



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

Kairos

**Aplicación de compra y venta
de relojes por subasta.**



Presentado por Rodrigo Pérez Ubierna
en Universidad de Burgos — 26 de junio
de 2024

Tutor: Dra. Sandra Rodríguez Arribas
y Dr. Jose Antonio Barbero Aparicio



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



D. nombre tutor, profesor del departamento de nombre departamento, área de nombre área.

Expone:

Que el alumno D. Rodrigo Pérez Ubierna, con DNI dni, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado título de TFG.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 26 de junio de 2024

Vº. Bº. del Tutor:

Vº. Bº. del co-tutor:

D. nombre tutor

D. nombre co-tutor

Resumen

En este primer apartado se hace una **breve** presentación del tema que se aborda en el proyecto.

Descriptores

Palabras separadas por comas que identifiquen el contenido del proyecto Ej: servidor web, buscador de vuelos, android ...

Abstract

A **brief** presentation of the topic addressed in the project.

Keywords

keywords separated by commas.

Índice general

Índice general	iii
Índice de figuras	v
Índice de tablas	vi
1. Introducción	1
2. Objetivos del proyecto	5
3. Conceptos teóricos	7
3.1. Definición de subasta y tipos de subastas	7
3.2. Definición de Machine Learning, algoritmos y evaluación . . .	8
3.3. Arquitectura de aplicaciones móviles	12
3.4. Bases de datos en la Nube	12
4. Técnicas y herramientas	15
4.1. Flutter	15
4.2. Firestore database	16
4.3. Machine Learning con Visual Studio Code	17
4.4. Render	17
4.5. MVC	17
5. Aspectos relevantes del desarrollo del proyecto	19
5.1. Creación y conexión del modelo	19
5.2. Entrenamiento del modelo y conexión con aplicación	24
5.3. Estructura general de la aplicación	26

6. Trabajos relacionados	27
7. Conclusiones y Líneas de trabajo futuras	29

Índice de figuras

4.1. Esquema patrón MVC	18
-----------------------------------	----

Índice de tablas

5.1. Campos vacíos en los datos	20
5.2. Comparación de métricas	23

1. Introducción

¿Sabrían decirme el nombre del Dios del tiempo en griego? Si la respuesta es Cronos, están en lo cierto. Si la respuesta es que forma más extraña de comenzar una memoria, también están en lo cierto. Aun así, déjenme centrarme en la primera respuesta. Cronos es el dios del tiempo. Para él tiempo es lineal. Y ahora, si les preguntase por el nombre del Dios de la oportunidad ¿qué me responderían? Si saben la respuesta, fantástico; si no, disfruten del título de este trabajo.

Kairós no es un trabajo cualquiera. Kairós nace de un *hoobie* que arrastro desde hace unos años, concretamente desde la herencia de mi primer reloj. Yo no solía fijarme mucho en ellos, pues siempre creí que no era más que una simple herramienta para saber qué hora en un determinado momento. Sin embargo, poco a poco decidí informarme sobre este mundo y pasé a verlo como lo que era: arte. Un reloj no es un simple accesorio. Un reloj define la identidad de la persona. Muchas personas afirman que un conjunto bonito pierde todo *glamour* si no es acompañado de un reloj en la muñeca. Por esta razón, nunca salgo sin que haya algo que pese en mi brazo.

Actualmente, el coleccionismo de relojes atrae a numerosas personas, especialmente hombres. Podríamos hablar de a que civilización se le atribuye la invención del reloj, aunque no sería de apoyo al trabajo. Sin embargo, una gran curiosidad es quién portaba los primeros relojes de pulsera en la historia: las mujeres. Los hombres siempre portaban relojes de bolsillo agarrados con una cadena. ¿Curioso verdad? Pues no es hasta la Primera Guerra Mundial cuando los hombres deciden utilizar los relojes de pulsera debido a la comodidad detrás de las trincheras.

¿Y por qué este apartado anterior? La respuesta es sencilla: las principales marcas de relojes hacen colecciones para el género masculino, pues son ellos

más propensos a portarlos de manera diaria. De ahí que la gran mayoría de relojes que veremos a lo largo del trabajo lleven la característica del género.

Tras un tiempo informándome, hablando con otros amantes de este mundo... ví que hay un gran obstáculo en esta sociedad cuando hablamos de relojes: la compra y venta de estos productos. No existen apenas sitios web o aplicaciones que se dediquen exclusivamente a la adquisición, bien por subasta o bien por compra directa, de relojes. Por esto, surge Kairós.

Kairós es una aplicación multiplataforma donde cualquier usuario podrá comprar y/o vender su reloj de una manera sencilla. La adquisición o venta de estos se realizará a través de la subasta ascendente, así como por venta directa.

Y ahora una de las preguntas que surgen dentro de este entorno: pero ¿cuánto vale mi reloj? Esta pregunta es la más formulada a la hora de vender esta pieza de arte. Desde Kairós queremos eliminar cualquier duda del usuario y, por ello, la aplicación cuenta con un sistema de predicción del precio del reloj según diversas características propias del accesorio.

Con todo situado, podemos marcar cuáles van a ser los principales apartados de la memoria del trabajo y la dinámica que se va a seguir para su realización:

1. Objetivos del proyecto
2. Conceptos teóricos
3. Técnicas y herramientas
4. Aspectos relevantes del desarrollo del proyecto
5. Trabajos relacionados
6. Conclusiones y líneas de trabajo futuras

Cada apartado expondrá distintas cosas que se han ido realizando durante el proceso, aunque todos compartirán una misma estructura. En todo ellos se explicarán tanto la parte de la aplicación como la parte del Machine Learning, de forma que todo quede lo más limpio y ordenado posible. Personalmente es un reto para mí ya que mis conocimientos en cada una de las materias son ínfimos, pero es lo que tienen los retos: a base de golpes y más golpes se llegará al objetivo final.

Entonces ¿por qué el nombre Kairos? Como les contaba, según la mitología griega, su nombre define al dios de la oportunidad. Kairos era representando con unas alas en sus pies y un par de pelos muy largos. Decían que si pasaba por tu lado, solo debías estar rápido y cogerle de su escasa melena. Esta era mi oportunidad de agarrar a Kairos y empezar a aprender cómo crear y lanzar mi propia aplicación. Disfruten del trabajo.

2. Objetivos del proyecto

Definir los objetivos a la hora de realizar un proyecto no es tarea sencilla, pues un mismo trabajo puede tener enfoques diferentes. Como hablaba en la introducción, esto nace de juntar dos de mis pasiones en un mismo sitio. La oportunidad que me brinda la universidad para combinar ambos mundos es única y, por eso mismo, me gustaría llegar a alcanzar los siguientes objetivos:

1. Crear una aplicación intuitiva y amigable para cualquier tipo de usuario.
2. Estructurar las vistas de la aplicación de forma que el usuario no se pierda en ningún momento y tenga la certeza de que ninguno de sus movimientos es erróneo.
3. Lanzar una aplicación disponible desde cualquier dispositivo móvil y que responda a los cánones de belleza que una aplicación de este nivel merece.
4. No entablarse en un solo sistema operativo y conseguir que nuestra aplicación sea multiplataforma.
5. Conocer las bases de Flutter y aplicar estos conocimientos de la manera más efectiva posible.
6. Conseguir una comunicación rápida y precisa entre el marco de desarrollo Flutter y la base de datos Firestore Database, consiguiendo así una rapidez comunicativa en el traspaso y operaciones de datos.
7. Tras varios momentos pensando como esta aplicación podría situarse como una clara competidora en el mercado, me di cuenta de que

muchas aplicaciones no se centran en lo que de verdad importa: el cliente. Siempre se ha dicho que el cliente tiene la razón, que el cliente debe encontrarse cómodo... pero a la hora de la verdad, las aplicaciones no destinan el esfuerzo suficiente a dejar todo lo más sencillo, amigable y accesible que el cliente merece. Por ello, lo primero que hice fue ponerme en su piel y pensar que me gustaría encontrarme en mi aplicación, que problemas podría a llegar a tener... y di con ello: desconozco cuánto vale mi reloj. Sé que puede sonar raro, pero es así. Son tantos los precios que se estipulan a un mismo reloj que sé, a ciencia cierta, que sería incapaz de marcar un precio de venta directa donde consiguiese el mayor valor de venta posible. De aquí nace la idea de investigar acerca del aprendizaje automático o Machine Learning. Mi objetivo principal es crear un modelo capaz de predecir cuál es el precio más recomendable para la venta de mi reloj, apoyándome en un dataset de más de 280000 relojes de diferentes marcas, estados, géneros, precios...

8. Crear una API que permita comunicar nuestra aplicación con el modelo de predicción de precios de manera rápida.
9. Dar libertad al usuario para poder subir sus relojes y ponerlos en venta, siguiendo unas directrices, pero pudiendo marcar precios y tiempos de finalización según su elección.
10. Controlar muy bien todos los casos posibles que pueden darse tanto a la hora de subir un reloj como a la hora de crear y llevar a cabo una subasta. El control de esto es fundamental para no experimentar problemas monetarios.
11. Seguir el patrón de arquitectura Modelo-Vista-Controlador de forma que la aplicación pueda ser escalable de manera sencilla.

Estos objetivos marcan las bases fundamentales para que la nuestra aplicación sea robusta. A medida que escalemos esta aplicación, el número de objetivos crecerá, consiguiendo cimas que todo programador quiere alcanzar.

3. Conceptos teóricos

En este apartado del informe trataré de abordar los puntos teóricos necesarios para afrontar de manera más clara el resto de secciones. La razón principal de que aparezcan unos sí y otros no es puramente personal. El objetivo de este apartado es resumir puntos clave que me han servido a mí para conseguir los objetivos marcados en el punto anterior.

3.1. Definición de subasta y tipos de subastas

El término subasta se define como la situación donde dos o más compradores pujan distintas cantidades de dinero por un mismo producto, siendo uno de ellos el comprador final. Para llevarla a cabo, se deben seguir una serie de reglas y directrices marcadas antes del comienzo de la subasta. El objetivo de este tipo de ventas es conseguir una mejor venta posible, llegando a un acuerdo justo entre vendedor y comprador.

Actualmente, existen muchos tipos de subasta, pero todas ellas se basan en cuatro pilares fundamentales o tipos:

Precio descendente u holandesa: la subasta comienza con un precio elevado y va disminuyendo a medida que nadie interrumpe al subastador. En el momento que alguien quiere aplicar, para al subastador en el precio que desee. Este hombre será el ganador de la subasta.

Precio ascendente o inglesa: la subasta comienza con un precio mínimo y va subiendo a medida que los posibles compradores lanzan pujas de

igual o mayor precio de lo que el subastador indica. El ganador de la subasta será aquella persona que lance la puja más alta.

De sobre cerrado a primer precio: los pujadores introducen el precio en un sobre cerrado. Una vez estén todos listos, se abren los sobres y gana la persona que marcó la puja más alta, teniendo que pagar tal cantidad por lo subastado.

De sobre cerrado a segundo precio o Vickrey: la dinámica es igual que el anterior tipo de subasta, pero con una diferencia: la persona con la puja más alta pagará el segundo precio más alto por el producto.

Una pregunta que puede surgirnos es por qué existen tantos tipos si el objetivo es el mismo. ¿Por diversión? ¿Por cambiar? La respuesta reside en el objetivo de alcanzar el mejor acuerdo posible vendedor-comprador, es decir, sacar el máximo grado de objetividad a la venta.

En una subasta podemos diferenciar entre las pujas normales y el precio reserva, este último siendo nuestra máxima cantidad a pujar. Referenciando a los tipos explicados hace unas líneas, la subasta de sobre cerrado a primer precio no es una subasta del todo óptima. El pujador no trata de lanzar su precio de reserva y esto puede hacer que, si la subasta se repitiese, el precio se ajuste mucho y el mercado se vuelva volátil. Sin embargo, la subasta de sobre cerrado a segundo precio o Vickrey incita al pujador a adoptar una estrategia dominante y lanzar su puja máxima, obteniendo un grado de venta muy satisfactorio para ambas partes.

En nuestro caso, el tipo de subasta elegido para nuestra aplicación será subasta de precio ascendente o inglesa, sin duda el tipo de subasta más frecuente.

3.2. Definición de Machine Learning, algoritmos y evaluación

El aprendizaje automático o Machine Learning es una parte de la informática muy ligada al concepto de AI o Inteligencia Artificial. Definir el objetivo de esta es simple: conseguir que una máquina se comporte como un ser humano. Consegir el objetivo de esto es algo más complicado. Principalmente se busca entrenar a la máquina con el uso de algoritmos y datos de forma que esta aprenda de manera gradual. No hay que decir que cuánto

mayor sea el número de datos y cuánto mayor sea la calidad de estos, mayor precisión se conseguirá.

El término Machine Learning podemos atribuírselo a Arthur Samuel quien, en 1952, creó un software capaz de aprender y jugar a las damas. Aun así, este campo ha evolucionado a pasos agigantados, siendo actualmente uno de los puntos más relevantes dentro del campo de la informática. Entre los hitos más significativos, destacan principalmente unos estudiantes de la Universidad de Stanford. Estos alumnos consiguieron desarrollar en 1979 un software capaz de pilotar un carro de manera autónoma sin que este chocara con ningún obstáculo. No podemos olvidarnos de uno de los momentos más significativos de nuestra época cuando el gran Garri Kaspárov fue derrotado por Deeper Blue en una partida de ajedrez en 1997, siendo la primera vez que el ajedrecista perdía contra una máquina.

Como se ha marcado en esta introducción, esta rama de la informática se centra en desarrollar algoritmos y modelos autosuficientes, es decir, que sean capaces de aprender de manera automática a través de datos y la experiencia. Dentro de esta rama, encontramos infinidad de técnicas y algoritmos, aunque me voy a quedar con los más comunes.

El aprendizaje supervisado es una rama del aprendizaje automático donde se entrena a un modelo con una serie de datos que contienen cuál es su entrada y cómo debe ser su salida. Un ejemplo de ello es la venta de un local: el modelo recibe como variables de entrada el número de baños, los metros cuadrados, el año de edificación... y el modelo procesa todo dando como salida el precio del local. Dentro de este tipo de aprendizaje pueden abordarse dos tipos de problemas: regresión si lo que devuelve la máquina es un valor justo (siguiendo con el ejemplo anterior, el precio del local), o clasificación si lo que devuelve es una categoría (clasificar imágenes de vehículos entre coches o camiones). Hay que añadir que son muchos los algoritmos de los que consta, aunque destacan principalmente:

Regresión lineal: se modela la relación de una variable dependiente con una o varias variables independientes.

Regresión logística: se modela para solventar problemas de clasificación binaria.

SVM: se utiliza tanto en regresión como en clasificación. El objetivo es separar las clases en el espacio de características a través de la búsqueda del hiperplano.

Árboles de decisión: el objetivo es dividir el espacio de características en regiones y asignar una etiqueta a cada una de ellas.

Random forest: conjunto de árboles de decisión que trabajan de manera combinada para alcanzar una mayor precisión y así evitar el sobreajuste.

Por otro lado estaría el aprendizaje no supervisado donde los datos no tienen ni etiquetas ni salidas asociadas. Su objetivo es que la máquina aprenda a base de identificar patrones en los datos y extraer información útil de ellos. Para explicar mejor esto, pongamos un ejemplo: teniendo un conjunto de transacciones bancarias, determinar grupos de personas según las similitudes de compras. Dentro de ello podemos distinguir tres tipos de problemas:

Clustering: el objetivo es agrupar los datos en conjuntos o clusters según existan similitudes entre ellos. Por indicar alguno de los algoritmos dentro de este tipo, destacan:

K-Means: divide los datos en k grupos teniendo en cuenta la minimización de la varianza. El objetivo es que los centros de estos grupos se vayan actualizando de manera iterativa minimizando así el sumatorio de las distancias cuadradas entre los datos y el centro más cercano.

Clustering Jerárquico: consiste en agrupar datos a través de la fusión iterativa de los clusters más similares. El resultado suele ser representado con un dendrograma.

Density-Based Spatial Clustering of Applications with Noise (DBSCAN): agrupa los datos en función de la densidad. Destaca por ser un algoritmo de gran ayuda a la hora de detectar ruido.

Reducción de dimensionalidad: el objetivo es reducir el número de variables en un conjunto de datos sin perder la mayor parte de la información. Ejemplos de algoritmos son:

Análisis de Componentes Principales (PCA): reduce dimensionalidad transformando los datos en nuevas variables ordenadas según la cantidad de varianza que explican.

T-Distributed Stochastic Neighbor Embedding (t-SNE): reduce la dimensionalidad disminuyendo el espacio y manteniendo las relaciones de proximidad locales. Muy útil para la búsqueda de patrones.

Asociación: el objetivo de la máquina es que aprenda a base de relacionar variables en un conjunto de datos determinado. Un ejemplo dominante dentro de este grupo son la sección de recomendación de webs de compras. Ejemplos de algoritmos dentro de ello son:

Equivalence Class Clustering and bottom-up Lattice Traversal (Eclat):
permite encontrar conjuntos de datos frecuentes en bases de datos transaccionales.

Frequent Pattern Growth (FP-Growth): mismo objetivo que el anterior pero la dinámica consiste en la contrucción de un árbol de patrones frecuentes.

Si combinamos las ideas fundamentales de estos dos aprendizajes anteriores, damos lugar al aprendizaje semisupervisado. Simplemente permite que la máquina aprenda a base de datos sin etiqueta y con ella.

Otro tipo de Machine Learning es el conocido como aprendizaje por refuerzo. La idea fundamental de este tipo de aprendizaje es que la respuesta del modelo mejore a partir de una retroalimentación en forma de recompensas. En otras palabras, el algoritmo va a aprender basándose en lo que le rodea y siguiendo la filosofía “ensayo-error”. Se utiliza en infinidad de sectores y los algoritmos por excelencia dentro de este aprendizaje son:

Q-Learning: actualiza los valores teniendo en cuenta las recompensas recibidas y las estimaciones futuras.

State-Action-Reward-State-Action (SARSA): actualiza los valores teniendo en cuenta las acciones presentes y las futuras.

Algoritmo de Policy Gradient: actualiza los valores según la dirección del gradiente de la recompensa esperada.

Por último, otro aprendizaje muy presente en diversos sectores es el Deep Learning. El objetivo de este campo es emular el comportamiento de un cerebro humano a partir de algoritmos de redes neuronales artificiales formadas por múltiples capas de procesamiento capaces de representar datos de alto nivel. Ejemplos muy característicos son el reconocimiento de imágenes, detección de objetos, detección de fraudes, traducción automática...

En nuestro caso, nuestro problema responde a un aprendizaje supervisado. Aunque lo veremos más adelante, se han probado dos algoritmos distintos, siendo Regresión Lineal el definitivo.

3.3. Arquitectura de aplicaciones móviles

Cuando hablamos de la arquitectura de aplicaciones móviles hablamos tanto de los componentes y su interacción como de la estructura y diseño que albergan dentro de ellas. Para explicarlo de una manera óptima, podemos dividir esta estructura en una serie de capas:

Capa visual o Frontend: esta capa recoge todo lo relacionado con la interfaz y la experiencia de usuario (UI y UX respectivamente). Sus componentes principales (siguiendo el patrón MVC explicado en el punto 4 de este informe: Técnicas y herramientas) son:

1. Controladores : albergan la lógica de presentación de la aplicación.
2. Vistas: muestran al usuario la información que les llega desde los controladores.

Capa lógica : dentro de esta capa se encuentran los modelos según el patrón MVC. No soy partidario de meterlo ni en la capa anterior ni en la posterior porque no responde como tal a ninguna de las dos. En este caso, los modelos son el puente entre ambas partes, pues se encargan del manejo de los datos.

Capa de datos o Backend: capa que tiene como objetivo almacenar datos y prepararlos para su posterior uso en la aplicación. Dentro de esta capa encontramos componentes como:

1. APIs : permiten hacer llamadas a servicios externos con el fin de recuperar datos.
2. Bases de datos: almacenan datos de manera persistente.

En nuestro caso, Flutter será quién protagonice la capa visual y la capa lógica, mientras que Firestore Database será quien marque la capa de datos.

3.4. Bases de datos en la Nube

Como hemos apuntado en la sección anterior, una base de datos es un conjunto de datos almacenados y que tienen relación entre sí. Los SGBD o Sistemas Gestores de Bases de datos son los programas que gestionan todo esto. Dentro de las bases de datos, podemos diferenciar dos grandes tipos:

1. Bases de datos SQL o relacionales

2. Bases de datos NoSQL o no relacionales

Las bases de datos SQL almacenan una serie de datos y sus relaciones con otras tablas a través de claves, datos que son los mismos en tablas distintas. Cada registro se ve representado por filas mientras que las columnas representan los campos de cada registro. Esta filosofía de almacenamiento nos permite acceder de manera sencilla a cualquier dato. Para ello, contamos con el lenguaje SQL.

Por otro lado se encuentran las bases de datos NoSQL, las cuales surgen por la necesidad de almacenar la información sin seguir una estructura determinada. Se utilizan cuando el número de datos a albergar es muy grande y necesitamos rapidez en el tratamiento de estos datos, cosa que las anteriores, en general, no permiten.

Siguiendo con nuestro ejemplo, la base de datos a utilizar es Firestore Database, base de datos NoSQL.

4. Técnicas y herramientas

Habiendo definido los aspectos teóricos más clave para entender un poco mejor el trabajo, se define a continuación cuáles han sido las técnicas y herramientas utilizadas para alcanzar nuestro objetivo. Aunque las hemos nombrado en alguno de los apartados anteriores, esta sección del informe explica de manera más particular qué nos aportan y cuáles son las ventajas frente a otras.

4.1. Flutter

Flutter es un SDK (Software Development Kit) de código abierto desarrollado por Google. Su objetivo principal es la creación de aplicaciones multiplataforma con un único código base. Se basa en el lenguaje de programación Dart, el cual es un lenguaje de programación orientado a objetos desarrollado por Google con el fin de crear aplicaciones web y de escritorio de una manera rápida y sencilla.

Volviendo al framework, Flutter es una herramienta que permite crear interfaces de usuario atractivas visualmente y de una alta personalización. Hay que añadir que dispone de una amplia gama de widgets predefinidos y listos para ser utilizados por cualquier usuario.

En cuanto a su rendimiento, Flutter destaca por poseer su propio motor de renderizado eliminando así la necesidad de dependencia de componentes de interfaz de usuarios nativos.

Aún que son muchas las ventajas que presenta esta herramienta, hay que destacar una frente a las demás: la posibilidad de crear multiplataforma. Flutter nos va a permitir reutilizar el mismo código para crear nuestra

aplicación en las plataformas iOS, Android, web, Windows, MacOS y Linux. A esta ventaja, añadir la posibilidad de recargar el código en caliente, aspecto muy demandado por cualquier persona que se dedique a la programación.

Para explicarlo de una manera sencilla, la unidad mínima de este lenguaje es el widget. Estos objetos son los que van a conformar cualquiera de las vistas que tenga nuestra aplicación. Estos widgets pueden estar formados a su vez por dos o más widgets, consiguiendo una libertad de edición y reutilización de widgets enorme. A esto añadir la gran cantidad de librerías con widgets ya predefinidos y listos para ser usados.

En lo que a este trabajo respecta, no todo son ventajas. La curva de aprendizaje de Flutter tiene muy poca pendiente y esto hace que conseguir una aplicación base llegue a ser frustrante. Solo para poder descargar el entorno debes tener unos requisitos e instalar una serie de programas previos que hacen que la probabilidad de error sea elevada. A esto hay que añadir que desde los últimos años, debido a la aparición de otros competidores, Flutter ha dejado de ser protagonista y su comunidad y documentación a disminuido de manera drástica, llegando a salir noticias actuales como el despido de gran parte del equipo de Flutter en Google.

4.2. Firestore database

Siguiendo un poco la herramienta utilizada en la capa visual, buscamos qué Sistema de Gestión de Base de Datos podría ser el óptimo para trabajar con Flutter. La verdad que encontrarlo resuelto sencillo pues Firestore Database también está desarrollado por Google y ofrece facilidades siempre que lo conectes a un SDK como Flutter.

Firestore Database es una base de datos alojada en la nube que responde al tipo NoSQL. Entre sus características principales destacan la rapidez de respuesta y su sincronización de datos en tiempo real. La información se organiza en colecciones y documentos, los cuales podríamos comparar con tablas y registros respectivamente. Sin embargo, es una base de datos que no requiere de esquemas predefinidos, permitiendo así cambios en la estructura de los datos.

Si tuviésemos que definir alguna ventaja más, podríamos afirmar que Firestore es capaz de gestionar grana cantidad de datos y sincronizarlos de manera rápida, así como una gran seguridad a la hora de autenticarnos. Sin embargo, existen algún que otra desventaja difícil de evitar durante este trabajo.

Nunca había tratado directamente con una base de datos distinta al formato SQL, lo que provocó que mi forma de definir las tablas y tratar los datos no fuese del todo correcta en varios momentos críticos del trabajo. Por otro lado, la conexión con la base de datos es una etapa con muchos pequeños pasos intermedios que, al igual que la preparación de Flutter, pueden desembocar en errores si no se cumplen al pie de la letra.

4.3. Machine Learning con Visual Studio Code

Aunque existen otras formas de realizarlo, la parte de preparación de los datos y entrenamiento del modelo lo he realizado con Visual Studio Code. En otras asignaturas de esta carrera hemos utilizado Anaconda para trabajar sobre Jupyter Notebook, pero gracias a una extensión he trabajado de manera más sencilla así.

Por otro lado, tanto el script para lanzar ejecutar el modelo y la API han sido realizados con la misma herramienta pero con extensión .py

4.4. Render

Render es un sitio web que permite desplegar infinidad de aplicaciones web, servicios backend, bases de datos... de una manera sencilla. Aunque hablaremos más a fondo en otros apartados de cómo ha sido la conexión, me gustaría destacar dos ventajas frente a otros competidores como Fly o Heroku:

1. Es gratuita. Las otras parecen ofrecer servicios gratis pero acaban pidiendo un método de pago.
2. Si tenemos conectado nuestro espacio a un repositorio de Git, se realizará un autodeploy cuando esta detecte algún commit en la carpeta indicada.

4.5. MVC

Las siglas MVC responden al patrón Modelo-Vista-Controlador utilizado con el objetivo de llevar a cabo una aplicación con una arquitectura concreta. Se compone de tres partes:

Modelo: podríamos definirlo como el script donde se van a definir las consultas a la base de datos. Representa a la capa lógica de nuestro programa.

Controlador: es el puente intermedio entre el modelo y la vista. Recibe los datos introducidos por el usuario a través de la vista, los lleva al modelo para realizar las acciones necesarias, los recupera de nuevo y los envía a la vista.

Vista: es la interfaz de usuario. Su objetivo es presentar al usuario los datos que le envía el controlador.

Para ser más fácil su asimilación, se puede apreciar el flujo de datos en la Figura 4.1. Se ha intentado seguir este patrón debido a que lo he aprendido en mis prácticas curriculares y quería intentar seguirlo, pero Flutter no permite aplicarlo tan puro como otros lenguajes como PHP.

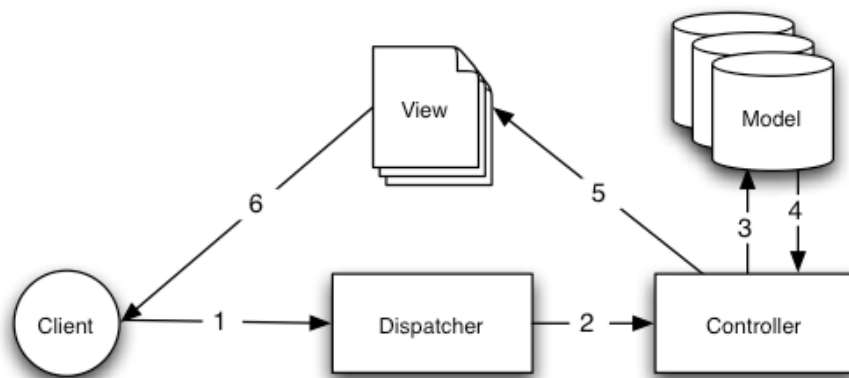


Figura 4.1: Esquema patrón MVC

5. Aspectos relevantes del desarrollo del proyecto

Con todo introducido, pasamos a desarrollar uno de los puntos clave de la memoria de este trabajo: explicar cómo y por qué se ha realizado el trabajo de esta forma. Aunque es cierto que se podría explicar todo el proceso seguido, creo que es conveniente centrarnos en los aspectos que han sido claves para llegar al proceso final y que, muchos de ellos, han sido verdaderos quebraderos de cabeza.

Si algo define este trabajo es la capacidad de una persona para conseguir realizar una aplicación y un modelo de Machine Learning comenzando completamente de cero, pues partía de no conocer absolutamente nada de lo que hoy es el resultado final.

A modo de ser ordenado, vamos a dividir este apartado en tres partes clave:

- Preparación del dataset.
- Entrenamiento del modelo y conexión con aplicación.
- Estructura general de la aplicación y control de estados.

5.1. Creación y conexión del modelo

Como apuntaba al principio de este apartado, mis conocimientos en el ámbito del Machine Learning eran ínfimos. Había leído algún artículo relacionado con ello, pero nunca había entrado en detalle. Por esta razón,

Nombre	Campos vacíos	Nombre	Campos vacíos
Unnamed	0	name	72585
price	406	brand	131
model	30466	ref	43152
mvmt	196685	casem	164271
bracem	174896	yop	134
cond	75987	sex	95805
size	23597	condition	212922

Tabla 5.1: Campos vacíos en los datos

la parte de creación del modelo de predicción de precios fue la primera en realizarse.

Tras hablar con mis tutores como podríamos afrontarla, marcamos como primer objetivo la búsqueda de un dataset que respondiera de manera eficaz a nuestras necesidades. Fueron muchos los que encontramos, teniendo como palabras clave de búsqueda ".E-commerce." "Watches", entre otras. La búsqueda no fue sencilla pues mi desconocimiento de la materia y la gran cantidad de dataset presentes en la red no me permitía elegir con precisión cual podía ser un buen punto de partida.

Tras varios días de búsqueda, el sitio web "Kaggle" me brindó el dataset definitivo: "Luxury Watch Listings". Este dataset contenía nada más y nada menos que más de 280000 relojes con sus características y su precio de venta. Tras ponerlo en común, fue el elegido.

Una vez elegido, busqué toda información necesaria para tratarlo y poder crear un modelo que devolviese una predicción del precio de cualquier reloj. Leyendo en diversas fuentes de información pude darme cuenta que el problema principal a la hora de entrenar un modelo es la presencia de valores nulos. Concretamente, nuestro dataset poseía los datos nulos indicados en la Tabla 5.1.

Aparentemente no eran muchos los campos nulos en proporción a las 280000 líneas del dataset. Sin embargo, esto no fue del todo cierto. Revisando previamente el archivo, vi que existían múltiples datos que nos iban a dificultar nuestro trabajo. Por esta razón, se procedió a hacer un estudio previo desde la aplicación Microsoft Excel. El proceso consistió en crear una

tabla dinámica con todos los datos y encabezados, de forma que se pudiera ver los distintos nombres o números que formaban el rango de datos.

Tras estudiar los datos y saber que la siguiente fase del proceso era convertir todo esa información en números para poder entrenar el modelo, se llegó a una serie de soluciones a problemas que se plantean a continuación:

Columna unnamed: esta columna, a primera vista, parecía que marcaba de manera incremental el número de filas del archivo. Sin embargo, cuando cambiábamos de marca de reloj, volvía a comenzar en uno. Por tanto, trataba de marcar el número de relojes por marca. Resumiendo, fue inútil trabajar con ella para conseguir nuestro objetivo. La solución que se planteó fue el borrado de esta.

Columna name: sin duda fue la columna más problemática de todo el archivo. Más que ser el nombre del reloj, era una descripción de este. El problema principal fue que la descripción no seguía ninguna estructura: marcaba los datos que quería, como quería, repetía datos de otras columnas... En resumen, ha sido y es una columna intratable. Por tanto, la solución fue su borrado.

Columna price: esta columna no presentaba apenas errores. Simplemente marcaba el precio de cada reloj. Sin embargo, aquel producto que no se le fuese marcado un precio, adquiriría el valor “Price on request” o directamente se encontraba en blanco. La solución que se planteó fue dar a estas dos excepciones el valor -1 para entender que no existe información relativa sobre ese reloj. Hay que añadir que se eliminó el símbolo dolar para evitar errores.

Columna brand: no presentó ningún error. Simplemente, se categorizó para su posterior uso en el modelo a crear. Las celdas sin información pasaron a valer -1.

Columna model: no presentó ningún error. Simplemente, se categorizó para su posterior uso en el modelo a crear. Las celdas sin información pasaron a valer -1.

Columna ref: siendo como soy, un apasionado de los relojes, sabía que tener la referencia del reloj que buscamos es uno de los puntos más claves a la hora de reconocer el reloj. Sin embargo, la columna de nuestro dataset no se encontraba del todo limpia. Aun así, no quise perder tan valiosa información y me di cuenta de que toda referencia que lleva una letra siempre aparecería en mayúscula. Por tanto, la

solución que se dió fue eliminar toda aquella palabra que tuviera al menos una minúscula.

Columna mvmt: no presentó ningún error. Simplemente, se categorizó para su posterior uso en el modelo a crear. Las celdas sin información pasaron a valer -1.

Columna casem: no presentó ningún error. Simplemente, se categorizó para su posterior uso en el modelo a crear. Las celdas sin información pasaron a valer -1.

Columna bracem: no presentó ningún error. Simplemente, se categorizó para su posterior uso en el modelo a crear. Las celdas sin información pasaron a valer -1.

Columna yop: la información se brindaba de cuatro formas diferentes:

- Un único año (Ej. 2024)
- Varios años separado por comas (Ej. 2021, 2022, 2023)
- Un año seguido de la palabra aproximación (Ej. 2020 (Approximation))
- No se sabe el año y se marca como Unknown, o se deja en blanco.

La solución que se aportó fue únicamente marcar un año por celda. Por tanto, para el primer caso no se hizo nada; el segundo caso se hizo la media entre todos los años que se marcaban; para el tercero se quitó la palabra “Approximation”; y para el último se cambió el valor “Unknown” o celda blanca por -1.

Columna cond: esta columna tenía los datos perfectamente dispuestos para ser categorizados. Sin embargo, se vió que terminaba a mitad de camino, siendo la columna “condition” quien seguía el trabajo de esta. La solución que se aportó fue combinar ambas columnas en una misma y categorizar, marcando como -1 aquellas celdas que se encontraban vacías.

Columna sex: no presentó ningún error. Simplemente, se categorizó para su posterior uso en el modelo a crear. Las celdas sin información pasaron a valer -1.

Columna size: debido a que fue un campo que opuso muchas dificultades, la solución que se aportó fue simple: borrar la columna. No es un campo que aportara información útil comparado con otros campos

	Regresion Lineal	Random-Forest-Regresor
Mean-Squared-Error	2033534792.8564095	1380488875.0762675
Root-Mean-Squared-Error	45094.73132037056	37154.930696695796
R2-score	0.4078856446303635	0.598036245442073
R2-score-ajustado	0.4078464450275924	0.5980096343333164
Mean-absolute-percentage-error	1.2132659733005473	0.23691091500423903

Tabla 5.2: Comparación de métricas

como la marca, el estado, el precio... que son variables que van a definir casi por completo la predicción objetivo.

Columna condition: explicada en apartado "Columna cond".

Para llegar a este momento pasaron par de semanas hasta que creí encontrar la solución más óptima para poder empezar a crear y entrenar un modelo. Realicé dos líneas de entrenamiento: Regresion Lineal y Random-Forest-Regresor. Los resultados de las métricas fueron los indicados en la Tabla 5.2.

Tras una reunión vimos que la métrica "Mean Absolute Percentage error." era muy buena, mientras que las demás no lo eran tanto. Esto nos hizo dudar de cómo se estaban tratando, llegando tras varios días a la conclusión de que había cuatro errores clave en la forma de tratar mis datos:

- Al categorizar por mi cuenta, estaba queriendo decir al modelo de alguna forma que la marca de relojes que recibía el valor 1 era peor que la marca de relojes que recibía el valor 2, y esto no tenía por qué ser así.
- Dar el valor -1 a aquellos campos nulos o sin información relevante no fue una buena práctica para entrenar al modelo.
- Durante la realización de esta parte, la consola me devolvía error con el tipo de datos. Tras dos semanas intentado ver que ocurría vi que estaba cometiendo un error: los valores int no tienen soporte directo para representar NaN, sin embargo float sí. Esto me hizo tener que volver a empezar de nuevo y pensar de manera distinta el proceso.
- La librería pandas trata los valores faltantes de diferente forma que la librería numpy, siendo pandas pd.NA y numpy np.NaN.

En conclusión, tuvimos que cambiar de nuevo la forma de trabajar. Dicen que a veces es más fácil desaprender y volver a realizar las cosas de nuevo. En mi caso, me surgió efecto.

Lo primero que hice fue ver como podía limpiar los datos, cosa que reutilicé parte de lo que ya había pensado anteriormente. No traté de categorizar, simplemente me quedé con lo que me interesaba y lo que no lo vacié. Con esto listo, pensé en como podía completar los campos vacíos:

- Para la columna precio, intenté aplicar la media de aquellos que comparten marca y modelo. No sirvió, pues existen relojes de la misma marca y modelo que se diferencian mucho, bien por la correa, la edición, el año...
- No podía dar valores porque sí al sexo, brazalete, caja, año...

Debido a no tener forma de dar valores a los campos faltantes, me salieron bastantes campos sin información. Sin embargo, empecé a probar combinaciones de columnas para ver con cuantas me podía quedar sin perder parte de las filas del dataset. Tras varias pruebas, las candidatas fueron:

- brand
- model
- condition
- yop
- price

El resultado final fue un dataset de 160000 líneas con tres columnas que pasarían a ser categóricas en un futuro, una columna numérica y una columna objetivo.

5.2. Entrenamiento del modelo y conexión con aplicación

Referenciando al apartado anterior, nos situamos en el dataset final. Tras semanas de trabajo y reuniones con tutores, vimos que debíamos centrarnos en cómo íbamos a comunicar el modelo de Machine Learning con la aplicación

de Flutter. Este proceso fue uno de los más duros, pues la información que había en la red era escasa y diferente. Si tuviera que definir este apartado en una frase sería: prueba y error.

Con el dataset preparado, descubrí OneHotEncoder (explicado en parte teórica) y entrené al modelo, consiguiendo un modelo que predecía el precio del reloj. Tras revisar las pocas fuentes de información disponibles, ví que existía una herramienta para conectar el modelo: tflite. Al parecer, todo era muy sencillo. Simplemente debía conseguir guardar el modelo con la extensión .h5, agregar una serie de dependencias en mi programa y descargar algún que otro paquete. Sin embargo, esto o fue así. Cada vez que lanzaba de alguna manera peticiones al modelo, el problema con los tipos de datos afloraba.

Esto no fue el único problema. Un error mío fue entrenar siempre por completo el modelo cada vez que intentaba una cosa nueva. Esto me suponía cerca de tres cuartos de hora esperando a que terminara, muchas veces con el resultado final de no poder ni guardar el modelo.

Al igual que con el dataset, dar con el problema fue un mundo. Traté de cambiar el tipo de datos, traté hacer conversiones extensiones sobre el modelo... pero nada. El modelo no respondía. Volviendo un poco al primer apartado, pensé que podía ser problema de las versiones de "tensorflow", "Flask"... y así fue: cada vez que realizaba un "pip install tensorflow" se descargaba por detrás una librería "tensorflow-intel" que descuadraba la compatibilidad de unas librerías con otras. Resumiendo, había que buscar otra forma.

Ojeando par de vídeos y leyendo en algún sitio web, ví que podía conseguir mi objetivo dando algo más de vuelta. Como veía que entrenar mi modelo suponía una gran cantidad de tiempo, tomé el consejo de mis tutores y traté de reducir el número de líneas de entrenamiento ya que mi objetivo era que se conectase a la aplicación como fuera.

La extensión elegida fue ".joblib". Una vez conseguida, realicé un pequeño script en python para poder lanzar en local. Con todo ello, conseguí que el script funcionase, pero tuve problemas con la forma de lanzar peticiones desde la terminal. Esto fue un poco frustrante, ya que veía como siguiendo la información disponible en red funcionaba y yo no lo conseguía.

No podía destinar el tiempo del que no disponía en ello, así que intenté trabajar con Postman, herramienta muy intuitiva que yo había utilizado tanto en el mundo estudiantil como en el mundo laboral. Tras un tiempo

configurando todo y corrigiendo errores, conseguí dar respuesta a mi petición POST, cosa que celebré como un gol en el ámbito futbolístico.

La siguiente fase era desplegarlo de alguna forma para que mi aplicación Flutter pudiera hacer la llamada que había conseguido con Postman. Muchos vídeos me recomendaban Fly.io y Heroku. Ambas presumían de ser entornos gratuitos, pero a la hora de la verdad, te obligaban a ingresar un método de pago. Aun así, comenté mi problema con algún compañero y me recomendó Render, el cual sería mi opción final.

El camino de Render era aparentemente sencillo: conectar con mi repositorio de Git y desplegar. Como marqué al principio del apartado, el proceso fue prueba y error. Tras varios commits, horas de intentos y pruebas, dí con los cuatro archivos necesarios para conseguir desplegar la API que haría las peticiones al modelo: Profile.txt, requirements.txt, el modelo con extensión .joblib y el script con extensión .py.

Con todo listo, la predicción y conexión con mi aplicación estaban listas.

5.3. Estructura general de la aplicación y control de estados

Como se ha indicado en el punto 4 de este informe, se ha intentado aplicar el patrón MVC en el trabajo. Al tener tres tablas en nuestra base de datos, es necesario crear tres modelos en nuestro programa. Estos han recibido los nombres de:

1. user.dart (usuario)
2. watch.dart (reloj)
3. auction.dart (subasta)

Los tres archivos siguen la misma filosofía:

1. Se declaran cada uno de los objetos.
2. Se estipula cómo se recibe y envía información a la base de datos.
3. Se declara cualquier función que trate los datos.

Por otro lado, se han definido las vistas y los controladores en el mismo script ya que Flutter requiere de ello. Como indicábamos en otros apartados, Flutter trabaja con widgets y estos necesitan que las funciones sean definidas para recargarlas cada vez que se producen cambios por diversas acciones, aunque esto lo hablaremos con más detalle en el apartado del manual de usuario.

Un aspecto importante de este trabajo es controlar muy bien los estados de cada reloj y subastas tras realizar una acción cualquiera. Por ello, se enumera a continuación todos los casos tenidos en cuenta:

1. Cuando un usuario crea un reloj, no puede crear dos relojes con un mismo watch nickname ya que luego van a depender de este campo a la hora de crear una subasta.
2. Una vez creado el reloj, el estado del reloj pasará a estar subido, lo que indicará que ni se encuentra en subasta ni nunca ha sido partícipe de una. Por tanto, el usuario podrá editar y/o eliminar el reloj sin ningún problema.
3. Cuando un usuario quiera crear una subasta, deberá crearla con un nombre de los relojes asociados a su cuenta. En el momento de su creación, el estado de la subasta pasará a activa y el estado del reloj pasará a estar en subasta. Por tanto, se deben bloquear los botones de editar y eliminar del reloj. En cuanto a la subasta, el creador de la subasta tendrá bloqueados los botones tanto de compra directa como el botón de puja, mientras que el botón de eliminar estará disponible siempre que no haya aplicado nadie a la subasta. En cuanto a otros usuarios distintos al creador, tendrán disponibles los botones de manera contraria a este.
4. Si el creador de una subasta elimina esta, el estado de la subasta pasará a finalizado y el estado del reloj pasará a disponible, volviendo a habilitar de nuevo los botones de edición y eliminación del reloj.
5. Si por la razón que sea la subasta terminase, el estado de esta pasaría también a finalizado, pero el estado del reloj pasaría a vendido y los botones seguirían deshabilitados.

Estos no han sido los únicos condicionantes a la hora de realizar acciones. Uno muy importante ha sido controlar bien las pujas de los usuarios. Siempre que usuario iba a realizar una puja o la compra directa del reloj, se han tenido en cuenta dos condicionantes:

1. El monedero del usuario debe ser mayor que la cantidad a pujar o la cantidad de compra directa.
2. El monedero del usuario debe ser mayor al sumatorio de todas las pujas realizadas por un usuario.

Refiriéndome al último condicionante, nos ponemos en el supuesto de que un usuario gane todas las pujas que aplica. Si ganase todas, debe poder pagar todas. De ahí que se haga la comprobación y, si no pudiese hacer frente a ello, se le avise con una alerta informativa explicando el porqué de todo.

6. Trabajos relacionados

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

7. Conclusiones y Líneas de trabajo futuras

Con todos los apartados anteriores explicados, llega el momento de realizar una valoración global del trabajo realizado. Es difícil valorarse de manera objetiva a uno mismo, pero creo que una actividad muy efectiva para seguir progresando tanto laboralmente como psicológicamente.

Como marcaba al principio de este informe, el trabajo realizado ha sido un verdadero quebradero de cabeza. En muchas ocasiones lo he comparado con la acción de intentar tirar una pared a cabezazos. Con esto último me quiero referir principalmente a lo difícil que ha sido para mí afrontar una aplicación de este nivel.

Es cierto que las herramientas utilizadas permiten realizar infinidad de cosas increíbles, pero creo que la curva de aprendizaje que carga a su espalda apenas tiene pendiente. Aun así, a base de esfuerzo y dedicación, se ha llegado a crear una aplicación operativa que responde a la mayoría de los objetivos definidos en este informe.

El proyecto puede ser escalado de mil formas, pues existen una gran cantidad de nuevas funciones que podría añadir esta aplicación. Si tuviera que marcar alguna en concreto, me quedaría con:

1. Inserción de nuevos tipos de subasta: aunque hemos definido la más común y creo firmemente que sería la más utilizada si hubiese más, podríamos tener en cuenta los tres tipos de subastas restantes explicados en la parte teórica de este informe.

2. Conexión con el entorno Google: muchas aplicaciones y sitios web permiten registrarse con la cuenta de Google, cosa que podría incluirse para aumentar el nivel de profesionalidad de nuestra aplicación.
3. Diseños más profesionales: la aplicación mostrada es una base de una aplicación que podría incluir diseños y efectos de cambios de vista más espectaculares, dando una experiencia de usuario mucho más satisfactoria.
4. Perfeccionamiento del modelo: este trabajo puede llevarse la mayor cantidad de tiempo a invertir, pues es fundamental que la predicción de precios sea lo más perfecta posible.

En conclusión, las líneas de mejora de esta aplicación son infinitas y tengo claro que se irán alcanzando con el aprendizaje tanto en mi mundo laboral como en el estudiantil, mundos que son un punto y seguido tras la finalización de este trabajo.