cuevas, casas de madera y barro. Hace 200 años, aún no existía la luz (luz eléctrica). No fue hasta 1879 [1] que Thomas Alva Edison consiguió que la primera bombilla se iluminara durante 13 horas y media, antes de romperse.

La tecnología que nació de la luz eléctrica ha seguido evolucionando, alcanzando un ritmo incesante, que comenzó en el año 1890 con la invención de la primera máquina tabuladora [2] por Herman Hollerith. Posteriormente, poco antes de la segunda guerra mundial, Alain Turing definió el concepto moderno de algoritmo y de la Máquina de Turing [3], que sentarían las bases de lo que posteriormente se convertiría en el primer ordenador.

Debido a la definición un poco ambigua del concepto de “computadora”, no se puede saber con seguridad cuál fue el primero de todos. Esto se debe a que si nos basamos en la definición de “máquina que computa” [4] entraría por ejemplo la “máquina analítica” desarrollada por Charles Babbage entre 1880 y 1910 siendo esta mecánica y no eléctrica. Si nos queremos centrar en la primera computadora digital, tenemos que avanzar hasta el año 1944 (hace tan solo 80 años) donde se creó ENIAC (Electronic Numerical Integrator and Computer), como la primera computadora de propósito general [5].

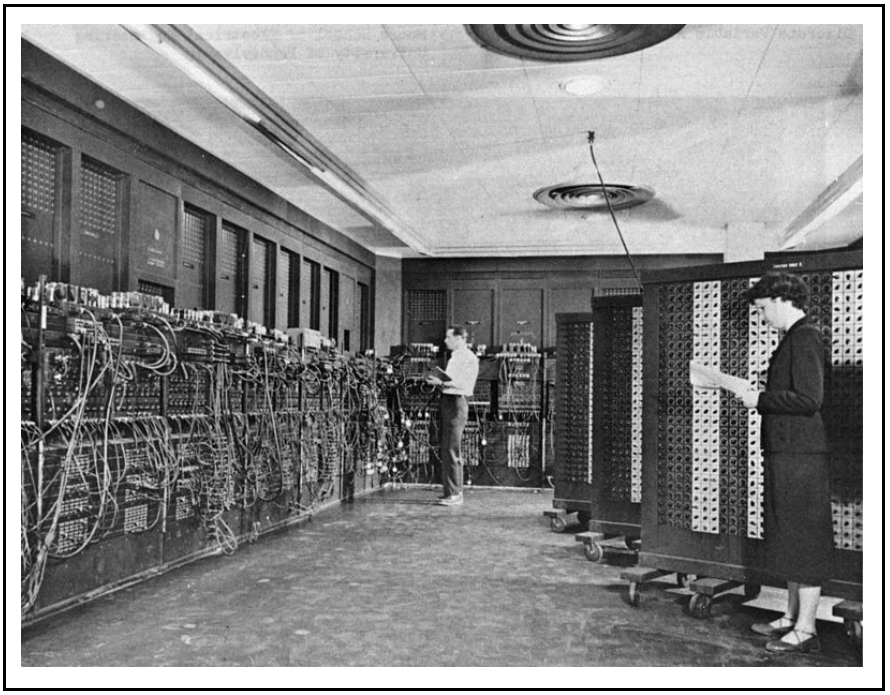

Figura 1.1: ENIAC, primer ordenador digital, año 1947

Fuente: Wikipedia [6]

A partir de aquí, el ordenador ha ido evolucionando. Tienen mucha más memoria, pasando de usar tarjetas perforadas y tubos de vacío, a microchips. Son infinitamente más rápidos (el intel 4004 inventado en 1971 trabajaba a 740 kHz y los ordenadores domésticos hoy en día trabajan a 3-4 GHz). Y una de las cosas más importantes, se han hecho pequeños. Antes los ordenadores abarcaban salas enteras de almacenes, las tarjetas perforadas llenaban cajas y cajas. Ahora tenemos un aparato en la mano, llamado móvil, que nos acompaña en nuestro día a día, que es millones de veces más rápido y con más memoria que dichos ordenadores.

1.2 Objetivos

El proyecto que se va a presentar, se basa en la implementación de una aplicación móvil capaz de representar datos en tiempo real extraídos de diferentes fuentes de información online. Como tal, se buscan los siguientes objetivos:

- **Adaptabilidad**: Se busca que mediante ligeros cambios en la aplicación, esta se pueda usar para diferentes clientes. Por ejemplo, cambiando la fuente de la que se extraen los datos y a su vez, modificando ligeramente el correspondiente parser, la estructura de la aplicación sirve para mostrar otros datos a nuevos clientes

- **Accesibilidad**: El objetivo principal de esta aplicación es mostrar los datos, pero una parte importante es que esos datos sean accesibles a la mayor parte posible de clientes. Eso engloba a la gente mayor, a la cual le cuesta más adaptarse a los cambios [7]. Por ello, se usarán diagramas más antiguos a los cuales, las anteriores generaciones ya están habituados y no se usarán nuevos diseños.

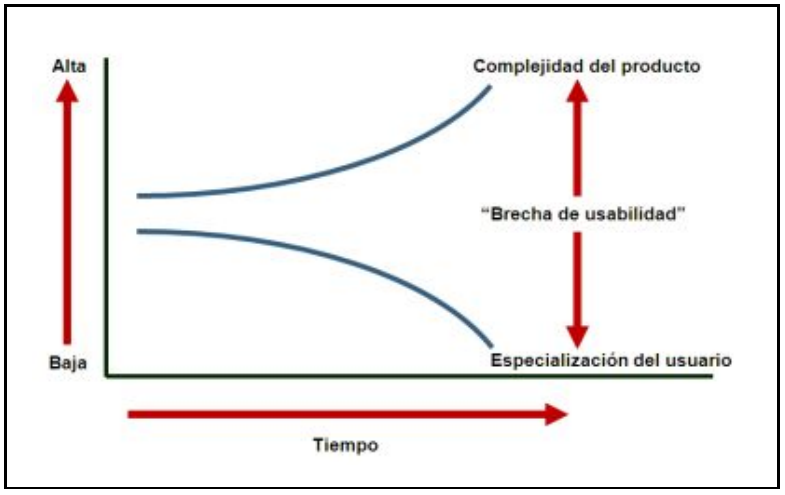

Figura 1.2: Diagrama brecha usabilidad con la edad

Fuente: Centro Estatal de Autonomía Personal y Ayudas Técnicas (CEAPAT)

- **Limpieza visual**: Siguiendo la línea del anterior objetivo, a más sencilla sea visualmente la aplicación, más sencillo será que sea accesible para todos los usuarios. Siguiendo esta norma, se desarrollarán únicamente las funciones por las que nace LiveDiagrams y no tendrá funcionalidades adicionales.

Capítulo 2 Estado del arte

2.1 El Proyecto

En este documento, se presenta el proyecto que se ha realizado y se expone las diferentes etapas que se han llevado a cabo para el diseño, implementación y testing de la aplicación que se ha desarrollado. En este capítulo, se situará el contexto histórico de la información y se hablará de la importancia de los diagramas y gráficas en la vida moderna, además del OpenData.

2.2 Antecedentes - Contexto Histórico de la Información

En el mundo actual, vivimos en la era de la información. La información es algo que siempre ha estado relacionado con el ser humano, al igual que la cultura. Desde tiempos remotos, el ser humano acumula información. Al principio se hacía dibujando en las paredes con pinturas en lugares que hoy son tan famosos como las Cuevas de Altamira. Con el tiempo, el medio donde reflejar la información evolucionó y se empezaron a hacer tallados en piedra con cinceles. Grabados que en ocasiones, aún hoy perduran como los realizados por la Civilización Maya o por la Egipcia. Con el paso de las décadas, llegó el papiro. Su principal ventaja era su reducido volumen. En una lámina muy fina, se podía acumular mucha información y así nacieron los pergaminos, que eran rollos muy largos de papiro. Este concepto fue evolucionando hasta que se llegó al concepto del libro que hoy en día conocemos, pero que no es algo moderno, sino que existe desde hace siglos.

Con el paso de los años, los ordenadores y móviles han ido evolucionando y han ido bajando de precio, haciendo que se haya pasado de tener unos pocos ordenadores en todo un país, a que ahora mismo todo el mundo tiene un ordenador en la palma de la mano con conexión a internet de alta velocidad.

Debido al gran acceso a la red, han ido naciendo diferentes negocios entorno a la información que se mueve por internet y con el aumento de usuarios que se ha experimentado, se ha llegado a un crecimiento exponencial del volumen de datos que se genera cada día en todo el planeta.

Con las nuevas tecnologías desarrolladas para los canales de información de internet (de fibra óptica de alta velocidad, 4G, 5G), han nacido numerosos sitios web dedicados al streaming, algo que hasta hace una década era impensable.

Esto se puede observar en la Figura 2.1 que se muestra a continuación, donde se ve la evolución del volumen de datos generados y los valores que se esperaban para los años futuros. “Se esperaban” porque ya estamos en esos años futuros y de momento se han cumplido.

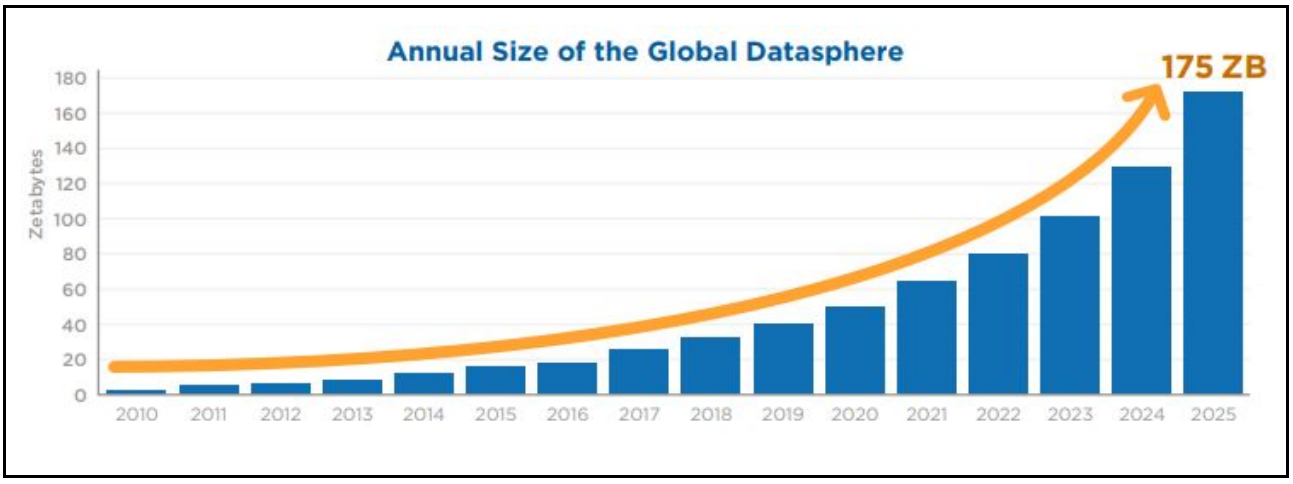

Figura 2.1: Evolución del volumen de datos, año 2018

Fuente: Fundación Bankinter [8]

2.3 Open Data

Habiéndonos situado en el contexto histórico de cómo se almacena la información, ahora toca hablar de cómo se hace hoy en día. En la actualidad, como ya se ha mencionado, vivimos en la era de la información y gracias al internet y a la tecnología de la que disponemos, podemos transferir información a alta velocidad.

Gracias a ello, se han desarrollado sistemas de almacenamiento basados en la nube, donde ahora la información ya no se guarda en disco duro a nivel local, sino que hay grandes servidores repartidos por el mundo [9], que se usan para almacenar la información. Las instituciones y otros organismos, se apoyan en ellos, para almacenar su información y así crear grandes fuentes de información. A esto hay que sumar el hecho de que en muchos países se ha instaurado la filosofía de compartir información de forma gratuita, ya sea para mostrar transparencia o para mejorar su modelo de negocio.

Partiendo de las dos bases (almacenamiento en la nube y grandes fuentes de información gratuita), nace el concepto de OpenData [10]. Este concepto, se basa en la filosofía de compartir la información a través de portales en internet, para que cualquier individuo u organismo pueda usarlos tanto para sacar un rendimiento económico, como para uso particular.

Esta información se puede presentar en diferentes formatos de archivo, además de que en España aún no existe un estándar para cómo almacenar los valores. Con esto, nos referimos a que el portal de opendata de fuerteventura ofrece datos desglosados por municipios y el portal de lanzarote lo hace desglosados por barrios. Debido a esto, en ocasiones se hace muy difícil sacar correlaciones de datos entre ambos, porque no están estructurados igual. Los formatos más comunes son: JSON, csv, geoJSON, KML, XLS.

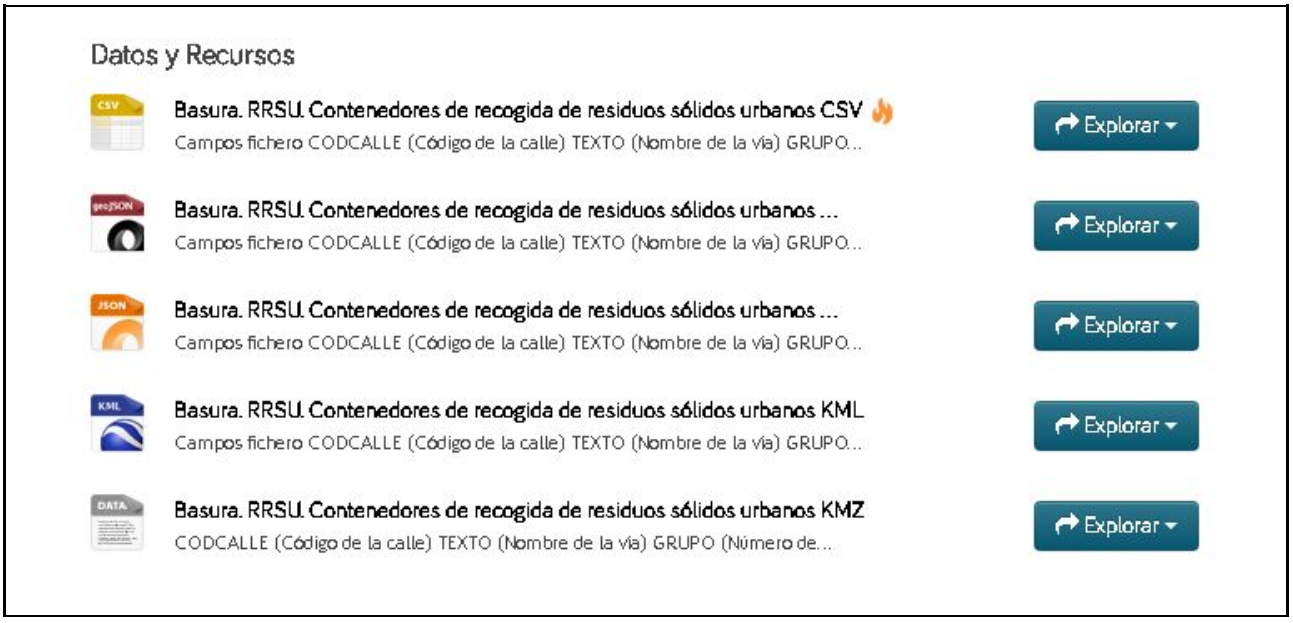

Figura 2.2: OpenData. Ubicación de contenedores en Santa Cruz.

Fuente: Portal OpenData Santa Cruz [11]

No obstante, en España está definida una guía de buenas prácticas [12] . Además, se analizan mediante barómetros, la calidad de los datos que se ofrecen en los distintos portales del gobierno, basándose en los siguientes aspectos:

- Disponibilidad de datos online
- Formatos Reutilizables
- Completitud
- Gratuidad
- Licencias abiertas
- Actualización
- Facilidad de acceso

Los ficheros de datos se pueden separar en 5 categorías de acuerdo con Tim Berners-Lee, inventor del esquema “5 estrellas de Datos abiertos” [13], las cuales son cada una mejor que la anterior.

- **1 Estrellas**: son aquellos ficheros, cuyos datos no siguen ningún estándar y están almacenados en cualquier formato de archivo. P.e: una imagen o pdf.

- **2 Estrellas**: son aquellos ficheros de categoría 1, pero que ahora tienen un formato legible. P.e: un excel.

- **3 Estrellas**: son aquellos ficheros de categoría 2, que usan un formato legible, pero que es un formato libre. Es decir, no es un excel, sino un CSV.

- **4 Estrellas**: En esta categoría los datos ya no vienen en un fichero, sino que están incrustados en una URL de la cual se pueden descargar.

- **5 Estrellas**: En la última categoría nos encontramos con datos que cumplen la categoría anterior, y que además están enlazados con otros datos similares publicados por otras organizaciones.

Teniendo en cuenta este sistema de clasificación de calidad de los datos, se entiende por qué España está situada como un referente dentro de Europa, en lo relativo a Opendata [14]. Según el barómetro de calidad del gobierno, en la mayoría de casos, se considera que los datos ofrecidos por los portales autonómicos o regionales, son de buena calidad (el 80% son de categoría 3 [15]).

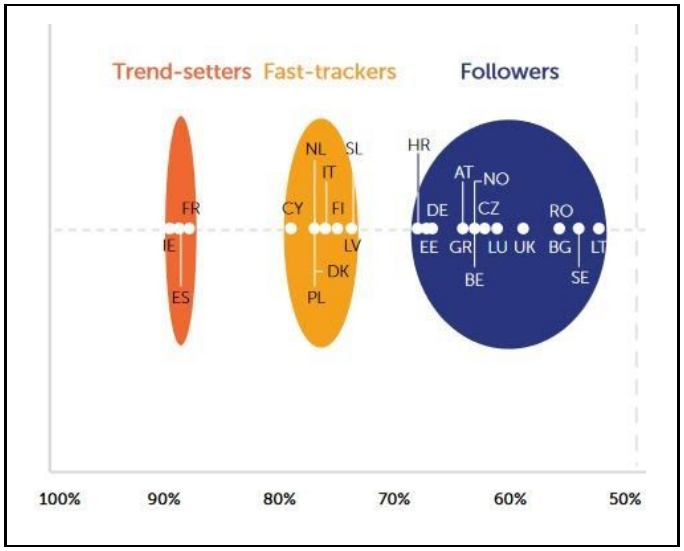

Figura 2.3: Ranking Europa mejores puntuaciones en OpenData, año 2019

Fuente: Gobierno de España [14]

2.4 Diagramas y Gráficas en la Actualidad

Una vez explicado lo que es el Open Data, hay que dar paso al siguiente nivel. Antes de la llegada de internet, el volumen de datos era muy inferior, con lo que fácilmente la gente los podía interpretar, pero en la actualidad, el volumen de datos es cientos de veces mayor. No es raro ver tablas de XML de miles de líneas, en las que llega un punto, que no sabes lo que se está describiendo porque es demasiada información. A partir de ahí, nace la necesidad de representar dichos datos mediante gráficas y diagramas [16].

En algunos casos, no hace falta que sean documentos muy grandes. Basta con que no se sepa muy bien interpretar los datos de una tabla, para que sea imposible entenderlos incluso pasado un tiempo. Sin embargo, será mucho más sencillo y sobretodo más rápido, interpretar los datos de la tabla si están representados con un gráfico. Gracias a la ilustración mediante diagramas, se puede ver más fácilmente por ejemplo, la tendencia a la baja de un valor en bolsa, sin tener que estudiar toda la tabla de índices previos de valor.

A nivel histórico, no es algo moderno el uso de ilustraciones para mostrar información. Ya se hacía en la antigüedad mediante el uso de la cartografía [17]. Durante siglos, los mapas han sido muy usados, no solo para la orientación y navegación, sino para representar información del entorno.

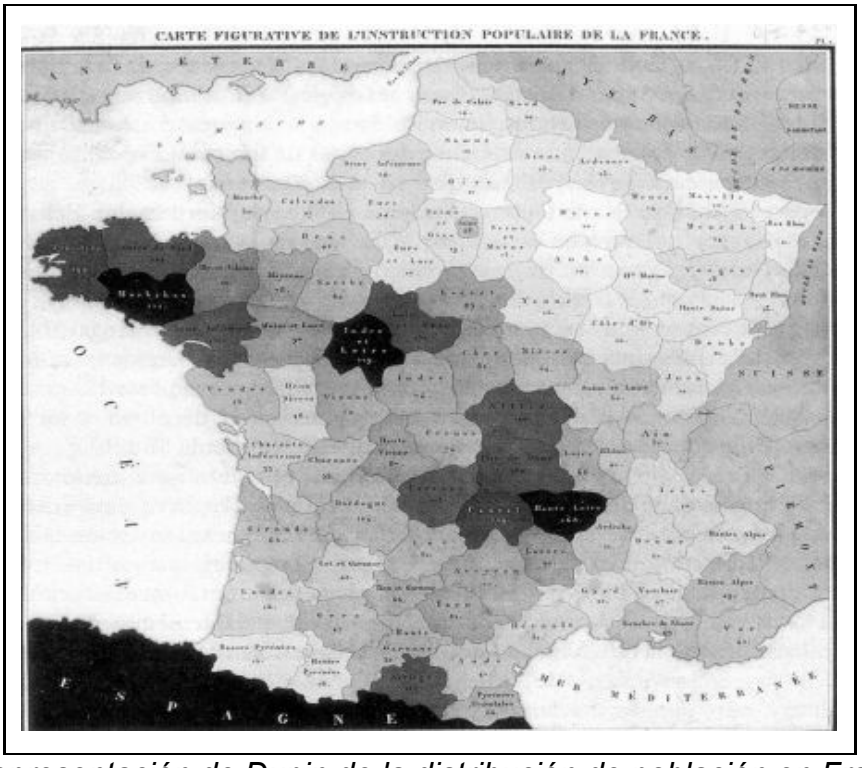

Figura 2.4: Representación de Dupin de la distribución de población en Francia en 1826
Fuente: Informe herramientas. Gobierno de España [18]

El uso de ilustraciones para representar información valiéndose de un mapa, no es algo que solo se haya hecho en la antigüedad. Un claro ejemplo de ello, es la Figura 2.4 en la que se observa la incursión de Napoleón a Rusia. Esta ilustración nos vale también, para entender que la dificultad de representar información mediante gráficas, se centra en que dependiendo de lo que se busque representar, hay mejores gráficas.

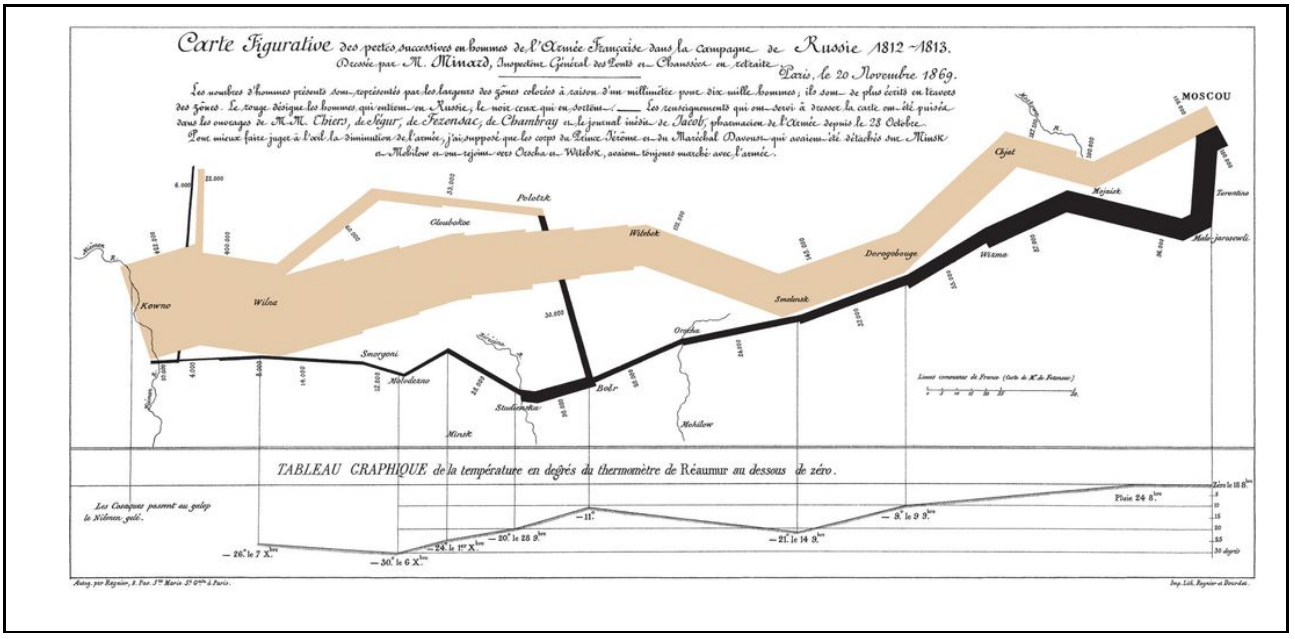

Figura 2.5: Incursión Napoleónica en Rusia
Fuente: Wikipedia [19]

En este diagrama se representa mucha información, pero a la vez, hay que concentrarse para descifrarla. Por un lado, tenemos una línea en el margen inferior, que representa la temperatura que hacía en cada punto de la avanzada de Napoleón en su camino a invadir Rusia. A su vez, las líneas beige y negras, representan vistas desde arriba, el trayecto sobre un mapa que realizó el ejército, pasando por cada ciudad y como se dividieron las tropas. Por otro lado, el grosor de cada línea, indica el número de tropas que quedaban con vida en cada momento y se observa claramente cómo según van avanzando, se van reduciendo de forma drástica debido a las numerosas bajas por el frío de Rusia. Por último, los dos colores de las barras. Estos colores representan el momento en el que cada uno de los dos frentes del ejército, inició su retirada táctica.

El problema de estos diagramas es que a pesar de ser muy completos, necesitan de un gran análisis para entenderlos y el público general no será capaz de hacerlo sin una explicación previa. Por otro lado, tenemos los diagramas comunes como son el diagrama de barras, el diagrama de tarta o histogramas, los cuales son tan habituales que por norma general, cualquier persona los puede interpretar de manera clara, sencilla y rápida.

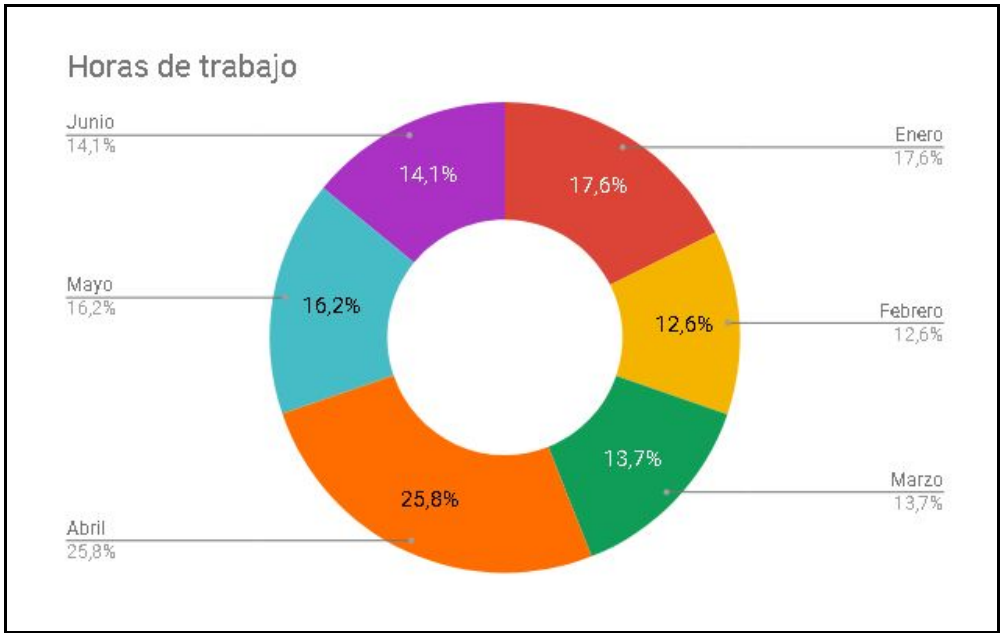

Figura 2.6: Diagrama de Tarta. Representación gráfica horas de trabajo

En el diagrama superior (Figura 2.5) se observa un diagrama de tarta que representa las horas de trabajo que realiza un vendedor (la imagen es ilustrativa, no real). Al primer vistazo, se puede entender claramente que del total de horas trabajadas, no han sido las mismas en cada mes, viendo que por ejemplo en Abril han sido casi el doble que en Junio.

Lo mismo sucede con el siguiente diagrama (figura 2.6). Esta gráfica representa las horas de trabajo autónomo de un estudiante durante un semestre (la imagen es ilustrativa). Cualquier persona sin tener un conocimiento previo sobre la materia, es capaz de interpretar con bastante exactitud la información que refleja el gráfico, viendo que la línea azul representa la evolución de las horas mensuales y la línea roja es la acumulación de las mismas.

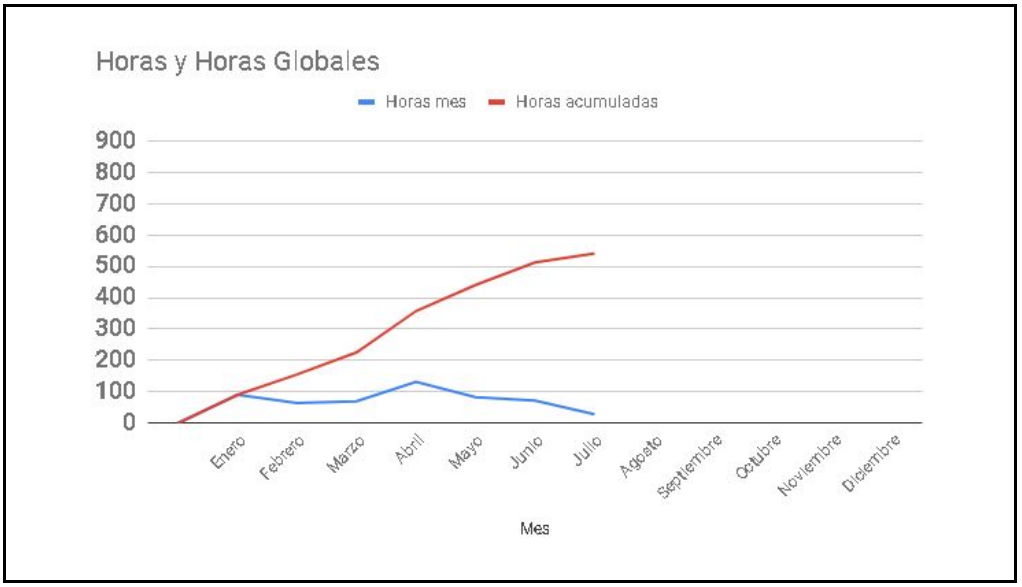

Figura 2.7: Histograma. Representación gráfica de las horas de estudio.

2.5 Tipos de visualización de datos

En la mayoría de casos, la visualización se centra exclusivamente en el uso de gráficas, pero es un entorno mucho más amplio.

En primer lugar se encuentran los elementos más básicos de la representación de datos:

- Los gráficos: entre ellos se encuentran los de barras, histogramas, puntos, tarta...

- Mapas: no se refiere a mapa geográfico, sino mapas conceptuales o de burbujas.

- Tablas de contenidos: una de las formas más sencillas de representar datos es el uso de tablas, pero dificulta su interpretación.

En segundo lugar aparecen los cuadros de mando. Estos son un conjunto de representaciones básicas, que al mostrarse a la vez, dotan de un mayor sentido al resto de datos. Se usan mucho en el ámbito empresarial para por ejemplo informar en tiempo real del estado de los servidores de una empresa.

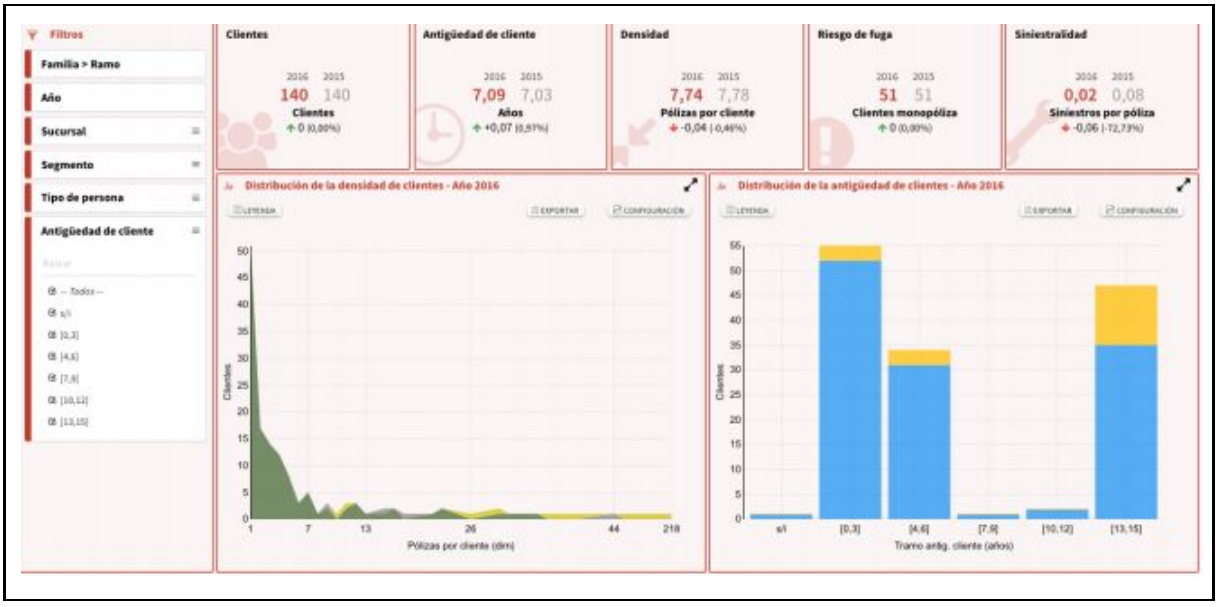

Figura 2.8: Ejemplo Cuadro de Mandos
Fuente: Informe herramientas. Gobierno de España [18]

En tercer y último lugar aparecen las infografías. Esta es muy similar al cuadro de mandos en el sentido de que también es una composición de visualizaciones, pero lo que es diferente es su objetivo. Las infografías tienen como objetivo contar una historia mediante la visualización de los gráficos. Un claro

ejemplo de esto, lo podemos ver en la Figura 2.4 .

2.6 Aplicaciones similares a nivel nacional

El estado del arte de este tipo de aplicaciones es un poco reducido. Esto se debe a que hay aplicaciones como las bancarias, que implementan sistemas similares para proveer de información bursátil a sus clientes a través de sus propias aplicaciones móviles, con la diferencia de que no lo hacen mediante Open Data, sino mediante información en tiempo real que les llega desde Madrid.

Por otro lado, se han podido encontrar algunas aplicaciones similares desarrolladas por gobiernos autonómicos para cosas puntuales. Por ejemplo, recientemente el gobierno de Aragón [20][21] mantuvo en activo (ya no) una aplicación para representar los casos de COVID-19 de la comunidad autónoma, en la que actualizaban las bases de datos de forma periódica para mantener informada a la sociedad. Este es un claro ejemplo de posibles usos que se le pueden dar a este tipo de aplicaciones. Si bien es cierto, este tipo de aplicaciones, pueden ser más útiles, juntando más funcionalidades en una misma app, como por ejemplo, la realización de trámites administrativos mediante firma digital con un ayuntamiento o con la comunidad autónoma.

2.7 Dart

En los anteriores apartados se ha introducido al proyecto, hablando de los antecedentes, explicando el Open Data, dando la importancia que requiere a los diagramas y poniendo en situación respecto al estado del arte de este tipo de aplicaciones. Ahora se va a pasar a explicar las características de esta aplicación como son el lenguaje de programación y las herramientas que se han utilizado para su desarrollo.

La aplicación que se ha desarrollado para este proyecto, se ha realizado con el lenguaje de programación Dart [22]. Este, es un lenguaje de código abierto desarrollado por la empresa tecnológica Google. Originalmente llamado “Dash”, salió al mercado con el nombre definitivo en el año 2011.

Su principal objetivo es convertirse en el lenguaje por excelencia para el desarrollo de aplicaciones móviles, desbancando así a JavaScript. Para ello, está totalmente optimizado para el desarrollo de UI (User Interface - Interfaz de Usuario). Una de sus mayores ventajas es la portabilidad entre sistemas operativos. Una aplicación desarrollada con Dart, es compatible en su totalidad con sistemas operativos IOS y Android, sin la necesidad de realizar modificaciones en el código, por lo que a nivel económico es una gran ventaja ya que las horas de programación son la mitad que con otros lenguajes. Otra de sus ventajas es la recarga en caliente. Esto es, que se pueden ir viendo los cambios en la aplicación final al tiempo que se va programando. Cada vez que se guarda el fichero, la aplicación se relanza muy rápidamente y sin consumir una

excesiva carga de recursos, de forma que podemos ver al momento el resultado e ir ajustando los detalles según se va desarrollando.

Otro de los factores que hacen que Dart sea una buena opción para el desarrollo de aplicaciones en el futuro, es su fácil aprendizaje debido a sus grandes similitudes estructurales con otros lenguajes, por lo que si se conoce JavaScript, no resultará difícil adaptarse a este nuevo lenguaje. Al estar centrado en el desarrollo de UI, tiene muchas librerías y paquetes para realizar una interfaz gráfica estética, en la cual sea fácil navegar entre los distintos menús.

2.8 Herramientas

En el desarrollo de este proyecto, han sido numerosas las herramientas de las que se ha hecho uso, unas en más medida que otras.

Para el hecho de programar en sí, se ha utilizado el IDE de Android Studio [23] ya que es el entorno oficial para el desarrollo de aplicaciones basadas en Android. En un principio se valoró la posibilidad de usar Eclipse para esta tarea, pero debido a diferentes factores, nos hemos decantado por el uso de Android Studio.

Para poder visualizar el resultado de la aplicación sin la necesidad de conectar un dispositivo móvil, se ha hecho uso de Android Studio Emulator [24], ya que nos ofrece la posibilidad de como el propio nombre indica, emular un dispositivo móvil donde se va a ejecutar la aplicación. Cabe resaltar que el uso de Android Studio, junto con el emulador consume una cantidad de recursos considerable, hecho a tener en cuenta.

Este IDE no se ha utilizado en solitario, sino que se le han añadido SDK para este proyecto. Ya que se va a realizar una aplicación en la plataforma oficial para android (sistema operativo de Google), hemos hecho uso de Flutter [25]. Flutter es un kit de herramientas de interfaz de usuario, desarrollado por Google y que usa el lenguaje de programación Dart. Está diseñado enfocándose en los widgets de la interfaz que funcionan a modo de estructuras simples, como pueden ser slices, textos o imágenes, de los que se pueden hacer widgets más complejos combinando varios simples.

Las principales ventajas de usar Flutter es que nos ofrece una UI muy estética con multitud de elementos, además de su alto rendimiento gráfico, teniendo en cuenta que las aplicaciones desarrolladas con flutter se ejecutan a 60 fps (fotogramas por segundo) lo que genera una sensación de fluidez muy alta.

La otra herramienta que se ha utilizado para el desarrollo del proyecto es github [26]. Github es una plataforma online de desarrollo colaborativo en la que se alojan los proyectos para llevar a cabo un sistema de control de versiones. Esto se ha hecho mediante el uso de la versión de escritorio Github Desktop.

Capítulo 3 Metodología

El primer paso en la realización de este proyecto, fue el establecimiento de los objetivos conjuntamente con la tutora para así poder luego planificar una hoja de ruta y poder cumplir los plazos que eran necesarios.

La metodología usada para el desarrollo de este proyecto ha sido la metodología en **cascada**. Debido a ello, se ha realizado una gran tarea de preparación y organización para establecer desde un principio las fases del proyecto y su secuencia.

En esta metodología, cobra especial importancia la búsqueda de información. Esto se debe a que si en determinada fase del proyecto, falta información para poder realizarlo de manera correcta, el proyecto en su conjunto sufrirá retrasos. Una vez se ha investigado a fondo todo lo relevante del proyecto, se puede dar comienzo.

Fases	**Descripción**
1ª Fase: Establecimiento de objetivos	Intercambio de correos electrónicos con la tutora para establecer los objetivos del proyecto de forma mútua
2ª Fase: Estudio previo	En esta fase, se ha busca información relativa al proyecto, así como a las herramientas que se van a usar.
3ª Fase: Planificación	Se establece una hoja de ruta y se secuencian las fases del proyecto
4ª Fase: Implementación	Se desarrolla el proyecto siguiendo la planificación previa
5ª Fase: Evaluación	Se prueba la aplicación, se evalúa y se transmiten los problemas o sugerencias encontrados
6ª Fase: Corrección	Se revisan las evaluaciones y se procede a las correcciones

Tabla 3.1: Fases del proyecto

Capítulo 4 Evaluación

Para poner a prueba el funcionamiento de la aplicación, se ha usado a dos personas para que la usen y así ver posibles fallos o aspectos a mejorar.

La evaluación se ha llevado a cabo siguiendo exclusivamente el criterio del feedback. No se ha indicado puntos en los que deban fijarse, sino que se han podido mostrar cualquier opinión que les haya surgido al utilizar la aplicación.

Tarea	Fecha de evaluación	Aspectos señalados
Evaluación 1ª Persona	28 / 07 / 2020	- Imágenes no cargan correctamente siempre - Leyenda del gráfico muy pequeña - Falta de información relativa al proyecto
Evaluación 2ª Persona	15 / 08 / 2020	- Texto del menú demasiado pequeño

Tabla 4.1: Evaluación del funcionamiento

Una vez realizada la evaluación de la primera persona, se ha procedido a corregir dichos errores o sugerencias antes de pasar a la segunda evaluación. A su vez, después de la segunda evaluación, se ha procedido a corregir el aspecto que debía ser cambiado.

La mayoría de los comentarios recibidos por parte de las personas encargadas de la evaluación, vienen derivados de la usabilidad de la aplicación. Este es uno de los objetivos que se estableció en un principio, con lo cual, resulta un feedback muy positivo para la mejora de la aplicación en estos aspectos.

Capítulo 5 Desarrollo del Proyecto

En este capítulo se va proceder a la explicación de las diferentes fases del desarrollo del proyecto. Estas se dividen por orden cronológico en: Fase Inicial, la Fase de Diseño, la Fase de Desarrollo y por último la Fase de Pruebas

5.1 Fase inicial

En la fase inicial del proyecto, nos hemos centrado en la investigación sobre la materia. Lo primero que se ha hecho ha sido una recopilación de información sobre los diagramas y gráficas existentes ya que se le quiere dar la importancia que tienen en esta aplicación. Dicha importancia viene de que la aplicación gira entorno a poder representar datos en tiempo real mediante gráficas, pero no vale con usar cualquiera.

En un principio, se había propuesto el uso de diagramas modernos, capaces de representar mucha información, pero según transcurría la investigación, se ha determinado que es mejor usar gráficos convencionales. El motivo de esto, no es otro que la accesibilidad a la información.

El principal objetivo de esta aplicación es que cualquier persona pueda usarla y así poder obtener la información que busca, pero para ello, los gráficos deben ser entendibles por gran parte del público. Si bien es cierto que las generaciones jóvenes, son más capaces de adaptarse a los nuevos gráficos o diagramas, las generaciones mayores, no tienen ya esa facilidad de adaptación. Puesto que ese sector del público está más acostumbrado a los gráficos convencionales, se ha optado por el uso de estos.

Una vez realizada la investigación pertinente respecto a los tipos de gráficos y diagramas, se ha pasado a la documentación sobre el lenguaje de programación. En este caso, al tratarse de un lenguaje nuevo, se desconoce la filosofía y el estilo de programación del mismo, por lo que se ha tenido que proceder a la recopilación de información respecto al mismo. Una de las ventajas y/o facilidades que se ha encontrado es su similitud con otros lenguajes de programación como JavaScript, con lo que ha sido más sencillo encontrar información y tutoriales sobre su estilo de programación.

5.2 Diseño

Desde un principio en este proyecto, este es uno de los apartados que más claros ha estado. Esto se debe a que recientemente he podido cursar la asignatura de Diseño Arquitectónico y Patrones. Al haber tenido esta materia (que considero que todos los alumnos deberían cursar), se ha obtenido otra visión sobre la forma de programar y se han interiorizado las ventajas que tiene el uso de patrones para el desarrollo de un proyecto.

Para el diseño de la aplicación, se ha desarrollado una aplicación similar pero más básica en Java, para poder ver cómo funcionan los patrones que a continuación se van a explicar. En el caso de este proyecto, hay un patrón de diseño que cumple la función que buscamos con esta aplicación, en su totalidad. Este es el patrón MVC, también conocido como Patrón Modelo Vista Controlador.

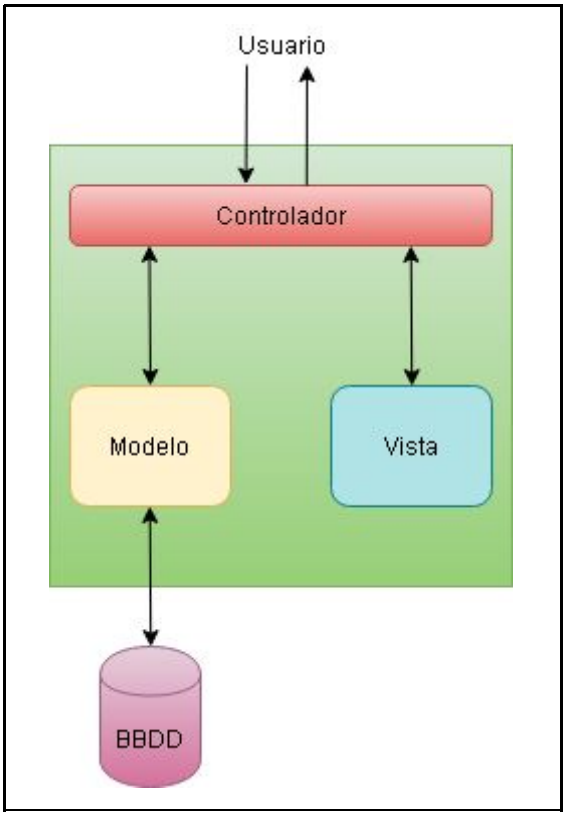

Figura 5.1: Diagrama Patrón MVC (Modelo-Vista-Controlador)

Este patrón busca la separación en tres partes bien diferenciadas del modelo, la vista y el controlador ya que cada uno tiene unas tareas específicas (Figura 5.1).

El modelo es el encargado de obtener los datos de la fuente correspondiente, ya sea una base de datos, una web, un fichero...y parsear los datos. Esto consiste en recopilar y almacenar en un objeto iterable, únicamente los datos que necesitamos,

desechando el resto de ellos.[

La vista es la encargada de una vez recibido el objeto sobre el que se itera, representar los datos que se han obtenido mediante una gráfica o diagrama. Este diagrama será indicado por una fuente externa.

El controlador es el encargado de manejar la interacción entre el modelo y la vista. En primer lugar, le indica al modelo qué datos son los que se requieren para que este los obtenga de la base de datos. A continuación, recibe dichos datos mediante un objeto y se los pasa a la vista indicando qué gráfico se tiene que utilizar en cada caso. Como medio externo está el usuario el cual interactúa con el controlador a través de la interfaz de usuario indicando las diferentes opciones.

Una vez explicado el patrón MVC, que es el usado para diseñar la estructura general de clases del programa, hay otro patrón que destaca en menor medida en dicho diseño. Este es el patrón Estrategia. Este patrón permite clasificar las clases en diferentes tipos, para así abstraer las funciones comunes y evitar duplicidad de código, mientras que las funciones propias e individuales de cada tipo, las tienen las clases hija. Esto se puede ver más claro en la siguiente figura (Figura 5.2):

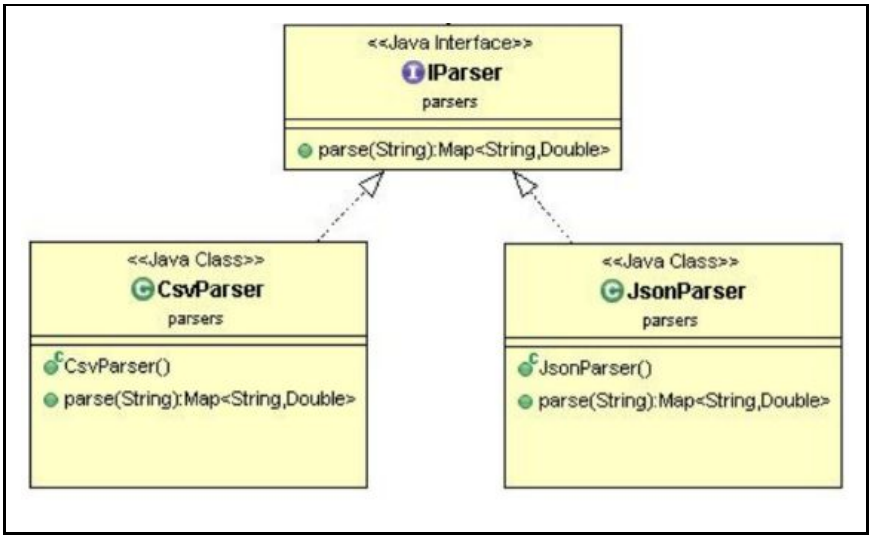

Figura 5.2: Diagrama UML del Patrón Estrategia

En nuestro caso, el patrón MVC va a ser usado una única vez, para desarrollar la estructura general del proyecto. Por otro lado, finalmente el patrón Estrategia no va ser necesario debido al funcionamiento del lenguaje Dart en el cual, la forma de parsear los datos y de representarlos, requiere de mucha menos programación que en Java.

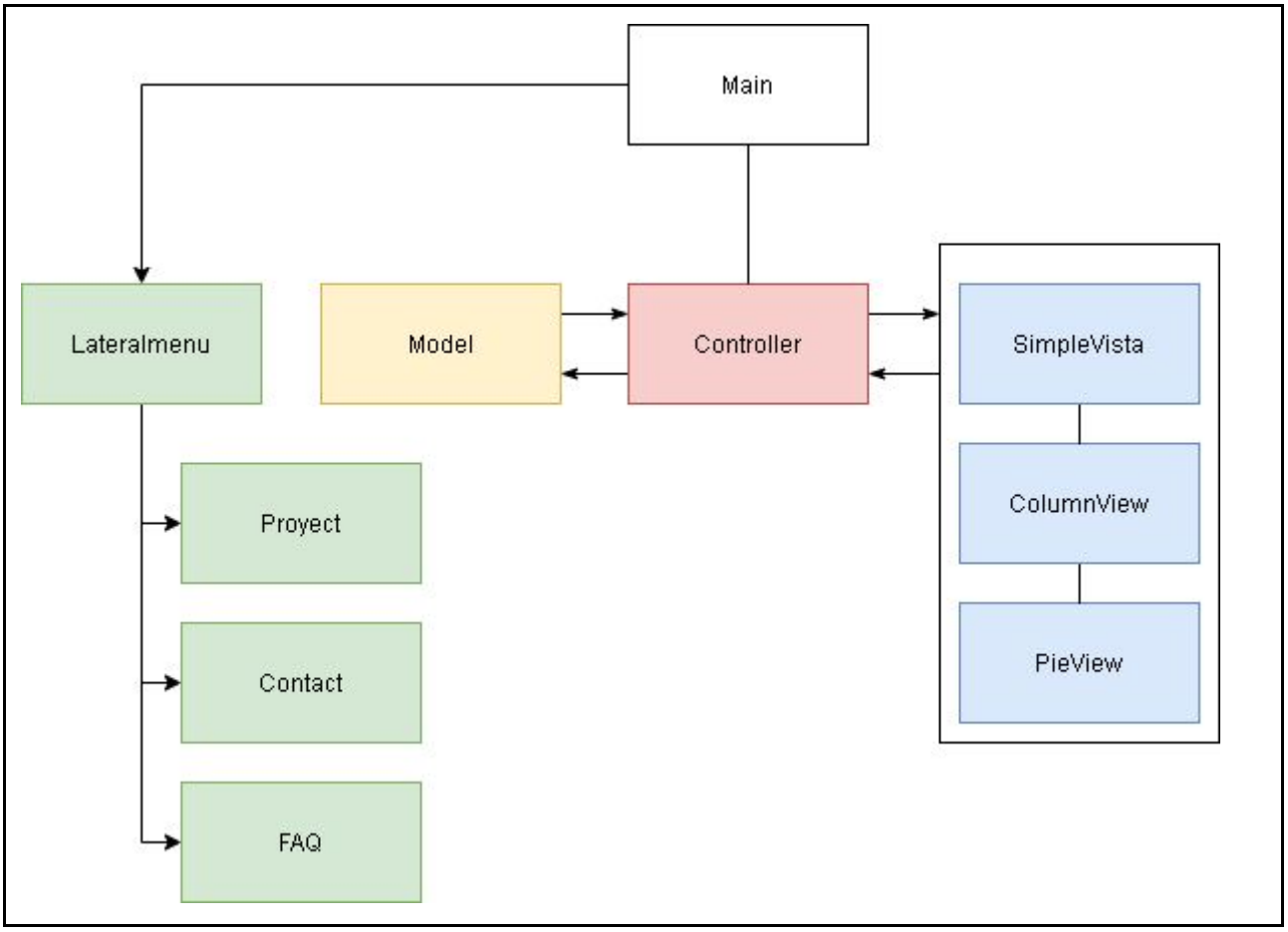

Figura 5.3: Estructura Final del Proyecto

En la Figura 5.3 se puede observar la estructura que tiene el proyecto una vez finalizado. Esta estructura se divide en dos partes. Una primera en la que existen diferentes ventanas informativas respecto del proyecto (fondo verde) y el resto que es la parte de programación de la aplicación.

En la parte de programación, se encuentra el patrón MVC explicado anteriormente en el cual se sitúa el controlador en medio haciendo de intermediario entre el modelo que se va a usar y las distintas vistas de las que dispone el proyecto.

5.3 Desarrollo

Para el desarrollo de la aplicación, se ha iniciado por la creación de una estructura base a partir de la cual se irán introduciendo elementos. En Android Studio, al crear un proyecto flutter, se genera automáticamente una aplicación de ejemplo (Figura 5.4)

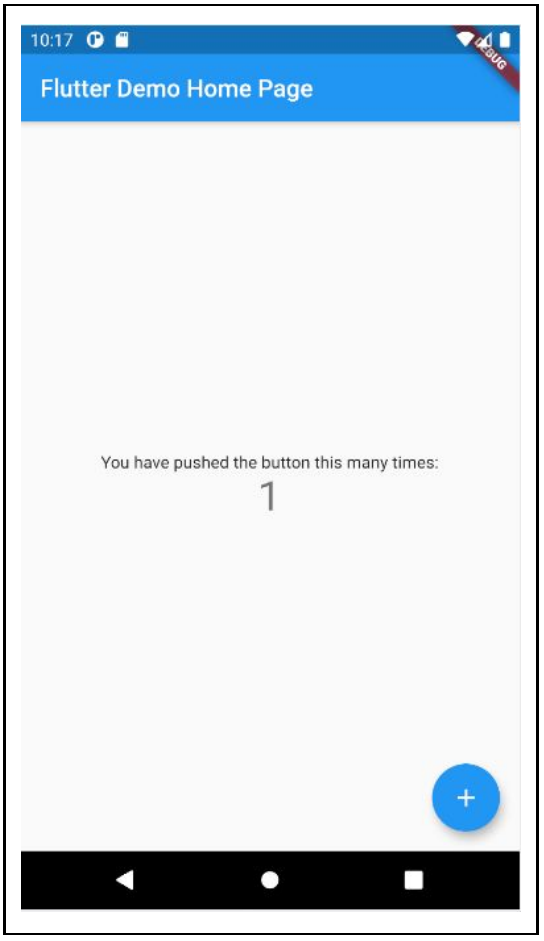

Figura 5.4: Aplicación de demostración

En esta aplicación inicial, lo únicos elementos que aparecen es un mensaje de texto, un pequeño contador y un botón inferior. Cada vez que se presiona dicho botón, el contador aumenta. En el código de esta demo, aparecen comentados con bastante claridad, lo que hace cada elemento declarado y para qué sirve cada sección.

Partiendo de esta base, y teniendo en cuenta que Dart es un lenguaje de programación diseñado para centrarse en la parte gráfica de la aplicación, se ha iniciado con la integración de un menú lateral o Lateralmenu similar al que se puede ver en la web del campus virtual de la universidad de la laguna. En dicho menú, se crearán las distintas ventanas de las que se compone el proyecto para informar sobre el funcionamiento de la aplicación, el objetivo del proyecto, preguntas frecuentes… y se añadirán elementos de estilo para ir comenzando con el diseño final de la aplicación (Figura 5.5)

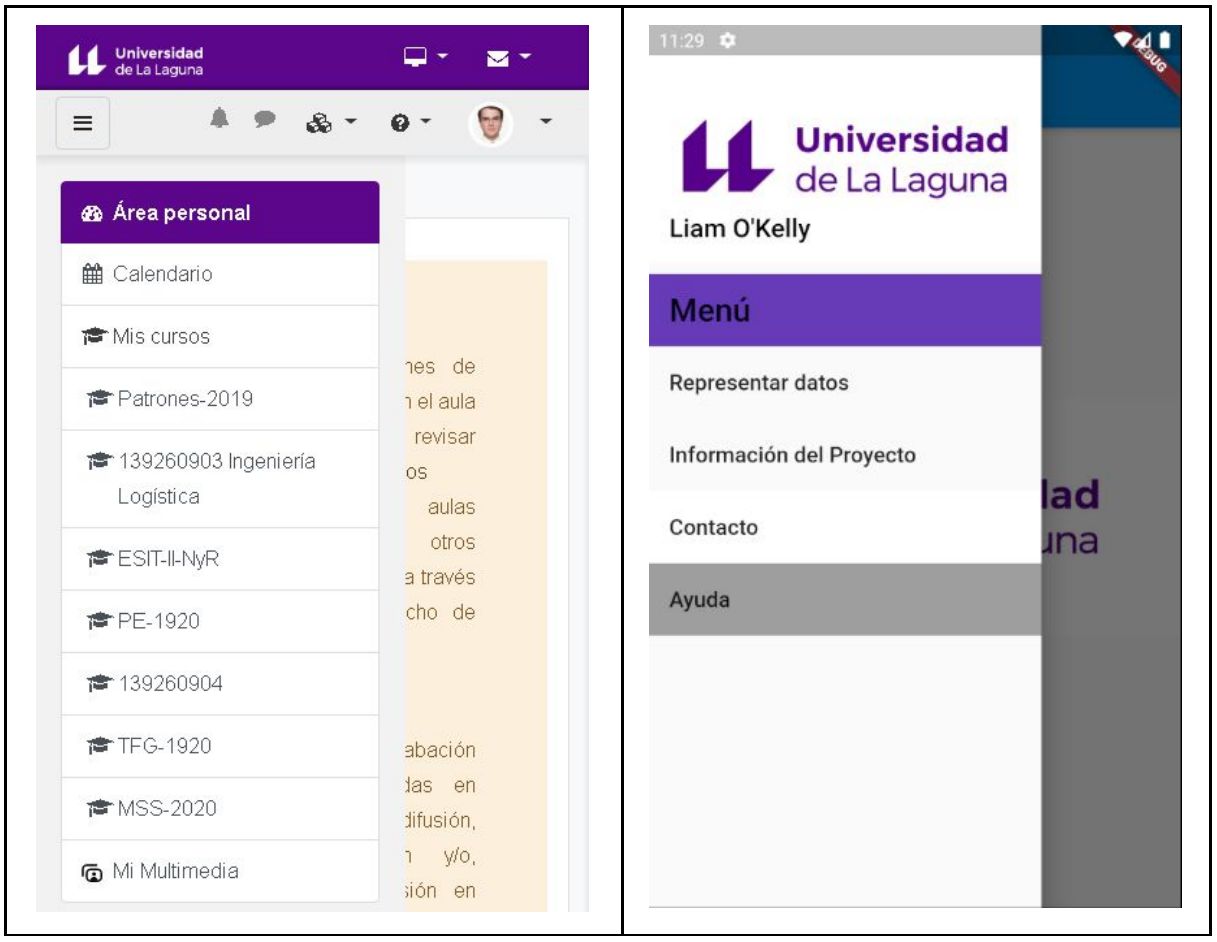

Figura 5.5: Comparativa menú lateral campus virtual respecto a LiveDiagrams

Fuente: Campus Virtual de la Universidad de La Laguna

En la figura anterior (Figura 5.5) se observa a la izquierda una imagen del campus virtual de la universidad de la laguna a modo ilustrativo de un menú lateral. Al lado derecho se muestra el menú lateral de la aplicación, el cual se ha intentado asemejar al del campus para mantener un estilo gráfico similar al de la universidad.

Una vez realizado el menú, se ha empezado por las páginas más informativas de la aplicación. Se ha realizado una ventana *Ayuda*, con las preguntas frecuentes de los usuarios, una ventana *Contacto* con los datos del alumno y una ventana *Información del Proyecto*, en la que se ha recopilado la información más relevante de este Trabajo Fin de Grado, de una forma sencilla y fácil de leer para que no sea muy pesado para el usuario.

Cada una de ellas, se ha realizado siguiendo la hoja de estilos que se ha establecido para mantener una estructura y orden similar en todas las páginas, creando así la sensación de uniformidad en todo su contenido. Para algunas de ellas, debido a la longitud de su información, que no cabía en una sola ventana, se le han añadido los widget SlideBar para poder desplazarnos por la página

Finalmente cada una de las páginas, se ha revisado y se ha linkeado con el menú lateral para que sean alcanzables desde el inicio de la aplicación y se les ha añadido un botón superior para volver a la ventana inicial.

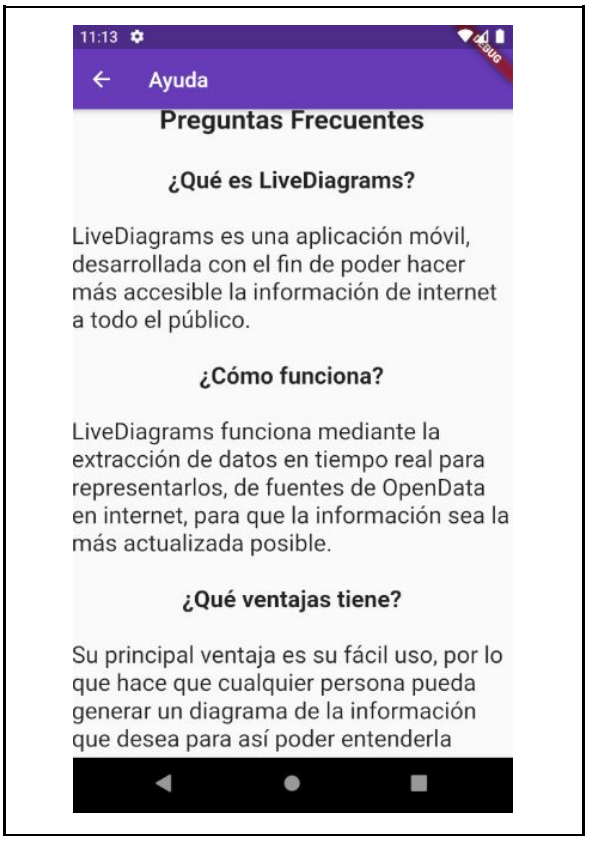

Figura 5.6: Ventana Página Ayuda

Figura 5.7: Ventana Página Contacto

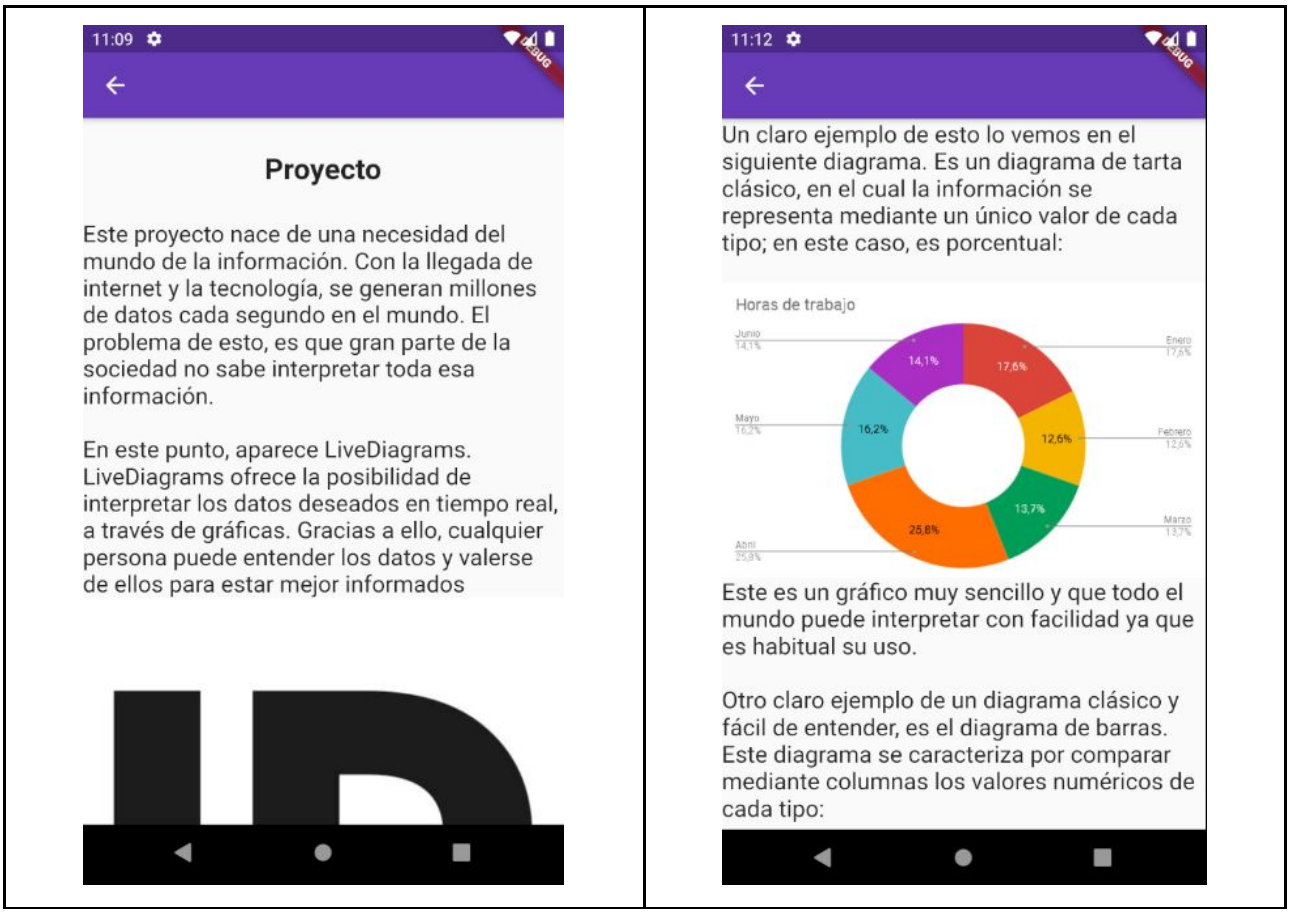

Figura 5.8: Ventana Página Información del Proyecto

A continuación, se ha creado una página adicional en el menú lateral llamada “Representar datos” en la cual luego se mostrarán los distintos gráficos. Ahora sí, está terminada la estructura de la interfaz de la aplicación y se pasará al desarrollo del backend de la aplicación.

El funcionamiento explicado a alto nivel sería el siguiente:

Al entrar a la ventana de “Representación de datos” hay un menú superior para tener los 3 tipos de representación (diagrama de tarta, modo de texto, diagrama de barras) (Figura 5.9). Justo al momento de entrar en cualquiera de las representaciones, la aplicación le dice a la clase “Model” que descargue los datos de internet y los almacene porque van a hacer falta. Una vez los tiene listos, se los envía a través de la clase “controller” a la clase “vista” encargada de generar el gráfico. El controlador sabe qué tipo de gráfica debe generar sin que se le indique nada, por el hecho de que cada gráfica está separada en ventanas diferentes. Con lo cual una vez que nos movemos de ventana, el programa comprueba a cuál se ha movido el usuario para saber cuál debe crear.

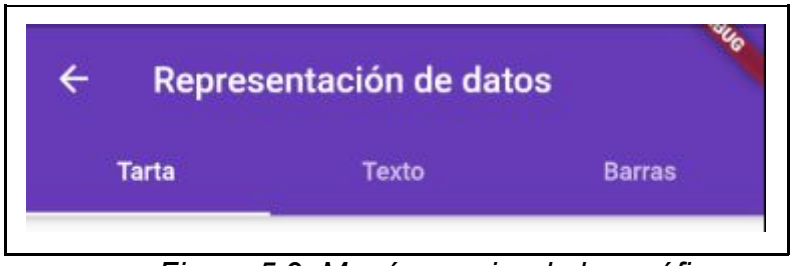

Figura 5.9: Menú superior de las gráficas

Ahora pasando a explicar con mayor detalle el funcionamiento, se empezará por explicar el funcionamiento del Modelo. Como se puede ver en la figura 5.3 , el modelo está conectado solamente con el controlador. Esto es debido a que el controlador es el cerebro de la aplicación y le tiene que indicar al resto partes del programa (Vista y Modelo) lo que tienen que hacer y cómo lo tienen que hacer. Una vez el modelo esté cargado, el controlador es de nuevo, el único intermediario con la clase Vista, a la cual le indicará qué vista debe generar.

Cuando se accede a la ventana de “Representación de datos” que se puede observar en la figura 5.9 , el controlador detecta que se quiere representar una gráfica, por lo que procede a Construir una instancia de Modelo. A continuación, se hace una llamada al método getData del modelo para obtener los datos. Esta llamada es muy importante porque se hace mediante el uso de programación asíncrona usando la clase Future. Esto se debe a que los datos tardan en descargarse y hay que esperar a que estén listos antes de llamar al constructor de la gráfica.

El siguiente paso, es comprobar que los datos se han almacenado correctamente y en ese caso, se procede a crear una instancia de la clase Vista a la cual se le pasan los datos por parámetro. Los tres tipos de vista que se aprecian en la figura 5.9, están estructurados como una lista de contenedores. La acción de movernos entre contenedores, es detectada por el controlador, que lo utiliza para saber cuál de las vistas es la que debe construir en cada caso.

En la clase Modelo hay dos partes diferenciadas:

- En una de ellas se define una estructura de datos, que será la que posteriormente se use para generar la lista de celdas en la que se almacenará la información.

- En la otra parte, se definen los métodos:

    - getData: es el encargado de hacer la llamada a la url del portal de opendata y comprobar si se pudo leer los datos del fichero. Este método tiene un async ya que tiene que esperar a que se resuelva toda la ejecución.

      Para la obtención de datos de internet, se ha hecho uso del paquete “Http” [27] de flutter, que nos dota de las funciones necesarias. Para ello, se ha añadido a las dependencias del fichero pubspec.yml donde se sitúan todas las demás. A continuación, se hace la petición web a la url indicada para obtener los datos.

    - parseResponse: en caso de que se pudieran leer correctamente los datos, este método es invocado desde el getData y tiene como objetivo mapear el json de la url en un map de Strings y devolverlo. De esta forma, sólo almacenamos los datos que se requieren y no el fichero completo, ahorrando así memoria y simplificando la forma de iterar sobre él mismo.

Una vez ya se tienen los datos parseados, el controlador llama al constructor de la gráfica pasandole la lista como argumento para que lo construya. Como se ha mencionado antes, en función de en qué ventana se esté, se llama a un constructor o a otro. Las clases de las gráficas solo tienen un constructor en el que se definen algunos parámetros como la posición de la gráfica, la leyenda, el color…

Barrio	Población
Igueste	12208
Toscal	17682
Almáciga	22371
Taborno	31394
La Salle	38347
Los Campitos	42029

Tabla 5.1: Ejemplo Formato de datos una vez realizado el parseo

Una vez se ha probado el correcto funcionamiento de la aplicación tanto a la hora de descargar datos desde portales de opendata, como a la hora de representar mediante gráficas dichos datos, se ha procedido a la prueba usando dataset de ejemplos, de generar distintas gráficas para visualizar algunas alternativas visuales que ofrece flutter:

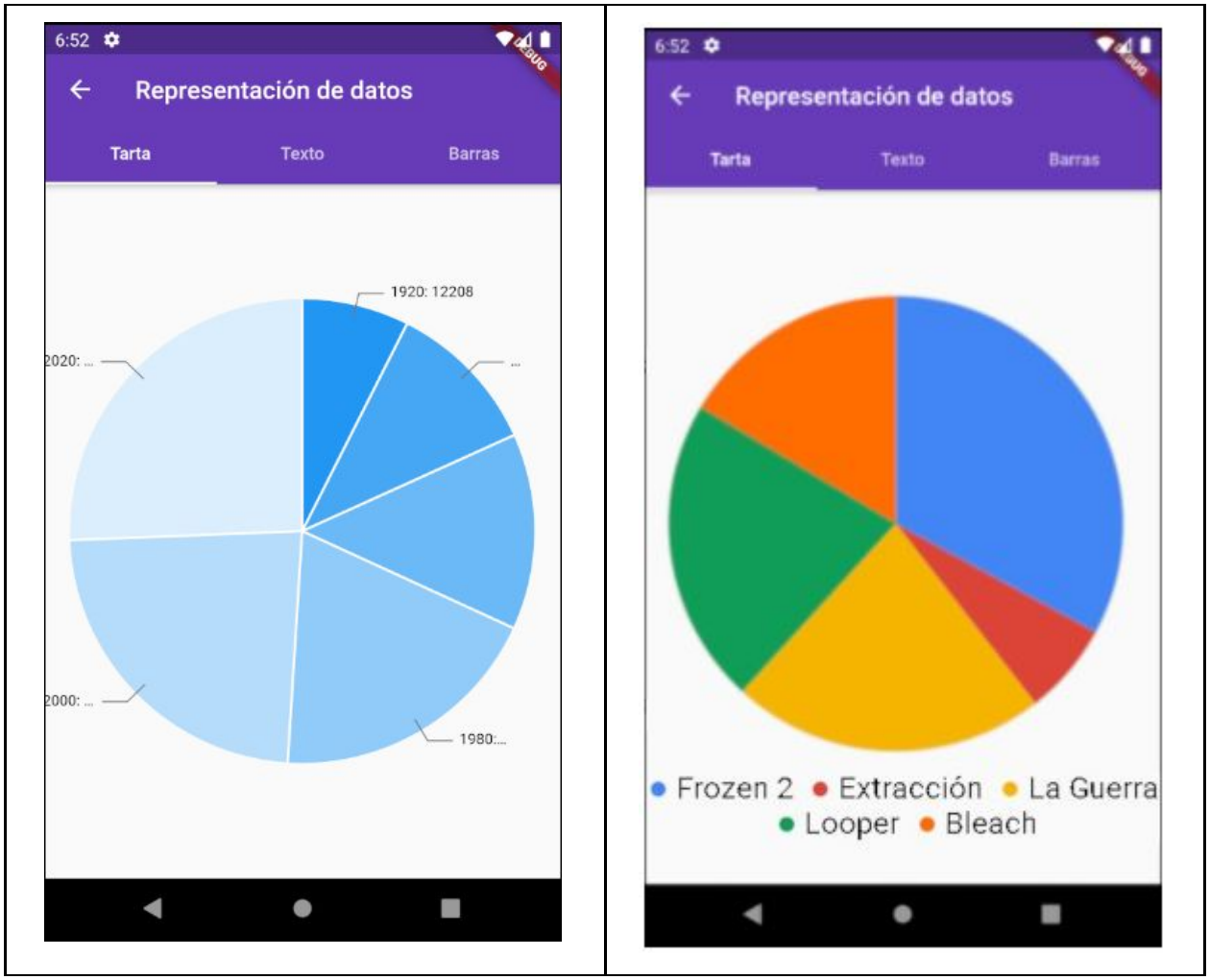

Figura 5.10: Representación diagramas de tarta

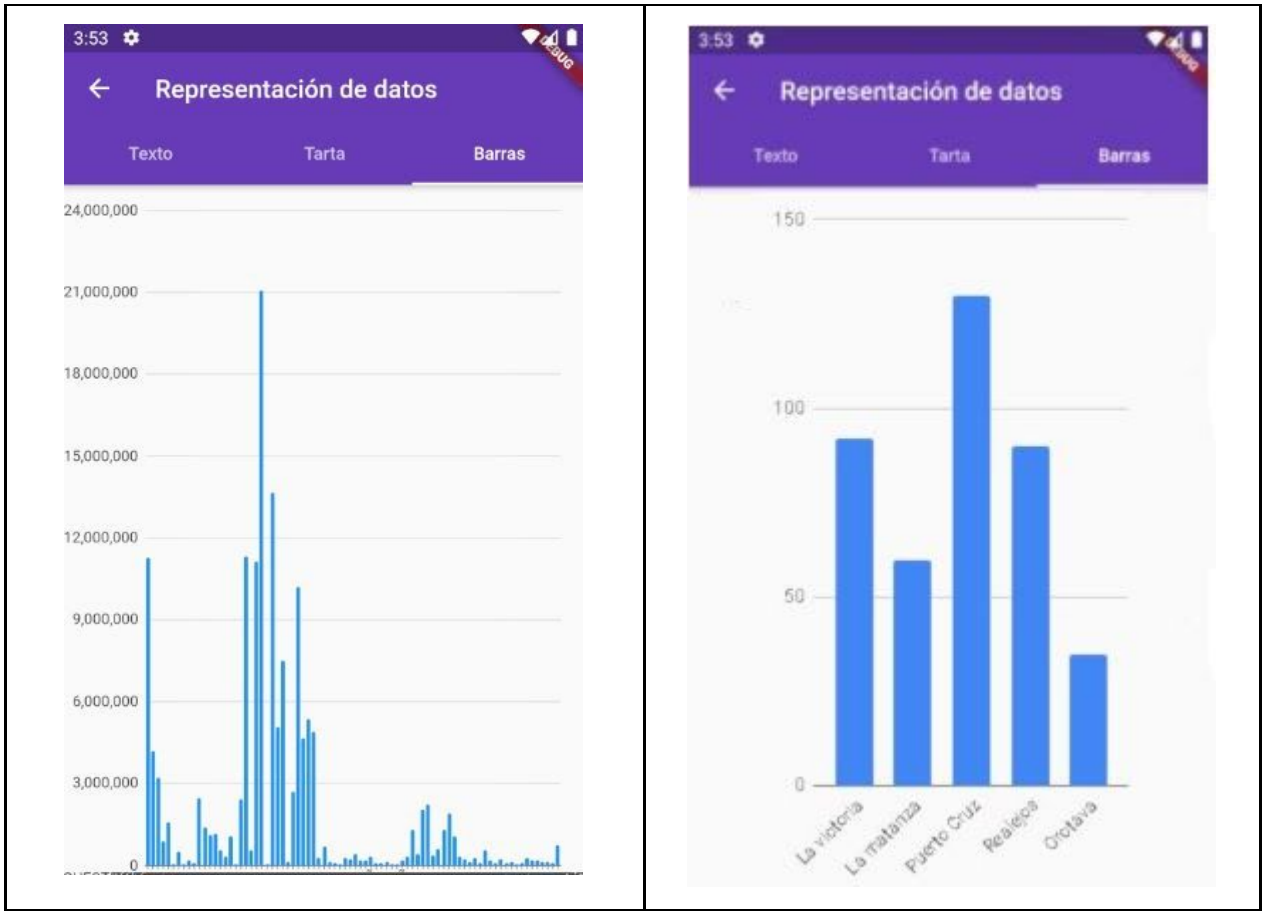

Figura 5.11: Representación diagramas de Barras

Figura 5.12: Representación Modo Texto

Capítulo 6 Conclusiones y líneas futuras

6.1 Conclusiones

A nivel general se consideran cumplidos los objetivos iniciales. Se ha conseguido crear una aplicación móvil tanto para Android, como para IOS, la cual descarga los datos en tiempo real desde el portal de opendata del Cabildo de tenerife.

Uno de los objetivos que surgió mientras se planteaba el proyecto, era la usabilidad de la aplicación. Se ha buscado una gran simplicidad a nivel de menús de usuario y de uso de la app, para facilitar su uso. Además, se han puesto los textos de un mayor tamaño para la gente con problemas de visión. Las gráficas que se han usado, han cumplido su objetivo de ser entendibles para gente de diferentes edades.

A nivel técnico, se han encontrado algunas dificultades en algunas partes como la extracción de datos. Esto se debe a que Flutter busca crear apps muy fluidas y de respuesta rápida al usuario. Esto provocaba que la ventana de la gráfica se cargase antes de siquiera haber obtenido los datos y haberlos parseado.

Pero la mayor dificultad encontrada en este proyecto ha sido respecto a flutter y dart. La gran cantidad de versiones existentes en un corto periodo de tiempo hace muy difícil realizar algunas implementaciones ya que las librerías han cambiado mucho entre algunas versiones, eliminando así métodos de la aplicación o modificando su comportamiento de una forma con la que no se podía trabajar en conjunto con el resto de métodos que se habían creado. Además, la documentación encontrada en internet en muchos casos se contradecía porque ésta cambia con mucha frecuencia.

6.2 Líneas Futuras

Poniendo la vista en posibles líneas futuras para esta aplicación, hay una idea que es potencialmente exitosa. En la situación actual, con el mundo pendiente de una pandemia con una cifra de contagios que evoluciona constantemente, existen algunas aplicaciones que representan los datos diarios del coronavirus [28].

La aplicación LiveDiagrams no es más que una plataforma base, la cual se puede modificar en función del cliente para representar otros datos. Consideramos interesante una versión de la aplicación quizás a nivel autonómico o en el caso de Canarias, insular, en la cual exista una lista desplegable con todos los municipios que existen en la isla y una vez se elija el deseado, podamos ver un histograma de la evolución de los casos en el municipio.

Además, se le pueden añadir ventanas a la aplicación en las cuales se indiquen medidas de precaución, como lavarse las manos, la distancia de seguridad, el uso de mascarilla, etc.

De esta forma, se tendría una aplicación complementaria a la actual del Cabildo de la Gomera [29]. En el caso de LiveDiagrams, debido a su sencillez y accesibilidad, mayor cantidad de gente podría entender las cifras de la pandemia y la evolución de la misma.

Capítulo 7 Summary and Conclusions

7.1 Conclusions

At a general level, the initial objectives are considered fulfilled. It has been possible to create a mobile application for both Android and IOS, which downloads data in real time from the opendata portal of the Tenerife council.

One of the objectives that appeared while the project was being planned was the usability of the application. Great simplicity has been sought in the menus and in the use of the app. In addition, the texts of a larger size have been placed for people with vision problems. The graphs that have been used have fulfilled their objective of being understandable to people of different ages.

On a technical level, some difficulties have been encountered in some parts such as data extraction. This is because Flutter seeks to create very fluid apps with fast response to the user. This caused the graph window to load before the data was even obtained and parsed.

But the biggest difficulty encountered in this project has been regarding flutter and dart. The large number of existing versions in a short period of time makes it very difficult to carry out some implementations since the libraries have changed a lot between some versions, thus eliminating application methods or modifying their behavior in a way that could not be worked with in set with the rest of the methods that had been created. Furthermore, the documentation found on the internet in many cases contradicted itself because it changes very frequently.

7.2 Futures Lines

Looking at possible future lines for this application, there is an idea that is potentially successful. In the current situation, with the world pending a pandemic with a constantly evolving number of infections, there are some applications that represent the daily data of the coronavirus.

The LiveDiagrams application is nothing more than a base platform, which can be modified depending on the client to represent other data. We consider a version of the application interesting, perhaps at the regional level or in the case of the Canary Islands, insular, in which there is a drop-down list with all the municipalities that exist on the island and once the desired one is chosen, we can see a histogram of the evolution of cases in the municipality.

In addition, windows can be added to the application in which precautionary measures are indicated, such as washing hands, the safety distance, the use of a mask, etc.

In this way, this would be a complementary application to the current one of the Cabildo of La Gomera. In the case of LiveDiagrams, due to its simplicity and accessibility, more people could understand the numbers of the pandemic and the evolution of it.

Capítulo 8 Presupuesto

El presupuesto para el desarrollo de este proyecto viene establecido por un único factor y es las horas de desarrollo. Para el cálculo, se ha establecido un precio de 20€ la hora para el cliente, pero de ahí sale el sueldo de 19€ la hora, dejando un 5% del precio/hora como margen por si surge algún inconveniente o problema durante el desarrollo del proyecto. En total, se calcula que han sido unas 180 horas de trabajo.

8.1 Desglose

Tipos	Descripción	Horas	Precio
Documentación	Estudio de los distintos diagramas, gráficos y lenguaje de programación	35	700
Estado del arte	Estudio del estado del sector y de aplicaciones similares	25	500
Diseño	Diseño de la aplicación, tanto de su estructura, como de su estilo	30	600
Implementación	Desarrollo del programa de la aplicación	90	1800
Total		**180 h**	**3600 €**

Tabla 7.1: Desglose presupuesto

Capítulo 9 Anexo Primero

9.1 Estructura de Celdas

```
*******************************************************************************
* Fichero: Model.dart
*******************************************************************************

class Data{
        //Attributes
        final double area;
        final String neighborhood;

        //builder
        Data({
                @required this.area,
                @required this.neighborhood,
        });

        factory Data.fromJson(Map<String, dynamic> json) {
                return Data(
                        area: json['AREA'] as double,
                        neighborhood: json['BARRIO'] as String,
                );
        }

        @override
        String toString(){
                return"Area: ${area}\n Barrio: ${neighborhood}";
        }
}
```

Esta es la estructura para cada celda, de la cual luego se hace un List para almacenar los datos provenientes del fichero Json que se ha descargado desde el portal.

9.2 Petición de Servidor

```
*******************************************************************************
* Fichero: Model.dart
*******************************************************************************

Future<List<Data>> getData() async {
        Response res = await get(URL);
        if (res.statusCode == 200) {
                return  parseResponse(res);
        } else {
                throw "No se pudo obtener el fichero json";
        }
 }

 List<Data> parseResponse (Response res){
        List<Data> pars = new List<Data>();
        Map<String,dynamic> body = jsonDecode(res.body);
        for(int i=0; i<body['docs'].length;i++){
                pars.add(Data.fromJson(body['docs'][i]));
        }

        return pars;
}
```

Aquí se aprecia el método “getData” de la clase modelo. Este método hace una petición al servidor y espera a la respuesta. En caso de que la respuesta no sea “200” se lanza un mensaje de error de lectura. En cualquier otro caso, se llama al método “parseResponse” que define una lista de tipo data (ver apartado 9.1) y se parsea el fichero Json.

El método “getData” es invocado desde el controlador, que recibirá como respuesta un List<Data> con la información del servidor.

9.3 Diagrama de Tarta

```
*******************************************************************************
* Fichero: Pieview.dart
*******************************************************************************

class PieView extends StatelessWidget {
        List<charts.Series<DataFormat,String>> seriesList;
        final bool animate;

        void setSeriesList(List<charts.Series<DataFormat,String>> s){
                seriesList = s;
        }

        PieView(this.seriesList, {this.animate});
        @override
        Widget build(BuildContext context) {
                return new charts.PieChart(
                        seriesList,
                        animate: animate,
                        behaviors: [new charts.DatumLegend(
                                position: charts.BehaviorPosition.bottom,
                                outsideJustification: charts.OutsideJustification.middleDrawArea,
                                desiredMaxColumns: 2,
                                desiredMaxRows: 10,
                        )],
                        defaultRenderer: new charts.ArcRendererConfig (arcRendererDecorators: [
                                new charts.ArcLabelDecorator(
                                        labelPosition: charts.ArcLabelPosition.auto,
                                        insideLabelStyleSpec: charts.TextStyleSpec(
                                                fontSize: 16, // size in Pts.
                                                color: charts.MaterialPalette.black
                                        )
                                )
                        ]),
                );
        }
}
```

Esta es la clase Pieview, que define cómo se comporta y se crea el diagrama de tarta.
En el apartado “ArcRendererConfig”, es el lugar en el que se han llevado a cabo algunas de las correcciones que salieron a partir de la evaluación de los usuarios.

Bibliografía

En este apartado se describen los enlaces de los cuales se ha obtenido información que se ha usado para el desarrollo de este proyecto. Del mismo modo, se referencia a los autores de las figuras obtenidas de internet con derechos de reutilización.

[1] Energya, 2019.

"https://www.energyavm.es/quien-fue-el-inventor-de-la-luz-electrica/"

[2] Wikipedia.

"https://es.wikipedia.org/wiki/Tabuladora"

[3] Wikipedia.

"https://es.wikipedia.org/wiki/M%C3%A1quina_de_Turing"

[4] Diccionario de la Real Academia Española de la Lengua.

"https://dle.rae.es/computador"

[5] Wikipedia.

"https://es.wikipedia.org/wiki/Primera_generaci%C3%B3n_de_computadoras"

[6] Biblioteca de recursos de Wikipedia.

"https://commons.wikimedia.org/wiki/File:Eniac.jpg"

[7] Centro de Referencia Estatal de Autonomía Personal y Ayudas Técnicas

"http://www.ceapat.es/InterPresent1/groups/imserso/documents/binario/reto_8.pdf"

[8] Bárbara Yuste, 2018.

"En 2025 el volumen de datos en el mundo será 175 veces más que en 2011"

[9] Stackscale,2020.

"https://www.stackscale.com/es/blog/servidores-internet-mapa/"

[10] Open knowledge foundation.
“https://opendatahandbook.org/guide/es/what-is-open-data/”

[11] Portal OpenData Santa Cruz.
“https://www.santacruzdetenerife.es/web/gobierno-abierto/opendata”

[12] Ministerio de Energía, Turismo y Agenda Digital, 2017
“Buenas prácticas en el Open Data”

[13] 5 Star Data
“Categorías de datos”

[14] Datos.gob.es
“Ranking países europeos”

[15] Datos.gob.es
“Calidad de los datos en España”

[16] Ministerio de industria, energía y turismo, 2016.
“Visualización de datos”

[17] Wikipedia
“https://es.wikipedia.org/wiki/Cartograf%C3%ADa”

[18] Gobierno de España, 2016.
“https://datos.gob.es/sites/default/files/doc/file/informe_herramientas_visualizacion.pdf”

[19] Diagrama incursión Napoleón a Rusia
“https://es.wikipedia.org/wiki/Archivo:Minard.png”

[20] Curva Aragón.
“https://curvaenaragon.com/”

[21] Evolución de la curva Aragón.
“datos.gob.es”

[22] Dart.
“dart.dev”

[23] Android Studio.
“https://developer.android.com/studio/intro”

[24] Android Studio Emulator.
“https://developer.android.com/studio/run/emulator?hl=es-419”

[25] Flutter.
“flutter.dev”

[26] Github.
“github.com/”

[27] Paquete Http Flutter.
“https://flutter-es.io/docs/cookbook/networking/fetch-data”

[28] Coronavirus Mundial por la Universidad Johns Hopkins
“Dashboard”

[29] App Coronavirus cabildo de la Gomera
“Período el Día”

Referencias

En este apartado se van a dar las referencias a las imágenes que se han usado para para este TFG, no aparecen en la memoria, pero si en la aplicación. Estas han sido obtenidas de internet y tienen licencia de uso libre, pero deben ser referenciadas.

[12] Diagrama Patrón MVC.

“https://es.m.wikipedia.org/wiki/Archivo:ModelViewControllerDiagram_es.svg”