Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

DESARROLLO DE UNA PLATAFORMA PARA LA APLICACIÓN DE MÉTODOS DE DEEP LEARNING

Alumno: Adriano Bravo Guzmán

Tutor: Prof. D. Antonio Jesús Rivera Rivas
Dpto: Departamento de Informática

Tutor: Prof. Dña. María Dolores Pérez Godoy
Dpto: Departamento de Informática

Junio, 2023

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Antonio Jesús Rivera Rivas y Doña María Dolores Pérez Godoy, tutores del Proyecto Fin de Carrera titulado: Desarrollo de una plataforma para la aplicación de métodos de Deep Learning, que presenta Adriano Bravo Guzmán, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Junio de 2023

El alumno:

BRAVO GUZMAN ADRIANO - 21025551D
Firmado digitalmente por BRAVO GUZMAN ADRIANO - 21025551D
Fecha: 2023.06.18 12:00:55 +02'00'

Adriano Bravo Guzmán

Los tutores:

RIVERA RIVAS ANTONIO JESUS - 26015863B
Firmado digitalmente por RIVERA RIVAS ANTONIO JESUS - 26015863B
Fecha: 2023.06.16 10:16:07 +02'00'

Antonio Jesús Rivera Rivas

PEREZ GODOY MARIA DOLORES - 75093149N
Firmado digitalmente por PEREZ GODOY MARIA DOLORES - 75093149N
Fecha: 2023.06.16 09:40:50 +02'00'

María Dolores Pérez Godoy

Índice

1. INTRODUCCIÓN Y MOTIVACIÓN....13
1.1. Introducción....13
1.2. Motivación....15
1.3. Objetivos del trabajo....15
1.4. Estructura de la memoria....16
2. CONTEXTO DEL TRABAJO....18
2.1. Inteligencia Artificial vs Machine Learning vs Deep Learning....18
2.1.1. Redes Neuronales Artificiales....19
2.1.1.1. Arquitectura....20
2.1.1.2. Tipos de redes neuronales....22
2.1.2. Capacidades de las redes neuronales....23
2.1.3. ¿Dónde se utilizan las redes neuronales?....24
2.2. TensorFlow Hub....25
2.2.1. Introducción TensorFlow Hub....25
2.2.2. TensorFlow Hub....26
2.2.3. Modelos de TensorFlow Hub....28
2.2.4. Dominios que utiliza DL4Vision....29
2.3. Herramientas....30
2.3.1. Framework y lenguaje de programación....30
2.3.2. IDL de desarrollo....35
2.3.3. Base de datos....36
2.3.4. Bibliotecas Python....37
2.3.5. Docker y Docker Hub....38
2.3.6. Sistemas Operativos....40
2.3.7. Drivers de Nvidia....42
2.3.8. Google Collab....43
2.3.9. Gestión de código....44
2.4. Metodologías de desarrollo software....44
2.4.1. Metodologías de desarrollo de software tradicionales....45
2.4.2. Metodologías de desarrollo de software ágiles....46
2.4.3. Elección de metodología del software....46
2.4.4. Desarrollo incremental....47
3. ANÁLISIS Y PLANIFICACIÓN....49
3.1. Captura de requisitos....49

3.2. Requisitos funcionales 51
3.2.1. Actor usuario 51
3.2.2. Actor principal administrador 59
3.3. Requisitos no funcionales 67
3.4. Planificación temporal 68
3.5. Estimación de costes 72
4. DISEÑO 74
4.1. Diagramas de secuencia 74
4.1.1. Diagrama de secuencia de usuario normal 74
4.1.2. Diagrama de secuencia de usuario administrador 83
4.2. Diagrama de clases 93
4.3. Base de datos y elementos de la aplicación 94
4.3.1. Diagrama de la base de datos 95
4.3.2. Valores de las clases implementadas 96
4.3.2.1. Categorías 97
4.3.2.2. Datasets 97
4.4. Arquitectura de la aplicación 100
5. DESARROLLO 102
5.1. Descripción de la solución propuesta 102
5.2. Evolución del proyecto 102
5.3. Front 103
5.4. Back. 105
5.4.1. Cargar modelos 105
5.4.2. Flujo de las vistas 107
5.4.3. Cámara 108
5.4.4. Script de instalación 109
5.5. Incrementos realizados. 110
6. PROBLEMÁTICAS ENCONTRADAS Y SOLUCIONES 112
6.1. Generalización de la aplicación para cualquier modelo 112
6.2. Limitación de portabilidad 113
6.3. Pruebas lentas 113
6.4. Problemas con el equipo 114
7. PRUEBAS CON USUARIOS 115
8. MEJORAS DE FUTURO 118
9. CONCLUSIÓN 119

10. APENDICE 120
10.1. Guía original del Trabajo Fin de Grado. 120
11. MANUALES 122
11.1. Guion de instalación 122
11.1.1. Comprobación 122
11.1.2. Descargas 125
11.1.2.1. Comprobar drivers de Nvidia e instalarlos 125
11.1.2.2. Ejecutar script 126
11.1.3. Ejecución 128
11.1.4. Recomendación 131
11.2. Guion de usuario 133
11.2.1. Introducción 133
11.2.2. Vista de inicio 134
11.2.3. Vista de categorías 135
11.2.4. Vista de bases de datos 136
11.2.5. Vista de opciones 137
11.2.6. Vista de resultado 142
11.3. Guion de administrador 143
11.3.1. Introducción 143
11.3.2. Entrar en modo administrador 143
11.3.3. Añadir elementos a la interfaz 145
11.3.4. Insertar modelo funcional 151
11.3.4.1. TensorFlow Hub 152
11.3.4.2. Insertar modelo en la aplicación 156
11.3.5. Consejo 160
12. DEFINICIONES Y ABREVIATURAS 163
12.1. Siglas 163
12.2. Referencias 164

Índice de ilustraciones

Ilustración 1: Comparativa entre Inteligencia Artificial, Deep Learning y Machine Learning. Fuente. 18

Ilustración 2: Capas de red neuronal sencilla. Fuente. 20

Ilustración 3: Capas de red neuronal compleja. Fuente. 22

Ilustración 4: Logo TensorFlow Hub (1). Fuente. 26

Ilustración 5: Página web de inicio de TensorFlow Hub (1) 27

Ilustración 6: Página web de inicio de TensorFlow Hub (2) 28

Ilustración 7: Captura de pantalla de los problemas de dominio de la página TensorFlow Hub 29

Ilustración 8: Logo del Framework Spring Boot y Angular. Fuente 1 y Fuente 2. 30

Ilustración 9: Logo de Java. Fuente 31

Ilustración 10: Logo de Python. Fuente. 31

Ilustración 11: Logo de FastAPI. Fuente. 32

Ilustración 12: Logo de Flask. Fuente. 32

Ilustración 13: Logo de Django. Fuente. 33

Ilustración 14: Comparación entre Flask y Django. Fuente. 34

Ilustración 15: Logo Visual Studio Code. Fuente. 35

Ilustración 16: Logo PyCharm. Fuente. 35

Ilustración 17: Logo SQLite. Fuente. 36

Ilustración 18: Logo PostgreSQL. Fuente. 37

Ilustración 19: Logos de algunas bibliotecas de Python usadas en DL4Vision 38

Ilustración 20: Logo de Docker. Fuente. 39

Ilustración 21: Logo de Docker Hub. Fuente. 40

Ilustración 22: Logos de Windows y Linux. Fuente. 41

Ilustración 23: Logo de Nvidia Cuda y drivers cuDNN. Fuente 1 y Fuente 2. 43

Ilustración 24: Logo de Docker Hub. Fuente. 43

Ilustración 25: Logo de Git y Github. Fuente 1 y Fuente 2 44

Ilustración 26: Etapas de un incremento en un modelo incremental. Fuente. 48

Ilustración 27: Diagrama de casos de uso del sistema de un usuario 50

Ilustración 28: Diagrama de casos de uso del sistema de un administrador 51

Ilustración 29: Diagrama de secuencia de código de usuario normal 1 75

Ilustración 30: Diagrama de secuencia de código de usuario normal 2 75

Ilustración 31: Diagrama de secuencia de código de usuario normal 3 76

Ilustración 32: Diagrama de secuencia de código de usuario normal 4 77

Ilustración 33: Diagrama de secuencia de código de usuario normal 5 77

Ilustración 34: Diagrama de secuencia de código de usuario normal 6 79

Ilustración 35: Diagrama de secuencia de código de usuario normal 7 80

Ilustración 36: Diagrama de secuencia de código de usuario normal 8 81

Ilustración 37: Diagrama de secuencia de código de usuario normal 9 81

Ilustración 38: Diagrama de secuencia de código de usuario normal 10 82

Ilustración 39: Diagrama de secuencia de código de usuario normal 11 82

Ilustración 40: Diagrama de secuencia de código de usuario administrador 1 83

Ilustración 41: Diagrama de secuencia de código de usuario administrador 2 84

Ilustración 42: Diagrama de secuencia de código de usuario administrador 3 85

Ilustración 43: Diagrama de secuencia de código de usuario administrador 4 86

Ilustración 44: Diagrama de secuencia de código de usuario administrador 5 87

Ilustración 45: Diagrama de secuencia de código de usuario administrador 6 88

Ilustración 46: Diagrama de secuencia de código de usuario administrador 7 89

Ilustración 47: Diagrama de secuencia de código de usuario administrador 8 90

Ilustración 48: Diagrama de secuencia de código de usuario administrador 9 91

Ilustración 49: Diagrama de secuencia de código de usuario administrador 10 92

Ilustración 50: Diagrama de secuencia de código de usuario administrador 11 93

Ilustración 51: Diagrama de clases....94

Ilustración 52: Diagrama Entidad-Relación de DL4Vision....95

Ilustración 53: Logo COCO. Fuente....98

Ilustración 54: Logo Imagenet. Fuente. 99

Ilustración 55: Imagen resultante de un modelo de detección de objetos entrenado con ADE20K. Fuente. 100

Ilustración 56: Imagen de arquitectura de Django. Fuente....101

Ilustración 57: Imagen del flujo de todas las vistas....104

Ilustración 58: Imagen del flujo de los modelos de ejemplo de TensorFlow Hub 106

Ilustración 59: Imagen del flujo de los modelos de DL4Vision 107

Ilustración 60: Imagen del flujo de las vistas....108

Ilustración 61: Cartel de La noche Europea de l@s Investigador@es 2022. Fuente....115

Ilustración 62: Foto de usuarios interaccionando con la aplicación....116

Ilustración 63: Campus GEM....117

Ilustración 64: Captura de pantalla de escritorio de Linux....122

Ilustración 65: Captura de pantalla de buscador de Linux 123

Ilustración 66: Captura de búsqueda de aplicación Cheese de Linux....123

Ilustración 67: Captura de búsqueda del Terminal de Linux 124

Ilustración 68: Captura del Terminal al ejecutar el comando sudo apt-get install cheese....124

Ilustración 69: Captura del Terminal al ejecutar el comando nvidia-smi....125

Ilustración 70: Captura del Terminal al ejecutar el comando ubuntu-drivers devices....125

Ilustración 71: Captura del Script de instalación....126

Ilustración 72: Captura de la carpeta DL4Vision con el Script dentro....127

Ilustración 73: Captura de mensaje al ejecutar el script (1) 127

Ilustración 74: Captura de mensaje al ejecutar el script (2) 128

Ilustración 75: Captura de mensaje al ejecutar el script (3) 128

Ilustración 76: Captura carpeta DL4Vision con el archivo DL4Vision.sh 129

Ilustración 77: Captura del terminal al ejecutar DL4Vision.sh 130

Ilustración 78: Captura del terminal donde muestra que los modelos se han cargado.....130

Ilustración 79: Captura de la vista inicial de DL4Vision.....131

Ilustración 80: Captura de la ventana Configuración en la pestaña de Energía 132

Ilustración 81: Captura de la ventana Preferencias 133

Ilustración 82: Captura de vista de inicio de DL4Vision 134

Ilustración 83: Captura de vista de categorías de DL4Vision.....136

Ilustración 84: Captura de vista de base de datos de DL4Vision 137

Ilustración 85: Captura de vista de base de opciones de DL4Vision (1) 137

Ilustración 86: Captura de vista de base de opciones de DL4Vision (2) 138

Ilustración 87: Captura de vista de elegir imagen de DL4Vision 138

Ilustración 88: Captura de vista de elegir una url de DL4Vision.....139

Ilustración 89: Captura de vista de elegir una url al introducir una url no valida de DL4Vision 140

Ilustración 90: Captura de vista de elegir una url al introducir una url valida de DL4Vision.140

Ilustración 91: Captura de vista de tomar una imagen.....141

Ilustración 92: Captura de vista de tomar una imagen después de pulsar el botón de la cámara.....141

Ilustración 93: Captura de vista de tiempo real.....142

Ilustración 94: Captura de vista de resultado (1).....142

Ilustración 95: Captura de vista de resultado (2).....143

Ilustración 96: Captura de vista de login.....144

Ilustración 97: Captura de vista de administrador.....144

Ilustración 98: Captura de vista de categorías y vista inicial de DL4Vision 145

Ilustración 99: Captura de vista de categorías y vista de categorías de DL4Vision.....146

Ilustración 100: Captura de ventana para añadir categoría 146

Ilustración 101: Captura de ventana de paleta de colores 147

Ilustración 102: Captura de vista de categorías con mensaje de confirmación y vista de categorías al ser refrescada 148

Ilustración 103: Captura de vista de categorías al seleccionar una categoría.....149

Ilustración 104: Captura de ventana de emergente para confirmar si se desea borrar 150

Ilustración 105: Captura de vista de modelos en la parte de administrador 151

Ilustración 106: Captura de página de inicio de TensorFlow Hub 152

Ilustración 107: Captura de página TensorFlow Hub con el filtro del dominio de imágenes153

Ilustración 108: Captura de página TensorFlow Hub con el filtro del dominio de imágenes y todos los filtros borrados 153

Ilustración 109: Captura de página TensorFlow Hub del filtro TF Version 154

Ilustración 110: Captura de página TensorFlow Hub señalando donde aparece el dataset de un modelo 154

Ilustración 111: Captura de página TensorFlow Hub al haber seleccionado un modelo 155

Ilustración 112: Captura de vista añadir un modelo de la parte de administrador y la página TensorFlow Hub al haber seleccionado un modelo 156

Ilustración 113: Captura de vista añadir un modelo de la parte de administrador relleno y la página TensorFlow Hub al haber seleccionado un modelo señalando el botón Copy URL.157

Ilustración 114: Captura de vista añadir un modelo de la parte de administrador relleno y la página TensorFlow Hub al haber seleccionado un modelo señalando el ancho y alto.....157

Ilustración 115: Captura comparando las opciones para que sean iguales.....158

Ilustración 116: Captura de la vista de modelos en la parte de administrador con el nuevo modelo 159

Ilustración 117: Captura de la vista de resultados al haber utilizado el nuevo modelo.....160

Ilustración 118: Captura de página de TensorFlow Hub al elegir un modelo que tiene asociado un notebook de Colab asociado 161

Ilustración 119: Captura de Colab con modelos de TensorFlow Hub (1) 161

Ilustración 120: Captura de Colab con modelos de TensorFlow Hub (2) 162

Índice de tablas

Tabla 1: Tabla de casos de uso de un usuario normal 52

Tabla 2: Tabla de código de usuario normal 1....53

Tabla 3: Tabla de código de usuario normal 2....53

Tabla 4: Tabla de código de usuario normal 3....54

Tabla 5: Tabla de código de usuario normal 4....54

Tabla 6: Tabla de código de usuario normal 5....55

Tabla 7: Tabla de código de usuario normal 6....56

Tabla 8: Tabla de código de usuario normal 7....56

Tabla 9: Tabla de código de usuario normal 8....57

Tabla 10: Tabla de código de usuario normal 9....57

Tabla 11: Tabla de código de usuario normal 10....58

Tabla 12: Tabla de código de usuario normal 11....58

Tabla 13: Tabla de casos de uso de un usuario administrador....59

Tabla 14: Tabla de código de usuario administrador 1 60

Tabla 15: Tabla de código de usuario administrador 2 60

Tabla 16: Tabla de código de usuario administrador 3 61

Tabla 17: Tabla de código de usuario administrador 4 62

Tabla 18: Tabla de código de usuario administrador 5 63

Tabla 19: Tabla de código de usuario administrador 6 63

Tabla 20: Tabla de código de usuario administrador 7 64

Tabla 21: Tabla de código de usuario administrador 8 65

Tabla 22: Tabla de código de usuario administrador 9 65

Tabla 23: Tabla de código de usuario administrador 10 66

Tabla 24: Tabla de código de usuario administrador 11 67

Tabla 25: Tabla resumen de la planificación temporal....69

Tabla 26: Diagrama de Gantt (1) 70

Tabla 27: Diagrama de Gantt (2) 70

Tabla 28: Diagrama de Gantt (3) 71

Tabla 29: Diagrama de Gantt (4) 71

Tabla 30: Diagrama de Gantt (5) 72

Tabla 31: Tabla resumen de gastos 73

1. INTRODUCCIÓN Y MOTIVACIÓN

En este capítulo se presenta la especificación del trabajo, con una estructura y contenidos inspirados en los criterios y recomendaciones que establece la norma UNE 157801:2007 - “Criterios Generales para la elaboración de proyectos de Sistemas de información”.

A lo largo del documento se utilizarán términos y acrónimos cuya descripción aparecen en el apartado 12 (Definiciones y abreviaturas).

1.1. Introducción

Actualmente vivimos en un mundo donde la tecnología marca la diferencia en todos los aspectos de nuestras vidas, hasta tal punto que hace que se establezca algo similar a una jerarquía entre las personas, así como lo hace el dinero. Aquellas empresas que dispongan de mayores avances tecnológicos, serán las que probablemente tengan más éxito entre los clientes. Hoy en día vivimos en una era digital, en la cual encontramos pantallas por todos lados, llenas de información y datos que nos bombardean constantemente, y es imprescindible saber cómo gestionarlo para poder sacar el máximo beneficio, tal y como dijo el matemático e investigador Clive Humby en 2006 “los datos son el nuevo petróleo”.

Esto es algo que va en aumento, del mismo modo que la humanidad evoluciona a pasos agigantados. A medida que vamos evolucionando, la tecnología pasa a ser más asequible y a estar a disposición de una mayor cantidad de gente. Es por esto que las nuevas generaciones están creciendo con distintos dispositivos que les facilitan la vida y es cuestión de tiempo que introduzcan estas tecnologías definitivamente en sus vidas. Encontramos tecnología desde que nacemos, con la cámara de videovigilancia que nos ponen nuestros padres cuando somos bebés para controlar si nos despertamos o lloramos; cuando somos niños, todos esos dispositivos tradicionales cada vez más informatizados; cuando somos adolescentes, con teléfonos, tablets u ordenadores de nuestros padres o hermanos; y por supuesto cuando somos adultos adquirimos una gran cantidad de aparatos y utensilios

tecnológicos sin realmente darnos cuenta. Y ya no solo es la gran variedad que hay disponible, sino también la cantidad de posibilidades que nos ofrecen, desde algo tan sencillo como tomarnos una foto hasta poder aplicar filtros en los que cambia el color de la imagen, e incluso algunos que usan el reconocimiento facial para añadir objetos audiovisuales alrededor, aquí es donde entra, la Inteligencia Artificial.

La Inteligencia Artificial aparenta ser una rama lejana y misteriosa, ya que es la responsable de simular las capacidades del ser humano mediante algoritmos, y, aunque esto nos suene muy lejos de la realidad, encontramos algoritmos en todas partes; en el contenido audiovisual que Netflix nos recomienda en base a lo que hemos visto con anterioridad, la probabilidad que tiene un cliente de devolver un crédito en base a sus ingresos e incluso en un teléfono móvil con el reconocimiento facial.

Estos son algunos ejemplos en dónde se puede ver de lo que se encarga la Inteligencia Artificial, pero para poder comprenderla un poco mejor, a lo largo del documento, la definiremos, entenderemos cómo esta repercute en la actualidad y los subconjuntos en los que se divide (entre los que se encuentra Machine Learning).

Este proyecto está englobado dentro del subconjunto de Machine Learning, más exactamente, está clasificado en la subcategoría de Deep Learning. A lo largo de este escrito profundizaremos en ambos conceptos.

Para la realización de esta aplicación, denominada DL4Vision, se han usado varios modelos de Deep Learning sobre imágenes para predecir una salida, concretamente podemos dividir los modelos en 4 grandes categorías: clasificación de imágenes, detección de objetos, segmentación de objetos y detección de pose. Con estos modelos se puede ver la importancia que pueden llegar a tener unas líneas de código y el motivo por el cual se les da tanto valor a los datos en la actualidad.

Los modelos que han sido usados en el desarrollo de la aplicación, son modelos de una biblioteca llamada TensorFlow Hub, por lo que podríamos decir que la aplicación realizada con motivo de este **TFG**, hace de intermediaria entre los modelos y el usuario.

1.2. Motivación

Actualmente, si queremos usar un modelo de Deep Learning hay que saber programar y todo el proceso que conlleva (datasets, tipos…). Es difícil de creer que haya modelos hechos en internet y que estos estén desaprovechados porque no pueden ser usados por cualquier usuario. Por tanto, el objetivo de esta aplicación es darle la posibilidad a los usuarios de añadir nuevos modelos, y usar los ya implementados, basándose en que todos tienen un código común, sin la necesidad de tener conocimientos de programación, es decir, darles una herramienta para que puedan testear y probar modelos de Deep Learning. Pero sobre todo, al proyecto se le ha dado un enfoque de escalabilidad, para que la aplicación no solo se pueda usar a nivel de cliente, sino que además, se pueda desplegar en un servidor y poder añadir desde la administración de la aplicación cualquier modelo de forma rápida y sencilla.

En resumen, este TFG permite que, al ejecutar la aplicación, cualquier usuario pueda usar los modelos de Deep Learning de una manera cómoda, introduciendo imágenes desde una carpeta o con un enlace. Incluso encontraremos algunos modelos que actúan en tiempo real. Adicionalmente, y con motivo de ciertos contactos con el Parque de las Ciencias de Granada, se concertó desarrollar esta aplicación para incluirla en la exposición "Inteligencia Artificial. Una exposición sobre las personas, los datos y el control"

1.3. Objetivos del trabajo

Este proyecto denominado “Desarrollo de una plataforma para la aplicación de métodos de Deep Learning”, ha sido realizado teniendo en cuenta el cumplimiento de los siguientes objetivos.

- Desarrollo de un módulo que permita la obtención y preprocesamiento de datos (imágenes).
- Desarrollo de diferentes módulos que permitan la aplicación de modelos Deep Learning en diferentes tareas de visión por computador.

- Desarrollo de un módulo que permita el análisis y visualización de los resultados.
- Desarrollo de una herramienta que permita la implantación de los módulos anteriores.

1.4. Estructura de la memoria

La presente memoria se estructura de la siguiente manera:

- **Primer capítulo: Introducción y motivación.** En este capítulo se realiza una introducción sobre los campos de la Inteligencia Artificial, y en específico el Deep Learning, actualmente, haciendo énfasis en su importancia y su utilidad.
- **Segundo capítulo: Contexto del trabajo.** Aquí se profundiza en el Deep Learning, diferenciando entre Machine Learning y la Inteligencia Artificial. Además, se comenta cómo son, algunos tipos y las utilidades de las redes neuronales. Para finalizar el apartado de contexto, se va a mostrar la plataforma de TensorFlow Hub y qué se puede encontrar en ella.
- **Tercer capítulo: Análisis y planificación.** Se verán todas las herramientas que han ido apareciendo a lo largo del desarrollo y las razones por las que se ha elegido cada una comparándolas con algunas parecidas.
- **Cuarto capítulo: Diseño.** En este apartado se especifica mediante diferentes herramientas el diseño o arquitectura del software.
- **Quinto capítulo: Desarrollo.** Se comenta sin entrar en el código como está estructurada la aplicación lógicamente.
- **Sexto capítulo: Problemáticas encontradas y soluciones.** Se verán todas las problemáticas que han ido apareciendo durante el desarrollo de la aplicación y cómo se han superado.

- **Séptimo capítulo: Pruebas con usuarios**: En este apartado se van a comentar las distintas ocasiones en las que usuarios han podido interactuar con la aplicación.

- **Octavo capítulo: Mejoras de futuro:** Se verán posibles mejoras de futuro para la aplicación.

- **Capítulos finales:** Conclusión, apéndices, manuales, definiciones y abreviaturas.

2. CONTEXTO DEL TRABAJO

2.1. Inteligencia Artificial vs Machine Learning vs Deep Learning

Esta sección comprende 3 conceptos que se suelen confundir, de forma que a lo largo de este escrito los definiremos, veremos las diferencias y por último profundizaremos en el Deep Learning que es el concepto más importante de este TFG.

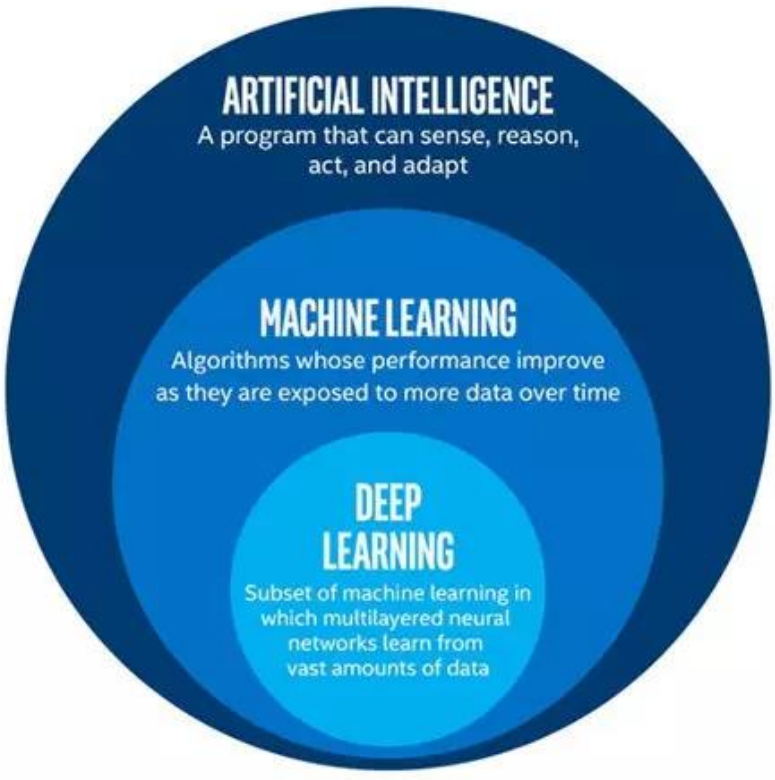


Ilustración 1: Comparativa entre Inteligencia Artificial, Deep Learning y Machine Learning. Fuente.

Como se puede ver en la Figura 1, el Deep Learning es un subtipo de Machine Learning, y este, a su vez, está englobado en la Inteligencia Artificial.

La **Inteligencia Artificial** consiste en la necesidad de imitar el comportamiento del ser humano, pero en un ámbito muy general, ya que engloba muchos más campos. El **Machine Learning** o Aprendizaje Automático, tiene un comportamiento similar al de las personas, ya que aprende a partir de los datos y nos da un resultado. Un ejemplo representativo es el algoritmo de recomendación que usa Netflix para proponer a los usuarios contenido.[1]

Dentro el Machine Learning hay problemáticas que requieren modelos más complejos que intenten comprender los conceptos con mayor precisión, analizando los datos a un alto nivel de abstracción, y aquí es donde aparece el concepto de Deep Learning o Aprendizaje Profundo.

El **Deep Learning** se asemeja al cerebro humano, de ahí el nombre del método que usa, redes neuronales. Las redes neuronales están formadas por nodos o neuronas que forman distintas capas interconectadas. Un ejemplo de uso del Deep Learning es el reconocimiento de rostros humanos.

2.1.1. Redes Neuronales Artificiales

Una red neuronal es un método de inteligencia artificial que enseña a las computadoras a procesar datos. El cerebro humano es la base de las redes neuronales, es la inspiración para realizar su arquitectura. Las células que encontramos en el cerebro se llaman neuronas y en su conjunto forman una red muy compleja con un alto nivel de interconexión. Estas, envían señales eléctricas entre sí, para procesar la información. Una red neuronal artificial (RNA) funciona de manera similar. Las redes neuronales están formadas por neuronas artificiales, que trabajan juntas para resolver un problema. Las neuronas artificiales son módulos software, llamados nodos, y las redes neuronales artificiales son programas de software o algoritmos que, en esencia, utilizan sistemas informáticos para resolver problemas matemáticos.

2.1.1.1. Arquitectura

Una red neuronal simple tiene neuronas artificiales interconectadas en tres capas como podemos ver en la Figura 2. Son:

- **Capa de entrada:** todos los datos que llegan a la red neuronal desde fuera tienen que pasar por esta capa. Los nodos de entrada procesan los datos y los analizan antes de pasar a la siguiente capa.
- **Capa oculta:** las capas ocultas reciben los datos de entrada de la capa anterior, procesa aún más los datos y los pasa a la siguiente capa. Las redes neuronales pueden llegar a estar formadas por una gran cantidad de capas ocultas.
- **Capa de salida:** la capa de salida nos da un resultado de todo el procesamiento de datos que se ha ido realizando a lo largo de la red neuronal artificial. Puede tener uno o varios nodos. Algunos problemas de regresión, con un nodo que nos dé un valor real. Pero si tenemos un problema de clasificación multiclase, la capa de salida estará formada por varios nodos de salida.

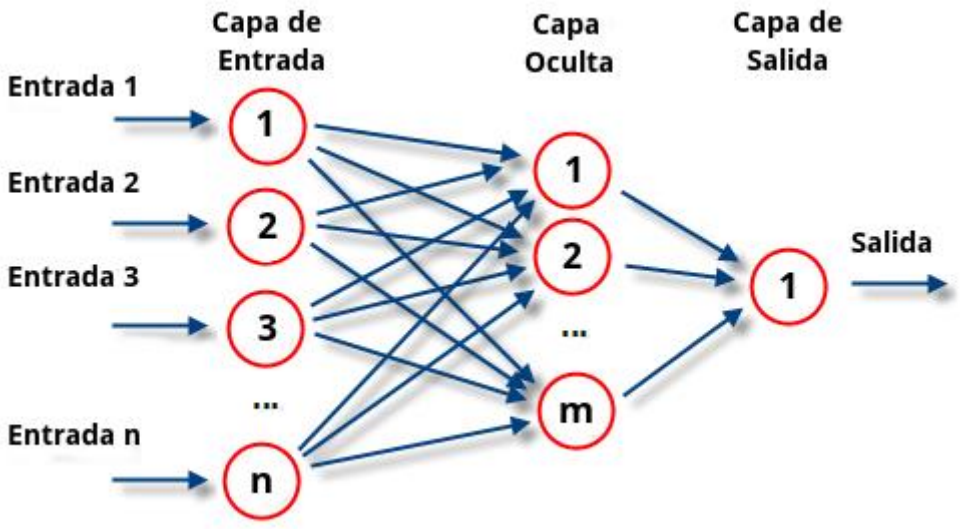


Ilustración 2: Capas de red neuronal sencilla. **Fuente**.

Sin embargo, una red neuronal con 3 capas es muy básica, en la realidad las redes neuronales tienen mucha más profundidad, es decir, tienen varias capas ocultas con millones de neuronas conectadas entre sí. Estas redes se definen como redes neuronales profundas.

Para entenderlo mejor veamos un ejemplo. Imaginemos un modelo de Aprendizaje Profundo que sea capaz de detectar rostros humanos. La primera capa concibe que hay pixeles, las siguientes se dan cuenta que varios pixeles forman un borde, y así sucesivamente, hasta que adquieren la noción de “cara”.

Hasta llegar a la capa final, los datos tienen que pasar por varios nodos conectados. Esta conexión entre nodos es representada por un número que se denomina peso. El peso es un número positivo si un nodo estimula a otro, o negativo si un nodo suprime a otro. Los nodos con valores de peso más altos tienen mayor influencia en los demás nodos.

Las redes neuronales profundas (véase imagen 3) pueden parecer perfectas, pero tienen un pequeño defecto, ya que para entrenarse necesitan millones de ejemplos. Así que, dependiendo del problema, puede ser mejor usar una red neuronal más simple que únicamente necesite cientos de datos o miles, para dar buenos resultados.

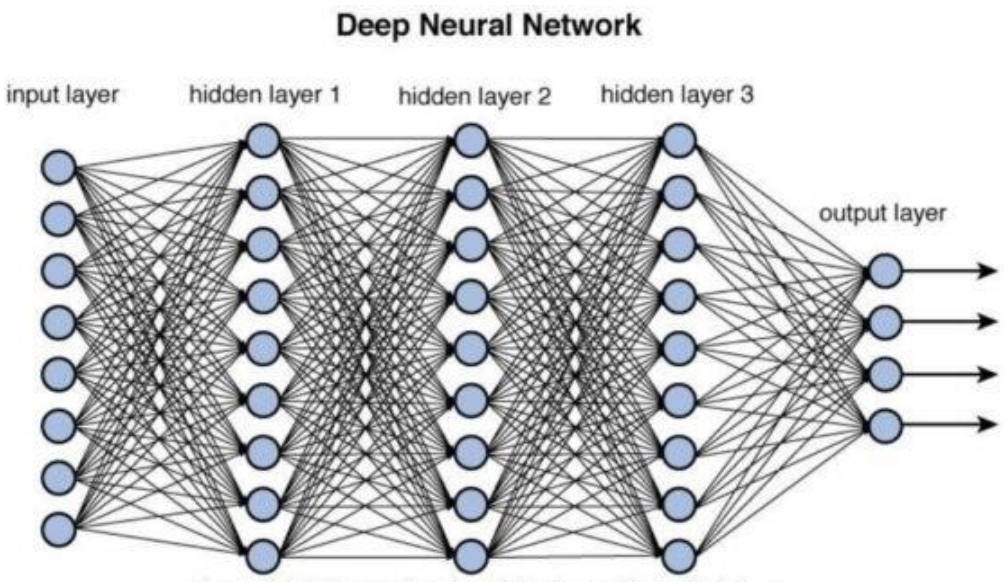


Ilustración 3: Capas de red neuronal compleja. **Fuente**.

2.1.1.2. Tipos de redes neuronales

A continuación, se van a mencionar algunos tipos de redes neuronales, algunas de las cualidades que tienen y la utilidad de las mismas. Hay muchos tipos, pero vamos a ver las más importantes[2]:

- **Convolutional Neural Networks (CNN, Redes Neuronales Convolucionales):** estas redes neuronales artificiales están programadas para procesar matrices estructuradas, como las imágenes. Su misión es clasificar imágenes buscando una serie de objetos y patrones dentro de dichas imágenes, desde lo más simple (líneas o círculos) hasta algo más complejo (ojos o caras).

  Las CNN sobre todo se usan en *computer vision* o visión artificial, porque son capaces de trabajar con imágenes brutas o *raw images* y no es necesario preprocesarlas. Esta habilidad hace que sean muy prácticas en aplicaciones visuales como la de clasificación de imágenes, pero también se usan para la clasificación de textos o **PLN** (Procesamiento del Lenguaje Natural).

- **Recurrent Neural Networks (RNN, redes neuronales recurrentes):** este tipo de redes neuronales son usadas para tratar problemas ordinales o temporales, y un claro ejemplo de esto lo encontramos en la traducción a otro idioma o el reconocimiento de voz (como Siri y Google Translate). Además, este tipo de redes son capaces de diferenciar el sexo de la persona que emite la voz, la edad de la misma e incluso el acento.

- **Generative Adversarial Networks (GAN, redes generativas antagónicas):** Están compuestas por dos redes neuronales artificiales para posteriormente enfrentarlas entre ellas para conseguir nuevos datos más realistas. Una de las redes se encarga de "crear" y la otra de "discriminar". Esta última recibe el nombre de red antagónica y se encarga de filtrar aquel contenido que considera real, enviado por la primera red, entrando en un bucle en el que la segunda va mejorando con el tiempo el contenido de la primera. Por esta razón, este tipo de redes son utilizadas para desarrollar nuevas imágenes, vídeos y voces.

- **Redes Transformer:** Esta arquitectura aparece como solución a los problemas de aprendizaje supervisado en Procesamiento del Lenguaje Natural, dando lugar a mejores resultados que los modelos utilizados hasta ahora. El transformer es capaz de realizar la traducción de un idioma a otro mientras el modelo va entrenándose al mismo tiempo[3]. Aunque actualmente se están utilizando en otras muchas áreas.

2.1.2. Capacidades de las redes neuronales

Las redes neuronales se caracterizan por ser capaces de realizar las siguientes tareas[4]:

- **Hacer generalizaciones y sacar conclusiones:** las redes neuronales son capaces de entender datos no estructurados y hacer observaciones generales. Por ejemplo, son capaces de darse cuenta que dos datos de entrada son semejantes, como estas dos oraciones: "¿Puede explicarme

cómo hacerte una transacción?" y "¿Cómo puedo transferirte el dinero?" ambas tienen el mismo significado.

- **Revelar relaciones y patrones ocultos:** las redes neuronales son capaces de analizar los datos con una mayor profundidad y descubrir conocimientos nuevos para los que no han sido entrenadas. Por ejemplo, una red neuronal capaz de detectar patrones en las compras de los usuarios, podría sugerir artículos que le podrían interesar basándose en estos patrones.

- **Crear sistemas de autoaprendizaje autónomos:** las redes neuronales son capaces de aprender con facilidad y mejorar con el paso del tiempo, en función del comportamiento del usuario. Pongamos el caso de una red neuronal que corrija o sugiera palabras automáticamente, basándose en el análisis de cómo escribe un usuario. Supongamos que el modelo fue entrenado en inglés y corrige la ortografía de las palabras en inglés. Sin embargo, si el usuario escribe a lo largo del texto una palabra que no está en inglés, la red neuronal aprende y puede corregir estas palabras de forma automática. Es decir, si en el texto aparece frecuentemente la palabra "gazpacho", al cometer una errata y escribir "lazpacho" es capaz de enmendarla, aunque esté entrenado para corregir en inglés.

- **Aprender a modelar datos muy volátiles:** algunos de los datos, como los importes de los reembolsos de los préstamos en un banco, pueden llegar a ser muy distintos. Las redes neuronales también pueden modelar estos datos, de modo que, siguiendo con el caso anterior, estas son capaces de analizar las transacciones y detectar si algunas de ellas son fraudulentas.

2.1.3. ¿Dónde se utilizan las redes neuronales?

Como ya hemos visto estas tienen muchas utilidades por lo que las podemos encontrar formando parte de diferentes sectores:

- Diagnosticar a pacientes con el uso de clasificación de imágenes médicas.
- En marketing las podemos encontrar orientadas al análisis de las **RRSS** y el comportamiento de los usuarios.
- En el sector financiero se usan para procesar datos y obtener información de ellos.
- Estimación de la demanda de energía y carga eléctrica.
- Establecimiento de compuestos químicos.

2.2. TensorFlow Hub

Una vez vista la diferencia entre Inteligencia Artificial, Machine Learning y Deep Learning hemos conocido un poco más sobre ellas. A continuación, veremos algunos conceptos sobre la página base de este TFG, TensorFlow Hub.

2.2.1. Introducción TensorFlow Hub

En la actualidad el software ha adquirido un papel muy importante, pero en ocasiones no se tiene demasiado en cuenta la idea de “código compartido” que podría hacer prosperar el software de los desarrolladores. La existencia de repositorios con código libre ya implementado, o incluso bibliotecas de cualquier lenguaje, hace que una tarea compleja se convierta en una mucho más sencilla. Estas bibliotecas hacen cambiar el estilo de programación y la manera de pensar de los desarrolladores ya que estructuran su código en base a estas.

¿Pero, si en lugar de compartir sólo código, se compartieran también redes neuronales? Esta pregunta nos lleva a pensar en, ¿qué pasaría si se entrenan redes neuronales y posteriormente se distribuyen? La realidad es que esto ya existe, y es que, además de compartir código, se pueden compartir redes entrenadas con anterioridad.

Una de las muchas ventajas que tiene, es que cualquier desarrollador puede personalizar dicha red para su dominio, a pesar de no tener los recursos computacionales necesarios o incluso conseguir los datos con suficiente calidad y cantidad para que estas redes sean entrenadas.

Ilustración 4: Logo TensorFlow Hub (1). Fuente.

TensorFlow Hub (ilustración 4) es una plataforma donde cualquiera puede subir, utilizar o rehusar cualquiera de las redes neuronales. Estas redes tienen una amplia funcionalidad, como añadir más capas a la red y entrenarlas con una cantidad de datos menor porque ya han sido pre-entrenadas. En el desarrollo de esta aplicación web nos vamos a limitar solo a usarlas, no vamos a entrenarlas ni tampoco aprender en base a los resultados de sus predicciones (como hacen algunas redes). Estas modificaciones no son competencia de este proyecto por lo que se podría considerar realizarlas en un futuro si se quisiera mejorar.

2.2.2. TensorFlow Hub

Una vez entendida la funcionalidad de la plataforma y lo que podemos encontrar en ella, se va a profundizar en el uso de la misma. Para acceder basta con escribir en Google “TensorFlow Hub” y aparecerá en el primer resultado, aunque también se puede acceder a través de la url https://www.tensorflow.org/hub.

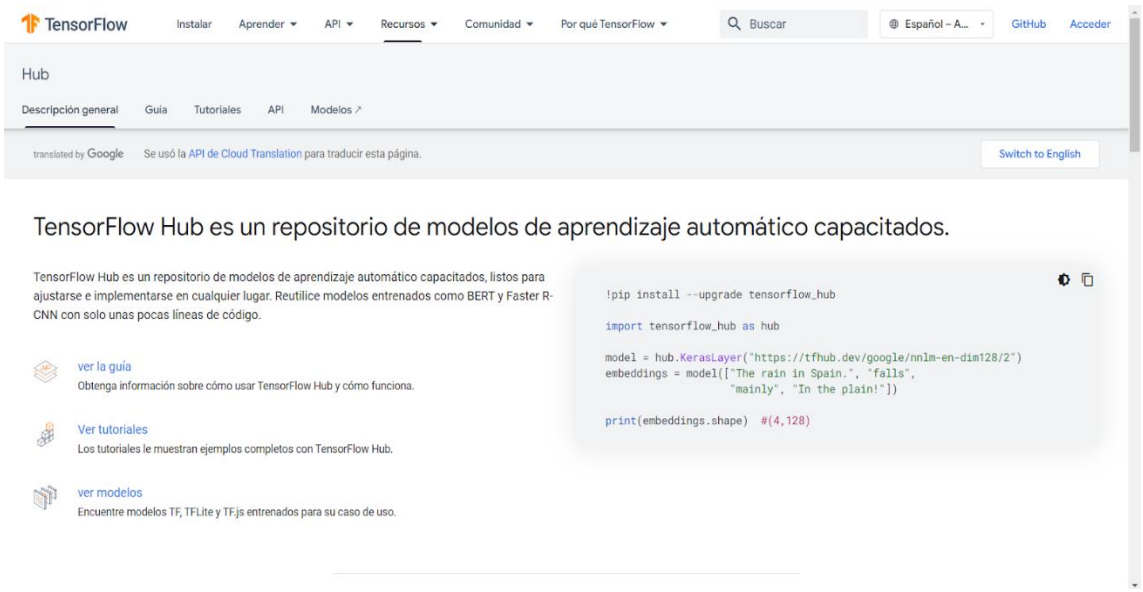


Ilustración 5: Página web de inicio de TensorFlow Hub (1)

En la imagen anterior (ilustración 5), podemos apreciar que, a simple vista, no hay mucha diferencia con otras páginas webs. En la página principal podemos encontrar una pequeña introducción en la que se explica en qué consiste la web y se explica el funcionamiento de uno de sus modelos a modo de ejemplo.

En la parte superior se observan varias pestañas en las que, al clicar, se dará información sobre la web, quiénes son y cómo instalar la biblioteca TensorFlow, entre otros.

Esta plataforma destaca porque tiene unos guiones en los que se explica cómo se usan sus modelos y los distintos dispositivos o lenguajes en los que se pueden usar. Además, incluye una **API** que permite probar modelos y mostrar su funcionamiento.

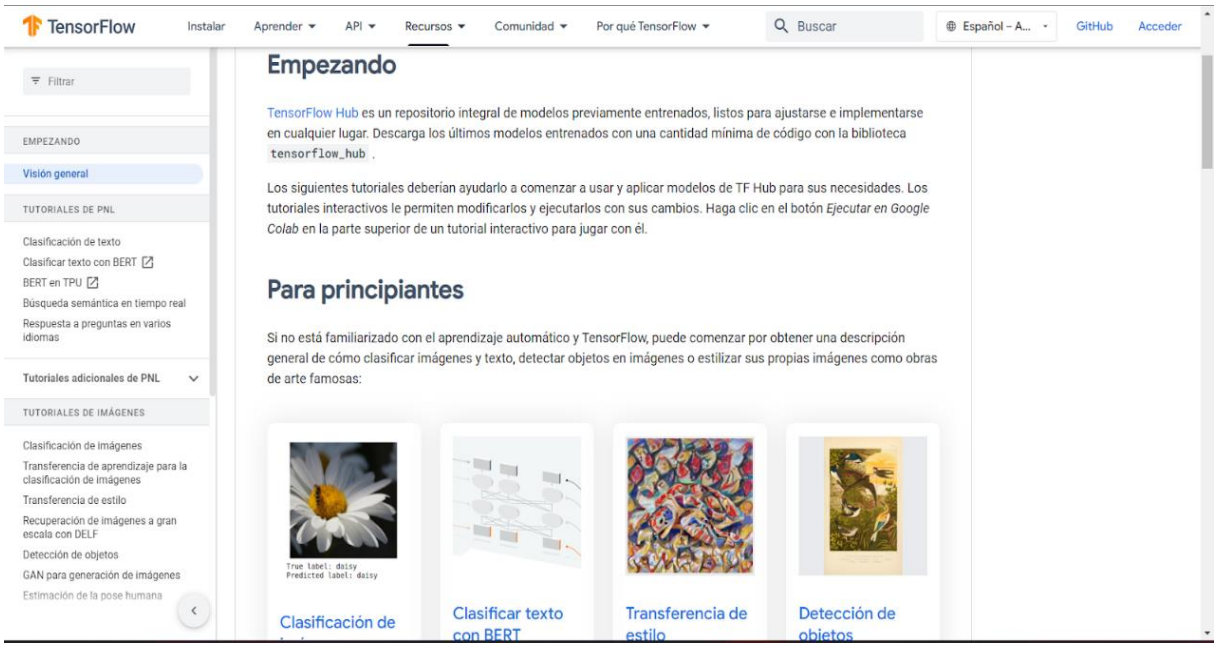


Ilustración 6: Página web de inicio de TensorFlow Hub (2)

2.2.3. Modelos de TensorFlow Hub

Esta página web contiene una gran cantidad de modelos en función del dominio del problema, de la arquitectura…, aunque no se va a profundizar en ellos.

En primer lugar, lo que hay que hacer es acceder a la interfaz de los modelos a través de la página principal (ilustración 6) o con el siguiente enlace https://tfhub.dev/?utm_source=www.tensorflow.org&utm_medium=referral.

En la página principal, al clicar en la zona superior izquierda, se encuentran distintos filtros para poder buscar modelos. El filtro de mayor utilidad de la plataforma es “problem domains” y en su interior podemos encontrar cuatro grandes grupos diferenciados (Figura 7) en base al tipo de dato de entrada con el que se vaya a trabajar. Estos grupos, a su vez, engloban distintos modelos.

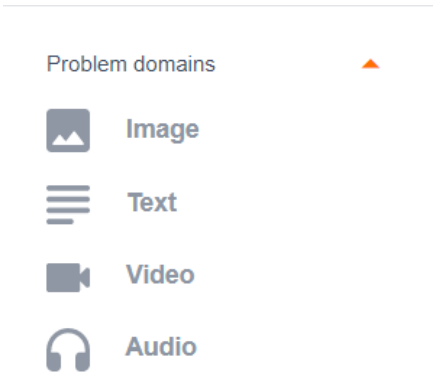


Ilustración 7: Captura de pantalla de los problemas de dominio de la página TensorFlow Hub

En el dominio de texto y audio destacan los modelos de “Embedding”. El dominio de texto alberga 351 modelos, y el de voz 65, de los cuales 208 de texto y 15 de audio son de “Embedding”. Por otro lado, tanto para las imágenes como para vídeo destacan los modelos de “Classification”.

2.2.4. Dominios que utiliza DL4Vision

DL4Vision, como su propio nombre indica, es una aplicación que trabaja en el dominio de imágenes, es decir, las utiliza para predecir, dependiendo del tipo de modelo que se use, nos dará un resultado u otro.

Cabe destacar que, aunque haya modelos de video que se podrían utilizar para predecir en tiempo real, no se han usado para la opción de *stream* implementada en la app. En su lugar, se ha optado por emplear modelos de imagen para predecir todos los *frames* que capte la cámara, es decir, si la cámara capta 32 **FPS**, el modelo consigue predecir 32 imágenes por segundo.

Posteriormente, en el apartado 4.3.2.1., se definirán con más detalle las cuatro categorías de DL4Vision:

- Clasificación de imágenes.
- Detección de Objetos

- Segmentación
- Detección de Pose

2.3. Herramientas

En este apartado se van a comentar todas las herramientas y software que se han utilizado. Algunas de estas comparándolas con otras tecnologías que realizan las misma función y la razón de porque se ha escogido.

2.3.1. Framework y lenguaje de programación

La elección del framework ha sido compleja ya que existen muchos frameworks disponibles para los usuarios que facilitan el trabajo a la hora de desarrollar una aplicación web como esta. En primera instancia se eligió Spring Boot (ilustración 8) para la parte de backend y para la parte de frontend se usó Angular.

Ilustración 8: Logo del Framework Spring Boot y Angular. Fuente 1 y Fuente 2.

Spring Boot es un framework de código abierto que utiliza los lenguajes de programación como Java, Kotlin o Groovy[5]. Por otro lado, encontramos Angular, un framework de código abierto creado por Google, que utiliza lenguajes de programación como **HTML**, **CSS** o JavaScript (ilustración 9)[6].

Estos framework hubieran sido una opción acertada porque son cómodos y sencillos a la hora de trabajar con ellos, además de que habían sido usados con anterioridad para otros proyectos. También cabe destacar que durante la formación de la carrera de ingeniería informática se ha trabajado con Java por lo que Sprint Boot hubiera sido una buena elección.

Ilustración 9: Logo de Java. Fuente

Pero durante el proceso de desarrollo del backend de la aplicación, se produjo un problema que haría de este proyecto algo mucho más complicado, ya que esta aplicación está basada en los modelos de TensorFlow Hub, los cuales han sido implementados en lenguaje Python (ilustración 10). Spring Boot al no trabajar con este lenguaje y trabajar con Java ha hecho que se descarte este framework y se haya escogido otro distinto.

Ilustración 10: Logo de Python. Fuente.

No se sabía la dificultad que iba a suponer gestionar la entrada y salida de estos modelos con un framework en un lenguaje distinto, por lo que se optó por usar un framework cuyo lenguaje principal fuese Python.

Tras investigar se encontró que los frameworks, más utilizados y cómodos para usar con lenguaje Python fueron FastAPI, Flask y Django.

Ilustración 11: Logo de FastAPI. Fuente.

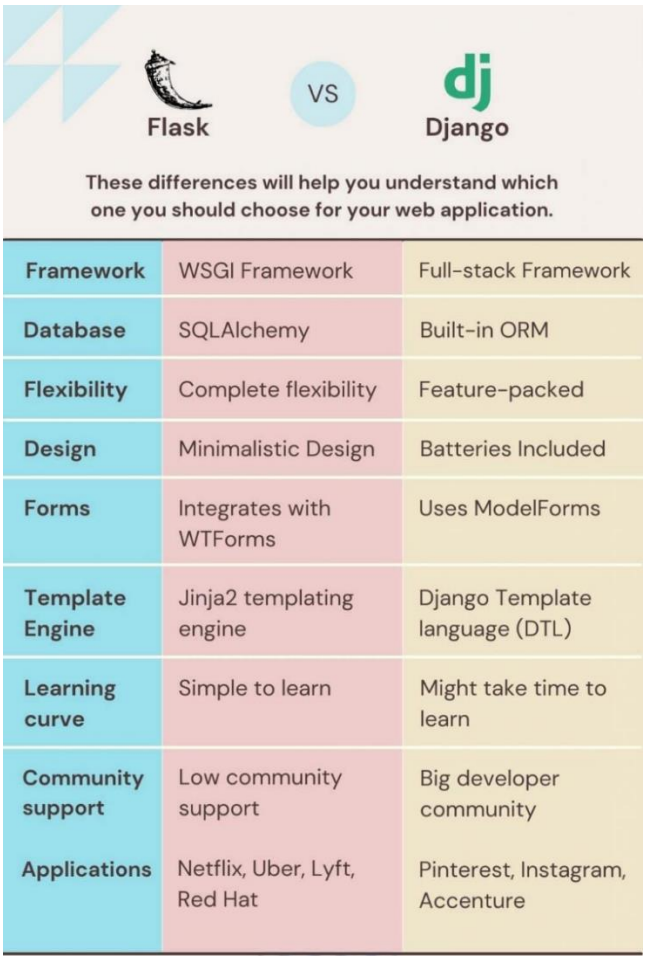
FastAPI (imagen 11) es el menos usado de los tres mencionados ya que es el más nuevo, pero se caracteriza por ser el menos complejo y el más veloz, porque con menos líneas de código se consigue mayor funcionalidad. Pero para el desarrollo de esta aplicación se buscaba un framework más consolidado y potente, a pesar de que este pudiera presentar complicaciones. Por eso, se optó por elegir un framework distinto[7].

Ilustración 12: Logo de Flask. Fuente.

Flask (ilustración 12) es un framework que permite desarrollar aplicaciones web de forma sencilla. Una de las propiedades más interesantes es la capacidad para instalar complementos o extensiones en función del tipo de proyecto que se vaya a elaborar. Se instalarán únicamente las funcionalidades que vayan a ser útiles para el desarrollo de la aplicación.

Ilustración 13: Logo de Django. Fuente.

Django (imagen 13) está diseñado para trabajar con una arquitectura **MVC** (modelo, vista, controlador), aunque también se puede definir como **MVT** (modelo, vista, plantilla). Se profundizará algo más en el apartado 4.4. dónde se explicará esta arquitectura.



	Flask	Django
Framework	WSGI Framework	Full-stack Framework
Database	SQLAlchemy	Built-in ORM
Flexibility	Complete flexibility	Feature-packed
Design	Minimalistic Design	Batteries Included
Forms	Integrates with WTForms	Uses ModelForms
Template Engine	Jinja2 templating engine	Django Template language (DTL)
Learning curve	Simple to learn	Might take time to learn
Community support	Low community support	Big developer community
Applications	Netflix, Uber, Lyft, Red Hat	Pinterest, Instagram, Accenture

Ilustración 14: Comparación entre Flask y Django. Fuente.

Se puede ver una comparación más detallada en la Ilustración 14 que compara los framework Flask y Django. Tras conocer un poco sobre estos tres framework comentados a lo largo de este apartado llega el momento de decidir cuál es mejor para el desarrollo de esta aplicación. Es cierto que esto depende del proyecto y del desarrollador, por lo que en esta ocasión se ha optado por usar Django gracias a la posibilidad que ofrece de poder utilizar páginas HTML dinámicas. Al carecer de experiencia trabajando con frameworks destinados al front esta característica permite desarrollar un front sencillo con solo saber HTML y CSS.

2.3.2. IDL de desarrollo

Durante el desarrollo de la aplicación, al principio, se usó el **IDL** Visual Studio Code, pero posteriormente se cambió a Pycharm.

Ilustración 15: Logo Visual Studio Code. Fuente.

Visual Studio Code (**VS** Code) (ilustración 15) es un framework desarrollado por Windows, este es multiplataforma y contiene software libre. Podemos encontrarlo para los principales sistemas operativos, tanto para Windows como para **GNU**/Linux o macOS[8].

En primer lugar, se optó por usar este IDL porque es capaz de soportar un gran número de lenguajes y permite programar con HTML, CSS o Python con solo instalar los plugins necesarios.

Ilustración 16: Logo PyCharm. Fuente.

Durante el desarrollo, para que la aplicación se pudiera ejecutar en cualquier **PC** se decidió usar una imagen de Docker (véase punto 2.3.5). Pycharm (imagen 16), en su versión de pago, nos permite ejecutar la aplicación dentro del Docker sin necesidad de insertar comandos. Esta fue la razón principal por la que se eligió este programa, además de la variedad de utilidades que incorpora. Por ejemplo, el propio IDL crea un entorno virtual en el que podemos encontrar todas las librerías que usa la aplicación.

2.3.3. Base de datos

En el desarrollo de la aplicación se han usado dos bases de datos distintas, SQLite (ilustración 17) y PostgreSQL.

Ilustración 17: Logo SQLite. Fuente.

La base de datos diseñada para esta aplicación no es muy compleja, por lo que en un principio se eligió el sistema de gestión de bases de datos SQLite que es mucho más sencillo. Cuando surge la necesidad de usar la tecnología de Docker, se prefirió emplear PostgrestSQL (imagen 18) porque al tener que generar un Docker-Compose para la aplicación, era conveniente añadir una base de datos de alto nivel. Además, proporciona una interfaz para interactuar con todos los elementos que hay en la base de datos.

Ilustración 18: Logo PostgreSQL. Fuente.

Durante el desarrollo surgió un problema porque al crear un contenedor de Docker y añadirle una imagen PostgreSQL, los elementos que contenía la base de datos no se subían a esta imagen y había que añadirlos manualmente cada vez que se ejecutaba. Por tanto, se volvió a utilizar SQLite, que sube la copia en local de la base de datos con el Dockerfile, y si es necesario gestionar algún elemento de la base de datos, se usa el modo de administrador.

2.3.4. Bibliotecas Python

Una vez comentado el framework y el lenguaje, se van a mencionar algunas librerías más importantes de Python que se han usado a lo largo del desarrollo de la aplicación.

- Pillow: también conocido como **PIL** se utiliza para manipular imágenes. Por ejemplo, se usa esta biblioteca para leer una imagen en bytes.
- Django: esta librería es imprescindible para que el framework funcione correctamente, de ahí que tenga el mismo nombre.
- TensorFlow: es muy conocida puesto que se usa para crear modelos de aprendizaje automático. En la aplicación desarrollada, algunos modelos usan esta librería tanto para cargar modelos como para predecir un resultado.
- Numpy: esta biblioteca es eficaz en cuanto se refiere a la gestión de vectores y matrices. En la aplicación, muchos de los modelos necesitan

que los valores de entrada de la imagen sean en un vector de numpy o como los llama la propia biblioteca, ndarray.

- Keras: es muy similar a la librería de TensorFlow, ya que lo único en lo que se diferencian es en el nombre de las funciones y las estructuras que usan. Algunos modelos están implementados con esta tecnología para crear redes neuronales.

- Matplotlib: es una de las librerías más importantes de la aplicación, ya que todas las imágenes que se envíen van a pasar por aquí. Además, tiene la función de editar imágenes, como por ejemplo añadir un título o pintar algún elemento.

- OpenCV: esta biblioteca es la que ha sido usada principalmente para las opciones de entrada de imagen "tomar imagen" o "tiempo real", ya que conecta la cámara conectada al equipo con la aplicación.

Ilustración 19: Logos de algunas bibliotecas de Python usadas en DL4Vision

Estas librerías (ilustración 19), junto con el resto, se encuentran preinstaladas en la imagen de Docker.

2.3.5. Docker y Docker Hub

Como se ha comentado en el apartado anterior, la intención de esta aplicación es que fuera escalable y que se pudiera usar en cualquier PC. La idea inicial era realizar un ejecutable de la aplicación que incluyera en su interior todo el software

necesario, pero llevarlo a cabo era demasiado complejo, por lo que finalmente se ha decidido usar la tecnología de Docker. Además, para mejorar la portabilidad de Docker, se ha usado la herramienta Docker Hub.

Docker es una plataforma de software que permite desarrollar, experimentar y crear aplicaciones de forma veloz. Empaqueta software en unidades estandarizadas, las cuales reciben el nombre de contenedores y en su interior albergan todo lo necesario para que se ejecute la aplicación. En ellos podemos encontrar bibliotecas, código y herramientas de sistema. Docker (ilustración 20) permite implementar y ajustar la escala de aplicaciones rápidamente en cualquier entorno, con la certeza de saber que su código se ejecutará[9].

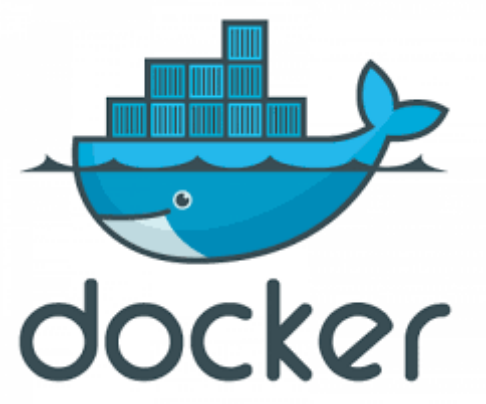


Ilustración 20: Logo de Docker. Fuente.

Para el desarrollo de la aplicación, se implementó una imagen en Docker utilizando el sistema operativo de Ubuntu 20.04 (apartado 2.3.6), incluyendo todas las bibliotecas (apartado 2.3.4) y los drivers (apartado 2.3.7). Además, se ha usado Docker-Compose que permite, de manera más intuitiva, ejecutar los comandos necesarios dentro de la imagen, para que la aplicación se inicie. Del mismo modo permite que Linux reconozca tanto los drivers de nvidia como la cámara conectada al equipo. Además, esta herramienta sirve para probar varias imágenes o ver cómo afecta la instalación de un nuevo software a la imagen.

Esta imagen de Docker es muy útil y permite que sea mucho más portable, pero la imagen hay que trasladarla al PC donde quiera ser ejecutada, y es entonces cuando se va a empezar a usar esta herramienta, Docker Hub.

Docker Hub (imagen 21) es un repositorio donde los usuarios pueden subir imágenes de Docker y pueden ser descargadas desde cualquier dispositivo, solamente basta con saber el nombre y ejecutar un comando.

Ilustración 21: Logo de Docker Hub. Fuente.

2.3.6. Sistemas Operativos

En este apartado se verán los diferentes sistemas operativos y la medida en la que estos aparecen. Los usados en esta aplicación han sido Windows y Linux (ilustración 22), de los cuales se irá explicando cómo han interactuado durante el desarrollo de la aplicación, desde que empezó a desarrollarse hasta una vez terminada.

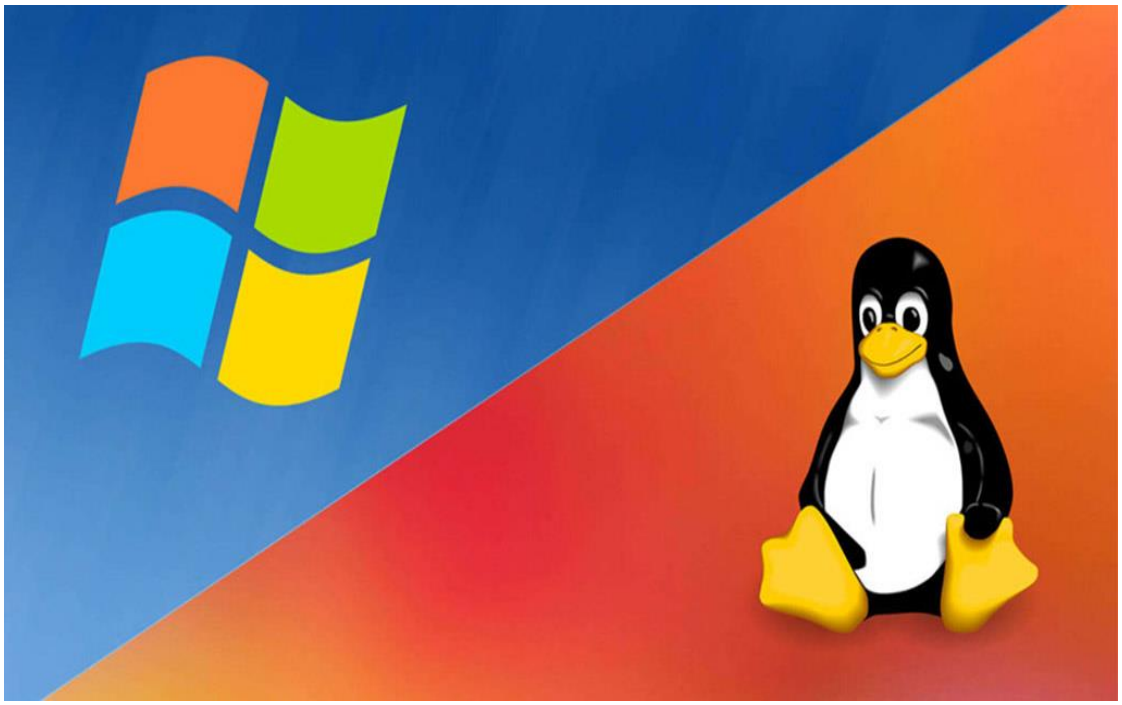


Ilustración 22: Logos de Windows y Linux. Fuente.

La aplicación empezó a desarrollarse en Windows porque es el sistema operativo que traía el portátil en el que se comenzó el proyecto. La idea inicial de la aplicación era desarrollarla para que funcionase en cualquier sistema operativo, pero al introducir la tecnología de Docker surgió un problema, ya que este no era capaz de coger la cámara del equipo. Este fue bastante tedioso de resolver, ya que en la aplicación algunas de las opciones de entrada de imagen para los modelos, es a través de la cámara, de forma que era importante dar solución a esto. Tanto el problema (apartado 6.2.) como las distintas opciones para insertar imágenes (apartado 11.2.5.) las veremos posteriormente.

Ante esta problemática, para que la aplicación funcione se tiene que ejecutar en Linux. Existe la posibilidad de ejecutarla en Windows, pero la funcionalidad de la cámara no estará, aunque el resto de elementos funcionarán perfectamente.

2.3.7. Drivers de Nvidia

En esta aplicación se han usado modelos que pueden llegar a ser bastante complejos, como por ejemplo, pueden estar entrenados para procesar imágenes de gran tamaño para aumentar la probabilidad de acierto en la predicción, pero esto tiene un coste computacionalmente hablando. Este coste se puede apreciar en el tiempo que tarda la aplicación en predecir dicha imagen, por eso se ha intentado sacar el máximo rendimiento de un ordenador, o mejor dicho, de su tarjeta gráfica.

Las tarjetas gráficas de Nvidia junto con las de **AMD** son las que dominan el mercado actual, pero la diferencia entre ellas es que Nvidia te permite trabajar con una tecnología llamada **CUDA**.

CUDA es un acrónimo que significa Compute Unified Device Architecture y es una plataforma de computación paralela, pero además, es una interfaz de programación de aplicaciones (API). Esta tecnología fue diseñada específicamente por Nvidia, para permitir que los desarrolladores de software tengan un mejor control sobre los recursos físicos que se encuentran a su disposición. Dentro de esta tecnología, podemos encontrar una librería llamada NVIDIA **cuDNN** (ilustración 23).

La biblioteca NVIDIA CUDA Deep Neural Network (cuDNN) es una biblioteca que aprovecha al máximo el rendimiento de la **GPU** para las redes neuronales profundas. Por eso, a nivel mundial, los investigadores de Deep Learning confían en cuDNN para la implementación de sus modelos. Gracias al trabajo que hace esta librería, permite a los desarrolladores enfocarse en entrenar redes neuronales e implementar aplicaciones de software en lugar de desaprovechar el tiempo en pulir los tiempos de predicción o el rendimiento de la GPU[10].

Ilustración 23: Logo de Nvidia Cuda y drivers cuDNN. **Fuente 1 y Fuente 2.**

2.3.8. Google Collab

Colaboratory o "Colab" es un producto desarrollado por Google Research (ilustración 24). Da la posibilidad a cualquier usuario para escribir y programar desde cualquier navegador. Es muy conveniente para realizar tareas de aprendizaje automático y análisis de datos para usuarios que desean aprender a programar. Si se observa desde una perspectiva un poco más técnica, se puede ver que Colab es un servicio de cuaderno alojado de Jupyter que no requiere configuración y que ofrece acceso sin coste adicional a recursos informáticos, como GPUs.

Esta herramienta ha sido usada con el objetivo de realizar pruebas para el desarrollo de los modelos, ya que desarrollarlos dentro de la aplicación lleva mucho tiempo. Además, se pueden probar distintas librerías sin necesidad de instalar nada.

Ilustración 24: Logo de Docker Hub. **Fuente**.

2.3.9. Gestión de código

Para finalizar este apartado, se va a explicar el método utilizado para gestionar el código. Se han escogido las herramientas de Git ya que son muy prácticas, sobre todo, para añadir elementos sin perder el progreso anterior. Aprovechando esta tecnología y el repositorio de Github (imagen 25), se ha conseguido que, una vez acabada la aplicación, las modificaciones realizadas para mejorar la aplicación queden bien diferenciadas de la aplicación original y que ambas puedan ser usadas simplemente cambiando de rama.

Ilustración 25: Logo de Git y Github. Fuente 1 y Fuente 2

2.4. Metodologías de desarrollo software

Las metodologías de desarrollo de software son muy utilizadas en el ámbito de la programación, entre otros, con la misión de conseguir trabajar conjuntamente de la manera más eficiente posible. Las metodologías con el transcurso de los años han evolucionado, antes eran un mero trámite de una empresa, algo que no tenía mucha importancia, a ahora en la actualidad a ser la base de cualquier proyecto de desarrollo software, para que cualquier proyecto se realice de manera eficaz y productiva.

En la actualidad se dividen todas las metodologías en dos grupos, las ágiles y las tradicionales. Veremos la diferencia y algunas metodologías representativas de cada grupo.

2.4.1. Metodologías de desarrollo de software tradicionales

Las metodologías de desarrollo de software tradicionales tienen como características principales definir total y rígidamente los requisitos al comenzar un proyecto. Los ciclos no permiten realizar cambios durante el transcurso del proyecto y son poco flexibles[11].

Además, este tipo de metodologías se caracteriza por organizarse de manera lineal, es decir, las etapas ocurren una tras otra y hasta que no acabe una etapa no se puede comenzar con la siguiente. Tampoco se puede ir en sentido contrario, ir de una etapa hacia una etapa anterior. Algunas de las principales metodologías tradicionales o clásicas son:

- **Waterfall (cascada)**: Esta metodología está organizada en un orden inalterable de arriba abajo, de ahí su nombre. Tiene las etapas bien diferenciadas unas entre otras y se revisa el producto rigurosamente antes de pasar a la siguiente etapa. Esto provoca que para ver resultados el proyecto debe estar en una etapa avanzada.
- **Prototipado**: Esta metodología consiste en realizar un prototipo muy simple con la suficiente funcionalidad para que lo puedan utilizar unos usuarios y así recibir un feedback de ellos.
- **Espiral**: en esta metodología se combinan las dos anteriores y además introduce la idea de análisis de riesgo. Esta se divide en 4 etapas: planificación, análisis de riesgo, desarrollo de prototipo y evaluación del cliente. Las etapas se van procesando las etapas en forma de espiral. Mientras más se acerque al centro, más avanzado está el proyecto.
- **Incremental:** en esta última metodología tradicional el software se construye de forma progresiva. En cada etapa se va añadiendo una nueva funcionalidad. Esta metodología tiene muchas ventajas como que el software se puede usar incluso antes de que el producto este acabado y, en general, es más flexible que el resto de metodologías.

2.4.2. Metodologías de desarrollo de software ágiles

Las metodologías ágiles de desarrollo de software son las que más se usan debido a su alta flexibilidad y agilidad. Los equipos de trabajo se organizan más eficientemente ya que cada persona del equipo sabe qué hacer en cada momento.

Las metodologías ágiles nacen a partir de la idea de mejorar metodología incremental. La diferencia es que en estas metodologías los ciclos son mucho más cortos y rápidos, la funcionalidades se granulan más para ir añadiendo pequeños cambios.

Las principales metodologías ágiles son:

- **Kanban**: esta metodología fue creada por la empresa de automóviles Toyota. Consiste en granular las tareas lo más pequeñas posibles y organizarlas en un tablero donde están reflejadas todas las tareas pendientes, en curso o finalizadas. De esta manera, las tareas se ven de una manera más visual.
- **Scrum**: es también una metodología incremental que divide los requisitos y tareas de forma similar a Kanban. Se utilizan bloques de tiempos cortos y fijos para conseguir un resultado completo en cada iteración. Las etapas son: planificación de la iteración (planning sprint), ejecución (sprint), reunión diaria (daily meeting) y demostración de resultados (sprint review). Cada iteración por estas etapas se denomina también sprint.

2.4.3. Elección de metodología del software

La metodología elegida para este TFG es la metodología incremental, hemos visto que las mejores son las ágiles, pero al ser un proyecto donde solo hay un desarrollador es innecesaria tanta organización, más adecuada para un proyecto donde se tengan que organizar a más personas.

Dentro de las tradicionales la que se ha elegido es la incremental porque es la mejor, es la más flexible a cambios ya que los profesores pueden querer introducir

cambios en cualquier momento y se ha desarrollado en pequeñas etapas en la que cada etapa se ha desarrollado una funcionalidad de la aplicación.

2.4.4. Desarrollo incremental

Una vez elegida la metodología software vamos a entrar más en profundidad, en conocerla mejor y como aplicarla al proyecto. El modelo incremental consiste en un desarrollo inicial de la arquitectura completa del sistema, seguido de sucesivos incrementos funcionales. Cada incremento tiene su propio ciclo de vida y se basa en el anterior, sin cambiar su funcionalidad ni sus interfaces.

Una vez entregado un incremento, no se realizan cambios sobre el mismo, sino únicamente corrección de errores. Dado que la arquitectura completa se desarrolla en la etapa inicial, es necesario conocer los requerimientos completos al comienzo del desarrollo.

A continuación, vamos a comentar las etapas que componen cada iteración (imagen 26):

- **Comunicación**: en esta etapa se lleva a cabo la comunicación con el cliente y se levantan los requerimientos que tendrá el software.
- **Planeación**: describe las tareas técnicas pendientes de realizar, los riesgos probables, los recursos que se requieren, los productos del trabajo que se obtendrán y una programación de las actividades.
- **Modelado**: un ingeniero de software crea modelos con el fin de entender mejor los requerimientos del software y el diseño que los satisfará.
- **Construcción**: esta actividad combina la generación de código (ya sea manual o automatizada) y las pruebas que se requieren para descubrir errores en éste.
- **Despliegue**: el software (como entidad completa o como un incremento parcialmente terminado) se entrega al consumidor que lo evalúa y que le da retroalimentación.

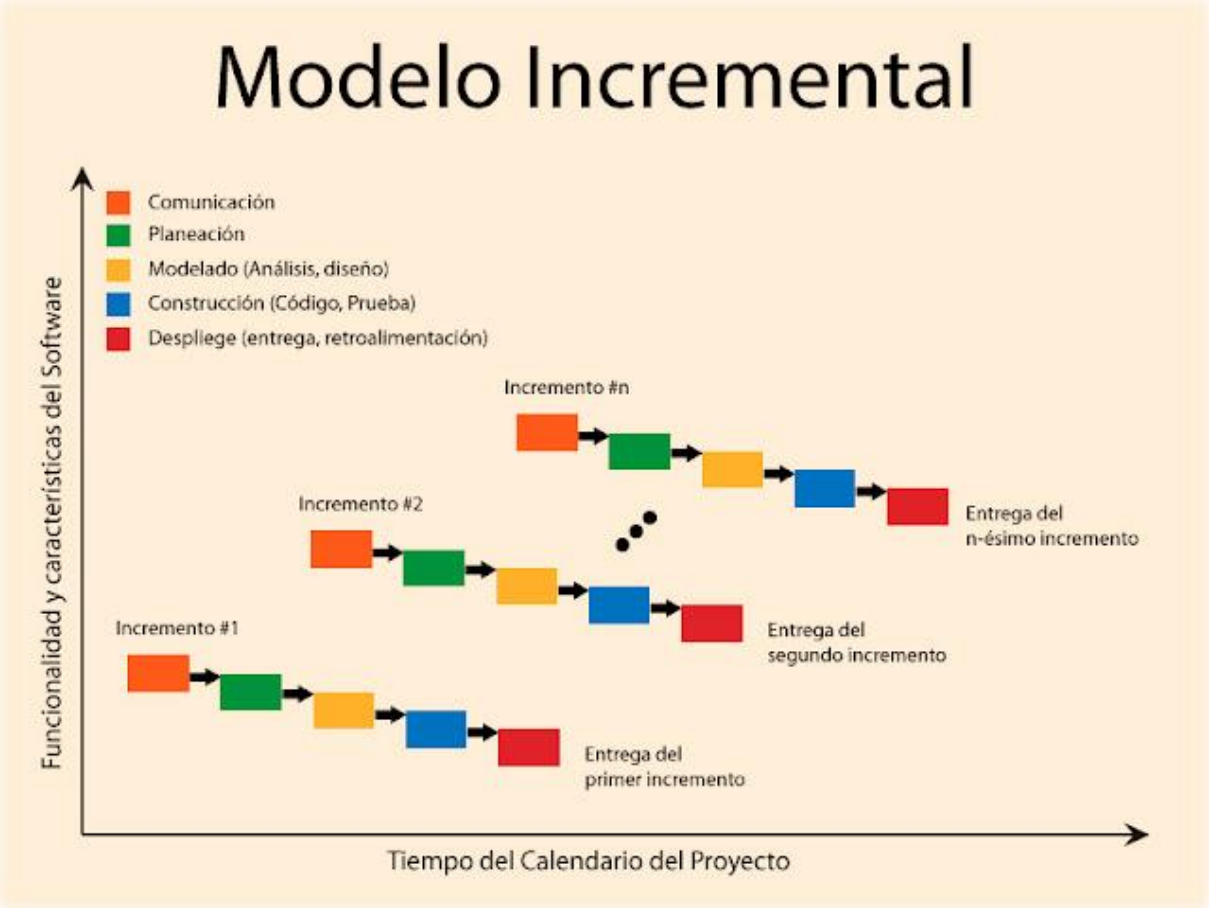


Ilustración 26: Etapas de un incremento en un modelo incremental. Fuente.

3. ANÁLISIS Y PLANIFICACIÓN

Un buen diseño software permite el buen entendimiento de los requisitos haciendo que el software tenga la funcionalidad necesaria, creando un sistema robusto y en el que sus piezas se pueden usar en otros sistemas. Siempre un buen diseño ayuda a conseguir un buen software, aunque un buen diseño no implica un buen software final.

3.1. Captura de requisitos

Para fijar los requisitos funcionales del sistema se han reflejado en dos diagramas de casos de uso (ilustración 27 y 28), dependiendo si es un usuario o un administrador. El diagrama de casos de uso permite reflejar el comportamiento del sistema desde el punto de vista del cliente.

En la fase inicial del desarrollo de estos casos de uso fueron empleados para construir la base del proyecto, y a través de los diferentes incrementos, se irá mejorando el diseño y comportamiento de nuestro sistema.

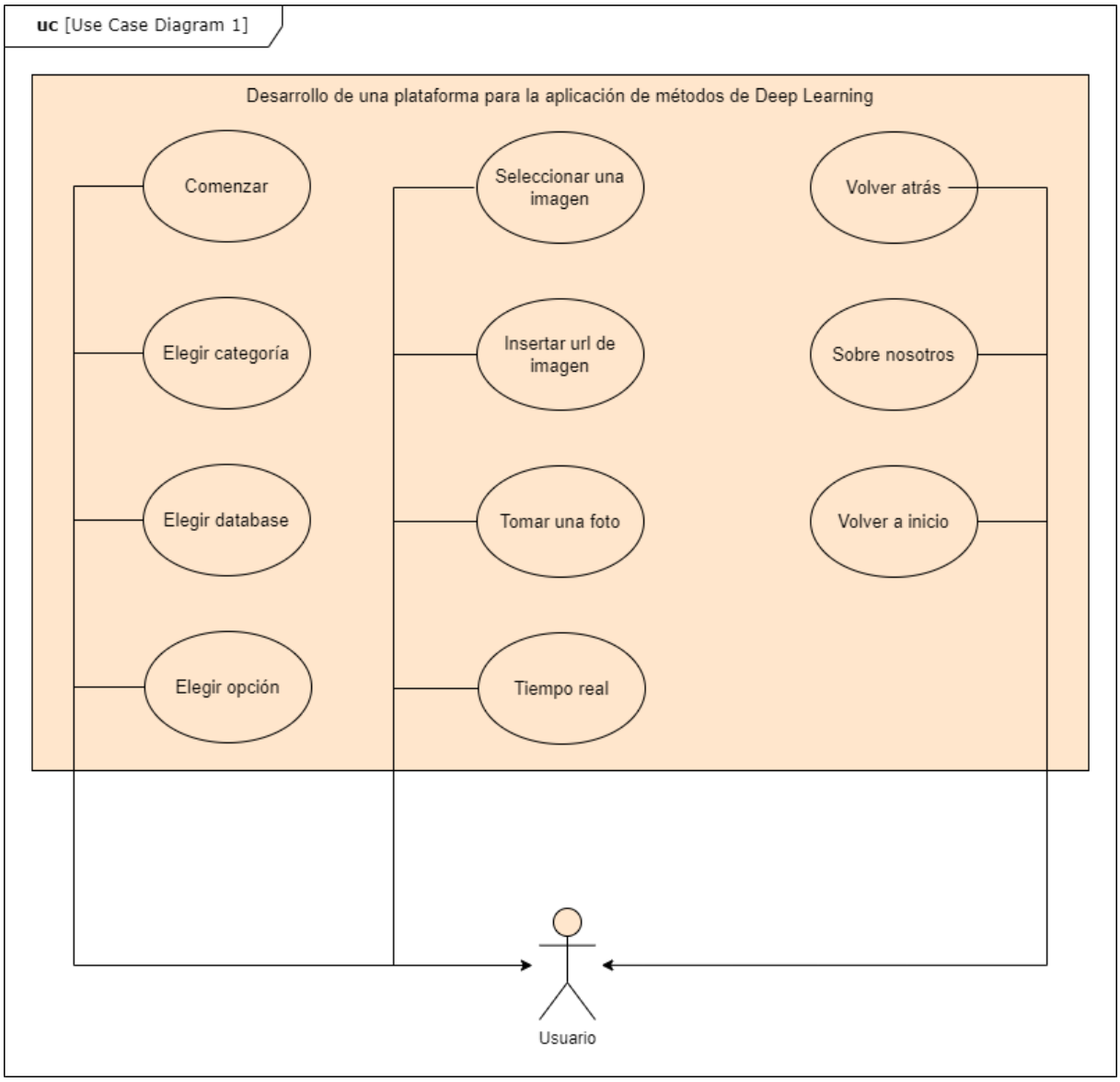


Ilustración 27: Diagrama de casos de uso del sistema de un usuario

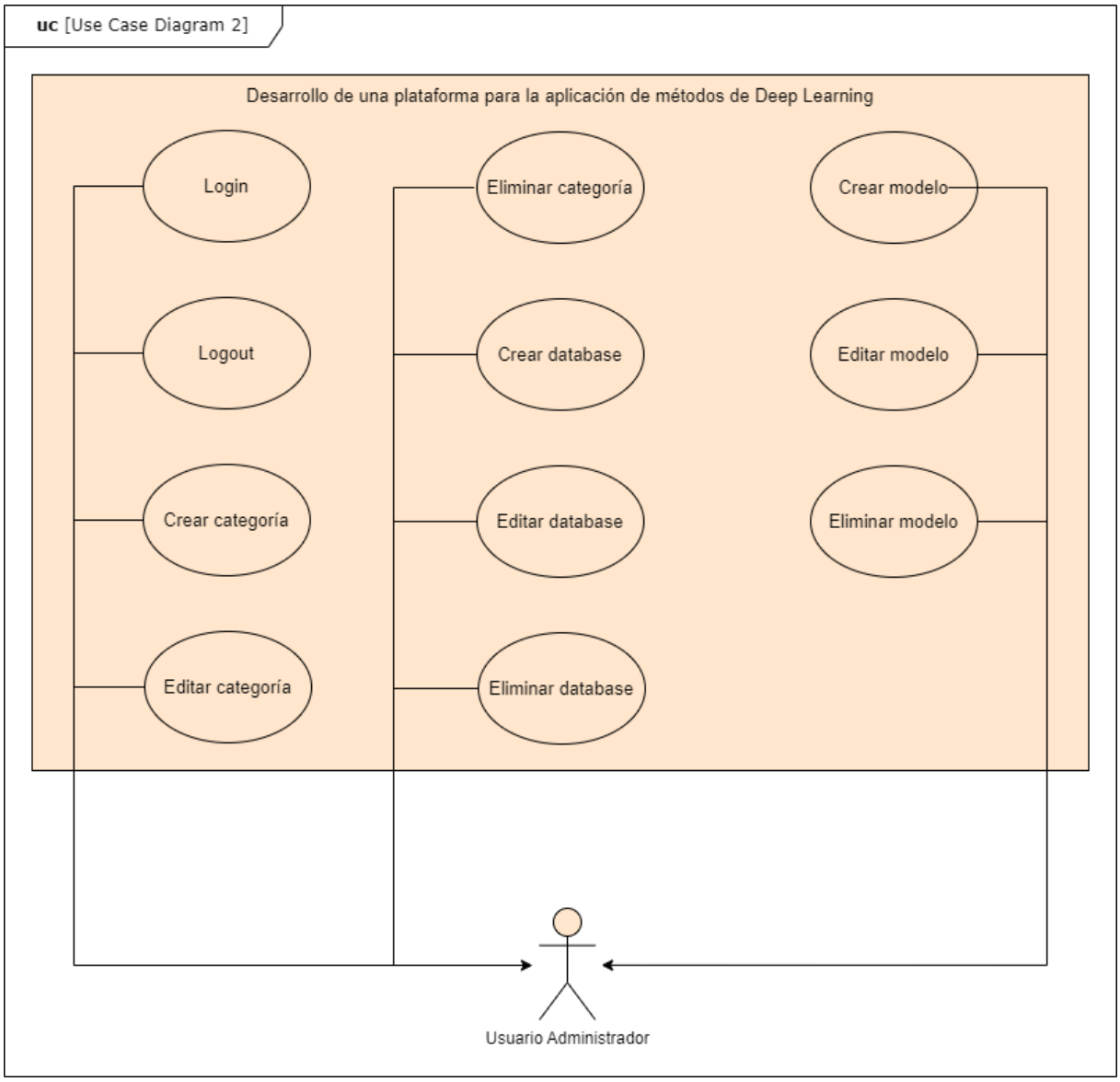


Ilustración 28: Diagrama de casos de uso del sistema de un administrador

3.2. Requisitos funcionales

A continuación, se van a presentar los distintos casos de uso donde se podrá ver toda la funcionalidad y flujo de la aplicación. Para ello, se va a dividir en función de quién sea el actor principal, bien sea un usuario normal o un usuario administrador.

3.2.1. Actor usuario

En la siguiente tabla (tabla 1) veremos los casos de uso de un usuario normal o **CUN**.

Código	Caso de uso
CUN-1	Comenzar
CUN-2	Elegir categoría
CUN-3	Elegir database
CUN-4	Elegir opción
CUN-5	Seleccionar una imagen
CUN-6	Insertar url de imagen
CUN-7	Tomar una foto
CUN-8	Tiempo real
CUN-9	Volver atrás
CUN-10	Sobre nosotros
CUN-11	Volver a inicio

Tabla 1: Tabla de casos de uso de un usuario normal

Una vez cargada la aplicación, se muestra la página principal dónde se define la primera interacción CUN-1 en la que el usuario comienza a manejar la aplicación (tabla 2).

CUN-1	Comenzar
Actores	Usuario, sistema
Precondiciones	Página web cargada
Postcondiciones	Ninguna
Escenario principal	1. El sistema muestra la vista principal 2. El usuario hace click en el botón Acceder 3. Fin del escenario con éxito

Escenarios alternativos	Ninguno
Escenarios de excepción	Ninguno

Tabla 2: Tabla de código de usuario normal 1

El siguiente paso en este trayecto es elegir una categoría entre las diferentes opciones que se presentan (tabla 3).

CUN-2	**Elegir categoría**
Actores	Usuario, sistema, base de datos
Precondiciones	CUN-1
Postcondiciones	Ninguna
Escenario principal	1. La base de datos devuelve todas las categorías 2. El sistema muestra la vista de categorías con las categorías 3. El usuario elige una categoría entre las disponibles 4. Fin del escenario con éxito
Escenarios alternativos	Ninguno
Escenarios de excepción	Ninguno

Tabla 3: Tabla de código de usuario normal 2

Continuamos, eligiendo la base de datos (tabla 4).

CUN-3	Elegir base de datos
Actores	Usuario, sistema, base de datos
Precondiciones	CUN-2
Postcondiciones	Ninguna
Escenario principal	1. La base de datos devuelve todas las bases de datos 2. El sistema muestra la vista de bases de datos con las bases de datos 3. El usuario elige una base de datos entre las disponibles 4. Fin del escenario con éxito
Escenarios alternativos	Ninguno
Escenarios de excepción	Ninguno

Tabla 4: Tabla de código de usuario normal 3

Con todos estos elementos seleccionados, solo queda elegir la opción con la que se van a insertar imágenes a la aplicación (tabla 5).

CUN-4	Elegir opción
Actores	Usuario, sistema, base de datos
Precondiciones	CUN-3
Postcondiciones	Ninguna
Escenario principal	1. La base de datos devuelve todas las opciones 2. El sistema muestra la vista de opciones con las opciones 3. El usuario elige una opción entre las disponibles 4. Fin del escenario con éxito
Escenarios alternativos	Ninguno
Escenarios de excepción	Ninguno

Tabla 5: Tabla de código de usuario normal 4

La siguiente interacción dependerá de la opción elegida para insertar imágenes. Si se ha elegido la opción de “Elegir una imagen” llegamos a la CUN-5 (tabla 6).

CUN-5	Seleccionar una imagen
Actores	Usuario, sistema, base de datos
Precondiciones	En CUN-4 haber elegido "Elegir imagen"
Postcondiciones	Ninguna
Escenario principal	1. El sistema muestra la vista de elegir imagen 2. El usuario elige una opción entre las disponibles 3. Fin del escenario con éxito
Escenarios alternativos	Ninguno
Escenarios de excepción	Ninguno

Tabla 6: Tabla de código de usuario normal 5

Si se selecciona la opción insertar url (tabla 7):

CUN-6	Insertar url de imagen
Actores	Usuario, sistema, base de datos
Precondiciones	En CUN-4 haber elegido "Elegir una url"
Postcondiciones	El sistema enseña la imagen de la url al clickar en "Elegir"
Escenario principal	1. El sistema muestra la vista de insertar url 2. El usuario inserta una url de una imagen de internet y hace click en "Elegir" 3. El sistema descarga la url y muestra la vista para elegir otra url o enviar la elegida 4. Hace click en "Enviar foto elegida" 5. Fin del escenario con éxito
Escenarios alternativos	4.a El usuario decide enviar otra imagen 1. El usuario escribe otra url

	2. Vuelve al estado 3
Escenarios de excepción	2.a El url insertado no es válido 1. Vuelve al estado 1 con un mensaje de error 4.a.a El url insertado no es válido 1. Vuelve al estado 1 con un mensaje de error

Tabla 7: Tabla de código de usuario normal 6

Por otro lado, si elegimos la opción tomar una foto (tabla 8):

CUN-7	Tomar una foto
Actores	Usuario, sistema, base de datos
Precondiciones	En CUN-4 haber elegido "Tomar una foto"
Postcondiciones	El sistema enseña la foto al clicar en el botón de la cámara
Escenario principal	1. El sistema muestra la vista de tomar una foto 2. El usuario se toma una foto al hacer click en el botón de la cámara 3. El sistema muestra la vista para tomar otra foto o enviar la ya tomada 4. Hace click en "Enviar foto elegida" 5. Fin del escenario con éxito
Escenarios alternativos	4.a El usuario decide enviar otra imagen 1. El usuario escribe otra url 2. Vuelve al estado 3
Escenarios de excepción	Ninguno

Tabla 8: Tabla de código de usuario normal 7

Como última opción está la interacción CUN-8 (tabla 9).

CUN-8	Tiempo real
Actores	Usuario, sistema, base de datos
Precondiciones	En CUN-4 haber elegido "Tiempo real"
Postcondiciones	Muestra en tiempo real imágenes entrenadas
Escenario principal	1. El sistema muestra la vista de tiempo real y entra en un bucle de cojo fotograma de la cámara, lo entrena y lo muestra 2. El usuario interactúa moviéndose y observando las predicciones del modelo 3. Fin del escenario con éxito
Escenarios alternativos	Ninguno
Escenarios de excepción	Ninguno

Tabla 9: Tabla de código de usuario normal 8

En último lugar, se muestra una vista en la que se enseña el resultado de lo elegido. Además, una última interacción para volver a comenzar (tabla 10).

CUN-9	Volver a inicio
Actores	Usuario, sistema
Precondiciones	CUN-5 o CUN-6 o CUN-7 o CUN-8 o CUN-11
Postcondiciones	Ninguna
Escenario principal	1. El sistema predice la imagen y muestra la vista de resultado con el resultado 2. El usuario hace click en el botón "Volver a empezar" para volver a CUN-1 3. Fin del escenario con éxito
Escenarios alternativos	Ninguno
Escenarios de excepción	Ninguno

Tabla 10: Tabla de código de usuario normal 9

Además de toda esta funcionalidad, independientemente de la vista en la que se encuentre el usuario puede volver hacia atrás con un botón (tabla 11).

CUN-10	Volver atrás
Actores	Usuario, sistema
Precondiciones	Haber seleccionado algún CUN del 2 al 9
Postcondiciones	Ninguna
Escenario principal	1. El sistema muestra una vista 2. El usuario hace click en el botón con el dibujo de una flecha 3. Fin del escenario con éxito
Escenarios alternativos	Ninguno
Escenarios de excepción	Ninguno

Tabla 11: Tabla de código de usuario normal 10

Para finalizar, si el usuario desea visualizar un poco de información sobre la aplicación, puede hacerlo a través de la interacción CUN-11 (tabla 12).

CUN-11	Sobre nosotros
Actores	Usuario, sistema
Precondiciones	La aplicación esté cargada
Postcondiciones	Ninguna
Escenario principal	1. El sistema muestra cualquier vista 2. El usuario hace click en el botón "Sobre nosotros" 3. Fin del escenario con éxito
Escenarios alternativos	Ninguno
Escenarios de excepción	Ninguno

Tabla 12: Tabla de código de usuario normal 11

3.2.2. Actor principal administrador

Se va a proceder a ver las distintas interacciones de un usuario administrador con el sistema y con la base de datos (tabla 13).

Código	Caso de uso
CUA-1	Login
CUA-2	Logout
CUA-3	Crear categoría
CUA-4	Editar categoría
CUA-5	Eliminar categoría
CUA-6	Crear database
CUA-7	Editar database
CUA-8	Eliminar database
CUA-9	Crear modelo
CUA-10	Editar modelo
CUA-11	Eliminar modelo

Tabla 13: Tabla de casos de uso de un usuario administrador

Esta aplicación, además de toda la funcionalidad vista anteriormente, permite a un usuario administrador añadir elementos a la base de datos (tabla 14).

CUA-1	Login
Actores	Administrador, sistema, base de datos
Precondiciones	Página web cargada y el administrador entre manualmente al enlace
Postcondiciones	Ninguna
Escenario principal	1. El sistema muestra la vista de login 2. El administrador inserta las credenciales 3. La base de datos comprueba si existe las credenciales 4. El sistema recibe si existe o no 5. Fin del escenario con éxito
Escenarios alternativos	Ninguno
Escenarios de excepción	2.a Las credenciales no son correctas 1. Vuelve al estado 1 con un mensaje de error

Tabla 14: Tabla de código de usuario administrador 1

Por contraposición, existe la interacción de poder hacer logout (tabla 15).

CUA-2	Logout
Actores	Administrador, sistema
Precondiciones	CUA-1
Postcondiciones	Ninguna
Escenario principal	1. El sistema muestra una vista de administrador 2. El administrador hace click en Logout 3. Fin del escenario con éxito
Escenarios alternativos	Ninguno
Escenarios de excepción	Ninguno

Tabla 15: Tabla de código de usuario administrador 2

A continuación, en las siguientes tablas, se van a mostrar las distintas interacciones, como la opción de añadir, editar y borrar aplicada para cada objeto o elemento de la base de datos (tabla 16-24).

CUA-3	Crear categoría
Actores	Administrador, sistema, base de datos
Precondiciones	CUA-1
Postcondiciones	Mensaje de confirmación
Escenario principal	1. El sistema muestra una vista con un formulario de categorías 2. El usuario rellena el formulario 3. El sistema recibe el formulario, lo valida y lo envía a la base de datos 4. La base de datos recibe la categoría, la almacena y devuelve un mensaje de confirmación 5. El sistema recibe el mensaje de confirmación y lo devuelve al usuario 6. Fin del escenario con éxito
Escenarios alternativos	Ninguno
Escenarios de excepción	3.a El sistema recibe un formulario con datos no válidos 1. Vuelve al estado 1 con un mensaje de error

Tabla 16: Tabla de código de usuario administrador 3

CUA-4	Editar categoría
Actores	Administrador, sistema, base de datos
Precondiciones	CUA-1 y que exista alguna categoría
Postcondiciones	Mensaje de confirmación
Escenario principal	1. El sistema le pide a la base de datos todas las categorías

	2. La base de datos devuelve todas las categorías 3. El sistema muestra una vista con todas las categorías 4. El usuario elige que categoría quiere editar 5. El sistema devuelve una vista con un formulario relleno con los datos actuales de esa categoría 6. El usuario edita el atributo que desee editar y le da a guardar 7. El sistema recibe el formulario, lo valida y lo envía a la base de datos 8. La base de datos recibe la categoría, la almacena y devuelve un mensaje de confirmación 9. El sistema recibe el mensaje de confirmación y lo devuelve al usuario 10. Fin del escenario con éxito
Escenarios alternativos	Ninguno
Escenarios de excepción	7.a El sistema recibe un formulario con datos no válidos 1. Vuelve al estado 1 con un mensaje de error

Tabla 17: Tabla de código de usuario administrador 4

CUA-5	Eliminar categoría
Actores	Administrador, sistema, base de datos
Precondiciones	CUA-1 y existe una categoría
Postcondiciones	Mensaje de confirmación
Escenario principal	1. El sistema le pide a la base de datos todas las categorías 2. La base de datos devuelve todas las categorías 3. El sistema muestra una vista con todas las categorías 4. El usuario elige que categoría quiere borrar y le da a borrar 5. El sistema envía la categoría que quiere borrar a la base de datos 6. La base de datos elimina la categoría y devuelve un mensaje de confirmación

	7. El sistema recibe el mensaje y se lo devuelve al usuario 8. Fin del escenario con éxito
Escenarios alternativos	Ninguno
Escenarios de excepción	Ninguno

Tabla 18: Tabla de código de usuario administrador 5

CUA-6	Crear database
Actores	Administrador, sistema, base de datos
Precondiciones	CUA-1
Postcondiciones	Mensaje de confirmación
Escenario principal	1. El sistema muestra una vista con un formulario de database 2. El usuario rellena el formulario 3. El sistema recibe el formulario, lo valida y lo envía a la base de datos 4. La base de datos recibe la database, la almacena y devuelve un mensaje de confirmación 5. El sistema recibe el mensaje de confirmación y lo devuelve al usuario 6. Fin del escenario con éxito
Escenarios alternativos	Ninguno
Escenarios de excepción	3.a El sistema recibe un formulario con datos no válidos 1. Vuelve al estado 1 con un mensaje de error

Tabla 19: Tabla de código de usuario administrador 6

CUA-7	Editar database
Actores	Administrador, sistema, base de datos
Precondiciones	CUA-1
Postcondiciones	Mensaje de confirmación
Escenario principal	1. El sistema le pide a la base de datos todas las databases 2. La base de datos devuelve todas las database 3. El sistema muestra una vista con todas las database 4. El usuario elige que database quiere editar 5. El sistema devuelve una vista con un formulario relleno con los datos actuales de esa database 6. El usuario edita el atributo que desee editar y le da a guardar 7. El sistema recibe el formulario, lo valida y lo envía a la base de datos 8. La base de datos recibe la database, la almacena y devuelve un mensaje de confirmación 9. El sistema recibe el mensaje de confirmación y lo devuelve al usuario 10. Fin del escenario con éxito
Escenarios alternativos	Ninguno
Escenarios de excepción	7.a El sistema recibe un formulario con datos no válidos 1. Vuelve al estado 1 con un mensaje de error

Tabla 20: Tabla de código de usuario administrador 7

CUA-8	Eliminar database
Actores	Administrador, sistema, base de datos
Precondiciones	CUA-1 y existe una categoría
Postcondiciones	Mensaje de confirmación
Escenario principal	1. El sistema le pide a la base de datos todas las database

	2. La base de datos devuelve todas las database 3. El sistema muestra una vista con todas las database 4. El usuario elige que database quiere borrar y le da a borrar 5. El sistema envía la database que quiere borrar a la base de datos 6. La base de datos elimina la database y devuelve un mensaje de confirmación 7. El sistema recibe el mensaje y se lo devuelve al usuario 8. Fin del escenario con éxito
Escenarios alternativos	Ninguno
Escenarios de excepción	Ninguno

Tabla 21: Tabla de código de usuario administrador 8

CUA-9	Crear modelo
Actores	Administrador, sistema, base de datos
Precondiciones	CUA-1
Postcondiciones	Mensaje de confirmación
Escenario principal	1. El sistema muestra una vista con un formulario de modelo 2. El usuario rellena el formulario 3. El sistema recibe el formulario, lo valida y lo envía a la base de datos 4. La base de datos recibe el modelo, la almacena y devuelve un mensaje de confirmación 5. El sistema recibe el mensaje de confirmación y lo devuelve al usuario 6. Fin del escenario con éxito
Escenarios alternativos	Ninguno
Escenarios de excepción	3.a El sistema recibe un formulario con datos no válidos 1. Vuelve al estado 1 con un mensaje de error

Tabla 22: Tabla de código de usuario administrador 9

CUA-10	Editar modelo
Actores	Administrador, sistema, base de datos
Precondiciones	CUA-1
Postcondiciones	Mensaje de confirmación
Escenario principal	1. El sistema le pide a la base de datos todos los modelos 2. La base de datos devuelve todos los modelos 3. El sistema muestra una vista con todos los modelos 4. El usuario elige que modelo quiere editar 5. El sistema devuelve una vista con un formulario relleno con los datos actuales de ese modelo 6. El usuario edita el atributo que desee editar y le da a guardar 7. El sistema recibe el formulario, lo valida y lo envía a la base de datos 8. La base de datos recibe el modelo, lo almacena y devuelve un mensaje de confirmación 9. El sistema recibe el mensaje de confirmación y lo devuelve al usuario 10. Fin del escenario con éxito
Escenarios alternativos	Ninguno
Escenarios de excepción	7.a El sistema recibe un formulario con datos no válidos 1. Vuelve al estado 1 con un mensaje de error

Tabla 23: Tabla de código de usuario administrador 10

CUA-11	Eliminar database
Actores	Administrador, sistema, base de datos
Precondiciones	CUA-1 y existe una categoría
Postcondiciones	Mensaje de confirmación
Escenario principal	1. El sistema le pide a la base de datos todos los modelos 2. La base de datos devuelve todos los modelos 3. El sistema muestra una vista con todos los modelos

	4. El usuario elige que modelo quiere borrar y le da a borrar 5. El sistema envía el modelo que quiere borrar a la base de datos 6. La base de datos elimina el modelo y devuelve un mensaje de confirmación 7. El sistema recibe el mensaje y se lo devuelve al usuario 8. Fin del escenario con éxito
Escenarios alternativos	Ninguno
Escenarios de excepción	Ninguno

Tabla 24: Tabla de código de usuario administrador 11

3.3. Requisitos no funcionales

Los requisitos no funcionales son los elementos impuestos a un determinado software para garantizar su calidad. En la mayoría de los proyectos se denotan con adjetivos como seguridad, rendimiento y escalabilidad. Los requisitos no funcionales son imprescindibles porque aseguran que el software cumpla con las necesitades del usuario.

A continuación, se van a comentar los requisitos no funcionales de la aplicación:

- **Seguridad**: el sistema debe estar implementado para que un usuario cualquiera no pueda entrar en la interfaz de administrador por ejemplo.
- **Escalabilidad:** el software debe poder escalar tanto hacia arriba o hacia abajo.
- **Mantenimiento:** el software debe ser fácil de actualizar y de mantener.
- **Portabilidad:** puede ejecutarse en distintos equipos con pocos cambios.
- **Fiabilidad:** debe cumplir con los requisitos de usuario.
- **Usabilidad:** el software tiene que ser fácil tanto de usar como de comprender.

3.4. Planificación temporal

Una vez que ya estaba todo claro, se sabía por dónde empezar y se tenía una idea general sobre la aplicación se empezó a trabajar. Se va a mostrar una planificación del tiempo que ha llevado todo el desarrollo de la app

- Hubo una primera etapa que se dedicó a la elección de las herramientas de entre todas las posibilidades vistas.
- Una vez escogidas, vino una segunda etapa de reflexión sobre cómo conseguir todo lo que había sido plasmado en el plan de proyecto.
- En esta tercera etapa, se desarrolla toda la ingeniería del software que se va a mostrar ahora y se diseñó la base de datos que hemos visto en el punto anterior.
- La cuarta y quinta etapa es el desarrollo de la aplicación. Se divide en dos fases ya que durante la primera se desarrolla la aplicación (la lógica, las vistas, etc.) y durante la segunda se le aplican mejoras. Esta última se debe a que durante la exposición de la aplicación en “La Noche Europea de los Investigadores” se detectaron cambios que podrían perfeccionar la aplicación para su posterior exposición en el Parque de las Ciencias de Granada en la zona de inteligencia artificial.
- Una vez instalada la aplicación, la memoria se ha ido implementando a lo largo de los puntos anteriores y este punto se ha centrado en acabar de redactar la memoria para la posterior exposición del TFG.
- Por último, una etapa de mejoras, que no se sabe la duración de esta ya que variará en función de las necesidades y actualizaciones que requiera la aplicación.

Tarea	Duración
Elección de herramientas	1 sem
Reflexionar sobre la lógica de la aplicación	1 sem
Aprender sobre las herramientas	3 sem
Desarrollo fase 1	15 sem
Desarrollo fase 2	10 sem
Acabar la redacción de la memoria	15 sem
Desarrollo fase 3	Indefinido
Duración Total	**45 sem**

Tabla 25: Tabla resumen de la planificación temporal

Ya que el modelo de desarrollo elegido es el modelo incremental, la parte de desarrollo del software y la fase de pruebas del mismo no se separa, es decir, estas fases se irán intercalando.

Por tanto, todo el proceso que se ha implicado en esta memoria consta de un total de 45 semanas. Para visualizar de forma más sencilla la estimación de la tabla anterior, se va a mostrar la misma estimación, pero con el diagrama de Gantt en las siguientes tablas (tabla 26-30).

Tarea	Duración	1	2	3	4	5	6	7	8	9	10
Elección de herramientas	1 sem										
Reflexionar sobre la lógica de la aplicación	1 sem										
Aprender sobre las herramientas	3 sem										

Desarrollo fase 1	15 sem										
Desarrollo fase 2	10 sem										
Acabar la redacción de la memoria	15 sem										

Tabla 26: Diagrama de Gantt (1)

Tarea	**Duración**	11	12	13	14	15	16	17	18	19	20
Elección de herramientas	1 sem										
Reflexionar sobre la lógica de la aplicación	1 sem										
Aprender sobre las herramientas	3 sem										
Desarrollo fase 1	15 sem										
Desarrollo fase 2	10 sem										
Acabar la redacción de la memoria	15 sem										

Tabla 27: Diagrama de Gantt (2)

Tarea	**Duración**	21	22	23	24	25	26	27	28	29	30
Elección de herramientas	1 sem										

Reflexionar sobre la lógica de la aplicación	1 sem										
Aprender sobre las herramientas	3 sem										
Desarrollo fase 1	15 sem										
Desarrollo fase 2	10 sem										
Acabar la redacción de la memoria	15 sem										

Tabla 28: Diagrama de Gantt (3)

Tarea	**Duración**	31	32	33	34	35	36	37	38	39	40
Elección de herramientas	1 sem										
Reflexionar sobre la lógica de la aplicación	1 sem										
Aprender sobre las herramientas	3 sem										
Desarrollo fase 1	15 sem										
Desarrollo fase 2	10 sem										
Acabar la redacción de la memoria	15 sem										

Tabla 29: Diagrama de Gantt (4)

Tarea	Duración	41	42	43	44	45
Elección de herramientas	1 sem					
Reflexionar sobre la lógica de la aplicación	1 sem					
Aprender sobre las herramientas	3 sem					
Desarrollo fase 1	15 sem					
Desarrollo fase 2	10 sem					
Acabar la redacción de la memoria	15 sem					

Tabla 30: Diagrama de Gantt (5)

El desarrollo de la fase 3 no la vamos a tener en cuenta, puesto que es posterior al desarrollo de la memoria.

3.5. Estimación de costes

Una vez estimado el tiempo aproximado de la duración del desarrollo del Trabajo de Fin de Grado se procederá a calcular el coste teniendo en cuenta las 45 semanas totales del desarrollo y la participación de una única persona durante todo el proceso.

Según las consultas recientes, se ha estimado que el salario de un programador web promedio en España es de € 1.536,78 al mes o € 9,61 (Boletín oficial del estado, 2022). Los cargos de nivel inicial comienzan con un ingreso de € 19.500 al año, mientras que profesionales más experimentados perciben hasta € 33.000 al año.

Con esta media de 9,61€ la hora, el desarrollador de este TFG además de la realización del mismo, trabaja 8 horas diarias, por lo tanto, saldría una media de trabajo diaria de 5 horas. Hay que tener en cuenta que en 45 semanas existen (45 *

5) 225 días laborales. Con este último parámetro podemos valorar el trabajo del desarrollador, en 225 días trabajados, se han alcanzado un total de 300 horas y por 9.61 euros la hora, se obtiene un resultado total de 2.883€.

Además, habría que añadir los gastos del hardware, tanto la parte del desarrollador como la de la universidad, ya que para que funcione de forma eficiente y fluida, se requiere de una tarjeta gráfica potente.

El desarrollador ha utilizado un ordenador portátil Lenovo de 1.095 euros de 2018, un monitor de 299 euros y 50 euros en dispositivos periféricos (teclado, ratón, cascos…).

La universidad por su parte, ha comprado equipo específicamente para el uso de esta aplicación valorado en 2180 euros.

Suponiendo que los dos equipos tienen una vida útil de 5 años, el proyecto dura 45 semanas, o lo que es lo mismo, 10.5 meses. Si dividimos estos presupuestos entre 60, que son la cantidad de meses en 5 años nos sale el precio por mes. Al multiplicarlo por los 10.5 meses tendríamos el costo de cada equipo.

- Equipo desarrollador: (1444 euros / 60 meses) x 10.5 meses = 252,7
- Equipo Universidad: (2180 euros / 60 meses) x 10.5 meses = 381,5

Por último, todas las licencias de software son gratuitas, por lo que no supondría ningún gasto. A continuación, vemos un resumen en la tabla 31.

Concepto	Precio en euros
Trabajo de desarrollador total	2.883
Gastos en equipo del desarrollador	252,7
Gastos por parte de la Universidad	381,5
Total Gastos	3.517,2

Tabla 31: Tabla resumen de gastos

4. DISEÑO

Un buen diseño de una aplicación proporciona un mejor entendimiento de los requisitos haciendo que el software tenga la funcionalidad necesaria, creando un sistema robusto en el que sus piezas se pueden usar en otros sistemas. Un buen diseño siempre ayuda a conseguir un buen software, aunque un buen diseño no implica un buen software final.

4.1. Diagramas de secuencia

A partir de los casos de uso se pueden realizar los diagramas de secuencia, que van a permitir el diseño de la interacción que va a tener el usuario con el software.

En los diagramas se diferencian tres actores principales, el usuario que actúa con la aplicación, el sistema que funciona como el cerebro de la aplicación y la base de datos donde se guardan todos los objetos creados.

Tal y como se hizo para los casos de usuario, se va a dividir en función de que el actor usuario sea un usuario normal o un usuario administrador.

4.1.1. Diagrama de secuencia de usuario normal

En el primer diagrama de secuencia se puede ver cómo el usuario da el primer paso para entrar en la aplicación (ilustración 29).

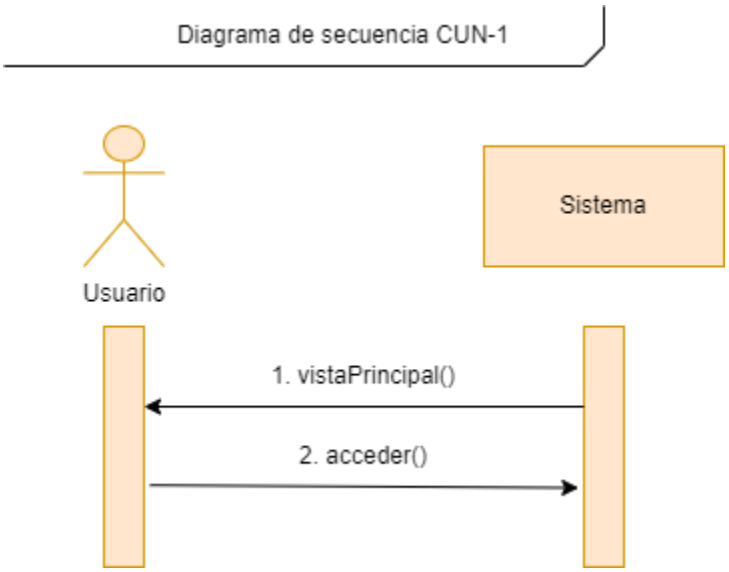


Ilustración 29: Diagrama de secuencia de código de usuario normal 1

En el siguiente diagrama el usuario elige la categoría entre todas las disponibles (ilustración 30).

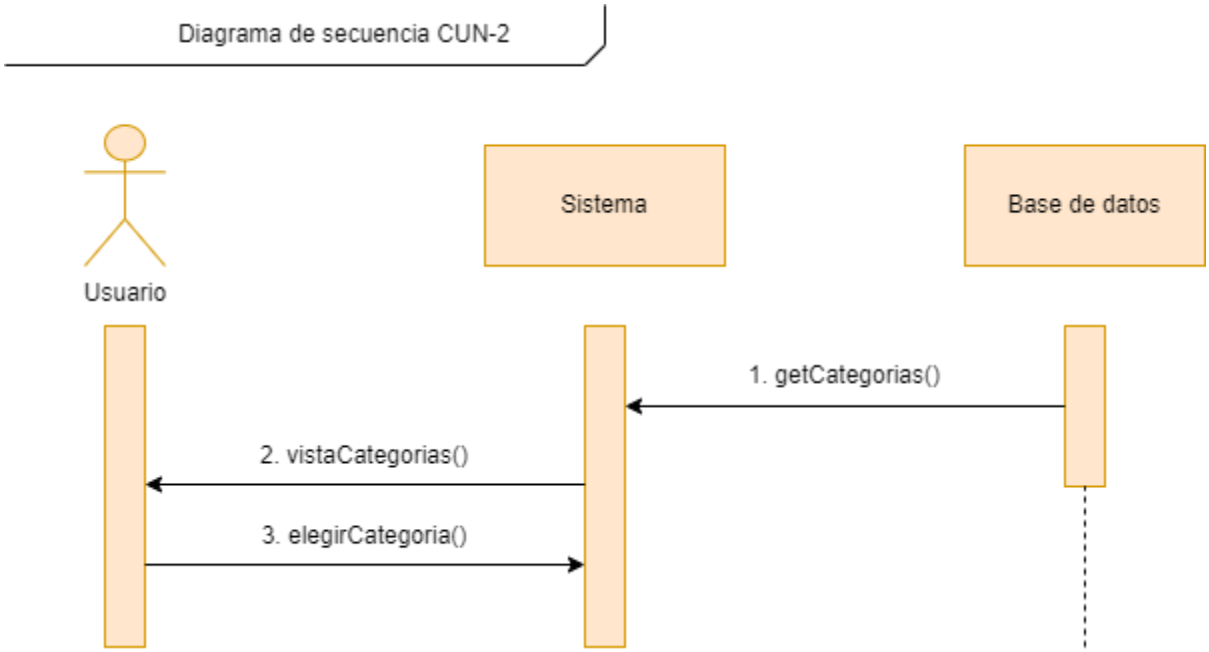


Ilustración 30: Diagrama de secuencia de código de usuario normal 2

A continuación, se ve cómo el usuario elige la base de datos (ilustración 31).

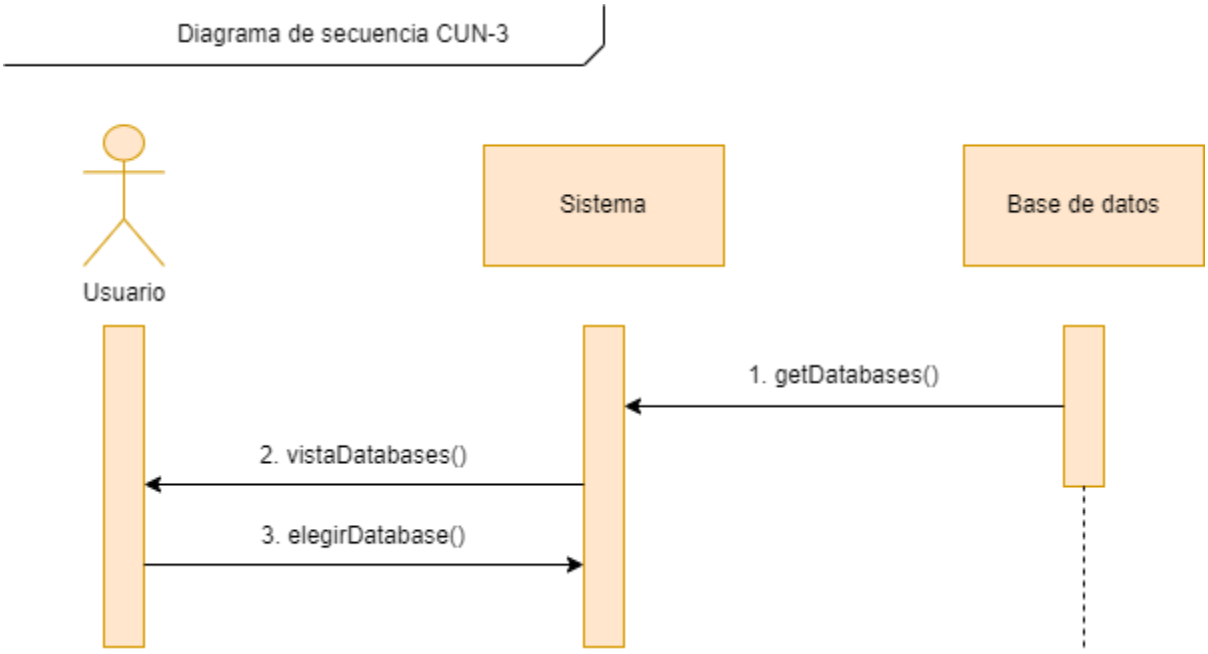


Ilustración 31: Diagrama de secuencia de código de usuario normal 3

Una vez elegida la base de datos y la categoría, a la hora de crear el modelo la base de datos devuelve las opciones que se han seleccionado. El modelo creado se tiene que seleccionar como “modelo por defecto “porque cada base de datos/categoría ha de tener un único modelo seleccionado como “por defecto” (ilustración 32).

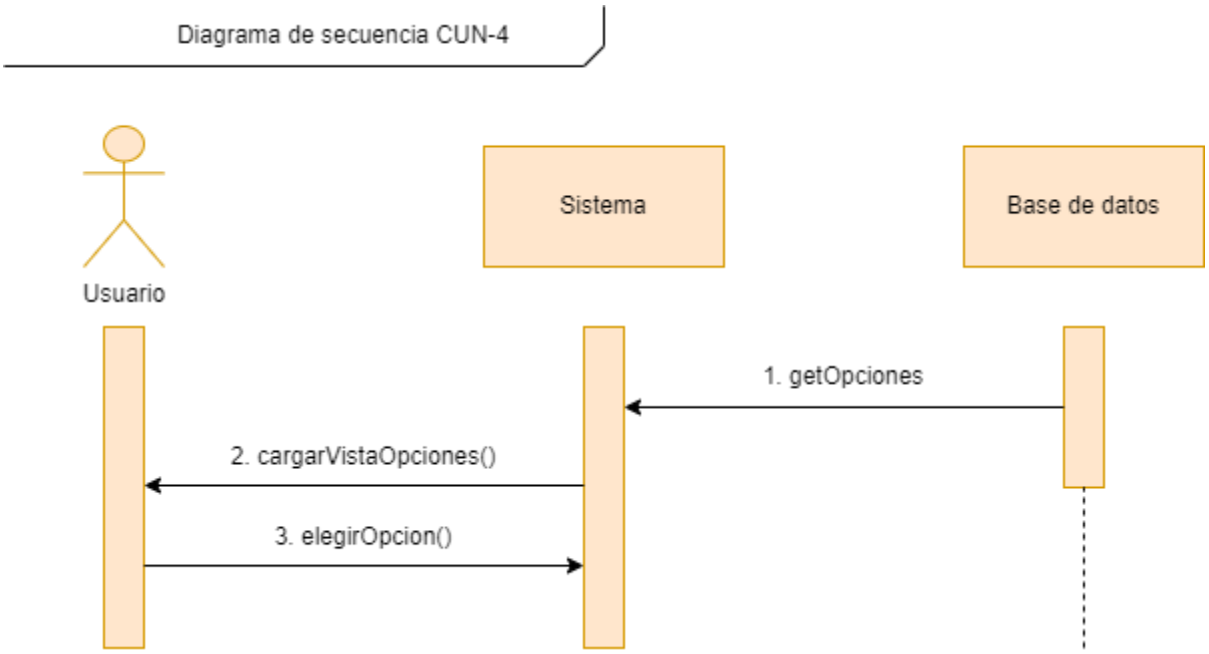


Ilustración 32: Diagrama de secuencia de código de usuario normal 4

Los próximos diagramas describen las distintas interacciones que le permiten al usuario insertar imágenes. En este primer diagrama (ilustración 33), si elegimos la opción “Elegir imagen” nos aparecerá una vista con imágenes, en la que dependiendo de la base de datos y la categoría proporcionará unas u otras, ya que estas son únicas para cada uno.

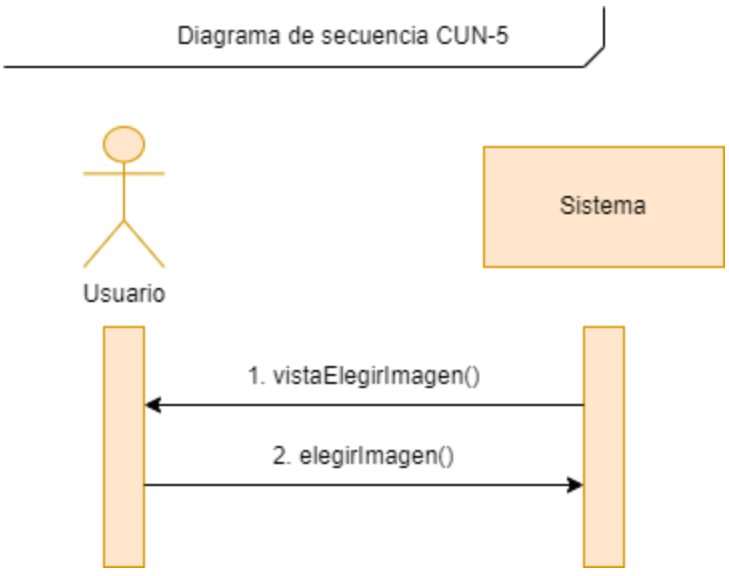


Ilustración 33: Diagrama de secuencia de código de usuario normal 5

El siguiente diagrama (ilustración 34) es para la opción “Insertar URL”, que como ya se veía en el caso de usuario, es un poco más complejo. En primer lugar, aparece la vista, en la se puede ver un cuadro de texto habilitado para insertar la url. Una vez insertada se pueden dar dos supuestos; que de error al intentar descargar la imagen, ya sea porque el enlace no sea válido o porque se ha dejado vacío el hueco (se indicará con un mensaje de error en color rojo independientemente de la razón causante del error) o que el enlace introducido sea correcto y por tanto aparecerá una nueva vista donde se verá la imagen cargada, dos botones y otra caja de texto.

Los botones tienen la funcionalidad de enviar la imagen o insertar un nuevo enlace y enviarla en el caso de que el usuario haya cambiado de opinión o incluso haya errado al insertar el enlace. Si al introducir el enlace, falla, volvería al mensaje de error comentado con anterioridad. Por esta razón, en este diagrama siguiente se aprecia un bucle, ya que cada vez que se equivoque el usuario, vuelva al principio de este diagrama de secuencia.

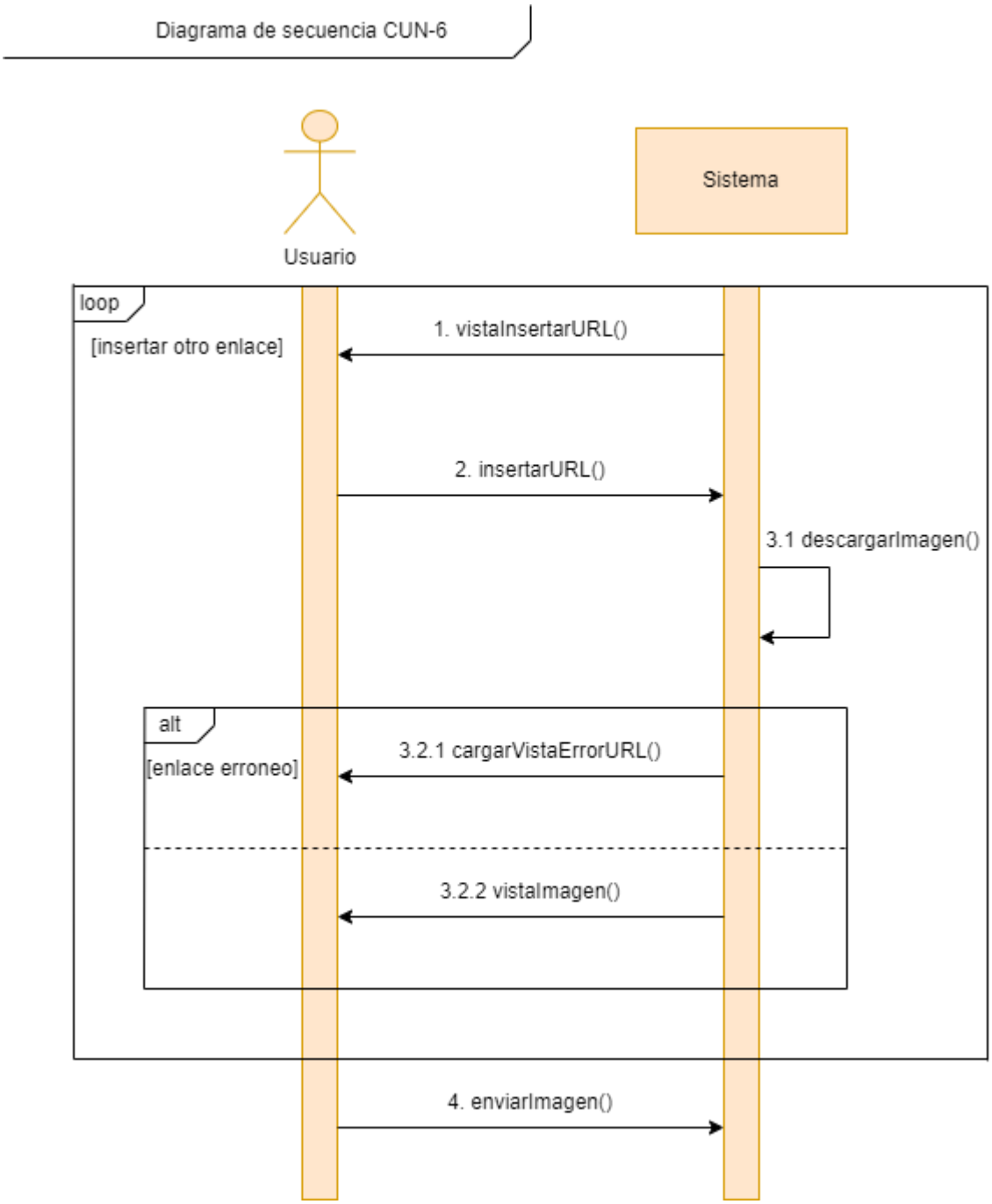


Ilustración 34: Diagrama de secuencia de código de usuario normal 6

Continuamos con el siguiente diagrama referente al caso de usuario normal 7 (diagrama 35). En este se explica qué pasaría si se ha elegido la opción de tomar una foto. Al usuario le aparecerá una vista en la cual podrá verse en el centro de la pantalla, en la parte inferior en la zona central hay un botón con la figura de una cámara en el

que al hacer click, echará una foto. Esta vista se refresca mostrando la imagen y además ofrece la oportunidad al usuario de echarse otra o enviar la ya tomada.

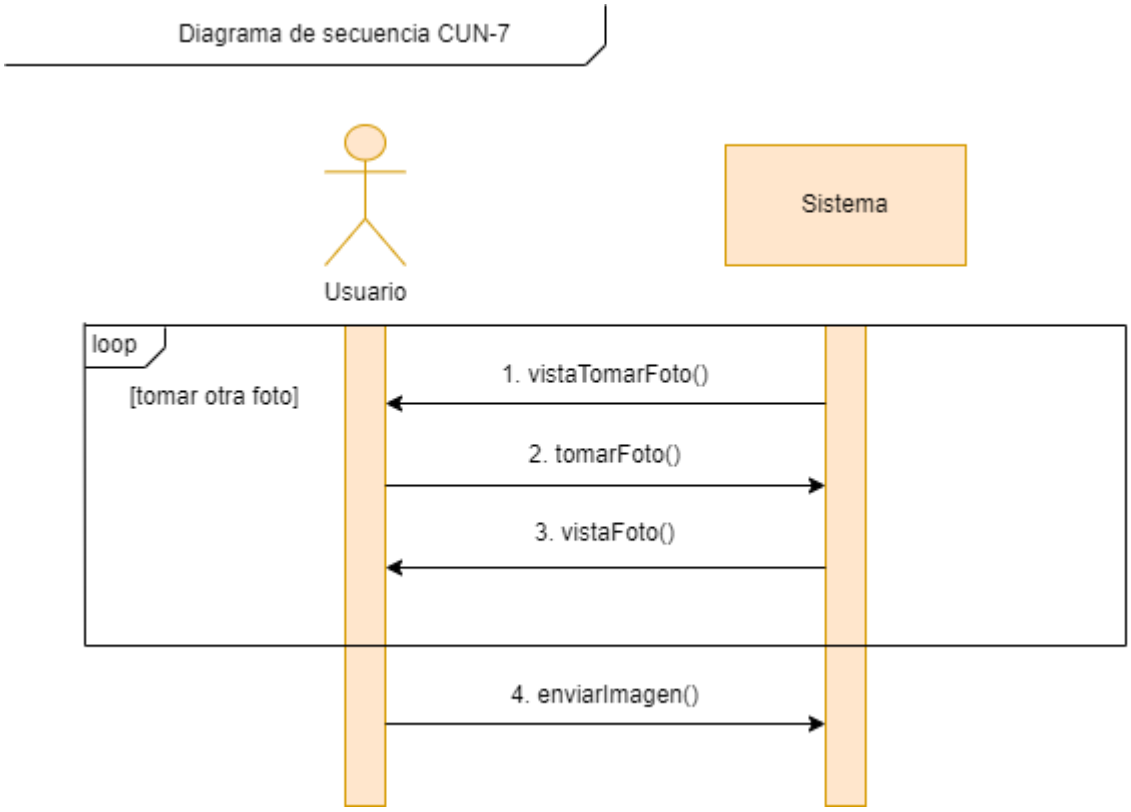


Ilustración 35: Diagrama de secuencia de código de usuario normal 7

Seguidamente, se encuentra el último diagrama (ilustración 36) de secuencia referente a las distintas opciones de introducir imágenes. Es un poco peculiar, pues parece un video, aunque realmente son imágenes que se están entrenando sin parar. Al entrar, aparecerá una vista con la imagen captada por la cámara (entrenada por el modelo) y el usuario podrá poner a prueba el modelo interactuando, moviéndose en el caso de detección de pose, por ejemplo.

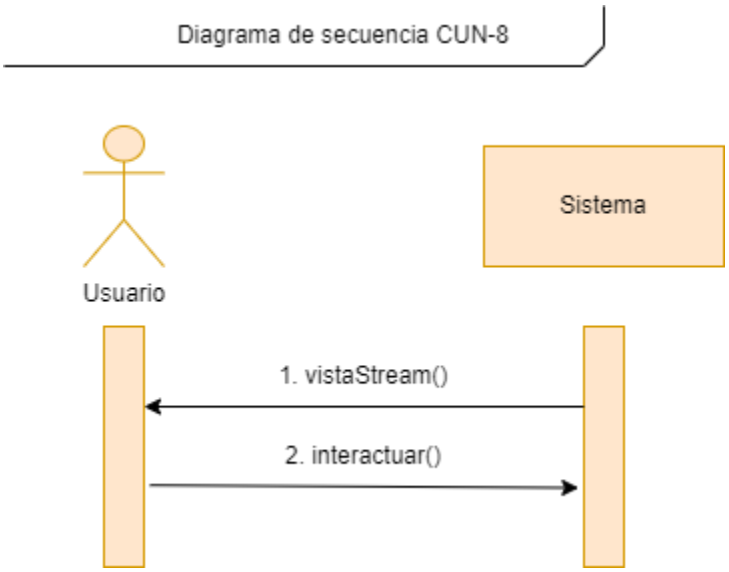


Ilustración 36: Diagrama de secuencia de código de usuario normal 8

La siguiente interacción es para volver a empezar, ya que una vez se ha llegado al final se clicará sobre un botón que volverá al inicio (ilustración 37).

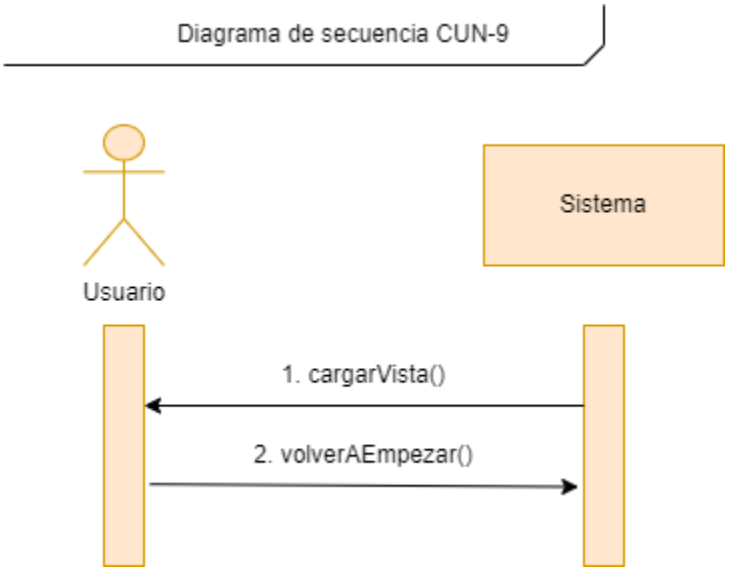


Ilustración 37: Diagrama de secuencia de código de usuario normal 9

Tanto la interacción anterior como esta, están hechas para mejorar la fluidez, y es que en la mayoría de las vistas aparece un botón para volver atrás, con el objetivo

de que el usuario no tenga que llegar hasta el final en el caso de que cambie de opinión en lo que ha elegido o prefiera usar algo distinto (ilustración 38).

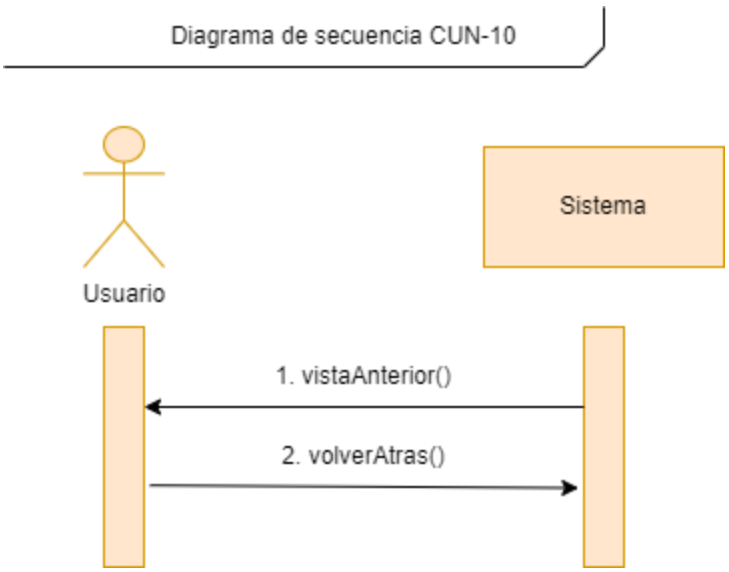


Ilustración 38: Diagrama de secuencia de código de usuario normal 10

Para acabar, la última interacción de un usuario normal, es un botón que aparece en todas las vistas, y es el que te lleva a una interfaz con un poco de información sobre los profesores que tutelan el proyecto y sobre el desarrollador (ilustración 39).

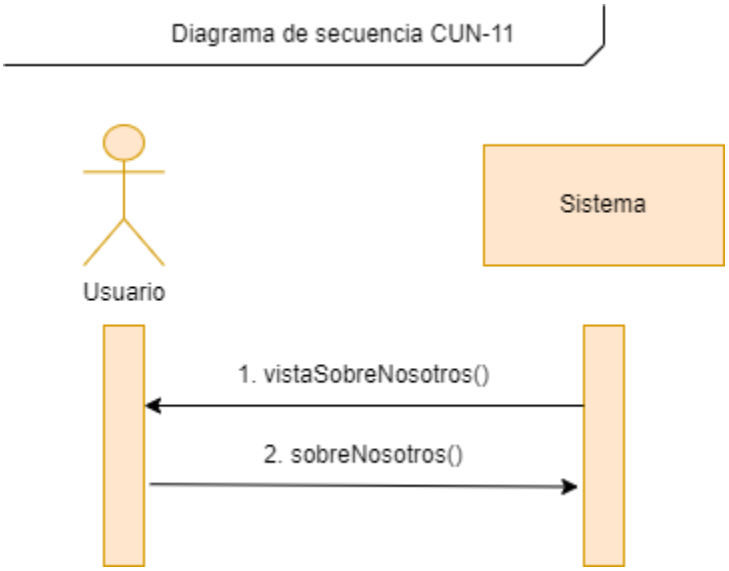


Ilustración 39: Diagrama de secuencia de código de usuario normal 11

4.1.2. Diagrama de secuencia de usuario administrador

Una vez vistas todas las interacciones de un usuario normal, se van a ver ahora los diagramas de secuencia de un usuario administrador.

La primera interacción, es un login para verificar que la persona que se dispone a entrar es un administrador ya que necesita una contraseña y un usuario (ilustración 40).

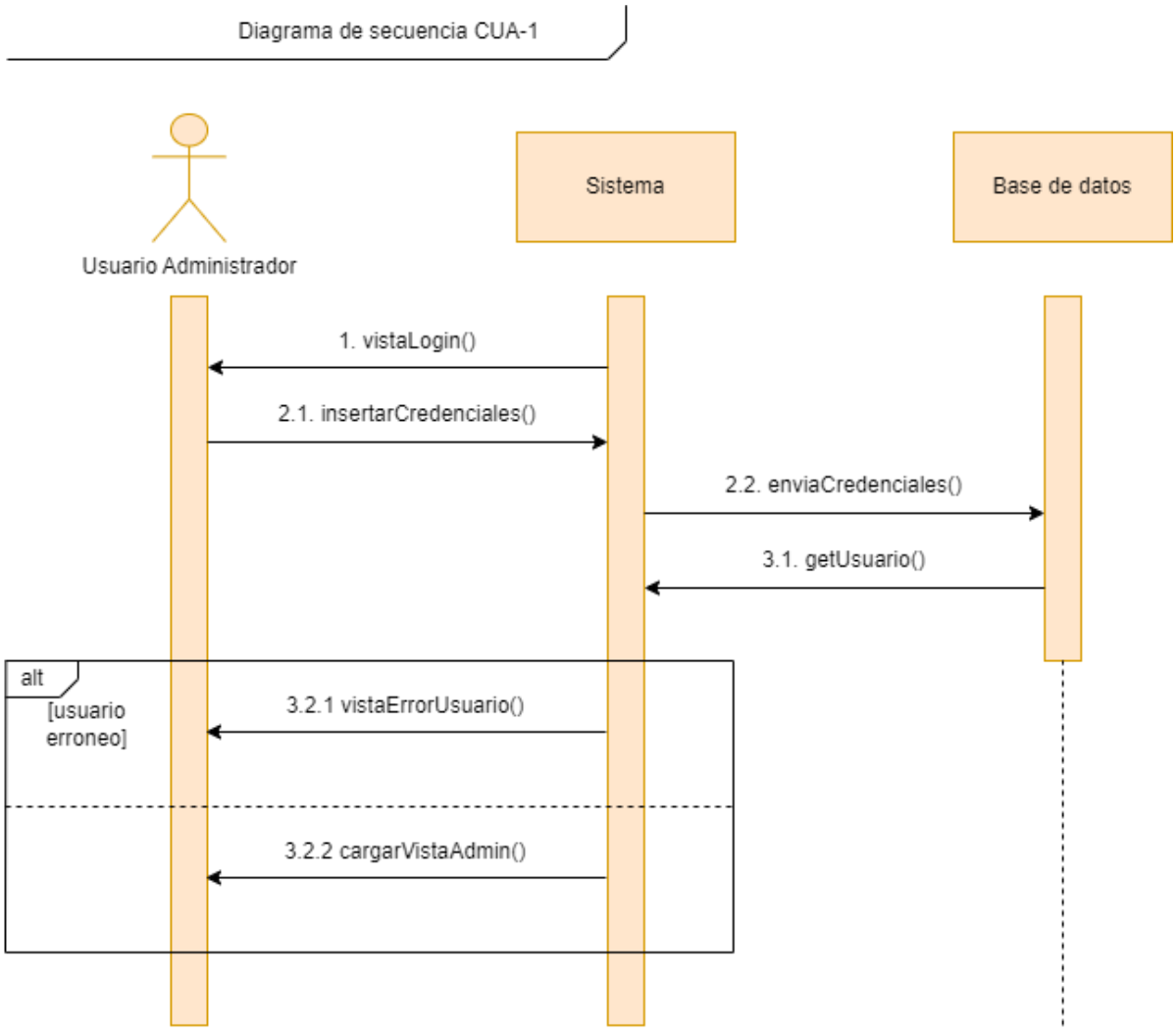


Ilustración 40: Diagrama de secuencia de código de usuario administrador 1

Por otro lado, tenemos el diagrama de usuario para logout, que con un solo click permite al usuario administrador salir de la vista de administrador (ilustración 41).

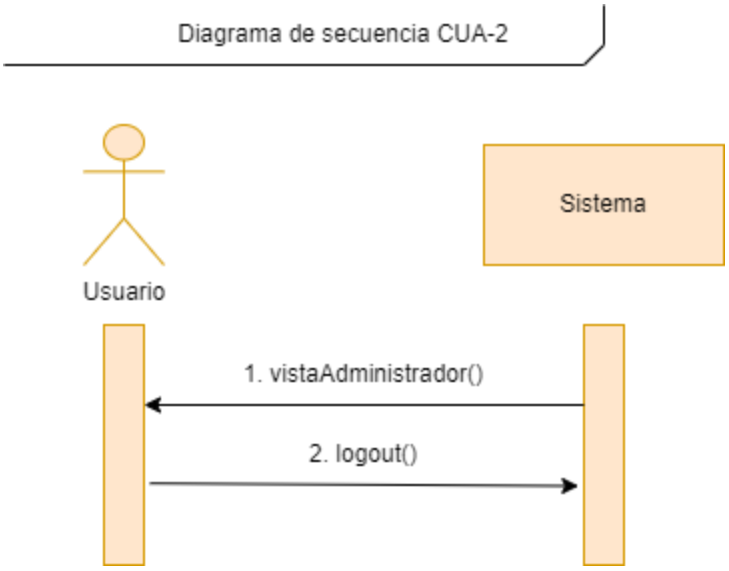


Ilustración 41: Diagrama de secuencia de código de usuario administrador 2

A continuación, se van a mostrar los diagramas de usuario para los distintos tipos de objetos que tenemos en la aplicación (categorías, database y modelo) y además se mencionarán las tres acciones que podemos realizar para cada una de ellas (crear, editar y borrar) (ilustración 42-50).

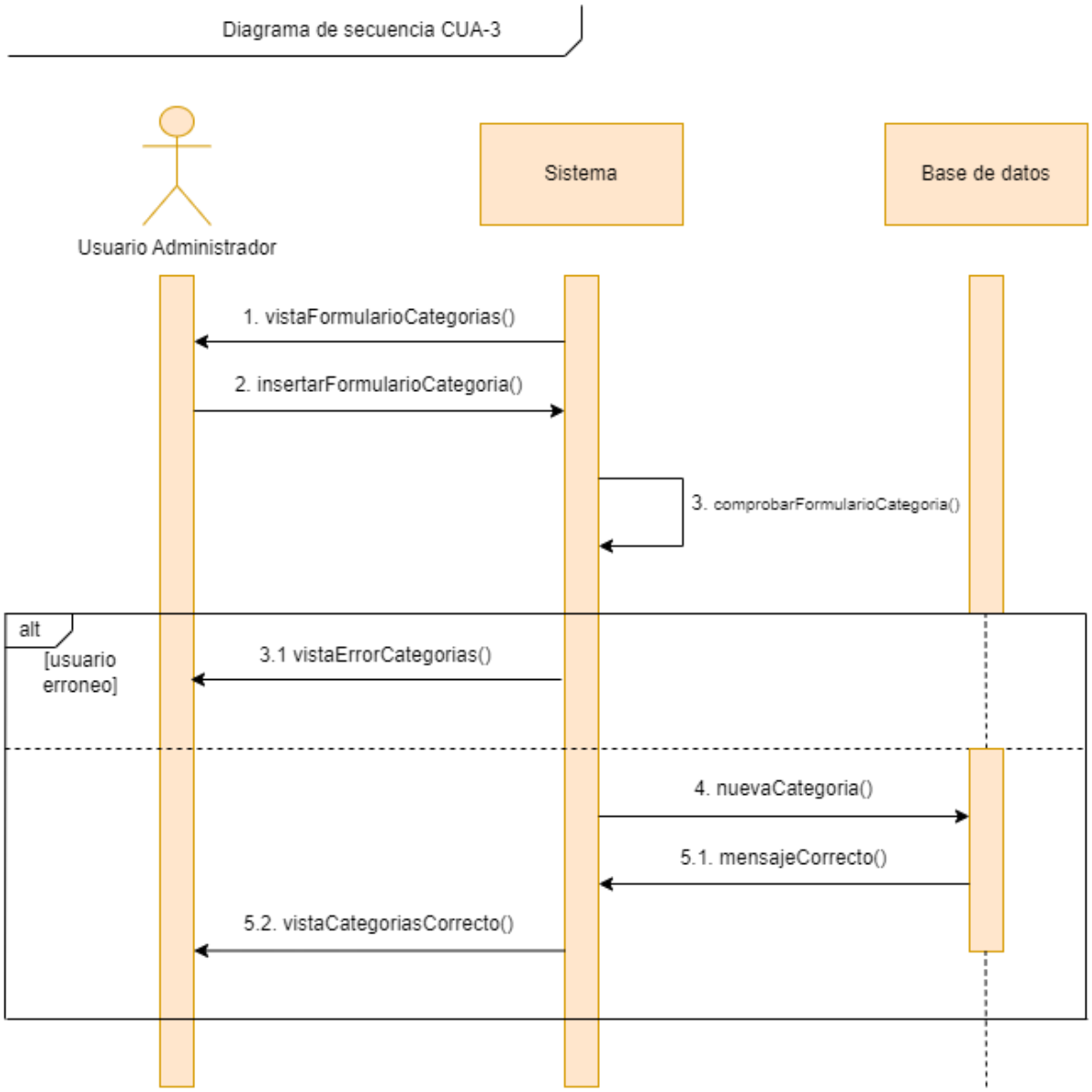


Ilustración 42: Diagrama de secuencia de código de usuario administrador 3

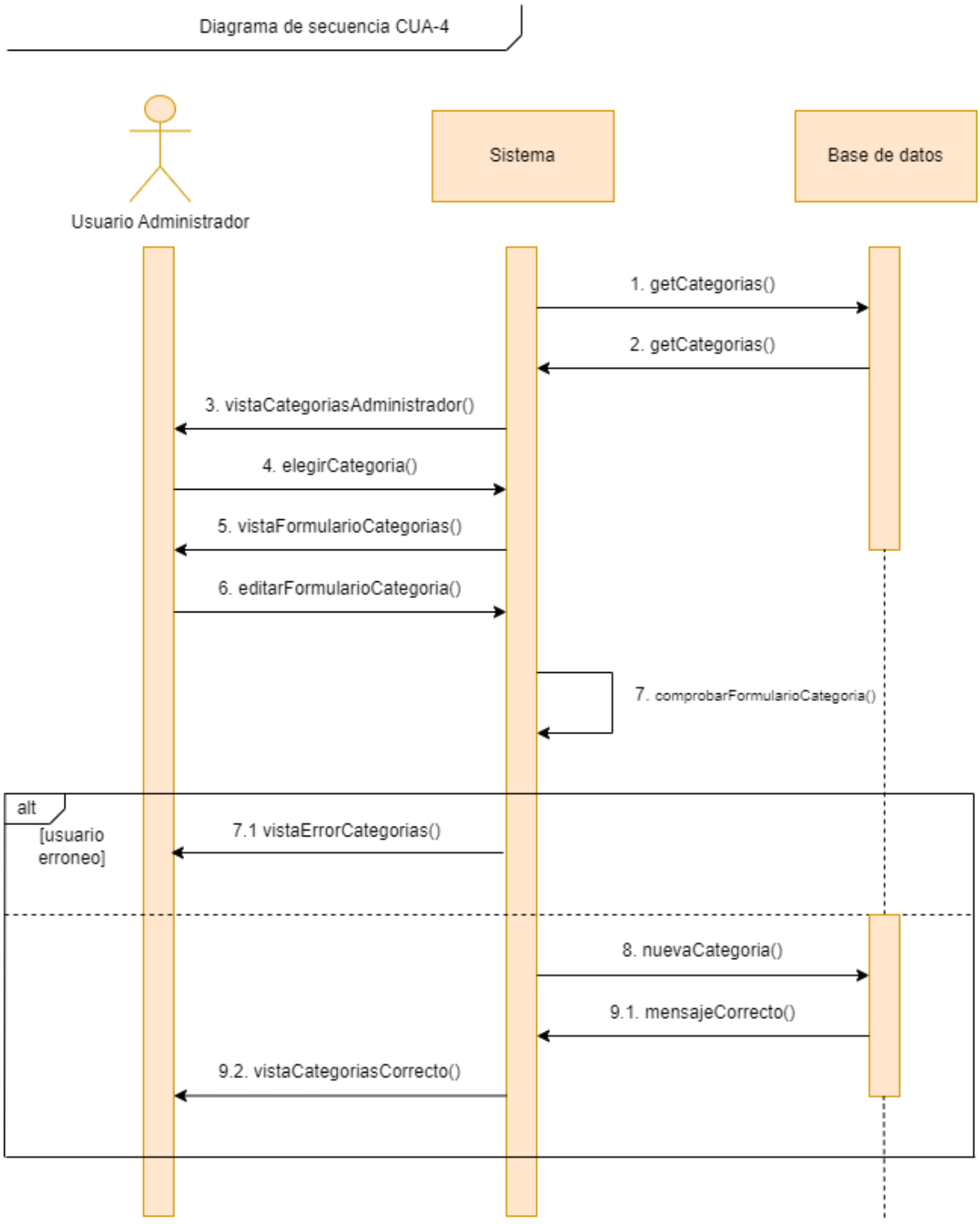


Ilustración 43: Diagrama de secuencia de código de usuario administrador 4

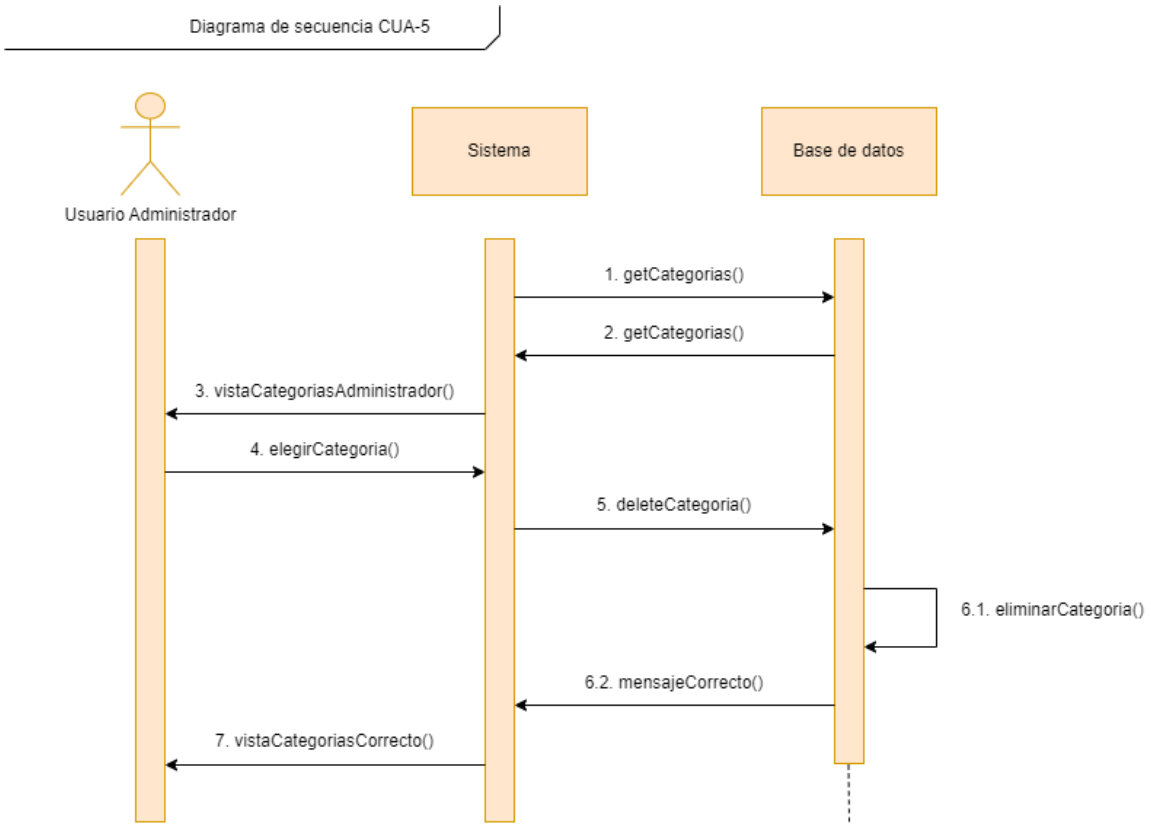


Ilustración 44: Diagrama de secuencia de código de usuario administrador 5

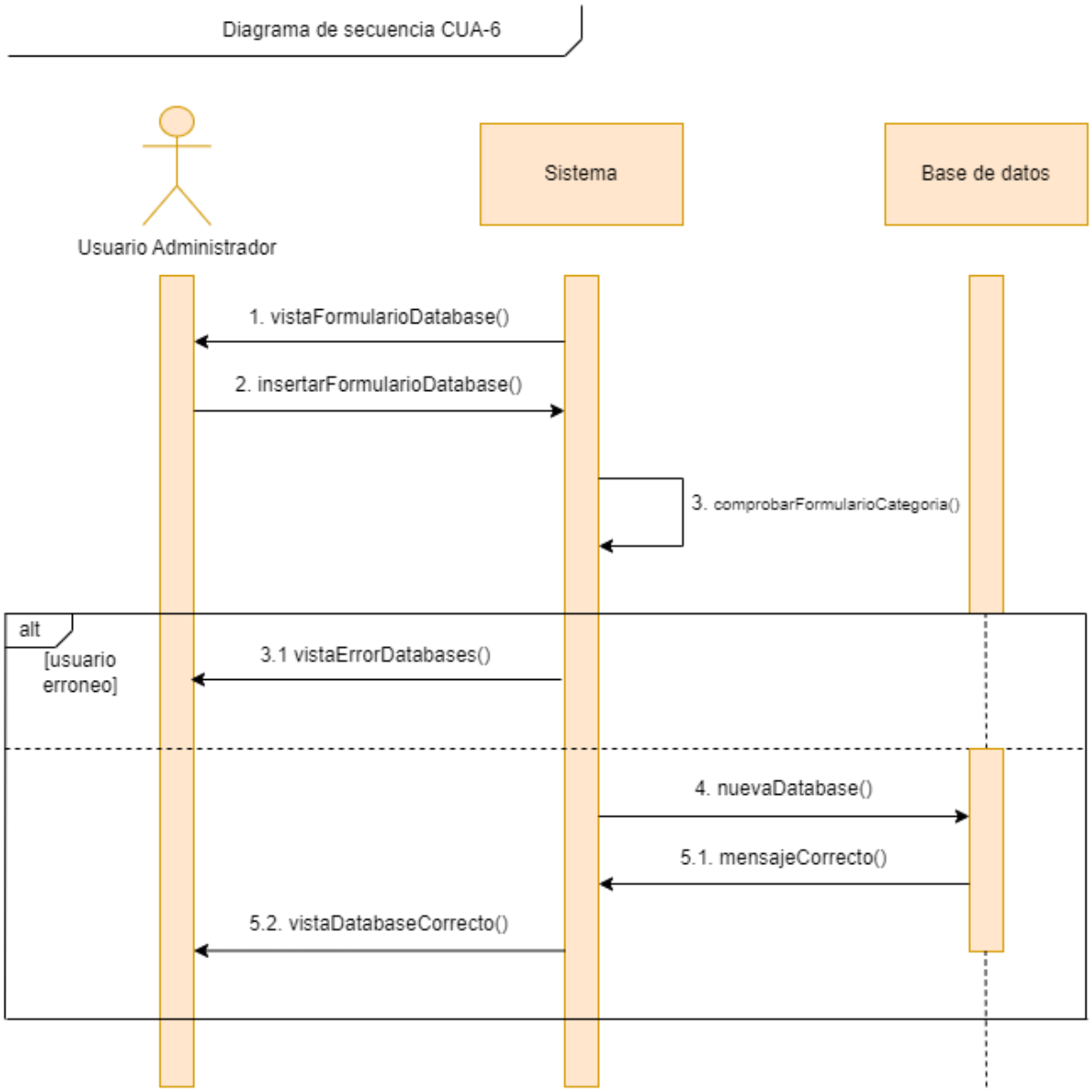


Ilustración 45: Diagrama de secuencia de código de usuario administrador 6

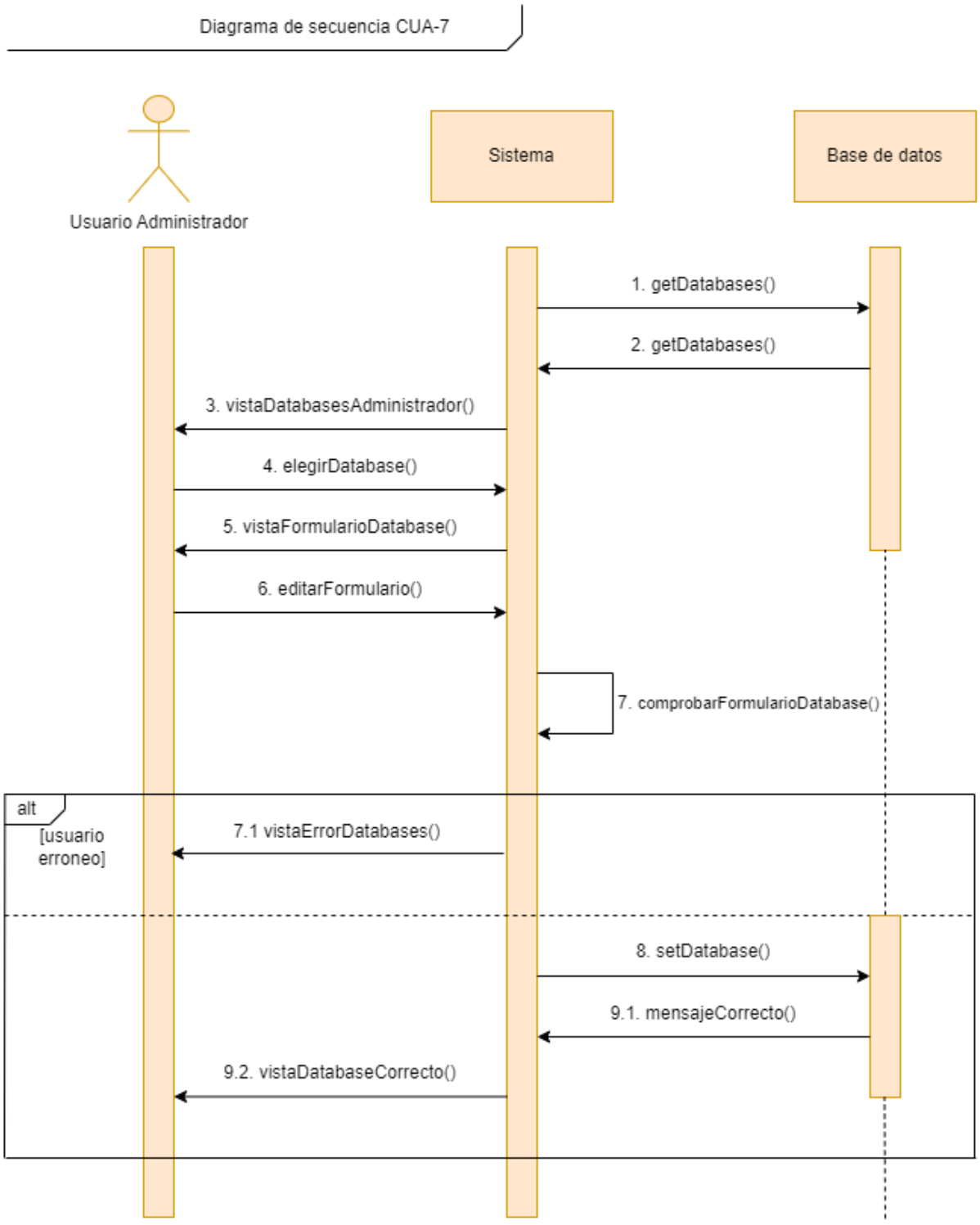


Ilustración 46: Diagrama de secuencia de código de usuario administrador 7

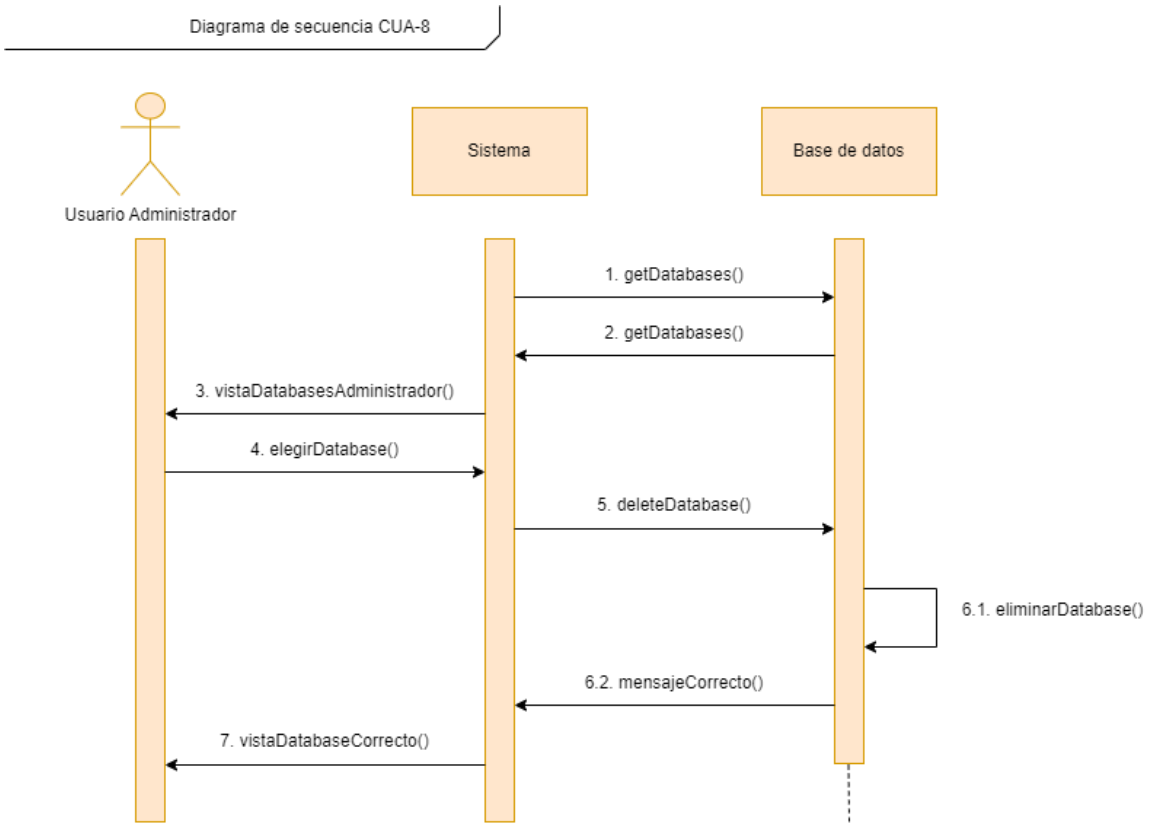


Ilustración 47: Diagrama de secuencia de código de usuario administrador 8

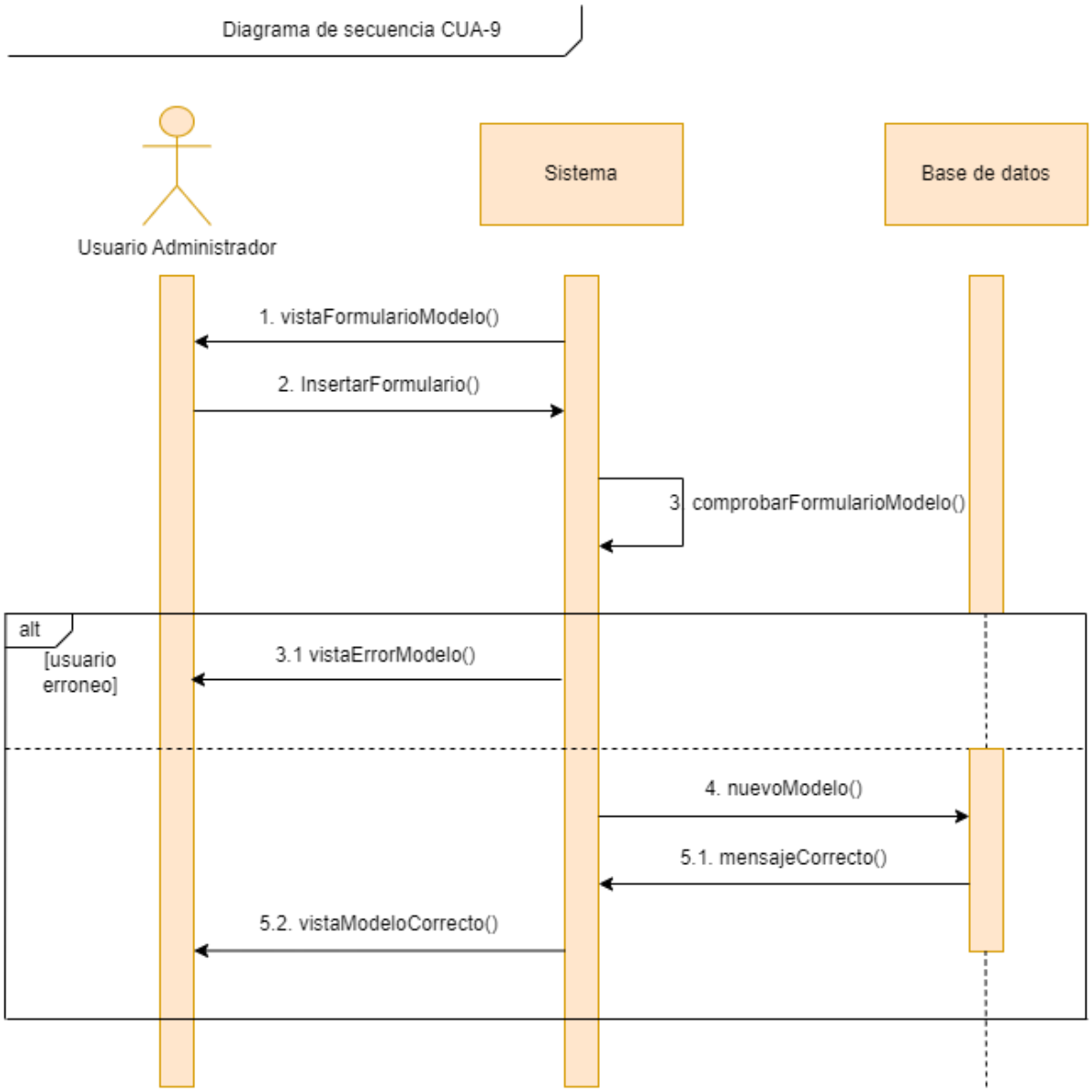


Ilustración 48: Diagrama de secuencia de código de usuario administrador 9

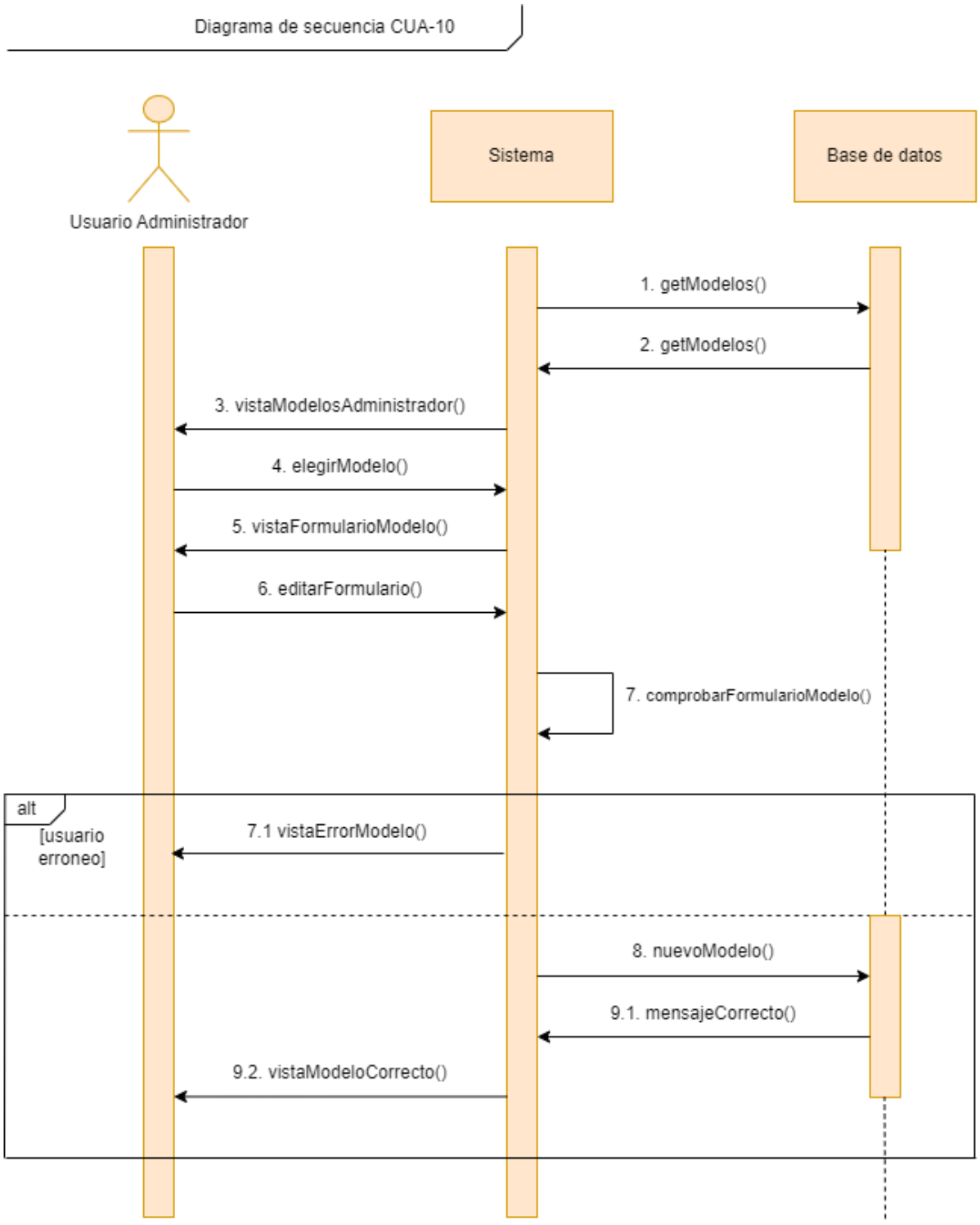


Ilustración 49: Diagrama de secuencia de código de usuario administrador 10

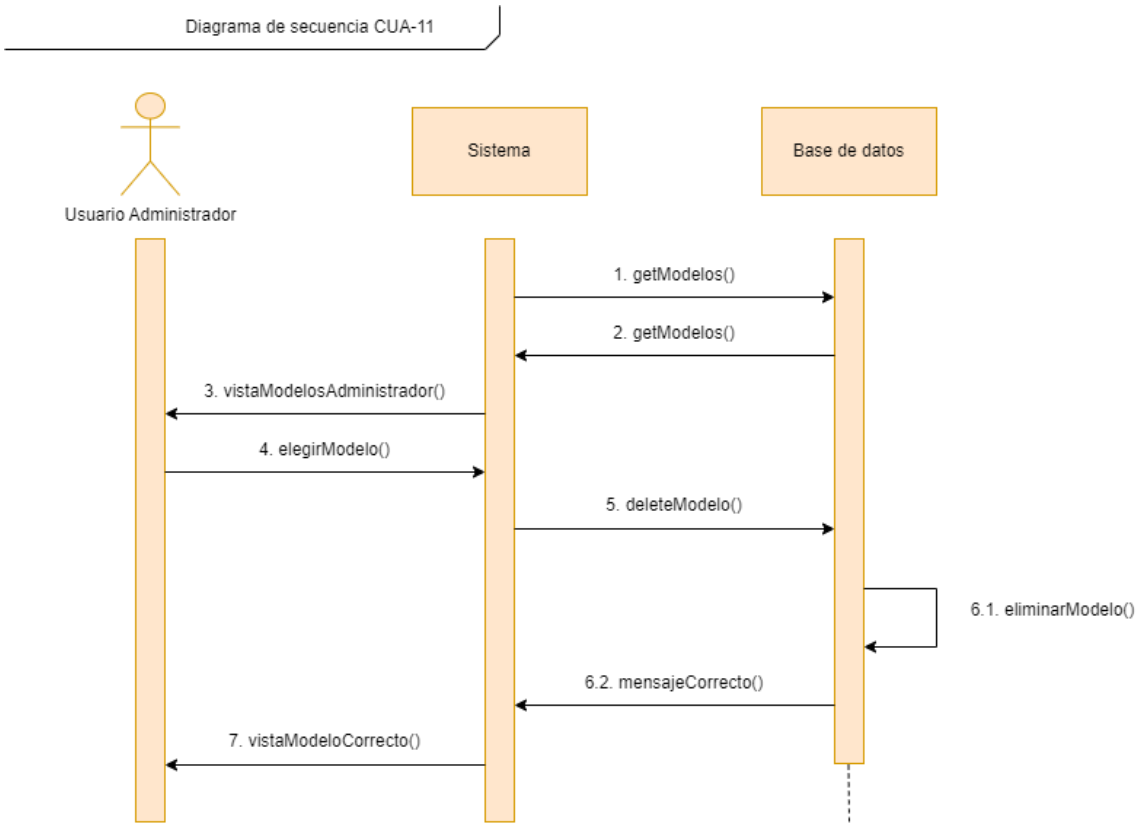


Ilustración 50: Diagrama de secuencia de código de usuario administrador 11

4.2. Diagrama de clases

A continuación, vamos a ver el diagrama de clases que es un diagrama basado en el modelo de programación orientado a objetos, ya que define las clases que se utilizarán cuando se pase a la fase de construcción y la manera en que se relacionan las mismas.

Podemos ver el diagrama de clases en la ilustración 51 y los objetos que destacan son, el sistema y la base de datos. El sistema es el objeto que crea las interfaces, recibe todas las interacciones del usuario y pide a la base de datos los objetos que necesita. Mientras que la base de datos es donde se almacenan todos los objetos de categoría, el modelo y database o base de datos de un modelo. Estos últimos los veremos con más detalle en el apartado 4.3.1.

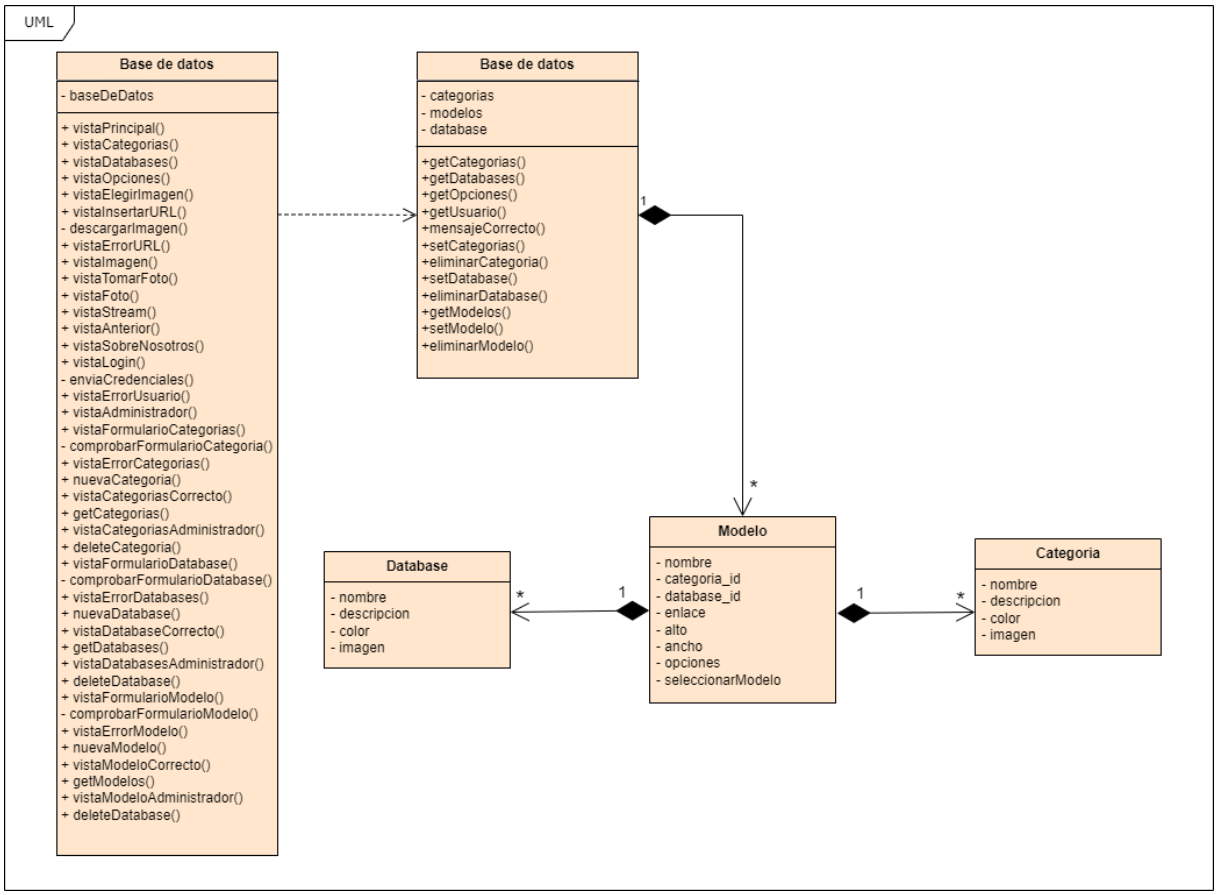


Ilustración 51: Diagrama de clases

4.3. Base de datos y elementos de la aplicación

La idea principal que se ha mantenido firme a lo largo del desarrollo de la aplicación es que a esta se le puedan añadir varios modelos capaces de tratar imágenes, independientemente de la problemática, la arquitectura, las capas internas del modelo, de cómo esté entrenado, del dataset con el que esté entrenado, etc. Para que sea independiente se han de fijar los elementos constantes en la aplicación y además establecer los que irán cambiando. Cuantos menos elementos variables haya, más fácil será llegar a esa idea idílica de independencia.

La base de datos es uno de los elementos más importantes de la aplicación, todo depende de ella y de cómo esté diseñada. La base de datos diseñada es muy simple, pero es complicado detectar en los modelos de deep learning qué elementos varían y cuales son siempre fijos. Probando y desarrollando distintos modelos se ha

conseguido descubrir cómo se podrían ejecutar varios modelos cambiando solo algunos elementos.

Dejando atrás las abstracciones, se ha conseguido diseñar una aplicación a la cual se le pueden añadir distintos modelos, haciendo que esta sea mucho más moldeable. Por ejemplo, en un ordenador promedio, la ejecución de los elementos podría tardar bastante tiempo en ejecutar. Gracias a esta facilidad de cambiar modelos, se puede cambiar uno por otro que esté entrenado con imágenes más pequeñas, (un ancho y alto más pequeño) y obtener resultados quizá no tan buenos pero si en un tiempo aceptable para un usuario.

A continuación, se va a definir el diagrama de la base de datos y se explicarán las clases que la forman junto con sus atributos. Los atributos se podrían subdividir en dos tipos; los atributos que hacen que los modelos funcionen correctamente y los que hacen que la interfaz sea más bonita visualmente y que permiten que sea mucho más fácil añadir nuevos elementos a la interfaz.

4.3.1. Diagrama de la base de datos

En este apartado vamos a ver el Diagrama Entidad-Relación en la ilustración 51 de la base de datos de DL4Vision y posteriormente se procederá a explicar cada una de las clases.

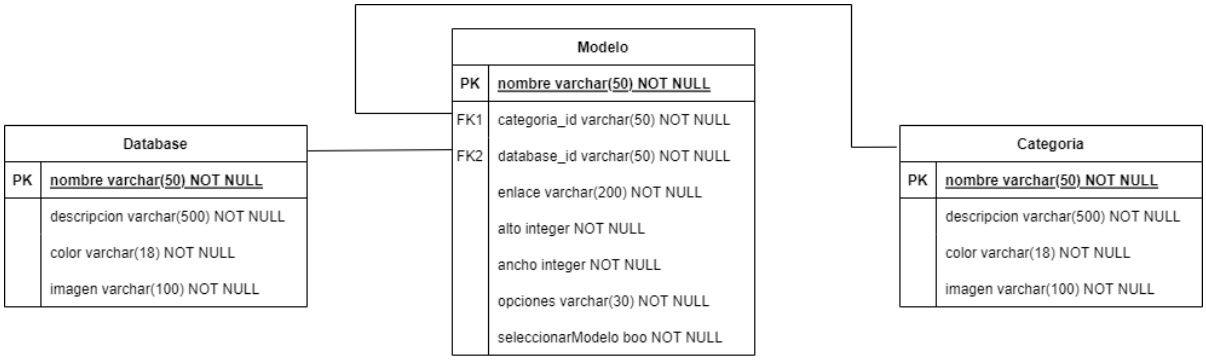


Ilustración 52: Diagrama Entidad-Relación de DL4Vision

- Categoría: como ya se ha comentado, en TensorFlow Hub se dividen los modelos en dominios de problemas dependiendo de la entrada. En este

caso, la aplicación trabaja con el dominio de las imágenes, dentro de este dominio hay varias subcategorías como por ejemplo clasificación de imágenes o detección de objetos. La clase categoría representa a esta subcategoría dentro del dominio de problema de imágenes. Los atributos son para hacer la aplicación más atractiva. En su interior se encuentra el nombre, una pequeña descripción dónde se puede comentar en qué consiste esta categoría, un color para diferenciarlo del resto y una imagen elegida por el administrador para ayudar visualmente al usuario a identificar lo que se obtendrá al final de la ejecución.

- Conjunto de datos: también recibe el nombre de base de datos o dataset. Es un conjunto de datos, en este caso, los datasets son conjuntos de imágenes con las cuáles los modelos han sido entrenados. Los atributos son iguales que los de la categoría.

- Modelo: es la clase principal de la aplicación, el modelo de deep learning, contiene la información necesaria para poder ejecutarlo. Por un lado, se encuentran los atributos necesarios para la aplicación como el nombre del modelo o las opciones (distintas maneras de insertar imágenes a la aplicación, ya sea por enlace, una foto…). Por otro lado, en los atributos que son utilizados para la lógica de los modelos y de la aplicación, tenemos la categoria_id y la database_id, que se relacionan con la Categoría y con la Base de Datos respectivamente. SeleccionarModelo es un booleano que indica cuál de los modelos está activo por defecto, ya que puede haber varios modelos con la misma base de datos y categoría, pero solo uno de ellos estará marcado por defecto para que la aplicación lo use. El resto de atributos son para que el modelo funcione correctamente, y se verán posteriormente.

4.3.2. Valores de las clases implementadas

A continuación, se van a ver qué elementos incluye la aplicación. Esta por defecto, tiene varios elementos fijos para poder probar los distintos modelos. Con esto

quiere decirse que se pueden añadir más elementos (categorías, bases de datos o modelos) además de los que ya trae, aunque esto conlleva algunas restricciones que veremos después.

4.3.2.1. Categorías

Dentro del dominio de imágenes, se han implementado 4 subtipos a las que se le ha denominado categorías.

- Clasificación de Imágenes: en esta categoría los modelos están entrenados para que al proporcionarle una imagen de entrada intente adivinar qué hay en esta. Devuelve una predicción, por ejemplo, el modelo que trae por defecto la aplicación está enfocado sobre todo a animales, es decir, al pasarle una imagen de un perro puede incluso predecir la raza.
- Detección de objetos: esta categoría está entrenada para detectar objetos, devuelve una predicción y además muestra un porcentaje de probabilidad con la que está seguro de que su predicción es correcta. Esta categoría se diferencia principalmente de la anterior en que ofrece varias predicciones y las pinta en la imagen.
- Detección de pose: esta categoría de modelos detecta en las personas los puntos más representativos de la figura humana y es capaz de identificar la postura humana.
- Segmentación: la segmentación de imágenes muestra, a partir una imagen, los distintos trozos o partes que contiene basándose en las características de los píxeles de la imagen.

4.3.2.2. Datasets

Seguidamente, se comentarán los datasets. De estos se sacan los datos necesarios de los distintos modelos para entrenarse. Incluso algunos modelos usan una parte, para entrenar y otra parte mucho más pequeña, para evaluar y saber cómo

de bueno es su modelo. En este caso, al estar trabajando con imágenes, los datasets utilizados son grandes cantidades de imágenes. En el mundo del deep learning se pueden encontrar algunos datasets bastante conocidos, y estos suelen ser los que usan los modelos de TensorFlow Hub. Es por esto que se van a ver los principales datasets que utiliza la aplicación.

- **COCO**: es un dataset diseñado por Microsoft y significa Common Objects in Context. Es un conjunto de datos de imágenes que se creó con el objetivo de avanzar en el reconocimiento de imágenes. El conjunto de datos de COCO (ilustración 52) contiene una gran cantidad de datos visuales desafiantes y de alta calidad para la visión por computadora (en su mayoría, redes neuronales de última generación). En la aplicación, los modelos de segmentación y de detección de pose han sido entrenados con este dataset[12].

Ilustración 53: Logo COCO. Fuente.

- **ImagiNet**: En la actualidad, ImageNet (ilustración 53) es el conjunto de datos por excelencia para evaluar algoritmos de clasificación, localización y reconocimiento de imágenes. ImageNet, más que un conjunto de datos, es un proyecto a gran escala que involucra a diversas instituciones educativas, como Stanford y Princeton. Su objetivo es ser un banco de datos visual enorme para la investigación y desarrollo de software especializado en reconocimiento de imágenes. Aunque en el modelo que se usa, realmente no se utiliza la base de datos entera, sino una pequeña

parte llamada **ILSVRC** (ImageNet Large Scale Visual Recognition Challenge)[13].

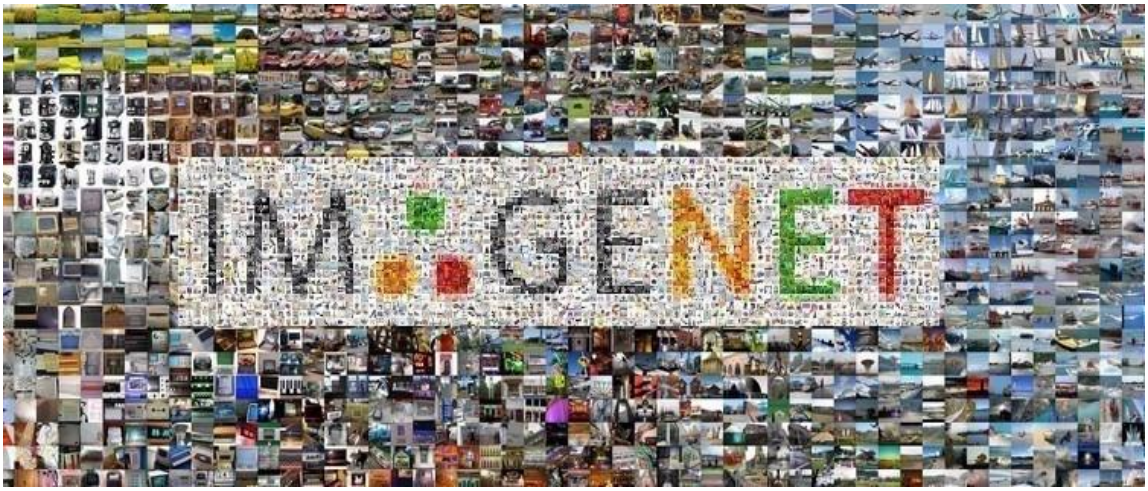


Ilustración 54: Logo Imagenet. Fuente.

- **ADE20K**: El conjunto de datos de segmentación semántica ADE20K (ilustración 54) contiene más de 20.000 imágenes centradas en escenas anotadas exhaustivamente con objetos a nivel de píxel y etiquetas de partes de objetos. Hay un total de 150 categorías semánticas, que incluyen cosas como el cielo, un camino, la hierba y objetos discretos como una persona, un automóvil, una cama, etc[14].

Ilustración 55: Imagen resultante de un modelo de detección de objetos entrenado con ADE20K. **Fuente**.

4.4. Arquitectura de la aplicación

Una vez se conoce el framework y todo lo que ha de tener la aplicación, hay que saber cómo se va a estructurar el código, es decir, como se van a relacionar las distintas clases y métodos entre ellas. Al usar un framework con distintos lenguajes de programación hay que tenerlo en cuenta y saber cómo se va organizar el código. Por tanto, se procederá a hablar sobre la arquitectura de la aplicación.

Django sigue el patrón MVC y las separa de la siguiente manera:

- M, representa a la parte de la base de datos, está controlada por la capa de la base de datos de Django.
- V, selecciona qué datos mostrar y como se van a representar, es manejada por las plantillas y las vistas.
- C, muestra la vista correspondiente dependiendo de la entrada del usuario, esto se encarga el framework.

Ya que "C" es controlada por el mismo framework y lo más importante ocurre en las plantillas, las vistas y los modelos, Django es conocido como un Framework MTV (ilustración 55):

- M de "Model" (Modelo), la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos.
- T de "Template" (Plantilla), la capa de presentación. Esta capa contiene las decisiones relacionadas con la presentación: como las cosas son mostradas sobre una página web u otro tipo de documento.
- V de "View" (Vista), la capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla correspondiente: puede verse como un puente entre los modelos y las plantillas.

Podría decirse que las vistas de Django pueden ser el "controlador" y las plantillas, la "vista".

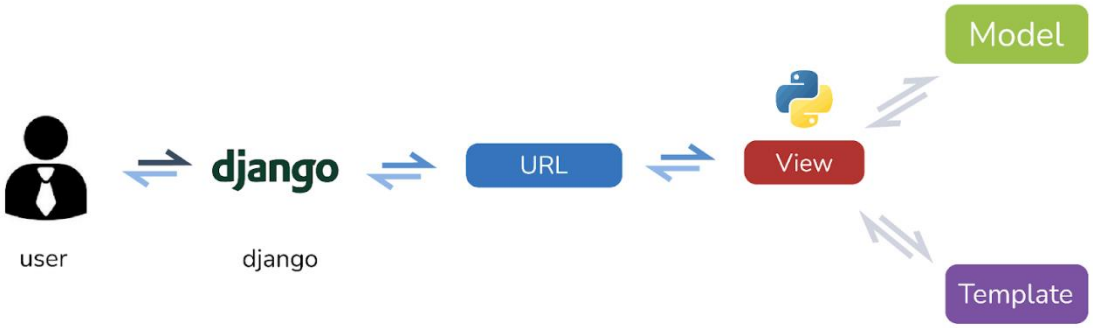


Ilustración 56: Imagen de arquitectura de Django. Fuente.

5. DESARROLLO

Esta sección va a abordar todo el proceso de desarrollo de software para la realización del TFG.

La estrategia de desarrollo que se ha llevado a cabo es un modelo incremental, como ya hemos visto en la sección anterior (apartado 2.4.3.). El proceso de desarrollo se dividirá en iteraciones cuyo resultado es un incremento, la sucesión de las iteraciones nos dará el producto final.

Una vez vista toda la parte de diseño, sabemos lo que tiene que hacer la aplicación y cómo se ha ido mejorando a lo largo del tiempo, se va proceder a comentar los entresijos de la misma, explicar cómo funciona esta herramienta, y responder a la pregunta de ¿Qué está ocurriendo por dentro? Obviamente, al final todo es código, pero no todo tiene la misma función, por tanto, se hablará de la funcionalidad del mismo. Para ello, se va a diferenciar entre la parte visual o también llamada front y la parte no visible a la que se le llamará back.

5.1. Descripción de la solución propuesta

La solución propuesta es desarrollar una plataforma para la aplicación de métodos de Deep Learning donde el usuario pueda utilizar modelos de distintas categorías relacionadas con la visión por computador. Además, se podran añadir más modelos de los que ya vienen por defecto.

5.2. Evolución del proyecto

Este proyecto se ha ido desarrollando a lo largo de varios meses, la idea original era implementar una aplicación interactiva en la que cualquier usuario pudiera usar un modelo de visión por computador. Tras el primer incremento conseguimos tener una aplicación con la que interactuar y con la que los usuarios pueden entrenar. A medida que ha ido pasando el tiempo y los incrementos, se han ido añadiendo en cada interacción nuevas opciones para que el usuario pueda introducir imágenes a la

aplicación. Se ha mejorado la portabilidad, la seguridad e incluso se han añadido más modelos de distintas categorias.

5.3. Front

En este apartado, sobre todo, se va a comentar cómo funciona Django y todas las ventajas que proporciona la propia API de Django. Esta hace que el desarrollo de la aplicación sea mucho más cómodo, ya que ahorra muchos elementos a la hora de programar porque ya los incluyen. Cabe destacar los siguientes elementos, ya que han sido muy útiles.

- La propia API contiene unas vistas de administrador por defecto, de forma que solo ha hecho falta insertar la base de datos en código en un archivo específico. Esto nos proporciona una base de datos y una interfaz para poder interaccionar con ella.
- La aplicación sabe qué modelo usar, ya que para cada categoría/database solo permite que tenga asignada uno por defecto, la propia vista de administrador no lo permite. El usuario elige la categoría y el database a través de las distintas vistas a medida que avanza por la aplicación. Toda esta información viaja a través de métodos http GET o POST como en cualquier página web. Por último, se elige una opción con la que se va a introducir la imagen a entrenar y la devuelve. En la siguiente imagen se observa el flujo de toda la aplicación con todas las vistas y todas las posibilidades para el usuario (ilustración 56).

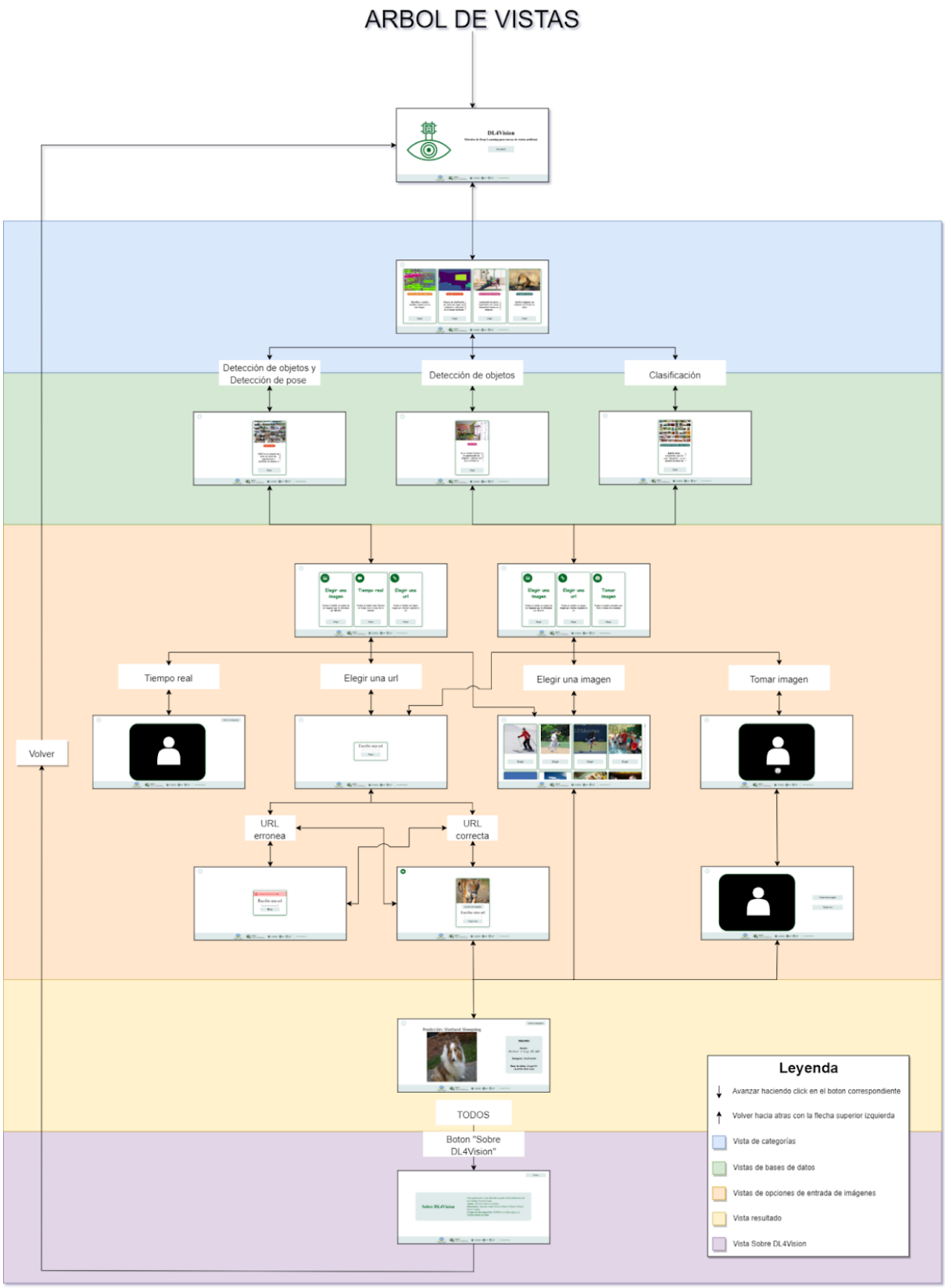


Ilustración 57: Imagen del flujo de todas las vistas

- Uno de los elementos que más útil ha sido la cantidad de opciones que te aporta en el front en lenguaje Python, para un desarrollador que no está especializado en lenguajes de programación de front. Por ejemplo, se ha hecho una aplicación que llame a la base de datos y muestre los elementos en tarjetas, independientemente del valor de la clase en la base de datos. En otras API, hubiera hecho falta aprender JavaScript por ejemplo, sin embargo, con esta API trae elementos de código que podemos usar con lenguaje Python dentro de HTML. Esto nos permite enviar al front variables, simplemente con poner el nombre de la variable que hemos enviado desde el back y se sustituye en el front, es decir, si enviamos desde el back una variable llamada “categoría” con el valor “Clasificación de textos” te la sustituye en el front con este valor. Además, te permite hacer condicionales (se utiliza si hay un error imprimo un mensaje) o bucles (estos se usan partir para que se vean todas las tarjetas de categorías y de database independientemente de la cantidad, si se ha añadido alguna, si se elimina alguna, etc. en cuanto se volviese a ver la vista, ya se podría ver actualizada).
- Permite usar plantillas haciéndolo muy útil ya que no es necesario reutilizar código.

5.4. Back.

Una vez se ha explicado la parte más visual de la aplicación, se va a explicar la parte que no se ve, o dicho de otra forma, el cerebro de la misma.

5.4.1. Cargar modelos

Para empezar, se ha considerado que es importante comentar lo primero que nos encontramos cuando ejecutamos la aplicación. Esta tarda en ejecutar algunos minutos y durante este tiempo la aplicación se dedica a cargar los modelos sin necesidad de tener el código de estos dentro. Y puede surgir la pregunta de querer saber cómo consigue cargar los modelos sin tenerlos programados dentro del código.

Esto es posible gracias al atributo enlace de la clase modelo que permite cargar el modelo con la librería Keras o TensorFlow, dependiendo de cómo esté programado.

Al observar los códigos de ejemplo o los notebooks de prueba, se aprecia que normalmente la visión de TensorFlow Hub usa modelos que siguen el siguiente flujo (ilustración 57):

1. Elegir la imagen que se desea pasar al modelo.
2. Se carga el modelo a través de un método, con el enlace del modelo de TensorFlow Hub.
3. Llama al método que predice la imagen pasándole los parámetros de entrada que necesite, normalmente es la imagen, el alto y el ancho.
4. Recoge la predicción y la muestra.
5. Si se desea volver a pasar otra imagen, regresar al paso 1.

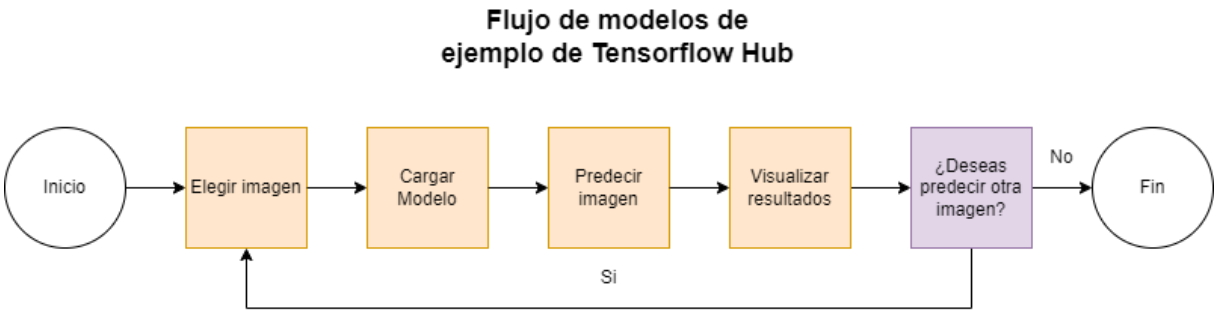


Ilustración 58: Imagen del flujo de los modelos de ejemplo de TensorFlow Hub

Este flujo para realizar algunas pruebas en un momento concreto está bien, pero para una aplicación de usuario no es el mejor. Esto se debe a que el modelo tarda un tiempo en cargarse y cada vez que queramos predecir una imagen la aplicación tardará un tiempo determinado que hará que esta sea lenta y poco fluida. Por tanto se llegó a la conclusión de que lo mejor era cargar todos los modelos marcados por defecto al iniciar la aplicación. De este modo, los modelos estarán en memoria, y el tiempo de espera para el usuario será el que tarde la aplicación en predecir la imagen. La diferencia de tiempo es bastante grande ya que el tiempo que tarda en cargar la

imagen es muy pequeño en comparación con el que tarda en cargar el modelo. Por tanto, el flujo quedaría (ilustración 58):

1. Se carga el modelo a través de un método con el enlace del modelo de TensorFlow Hub.
2. Se elige la imagen que se desea pasar al modelo.
3. Llama al método que predice la imagen pasándole los parámetros de entrada que necesite, normalmente es la imagen, el alto y el ancho.
4. Recoge la predicción y la muestra.
5. Si se desea volver a pasar otra imagen volvemos al paso 2.

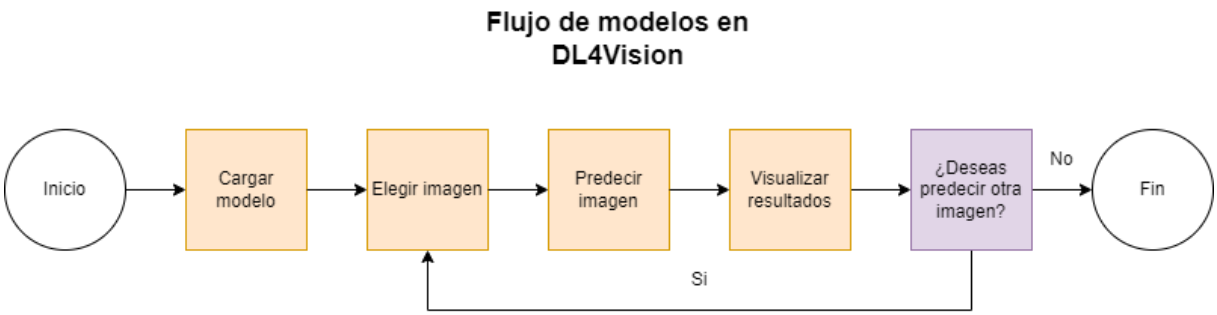


Ilustración 59: Imagen del flujo de los modelos de DL4Vision

5.4.2. Flujo de las vistas

Además de lo que ya se ha comentado del front de Django, hay que mencionar varios elementos. Django controla prácticamente todo, ya que por defecto trae algunos archivos para gestionar la seguridad, las urls, las bases de datos, las vistas, etc.

Básicamente el flujo de cómo se crea una vista y se muestra así (ilustración 59):

1. Primero hay un archivo que relaciona la dirección url con su vista correspondiente, el usuario entra en la url de la aplicación y en este archivo esta busca la vista correspondiente. Por ejemplo, el usuario al introducir la url “/home” la aplicación sabe que tiene que mostrarle al

usuario la vista llamada home gracias a que en este archivo están relacionadas.

2. La aplicación lee el código del método correspondiente, donde la última línea indicará siempre hacia qué plantilla quiere dirigirse y qué datos quiere enviar. Por ejemplo, podemos llamar a la base de datos para que nos devuelva todas las categorías y al llamar a la plantilla, le pasa esa información también.

3. La aplicación va a la plantilla y la muestra.

4. El usuario inserta un nuevo enlace a través de las distintas interacciones de la vista (por ejemplo, un botón).

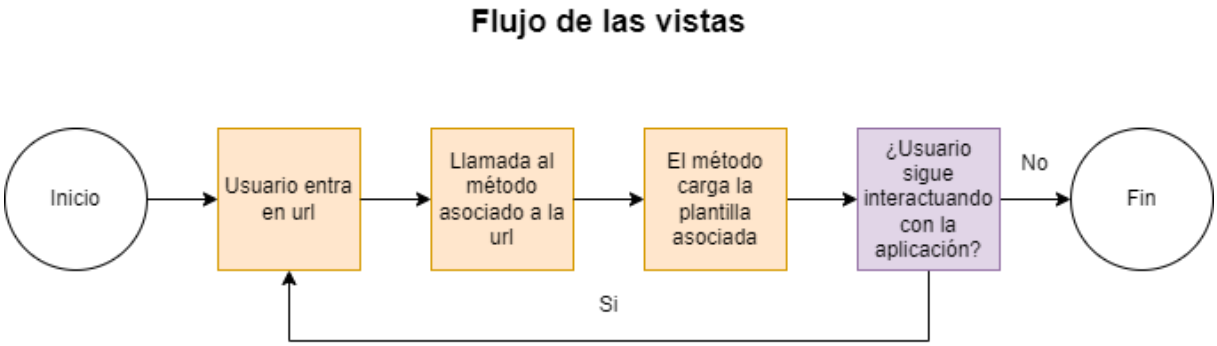


Ilustración 60: Imagen del flujo de las vistas

5.4.3. Cámara

La aplicación DL4Vision entre todas sus opciones, cabe destacar el uso de la cámara, ya que las opciones de elegir una foto o descargar una url son más sencillas a nivel de programación.

En un primer momento se intentó implementar la cámara para que echara fotos y buscando información se encontró la biblioteca de Python OpenCV. Esta biblioteca busca una cámara conectada en el equipo y cuando se carga la cámara, esta te devuelve un frame. Por tanto, es tan sencillo como que devuelva frames sin parar y que los vaya mostrando. Django tiene implementadas una clase de streaming al que

se le pasa este objeto cámara que devuelve frames en bucle y es capaz de mostrarlo sin tener que cargar toda la vista, es decir, se refresca la imagen automáticamente.

La opción de tomar una foto, lo que hace básicamente es que al hacer click en el botón para tomar la foto guarda el último frame enviado por este objeto cámara y se muestra en la vista.

La opción de streaming es exactamente igual que la anterior, a diferencia de que justo antes de devolver el frame, se lo pasa al modelo y este lo devuelve con la predicción. Por eso, dependiendo del modelo, la calidad de la imagen puede variar, por ejemplo, si hubiera un modelo que tuviera el alto y el ancho de la imagen muy pequeño, al devolver los frames al tamaño original se notaría una pérdida de calidad, cuando en la cámara el frame muestra lo que capta.

5.4.4. Script de instalación

Anteriormente se ha mencionado que la aplicación está desarrollada sobre una imagen de Docker Compose que alberga todo lo necesario (bibliotecas, drivers, la propia aplicación, etc.). Esta imagen está en Docker Hub para que desde cualquier sitio se pueda acceder a esta mejorando así su portabilidad. Pero aun teniendo todo esto, puede que no funcione, ya que tener la imagen de Docker no es requisito único, sino que el equipo dónde se quiere usar esta aplicación necesita algunos requisitos más. Algunos de los principales requisitos son, que el sistema operativo tenga Linux, que tenga instalado Docker y Docker hub para poder ejecutar esta imagen, instalar los drivers de Nvidia para que el equipo sea capaz de reconocer la tarjeta gráfica y opcionalmente se recomienda tener instalado Chrome.

Tras ver lo necesario para la instalación se puede pensar que esta es muy compleja, pero para hacerlo más sencillo se ha programado un archivo bash en el que con una serie de comandos se puede ejecutar todo esto. Para instalarlo se debe de ejecutar este script y previamente son necesarios los requisitos mencionados. Además, crea un segundo ejecutable para que cada vez que se quiera ejecutar la

aplicación arranque la imagen y abra el Chrome con la url. Posteriormente, esto se verá en un guion de instalación en el que se explica más detalladamente el proceso.

5.5. Incrementos realizados.

Como hemos visto en el punto 2.4.3, la metodología que se va a usar para este proyecto es una metodología incremental, es decir, se ha desarrollado por incrementos. A continuación, se va a explicar brevemente cada incremento.

El primer incremento ha estado enfocado en desarrollar la interfaz de la aplicación software. La idea era implementar un software básico con el que interactuar con un modelo. Se realizó un software lineal desde una pantalla de inicio hasta la pantalla final donde una vez elegida la categoría y la base de datos podíamos predecir una imagen. Esta podía ser seleccionada entre muchas otras predeterminadas que contiene la aplicación. Además, se hizo una vista de administrador, que para acceder a ella, se necesita un usuario o una contraseña. En esta última vista mencionada la idea era añadir, editar o eliminar categorías, bases de datos o modelos.

En el segundo incremento se implementaron más opciones para que el usuario pudiera predecir una imagen, concretamente se añadieron dos. Primero se añadió una funcionalidad para que el usuario pudiera insertar el enlace de una imagen, la aplicación la descarga y se la envía al modelo. La segunda opción consistía en que el usuario se tome una foto que será enviada al modelo.

En el tercer incremento, aprovechando el código implementado para tomar una foto, se añadió una opción más que consiste en la predicción de imágenes en tiempo real a través de la cámara. Pero esta funcionalidad no podía ser aprovechada porque hasta el momento la única implementada era la categoría de clasificación de imágenes y esta no podía aprovechar esta funcionalidad. De esta forma, posteriormente se implementó la categoría de detección de objetos.

En el siguiente incremento se añadieron dos categorias más, la categoría de segmentación y la de detección de pose.

El último incremento se enfocó en mejorar la aplicación, teníamos una aplicación con mucha funcionalidad, pero era muy poco portable. De este modo, se introdujo en el proyecto la tecnología de Docker, se creó una imagen de Docker que contenía el sistema operativo, bibliotecas de Python, drivers necesarios y la aplicación. Además, se implementó un script para que pudiera descargar esta imagen de docker e instalarla.

6. PROBLEMÁTICAS ENCONTRADAS Y SOLUCIONES

Durante el desarrollo de la aplicación han ido surgiendo varias problemáticas a las que se le han ido buscando soluciones. En algunas ocasiones, se han conseguido resolver utilizando otro camino, pero en otras se han tenido que limitar las expectativas ya que no era posible llevarlo a cabo en el tiempo que lleva realizar un TFG.

6.1. Generalización de la aplicación para cualquier modelo

Una de las primeras dificultades que se presentaron fue el poder encontrar la manera para conseguir que cualquier modelo, con solo introducirlo en la base de datos, funcione. Cada modelo está implementado de manera distinta, ya que algunos usan TensorFlow o Keras, es decir, el método de carga del modelo es diferente según la biblioteca que se use. De hecho, si surgiera una biblioteca con características similares a las mencionadas, pero con algunas mejoras sería conveniente tenerla en cuenta. Además, que de base, no se saben qué métodos ni qué clase se van a usar. No se conocen los parámetros que hay que pasarle a la aplicación ni los que va a devolver, ya que, mientras que algunos modelos basta con pasarle solo la imagen, hay otros que necesitan el alto, el ancho, que se redimensione la imagen, que se le añada una dimensión, que la imagen se pase a una biblioteca en específico, que sean bytes o un array de numpy, etc. A la salida del método pasa exactamente lo mismo, no se sabe que va a devolver.

Por tanto, se ha diseñado para que en el caso de que sean similares, es decir, que tenga la misma categoría, el mismo database, los mismos parámetros de entrada y de salida, se pueden introducir varios modelos. Parece un tanto complejo, pero normalmente en TensorFlow Hub los modelos que tienen la misma categoría y el mismo database la forma en la que están implementados los modelos es la misma porque suelen ser de la misma organización. Por ejemplo, Google tiene una gran cantidad de modelos, más de 70 de clasificación de imágenes. Todos estos se podrían introducir y funcionarían a la perfección. En la sección de guiones se enseñará cómo seleccionar estos modelos que se asemejan y cómo añadirlos.

Para concluir se podría decir que se puede añadir cualquier modelo si ya existe un modelo anterior que tenga características similares. En el caso de querer añadir algo distinto, se tendría que entrar en el código para implementarlo, porque a pesar de verse en las vistas esto daría error.

6.2. Limitación de portabilidad

Uno de los objetivos que se plantearon al inicio del desarrollo de la aplicación fue que se pudiera implementar con facilidad en cualquier lugar, pero al elegir hacer la aplicación web con un framework como Django, implicaba que no podría ser como una aplicación de escritorio en la que se lanza un ejecutable y ya estaría instalado todo.

No se consiguió la manera de encontrar un ejecutable que se bajara la aplicación web y la ejecutara, ante esta situación, se observaron dos posibilidades nuevas. Por un lado, se podría pasar el código y ejecutarlo con el IDL de Pycharm en local, lo que es muy tedioso, pues solo para ejecutar la aplicación se necesitan muchos requisitos previos. Por otro lado, existía la posibilidad de usar Docker, introduciendo todo en una imagen, ya que lo único que se requería era tener Docker para ejecutarla.

Nos quedamos con la opción del Docker, pero aun así encontramos varias problemáticas, puesto que en Windows cuando ejecutas la imagen en un contenedor no es capaz de encontrar la cámara. Se intentó de muchísimas maneras, pero Windows no está implementado para coger la cámara. Por tanto, la aplicación se vio limitada a que solo pueda ser usada en un dispositivo con Linux.

6.3. Pruebas lentas

Como se comentaba anteriormente, los modelos tardan varios minutos en cargarse, además del tiempo que tarda el modelo en predecir una imagen, por tanto las pruebas en un principio eran demasiado lentas, ya que si se quería comprobar cualquier cambio había que esperar a que se ejecutara. Cuando se implementó para

que cargara todos los modelos asignados por defecto fue incluso peor ya que para probar cualquier cosa había que esperar a que todos los modelos se cargarán.

Por todo este se decidió mejorar los tiempos tanto de cargar los modelos como el tiempo que tarda en predecir un modelo gracias a utilizar los drivers Nvidia Cuda. Esta mejora no fue tan evidente en el portátil en el que se estaba desarrollando la aplicación porque la tarjeta gráfica de este no era de las mejores, pero una vez que se probó en los equipos de la universidad que poseían una tarjeta gráfica mejor, realmente se notó el cambio, sobre todo mejoró el delay que existía en la opción de streaming. Pasó de verse a cámara lenta a tiempo real.

6.4. Problemas con el equipo

El portátil del desarrollador contiene una tarjeta gráfica de baja calidad, por lo que las pruebas eran lentas, pero al menos podían llevarse a cabo. El problema llegó cuando fueron necesarias las pruebas en Linux para hacer funcionar la aplicación. El portátil que se usó tiene Windows 11 y no Linux, por lo que se intentó usar una máquina virtual como VMWare o VirtualBox, pero si ya era complejo que la cámara cogiera la imagen de Docker y los drivers, que además los coja la máquina virtual y luego la imagen de esta, era bastante más complicado. Al final, gracias a los docentes se pudo superar esta problemática ya que se le proporcionó al desarrollador un PC de una de las aulas de la universidad para usarlo y realizar las pruebas necesarias. Se preparaba todo lo que se quería probar y se planteaban alternativas por si fallaba aprovechar el tiempo lo mejor posible. Si no llega a ser por esta facilidad hubiera surgido la necesidad de hacer una partición solo de Linux para hacer pruebas.

7. PRUEBAS CON USUARIOS

Hay que destacar que la aplicación desarrollada en este TFG fue concertada con el Parque de las Ciencias de Granada para incluirse en su exposición "Inteligencia Artificial. Una exposición sobre las personas, los datos y el control".

Previamente se ha podido probar con usuarios en otros eventos de la universidad, como el evento de “LA NOCHE EUROPEA DE L@S INVESTIGADOR@S 2022” (ilustración 60).

Ilustración 61: Cartel de La noche Europea de l@s Investigador@es 2022. Fuente.

En este evento donde muchos investigadores de la universidad mostraban a la gente curiosidades científicas, se expuso esta aplicación para acercar a los usuarios la Inteligencia Artificial. Se instalo en dos equipos, en una con una pantalla grande donde se utilizaban sobre todo modelos en tiempo real, sobre todo para captar la atención del usuario y se viera en la pantalla con alguna predicción (véase imagen 61).

Ilustración 62: Foto de usuarios interaccionando con la aplicación

El segundo equipo era un ordenador con un ratón para que pudieran interactuar con el resto de la aplicación.

Además, la Universidad de Jaén, gracias al proyecto piloto cofinanciado por la Unión Europea denominado campus GEM, acogió entre el 11 y el 15 de julio del 2022 a 60 niñas entre 13 y 18 años y los profesores que tutorizan este TFG les enseñaron de una manera entretenida y sencilla algunos conceptos sobre inteligencia artificial y Deep Learning. Estas chicas tuvieron la oportunidad de probar esta aplicación y se tomaron las siguientes imágenes (ilustración 62):

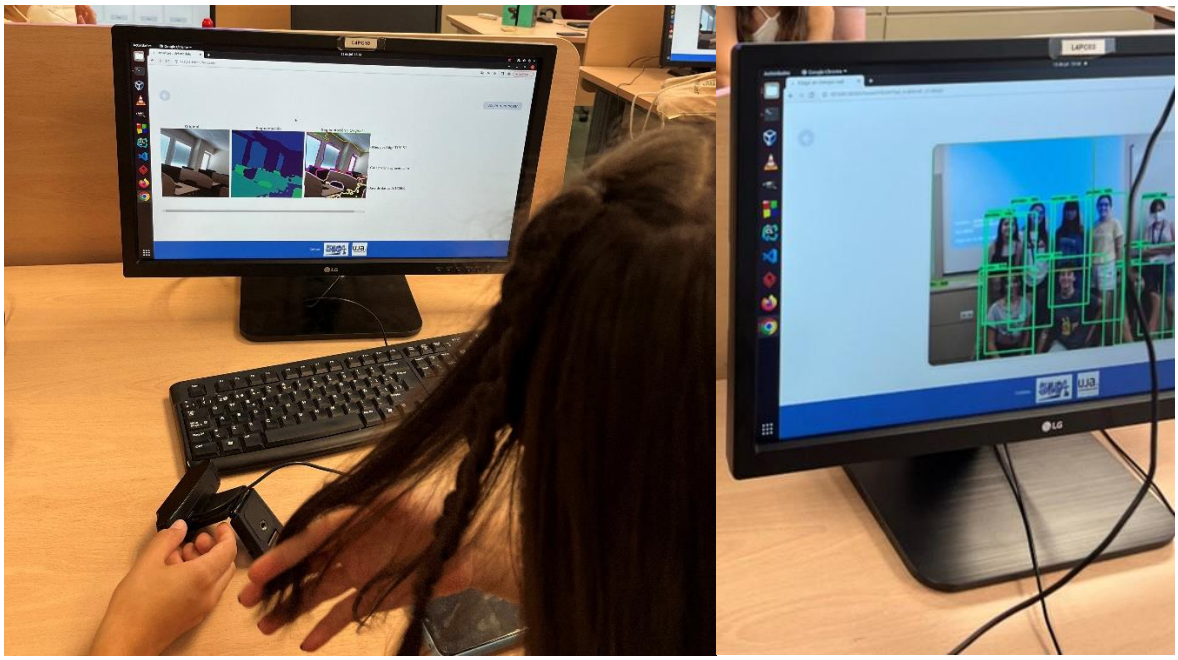


Ilustración 63: Campus GEM

8. MEJORAS DE FUTURO

En este apartado se van a comentar las mejoras que han ido apareciendo durante el desarrollo, pero por ausencia de tiempo suficiente no se han investigado o no han sido añadidas. Esta aplicación tiene un nivel de mejora muy amplio, empezando por añadir más categorías y más bases de datos, es decir, programarlas en el código. Pero se van a mencionar principalmente dos mejoras que son las que se han considerado más importantes.

Añadir la aplicación a un servidor sería una de las mejoras. No se ha pensado un servidor en especial, pero Google Cloud podría ser un buen ejemplo. Al tener la aplicación desplegada en un servidor se podría usar con tan solo conexión a internet y un navegador. Si en un futuro, la aplicación se sigue utilizando y tiene éxito se podría desplegar y configurar para que usuarios de todo el mundo pudieran añadir sus propios modelos y usarlos.

La segunda opción que se había pensado cambiar, es la manera de introducir los modelos en la aplicación, meterlos a través de un fichero ejecutable de Python con dos métodos, donde el primero sea para cargar el modelo y el segundo para predecir una imagen. A partir de aquí no se ha investigado mucho, no se conoce la posibilidad de ejecutar un fichero, aunque se sospecha que ha de haber algo parecido para hacerlo con todas las bibliotecas de Python. Se establecería una plantilla descargable donde se explique que todos los ficheros tienen que tener en común estos dos métodos con sus parámetros de entrada y salida en común. Aunque puede que a la hora de la verdad, cuando se empiece el desarrollo no sea tan factible, pero es una idea que se ha barajado durante la fase final del desarrollo de la aplicación.

9. CONCLUSIÓN

Se ha realizado una plataforma, DL4Vision, que es capaz de usar modelos de TensorFlow Hub para predecir imágenes con 4 categorias de Deep Learning distintas, clasificación, detección de objetos, segmentación y detección de pose. Incluso es flexible a que se le puedan añadir más categorías, mas bases de datos o más modelos. Esta te permite en unos segundos predecir una imagen o detectar la pose. Con muchas facilidades de entrada de imagen. Se podría llegar a mejorar mucho más como hemos comentado antes, pero los objetivos especificados para este TFG se han cumplido. Incluso a los usuarios les ha gustado y se ha visto reflejado lo que se quería en ellos, acercarles la Inteligencia Artificial.

Personalmente, es un TFG que me ha gustado realizar. He aprendido mucho sobre Deep Learning, bibliotecas de Python y programar aplicaciones. Espero que en el futuro, los modelos que utilice sean míos.

10. APENDICE

A continuación, se introduce documentación adicional de utilidad para comprender correctamente el proyecto.

10.1. Guía original del Trabajo Fin de Grado.

Desarrollo de una plataforma para la aplicación de métodos de Deep Learning

- Tutor del TFG: **MARÍA DOLORES PÉREZ GODOY**
- Modalidad: Proyecto de Ingeniería | Tipo: TFG Específico
- Número máximo de estudiantes: 1 (0 asignados)
- Idioma: Castellano
- Segundo tutor del TFG: **ANTONIO JESÚS RIVERA RIVAS**

Actualmente es mucha la información disponible en todos los ámbitos de la sociedad y las expectativas se dirigen hacia la extracción de conocimiento partiendo de dicha información. Por eso, en las últimas décadas, la ciencia de datos está en continuo desarrollo y avance. Un aspecto muy importante de la ciencia de datos es la extracción de conocimiento a partir de imágenes (visión por computador). Una de las técnicas de machine learning más utilizadas para la extracción de conocimiento de las imágenes son las redes neuronales profundas (deep learning), dados los buenos resultados que se obtienen con ellas. Esto ha hecho que este tipo de técnicas se constituyan en uno de los modelos más investigados. El objetivo de este trabajo es el desarrollo de una plataforma que permita a los usuarios finales la utilización de modelos de deep learning aplicados a diferentes tareas sobre imágenes.

Conocimientos Previos: No requiere

Objetivos del TFG:

- Desarrollo de un módulo que permita la obtención y preprocesamiento de datos (imágenes).
- Desarrollo de diferentes módulos que permitan la aplicación de modelos *Deep learning* en diferentes tareas de visión por computador.
- Desarrollo de un módulo que permita el análisis y visualización de los resultados.
- Desarrollo de una herramienta permita la implantación de los módulos anteriores.
- Escritura de la memoria del Trabajo Fin de Grado.

Metodología a Desarrollar:

- Análisis y estudio de las bibliotecas y métodos disponibles en *deep learning* para visión por computador.
- Análisis del sistema que incluya los elementos que contendrán las fases de preprocesamiento, obtención del conocimiento y análisis y visualización de resultados.
- Análisis y estudio de las tecnologías de desarrollo existentes, que permitan la ejecución de métodos de *deep learning* y justificación de la propuesta seleccionada.
- Diseño de la aplicación.
- Codificación e implantación.
- Pruebas y generación de la documentación asociada.

Documentos y Formatos de Entrega:

- Memoria del trabajo fin de grado en formato digital.
- Prototipo software.

11. MANUALES

En este apartado vamos a ver todos los guiones para aprender bien cómo usar la aplicación, ver todas las utilidades y elementos que tiene. Además de instalarlo correctamente.

11.1. Guion de instalación

11.1.1. Comprobación

Este es un paso para asegurarse de que está todo bien antes de ejecutar la aplicación. Hay que comprobar que se disponga de conexión a internet y que la webcam esté conectada. Para comprobar esto, se hará una pequeña búsqueda rápida en Google para comprobar la conexión.

Respecto a la cámara, el cable debería estar conectado a un puerto **USB** del equipo. Si desea comprobar que funciona, Ubuntu trae una aplicación llamada Cheese para ver que efectivamente funciona, para encontrar la aplicación Cheese sigue los siguientes pasos:

- Arriba a la izquierda del todo aparece un botón llamado actividades (ilustración 63).

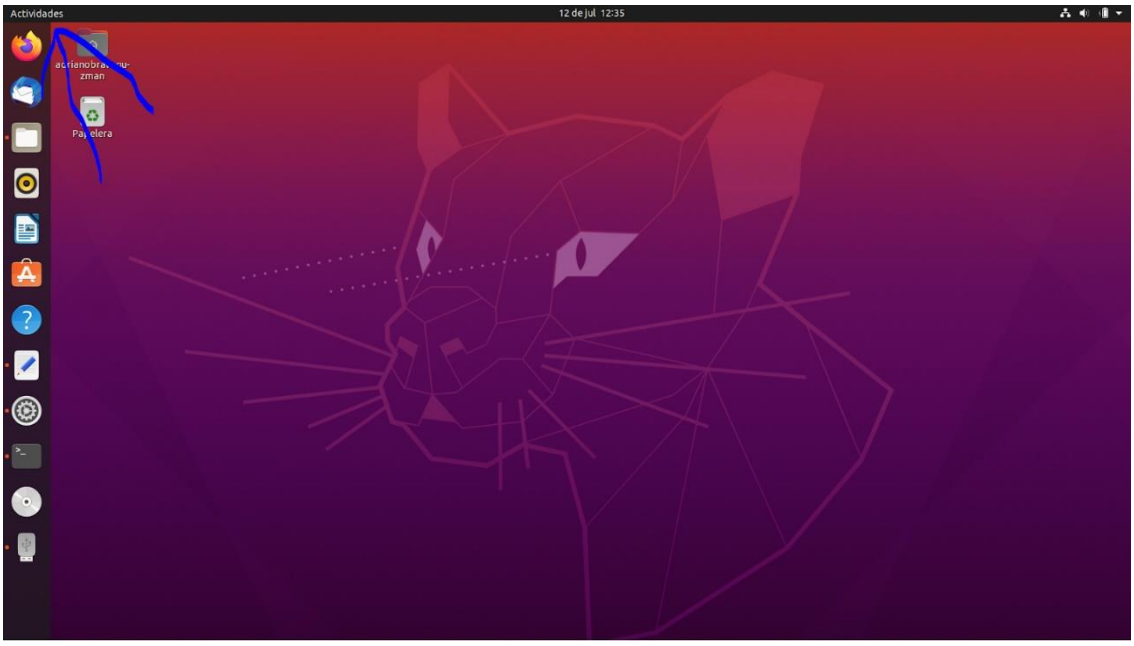


Ilustración 64: Captura de pantalla de escritorio de Linux

- Hacemos click y se abrirá la siguiente ventana (ilustración 64):

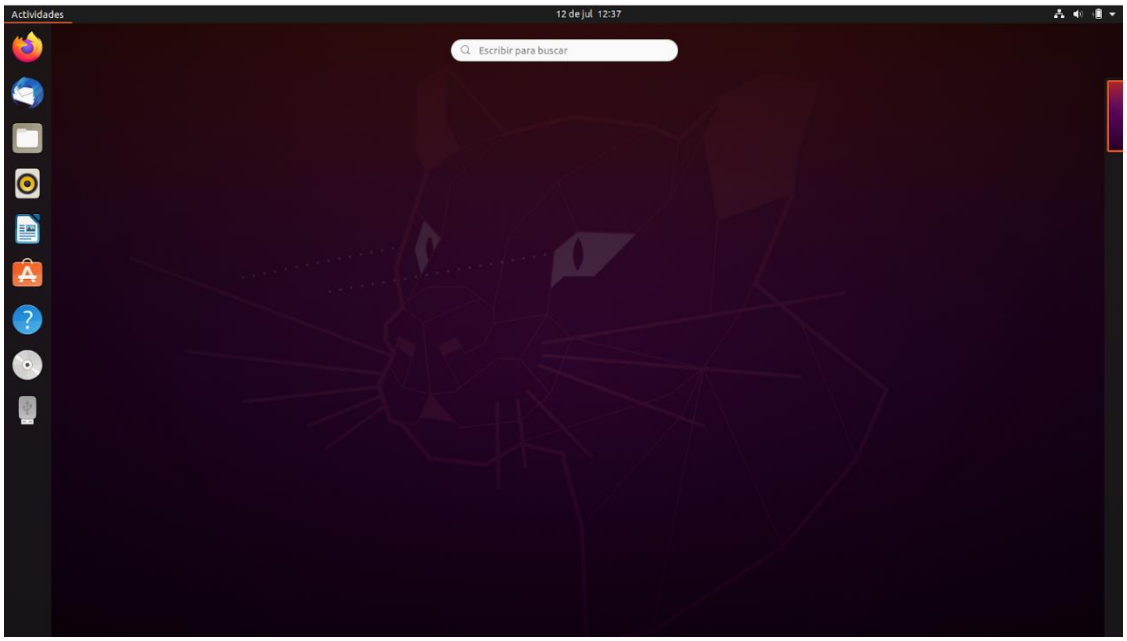


Ilustración 65: Captura de pantalla de buscador de Linux

- En la barra de búsqueda se escribe Cheese (ilustración 65):

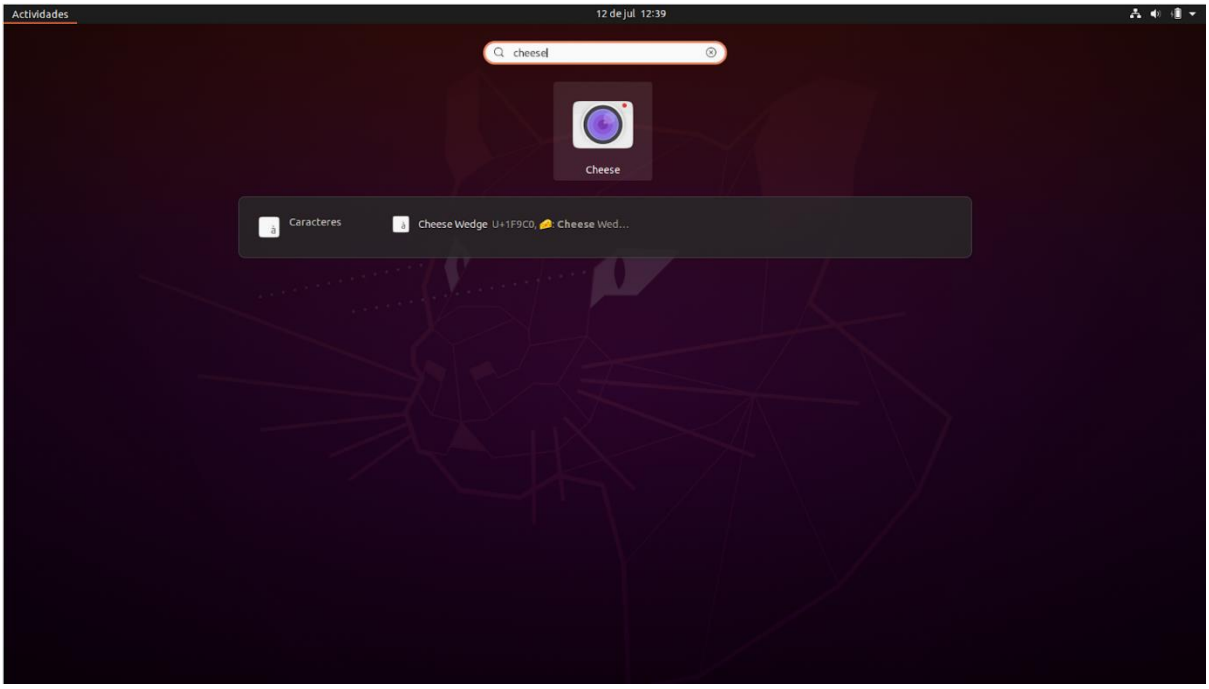


Ilustración 66: Captura de búsqueda de aplicación Cheese de Linux

Se clica en el icono de la cámara y aparecerá la imagen que está captando la cámara. En el caso de que no traiga esta aplicación se instalará desde el terminal.

- Para abrir un terminal, se parte de la imagen 2 y se escribe terminal en la barra de búsqueda. Se clica en el icono negro rectangular que aparece (ilustración 66).

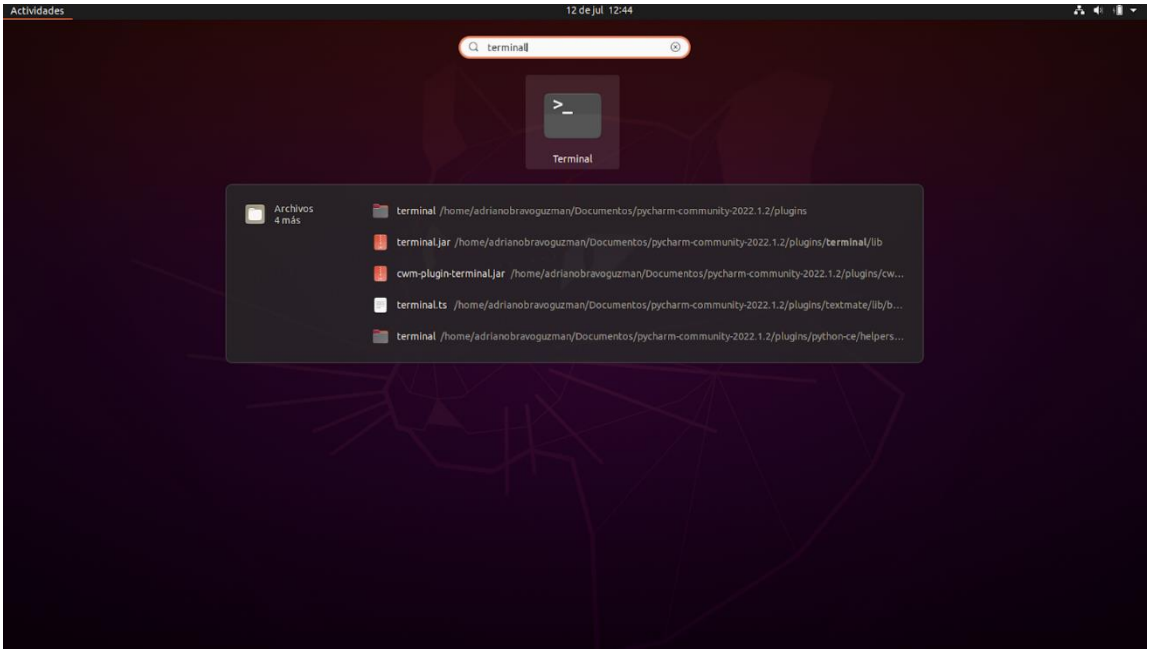


Ilustración 67: Captura de búsqueda del Terminal de Linux

Se abrirá una ventana con unas letras verde y se escribe el siguiente comando *"sudo apt-get install cheese"* y se le da a intro, aparecerá esto (ilustración 67):

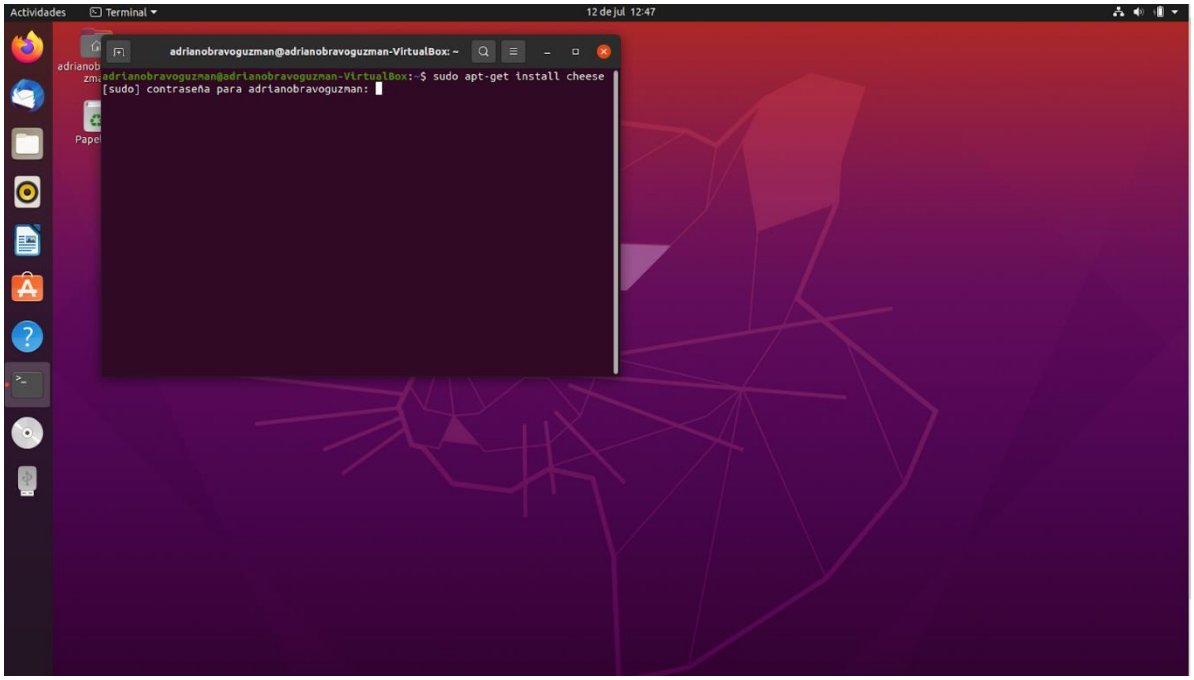


Ilustración 68: Captura del Terminal al ejecutar el comando sudo apt-get install cheese

- Ahora introducimos la contraseña de administrador (si no se es el propietario del equipo, se tiene que preguntar al administrador a cargo).

Una vez, todo instalado, si pregunta si se desea continuar escribimos la letra s y le damos a intro. Cerramos el terminal.

11.1.2. Descargas

11.1.2.1. Comprobar drivers de Nvidia e instalarlos

Para comprobar si los drivers de nvidia están instalados abrimos un terminal y ejecutamos el comando *"nvidia-smi"* y si están instalados debería aparecer algo así (ilustración 68):

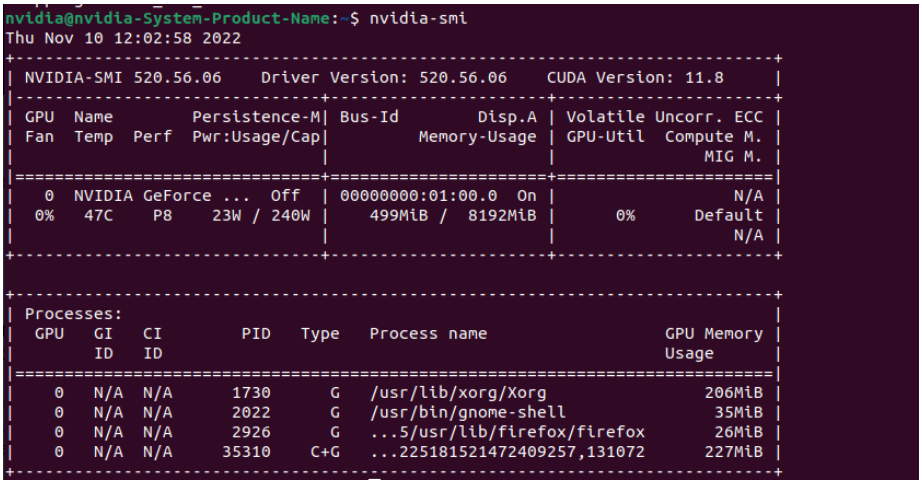


Ilustración 69: Captura del Terminal al ejecutar el comando nvidia-smi

En el caso de que no aparezca, como en la imagen anterior, insertamos el comando *"ubuntu-drivers devices"* (ilustración 69).

Ilustración 70: Captura del Terminal al ejecutar el comando ubuntu-drivers devices

Aparecerá la tarjeta gráfica de Nvidia que tenemos y el número de controlador que nos recomienda. En este caso, en la imagen podemos ver en la penúltima línea, que recomienda el driver nvidia-driver-520-open por la palabra *recommended* al final de la línea.

Una vez que sabemos el nombre del driver vamos a instalarlo con la orden "*sudo apt install nvidia-driver-520-open"*.

Reiniciamos el sistema y si se desea reiniciar el terminal también puede hacerse con la orden *"sudo reboot"*. Ahora, con el comando *"nvidia-smi"* que usamos al principio, debería salirnos algo parecido a la primera imagen.

11.1.2.2. Ejecutar script

Para descargar todo lo necesario para que funcione la aplicación, lo único que tenemos que hacer es descargar el script, darle permisos de ejecución y ejecutarlo. Para descargar el script hacer click aquí, y nos llevará a un archivo de drive donde deberemos clicar donde aparece el ratón (ilustración 70).

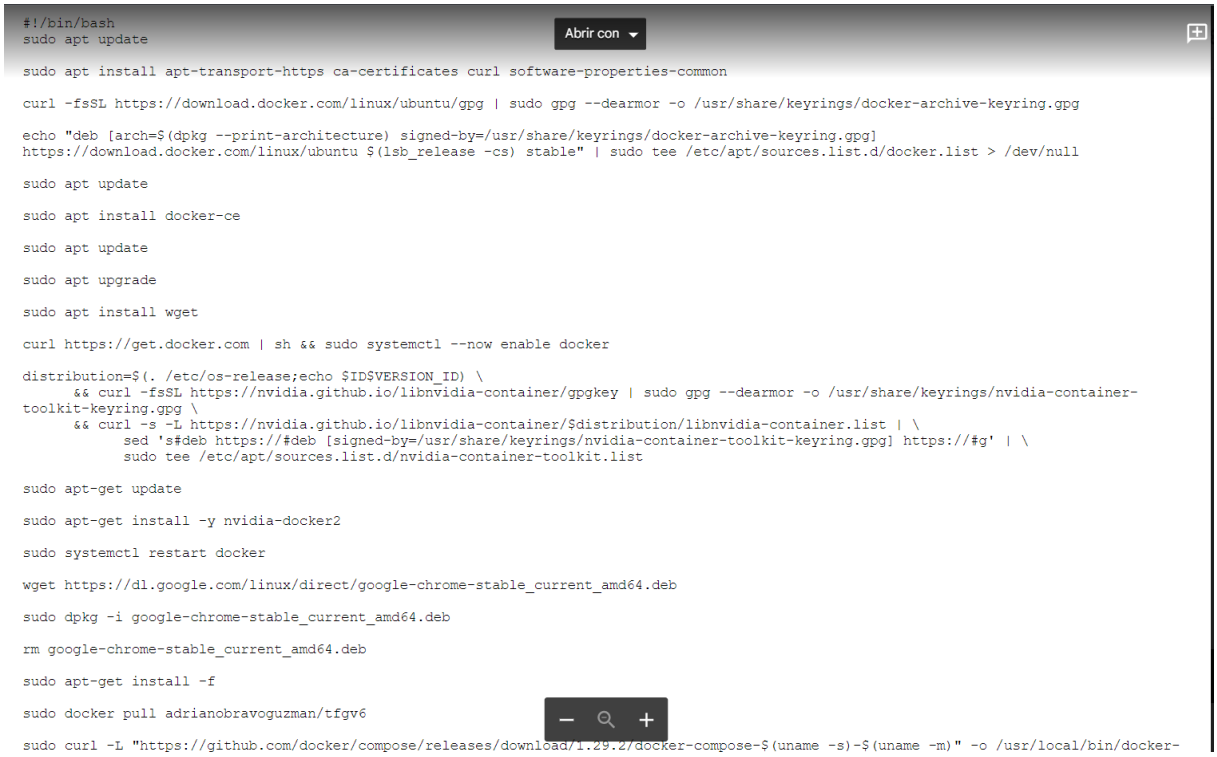


Ilustración 71: Captura del Script de instalación

Una vez descargado el archivo, abrimos el terminal donde esté el archivo descargado. Lo normal es que esté en Descargas o Downloads, lo podemos mover a donde queramos, en este caso se pondrá en el Escritorio o Desktop, en una carpeta llamada DL4Vision y se meterá el script ahí (ilustración 71).

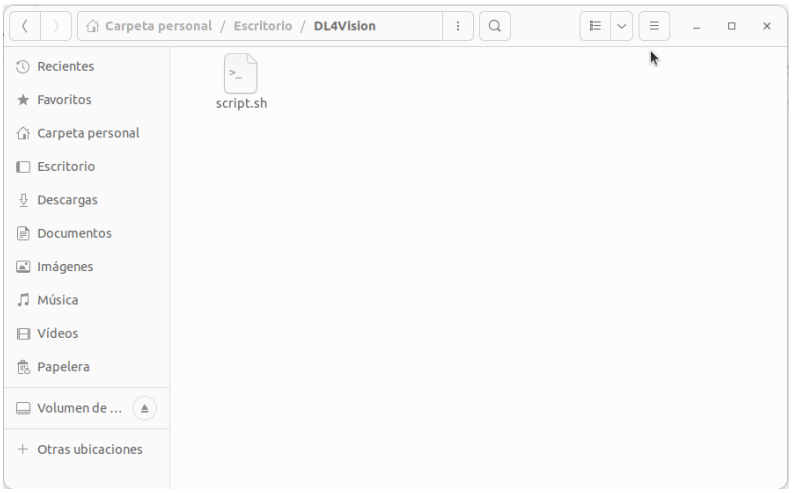


Ilustración 72: Captura de la carpeta DL4Vision con el Script dentro

Si hacemos click derecho entre todas las opciones que aparecen, nos saldrá una para abrir un terminal, clicamos en esa. Una vez abierto, le damos permisos de ejecución con el comando *"chmod +x script.sh"* y lo ejecutamos con el comando *"./script.sh"*.

A continuación, empezarán a descargarse varios archivos, aparecerá un mensaje como este (ilustración 72).

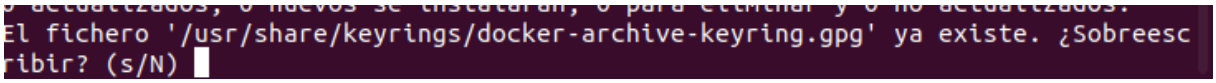


Ilustración 73: Captura de mensaje al ejecutar el script (1)

Escribimos la tecla de la letra s y damos a la tecla intro.

Además, habrá un momento que se quedará parado y nos saldrá un mensaje como este (ilustración 73):

```
# Executing docker install script, commit: 4f282167c425347a931ccfd95cc91fab041d4
14f
Warning: the "docker" command appears to already exist on this system.

If you already have Docker installed, this script can cause trouble, which is
why we're displaying this warning and provide the opportunity to cancel the
installation.

If you installed the current Docker package using this script and are using it
again to update Docker, you can safely ignore this message.

You may press Ctrl+C now to abort this script.
+ sleep 20
```

Ilustración 74: Captura de mensaje al ejecutar el script (2)

Esto es normal, solo hay que esperar unos segundos y la ejecución seguirá su curso. Hay que tener paciencia porque tiene que descargar muchos archivos y la imagen de la aplicación pesa casi 9 GB, por tanto, la descarga va a llevar un tiempo. Cuando aparece la versión de Docker (véase en la imagen posterior) significa que la instalación ha acabado (ilustración 74).

```
Docker Compose version v2.12.2
nvidia@nvidia-System-Product-Name:~/Escritorio/DL4Vision$
```

Ilustración 75: Captura de mensaje al ejecutar el script (3)

11.1.3. Ejecución

Si nos fijamos, donde hemos ejecutado el script se nos ha creado un archivo llamado DL4Vision.sh, este nos va a servir para ejecutar la aplicación (ilustración 75).

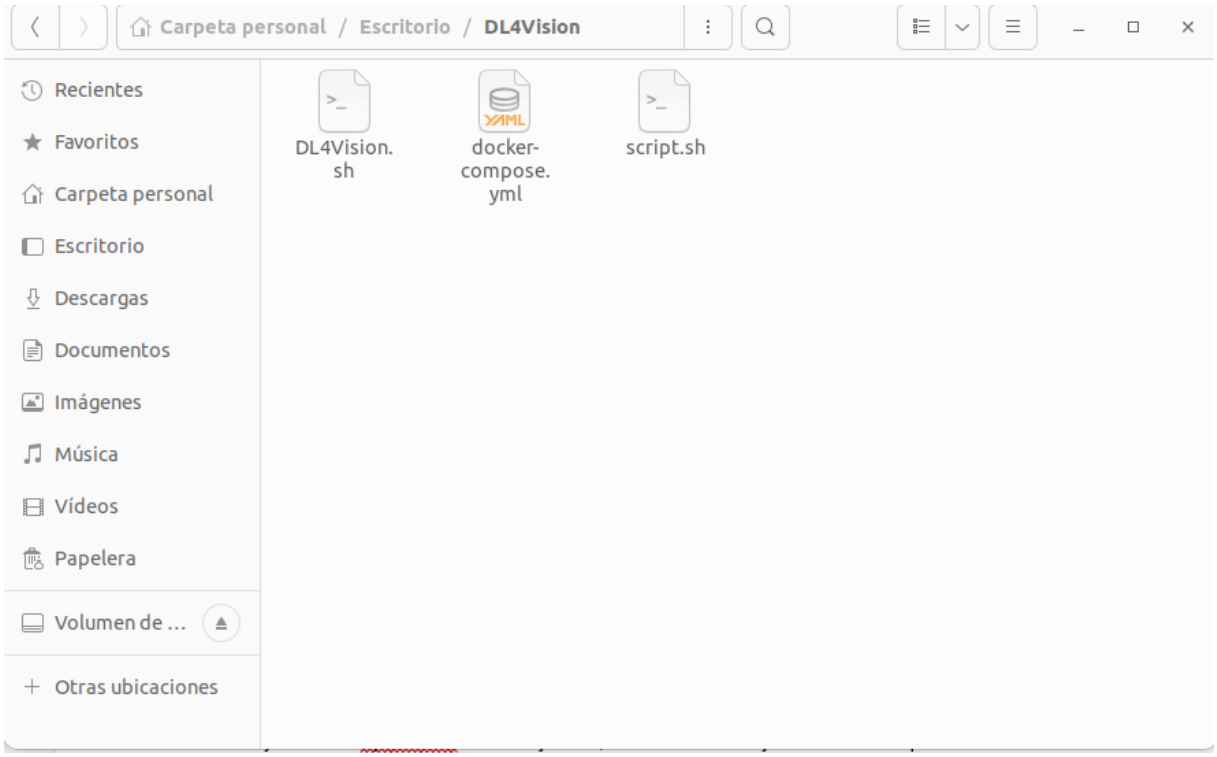


Ilustración 76: Captura carpeta DL4Vision con el archivo DL4Vision.sh

Abrimos el terminal y escribimos el comando *"./DL4Vision.sh"*. Se nos abrirá una ventana de Chrome con la ruta de la aplicación. La dejamos a un lado por ahora, y nos centramos en el terminal. Pueden pasar dos cosas:

- Que se quede parado como en la siguiente imagen (ilustración 76):

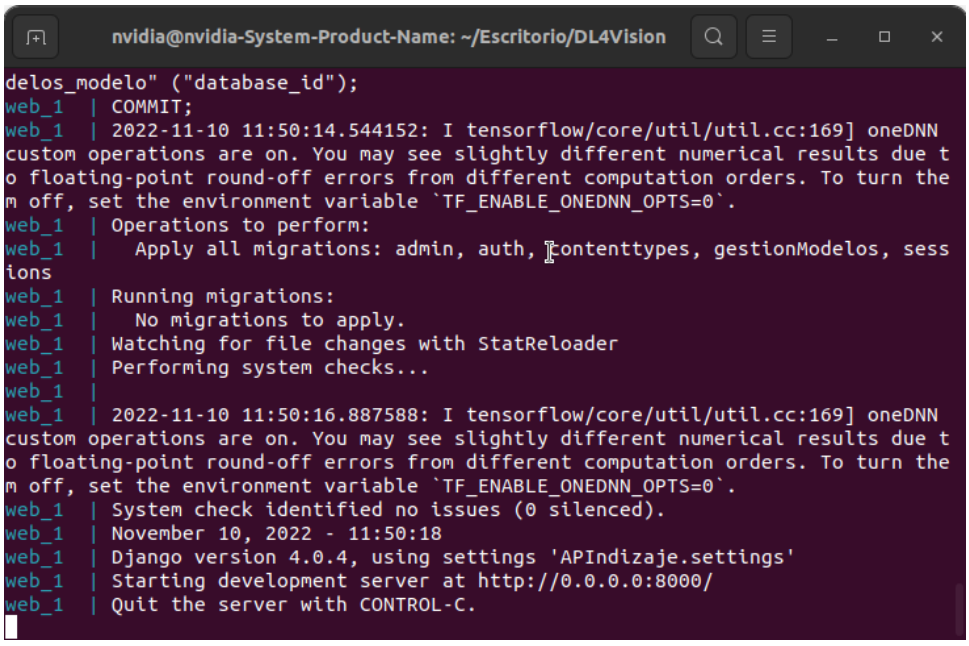


Ilustración 77: Captura del terminal al ejecutar DL4Vision.sh

En este caso lo único que hay que hacer es click en la ruta en Chrome y darle a intro o al botón de recargar página y volvemos a la terminal.

- Que continúe solo, en ese caso seguimos en la terminal. Ahora va a cargar los modelos y puede tardar un poco, sabremos que están los 4 modelos de la aplicación cargados cuando aparezca en el terminal esto (ilustración 77):

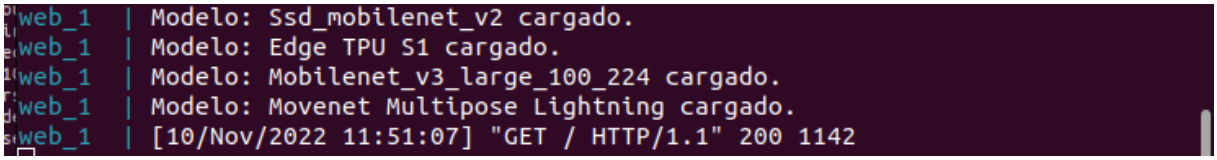


Ilustración 78: Captura del terminal donde muestra que los modelos se han cargado

Una vez aparezca la última línea de la imagen ya está lista la aplicación para usarla (ilustración 78):

Ilustración 79: Captura de la vista inicial de DL4Vision

11.1.4. Recomendación

Aquí se va a comentar la configuración que usamos en el equipo:

- Usar pantalla completa para que el usuario no pueda salir de la aplicación, la propia aplicación está diseñada para que el usuario pueda moverse por ella sin necesidad de usar el navegador con varios botones.
- Quitar que la pantalla se suspenda. Durante varios minutos puede que nadie interactúe con la aplicación porque los usuarios estén observando la detección de objetos o en tiempo real. Por tanto, para evitar que se ponga la pantalla negra, se mostrará a continuación cómo quitar la suspensión de pantalla (ilustración 79).

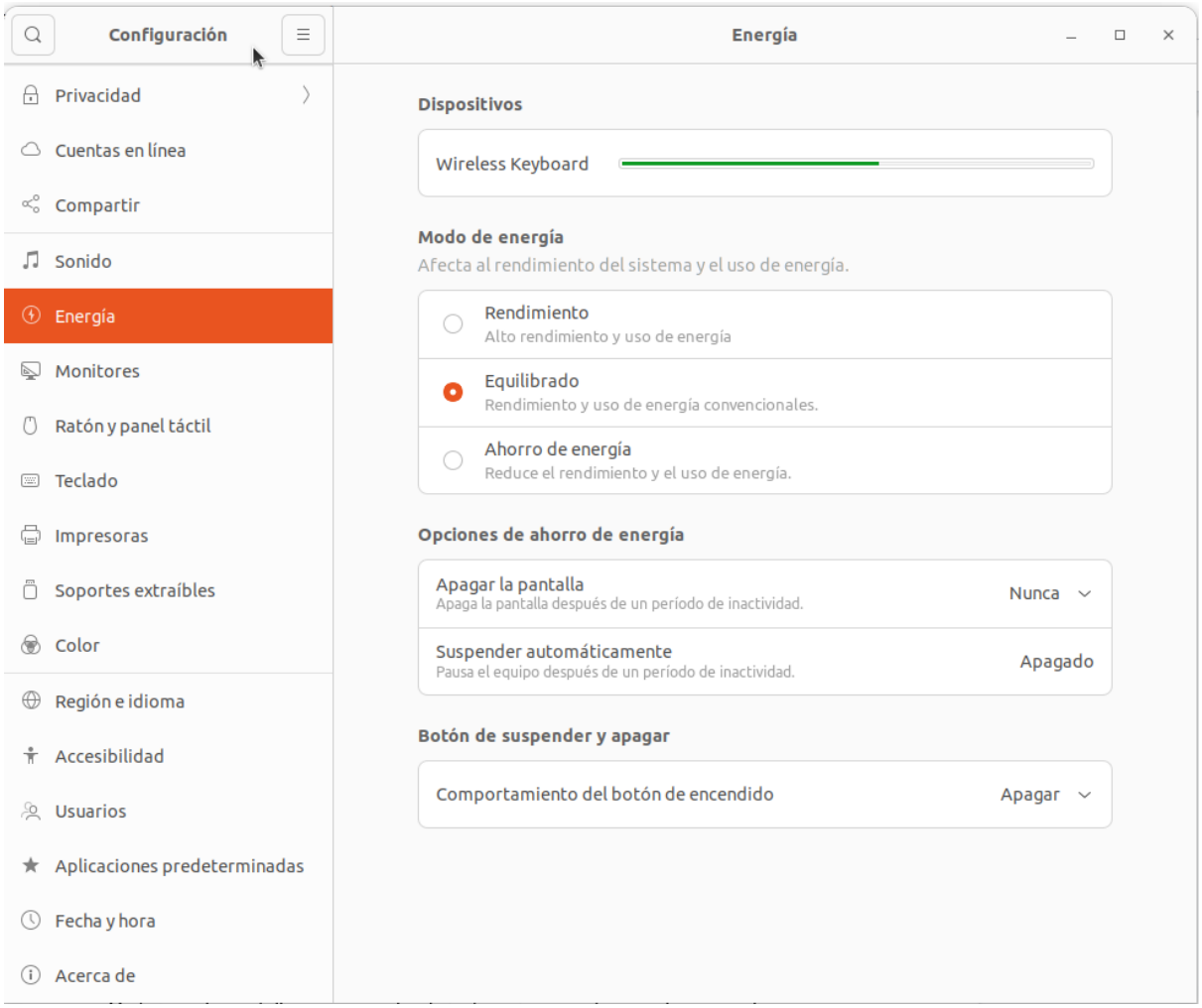


Ilustración 80: Captura de la ventana Configuración en la pestaña de Energía

- Habilitar dos clicks de ratón:

Con la intención de facilitar el uso del proyecto vamos a comentar la forma en la que se podría ejecutar “con un par de clics”.

Lo primero que hay que hacer es meter al usuario en un grupo en el que no se necesiten contraseñas para realizar la ejecución de las órdenes. Esto puede realizarse con el comando *“sudo usermod (nombreUsuario) -G docker”.* Tras esto habría que reiniciar.

Después hay que configurar el archivo para que se pueda ejecutar con dos clics. Hay que tener en cuenta que en función de las versiones esto puede complicarse. Si se está en un Linux 20.04 o previos, con ir a la configuración de la carpeta,

Preferencias -> comportamiento -> archivos de texto ejecutables ->ejecutar, lo tendríamos ya listo (ilustración 80).

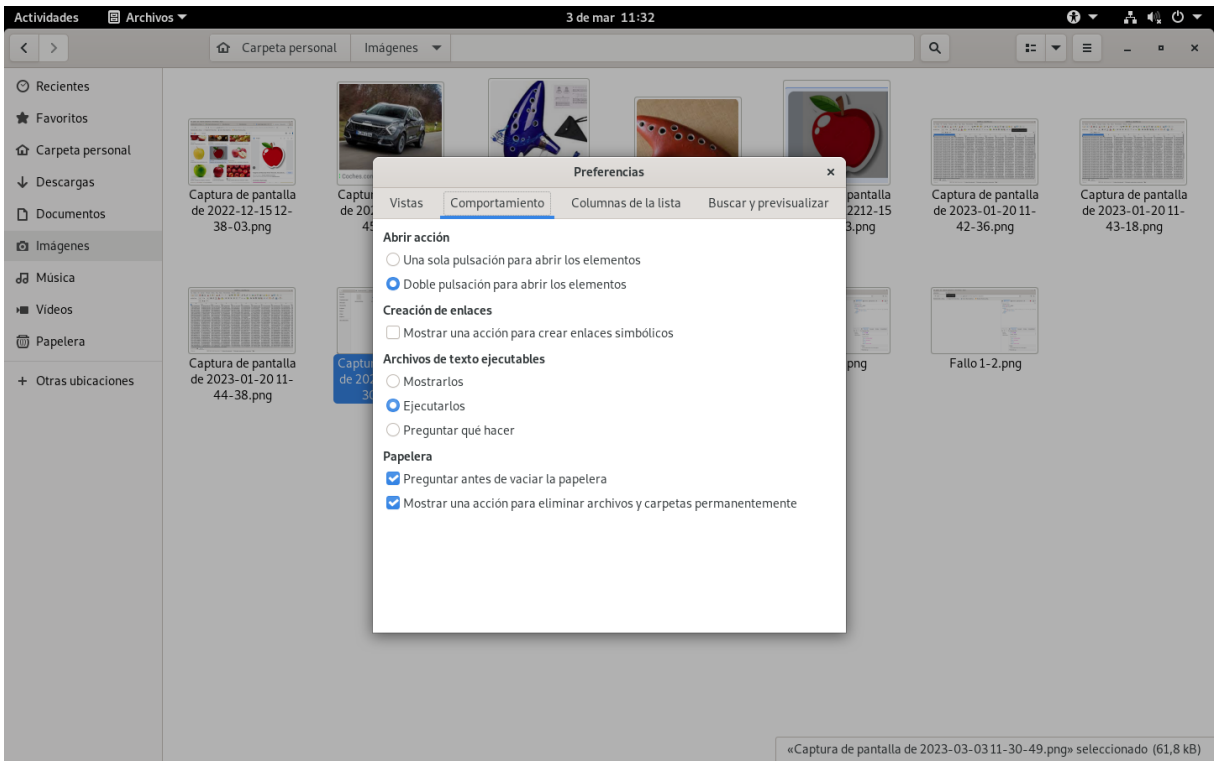


Ilustración 81: Captura de la ventana Preferencias

Si se está en una versión 22.04 la cosa diferirá en lo siguiente: habría que hacer click con el botón secundario y “Ejecutar como un programa”. Se tiene que tener en cuenta que en ambos casos el archivo debe tener permiso de ejecución.

11.2. Guion de usuario

11.2.1. Introducción

A continuación, se presenta un breve guion en el que se explica a los usuarios como usar la aplicación, todas las posibilidades que esta ofrece y todo lo que contiene. Comenzaremos, en primer lugar, con la página de inicio de la aplicación y se explorará cada rincón de la misma. Además, en última instancia se podrá ver una imagen en la que se muestra un resumen del flujo de la aplicación.

En ella se usan modelos de deep learning, los cuales son entrenados con unas bases de datos y son capaces de predecir a partir de lo que han aprendido

Esta aplicación destaca porque los modelos mencionados anteriormente no están dentro del software de la aplicación, sino que estos están en una nube de una página llamada TensorFlow Hub.

También está diseñada para educar a los usuarios y que estos vean las distintas categorías y bases de datos que trabajan con imágenes. Además, se les proporcionan varias herramientas para que usen los modelos que veremos posteriormente.

11.2.2. Vista de inicio

Esta es la página de inicio, en la que encontramos el logo, el nombre y una breve explicación de lo que veremos a continuación. El nombre “DL4Vision” es una composición de palabras proveniente del inglés “Deep Learning for Vision”. Su misión es predecir las imágenes y realizar las principales tareas de visión por ordenador, las imágenes de entrada se pueden introducir de diferentes maneras en la aplicación. Al hacer clic en “Acceder” nos llevará a la siguiente vista, la vista de categorías (ilustración 81).

Ilustración 82: Captura de vista de inicio de DL4Vision

11.2.3. Vista de categorías

En esta vista encontramos todas las categorías disponibles en la aplicación. Estas sirven para diferenciar de manera más sencilla entre los distintos modelos que vamos a usar.

Visualmente podemos diferenciar varios elementos:

- Nombre: es el nombre de la categoría.
- Imagen: es una imagen que nos muestra lo que el modelo puede hacer antes de usarlo y darle un feedback a un usuario que no sepa nada sobre el tema.
- Una pequeña descripción: para reforzar la imagen se ofrece una pequeña descripción en la que se explica brevemente qué hacen los modelos de cada categoría.

En esta aplicación hay 4 categorías distintas (ilustración 82):

- Detección de objetos: en la imagen resultante nos da una predicción sobre los objetos que ha detectado y un porcentaje que nos indica la probabilidad con la que está seguro de que su predicción es correcta. Puede detectar varios objetos al mismo tiempo.

- Segmentación: este tipo de modelos detecta objetos y los colorea, además, nos dibuja sobre la imagen original el contorno de los objetos que ha detectado.

- Detección de pose: dibuja en la imagen los puntos más importantes y característicos del cuerpo humano, dando lugar a un muñeco con una postura. Puede detectar la postura de hasta 6 personas.

- Clasificación: esta categoría está destinada principalmente a animales, es capaz de identificar el animal que aparece en la imagen e incluso la raza.

Posteriormente veremos una imagen de ejemplo.

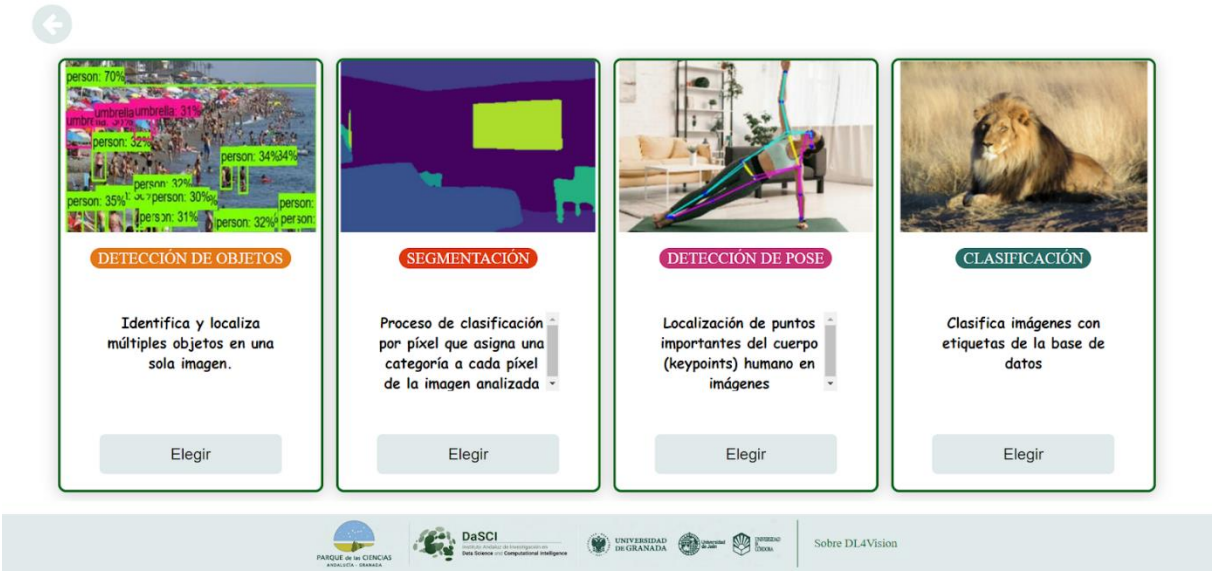


Ilustración 83: Captura de vista de categorías de DL4Vision

11.2.4. Vista de bases de datos

En esta vista encontramos las bases de datos, que, dependiendo de lo que hayamos elegido en la vista anterior, nos saldrá una u otra. Las bases de datos son un conjunto de datos, en este caso, un conjunto de imágenes, con las que los modelos han sido entrenados. Estos modelos se entrenan como si de una persona se tratara, se les entrena para que sean capaces de realizar una determinada tarea.

Para comprender esto haremos un símil; los modelos de imágenes funcionan del mismo modo que lo hace el cerebro humano, se les enseñan varias imágenes de pelotas y se le dice que cualquier objeto que cumpla estas características será una pelota, de forma que conseguimos que aprenda lo que es una pelota, es decir, lo entrenamos.

Visualmente podemos diferenciar varios elementos (ilustración 83):

- Nombre: se corresponde con el nombre de la base de datos.
- Imagen: le proporciona un aspecto más visual
- Una pequeña descripción: sirve para explicar un poco la base de datos.

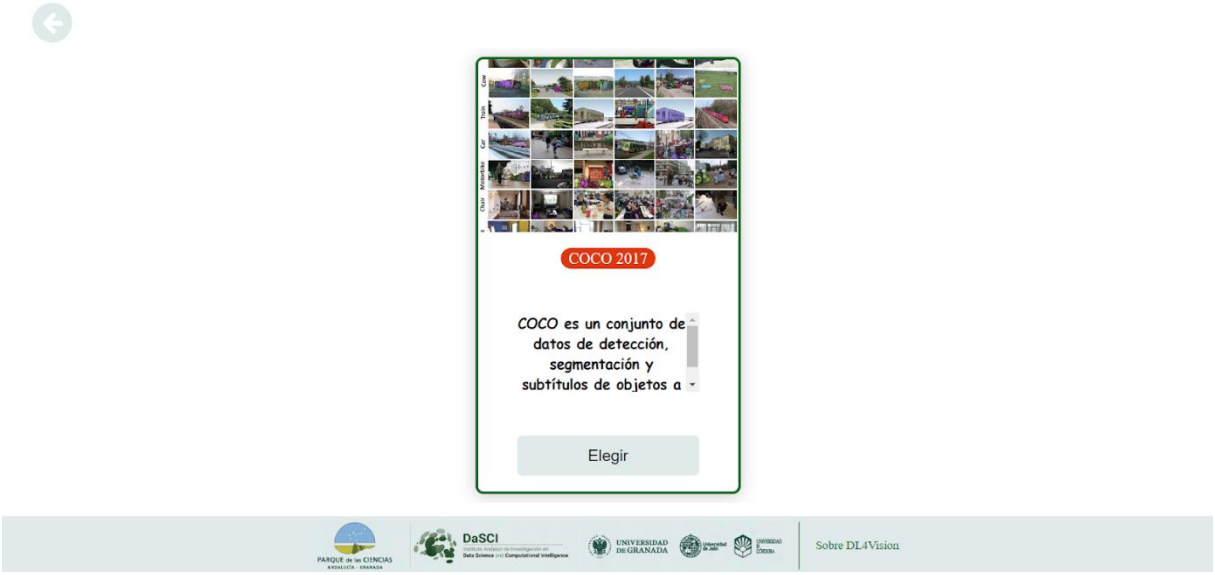


Ilustración 84: Captura de vista de base de datos de DL4Vision

11.2.5. Vista de opciones

Una vez elegidas la categoría y la base de datos de forma implícita hemos elegido también el modelo del que podremos ver su nombre en la vista de resultados. En esta vista podemos elegir cómo vamos a darle al modelo la imagen, en función de lo que hayamos elegido anteriormente, nos saldrán unas opciones u otras. Como máximo tendremos tres opciones por modelo, pero puede haber algunas más que el desarrollador haya decidido ocultar (ilustración 84-85).

Ilustración 85: Captura de vista de base de opciones de DL4Vision (1)

Ilustración 86: Captura de vista de base de opciones de DL4Vision (2)

A continuación, se mostrará el contenido de cada una de las opciones:

- Elegir una imagen: nos aparecerán varias imágenes entre las que tendremos que seleccionar una (ilustración 86).

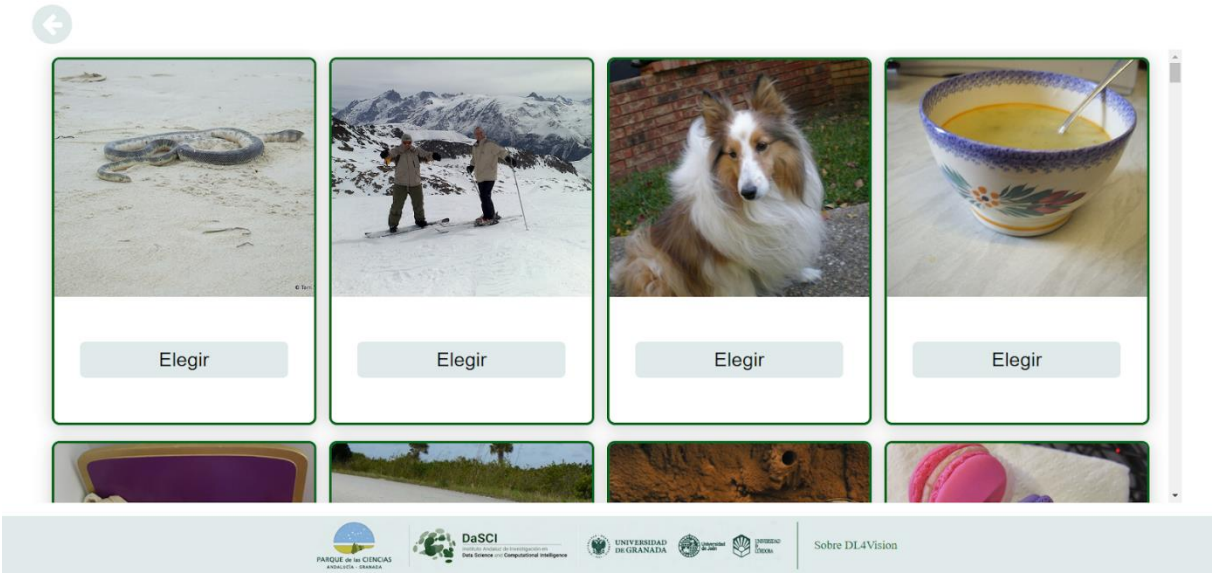


Ilustración 87: Captura de vista de elegir imagen de DL4Vision

- Elegir una url: podemos elegir una foto de internet e introducir el url para pasársela al modelo. Una vez hemos introducido el contenido en la caja de texto pueden darse dos situaciones (ilustración 87):
    - Si la imagen no sirve o ponemos algo que no sea un enlace → la página dará error (ilustración 88).
    - Si el enlace es correcto → nos dará la opción de elegir otra o enviar la que acabamos de introducir (ilustración 89).

Ilustración 88: Captura de vista de elegir una url de DL4Vision

Ilustración 89: Captura de vista de elegir una url al introducir una url no valida de DL4Vision

Ilustración 90: Captura de vista de elegir una url al introducir una url valida de DL4Vision

- Tomar imagen: el equipo tendrá una cámara conectada con la cual se podrá echar una foto y enviarla (ilustración 90). Para ello debemos de darle al botón de la cámara, al pulsarlo echará la foto y aparecerá una

pantalla en la que nos dará la opción de echar otra foto o enviar la que acabamos de tomar (ilustración 91).

Ilustración 91: Captura de vista de tomar una imagen

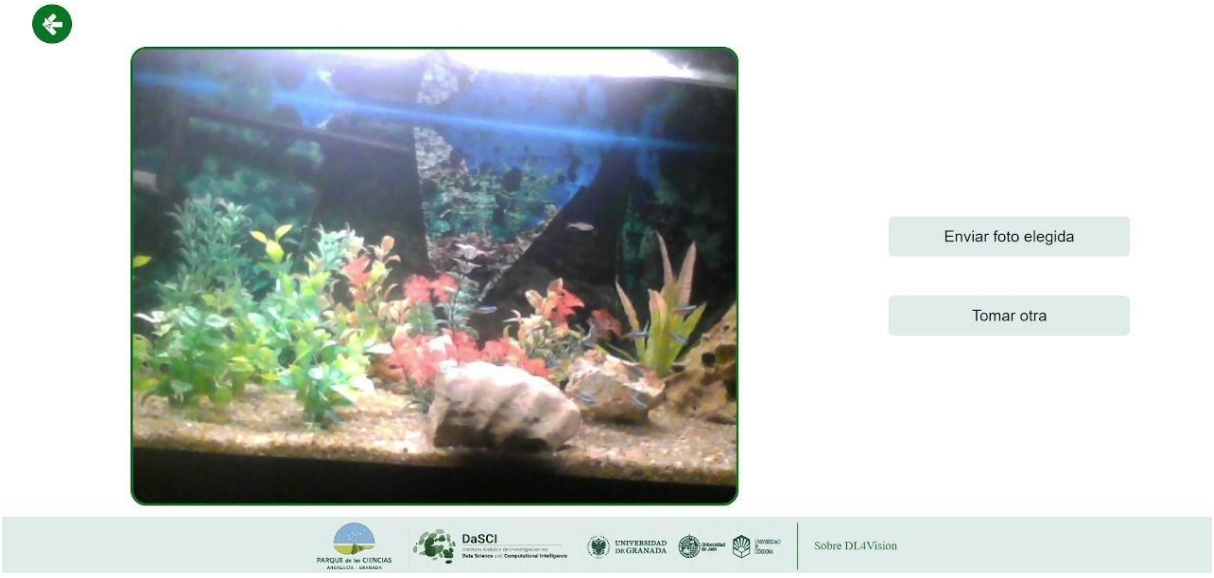


Ilustración 92: Captura de vista de tomar una imagen después de pulsar el botón de la cámara

- Tiempo real: podemos ver como el modelo trabaja en tiempo real (ilustración 92).

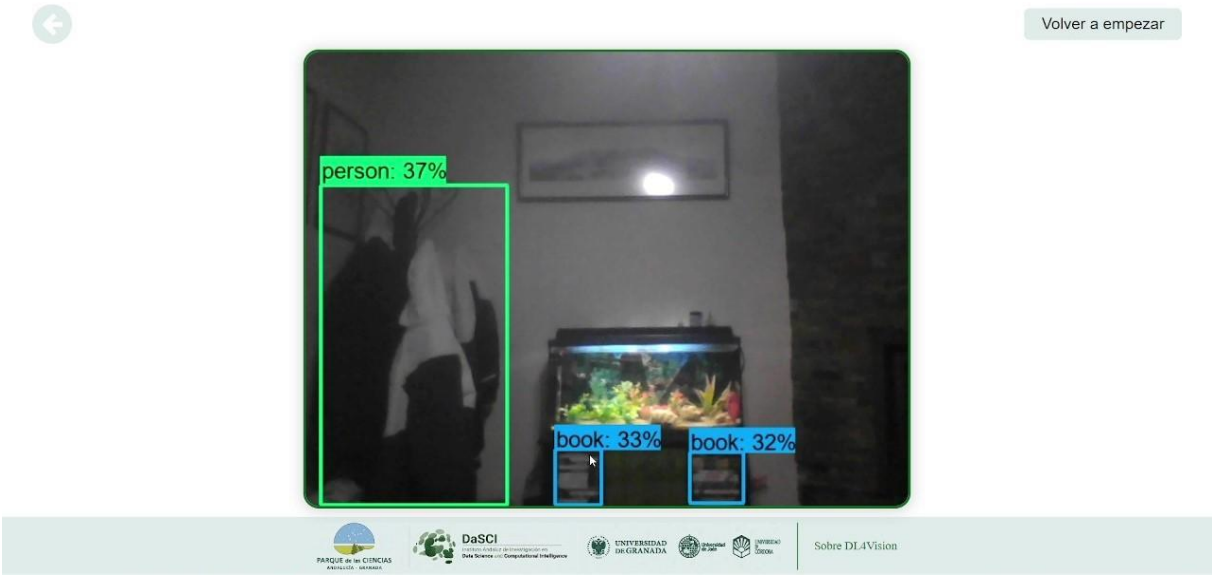


Ilustración 93: Captura de vista de tiempo real

11.2.6. Vista de resultado

En esta vista podemos ver el resultado de la predicción del modelo y un resumen de lo que hemos elegido. Llegaremos a esta vista a través de cualquiera de las opciones menos la de tiempo real (ilustración 93).

Ilustración 94: Captura de vista de resultado (1)

Si elegimos la categoría de segmentación difiere un poco a esta, ya que realmente para dibujar el borde de los objetos primero tiene que pintarlos, por lo que encontramos estas dos imágenes (ilustración 94):

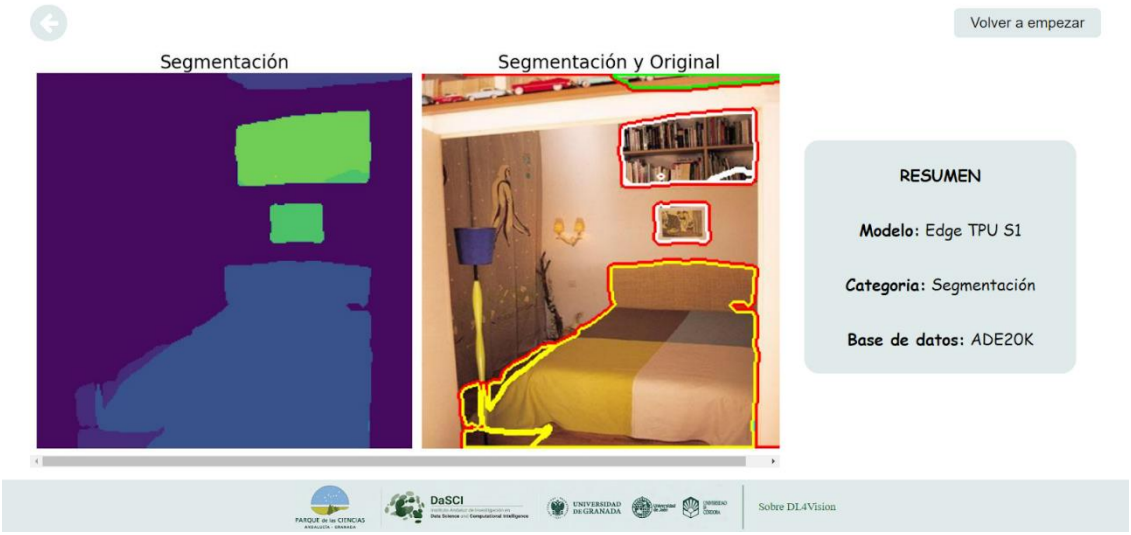


Ilustración 95: Captura de vista de resultado (2)

11.3. Guion de administrador

11.3.1. Introducción

Para introducir este guion me gustaría comentar que la vista de administrador, hasta el momento, solo sirve para añadir elementos a la interfaz y determinados modelos (diferentes o variantes de los ya existentes). Por tanto, el objetivo es que los desarrolladores puedan añadir elementos a la interfaz con mayor facilidad sin tener que modificar el código y además, algunos modelos a la funcionalidad de la aplicación que serán explicados posteriormente.

11.3.2. Entrar en modo administrador

Para entrar tenemos que escribir la siguiente url en el navegador: http://127.0.0.1:8000/admin. Una vez insertado este enlace nos pedirá unas credenciales (ilustración 95):

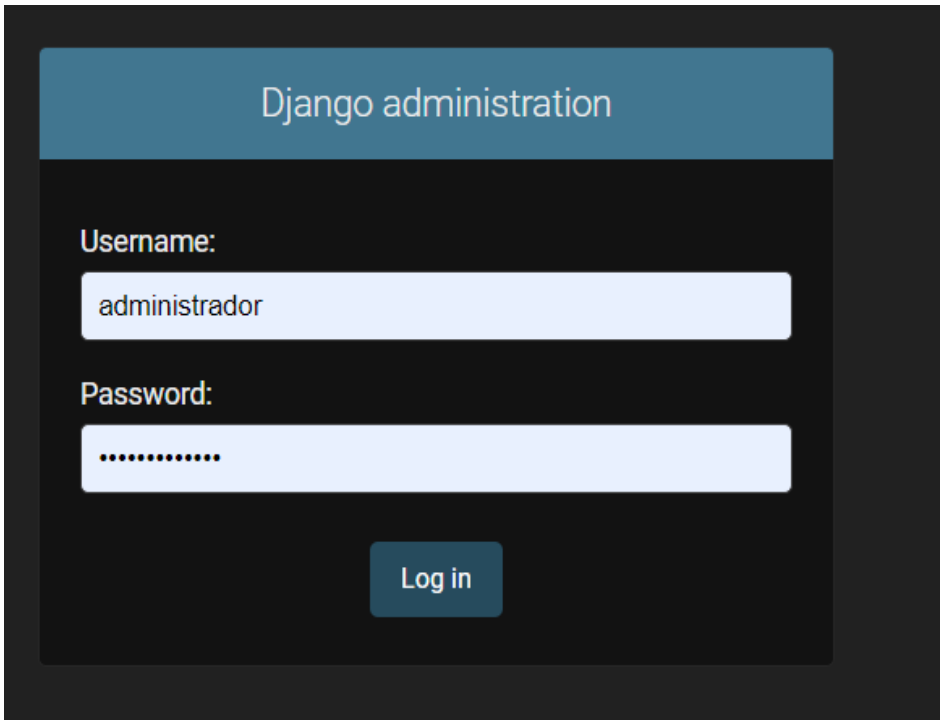


Ilustración 96: Captura de vista de login

Tanto en el username como en la contraseña pondremos la palabra “administrador”. El color depende del tema del equipo (oscuro o claro). Hacemos click en “Log in” y encontraremos la siguiente vista (ilustración 96):

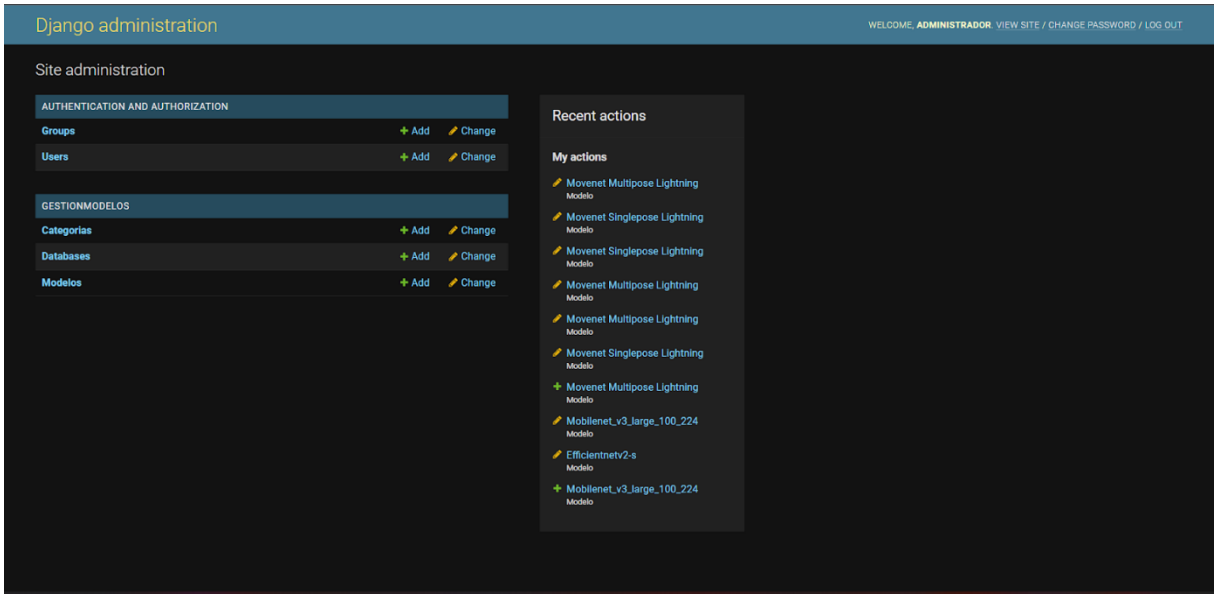


Ilustración 97: Captura de vista de administrador

El propio framework gestiona la autenticación y autorización de la aplicación mediante Usuarios y Grupos, aunque en principio no es imprescindible, puede servir en el futuro. En este guion no los utilizaremos. Ahora vamos a ver las dos utilidades que tiene este modo de administrador.

11.3.3. Añadir elementos a la interfaz

Como se ha comentado en la introducción, la primera utilidad es añadir elementos a la interfaz. Podemos añadir tanto Categorías, como Datasets, como Modelos. Para poder ver bien cómo funciona vamos a tener dos pestañas al mismo tiempo, por un lado, la vista de administrador y por otro, la aplicación. Hacemos click en Categorías y se observa lo siguiente (ilustración 97):

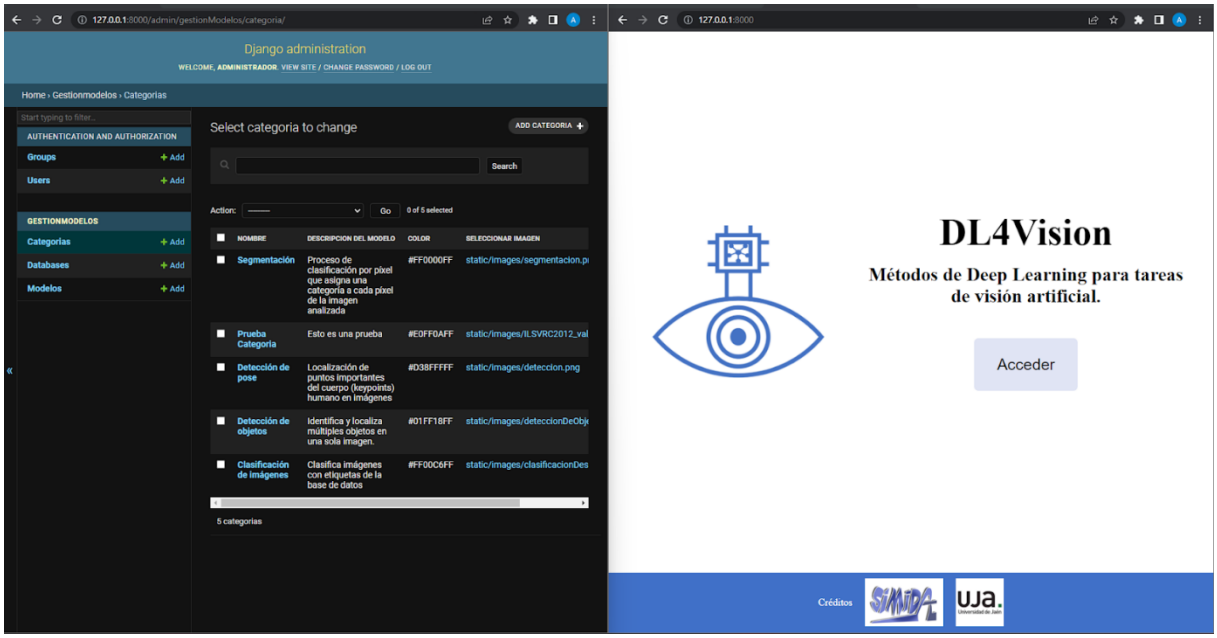


Ilustración 98: Captura de vista de categorías y vista inicial de DL4Vision

En la vista de administrador, podemos ver a la izquierda los tres elementos principales con los que poder ir navegando entre ellos, además, disponemos de un botón de añadir (“+ Add”) por si deseamos crear algún nuevo elemento. También, en la parte superior tenemos un buscador para poder encontrar cualquiera de los atributos que tiene cada categoría.

A continuación, en la vista de la aplicación, al hacer click en Acceder veremos (ilustración 98):

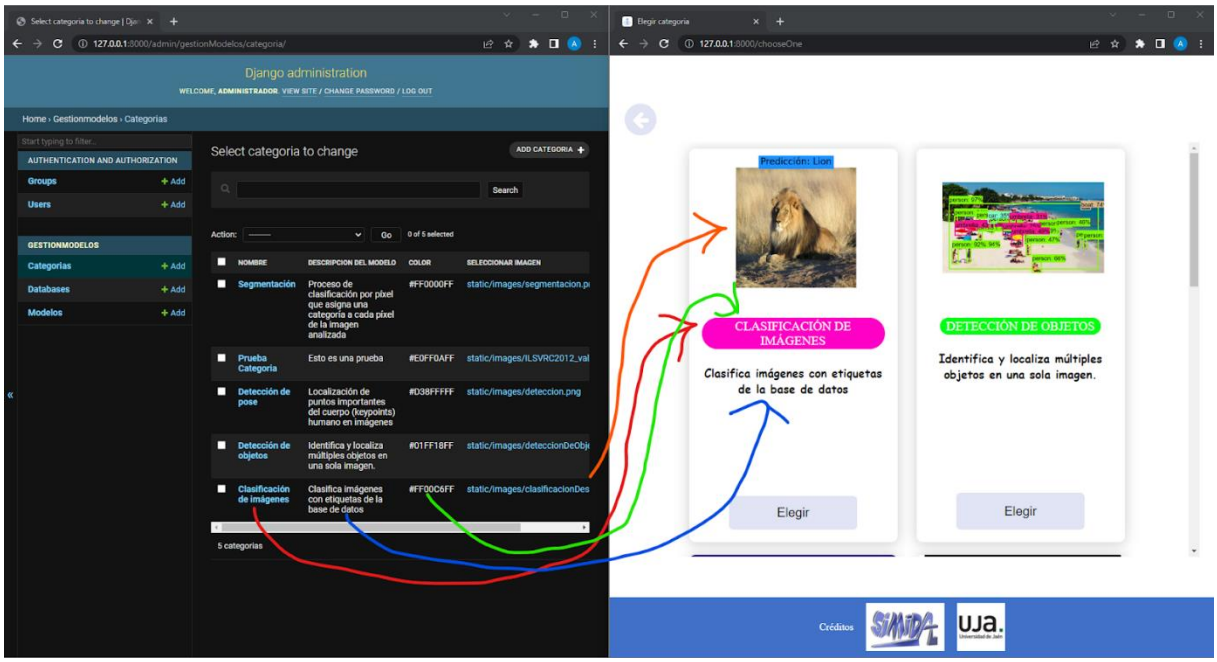


Ilustración 99: Captura de vista de categorías y vista de categorías de DL4Vision

La primera columna pertenece al nombre de la categoría, la segunda una breve descripción de la categoría, la tercera columna es el color que aparece para hacer la interfaz más ilustrativa y la última columna es la url dentro del proyecto de la imagen.

Para añadir una categoría hacemos click en “Add Categoria” (arriba a la derecha) y rellenamos todos los datos (podemos poner los datos que queramos) (ilustración 99).

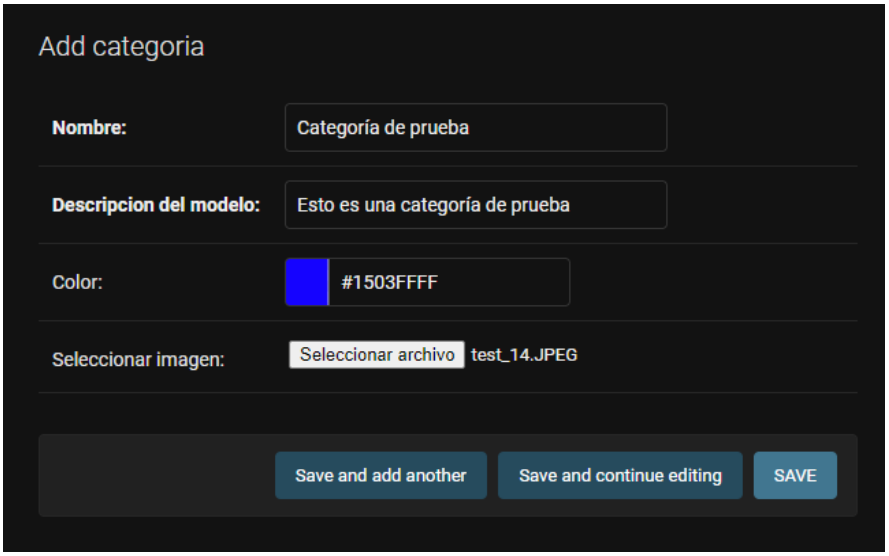


Ilustración 100: Captura de ventana para añadir categoría

El color se elige a través de una paleta de colores y la imagen se selecciona buscando una imagen del equipo (ilustración 100).

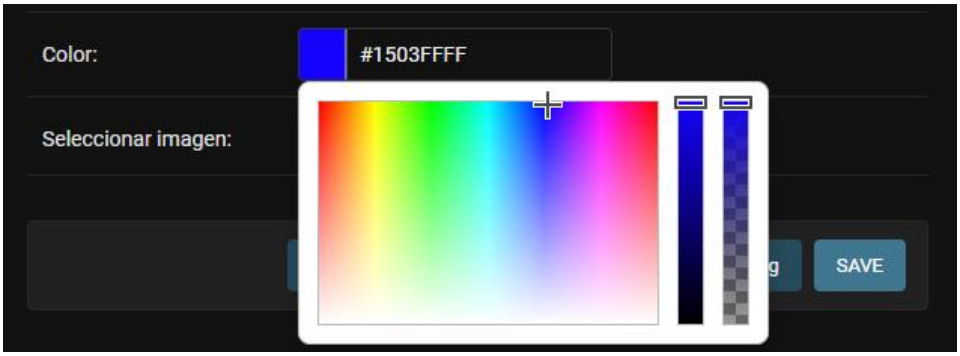


Ilustración 101: Captura de ventana de paleta de colores

Una vez todo relleno hacemos click en “Save” para guardar, aunque hay otras opciones en el caso de que se desee realizar otra cosa.

Nos devolverá a la pantalla anterior con un mensaje en verde que nos confirmará que la categoría se ha guardado correctamente, recargamos la aplicación y ya podremos visualizar la nueva categoría (ilustración 101).

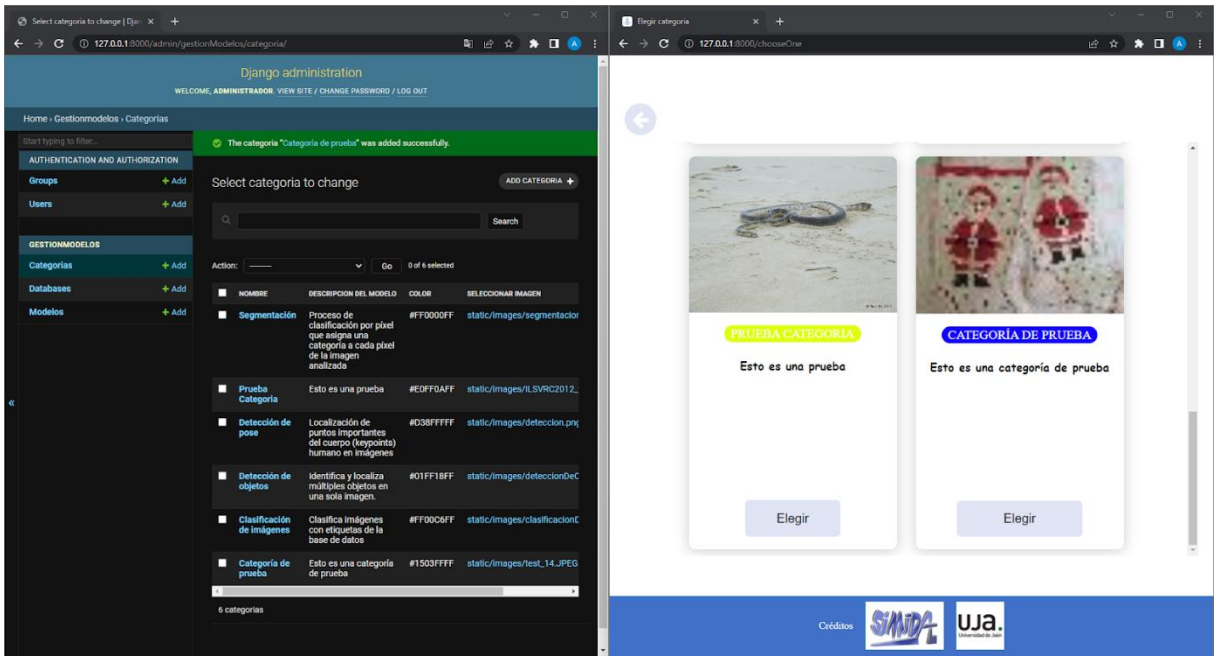


Ilustración 102: Captura de vista de categorías con mensaje de confirmación y vista de categorías al ser refrescada

Como podemos ver había una categoría de prueba (anteriormente creada), para borrarla marcamos el check que aparece a la izquierda del nombre de la categoría. Donde pone Action desplegamos, pinchamos en “Delete selected categorias” y pulsamos en el botón Go, quedando eliminada(s) la(s) categoría(s) (ilustración 102).

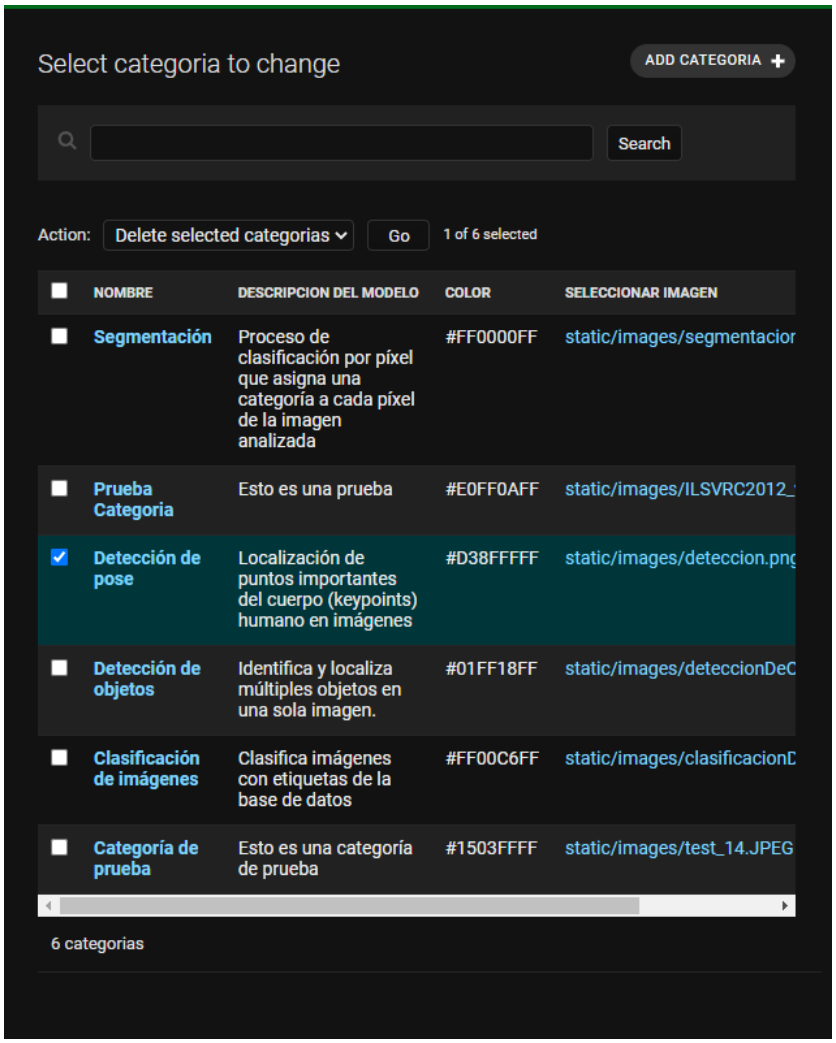


Ilustración 103: Captura de vista de categorías al seleccionar una categoría

Se preguntará si está seguro y haremos click en sí. En el caso de que tengamos un modelo asociado, también nos lo dirá. Elegimos otra categoría para demostrarlo (ilustración 103).

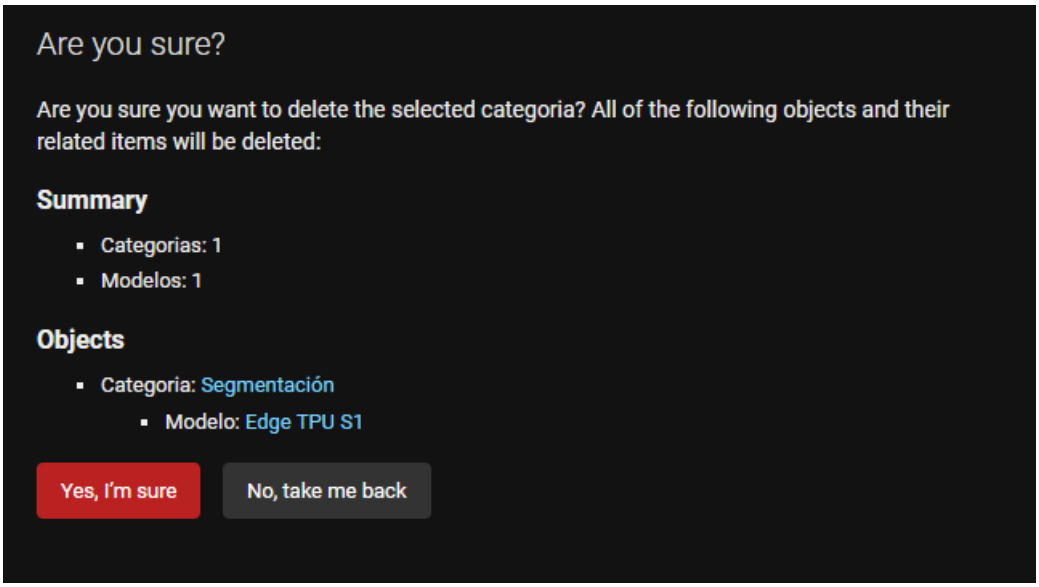


Ilustración 104: Captura de ventana de emergente para confirmar si se desea borrar

Algo muy importante en tener en cuenta es no borrar ni editar el nombre de ninguna de las 4 categorías que ya están creadas, no se puede cambiar ni una tilde porque si no dará error el código.

La base de datos o dataset sigue la misma lógica que las categorías por lo que pasaremos directamente a los modelos.

En la interfaz de la aplicación, al pinchar en una categoría que no tiene ningún modelo asociado (por tanto, ninguna base de datos asociada) nos llevará a una pantalla vacía. A continuación, vamos a crear un modelo que nos asigne una base de datos con una categoría. Si no está creada la base de datos no pasa nada, en el momento de crear el modelo se podrá crear.

Para ver los diferentes modelos, hacemos click en el menú de la izquierda en "Modelos" y veremos lo siguiente (ilustración 104):

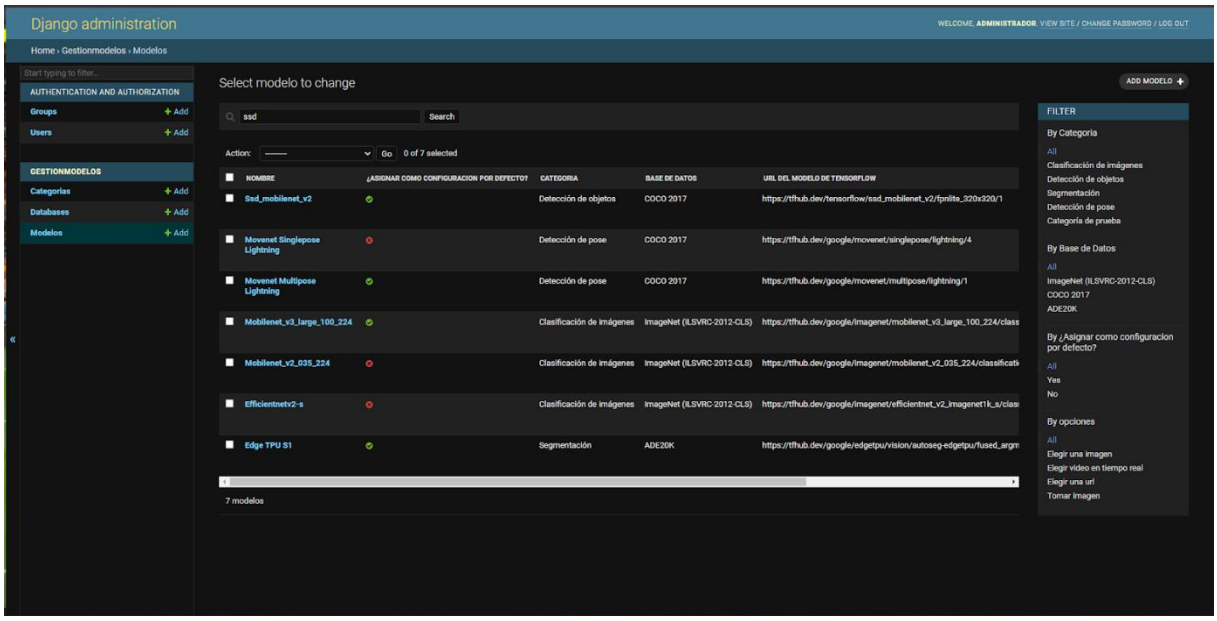


Ilustración 105: Captura de vista de modelos en la parte de administrador

Los modelos no se seleccionan en la interfaz, esto es trabajo del administrador, cada columna son datos que hay que añadir para que funcione el código. Además, a la derecha tenemos varios filtros para encontrar el modelo que queremos con más facilidad. Para añadir un modelo funcional, tendremos que pasar al siguiente apartado.

11.3.4. Insertar modelo funcional

Ahora vamos a insertar un modelo distinto a los que ya hay insertados en la aplicación. Para insertar un modelo funcional, es decir, que funcione correctamente, hay que insertar un modelo cuyo código esté ya implementado dentro de la aplicación. Para implementar un modelo funcional se pueden dar distintas situaciones:

- **Importar un modelo funcional con una base de datos o una categoría nueva.** Si queremos añadir un modelo con una categoría o una base de datos nueva habría que cambiar el código y añadirlo manualmente para que funcione, no puede ser realizado en la vista de administrador.

- **Importar un modelo con la misma base de datos y con la misma categoría que ya hay creada, pero con distintos parámetros de**

entrada y salida. Los parámetros de entrada y salida se pueden mirar en TensorFlow Hub comparándolos con el modelo que tiene por defecto la aplicación para esa categoría y esa base de datos. Si los parámetros son distintos, la aplicación no funcionará, por eso, si al insertar un modelo de este tipo no funciona, esta será la razón. Ante esta situación, será necesario añadirlo al código.

- **Importar un modelo con la misma base de datos y con la misma categoría que ya hay creada y tiene los mismos parámetros de entrada y salida.** Estos son los modelos con los que se puede ir alternando, ahora se explicará con más detalle.

11.3.4.1. TensorFlow Hub

En este apartado vamos a ver TensorFlow Hub y cómo buscar en él modelos funcionales para la aplicación (ilustración 105). Para ello, entramos en el siguiente enlace.

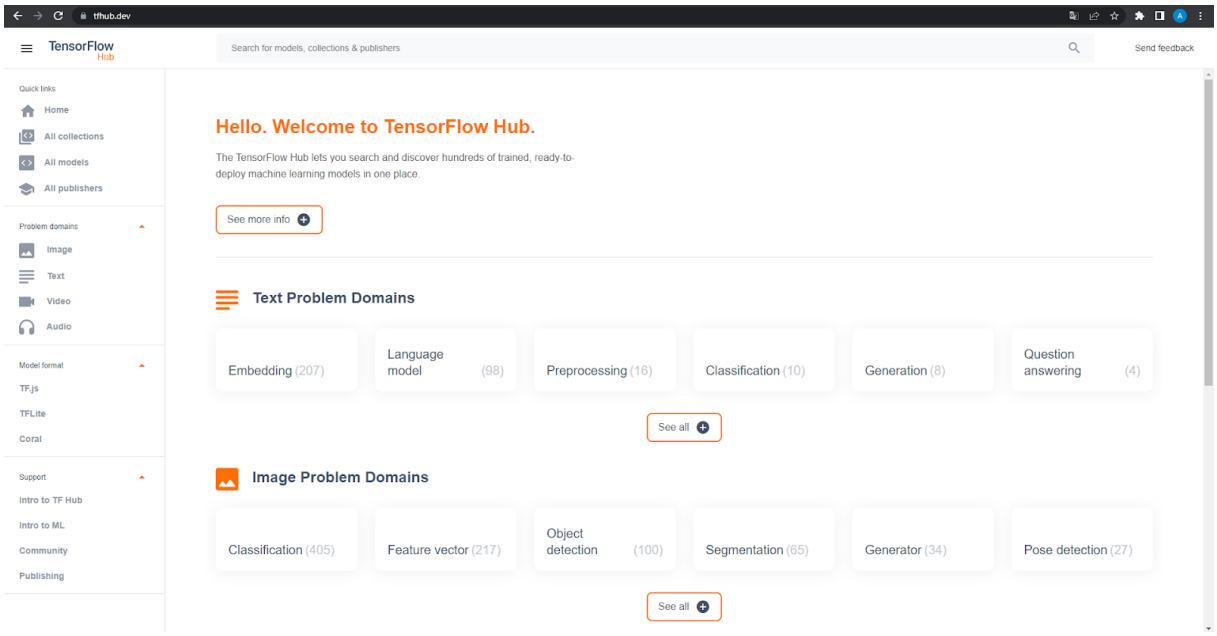


Ilustración 106: Captura de página de inicio de TensorFlow Hub

A la izquierda, podemos ver varios filtros y en la parte superior encontramos un buscador para poder encontrar el modelo según el nombre específico. Por ejemplo,

vamos a filtrar para ver los modelos de clasificación de imágenes, para ello haremos click en Image situado a la izquierda del menú (ilustración 106):

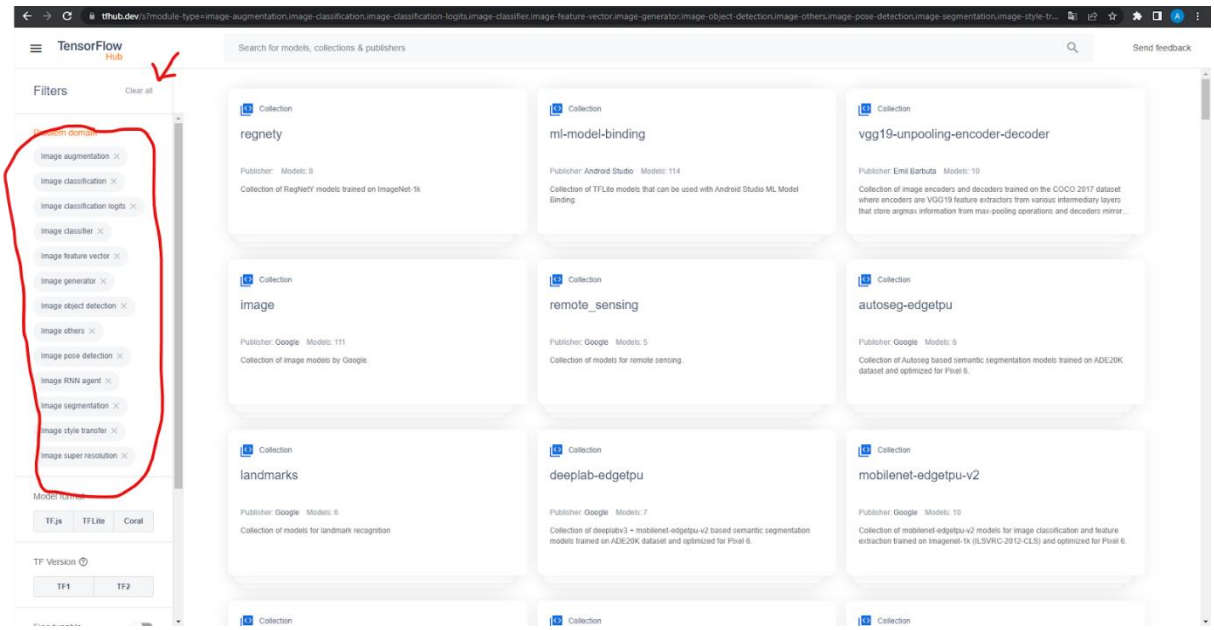


Ilustración 107: Captura de página TensorFlow Hub con el filtro del dominio de imágenes

La propia página te selecciona varios filtros automáticamente, para eliminarlos, haremos click en “Clear All” y nos saldrá la siguiente pantalla (ilustración 107):

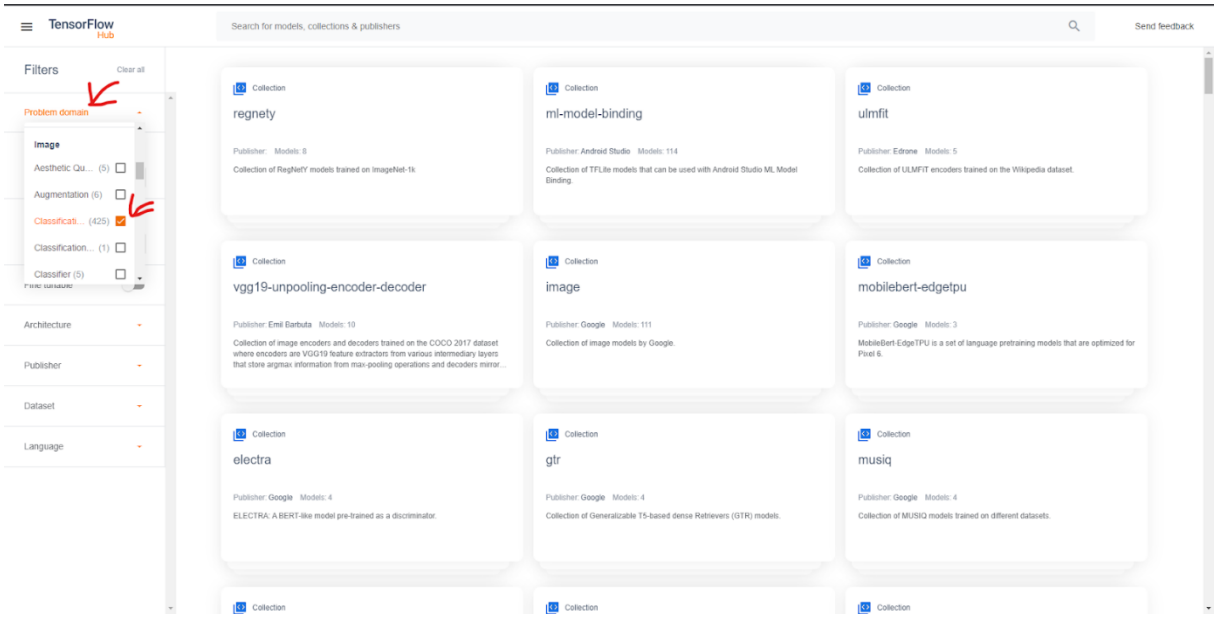


Ilustración 108: Captura de página TensorFlow Hub con el filtro del dominio de imágenes y todos los filtros borrados

A continuación, clicamos en el desplegable “Problem domain”, y deslizamos la barra de la derecha hasta encontrar la palabra “Image” y marcamos “Classification” que es una de las categorías que ya tenemos en la aplicación. Para aplicar el filtro pinchamos fuera y este se aplicará automáticamente.

Un poco más abajo, marcaremos el filtro “TF2” en “TF Version” y podremos ver todos los modelos con un símbolo de color naranja (ilustración 108).

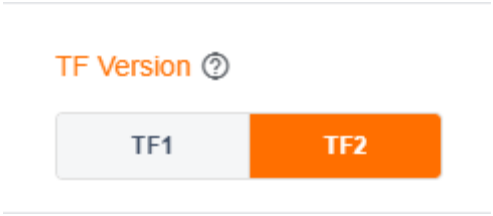


Ilustración 109: Captura de página TensorFlow Hub del filtro TF Version

Como podemos ver, hay algunos modelos que nos aparece la base de datos designada con la palabra dataset (ilustración 109).

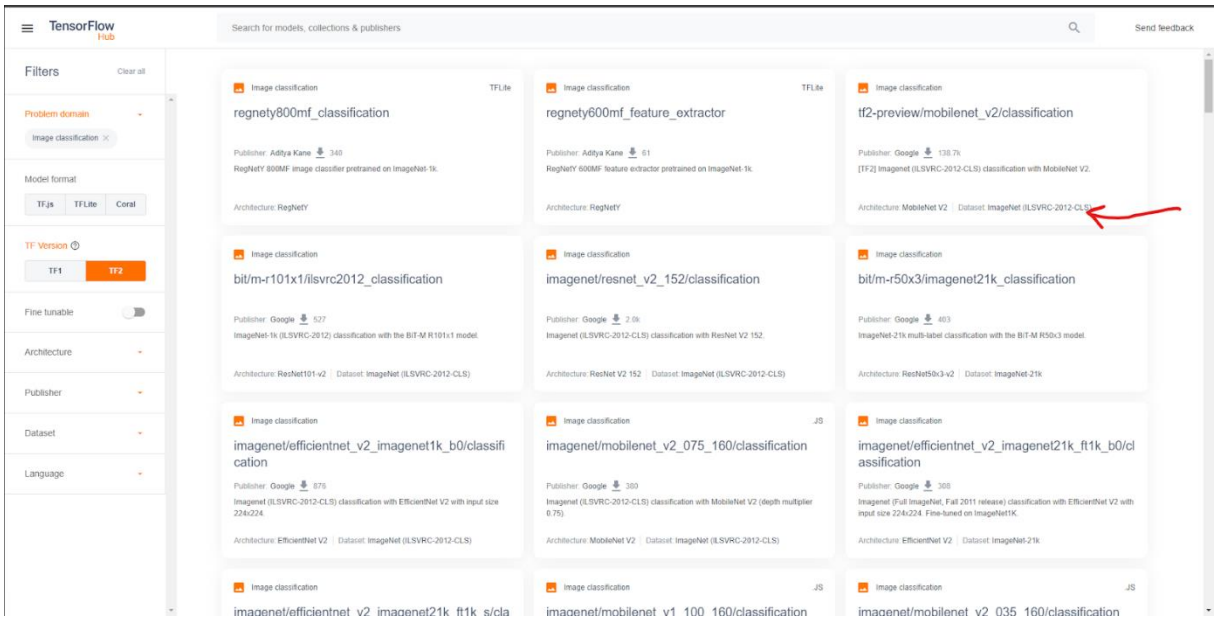


Ilustración 110: Captura de página TensorFlow Hub señalando donde aparece el dataset de un modelo

Como podemos ver, hay algunos modelos que nos aparece la base de datos designada con la palabra dataset. La base de datos que usamos en la aplicación es ImagiNet, por tanto, hemos encontrado un modelo donde la categoría y la base de datos con la que ha sido entrenado es la misma. Por eso, necesitaremos comparar el modelo nuevo con el modelo que tenemos predeterminado (se encuentra en la vista de administrador marcado con un tick verde, el resto saldrán de color rojo con una "x").

En primer lugar, tendremos que ir a la vista de administrador y buscar en los modelos cuál es el que está como predeterminado, una vez hecho esto copiamos el nombre del modelo y lo buscamos en la página de TensorFlow Hub. Por último, tendremos que comparar los códigos de ambos modelos en la página web anteriormente mencionada. En el ejemplo, hemos escogido el modelo imagenet/efficientnet_v2_imagenet21k_ft1k_b0/classification que se encuentra en la izquierda de la imagen que vamos a comparar con el que nosotros tenemos como predeterminado que queda en la derecha de la imagen (ilustración 110).

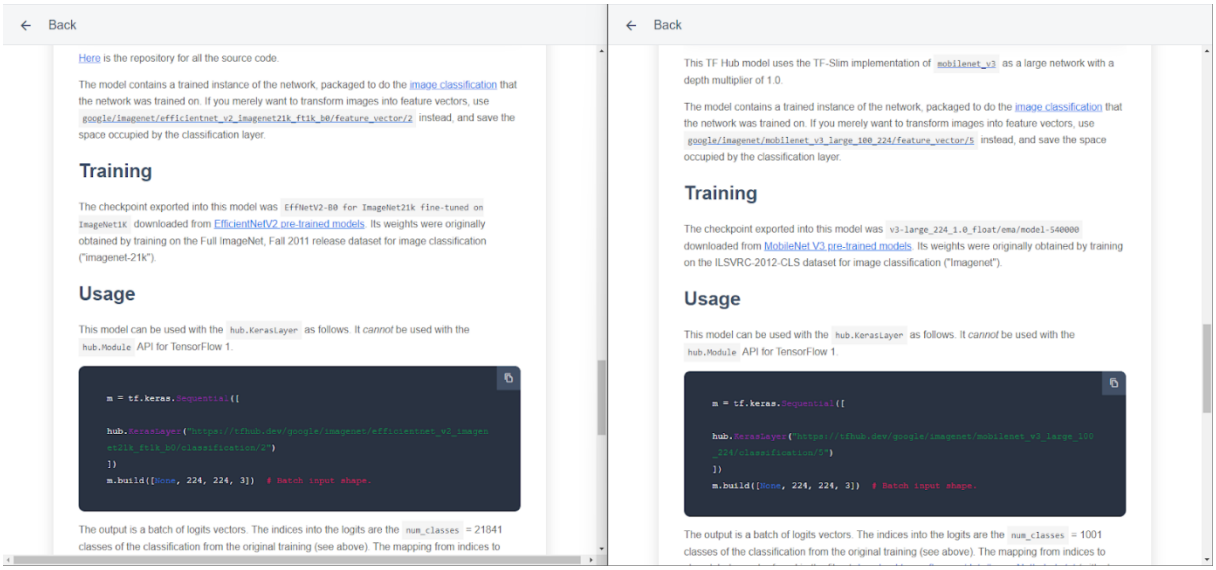


Ilustración 111: Captura de página TensorFlow Hub al haber seleccionado un modelo

Observamos que el código es exactamente igual. Por tanto, este modelo podrá ser usado. Si observamos bien el código, encontramos el número 224, este es el parámetro de altura y anchura de la imagen de entrada. Estos valores son los únicos que pueden ser distintos a la original, junto con el url del modelo, que es lo que

podemos ver de color verde. Pero estos valores no pueden ser aleatorios, en esta misma página podemos ver para cada modelo su ancho y su alto (Esto lo veremos en el siguiente apartado).

11.3.4.2. Insertar modelo en la aplicación

Una vez que hemos elegido el modelo, vamos a insertarlo en la aplicación y para ello, vamos a abrir en una ventana la aplicación con el modo de administrador y en la otra TensorFlow Hub. Una vez en la ventana de modelos, hacemos click en “Add modelo” (ilustración 111):

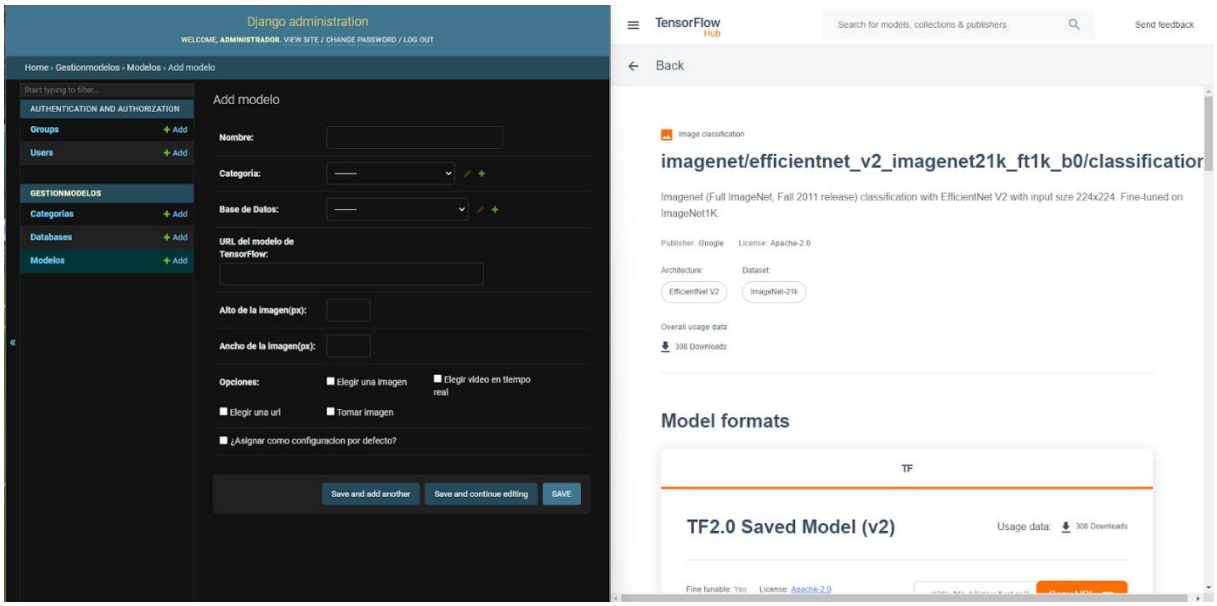


Ilustración 112: Captura de vista añadir un modelo de la parte de administrador y la página TensorFlow Hub al haber seleccionado un modelo

Vamos a ver atributo por atributo (ilustración 112):

- Nombre: el nombre no es importante, pero sirve para identificarlo de los demás y no se podrá repetir.
- Categoría y Base de Datos: aquí seleccionamos las ya creadas, en este caso, Clasificación de Imágenes e ImagiNet respectivamente.
- URL del modelo de TensorFlow: en TensorFlow, el URL lo podemos encontrar tanto en el código (como hemos visto en la imagen en la que

hemos comparado modelos) como pulsando en el botón “Copy URL” de color naranja.

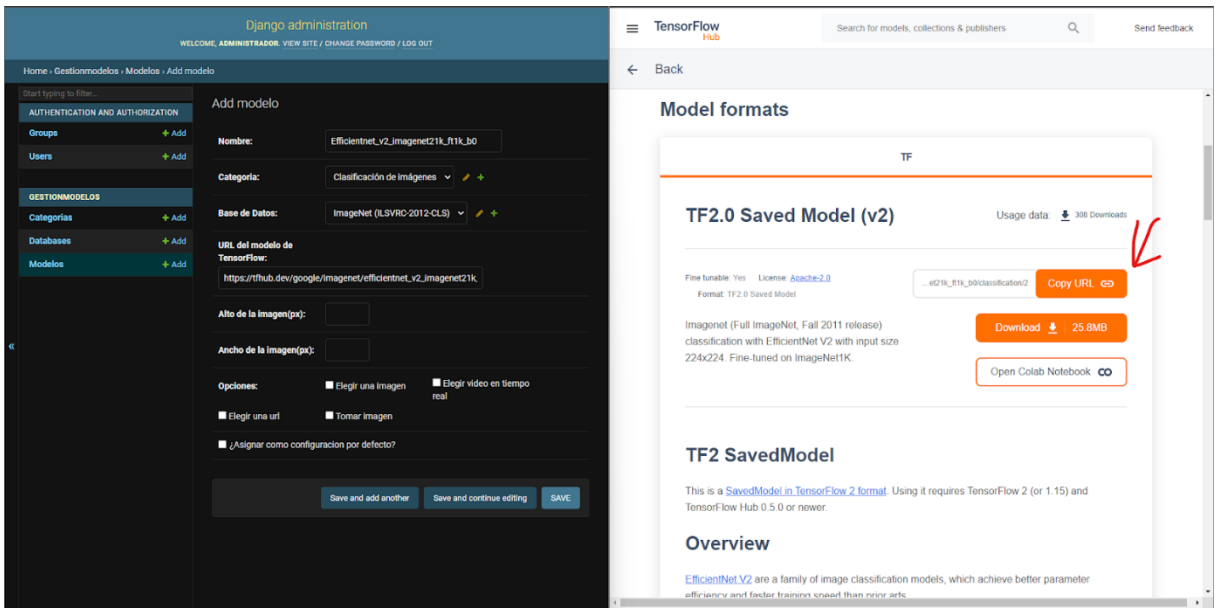


Ilustración 113: Captura de vista añadir un modelo de la parte de administrador relleno y la página TensorFlow Hub al haber seleccionado un modelo señalando el botón Copy URL

- Alto y ancho: lo podemos encontrar leyendo en la página del modelo o en el código (marcado en la imagen inferior) (ilustración 113).

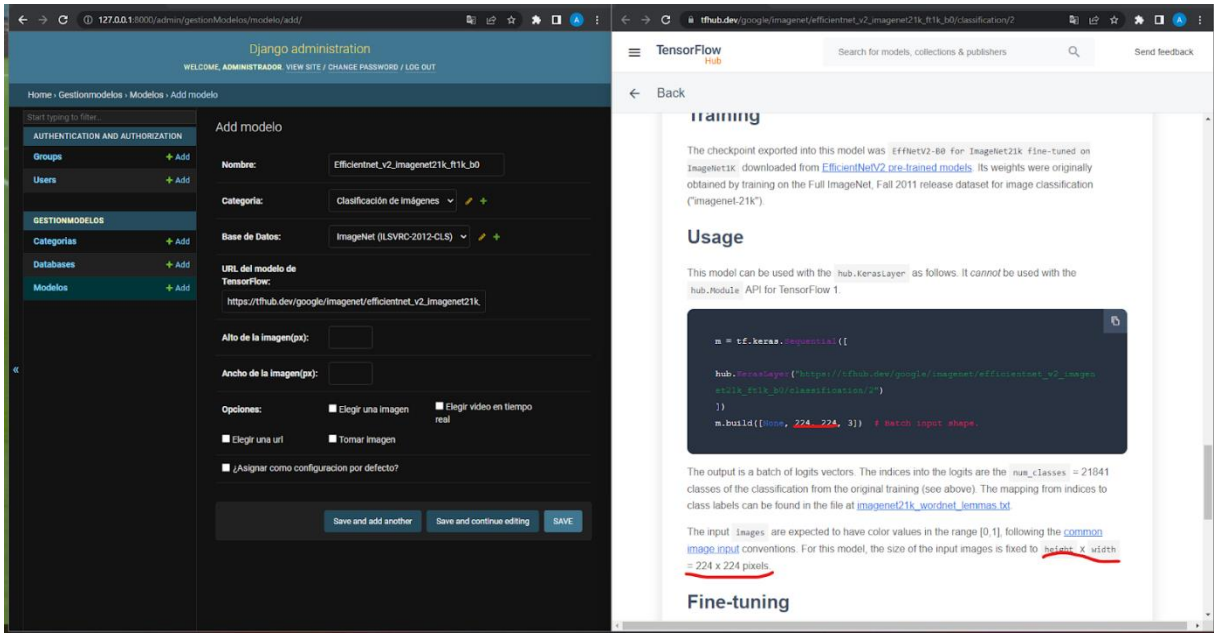


Ilustración 114: Captura de vista añadir un modelo de la parte de administrador relleno y la página TensorFlow Hub al haber seleccionado un modelo señalando el ancho y alto

- Opciones: en este apartado hay que poner las mismas que el modelo que ya está insertado (ilustración 114).

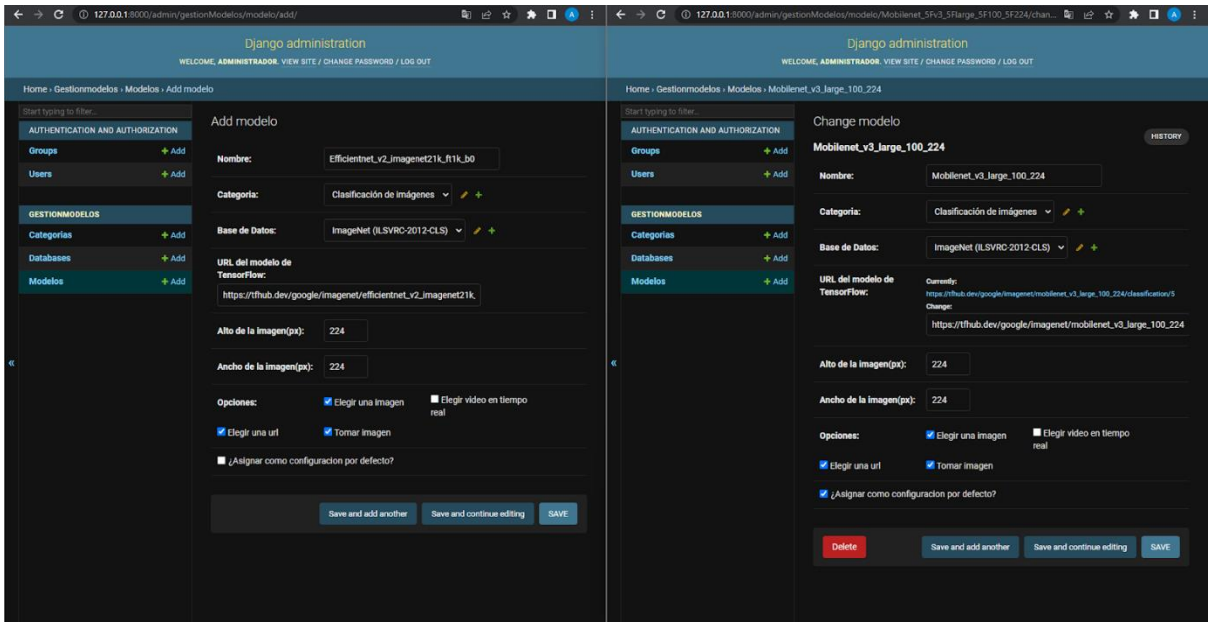


Ilustración 115: Captura comparando las opciones para que sean iguales

- Asignar como configuración por defecto: la aplicación coge el modelo por defecto, el usuario no puede elegir el modelo a usar, por tanto, entre todos los que tiene el modo administrador elige el que tenga marcado por defecto. Ya hay un modelo marcado por defecto, por tanto, al añadir uno nuevo y marcar esta opción, el nuevo modelo se guardará correctamente y esta opción quedará desmarcada. Se tiene primero que desmarcarla del modelo antiguo y marcarla en el nuevo. Para editar un modelo, lo buscamos y hacemos click en el que queremos editar, hacemos los cambios que deseamos y guardamos.

Algo importante a tener en cuenta es que si no hay marcado para cada categoría/base de datos un modelo por defecto, dará error.

Al filtrar a la derecha de los modelos podemos ver que está activado el nuevo modelo que acabamos de añadir y el resto están desactivados (ilustración 115).

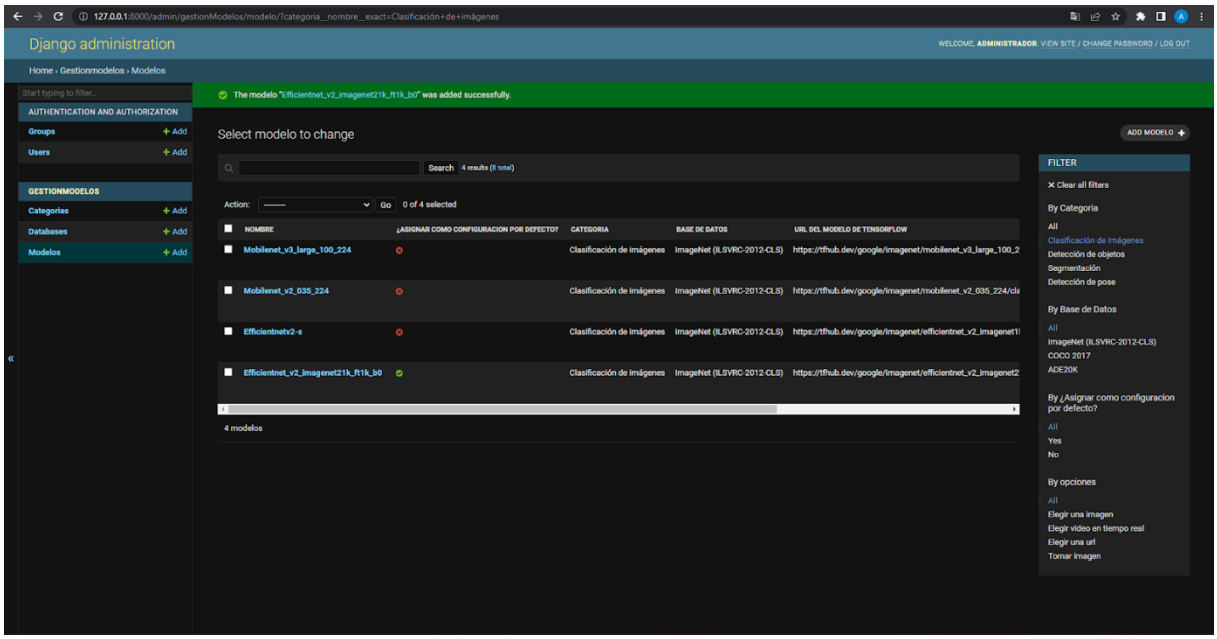


Ilustración 116: Captura de la vista de modelos en la parte de administrador con el nuevo modelo

Por último, en la aplicación, si seleccionamos la categoría “Clasificación de imágenes” y base de datos “ImagiNet” podremos ver si es realmente un modelo funcional y es capaz de predecir una imagen. Para probarlo elegiremos una imagen de las que trae la aplicación por defecto. Puede tardar un par de segundos en cargar la aplicación, esto significa que está cargando el nuevo modelo por defecto que hemos añadido (ilustración 116).

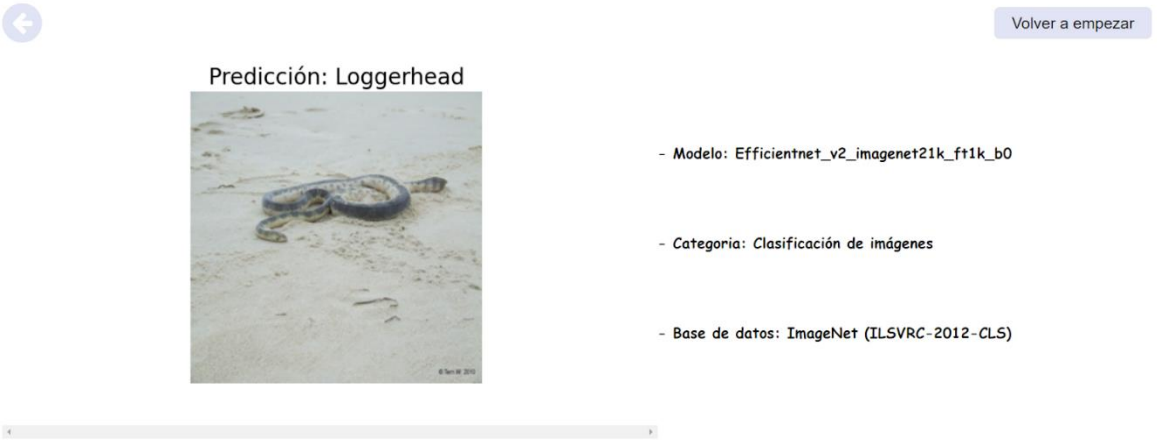


Ilustración 117: Captura de la vista de resultados al haber utilizado el nuevo modelo

11.3.5. Consejo

Algunos de los modelos funcionales de la aplicación traen un archivo donde se pueden probar los modelos, en estos archivos de código normalmente no solo viene el modelo desde el que hemos abierto dicho archivo, además suelen venir varios modelos más que comparten los mismos elementos (mencionados a lo largo de la guía) con el que está implementado en la aplicación. Estos modelos alternativos aparecen con su nombre, el URL, el ancho y el alto. Esto nos puede servir para buscar un modelo más funcional que el que ya tenemos de forma mucho más rápida. Para ilustrar este consejo veamos un ejemplo:

En la siguiente imagen (ilustración 117) seguimos en la página del modelo imagenet/efficientnet_v2_imagenet21k_ft1k_b0/classification, justo debajo del botón donde hemos copiado el URL del modelo hay un botón llamado “Open Colab Notebook”.

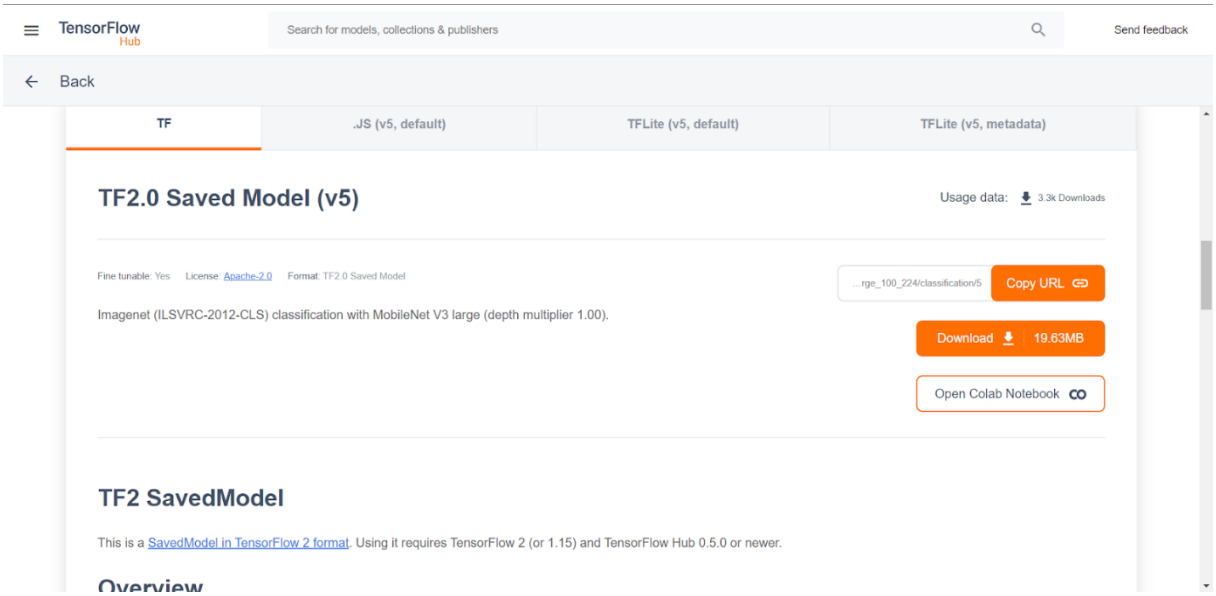


Ilustración 118: Captura de página de TensorFlow Hub al elegir un modelo que tiene asociado un notebook de Colab asociado

Al pulsarlo nos llevará a un notebook de Collab donde hay código para probar el modelo, no es necesario saber programar, solo echando un vistazo al código se pueden diferenciar los modelos. En nuestro ejemplo si bajamos un poco podemos ver el nombre y el enlace (ilustración 118).

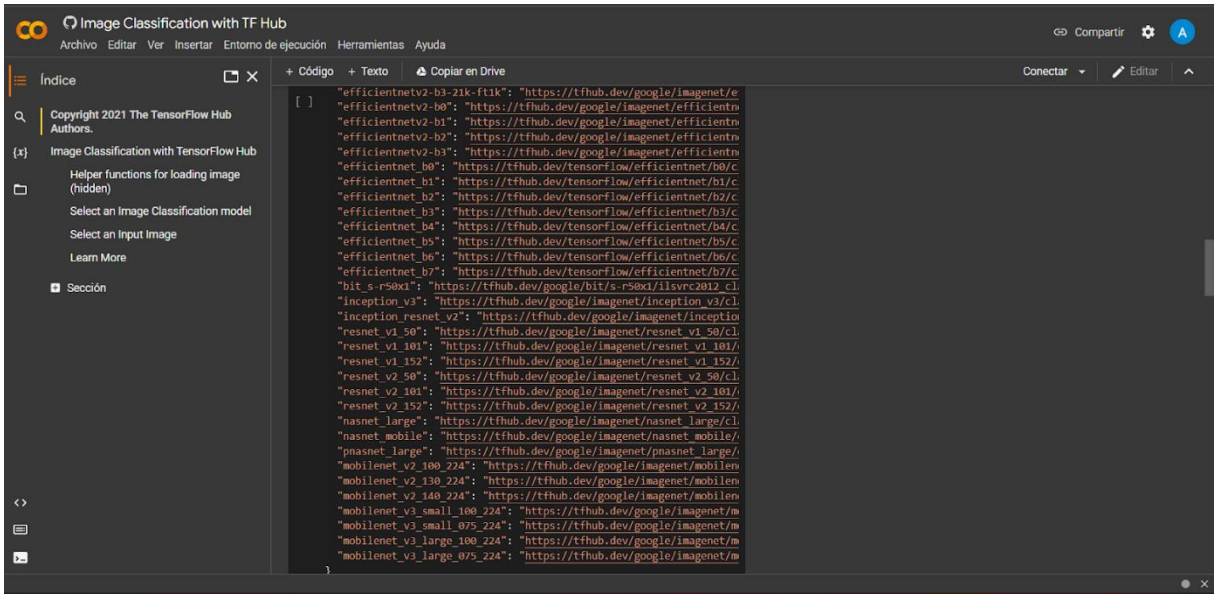


Ilustración 119: Captura de Colab con modelos de TensorFlow Hub (1)

Y un poco más abajo, el nombre con la altura o anchura (ya que el valor de la anchura y el de la altura es el mismo, aunque el valor cambia según el modelo) (ilustración 119).

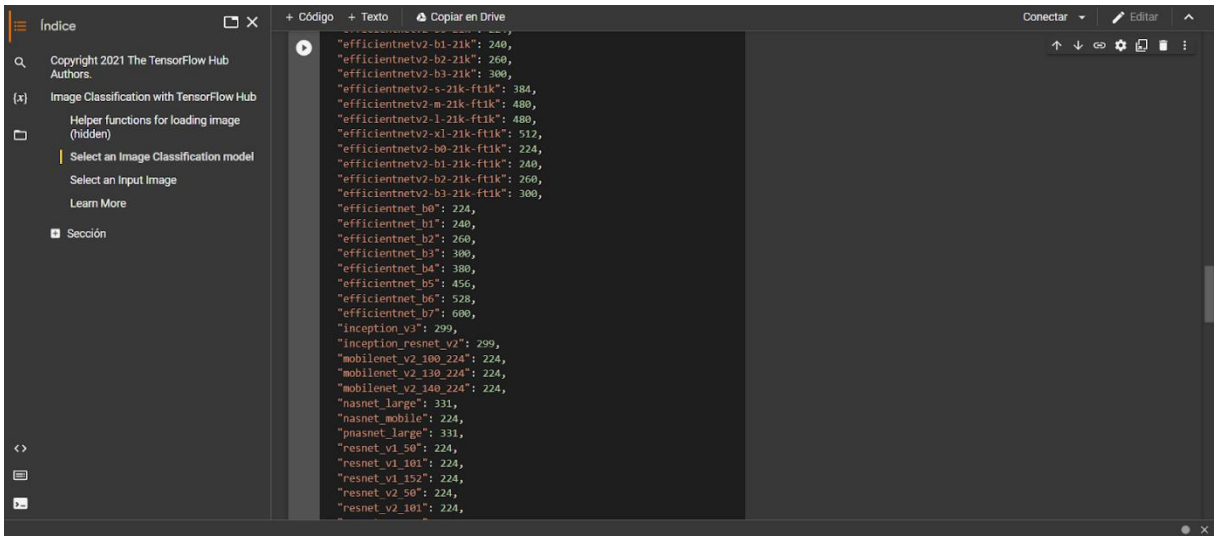


Ilustración 120: Captura de Colab con modelos de TensorFlow Hub (2)

Con estos datos ya podemos crear un modelo funcional y además, disponemos de otros muchos modelos funcionales alternativos para poder encontrar el más preciso, si así lo deseamos.

12. DEFINICIONES Y ABREVIATURAS

12.1. Siglas

- **TFG**: Trabajo Fin de Grado
- **CNN**: Convolutional Neural Networks
- **PLN**: Procesamiento del Lenguaje Natural
- **RNN**: Recurrent neural networks o redes neuronales recurrentes
- **GAN**: Generative Adversarial Networks o redes generativas antagónicas
- **RRSS**: Redes sociales
- **API**: Application Programming Interfaces o Interfaz de Programación de Aplicaciones
- **FPS**: Frames Per Second o imágenes por segundo
- **HTML**: HyperText Markup Language o Lenguaje de Marcas de Hipertexto
- **CSS**: Cascading Style Sheets o Hojas de Estilo en Cascada
- **MVC**: Modelo-Vista-Controlador
- **MVT**: Modelo-Vista-Template
- **IDL**: Interface definition language o Lenguaje de descripción de interfaz
- **VS**: Visual Studio
- **GNU**: acrónimo recursivo de «GNU No es Unix»
- **PC**: Personal Computer
- **PIL**: Python Imaging Library
- **AMD**: Advanced Micro Devices

- **CUDA**: Compute Unified Device Architecture o Arquitectura Unificada de Dispositivos de Cómputo
- **GPU**: Graphics Processing Unit y Unidad de Procesamiento Gráfico
- **cuDNN**: CUDA Deep Neural Network
- **COCO**: Common Objects in Context
- **ILSVRC**: ImageNet Large Scale Visual Recognition Challenge
- **CUN**: Caso de uso de Usuario Normal
- **CUA**: Caso de uso de Usuario Administrador
- **USB**: Universal Serial Bus

12.2. Referencias

1. González, L. (2021, marzo 30). *Inteligencia Artificial vs Machine Learning vs deep learning*. 🤖 Aprende IA. https://aprendeia.com/inteligencia-artificial-vs-machine-learning-vs-deep-learning/

2. Portal, T. I. C. (2021, abril 5). *Deep learning: ¿se pueden programar las máquinas para pensar como humanos?* TIC Portal. https://www.ticportal.es/glosario-tic/deep-learning-dl

3. *¿Cómo funcionan los Transformers? en Español*. (2022, noviembre 8). Aprende Machine Learning. https://www.aprendemachinelearning.com/como-funcionan-los-transformers-espanol-nlp-gpt-bert/

4. *Machine Learning e IA*. (s. f.). Amazon.com. https://aws.amazon.com/es/what-is/neural-network/

5. *¿Qué es Java Spring Boot?* (s. f.). Microsoft.com. https://azure.microsoft.com/es-mx/resources/cloud-computing-dictionary/what-is-java-spring-boot/

6. Tokio School. (2022, septiembre 19). *¿Qué es Angular? El framework de JavaScript para crear aplicaciones web dinámicas*. Tokio School. https://www.tokioschool.com/noticias/que-es-angular/

7. *¿Qué sigue después de aprender Python? Django vs. Flask vs. FastAPI*. (2021, octubre 26). Platzi. https://platzi.com/blog/django-flask-fastapi/

8. Flores, F. (2022, julio 22). Qué es Visual Studio Code y qué ventajas ofrece. *Openwebinars.net*. https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/

9. *¿Qué es Docker?* (s. f.). Amazon.com. https://aws.amazon.com/es/docker/

10. *NVIDIA CUDA deep neural network (cuDNN)*. (2014, septiembre 2). NVIDIA Developer. https://developer.nvidia.com/cudnn

11. *Metodologías de desarrollo de software: ¿qué son?* (s. f.). Becas-santander.com. Recuperado 10 de mayo de 2023, de http://becas-santander.com/es/blog/metodologias-desarrollo-software.html

12. Meel, V. (2023, enero 1). *What is the COCO Dataset? What you need to know in 2023*. Viso.Ai. https://viso.ai/computer-vision/coco-dataset/

13. Jesús. (2019, octubre 19). *Qué es ImageNet y Por Qué Necesitas Conocerlo*. DataSmarts. https://www.datasmarts.net/que-es-imagenet-y-por-que-necesitas-conocerlo/

14. *ADE20K*. (s. f.). Mit.edu. https://groups.csail.mit.edu/vision/datasets/ADE20K/