UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Reconocimiento Nutricional, Aplicación móvil de información nutricional por reconocimiento de imágenes

Presentado por Miguel Díaz Hernando
en Universidad de Burgos — 5 de julio de 2023
Tutor académico: Bruno Baruque Zanón
Tutor empresarial: Javier Melús Carruez

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. Bruno Baruque Zanón, profesor del departamento de Ingeniería Informática, área de Ciencia de la computación e Inteligencia Artificial.

Expone:

Que el alumno D. Miguel Díaz Hernando, con DNI 71297179P, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado Reconocimiento Nutricional, Aplicación móvil de información nutricional por reconocimiento de imágenes 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, 5 de julio de 2023

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. Bruno Baruque Zanón	D. Javier Melús Carruez

Resumen

Este documento aborda la creación de una aplicación para Android con el objetivo de mantener un control de los alimentos consumidos por el usuario, por medio de la detección de imágenes automática de la comida y la obtención de su información nutricional. La aplicación permitirá la toma de imágenes para su reconocimiento y análisis, devolviendo al usuario la información nutricional de la comida que se encuentre en las mismas.

La aplicación ha sido desarrollada en Unity para Android, e integra consultas a la API de Google Cloud Vision, comúnmente conocida también como Google lens, como una parte fundamental de la misma.

Frente a otras aplicaciones similares ya existentes en el mercado, esta cuenta con la ventaja del reconocimiento de imágenes integrado, por lo que proporciona una mayor comodidad a la hora de introducir la información de los alimentos.

Descriptores

Aplicación móvil, información nutricional, Unity, Android, reconocimiento de imágenes, cloud vision.

Abstract

This document covers the creation of an Android application with the objective of maintaining control of the food consumed by the user, through the automatic detection of images of the food and obtaining its nutritional information. The application will allow the taking of images for their recognition and analysis, returning to the user the nutritional information of the food that is in them.

The application has been developed in Unity for Android, and integrates queries to the Google Cloud Vision API, also commonly known as Google lens, as a fundamental part of it.

Compared to other similar applications already on the market, this one has the advantage of integrated image recognition, thus providing greater comfort when entering food information.

Keywords

Mobile app, nutritional information, Unity, Android, image recognition, Cloud Vision.

Índice general

Índice de figuras

Índice de tablas

Introducción

El entorno todavía emergente de la inteligencia artificial ofrece la oportunidad de crear aplicaciones originales basadas en estas nuevas tecnologías, pero también un rango nuevo de herramientas y soluciones que integrar en ideas o aplicaciones ya existentes para proporcionar un resultado mejorado o un proceso de uso o desarrollo más cómodo de las mismas.

La aplicación que se va a desarrollar, destinada a ayudar al usuario a controlar los alimentos que consume y ofreciendo información nutricional y el registro de los mismos, es uno de estos últimos casos. Existen varias aplicaciones similares comúnmente conocidas en el mercado con diferente enfoque y alcance del seguimiento[39, 68], pero una importante barrera de usabilidad es la necesidad de introducir manualmente los alimentos que se deseen registrar. Es aquí donde nuestra aplicación destaca ofreciendo algo único con respecto a otras similares, permitiendo una introducción de los alimentos mucho más rápida y cómoda a través del reconocimiento de imágenes integrado. El reconocimiento de imágenes es una tecnología ya establecida, y proporciona una nueva solución para este tipo de aplicaciones permitiendo en este caso identificar automáticamente los alimentos que se quieren registrar, en vez de introducir la información nutricional que habría que conocer de antemano, o buscarlos en una base de datos manualmente.

De esta forma, este proyecto se plantea como el desarrollo completo de una aplicación móvil como producto software con una utilidad concreta, pero además como un trabajo de investigación para ITCL motivado por el uso de estas tecnologías en futuros proyectos similares de la empresa.

Estructura de la memoria

- **Introducción:** Descripción del proyecto, estructura de la memoria y materiales adjuntos.
- **Objetivos del proyecto:** Descripción de los objetivos que pretende abarcar el proyecto.
- **Conceptos teóricos:** Puntos importantes que se mencionarán a lo largo de la memoria y que conviene tener claros.
- **Técnicas y herramientas:** Explicación de las técnicas, herramientas, metodologías y prácticas que se han seguido en el proyecto.
- **Aspectos relevantes del desarrollo del proyecto:** Aquellos detalles más interesantes del proyecto, curiosidades, dificultades y soluciones que se han ido afrontando a lo largo del desarrollo del mismo.
- **Trabajos relacionados:** Otros proyectos relacionados o similares.
- **Conclusiones y líneas de trabajo futuras:** Conclusiones sobre el proyecto y hacia dónde conduce el mismo en un futuro.

Anexos

- **Plan del proyecto:** Aspectos y estudios relacionados con la planificación del proyecto.
- **Especificación de Requisitos:** Requisitos extraídos de los objetivos del proyecto.
- **Especificación de Diseño:** Fase de diseño del software y diagramas relacionados.
- **Manual del programador:** Aspectos más relevantes relacionados con los ficheros fuente.
- **Manual de usuario:** Guía de uso de la aplicación.
- **Pruebas:** Resumen de las pruebas con usuarios realizadas.

Objetivos del proyecto

El objetivo principal del proyecto es el desarrollo de una aplicación funcional, aunque no para uso público o comercial. La aplicación permitirá la toma de imágenes y la realización de consultas sobre la información nutricional de los alimentos que se encuentren en las mismas. Para esto se requerirá el uso del reconocimiento de imágenes a través de Cloud Vision[23], lo que supondrá una ventaja destacable sobre otros productos con una finalidad similar.

Un objetivo secundario, pero tal vez más importante para ITCL, es la investigación y documentación de los aspectos técnicos derivados del uso de Cloud Vision y su integración en Unity, con todo lo que ello conlleva. Este aspecto tendrá junto a los elementos más ligados al desarrollo en Unity un peso importante en el apartado de técnicas y herramientas (3.4), y sobre todo en el anexo del Manual del programador.

2.1. Objetivos principales

- Desarrollar una aplicación que permita la identificación de alimentos en imágenes y la obtención de su información nutricional.
- Investigar y documentar el proceso de uso de Cloud Vision y su integración en Unity.
- Proporcionar un prototipo o punto de partida en el que basar futuros proyectos de ITCL desarrollados con estas tecnologías.

2.2. Objetivos técnicos

- Construir una aplicación para Android en Unity usando técnicas y elementos propios de los entornos móviles.
- Conseguir acceso y comunicarse con la API de Cloud Vision desde Unity.
- Cumplir con los anteriores objetivos de forma que el proyecto resultante sea fácilmente extensible y escalable, disponiendo de una buena documentación.

2.3. Objetivos personales

- Afianzar y mejorar mis conocimientos y mi manejo de Unity, herramienta en la que ya he adquirido un nivel de uso medio y que me gustaría llegar a dominar.
- Conocer técnicas específicas del desarrollo móvil, concretamente en Unity como motor de videojuegos.
- Entender el proceso completo de vida de un proyecto software, desde la idea hasta el producto final siendo responsable de todas sus etapas y elementos.
- Acostumbrarme al uso de metodologías ágiles y sus ritmos de producción en este tipo de proyectos de desarrollo de aplicaciones o videojuegos.
- Aplicar en un caso práctico, real y completo, todos los conocimientos adquiridos a lo largo de los años en el estudio de la titulación de Ingeniería Informática en diversas materias.

Conceptos teóricos

Aunque el proyecto consiste en una aplicación para móvil, y lo lógico parecería producirla en Android Studio[10], la decisión es utilizar Unity, como se explicará en el capítulo de Técnicas y herramientas.

Unity, aunque en este caso se vaya a utilizar para crear una aplicación móvil que no es un juego, no deja de ser un motor de videojuegos, y como entorno de desarrollo de videojuegos en tiempo real, supone una herramienta completamente distinta a otros entornos de programación clásicos. Por ello, es especialmente importante éste apartado para comprender los conceptos básicos y más habituales del desarrollo en Unity.

3.1. Motor de videojuegos

Un motor abstrae el código del juego en el hardware en el que se pretende hacer funcionar, y provee de funciones clave, como renderización, animación, simulación de físicas o ingreso de comandos del usuario. De esta forma, los motores de videojuego suponen un entorno de desarrollo único que es necesario conocer, y a continuación se verán algunos de sus elementos propios más relevantes, y en específico de Unity.

Conceptos básicos

GameObject[55]

El concepto básico de desarrollo para videojuegos es el GameObject, que se puede asimilar de cierta forma a un objeto en un entorno clásico de programación orientada a objetos. La aplicación se compondrá de un sistema de GameObjects que coexisten en un mundo o escena[3.1] en una

ubicación concreta definida por unas coordenadas. Este GameObject cuenta con una serie de componentes asignados, que pueden ser elementos básicos proporcionados por Unity, o scripts propios creados por el programador. Estos componentes son los que definirán sus características y su comportamiento en todo momento, desde su ubicación, forma, color o texturas hasta sus reacciones o interacciones con otros objetos o con el usuario, pudiendo modificar a su vez la información de otros componentes de su mismo GameObject o de otros con los que interactúen. Todos los elementos que podemos ver en nuestra aplicación en ejecución son un GameObject, desde el fondo de la interfaz, los botones, cada texto, e incluso los que no vemos, como el MainMenuManager (manager general de la escena) o la Main Camera[3.1].[fig.3.1]

Escenas[58]

Una escena es el mundo o contexto en el que un conjunto de GameObjects[3.1)] existen, y son el primer nivel de modulación de la aplicación. El concepto de distintas escenas como entornos separados viene derivado de las limitaciones hardware de los dispositivos, ya que si se fuera a construir toda la aplicación en un solo escenario que tendría que estar en memoria, ésta se sobrecargaría en poco tiempo, o supondría una limitación inaceptable en cuanto al tamaño de la aplicación. De esta forma, la aplicación se compondrá de distintas escenas que se irán cargando o descargando según el usuario navegue por ella, aunque esto no nos limita a trabajar únicamente con una escena en cada momento. Es posible la interacción de objetos entre escenas, y la coexistencia de más de una escena en un mismo instante mediante técnica conocida como escenas aditivas[50] que es bastante común, y aunque no se ha necesitado en éste proyecto, se profundiza algo más en la misma en el capítulo de Técnicas y Herramientas[4.9].

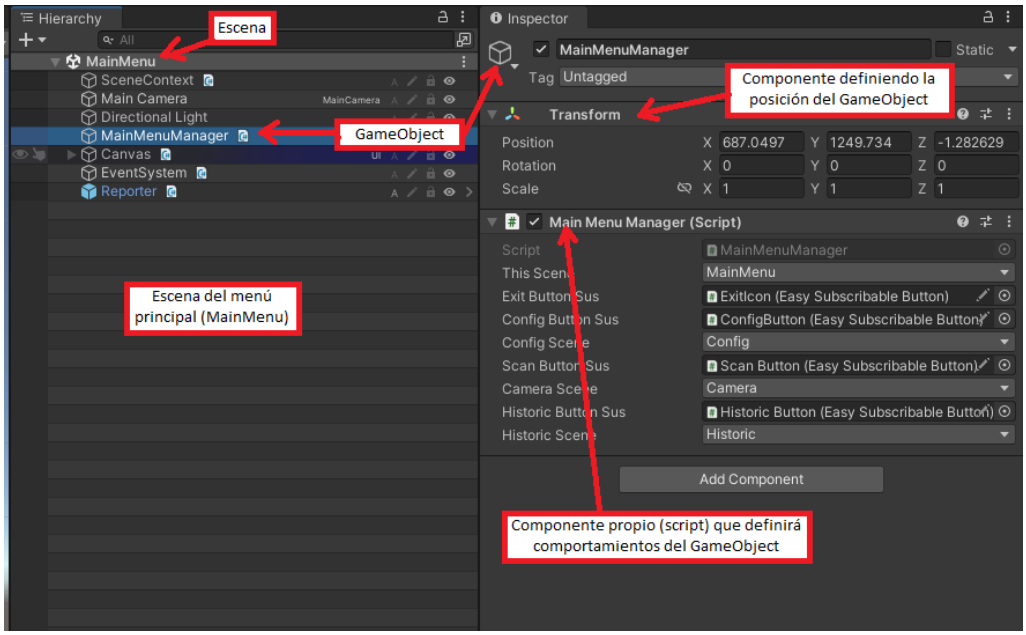

Figura 3.1: Vista en el editor de Unity de una escena con sus objetos (Hierarchy) y el detalle de un objeto (Inspector).

Prefabs[56]

El sistema de prefabs de Unity permite la creación de nuevos GameObject con una serie de componentes establecidos, con unas características ya definidas. Esto se hace en base a una definición de un GameObject que no se encuentra en ninguna escena, y que actúa como una plantilla para la creación de instancias de ese prefab en la escena.

Si el GameObject se puede comparar un entorno clásico de programación orientada a objetos con un objeto, el prefab sería una clase en este símil. Pero no conviene apegarse mucho a esta idea ya que las diferencias de los entornos son sustanciales, ya que ni es necesaria la existencia de un prefab para la creación de objetos, ni la definición del prefab limita al objeto instanciado, que puede posteriormente añadir, eliminar o modificar sus componentes.

Ejemplos de este concepto serían casi todos los botones de la aplicación, creados a partir del mismo prefab de botón para tener una imagen, forma y fuente de letra idénticas, además de una serie de componentes fijos inherentes a un objeto botón interactuable.

Interfaz de usuario

La interfaz de usuario es tratada como un elemento especial en Unity, que se ubica siempre dentro de un mismo objeto en cada escena llamado canvas, y cuenta con sus propios tipos de GameObjects específicos. Éste canvas es un objeto de dos dimensiones, independientemente de la naturaleza de la aplicación o de la escena, y representa la forma en la que se verán los elementos de la interfaz superpuestos al resto de la escena de la misma forma que aparecerán luego en una pantalla.

Cámara[52]

Un componente indispensable en Unity es el de cámara. Al añadir este componente a un objeto, éste es tratado como un objeto especial, que actuará como la ventana del usuario para visualizar el mundo del juego. Todas las escenas se crean por defecto con un objeto llamado cámara que consta del componente homónimo, lo cual es indispensable para que la escena tenga un sentido, y una escena puede disponer de varias cámaras para posibilitar el cambio de puntos de vista de la escena, efectos de pantalla dividida, etc. También sería el medio para desplazarse visualmente por la escena si la visualización del usuario cubriese solamente un fragmento de la misma, dado que se le puede aplicar al objeto cámara un movimiento en función de la entrada del usuario, por ejemplo.

En este proyecto no contamos con este tipo de escenas tan grandes, que requerirían un movimiento de la cámara, si no que todas las escenas albergan lo que será una pantalla de la app. Por lo tanto habrá siempre una y solo una cámara, y permanecerá fija en la escena, así que no debemos preocuparnos por éste elemento más allá de su presencia y su correcta configuración inicial.

Desarrollo para Android

Al tratarse de una aplicación para Android, se diseña teniendo en mente un hardware y un sistema operativo específicos. Esto supone por un lado una facilidad al no tener que preocuparse por todos los sistemas y formatos en los que se podría ejecutar, pero por otro lado una complicación, puesto que hay que tener en cuenta las complejidades específicas del desarrollo para móvil.

Hardware específico

Teniendo la seguridad del dispositivo en el que se va a ejecutar la aplicación, un teléfono móvil, se puede confiar en la existencia de una cámara trasera en el hardware. Esto facilita el desarrollo del código puesto que no son necesarias diversas comprobaciones y consideración de posibilidades en cuanto a la existencia de cámaras, su selección en caso de existir varias, su posible desconexión, etc. También es relevante la certeza de una pantalla táctil, que se ha tenido en cuenta para el diseño de un sistema interno de zoom de la imagen. Finalmente se consideró un desvío considerable de las funcionalidades necesarias, pero una función de zoom basada en clicks en pantalla es sólo posible si se cuenta con pantalla táctil. Otro aspecto a mencionar es el diseño de pantallas. A pesar de la diversidad de resoluciones y dimensiones de las pantallas de los teléfonos móviles del mercado, se puede confiar en unas proporciones de pantalla aproximadas y una orientación vertical, lo cual limita, pero a su vez facilita el diseño de las interfaces.

Permisos del sistema[12]

Este detalle puede parecer una trivialidad, pero nada más lejos de la realidad. Si no se tiene en cuenta, o no se configuran bien, ni una sola parte de la aplicación podría funcionar. Para prácticamente toda aplicación móvil es necesario acceder a algún sistema de almacenamiento de datos interno del teléfono, en este caso en concreto para guardar la base de datos así como la configuración de la aplicación, donde se definen aspectos como el idioma seleccionado. Además en nuestra aplicación es indispensable el uso de la cámara, por lo que también se necesitan sus permisos.

La solicitud de permisos está configurada en el inicio de la aplicación, en la mencionada primera pantalla de carga responsable de gestiones de éste tipo, entre otras. Los permisos se piden al usuario la primera vez que abre la aplicación y, puesto que son críticos, no se avanza de la primera pantalla de carga hasta que no se cuenta con ellos.

Rendimiento

En un videojuego común, o incluso una aplicación cualquiera para PC, lo más común es darle una gran importancia al aspecto visual, requiriendo mantener unos fps altos en todo momento en base a unos requisitos hardware previos de tarjetas gráficas y procesadores de gran potencia, a través de la eficiencia extrema del código o normalmente una mezcla de ambos.

Aunque esta aplicación no requiere un gran rendimiento gráfico, ya que casi todas las escenas se componen únicamente de imágenes estáticas, pero tampoco significa que se pueda descuidar éste aspecto. El rendimiento gráfico puede ser importante a la hora de mostrar las imágenes de la cámara del dispositivo, pero más relevante puede resultar el rendimiento en cuanto a procesador. Actualmente los procesadores de Android más comunes son considerablemente potentes, pero conviene asegurar la correcta ejecución de la aplicación en todos los tipos de procesador. No podemos entonces en este caso confiar en un holgado margen en cuanto al procesador, y tenemos entonces que tomar esto en cuenta en el código y en el diseño de las escenas, buscando un nivel aceptable de optimización.

También cabe tener en cuenta a este respecto la existencia de diferentes arquitecturas de Android, para las que podría haber aspectos específicos a considerar. Más información al respecto se expondrá en el capítulo de Aspectos Relevantes[5.1].

3.2. Lenguaje de programación en Unity

Como ha quedado ya claro, esta aplicación va a ser desarrollada en el motor Unity3D[60]. Este motor es uno de los más utilizados en el mercado por su versatilidad a la hora de crear diferentes productos, abarcando desde sencillas demostraciones hasta complejos videojuegos, pasando por simuladores, películas de animación, etc.

Como todos los motores, Unity se sirve de un lenguaje de programación para describir el comportamiento de los objetos del juego, y Unity originalmente contaba con dos opciones de lenguajes: C#[8] y UnityScript, un lenguaje propio cuya intención era asimilarse a JavaScript[29]. Sin embargo, la gran mayoría de desarrolladores de Unity han utilizado siempre el primero, debido a su recomendación por parte del propio Unity, y por encontrarse en éste la gran mayoría de ejemplos previos. Esto causa un efecto de retroalimentación que enmarca a C# como la opción más viable para el desarrollo en Unity. Por estos motivos, en agosto de 2017, el soporte a UnityScript fue oficialmente eliminado de Unity[65], dejando a C# como la única posibilidad para el desarrollo en Unity.

Pese a que C# es un lenguaje muy similar a Java y C++ y, como ellos, está orientado a objetos, Unity utiliza un tipo de programación diferente, orientada a componentes[42]. Una explicación más detallada sobre este tipo de programación se puede encontrar en el apartado de diseño de los anexos, y

en los conceptos básicos de este mismo capítulo [GameObject: 3.1], [Escenas: 3.1].

C# en Unity[61]

C# ha sido desde la creación de Unity su lenguaje de programación por excelencia, ya que se basa en una implementación del framework .NET, creado por Microsoft[37]. Enfocarse hacia el desarrollo con este framework fue de las primeras decisiones que se tomaron por el equipo de Unity, por lo que la adecuación de C# era indiscutible.

A lo largo de los años, Unity ha ido alcanzando varios hitos clave en la puesta al día de sus integraciones, como en 2016 con la inclusión de su propio framework de compilación y tiempo real, o en 2018 con una serie de grandes actualizaciones en el motor, convirtiendo a ésta versión en la más popular durante mucho tiempo. Este hecho es relevante puesto que aún a día de hoy hay proyectos en activo que se basaron en la versión de Unity 2018, además de que una inmensa cantidad de discusiones y ejemplos en foros u otras plataformas no oficiales corresponden a esta versión. Se ha encontrado justamente este problema con respecto a las librerías de SQLite, indagando más al respecto en el manual del programador.

Un nuevo hito se ha alcanzado recientemente en las versiones de 2020 y 2021, cuando nuevamente se actualizan las versiones de C# y las APIs .NET, por lo que se espera que la versión 2021, en la que se realiza el proyecto, se convierta en la versión principal durante otra larga temporada.

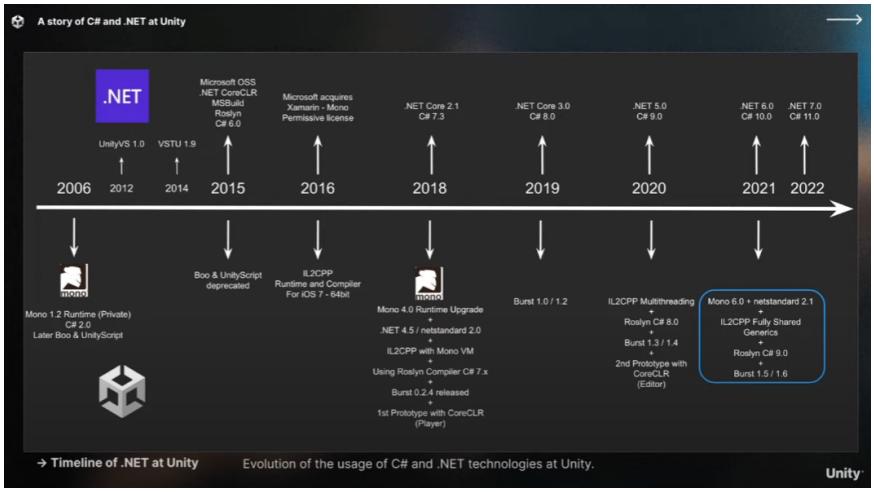

Figura 3.2: Timeline de C# y .NET en Unity.

3.3. Interfaz de programación de aplicaciones

Conviene conocer en profundidad el concepto de API ya que la API Cloud Vision[23] de Google se trata de una pieza fundamental del proyecto.

Una API, o application programming interface, es una sección de código que permite a dos aplicaciones distintas comunicarse entre sí, comportándose como un intermediario entre ambas y facilitando el intercambio de datos y funcionalidades. En nuestro caso, estaremos intercambiando datos entre nuestra aplicación de Unity y la aplicación de reconocimiento de imágenes de Google.

Para este intercambio, nuestra aplicación debe adecuarse a las estructuras de datos que requiere la API en el momento del envío de esos datos, y a su vez es necesaria una correcta interpretación de los datos recibidos. Para esto, Google cuenta con una documentación[24] que sirve de guía básica para la comunicación con Cloud Vision, que ha sido complementado con algunos ejemplos más específicos dentro del contexto de Unity, encontrándose el más relevante en un repositorio público de git[6] detallando las clases y estructuras de datos concretas de la API. Se profundizará en este tema en el anexo del Manual del programador.

3.4. Visión artificial

Aunque no se va a trabajar directamente esta materia, puesto que queda delegada a la herramienta de Google Cloud y para eso se utiliza la API, conviene también conocer la base de ésta tecnología y sus conceptos básicos[47].

La visión artificial es una técnica que se basa en el procesamiento de imágenes del mundo real con el fin de producir una información sobre las mismas que pueda ser tratada por un ordenador. Su objetivo final es conseguir el desarrollo de estrategias automáticas para el reconocimiento de patrones complejos en imágenes de cualquier ámbito.

Es una forma de inteligencia artificial, puesto que se basa en este concepto entrenando un algoritmo con multitud de imágenes, para que luego éste sea capaz de reconocer diferentes conceptos, objetos o información en cualquier imagen proporcionada.

Actualmente muchos campos distintos se benefician de esta tecnología, como el de la robótica, que se sirve de estos modelos para que un robot pueda

por ejemplo reconocer objetos de forma autónoma para interactuar con ellos. También en el campo de la conducción y la seguridad vial, introduciendo la visión artificial en el reconocimiento de los elementos de tráfico reales para la ayuda a la conducción por ejemplo, o en la reconstrucción en 3D a partir de imágenes en 2D, para reproducir las dimensiones reales de los elementos de una fotografía. Algunos de estos campos se trabajan de hecho en ITCL, proporcionándome una experiencia de primera mano al respecto.

Técnicas y herramientas

En este capítulo se presentan las técnicas y herramientas que se han escogido para el desarrollo del proyecto, los motivos de su elección y las alternativas que se consideraron.

4.1. Metodologías

Scrum

Inicialmente, un poco por defecto, scrum[45] se supuso como la elección obvia para la gestión. Se adecuaba muy bien a la duración y naturaleza de un proyecto corto con unos límites bien definidos. También es la metodología usada generalmente en ITCL por lo que, tras realizar las prácticas en ésta misma empresa, ya estaba familiarizado con ella.

Kanban

Al poco tiempo, tras buscar una plataforma adecuada para realizar la gestión mediante scrum, las opciones al respecto se valoraron algo complicadas, empezando a consumir un tiempo mayor al que se le quería dedicar al respecto. Se empezó entonces a considerar la gestión mediante kanban[35], que de igual manera se adecuaba perfectamente a las necesidades del proyecto, me resultaba conocido por haber trabajado antes con él y también se usaba en la empresa, aunque en menor medida, un sistema similar con tarjetas y tablero.

Scrumban

Kanban parecía una mejor opción en cuanto a versatilidad y flexibilidad, ya que al tratarse de un proyecto de una sola persona con poca experiencia, se esperaba que las estimaciones no fuesen siempre acertadas, hubiese complicaciones imprevistas, o espacio para mejoras adicionales.[4]

Pero aún así no se quería abandonar la organización en sprints. Fue por esto, tras un corto periodo de investigación, que se descubrió y decidió adoptar la metodología scrumban[1], que permite exactamente combinar ambos aspectos deseados de las dos metodologías. Scrumban se originó como un método de transición para equipos de un sistema al otro, pero en algunos casos resultó ser más eficiente que ambos.

Éste es el que finalmente se decidió para gestionar el proyecto.

4.2. Patrones de diseño

A pesar de que Unity se basa en la programación orientada a componentes, algunos patrones de diseño de la programación orientada a objetos[18] y la funcional pueden aplicarse también en este entorno.

Managers

Es habitual en el desarrollo en Unity tener scripts de carácter general que se asocian como componentes a objetos vacíos. Estos managers se encargan de gestionar el funcionamiento global de la aplicación o de una escena, como es el caso de todas las escenas de la aplicación, que cuentan con su manager de escena correspondiente responsable de las interacciones entre objetos de su escena u otros comportamientos generales en la misma[ver Fig.[3.1]. También pueden utilizarse para ofrecer funciones de utilidad, como la conexión con la base de datos o las APIs. En ocasiones como esta última, el manager no es más que otra forma de llamar al patrón fachada.

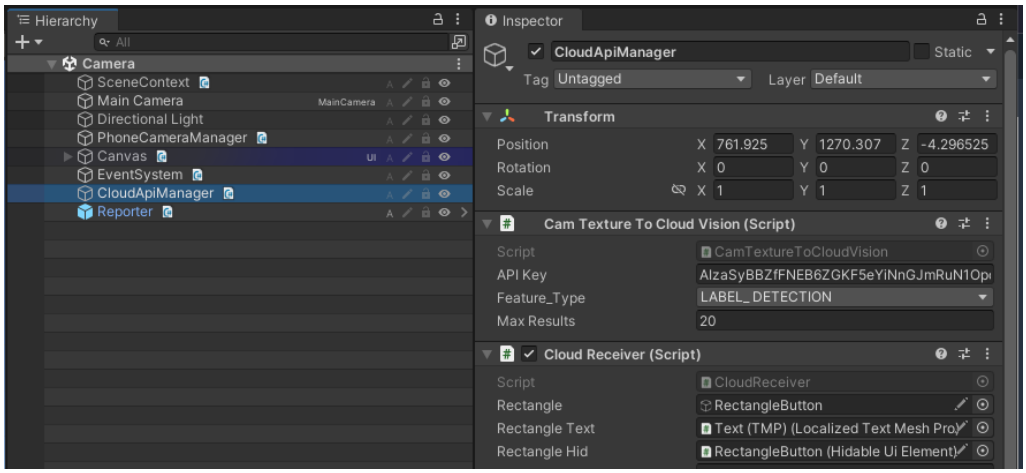

Figura 4.1: Escena de la cámara, que cuenta con un manager encargado de gestionar la API y sus llamadas, aparte de su manager general de la escena.

Singleton

El Singleton es un patrón utilizado cuando tenemos que asegurar la existencia de una y solo una instancia de una clase, como es el caso de los managers mencionados anteriormente. En Unity tienen otra utilidad, y es que permiten transferir datos a través de escenas, lo cual se aprovecha en la aplicación para mantener información como la sesión actual.

```
/// <summary>
/// Singleton that controls the Loading Screen.
/// </summary>
[RequireComponent(typeof(CanvasGroup))]
1 asset usage  5 usages  Miguel Díaz
public class LoadingScreen : Singleton<LoadingScreen>
{
```

Figura 4.2: Definición del componente LoadingScreen como singleton, puesto que solo debe existir una pantalla de carga.

Cadena de responsabilidad

Cuando una escena empieza a ser compleja debido a la cantidad de elementos que contiene, muchos de ellos necesitando ser referenciados en un script, referenciarlos todos en un mismo manager general de la escena puede resultar confuso y sobrecargado.

La solución es la cadena de responsabilidad, que delega las funcionalidades en más managers a distintos niveles, de forma que queden encapsuladas, por así decirlo. En el proyecto se ha aplicado por ejemplo con las diversas ventanas de la escena de la cámara, o para la pantalla de gráfica del histórico, que cuentan con un manager aparte gestionando sus funciones y referencias.

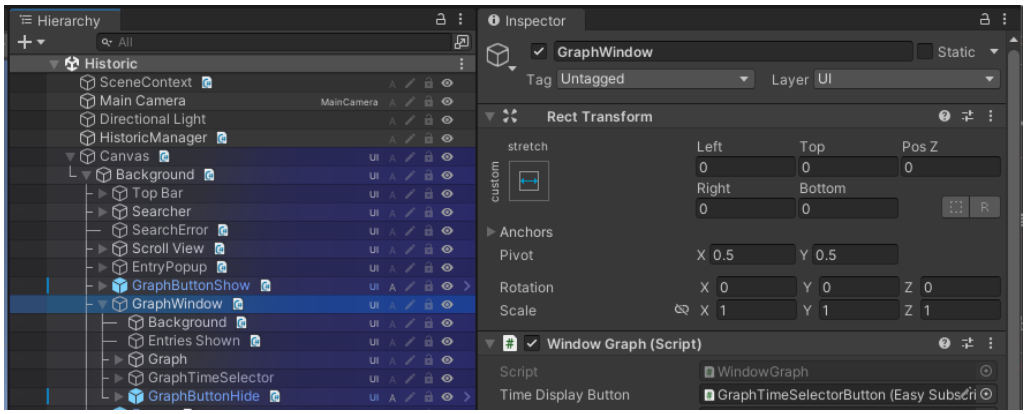

Figura 4.3: Componente encargado de la gráfica del histórico con su propio manager (Window Graph), desglosando funciones y referenciado en el manager general de la escena de histórico (HistoricManager).

4.3. Herramientas de gestión

Cuando se estaba considerando usar scrum, la principal candidata para su aplicación era Pivotal Tracker[36], debido a su uso en el pasado en ITCL. También conocía Zenhub[69], pero fue rápidamente descartada debido a su enfoque hacia GitHub, que no iba a ser la herramienta de control de versiones del proyecto.

Al pensar luego en kanban, la primera opción fue Trello[64], ya que contaba con cierta experiencia en su uso. Se consideró también Jira[3], que es una de las herramientas usadas actualmente en ITCL. Finalmente se optó por Trello por su mayor simpleza y agilidad.

4.4. Control de versiones

En este aspecto no hubo margen para opciones ni discusión, al tratarse de un proyecto para la empresa, se debía alojar en Bitbucket[2], herramienta estándar en la empresa, en la cual disponen de servidor privado. Por lo tanto, Git[7] será el sistema utilizado en el control de versiones del proyecto, ubicado en un repositorio privado de Bitbucket.

En cuanto al cliente, se utilizó el propio Rider[34], que integra estas herramientas en su entorno, así como GitKraken[20], cliente con el que estaba más familiarizado para gestionar la clonación de repositorios, la definición de archivos .gitignore, etc.

No obstante, para la entrega y presentación del proyecto, se requería la disponibilidad del repositorio conteniendo los archivos fuente y la visualización del ritmo de trabajo mediante los commits, de modo que una vez acabado el proyecto fue necesario trasladar el repositorio[46] a una copia en GitHub[19].

4.5. Motor gráfico

A pesar de que en ITCL se está comenzando a utilizar Unreal[16], es Unity el punto fuerte de la empresa y del departamento en el que realizo el proyecto, además de contar yo con experiencia en el mismo.

Por esto, como se menciona brevemente al inicio de los Conceptos teóricos, no tendría sentido para ITCL el uso de Android Studio[10], de manera que el uso de Unity es un requisito por parte de la empresa. Tampoco hay que preocuparse por la adecuación, puesto que los puntos fuertes de Android Studio frente a Unity en cuanto a aplicaciones móviles recaen sobre:

- El acceso a componentes hardware de Android, aspecto que Unity cubre también sin problema, gracias a la integración que proporciona su módulo de Android Build Support.
- Soporte integrado de elementos de interfaz de usuario nativos de Android, lo que tampoco es un impedimento ya que cuento con experiencia en el manejo de interfaces en Unity.

Es destacable además, que en la actualidad una gran mayoría de juegos desarrollados para móvil se producen con Unity, y aunque ésta aplicación no

vaya a ser un juego, demuestra la gran adaptación de un proyecto en Unity al entorno móvil.

Por estos motivos, no se planteó en ningún momento el desarrollo en otro entorno distinto, siendo además la premisa de trabajar con una empresa el acceso a la experiencia y recursos previos de la misma y de mis compañeros de trabajo. También es relevante considerar que Unity consta de una enorme cantidad de plugins, tanto de pago como gratuitos, lo cual supone una gran ventaja a la hora de realizar cualquier tipo de proyecto.

4.6. Entorno de programación

El entorno más común y más usado en Unity es Visual Studio[38], pero personalmente nunca me ha gustado, debido a su complejidad excesiva derivada de su integración de más herramientas de las necesarias. Debido a ésta experiencia negativa, Visual Studio quedaba relegado al final de las opciones viables para el IDE (Integrated Development Enviroment).

Rider

Convenientemente, Unity introdujo en 2017 la integración con Rider[34], un IDE de Jetbrains[32] inspirado por el propio Visual Studio pero basado en la plataforma IntelliJ[31] y ReShaper[33], aportando una mayor simpleza y elegancia. Ésta fue la opción recomendada por los compañeros con más experiencia y la elección final, y demuestra no tener nada que envidiar a Visual Studio.

Datagrip

Otra IDE de Jetbrains utilizada ha sido Datagrip[30], diseñada para trabajar con bases de datos SQL. A causa del uso de un plugin para la base de datos, y su entera programación integrada en el código mediante Rider, ésta se utilizó tan solo brevemente para la obtención automática de los diagramas de la base de datos del proyecto. Aún así resultó ser efectiva y fácil de usar, aumentando mi apoyo por los productos de Jetbrains.

4.7. Plugins de Unity

En el desarrollo de un proyecto de Unity3D, las herramientas más importantes que se pueden incorporar al proyecto son los plugins o assets externos.

Éstos se pueden obtener en la Asset Store[51], o pueden ser paquetes creados por el propio desarrollador u otros desarrolladores. Estos plugins pueden tener una gran variedad de funcionalidades, desde librerías de utilidades hasta modelos y texturas.

WhateverDevs[66]

Los paquetes de recursos más importantes del proyecto son precisamente una creación de un compañero desarrollador del departamento, en colaboración con otro exmiembro de la empresa. Se trata de una recopilación de funcionalidades añadidas a Unity en forma de scripts que añaden utilidades diversas para facilitar el trabajo con el desarrollo de código, el manejo de datos, los componentes de los objetos, los logs de la aplicación o incluso aspectos del editor de Unity.

En el proyecto he usado varios de estos paquetes, aportando a las funcionalidades básicas de los objetos (core), a la localización, al manejo de escenas, a la inyección de dependencias...

Es un recurso open source, aunque usado apenas en el entorno del departamento. Aún así, ha sido y seguirá siendo de gran ayuda para el desarrollo en Unity, por la gran comodidad que proporciona.

Enhanced Hierarchy

Enhanced hierarchy[44] es un plugin muy elaborado, que aporta a la interfaz de Unity una mayor claridad y elegancia, y un acceso más rápido y claro a la información más relevante y usada de los objetos en escena. Es fácil de añadir al proyecto ya que no requiere ningún tipo de configuración, y aporta una gran ayuda no solo a la hora de construir escenas, si no también al depurar y debuggear.

Console Pro

Otro paquete aportando a la interfaz del editor[14], que proporciona un mejor desglose y visualización de la consola de logs del editor de Unity. Tanto éste como el anterior plugin son de pago, pero no se incluyeron en el apartado de viabilidad económica al estar comprados ya hace tiempo por los miembros del equipo, que fueron quienes me facilitaron dichos plugins.

Rider

A pesar de que ya se menciona como IDE, cabe destacar que Rider añade al proyecto de Unity un pequeño plugin que le permite conectarse a herramientas como el debbuger o la consola.

Textmesh Pro

Textmesh Pro[59] surgió como un plugin gratuito independiente, pero fue adquirido por Unity y ahora forma parte de las bases de Unity, siendo mantenido y desarrollado por su equipo. Aunque aún se pueden usar los componentes de texto anteriores más básicos, no se recomienda su uso, y permanecen solamente por motivos de soporte a proyectos antiguos.

Textmesh Pro es una herramienta que mejora la renderización de los textos en Unity, haciéndolos mucho más nítidos y con más opciones de personalización.

Animnated Loading Icons

Un sencillo plugin gratuito[17] que proporciona diversos iconos animados, principalmente para procesos de carga o espera. Se ha usado evidentemente para darle un mejor aspecto a la pantalla de carga del proyecto.

UX Flat Icons

Un recurso que facilita una serie de iconos 2D para la composición de pantallas e interfaces[13]. Se ha utilizado para diversos botones de la aplicación para conseguir una interfaz más intuitiva y amigable.

White & Black GUI

Otro paquete de iconos para interfaces[15], que se ha usado para cubrir alguna carencia específica del anterior.

Unity Native Gallery

Un plugin para Android e iOS en Unity[67] que permite un fácil acceso a la galería de fotos y vídeos del teléfono. Se ha usado a la hora de tomar las fotos de la comida, pera guardarlas en la galería de forma simple, pero se añadió pensando en la escalabilidad, en futuras funciones que puedan recuperar esas fotos para mostrarlas dentro de la aplicación.

DOTween

Un motor de animación para Unity enfocado a mejorar el tweening, esto son las imágenes generadas en una animación entre puntos clave de la misma. A pesar de contar con pocas animaciones en el proyecto, su uso ha sido recomendado por compañeros de trabajo.

4.8. Herramientas adicionales

SQLite

Una librería compatible con lenguajes C que contiene un pequeño pero completo motor de base de datos SQL, elegida como la herramienta para crear y gestionar la base de datos local del proyecto.

SQLite Viewer

Al implementar y depurar la base de datos y su código, se puede volver algo tedioso e incluso complicado comprobar la causa de los errores sin poder ver directamente el contenido de la base de datos. Situación que se daba precisamente al probar la base de datos sobre la aplicación en marcha, no pudiendo distinguir fácilmente si el origen estaba en la construcción de objetos con Unity, en la base de datos, o de qué tipo era el error de SQL. Esto se solucionó con SQLite Viewer[41], una sencilla herramienta sobre navegador que permite la visualización del contenido de una base de datos (extensiones .s3db, .sqlite, .db entre otras), facilitando enormemente el proceso de detección y corrección de errores.

Google Cloud prueba

En el proceso de investigación y desarrollo relacionado con la API de Cloud Vision fue de gran ayuda el uso de esta función[28] que permite testear el envío de imágenes a Cloud Vision para obtener el resultado que ofrece la API sobre las mismas. La posibilidad de visualizar los datos recibidos y su formato resultaron de gran relevancia en el desarrollo de los scripts encargados de comunicarse con la API.

Postman

Una plataforma que permite y hace más sencilla la creación y el uso de APIs. Es un cliente http que nos da la posibilidad de testear 'http requests'

a través de una interfaz gráfica de usuario, visualizando de forma clara la respuesta de esa petición http y sus datos[43]. Ha sido de gran ayuda a la hora de actualizar las comunicaciones http al nuevo patrón de Unity[5.2].

4.9. Técnicas y herramientas descartadas

Escenas aditivas

Frente a las escenas individuales que se han planteado en los conceptos teóricos[3.1], se menciona una técnica llamada escenas aditivas[50]. Esta técnica permite cargar una escena sobre otra, uniéndolas y mostrando los objetos de ambas. Esta técnica se suele usar cuando se quiere contar con una escena maestra en la que estarán los managers, la cámara (en muchos casos de realidad virtual) y otros ajustes, y añadirle en cada situación otras escenas de distintos escenarios o interfaces según sea necesario.

Esta técnica se valoró para la escena de la cámara, creando una escena con la interfaz independiente de las imágenes de la cámara. Finalmente se consideró añadir una complejidad por encima de las necesidades del proyecto, pero se sigue teniendo en cuenta para futuras versiones o el posterior proyecto similar de ITCL que sí podría implementar realidad aumentada.

Realidad aumentada

También se planteó como una posibilidad al inicio del proyecto aplicar realidad aumentada[57] al tomar la foto de la comida, y para mostrar la información sobre la misma representada sobre el entorno en 3D. Aunque prometía una gran atracción visual y buena utilidad como ejercicio práctico, fue descartada por la poca relevancia que aportaría al concepto de la aplicación, siendo solamente eso, llamativo, y no realmente útil.

SQLite Browser

Se consideraba como alternativa previa a SQLite Viewer[4.8]. Como éste, permite visualizar el contenido de bases de datos SQL, pero demostraba una complejidad inicial de instalación y configuración que desaparecieron al descubrir SQLite Viewer, por lo que SQLite Browser[5] fue entonces descartado.

Aspectos relevantes del desarrollo del proyecto

Este capítulo recoge los aspectos más interesantes del desarrollo del proyecto, así como los principales problemas que se encontraron, y cómo se abordaron y resolvieron.

5.1. Proyecto para Android

Comenzando el proyecto

El Unity Hub[53] es una aplicación independiente de Unity pero muy necesaria para su uso, sirviendo como punto de acceso y configuración a los distintos proyectos. Es el sitio también donde descargar nuevas versiones del software de Unity, y por defecto al hacerlo se descarga una versión orientada a una aplicación para Windows.

Si queremos sin embargo, como es nuestro caso, compilar una versión para Android, es necesario descargar los paquetes y librerías necesarios para esa versión de Unity. Esto se hace desde el mismo Unity Hub en la sección de instalaciones, seleccionando la versión de Unity en la que queremos trabajar y habilitando en la misma los componentes necesarios para Android. De esta forma contaremos en nuestro proyecto no solo con la posibilidad de compilarlo para Android, si no también invocar funciones personalizadas nativas de C o C++ de forma directa en nuestro código C#[54].

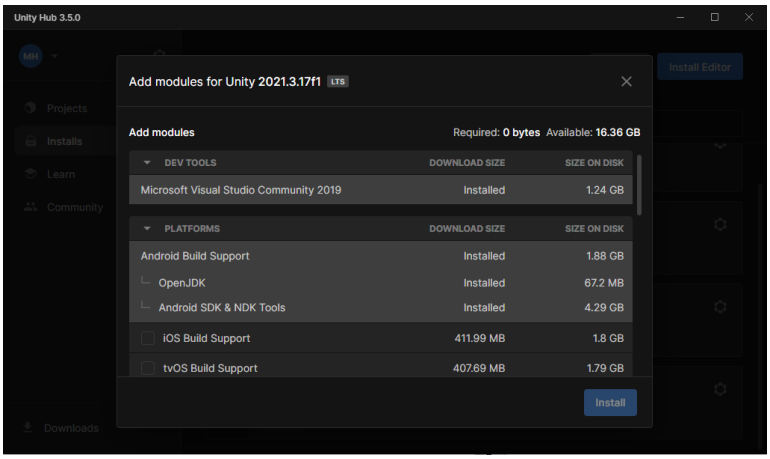

Figura 5.1: Unity Hub Android Setup

Arquitecturas

Como se menciona brevemente en los conceptos teóricos[3.1], los teléfonos móviles Android existen actualmente en 4 variedades de arquitecturas[9]; ARM, ARM64, x86 y x86_64. Tanto ARM como x86 son para procesadores de 32 bits, los cuales están desactualizados y son muy ratos de encontrar a día de hoy. Lo más común en los móviles Android que podemos encontrar en la actualidad son procesadores de 64 bits, más concretamente con la arquitectura ARM64[11]. En principio se trata de un detalle que no tiene por qué afectar a nuestro proyecto, pero es conveniente conocer bien estos conceptos al desarrollar para Android ya que podría resultar relevante.

Tal es el caso de la integración de SQLite, cuyos ejemplos para Unity están la mayoría desactualizados y tan solo constan con una librería para ARM de 32 bits. Los primeros intentos de integración resultaron por lo tanto en diversos errores, que se fueron depurando hasta localizar su origen en las librerías, provocando así la investigación necesaria al respecto de las arquitecturas, y la posterior resolución de los errores de integración al encontrar las librerías necesarias para dichas arquitecturas. Se puede apreciar esta complicación en el progreso del sprint 5 en el anexo del Plan de proyecto, con su tarea relacionada (Corrección de los errores de conexión con la base de datos) se prolongó más allá de lo planificado.

En la versión final SQLite cuenta con librerías para la integración con 3 de las 4 arquitecturas, faltando tan solo la de x86_64 debido a no haber sido posible encontrarla. De este modo se minimiza la posibilidad de cualquier error de la base de datos relativo a este aspecto.

5.2. Comunicación www

La interacción con servidores externos a través de la red es una parte vital de la aplicación, y es uno de los aspectos en los que más tiempo se invirtió hasta conseguir un buen resultado.

El problema se planteaba por la depreciación del tipo de objeto www de Unity, que era el encargado de establecer peticiones y obtener respuestas http, y su sustitución por el similar UnityWebRequest[49]. Este nuevo tipo requiere de un código bastante distinto y más complejo que el de su predecesor, contemplando más parámetros y opciones.

Al ser un cambio relativamente reciente (se deprecó a lo largo de los últimos 2-3 años), mis compañeros no estaban del todo familiarizados con el nuevo formato, usándose todavía en algunos proyectos longevos el antiguo www a pesar de estar desactualizado y sin soporte. Asimismo, resultaba difícil encontrar ejemplos útiles del nuevo UnityWebRequest. Por estos motivos, se dedicó gran cantidad de tiempo a investigar y testear al respecto[62], hasta que se dio con la fórmula correcta para el establecimiento de la conexión, envío de datos y recuperación de las respuestas (Principal carga de trabajo en el sprint 4).

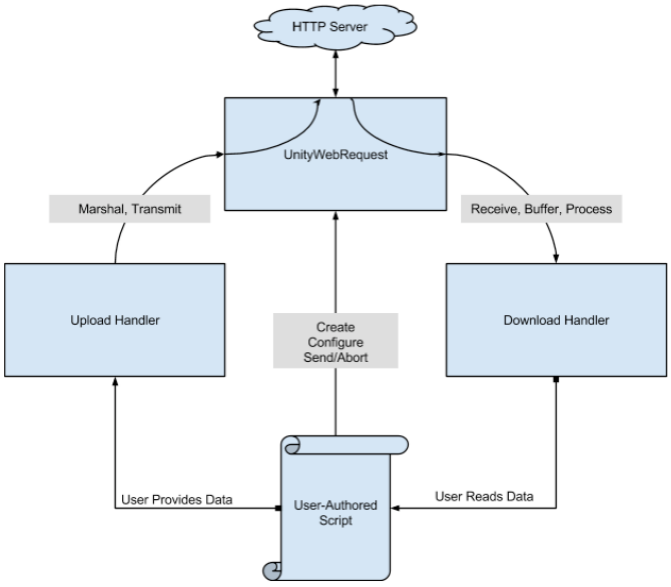

Figura 5.2: Arquitectura de un UnityWebRequest, tal y como viene descrita por su documentación oficial[49]

5.3. Base de datos

Una base de datos no era indispensable en este proyecto debido a su naturaleza simple, así como tampoco era un requisito de la aplicación. Aún así, consideré apropiada su implementación, tanto como ejercicio práctico y prueba de su viabilidad, como por una mayor adaptación a lo que sería una versión final. Una versión final comercial de la aplicación seguramente sí necesitase de una base de datos global en un servidor.

La opción de crear la base de datos en un servidor, como era esperable, fue rápidamente descartada por parte de ITCL y mis compañeros, desaconsejándome la aplicación de tales recursos a un proyecto de poca envergadura.

La opción de una base de datos global sin embargo, alojada en un servicio web, sí que fue brevemente considerada, pero nuevamente se valoró demasiado compleja para una funcionalidad añadida no principal.

No se llegó a descartar aún así la implementación de la base de datos totalmente, ya que la idea de desarrollar esta versión ya con su integración sí merecía la pena. Se decidió así la implementación de base de datos más sencilla, con una copia local.

Para su implementación fue propuesta desde el principio la herramienta SQLite[48]. Se trata de una librería compatible con lenguajes C que contiene un pequeño pero completo motor de base de datos SQL, que ya era conocida y de uso común en el departamento, con compañeros con experiencia en su integración en Unity.

A la hora de su implementación, resultó dar algún problema puesto que dicha experiencia venía dada por un proyecto sobre la versión de Unity 2018. Esto provocó relegar en algún elemento desactualizado, y obligó posteriormente a una investigación al respecto en pos de actualizarlos.

Aparte de este incidente, sobre el que se profundiza en el manual del programador, SQLite o la base de datos no supuso complicaciones adicionales, siendo la base de datos simple y pequeña (solo 3 tablas, con pocos datos de momento).

5.4. API de Cloud Vision

El elemento central del proyecto. La documentación relativa a la investigación sobre su integración en Unity, y su uso específico para reconocer y ubicar alimentos en imágenes es además un objetivo principal para ITCL, de modo que los aspectos referentes a conseguir su acceso, configurarla, y la obtención específica de dichos datos se aborda en profundidad en el manual del programador.

No obstante, en este apartado se expondrán otros detalles como la diversidad de posibilidades que ofrece la API o Google Cloud, o el detalle de la información proporcionada por la API y la facilidad de su uso.

Funcionalidades de Google Cloud

Cloud Vision no es, evidentemente, el único servicio que provee el sistema de Google Cloud[25]. Estos otros servicios no se han explorado, ya que no formaban parte del proyecto, pero sí se han podido apreciar por encima mientras se configuraba el servicio de Cloud Vision, y se consideran lo suficientemente interesantes como para mencionar los siguientes:

- **Cloud Storage[22]:** Servicio administrado para almacenar datos no estructurados. Utiliza contenedores (o buckets) básicos de datos en la nube, y puede albergar copias de seguridad o cualquier tipo de datos.
- **Google Kubernetes Engine (GKE)[26]:** Kubernetes es una plataforma de sistema distribuido de código libre para la automatización del despliegue, ajuste de escala y manejo de aplicaciones en contenedores (buckets).
- **Big Query[21]:** Almacén de datos sin servidor que funciona en diferentes nubes y se adapta a diferentes volúmenes de datos. Utiliza funciones integradas de IA y aprendizaje automático para obtener información valiosa a gran escala.

Todos sus servicios cuentan con sus respectivas APIs, además de disponer de otra enorme cantidad de integraciones, como con el traductor de Google en la nube, Google Drive, Google Maps, funciones de YouTube, etc. En definitiva, Google se esfuerza por ofrecer prácticamente cualquier servicio, y si contamos con una cuenta en Google Cloud es muy probable que nos sirva para cualquier tipo de proyecto, sea cual sea la integración o funcionalidad que necesite.

No obstante, para muchas de ellas es posible que podamos encontrar servicios más específicos y por lo tanto más eficientes o mejor adaptados, pero no deja de ser impresionante la unificación de tantas funcionalidades en un mismo proveedor de servicios.

Más opciones de Cloud Vision

Volviendo al tema central, el sistema de reconocimiento de imágenes de Cloud Vision no se limita a proporcionar etiquetas y objetos encontrados sobre la imagen. Dispone de una variedad de opciones de reconocimiento, y así como las etiquetas y objetos cuentan con una estructura de datos propia, también es el caso de cada una de las características que permite reconocer.

Los diferentes tipos de reconocimiento son:

- **Facial:** Probablemente el más conocido, y también el más específico. Así como casi todo el resto de opciones usan el mismo conjunto de clases y tipos, el reconocimiento facial consta de su propio tipo de datos. Este está acomodado para albergar la información de la ubicación de la cara, la ubicación de los elementos reconocibles de la misma (ojos, boca, nariz), su ángulo de inclinación, y el grado de una variedad de emociones que muestra el rostro, como felicidad, enfado, sorpresa...
- **Ubicaciones:** Otro más simple pero igualmente útil, devuelve una lista de las posibles ubicaciones en el mundo real de por ejemplo un paisaje, un edificio... Cabe destacar que los resultados son evidentemente muy acertados en el caso de fotos de lugares reconocibles, como la Torre Eiffel o las pirámides de Giza, pero no lo es tanto con cualquier otro paisaje genérico.
- **Etiquetas:** Uno de los utilizados en la aplicación. Proporciona una lista de etiquetas básicas asociables a la imagen analizada, como en nuestra aplicación por ejemplo devolviendo "Food", "Bread", "Sandwich", "Pasta", o para una imagen urbana "Street", "Town", Çar"... El nivel de detalle de las etiquetas y su precisión varía con aspectos como el enfoque de la imagen, o la iluminación.
- **Localización de objetos:** El otro utilizado en la aplicación. Capaz de encontrar diferentes objetos en la imagen y marcar su ubicación en la misma. En el caso de nuestra aplicación nos interesa que encuentre el objeto que coincida con la etiqueta de comida que nos interesa para obtener su ubicación, o en el mismo ejemplo de la imagen urbana encontraría el objeto Çarz su posición.

- **Logos:** Similar al reconocimiento de objetos, es capaz de identificar logotipos o iconos de marcas como el de Google, Unity, CocaCola... y ubicarlos en la imagen.
- **Texto:** Otro bastante útil, ubica texto escrito en la imagen, y en la mayoría de ocasiones es capaz de reconocer los caracteres que lo forman. Este tipo de reconocimiento se utiliza en diversas aplicaciones específicas para extraer texto escrito desde imágenes.
- **Contenido sensible:** Otra que se escapa de las clases y tipos de datos comunes, consta de tipos de datos propios para analizar en una imagen si tiene algún tipo de contenido sensible, como violencia, racismo, médico, contenido adulto...
- **Propiedades de la imagen:** Extrae datos como la paleta de colores de la imagen o la proporción de los colores por píxeles.
- **Sugerencia de bordes:** Indica los elementos más destacables que encuentra en la imagen y proporciona unos bordes de los mismos para su extracción. Es sin duda el menos eficiente de todos, ya que muchas veces no enmarca ningún objeto en concreto, o directamente proporciona resultados que parecen absurdos.

Es remarcable que para absolutamente todas las opciones, cada uno de los resultados ofrecidos está acompañado siempre de una puntuación en forma de porcentaje del 0 al 1, indicando la confianza con la que el algoritmo ha reconocido ese resultado. Los primeros elementos de una ejecución que se puede considerar exitosa suelen tener valores entre el 1 y 0.85, y los resultados con valores de confianza menor del 0.5 pueden ser casi siempre descartados.

Resultados obtenidos en la app

Como se ha indicado, la aplicación utiliza las consultas sobre etiquetas y ubicación de objetos de la API, aunque tampoco se han testeado ampliamente, ya que se han probado únicamente con ejemplos de comidas, que era el objetivo principal de su uso.

Los resultados más comunes que se obtienen con imágenes de diferentes alimentos son en general mejorables pero buenos, reconociendo por ejemplo en la imagen de un sándwich vegetal, en la que podía apreciarse lechuga o huevo, etiquetas de “Food” y “Sandwich”, incluso de “Vegetable”, pero no “Lettuce” o “Egg”. Para cualquier alimento es capaz de reconocer la etiqueta

de “Food", por lo que es una buena forma de aplicar un primer filtro y determinar si hay o no algún tipo de comida en la imagen, lo cual se ha implementado en la aplicación. Pero también parece que reconoce siempre otras etiquetas genéricas como “Ingredient”, “Recipe”, “Cuisine”, “Staple Food”... Estas etiquetas no interfieren con la detección de alimentos, ya que es improbable que se quieran registrar alimentos con esos nombres, pero aún así revela las limitaciones de precisión que tiene la API en este tipo de reconocimiento. También es destacable la gran cantidad de etiquetas que devuelve, muchas de ellas con un bajo porcentaje de confianza, por lo que en la aplicación se ha decidido filtrar éstos resultados y considerar como útiles solamente los que cuentan con una confianza del 70 % o superior.

En cuanto a los objetos encontrados, suele ofrecerse mucha menor cantidad, por ejemplo con la imagen del sándwich suele dar solo dos resultados, “Sandwich” y “Food”, por lo que vemos que este modo falla mucho menos, pero también pierde precisión, ya que no marca objetos como “Bread” o “Vegetable”, que sí encontramos en las etiquetas.

No obstante, los resultados la mayoría de las ocasiones nos permiten encontrar una etiqueta suficientemente ajustada al alimento, y luego encontrar una ubicación de objeto con el mismo nombre que esa etiqueta. Para los casos en los que no podemos establecer esta relación directa, nos queda todavía la seguridad de haber encontrado un objeto “Food”, por lo que si la etiqueta coincide, podemos asumir que éste objeto deberá corresponderse con la comida encontrada en la etiqueta.

Integración de los resultados de la API con la base de datos

Al haber seguido un diseño de los datos de abajo hacia arriba, primero se analizaron las respuestas de la API, y después se produjo la base de datos y su formato, por lo que difícilmente se encontrarán problemas de integración entre estos dos elementos. En etapas posteriores de la aplicación, bajo el supuesto de continuar su desarrollo y aumentar la base de datos, habría que seguir este mismo principio, e ir añadiendo los alimentos con nombres acorde a las etiquetas reconocidas por Cloud Vision.

Trabajos relacionados

Para este capítulo, aunque no pueda considerar haber trabajado nunca en el desarrollo de una aplicación para móvil, sí contaba con alguna experiencia en el proceso inicial de creación de un proyecto, y su desarrollo en Unity, puesto que realicé las prácticas de empresa los meses anteriores, y en la misma empresa y departamento. Se consideran entonces dignos de mención algunos proyectos en los que trabajé en ese periodo.

6.1. Aplicación de rehabilitación

Un proyecto que comenzó poco después de mi incorporación como becario de prácticas, y al que por su independencia y sencillez se me asignó a trabajar. Se trata de una aplicación de PC desarrollada con Unity con el objetivo de realizar ejercicios aeróbicos, destinada a procesos de rehabilitación. Se ubica en un escenario en 3D, pero un trabajo importante que realicé en la misma es el diseño de las interfaces, que son independientemente en 2D.

Esta experiencia ha tenido una enorme relevancia, ya que una gran carga de trabajo de mi aplicación móvil recae precisamente en el diseño de sus pantallas e interfaces, por lo que éste proceso resultó mucho más rápido y eficiente.

También pude colaborar como he dicho, en los pasos iniciales de creación del proyecto, conociendo de primera mano las arquitecturas y estructuras de datos usadas comúnmente en el departamento. Esto también sirvió de ayuda al plantear la estructura de mi proyecto, a pesar de tratarse de una considerablemente menos compleja.

Figura 6.1: Captura de la aplicación de rehabilitación mostrando elementos de la interfaz

6.2. Aplicación de triaje médico

Otra aplicación en la que colaboré como becario, también diseñada en 3D pero ésta sí para dispositivos móviles Android. En su momento no realicé gran cantidad de trabajo en ella, ni tareas específicas relacionadas con el desarrollo para móvil o Android, pero sirvió como acercamiento al entorno. Ha proporcionado una valiosa guía y ejemplos que se tuvieron en cuenta al desarrollar mi aplicación, ya que conté con acceso a ella durante mi trabajo.

Figura 6.2: Captura de la aplicación de triaje

6.3. Aplicaciones similares

Obviamente, también se han tenido en cuenta ejemplos externos, principalmente otras aplicaciones móviles similares de información nutricional.

Una de las más relevantes ha sido My Fitness Pal[39], que es principalmente una aplicación web con multitud de funcionalidades, pero cuenta con aplicación para móvil[40] destinada al registro de valores nutricionales.

Otra que también se ha tenido en cuenta es Yazio[68], otra aplicación móvil de las más usadas en el mercado a este respecto.

Se han probado otra multitud de aplicaciones relacionadas, pero lo cierto es que no existen grandes diferencias entre casi todas ellas a nivel de usuario, sobre todo pasada la minoría más descargada.

En el ámbito de la visión artificial, existe sin ir más lejos la aplicación de Google Lens[27], que utiliza este mismo sistema de reconocimiento de imágenes, varios ejemplos de aplicaciones que registran tickets de compra automáticamente haciendo uso del reconocimiento de texto en imágenes[63], o reconocen códigos de barras o patrones QR.

Aún así este tipo de aplicaciones no se ha tomado como referencia, pues lo relevante de las mismas es o bien el tipo de servicio que ofrecen (reconocer texto, patrones o imágenes completas), lo cual no tiene demasiado que ver con mi proyecto, o su método de reconocimiento de imágenes y sus posibilidades, pero este aspecto permanece opaco al usuario, de modo que tampoco podría obtener nada de valor.

Conclusiones y Líneas de trabajo futuras

7.1. Conclusiones generales

La aplicación resultante es una buena primera versión funcional del concepto, y un gran punto de partida sobre el que desarrollar una versión completa. Contando con una gran carga de trabajo por parte de la gestión de la API y la base de datos ya resuelta, el grueso del proyecto similar que se espera realizar en ITCL recaerá principalmente en el rediseño de la aplicación y sus interfaces, y la parte artística.

Precisamente la parte artística es un detalle que puede apreciarse descuidado en ésta versión, puesto que no he contado con la disponibilidad del equipo de arte del departamento por su amplia carga de trabajo durante mi proyecto. Aún así se contó con su consejo y considero que se consiguió un resultado visual tal vez no profesional, pero sí suficientemente agradable y ameno.

Así, la aplicación, aunque funcional, se encuentra en lo que se puede considerar como un estado beta a lo sumo, con gran cantidad de espacio para mejoras, refinamiento y adición de funcionalidades. A continuación se plantean las principales:

7.2. Líneas de trabajo futuras

- **Mejora de la interfaz:** Como se ha comentado, ciertas partes de la interfaz pueden resultar poco intuitivas o complejas, por lo que habría que dedicar tiempo a su mejora o explicación. Además entra

aquí la mencionada parte artística, necesitando el proyecto una mejor unificación de los diseños visuales de pantallas, los colores, las fuentes de texto, la forma de los botones, o lo más importante: un icono real de aplicación.

- **Ampliación de la base de datos:** La parte más restrictiva de la aplicación sin duda, es el escaso número de comidas registradas en la base de datos inicial. Esto hace que la aplicación pierda atractivo al parecer que funciona bien solamente con 3 o 4 comidas concretas. Para una versión más completa de la aplicación, sería indispensable dedicar tiempo a su aplicación, tiempo con el que no se ha contado al ser un trabajo muy pesado y repetitivo y con poco impacto en la funcionalidad técnica del proyecto.

- **Aumento de las interacciones con la base de datos:** La información de los alimentos en la base de datos es actualmente una información solo de lectura, y aunque se puede modificar a la hora de registrar una comida, esos cambios solo se efectúan en la entrada, pero no en la comida de la base de datos. Sería conveniente disponer de una funcionalidad para poder visualizar los alimentos desde la base de datos, y poder modificar sus valores nutricionales base o nombre representado al gusto del usuario. Esta funcionalidad ya está considerada por la gestión de la base de datos, pero es necesario el diseño e implementación de nuevas ventanas e interfaces para facilitárselas al usuario. En la misma situación se encuentran funcionalidades como añadir o eliminar usuarios.

- **Flexibilidad en las cantidades de comida:** Otra restricción es la forma de indicarlos valores nutricionales, con una representación fija por 100g de alimento. En futuras versiones se quiere poder modificar ésto, para que el usuario decida la cantidad de alimento que consume, 200g, 350g, etc, y los valores se adapten automáticamente a dicha cantidad.

- **Ampliar funcionalidades nutricionales:** Más allá de las funcionalidades planteadas en el proyecto, es interesante como aplicación de nutrición contar con algún tipo de evaluación de los valores nutricionales registrados, indicando si el nivel de consumo de cada uno de ellos es el adecuado, si se debería aumentar o disminuir en alguno, etc. Para esto sería necesaria la colaboración de personal con alguna experiencia en el campo de la nutrición, lo cual es una posibilidad en el planteamiento de proyecto futuro por parte de ITCL.

7.3. Conclusiones técnicas

Al inicio del proyecto, uno de mis objetivos principales era aumentar y afianzar mis conocimientos y manejo de Unity. También se buscaba desde ITCL una investigación por mi parte sobre la API de Cloud Vision, su viabilidad y su capacidad de integración con Unity, y hacerlo específicamente en un proyecto para Android.

Estos objetivos técnicos han quedado satisfechos por ambas partes, habiendo conseguido yo aumentar en gran medida mi entendimiento del entorno de Unity, sus recursos, técnicas y plugins. Además, he conseguido todo un conocimiento nuevo en cuanto a desarrollo en Unity para móvil, que ha demostrado no tener nada que envidiar a otros entornos de desarrollo como el destacado Android Studio, pues las únicas complicaciones relacionadas con elementos propios de Android fueron causadas por un plugin de terceros (SQLite) y no por la propia adaptación de Unity a Android.

En cuanto a la investigación sobre Cloud Vision se ha demostrado perfectamente la viabilidad de su uso en Unity y su utilidad como herramienta de visión artificial. Si bien esto último queda a expensas de un análisis más exhaustivo por parte de ITCL para su uso en proyectos posteriores, sí ha demostrado ser suficiente para esta versión de la aplicación, cumpliendo sin problemas con su necesidad de detalle en cuanto a los resultados proporcionados en las etiquetas y objetos encontrados.

7.4. Conclusiones personales

Si bien algunas de las decisiones técnicas venían definidas por parte de la empresa, y alguna se escogió teniendo en cuenta mi conocimiento previo de las mismas, no ha sido el criterio general, ya que no se obtendrá un gran valor de trabajo ciñéndose a métodos y herramientas que ya conozco, aunque sea jugar sobre seguro. Lo que se buscaba con éste proyecto al final, era ampliar mis conocimientos sobre diversos campos, y no solo en un sentido. Por esto se han tomado decisiones que tal vez me hayan hecho abarcar mucho en distintas direcciones, suponiendo complicaciones en ocasiones en múltiples aspectos simultáneamente, dando la sensación de no estar consiguiendo ningún avance en ninguno de ellos, mucho menos en el conjunto del proyecto.

Sin embargo, siempre he considerado que ésta apuesta era la decisión acertada, sobre todo contando con la experiencia y ayuda de mis compañeros de departamento en mayor o menor medida en algunos de estos campos, la cual ha sido tal vez puntual, dado que siempre he intentado encontrar

una solución por mi cuenta, pero sin duda muy valiosa, aportándome una experiencia que con total seguridad me será muy útil de ahora en adelante.

Tomé la decisión de realizar mi TFG en ITCL y con un proyecto en Unity guiado principalmente por mi deseo de entrar a continuación en el entorno laboral de la programación en Unity, desarrollando aplicaciones en 3D para realidad aumentada y virtual, y este periodo no ha hecho sino alimentar ese deseo. No solo eso, también me ha gustado la experiencia de la creación de una aplicación móvil, y las posibilidades que supone este ámbito.

Cabe destacar también la brevedad del proyecto, ya que debido a las restricciones en la organización temporal de las prácticas en empresa junto con el TFG, tan solo pude empezar éste proyecto a mediados del semestre. Este margen de tiempo ha supuesto otro reto añadido, ya que personalmente me había propuesto llegar a la entrega de Julio. Esto no habría sido posible si no hubiera conseguido un aprendizaje intensivo en cuanto a gestionar eficientemente mi tiempo. No sólo en lo relativo al proyecto, aprendiendo a priorizar tareas y funcionalidades o descartar otras a tiempo, si no también con respecto al resto de aspectos de mi día a día, eliminando muchas distracciones o pasatiempos triviales.

Por estos motivos considero que me ha servido como una experiencia más que real en cuanto a gestión de proyectos, incluidos los aspectos burocráticos, pues la elaboración de este documento y sus anexos es considerablemente exhaustiva al respecto.

Finalmente y por todo lo expuesto, considero haber logrado con creces los objetivos propuestos al inicio del proyecto, tanto personales como externos, y quedo bastante satisfecho con la aplicación que he logrado crear. Aunque se trate tan solo de una versión inicial de un producto, podré guardarlo siempre como recuerdo de mis inicios como desarrollador y de mi progreso en el área.

Bibliografía

[1] Asana. Scrumban: The best of two agile methodologies. https://asana.com/resources/scrumban, 2023.

[2] Atlassian. Bitbucket. https://bitbucket.org/, 2023.

[3] Atlassian. Jira software. https://www.atlassian.com/es/software/jira, 2023.

[4] Max Rehkopf Atlassian. Kanban vs scrum. https://www.atlassian.com/agile/kanban/kanban-vs-scrum, 2023.

[5] SQLite Browser. Sqlite browser. https://sqlitebrowser.org/, 2023.

[6] Akihiro Komori (Comoc). Unity cloud vision repo. https://github.com/comoc/UnityCloudVision, 2023.

[7] Software Freedom Conservancy. Git. https://git-scm.com/, 2023.

[8] Csharp. Tutorial de C#. https://csharp.com.es/, 2023.

[9] Android Developers. Arquitectura de la plataforma. https://developer.android.com/guide/platform?hl=es-419, 2020.

[10] Android Developers. Android studio. https://developer.android.com/studio, 2023.

[11] Android Developers. Arm processors. https://developer.android.com/google/play/requirements/64-bit?hl=es-419, 2023.

[12] Android Developers. Permissions on android. https://developer.android.com/guide/topics/permissions/overview, 2023.

[13] Heathen Engineering. Ux flat icons. https://assetstore.unity.com/packages/2d/gui/icons/ux-flat-icons-free-202525, 2023.

[14] FlyingWorm. Editor console pro. https://assetstore.unity.com/packages/tools/utilities/editor-console-pro-11889, 2023.

[15] Gamertose. White & black gui. https://assetstore.unity.com/packages/2d/gui/icons/white-black-gui-by-gamertose-168805, 2023.

[16] Epic Games. Unreal engine. https://www.unrealengine.com/en-US/unreal-engine-5, 2023.

[17] Infima Games. Animated loading icons. https://assetstore.unity.com/packages/2d/gui/icons/animated-loading-icons-47844, 2023.

[18] Erich Gamma. Patrones de diseño : elementos de software orientado a objetos reutilizable[s]. Madrid etc. : Addison Wesley, 2006.

[19] GitHub. Github. https://github.com/, 2023.

[20] GitKraken. Gitkraken. https://www.gitkraken.com/, 2023.

[21] Google. Bigquery. https://cloud.google.com/bigquery, 2023.

[22] Google. Cloud storage. https://cloud.google.com/storage, 2023.

[23] Google. Cloud vision. https://cloud.google.com/vision, 2023.

[24] Google. Cloud vision api. https://cloud.google.com/vision/docs/reference/rest/?apix=true#rest-resource:-v1.images, 2023.

[25] Google. Google cloud. https://cloud.google.com/, 2023.

[26] Google. Google kubernetes engine (gke). https://cloud.google.com/kubernetes-engine, 2023.

[27] Google. Lens. https://lens.google/, 2023.

[28] Google. Pruébala, api de cloud vision. https://cloud.google.com/vision/docs/drag-and-drop?hl=es-419, 2023.

[29] Javascript. Javascript. https://www.javascript.com/, 2023.

[30] Jetbrains. Datagrip. https://www.jetbrains.com/datagrip/, 2017.

[31] JetBrains. Intellij. https://www.jetbrains.com/es-es/idea/, 2023.

[32] JetBrains. Jetbrains. https://www.jetbrains.com/es-es/, 2023.

[33] JetBrains. Reshaper. https://www.jetbrains.com/es-es/resharper/, 2023.

[34] JetBrains. Rider. https://www.jetbrains.com/rider, 2023.

[35] Kanbanize. What is kanban? https://kanbanize.com/kanban-resources/getting-started/what-is-kanban, 2023.

[36] Pivotal Labs. Pivotal tracker. https://www.pivotaltracker.com, 2023.

[37] Microsoft. .net framework. https://www.microsoft.com/net/, 2023.

[38] Microsoft. Visual studio. https://visualstudio.microsoft.com/, 2023.

[39] MyFitnessPal. My fitness pal. https://www.myfitnesspal.com/es/welcome/learn_more, 2023.

[40] MyFitnessPal. My fitness pal mobile app. https://play.google.com/store/apps/developer?id=MyFitnessPal,+Inc.&hl=es_419&gl=US&pli=1, 2023.

[41] Juraj Novák. Sqlite viewer. https://inloop.github.io/sqlite-viewer/, 2020.

[42] Alex Okita. Learning C# programming with unity 3d, 2015.

[43] Postman. Postman. https://www.postman.com/, 2023.

[44] Muka Schultze. Enhanced hierarchy 2.0. https://assetstore.unity.com/packages/tools/utilities/enhanced-hierarchy-2-0-44322, 2023.

[45] Scrum.org. What is scrum? https://www.scrum.org/resources/what-is-scrum, 2023.

[46] Rushabh Shah. Let's move repository from bitbucket to github with all branches and commits! https://rushabhshah065.medium.com/lets-move-repository-from-bitbucket-to-github-with-all-branches-and-commits-f93c7d3bda67, 2023.

[47] Juan Humberto Sossa Azuela. Visión artificial : rasgos descriptores para el reconocimiento de objetos. Paracuellos de Jarama Madrid : Ra-Ma, 2013.

[48] SQLite. Sqlite. https://www.sqlite.org/index.html, 2023.

[49] Unity Technologies. Unitywebrequest manual. https://docs.unity3d.com/es/530/Manual/UnityWebRequest.html, 2016.

[50] Unity Technologies. Additive scenes. https://docs.unity3d.com/Manual/MultiSceneEditing.html, 2023.

[51] Unity Technologies. Asset store. https://assetstore.unity.com/, 2023.

[52] Unity Technologies. Cameras. https://docs.unity3d.com/Manual/Cameras.html, 2023.

[53] Unity Technologies. Download unity. https://unity.com/download, 2023.

[54] Unity Technologies. Empezando con el desarrollo en android. https://docs.unity3d.com/es/530/Manual/android-GettingStarted.html, 2023.

[55] Unity Technologies. Gameobjects. https://docs.unity3d.com/Manual/GameObjects.html, 2023.

[56] Unity Technologies. Prefabs. https://docs.unity3d.com/Manual/Prefabs.html, 2023.

[57] Unity Technologies. Realidad aumentada. https://unity.com/es/unity/features/ar, 2023.

[58] Unity Technologies. Scenes. https://docs.unity3d.com/Manual/CreatingScenes.html, 2023.

[59] Unity Technologies. Textmesh pro. https://docs.unity3d.com/Manual/com.unity.textmeshpro.html, 2023.

[60] Unity Technologies. Unity 3d. https://unity3d.com/, 2023.

[61] Unity Technologies. Unity and .net, what's next? https://blog.unity.com/engine-platform/unity-and-net-whats-next, 2023.

[62] Unity Technologies. Unitywebrequest class. https://docs.unity3d.com/ScriptReference/Networking.UnityWebRequest.html, 2023.

[63] Easy Expense Tracker. Easy expense. https://easy-expense.com/, 2023.

[64] Trello. Trello. https://trello.com/es, 2023.

[65] Richard Fine Unity Technologies. Unityscript's long ride off into the sunset. https://blog.unity.com/community/unityscripts-long-ride-off-into-the-sunset, 2023.

[66] Rodrigo Varga. Whateverdevs. https://github.com/WhateverDevs/Core, 2023.

[67] Yasirkula. Unity native gallery. https://github.com/yasirkula/UnityNativeGallery/releases, 2023.

[68] Yazio. Yazio. https://www.yazio.com, 2023.

[69] Zenhub. Zenhub. https://www.zenhub.com/, 2023.

Esta obra está bajo una licencia Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional (https://creativecommons.org/licenses/by-nc-sa/4.0/).