Tecnológico de Costa Rica

Escuela de Ingeniería en Computadores
(Computer Engineering School)

Programa de Licenciatura en Ingeniería en Computadores
(Licentiate Degree Program in Computer Engineering)

Implementación de un sistema embebido para control de acceso en gimnasios mediante reconocimiento facial

(Implementation of an embedded system for access control in gyms using facial recognition)

Informe de Trabajo de Graduación para optar por el título de Ingeniero en Computadores con grado académico de Licenciatura

(Report of Graduation Work in fulfillment of the requirements for the degree of Licentiate in Computer Engineering)

Luis Pedro Morales Rodríguez

Cartago, junio de 2025
(Cartago, June, 2025)

Declaro que el presente Proyecto de Graduación ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos propios.

En los casos en que he utilizado bibliografía he procedido a indicar las fuentes mediante las respectivas citas bibliográficas. En consecuencia, asumo la responsabilidad total por el trabajo de graduación realizado y por el contenido del correspondiente informe final.

Luis Pedro Morales Rodríguez

Cartago, 30 de junio de 2025

Céd: 9-0109-0674

Instituto Tecnológico de Costa Rica
Área Académica de Ingeniería en Computadores
Proyecto de Graduación
Tribunal Evaluador

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniero en Computadores con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal

Profesor Lector	Profesor Lector

Pedro Gutiérrez García
Profesor Asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por el Área Académica de Ingeniería en Computadores.

Cartago, junio de 2025

Instituto Tecnológico de Costa Rica
Área Académica de Ingeniería en Computadores
Proyecto de Graduación
Tribunal Evaluador
Acta de Evaluación

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniero en Computadores con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Estudiante: *Luis Pedro Morales Rodríguez*

Nombre del Proyecto: *Implementación de un sistema embebido para control de acceso en gimnasios mediante reconocimiento facial*

Miembros del Tribunal

_____ _____

Profesor Lector Profesor Lector

_____

Pedro Gutiérrez García
Profesor Asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por el Área Académica de Ingeniería en Computadores.

Nota final del Proyecto de Graduación: _____

Cartago, junio de 2025

Resumen

Palabras clave: PALABRAS CLAVE

Abstract

Keywords:

DEDICATORIA

Agradecimientos

Luis Pedro Morales Rodríguez

Cartago, 30 de junio de 2025

Índice general

Índice de figuras

Índice de tablas

Lista de símbolos y abreviaciones

Abreviaciones

API	Interfaz de Programación de Aplicaciones
AUGE	Agencia Universitaria para la Gestión Emprendedora
BaaS	Backend como Servicio
BSP	Paquete de Soporte de Placa
CAMTIC	Cámara de Tecnologías de Información y Comunicación
CDN	Red de Distribución de Contenidos
CNN	Redes Neuronales Convolucionales
FLOPs	Operaciones de Punto Flotante
FPS	Cuadros por Segundo
HOG	Histograma de Gradientes Orientados
IaaS	Infraestructura como Servicio
IoT	Internet de las Cosas
ITCR	Instituto Tecnológico de Costa Rica
LTS	Soporte a Largo Plazo
MMU	Unidad de Manejo de Memoria
MVP	Producto Mínimo Viable
OE	Objetivo Específico
PaaS	Plataforma como Servicio
PIT	Proyectos de Innovación Tecnológica
SaaS	Software como Servicio
SBD	Sistema de Banca para el Desarrollo
SDK	Kit de Desarrollo de Software
UAT	Pruebas de Aceptación de Usuario

Capítulo 1

Introducción

En la última década, el crecimiento de la industria del *fitness* ha impulsado una mayor demanda de servicios personalizados y sistemas tecnológicos que optimicen la gestión de gimnasios y centros deportivos. Esta evolución ha motivado a las organizaciones a implementar soluciones innovadoras para mejorar la experiencia del usuario y aumentar su eficiencia operativa. Sin embargo, uno de los desafíos comunes en gimnasios de mediana y gran escala es la necesidad de un sistema de control del acceso de sus clientes a las instalaciones, que facilite la validación de sus respectivas membresías de una forma segura y automatizada, optimizando así el uso de recursos y protegiendo la seguridad de los usuarios.

El presente proyecto propone desarrollar un sistema de control de acceso basado en reconocimiento facial, diseñado específicamente para gimnasios y centros de *fitness* que utilizan los servicios de **Fito App**, una *start up* tecnológica costarricense enfocada en ofrecer soluciones *SaaS* para la administración de dichos establecimientos. Este sistema permitirá validar la identidad de los usuarios que ingresen al gimnasio, de forma automática y sin contacto, asegurando que solo aquellos con una membresía activa puedan acceder a las instalaciones. A través de un sistema embebido de propósito específico, el proyecto integra tecnologías de reconocimiento facial y hardware de relativo bajo costo (con relación a otras alternativas del mercado), con el resto de la infraestructura de Fito App, lo que permitirá una gestión centralizada y eficiente de los datos de los usuarios. Este proyecto busca no solo mejorar la seguridad y eficiencia del control de acceso en gimnasios, sino también posicionar a Fito App como una alternativa competitiva e innovadora en el mercado de sistemas de gestión para el sector *fitness*.

1.1 Antecedentes del proyecto

En esta sección se describe la organización en la cual se desarrolla el proyecto, incluyendo una breve reseña histórica de la misma, el tipo de negocio y área a la que se dedica, así como condiciones de su dimensión y estructura organizacional. También se mencionan

las áreas de la Ingeniería en Computadores que tienen aplicabilidad en el proyecto, y se presentan trabajos similares que se han realizado en el área, indicando cómo se relacionan y cómo se diferencian de cada uno.

1.1.1 Descripción de la organización

El proyecto se desarrolla dentro de una organización catalogada como un emprendimiento tecnológico o *start up*, de origen costarricense, llamado **Fito App** [2]. Esta es una plataforma que busca ayudar a los administradores de gimnasios a gestionar su centro, mediante un software como servicio (SaaS) que permite la gestión administrativa de membresías, creación y distribución de rutinas a los clientes y el manejo de reservas de clases.

En procesos de entrevistas realizadas a potenciales clientes, se identificó que hay una deficiencia en las aplicaciones y otros sistemas manuales que actualmente utilizan los gimnasios, lo que genera frustración tanto a los administradores, como a los colaboradores y clientes de los gimnasios. Así, el objetivo de la organización es mejorar la eficiencia operativa de los centros de *fitness* y la experiencia de sus clientes, mediante el uso de tecnología y la automatización de procesos.

Fito App nació en 2023, a manos de Lic. Ignacio Castro Hidalgo, actual director ejecutivo y quien impulsó la participación de la organización en el concurso para optar por los fondos de prototipado (capital semilla) otorgados por la Agencia Universitaria para la Gestión Emprendedora (AUGE) de la Universidad de Costa Rica a finales de ese mismo año. Los fondos de prototipado son recursos concursables y no reembolsables auspiciados por el Sistema de Banca para el Desarrollo (SBD), que brinda recursos financieros a iniciativas desarrolladas dentro del Programa Proyectos de Innovación Tecnológica (PIT) [3].

La participación en el concurso por los fondos fue exitosa, y a partir de febrero del año 2024, se empezó a hacer efectiva la ejecución del capital semilla para apoyar la construcción de un prototipo que permitiera realizar validaciones comerciales del modelo de negocio. La construcción del producto mínimo viable (MVP, por sus siglas en inglés) duró aproximadamente 6 meses, y en agosto del 2024 se hizo el primer lanzamiento del producto en su versión *Beta*, lo que permitió establecer las primeras relaciones comerciales con clientes a manera de usuarios pioneros o *early adopters* (en inglés).

A finales del 2024, se concursó nuevamente por fondos de capital semilla; esta vez como parte del programa de *Puesta en Marcha*, que va dirigido a *Startups* en su proceso de lanzamiento y consolidación en el mercado [4]. Dicho programa fue auspiciado por la Cámara de Tecnologías de Información y Comunicación (CAMTIC), quienes recibieron en ese mismo año su primera designación como Agencia Operadora del SBD, y así poder canalizar parte de los ₡3300 millones de colones que el Sistema de Banca para el Desarrollo destinó a la promoción de la innovación y el emprendimiento en Costa Rica [4].

Los resultados de la aplicación al programa de Puesta en Marcha de CAMTIC fueron positivos, y a partir de enero del 2025, se empezó a ejecutar una estrategia de comer-

cialización y crecimiento del producto, aprovechando los fondos no reembolsables que se obtuvieron. En este contexto, el presente proyecto se desarrolla como parte de la estrategia de crecimiento de la organización, buscando mejorar la propuesta de valor del producto y su competitividad en el mercado.

Se cree que hay una oportunidad de negocio importante no solo a nivel nacional, sino también regional, considerando que la industria del *fitness* ha tenido un crecimiento importante en la última década y no se visualiza que merme en un futuro cercano [5],[6],[7]. Por el contrario, se considera que es un nicho de mercado que ofrece un amplio margen de oportunidad para la innovación y la mejora del estado actual de sus procesos y herramientas. En Fito App, desde el momento de su gestación, se ha trabajado con miras a la expansión hacia el mercado latinoamericano, considerando que su modelo de negocio basado en SaaS permite una escalabilidad considerable.

Actualmente, el producto sigue en desarrollo y en un proceso de mejora continua en el que se trabaja muy estrechamente con los clientes actuales para seguir identificando puntos de dolor en la gestión de los gimnasios y cómo poder atacarlos con el sistema que se está desarrollando. Así, el equipo de trabajo encargado de ejecutar los objetivos de Fito App está conformado por 3 socios fundadores: **Lic. Ignacio Castro Hidalgo**, **Bach. Ángel Villalobos Peña** y **Luis Pedro Morales Rodríguez**, el autor de este documento y encargado de ejecutar el proyecto del Trabajo Final de Graduación. Además, la organización cuenta con el apoyo de **dos estudiantes más** de las carreras de Ing. en Computadores e Ing. en Computación, que trabajan en el proyecto en la calidad de pasantes. La organización no cuenta con una sede física, por tanto la modalidad de trabajo es de tipo remota en la mayoría del tiempo.

1.1.2 Descripción del área de conocimiento del proyecto

El presente proyecto atañe a varias áreas de la Ingeniería en Computadores, entre las que destacan:

Sistemas empotrados

El sistema de control de acceso propuesto se implementa sobre una plataforma de hardware de propósito específico, utilizando una Raspberry Pi 4 con un sistema operativo personalizado, y ciertos periféricos como una cámara, una pantalla táctil y una bocina. Este entorno es representativo de un sistema empotrado, donde se deben considerar restricciones de recursos, rendimiento y estabilidad. La solución requiere diseño y desarrollo de software optimizado para operar en un hardware con capacidades limitadas, lo cual es una competencia central en esta área.

Internet de las cosas (IoT)

El sistema embebido debe conectarse a internet para realizar validaciones de identidad en la nube y registrar accesos en una base de datos remota. Esta característica de conectividad constante y la interacción con servicios distribuidos a través de internet posicionan el sistema dentro del área del Internet de las Cosas, en la que los dispositivos físicos se integran a una red digital para ofrecer funcionalidades inteligentes e interconectadas.

Sistemas operativos

Para optimizar el uso de recursos y enfocar el sistema en su funcionalidad principal, se desarrolla un sistema operativo personalizado utilizando *Yocto Project*, el cual permite configurar una distribución de Linux específicamente adaptada a los requerimientos del sistema embebido y de menor tamaño que un sistema operativo de uso general. Esto implica una comprensión importante de la estructura de los sistemas operativos, sus capas, manejo de servicios, drivers y optimización de recursos.

Reconocimiento de patrones

La validación de identidad se realizará mediante técnicas de reconocimiento facial, que consisten en la extracción y comparación de características biométricas del rostro del usuario. Estas técnicas se basan en algoritmos de reconocimiento de patrones, incluyendo redes neuronales convolucionales (CNN) y otros métodos de aprendizaje profundo, que permiten identificar y clasificar imágenes de rostros con alta precisión. Es importante destacar que la relación del proyecto con esta área de conocimiento será a través del uso de modelos ya existentes y pre-entrenados, y servicios de terceros como *Amazon Rekognition* [8], que permiten realizar la validación de identidad de los usuarios a través de su API. Esto para buscar un enfoque práctico y aplicado en el uso de tecnologías avanzadas de reconocimiento facial, sin necesidad de desarrollar algoritmos desde cero.

1.1.3 Trabajos similares

Debido a que la naturaleza de la organización en la que se desarrolla el proyecto es la de un emprendimiento tecnológico en sus primeras etapas, se busca que el producto resultante represente un aporte significativo a su proceso de generar tracción comercial y presencia en el mercado local de sistemas de gestión para gimnasios. En ese sentido, el presente apartado se enfoca en describir a otras soluciones que han sido desarrolladas en el área de control de acceso para centros de *fitness*, y que podrían considerarse como competidores directos o indirectos de Fito App. Estas alternativas se han identificado a partir de un análisis de mercado realizado por el equipo de trabajo de la organización.

Como resultado de dicho análisis de mercado, se identificó que las alternativas de sistemas

de control de acceso que existen actualmente en el mercado local, pueden clasificarse en dos categorías: las que funcionan de forma independiente y las que se integran con el sistema de gestión que ya usa el gimnasio.

Los sistemas independientes, al no poder integrarse con el resto de la gestión administrativa, tienen la gran desventaja de que no pueden sincronizar la información de los usuarios, lo que implica que no contemplen el estado de la membresía de cada cliente al validar su acceso. Esto puede llevar a que personas que no tengan una membresía activa, ya sea porque se atrasaron con el pago o simplemente dejaron de pagar, puedan seguir ingresando a las instalaciones si en algún momento fueron registradas en el sistema. Para evitar esto, se requiere que el gimnasio esté constantemente actualizando la información de los usuarios activos en el sistema de control de acceso, lo que representa un esfuerzo adicional y puede llevar a errores humanos.

En esta categoría se encuentran opciones como las que ofrece *Daosafe Technology Co., Ltd.* [9], una empresa fundada en China, que fabrica sistemas de seguridad para la entrada y salida de personas de edificios y espacios de acceso restringido. En particular, el modelo *DS112 Tripod Turnstile* es un torniquete de acceso peatonal de alta durabilidad, diseñado para controlar de manera segura y eficiente, el ingreso de personas mediante un torniquete de brazos giratorios. Tiene capacidad de integración a múltiples dispositivos de control, tales como lectores de tarjeta, de huella, de código QR o de reconocimiento facial [10]. En la *Figura 1* se muestra este modelo de torniquete [1].

Fig. 1: Imagen del modelo DS112 Tripod Turnstile de Daosafe.

Se identificó que algunos potenciales clientes de Fito App han adquirido este sistema (o similares), comprándolo directamente con el fabricante. El costo de adquisición reportado, ronda entre los $2000 y $2500 dólares, que incluye el torniquete y el sistema de control, para el cual se suele preferir el reconocimiento facial, aunque no es la única opción disponible. Estos gimnasios utilizan otros sistemas para la gestión administrativa,

[1] Imagen tomada del manual de referencia del modelo DS112 [10].

que no son compatibles con el sistema de control de acceso. Esta falta de integración entre los sistemas ha llevado a que los administradores de estos gimnasios hayan consultado a los proveedores de sus sistemas de gestión sobre la posibilidad de integrar el sistema de control de acceso, pero la respuesta ha sido negativa.

Por otro lado, los sistemas que sí se integran con el resto de la gestión del gimnasio, permiten validar no solo la identidad del usuario, sino también el estado de su membresía. Esto hace que no solo sean más seguros, sino que también permiten generar estadísticas de acceso que sean accesibles desde el mismo sistema de gestión del gimnasio. Estas características hacen que los sistemas de esta categoría sean más atractivos, y por ende, los más comunes entre los gimnasios que fueron consultados.

En esta categoría se encuentran opciones como las que ofrece la empresa *Latinsoft*, con su sistema llamado *Fitness 24/7 Gym* [11]. Esta es una de las soluciones que tiene mayor adopción en los gimnasios medianos y grandes (ver la *Tabla 4* para entender la referencia de tamaño de un gimnasio) en el país. Para entender mejor el contexto de esta solución, se realizó una cotización directamente con un agente de ventas de dicha empresa, lo que permitió obtener información sobre sus características, precios y funcionalidades específicas.

El producto *Fitness 24/7 Gym* se ofrece como un sistema de gestión de gimnasios general que se puede obtener pagando una membresía, cuyo costo dependerá del paquete seleccionado, pero que puede ir desde los $75 mensuales para PYMEs, hasta alrededor de $500 mensuales para grandes empresas y paquetes personalizados. Las funcionalidades que incluye cada paquete son variables, y no se ahondará en ellas en este documento; sin embargo, si se quiere incluir la funcionalidad de control de acceso, el gimnasio deberá asumir el costo de la compra del hardware y el sistema mecánico de acceso que deseen. En la *Tabla 1* se muestra un resumen de los precios aproximados de los diferentes dispositivos que *Latinsoft* ofrece para el control de acceso. Posteriormente, en la *Tabla 2* aparecen los precios de los diferentes sistemas mecánicos que se pueden integrar con los dispositivos anteriormente mencionados.

Dispositivo	**Precio aproximado**
Control por pin numérico	$15
Lector de código QR	$90 - $100
Lector de huella dactilar	$120
Dispositivo de reconocimiento facial	$1000 - $1200

Tabla 1: Precios aproximados de los dispositivos de control de acceso ofrecidos por el proveedor de *Latinsoft*.

El tipo de dispositivo que es más común entre los gimnasios que utilizan este servicio es el lector de huella dactilar. Se presume que esto es por ser el sistema que utiliza datos biométricos, lo que le da un mayor nivel de seguridad al sistema, pero que también es económicamente accesible. La otra opción con esta característica es el dispositivo de

reconocimiento facial, pero la diferencia de precios puede llegar a ser hasta 10 veces mayor, lo que hace que no sea una opción viable para la mayoría de los gimnasios.

Ahora bien, a partir de entrevistas realizadas a administradores de gimnasios que utilizan el sistema de *Latinsoft*, se ha identificado que el lector de huella dactilar tiene un margen de error importante, lo que afecta la experiencia de los usuarios. No se ha cuantificado el porcentaje de error de los sensores de huella dactilar utilizados, pero sí se ha evidenciado un descontento general en los usuarios de estos sistemas. Desde el punto de vista de Fito App, esto representa una oportunidad de negocio si se logra implementar una solución de reconocimiento facial que sea capaz de operar de forma rápida, confiable y sin contacto físico; pero que también sea más económica que las alternativas actuales que ofrece *Latinsoft*.

Sistema mecánico	**Precio aproximado**
Puerta automática accionada por magnetismo	$400 - $500
Torniquete de brazos giratorios	$3800 - $5000

Tabla 2: Precios aproximados de los sistemas mecánicos de control de acceso ofrecidos por el proveedor de *Latinsoft*.

En cuanto a los sistemas mecánicos disponibles, se sabe que estos pueden ser utilizados con cualquiera de los dispositivos de control de acceso, por lo que su escogencia pasa por considerar factores como el costo, seguridad y facilidad de instalación. Es claro que la diferencia de precio entre la opción de puerta automática y el torniquete es considerable, siendo la segunda opción hasta 10 veces más cara. Esto hace que la combinación más popular entre los clientes de *Latinsoft* sea la de lector de huella dactilar con puerta automática, ya que es la opción más económica y que ofrece un nivel de seguridad aceptable.

Otra alternativa que se ha encontrado en el mercado local y que ha ganado popularidad en los últimos 2 años, es la de *MNT* [12] . Esta es una empresa costarricense que ofrece sistemas de control de acceso para cualquier tipo de establecimiento. Recientemente, se han enfocado en el sector de gimnasios al integrar su producto con un sistema de gestión de membresías básico. Su línea de productos se enfoca principalmente en torniquetes automáticos controlados por un sistema de reconocimiento facial. En la Figura 2 se muestra una imagen de este sistema.[2]

Se utilizó la misma estrategia que con el caso anterior y se realizó una cotización directamente con un agente de ventas de *MNT*. A diferencia de *Fitness 24/7 Gym*, el sistema de *MNT* no tiene un costo inicial de adquisición, sino que se ofrece a manera de alquiler con un costo mensual. En la *Tabla 3* se resumen los precios de los paquetes ofrecidos en función de la cantidad de clientes que se manejen.

La información recopilada de la cotización con *MNT* permitió concluir algunos puntos importantes. Primero, el modelo de negocio basado en un alquiler del sistema, en lugar

[2]Imagen promocional tomada del sitio web de MNT [12].

Cantidad de clientes	Costo mensual
Hasta 300	$113
Hasta 1500	$296
Hasta 1500 con dos líneas de acceso	$496
Hasta 1500 solo con dispositivo de reconocimiento facial (sin torniquete)	$78

Tabla 3: Precios de los paquetes de sistemas de control de acceso ofrecidos por *MNT*.

de una sola inversión inicial elevada, puede ser de un mayor atractivo para los negocios que buscan una transición hacia estos sistemas que sea conveniente desde el punto de vista económico. En segundo lugar, se plantea la opción de solo adquirir el sistema de reconocimiento facial, sin el torniquete, lo que da lugar a cuestionarse si realmente es necesario contar con un sistema mecánico de control de acceso, o si el reconocimiento facial por sí solo es suficiente para satisfacer las necesidades de control de acceso.

En Fito App, se plantea la hipótesis comercial de que si se combina el sistema de reconocimiento facial con una barrera de tipo psicológica, como un altavoz que emita un sonido al momento de validar el acceso, se puede lograr un efecto similar al de una barrera física, pero con un costo mucho menor. Esto representaría una ventaja competitiva importante frente a otras soluciones del mercado. La intención es que el alcance de este proyecto permita validar esta hipótesis en etapas posteriores, y que se pueda evaluar la posibilidad de ofrecerlo como una opción adicional a los gimnasios que utilicen el sistema de Fito App.

Fig. 2: Imagen del sistema de control de acceso que ofrece MNT.

1.2 Planteamiento del problema

En esta sección se describe el contexto del problema identificado y la justificación que respalda la necesidad de implementar, desde Fito App, una solución que lo aborde.

1.2.1 Contexto del problema

Como se mencionó anteriormente, Fito App desarrolla y mantiene un SaaS que le permite a la administración de los gimnasios, gestionar: sus ventas a través de membresías, las rutinas personalizadas de sus clientes y la reserva de cupos a eventos tales como clases o citas de medición. Esto mediante una aplicación web para uso de la administración del gimnasio, y una aplicación móvil para uso de los clientes. Este fue el alcance definido para la fase de prototipado del producto y representa el estado actual de los servicios que Fito App ofrece a sus clientes.

La validación comercial del prototipo, ha arrojado resultados positivos en cuanto a la aceptación del producto; sin embargo, es sabido que el alcance del prototipo actual no es suficiente para poder competir contra otras soluciones ya establecidas en el mercado local, como lo son *WStudio* [13] y *Fitness 24/7 Gym* [11]. En etapas previas a la implementación del prototipo, se condujeron investigaciones de campo en las que se entrevistó a administradores de gimnasios o centros de *fitness* de diversos tipos, para poder entender mejor sus necesidades y se capturaron requerimientos importantes que principalmente son aplicables a gimnasios de dimensión mediana y grande. Para clasificar a los gimnasios por su tamaño, se utiliza la cantidad de clientes que tienen asociados. Las categorías que se identificaron de las investigaciones internas realizadas son las siguientes:

Tamaño	**Cantidad de clientes**
Micro	$0 < N \leq 100$
Pequeño	$100 < N \leq 300$
Mediano	$300 < N \leq 1000$
Grande	$N > 1000$
Multi-sede	$N > 1000$

Tabla 4: Categorización de los centros de *fitness* según su tamaño

El modelo de negocio por defecto de los gimnasios es utilizar la figura de membresía, la que permite que sus clientes, por un lapso determinado, tengan acceso a sus instalaciones y servicios. Así, es imprescindible que los gimnasios, principalmente los de mayor tamaño, cuenten con un **sistema de control de acceso** a sus instalaciones para poder filtrar y autorizar el ingreso únicamente de las personas que cuenten con una membresía activa.

1.2.2 Justificación del problema

Tener un control activo de quiénes ingresan y salen de las instalaciones de un gimnasio es fundamental para la administración del mismo. Esto le permite no solo poder disminuir el riesgo de ingresos de personas que no hayan pagado su derecho de uso de las instalaciones (lo que representa pérdidas económicas), sino también generar estadísticas de interés con relación al flujo de clientes; como por ejemplo, la cantidad de personas que ingresan en un día, el promedio de ingresos que hay por semana, o las horas del día que tienen mayor concurrencia.

Las estadísticas relacionadas con el flujo de ingreso de personas son valiosas para la administración porque les permite entender mejor el comportamiento de sus clientes, y a partir de ellas, tomar decisiones de negocio más informadas y basadas en datos. Ahora bien, esta información no solo es valiosa para el uso interno de la organización. Se pudo identificar del proceso de entrevistas realizadas a administradores, que también existe la necesidad de llevar el control del flujo de clientes que ingresan y salen de un local porque hay ocasiones en las que funcionarios del Ministerio de Salud de Costa Rica ejecutan auditorías o inspecciones en los gimnasios y solicitan estos datos. En caso de no brindar esta información con precisión, es posible que esto tenga repercusiones negativas de índole legal.

Como es esperable, cuando un gimnasio alcanza un tamaño mediano o grande, el control de acceso deja de ser una tarea viable para un proceso manual, por lo que surge la necesidad de implementar sistemas automatizados. En el mercado local, existen varias soluciones que ofrecen sistemas de control de acceso automatizados. Las principales variantes de estos sistemas que se han encontrado son las siguientes:

- **Acceso con pin**: el cliente tiene asociado un pin único el cual deberá utilizar cada vez que ingrese al local, por ejemplo mediante un teclado numérico que el gimnasio pone a su disposición. Luego, el sistema evalúa si corresponde a un cliente con una membresía activa. Es una solución de bajo costo, pero tiene la desventaja de que su seguridad puede ser sorteada fácilmente si un usuario comparte su pin a otra persona sin autorización de ingreso. También puede ser un inconveniente para usuarios olvidadizos que les cueste trabajo memorizar el código.

- **Acceso con código QR**: el funcionamiento es similar al acceso por pin, ya que el usuario debe mostrar el código QR cada vez que ingresa al local. Es una solución que se popularizó durante la pandemia ya que evita el contacto físico directo con algún medio compartido. La lectura se puede hacer con un dispositivo específico o con un dispositivo móvil, de forma eficaz y rápida, con muy poco margen de error. Tiene la misma desventaja que el punto anterior de la posibilidad de compartir el código de forma deshonesta.

- **Lector de huella dactilar**: es una de las soluciones más populares actualmente. Tiene la gran ventaja de utilizar datos bio-métricos para el acceso, por lo que se

eliminan los problemas de seguridad de las dos soluciones anteriores. Aunque tiene la desventaja de que hay un cierto margen de error en el sistema, que tiende a arrojar falsos negativos, y que ha afectado la experiencia de los usuarios a los que se les ha consultado. No se ha cuantificado el porcentaje de error de los sensores de huella dactilares utilizados, pero sí se ha evidenciado un descontento general en los usuarios de estos sistemas. Otro punto negativo es el contacto que podría generar un foco de contagio sino se tienen medidas adecuadas de higiene al tocar el dispositivo.

- **Reconocimiento facial**: es una solución innovadora que está ganando popularidad en los sistemas de control de acceso en general. Al igual que el lector de huella dactilar, utiliza datos bio-métricos, lo que favorece la seguridad del sistema, aunque esta solución utiliza tecnología de reconocimiento de patrones que garantizan un mayor porcentaje de acierto que los sensores de huella, que pueden perder precisión si se ensucian o se ven afectados por la humedad característica de lugares como los gimnasios. También, al evitar el contacto físico, el reconocimiento facial se coloca como una alternativa más segura para la salud de las personas evitando focos de contagio de enfermedades. Ahora bien, se ha identificado que soluciones de este tipo que ofrecen otros SaaS similares al de Fito App, tienen problemas de rendimiento y presentan latencias elevadas a la hora de procesar las solicitudes de acceso, afectando de manera importante la experiencia de usuario. También, esta se suele presentar como la alternativa de mayor costo económico, lo que puede llevar a que centro de *fitness* prefieran otro tipo de soluciones más accesibles. En Fito App, esto se interpreta como una oportunidad de negocio si se logra implementar una solución más eficiente y de menor costo que las alternativas actuales en el país.

1.2.3 Enunciado del problema

En síntesis, el proyecto busca atacar el problema de los ingresos no autorizados a los gimnasios, que afecta negativamente la seguridad y la experiencia de los usuarios que sí tienen membresías activas, así como los réditos económicos de la administración. Esto mediante un sistema automatizado que controle y registre el acceso de los usuarios a los diferentes centros de *fitness*, los cuales son potenciales clientes de Fito App, lo que permitiría robustecer el producto actual y colocarlo como una alternativa más competente en el mercado local.

Dicho esto, se considera que una solución de reconocimiento facial que sea capaz de operar de forma rápida, confiable y sin contacto físico, mejorando la seguridad y comodidad para los usuarios, le daría una ventaja competitiva al producto. El desarrollo del producto descrito en este documento busca desarrollar un sistema de bajo costo, que minimice la latencia en el procesamiento de reconocimiento facial y se integre eficientemente con el sistema de gestión de membresías de Fito App, con un enfoque en la mantenibilidad y escalabilidad, aumentando así su competitividad en el mercado y su valor para los gimnasios que tengan dicha necesidad.

1.3 Objetivos del proyecto

En este apartado se definen los objetivos del proyecto, tanto el objetivo general como los específicos.

1.3.1 Objetivo general

Desarrollar un sistema de control de acceso para gimnasios, que mejore la seguridad y eficiencia en la validación de membresías en el sistema de gestión de Fito App, utilizando una plataforma de *hardware* embebido.

1.3.2 Objetivos específicos

Los objetivos específicos se asocian con un identificador único para poder ser referenciados en secciones posteriores del documento con mayor facilidad.

- **OE1**: Diseñar la arquitectura de un sistema embebido de control de acceso, que permita la captura y análisis de datos biométricos, mediante el uso de una plataforma de bajo costo con un sistema operativo de propósito específico.
- **OE2**: Desarrollar un módulo de reconocimiento facial, para la validación de la identidad de los usuarios en el punto de ingreso al gimnasio, mediante la integración de una cámara y el uso de servicios en la nube.
- **OE3**: Integrar un módulo de reportes en el sistema de administración de Fito App, que permita la consulta de estadísticas de acceso de los usuarios, mediante una interfaz web.
- **OE4**: Evaluar el rendimiento del sistema en un entorno real, para la verificación del cumplimiento de los requisitos de latencia, precisión y experiencia de usuario, mediante pruebas de campo controladas.

1.4 Alcances, entregables y limitaciones del proyecto.

En esta sección se describen los alcances, entregables y limitaciones del proyecto, con el objetivo de delimitar claramente lo que se espera lograr, los productos que se generarán como resultado del desarrollo y las restricciones que podrían influir en la ejecución del mismo.

1.4.1 Alcances

El proyecto contempla el diseño, implementación e integración de un sistema de control de acceso para gimnasios, utilizando reconocimiento facial como método de validación de identidad. El sistema será desarrollado sobre una plataforma embebida de propósito específico, conectada a servicios de reconocimiento facial en la nube, y comunicada con el sistema administrativo actual de Fito App.

El alcance del proyecto incluye:

- El diseño e implementación de la arquitectura de hardware y software del sistema embebido, incluyendo su sistema operativo personalizado.
- El desarrollo del módulo de captura y reconocimiento facial para validación de acceso.
- La implementación de un mecanismo de retroalimentación auditiva y visual para el usuario al momento de validar su acceso.
- La integración con el *backend* de Fito App para registrar accesos y generar estadísticas.
- La integración de un módulo de reportes en la aplicación web de Fito App, que permita a los administradores consultar los registros de acceso.
- La ejecución de pruebas de validación funcional y de rendimiento en un entorno real de operación.

El proyecto no contempla el desarrollo de modelos de reconocimiento facial propios, ya que se utilizarán servicios preexistentes. Tampoco se considera como parte del alcance la integración con mecanismos físicos de cierre o apertura de acceso, tales como puertas o torniquetes; el enfoque se centra en la validación y registro digital del acceso y en una retroalimentación adecuada al usuario.

1.4.2 Entregables

En la *Tabla 5*, se resumen los entregables del proyecto en función de los objetivos específicos planteados.

Tabla 5: Entregables del proyecto en función de los objetivos específicos.

Obj. Esp.	Entregable	Descripción
OE1	Documento de arquitectura	Especificación de la arquitectura hardware-software del sistema embebido, incluyendo diagramas y justificación técnica.
OE2	Unidad de interacción y detección facial	Conjunto hardware-software residente en el nodo embebido que captura las imágenes faciales, coordina la comunicación con la nube para su procesamiento y ofrece retroalimentación inmediata al usuario mediante indicadores audiovisuales.
OE2	Imagen de sistema operativo para nodo embebido	Distribución Linux minimalista personalizada que incluye todos los componentes de software requeridos por el nodo en borde, facilitando la replicación e instalación en campo.
OE2	Servicio de reconocimiento facial y autorización	Microservicio desplegado en infraestructura *cloud* que procesa las imágenes enviadas desde el módulo local, realiza la validación de identidad y estado de membresía del usuario, y devuelve el resultado al nodo en borde.
OE3	Módulo de reportes	Interfaz de visualización web integrada al sistema de Fito App para mostrar estadísticas de acceso a los administradores.
OE4	Informe de pruebas de rendimiento	Documento con los resultados de pruebas realizadas en entorno real, incluyendo métricas de latencia y precisión.

1.4.3 Limitaciones del proyecto

El desarrollo del proyecto se verá influenciado por diversas limitaciones que podrían afectar su ejecución y resultados. A continuación se enumeran las principales limitaciones identificadas:

1. La ejecución del proyecto deberá hacerse en un lapso de 16 semanas, es decir, un semestre lectivo del ITCR, planteado para iniciarse el 10 de febrero del 2025.

2. La organización cuenta con un dispositivo *Raspberry Pi 4 Model B*, del cual se dispone para el desarrollo del proyecto de ser necesario. El resto de componentes del sistema deberán ser adquiridos externamente. En la etapa de diseño deberá de tomarse la decisión sobre la plataforma de *hardware* y componentes que se utilizará.

3. Es posible que adquisición del *hardware* necesario para el proyecto implique hacer

una compra internacional, lo que supone una limitación en cuanto al tiempo de latencia asociado al proceso de envío y entrega de los productos.

4. La organización cuenta con un presupuesto limitado para la ejecución del proyecto, lo que podría restringir la adquisición de ciertos componentes o servicios necesarios para el desarrollo del sistema.

5. Los entregables generados por el proyecto, como código fuente y documentación, serán de carácter confidencial y propiedad intelectual de Fito App.

6. Las pruebas de funcionalidad completa podrían estar limitadas por la disponibilidad de un gimnasio dispuesto a colaborar, lo cual podría restringir el número y la duración de las pruebas en un entorno real.

Capítulo 2

Marco de referencia teórico

El éxito en el diseño y la implementación de un sistema de control de acceso basado en reconocimiento facial depende, en gran medida, de la solidez conceptual que sustente cada decisión de ingeniería. Por ello, en este capítulo se presentan los conceptos fundamentales que guiarán el desarrollo del sistema. Se articulan las definiciones, los modelos y buenas prácticas necesarias para justificar la arquitectura propuesta, orientar la selección de tecnologías y sustentar los criterios de validación.

En consecuencia con la naturaleza multidisciplinaria del sistema, que combina *hardware* especializado, algoritmos de inteligencia artificial y servicios distribuidos, el capítulo integra conceptos clave de diversas áreas, incluyendo:

1. Sistemas embebidos
2. Sistemas operativos embebidos y *Yocto Project*
3. Computación en la nube y su papel en arquitecturas IoT
4. Herramientas de reconocimiento y detección facial

La Figura 3 ilustra los ejes temáticos que se desarrollarán en este capítulo, proporcionando una visión general de cómo se interrelacionan estos conceptos para formar la base teórica del sistema propuesto. La profundización en cada uno de estos ejes busca no solo contextualizar el trabajo, sino también establecer un marco de referencia que permita contrastar la implementación del proyecto con el estado del arte.

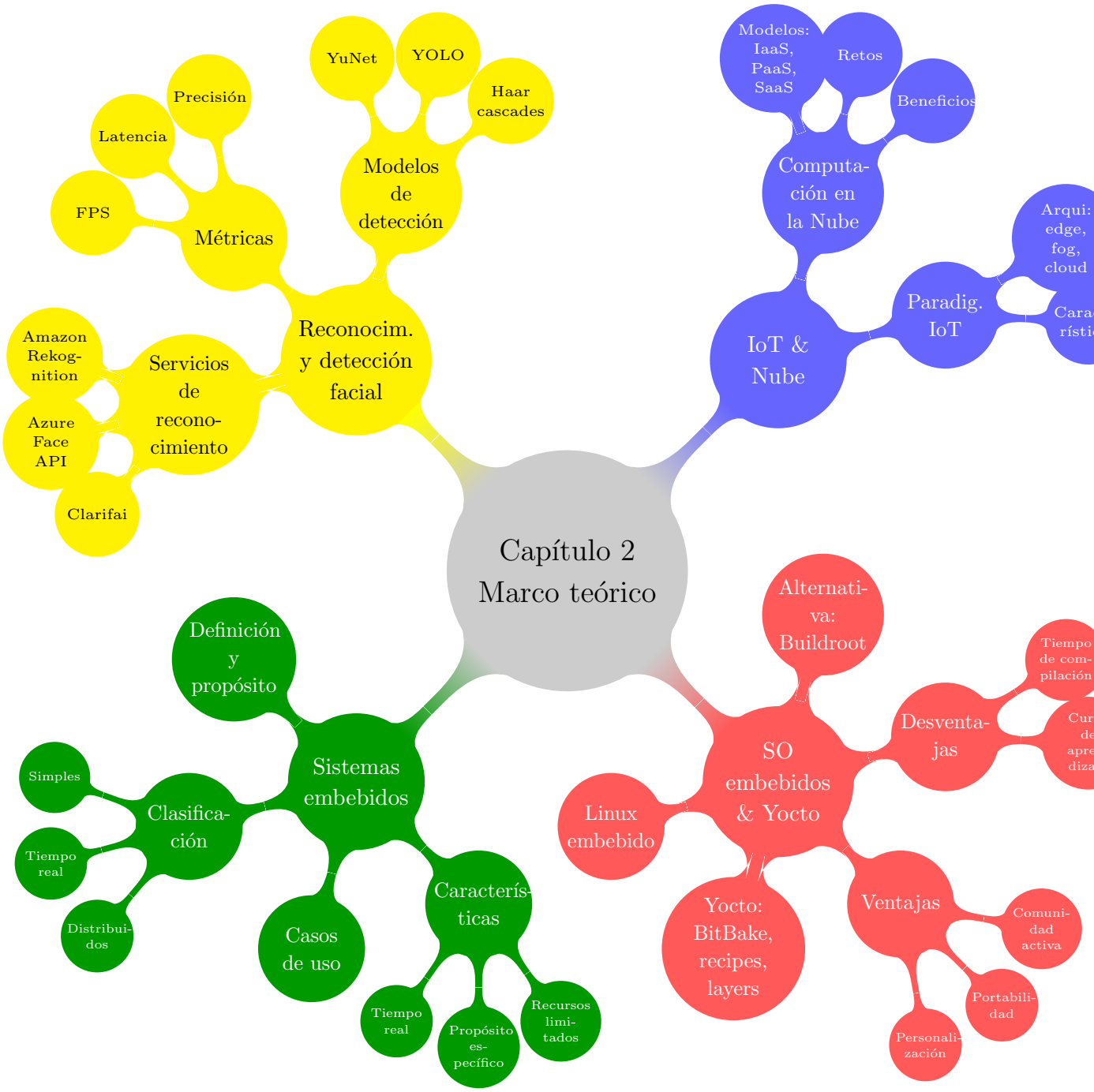

Fig. 3: Mapa conceptual de los ejes temáticos del sistema propuesto.

2.1 Sistemas embebidos

Un sistema embebido, como Wolf define en [14], "*es un dispositivo que incorpora una computadora programable, pero que no está destinado a funcionar como una computadora de propósito general*". Es un conjunto de componentes de hardware y software, y quizás

componentes mecánicos, dedicados a realizar una función específica, o un conjunto acotado de funciones, dentro de un sistema mayor, y que generalmente opera con recursos limitados y bajo requisitos estrictos de confiabilidad y tiempo real [15].

Los sistemas embebidos son ubicuos en la vida cotidiana, y se encuentran en una amplia variedad de aplicaciones. Su diseño y aplicación son una necesidad fundamental para múltiples áreas de la ingeniería, pues dispositivos como automóviles, teléfonos móviles y electrodomésticos dependen en gran medida de microprocesadores embebidos. Para integrar estos componentes en un sistema, los departamentos de ingeniería deben ser capaces de identificar tareas computacionales específicas dentro del producto, diseñar una plataforma de hardware con capacidad de entrada/salida que permita ejecutar dichas tareas, e implementar el software que las coordine de forma eficiente [14].

El diseño de sistemas embebidos enfrenta desafíos significativos que van más allá de la mera funcionalidad. Según Jebril y Abu Al-Haija, los sistemas embebidos deben operar bajo restricciones físicas y temporales estrictas, lo que complica su diseño y verificación [16]. Entre los retos más relevantes destacan: la selección adecuada del hardware para cumplir simultáneamente con los requisitos de desempeño y las limitaciones presupuestarias; el cumplimiento estricto de plazos de respuesta en aplicaciones sensibles al tiempo; la minimización del consumo energético, especialmente en dispositivos alimentados por batería; y la necesidad de asegurar confiabilidad en entornos de operación continua o críticos [14].

Adicionalmente, se espera que muchos de estos sistemas sean escalables y actualizables, lo cual añade complejidad al diseño inicial. Estos retos, que no suelen presentarse en la misma magnitud en sistemas de propósito general, requieren un enfoque metodológico riguroso y una comprensión profunda de la interacción entre hardware, software y entorno de operación.

2.1.1 Características principales

Los rasgos distintivos de los sistemas embebidos comúnmente citados en la literatura consultada, se resumen en la siguiente tabla:

Tabla 6: Principales características de los sistemas embebidos

Característica	Descripción
Restricción de recursos	CPU, memoria y energía limitadas obligan a optimizar código y hardware [17].
Propósito específico	El software se diseña para una tarea concreta (p. ej., validar credenciales biométricas) [14].
Operación en tiempo real	Se exige cumplir plazos máximos de respuesta (determinismo) [18].

Continúa en la siguiente página

Tabla 6: Principales características de los sistemas embebidos (continuación)

Característica	Descripción
Alta confiabilidad	El dispositivo suele operar de forma autónoma y continua, por lo que los fallos son inaceptables [19].
Integración hardware-software	El diseño busca optimizar electrónica, sistema operativo y aplicación de manera conjunta [14].
Interfaz con el entorno	Tienen la capacidad de interactuar con el entorno a través de unidades de entrada/salida e interfaces de comunicación. [14].

2.1.2 Clasificación de los sistemas embebidos según su nivel de complejidad

Los sistemas embebidos pueden clasificarse según diversos criterios, siendo uno de los más relevantes el nivel de complejidad, el cual abarca tanto aspectos del hardware como del software, así como la interacción con el entorno físico. Respecto a esto, Lee y Seshia definen a los sistemas embebidos como "*computadoras menos visibles que interactúan con procesos físicos, típicamente en la forma de sistemas ciberfísicos con lazos de retroalimentación entre los procesos físicos y el software*" [20]. Esta interacción impone restricciones y retos específicos de diseño que varían según el tipo de sistema.

Según Wolf [14], los sistemas embebidos pueden clasificarse en cuatro niveles de complejidad creciente:

1. **Sistemas embebidos autónomos simples**, tales como relojes digitales o termostatos, los cuales ejecutan una lógica de control elemental y suelen operar sin conectividad o interacción compleja.

2. **Sistemas de tiempo real**, los cuales requieren respuestas dentro de márgenes temporales estrictos. Estos sistemas son típicos en automóviles, dispositivos médicos y control industrial.

3. **Sistemas embebidos con múltiples procesadores o coprocesadores**, que integran arquitecturas heterogéneas optimizadas para tareas específicas, como decodificadores multimedia o estaciones base de comunicación.

4. **Sistemas embebidos distribuidos**, que incluyen componentes interconectados a través de redes, con alta complejidad de coordinación y sincronización, como los sistemas automotrices modernos y las infraestructuras inteligentes.

Esta categorización puede también relacionarse con el grado de criticidad del sistema. Tal como señalan Lee y Seshia, "*los sistemas ciberfísicos enfrentan desafíos únicos, como la necesidad de garantizar propiedades temporales en un entorno concurrente, donde la*

precisión depende del comportamiento tanto computacional como físico" [20]. Por tanto, mientras los sistemas simples pueden tolerar ciertos errores o retrasos, los sistemas complejos, como los sistemas embebidos distribuidos en la industria automotriz o cirugía robótica, requieren garantías formales sobre su comportamiento.

En resumen, el nivel de complejidad de un sistema embebido se encuentra determinado por factores como la criticidad temporal, la concurrencia, la heterogeneidad de hardware y la capacidad de integración en redes de sensores y actuadores. Esta clasificación no solo guía las decisiones de diseño, sino que también establece el enfoque metodológico a seguir en su desarrollo.

2.1.3 Aplicación de los sistemas embebidos en control de acceso

Los sistemas embebidos juegan un papel fundamental en los sistemas modernos de control de acceso, particularmente en aplicaciones donde se requiere una validación rápida, confiable y autónoma de la identidad del usuario. El uso de propósito específico, el bajo consumo energético y la capacidad de operar en tiempo real hacen que los sistemas embebidos sean ideales para este tipo de aplicaciones, donde se debe responder a eventos del entorno físico en un tiempo limitado.

En el estudio de Guerbaoui et al., se presenta una plataforma funcional basada en una Raspberry Pi 3 y una cámara Intel RealSense D415, utilizando clasificadores en cascada Haar para la detección y reconocimiento facial. Este sistema fue desarrollado específicamente para operar en tiempo real y en condiciones variables de iluminación y ángulos de visión, mostrando un desempeño adecuado para aplicaciones de vigilancia en instalaciones sensibles [21].

De manera complementaria, Kalkar et al. proponen una arquitectura distribuida en la que un sistema embebido realiza la detección de rostros a través de cámaras IP, y delega el reconocimiento facial a un servidor computacional. Se emplean modelos de detección como *YOLO* y redes neuronales ligeras (*MobileFaceNets*), lo que permite mantener un equilibrio entre velocidad y precisión. El diseño está orientado a aplicaciones en zonas restringidas como oficinas o centros educativos, donde se requiere monitoreo continuo con capacidad de respuesta en tiempo real [22].

Finalmente, Hammami y Alhammami desarrollan un sistema de control de acceso robusto que combina reconocimiento facial con códigos QR cifrados, ejecutado completamente sobre una Raspberry Pi 4. Esta solución, además de mejorar la precisión del reconocimiento, permite validar la identidad de visitantes mediante códigos de un solo uso y registra todos los eventos de acceso en una base de datos local, logrando un rendimiento de 8.27 FPS (cuadros por segundo) en condiciones reales de uso [23].

Estos estudios evidencian que los sistemas embebidos son una opción viable y efectiva para implementar soluciones de control de acceso basadas en reconocimiento facial. La combinación de hardware especializado, algoritmos optimizados y la capacidad de operar

en tiempo real permiten desarrollar sistemas que cumplen con los requisitos de seguridad y eficiencia necesarios en entornos críticos.

2.2 Sistemas operativos embebidos y *Yocto Project*

Un sistema operativo embebido es un software especializado que proporciona servicios básicos como gestión de hardware, planificación de tareas y comunicación entre procesos, adaptados a las restricciones y necesidades particulares de un sistema embebido. Según Holt y Huang, estos sistemas están diseñados para ofrecer un conjunto mínimo pero suficiente de funciones orientadas a una aplicación específica, con alta eficiencia, bajo consumo de recursos y fiabilidad en tiempo real, permitiendo controlar periféricos y tareas críticas de forma predecible [24].

Wang complementa esta definición al señalar que un sistema operativo embebido no es un sistema monolítico, sino un conjunto modular de funciones esenciales que permiten la ejecución coordinada y determinista de múltiples tareas en ambientes con limitaciones físicas y temporales [25]. Estas características lo diferencian de los sistemas operativos de propósito general, cuya arquitectura y funcionalidad están pensadas para entornos flexibles y de uso amplio, como computadoras personales o servidores.

Con el aumento del poder de cómputo en las plataformas embebidas, Linux se ha convertido en una opción popular para este tipo de aplicaciones, gracias a su flexibilidad, código abierto y ecosistema maduro. Sin embargo, su uso requiere de adaptaciones importantes. Existen variantes como *µLinux*, pensada para sistemas sin unidad de manejo de memoria (*MMU*), y *RTLinux*, que introduce una capa de emulación para manejar tareas críticas en tiempo real sobre el kernel de Linux estándar [24] [25].

Estas adaptaciones, aunque poderosas, implican un nivel de complejidad elevado en la construcción de un sistema embebido, pues es necesario personalizar diversos aspectos críticos del sistema operativo, como el gestor de arranque, la configuración del kernel, la gestión de memoria y la integración de controladores específicos para el hardware utilizado. En este contexto, herramientas como *Yocto Project* cobran una especial relevancia.

2.2.1 Yocto Project: conceptos básicos y herramientas principales

El *Yocto Project* es un marco de trabajo de código abierto que permite crear distribuciones GNU/Linux adaptadas a sistemas embebidos mediante la generación reproducible de imágenes y paquetes binarios [26]. Su piedra angular es el *OpenEmbedded Build System*, cuyo motor de ejecución es *BitBake*. A grandes rasgos, los elementos fundamentales que es necesario comprender son los siguientes:

- **BitBake**: herramienta de orquestación que interpreta *meta-datos* y ejecuta un gra-

fo de tareas (*parsing*, configuración, compilación, empaquetado, etc.) respetando dependencias y permitiendo compilación en paralelo [27]. Conceptualmente, es similar a la herramienta `make` en `C`, pero está especializada en la construcción de stacks embebidos y en la gestión de cruces de arquitectura (*cross-compilation*).

- **Poky**: distribución de referencia que incluye BitBake; un conjunto mínimo de capas (`meta`, `meta-poky`, `meta-yocto-bsp`, entre otras) y herramientas auxiliares. Poky sirve de punto de partida para crear distribuciones personalizadas evitando partir de cero [26]. Esta herramienta es especialmente útil para desarrolladores que buscan una solución rápida y funcional, ya que proporciona una base sólida sobre la cual construir.

- **Capas (*layers*)**: estructura modular donde se agrupan colecciones coherentes de meta-datos. Cada capa encapsula una política (p. ej. *Board Support Package*[1], distro, aplicación) y puede activarse o desactivarse de forma independiente, lo que favorece la escalabilidad y el mantenimiento. El modelo de capas fomenta la separación entre hardware (BSP), políticas de distribución y aplicaciones [26].

- **Meta-datos y *recipes***: los ficheros `.bb` (*recipes*) describen cómo obtener el código fuente, aplicar parches, compilar y empaquetar un componente. Los ficheros `.bbclass` (clases) encapsulan lógica común reutilizable, mientras que los `.conf` establecen configuración global (máquina, distro, usuario). Todo este conjunto constituye los *meta-datos* que BitBake analiza para generar el grafo de tareas [27].

- **Variables y sobrecargas**: la flexibilidad del sistema radica en la extensiva parametrización mediante variables (p. ej. `SRC_URI`, `DEPENDS`) y en los mecanismos de sobrecarga (`override`) que permiten adaptar el comportamiento según máquina, arquitectura o política de distribución [27].

- **SDK y eSDK**: el flujo de trabajo típico culmina en una imagen para el dispositivo y en un SDK extensible que facilita el desarrollo de aplicaciones para el ambiente objetivo, manteniendo coherencia con la cadena de compilación empleada durante el proceso de construcción [26].

El uso de *Yocto Project* permite a los desarrolladores crear distribuciones personalizadas de Linux para sistemas embebidos, optimizando el proceso de construcción y facilitando la integración de controladores y aplicaciones específicas. Esta flexibilidad es esencial para abordar los desafíos que presentan los sistemas embebidos, donde cada proyecto puede requerir un enfoque único en términos de hardware y software.

[1] Un *Board Support Package (BSP)* es un conjunto de controladores, bibliotecas y firmware específicos que permiten que un sistema operativo funcione en una plataforma de hardware particular. En el contexto de Yocto, un BSP incluye los componentes necesarios para adaptar Linux a un hardware específico como Raspberry Pi o BeagleBone.

2.2.2 Ventajas y desventajas de utilizar *Yocto* para sistemas de propósito específico

El uso de esta herramienta presenta beneficios, pero también desventajas que deben tomarse en cuenta para determinar su idoneidad en un proyecto específico. En la *Tabla 7* se contrastan los puntos más importantes encontrados en la literatura consultada [28], [29], [30], [31].

Tabla 7: Ventajas y desventajas del uso de *Yocto* en sistemas embebidos

Ventajas	Desventajas
Flexibilidad y personalización granular mediante el concepto de *layers* y *recipes*, que permite generar distribuciones altamente adaptadas al hardware y a los requisitos de la aplicación	Curva de aprendizaje pronunciada: la complejidad de los metadatos y la multitud de variables/sobrecargas de BitBake exige experiencia avanzada
Reproducibilidad y trazabilidad del proceso de construcción gracias a BitBake y los mecanismos de `sstate-cache`, lo que facilita auditorías y cumplimiento normativo.	Tiempos de compilación prolongados y elevado consumo de recursos (CPU, RAM, almacenamiento) en la primera construcción completa.
Soporte multiarquitectura y ecosistema industrial con BSPs mantenidos por la comunidad y por proveedores comerciales (Wind River, Intel, etc.) y versiones LTS.	Sobrecoste para prototipado rápido: en proyectos de ciclo de vida corto, herramientas más ligeras (p. ej. *Buildroot*) pueden resultar más ágiles.
Generación automática de SDKs coherentes con la imagen de tiempo de ejecución, lo que simplifica la integración continua y el desarrollo de aplicaciones de usuario.	Depuración compleja de recetas y dependencias: los fallos pueden propagarse a través de múltiples capas, dificultando la localización del origen.
Escalabilidad y mantenibilidad de proyectos de gran tamaño gracias al paradigma de capas, que favorece la modularidad y el versionado independiente de componentes.	Documentación extensa, pero dispersa: la rápida evolución de capas externas puede provocar incompatibilidades y fallos de descarga (*fetch errors*).

En síntesis, aunque *Yocto Project* constituye una alternativa de gran solidez y flexibilidad, capaz de orquestar dependencias complejas y sostener proyectos de gran escala, esa misma robustez conlleva una curva de aprendizaje pronunciada y procesos de compilación más extensos, lo que puede elevar los costos de adopción y mantenimiento. Ahora bien, para desarrollos de menor envergadura o prototipos que requieren iteraciones rápidas, es importante destacar que existen otras herramientas más livianas como *Buildroot* que resultan ventajosas, ya que su simplicidad reduce la barrera de entrada y acelera la generación de imágenes, favoreciendo tiempos de desarrollo significativamente más cortos,

como se señala en [30].

2.3 Internet de las Cosas y computación en la nube para sistemas embebidos

La implementación de sistemas de control de acceso basados en reconocimiento facial se posiciona en un contexto donde es necesario integrar múltiples tecnologías y paradigmas. En este sentido, se puede articular el sistema propuesto en torno a tres ejes tecnológicos, como lo son el Internet de las Cosas (IoT), la computación en la nube y el procesamiento de datos biométricos. En esta sección se describen los conceptos fundamentales de los dos primeros ejes, y cómo se interrelacionan para formar sistemas distribuidos. Posteriormente, en la siguiente sección se abordará el tema del procesamiento de datos biométricos en el contexto de técnicas y herramientas de reconocimiento y detección facial.

2.3.1 Internet de las Cosas en los sistemas embebidos

Según Nimodiya y Ajankar, "*el Internet de las Cosas es simplemente una interacción entre el mundo físico y el mundo digital, y constituye una red de objetos físicos o personas llamadas "cosas" que están embebidas con software, electrónica, sensores y conectividad de red para recopilar y compartir datos*" [32]. Por su parte, Dauda et al. resaltan que el paradigma IoT se ha convertido en un habilitador esencial de la innovación en numerosos ámbitos de la actividad humana, incluidos la manufactura, la agricultura, la salud, la educación, la seguridad y el transporte [33].

En el sector manufacturero, por ejemplo, sensores y actuadores conectados permiten supervisar el rendimiento de los equipos, rastrear inventarios en tiempo real, optimizar las cadenas de suministro y habilitar el mantenimiento predictivo, lo que reduce al mínimo los tiempos de inactividad, incrementa la eficiencia productiva y disminuye los costos operativos [33].

Según una publicación de Statista, se prevé que el número de dispositivos IoT conectados duplicará los 15,9 mil millones de dispositivos en 2023, alcanzando los más de 32,1 mil millones en 2030 [34]. Esta tendencia resalta el papel fundamental de la conectividad y la interoperabilidad entre dispositivos, lo que permite crear sistemas inteligentes y distribuidos, que son los que dominan el mercado actual.

De acuerdo con Nimodiya y Ajankar, la literatura converge en un conjunto de rasgos que definen la esencia operativa, tecnológica y social del IoT. En la *Tabla 8* se resumen las características citadas en [32].

Tabla 8: Principales características del paradigma IoT

Característica	Descripción
Inteligencia	Integración de algoritmos y cómputo embebido que dota a los dispositivos de capacidad para percibir su contexto, razonar y actuar de forma autónoma, habilitando la llamada *inteligencia de ambiente*.
Conectividad	Habilita la accesibilidad a redes y la compatibilidad entre dispositivos heterogéneos, reforzando el valor de la información conforme al principio de la *Ley de Metcalfe* [2].
Naturaleza dinámica	Los nodos cambian de estado (activo, inactivo, conectado, desconectado) y de contexto (p. ej., posición, velocidad, temperatura), generando datos en tiempo real que requieren gestión continua.
Escala masiva	El número de dispositivos que deben ser gestionados y conectados entre sí, supera con creces al número de conexiones en la Internet tradicional, multiplicando los retos de direccionamiento, almacenamiento y procesamiento eficiente de grandes volúmenes de datos.
Sensado	Uso extensivo de sensores que capturan variables del entorno físico, posibilitando la creación de representaciones digitales y actualizadas del mundo real.
Heterogeneidad	Coexistencia de plataformas de hardware, protocolos de comunicación y modelos de datos diversos; el diseño de soluciones IoT debe garantizar interoperabilidad, extensibilidad y modularidad.
Seguridad	Los dispositivos y los datos están expuestos a un amplio espectro de amenazas; la protección de los servicios, redes y procesos exige arquitecturas de confianza, cifrado y gestión de la privacidad.

Estos atributos hacen que el IoT se haya convertido en un habilitador transversal de la innovación; su conectividad, capacidad de sensado y procesamiento de datos permiten la creación de sistemas inteligentes y distribuidos cuyas posibles aplicaciones son prácticamente ilimitadas. Esta variedad en su aplicabilidad, ha dado lugar a un ecosistema diverso de tecnologías, protocolos y arquitecturas, que van desde dispositivos de bajo consumo y bajo costo hasta sistemas complejos de análisis de datos en tiempo real.

La literatura reciente señala que los sistemas IoT pueden clasificarse, a grandes rasgos, en tres estilos arquitectónicos complementarios: *on-cloud* (en la nube), *on-edge* (en el

[2] La *Ley de Metcalfe* establece que el valor de una red es proporcional al cuadrado del número de usuarios conectados.

borde) y *fog* (de niebla). Cada uno de estos estilos aborda diferentes aspectos del procesamiento y almacenamiento de datos, y su elección depende de los requisitos específicos de la aplicación, respondiendo a compromisos entre latencia, escalabilidad y consumo de recursos.

- **Arquitectura *on-cloud*.** Todo el flujo de datos se encamina a plataformas en la nube con capacidad prácticamente ilimitada de cómputo y almacenamiento. Este modelo facilita la orquestación y el mantenimiento centralizados, así como la explotación de servicios avanzados (analítica masiva, *machine learning as-a-service*, etc.), a costa de una mayor dependencia de la conectividad de red y de un retardo apreciable en aplicaciones sensibles al tiempo [33].

- **Arquitectura *on-edge*.** El procesamiento se traslada a los dispositivos periféricos o a unidades locales con el fin de acercar la computación a la fuente de los datos. Con ello se reduce la congestión debido a la red, se atenúa la latencia y se alivian problemas de privacidad, ya que, de ser necesario únicamente se envían a la nube datos filtrados o agregados. Al mismo tiempo, el software debe enfrentarse a desafíos como la heterogeneidad y a las limitaciones de capacidad de cómputo de los nodos del borde [33].

- **Arquitectura *fog*.** Actúa como un nivel intermedio y distribuido que combina la capacidad de integración global propia de la nube con la inmediatez del borde. Los nodos *fog* se ubican en la red de acceso y ofrecen servicios de cómputo, almacenamiento temporal y control de seguridad próximos al lugar donde se generan los eventos, al tiempo que mantienen sincronización con la nube para tareas de larga duración o con requisitos intensivos de recursos [35].

Este abanico de estilos no debe entenderse como excluyente, sino como un complemento sobre el que el sistema complejo distribuye su lógica según sus restricciones concretas de latencia, ancho de banda, autonomía energética o confidencialidad de los datos. Ahora bien, en el caso de que el sistema requiera de un procesamiento intensivo de datos, como es el caso del reconocimiento facial, las arquitecturas de tipo *on-cloud* y *fog* se presentan como opciones atractivas debido a su capacidad para manejar grandes volúmenes de datos y realizar análisis complejos. En ambos casos, la computación en la nube se convierte en un componente esencial del sistema.

2.3.2 Computación en la nube y su papel en arquitecturas IoT

La computación en la nube se define, de acuerdo con la National Institute of Standards and Technology (NIST), como "*un modelo para permitir el acceso ubicuo, conveniente y bajo demanda a un conjunto compartido de recursos informáticos configurables (p. ej., redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser aprovisionados y*

liberados rápidamente con un esfuerzo mínimo de gestión o interacción del proveedor" [36].

Se establecen cinco características esenciales (autoservicio bajo demanda, amplio acceso a la red, agrupación de recursos, elasticidad rápida y servicio medido), tres modelos de servicio (IaaS, PaaS, SaaS) y cuatro modelos de despliegue (nube pública, privada, comunitaria e híbrida), sentando las bases conceptuales que distinguen a la nube frente a enfoques tradicionales "*on-premises*" o locales [36].

La migración desde centros de datos locales hacia la nube se explica, en gran medida, por los beneficios de elasticidad y factura por uso, descritos tempranamente por Armbrust et al. [37] y reforzados por prácticas recomendadas en marcos industriales como el *AWS Well-Architected Framework - IoT Lens* [38]. En esta evolución, los servicios de infraestructura (IaaS) abstraen hardware y redes, las plataformas (PaaS) ofrecen entornos gestionados para desplegar aplicaciones sin lidiar con sistemas operativos subyacentes, y el software como servicio (SaaS) entrega soluciones completas listas para el usuario final. El resultado es un espectro de opciones que permiten a los sistemas IoT, delegar cargas intensivas de cómputo a la nube, aprovechar economías de escala y acortar el "*time-to-market*" o tiempo de comercialización mediante el consumo de servicios gestionados.

Para ejemplificar el papel de la computación en la nube en arquitecturas IoT, se puede considerar el trabajo de Thilakarathne et al. [39], que presenta una plataforma de gestión de cultivos para un invernadero de tomates que combina dos microcontroladores (NodeMCU y Arduino UNO) con una red de sensores ambientales y de suelo. Los datos se envían mediante *Wi-Fi* a la nube pública de *Thinger.io*, donde se almacenan en *buckets*, se visualizan en diagramas de seguimiento y se habilita el control remoto de actuadores (bombas peristálticas e iluminación).

Esta arquitectura *edge–cloud* ilustra la transición desde soluciones puramente locales a modelos PaaS/SaaS: el borde (microcontroladores) capta y pre-procesa la información, mientras que la nube ofrece capacidad elástica de cómputo, persistencia y difusión en tiempo real, simplificando la analítica y la automatización de decisiones agrícolas.

Por su parte, Permana et al., describen en [40] un sistema inteligente de control de acceso basado en cerraduras electrónicas gestionadas por un ESP32 que integra sensor de huellas y teclado. El dispositivo registra eventos y recibe comandos a través de *Firebase Realtime Database*, un servicio de *Backend-as-a-Service* que abstrae la infraestructura subyacente.

Gracias a la computación en la nube, el sistema logra sincronización inmediata entre la cerradura y la aplicación Android del usuario, mantiene históricos de actividad y aplica principios de *Zero Trust* al delegar la autenticación y la autorización en servicios gestionados. Este caso evidencia cómo la computación en la nube potencia a la IoT en escenarios de seguridad física, al proporcionar escalabilidad, baja latencia de notificación y alta disponibilidad sin necesidad de servidores locales.

En síntesis, la incorporación de la computación en la nube en arquitecturas IoT aporta elasticidad para escalar el procesamiento según la variabilidad de la carga, alta disponibi-

lidad gracias a infraestructuras redundantes distribuidas globalmente, y un modelo pago por uso que traslada el gasto de capital a gasto operativo.

Además, los servicios gestionados, como PaaS o BaaS, reducen la complejidad de mantener servidores propios, facilitan la integración de *dashboards* y analítica avanzada, acelerando la iteración de prototipos y el tiempo de comercialización. En conjunto, estas ventajas permiten a los equipos de desarrollo concentrarse en la lógica de dominio mientras delegan la gestión de la infraestructura a proveedores de nube consolidados.

No obstante, persisten retos significativos como: la latencia y la dependencia de conectividad que pueden comprometer funciones críticas en entornos con redes inestables [41]; la protección de datos sensibles y el cumplimiento regulatorio exigen políticas de cifrado extremo a extremo, segmentación y *Zero Trust* [42]. Además, el riesgo de *vendor lock-in*[3] limita la portabilidad de las soluciones [43]; y la naturaleza variable e impredecible de la facturación en la nube introduce incertidumbre presupuestaria [44].

Paralelamente, la sostenibilidad energética de grandes centros de datos y el dimensionamiento correcto entre lógica en el borde y en la nube continúan siendo áreas activas de investigación. Estos desafíos impulsan el interés por arquitecturas híbridas *edge/fog-cloud* y mecanismos estandarizados de interoperabilidad que mitiguen las limitaciones sin renunciar a los beneficios del paradigma de la computación en la nube [45], [46].

2.4 Herramientas de reconocimiento y detección facial

El procesamiento y análisis automatizado de datos biométricos es un aspecto fundamental en el desarrollo de sistemas de control de acceso. Particularmente, los sistemas que utilizan la verificación de identidad por medio de aspectos faciales, suelen dividir el proceso en dos etapas: la detección de rostros y el reconocimiento facial. El propósito de esta sección es presentar una visión general de estas dos etapas, así como de las herramientas y técnicas más relevantes en el ámbito del reconocimiento facial.

La detección facial se define como el procedimiento que, a partir de una imagen fija o un flujo de vídeo, localiza las regiones que contienen rostros humanos y las aísla del fondo y de otros objetos presentes en la escena, entregando como resultado la posición espacial del rostro mediante un recuadro o una máscara [47]. Por su parte, el reconocimiento facial comprende el análisis del rostro previamente detectado para extraer un conjunto de características discriminantes (vector de rasgos) y compararlas con plantillas almacenadas en una base de datos, con el fin de verificar si corresponde a una identidad conocida o determinar la identidad más probable de la persona [47].

[3]El riesgo de *vendor lock-in* se refiere a la dificultad de cambiar de proveedor de servicios en la nube debido a problemas como la dependencia de tecnologías muy específicas o incapacidad para migrar datos.

2.4.1 Algoritmos y modelos populares de detección facial

En las últimas dos décadas la detección automática de rostros ha evolucionado desde técnicas basadas en descriptores manuales hasta detectores profundos de una sola etapa. Algunas de las soluciones de mayor adopción académica e industrial son las siguientes:

- **Viola–Jones / Haar Cascade**. Pionero en la detección en tiempo real mediante *features* Haar y un clasificador *AdaBoost* en cascada [48].
- **HOG + SVM (dlib)**. Emplea histogramas de gradientes orientados (HOG) y un clasificador lineal SVM; mantiene buena precisión con bajo coste computacional [49].
- **MTCNN**. Red convolucional en cascada que combina detección y alineamiento facial en un mismo flujo multi-tarea [50].
- **S^{3}FD**. Detector de una sola pasada (*single-shot*) diseñado para ser invariante a la escala, especialmente robusto para rostros pequeños [51].
- **BlazeFace**. Arquitectura ligera optimizada para GPU móviles, capaz de superar las 200 FPS en dispositivos de gama alta [52].
- **Familia YOLO (p. ej. YOLOv5-Face)**. Detectores de una sola pasada que combinan alta velocidad y buena exactitud; versiones afinadas para rostros añaden anclas y pérdidas específicas [53].
- **YuNet**. Detector ultraligero (∼76 mil parámetros) diseñado para dispositivos *edge*; alcanza 81 % *mAP* [4] en el *benchmark* WIDER FACE con una latencia de apenas 1.6 ms por imagen [1].

Los enfoques clásicos de detección facial se basan en la extración explícita de características (*features*) "hechas a mano", tales como los descriptores Haar combinados con *AdaBoost* [48] o histogramas de gradientes orientados [49]. Estos métodos utilizan un proceso en el que los investigadores definen manualmente el conjunto de descriptores matemáticos que capturan rasgos visuales considerados relevantes, tales como bordes, texturas, contrastes de intensidad, o formas geométricas, antes de entrenar el clasificador. Una vez extraídas estas características, fijas y de dimensión limitada, se alimentan clasificadores ligeros, como *AdaBoost* o SVM, que permiten detectar rostros en imágenes.

Estos métodos, por lo general, ofrecen una ejecución muy ligera sobre CPU al no requerir operaciones convolucionales profundas. Sin embargo, su desempeño se degrada en presencia de variaciones de iluminación, obstrucciones parciales y poses fuera del plano, y suelen mostrar menores valores de precisión (mAP) en bases de datos desafiantes frente a los detectores modernos.

[4] El *mAP* es el promedio de precisión (*mean Average Precision*) utilizado para evaluar la precisión de los modelos de detección de objetos.

En contraste, los detectores basados en aprendizaje profundo emplean redes neuronales convolucionales entrenadas de extremo a extremo y se dividen según su arquitectura, en esquemas de dos etapas (p. ej., MTCNN) y de una sola etapa (p. ej., YOLO, S^3FD). Estos modelos aprenden representaciones jerárquicas de características y exhiben un rendimiento superior ante variaciones de escala, rotación y obstrucciones, aunque requieren de un mayor consumo de recursos computacionales y, en muchos casos, demanda de aceleradores GPU o VPU para lograr latencias aceptables en tiempo real [50], [52], [53].

Ahora bien, la tendencia hacia arquitecturas ligeras y eficientes ha llevado al desarrollo de modelos como *YuNet*, que equilibra minuciosamente exactitud y eficiencia, permitiendo su uso en dispositivos de bajo consumo energético. Este modelo prescinde del uso de anclas, que son comunes en los detectores de una sola etapa, y emplea una arquitectura totalmente basada en convoluciones separables en profundidad, complementada con un *Tiny FPN* simplificado. Esto se traduce en únicamente 75 856 parámetros (∼300 kB) y 149 MFLOPs para una entrada de 320 × 320 píxeles, cifras que lo situan por debajo de la quinta parte del tamaño de otros detectores ligeros [1].

A pesar de su diseño compacto, el modelo YuNet alcanza un rendimiento competitivo, logrando 81.1 % de mAP en el set de alta dificultad de WIDER FACE [5] y registra una latencia media de 1,6 ms por imagen en CPU *Intel i7-12700K*, lo que habilita velocidades superiores a 600 FPS en resoluciones habituales y elimina la necesidad de aceleradores GPU externos [1].

En la *Tabla 9* se comparan las principales características de YuNet con otros detectores ligeros populares, como RetinaFace-0.25 y YOLO5-Face-n. En la tabla se subrayan los valores con mejor rendimiento en cada columna. Los modelos *YuNet-s*, *RetinaFace* y *SCRFD-10g* se incluyen en la lista como referencia pero no se consideran para la comparación, ya que el primero es una versión reducida de YuNet y los otros dos son detectores de alta complejidad que por su tamaño y latencia no son adecuados para sistemas embebidos.

Tabla 9: Comparación de YuNet con otros detectores faciales (adaptado de [1])

Tamaño imagen	**Método**	**Num. Parámetros** (proporción)	**FLOPs** (M)	**AP**$_{easy}$	**AP**$_{med}$	**AP**$_{hard}$	**Latencia** (ms)
320×320	SCRFD-0.5g	631 410 (8.32×)	195	0.850	0.754	0.372	3.4
	RetinaFace-0.25	426 608 (5.62×)	245	0.765	0.611	0.271	4.2
	YOLO5Face-n	446 376 (5.88×)	185	0.858	0.793	0.445	7.2
	YuNet	**75 856 (1.00×)**	149	0.836	0.747	0.395	2.2
	YuNet-s	54 608 (0.72×)	96	0.785	0.668	0.309	1.9
	RetinaFace	27 293 600 (359.81×)	11 070	0.868	0.742	0.341	49.1
	SCRFD-10g	4 229 905 (55.76×)	3 359	0.923	0.862	0.504	17.3
640×640	SCRFD-0.5g	–	779	0.907	0.882	0.684	17.8
	RetinaFace-0.25	–	981	0.893	0.831	0.541	22.0
	YOLO5Face-n	–	741	0.907	0.880	0.734	20.1
	YuNet	–	595	0.899	0.869	0.691	11.3
	YuNet-s	–	386	0.876	0.834	0.591	8.7

Continúa en la siguiente página

[5] WIDER FACE es un conjunto de datos de referencia para detección facial que agrupa más de 32 000 imágenes con casi 400 000 rostros anotados en condiciones extremadamente variadas de escala, pose, iluminación y oclusión, lo que lo convierte en el estándar más exigente para evaluar la robustez de los detectores modernos [54].

Tabla 9: Comparación de YuNet con otros detectores faciales (continuación)

Tamaño imagen	**Método**	**Num. Parámetros** (proporción)	**FLOPs** (M)	**AP**$_{easy}$	**AP**$_{med}$	**AP**$_{hard}$	**Latencia** (ms)
	RetinaFace	–	44 260	0.943	0.908	0.659	232.7
	SCRFD-10g	–	13 435	0.949	0.935	0.814	95.0
Tamaño orig.	SCRFD-0.5g	–	–	0.892	0.885	0.820	25.0
	RetinaFace-0.25	–	–	0.907	0.883	0.742	57.0
	YuNet	–	–	0.892	0.883	0.811	16.3
	YuNet-s	–	–	0.887	0.871	0.768	13.8
	RetinaFace	–	–	0.955	0.941	0.847	463.7
	SCRFD-10g	–	–	0.923	0.925	0.885	137.8

La tabla anterior permite observar que YuNet, a pesar de su reducida cantidad de parámetros, en los 3 casos logra el mejor rendimiento en términos de latencia, por encima de otros detectores ligeros que son 5 veces más grandes. Esto tiene relación con su rendimiento en términos de FLOPs, que también tiene los registros más bajos entre los detectores comparados, lo que se traduce en una menor carga computacional y, por ende, una mayor velocidad de procesamiento.

Además, YuNet alcanza un rendimiento competitivo en términos de precisión (AP), con valores muy cercanos a los del resto de detectores, y en algunos casos incluso superándolos, como en la métrica AP$_{medium}$ para el caso de imágenes de 320×320 píxeles, en el que YuNet supera RetinaFace a pésar de que este es 359 veces más grande en términos de número de parámetros.

Por otro lado, desde la óptica de la integración, el modelo se distribuye en formato ONNX y ha sido incorporado en la librería de modelos de OpenCV [6], de modo que puede invocarse con una pocas líneas de código en C++ o Python, lo que facilita su adopción en proyectos de sistemas embebidos. Su bajo consumo de memoria, junto con la política de entrenamiento a partir de una única escala de entrada y el uso de asignación *simOTA*, simplifica el ajuste de hiperparámetros y facilita su despliegue en dispositivos con recursos limitados [1].

2.4.2 Servicios en la nube para reconocimiento facial

A pesar de los avances en eficiencia y precisión de los algoritmos de detección y reconocimiento facial para dispositivos embebidos, estos sistemas enfrentan limitaciones inherentes de procesamiento, almacenamiento y consumo energético. Como solución a estas restricciones, los servicios de reconocimiento facial en la nube permiten delegar tareas computacionalmente intensivas a infraestructuras remotas, garantizando mayor escalabilidad, mantenimiento simplificado y actualizaciones constantes, como ya fue discutido en la sección 2.3. En la presente sección, se presenta una revisión de los principales servicios gestionados en la nube que ofrecen capacidades de reconocimiento facial, y que pueden integrarse con plataformas embebidas.

[6] OpenCV es una biblioteca de visión por computadora de código abierto que tiene una amplia gama de herramientas y algoritmos para el procesamiento de imágenes y videos.

- **Amazon Rekognition (AWS).** Servicio totalmente administrado (el proveedor se encarga de gestionar la configuración, mantenimiento, escalabilidad, actualizaciones y seguridad de la infraestructura subyacente) que detecta y analiza rostros en imágenes y video. Provee comparación uno-a-uno y uno-a-muchos, métricas de atributos (edad aparente, emociones, uso de gafas, etc.) y flujos de moderación de contenido. Se integra con otros servicios AWS (S3, Lambda, Kinesis) y ofrece precios por volumen de llamadas y por segundo de video procesado [55].

- **Azure Face API (Microsoft).** Incluye algoritmos para detección de rostros, verificación de identidad, agrupamiento y búsqueda en grandes colecciones. Soporta atributos faciales avanzados (pose, expresión, barba, accesorios) y perfiles de seguridad con Azure Active Directory. Dispone de niveles gratuitos y de pago por transacción, así como contenedores locales para escenarios *edge* [56].

- **Clarifai Face Recognition.** Plataforma basada en aprendizaje profundo que expone modelos pre-entrenados y personalizables para detección, verificación y búsqueda de rostros. Brinda SDKs en varios lenguajes, endpoints REST y gRPC, así como despliegue *on-premise* (local) para requisitos de privacidad estrictos. Se factura bajo un modelo de pago por uso con niveles gratuitos y escalado automático [57].

- **Face++ (Megvii).** Plataforma china con API REST y SDKs móviles que ofrece detección, verificación, búsqueda y análisis demográfico. Opera bajo un modelo de créditos pre-pago y cuenta con planes empresariales personalizados [58].

En la literatura se han documentado casos de uso exitosos de soluciones de reconocimiento facial en arquitecturas *edge-cloud*, que ilustran el potencial de estas herramientas, principalmente la de Amazon Rekognition, para ampliar las capacidades de sistemas empotrados con recursos limitados. Por ejemplo, el trabajo de Patel et. al. [59], desarrolla un sistema de videovigilancia doméstico y accionamiento remoto de cerraduras basado en una Raspberry Pi 3B+.

El sistema utiliza un nodo local que captura únicamente fotografías, en lugar de video continuo, cuando se detecta presencia de movimiento por sensor infrarrojo o se pulsa el timbre. Las imágenes se envían a un *bucket* de Amazon S3; allí, una función AWS Lambda entrega la imagen a Amazon Rekognition, almacena los vectores faciales en DynamoDB y notifica al usuario vía aplicación móvil y correo electrónico. Si la persona es reconocida, un disparador Lambda instruye a la Raspberry Pi para accionar el relé de la cerradura y mostrar un saludo personalizado en pantalla LCD. Este diseño traslada la pesada tarea de inferencia a la nube, reduce el ancho de banda (solo se suben fotografías instantáneas) y aprovecha los servicios gestionados de AWS para orquestar elasticidad, notificaciones y persistencia de eventos [59].

Otro ejemplo es el de Kanna et. al. [60], en el que se demuestra la viabilidad de un sistema inteligente de asistencia en campus que fusiona RFID y reconocimiento facial. Una Raspberry Pi 4B gestiona simultáneamente la lectura de etiquetas MFRC522 y la

captura de 30 FPS con una cámara USB. Cada segundo se genera un fotograma compuesto que se envía a una colección de Amazon Rekognition, donde se comparan los rostros con los registrados, aplicando un umbral de similitud del 85%. Cuando el UID RFID y la cara coinciden, la asistencia se marca automáticamente y se sincroniza en tiempo real con Firebase, permitiendo a administradores y estudiantes consultar registros desde una aplicación Android. La arquitectura aprovecha a Rekognition para la identificación fiable bajo iluminación y poses variables, mientras que Firebase simplifica el *backend* de datos y las notificaciones, conservando en la Raspberry Pi solo el procesamiento ligero de captura y transmisión.

Ambos casos prácticos evidencian la estrategia de delegar las operaciones de visión por computador a un servicio gestionado, manteniendo en el dispositivo empotrado tareas de adquisición de datos y control de actuadores. La combinación de hardware de bajo costo (Raspberry Pi), Amazon Rekognition como motor de análisis centralizado y servicios auxiliares (S3, DynamoDB, Lambda, Firebase) permite construir soluciones robustas, escalables y de fácil mantenimiento, superando las limitaciones de cómputo, almacenamiento y actualización de firmware típicas de los sistemas embebidos tradicionales.

Capítulo 3

Marco metodológico

El presente capítulo describe la estrategia metodológica empleada para el desarrollo del sistema propuesto, detallando las diferentes fases, enfoques y herramientas utilizadas. El enfoque metodológico adoptado se fundamenta en un híbrido entre dos paradigmas: el modelo en cascada, que es más tradicional y estructurado, y las prácticas ágiles, que permiten una mayor flexibilidad y retroalimentación temprana al utilizar un desarrollo incremental.

Esta combinación se justifica por la naturaleza del proyecto, en condición de Trabajo Final de Graduación, que requiere un enfoque riguroso y extensamente documentado, con poco margen para cambios significativos en los requisitos una vez definidos, pero que también se beneficia de la iteración y la retroalimentación continua para asegurar que el sistema cumple con las expectativas del usuario final.

A continuación, se describe con mayor detalle cada una de las etapas del proceso, así como un desglose de la correlación entre el marco metodológico y los objetivos específicos del proyecto, planteados en la sección 1.3.2.

3.1 Explicación del enfoque metodológico

El modelo en cascada o *waterfall*, descrito en [61], se caracteriza principalmente por su enfoque secuencial, donde cada fase del desarrollo debe completarse antes de pasar a la siguiente. A grandes rasgos, un proceso de este tipo sigue las siguientes etapas:

1. **Requerimientos:** En esta fase se recogen y documentan los requisitos del sistema, asegurando que se entienden las necesidades del cliente y los objetivos del proyecto.
2. **Diseño:** Se elabora un diseño detallado del sistema, incluyendo la arquitectura, los componentes y las interfaces necesarias para cumplir con los requisitos establecidos.
3. **Implementación:** En esta etapa se desarrollan los componentes del sistema según el diseño previamente definido, utilizando las herramientas y tecnologías adecuadas.

4. **Verificación:** Se realizan pruebas para verificar que el sistema cumple con los requisitos especificados, asegurando su funcionalidad y calidad.

5. **Mantenimiento:** Una vez que el sistema está en funcionamiento, se realizan ajustes y mejoras según sea necesario, abordando cualquier problema que surja durante su operación.

A pesar de que este modelo no suele ser la primera opción para proyectos de hardware y software, para los cuales normalmente se prefiere un enfoque ágil, sigue siendo una metodología que presenta ventajas para proyectos con ciertas particularidades. Por ejemplo, en proyectos donde el alcance y los entregables están bien definidos desde el inicio, la estructura secuencial del modelo en cascada funciona bastante bien. Esta característica se alinea convenientemente con la naturaleza del Trabajo Final de Graduación, donde los objetivos y requisitos deben ser claros desde las primeras etapas del proyecto, y aunque es posible modificar el alcance, idealmente se busca evitar cambios significativos una vez que se ha comenzado el desarrollo.

Otra ventaja del modelo en cascada es su enfoque en la documentación exhaustiva, lo que facilita la comprensión del sistema por parte de todos los involucrados y proporciona una base sólida para futuras referencias. Esto es particularmente útil para proyectos académicos, en los que la documentación técnica y el registro de decisiones son esenciales para la evaluación y la validación del trabajo realizado. También, el modelo en cascada permite dar un seguimiento claro del progreso del proyecto, ya que cada fase debe completarse antes de avanzar a la siguiente, lo que puede facilitar la gestión del tiempo, el cual es limitado en este contexto.

Ahora bien, algunos de los puntos débiles del modelo *waterfall* que se buscaron mitigar con la incorporación de prácticas ágiles son la rigidez en la adaptación a cambios y la falta de retroalimentación continua. En la metodología híbrida utilizada, se pretendió aplicar algunos de los 12 principios del Manifiesto Ágil, principalmente los siguientes:

1. *"Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software valioso [62]."*

2. *"Los cambios en los requisitos son bienvenidos, incluso en las etapas tardías del desarrollo. Los procesos ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente [62]."*

Se incorporaron estos dos principios en la **fase de implementación** del modelo en cascada mediante iteraciones cortas y revisiones frecuentes del sistema desarrollado. Esto implicó que se siguieran las etapas secuenciales de captura de requerimientos y diseño de alto nivel de la solución; sin embargo, durante la etapa de implementación, se realizaron ciclos cortos de planificación, diseño más preciso de cada módulo, desarrollo de las tareas comprometidas y procesos de verificación. Esto con la intención de que con base en

la retroalimentación de la contraparte de la organización, fuera posible hacer ajustes y mejoras necesarias, pero que no afectaran significativamente el alcance del proyecto.

Posteriormente, se procede con la fase de verificación, donde se evalúa el sistema en su totalidad con pruebas de punto a punto, para las cuales se esperaba un alto grado de éxito, dado que se realizaron pruebas parciales durante la fase de implementación. La etapa de mantenimiento quedó fuera del alcance de este proyecto en su calidad de Trabajo Final de Graduación debido a las restricciones de tiempo, sin embargo, el sistema quedará integrado en la infraestructura de Fito App, por lo tanto, en su condición de producto comercial, será objeto de mantenimiento y mejoras continuas por parte del equipo de desarrollo de la organización.

En la Figura 4 se presenta un diagrama que ilustra el enfoque metodológico adoptado, destacando las fases del modelo en cascada y la integración de prácticas ágiles durante la fase de implementación.

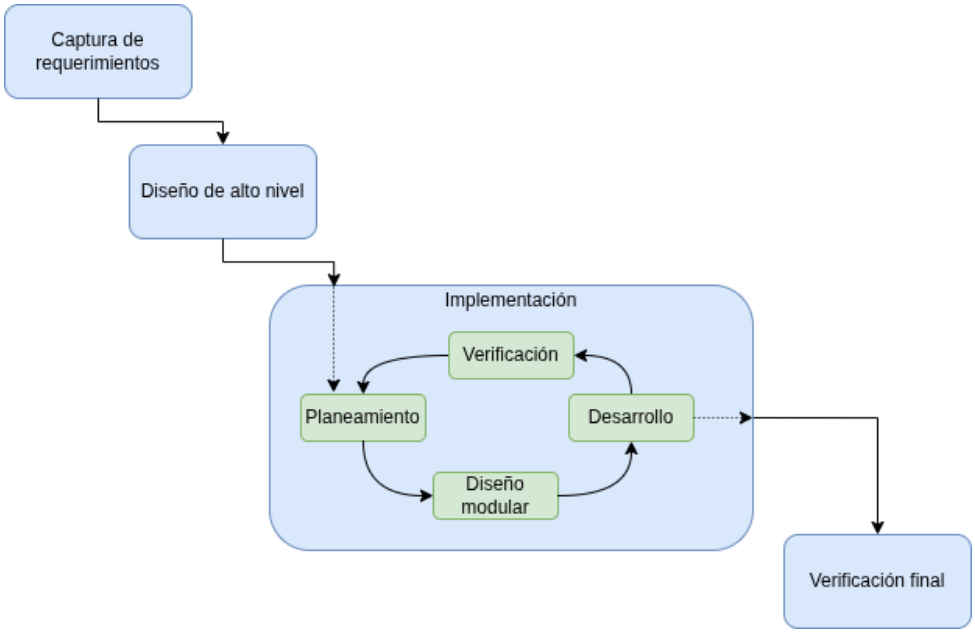

Fig. 4: Proceso metodológico adoptado para el desarrollo del sistema propuesto.

Algunos puntos importantes a considerar en la implementación de este enfoque metodológico son:

- Los ciclos de implementación tuvieron una duración de dos semanas.
- Cada ciclo incluyó una reunión de planificación con la contraparte de la organización, donde se definió el alcance y las tareas a realizar.
- Cada ciclo incluyó una etapa de diseño específica para cada módulo a desarrollar, pero que partió del diseño de alto nivel previamente establecido.
- Se mantuvo un tablero virtual al estilo Kanban para el seguimiento del estado de las tareas y el progreso del proyecto.

- Al final de cada ciclo se hizo una revisión del trabajo realizado con la contraparte organizacional, las cuales tomaron forma de pruebas de aceptación de usuario o demostraciones del sistema.
- En las sesiones de verificación se obtuvo retroalimentación continua, lo que permitió realizar ajustes y mejoras que se reflejaron en los ciclos siguientes.

3.2 Actividades desarrolladas para cada objetivo específico del proyecto

En la Tabla 10 se presenta un desglose de las actividades realizadas para cada uno de los objetivos específicos planteados en la sección 1.3.2, junto con sus entregables asociados, técnicas o herramientas utilizadas y estrategias de verificación implementadas. Para mayor facilidad de lectura, se utiliza el identificador de cada objetivo específico:

- **OE1**: Diseñar la arquitectura de un sistema embebido de control de acceso, que permita la captura y análisis de datos biométricos, mediante el uso de una plataforma de bajo costo con un sistema operativo de propósito específico.
- **OE2**: Desarrollar un módulo de reconocimiento facial, para la validación de la identidad de los usuarios en el punto de ingreso al gimnasio, mediante la integración de una cámara y el uso de servicios en la nube.
- **OE3**: Integrar un módulo de reportes en el sistema de administración de Fito App, que permita la consulta de estadísticas de acceso de los usuarios, mediante una interfaz web.
- **OE4**: Evaluar el rendimiento del sistema en un entorno real, para la verificación del cumplimiento de los requisitos de latencia, precisión y experiencia de usuario, mediante pruebas de campo controladas.

Los entregables asociados a cada objetivo específico se pueden consultar en la Tabla 5, donde se añadió una breve descripción de cada entregable. Ahora bien, a continuación se enumeran dichos entregables con su respectivo identificador único que será referenciado en la tabla:

- **E1:** Documento de arquitectura
- **E2:** Unidad de interacción y detección facial
- **E3:** Imagen de sistema operativo para nodo embebido
- **E4:** Servicio de reconocimiento facial y autorización

- **E5:** Módulo de reportes
- **E6:** Informe de pruebas de rendimiento

El listado de actividades desarrolladas para alcanzar el cumplimiento de los objetivos específicos del proyecto se presenta a continuación. Cada actividad también está asociada a un identificador único que se utiliza en la tabla para facilitar su referencia:

- **A1-1:** Levantamiento de requisitos del sistema.
- **A1-2:** Selección comparativa de la plataforma de hardware.
- **A1-3:** Diseño de arquitectura física del hardware: mapeo de pines y buses de comunicación.
- **A1-4:** Diseño de arquitectura lógica del sistema.
- **A2-1:** Modelado de la experiencia de usuario y guion de pantallas.
- **A2-2:** Desarrollo del módulo de detección facial en nodo *on-edge*.
- **A2-3:** Integración del módulo de detección facial con la interfaz de usuario.
- **A2-4:** Configuración de Yocto Project y generación de la primera imagen mínima.
- **A2-5:** Personalización de la imagen.
- **A2-6:** Adaptación de base de datos para el registro de accesos.
- **A2-7:** Habilitación de repositorio para almacenamiento de imágenes de los rostros de los usuarios.
- **A2-8:** Desarrollo de servicio *serverless* para el registro de usuarios.
- **A2-9:** Desarrollo de servicio *serverless* para el reconocimiento de usuarios y validación de membresía.
- **A2-10:** Publicación de *endpoints* para utilizar los servicios.
- **A3-1:** Especificación de requisitos de información de reportes de acceso.
- **A3-2:** Implementación de servicios REST para el consumo de reportes de acceso.
- **A3-3:** Desarrollo de la interfaz de usuario para la visualización de reportes.
- **A4-1:** Elaboración del plan de pruebas.
- **A4-2:** Preparación del entorno de pruebas en gimnasio piloto.
- **A4-3:** Ejecución de pruebas de punto a punto del sistema.
- **A4-4:** Análisis estadístico de resultados y comparación con los objetivos de desempeño.

Tabla 10: Desglose de actividades desarrolladas para cada objetivo específico.

Obj. Esp.	Entregables	Actividades	Herramientas/Técnicas	Estrategias de verificación
OE1	E1	A1-1, A1-2, A1-3, A1-4	• Entrevistas con partes interesadas para el levantamiento de requisitos. • Revisión de hoja de datos de Raspberry Pi 4B (plataforma disponible en la organización) y de sus posibles alternativas. • Revisión de hoja de datos de cámaras compatibles. • Revisión de hoja de datos de pantalla táctil compatible. • Uso de software Draw.io para diagramas de arquitectura.	• Aprobación de los requisitos del sistema por parte de la contraparte de la organización. • Aprobación del documento de arquitectura física y lógica del sistema por parte de la contraparte de la organización. • Verificación que el costo total de la plataforma de hardware no excede los $300 USD.

Continúa en la siguiente página

Tabla10: Desglose de actividades desarrolladas para cada objetivo específico (continuación).

Obj. Esp.	Entregables	Actividades	Herramientas/Técnicas	Estrategias de verificación
OE2	E2, E3, E4	A2-1, A2-2, A2-3, A2-4, A2-5, A2-6, A2-7, A2-8, A2-9, A2-10	• Generación bosquejo (*mockup*) de interfaz gráfica de usuario usando *Draw.io*. • Uso de *git* para el control de versiones del código fuente. • Implementación de detección facial con *C++* y *OpenCV*. • Generación de imagen Linux reproducible con *poky* (Yocto). • Orquestación de *builds* con *BitBake* (Yocto). • Activación de capas como *meta-yocto-bsp*, *meta-poky* y *meta-raspberrypi* para definir la plataforma y distribución (Yocto). • Generación de SDK para permitir el desarrollo de aplicaciones sobre la imagen (Yocto). • Creación o modificación de *recipes* y archivos *append* para incluir dependencias y configuraciones específicas (Yocto). • Configuración de caché de estado para acelerar el proceso de compilación mediante el uso de *sstate-cache* (Yocto).	• Pruebas de usabilidad con los bosquejos de interfaz gráfica de usuario. • Prueba de despliegue de la imagen Yocto en la Raspberry Pi 4B. • Pruebas unitarias del módulo de detección facial que validan tasa de detección superior al 95% con un banco de imágenes de referencia. • Pruebas de *endpoints* de servicios REST con *Postman* (tipos de solicitudes, códigos de error, esquema de respuesta). • Pruebas de precisión (mayor al 95%) y latencia (menor a 500 ms) del servicio de reconocimiento facial con un banco de imágenes de referencia. • Pruebas de carga y resiliencia: hasta 50 peticiones concurrentes, sin degradación significativa de latencia ni errores 5XX. • Validación de registros de acceso en la base de datos de Fito App.

Continúa en la siguiente página

Tabla10: Desglose de actividades desarrolladas para cada objetivo específico (continuación).

Obj. Esp.	Entregables	Actividades	Herramientas/Técnicas	Estrategias de verificación
			• Uso de *PostreSQL* para la modificación de la base de datos de Fito App y su adaptación a los requerimientos del sistema. • Configuración de un *bucket* en *Amazon S3* para el almacenamiento de imágenes de rostros. • Configuración de Red de Distribución de Contenidos (*CDN*) para el consumo y distribución de imágenes. • Uso de *AWS Lambda* para el desarrollo de servicios *serverless*. • Uso de *API Gateway* para la creación de *endpoints REST*. • Uso de API de *Amazon Rekognition* para el reconocimiento facial. • Uso de *Python* para el desarrollo de servicios *serverless* y comunicación con APIs.	

Continúa en la siguiente página

Tabla10: Desglose de actividades desarrolladas para cada objetivo específico (continuación).

Obj. Esp.	Entregables	Actividades	Herramientas/Técnicas	Estrategias de verificación
OE3	E5	A3-1, A3-2, A3-3	• Uso de *Node.js*, *Drizzle ORM* y *Neon SDK* para el desarrollo de servicios REST. • Implementación con *React* y *Next.js* para la interfaz de usuario.	• Pruebas de usabilidad de la interfaz de reportes con usuarios administradores para evaluar claridad de las gráficas, tablas y filtros. • Validación de consistencia de datos: comparación de un subconjunto de estadísticas contra los registros de acceso en la base de datos garantizando una desviación menor al 1%.
OE4	E6	A4-1, A4-2, A4-3, A4-4	• Medición de métricas de latencia en el reconocimiento facial, mediante la librería `std::chrono` en *C++*. • Registro manual de tasas de aciertos y fallos en el reconocimiento facial. • Implementación de pruebas de aceptación de usuario (*UAT*) con usuarios reales del gimnasio piloto. • Uso de librerías de análisis estadístico y visualización de datos como *Matplotlib* y *NumPy* en *Python* para el análisis de resultados.	• Revisión del documento de los resultados y análisis por parte de la contraparte de la organización.

Bibliografía

[1] W. Wu, H. Peng, and S. Yu, "YuNet: A tiny millisecond-level face detector," *Machine Intelligence Research*, vol. 20, no. 5, pp. 656–665, 2023. [Online]. Available: https://doi.org/10.1007/s11633-023-1423-y

[2] Fito, "Fito App - Inicio," Online, available: https://fito-app.com/, Accessed: Nov. 5, 2024.

[3] Agencia Universitaria para la Gestión del Emprendimiento (AUGE-UCR), "Fondos de Capital Semilla," Online, available: https://augeucr.com/home/fondosauge/, Accessed: Apr. 8, 2025.

[4] Sistema de Banca para el Desarrollo, "Emprendimientos tendrán acceso a ₡3.300 millones para capital semilla," Online, available: https://sbdcr.com/emprendimientos-tendran-acceso-a-%E2%82%A13-300-millones-para-capital-semilla/, Accessed: Apr. 8, 2025.

[5] CMD Sport, "Animan al sector del fitness a seguir invirtiendo en Latinoamérica," Online, available: https://www.cmdsport.com/fitness/actualidad-fitness/animan-al-sector-del-fitness-seguir-invirtiendo-latinoamerica/, Accessed: Apr. 8, 2025.

[6] Mercado Fitness, "En Latinoamérica, las membresías aumentaron un 2.4% en 2023," Online, available: https://mercadofitness.com/latinoamerica-membresias-creciimiento-2023/, Accessed: Apr. 8, 2025.

[7] Mordor Intelligence, "Industria del fitness en América del Sur - Análisis de tamaño y participación - Tendencias y pronósticos de crecimiento (2024-2029)," Online, available: https://www.mordorintelligence.com/es/industry-reports/south-america-health-and-fitness-club-market, Accessed: Apr. 8, 2025.

[8] Amazon Web Services, "What is Amazon Rekognition?" Online, available: https://docs.aws.amazon.com/rekognition/latest/dg/what-is.html, Accessed: Apr. 8, 2025.

[9] Daosafe, "Daosafe - Turnstile Gates & Access Control Systems," Online, available: https://www.daosafe.com/, Accessed: Apr. 26, 2025.

[10] Daosafe Technology Co., Ltd., "Daosafe DS112 Tripod Turnstile Datasheet," Online, available: https://www.daosafe.com/, Accessed: Apr. 26, 2025.

[11] LatinSoft, "Fitness 24/7 Gym," Online, available: https://www.latinsoftcr.net/fitness247, Accessed: Apr. 20, 2025.

[12] Multinegocios Tecnológicos de Costa Rica S.A., "¿Qué ofrecemos en Multinegocios Tecnológicos para tu gimnasio?" Online, available: https://mntcr.com/, Accessed: Apr. 26, 2025.

[13] WStudio, "WStudio - Funciones," Online, available: https://www.wstudio.app/#funciones, Accessed: Apr. 20, 2025.

[14] M. Wolf, *Computers as Components: Principles of Embedded Computing System Design*, 3rd ed. Burlington, MA: Elsevier, 2012.

[15] M. Barr, *Programming Embedded Systems in C and C++*, 1st ed. Sebastopol, CA: O'Reilly & Associates, 1999.

[16] N. A. Jebril and Q. A. Al-Haija, "Review of Challenges in Embedded Systems Design: Robustness, Predictability, Security," *International Journal of Current Research in Embedded System & VLSI Technology*, vol. 3, no. 1, pp. 1–5, 2017.

[17] D. Henriksson, "Resource-Constrained Embedded Control and Computing Systems," Doctoral Thesis, Lund University, Lund, Sweden, 2006. [Online]. Available: https://portal.research.lu.se/files/4452427/546030.pdf

[18] R. K. Shyamasundar and J. Aghav, "Validating Real-Time Constraints in Embedded Systems," in *Proceedings of the 2001 Pacific Rim International Symposium on Dependable Computing (PRDC)*. IEEE, 2001, pp. 99–106.

[19] Wind River, "What Is Embedded Systems Security?" Online, 2025, available: https://www.windriver.com/solutions/learning/embedded-systems-security, Accessed: Apr. 29, 2025.

[20] E. A. Lee and S. A. Seshia, *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*, 2nd ed. MIT Press, 2017.

[21] M. Guerbaoui, N. Misbah, A. Selmani, S. E. Faiz, B. Benhala, A. Ed-Dahhak, and A. Lachhab, "Advanced Facial Recognition Systems for Real-Time Embedded Applications," in *E3S Web of Conferences*, vol. 601. EDP Sciences, 2025, p. 00109.

[22] S. Kalkar, S. Kelkar, R. Kajale, and S. M. Kulkarni, "Face Recognition in Embedded Systems for Security Surveillance," *International Research Journal of Engineering and Technology (IRJET)*, vol. 7, no. 6, pp. 3655–3662, 2020. [Online]. Available: https://www.irjet.net/archives/V7/i6/IRJET-V7I6683.pdf

[23] S. M. Hammami and M. Alhammami, "Robust Embedded Access Control System Based on Face and Encrypted QR with RPi4," *International Journal of Reconfigurable and Embedded Systems (IJRES)*, vol. 13, no. 3, pp. 586–594, 2024.

[24] A. Holt and C.-Y. Huang, *Embedded Operating Systems: A Practical Approach*, 2nd ed. Cham, Switzerland: Springer, 2018.

[25] K. Wang, *Embedded and Real-Time Operating Systems*. Cham, Switzerland: Springer, 2023.

[26] *Yocto Project Overview and Concepts Manual*, The Yocto Project, 2025, version 5.1.999. Accessed: May 3, 2025. [Online]. Available: https://docs.yoctoproject.org/overview-manual/index.html

[27] *BitBake User Manual*, The Yocto Project, 2025, documentación *dev*. Accessed: May 3, 2025. [Online]. Available: https://docs.yoctoproject.org/bitbake/

[28] R. J. Streif, *Embedded Linux Systems with the Yocto Project*. Addison-Wesley Professional, 2016.

[29] D. Abbott, *Linux for Embedded and Real-Time Applications*, 4th ed. Newnes, 2018.

[30] T. Perä, "Comparison of Custom Embedded Linux Build Systems: Yocto and Buildroot," Master's thesis, Aalto University, Espoo, Finland, 2022.

[31] H. Karacali, N. Donum, and E. Cebel, "Enhancing Workflow Efficiency in Yocto Project: A Build Tool for Fetch Error Detection and Fixing," *The European Journal of Research and Development*, vol. 4, no. 2, pp. 49–76, 2024.

[32] A. R. Nimodiya and S. S. Ajankar, "A Review on Internet of Things," *International Journal of Advanced Research in Science, Communication and Technology*, vol. 2, no. 1, pp. 135–144, January 2022.

[33] A. Dauda, O. Flauzac, and F. Nolot, "A Survey on IoT Application Architectures," *Sensors*, vol. 24, p. 5320, August 2024. [Online]. Available: https://doi.org/10.3390/s24165320

[34] L. S. Vailshery, "Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2023, with forecasts from 2024 to 2030," Statista, July 2023, accessed: May 9, 2025. [Online]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/

[35] P. Sethi and S. R. Sarangi, "Internet of Things: Architectures, Protocols, and Applications," *Journal of Electrical and Computer Engineering*, vol. 2017, no. 1, p. 9324035, 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1155/2017/9324035

[36] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," National Institute of Standards and Technology, Special Publication 800-145, Sep. 2011.

[37] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A View of Cloud Computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010.

[38] Amazon Web Services, "AWS Well-Architected Framework – IoT Lens," Whitepaper, Jan. 2024, accessed: 14-May-2025.

[39] N. N. Thilakarathne, M. S. A. Bakar, P. E. Abas, and H. Yassin, "Towards Making the Fields Talk: A Real-Time Cloud-Enabled IoT Crop-Management Platform for Smart Agriculture," *Frontiers in Plant Science*, vol. 13, p. 1030168, 2023.

[40] K. A. K. Permana, I. N. Piarsa, and A. A. K. A. C. Wiranatha, "IoT-Based Smart Door Lock System with Fingerprint and Keypad Access," *Journal of Information Systems and Informatics*, vol. 6, no. 3, pp. 2086–2095, 2024.

[41] F. C. Andriulo, M. Fiore, M. Mongiello, E. Traversa, and V. Zizzo, "Edge Computing and Cloud Computing for Internet of Things: A Review," *Informatics*, vol. 11, no. 4, p. 71, 2024.

[42] C. Zanasi, S. Russo, and M. Colajanni, "Flexible Zero Trust Architecture for the Cybersecurity of Industrial IoT Infrastructures," *Ad Hoc Networks*, vol. 156, p. 103414, 2024.

[43] A. Alhosban, S. Pesingu, and K. Kalyanam, "CVL: A Cloud Vendor Lock-In Prediction Framework," *Mathematics*, vol. 12, no. 3, p. 387, 2024.

[44] S. Deochake, "ABACUS: A FinOps Service for Cloud Cost Optimization," *arXiv preprint arXiv:2501.14753*, 2024.

[45] A. Shehabi, S. J. Smith, A. Hubbard, A. Newkirk, N. Lei, M. A. Siddik, B. Holecek, J. G. Koomey, E. R. Masanet, and D. A. Sartor, "2024 United States Data Center Energy Usage Report," Lawrence Berkeley National Laboratory, Tech. Rep., 2024.

[46] N. Fernando, S. Shrestha, S. W. Loke, and K. Lee, "On Edge-Fog-Cloud Collaboration and Reaping Its Benefits: A Heterogeneous Multi-Tier Edge Computing Architecture," *Future Internet*, vol. 17, no. 1, p. 22, 2025.

[47] S. Z. Li and A. K. Jain, *Handbook of Face Recognition*, 2nd ed. London: Springer, 2011.

[48] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001, pp. I–511–I–518.

[49] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005, pp. 886–893.

[50] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint Face Detection and Alignment using Multitask Cascaded Convolutional Networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.

[51] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li, "S3FD: Single Shot Scale-invariant Face Detector," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 192–201.

[52] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran, and M. Grundmann, "BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs," *arXiv preprint arXiv:1907.05047*, 2019.

[53] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[54] S. Yang, P. Luo, C. C. Loy, and X. Tang, "WIDER FACE: A Face Detection Benchmark," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[55] Amazon Web Services. (2025) Detecting and Analyzing Faces - Amazon Rekognition. Accessed: 27-May-2025. [Online]. Available: https://docs.aws.amazon.com/rekognition/latest/dg/faces.html

[56] Microsoft Corporation. (2025) What is the Azure AI Face service? Accessed: 27-May-2025. [Online]. Available: https://learn.microsoft.com/en-us/azure/ai-services/computer-vision/overview-identity

[57] Clarifai, Inc. (2025) Clarifai API Overview - Face Recognition. Accessed: 28-May-2025. [Online]. Available: https://docs.clarifai.com/resources/api-overview/

[58] Megvii Technology Limited. (2025) Face Detection - Face++ Cognitive Services. Accessed: 27-May-2025. [Online]. Available: https://www.faceplusplus.com/face-detection/

[59] J. Patel, S. Anand, and R. Luthra, "Image-based smart surveillance and remote door-lock switching system for homes," in *Procedia Computer Science*, vol. 165, 2020, pp. 624–630.

[60] P. V. Kanna, K. Anusuya, and P. Vaishnavi, "Smart attendance system using face recognition and rfid technology," in *Proc. 7th Int. Conf. on Computer Applications (ICCAP 2021)*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, 2021.

[61] W. W. Royce, "Managing the development of large software systems," in *Proceedings of the Technical Papers of the Western Electronic Show and Convention (WESCON)*. Los Angeles, CA, USA: IEEE, Aug. 1970, pp. 1–9.

[62] Agile Alliance, "Principles behind the Agile Manifesto," [Online], 2001, accedido: 22-jun-2025. [Online]. Available: https://agilemanifesto.org/principles.html