UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Voice-Assisted Computer Accessibility

Presentado por Víctor Manuel Martínez Garcia
en Universidad de Burgos — 9 de junio
de 2025
Tutor: Pedro Luis Sanchez Ortega

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

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

Expone:

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

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

En Burgos, 9 de junio de 2025

Vº. Bº. del Tutor:

D. Pedro Luis Sanchez Ortega

Resumen

El proyecto **Voice-Assisted Computer Accessibility (VACA)** busca desarrollar un software accesible para personas con movilidad reducida, utilizando tecnologías de **Speech-to-Text (SST)**, **Text-to-Speech (TTS)** y **modelos de lenguaje avanzados**. La solución permite a los usuarios controlar su ordenador mediante prompts de voz, realizando tareas cotidianas sin necesidad de hardware especializado ni conocimientos técnicos. **VACA** promueve la autonomía digital e inclusión laboral, integrándose de manera sencilla en sistemas Windows.

Descriptores

Accesibilidad, **tecnologia de asistencia**, **agentes inteligentes**, **tecnología de voz**, **speech-to-text (SST)**, **text-to-speech (TTS)**, **IA**, **Windows**, **desarrollo de software**, **integración tecnológica**,**modelos de lenguaje (LLM)**,**tecnologías de la salud**,**autonomía digital**

Abstract

The **Voice-Assisted Computer Accessibility (VACA)** project aims to develop accessible software for people with reduced mobility, using **Speech-to-Text (SST)**, **Text-to-Speech (TTS)**, and **advanced language models** technologies. The solution allows users to control their computer through voice prompts, performing everyday tasks without the need for specialized hardware or technical knowledge. **VACA** promotes digital autonomy and workforce inclusion, easily integrating into Windows systems.

Keywords

Accessibility, **Assistive technology**, **Intelligent agents**, **Voice technology**, **Speech-to-text (SST)**, **Text-to-speech (TTS)**, **AI**, **Windows**, **Software development**, **Technological integration**, **Language models (LLM)**, **Health technologies**, **Digital autonomy**

Índice general

Índice general iii

Índice de figuras v

Índice de tablas vi

1. Introducción 1

2. Objetivos del proyecto 5
2.1. Objetivos específicos 5
2.2. Objetivos adicionales a futuro. 6
2.3. Objetivos personales 7

3. Conceptos teóricos 9
3.1. Agentes de uso de computadora (CUA) 9
3.2. Speech-to-Text (STT) 10
3.3. Text-to-Speech (TTS) 10
3.4. Inteligencia Artificial (IA) y Agentes Inteligentes 10
3.5. Modelos de Lenguaje Grande (LLM) 10
3.6. Modelos multimodales 11
3.7. Modelos de visión 11

4. Técnicas y herramientas 13
4.1. Herramientas utilizadas 14

5. Aspectos relevantes del desarrollo del proyecto 17
5.1. Inicio del Proyecto 17
5.2. Proceso de Adquisición de Conocimientos Técnicos 18

5.3. Implementación Inicial 19
5.4. Análisis de Herramientas Comunes en CUAs 19
5.5. Frameworks de Desarrollo 20
5.6. Librerías y Herramientas Destacadas 20
5.7. Modelos de Reconocimiento de Voz (ASR) 20
5.8. Limitaciones del Hardware y Solución 21
5.9. Conclusión del proceso 21

6. Trabajos relacionados **25**
6.1. Agentes de uso general basados en LLM 25
6.2. Sistemas alternativos de asistencia para personas con movilidad reducida 26
6.3. Comparativa funcional 26
6.4. Valor diferencial de VACA 26

7. Conclusiones y líneas de trabajo futuras **29**
7.1. Conclusiones 29
7.2. Líneas de trabajo futuras 32

Bibliografía **33**

Índice de figuras

Índice de tablas

6.1. Comparativa entre VACA y otras herramientas de asistencia basadas en modelos de lenguaje 28

1. Introducción

En la actualidad, la tecnología forma una parte esencial para la vida cotidiana, el acceso equitativo a sistemas informáticos sigue siendo un desafío para personas con movilidad reducida. El proyecto **Voice-Assisted Computer Accessibility (VACA)** busca desarrollar un software que facilite el uso del ordenador a individuos con movilidad reducida mediante la combinación de **computer use agents (CUA)** e integración de tecnologías **speech-to-text (SST)** y **text-to-speech (TTS)**.

Actualmente, los **usuarios con movilidad reducida** solo pueden acceder a sistemas operativos mediante el uso de **hardware específico**, los cuales pueden tener un **costo significativo**, generando así una **barrera tanto práctica como económica** para los usuarios.[4, 15, 10, 16]

Y otras soluciones actuales plantean el **conocimiento de 'comandos'** para la gestión de sistemas operativos específicos y los paquetes integrados del mismo como podría llegar a ser **Microsoft**[2], generando una **barrera técnica** para usuarios de edades más avanzadas, quienes a menudo no cuentan con el conocimiento necesario para interactuar con dichos comandos.

Además, **Windows Speech Recognition** presenta una limitación importante al estar diseñado principalmente para interactuar solo con aplicaciones del ecosistema de **Microsoft**, lo que restringe su funcionalidad.

A mayores, la creciente digitalización de los servicios, el **teletrabajo**, la **administración electrónica** y el **comercio online** ha generado nuevas oportunidades, pero también ha ampliado significativamente las **brechas digitales**. Este fenómeno no solo afecta a personas con movilidad reducida, sino también a otros colectivos que enfrentan dificultades en el acceso y uso de la tecnología.

En el caso de las personas con discapacidad motriz, estas barreras suelen manifestarse de manera doble: por un lado, la **falta de dispositivos asequibles** adaptados a sus necesidades; por otro, la **complejidad de los sistemas actuales**, que requieren **conocimientos técnicos específicos** para su manejo. De igual forma, muchos **usuarios de edad avanzada** también se ven afectados, al no contar con la formación tecnológica necesaria para desenvolverse en un entorno digital que evoluciona rápidamente.

Por todo ello, resulta **imprescindible** el desarrollo de **soluciones tecnológicas accesibles, intuitivas, abiertas y económicas** que promuevan una **verdadera inclusión digital**. La **accesibilidad informática** no debe entenderse como un añadido opcional, sino como una **garantía básica de participación social, laboral y educativa** en la sociedad actual.

Es esta base la motivación de la creación del proyecto **VACA** en conjunto con **ITCL** *(Instituto Tecnológico de Castilla y León)*, centro tecnológico el cual ha mostrado un gran compromiso con el **desarrollo de tecnologías orientadas a mejorar la calidad de vida y la salud en personas** con proyectos como:

- ***HostSmartAI*** [7]
- ***Iberus*** [8, 9]
- ***Wellsa***[6, 9]
- ***Rererevi***[5]

La propuesta de **VACA** incide en mejorar el **estado del arte** de las tecnologías de accesibilidad actuales, promoviendo un programa plug-and-play diseñado específicamente para integrarse en un **sistema Windows** con el uso de **tecnologías avanzadas** actuales como **modelos de lenguaje (LLM)** capaces de interpretar instrucciones complejas y contextuales mediante voz, integradas con sistemas **Speech To Text** y **Text To Speech** y tecnologías de interacción gráfica como **Omniparser** o **Browser-Use**. Siendo las únicas herramientas para el uso de esta tecnología, un micrófono y unos auriculares de uso cotidiano.

Estas herramientas proporcionarán al usuario la **capacidad de realizar acciones cotidianas** como la gestión de un correo electrónico, realización de compras en línea, tramitación de documentos, etc... Sin la necesidad de conocimiento previo, dado que los **modelos de lenguaje serán el cerebro** de las operaciones que el usuario quiera realizar.

El objetivo final de este proyecto será conseguir la implementación de un sistema que no solo contribuya a la **autonomía digital** de las personas con movilidad reducida, sino también **impulsar la inclusión laboral**, permitiendo a los usuarios desempeñar actividades profesionales que requieren principalmente del uso de un ordenador.

Además de lo anteriormente mencionado, el proyecto **generará conocimiento técnico de vanguardia** que podrá transferirse a los **ámbitos del sector salud**, fortaleciendo la capacidad tecnológica de **ITCL** en futuras iniciativas.

A futuro, dado que el proyecto requerirá de recursos computacionales significativos, el **ITCL** evaluará diferentes opciones técnicas para el soporte del mismo, desde el uso de servicios externos mediante **API (OpenAI - Anthropic)** hasta la implementación de servidores propios con GPUs capaces de ejecutar **modelos de lenguaje abiertos (Llama - Deepseek)**.

Desafíos

El proyecto podrá enfrentarse a grandes desafíos como la **precisión de reconocimiento de voz, reconocimiento de objetos en la interfaz gráfica, realización adecuada de los prompts del usuario**. Para superar las dificultades se procederá a usar **bibliotecas de IA preentrenadas de VLM (Anthropic-Claude)** y otras tecnologías que se puedan incorporar para el objetivo final como **HuggingFace-Coqui**[3] o **OpenAI-Whisper**[14].

Metodología y fases

El proyecto usa una metodología ágil para el seguimiento del mismo y los indicadores de éxitos finales incluirán la precisión en el reconocimiento de prompts de voz, la realización correcta de dichos prompts y la satisfacción del usuario final.

El proyecto será realizado desde **febrero hasta junio**, dividiéndose en las siguientes etapas/fases, las cuales estarán por dentro divididas en sprints:

- Investigación
- Desarrollo del software **back-end**
- Desarrollo del software **front-end**
- Pruebas del software
- Despliegue

2. Objetivos del proyecto

El proyecto **Voice-Assisted Computer Accessibility (VACA)** tiene como propósito principal mejorar la accesibilidad de las personas con movilidad reducida en el uso de sistemas informáticos. A través de la integración de tecnologías de **speech-to-text (SST)**, **text-to-speech (TTS)**, y **agentes inteligentes**, se busca desarrollar una solución accesible y eficiente que permita a los usuarios controlar el ordenador mediante prompts de voz y asistencia basada en inteligencia artificial.

2.1. Objetivos específicos

- **Desarrollar un software accesible** que permita a personas con movilidad reducida utilizar un ordenador sin necesidad de dispositivos físicos adicionales, utilizando solo prompts de voz y agentes inteligentes.

- **Integrar tecnologías de Speech-to-Text (SST)** para convertir el habla en texto y permitir que los usuarios controlen el sistema mediante prompts de voz.

- **Implementar la tecnología Text-to-Speech (TTS)** para proporcionar retroalimentación auditiva, mejorando la interacción del usuario con el sistema.

- **Incorporar agentes inteligentes basados en Inteligencia Artificial (IA)** que proporcionen asistencia personalizada y adaptativa a las necesidades individuales de los usuarios, permitiendo una mayor eficiencia en el control del ordenador.

- **Reducir la dependencia de hardware especializado** mediante el uso de soluciones basadas en software, lo que facilita el acceso a personas con movilidad reducida a un costo más bajo.

- **Evaluar la precisión del reconocimiento de voz** mediante pruebas de usabilidad, a fin de garantizar que el sistema cumpla con los requisitos de accesibilidad y facilidad de uso.

- **Evaluar la precisión del CUA** mediante pruebas de usabilidad, afinar el reconocimiento grafico del LLM con el entorno del sistema.

- **Realizar un despliegue efectivo del software,** permitiendo que los usuarios finales puedan instalar y utilizar el sistema en entornos de Windows.

2.2. Objetivos adicionales a futuro.

- **Soporte multilingüe:** Incorporar soporte para varios idiomas en el software, permitiendo que usuarios de diferentes partes del mundo puedan beneficiarse del sistema, adaptando el sistema de reconocimiento de voz y las respuestas TTS a distintos idiomas.

- **Interfaz de usuario adaptativa:** Desarrollar una interfaz gráfica de usuario (GUI) que se adapte a las necesidades y preferencias de cada usuario.

- **Compatibilidad con sistemas operativos adicionales:** Ampliar la compatibilidad del software para que sea accesible en otros sistemas operativos además de Windows, como macOS o Linux.

- **Funcionalidad de aprendizaje automático para mejorar la precisión del reconocimiento de voz:** Implementar un sistema de retroalimentación en tiempo real que permita al software aprender y adaptarse a las peculiaridades del habla del usuario, mejorando continuamente la precisión del reconocimiento de voz mediante el uso de algoritmos de aprendizaje automático.

- **Reduccion de costo de los modelos de lenguaje** utilizando prompt engineering para minimizar el uso de tokens, así mismo el costo del modelo, o integrando modelos de lenguaje de código abierto.

- **Implementación de una solución compacta y portatil** basandose en Jetson Orin, para proteger la privacidad y autonomía del usuario, permitiendo una gestión completamente local y segura de su información.

2.3. Objetivos personales

- **Formacion en LLM:** Como a futuro me gustaría llevar mi carrera profesional en la dirección de la rama de inteligencia artifical esto podría ser un buen comienzo y una buena toma de contacto con estas tecnologías.
- **Desarrollo de una aplicación con futuro de producción:** Poder crear una aplicación para futuro uso del ITCL me dará la oportunidad de enfocarme en detalles como la escalabilidad y modularidad del proyecto.
- **Dejar una huella en la empresa ITCL:** Dejar una marca por mi paso por el ITCL me haría ilusión y dejandoles este proyecto y el conocimiento de estas herramientas en sus manos, sería un gran logro para esto.

3. Conceptos teóricos

En este capítulo se presentan los conceptos teóricos clave que sustentan el desarrollo del proyecto **Voice-Assisted Computer Accessibility (VACA)**. Estos conceptos son fundamentales para entender el funcionamiento del sistema, las tecnologías empleadas y la problemática a la que se enfrenta. A continuación, se explican los principales términos relacionados con el **Computer Use Agents (CUA)**, **speech-to-text (SST)**, **text-to-speech (TTS)** y otros métodos empleados en la creación del software.

Estas tecnologías, aun que son diversas en naturaleza entre sí, deben integrarse sinergicamente entre ellas para proporcionar una experiencia fluida. Comprender el funcionamiento individual de estas permitira entender los retos a los que me enfrentare a la hora de la realización del proyecto.

3.1. Agentes de uso de computadora (CUA)

Los **Computer Use Agents (CUA)** son sistemas inteligentes diseñados para asistir a los usuarios en tareas específicas a través de interfaces graficos, usando los recursos disponibles del sistema, automatizando asi ciertas tareas que realizaria un usuario.

En el contexto de este proyecto, los CUA se utilizan para controlar el ordenador mediante prompts de voz haciendo asi el proceso de controlar un sistema mas accesible y eficiente para personas con movilidad reducida.

3.2. Speech-to-Text (STT)

El **Speech-to-Text** (STT) es una tecnología que convierte las palabras habladas en texto escrito. Este proceso se logra mediante el uso de algoritmos de reconocimiento de voz, que permiten a las máquinas comprender el lenguaje hablado. SST es una herramienta clave en este proyecto, ya que permite que los usuarios controlen el sistema informático sin necesidad de usar el teclado o el ratón, lo que facilita la accesibilidad para personas con movilidad reducida.

3.3. Text-to-Speech (TTS)

El **Text-to-Speech** (TTS) es una tecnología que convierte texto escrito en voz. Esta tecnología es crucial para proporcionar retroalimentación auditiva a los usuarios, permitiéndoles interactuar con el sistema de manera más eficiente. En este proyecto, el TTS se usa para leer instrucciones, mensajes de error o cualquier otra información relevante para el usuario.

3.4. Inteligencia Artificial (IA) y Agentes Inteligentes

La **Inteligencia Artificial** (IA) se refiere a la simulación de procesos de inteligencia humana mediante algoritmos y sistemas computacionales. En el contexto de este proyecto, la IA se utiliza para crear **agentes inteligentes** capaces de aprender y adaptarse a las necesidades específicas de los usuarios. Estos agentes son esenciales para ofrecer una experiencia personalizada y eficiente, ajustando el comportamiento del software a las características individuales de cada usuario.

3.5. Modelos de Lenguaje Grande (LLM)

Los **Modelos de Lenguaje Grande** son sistemas basados en inteligencia artificial que permiten a las máquinas comprender y generar texto de manera coherente. En este proyecto, los modelos de lenguaje son fundamentales para que los **agentes inteligentes** comprendan y respondan a los prompts de voz de los usuarios. Estos modelos mejoran la precisión y adaptabilidad del sistema, permitiendo una interacción más fluida y personalizada, especialmente para personas con movilidad reducida.

3.6. Modelos multimodales

Los **modelos multimodales** son sistemas de inteligencia artificial capaces de procesar y relacionar información proveniente de múltiples modalidades, como texto, voz, imágenes y video. Estos modelos integran distintos tipos de datos para comprender contextos complejos y ofrecer respuestas más precisas y contextualizadas. En el proyecto VACA, los modelos multimodales permiten combinar prompts de voz con información visual en pantalla, mejorando así la interacción entre el usuario y el sistema. Esta capacidad es totalmente necesaria para que el programa pueda ser consciente de lo que quiere transmitir el usuario y entender el entorno gráfico del usuario.

3.7. Modelos de visión

Los **modelos de visión por computadora** son algoritmos diseñados para interpretar y analizar imágenes o secuencias visuales, permitiendo que los sistemas comprendan elementos visuales como ventanas, botones, íconos o texto en pantalla. En el proyecto VACA, estos modelos seran utilizados para identificar componentes gráficos del sistema operativo y permitir al **CUA** ser consciente de lo que le rodea en el entorno gráfico.

4. Técnicas y herramientas

El desarrollo del proyecto **Voice-Assisted Computer Accessibility (VACA)** implica la utilización de diversas técnicas y herramientas que permiten alcanzar los objetivos definidos, garantizando que el software sea accesible, eficiente y funcional para personas con movilidad reducida.

A continuación, se describen las principales técnicas empleadas, así como las herramientas tecnológicas utilizadas durante las distintas fases del proyecto, incluyendo desarrollo, entrenamiento, pruebas y despliegue.

- **Procesamiento de Lenguaje Natural (PLN):** Se implementan tecnologías de reconocimiento y síntesis de voz, como **Speech-to-Text (STT)** y **Text-to-Speech (TTS)**, así como modelos de lenguaje para la comprensión e interpretación del texto generado. Estas técnicas permiten una interacción natural entre el usuario y el sistema.
- **Agentes Inteligentes basados en IA:** Se emplean agentes inteligentes que actúan como **Computer Use Agents (CUA)**, diseñados para razonar sobre el entorno gráfico y ejecutar acciones en nombre del usuario. Estos agentes combinan razonamiento contextual y visión artificial.
- **Procesamiento de interfaz gráfica y visión por computadora:** Se utilizan técnicas de detección de objetos, segmentación semántica y captioning de imágenes mediante modelos como YOLO y Florence, para que el sistema identifique y entienda los elementos gráficos del sistema operativo.
- **Metodología ágil (Agile):** Se adopta una metodología ágil basada en sprints y entregas iterativas, que permite adaptar el desarrollo

a cambios y mejoras continuas. Se aplica principalmente mediante herramientas de planificación como Zube.io.

- **Ingeniería de prompts:** Se utiliza *prompt engineering* para optimizar la interacción con modelos de lenguaje, reducir el uso de tokens y mejorar la eficiencia en respuestas.

4.1. Herramientas utilizadas

A continuación, se presentan las herramientas y tecnologías que se utilizarán para desarrollar el proyecto:

- **Lenguaje de programación Python:** Utilizado como lenguaje principal por su versatilidad, amplio ecosistema de bibliotecas y compatibilidad con los principales modelos de inteligencia artificial, visión artifical y procesamiento de lenguaje.
- **Modelos de inteligencia artificial:** Se emplean modelos como **Whisper** (STT) y **CUQUI-TTS** (TTS), además de modelos de lenguaje como **Claude (Anthropic)** y **GPT (OpenAI)**. También se integran modelos visuales como **YOLOv8** y **FlorenceV2**.
- **Interfaces de usuario:** Se desarrolla la interfaz utilizando **CTkinter**, una versión moderna y personalizable de Tkinter, con un enfoque minimalista y de alto contraste, ideal para accesibilidad.
- **Sistema operativo:** El software está optimizado inicialmente para **Windows**, aprovechando su compatibilidad con bibliotecas de automatización y visión. Se contempla la portabilidad futura a **Linux** y **macOS**.
- **Control de versiones:** Se utiliza **Git** como sistema de control de versiones, con almacenamiento en **BitBucket**, lo que permite un seguimiento detallado de cambios y colaboración eficiente.
- **Gestión de proyecto:** Se emplea la plataforma **Zube.io**, basada en tableros Kanban, para la gestión de tareas, seguimiento de sprints y control de entregables.
- **Entornos de desarrollo (IDE):** Se utiliza **Visual Studio Code (VSCode)** como entorno principal de desarrollo, dada su compatibilidad con extensiones, terminal integrada y soporte para Python y Docker.

- **Bibliotecas para modelos de lenguaje:** Se integran **LangChain** y **LangGraph** para la gestión de agentes y flujos conversacionales, así como **Transformers (HuggingFace)** para la ejecución de modelos personalizados.
- **Bibliotecas de automatización de escritorio: PyAutoGUI** se emplea para simular movimientos del ratón, clics, escritura de texto y otras acciones típicas de usuario. También se evalúa el uso de **Browser-Use** para exploración automatizada de interfaces web.
- **Docker y contenedores:** Se utiliza **Docker** para empaquetar y desplegar los modelos en contenedores independientes. Se desarrolla una arquitectura cliente-servidor mediante **FastAPI**, lo que permite migrar los procesos de inferencia a servidores ITCL y facilitar su reutilización.
- **OpenCV:** Librería clave para el procesamiento de imágenes, que permite transformar y manipular los outputs de los modelos visuales como YOLO o Florence.

5. Aspectos relevantes del desarrollo del proyecto

En este capítulo se detallan los aspectos más significativos durante el desarrollo del proyecto, desde las decisiones técnicas adoptadas, los obstáculos encontrados, hasta las soluciones implementadas. Se busca reflejar la evolución progresiva del sistema, la adquisición de competencias técnicas y la justificación de elecciones a la hora de dar forma al proyecto.

5.1. Inicio del Proyecto

La idea del proyecto surge como consecuencia del interés por una nueva línea de investigación en el ámbito de la inteligencia artificial: los **Computer-Use-Agents (CUAs)** [18, 11], una tecnología emergente destinada a automatizar la interacción con sistemas operativos mediante agentes inteligentes.

A partir de dicha investigación, se plantea el diseño y desarrollo de un agente asistencial que facilite el uso del ordenador a personas con movilidad reducida, combinando tecnologías como visión artificial, agentes con ASR.

5.2. Proceso de Adquisición de Conocimientos Técnicos

El desarrollo del proyecto requirió una fase intensiva de autoformación para adquirir los conocimientos técnicos necesarios. Se utilizaron plataformas formativas como DeepLearning.ai[1] y la documentación oficial de los modelos utilizados.

Aprendizaje sobre LLMs

Se profundizó en el funcionamiento y aplicación de los **Large Language Models (LLMs)**, abordando los siguientes conceptos:

- Procesamiento de texto y *text overflow*
- Recuperación aumentada con generación (RAG)
- Diseño de agentes
- Seguridad y mitigación de alucinaciones

Estudio de Computer-Use Agents

- Análisis técnico del CUA de Anthropic.
- Investigación del modelo **OmniParser** de Microsoft.
- Comparativa entre agentes existentes.

Tecnologías de Visión Artificial

- Implementación de detección de objetos con **YOLOv8** [17].
- Uso del modelo **FlorenceV2** de Microsoft para tareas de captioning visual [12].
- Integración con OpenCV para procesamiento intermedio de imágenes.

5.3. Implementación Inicial

Durante las primeras etapas, se realizaron pruebas con el modelo de Anthropic, encontrando diversas limitaciones:

- Alto consumo de memoria RAM en entornos Docker.
- Inestabilidad del WebSocket de comunicación.
- Escasa comprensión del entorno gráfico por parte del agente.

Posteriormente se experimentó con **OmniParser**, un modelo más avanzado pero con limitaciones clave:

- Requiere entorno Docker exclusivo.
- Falta de soporte para STT (Speech-to-Text) y TTS (Text-to-Speech).
- Proceso de instalación complejo y poco robusto.

5.4. Análisis de Herramientas Comunes en CUAs

Del estudio de los modelos existentes se extrajo una arquitectura común compuesta por tres pilares imprescindibles:

1. Un **modelo de visión** capaz de detectar y clasificar iconos o elementos visuales.
2. Un **modelo multimodal** que entienda imágenes y razone sobre ellas.
3. Un **sistema de simulación de entradas** (mouse, teclado) para ejecutar acciones.

Para dar soporte a este esquema, se integraron los siguientes elementos:

- YOLOv8 para bounding boxes (ajustando parámetros para reducir falsos positivos).
- FlorenceV2 como modelo de captioning descriptivo.
- OpenCV como middleware de visualización y conexión.

5.5. Frameworks de Desarrollo

El agente fue diseñado como un sistema modular utilizando los frameworks **LangChain** y **LangGraph**, adaptando sus herramientas para admitir llamadas a modelos como OpenAI y Anthropic.

Se identificó un problema en el uso intensivo de tokens en el análisis de imágenes por parte de LangChain (hasta 100,000 tokens), en contraste con implementaciones dadas por las librerias de los propios modelos. (10,000 tokens con Anthropic).

5.6. Librerías y Herramientas Destacadas

Para el desarrollo del sistema y la interacción con el entorno operativo se utilizaron librerías clave:

- **Tkinter**: interfaz gráfica de usuario.
- **PyAutoGUI**: simulación de eventos de teclado y ratón.
- **OpenCV**: procesamiento de imagen.

5.7. Modelos de Reconocimiento de Voz (ASR)

Para la transcripción y síntesis de voz se emplearon los siguientes modelos los cuales se encontraban alojados en los servidores del ITCL:

- **Whisper (OpenAI)**: transcripción STT.
- **CUQUI-TTS**: síntesis TTS.

Las llamadas a estos servicios se integraron mediante endpoints personalizados del ITCL, produciendo archivos `.wav` y texto interpretado para acciones.

5.8. Limitaciones del Hardware y Solución

El entorno de desarrollo ofrecido por ITCL disponía de hardware limitado, afectando directamente al rendimiento de los modelos más pesados como Florence.

Como solución, se desplegó un entorno Docker para Florence y YOLO en servidores ITCL, permitiendo:

- Aceleración por GPU del servidor.
- Disponibilidad de los modelos para otros proyectos.
- Reducción de los tiempos de inferencia en un **79 %**.

Se crearon dos variantes:

- Contenedor único con ambos modelos integrados.
- Dos contenedores independientes comunicados por red.

5.9. Conclusión del proceso

El desarrollo de este proyecto representó un desafío considerable, especialmente debido al desconocimiento inicial sobre muchas de las tecnologías involucradas. Ámbitos como la visión artificial, el procesamiento del lenguaje natural y la automatización de entradas, etc...

Errores y resolución de problemas

Durante el desarrollo, se presentaron numerosos errores y dificultades técnicas que fueron resueltos de forma iterativa. A continuación, se enumeran los principales problemas detectados y las soluciones implementadas:

- **Compatibilidad entre librerías**

  Se detectaron conflictos entre la versión de Python y diversas dependencias, especialmente con LangChain. Estos problemas se solventaron utilizando entornos virtuales aislados y controlando manualmente las versiones instaladas mediante archivos `PyProject.toml`.

- **Problemas de rendimiento con modelos pesados**

  El equipo de desarrollo contaba con una GPU no compatible con CUDA, lo que afectó al rendimiento de modelos como Florence. Como solución, se dockerizaron tanto YOLOv8 como Florence y se desplegaron en los servidores del ITCL, aprovechando así recursos de cómputo con aceleración por GPU.

- **Evaluación de Streamlit**

  Se valoró inicialmente utilizar Streamlit como interfaz, pero tras discusión con el tutor del ITCL, se optó por desarrollar una aplicación autónoma con su propia interfaz gráfica. Esta decisión permite una mayor escalabilidad y posibilidad de integración con procesos del sistema operativo, como la ejecución al inicio del sistema.

- **Falsos positivos en YOLOv8**

  El modelo de visión YOLOv8 presentaba falsos positivos en la detección de elementos gráficos. Se mitigaron ajustando los umbrales de confianza y refinando las etiquetas asociadas a los objetos.

- **Uso elevado de tokens en LangChain**

  LangChain llegaba a consumir hasta 100.000 tokens al procesar imágenes. Esto fue optimizado utilizando directamente las llamadas de los modelos sin pasar por la interfaz de LangChain, reduciendo el uso de tokens a aproximadamente 10.000.

- **Gestión de memoria en ejecución**

  El sistema de memoria no cuenta actualmente con una ventana de contexto adaptable, lo que provoca un crecimiento continuo de la misma. Se identificó un problema al intentar resumir mensajes usando herramientas de LangChain, ya que las respuestas esperaban una estructura distinta. Se propone como solución futura separar mensajes de usuario/LLM de las herramientas, resumir solo los primeros y reenviarlos procesados.

- **Problemas con PyAutoGUI**

  Esta librería no gestionaba correctamente caracteres UTF-8 como tildes o la letra "ñ". Se resolvió sustituyendo esta funcionalidad por la librería `keyboard`, que ofrecía una mayor compatibilidad.

- **Errores al escribir código**

El agente generaba código con errores de indentación. Se solucionó utilizando la librería `pyperclip`, que permite copiar el texto generado y pegarlo con la indentación original preservada.

- **Limitaciones con VPN y servidores remotos**

  Al utilizar los workers del ITCL desde entornos externos, se detectaron problemas relacionados con la conexión VPN. Esta cuestión depende directamente de la administración del centro y escapa a mi control.

- **Resolución de imágenes para Florence**

  Las descripciones generadas por Florence fallaban si la imagen recortada era demasiado pequeña. Se resolvió escalando las imágenes a una resolución mínima de 64x64 píxeles. Aunque sería posible implementar un escalado proporcional dinámico, se decidió mantener esta solución por simplicidad y eficiencia.

- **Prompts inconsistentes con Claude**

  Claude, en ocasiones, ignoraba las instrucciones relacionadas con el uso del diccionario de coordenadas, inventando valores. Este comportamiento se corrigió parcialmente mediante prompts más restrictivos. No obstante, persisten ocasionales alucinaciones. Una solución definitiva podría pasar por un fine-tuning de un modelo opensource o implementar validación externa sobre las coordenadas generadas.

- **Fallos con modelos multimodales opensource**

  Se intentó utilizar modelos alojados en los servidores del ITCL con capacidades multimodales, así como integrarlos con `browser_use`. Sin embargo, fallaron al interpretar correctamente imágenes o establecer conexiones con el navegador. Dado que otros usuarios también reportaron errores similares con modelos como DeepSeek u Ollama, se decidió posponer la integración de modelos opensource y continuar con APIs comerciales como Claude y GPT.

Estas situaciones no solo requirieron solución técnica, sino también habilidades de investigación, prueba y documentación, fortaleciendo el aprendizaje obtenido a lo largo del proyecto.

Lecciones aprendidas y competencias adquiridas

A lo largo del proceso, no solo se lograron los objetivos propuestos, sino que también se adquirieron competencias fundamentales como :

- Diseño e implementación de agentes inteligentes capaces de interactuar con entornos visuales complejos.
- Optimización del uso de recursos en entornos de cómputo limitados, tanto a nivel local como distribuido.
- Evaluación crítica de tecnologías emergentes en términos de rendimiento, escalabilidad y aplicabilidad.
- Aplicación de principios de modularidad y escalabilidad en el diseño de sistemas, favoreciendo su mantenimiento y evolución futura.

6. Trabajos relacionados

Dado que los **Computer Use Agents (CUA)** son una tecnología emergente, actualmente no existen desarrollos consolidados que funcionen de forma nativa en sistemas operativos como Windows y que estén específicamente dirigidos a personas con movilidad reducida. Sin embargo, sí es posible identificar ciertos proyectos y prototipos relevantes que han explorado esta línea, ya sea desde la perspectiva de agentes inteligentes o desde soluciones de accesibilidad alternativas.

6.1. Agentes de uso general basados en LLM

En el campo de los CUAs genéricos se han desarrollado algunos agentes capaces de interactuar con interfaces gráficas mediante razonamiento sobre el estado del sistema. A continuación, se presentan los más destacados:

- **OmniParser** [18]: agente de Microsoft enfocado en la interacción con interfaces de usuario mediante visión artificial, soportando múltiples modelos de lenguaje. Funciona exclusivamente sobre entornos Docker.
- **Anthropic CUA** [11]: primer intento funcional de agente multimodal orientado al control de interfaz gráfica, basado en el modelo Claude. Requiere entorno dockerizado y presenta limitaciones de comprensión visual.
- **Operator (OpenAI)** [13]: demo cerrada y de pago desarrollada por OpenAI, orientada a la interacción autónoma con entornos web mediante el uso exclusivo de sus propios modelos. Sin código abierto ni soporte extensible.

6.2. Sistemas alternativos de asistencia para personas con movilidad reducida

Aunque los CUAs específicos para accesibilidad aún están en fase exploratoria, existen múltiples investigaciones y sistemas orientados a mejorar la interacción de personas con movilidad reducida con los sistemas informáticos anteriores a estes agentes, entre ellos,los que destacan son:

- **Eye Control** [4]: sistemas de seguimiento ocular que permiten controlar el cursor del ratón mediante el movimiento de los ojos.
- **Hands-Free** [16]: interfaces basadas en movimientos de cabeza y reconocimiento de gestos para simular entradas de usuario.
- **Brain-Computer Interface (BCI)** [15]: dispositivos que detectan señales eléctricas del cerebro para traducirlas en acciones computacionales, aunque de uso limitado por su complejidad y coste.
- **Tongue Drive** [10]: tecnología que permite el control de dispositivos mediante movimientos de la lengua, especialmente útil para usuarios con cuadriplejía.

6.3. Comparativa funcional

La siguiente tabla resume una comparativa entre VACA y otros CUAs genéricos disponibles, considerando aspectos clave como integración con voz, ejecución en entorno local, modelos soportados y accesibilidad de uso:

6.4. Valor diferencial de VACA

El sistema VACA se presenta como una alternativa accesible, extensible y de bajo coste, centrada desde su origen en usuarios con movilidad reducida. A diferencia de otras soluciones que actúan como pruebas de concepto o herramientas de desarrollo, VACA ofrece:

- Integración completa de entrada y salida por voz.
- Soporte nativo para Windows sin necesidad de contenedores.
- Modularidad para incorporar diferentes modelos de lenguaje (Aunque no esté implementada actualmente al 100 %).

- Interfaz gráfica simple, funcional y accesible.

Característica	**VACA**	**OmniParser**	**Operator**	**Anthropic**
STT y TTS integrados	Sí	No	No	No
Ejecución en local (nativo Windows)	Sí	Solo Docker	Solo Docker	Solo Docker
Detección de objetos	Sí	Sí	Sí	No
Integración de voz	Sí	No	No	No
Modelos multimodales soportados	Cualquiera (configurable)	Qwen, DeepSeek, OpenAI, Claude	Solo OpenAI	Solo Claude
Código abierto	Sí	Sí	No	Sí
Interfaz gráfica	GUI nativa	WebUI	WebUI	WebUI
Requiere pocos recursos	Sí	No especificado	No especificado	No especificado
Costo de uso	Gratuito	Gratuito	200€ (demo)	Gratuito

Tabla 6.1: Comparativa entre VACA y otras herramientas de asistencia basadas en modelos de lenguaje

7. Conclusiones y líneas de trabajo futuras

7.1. Conclusiones

En este apartado se exponen las principales conclusiones obtenidas a lo largo del desarrollo del proyecto **Voice-Assisted Computer Accessibility (VACA)**. Este trabajo ha supuesto no solo un reto técnico, sino también una experiencia de aprendizaje integral, tanto a nivel tecnológico como en la gestión de un proyecto de software con potencial de aplicación real.

Cumplimiento de los objetivos

Objetivos funcionales

Se han alcanzado los objetivos funcionales planteados al inicio del proyecto. Entre los logros más destacados se encuentran:

- El desarrollo de una solución accesible que permite controlar un sistema operativo mediante comandos de voz, sin necesidad de dispositivos de entrada físicos como ratón o teclado.
- La integración completa de tecnologías de **Speech-to-Text (STT)** y **Text-to-Speech (TTS)** usando modelos como *Whisper* y *CUQUI-TTS*, ofreciendo una interacción fluida con el sistema.
- La implementación de **Computer Use Agents** capaces de combinar visión artificial, modelos de lenguaje y automatización de acciones dentro del entorno gráfico del sistema operativo.

- La ejecución local del sistema en Windows de forma nativa, sin necesidad de contenedores o entornos virtualizados, facilitando su uso desde cualquier equipo personal.
- El despliegue eficiente de los modelos más pesados (como FlorenceV2 y YOLO) mediante Docker en servidores del ITCL, lo que permitió reducir los tiempos de inferencia en más de un 70 %.

Objetivos no funcionales

- Se ha priorizado la modularidad y escalabilidad del sistema, facilitando futuras mejoras, mantenimiento y extensión a nuevos contextos.
- El sistema ha sido desarrollado como una solución gratuita y de código abierto, con un bajo consumo de recursos, lo que lo hace viable en entornos con limitaciones técnicas o económicas.
- Se ha diseñado una interfaz gráfica funcional y sencilla usando **CTkinter**, con el objetivo de que cualquier persona pueda utilizarla, incluso con pocos conocimientos informáticos.

Objetivos personales

- Este proyecto me ha permitido adentrarme formalmente en el campo de los **modelos de lenguaje (LLM)** y en el uso de agentes inteligentes, un área en la que quiero seguir especializándome.
- He aplicado metodologías ágiles a lo largo del desarrollo del proyecto, gestionando tareas, sprints y versiones de código mediante herramientas como *Zube.io*, *BitBucket* y *GitHub*.
- Me enorgullece haber creado una herramienta con potencial de uso real, que quedará como una aportación al ITCL y podrá servir como base para desarrollos futuros.
- Este trabajo ha sido el broche final de mi formación como ingeniero informático, permitiéndome aplicar muchos de los conocimientos adquiridos durante el grado de una manera práctica y con sentido.

Reflexiones sobre el proceso

Uno de los retos más grandes fue trabajar con tecnologías que están aún en desarrollo y para las que, muchas veces, apenas existe documentación. Cada pocas semanas aparecían nuevos modelos o herramientas, lo que me obligó a adaptarme constantemente y tomar decisiones rápidas.

Algunas de las dificultades más destacadas fueron:

- Las limitaciones de algunos modelos existentes, como el CUA de Anthropic o el Omniparser de Microsoft, que llevaron a crear soluciones propias más ajustadas a las necesidades reales del proyecto.
- El trabajo con hardware limitado, que exigió optimizar recursos al máximo y trasladar parte de los procesos a servidores remotos del ITCL mediante Docker.
- La integración de herramientas complejas como LangChain, PyAutoGUI, YOLOv8, FlorenceV2, FastAPI y Docker, que requirió tiempo, paciencia y muchas pruebas hasta lograr un sistema robusto y funcional.

Valoración personal

Para mí, el desarrollo de VACA ha sido mucho más que un proyecto de clase. Ha sido una oportunidad para crear algo útil, con impacto potencial en la vida de personas con movilidad reducida. Me siento orgulloso de haber conseguido una solución funcional.

Además, ha sido una excelente forma de comenzar mi camino dentro del mundo de la inteligencia artificial, un área que me motiva especialmente y en las que quiero seguir creciendo profesionalmente.

VACA no es solo un software funcional; representa una forma de poner la tecnología al servicio de quienes más la necesitan.

7.2. Líneas de trabajo futuras

Aunque VACA ya es un sistema funcional, su arquitectura modular permite ampliaciones y mejoras que podrían marcar una gran diferencia. Algunas de las líneas de trabajo que me gustaría destacar son:

- **Soporte multilingüe:** Añadir compatibilidad con varios idiomas para que personas de diferentes países puedan beneficiarse del sistema.
- **Versión embebida:** Crear una versión del sistema que funcione de forma local en dispositivos compactos como Jetson Orin, garantizando autonomía y privacidad total.
- **Entrenamiento personalizado del STT:** Permitir que el sistema se adapte a la voz concreta del usuario, mejorando la precisión y la comodidad en el uso diario.
- **Compatibilidad con otros sistemas operativos:** Llevar VACA a Linux y macOS, ampliando así su disponibilidad y uso en otros entornos.
- **Interfaz gráfica adaptativa:** Mejorar la GUI para que se ajuste a las necesidades visuales o cognitivas del usuario, permitiendo configuraciones avanzadas y personalizadas.
- **Sistema de retroalimentación:** Añadir una función que permita al usuario corregir errores o malentendidos del agente para mejorar la experiencia general.
- **Conexión con herramientas externas:** Integrar el sistema con aplicaciones como navegadores, calendarios o gestores de tareas para extender su funcionalidad como asistente personal.
- **Validación con usuarios reales:** Realizar pruebas con personas con movilidad reducida para validar la utilidad del sistema, recoger sugerencias reales y priorizar futuras mejoras.
- **Mejora del rendimiento de los modelos de visión:** Entrenar los modelos YOLOv8 y FlorenceV2 con datasets propios para afinar su precisión y adaptarlos aún más al contexto de escritorios y sistemas operativos.

Bibliografía

[1] Home — deeplearning.ai. https://www.deeplearning.ai/. [Accessed 10-03-2025].

[2] Windows Speech Recognition commands - Microsoft Support — support.microsoft.com. https://support.microsoft.com/en-us/windows/windows-speech-recognition-commands-9d25ef36-994d-f367-a81a-a326160128c7. [Accessed 11-03-2025].

[3] Coqui. XTTS-v2 — Hugging Face. https://huggingface.co/coqui/XTTS-v2, 2024. [Accessed 3-06-2025].

[4] James Gips and Peter Olivieri. Eagleeyes: An eye control system for persons with disabilities. In *The eleventh international conference on technology and persons with disabilities*, pages 1–15, 1996.

[5] ITCL. RERÉREVI - Rehabilitación con Realidad Virtual — itcl.es. https://itcl.es/proyectos-srv/rererevi-rehabilitacion-residencias-realidad-virtual/, 2015. [Accessed 12-03-2025].

[6] ITCL. Monitorización en tiempos de coronavirus mediante telemedicina — itcl.es. https://itcl.es/blog/monitorizacion-en-tiempos-de-coronavirus/, 2020. [Accessed 12-03-2025].

[7] ITCL. HosmartAI: Soluciones con IA en Medicina — itcl.es. https://itcl.es/proyectos-europeos/hosmartai-desarrollo-inteligente-en-hospitales/, 2021. [Accessed 12-03-2025].

[8] ITCL. IBERUS- Ingeniería Biomédica para Enfermedades Degenerativas — itcl.es. https://itcl.es/proyectos-nacionales/iberus-

ingenieria-biomedica-enfermedades-degenerativas/, 2021. [Accessed 12-03-2025].

[9] ITCL. ITCL presenta en (FITECU) Wellsa e Iberus: dos proyectos tecnológicos que suman IA a la mejora de la calidad de vida — itcl.es. https://itcl.es/itcl-noticias/itcl-presenta-en-fitecu-wellsa-e-iberus-dos-proyectos-tecnologicos-que-suman-ia-a-la-mejora-de-la-calidad-de-vida/, 2023. [Accessed 12-03-2025].

[10] G. Krishnamurthy and M. Ghovanloo. Tongue drive: a tongue operated magnetic sensor based wireless assistive technology for people with severe disabilities. In *2006 IEEE International Symposium on Circuits and Systems*, pages 4 pp.–, 2006.

[11] Zak Lee, Alex Albert, Nathan McCandlish, Zachary Lee, Brandon Macer, Andrew Edstrom, Chris Gorgolewski, Dave Hansen, Hisham Khalifa, Ikko Eltociear Ashimine, James Ward, Jesse Hu, Jesús Ferretti, Kevin Ji, Kirill Pertsev, Peter Raboud, Rahim Nathwani, Tynan Daly, Yijing Barry Zhang, and laiso. *anthropics/anthropic-quickstarts*. 2 2025.

[12] Microsoft. Florence-2-base — Hugging Face. https://huggingface.co/microsoft/Florence-2-base, 2024. [Accessed 02-05-2025].

[13] OpenAI. Operator OpenAI. https://openai.com/index/introducing-operator/. [Accessed 10-03-2025].

[14] OpenAI. Whisper-large-v3 — Hugging Face. https://huggingface.co/openai/whisper-large-v3, 2024. [Accessed 3-06-2025].

[15] Gabriel Pires, Urbano Nunes, and Miguel Castelo-Branco. Evaluation of brain-computer interfaces in accessing computer and other devices by people with severe motor impairments. *Procedia Computer Science*, 14:283–292, 2012. Proceedings of the 4th International Conference on Software Development for Enhancing Accessibility and Fighting Info-exclusion (DSAI 2012).

[16] Atieh Taheri, Ziv Weissman, and Misha Sra. Exploratory design of a hands-free video game controller for a quadriplegic individual. In *Proceedings of the Augmented Humans International Conference 2021*, AHs '21, page 131–140, New York, NY, USA, 2021. Association for Computing Machinery.

[17] Ultralytics. Inicio — docs.ultralytics.com. https://docs.ultralytics.com/es. [Accessed 10-03-2025].

[18] yadong lu, Thomas Dhome-Casanova, Billy Cao, Neil Stoker, Andriy Tkach, Krishna Srinivasan, and microsoft-github-policy service[bot]. *microsoft/OmniParser*. 2 2025.