UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Graduado en Ingeniería del Software

Identificación de alimentos en comedores universitarios mediante técnicas de visión artificial y aprendizaje profundo

Food identification and Price estimation of the university canteen through artificial vision techniques and deep learning

Realizado por
Adolfo Gregorio Ingelmo Moyano

Tutorizado por
Rafael Marcos Luque Baena
Jorge García González

Departamento
Lenguajes y Ciencias de la Computación

MÁLAGA, junio de 2020

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADUADO EN INGENIERÍA DEL SOFTWARE

Identificación de alimentos en comedores universitarios mediante técnicas de visión artificial y aprendizaje profundo

Food identification and price estimation of the university canteen menu through artificial vision techniques and deep learning

Realizado por
Adolfo Gregorio Ingelmo Moyano

Tutorizado por
Rafael Marcos Luque Baena
Jorge García González

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, JUNIO DE 2020

Fecha defensa: julio de 2021

Resumen

El desarrollo de una plataforma que consiga disminuir los tiempos de espera proporcionaría una mayor fluidez a las colas de los comedores y aumentaría la eficiencia en las horas punta cuando hay más afluencia de personas al comedor. El pago de los alimentos suele ser un cuello de botella, provocando un retraso tanto del usuario como del empleado que los atiende aumentando el tiempo de espera y por consiguiente disminuyendo la satisfacción del cliente.

En este trabajo de fin de grado, se ha desarrollado una serie de utilidades para cubrir la necesidad de reducir los tiempos de espera en la cola del comedor. La aplicación móvil desarrollada tomará una foto de la bandeja con comida y la enviará a un servidor externo donde estará cargado un modelo de detección de objetos encargado de la detección e identificación del alimento. Una vez analizada la imagen, el usuario obtendrá el precio total y los productos que ha seleccionado. La detección de objetos se hace en base a un modelo de detección basado en redes neuronales convolucionales.

Palabras clave: detección de objetos, aplicación, android, servidor REST.

Abstract

The development of a platform that manages to reduce waiting times would increase fluidity of the queues of the canteens and improve efficiency at the most busy hours, when there is a higher influx of people to the canteen. The moment of food payment is usually a bottleneck causing a delay for both the user and the attending employee, therefore increasing waiting time and diminishing customer satisfaction.

In this final degree project, a series of utilities have been developed to cover the need to reduce waiting times in the canteen queue. The mobile application developed will take a photo of the tray with food and send it to an external server where an object detection model in charge of detecting and identifying food will be loaded. Once the image has been analyzed, the user will recieve the total price and the products selected. Object detection is performed through a detection model based on convolutional neural networks.

Keywords: object detection, android application, API REST server.

Índice

1
Introducción

1.1. Motivación

La Universidad de Málaga es una institución con casi 50 años de antigüedad, se encuentra dividida en dos campus principales, el campus de Teatinos y el ubicado en el barrio de El Ejido. Todas las facultades que componen ambos campus tienen integrado un servicio de cafeterías y comedores con un uso potencial diario de mas de 40000 usuarios (34202 estudiantes tanto de grado como de máster y aproximadamente 7000 personas que componen el personal tanto docente como no docente [1]). Mas concretamente, los estudiantes potenciales que pueden visitar el comedor de la facultad de informática son aproximadamente 1500(como se muestra en la imagen 1), por lo que la fluidez del comedor es vital para un correcto funcionamiento del mismo.

Los comedores colectivos son puntos de encuentro de muchos usuarios congregados usualmente en franjas horarias similares, por ello la fluidez del servicio tiene que ser alta ya que los usuarios suelen tener un tiempo limitado para comer antes de volver a sus tareas. Es necesario considerar los inconvenientes que puede acarrear que esta fluidez no sea óptima: tiempos de espera elevados con la consiguiente pérdida de tiempo de estudio o trabajo e incomodidad derivada del hecho de estar esperando particularmente importante en personas con movilidad reducida y dificultades de acceso. Además, de forma muy manifiesta en la situación actual de pandemia, las aglomeraciones implican una mayor dificultad para guardar la distancia interpersonal y ello aumenta el riesgo de contagio. Añadido a esos puntos anteriormente mencionados, la monitorización de los alimentos es una buena manera de saber qué comen los usuarios y así conseguir información para poder definir los menús en relación a las preferencias así como ampliar el conocimiento sobre salud y hábitos de nutrición de la gente.

El origen del proyecto surgió como una solución propuesta al *Smart Campus* de la UMA para solventar el problema de fluidez a la hora de tramitar el pedido en el comedor, por lo que la idea principal era integrarla en la aplicación de la Universidad de Málaga para así ofrecer una nueva funcionalidad al alumnado. La edad media de los usuarios del comedor de las facultades de la UMA hace que sean conocedores de las tecnologías actuales por lo que la implementación de una aplicación simple de usar en dispositivos móviles no supondría ningún problema y añadiría un valor superior al uso del servicio de comedor.

El uso de tecnologías punteras como son el Aprendizaje Profundo puede afectar positiva-

Centro	Titulación	Sexo	2015-16	2016-17	2017-18	2018-19
Escuela Técnica Superior de Ingeniería Informática	Graduado en Ingeniería Informática por la Universidad de Málaga	HOMBRE	337	393	426	486
		MUJER	41	40	44	49
		Total	**378**	**433**	**470**	**535**
	Graduado en Ingeniería del Software por la Universidad de Málaga	HOMBRE	280	309	329	356
		MUJER	27	26	31	36
		Total	**307**	**335**	**360**	**392**
	Graduado en Ingeniería de la Salud por la Universidad de Málaga y la Universidad de Sevilla	MUJER	111	128	137	148
		HOMBRE	109	120	136	120
		Total	**220**	**248**	**273**	**268**
	Graduado en Ingeniería de Computadores por la Universidad de Málaga	HOMBRE	146	160	153	164
		MUJER	11	15	18	16
		Total	**157**	**175**	**171**	**180**
	Programa de Doctorado en Tecnologías Informáticas por la Universidad de Málaga	HOMBRE	33	47	53	52
		MUJER	7	6	6	10
		Total	**40**	**53**	**59**	**62**
	Máster Universitario en Ingeniería Informática por la Universidad de Málaga	HOMBRE	15	16	22	26
		MUJER	2	2	4	5
		Total	**17**	**18**	**26**	**31**
	Máster Universitario en Ingeniería del Software e Inteligencia Artificial por la Universidad de Málaga	HOMBRE	13	15	17	30
		MUJER	1	1	7	6
		Total	**14**	**16**	**24**	**36**
	Total		**1.133**	**1.278**	**1.383**	**1.504**
Total			**1.133**	**1.278**	**1.383**	**1.504**

Figura 1: Alumnos activos matriculados en ETSII y ETSIT [1]

mente en la resolución de este problema, ya que aplicándolo a Visión por Computador podemos detectar los alimentos seleccionados por el usuario y proporcionar información tanto al cliente como al servicio de los productos elegidos, obteniendo así una mayor rapidez en el momento del cobro y menos tiempos de espera.

1.2. Objetivos

El objetivo de este proyecto es el desarrollo de un sistema software que agilice la labor de cobro en los comedores a través de las redes neuronales que identifique en una imagen con una bandeja de comedor los platos que se encuentran en la misma. Para lograr este objetivo, se hace uso de una aplicación Android que se ha desarrollado para la toma de una fotografía de la bandeja, dicha fotografía será enviada a un servidor API donde se procesará devolviendo la información con los platos detectados y un precio del total a pagar. Estas funcionalidades se pueden dividir en los siguientes subobjetivos:

- Desarrollo de una API REST con microservicios [2], donde cada microservicio atienda a una funcionalidad del sistema.
- Creación de una base de datos relacional que contenga la información relativa a los

alimentos disponibles en los menús, el precio de los mismos y una información corta sobre usuarios para los servicios de login. La lectura y escritura en la base de datos se ha hecho a partir de microservicios.

- Adaptación de una red neuronal a los objetivos del proyecto, entrenada con imágenes de bandejas de comedores con platos de comidas.
- Desarrollo de un microservicio en el que dado una imagen de una bandeja con platos de comida se devuelva la localización de dichos platos y los códigos de los mismos.
- Desarrollo de un microservicio en el que dado los códigos de comidas se calcule el precio final a pagar con la suma de todos los platos.
- Desarrollo de una aplicación web donde gestionar los menús de los comedores con operaciones CRUD (Operaciones fundamentales de sistemas con bases de datos: Create, Read, Update, Delete) [3].
- Desarrollo de una aplicación Android desde donde enviar una foto con la bandeja de comida, a través de la API REST, y recibir la información sobre los elementos y precio de la misma.

1.3. Estructura del documento

El documento de este proyecto se estructura de la siguiente manera:

1. **Introducción**

   Breve introducción sobre el proyecto, incluye las motivaciones que han impulsado la creación de una solución al problema de la falta de fluidez en comedores. Además se definen los objetivos a alcanzar al finalizar el desarrollo de las aplicaciones.

2. **Antecedentes**

   En esta sección se encuentran unas breves definiciones sobre las tecnologías mas complejas que se usarán en este proyecto para así tener una mejor comprensión del resto del documento.

3. **Metodología y Planificación**

   En esta sección se especifican las metodologías que se han elegido y que se aplicarán al desarrollo del proyecto, la aplicación correcta de dicha tecnología aportará orden al

desarrollo del sistema. También se podrá consultar las fases del proyecto y el tiempo que se prevé usar en cada una de ellas.

4. **Análisis de Especificaciones**

   En esta sección se desarrollan los requisitos tanto funcionales como no funcionales del sistema, son necesarios a la hora de no caer en confusiones sobre la implementación de las funcionalidades que se quieren añadir al sistema. También se aportarán casos de usos sobre algunas de las funcionalidades que así lo requieran.

5. **Diseño del Sistema**

   En esta sección se definirán las arquitectura que se tendrán como referencia para el desarrollo de las aplicaciones web y Android. Se identificarán los subsistemas que se desarrollarán para cumplir los objetivos del proyecto y se revisarán los casos de uso definiendo los subsistemas de los que harán uso.

6. **Implementación**

   En esta sección se desarrolla el flujo de trabajo que se ha seguido para la implementación de los subsistemas anteriormente definidos y por consiguiente para el desarrollo del proyecto. En este apartado se hace mayor incapié en el código empleado para cada una de las partes que componen el sistema.

7. **Conclusiones y Líneas Futuras**

   En esta sección se analizan si los objetivos han sido conseguidos. Se habla de las dificultades solventadas durante el desarrollo, los conocimientos que han sido necesario adquirir y se proponen varias ideas de líneas de trabajo a partir del proyecto actual.

2
Antecedentes

2.1. Estado del Arte

La detección de objetos lleva tiempo realizándose aplicándose a distintos contextos y en particular a la identificación de platos en bandejas de comedores fue abordada por Schettini et al.[4] en 2015. Presentaron un *dataset* con 1027 fotografías de bandejas de comedor, 3616 platos de comida y 73 clases de alimentos. Además realizaron un trazado manual de las cajas de identificación.Sobre estas fotografías realizaron un análisis basado en Redes Neuronales Convolucionales y reportaron una precisión del 79 %. Tanto el *dataset* como el trazado de las cajas de identificación están a disposición de la comunidad científica. Por tanto nos enfrentamos a un problema que ya ha sido abordado con éxito previamente así que nosotros pretendemos hacer una aplicación útil de ese conocimiento teórico.

2.2. Aprendizaje computacional

El Aprendizaje Computacional o *Machine Learning* es una rama de la Inteligencia Artificial centrada en desarrollar técnicas para que las computadoras tengan capacidad de aprender de manera automática a resolver un problema concreto sin la necesidad de que se les programen reglas específicas para dicho problema. Se busca el desarrollo de algoritmos y heurísticas que conviertan los datos en programas funcionales sin necesidad de escribirlos explícitamente.

2.2.1. Redes Neuronales

Las redes neuronales son un modelo de Aprendizaje Computacional formado por un conjunto de unidades de proceso conectadas entre sí y que reciben su nombre del intento de imitar el comportamiento de las neuronas biológicas. La agrupación de estas neuronas artificiales se realizan en capas y se comunican entre si generando unos valores de salidas según la información que recibe de entrada.

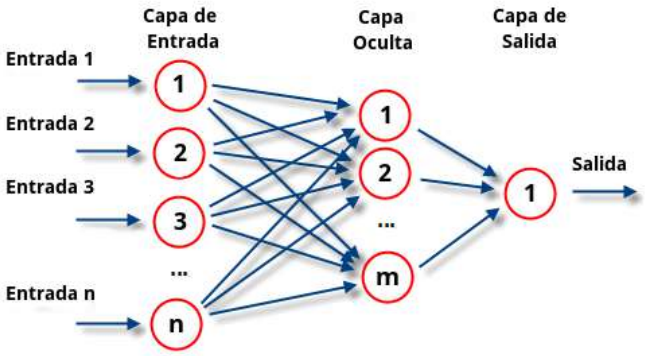


Figura 2: Red neuronal artificial [5]

Las redes neuronales son capaces de aprender gracias al algoritmo de retropropagación (*Backpropagation* en la bibliografía). Cada red neuronal tendrá una función de error que le permite la medición de lo preciso que es su funcionamiento, esta función se llama *función de pérdida*. La función de pérdida determina el error entre el resultado deseado suministrado al algoritmo de entrenamiento y la predicción que ha hecho la red. Además la red neuronal contiene una función denominada *optimizador*, el encargado de ejecutar la retropropagación. que ayuda a la reducción del error en función de los pesos, los pesos son coeficientes adaptables dentro de la red que determinan la intensidad de la señal de entrada registrada por una neurona. Las redes neuronales hacen una predicción que es analizada por la función de pérdida y el optimizador determina una manera para mejorar ese error, modificando los pesos. Este proceso se lleva a cabo hasta que se obtiene el error deseado.

2.2.2. Redes Neuronales Convolucionales

Dentro de las redes neuronales se encuentran las denominadas Redes Neuronales Convolucionales (CNN). Las CNN consisten en la creación de múltiples capas de filtros convolucionales de una o más dimensiones. Utilizada mayoritariamente para trabajos de análisis de imagen, la principal razón por la que este tipo de redes se utilizan para estas tareas es su capacidad para observar patrones en una imagen independientemente del lugar que se encuentre en la imagen.

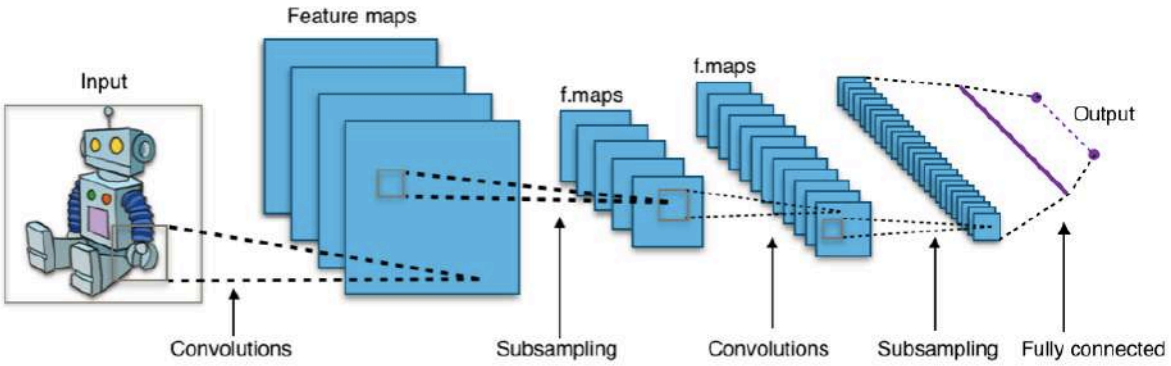


Figura 3: Red neuronal convolucional [6]

- **Capas convolucionales**

  Entre las capas en las que se compone la CNN, se encuentran las capas convolucionales. Estas capas son las encargadas de extraer los patrones que se usarán para el aprendizaje. Las capas están formadas por varios filtros de imagen tal que los valores matriciales de estos filtros son los pesos a entrenar. La aplicación iterativa del filtro a una imagen permite que el cálculo de las unidades de cómputo tengan en cuenta información espacial local. La función final de la capa del filtro es resaltar una característica de la imagen, por lo que cada capa buscará unas características mas compleja en base al cómputo que ha realizado la capa anterior. La salida de una capa convolucional será un una matriz de activaciones de dimensión N, siendo N el número de filtros, con todas las versiones de la imagen al aplicarle cada filtro.

- **Capas Pooling**

  Las capas *pooling* están diseñadas para reducir la información de la imagen reduciendo su tamaño. Su modo de acción es parecido al de las capas convolucionales pues recorren la imagen cuadrado de píxeles a cuadrado de píxeles y según el tipo de *pooling* se calculará un valor. Puede ser el mínimo, máximo, media, mediana, etc. Aunque generalmente se usa el máximo porque en combinación con las capas convolucionales, destacará las características con una activación mayor.

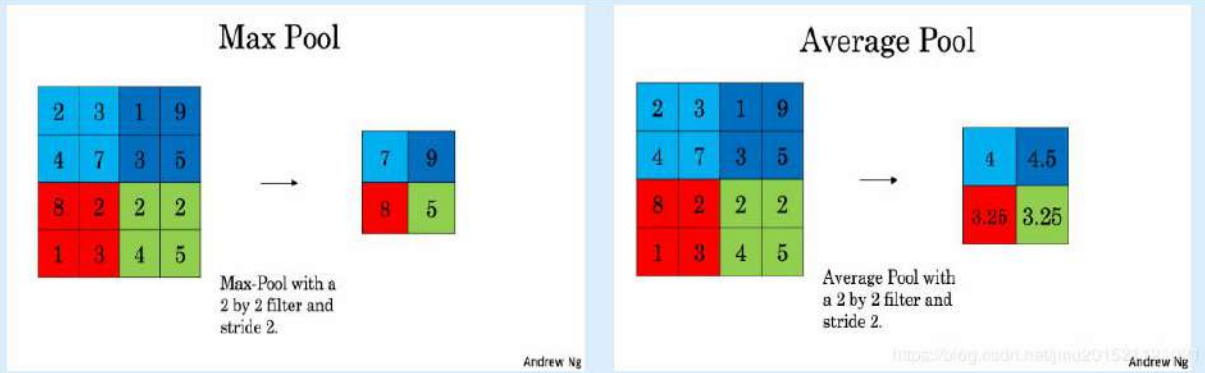


Figura 4: Capas pooling [7]

2.2.3. Transfer Learning

El *Transfer Learning* es una técnica consistente en la utilización de una red neuronal ya entrenada con un vasto conjunto de imágenes y adaptarla a un problema distinto a aquel para el que se entrenó. Este procedimiento permite empezar a trabajar con una red neuronal que han recibido un entrenamiento usando millones de imágenes (algo inabarcable para un proyecto de esta categoría), por lo que ha aprendido características básicas como bordes o formas simples y complejas, características genéricas que se podrían encontrar en otras imágenes. Todo ello nos ahorra la parte con mayor coste computacional, el entrenamiento de las capas convolucionales de la CNN.

El *Transfer Learning* también permite realizar un entrenamiento adecuado con un conjunto de datos reducido. Si los datos conseguidos no son suficientes para entrenar a una red desde 0, si que se podrán usar partiendo de una red ya entrenada.

2.3. API REST

REST (*Representational State Transfer*) es un estándar para la conexión entre sistemas que use HTTP para la obtención de datos o la generación de operaciones sobre los datos obtenidos en formatos como JSON.

La ventaja de este sistema es la especificación de un protocolo cliente/servidor sin estado donde el mensaje HTTP contiene la información necesaria para comprender la petición del cliente, evitando así el almacenamiento de información acerca de las conversaciones previas

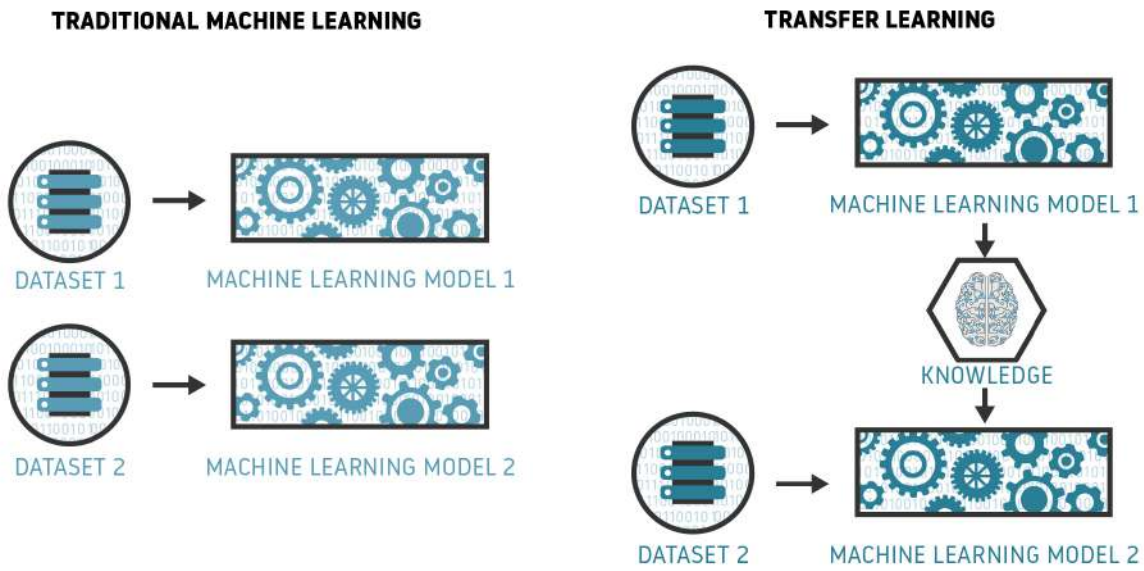


Figura 5: Esquema comparativo entre proceso de aprendizaje usual y Transfer learning [8]

entre el cliente y el servidor. HTTP contiene una serie de operaciones definidas, las mas usadas son:

- **GET:** lectura de un recurso.
- **POST:** creación de un nuevo recurso.
- **PUT:** actualización de un recurso.
- **DELETE:** eliminación de un recurso.

La comunicación entre el cliente y el servidor API REST se hará usando el formato de texto JSON, un formato de texto sencillo:

```
{
  "img_base64": "",
  "price": "9.99",
  "mealList": ["a","b","c"]
}
```

3

Metodología y Planificación

3.1. Metodologia

En el mundo actual del desarrollo de software se han introducido una serie de marcos de trabajo para obtener un desarrollo ágil del producto. Uno de estos marcos de desarrollo ágil [9] es el llamado *Scrum*: esta metodología permite el desarrollo colaborativo con un equipo de trabajo mientras se van realizando pequeñas entregas para concluir en la entrega del producto final. *Scrum* permite una gestión diaria del proyecto, lo que facilita la adaptación y flexibilidad del mismo a la hora de afrontar problemas que puedan surgir durante el desarrollo.

El método *Scrum* se divide en *sprints*, los *sprints* son unidades de tiempo comprendida entre 2 y 4 semanas en las cuales se realizarán las fases del proyecto que correspondan y encajen con la duración establecida. Al finalizar ese periodo se harán reuniones de control con el *Scrum Master*, el cual evaluará el trabajo realizado durante el sprint y a partir de esa evaluación se planearán las siguientes fases que estarán incluidas en el siguiente sprint.

Para poder llevar un control mas regular y permitir centrarse cada persona que compone el equipo en una tarea en específico, y así aumentar la productividad, se definen una serie de roles:

- **Scrum Master (o Facilitador).** La persona con este rol centra sus esfuerzos en el correcto cumplimiento de las normas marcadas para así conseguir que se alcancen las metas establecidas en cada sprint. Este rol le corresponde al tutor del trabajo de fin de grado, Rafael Marcos Luque Baena y al cotutor del trabajo de fin de grado, Jorge García González.
- **Desarrollador.** Se encarga del desarrollo del proyecto y son los responsables de que el producto se encuentre terminado al finalizar los *sprints*. Este rol lo asume el autor, Adolfo G. Ingelmo Moyano.

3.2. Planificación

La metodología *Scrum* requiere de una planificación previa para poder dar fluidez al desarrollo del proyecto. A continuación el lector puede observar una aproximación de las horas estimadas para las diversas partes en las que he visto a bien dividir el proyecto como se puede ver en el cuadro 3.1.

Fases del proyecto	
Fase	**Horas**
Documentación y estudio de las tecnologías	30
Obtención de Requisitos	15
Elaboración de diagramas	8
Implementación:	**208**
Configuración y Entrenamiento de la red neuronal	62
Desarrollo de la base de datos relacional	8
Desarrollo del servidor web	28
Integración de la red neuronal en el servidor web	39
Desarrollo de la aplicación web	36
Desarrollo de la aplicación móvil	35
Pruebas y reimplantaciones	35
Total	**296**

Cuadro 3.1: Distribución de las fases y horas del proyecto

Dentro del contexto de la metodología *Scrum* es necesario definir las horas que se van a dedicar a la semana al proyecto para así poder encajar las fases con los *sprints* que sean necesarios. En el caso que nos ocupa, y debido a la utilización de los fines de semana para el desarrollo del proyecto, se invertirán una media de 35 horas semanales o, dicho de otra manera, 5 horas diarias. Así que si definimos un *sprint* en el marco de dos semanas, la realización de cada sprint será de unas 70 horas por sprint.

El Cuadro 3.2 muestra la distribución de las fases del proyecto, se puede observar que el proyecto se ha completado en 6 *sprints*, algunas de las fases han invertido mas tiempo del estimado debido a complicaciones por lo que se ha empleado un *sprint* más del planificado inicialmente.

Sprints del proyecto	
Fase	**Sprint**
Documentación y estudio de las tecnologías	1
Obtención de Requisitos	1
Elaboración de diagramas	1
Configuración y Entrenamiento de la red neuronal	2-3
Desarrollo de la base de datos relacional	3
Desarrollo del servidor web	4
Implementación de la red neuronal entrenada con datos acordes a la aplicación	4
Desarrollo de la aplicación web	5
Desarrollo de la aplicación móvil	5
Pruebas y reimplantaciones	6

Cuadro 3.2: Distribución de las fases y sprints del proyecto realizados

4

Análisis de Especificaciones

4.1. Análisis de Requisitos

En esta sección se analizan los requisitos necesarios para solventar los problemas del proyecto (identificación de platos y cálculo de precios). Los requisitos se pueden dividir en dos tipos:

- **Requisitos Funcionales:** definen una función del sistema, es decir, un conjunto de entradas, comportamientos y salidas.
- **Requisitos No Funcionales:** describe atributos de calidad del sistema, normalmente restricciones o condiciones que el cliente impone al sistema.

Cada requisito es identificado con un código identificativo y una breve descripción. Los requisitos funcionales llevan una nomenclatura del tipo **RF-XX**, donde XX es un código numérico de dos cifras y los requisitos no funcionales se identificarán con el código **RNF-XX**, donde XX es un código numérico de dos cifras.

4.1.1. Requisitos funcionales

- **Reconocimiento e identificación de platos - RF-01**
  Se debe ser capaz de reconocer e identificar dichos alimentos en una bandeja de comida.

- **Calculo de precios - RF-02**
  Se debe, a partir de códigos de los platos identificados, calcular el precio final de la bandeja de comida seleccionada.

- **Devolución de la información de platos y precio final - RF-03**
  Se debe mostrar la información calculada anteriormente sobre los precios y una lista de platos seleccionados.

- **Edición de platos del menú - RF-04**
  Se debe, usando una aplicación web, poder editar datos básicos de los platos del menú, los datos editables serán: Nombre, precio y disponibilidad en el menú.

- **Realización de fotos - RF-05**
  A través de una aplicación Android, se podrá tomar una foto de la bandeja de comida haciendo uso de su dispositivo móvil.

- **Envío de fotos al servidor - RF-06**

  El sistema debe de ser capaz de enviar la fotografía realizada por el usuario a la API REST para su posterior análisis e identificación.

- **Login en Aplicación Web - RF-07**

  Se debe poder acceder a la aplicación a través de un sistema de registro que codifique las contraseñas para una mayor seguridad.

- **Login en Aplicación Web desde aplicación Android - RF-08**

  Se debe poder acceder a la aplicación usando un sistema de login realizando una petición al servidor.

- **Conexión con base de datos - RF-09**

  El sistema debe conectarse a la base de datos para recuperar información almacenada relativa tanto a usuarios como a las comidas. La conexión debe de ser segura y estable.

4.1.2. Requisitos no funcionales

- **Rendimiento - RNF-01**

  Debido a que buscamos una mayor fluidez en las colas, es indispensable un rendimiento óptimo en los dispositivos en los que se vayan a ejecutar para así alcanzar nuestros objetivos.

- **Estabilidad - RNF-02**

  Debido a su conexión con la red, es necesario que esté disponible en cualquier momento durante el horario del comedor.

- **Fiabilidad RNF-03**

  El sistema debe de aportar una información fiable y veraz sobre los alimentos que se han seleccionado.

- **Operabilidad RNF-04**

  La aplicación se debe de desarrollar con gran operabilidad, posibilitando su uso a toda clase de personas indistintamente de sus habilidades tecnológicas. El producto debe de ser sencillo y enriquecer la experiencia del usuario.

4.2. Casos de uso

A través de la definición de los casos de uso se pueden describir las acciones que se puede realizar dentro del sistema. La definición de estos casos de usos vienen acompañadas por diagramas que definen las actividades correspondientes al usuario o al sistema, los actores, para llevar a cabo los procesos. Los casos de usos se pueden definir como la secuencia de interacciones entre los actores desencadenada por un evento inicial ejecutado por el actor principal. Los diagramas muestran la interacción entre los distintos actores o, dicho de otra manera, entre los usuarios y el sistema.

Los Casos de Uso que se definirán contendrán una pequeña descripción del/de los Requisito/s Funcional/es que modela y estarán precedidos por un código con la siguiente forma: CU-XX, donde XX será un código numérico de dos cifras.

4.2.1. CU-01 - Login desde Aplicación Web

El siguiente diagrama de secuencia describe las acciones que se llevan a cabo a la hora del registro en la aplicación. El usuario rellenará los campos de usuario y contraseña y se envían al sistema que comprobar, desencriptando el valor *hash* de la contraseña almacenado en la base de datos, que los datos son correctos.

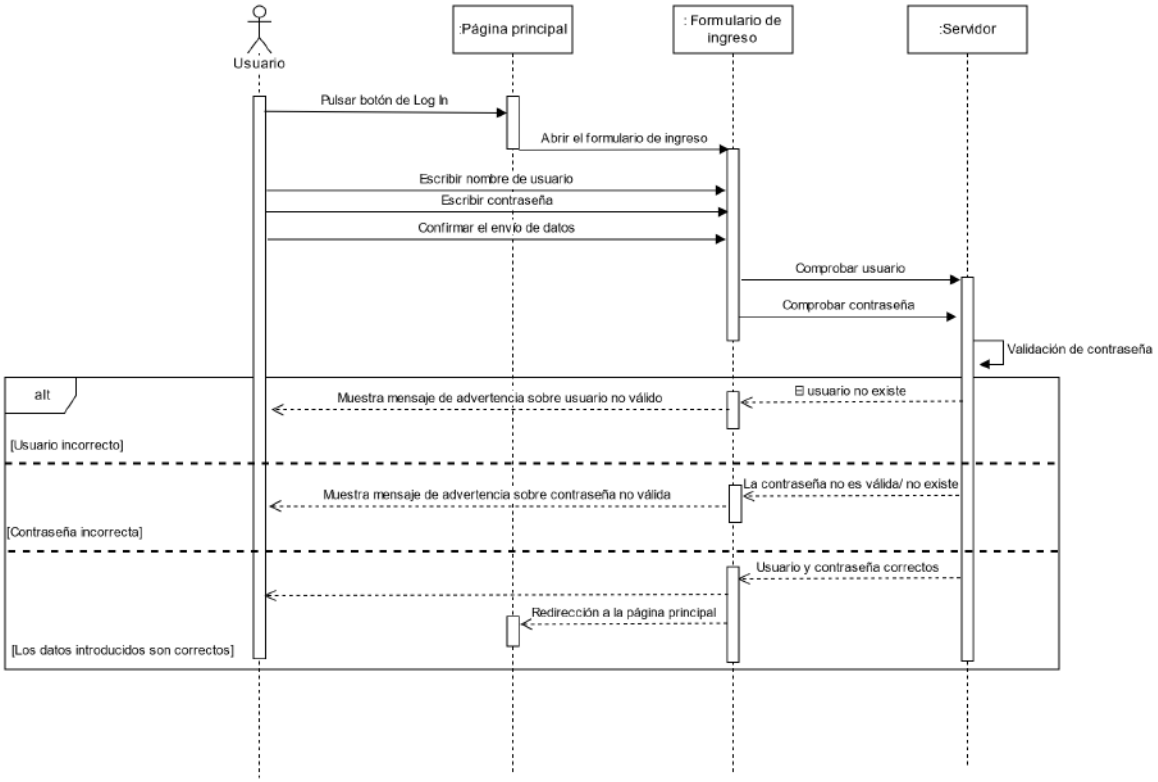


Figura 6: Diagrama de secuencia, Login en la aplicación web

4.2.2. CU-02 - Login desde Aplicación Android

El siguiente diagrama de secuencia describe las acciones que se debe llevar a cabo a la hora del registro en la aplicación. El usuario rellenará los campos de usuario y contraseña, la aplicación Android envía los valores introducidos en formato JSON para que el servidor de la Aplicación Web API compruebe la validez de los datos tal y como se han descrito en caso de uso CU-01 (4.2.1)

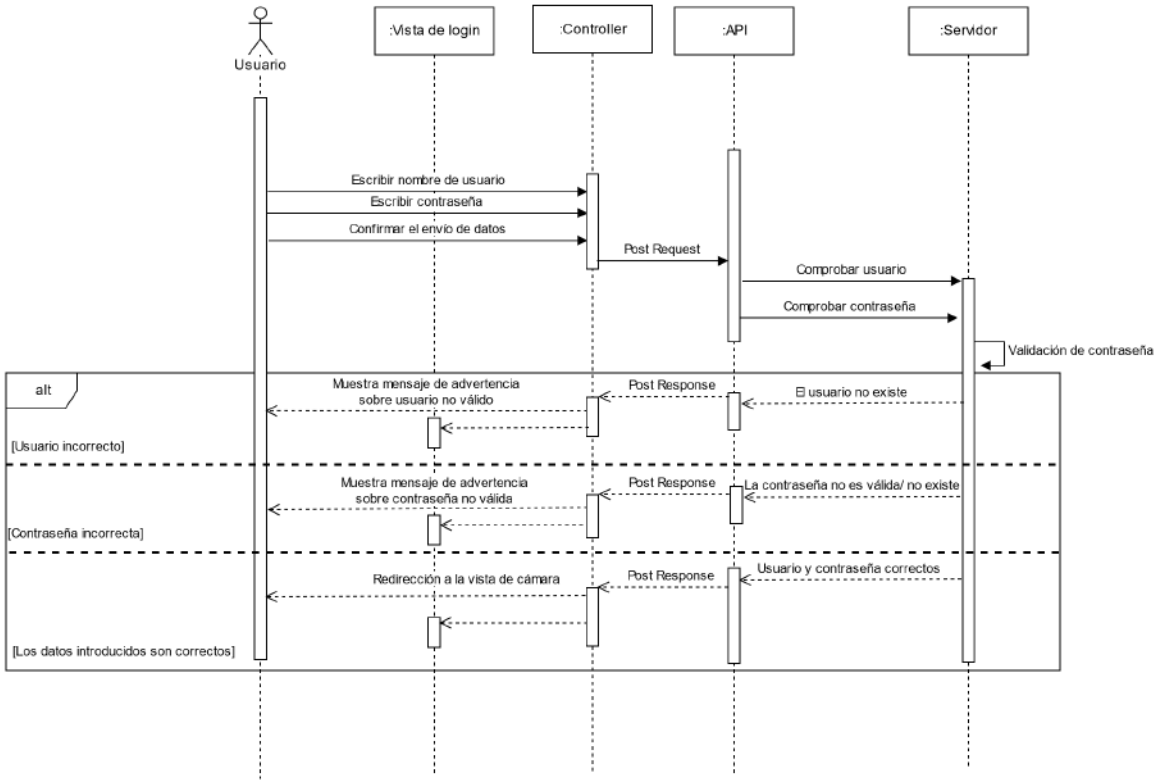


Figura 7: Diagrama de secuencia, Login desde aplicación Android

4.2.3. CU-03 - Mostrar precio y lista de alimentos

Para el análisis de los platos de comida en la bandeja es necesario que el usuario tome una foto de la misma. Una vez hecha, la foto se debe enviar al servidor API para su análisis. Una vez recibida en el servidor, la foto será analizada por el modelo de red neuronal entrenado anteriormete devolviendo el nombre de los elementos en la fotografía, el precio de cada uno de esos platos se consultará en la base de datos y la suma de los precios de los alimentos que están en el menú e identificados se reenviará junto con una lista de los elementos en la bandeja de comida.

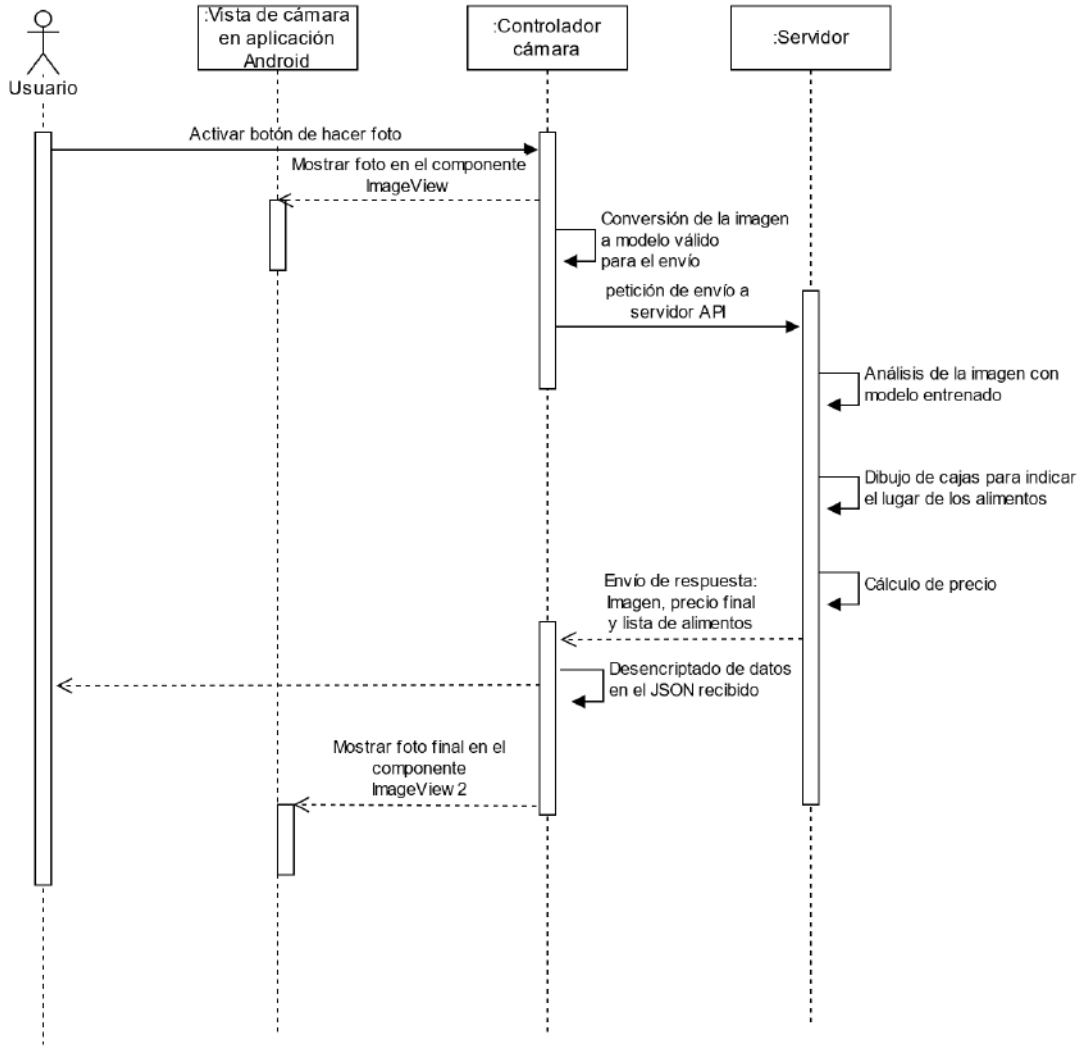


Figura 8: Diagrama de secuencia, Mostrar precio y lista de alimentos

4.2.4. CU-04 - Edición de platos del menú

La edición de los menús se harán desde la página principal de la aplicación web. El usuario registrado con roles de administrador debe ser capaz de añadir, eliminar y editar los platos disponibles y no disponibles en el menú. Los campos editables son: nombre del plato, precio, inclusión en el menú y código. La edición de los platos se debe poder realizar desde una vista de edición y se editarán por separado, el servidor debe leer los valores y los actualiza en la base de datos.

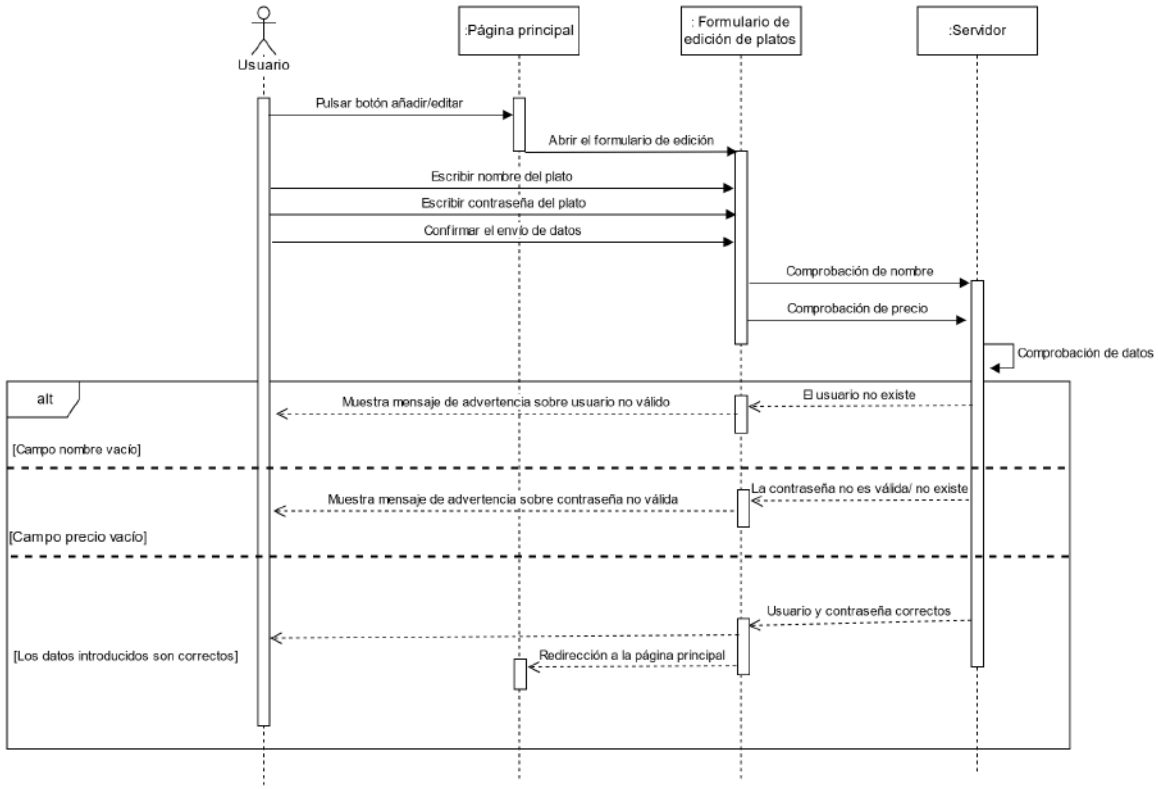


Figura 9: Diagrama de secuencia, Edición de platos del menú

5
Diseño del Sistema

5.1. Arquitectura del Sistema

Para un correcto desarrollo del proyecto, es recomendable tener claro los patrones en los que se basa para tener una referencia clara, el conjunto de patrones que se usan engloban lo denominado como Arquitectura del sistema. A través de esta sección se definirán el funcionamiento, estructura e interacción de las distintas partes del proyecto.

5.1.1. Microservicios

Debido a las distintas funcionalidades que contiene el sistema, utilizar una arquitectura basada en microservicios [10] es un acierto debido a su potencial de escalabilidad. La arquitectura de microservicios se basa en el desarrollos de pequeños servicios autónomos, independientes del resto y que implementan una funcionalidad distinta. Este aislamiento entre servicios hace que sean los propios servicios los encargados de asegurar y conservar la persistencia de los datos que manejan.

Otra de las partes positivas que tienen los microservicios es el manejo y corrección de errores, ya que si se produce un fallo en un servicio ninguna otra funcionalidad se verá afectada, esta burbuja de separación entre servicios permite también la actualización o modificación del mismo sin afectar al resto.

Los mayores puntos positivos que conlleva la utilización de una arquitectura basada en microservicios son la escalabilidad y reusabilidad de las funcionalidades. Al estar todo el proyecto dividido en servicios debidamente identificados, se podrá reutilizar dichos servicios para otras partes del mismo, algo que no se podría implementar si toda la funcionalidad de un requisito estuviera definida en un mismo servicio.

5.1.2. Modelo Vista Controlador (MVC)

Para la aplicación Android se ha usado la Arquitectura de sistema Modelo-Vista-Controlador (MVC) [11], este patrón de arquitectura separa los datos entre la lógica de negocio y la sección que se encarga de controlar los eventos que se producen. Al igual que la arquitectura basada en microservicios, su principal cometido es la reutilización de código y la separación de conceptos. Cada una de las partes en que se compone son:

- **Modelo:** Gestiona todos los accesos a la información, normalmente están referencia-

dos a una tabla de una base de datos pero para nuestro caso los modelos solo tendrán información de las respuestas del servidor API.

- **Controlador:** Responde a eventos y devuelve información relativa a los modelos y además, en el caso de la aplicación Android del proyecto, también gestiona los eventos que se puedan producir en su vista asociada. Es un intermediario entra la vista y el modelo
- **Vista:** Presenta en una interfaz de usuario los datos y la información del modelo.

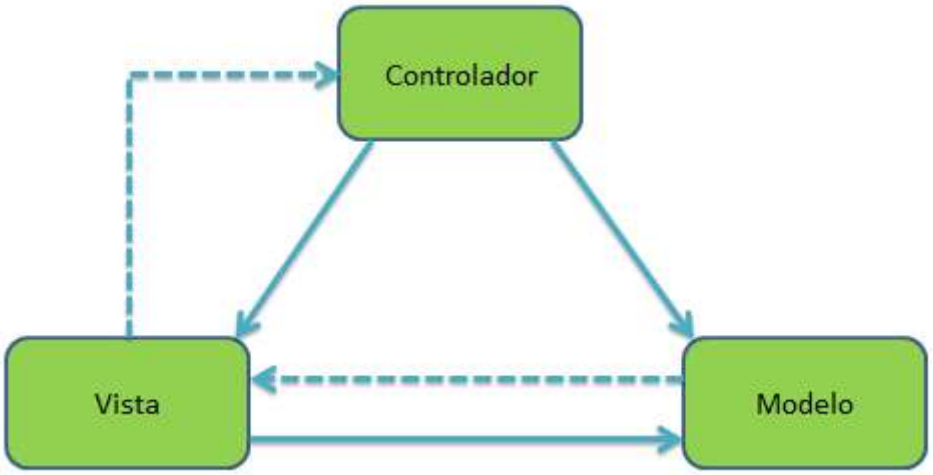


Figura 10: Diagrama MVC

5.2. Identificación de subsistemas

Es necesario una correcta división de los sistemas y así poder aplicar las arquitecturas que he seleccionado para el desarrollo de este proyecto. Tal y como se han definido en una de las secciones anteriores, estos subsistemas son los llamados microservicios, en esta sección se especificarán tanto la meta que representan como la conexión que puedan tener con el resto de microservicios.

El sistema se ha dividido en los siguientes microservicios:

- Gestor de menús
- Identificación de comidas
- Gestor de usuarios

- Toma de fotos con aplicación Android

Los primeras tres microservicios se encuentran englobados en la API REST, y sus funcionalidades estárán divididos en controladores, servicios y repositorios. La aplicación Android seguirá los patrones de MVC tal y como me he referido en la sección 5.1.2

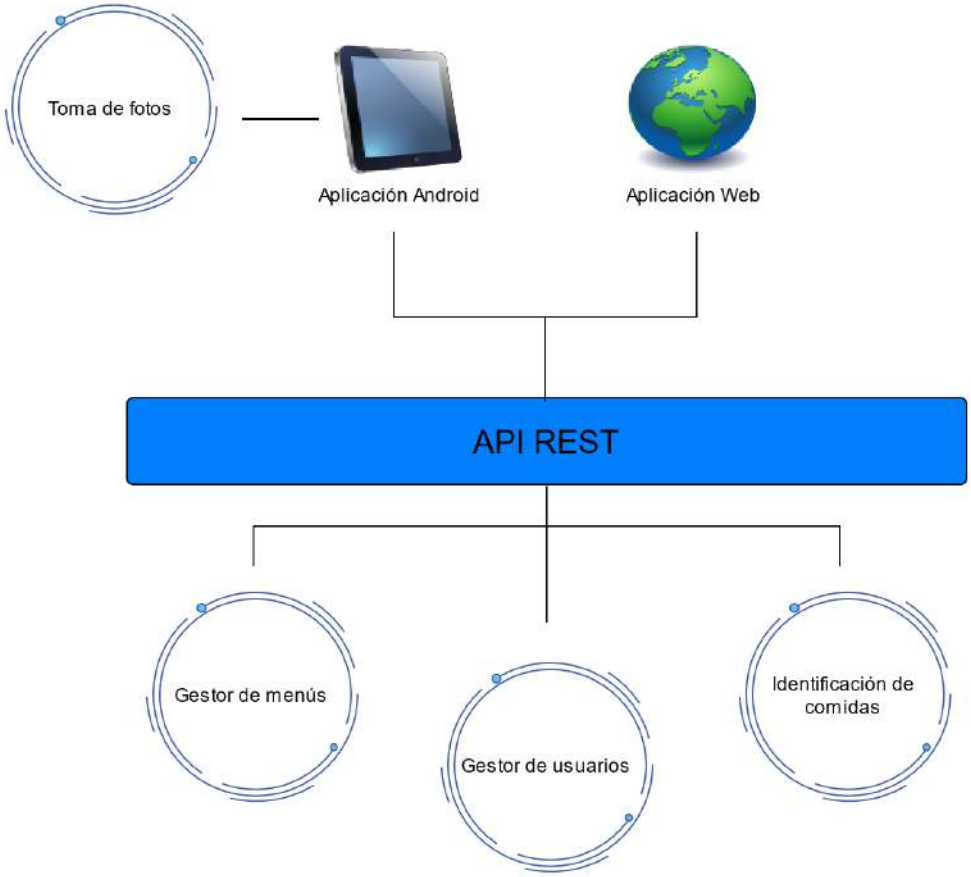


Figura 11: Arquitectura del sistema

5.2.1. Gestor de menús

Este microsistema se encargará de la edición de los menús, mediante el uso de la aplicación web se podrá crear un nuevo plato, editar o borrar uno existente. Los campos editables son el nombre, el precio y la inclusión en el menú pues otros campos como el código de la comida son autogenerados para que sean acordes a los que la red neuronal ha aprendido.

5.2.2. Identificación de comidas

El microsistema de identificación de comidas se encargan del análisis de la imagen de la bandeja, la identificación de los alimentos en la misma y el tratamiento de la captura para aportar los datos generados. A través de este microservicio se iniciará el modelo entrenado, una vez inicializado se identificarán los platos y devolverá los códigos de alimentos, las puntuaciones de dichas identificaciones y los puntos de las cajas que se dibujarán alrededor de los platos.

5.2.3. Gestor de usuarios

A través de este microsistema se ejecutarán las tareas de registro y login. Además se encargará de la seguridad en la edición de datos al controlar los accesos a ciertas vistas de la aplicación así como a las acciones restringidas por el rol al que pertenezca el usuario que las ejecute.

5.2.4. Toma de fotos con aplicación Android

Dentro de la toma de fotos se engloba también el login desde la aplicación Android ya que es unaadaptación de la integración en la aplicación de la UMA tal y como se explica en el apartado 1.1. La toma de fotos se encarga de la captura de la bandeja, su procesado y por último el recibimiento de la respuesta, su decodificación y la muestra de los mismos en la pantalla principal de la aplicación.

5.3. Revisión de casos de uso

Relación entre los casos de usos definidos en la sección 4.2 con los subsistemas anteriormente definidos.

- **CU-01 - Login desde Aplicación Web**

  La identificación de los usuarios para el uso de la aplicación web se lleva a cabo usando simplemente el microservicio de gestión de usuarios, a través de la comprobación del nombre de usuario y contraseña, en la cual es necesario la desencriptación de la misma, se puede acceder a la edición de los menú.
- **CU-02 - Login desde Aplicación Android**

  La identificación por medio de la aplicación Android se lleva a cabo de la misma manera que en CU-01, se usa el microservicio de Gestión de usuarios.
- **CU-03 - Mostrar precio y lista de alimentos**

  Para poder mostrar el precio y la lista de alimentos es necesario la intervención de tres de los microservicios anteriormente mencionados: primero se hace uso de la toma de fotos con la aplicación Android para obtener una captura de la bandeja de comida, tras enviar la captura a la API, esta se ve afectada por el servicio de identificación de comida que se encarga de analizarla y dibujar las cajas sobre los alimentos identificados, por último el microservicio de gestor de menús se encargará de obtener los precios de los platos identificados y devuelve el precio total y una lista con los nombres de los alimentos.
- **CU-04 - Edición de platos del menú**

  El caso especificado para la edición de menús es ejecutado por dos microservicios, el gestor de menús se encarga de la edición, creación y borrado de platos, además de actualizar la disponibilidad de los mismos en el menú y el gestor de usuario hará las comprobaciones pertinentes para que dichos datos sean editados por la persona con el rol correspondiente.

5.4. Estructura de la base de datos

El sistema de gestión de bases de datos que se ha decidido implementar en el proyecto es SQLite[12]. Este gestor de bases de datos está integrado dentro del programa a diferencia de otros sistemas de gestión de bases de datos que requieren de un cliente-servidor para su

funcionamiento. Todo lo que está contenido dentro de la base de datos (definiciones, tablas, índices y datos) está contenido en un solo fichero en el propio proyecto, aunque esto supone un punto negativo, pues el acceso a los datos se efectúa bloqueando el fichero completo, no afecta al funcionamiento del proyecto pues las operaciones que tardan mas tiempo son las de escritura y dichas operaciones solo se ejecutarán por el administrador del sistema en franjas horarias en las que no haya usuarios identificando comidas.

SQLite implementa parte del estándar SQL-92 [13], como los puntos de aislamiento, durabilidad, triggers o las transacciones de base de datos atómicas. A diferencia de otras bases de datos relacionales, SQLite admite cualquier tipo de valor independientemente del tipo al cual se haya definido la columna donde se vaya a insertar, esto puede verse como un avance ya que no produce problemas de compatibilidad de datos aunque puede verse como un escollo para la migración de los datos hacia otros gestores de bases de datos.

La base de datos que contiene el sistema está dividida en tres tablas (*meal, user, userRoles*) que contienen la información relativa a las comidas del menú, a los usuarios y a sus roles. Como se puede observar en la figura 12

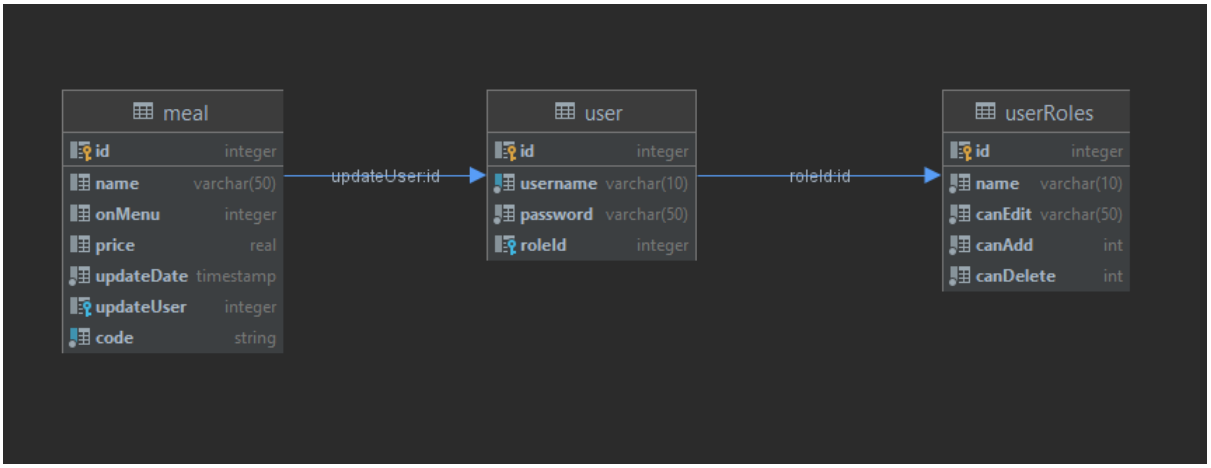


Figura 12: Diagrama entidad-relación de la base de datos del sistema

6
Implementación

En esta sección de la memoria se describe la implementación de los distintos aspectos de la aplicación y las tecnologías usadas para ello.

6.1. Tecnologías usadas

A través de esta sección, se describen las tecnologías y aplicaciones que se ha usado para el desarrollo del proyecto. Las tecnologías seleccionadas deben adecuarse a la arquitectura, definida en la sección Arquitectura del Sistema.

- **Proyecto Web API REST**

  Como lenguaje base para la aplicación web y el tratamiento de la red neuronal se ha seleccionado Python [14], Python se está convirtiendo en uno de los lenguajes mas usados a nivel mundial gracias a su simplicidad, portabilidad y flexibilidad a la hora de programar debido a la cantidad de herramientas de la que se dispone para crear código de manera flexible. Python es un lenguaje de programación multiparadigma, soporta la programación orientado a objetos y también incorpora elementos propios de la programación imperativa y de la programacion funcional. La flexibilidad de Python tanto de entornos de trabajo como de librerías existentes hacen de él un lenguaje mas que apto para el desarrollo de aplicaciones web.

  Habiendo seleccionado Python como lenguaje para la aplicación web, el *framework* que se usará para el desarrollo de la API REST es *Flask* [15], este *framework* se apoya principalmente en dos bibliotecas: *Jinja* [16](versión 3.0.1) para la introducción de plantillas en la parte *Front-End* de la aplicación y en *Werkzeug*(versión 2.0.1) [17] para la comunicación a través del protocolo HTTP usando el estándar *Web Server Gateway Interface* (WSGI) [18]. La elección de este *framework* para la realización de la aplicación web es debida a la variedad de bibliotecas que son compatibles actualmente, su claridad a la hora de hacer las operaciones más repetitivas propias de una API REST y la simplicidad a la hora de generar la nueva aplicación. El estilo que se presenta en la aplicación web es proporcionado por la librería *Bootstrap* [19](versión 5.0.1). En la Figura 13 se muestra la vista del panel principal de la aplicación

- **Base de Datos**

  Para la gestión de la base de datos se usa *SQLite* [12], un gestor de base de datos que no

Figura 13: Panel principal aplicación web

necesita de un servidor para su utilización y es totalmente independiente. El motor de la base de datos (BD) se ejecuta como parte de la aplicación. Comparado con *MySQL*, el tamaño que ocupa esta base de datos suele rondar los 250KB, mucho inferior a los 600MB que puede ocupar un servidor de *MySQL*, en el caso de *SQLite* la BD se encuentra en un solo fichero por lo que su portabilidad es muy fácil, el número de tipos de datos es menor en *SQLite* aunque no nos es necesario un rango mayor de tipos debido a los datos simples que se manejan. El mayor problema de *SQLite* es la poca escalabilidad que tiene, el acceso a los datos no es múltiple por lo que si hay muchas peticiones de escritura simultanea puede resultar un problema a largo plazo, en el caso que nos ocupa no debería afectar en ninguna medida pues las peticiones concurrentes únicamente serán de lectura, las cuales son bien manejadas por *SQLite*.

- **Entrenamiento y Uso de Redes neuronales**

  Para el entrenamiento del modelo de redes neuronales que se utilizan en el proyecto se ha utilizado *Tensorflow* [20], se trata de una biblioteca de código abierto desarrollado por Google dedicada al aprendizaje automático y centrada en la creación y entrenamiento de redes neuronales. La versión utilizada de esta biblioteca es la 2, apoyada por las extensiones de CUDA 8Compute Unified Device Architecture[21]). Para los entrenamientos se ha utilizado una RTX3080 y el servidor de la APP funciona sobre una tarjeta gráfica GTX970. Ademas de Tensorflow (versión 2.5.0), se han usado otras librerías de

Python como son NumPy (versión 1.19.5), Pandas (versión 1.1.5), Matplotlib (versión 3.4.2), OpenCV (versión 3.0.1), Pillow (versión 8.2.0). La biblioteca Numpy [22] proporciona herramientas para el tratamiento de vectores y arrays, Pandas [23] se ha usado para el tratamiento y lectura de los datos almacenados en ficheros excel, Matplotlib [24] es una librería para crear y manipular gráficos, la biblioteca OpenCV [25] se utiliza para la manipulación de imágenes y vídeos, Pillow [26] es una librería con multitud de herramientas para el tratamiento de imágenes y por último la biblioteca de *TensorFlow Object Detection* [27] nos da las funcionalidades para entrenar y aplicar modelos de detección de objetos basados en redes neuronales.

El entorno de desarrollo que se ha usado tanto para el proyecto de entrenamiento de la red neuronal como para el desarrollo de la aplicación web, ambos en lenguaje Python, es PyCharm. Es un entorno popular actualmente que proporciona una gran cantidad de herramientas que facilitan el desarrollo y depuración de los servicios web en los que se ha trabajado.

- **Aplicación Android**

  El cliente se ha desarrollado para dispositivos Android usando Java como lenguaje de programación y usando la aplicación Android Studio como entorno de desarrollo. Este entorno está diseñado especialmente para el desarrollo de este tipo de aplicaciones aportando una interfaz completa y eficaz para el diseño de interfaces, permite el compilado de las aplicaciones y la posibilidad de crear emuladores de dispositivos Android donde ejecutar la aplicación y hacer las pruebas oportunas. Para la gestión de la cámara de fotos se ha usado la librería *Fotoapparat* [28] (versión 2.7.0) y para la conexión con la API REST se ha empleado la librería *Retrofit2* [29] (versión 2.9.0). En la Figura 14 se muestra la vista de la funcionalidad de fotografía desarrollada para la aplicación Android.

Figura 14: Vista fotografía con resultados, la imagen analizada está cargada desde memoria

6.2. Entrenamiento de la red neuronal

Para el entrenamiento de la red neuronal se ha utilizado un dataset proveniente de un trabajo anterior sobre reconocimiento de comidas [30], el dataset proporciona 1027 fotografías de platos de comidas en bandejas con 73 categorías distintas de alimentos.

Figura 15: Imagenes proporcionadas por el dataset utilizado [30]

Las fotografías tienen una dimensión de 3264 x2448 píxeles. El estudio además proporciona

un fichero *annotations.mat* con anotaciones. Tras pasarlo del archivo del tipo matlab a uno mas fácil de manejar en Python como es una hoja de cálculo Excel, se obtuvieron los valores referentes al alimento, el nombre de la imagen en la que se encuentra y 8 puntos cardinales para la localización de las cajas de detección. Estos datos son vitales para el entrenamiento del modelo de detección de objetos pues necesitaremos administrarle la etiqueta del plato a analizar y la localización del mismo.

20151127_114556	patate/pure	2000	1200	2680	1200	2680	1950	2000	1950
20151127_114556	pasta_mare_	843	667	1623	667	1623	1467	843	1467
20151127_114946	pasta_mare_	2119	1256	2921	1256	2921	2002	2119	2002
20151127_114946	pizza	978	378	2004	378	2004	2067	978	2067
20151127_114946	budino	289	1301	729	1301	729	1714	289	1714
20151127_114946	mandarini	550	574	814	574	814	822	550	822
20151127_114946	mandarini	477	920	741	920	741	1157	477	1157
20151127_115133	pasta_zaffer	1819	513	2850	513	2850	1404	1819	1404
20151127_115133	arrosto	367	943	1215	943	1215	1584	367	1584
20151127_115133	patate/pure	545	1265	1269	1265	1269	1793	545	1793
20151127_115133	mandarini	257	378	531	378	531	640	257	640
20151127_115151	pizza	1376	147	2706	147	2706	1647	1376	1647
20151127_115229	yogurt	1110	250	1600	250	1600	756	1110	756
20151127_115229	pane	2158	334	2778	334	2778	808	2158	808
20151127_115229	pasta_zaffer	1715	889	2658	889	2658	1882	1715	1882
20151127_115229	torta_salata_	389	826	1230	826	1230	1671	389	1671
20151127_115229	patate/pure	332	1343	943	1343	943	1917	332	1917
20151127_115424	pasta_mare_	491	515	1408	515	1408	1391	491	1391

Figura 16: Datos de anotaciones pertenecientes al dataset

Por otra parte, los puntos de coordenadas facilitados estaban localizados con la imagen en una posición específica, la imagen debía estar en horizontal y con la barra de servicio del comedor en la parte superior, por lo que a falta de una herramienta para automatizar ese proceso, se tuvo que girar hasta la posición deseada en pos de una correcta detección y procesado de los platos por parte del modelo de detección de objetos como se puede observar en la figura 6.2

6.3. Evaluación del modelo de detección de objetos

Para poder evaluar el correcto entrenamiento del modelo de detección de objetos, se han usado varias métricas de COCO [31]. Para ello se han de definir varios conceptos.

- **Puntuación de confianza (o *Confidence score*)**: la probabilidad de que la caja detectada contenga un objeto. Normalmente la predice un clasificador.
- ***Intersection over Union* (IoU)**: se define como el área de intersección dividida entre el

Cuadro 6.1: Imágenes proporcionadas y reajustadas. La barra superior metálica indica la correcta orientación de la imagen

área de unión de un cuadro delimitador B_p y la caja de referencia B_{gt}

$$IoU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \tag{1}$$

- ***Precision* y *Recall***

$$precision = \frac{TP}{TP + FP} \tag{2}$$

$$recall = \frac{TP}{TP + FN} \tag{3}$$

con TP como verdaderos positivos, FP como Falsos Positivos y FN como Falsos Negativos resultantes de compara el resultado del modelo con la referencia. (*groundtruth*).

6.3.1. *Average Precision* (AP)

$$AP = \sum_{i=1}^{n-1} (r_{i+1} - r_i) precision_{interp}(r_{i+1}) \tag{4}$$

Con $precision_{interp}$ como la máxima precisión encontrada en cualquier nivel de *recall* dentro del rango en el que estamos calculando.

6.3.2. *Average Recall* (AR)

$$AR = 2 \int_n^m recall(x)dx \tag{5}$$

con $[n, m]$ como el rango de IoU evaluado.

6.4. FASTER R-CNN - Modelo de detección de objetos

El modelo de detección que se ha usado es la *Faster R-CNN* [32], se trata de un modelo que usa dos redes: una red que en una primera pasada propone regiones donde podría haber un objeto y una segunda red que utiliza esas propuestas para detectar los objetos. Las dos pasadas que hace el modelo hacen que, pese a obtener resultados realmente buenos, suele ser demasiado lento para aplicaciones que necesitan funcionar en tiempo real. En nuestro caso el tiempo, al tratarse de décimas de segundos de diferencia, nos da igual porque no necesitamos tanta velocidad sino que priorizamos la fiabilidad de la respuesta por lo que se convierte en un modelo ideal para nuestra aplicación, el tiempo de análisis es de 0.4 segundos y tarda aproximadamente 3 horas en completar el entrenamiento.

Hemos utilizado uno de los modelos *faster* pre-entrenados que proporciona tensorflow [27]. Estos modelos están entrenados en el conjunto de datos *Common Objects in Context de Microsoft* (COCO) [31]. Este conjunto de datos es uno de los más utilizados ya que contiene una gran cantidad de imágenes con mas de 80 clases distintas de objetos comunes, así que la variedad de filtros que tiene aprendidos es muy alta.

El *Transfer-Learning* se ha hecho utilizando la API de Tensorflow 2 [33]. Para el entrenamiento se han usado las 44 clases que tenían mas de 20 apariciones en el *dataset*, se decidió no coger el resto de clases por falta de datos para el entrenamiento. Se ha reservado el 10 % del *dataset* para evaluarlo seleccionado aleatoriamente. La evaluación se ha realizado en base a las métricas de COCO utilizando la librería que provee a tal fin y los resultados han sido suficientemente altos.

Atendiendo a los datos obtenidos (Tabla 6.2) podemos confirmar que al tener un *Average Precision* alto nuestro modelo no suele identificar platos de comida donde no los hay. El *Average Recall* alto nos indica que nuestro modelo de detección no suele dejar platos sin identificar.

Un análisis clase por clase de los resultados como se puede observar en el cuadro 6.3, nos muestra que la mayoría de las clases evaluadas obtienen un resultado razonablemente alto, lo que nos indica que nuestro modelo es fiable para la labor que lo hemos adaptado. Resulta llamativo que las dos clases con peores resultados sean *Mandarini* y *Arancia*, esto puede ser debido a que representan a la mandarina y a la naranja respectivamente. Dado el parecido entre estas dos frutas, es razonable deducir que el modelo de detección de objetos las confunde

	Intersección sobre la Unión	area	maxDets	Resultado
Average Precision (AP)	0.50:0.95	all	100	0.833
Average Precision (AP)	0.50	all	100	0.941
Average Precision (AP)	0.75	all	100	0.914
Average Precision (AP)	0.50:0.95	small	100	-1.000
Average Precision (AP)	0.50:0.95	medium	100	0.653
Average Precision (AP)	0.50:0.95	large	100	0.829
Average Recall (AR)	0.50:0.95	all	1	0.851
Average Recall (AR)	0.50:0.95	all	10	0.862
Average Recall (AR)	0.50:0.95	all	100	0.862
Average Recall (AR)	0.50:0.95	small	100	-1.000
Average Recall (AR)	0.50:0.95	medium	100	0.680
Average Recall (AR)	0.50:0.95	large	100	0.859

Cuadro 6.2: Resultado cuantitativos en base a las métricas COCO

provocando errores en ambas.

También es importante señalar que las clases que obtienen puntuaciones muy altas se encuentran en la mitad baja de la tabla dado que la tabla está ordenada por apariciones en el dataset, probablemente en el conjunto de evaluación haya pocos ejemplos de estas clases que se deben haber acertado todos y por ello el resultado es anormalmente alto.

Pese a que hemos entrenado con 44 clases, en el cuadro solo se ve el resultado de 42 porque en el grupo de evaluación no se encontraba ningún elemento de dos de las clases.

El cálculo de los precios de los menús se calcula usando los resultados de la identificación realizada por el modelo entrenado, se devuelve el código de cada uno de los alimentos identificados y se obtienen de la base de datos el precio final del menú haciendo la suma de todos ellos. El resultado final en la aplicación se puede observar en la Figura 14.

Nº oc.	Nombre	AP	AR	Nº oc.	Nombre	AP	AR
479	Pane	0.918	0.855	198	Mandarini	0.573	0.321
161	Carote	0.785	0.790	150	Patata/pure	0.758	0.783
148	Cotoletta	0.866	0.877	131	Fagiolini	0.711	0.738
130	Yogurt	0.826	0.878	112	Budino	0.722	0.733
110	Spinaci	0.866	0.900	93	Scaloppine	0.724	0.812
89	Pizza	0.819	0.900	80	Pasta_vegetariano	0.658	0.680
73	Mele	0.787	0.800	70	Pasta_pesto	0.969	0.983
67	Zucchine_umido	0.869	0.917	66	Arancia	0.484	0.537
66	Lasagna_bolognese	0.935	0.943	65	Pasta_sugo_pesce	0.960	0.980
64	Pasta_cozze_e_vongole	0.720	0.920	64	Patatine_fritte	0.760	0.825
62	Arrosto	0.878	0.900	57	Riso_bianco	0.950	0.950
56	Medaglioni_di_carne	0.950	0.950	54	Torta_salata_spinaci	0.875	0.900
52	Pasta_zafferano	1.000	1.000	49	Patate_prosciutto	0.761	0.775
48	Torta_salata_rustica	1.000	1.000	46	Insalata_mista	0.721	0.757
45	Polpette_di_carne	0.759	0.771	45	Pasta_mare_e_monti	0.901	1.000
42	Pasta_pancetta	1.000	1.000	21	Salmone	0.600	0.600
39	Orecchiette_(ragu)	0.866	0.900	38	Pizzoccheri	0.972	0.980
37	Finocchi_gratinati	0.850	0.850	35	Pere	0.767	0.800
31	Pasta_tonno	0.711	0.725	31	Riso_sugo	1.000	1.000
30	Pasta_tonno_e_piselli	1.000	1.000	26	Piselli	0.800	0.800
25	Torta_salata_3	0.900	0.900	22	Torta_salata	1.000	1.000

Cuadro 6.3: Resultado cuantitativos en base a las métricas COCO por cada clase. Nº oc.(Número de ocurrencias)

Cuadro 6.4: Resultados cualitativos. En estas imágenes podemos ver dos ejemplos de bandejas de comida con la comida identificada, nombre de los alimentos identificados y su puntuación

7

Conclusiones y Líneas Futuras

7.1. Conclusiones

En este trabajo se ha afrontado un problema de creación y desarrollo de un sistema capaz de reconocer platos de comidas sobre una bandeja de comedor. El problema se plantea ante la necesidad de disminuir los tiempos de espera a la hora de pagar.

El campo de la visión por computador y las redes neuronales es un campo en continuo desarrollo debido a la gran investigación que se hace, ya sea por su potencial económico o por el interés que pueda suscitar una tecnología tan nueva e innovadora. Aprender a entrenar una red neuronal usando las técnicas de Transfer Learning y Fine Tuning ha sido un reto personal pues ha aumentado mi conocimiento sobre estas tecnologías.

El uso de metodologías como los microservicios o la arquitectura MVC supone que ambas aplicaciones están abiertas a una mayor expansión con una ampliación de sus funcionalidades. Aunque el número de directorios y archivos generados puede ser mayor que los tipos de datos que se manejan y las funcionalidades creadas no supone un impedimento para su reutilización pues la mayoría de esas funciones son de uso general y serán utilizadas en esa escalabilidad antes nombrada.

La realización de este proyecto ha sido un reto pues, aunque sabía los fundamentos de algunas tecnologías, he tenido que aprender un lenguaje de programación nuevo como es Python para la realización del mismo y he tenido que ampliar mis conocimientos acerca de Android para así crear ciertas funcionalidades como son las peticiones a una API o la toma de fotografías.

7.2. Líneas Futuras

Una linea principal y clara de desarrollo para este proyecto es la implantación en la aplicación de la UMA como nueva funcionalidad ya que esa fue la idea original, adaptar el sistema de login y ofrecer un sistema de pago para la automatización del mismo supondría un avance hacía el objetivo final del proyecto.

La segunda linea de proyecto obvia es la utilización de los datos recopilados por la red neuronal para crear una red inteligente de recomendaciones alimentarias, si se une con la linea planteada en el párrafo anterior se podría crear una aplicación que, analizando la comida de un usuario, le proponga cambios en los alimentos que suele tomar para mejorar sus hábitos de

alimentación. La utilización de esos datos por parte del cliente, el comedor, para la realización de menús acorde a los gustos de los usuarios medios también sería un buen punto en el que trabajar. Se podría hacer análisis de de estadísticas si se asociara información de usuario a un registro de sus comidas.

Referencias

[1] Universidad de Málaga. https://www.uma.es/, 2021.

[2] Amazon AWS. ¿Qué son los microservicios? https://aws.amazon.com/es/microservices/.

[3] Wikipedia. Operaciones CRUD. https://es.wikipedia.org/wiki/CRUD.

[4] Gianluigi Ciocca, Paolo Napoletano, and Raimondo Schettini. Food recognition: a new dataset, experiments and results. *IEEE Journal of Biomedical and Health Informatics*, 21(3):588–598, 2017.

[5] Wikipedia. Perceptrón multicapa. https://es.wikipedia.org/wiki/Perceptr%C3%B3n_multicapa#/media/Archivo:RedNeuronalArtificial.png.

[6] Wikipedia. https://upload.wikimedia.org/wikipedia/commons/6/63/Typical_cnn.png.

[7] Programador Click. Capa de agrupamiento de aprendizaje profundo. https://programmerclick.com/article/2420177865/.

[8] Dario Martinez. Is Transfer Learning the final step for enabling AI in Aviation? . https://datascience.aero/transfer-learning-aviation/.

[9] Wikipedia. Agile Software Development. https://en.wikipedia.org/wiki/Agile_software_development.

[10] Óscar Blancante. Arquitectura de Microservicios. https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/microservicios.

[11] Wikipedia. Modelo-Vista-Controlador. https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador.

[12] SQLite. SQLite Documentation. https://www.sqlite.org/docs.html.

[13] Wikipedia. SQL-92. https://es.wikipedia.org/wiki/SQL-92.

[14] Python. Python Documentation. https://docs.python.org/3.7/. [Version used: 3.7].

[15] Flask. Flask web development, one drop at a time. https://flask.palletsprojects.com/en/2.0.x/#.

[16] Jinja. Jinja documentation. https://jinja.palletsprojects.com/en/3.0.x/.

[17] Werkzeug. Werkzeug documentation. https://werkzeug.palletsprojects.com/en/2.0.x/.

[18] Wikipedia. Web Server Gateway Interface. https://en.wikipedia.org/wiki/Web_Server_Gateway_Interface.

[19] Bootstrap. Bootstrap. https://getbootstrap.com/docs/5.0/getting-started/introduction/.

[20] Tensorflow. Tensorflow Documentation. https://devdocs.io/tensorflow~2.3/. [Version used: 2.3.1].

[21] Wikipedia. Computer Unified Devide Architecture. https://es.wikipedia.org/wiki/CUDA.

[22] Numpy. Numpy Documentation. https://numpy.org/doc/.

[23] Pandas. Pandas Documentation. https://pandas.pydata.org/docs/reference/index.html.

[24] Matplotlib. Matplotlib Documentation. https://matplotlib.org/stable/contents.html#.

[25] OpenCV2. OpenCV2 Documentation. https://docs.opencv.org/master/index.html.

[26] Pillow. Pillow Documentation. https://pillow.readthedocs.io/en/stable/reference/index.html.

[27] Tensorflow. Tensorflow Pre Trained Model. https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md.

[28] Fotoapparat Github. https://github.com/RedApparat/Fotoapparat.

[29] Retrofit2. https://square.github.io/retrofit/.

[30] Raimondo Schettini Gianluigi Ciocca, Paolo Napoletano. Food Recognition. http://www.ivl.disco.unimib.it/activities/food-recognition/.

[31] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

[32] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.

[33] Google. Tensorflow 2 Documentation. https://www.tensorflow.org/api_docs.

[34] Android Studio. Fuía de usuario. https://developer.android.com/studio/intro?hl=es-419.

Apéndice A
Manual de Instalación

A.1. Aplicación Web

Para el correcto uso de la aplicación web, será necesario la instalación del IDE de desarrollo Python: Pycharm. Se puede descargar desde la página oficial

Figura 17: Vista principal página web PyCharm

Una vez instalado PyCharm será necesario tener una versión compatible de Python, la versión correcta para el uso de esta aplicación web es Python 3.7. Será importante seleccionar, una vez que se ha ejecutado el instalador, la casilla de *Add Python 3.7 to PATH* para facilitar su uso. Como se puede observar en la figura A.1

Tras instalar las aplicaciones necesarias para ejecutar el proyecto, será necesario su descarga desde el repositorio en el que se encuentra y mediante el método más cómodo para el

Figura 18: Vista instalador Python 3.7

usuario (mediante GIT o descargando el proyecto empaquetado en un archivo ZIP) descargar en el directorio local deseado. Para abrirlo desde Pycharm únicamente habrá que seleccionar la opción File ->Open y seleccionarlo desde la localización donde se haya descargado en nuestro directorio local.

Es necesario la instalación de las bibliotecas a las que hace referencia la sección 6.1

Finalmente solo habrá que pulsar *Mayús+F10* para ejecutar el proyecto o pulsar el botón verde de inicio en la esquina superior derecha.

A.2. Aplicación Android

Para poder usar la aplicación Android es necesario descargar la aplicación de Android Studio desde la página oficial. Con la instalación de este software en la computadora también podremos instalar el sdk de Android y un dispositivo virtual para poder ejecutar la aplicación en nuestro equipo.

Una vez descargado el archivo de instalación, ha de ejecutarse para iniciar el proceso de

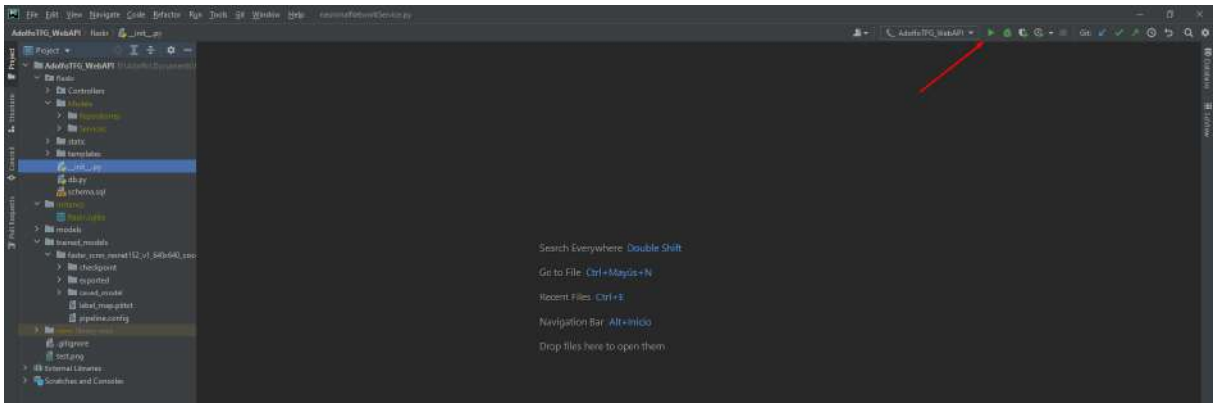


Figura 19: Vista principal PyCharm

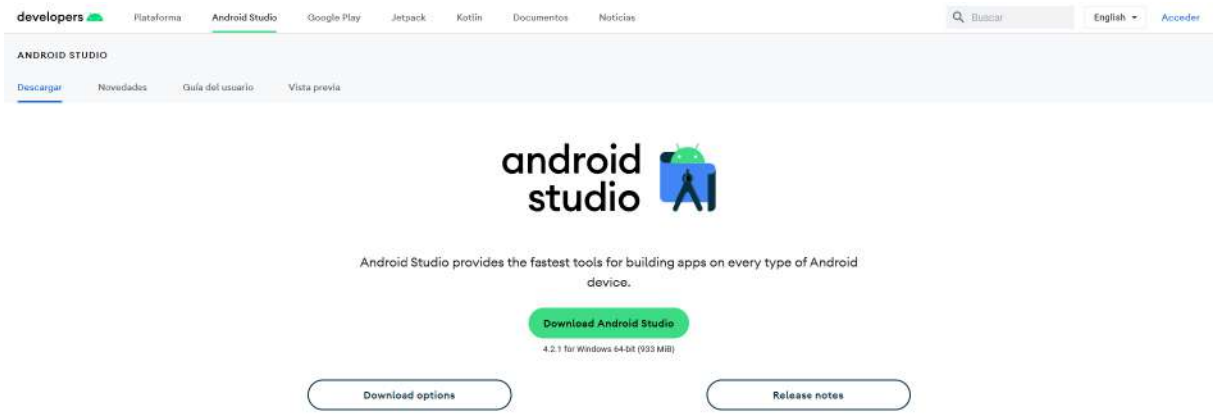


Figura 20: Página principal Android Studio

instalación. Se abrirá el instalador y habrá que tener especial atención en la ventana de componentes a instalar, pues deberemos marcar la casilla *Android Virtual Device* para así tener acceso a los dispositivos virtuales donde ejecutaremos la aplicación, como se puede observar en la Figura A.2

En la guía del usuario de Android Studio [34] se describe el resto de procesos de instalación y configuración del entorno.

Para descargar el proyecto deberemos ir al repositorio donde se encuentra y, mediante el método más cómodo para el usuario (mediante GIT o descargando el proyecto empaquetado en un archivo ZIP) descargar en el directorio local deseado. Una vez descargado, desde Android Studio, se selecciona usando la opción File ->Open.

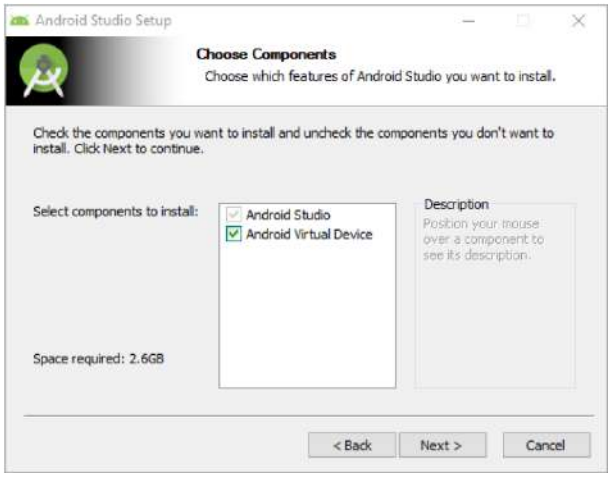


Figura 21: Opción de instalación de herramientas de dispositivos virtuales en Android Studio

Por último será necesaria la creación de un dispositivo virtual, en caso de no querer usar un dispositivo virtual se podrá conectar un dispositivo Android a la computadora en modo debug para que Android Studio lo reconozca y permita la instalación de la app en el dispositivo. Para crea el dispositivo web, habrá que pulsar el botón que se encuentra en la parte superior derecha para abrir el panel de *Android Virtual Device Manager*, seguidamente seleccionaremos el botón de *Create Virtual Device* donde se nos abrirá el panel de creación para seleccionar el dispositivo deseado, si pulsamos el botón *Next* podremos seleccionar la versión de Android a instalar, **Oreo** en nuestro caso.

Por último para ejecutar la aplicación solo tendremos que pulsar *Mayús+F10* y se instalará y ejecutará en la aplicación virtual que hemos creado.

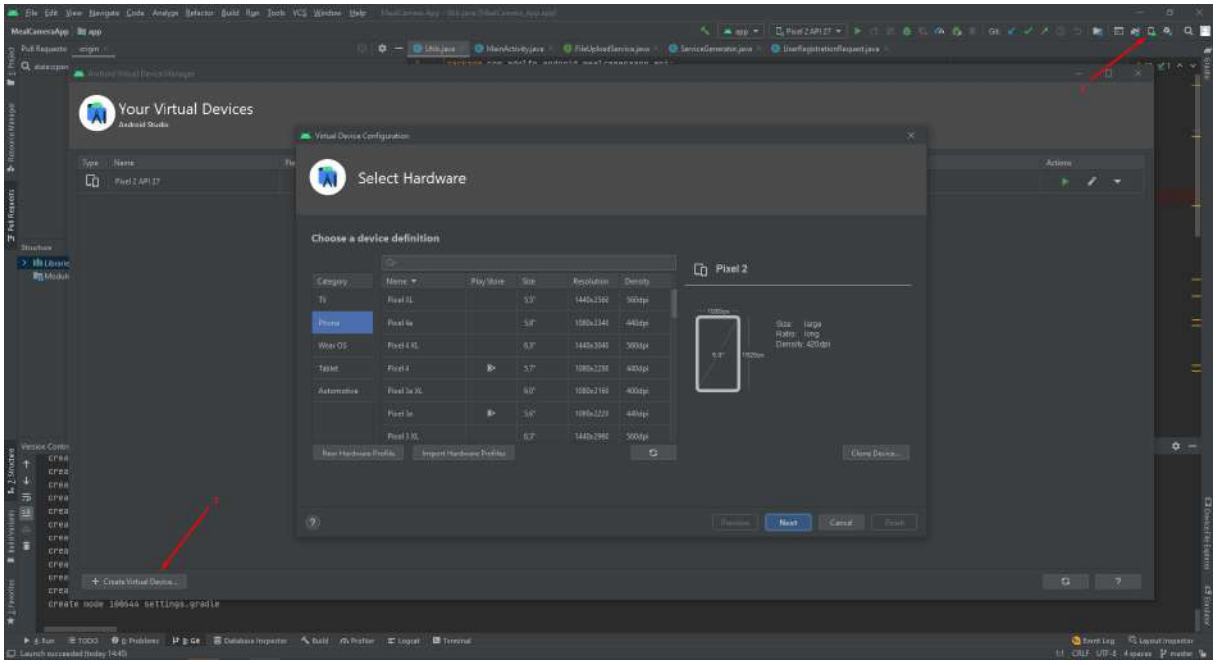


Figura 22: Menú creación Dispositivo virtual

Apéndice B
Manual de Usuario

B.1. Aplicación Web

Para acceder a la aplicación web se hará a través de la URL en la que se haya desplegado, por defecto será `http://127.0.0.1:5000/`. Las acciones sobre el menú estarán restringidas hasta que el usuario haya accedido con un usuario válido usando los *links* que se encuentran en la esquina superior derecha. Una vez seleccionado la opción de *Registro* o *Acceder*, el usuario será redirigido a un panel donde ejecutar la opción seleccionada, si la opción seleccionada era *Registro* y los datos que ha introducido son válidos se redirigirá de nuevo al panel de Acceso para poder ingresar en la aplicación.

Figura 23: Menú de login aplicación web

Una vez registrado en la aplicación y solo si tiene permisos para ello, el usuario será redirigido al menú principal y podrá añadir nuevos platos o editar los existentes. Para Añadir un nuevo plato simplemente tendrá que seleccionar el botón *Añadir* que se encuentra en la esquina superior izquierda justo encima del listado de platos y para la edición de un plato habrá de pulsar el botón *Editar* que se encuentra a la derecha de cada plato en la tabla que lista los platos disponibles. Es posible la inclusión o retirada de un plato del menú con la pulsación del *checkbox* bajo la columna *En el menú*.

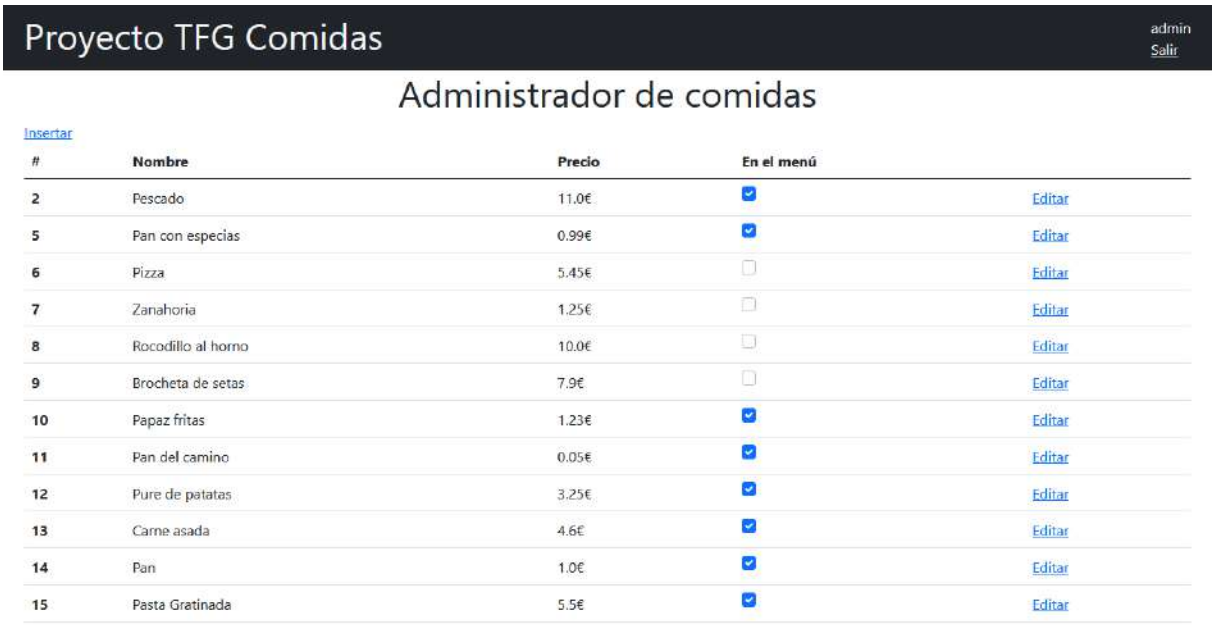


Figura 24: Menú principal aplicación web

El panel de edición contará con la posibilidad de editar tanto el nombre como el precio del plato seleccionado, o en el caso de haber seleccionado la opción *Añadir* en el menú principal, estarán vacíos para ingresar los nuevos datos de platos para el menú. El botón de borrado únicamente estará habilitado a través de la opción *Editar* en platos existentes.

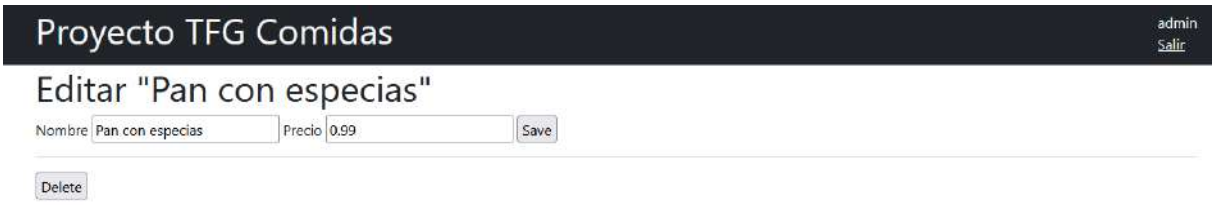
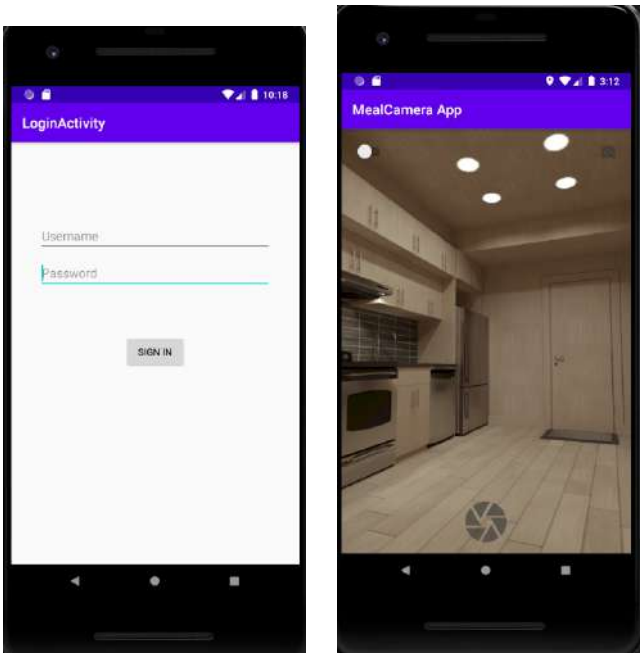


Figura 25: A la derecha se encuentra la pantalla de login y a la derecha la pantalla de la cámara

B.2. Aplicación Android

Al iniciar la aplicación Android aparecerá la pantalla de login, para poder acceder es necesario el registro previo a través de la aplicación web ya que se necesita un usuario válido. Se

introduce el usuario y contraseña correcto y se pulsa el botón *Sign in*. Si el usuario es válido será redirigido a la aplicación de toma de fotos.



Cuadro B.1: A la derecha se encuentra la pantalla de login y a la derecha la pantalla de la cámara

Figura 26: Ejemplo de bandeja fotografiada en la posición correcta

Figura 27: Para la realización de pruebas se ha usado un modelo para inicializar la app de la cámara, las fotos enviadas son para pruebas y estaban almacenadas en el dispositivo.

Es necesario la disposición de la bandeja en una posición en la que todos los platos sean visibles desde el eje vertical, una vez colocada se tomará la foto usando la aplicación, la cual la enviará al servidor devolviendo el precio y la lista con los alimentos identificados en el mismo.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA