



XUNTA
DE GALICIA

CONSELLERÍA DE CULTURA,
EDUCACIÓN, FORMACIÓN
PROFESIONAL E UNIVERSIDADES



IES SAN MAMEDE

© Rúa do Castelo N° 3. 32700. Maceda (Ourense)
Tlf: 988 788 561 ✉ ies.san.mamede@edu.xunta.gal
🌐 <http://www.iesanmamede.com>

Detector IA



Alumno: Iván Baños Piñeiro

Curso: 2º DAM

Módulo: Proyecto Final Ciclo

Github: https://github.com/IvanBanhosPinheiro/Detector_Ia_Trabajo_Final

01/04/2025

Índice:

1. Introducción:	5
2. Objetivos:	5
2.1. Objetivo general:	5
2.2. Objetivos específicos:	5
3. Situación previa:	6
4. Tecnologías y Bibliotecas Utilizadas:	6
5. Solución propuesta:	8
5.1. Cliente	8
5.2. Servidor	8
5.3. Características destacadas de la solución	8
6. Planificación del proyecto:	9
6.1. Modelo de desarrollo	9
6.2. Fases del proyecto	9
6.2.1. Análisis de requerimientos (Semana 1)	9
6.2.2. Diseño (Semanas 2-3)	9
6.2.3. Implementación (Semanas 4-7)	9
6.2.4. Pruebas (Semanas 8-9)	9
6.2.5. Despliegue (Semana 10)	9
6.2.6. Mantenimiento y mejoras (Semana 11 en adelante)	9
6.3. Diagrama de Gantt	10
6.4. Diagramación	10
6.4.1. Universo de la base de datos	10
6.4.1.1. Tablas de la base de datos:	11
6.4.1.2. Relación entre las tablas	11
6.4.2. Diagrama de la ER/EER	12
6.4.3. Modelo relacional	13
6.4.4. Diagrama de clases	14
6.4.5. Diagramas de casos de uso	14
6.4.5.1. Usuario	15
6.4.5.2. Cliente	16
6.4.6. Diagramas de secuencia	18
6.4.6.1. Cliente-servidor	18
6.4.6.2. Profesor-servidor	19
6.4.6.3. Administrador-servidor	20
6.4.6.4. Interno del servidor	21
6.5. Diseño de la interfaz de usuario	22
6.5.1. Principios de diseño utilizados:	22
6.5.2. Paleta de colores y estilo visual:	22
6.5.3. Tipografía utilizada:	22
6.5.4. Iconografía:	22
6.5.5. Mockups y estructura de navegación:	23

6.5.5.1.Index.....	23
6.5.5.2.Login.....	23
6.5.5.3.Panel de Control (administrador).....	24
6.5.5.4.Usuarios.....	25
6.5.5.5.Editar Keywords.....	25
6.5.5.6.Horarios.....	26
6.5.5.7.Perfil del usuario.....	26
6.6.Presupuesto del proyecto:.....	27
6.6.1 Coste del Hardware empleado.....	27
6.6.2 Software y herramientas empleadas.....	27
6.6.3 Total general.....	28
7. Desarrollo y ejecución.....	29
7.1.Descripción del diagrama.....	29
7.1.1.Cliente:.....	29
7.1.2.Servidor:.....	30
7.1.3.Interfaz web:.....	30
8.Validación, accesibilidad y testing.....	30
8.1.Métodos utilizados:.....	30
8.2.Resultados de la validación y mejoras aplicadas:.....	31
8.3.Conclusión:.....	31
9.Conclusiones, propuestas de mejora y valoración personal.....	31
9.1Conclusiones generales:.....	31
9.2.Problemas encontrados y soluciones:.....	32
9.3.Propuestas de mejora:.....	32
Bibliografía.....	33
Anexo I: Manual de instalación.....	35
1.Requisitos técnicos.....	35
2.Instalación del software.....	35
2.1.Servidor:.....	36
2.1.1.Opción 1: Anaconda (entorno virtual).....	36
2.1.2.Opción 2: Docker (recomendable).....	39
2.1.2.1.Con deploy.bat.....	40
2.1.2.2.Con docker-compose.....	42
2.2.Cliente.....	43
2.2.1.Anaconda (entorno virtual).....	43
2.2.2.Instalador (recomendable).....	45
3.Despliegue del sistema.....	48
3.1.Servidor.....	48
3.2.Cliente.....	49
Anexo II: Manual de usuario.....	50
1. Arquitectura general del sistema.....	50
2.Acceso y autenticación.....	50
3.Funciones según el rol del usuario.....	51



3.1. Para administradores.....	51
3.1.1. Gestión de usuarios:.....	51
3.1.2. Configuración de horarios:.....	53
3.1.3. Visualización de capturas:.....	56
3.1.4. Editar los keywords:.....	59
3.2. Para profesores.....	60
4. Cierre de sesión.....	60

1. Introducción:

En los últimos años, la inteligencia artificial ha transformado numerosos ámbitos, incluido el educativo. Herramientas como Chat GPT, Copilot o buscadores potenciados por IA están cada vez más al alcance de los estudiantes, lo que supone un nuevo desafío en contextos de evaluación.

El objetivo de este proyecto es dar respuesta a esa situación, mediante el desarrollo de un sistema de vigilancia capaz de detectar indicios del uso de IA en tiempo real durante exámenes o clases.

El sistema se basa en una arquitectura cliente-servidor, en la que los equipos de los alumnos envían capturas al servidor únicamente cuando se detecta un comportamiento sospechoso, como el uso de ciertas aplicaciones o palabras clave. Esto permite a los profesores revisar evidencias visuales de forma rápida y ordenada.

La solución se ha diseñado con tecnologías open source, como Flask, Tesseract OCR y PyAutoGUI, y con un enfoque modular, escalable y respetuoso con la privacidad de los usuarios. Además, su configuración mediante ficheros INI y su despliegue vía Docker o Anaconda permiten una implementación sencilla y flexible en distintos entornos educativos.

2. Objetivos

2.1. Objetivo general:

- Desarrollar una herramienta capaz de monitorizar el uso de inteligencia artificial en ordenadores de aula, mediante capturas de pantalla automáticas y análisis de texto, controladas por un sistema centralizado.

2.2. Objetivos específicos:

- Implementar un cliente ligero que detecte ventanas activas y realice capturas cuando se identifiquen patrones de uso de IA.
- Diseñar un servidor que reciba, almacene y clasifique las capturas asociadas a cada profesor y horario.
- Permitir que los profesores gestionen manualmente el estado del sistema durante su sesión (modo automático/manual).
- Proporcionar al administrador un panel para controlar usuarios, horarios y ajustes generales del sistema.
- Garantizar la seguridad, integridad y acceso limitado a los datos según los roles definidos.
- Ofrecer una interfaz sencilla, accesible y compatible con múltiples resoluciones.
- Facilitar la instalación del sistema mediante Docker o entornos virtuales (Anaconda).

3. Situación previa

A día de hoy, el uso de herramientas de inteligencia artificial en entornos educativos, especialmente durante exámenes o clases online, ha crecido de forma exponencial. Esto plantea un gran problema: los estudiantes pueden acceder a estas tecnologías sin ser detectados, comprometiendo así la integridad académica.

Aunque existen aplicaciones para gestionar aulas o bloquear el acceso a ciertos programas, estas soluciones suelen ser costosas, difíciles de implementar, o no ofrecen suficiente control en tiempo real. Además, muchas están pensadas para entornos muy específicos o requieren una infraestructura compleja.

Este proyecto surge como respuesta a esa necesidad: una solución efectiva, ligera y modular que permita detectar el uso de IA en ordenadores de clase, con capturas automáticas controladas desde un servidor central.

Al iniciar este desarrollo, no contaba con una alternativa que se ajusta completamente a mis necesidades, ni a nivel técnico ni económico. Las opciones comerciales existentes no permitían un control personalizado sobre las capturas, ni diferenciaban claramente entre profesores y administradores, y en muchos casos requerían instalación de software propietario o dependencias cerradas.

Por tanto, decidí crear un sistema propio con herramientas open source y configuraciones fácilmente replicables. Esto no solo permite resolver el problema de vigilancia en clase, sino también garantizar que cada profesor pueda acceder exclusivamente a sus capturas, mientras que el administrador conserva el control global de usuarios y horarios.

4. Tecnologías y Bibliotecas Utilizadas

La aplicación ha sido desarrollada empleando una combinación de tecnologías backend y frontend, así como librerías orientadas a la gestión de procesos automatizados y tratamiento de imágenes. El backend está construido sobre Flask, un micro framework web en Python que permite la definición de rutas, la gestión de peticiones HTTP y el renderizado de interfaces HTML mediante el motor de plantillas Jinja2. Además, se ha empleado el sistema de Blueprints para organizar la aplicación de forma modular, separando la lógica por funcionalidades como autenticación, control de capturas o gestión de horarios.

Para implementar el sistema de autenticación, se ha utilizado Flask-Login, que facilita el manejo de sesiones de usuario, protegiendo rutas mediante decoradores y proporcionando una interfaz sencilla para controlar el inicio y cierre de sesión. La gestión de usuarios está completamente integrada con el sistema ORM Flask-SQLAlchemy, que permite trabajar con una base de datos relacional sin necesidad de escribir consultas SQL manuales. Los modelos definidos en este ORM representan entidades como usuarios, equipos, capturas y horarios.

La base de datos empleada es SQLite, un motor ligero y embebido que guarda toda la información en un único archivo. Aunque resulta muy práctico en entornos pequeños, sus limitaciones ante múltiples escrituras simultáneas han llevado a complementar su uso con un sistema de cola basado en la librería estándar `queue` de Python. Este sistema permite serializar las escrituras en base de datos, evitando bloqueos por concurrencia. Además, se ha incorporado el módulo `sqlalchemy.exc` para detectar excepciones como errores operacionales o integridad referencial, lo que permite implementar mecanismos de recuperación en caso de colapso de la base de datos.

Para reforzar la seguridad del sistema, especialmente en el tratamiento de contraseñas, se ha integrado Werkzeug, que permite generar hashes seguros y verificar credenciales de forma confiable. La configuración de la aplicación, tanto del servidor como del cliente, se gestiona a través de archivos `.ini`, leídos mediante la librería `ConfigParser`, lo que aporta flexibilidad sin necesidad de modificar el código fuente.

En la parte del cliente, la detección del uso de inteligencia artificial o software indebido durante los exámenes se lleva a cabo mediante capturas de pantalla y análisis OCR. Las capturas se realizan utilizando PyAutoGUI, mientras que el procesamiento de imágenes y la preparación para su análisis se gestionan con Pillow. El reconocimiento de texto en las imágenes se realiza gracias a Pytesseract, un wrapper para Tesseract OCR. Además, para detectar cambios de ventana activa, se hace uso de la biblioteca PyGetWindow, que permite obtener información en tiempo real del título de la ventana en uso.

Las comunicaciones entre el cliente y el servidor se gestionan mediante la librería Requests, que permite enviar peticiones HTTP como la descarga de palabras clave o el envío de alertas. Los datos serializados, como la lista de palabras clave, se almacenan en formato binario utilizando Pickle. El sistema también hace uso de módulos estándar como `os`, para interactuar con el sistema operativo, y `datetime` y `time`, necesarios para registrar horarios, marcar eventos y controlar ciclos de espera.

En el frontend se han empleado tecnologías web estándar. Las páginas HTML se generan desde el backend usando plantillas dinámicas, mientras que el diseño visual se ha definido utilizando CSS, garantizando una interfaz clara y funcional. La interactividad se ha logrado gracias a JavaScript, permitiendo manejar eventos del usuario y actualizar contenido en tiempo real. Para la comunicación asíncrona sin recargar la página se ha utilizado AJAX, implementado a través de la API Fetch de JavaScript.

Gracias a esta combinación de tecnologías, la aplicación es capaz de funcionar en tiempo real, supervisar múltiples equipos, almacenar datos de manera eficiente y ofrecer una interfaz de administración intuitiva tanto para profesores como para el administrador del sistema.

5. Solución propuesta

Para dar respuesta al problema planteado, se propone el desarrollo de un sistema de detección de uso de inteligencia artificial en entornos educativos, compuesto por dos elementos principales: un cliente que se ejecuta en segundo plano en los ordenadores del aula, y un servidor central que recibe, clasifica y gestiona la información recogida.

El sistema se ha diseñado con un enfoque modular y escalable, utilizando tecnologías open source que permiten una instalación sencilla y una gestión eficiente tanto para profesores como para administradores.

5.1. Cliente

- El cliente es una aplicación desarrollada en Python, que se instala en los ordenadores de clase.
- Monitorea continuamente las ventanas activas en el equipo, y cuando detecta palabras clave relacionadas con inteligencia artificial (por ejemplo, “Chat GPT”, “Copilot”, “Bard”, etc.), realiza una captura de pantalla.
- La imagen capturada es procesada mediante Tesseract OCR, que extrae el texto visible en la pantalla.
- Si se confirma la presencia de términos sospechosos, el cliente envía automáticamente la imagen al servidor.
- El sistema puede trabajar en modo automático, según los horarios definidos por el administrador, o en modo manual, activado o desactivado por el propio profesor.

5.2. Servidor

- El servidor está desarrollado con Flask, y recibe las capturas enviadas por los clientes.
- Clasifica cada imagen según el equipo de origen y el horario activo, asignándole al profesor correspondiente.
- Ofrece una interfaz web responsive para profesores y administradores:
- Los profesores pueden ver, descargar y eliminar sus propias capturas.
- Los administradores gestionan usuarios, horarios y configuraciones generales.
- Toda la información se almacena en una base de datos SQLite, con posibilidad de migración futura a otros motores.

5.3. Características destacadas de la solución

- Sistema: Compatible con cualquier entorno Windows.
- Despliegue flexible: Puede instalarse vía ejecutable (cliente_hidden.exe), entorno virtual Anaconda y docker.
- Privacidad y control: Las capturas se realizan únicamente cuando hay sospecha de uso de IA, y se almacenan de forma segura.

- Personalización: Los profesores pueden modificar temporalmente el comportamiento del sistema durante su sesión.
- Escalabilidad: La arquitectura permite añadir nuevos clientes o ampliar funcionalidades en el futuro.

6. Planificación del proyecto

6.1. Modelo de desarrollo

Para llevar a cabo este proyecto se ha seguido un modelo de desarrollo en cascada que permite una organización clara y efectiva. Este modelo se divide en varias fases secuenciales donde cada fase depende del resultado de la anterior.

6.2. Fases del proyecto

6.2.1. Análisis de requerimientos (Semana 1)

- Identificación detallada de requisitos funcionales y no funcionales.
- Determinación de tecnologías a emplear.

6.2.2. Diseño (Semanas 2-3)

- Diagramas de clases y modelo relacional.
- Diseño de base de datos y diagramas EER.
- Diseño de interfaz de usuario mediante mockups.

6.2.3. Implementación (Semanas 4-7)

- Desarrollo del backend con Flask.
- Desarrollo del cliente Python con PyAutoGUI y Tesseract OCR.
- Creación y ajuste del frontend usando Bootstrap.

6.2.4. Pruebas (Semanas 8-9)

- Pruebas unitarias, de integración y de sistema.
- Validación de accesibilidad y usabilidad.

6.2.5. Despliegue (Semana 10)

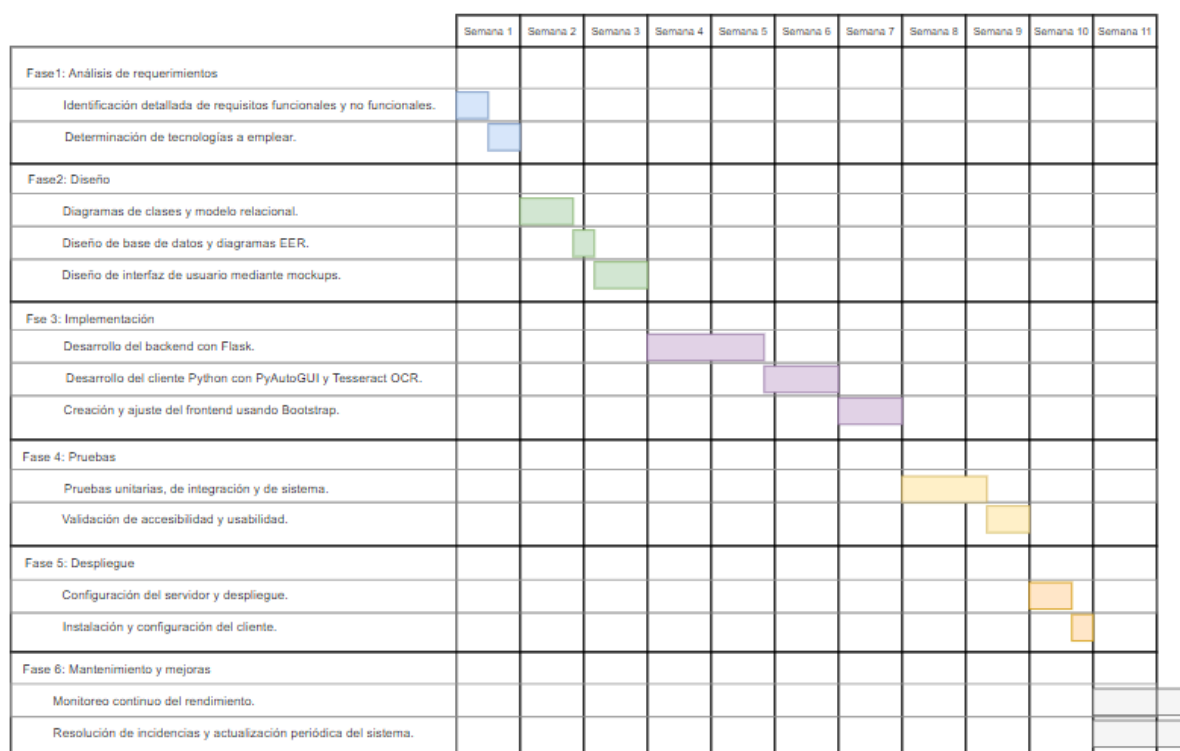
- Configuración del servidor y despliegue.
- Instalación y configuración del cliente.

6.2.6. Mantenimiento y mejoras (Semana 11 en adelante)

- Monitoreo continuo del rendimiento.
- Resolución de incidencias y actualización periódica del sistema.

Esta planificación asegura un desarrollo ordenado, eficiente y con alta calidad del producto final.

6.3. Diagrama de Gantt



6.4. Diagramación

6.4.1. Universo de la base de datos

Este programa está pensado para detectar cuando los estudiantes utilizan inteligencia artificial durante exámenes o clases normales. Para ello, necesitamos almacenar información sobre quién puede acceder al sistema, los horarios en los que el sistema debe vigilar los ordenadores, y las capturas de pantalla que se realizan automáticamente cuando se detecta algo sospechoso. Además, tenemos que registrar qué profesor está relacionado con cada captura, permitiéndole acceder únicamente a sus propias imágenes y controlar de forma sencilla cuándo activar o desactivar el modo automático de capturas según la necesidad del momento.

6.4.1.1. Tablas de la base de datos:

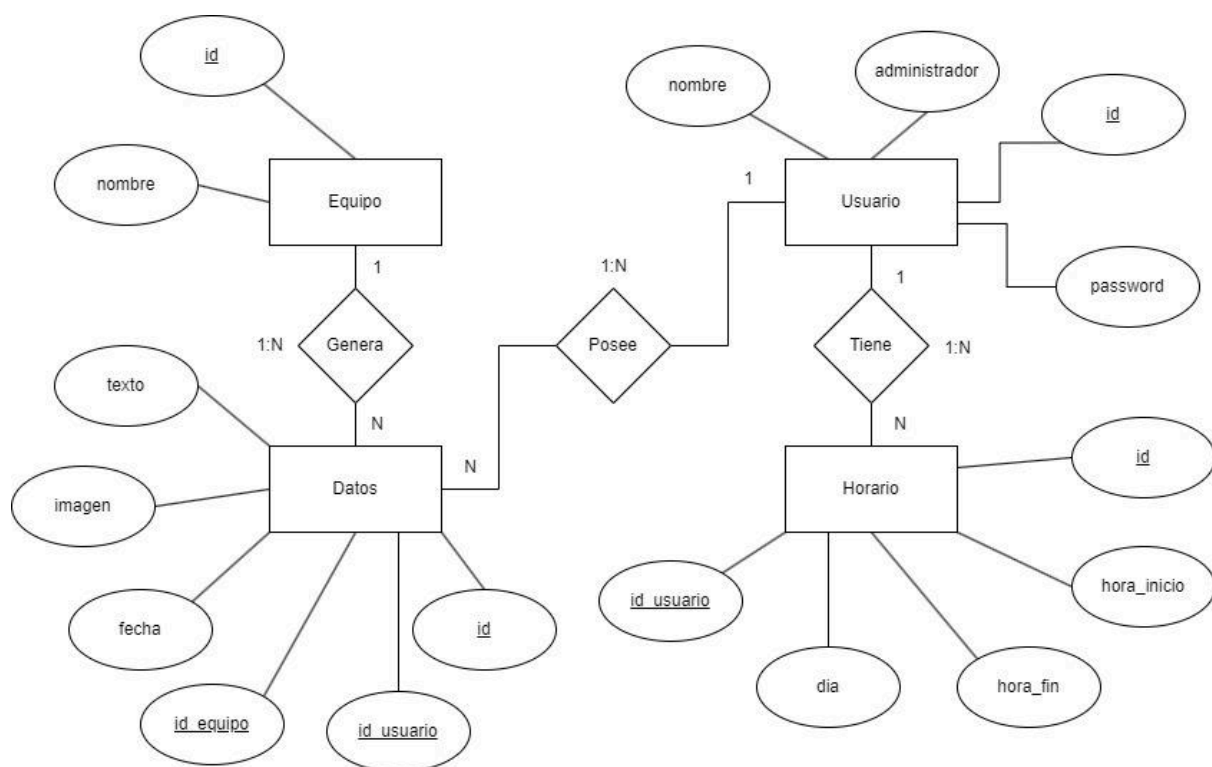
- **Tabla usuarios**
 - id (int, PK) → Identificador único del usuario.
 - nombre (str, único) → Nombre de usuario.
 - password_hash (str) → Hash de la contraseña.
 - administrador (bool) → Indica si el usuario tiene permisos de administrador.
- **Tabla equipos**
 - id (int, PK) → Identificador único del equipo.
 - nombre (str, único) → Nombre del equipo.
- **Tabla datos**
 - id (int, PK) → Identificador único de la captura.
 - id_usuario (int, FK) → Usuario propietario de la captura.
 - id_equipo (int, FK) → Equipo que generó la captura.
 - fecha (datetime) → Fecha y hora de la captura.
 - Imagen (LargeBinary) → Imagen capturada.
 - Texto (Text) → Texto extraído mediante OCR.
- **Tabla horarios**
 - id (int, PK) → Identificador único del horario.
 - hora_inicio (Time) → Hora de inicio de la captura.
 - hora_fin (Time) → Hora de finalización de la captura.
 - dia (int) → Día de la semana (0=Lunes, 6=Domingo).
 - id_usuario (int, FK) → Usuario asociado al horario.

6.4.1.2. Relación entre las tablas

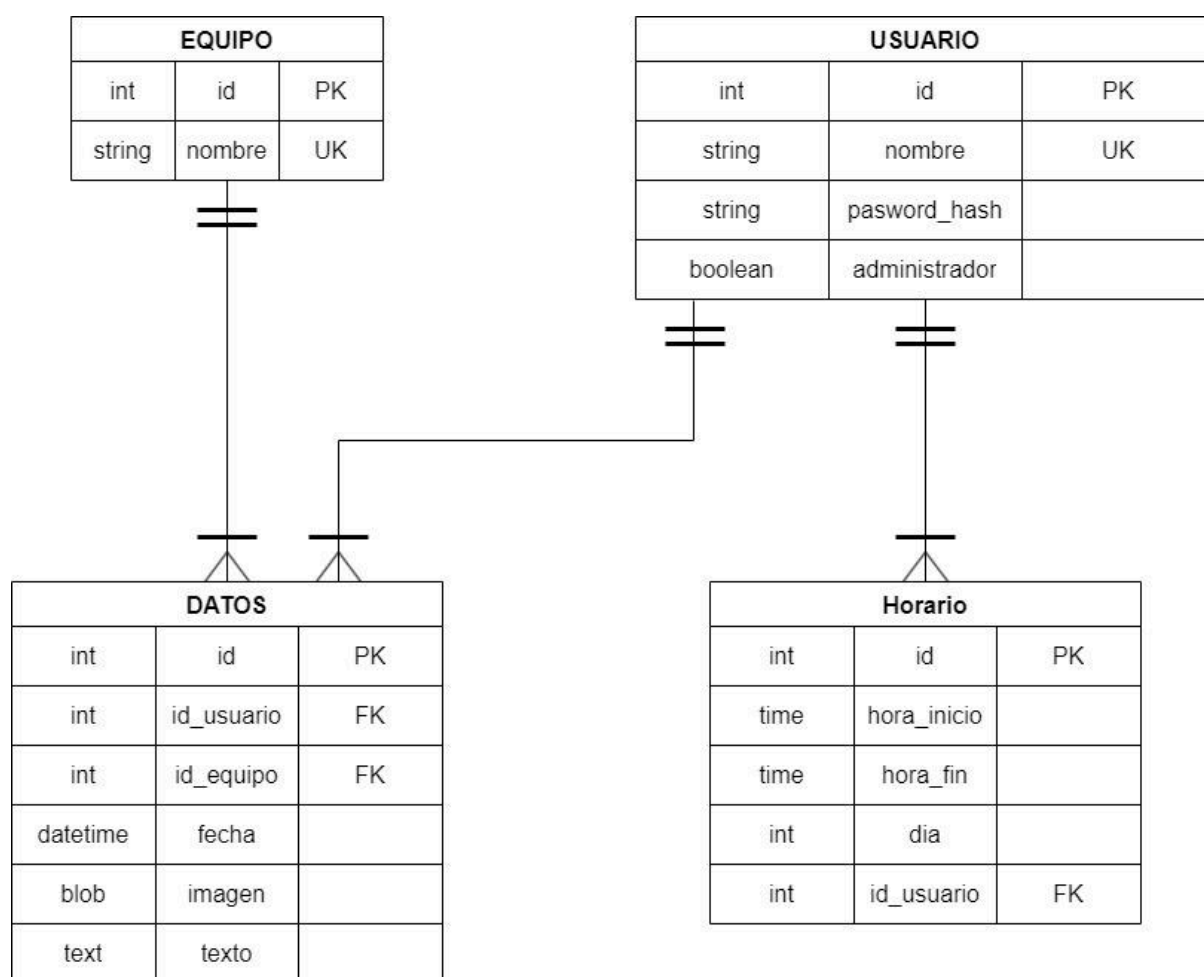
- Un usuario puede tener múltiples horarios asignados. Esto permite definir diferentes tramos de tiempo en los que ese usuario (normalmente un profesor) tiene clases y desea activar el sistema de capturas.
- Un usuario también puede estar asociado a múltiples datos (capturas), de forma que cada imagen enviada por los clientes quede registrada con el profesor correspondiente.
- Cada equipo puede enviar múltiples datos. Esto permite que desde un mismo ordenador se registren todas las capturas realizadas durante diferentes sesiones.
- La tabla datos sirve de nexo entre usuarios y equipos, reflejando qué profesor estaba a cargo en el momento de la captura y desde qué dispositivo se tomó.

Esta estructura de base de datos permite una gestión eficiente del monitoreo de IA en los exámenes, asegurando que cada usuario solo acceda a la información que le corresponde y garantizando la integridad del sistema.

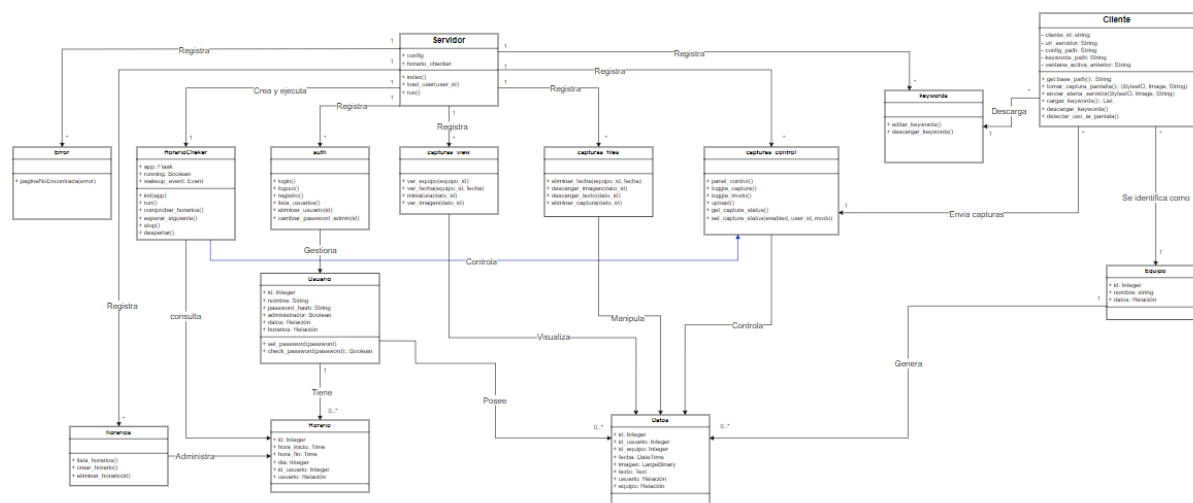
6.4.2. Diagrama de la ER/EER



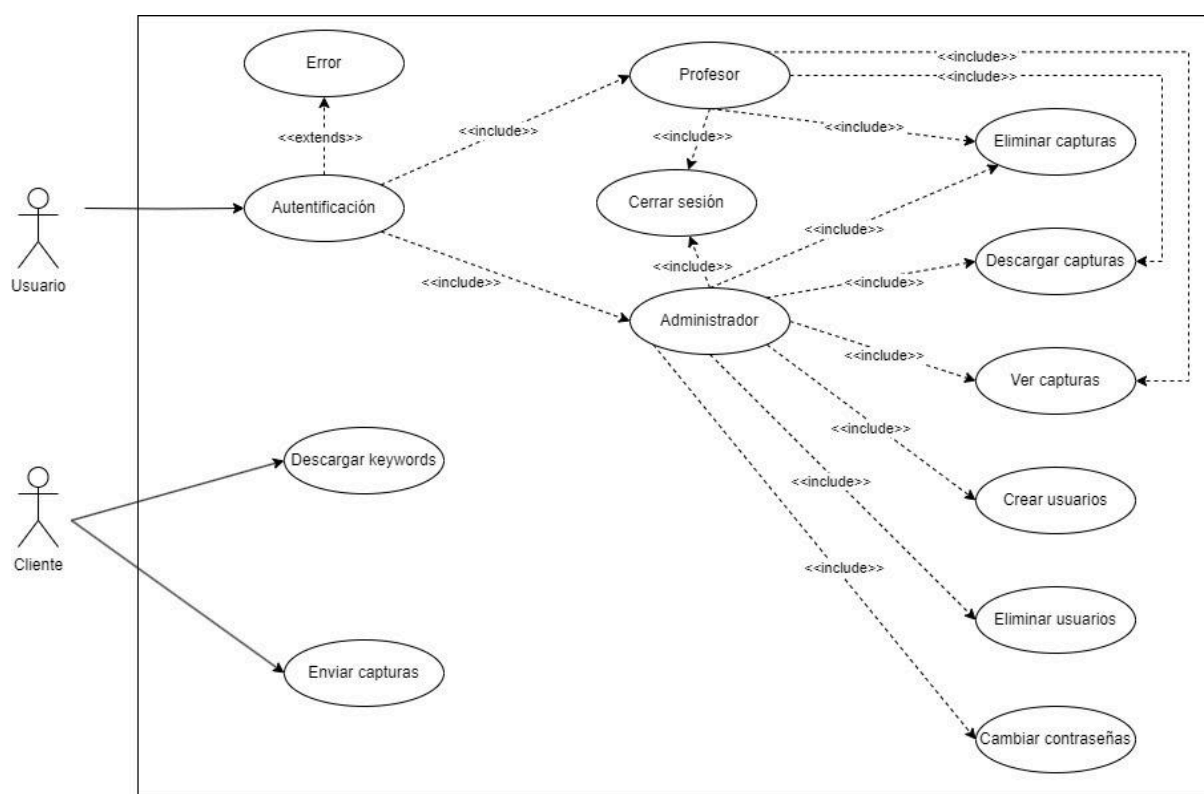
6.4.3. Modelo relacional



6.4.4. Diagrama de clases



6.4.5. Diagramas de casos de uso



6.4.5.1. Usuario

CU-Usuario	Uso general del sistema por parte de usuarios		
Descripción	El usuario accede al sistema, realiza acciones según su rol (profesor o administrador) y gestiona información a través de la interfaz web.		
Actores	Usuario (Profesor / Administrador)		
Precondiciones	El usuario debe estar registrado en la base de datos y el servidor debe estar en ejecución.		
Postcondiciones	El usuario accede al sistema y puede realizar las acciones disponibles según su nivel de acceso.		
Secuencia Normal	#	Acción	Reacción
	1	Accede a la URL del sistema.	Muestra la página de autenticación.
	2	Introduce nombre de usuario y contraseña.	Verifica las credenciales.
	3	Si son válidas, entra como profesor o administrador.	Redirige al panel correspondiente.
	4	El profesor accede a sus capturas.	Muestra listado de capturas filtradas por horario y equipo.
	5	El profesor elimina una captura.	Elimina la captura de la base de datos.
	6	El profesor descarga una captura.	Genera el archivo y lo descarga en el navegador.
	7	El administrador accede a la gestión de usuarios.	Muestra la tabla con todos los usuarios del sistema.
	8	El administrador crea un nuevo usuario.	Registra el nuevo usuario en la base de datos.
	9	El administrador elimina un usuario.	Borra el usuario y sus datos asociados.
	10	El administrador cambia la contraseña de un usuario.	Actualiza la contraseña en la base de datos.
	11	El usuario cierra sesión.	Termina la sesión activa y redirige al login.

Excepciones	#	Acción	Reacción
	p1	Introduce usuario o contraseña incorrecta.	Muestra mensaje de error e impide el acceso.
	p2	Intenta eliminar un usuario que no existe.	Muestra error de operación inválida.
	p3	Intenta eliminar al único administrador.	Bloquea la acción y muestra advertencia.
	p4	Intenta descargar una captura que ya no está disponible.	Muestra un mensaje de archivo no encontrado.
Rendimiento	Todas las acciones deben ejecutarse en menos de 2 segundos.		
Frecuencia	Login y visualización de capturas: diaria. Gestión de usuarios: ocasional.		
Importancia	Alta		
Urgencia	Inmediata		
Comentarios	El sistema debe ser accesible, seguro y con división clara entre profesor y administrador.		

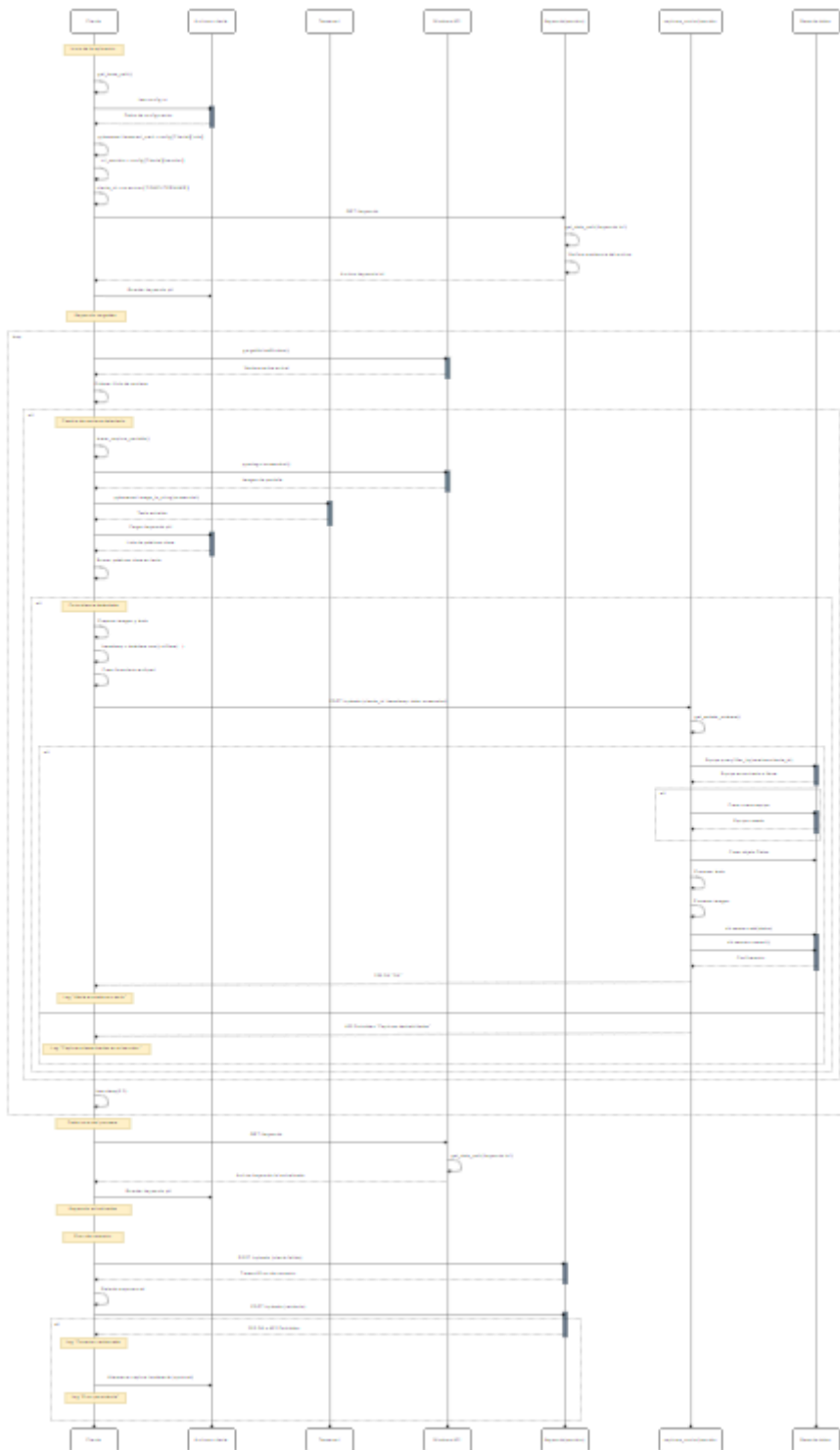
6.4.5.2. Cliente

CU-Cliente	Comportamiento del cliente en el sistema		
Descripción	El cliente, ejecutado en segundo plano en el equipo del alumno, monitoriza la actividad, detecta el uso de IA, captura la pantalla y envía los datos al servidor.		
Actores	Cliente (aplicación instalada en el equipo del alumno)		
Precondiciones	El cliente debe estar instalado, en ejecución, con conexión activa al servidor y configurado correctamente.		
Postcondiciones	Se envían las capturas al servidor y se descargan las keywords si hay cambios.		
Secuencia Normal	#	Acción	Reacción
	1	Se inicia el cliente al encender el equipo o al iniciar sesión.	Carga la configuración desde el archivo config.ini.
	2	Detecta cambio de ventana o actividad	Realiza una captura de pantalla.

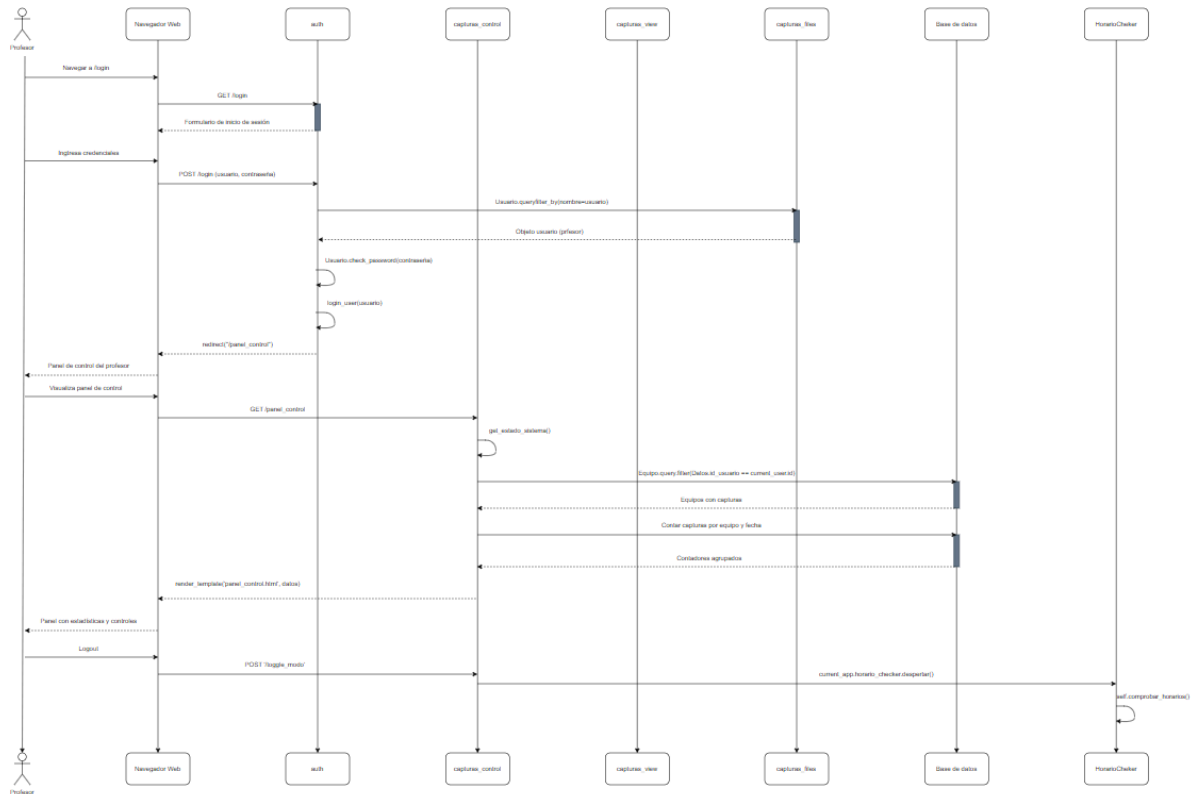
		sospechosa.	
	3	Extrae texto de la imagen mediante OCR.	Analiza el contenido y lo compara con la lista de keywords.
	4	Si hay coincidencia con palabras clave.	Prepara y envía la captura al servidor mediante una petición HTTP.
	5	Verifica si el envío ha sido exitoso.	El servidor recibe y almacena la captura.
	6	Consulta si hay nueva versión de las keywords.	Si hay actualización, descarga el nuevo archivo keywords.pkl.
Excepciones	#	Acción	Reacción
	p1	No hay conexión con el servidor al enviar una captura.	El cliente guarda la imagen en el local y reintentar el envío más adelante.
	p2	El archivo config.ini está corrupto o no existe.	El cliente muestra un error o no se ejecuta correctamente.
	p3	Fallo en el OCR o no se detecta texto.	El cliente descarta la imagen o la registra como "no válida".
	p4	El servidor devuelve el error al recibir la imagen.	El cliente guarda temporalmente la captura para reintentar la.
Rendimiento	El proceso de detección y envío debe tardar menos de 3 segundos por ciclo.		
Frecuencia	Dependiendo del comportamiento del alumno, puede generar de 0 a decenas de capturas por hora.		
Importancia	Crítica		
Urgencia	Inmediata		
Comentarios	El cliente debe ejecutarse de forma oculta y automática. Puede ser desplegado como ejecutable (cliente_hidden.exe)		

6.4.6. Diagramas de secuencia

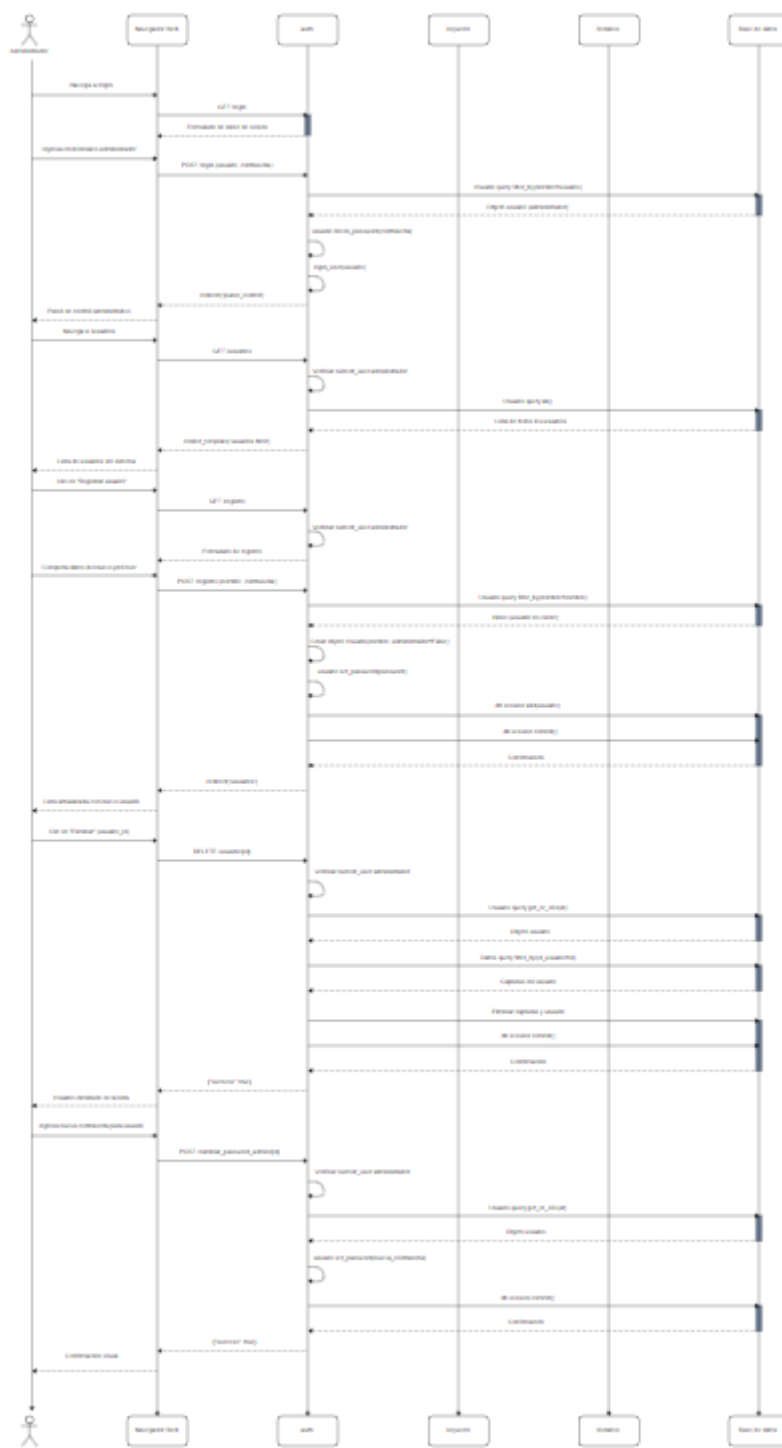
6.4.6.1. Cliente-servidor



6.4.6.2. Profesor-servidor

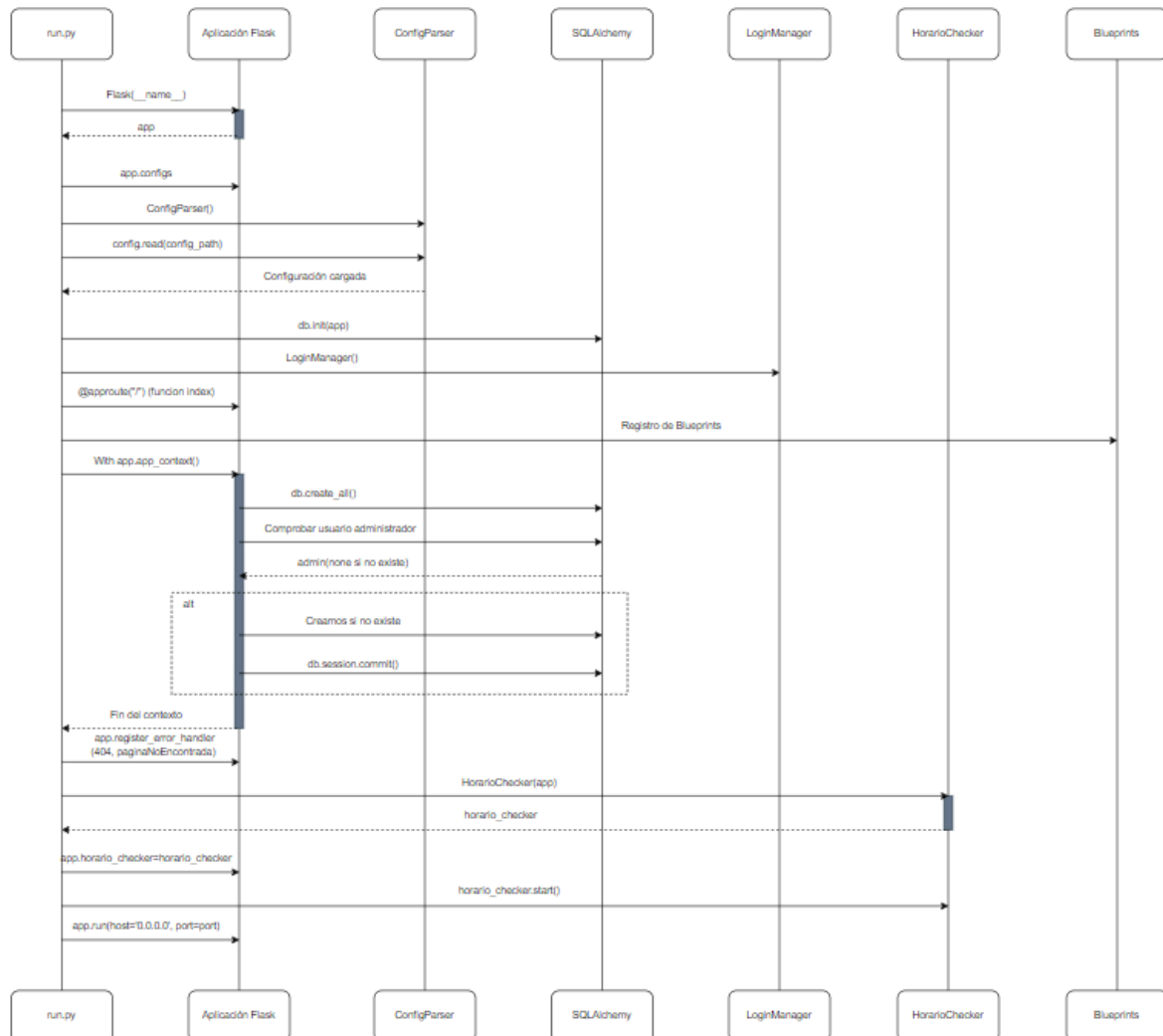


6.4.6.3.Administrador-servidor





6.4.6.4. Interno del servidor



6.5. Diseño de la interfaz de usuario

La interfaz de usuario del sistema ha sido diseñada con el objetivo de ser clara, intuitiva y eficiente para cada uno de los roles que interactúan con la aplicación. Se ha utilizado un enfoque basado en Bootstrap para garantizar un diseño responsive que se adapte a distintos dispositivos, facilitando la accesibilidad tanto en ordenadores como en tablets o móviles.

6.5.1. Principios de diseño utilizados:

- Simplicidad: La interfaz minimiza distracciones y presenta únicamente la información relevante.
- Consistencia: Se mantiene un estilo homogéneo en todas las vistas.
- Accesibilidad: Se han aplicado buenas prácticas para asegurar la usabilidad por parte de todos los usuarios.
- Optimización de flujo de trabajo: Cada acción importante se puede realizar en pocos pasos.

6.5.2. Paleta de colores y estilo visual:

Se ha optado por una paleta de colores sobria y profesional, combinando tonos oscuros y neutros para facilitar la lectura prolongada y reducir la fatiga visual. Los colores clave son:

- Negro (#212529): Color predominante para encabezados y elementos de énfasis.
- Gris claro (#F7F7F7): Fondo de las secciones principales para crear contraste.
- Blanco (FFFFFF): Utilizado en textos y fondos secundarios.
- Verde (#10B981): Indica acciones exitosas y estados positivos.
- Rojo (#EF4444): Se usa para alertas y acciones críticas, como eliminación de capturas.
- Azul (#0B5ED7): Para los botones de descargar imágenes, y textos (links)
- Azul claro (#0DCAF0): Para los botones de descargar texto
- Gris (#C7C8AF): Usado para textos del navBar

6.5.3. Tipografía utilizada:

El sistema emplea la fuente Inter, que proporciona una lectura clara y profesional en distintos tamaños. Se han definido los siguientes tamaños para mejorar la jerarquía visual:

- Títulos principales: 24px, negrita.
- Subtítulos: 18px, seminegrita.
- Texto normal: 16px.
- Etiquetas y botones: 14px.

6.5.4. Iconografía:

Se han utilizado imágenes obtenidas de Google y un icono de aplicación generado con DALLE.

6.5.5.Mockups y estructura de navegación:

El sistema cuenta con distintas pantallas diseñadas para la gestión eficiente de sus funciones. Las principales vistas implementadas son:

6.5.5.1.Index

Página de inicio del sistema.

IA Detector Login

Bienvenido a IA Detector

Sistema de detección y monitoreo de actividad sospechosa

Inicia sesión para acceder al sistema

Iniciar sesión

6.5.5.2.Login

Permite entrar en un usuario.

IA Detector Login

Inicio de sesión

Usuario

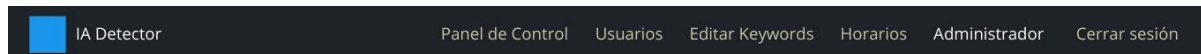
Contraseña

Iniciar sesión

(C) 2025

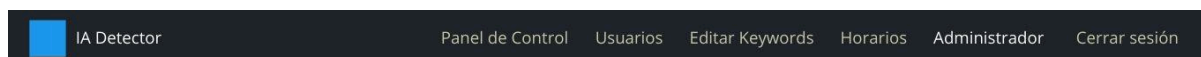
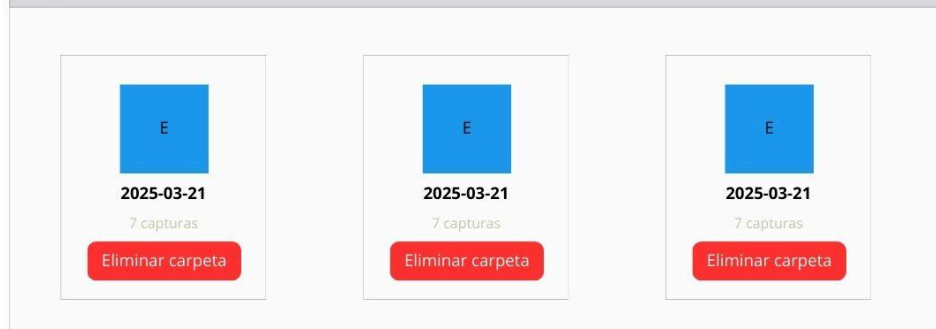
6.5.5.3. Panel de Control (administrador)

Sección principal para la gestión de capturas y estado del sistema.



[Panel de control](#) / PC-1

Capturas de PC-1




[Panel de control](#) / [PC-1](#) / 2025-03-21

Capturas del 2025-03-21



6.5.5.4. Usuarios

Administración de usuarios, accesible solo para administradores.

 IA Detector
 Panel de Control Usuarios Editar Keywords Horarios Administrador Cerrar sesión


Listado de Usuarios

[Nuevo Usuario](#)

ID	Nombre	Administrador	Acciones
1	Administrador	SI	Cambiar Contraseña Eliminar
2	Usuario 1	No	Cambiar Contraseña Eliminar
3	Usuario 2	No	Cambiar Contraseña Eliminar
4	Usuario 3	No	Cambiar Contraseña Eliminar

6.5.5.5. Editar Keywords

Gestión de palabras clave para detección de IA.

 IA Detector
 Panel de Control Usuarios Editar Keywords Horarios Administrador Cerrar sesión

Editar Palabras Clave

[Descargar Archivo](#)

Contenido del archivo keywords.txt


```

chatgpt
copilot
openai
bard
codeium
qwen
deepseek
davinci
codex
watson
lumen5
hivemind
chatbot
deepmind
  
```

[Guardar Cambios](#)

6.5.5.6. Horarios

Configuración de los períodos en los que se activará la monitorización.


IA Detector

Panel de Control
 Usuarios
 Editar Keywords
 Horarios
 Administrador
 Cerrar sesión

Gestión de Horarios

Añadir Nuevo Horario


Usuario: Seleccionar usuario ▼
 Día: Lunes ▼
 Hora inicio: --:--
 Hora fin: --:--
Añadir Horario

Lunes
 Martes
 Miércoles
 Jueves
 Viernes
 Sábado
 Domingo

Usuario	Hora inicio	Hora fin	Acciones
Usuario 1	18:20	19:20	Eliminar
Usuario 2	19:20	20:20	Eliminar

6.5.5.7. Perfil del usuario

Muestra el nombre del usuario activo y opciones de personalización.


IA Detector

Panel de Control
 Usuario 1
 Cerrar sesión

Panel de control


Control de Capturas


☐ Modo Automático


Estado del Sistema: Desactivado


Iniciar Capturas


Capturas por Equipo



PC-1
5 días


PC-2
7 días


PC-3
1 días


PC-4
15 días


PC-5
8 días


PC-6
12 días

Panel de control

Control de Capturas

☐ Modo Automático

Estado del Sistema: Activado
 Guardando para: **Usuario 1** (Modo Manual)

Detener captura

Capturas por Equipo

					
PC-1	PC-2	PC-3	PC-4	PC-5	PC-6
5 días	7 días	1 días	15 días	8 días	12 días

Este diseño garantiza una experiencia de usuario optimizada, permitiendo que administradores y profesores gestionen el sistema de manera eficiente y sin complicaciones innecesarias.

6.6.Presupuesto del proyecto:

6.6.1 Coste del Hardware empleado

Componente	Precio
Ordenador completo (procesador Intel i7, 16GB RAM, 1TB SSD)	950 €
Pantalla Full HD 24"	150 €
Periféricos (teclado y ratón)	50 €
Subtotal Hardware	1.150 €

6.6.2 Software y herramientas empleadas

Componente	Precio
Sistema Operativo (Windows 10)	110,99 €
Python 3.8.20	Gratuito

Anaconda (Gestión de entorno Python)	Gratuito
Google docs	Gratuito
Visual Studio Code (Editor de código)	Gratuito
GitHub (Control de versiones)	Gratuito
Draw.io	Gratuito
Google Chrome	Gratuito
Subtotal Software	110,99€

6.6.3 Total general

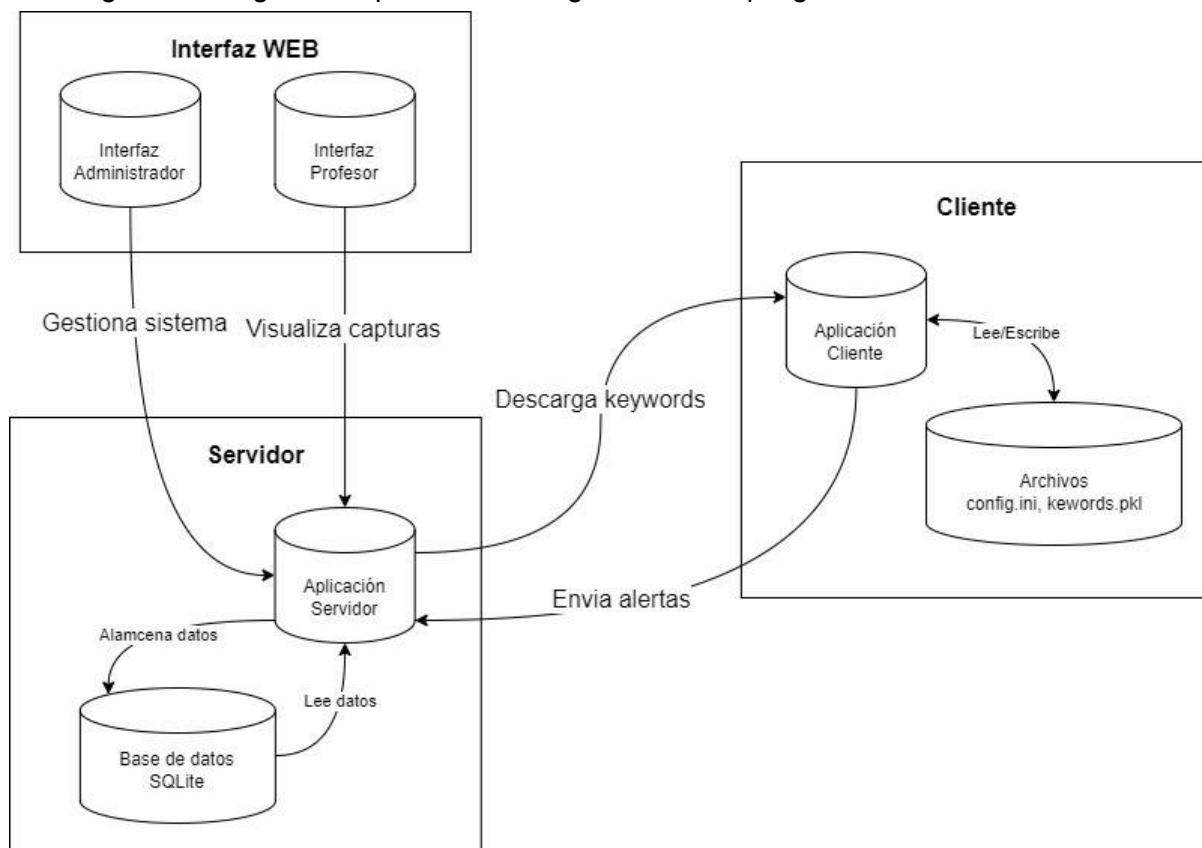
Concepto	Precio
Subtotal Hardware	1.150 €
Subtotal Software	110,99 €
Total General	1.250,99 €

Este presupuesto refleja un proyecto asequible basado principalmente en soluciones gratuitas y open-source, optimizando los costes totales.

7. Desarrollo y ejecución

El sistema desarrollado se basa en una arquitectura distribuida compuesta por dos elementos principales: el cliente (instalado en los ordenadores del aula) y el servidor (que centraliza la información y la gestión del sistema).

En la siguiente imagen se representa el diagrama de despliegue del sistema:



7.1.Descripción del diagrama

7.1.1.Cliente:

- Cada ordenador de aula ejecuta una aplicación cliente en segundo plano. Esta aplicación:
 - Lee la configuración desde los archivos config.ini y keywords.pkl.
 - Monitorea el sistema y, cuando detecta alguna coincidencia con las palabras clave, realiza capturas de pantalla.
 - Envía estas capturas al servidor como alertas de posible uso de inteligencia artificial.
 - También puede actualizar su lista de palabras clave descargándola desde el servidor.

7.1.2.Servidor:

- El servidor ejecuta la aplicación central desarrollada con Flask. Sus funciones son:
 - Recibir las capturas enviadas por los clientes.
 - Leer y almacenar datos en la base de datos SQLite.
 - Gestionar y enviar las palabras clave a los clientes.
 - Ofrecer acceso a través de una interfaz web diferenciada por roles.

7.1.3.Interfaz web:

- Los profesores acceden a una interfaz desde la cual pueden visualizar, descargar o eliminar las capturas correspondientes a su horario y cuenta.
- Los administradores disponen de un panel de gestión completo que les permite:
 - Añadir o modificar usuarios.
 - Configurar los horarios.
 - Editar la lista de palabras clave utilizadas para la detección.

Este sistema modular y distribuido permite un control efectivo y adaptable en tiempo real del uso de tecnologías de inteligencia artificial en el aula, asegurando que cada captura está relacionada con el contexto, usuario y equipo correspondientes.

8.Validación, accesibilidad y testing

Para garantizar que el sistema sea accesible, estable y funcional en diferentes entornos, se realizaron diversas pruebas manuales enfocadas en la estabilidad, usabilidad y rendimiento del sistema.

8.1.Métodos utilizados:

- Pruebas en hardware variado: Se ejecutó el sistema en un ordenador moderno y en un portátil con especificaciones limitadas para comprobar su rendimiento y fluidez.
- Evaluación visual manual: Se revisó la claridad de la interfaz, la legibilidad de los textos y la facilidad de navegación.
- Simulación de distintos escenarios de uso: Se probaron todas las funciones principales, incluyendo la activación y desactivación de capturas, la gestión de horarios y la administración de palabras clave.
- Pruebas funcionales: Se verificó que cada módulo del sistema ejecuta correctamente su función asignada, incluyendo la captura de pantalla, el análisis OCR y el envío de datos al servidor.
- Pruebas de integración: Se comprobó la interacción entre los distintos componentes del sistema, asegurando que la comunicación entre cliente y servidor se realiza sin errores.
- Pruebas de estrés: Se ejecutaron pruebas con múltiples clientes enviando capturas simultáneamente para evaluar la capacidad del servidor.
- Pruebas de compatibilidad: Se probó el sistema en diferentes dispositivos y resoluciones de pantalla para garantizar una correcta visualización.

8.2.Resultados de la validación y mejoras aplicadas:

Las pruebas realizadas confirmaron que el sistema funciona de manera estable en distintos entornos, aunque se realizaron algunos ajustes para optimizar su rendimiento:

- Optimización de capturas: Se mejoró el procesamiento de imágenes para reducir el tiempo de análisis.
- Corrección de errores en la interfaz: Se ajustaron ciertos elementos visuales que no se mostraban correctamente en resoluciones más bajas.
- Mejor manejo de datos en la base de datos: Se optimizó la consulta y almacenamiento de información para mejorar la velocidad de respuesta del servidor.
- Ajuste en el tamaño de ciertos elementos: Para mejorar su visibilidad en pantallas más pequeñas.
- Optimización de la carga de la interfaz: Para mejorar el rendimiento en equipos con menor capacidad.
- Verificación del acceso y flujo de trabajo: Para garantizar que los usuarios pueden realizar sus acciones sin inconvenientes.

8.3.Conclusión:

Gracias a estas validaciones y ajustes, el sistema se ha optimizado para ofrecer una experiencia estable y accesible en distintos dispositivos y condiciones de uso, permitiendo que administradores y profesores interactúen con la aplicación sin dificultades técnicas.

9.Conclusiones, propuestas de mejora y valoración personal

Después de todo el proceso de desarrollo, implementación y pruebas, se han obtenido diversas conclusiones que reflejan el desempeño del sistema y las posibles mejoras futuras.

9.1Conclusiones generales:

- El sistema cumple con su propósito principal de detectar el uso de inteligencia artificial en exámenes y clases, brindando una herramienta útil para administradores y profesores.
- La elección de Flask como framework backend ha permitido un desarrollo ágil y una estructura modular fácil de mantener.
- La integración de PyAutoGUI y Tesseract OCR ha demostrado ser efectiva en la captura y análisis de texto en pantalla, aunque con ciertas limitaciones en condiciones de baja calidad de imagen.
- El almacenamiento en SQLite ha resultado suficiente para la escala del proyecto, pero podría requerir una migración a una base de datos más robusta en caso de un crecimiento significativo.

9.2. Problemas encontrados y soluciones:

- Optimización del procesamiento OCR: Inicialmente, la detección de texto tenía fallos con ciertas tipografías y tamaños. Se soluciona ajustando la preprocesación de las imágenes antes del análisis.
- Fluidez en la interfaz de usuario: Algunas pruebas iniciales indicaron tiempos de carga altos en dispositivos con menor capacidad. Se resolvió optimizando consultas y reduciendo el peso de ciertos recursos gráficos.
- Gestión de horarios: Se encontró necesario mejorar la flexibilidad del sistema de horarios para permitir más personalización, lo cual fue implementado en una actualización posterior.
- Conflictos de escritura simultánea en base de datos: Uno de los problemas más críticos surgió al recibir múltiples capturas de diferentes clientes de forma simultánea. Esto provocaba errores de escritura en la base de datos SQLite, que no está diseñada para manejar accesos concurrentes intensivos. La solución implementada fue incorporar una cola de tipo Queue, que serializa las operaciones de escritura. De esta forma, las capturas se procesan una a una en segundo plano, evitando bloqueos y garantizando la integridad de los datos. Esta mejora ha permitido mantener el uso de SQLite como base de datos ligera y eficiente, incluso con múltiples clientes activos.

9.3. Propuestas de mejora:

- Migración a una base de datos más escalable: En un futuro, se podría utilizar PostgreSQL o MySQL para manejar grandes volúmenes de datos con mayor eficiencia.
- Implementación de WebSockets: Para mejorar la experiencia de los profesores, se podría desarrollar un sistema de notificaciones en tiempo real para nuevas capturas detectadas.
- Visualización de equipos conectados: Añadir una sección específica en el panel de administrador que muestre los equipos conectados recientemente o actualmente activos. Esto facilita el seguimiento en tiempo real, la detección de problemas de red y la confirmación de que los clientes están funcionando correctamente.

Bibliografía

Fuentes utilizadas en formato APA 7.^a edición.

Referencias

Anaconda Inc. (s.f.). Anaconda Documentation. Recuperado de

<https://docs.anaconda.com/>

Bootstrap. (s.f.). Bootstrap Documentation. Recuperado de

<https://getbootstrap.com/>

Codigofacilito. (s.f.). Cursos de programación y desarrollo web. Recuperado de

<https://codigofacilito.com/>

Dev.to. (s.f.). Comunidad de desarrolladores. Recuperado de

<https://dev.to/>

Docker Inc. (s.f.). Docker Documentation. Recuperado de

<https://docs.docker.com/>

Google. (s.f.). Tesseract OCR Documentation. Recuperado de

<https://github.com/tesseract-ocr/tesseract>

Medium. (s.f.). Artículos de tecnología y desarrollo. Recuperado de

<https://medium.com/>

OpenWebinars. (s.f.). Formación en tecnologías de la información. Recuperado de

<https://openwebinars.net/>

Pallets. (s.f.). Flask Documentation. Recuperado de

<https://flask.palletsprojects.com/>

SQLAlchemy. (s.f.). SQLAlchemy Documentation. Recuperado de

<https://docs.sqlalchemy.org/>

Stack Overflow. (s.f.). Foro de desarrollo y programación. Recuperado de

<https://stackoverflow.com/>

Sweigart, A. (s.f.). PyAutoGUI Documentation. Recuperado de

<https://pyautogui.readthedocs.io/>

Esta bibliografía ha sido fundamental para la correcta implementación del proyecto, permitiendo solucionar problemas técnicos y mejorar la calidad del desarrollo.

Anexo I: Manual de instalación

1.Requisitos técnicos

Requisitos mínimos recomendados:

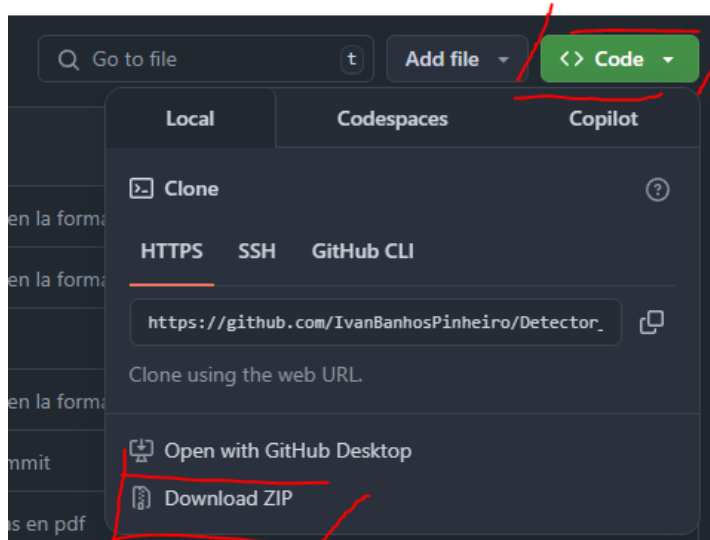
- Servidor:
 - Procesador: Intel Core i5 o superior.
 - Memoria RAM: 8GB.
 - Almacenamiento: 50GB de espacio libre.
 - Sistema Operativo: Windows 10 o superior.
 - Docker instalado para la gestión del servidor.
- Cliente (ordenador de aula):
 - Procesador: Intel Core i3 o superior.
 - Memoria RAM: 4GB.
 - Sistema Operativo: Windows 10 o superior.
 - Conexión a Internet local para comunicación con el servidor.

2.Instalación del software

Este manual describe los pasos necesarios para instalar y configurar correctamente el sistema, tanto en el servidor como en los equipos cliente.

Para ambos puntos necesitaremos descargar el proyecto de github, para eso nos dirigiremos a este link y lo descargamos.

https://github.com/IvanBanhosPinheiro/Detector_la_Trabajo_Final



Para el cliente también descargamos el ejecutable.

https://github.com/IvanBanhosPinheiro/Detector_la_Trabajo_Final/releases/tag/v0.1

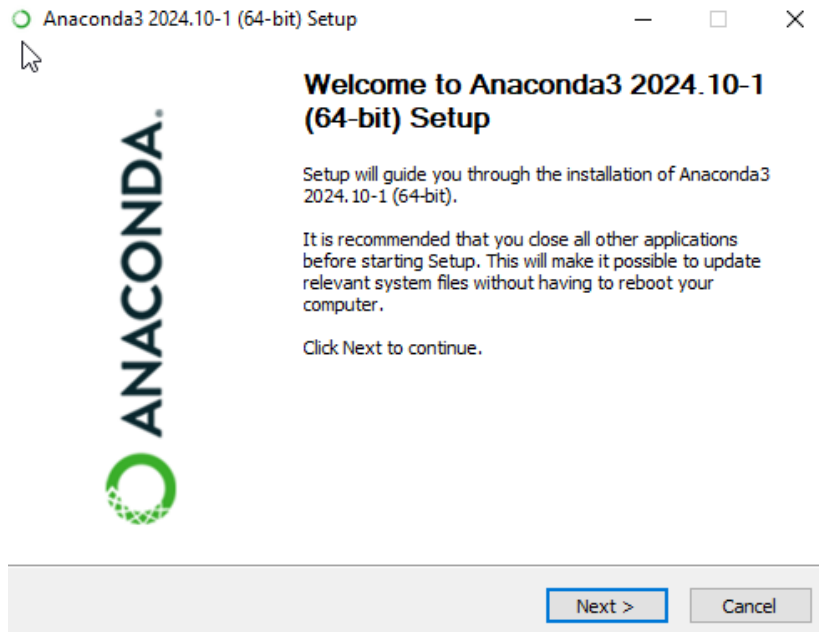
2.1.Servidor:

En ambas opciones te recomiendo eliminar de la carpeta `instace database.bd` para tener un entorno totalmente limpio, esta se creará la primera vez que se ejecute el programa.

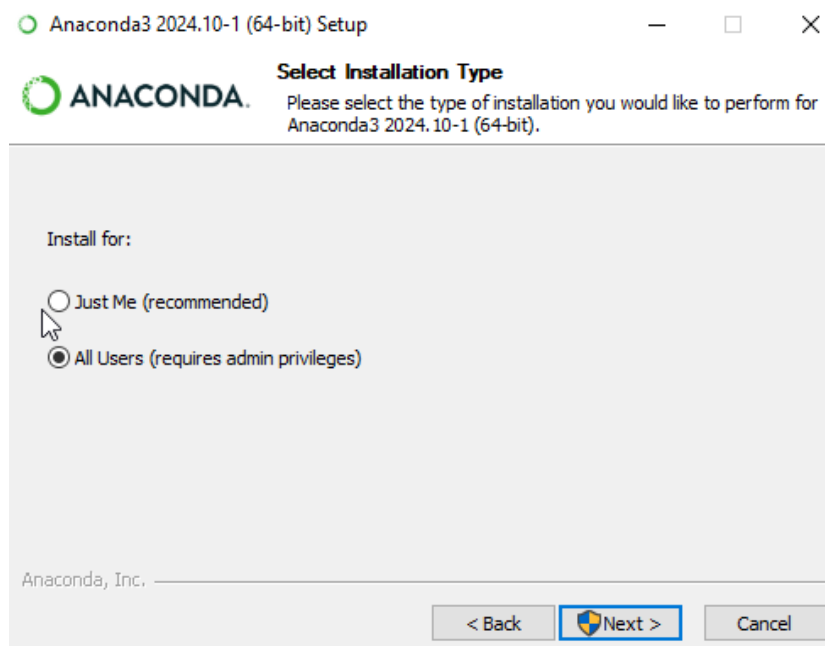
2.1.1.Opción 1: Anaconda (entorno virtual).

Descargamos anaconda <https://www.anaconda.com/download/success>

- Seguimos las instrucciones de instalación.

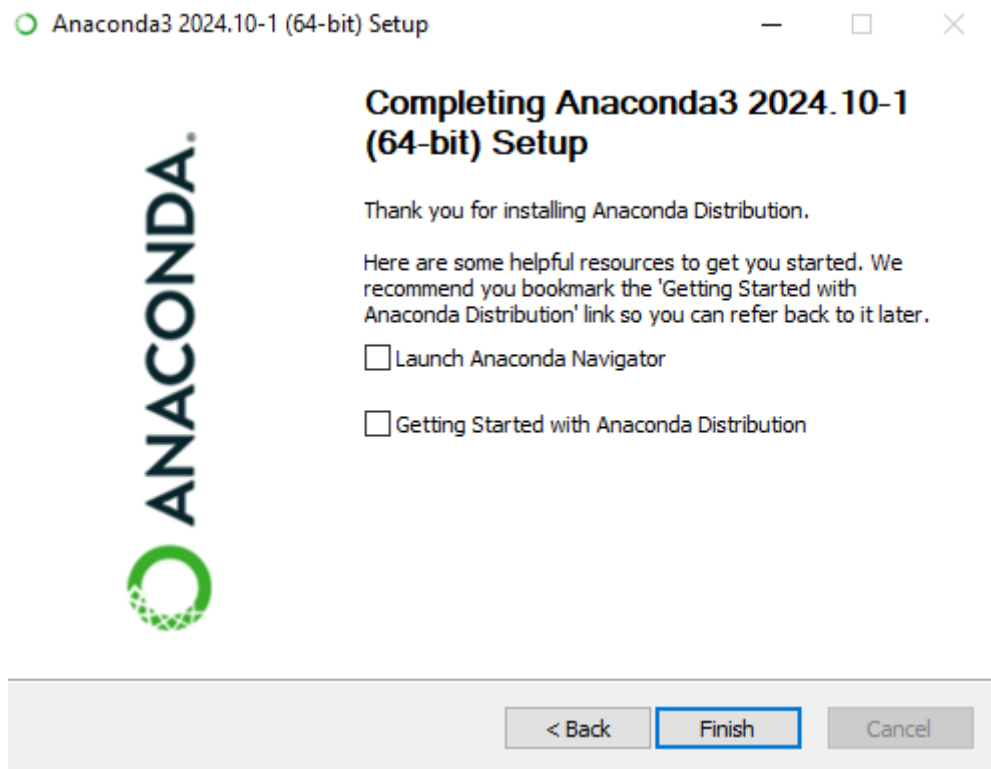


Yo seleccioné la opción de para todos que pedirá permisos de administrador

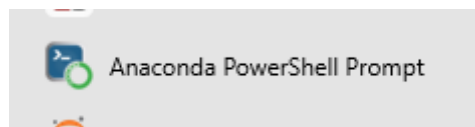




Siguiente siguiente... y finish



- Ahora vamos a abrir la consola de conda para crear un entorno virtual con python 3.8.20, en inicio



- Escribimos este comando: `conda create -n detector_env python=3.8.20`

Anaconda PowerShell Prompt

```
(base) PS C:\Users\Tutorial> conda create -n detector_env python=3.8.20_
```

Esto descargara todos los paquetes necesarios para crear un entorno con python 3.8.20 para poder usar nuestro programa.

- Una vez creado necesitamos activarlo con el siguiente comando: `conda activate detector_env` lo ejecutamos en la consola de conda tambien.

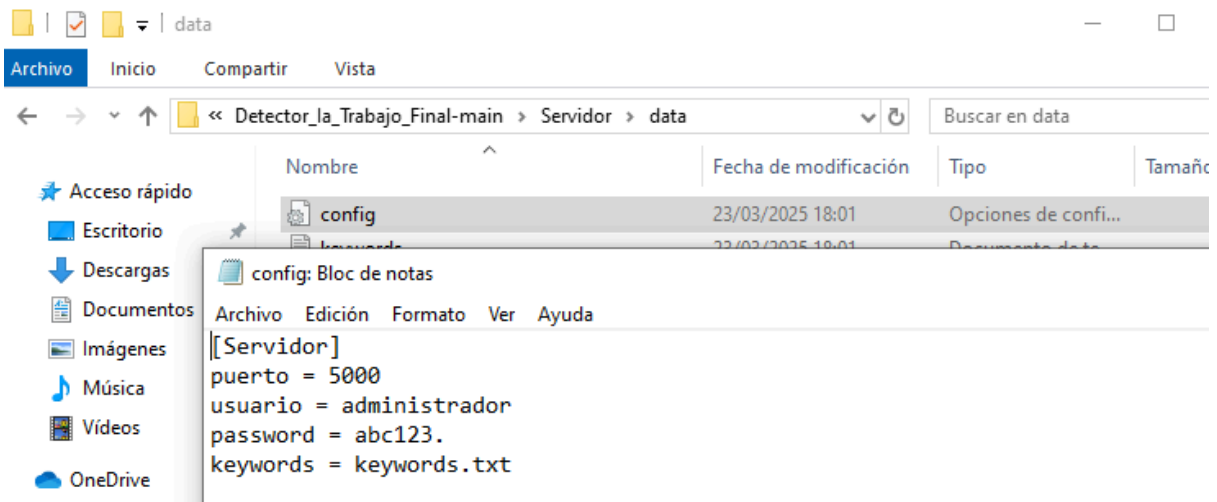
Anaconda PowerShell Prompt

```
(base) PS C:\Users\Tutorial>
(base) PS C:\Users\Tutorial> conda activate detector_env
(detector_env) PS C:\Users\Tutorial> _
```

- Nos dirigimos a la carpeta donde esta el servidor que descargamos de github anteriormente y ejecutamos `pip install -r requirements.txt` para instalar todo lo necesario

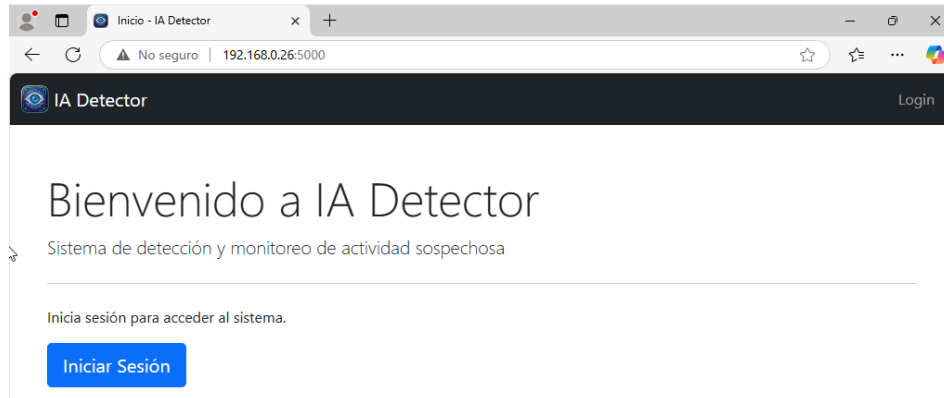
```
Anaconda PowerShell Prompt
(base) PS C:\Users\Tutorial> conda activate detector_env
(detector_env) PS C:\Users\Tutorial> cd C:\Users\Tutorial\Downloads\Detector_Ia_Trabajo_Final-main\Servidor
(detector_env) PS C:\Users\Tutorial\Downloads\Detector_Ia_Trabajo_Final-main\Servidor> pip install -r .\requirements.txt
```

- En la carpeta data del servidor modificaremos el archivo config.ini para cambiar la contraseña y el usuario del administrador y el puerto por donde se conectara nuestro cliente y nosotros.



- Ahora ejecutando el comando `python run.py` ejecutaremos el programa y ya estaría funcionando en tu localhost o en tu red local con tu ip y el puerto 5000 en nuestro caso 192.168.0.26:5000 ya podriamos acceder al panel de control y conectar nuestros clientes.

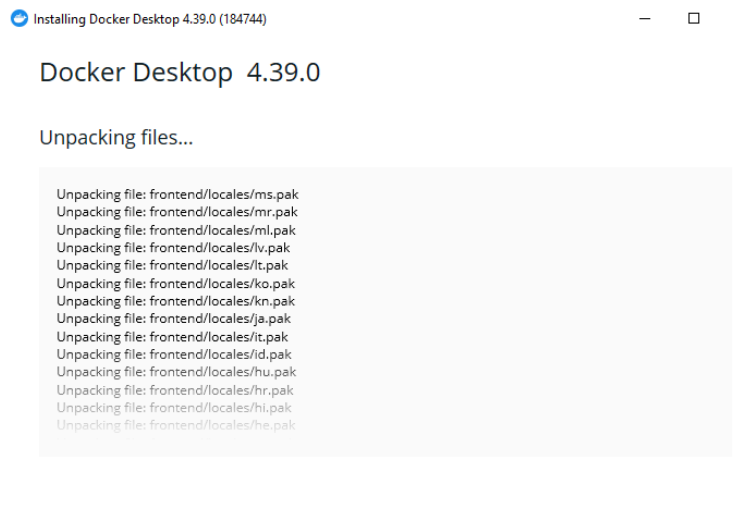
```
Selección Anaconda PowerShell Prompt
t-3.1.1 idna-3.10 importlib-metadata-8.5.0 itsdangerous-2.2.0 requests-2.32.3 typing-extensions-4.12.2 urllib3-2.2.3 z
3.20.2
(detector_env) PS C:\Users\Tutorial\Downloads\Detector_Ia_Trabajo_Final-main\Servidor> python run.py
Hilo de comprobación de horarios iniciado
* Serving Flask app 'run'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://192.168.0.26:5000
Press CTRL+C to quit
Comprobando horarios: Día 6 (Lunes=0), Hora 18:09
No se encontró horario activo para el momento actual
Próxima comprobación en 48.18 horas
```



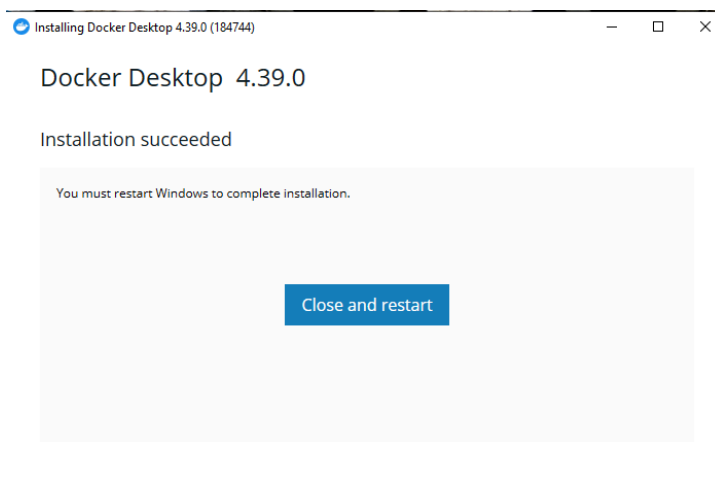
2.1.2.Opción 2: Docker (recomendable).

Descargamos Docker Desktop <https://www.docker.com/products/docker-desktop/>

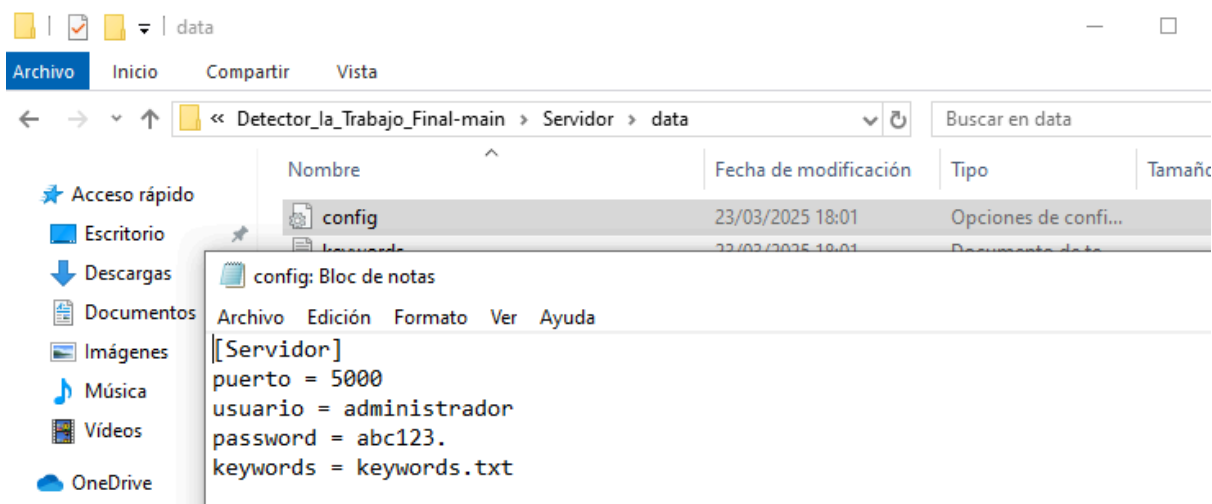
- Ejecutamos el instalador y seguimos los pasos instalara todo lo que necesite.



- Al terminar aparecerá un mensaje para reiniciar el equipo, le damos y ya tendríamos docker instalado

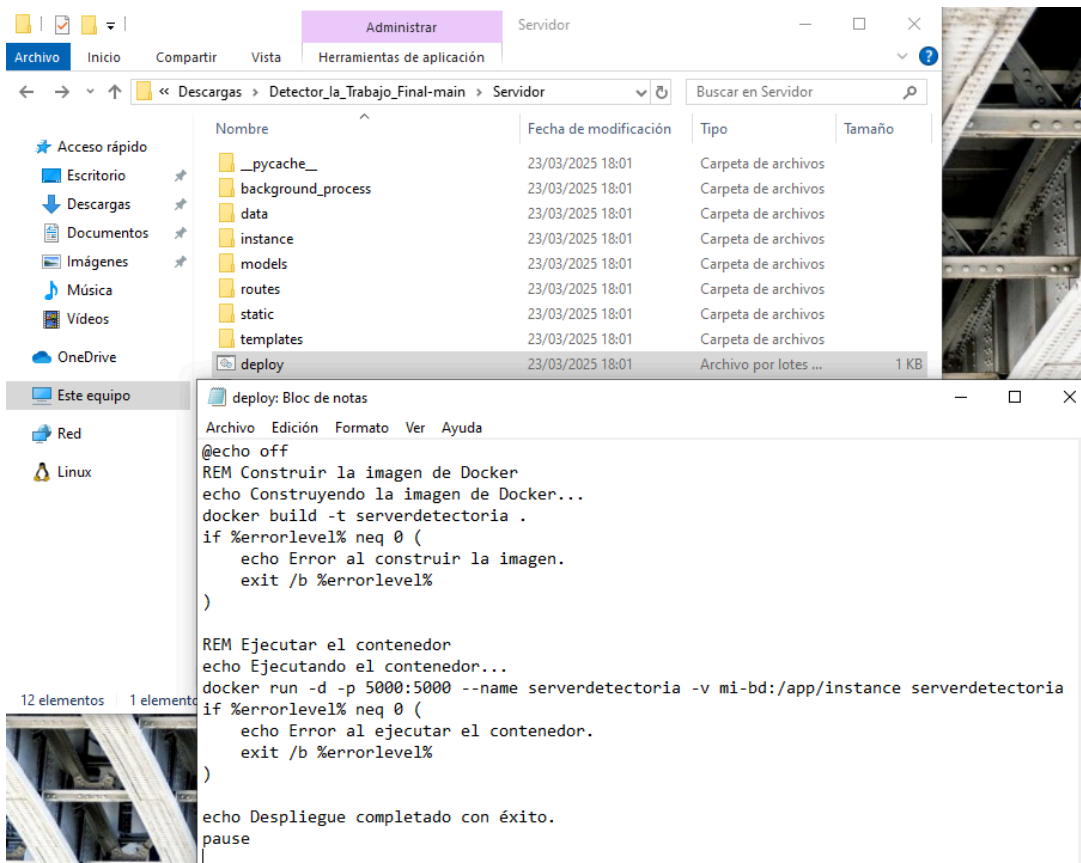


- Abrimos docker y nos dirigimos a la carpeta del servidor que descargamos de gitHub, en ella modificaremos en config.ini que se encuentra en la carpeta data. El puesto que pongamos es indiferente, yo recomiendo dejar el que está así la configuración será menor y más simple



2.1.2.1. Con deploy.bat

- Ahora vamos a editar el deploy.bat para configurar el puerto que queremos usar finalmente ya que el del config.ini será el interno del contenedor. El primer 5000 será nuestro puerto de enlace con el contenedor el otro es el interno al colocamos el puerto que nos dé la gana y ejecutamos el script.



- Se creará nuestro contenedor automáticamente y un volumen para el archivo de la base de datos al mismo tiempo que se iniciara.

```
C:\Windows\system32\cmd.exe
Construyendo la imagen de Docker...
[+] Building 3.0s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 652B
=> [internal] load metadata for docker.io/library/python:3.8.20-slim
=> [auth] library/python:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/5] FROM docker.io/library/python:3.8.20-slim@sha256:1d52838af602b4b5a831beb13a0e4d073280665ea7be7f69ce2382
=> => resolve docker.io/library/python:3.8.20-slim@sha256:1d52838af602b4b5a831beb13a0e4d073280665ea7be7f69ce2382
=> [internal] load build context
=> => transferring context: 7.02kB
=> CACHED [2/5] WORKDIR /app
=> CACHED [3/5] COPY requirements.txt .
=> CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt
=> CACHED [5/5] COPY . .
=> exporting to image
=> => exporting layers
=> => exporting manifest sha256:8751c7255d513d036d5dbbc15b9753704de520865ad1abde52b57aa26866e2df
=> => exporting config sha256:f071b38d977d4877dc39ed5fafd8f91fbb21b43d7baa2bf323a3cd057c08359b
=> => exporting attestation manifest sha256:472ea297ada33811777ab107427abc06d9cb4a7576da16441695ae5d3b9e666c
=> => exporting manifest list sha256:8ddd48c8eb15c06108a31e5e5b98b9ae5548ac051b41298db619e95c49c9ac45
=> => naming to docker.io/library/serverdetectoria:latest
=> => unpacking to docker.io/library/serverdetectoria:latest

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/vgrmx4abzryu82ptduf9p2m01
Ejecutando el contenedor...
9e09899af6e8df19f4716e49a30033f15ca372970273d056519683f1dceb2e6f
Despliegue completado con éxito.
Presione una tecla para continuar . . .
```

Containers [Give feedback](#)

View all your running containers and applications. [Learn more](#)

Container CPU usage 0.01% / 800% (8 CPUs available) Container memory usage 40.73MB / 7.55GB [Show charts](#)

Search Only show running containers

Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
serverdetectoria	9e09899af6e8	serverdetectoria	5000:5000	0.01%	1 minute ago	Restart Stop Logs Delete

Volumes [Give feedback](#)

Manage your volumes, view usage, and inspect their contents. [Learn more](#)

Search [Create](#)

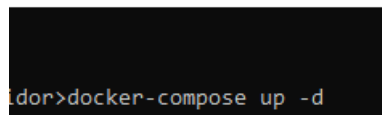
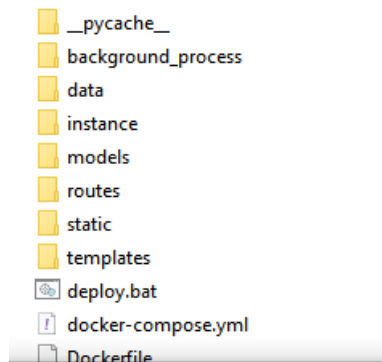
Name	Created	Size	Actions
mi-bd	1 minute ago	16.5 MB	Refresh Inspect Delete

2.1.2.2. Con docker-compose

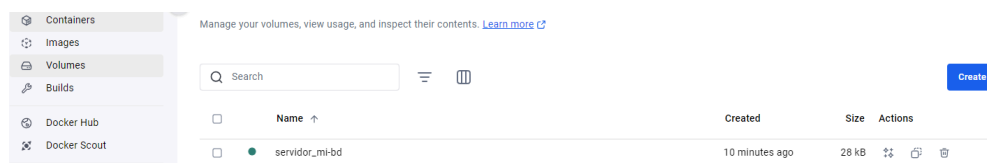
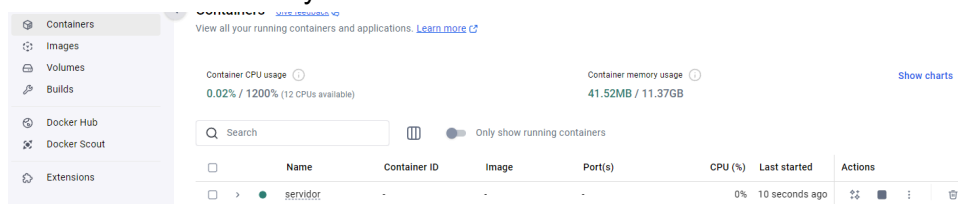
- Editaremos el archivo docker-compose.yml para decirle que puerto usaremos

```
docker-compose.yml
1 version: '3.8'
2
3 services:
4   serverdetectoria:
5     build:
6       context: .
7     image: serverdetectoria
8     container_name: serverdetectoria
9     restart: always
10    ports:
11      - "5000:5000"
12    volumes:
13      - mi-bd:/app/instance
14
15 volumes:
16   mi-bd:
```

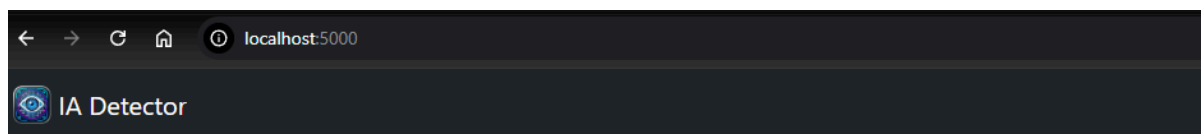
- Abrimos un cmd en la carpeta del servidor y ejecutamos este comando `docker-compose up -d` y también se montará automáticamente



- Se creará el contenedor y el volumen también



- Poniendo tu ip o localhost + el puerto en este caso 5000 podremos acceder al software.



Bienvenido a IA Detector

Sistema de detección y monitoreo de actividad sospechosa

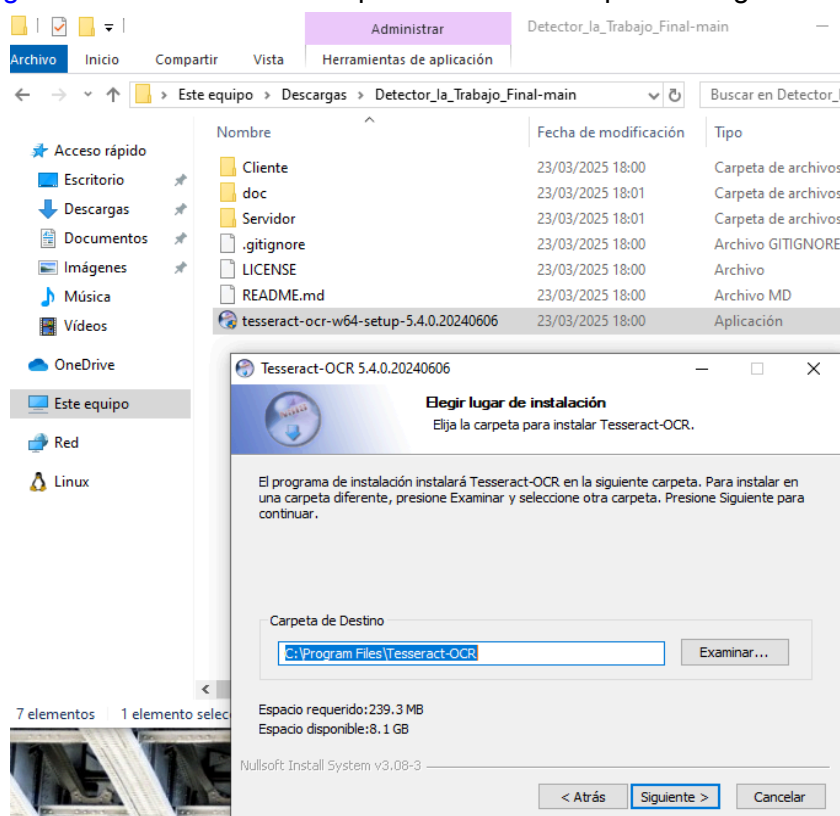
Inicia sesión para acceder al sistema.

Iniciar Sesión

2.2. Cliente

2.2.1. Anaconda (entorno virtual).

- En la carpeta del proyecto que descargamos de gitHub ejecutamos el tesseract, necesario para convertir las imágenes a texto. Seguimos los pasos del instalador y nos quedaremos con la ruta donde se ha instalado en este caso es [C:\Program Files\Tesseract-OCR](#) que necesitaremos para configurar el programa



- En la carpeta de Cliente editaremos el config.ini y pegaremos la ruta del tesseract justo con la ip y el puerto de nuestro servidor

Nombre	Fecha de modificación	Tipo	Tamaño
build	23/03/2025 18:00	Carpeta de archivos	
dist	23/03/2025 18:01	Carpeta de archivos	
cliente.py	23/03/2025 18:00	Archivo PY	9 KB
cliente_debug.spec	23/03/2025 18:00	Archivo SPEC	1 KB
cliente_hidden.spec	23/03/2025 18:00	Archivo SPEC	1 KB
config	23/03/2025 18:00	Opciones de confi...	1 KB
keywords.pkl	23/03/2025 18:00	Archivo PKL	1 KB
requirements	23/03/2025 18:00	Documento de te...	1 KB

```
config: Bloc de notas
Archivo Edición Formato Ver Ayuda
[Cliente]
url_servidor = http://192.168.0.14:5000
ruta = C:\\Program Files\\Tesseract-OCR\\tesseract.exe
```

- Para la instalación de anaconda seguir el punto **2.1.1.Opción 1: Anaconda (entorno virtual)**. del servidor en el apartado anterior.
- Abrimos una consola de Anaconda y con `cd` nos situamos en la carpeta del cliente y activamos el entorno de conda que creamos

```
Anaconda PowerShell Prompt
(base) PS C:\Users\Tutorial> cd C:\Users\Tutorial\Downloads\Detector_Ia_Trabajo_Final-main\Cliente
(base) PS C:\Users\Tutorial\Downloads\Detector_Ia_Trabajo_Final-main\Cliente> conda activate detector_env
(detector_env) PS C:\Users\Tutorial\Downloads\Detector_Ia_Trabajo_Final-main\Cliente>
```

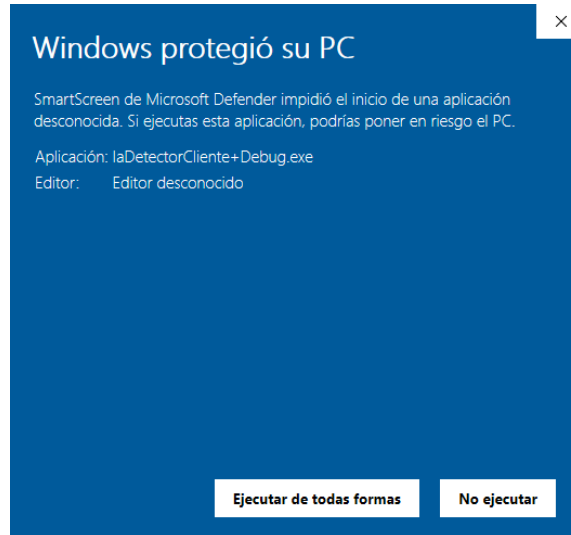
- Ejecutamos `pip instal -r requirements.txt` para instalar todas las librerías necesarias
- Ejecutamos el comando `python cliente.py` y el programa ya se pondría a correr

```
Anaconda PowerShell Prompt
Successfully built PyAutoGUI PyGetWindow pyscreeze pytweening mouseinfo pymsgbox pyrect pyperclip
Installing collected packages: pytweening, pyrect, pyperclip, pymsgbox, pyscreeze, PyGetWindow, packaging, mouseinfo, py
tesseract, PyAutoGUI
Successfully installed PyAutoGUI-0.9.54 PyGetWindow-0.0.9 mouseinfo-0.1.3 packaging-24.2 pymsgbox-1.0.9 pyperclip-1.9.0
pyrect-0.2.0 pyscreeze-1.0.1 pytesseract-0.3.13 pytweening-1.2.0
(detector_env) PS C:\Users\Tutorial\Downloads\Detector_Ia_Trabajo_Final-main\Cliente> python .\cliente.py
[2025-03-23 19:21:20.206011] Intentando descargar las palabras clave desde: http://192.168.0.18:5000/keywords
[2025-03-23 19:21:20.251495] Archivo de palabras clave descargado exitosamente.
[2025-03-23 19:21:20.251495] Cambio de ventana detectado: Anaconda PowerShell Prompt
['chatgpt', 'copilot', 'openai', 'bard', 'codeium', 'qwen', 'deepseek', 'davinci', 'codex', 'watson', 'lumen5', 'hivemind', 'chatbot', 'deepmind', 'cortex', 'automated', 'ai-based', 'ml-powered', 'ai-driven']
```

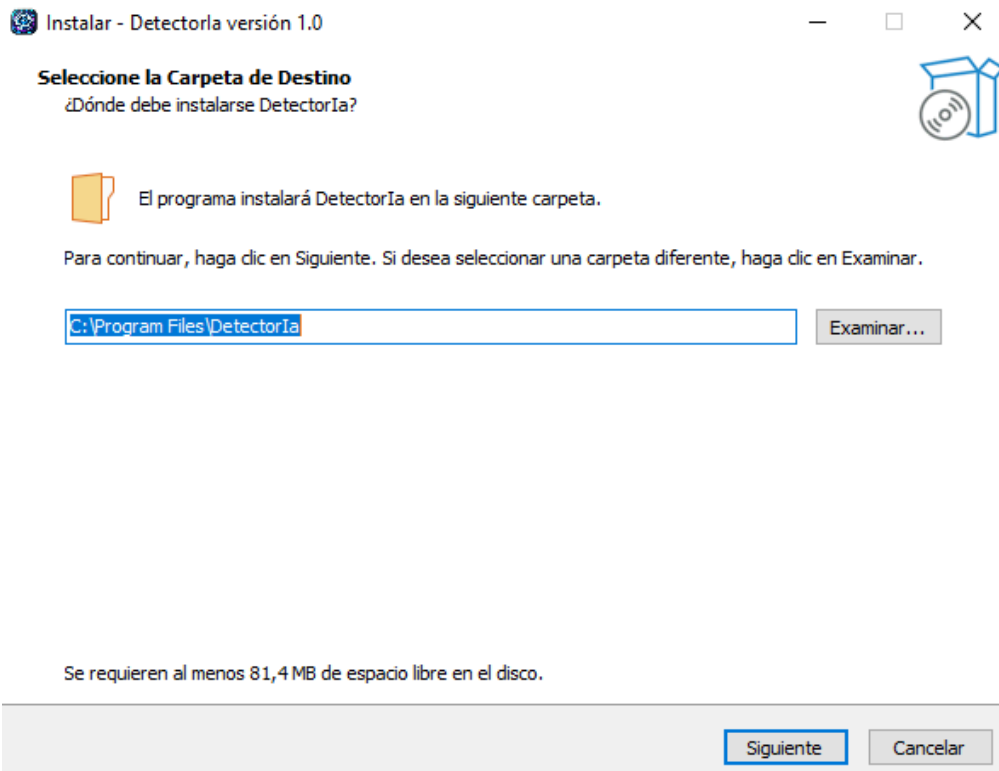


2.2.2.Instalador (recomendable).

- Ejecutamos el instalador `laDetectorInstaller.exe` y ya nos instalará el tesseract automáticamente, además nos copiará un ejecutable del cliente. Nos va a decir que es peligroso pero le damos a ejecutar igualmente.



- Elegimos donde queremos copiar nuestro programa. Deberemos ir más tarde para configurar el `config.ini`.





- Elegimos si queremos crear una carpeta en el menú de inicio

Instalar - DetectorIA versión 1.0

Seleccione la Carpeta del Menú Inicio
¿Dónde deben colocarse los accesos directos del programa?

☐ El programa de instalación creará los accesos directos del programa en la siguiente carpeta del
☐ Menú Inicio.

Para continuar, haga clic en Siguiente. Si desea seleccionar una carpeta distinta, haga clic en Examinar...

☐ No crear una carpeta en el Menú Inicio

- Si queremos crear un acceso directo en el menú o si queremos agregar al inicio de windows, marcamos la última casilla para que se inicie solo

Instalar - DetectorIA versión 1.0

Seleccione las Tareas Adicionales
¿Qué tareas adicionales deben realizarse?

Seleccione las tareas adicionales que desea que se realicen durante la instalación de DetectorIA y haga clic en Siguiente.

Accesos directos adicionales:
☐ Crear un acceso directo en el escritorio

Opciones de inicio:
☒ Iniciar automáticamente al encender Windows

- Introducimos la ip del servidor y la ruta de tesseract

Instalar - DetectorIA versión 1.0

Configuración del Detector IA

Configura los parámetros de conexión y OCR

Por favor, especifica la dirección del servidor y la ruta de Tesseract OCR:

URL del servidor:

http://192.168.0.23:5000

Ruta de Tesseract OCR:

C:\Program Files\Tesseract-OCR\tesseract.exe

Atrás

Siguiente

Cancelar

- Se instalará el tesseract al final y ejecutaremos el programa

Instalar - DetectorIA versión 1.0

Completando la instalación de DetectorIA

El programa completó la instalación de DetectorIA en su sistema. Puede ejecutar la aplicación utilizando los accesos directos creados.

Haga clic en Finalizar para salir del programa de instalación.

☒ Ejecutar DetectorIA

Finalizar

3.Despliegue del sistema

Configuración del entorno y ejecución en producción.

3.1.Servidor

El despliegue del servidor se realiza utilizando Docker, lo que permite ejecutar la aplicación de forma aislada, controlada y automática.

Gracias al script de despliegue incluido en el proyecto, el proceso de puesta en marcha es totalmente automático y solo requiere una configuración inicial mínima.

¿Por qué se usa Docker?

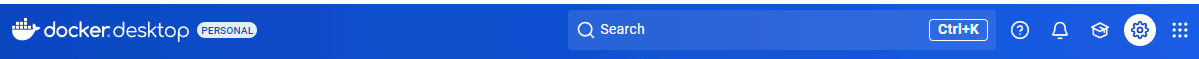
- Docker permite:
 - Ejecutar la aplicación con todas sus dependencias sin preocuparse por el sistema operativo.
 - Asegurar que el entorno de ejecución sea siempre el mismo (consistencia entre máquinas).
 - Controlar de forma sencilla el estado del servidor, incluso después de reinicios del sistema.

¿Qué hace exactamente el despliegue?

- Al ejecutar el script desplegar servidor.bat, se realiza lo siguiente:
 - Se construye la imagen Docker del servidor.
 - Se elimina cualquier contenedor anterior para evitar errores.
 - Se lanza el contenedor en segundo plano, con una configuración especial:
 - Se expone el puerto 5000 (puerto de Flask).
 - Se usa un volumen persistente para mantener los datos (base de datos SQLite).
 - Se aplica la política `--restart=always`, lo que significa que Docker reiniciará automáticamente el contenedor si se apaga o reinicia el sistema.

¿Qué se necesita hacer para que se inicie automáticamente?

- Nada más que asegurarse de que Docker Desktop está configurado para arrancar con Windows.
- Esto se hace una sola vez:
 - Abrir Docker Desktop.
 - Ir a Settings > General.
 - Activar la opción "Start Docker Desktop when you log in".



Con esto, no es necesario programar tareas adicionales, ni accesos directos, ni scripts extra.

Al encender el ordenador, Docker arranca, y con él se lanza automáticamente el servidor dentro del contenedor.

3.2. Cliente

El despliegue del cliente se basa en un instalador personalizado que automatiza el proceso completo de instalación y configuración. Este instalador copia el ejecutable [DetectorIA_hidden.exe](#) en la ruta protegida del sistema: [C:\Program Files\DetectorIA](#)

Una vez instalado, el instalador crea automáticamente un acceso directo al ejecutable dentro de la carpeta de inicio de Windows (shell:startup). Esto garantiza que el programa se inicie de forma automática cada vez que el usuario inicie sesión en el sistema, sin necesidad de ejecutar comandos adicionales ni intervenir manualmente.

El cliente está diseñado para ejecutarse en segundo plano, sin mostrar interfaz visible, lo que permite una vigilancia continua y discreta del sistema sin interrumpir el uso normal del equipo.

Conclusión

El cliente se integra de manera eficiente en el sistema operativo gracias a un instalador que lo ubica en una carpeta protegida y asegura su ejecución automática al inicio de sesión. Esta configuración evita que los alumnos interfieran con su ejecución y permite que el sistema funcione de forma autónoma desde el arranque, asegurando la monitorización activa desde el primer momento.

Anexo II: Manual de usuario

Este manual explica cómo utilizar el sistema para la detección de inteligencia artificial en exámenes y clases. Está dirigido a administradores y profesores, quienes gestionan y supervisan el monitoreo de los equipos en el aula.

1. Arquitectura general del sistema

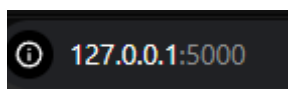
El sistema está compuesto por tres elementos clave: el servidor, la interfaz web y el cliente. Cada uno cumple una función específica y es necesario que al menos el servidor esté activo para que la aplicación funcione correctamente.

- **Servidor:** Es el núcleo del sistema. Gestiona la base de datos, expone las API y mantiene activa la lógica de negocio. Es indispensable que esté en ejecución para que tanto la interfaz web como el cliente pueda comunicarse con él.
- **Interfaz web:** Permite a los usuarios (profesores o administradores) acceder al sistema para gestionar usuarios, visualizar capturas, configurar horarios, etc. La interfaz web depende completamente del servidor. Si este no está activo, la interfaz no podrá cargar ni acceder a los datos.
- **Cliente:** Es una aplicación que debe estar en ejecución en los ordenadores que se desean monitorizar. Su función es tomar capturas de pantalla, analizarlas y, si detecta actividad sospechosa, enviarlas automáticamente al servidor. Aunque la interfaz web puede funcionar sin el cliente, no se recibirán nuevas capturas si el cliente no está activo.

Por tanto, para el funcionamiento completo del sistema (captura, análisis y visualización), se requiere que el servidor y el cliente estén en ejecución. La interfaz web actúa como herramienta de visualización y control, pero depende totalmente del servidor para operar.

2. Acceso y autenticación

Abrir un navegador web e ingresar la dirección del servidor proporcionada por el administrador.



Introducir el usuario y contraseña en la pantalla de inicio de sesión.

Según el rol del usuario, se mostrará el panel correspondiente:

- Administrador: Acceso completo a la gestión del sistema.
- Profesor: Acceso a la visualización de capturas y gestión de palabras clave

3.Funciones según el rol del usuario

3.1.Para administradores

3.1.1.Gestión de usuarios:

Crear, modificar y eliminar cuentas de profesores.

- Observar a todos los usuarios.

ID	Nombre	Administrador	Acciones
1	administrador	Si	
2	Janeiro	No	Cambiar Contraseña Eliminar
3	ivanB	No	Cambiar Contraseña Eliminar
4	Javier	No	Cambiar Contraseña Eliminar
5	Julia	No	Cambiar Contraseña Eliminar

- En el botón de Eliminar borraremos el usuario y todas sus capturas de pantalla

Eliminar

- El botón de cambiar contraseña nos abrirá un formulario para introducir una nueva contraseña aunque no sepas la anterior

Cambiar Contraseña

The screenshot shows the 'IA Detector' web application. On the left, there's a sidebar with 'Lista de Usuarios' and a 'Nuevo Usuario' button. The main area displays a table of users:

ID	Nombre
1	administrador
2	Janeiro
3	ivanB
4	Javier
5	Julia

A modal titled 'Cambiar Contraseña' is open, showing the process for user 'ivanB'. It has fields for 'Nueva Contraseña' and 'Confirmar Nueva Contraseña', with 'Cancelar' and 'Guardar Cambios' buttons at the bottom.

- Nuevo usuario nos llevará a una nueva ventana donde tendremos que introducir los datos del nuevo usuario

The screenshot shows the 'IA Detector' web application with the 'Registro de Usuario' form. The form includes fields for 'Nombre de usuario' (with 'Usuario' as a placeholder), 'Contraseña', and 'Confirmar contraseña' (with 'Confirmar contraseña' as a placeholder). A blue 'Registrarse' button is at the bottom. The footer shows '© 2025'.

- Seleccionaremos el día de la semana.

Día

Lunes

Lunes

Martes

Miércoles

Jueves

Viernes

Sábado

Domingo

- Seleccionamos hora de inicio

Hora inicio

--:--

11 05

12 06

13 07

14 08

15 09

16 10

17 11

- Seleccionamos hora de fin

Hora fin

--:--

🕒

11

06

12

07

13

08

14

09

15

10

16

11

17

12

- Pulsamos el botón de añadir horario. OJO nunca podrá haber 2 profesores en el mismo horario, te salta un error y el horario no se creará.

Añadir Horario

- En la tabla de abajo podremos seleccionar el día para ver los horarios que están creados y para quién están creados además de eliminarlos.

<div> Lunes Martes Miércoles Jueves Viernes Sábado Domingo </div>			
Usuario	Hora inicio	Hora fin	Acciones
Janeiro	18:20	18:22	Eliminar
ivanB	18:22	18:25	Eliminar
Javier	18:25	18:27	Eliminar
ivanB	18:37	18:40	Eliminar
Janeiro	18:45	18:47	Eliminar
ivanB	18:47	18:49	Eliminar
Javier	18:50	18:53	Eliminar

3.1.3. Visualización de capturas:

Consultar y gestionar las capturas realizadas.

- Entrar en el Panel de Control, ahí visualizamos todos los equipos que alguna vez han registrado una imagen.



Panel de Control

Control de Capturas

☒ Modo Automático

Estado del Sistema: Desactivado

Capturas por Equipo



I7-4790K

5 días



PORATILPOCHO

1 días

- Control de Capturas podemos cambiar de automático a manual y además ver para quien se están guardando las capturas en ese momento.
 - Modo Automático

Control de Capturas

☒ Modo Automático

Estado del Sistema: Desactivado



Control de Capturas

☒ Modo Automático

Estado del Sistema: Activado

Guardando para: **ivanB** (Modo Automático)

- Modo manual

Control de Capturas

☐ Modo Automático

Estado del Sistema: Activado

Guardando para: **administrador** (Modo Manual)

Detener Capturas

Control de Capturas

☐ Modo Automático


Estado del Sistema: Desactivado

Iniciar Capturas




- Capturas por Equipo podemos clicar en un equipo cualquiera, también nos da la información de cuántos días tiene capturas guardadas.

Capturas por Equipo



I7-4790K

5 días




PORATILPOCHO

1 días

- En el equipo podemos ver todos los días separados por carpetas, entrar en uno y ver las capturas.

[Panel de Control](#) / I7-4790K


Capturas de I7-4790K



2025-03-21

7 capturas


Eliminar carpeta



2025-03-18

1 capturas


Eliminar carpeta



2025-03-16

6 capturas


Eliminar carpeta



2025-03-13

8 capturas

Eliminar carpeta



2025-03-12

9 capturas

Eliminar carpeta

Panel de Control / I7-4790K / 2025-03-12

Capturas del 2025-03-12

20:39:35



Ordenador: I7-4790K Ventana: cliente.py - cliente - Visual Studio Code Texto detectado: %J File Edit...

Imagen Texto Eliminar

20:39:21



Ordenador: I7-4790K Ventana: cliente.py - cliente - Visual Studio Code Texto detectado: %J File Edit...

Imagen Texto Eliminar

20:38:40



Ordenador: I7-4790K Ventana: cliente.py - cliente - Visual Studio Code Texto detectado: %3 File Edit...

Imagen Texto Eliminar

20:38:29



Ordenador: I7-4790K Ventana: cliente.py - cliente - Visual Studio Code Texto detectado: %J File Edit...

Imagen Texto Eliminar

20:30:12



Fecha: 20250312_203012 Cliente: I7-4790K Ventana: cliente.py - cliente - Visual Studio Code Texto de...

Imagen Texto Eliminar

20:29:57



Fecha: 20250312_202957 Cliente: I7-4790K Ventana: config.ini - cliente - Visual Studio Code Texto de...

Imagen Texto Eliminar

- En el botón de imagen podremos descargar la Imagen, en el de Texto el .txt y en el botón Eliminar borraremos la captura de la base de datos.

3.1.4.Editar los keywords:

Consultar y modificar las palabras clave utilizadas para la detección de IA.

IA Detector

Panel de Control Usuarios Editar Keywords Horarios administrador Cerrar Sesión

Editar Palabras Clave

Descargar Archivo

Contenido del archivo keywords.txt:

chatgpt
copilot
openai
bard
codeium
qwen
deepseek
davinci
codex
watson
lumen5
hivemind
chatbot
deepmind
cortex
automated
ai-based
ml-powered
ai-driven

Guardar Cambios



- En el recuadro podremos escribir más palabras clave y darle a guardar, recomendable escribir una palabra clave por cada línea
- En el botón de Descargar Archivo dos descargamos el .txt por si lo queremos sustituir en otro equipo.

3.2. Para profesores

La parte de panel de control a la única que tiene acceso un profesor, el uso es igual que en el administrador.

- Visualización de capturas: Acceder a las imágenes capturadas por los clientes asignados.
- Descarga y eliminación de capturas: Gestionar las imágenes almacenadas en el sistema.
- Modo manual de capturas: Habilitar o deshabilitar la captura automática dentro de su sesión.
- Descarga y eliminación de capturas: Gestionar las imágenes almacenadas en el sistema.

4. Cierre de sesión

Al cerrar sesión te llevará de vuelta al login, el programa seguirá funcionando y modificando los horarios con normalidad.