ULPGC
Universidad de
Las Palmas de
Gran Canaria

ESCUELA DE INGENIERÍA DE
TELECOMUNICACIÓN Y ELECTRÓNICA

TRABAJO FIN DE GRADO

Aplicación de la Tecnología Blockchain de Worldcoin en la Seguridad y Control de Acceso

Titulación: Grado en Ingeniería en Tecnologías de la Telecomunicación

Mención: Telemática

Autor: Carlos Manuel Ortega Negrín

Tutores: Álvaro Suárez Sarmiento

Fecha: Mayo 2025

Resumen

La gestión de identidades digitales y el control de acceso en entornos institucionales sigue siendo uno de los mayores desafíos de seguridad a nivel informático, especialmente en el contexto universitario, en el cual se maneja información sensible de estudiantes, docentes y personal administrativo, es fundamental implementar sistemas que garanticen la integridad, la privacidad y la autenticación segura de los usuarios.

Con este propósito, en el presente TFG se plantea el desarrollo de una simulación funcional de una aplicación de verificación de identidad descentralizada, denominada *ULPGC Verify*, basada en la infraestructura de *Worldcoin*, aprovechando las capacidades y beneficios de la tecnología *Blockchain* y de la identidad digital única generada por *World ID*, mediante el uso de Zero-Knowledge Proofs (*ZKP)* y su sistema de verificación biométrica con el dispositivo *Orb*, de esta forma se analizara el estado actual de los métodos de autenticación tradicionales, se evalúan sus limitaciones y se justifica la adopción de un modelo descentralizado más seguro y realizando una simulación desarrollada que permite al usuario realizar un proceso de verificación desde la aplicación móvil oficial de *Worldcoin*, vinculándose a una *MiniApp* externa construida con herramientas necesarias.

El sistema está diseñado para no almacenar datos personales ni realiza consultas a bases de datos institucionales, sino que valida únicamente la unicidad del usuario de forma segura y anónima. Por último, se lleva a cabo una evaluación del rendimiento, midiendo tanto el consumo de recursos como el tiempo de respuesta durante la ejecución, demostrando la viabilidad técnica integrando un sistema de autenticación descentralizado en un entorno académico.

Abstract

The management of digital identities and access control in institutional environments remains one of the greatest cybersecurity challenges, especially within the university context, where sensitive information related to students, faculty, and administrative staff is handled. It is therefore essential to implement systems that ensure user integrity, privacy, and secure authentication.

To address this, the present Final Degree Project proposes the development of a functional simulation of a decentralized identity verification application called ULPGC Verify. This system is based on the Worldcoin infrastructure, leveraging the capabilities and benefits of Blockchain technology and the unique digital identity generated by World ID. It makes use of Zero-Knowledge Proofs (*ZKP*) and biometric verification through the Orb device. The project analyses the current state of traditional authentication methods, evaluates their limitations, and justifies the adoption of a more secure decentralized model. A simulation is developed that allows users to carry out a verification process through Worldcoin's official mobile application, linking to an external MiniApp built with the necessary tools.

The system is designed not to store personal data or query institutional databases, but instead to validate the uniqueness of each user in a secure and anonymous manner. Finally, a performance evaluation is carried out, measuring both resource consumption and response times during execution, demonstrating the technical feasibility of integrating a decentralized authentication system in an academic environment.

ÍNDICE

1. Introducción

Se introduce, en este capítulo, el contexto general que motiva la realización de este *Trabajo de Fin de Grado* (*TFG*), abordan los antecedentes relacionados con la ciberseguridad y la protección de datos, introduciendo el concepto de tecnología *blockchain* y se justifica la elección de *Worldcoin* como solución tecnológica principal.

1.1 Antecedentes y motivación

A medida que las tecnologías avanzan, el volumen de información gestionada por las organizaciones aumenta al igual que los riesgos asociados a su seguridad. Los ciberataques dirigidos a estas infraestructuras se han vuelto más sofisticados, afectando sectores sensibles como la banca, el educativo, las administraciones públicas y el sector sanitario. Un ejemplo destacado de esta vulnerabilidad es lo ocurrido en marzo de 2023, cuando el Hospital Clínic de Barcelona fue objeto de un ciberataque por parte del grupo *Ransom House*, especializado en *ransomware* [1]. Este ataque provocó la cancelación de más de 150 cirugías no urgentes y la reprogramación de unas 2.000 visitas médicas, generando una situación grave tanto a nivel asistencial como organizativo [2]. Este suceso puso de manifiesto no solo la fragilidad de los sistemas informáticos actuales, sino también las deficiencias en las estrategias de protección de datos sensibles.

La dependencia de sistemas centralizados para el almacenamiento, la verificación de identidad y el control de acceso incrementa significativamente los riesgos de filtraciones, accesos no autorizados y pérdida de información. A pesar de la evolución de las medidas de ciberseguridad convencionales, los ataques siguen evolucionando y superando las defensas tradicionales, evidenciando que los modelos actuales no son suficientes para proteger adecuadamente la información más sensible [3].

El análisis de los sistemas de autenticación utilizados actualmente muestra que muchas organizaciones (universidades, hospitales, bancos y administraciones públicas) emplean un número elevado de métodos diferentes para controlar el acceso a sus servicios: usuario y contraseña, *Multi-Factor Authentication* (*MFA*), certificados digitales, federación de identidades, entre otros. Esta diversidad provoca un sistema disperso, poco eficiente y difícil de gestionar, ya que cada sistema tiene sus propios riesgos y complejidades. Un claro ejemplo es la *Universidad de Las Palmas de Gran Canaria* (*ULPGC*), que utiliza múltiples métodos de autenticación para distintos servicios internos o el Gobierno de Canarias y los hospitales públicos, donde los usuarios deben recordar y gestionar diferentes credenciales y certificados. Esta situación se repite en entidades bancarias y en muchas grandes organizaciones.

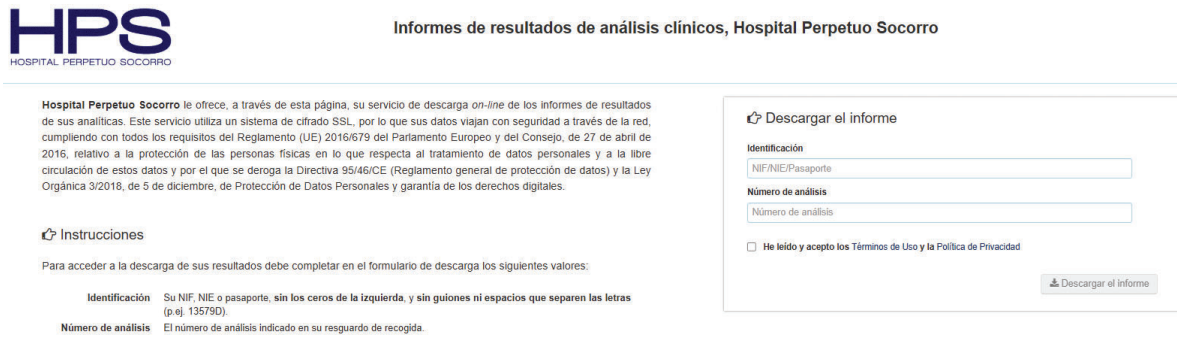

Figura 1.1 Acceso a informes médicos en el Hospital Perpetuo Socorro

Un ejemplo sería el actual sistema del Hospital Perpetuo Socorro de Las Palmas de Gran Canaria, para la consulta de los resultados médicos de sus pacientes. El usuario debe introducir su *Número de Identificación Fiscal* (*NIF*) o pasaporte junto con un número de análisis, como se muestra en la *Figura 1.1*. Este sistema resulta problemático, ya que depende de que el usuario conserve correctamente este número, lo que podría derivar en una pérdida de acceso a los datos médicos en caso de extravío, comprometiendo la seguridad y la accesibilidad a la información sanitaria.

Con esta problemática, el uso de múltiples métodos de autenticación descoordinados dentro de una Organización representa un problema de gestión y seguridad. La existencia de tantos mecanismos para autenticar usuarios añade complejidad innecesaria, dificulta el control de accesos y expone a mayores riesgos de vulnerabilidad. Lo ideal sería simplificar y fortalecer todos estos sistemas de acceso mediante una única tecnología robusta y segura, capaz de ofrecer mayores garantías en términos de seguridad, privacidad, integridad y resistencia frente a ataques. El desafío es encontrar soluciones que reduzcan la dependencia de intermediarios, reforzar la autenticación de usuarios y garanticen la trazabilidad de las acciones realizadas dentro de los sistemas digitales. Especialmente en sectores donde la protección de los datos resulta de gran importancia, como el sanitario o el académico [4].

1.2 La tecnología Blockchain

La tecnología *Blockchain* [5] es un sistema para el registro descentralizado, seguro e inmutable, agrupando los datos en bloques que se enlazan criptográficamente formando

una cadena continua. Cada bloque contiene un conjunto de transacciones o datos, y cada nuevo bloque está vinculado al anterior mediante un *hash* [6] criptográfico que garantiza la integridad de los datos. Esta tecnología elimina la necesidad del almacenamiento centralizado replicando la cadena en todos los nodos de la red y utiliza algoritmos de consenso como *Proof of Work* (*PoK*) [7] o *Proof of Stake* (*PoS*) [8] para validar y proteger la información. Cualquier intento de alterar la información requiere el consenso de un subconjunto cualificado de nodos, lo cual hace que los ataques o fraudes sean muy difíciles de llevar a cabo.

En los últimos años ha dejado de ser una tecnología exclusiva del ámbito financiero para evolucionar y ser aprovechada en sectores como la cadena de suministro, la gestión de derechos de autor, los sistemas de votación electrónica y la gestión de identidad digital (los usuarios pueden demostrar quiénes son sin depender de organismos que centralizadamente guardan esa información). En el contexto de la Identidad Digital Descentralizada (*DID*) [9], la *Blockchain* permite que los usuarios posean y controlen su propia identidad sin necesidad de confiar en un proveedor centralizado, como *Google*, *Microsoft* o una institución pública (por ejemplo, la ULPGC).

1.2.1 Elección de la cadena blockchain

Sabiendo que existen infinidad de *blockchain* hoy en día, en este TFG se opta por utilizar *Worldcoin* [10], construida sobre la *blockchain* de *Ethereum* [11]. Permite implementar sistemas de autenticación basados en *DID* con un enfoque innovador, que a diferencia de otras soluciones que utilizan únicamente *wallets* [12] o direcciones públicas. *Worldcoin* ha diseñado un sistema que garantiza que cada usuario sea único, utilizando pruebas criptográficas en lugar de almacenar los datos personales, este sistema se compone de cuatro elementos fundamentales (*Figura 1.2*):

- *Orb*: un dispositivo biométrico desarrollado por *Worldcoin* que escanea el iris del usuario y genera una codificación única (*hash*), que posteriormente se convierte en una prueba criptográfica [13].

- *World ID*: es la *DID* generada a partir del escaneo biométrico, que es verificada mediante *Zero-Knowledge Proofs* (*ZKP*) y registrada de forma privada en la *blockchain* [14].

- *MiniApp*: aplicación externa, que se ejecutan dentro del sistema de *Worldcoin* y heredando todos los elementos de este sistema, como es la identificación de *World ID* [15].

- *World App*: aplicación móvil desde la cual los usuarios pueden interactuar con las *MiniApp* y demostrar su identidad sin revelar ningún dato sensible a las aplicaciones externas [16].

La elección de *Worldcoin* frente a otras tecnologías *blockchain* (como son: Solana [17], Binance Smart Chain [18], Cardano [19], entre otras) se justifica por su capacidad para integrar de manera nativa, eficiente y segura los siguientes elementos:

- *Privacidad*: mediante *ZKP* [20], que evita el almacenamiento de datos biométricos en servidores centrales y protege la identidad del usuario [21].

- *Descentralización*: el *World ID* puede verificarse sin necesidad de comunicarse constantemente con los servidores de *Worldcoin*, lo que reduce la dependencia de terceros [14].

- *Enfoque global y escalable*: el protocolo fue diseñado para operar en cualquier parte del Mundo, permitiendo a desarrolladores e instituciones integrarlo fácilmente.

- Documentación abierta, *Software Development Kit* (*SDK*) [15] funcional y herramientas de integración accesibles, lo que facilitó el desarrollo de una demo completa sin necesidad de *hardware* especializado como el *Orb*.

- *Base tecnológica sobre Ethereum*: lo que proporciona un entorno seguro, probado y compatible con estándares ampliamente adoptados en el sistema *Web3* [22].

Además, *Worldcoin* ofrece un sistema de registro mediante su propio dispositivo biométrico (*Orb*), el cual garantiza que los usuarios sean únicos y evitando la existencia de *bots* dentro del sistema. Esto representa una ventaja competitiva frente a otras soluciones de identidad digital, que a diferencia de otras plataformas *blockchain* que dependen exclusivamente del uso de *wallets* como *MetaMask* [23], muchas de las cuales no son intuitivas ni prácticas para el público general. En cambio, *Worldcoin* está enfocada para integrarse fácilmente en organismos como la ULPGC, y facilitando las herramientas necesarias a los desarrolladores para que puedan integrar aplicaciones dentro del sistema *Worldcoin* [15].

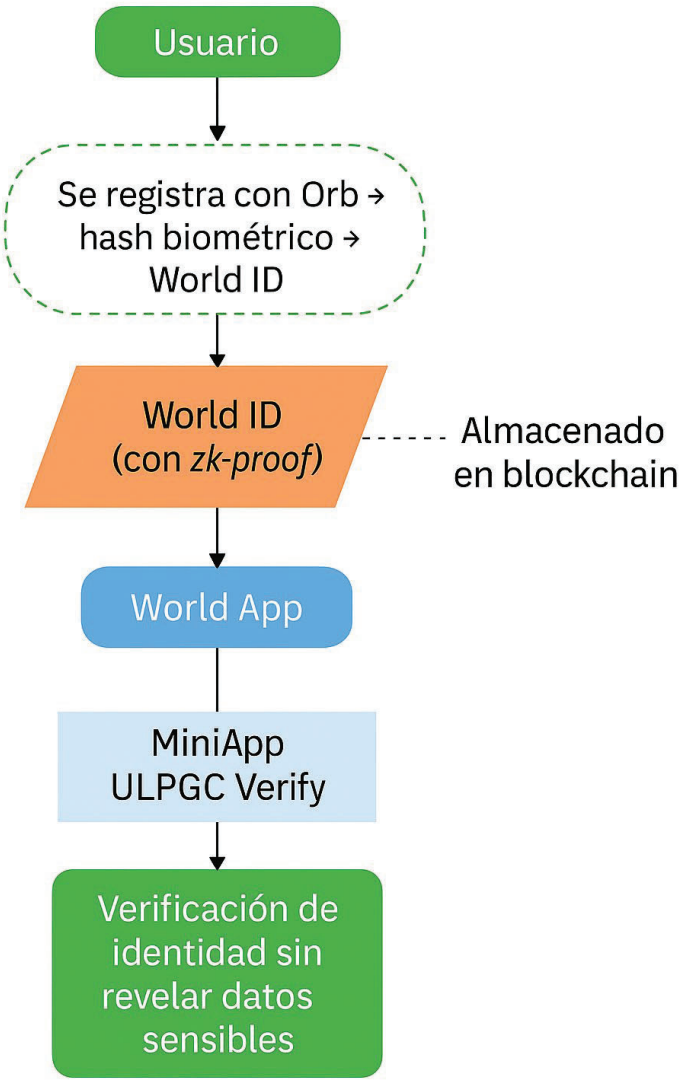

Figura 1.2 Diagrama de Flujo para la validación del usuario en el sistema Worldcoin

1.2.2 Arquitectura en niveles: de Ethereum a Worldcoin

Para comprender el funcionamiento técnico de *Worldcoin*, es fundamental situarlo dentro de la arquitectura de *Ethereum* y su modelo de ejecución basado en niveles. Inicialmente, sobre la red principal de *Ethereum Layer 1* (*L1*) [11] que es la plataforma de ejecución y validación de transacciones. Para solventar los problemas de escalabilidad y coste la *Ethereum Layer 2* (*L2*) [24] permite ejecutar operaciones eficiente y económicamente, sin comprometer la seguridad. En este TFG hemos elegido *OP Stack* [25], una infraestructura basada en *Optimistic Rollups* [26] que permite escalar aplicaciones sin congestionar la red principal.

Sobre *OP Stack* se despliega la aplicación descentralizada [27] *World* desarrollada por *Worldcoin*. Dentro de esta aplicación se organiza una infraestructura propia, que incluye su propia *blockchain* y un sistema de componentes con una arquitectura modular como se muestra en la *Figura 1.3*. *Worldcoin* ofrece un sistema de autenticación robusto, escalable y descentralizado, aprovechando la seguridad de *Ethereum*, la eficiencia del *OP Stack* y la flexibilidad de su propia arquitectura.

1.3 Objetivos

El propósito principal de este *TFG* es estudiar y explorar la viabilidad de la aplicación de la tecnología *Blockchain*, para garantizar la seguridad y privacidad de los datos en entornos comunes como pueden ser en los centros docentes universitarios, específicamente en lo que respecta a la gestión de identidad y acceso a sistemas de información sensibles. Se busca evaluar la idoneidad de la tecnología *Blockchain* para desarrollar un sistema de identificación y autenticación descentralizado que proteja la información de los usuarios.

El objetivo general de este TFG es analizar cómo la tecnología *Blockchain*, incluyendo la proporcionada por *World*, puede mejorar la seguridad y privacidad de los datos en centros docentes y otros contextos, reduciendo la dependencia de terceros para el almacenamiento y gestión de la información personal de los usuarios.

Para lograr este objetivo general, se han establecido los siguientes subobjetivos:

- O1. Comprender en profundidad el funcionamiento de la tecnología *Blockchain* en el contexto específico de la gestión de identidad y seguridad de datos.
- O2. Analizar la implementación de sistemas de gestión de identidad basados en *Blockchain* en diferentes ámbitos.
- O3. Evaluar la viabilidad del diseño e implementación de un sistema de gestión de identidad y acceso basado en *Blockchain* en diferentes entornos, considerando los protocolos y plataformas tecnológicas más adecuadas, incluyendo aquellos utilizados por *World*.

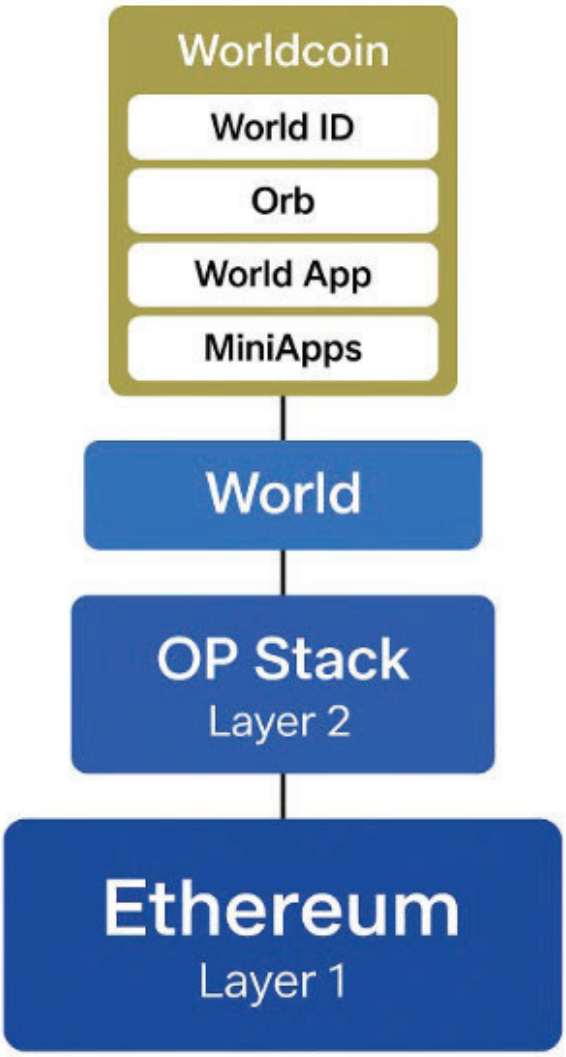

Figura 1.3 Arquitectura del sistema de Worldcoin, construido sobre Ethereum y OP Stack

1.4 Estructura de la memoria

En esta sección presentamos la estructura de la memoria de este TFG. Se aborda el estudio y aplicación de la tecnología *Blockchain* para la gestión de identidades digitales y el control de acceso en entornos institucionales.

En el capítulo 2, se realiza un análisis detallado de los métodos de autenticación convencionales empleados en entornos institucionales como universidades, hospitales y administraciones públicas. Se describen los riesgos y limitaciones de estos métodos, así como los problemas derivados de su implementación. En el capítulo 3, se analiza los conceptos básicos de la tecnología *blockchain*, mecanismos de consenso y la capacidad para gestionar identidades digitales de manera segura y descentralizada. En estos dos capítulos se aborda el objetivo 1 (*O1*).

En el capítulo 3, se realiza el estudio de *Worldcoin* como un sistema de gestión de identidad digital. Se detalla su arquitectura basada en *Ethereum*, sus soluciones de escalabilidad y los componentes principales.

En el capítulo 4, se describen las herramientas tecnológicas empleadas para desarrollar ULPGC Verify, la aplicación descentralizada diseñada para este TFG.

En conjunto, estos dos últimos capítulos engloban el objetivo 2 (*O2*).

En el capítulo 5, se lleva a cabo el desarrollo e implementación práctica de ULPGC Verify, describiendo en detalle su diseño, arquitectura y funcionamiento. Se presentan los resultados obtenidos tras la simulación, evaluando su rendimiento y capacidad para ofrecer un método seguro y descentralizado de autenticación en entornos académicos. Este capítulo abarca el objetivo 3 (*O3*).

Los anexos complementan el estudio principal, con información técnica detallada sobre conceptos relacionados con *Worldcoin*. En el Anexo A se describe el funcionamiento del dispositivo biométrico *Orb* y su papel en la verificación de identidad digital. En el Anexo B se presentan los conceptos biométricos que utiliza el sistema *Worldcoin*, que facilita la comprensión del TFG en su totalidad.

2. Métodos de autenticación: aplicación a ULPGC Verify

En este capítulo se analiza el estado actual de los principales métodos de autenticación utilizados en los sistemas digitales, con el objetivo de identificar sus fortalezas, limitaciones y vulnerabilidades. Abordamos las soluciones tradicionales y tecnologías más recientes como la biometría, los sistemas federados y la introducción de la autenticación basada en *World ID* como alternativa descentralizada. Se revisa los métodos de autenticación utilizados en la ULPGC en la sede electrónica, el correo electrónico, el servidor de información principal y el acceso a la *Wireless Fidelity* (*WiFi*) y se enuncian las ventajas de usar World ID en la ULPGC.

2.1 Basada en contraseñas

La autenticación mediante usuario y contraseña es uno de los métodos más usados y de los primeros en establecer un control de acceso a un sistema informático. Utiliza un método muy sencillo: primero pide al usuario un identificador, que puede ser un nombre de usuario o un correo eléctrico y después le solicita una contraseña previamente establecida, que debe coincidir con la almacenada en el sistema para obtener el acceso [28]. De forma técnica, la contraseña no debe almacenarse en el servidor sin procesarse, debido a los riegos de seguridad que esto puede implicar, siempre se recomienda utilizar funciones criptográficas de *hashing*, como *SHA-256* [29], que transforman la contraseña en una cadena de caracteres codificada.

Para aumentar la seguridad del *hash*, se utiliza una técnica llamada *salting*, que consiste en añadir un valor aleatorio (*salt*) a cada contraseña antes de aplicar la función de *hash*. De esta manera se evita que dos contraseñas iguales puedan generar el mismo *hash* y protegerse contra ataques mediante tablas arcoíris [30].

Este método sigue siendo muy utilizado hoy en día debido a su simplicidad, ya que no requiere una infraestructura compleja. Es ideal para aplicaciones *web*, servicios en línea y sistema de bajo riesgo, pero tiene una gran desventaja, ya que depende del usuario final y muchas veces utilizan contraseñas débiles, predecibles o reutilizadas, lo que los expone a ataques como:

- *Credential stuffing*: reutilización de contraseñas filtradas de otros servicios.
- *Phishing*: técnicas de ingeniería social para robar credenciales.
- *Fuerza bruta*: prueba sistemática de contraseñas comunes hasta encontrar la correcta.

Como resultado, aunque la autenticación por contraseña sigue siendo un método muy utilizando hoy en día, su uso aislado ya no se considera suficiente para proteger sistemas sensibles. En el caso de la ULPGC, utilizan este método de autenticación para acceder a servicios digitales como el campus virtual, donde se solicita al estudiante su *Documento*

Nacional de Identidad (DNI) sin letra como identificador y una contraseña establecida previamente, permitiendo acceso a MiULPGC [31].

2.2 Método de sesiones

La autenticación basada en sesiones (*cookies*) es uno de los métodos más antiguos y utilizados en aplicaciones *web*. Consiste en validar las credenciales del usuario, normalmente mediante el método de autenticación usuario y contraseña. Una vez verificadas estas credenciales, el servidor crea una sesión para ese usuario y envía un identificador de sesión al navegador en forma de *cookie*, la cual se utiliza para cada solicitud posterior para mantener al usuario autenticado [32].

Este sistema se basa en dos componentes clave:

- *Cookies*: es donde almacenan en el cliente el identificador de la sesión.
- *Sesiones*: se almacenan en el servidor la información del usuario autenticado.

En la *ULPGC*, este sistema se emplea para el acceso al *Campus Virtual*, donde los usuarios deben ingresar su *DNI* sin letra como nombre de usuario y una contraseña personal a través de un formulario *web*. Después de autenticarse de forma correcta, el servidor mantiene activa la sesión mediante las *cookies*, permitiendo al estudiante navegar sin tener que volver a autenticarse en cada acción [33].

Entre las ventajas de este método están:

- *Simplicidad*: fácil de implementar en servidores *web* y ampliamente compatible con tecnologías *frontend* y *backend*.
- *Persistencia de sesión*: permite mantener al usuario autenticado durante toda la sesión sin requerir nuevas validaciones.
- *Compatibilidad*: ampliamente soportado por navegadores y lenguajes de programación *web*.

Entre las desventajas y desafíos de este método están:

- *Vulnerabilidad*: si no se protege correctamente, las cookies pueden ser robadas o reutilizadas por atacantes [34].
- *Gestión de sesiones*: requiere mecanismos para almacenar y limpiar sesiones activas, lo que implica uso de recursos.
- *No es escalable por sí solo*: en sistemas distribuidos se necesitan mecanismos adicionales para sincronizar sesiones entre múltiples servidores.

2.3 Uso de tokens

La autenticación basada en *tokens* es una evolución de los sistemas de autenticación por sesiones. Es ampliamente utilizada en arquitecturas *web* modernas y especialmente útil en entornos de microservicios, *Application Programming Interface* (*API*) o aplicaciones móviles. El usuario, una vez autenticado de manera correcta, recibe un *token* digital firmado que actúa como una prueba de identidad. Este sistema permite que el estado de autenticación se externalice como un objeto portable, eliminando la necesidad de mantener sesiones con el servidor [35]. Los estándares más utilizados para implementar este sistema son el *JSON Web Token* (*JWT*) [36], el cual contiene tres partes codificadas en base64, que son:

- *Header*: indica al algoritmo de firma (como *HMAC-SHA256*).
- *Payload*: contiene los datos o *claims*, como el *ID* del usuario, su categoría o el tiempo de expiración.
- *Signature*: firma digital que asegura la integridad del *token*. Se genera usando una clave secreta o una clave privada según el método utilizado [37].

Los *JWT* pueden ser autocontenidos, contienen toda la información necesaria para poder validar la autenticación sin necesidad de consultar una base de datos, ideal para

aplicaciones *stateless*, en las que la escalabilidad y la independencia del *backend* son esenciales [38].

El proceso funciona de la siguiente manera:

- El usuario se autentica.
- El servidor valida las credenciales y genera un *JWT* firmado.
- El *JWT* se envía al cliente, que lo almacena en el navegador, que puede ser un *localStorage*.
- En cada petición futura, el cliente adjunta el token en la cabecera *HTTP Authorization*: *Bearer* <*token*>.
- El servidor verifica la firma del *token* y permite el acceso en caso de que sea válido.

Este método presenta ventajas significativas, al no depender de un servidor para almacenar la sesión. El sistema puede ser distribuido fácilmente en múltiples servidores sin replicar el estado. Los *tokens* pueden incluir información útil, como permisos y vigencia temporal, permitiendo que el servidor pueda aplicar políticas de control de acceso sin necesidad de hacer múltiples consultas a la base de datos [36]. No obstante, este modelo presenta algunos riesgos, ya que los tokens al ser portables y autocontenidos pueden ser robados si no se almacenan de manera segura.

La autenticación mediante tokens se ha vuelto especialmente utilizado en entornos modernos, como:

- *La API* RESTful
- Aplicaciones móviles.
- Sistemas de autenticación federada (*OAuth2* / *OpenID Connect*).
- Microservicios, donde cada servicio puede verificar la autenticidad del token sin depender de un sistema de sesiones centralizado.

2.4 Técnica Multi-Factor

La técnica *MFA* refuerza la autenticación del usuario mediante la combinación de dos o más métodos de verificación, provenientes de diferentes categorías. Por ejemplo, una contraseña, un número de teléfono para recibir un código vía mensaje de texto y la autenticación por biometría, consiguiendo reducir el riesgo de accesos no autorizados, incluso si una de las credenciales ha sido comprometida. La *MFA* más utilizada es la *Two-Factor Authentication* (*2FA*), que implica dos de las categorías mencionadas anteriormente, el más utilizado es el uso de una contraseña como primer factor, junto con un código temporal enviado por *Short Message Service* (*SMS*), generado por una *Application* (*App*) de autenticación como segundo factor.

El sistema *MFA* funciona de la siguiente manera:

- El usuario debe introducir sus credenciales para completar el primer factor (nombre de usuario y contraseña).
- Si las credenciales son válidas, el sistema solicitaría un segundo factor de autenticación.
- Este segundo factor podría generarse o validarse de varias formas:
    - Un código *Time-based One-Time Password* (*TOTP*) generado por una App.
    - Un código recibido por SMS o correo electrónico.
    - Una notificación push en el teléfono móvil.
    - La inserción de un token físico.
    - El uso de una huella dactilar, reconocimiento facial o escaneo de retina.
- Si el segundo factor también es verificado con éxito, el usuario podría acceder al sistema [39].

La *MFA* supera las limitaciones del método de autenticación basado en contraseñas, reduciendo de manera significativa el riesgo de acceso no autorizado y aumentar la seguridad. Esto es debido a que se refuerza con niveles (factores) de seguridad, exigiendo una prueba más difícil de replicar que en el caso de la técnica de usuario y contraseña (que estas pueden ser conocidas por terceros sin que el sistema lo detecte). Por ello, su uso se hace en sistemas críticos como son: la banca electrónica, servicios de salud, sistemas gubernamentales y plataformas en la nube, así como en servicios de consumo masivo como Google, Microsoft o Amazon [40].

En el caso de la ULPGC, este método de autenticación de doble factor se utiliza para el acceso al correo electrónico institucional, en el que los usuarios deben confirmar su identidad utilizando una combinación de contraseña y un código temporal, ayudando aumentar la seguridad de los usuarios que utilizan el correo institucional ante accesos no autorizados [41]. Entre las ventajas de usar este método de autenticación, es la reducción de ataques automatizados como el *credential stuffing* [42] y la disminución de la eficacia de técnicas de *phishing* [43]. Aunque la necesidad de múltiples pasos resulta incómoda para algunos usuarios, y la dependencia de dispositivos externos puede dificultar el acceso en caso de pérdida. También se podría vulnerar ciertos métodos de *MFA*, como ocurre con el *SIM swapping* [44], aparte tendrías que aportar tu información personal, como el número de teléfono a un tercer agente.

Tipos de factores usados en *MFA*

- *Conocimiento*: algo que el usuario sabe. Por ejemplo, contraseñas, *Personal Identification Number* (*PIN*), preguntas secretas.
- *Posesión*: algo que el usuario tiene. Por ejemplo, tokens físicos, dispositivos móviles, tarjetas inteligentes.
- *Cualidades personales*: algo que el usuario es. Por ejemplo, biometría (huella, rostro, iris, voz, etc.) [39].

2.5 Biométrica

La autenticación biométrica se basa en el reconocimiento y verificación de características físicas únicas de un individuo, como puede ser la huella dactilar, el rostro, el iris, la voz o la geometría de la mano, en lugar de depender de una contraseña. Este sistema se fundamenta en algo identitario del usuario. Es un método de autenticación de nivel alto porque reduce la dependencia de elementos externos y aprovecha las características que son intrínsecas e intransferibles del usuario [45].

A nivel funcional, un sistema de autenticación biométrica consta de dos fases principales:

- *Fase de registro*: donde se capturan y almacena la muestra biométrica del usuario en una base de datos o en un dispositivo seguro como plantilla codificada.

- *Fase de autenticación*: se captura una nueva muestra biométrica y se compara contra la plantilla previamente registrada para validar su identidad.

En esas fases se aplican algoritmos de procesado de señal, extracción de características y comparación con umbrales definidos, utilizando técnicas estadísticas y de aprendizaje automático para reducir la probabilidad de falsos positivos o negativos. Uno de los principales beneficios de este método es la comodidad, ya que elimina la necesidad de recordar contraseñas o portar dispositivos físicos. Se integra ampliamente en dispositivos móviles, sistemas de control de acceso en edificios, puntos de control fronterizo, servicios bancarios y aplicaciones sanitarias. Por ejemplo, muchos de los teléfonos móviles y portátiles utilizan sensores biométricos como *Face ID* [46] o lectores de huellas digitales para desbloquear el dispositivo o autorizar pagos.

Entre las ventajas de este método están:

- *Alta conveniencia*: los usuarios no necesitan recordar contraseñas ni llevar dispositivos físicos.

- *Difícil de falsificar*: las características biométricas son únicas y persistentes en el tiempo.

- *Rápida verificación*: el proceso es casi instantáneo en sistemas modernos [47].

Entre las desventajas y desafíos de este método están:

- *Privacidad y almacenamiento*: si las plantillas biométricas son comprometidas, no pueden ser cambiadas como una contraseña.

- *Falsos rechazos (FAR) y falsos accesos (FRR)*: los factores como el entorno, la calidad del sensor o alteraciones físicas pueden afectar los resultados en la lectura de características físicas de un individuo.

- *Coste del hardware*: aunque ha disminuido en los últimos años, algunos sistemas requieren sensores especializados.

- *Problemas éticos y de inclusión*: no todos los usuarios pueden proporcionar datos biométricos confiables, por ejemplo, personas con discapacidades o lesiones [47].

2.6 Empleo de certificado digital

La autenticación mediante certificado digital, también conocida como autenticación basada en *Public Key Infrastructure* (*PKI*), es un sistema de autenticación que se basa en criptografía asimétrica para verificar la identidad de usuarios, dispositivos o servicios, que es ampliamente utilizada especialmente en conexiones *HyperText Transfer Protocol Secure* (*HTTPS*), firmas digitales, acceso a redes empresariales y envío seguro de correos electrónicos [48].

El funcionamiento parte de la *PKI* con el uso de dos claves criptográficas, que son la clave privada, que se mantiene en secreto, y una clave pública, que puede ser compartida abiertamente. Estas claves están matemáticamente relacionadas, de modo que la información cifrada con una de ellas solo puede ser descifrada con la otra. El certificado

digital de un archivo electrónico que vincula la clave pública con la identidad del propietario. Este certificado es emitido por una entidad de confianza llamada *Autoridad Certificadora* (*CA*), que actúa como un notario digital validando que una clave pública pertenece a una persona, organización o dispositivo [49]. El proceso de autenticación funciona de la siguiente manera:

- Un usuario o cliente presenta su certificado digital al servidor o sistema al que quiere acceder.
- El sistema verifica la validez del certificado y que haya sido emitido por una *CA* confiable.
- Se realiza un reto criptográfico, donde el servidor envía un mensaje cifrado con la clave pública del usuario, y este debe descifrarlo con su clave privada para demostrar que es el propietario legítimo del certificado.
- Si el proceso se completa correctamente, se concede el acceso [50].

Este método es ampliamente utilizado en entornos corporativos, gubernamentales y académicos donde se necesita un alto nivel de seguridad, como en el acceso a *Virtual Private Network* (*VPN*) [51], correo electrónico cifrado, acceso a plataformas administrativas o servicios en la nube. Esto es común en la autenticación de dispositivos y en protocolos como *Secure Sockets Layer* (*SSL*) o *Transport Layer Security* (*TLS*) para proteger las conexiones *web HTTPS*. En el caso de la ULPGC [52], este método se aplica en el acceso a su Sede Electrónica y los usuarios pueden autenticarse mediante un certificado digital reconocido, como el del *Documento Nacional de Identidad Electrónico* (*DNIe*) o los emitidos por la *Fábrica Nacional de Moneda y Timbre* (*FNMT*), para firmar digitalmente documentos o realizar gestiones académicas y administrativas con plena validez legal y máxima seguridad.

Entre las ventajas de este método están:

- *Seguridad criptográfica*: la autenticación está basada en claves públicas y privadas, lo cual es muy difícil de vulnerar sin acceso a la clave privada.

- *No requiere contraseñas*: elimina los riesgos asociados al uso de contraseñas débiles o reutilizadas.
- *Escalabilidad*: es ideal para entornos con miles de usuarios o dispositivos.
- *Firma digital*: los certificados emitidos por una *CA* reconocida pueden tener validez legal [53].

Entre las desventajas y desafíos de este método están:

- *Gestión compleja*: requiere una infraestructura robusta para emitir, renovar, revocar y almacenar certificados de forma segura.
- *Riesgo si se compromete la clave privada*: si un atacante obtiene acceso a la clave privada de un usuario, puede suplantarlo completamente.
- *Dependencia de terceros*: la confianza depende de la *CA*, en caso de que esta es comprometida, se pone en riesgo toda la cadena de confianza.
- *Requerimiento de dispositivos seguros*: para almacenar la clave privada, como *tokens Universal Serial Bus* (*USB*), tarjetas inteligentes o módulos *Trusted Platform Module* (*TPM*)[53].

2.7 Aplicación de dispositivos de usuario

La autenticación basada en dispositivos es una técnica moderna y segura que elimina el uso de contraseñas tradicionales, sustituyéndolas por claves criptográficas almacenadas localmente en un dispositivo del usuario. Se apoya principalmente en los estándares *Fast Identity Online 2* (*FIDO2*) [54] y *WebAuthn* [55]. Desarrollado por la *FIDO Alliance* y el *World Wide Web Consortium* (*W3C*) [54] [56]. Utiliza criptografía de clave pública que se registra en el servidor y una clave privada que permanece protegida dentro del dispositivo del usuario. La autenticación se realiza mediante la firma de un reto criptográfico (*challenge*), que garantiza que la persona que intenta iniciar sesión es realmente quien dice ser [56].

El dispositivo se podría convertir en un *token* de *hardware* inteligente y utilizase como *MFA*: a) lo que tiene el usuario (dispositivo) y lo que sólo él sabe (su biometría o un PIN). Esto permite la autenticación sin contraseñas (*Passwordless*), que es altamente resistente a ataques de *phishing* y robo de credenciales. Este método es ampliamente utilizado en servicios de acceso a cuentas críticas como *Google*, *Microsoft* o *GitHub*, así como en entornos corporativos para autenticarse en sistemas, *VPN* o escritorios virtuales [57].

Entre las ventajas de este método están:

- *Elevado grado de seguridad*: al requerir presencia física del dispositivo y validación local, este sistema tiene una alta resistencia a los ataques de *phishing*, *man-in-the-middle* y la reutilización de credenciales.

- *Passwordless*: elimina la necesidad de recordar contraseñas, mejorando la experiencia del usuario y reduciendo riesgos asociados a contraseñas débiles.

- *Portabilidad*: el usuario puede utilizar su dispositivo para autenticarse en múltiples plataformas compatibles.

- *Resistencia al phishing*: el desafío criptográfico está ligado al dominio *web*, evitando ataques mediante sitios fraudulentos.

Entre las desventajas y desafíos de este método están:

- *Dependencia del dispositivo*: si se pierde o daña el dispositivo de autenticación, puede ser necesario un método de recuperación alternativo.

- *Costo inicial*: requiere inversión en hardware como llaves de seguridad o dispositivos compatibles con autenticación biométrica.

- *Implementación técnica*: integrar correctamente *FIDO2*/*WebAuthn* puede implicar cierta complejidad y requiere compatibilidad en los navegadores y en las distintas plataformas.

2.8 Federada para acceso a múltiples servicios

La autenticación federada es un método de autenticación que permite a los usuarios acceder a múltiples servicios digitales. Utiliza una única identidad proporcionada por un tercero confiable, conocido como *Identity Provider* (*IdP*). En lugar de que cada aplicación tenga que autenticar al usuario por separado, estas asignan el proceso de autenticación al *IdP*, lo que permite hacer un único inicio de sesión o un *Single Sign-On* (*SSO*) [58]. Cuando la autenticación es exitosa, el *IdP* genera una afirmación, que generalmente en formato *Security Assertion Markup Language* (*SAML*) o *JWT*, que contiene la información del usuario y la envía al Proveedor de Servicios (*SP*), quien confía en dicha afirmación para conceder el acceso [59].

En la ULPGC, se utiliza la autenticación federada a través del servicio *Education Roaming* (*eduroam*) [60], utilizada por una red de instituciones educativas. Eduroam permite a los estudiantes, docentes e investigadores que puedan utilizar su identidad digital institucional para poder conectarse a redes *WiFi* de otras universidades sin necesidad de crear una nueva cuenta. Este sistema se basa en una infraestructura jerárquica de servidores *Remote Authentication Dial-In User Servic* (*RADIUS*) [61], que son los encargados de enrutar las solicitudes de autenticación desde la institución anfitriona hasta la universidad de origen del usuario. Cada validación es realizada mediante el protocolo *802.1X* [62] y combinado con el *Protocolo de Autenticación Extensible* (*EAP*), conforme a lo establecido en la *RFC 7593* [63].

Los tres estándares más utilizados son:

- *OAuth 2.0*: protocolo de autorización que permite a las aplicaciones obtener acceso limitado a los recursos protegidos en nombre de un usuario [37].
- *OpenID Connect* (*OIDC*): protocolo de autenticación construido sobre *OAuth 2.0* que permite obtener información de identidad del usuario [64].
- *SAML*: estándar basado en *Extensible Markup Language* (*XML*) que permite intercambiar información de autenticación y autorización entre diferentes dominios [64].

Entre las ventajas de este método están:

- *Administración centralizada*: permite gestionar permisos, accesos y políticas de seguridad desde un único lugar.
- *Compatibilidad con MFA*: muchos proveedores de identidad permiten aplicar *MFA* sin que el *Service Provider* tenga que implementarlo.
- *Escalabilidad*: Es ideal para entornos con múltiples aplicaciones, ya que evita replicar mecanismos de autenticación en cada una [59].

Entre las desventajas y desafíos de este método están:

- *Dependencia de terceros*: si el *IdP* falla o es comprometido, todos los servicios que dependen de él pueden verse afectados.
- *Complejidad en la implementación*: configurar correctamente *OAuth*, *OIDC* o *SAML* requiere conocimiento técnico y una configuración segura.
- *Privacidad*: El *IdP* puede tener una visibilidad sobre qué servicios utiliza el usuario, lo cual puede generar preocupaciones.

2.9 World ID

World ID, desarrollado por *Worldcoin*, es un sistema de autenticación basado en pruebas de identidad única (*Proof of Personhood*) [65]. Tiene como objetivo proporcionar una forma de demostrar que un individuo es una persona real y única, sin necesidad de compartir datos personales, mediante la combinación de técnicas criptografía avanzada y dispositivos biométricos.

El sistema forma parte de una visión más amplia donde la verificación de *DID* y sin custodia puede reemplazar o complementar los métodos de autenticación actuales como contraseñas, *tokens*, certificados, biometría centralizada, entre otros. Permitiendo un

acceso más seguro, privado y descentralizado a los servicios digitales que estén integrados a este sistema [14].

El sistema se basa en tres componentes esenciales:

- *Dispositivo Orb*: escanea la biométrica de los usuarios, captura una imagen del iris del usuario para generar un *hash* cifrado único. Con este código no se almacena la imagen del iris, sino que se representa matemáticamente su unicidad (Anexo A).

- *Registro descentralizado*: el *hash* generado se compara con otros ya existentes para asegurar que no exista duplicidad, en caso de que el hash es único, el usuario obtiene un *World ID* vinculado a una *wallet Web3*, permitiendo su uso como sistema de autenticación.

- *Verificación mediante ZKP*: cuando el usuario necesita autenticarse, el sistema genera una prueba de conocimiento cero que valida en caso de que posea un *World ID* sin revelar su identidad ni sus datos biométricos originales [66].

Entre las ventajas de este método están:

- *Unicidad:* impide el registro múltiple de una misma persona, eliminando fraudes de identidad o suplantaciones, creando una red libre de *bots*.

- *Descentralización:* no depende de una autoridad central para validar la identidad, ya que el control reside en el usuario.

- *Privacidad y anonimato:* no se requiere compartir datos personales, ya que la verificación se realiza mediante hash y pruebas *Zero-Knowledge* (*ZK*).

- *Escalabilidad:* compatible con infraestructuras *Web3*.

- *Accesibilidad global:* permite autenticación confiable a personas sin necesidad de tener documentos oficiales o cuentas en instituciones.

Entre las desventajas y desafíos de este método están:

- *Dependencia de hardware:* para registrarse por primera vez es obligatorio el uso del Orb, el cual no está ampliamente disponible en todas las regiones, incluyendo a la ULPGC.

- *Requiere adaptación en sistemas tradicionales*: plataformas como el Campus Virtual o los sistemas de secretaría deben ser compatibles con el sistema para integrarlo completamente.

- *Limitaciones de adopción*: dado que es una tecnología emergente, aún no se ha implementado masivamente en sectores tradicionales como el académico.

La solución que puede aportar *World ID* respecto a los métodos actuales que usa el sistema de autenticación de la *ULPGC* son:

- *Contraseñas*: *World ID* eliminaría el uso de credenciales para acceder a MiULPGC, que son vulnerables a ataques de *phishing*, fuerza bruta o reutilización.

- *Correo institucional*: aunque la ULPGC implementa *MFA* como refuerzo de seguridad, *World ID* proporciona un segundo factor integrado, que sería la biometría que no depende de *SMS*, *App* o códigos temporales. Así reduciría la exposición a técnicas como *SIM swapping* o interceptación de códigos. Además, los usuarios no tendrían la necesidad de aportar información a terceros (*Microsoft*).

- *Sede electrónica*: *World ID* elimina la necesidad de instalar y mantener certificados en el navegador. La identidad de los usuarios se almacena en una wallet digital vinculada al hash biométrico correspondiente al usuario, lo cual podría servir como alternativa segura, verificable y portable.

- *Facilidad de acceso multiservicio*: World ID permitiría al alumnado y al personal autenticarse en todos los servicios de la *ULPGC* con una sola verificación, sin necesidad de múltiples inicios de sesión ni recordar diferentes credenciales.

2.10 Tabla comparativa de métodos de autenticación

En la *Tabla 2.1* se muestra una comparación entre los distintos métodos definidos anteriormente. Permitiendo extraer varias conclusiones relevantes, de esta manera se observa que los métodos tradicionales como las *contraseñas*, *formularios* y *sesiones* son utilizados para el acceso del *Campus Virtual*, cuando son los métodos más vulnerables en términos de seguridad. En particular, las *contraseñas* presentan una alta dependencia del usuario y poca resistencia al *phishing*.

Los métodos como la *MFA* o los *certificados digitales*, utilizados en el acceso al correo institucional y la sede electrónica respectivamente, mostrando un nivel de seguridad significativamente mayor. No obstante, siguen requiriendo intervención del usuario, como el uso de claves temporales o certificados instalados en el navegador.

En cuanto a la biometría y los dispositivos basados en *FIDO2*/*WebAuthn*, a pesar de ser sistemas altamente seguros y resistentes al *phishing*, no están implementados actualmente en la ULPGC. Lo mismo ocurre con la autenticación de *tokens*, es ampliamente usada en aplicaciones modernas, pero no se ha identificado como método activo en los servicios actuales de la universidad.

La autenticación federada, aunque no está implementada como método de acceso en plataformas internas, sí se utiliza en el acceso a *eduroam*, permitiendo autenticación entre instituciones. Finalmente, *World ID* se posiciona como un método emergente con un enfoque descentralizado, con una alta seguridad y resistencia al *phishing*.

2.11 Reflexiones sobre ULPGC Verify

A continuación, se detallan los principales sistemas utilizados, acompañados de una figura representativa de cada uno:

- *Usuario y contraseña*: utilizado principalmente en el acceso a *MiULPGC*. El estudiante debe introducir su DNI sin letra junto con una contraseña personal, como se muestra en la *Figura 2.1.*

Método de Autenticación	Requiere Infraestructura	Nivel de Seguridad	Dependencia de Usuario	Resistencia al *Phishing*	Uso en la ULPGC
Contraseña	No	Bajo	Alta	Baja	Acceso al Campus Virtual
Tokens	Sí	Medio	Media	Media	No implementado actualmente
Certificados Digitales	Sí	Alto	Baja	Alta	Sede Electrónica
Biométrica	Sí	Alto	Media	Alta	No implementado actualmente
MFA	Sí	Alto	Media	Alta	Correo Institucional
Formularios y Sesiones	No	Medio	Alta	Baja	Campus Virtual
Autenticación Federada	Sí	Alto	Media	Alta	eduroam
Dispositivos (*FIDO2/WebAuthn*)	Sí	Muy Alto	Baja	Muy Alta	No implementado actualmente
World ID	Sí	Muy Alto	Baja	Muy Alta	No implementado actualmente

Tabla 2.1 Comparativa de los métodos de autenticación

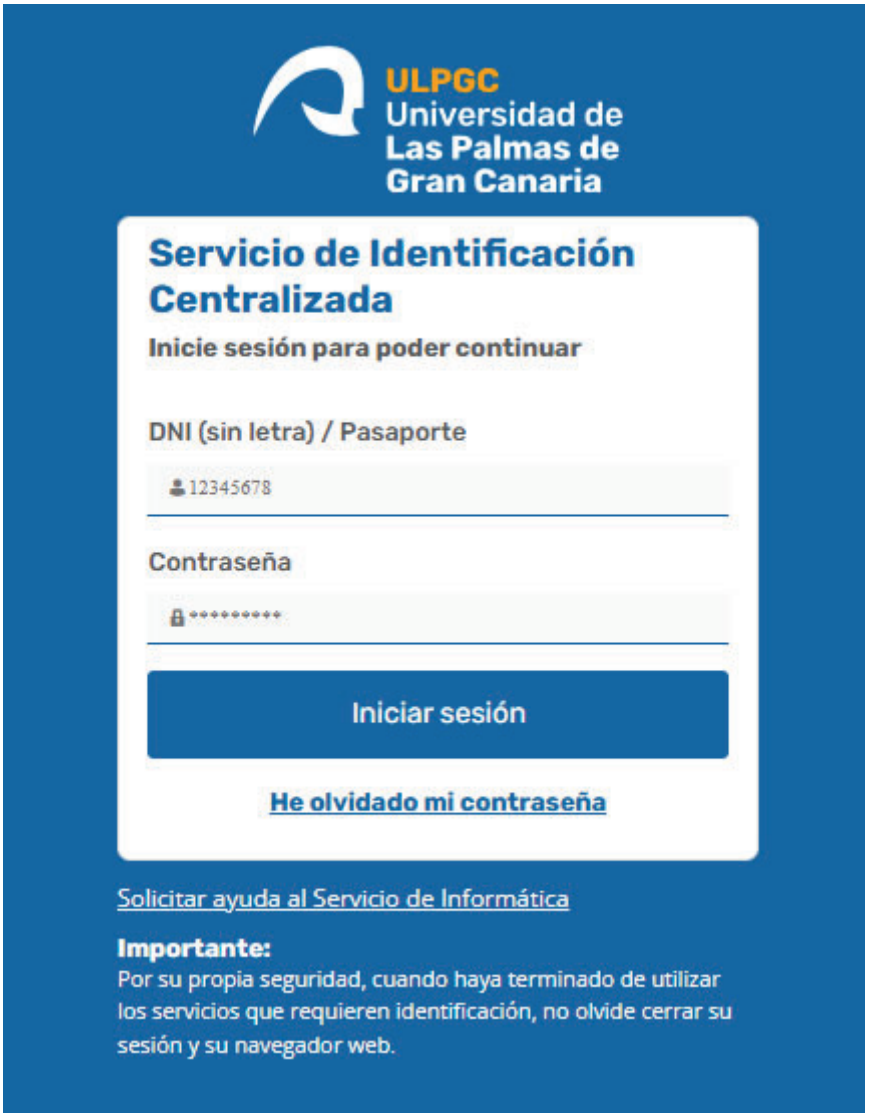

Figura 2.1 Pantalla de acceso a MiULPGC con usuario y contraseña

- *MFA*: aplicada en el acceso al correo institucional de la ULPGC. El usuario introduce su contraseña y debe confirmar su identidad mediante un código temporal generado por una *App* o enviado por *SMS*, como se muestra en la *Figura 2.2*.

- *Certificados digitales*: utilizados en la Sede Electrónica de la ULPGC. Permiten iniciar sesión en la Sede Electrónica, firmar digitalmente y realizar trámites administrativos mediante certificados como el del *DNIe* o los emitidos por la *FNMT*, como se muestra en la *Figura 2.3*.

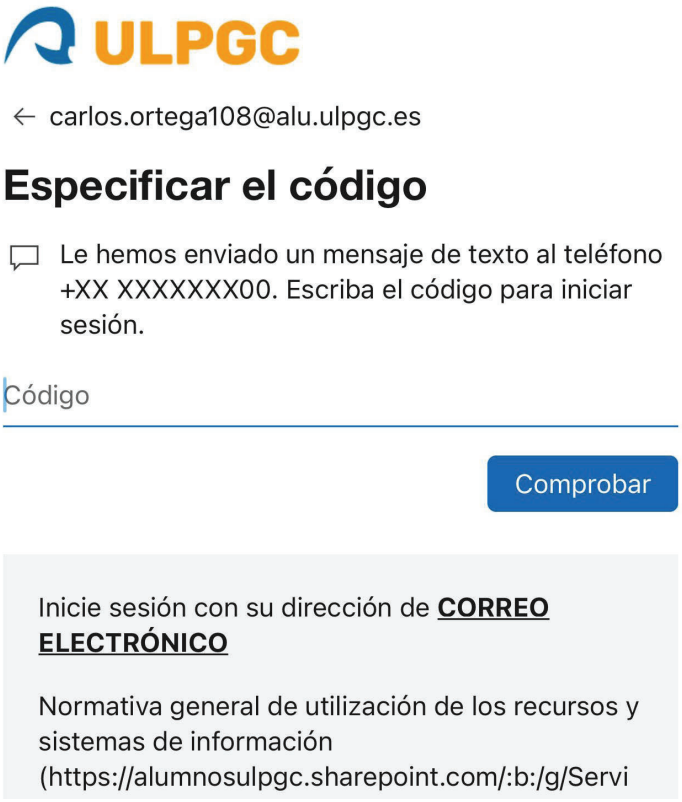

Figura 2.2: Interfaz de verificación MFA para confirmar la identidad mediante un código temporal por SMS

Figura 2.3 Acceso a la Sede Electrónica mediante certificado digital

- *Autenticación basada en sesiones:* implementada en varias plataformas *web* internas. Tras introducir las credenciales, el sistema mantiene la sesión activa mediante cookies, como se muestra en la Figura 2.4. Se implementada en la plataforma del portal MiULPGC. Tras introducir las credenciales en un formulario *web*, el sistema valida el acceso y mantiene la sesión activa mediante una *cookie* de sesión, generalmente identificada como *PHPSESSID*, tal como se muestra en la *Figura 2.4*. Esta cookie es almacenada en el navegador y enviada automáticamente en cada solicitud posterior para preservar el estado autenticado del usuario mientras navega por el sistema.

- *Autenticación federada (eduroam):* este método permite que el usuario se conecte utilizando con sus credenciales institucionales, como se muestra en la *Figura 2.5.*

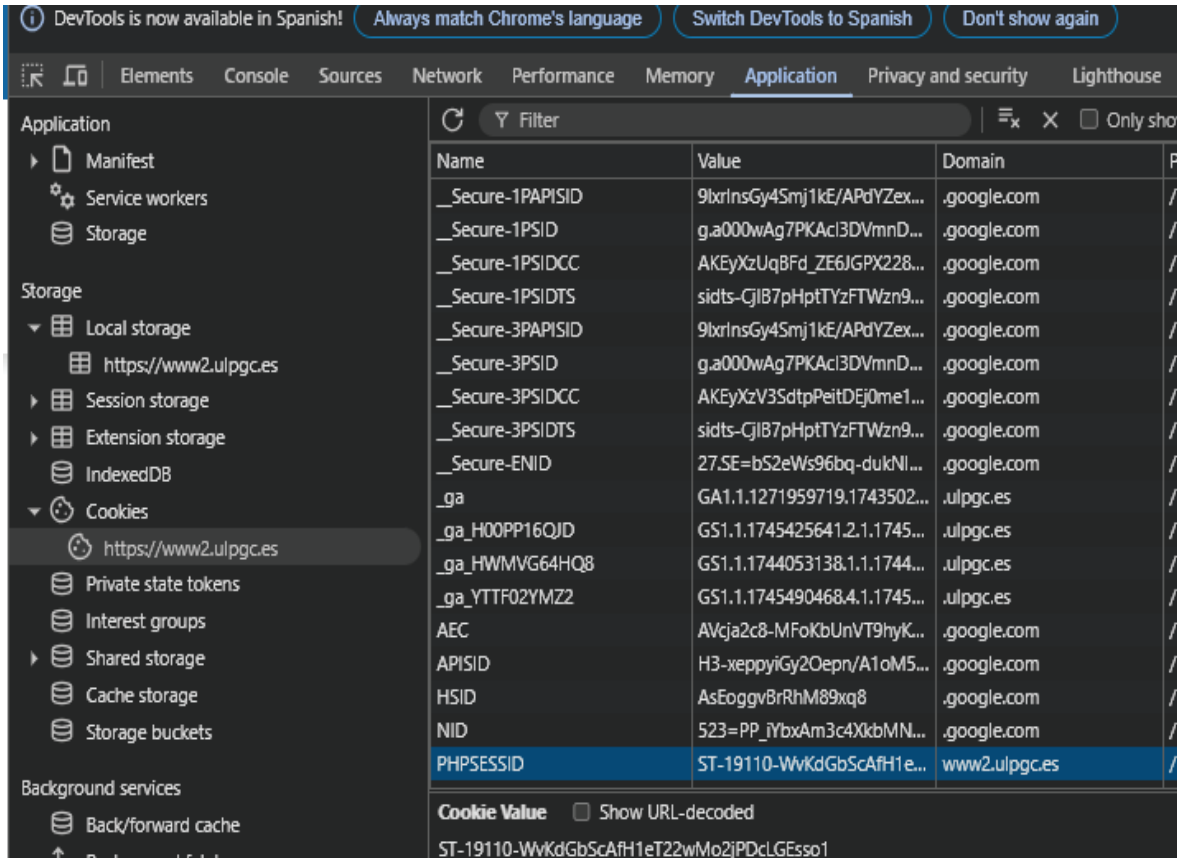

Figura 2.4 Cookie de sesión PHPSESSID en el portal MiULPGC

El principal objetivo que perseguimos con el diseño de *ULPGC Verify* es analizar si el sistema *World ID* puede ser útil para simular el acceso a estos métodos de autenticación tradicionales mediante una única verificación biométrica y descentralizada. La intención es ofrecer una alternativa moderna que simplifique el acceso a servicios digitales universitarios, manteniendo altos niveles de seguridad y privacidad.

Sin embargo, tras estudiar los requisitos técnicos de cada sistema, se concluye que *World ID* no puede sustituir directamente a ninguno de los métodos implementados actualmente. Esto se debe a que cada uno de ellos depende de componentes e infraestructuras muy específicas, como certificados emitidos por autoridades nacionales (*FNMT*, *DNIe*), integración con sistemas de autenticación federada o servicios externos para *MFA* (*Microsoft Authenticator* o *SMS*).

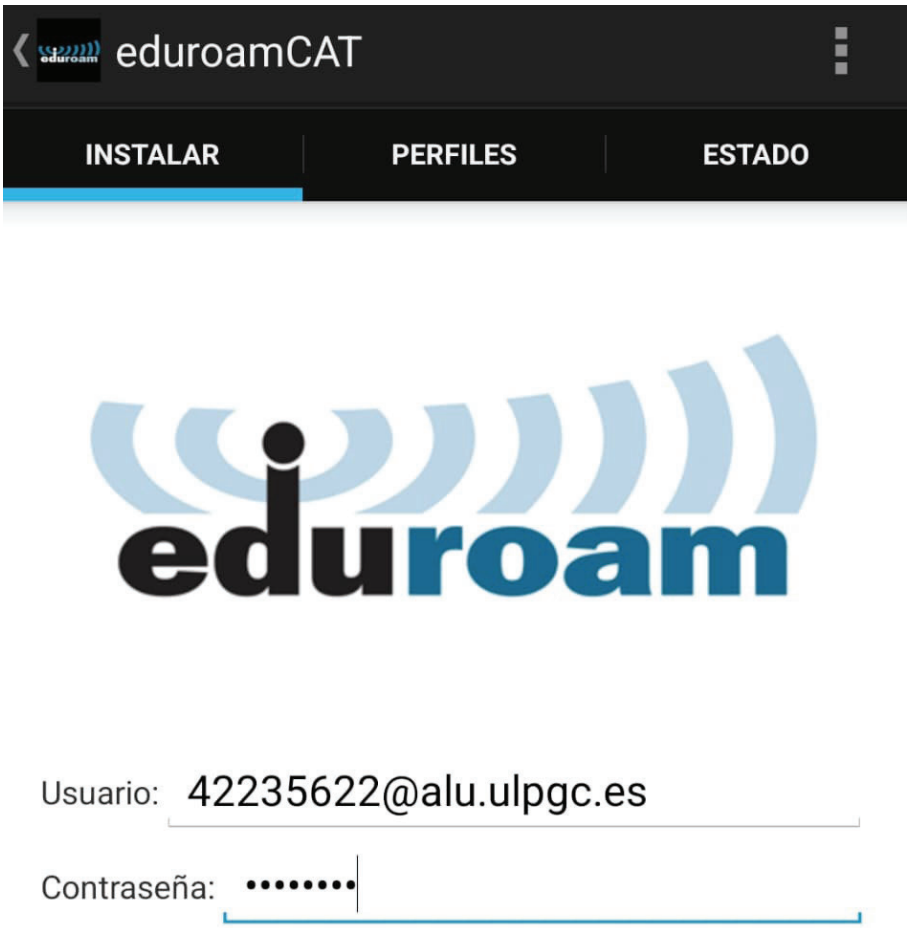

Figura 2.5 Configuración de acceso a la red eduroam con credenciales ULPGC

Aun así, *World ID* no debe entenderse como un sustituto directo, sino como una alternativa que ofrece una arquitectura completamente distinta: sin contraseñas, sin compartir datos personales, con validación criptográfica y control total por parte del usuario. Su aplicación en una institución como la ULPGC permitiría unificar el acceso a múltiples servicios mediante un único punto de autenticación, basado en *DID*.

Por ejemplo, el acceso mediante usuario y contraseña podría ser reemplazado por una verificación con *World ID*. A diferencia del *MFA*, *World ID* integra un segundo factor criptográfico-biometría sin necesidad de depender de un proveedor externo ni enviar datos personales. Frente a los certificados digitales, *World ID* no requiere instalación de hardware ni configuración adicional, y ofrece verificación directa desde una *wallet Web3*.

En la *Figura 2.6* se muestra la interfaz de acceso diseñada para ULPGC Verify. En esta propuesta, tras una única verificación con *World ID*, el usuario accedería a un menú con enlaces simulados a distintos servicios institucionales, demostrando así su potencial como solución unificada.

Para que esta solución se convierta en una propuesta real que supere a los métodos actuales, deberían implementarse funcionalidades como:

- Soporte para protocolos de identidad académica y federada.
- Integración con sistemas de gestión de usuarios de la ULPGC.
- Mecanismos de recuperación y verificación offline.
- Infraestructura propia (por ejemplo, una *blockchain* institucional o un nodo verificador).

En los próximos capítulos se explica en detalle cómo se ha implementado la propuesta *ULPGC Verify*, qué tecnologías se han utilizado y cómo se ha estructurado la demo funcional del sistema.

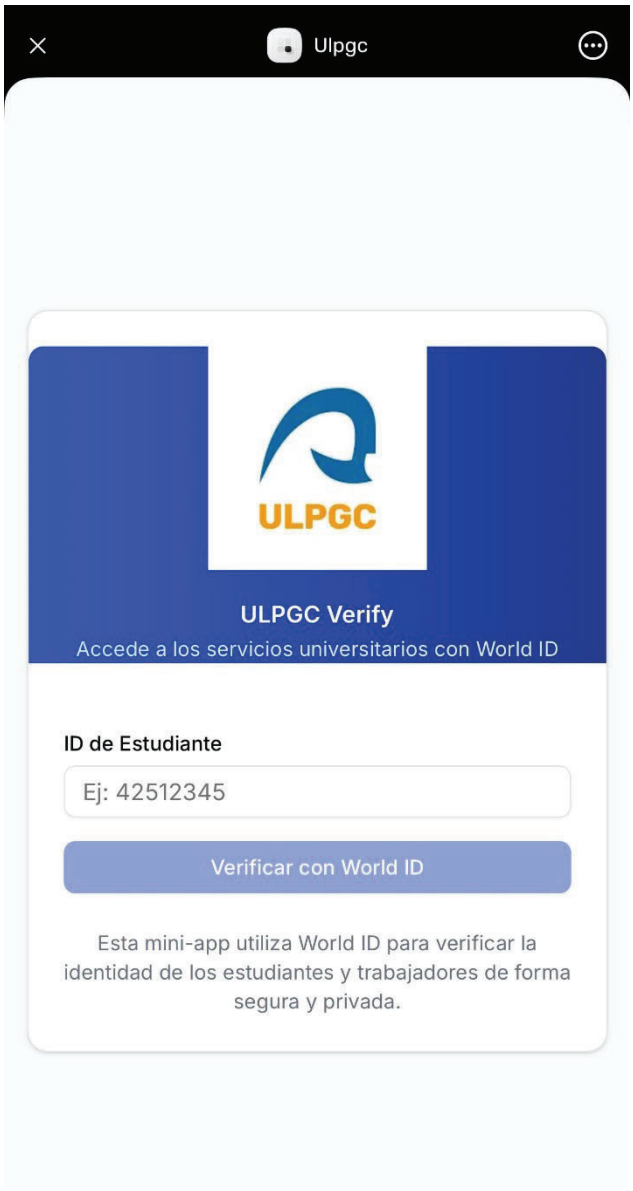

Figura 2.6. Menú principal de ULPGC Verify con World ID

3. Tecnología de Worldcoin

Se analiza en profundidad la tecnología sobre la que se construye el protocolo *Worldcoin*, comenzando por el sistema de Ethereum y sus niveles de escalabilidad, hasta llegar a los componentes de *Worldcoin*, detallando los conceptos de *Layer 2* (*OP Stack*), los *Smart Contracts* y las aplicaciones descentralizadas, y explicando los elementos fundamentales del sistema *Worldcoin*.

3.1 Ethereum

Ethereum es una *blockchain* de código abierto sobre la que: a) se ejecutan las aplicaciones descentralizadas (*DApp*) [27], b) a diferencia de la versión original de Bitcoin [67] permite ejecutar *Smart Contracts* [68], c) es compatible con estándares y herramientas ampliamente adoptados para el desarrollo *Web3*, lo que facilita la creación de soluciones interoperables y d) soporta a *Worldcoin* (que implementa verificación de identidad, la gestión *wallets* y la integración de las DApp) [69].

Destaca por su capacidad para:

- Ejecutar operaciones sin intermediarios mediante su modelo descentralizado.
- Asegurar la integridad de los datos mediante algoritmos de consenso.
- Almacena las aplicaciones y registros en una estructura inmutable.
- Adaptarse a casos de uso diversos mediante un sistema flexible y modular.

3.1.1 Conceptos básicos

Revisamos brevemente algunos conceptos básicos: su criptomoneda nativa *Ether* [70], el modelo de *cuentas* [71] utilizado para identificar y gestionar los usuarios en la red, la *Ethereum Virtual Machine* (*EVM*) [72] como entorno de ejecución de *Smart Contracts*, el sistema de *gas* para la medición de costos computacionales, y mecanismos avanzados como los *contratos multifirma* [73], que refuerzan la seguridad en la gestión de fondos o autorizaciones dentro de la red.

Gas

El *gas* es la unidad de medida del esfuerzo computacional que se ejerce cuando se ejecuta una transacción (operación que cambia el estado de la blockchain de Ethereum añadiendo bloques o ejecutando un Smart Contract) o un *Smart Contracts* en la red. Cada transacción contiene una tarifa base de *gas* (*Figura 3.1*). Dado que cada transacción implica un

consumo de recursos computacionales, el *gas*: regula el uso de la blockchain, evita usos abusivos, controlar envío masivo de transacciones fraudulentas o bucles de ejecución infinitas.

Ether

Los *validares* generan la *Ether* (*ETH*) proponiendo y confirmando bloques dentro de una *Epoch* [74]. Por ello reciben *ETH* dependiendo del número de validadores y de la cantidad que hayan apostado. El *burning* es el proceso para destruir o eliminar algunos *ETH*. Cuando aumenta la demanda de las *ETH* se controla la inflación eliminando una cantidad elevada de *ETH*. Con este mecanismo de *burning*, se promueve un mercado de comisiones más transparente y justo[70].

Cálculo del coste de una transacción

Para que un usuario pueda realizar una acción en la red, debe pagar una tarifa de *gas*. Su coste se calcula multiplicando la cantidad de *gas* que se requiere para la operación por el costo unitario del *gas* en ese momento. Las tarifas de *gas* se deben pagar con *ETH*.

Por lo general, el precio del *gas* se representa en *gwei* (10^{-9} *ETH*). Los usuarios pueden definir la cantidad de *gas* que están dispuestos a pagar cuando envían una transacción, es decir, cada usuario puja para que su transacción sea incluida en el bloque. La cantidad total de *gas* pagada se divide en dos partes: a) *Base fee*: valor establecido automáticamente por el protocolo, que indica el mínimo necesario para que una transacción sea válida (se estima entre 15 y 30 millones de unidades de gas en función de la congestión de la red y el tamaño del bloque y el *Wallet* lo calcula automáticamente [75]), y b) *Priority fee*: incentivo opcional que los usuarios pueden añadir para que su transacción sea procesada más rápido, ya que los validadores prefieren incluir las transacciones con mayor *priority fee*.

Por ejemplo, el *usuario1* envía 1 *ETH* al *usuario2*. Si ese envío requiere 21.000 unidades de *gas*, si la *base fee* es 10 *gwei* y *usuario1* añade una *priority fee* de 2 *gwei*, entonces el coste de *gas* sería: $\text{Unidad de Gas} \times (\text{Tarifa base} + \text{Tarifa Prioritaria})$, esto $21.000 \times (10 + 2) = 252.000\ \text{gwei} = 0.000252\ \text{ETH}$. Por tanto, la cuenta de *usuario1* se reduce en 1,000252 *ETH*.

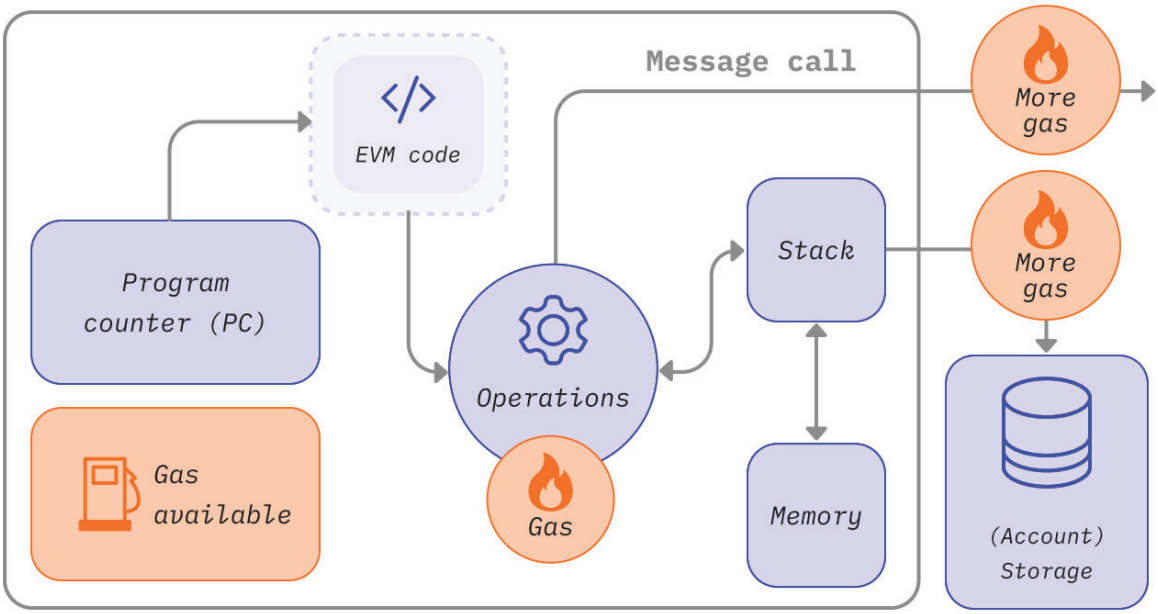

Figura 3.1 Coste de transacciones usando gas

Fuente*:*

https://ethereum.org/next/image/?url=%2Fcontent%2Fdevelopers%2Fdocs%2Fgas%2Fgas.png&w=828&q=75

Los usuarios pueden establecer la cantidad más alta que están dispuesto a pagar por una transacción (*maxFeePerGas*). Si Base *fee* + *Priority fee* < *maxFeePerGas*, entonces se le reembolsa la diferencia al usuario. Este mecanismo hace que la gestión de los costos sea más flexible sin la necesidad de pagar de más innecesariamente por una transacción.

Cuentas

Una cuenta se identifica mediante una dirección de 20 B y permite realizar transacciones, almacenar información y ejecutar *Smart Contracts*. Está compuesta por cuatro componentes:

- *Nonce*: contador que evita que una misma transacción se pueda procesar más de una vez.
- *Saldo en ETH*: cantidad de *ETH* que se almacenan en dicha cuenta, que puede ser utilizada como reserva de valor para pagar las comisiones.

- *Código de Contrato*: para las cuentas de contrato, se incluye un código de contrato para definir su comportamiento.

- *Almacenamiento Interno*: donde se guardan las variables y los datos relevantes para poder ser ejecutado.

Existen dos tipos de cuenta en *Ethereum*:

- *Externally Owned Accounts* (*EOA*): controladas por claves privadas y pueden enviar o recibir *ETH* mediante la firma de transacciones, operando como un programa autónomo que se activa cuando recibe una llamada de otra cuenta.
- *Contract Accounts*: contienen el código de *Smart Contract* y se ejecutan mediante instrucciones automáticas cuando reciben alguna transacción.

Estos contratos no deben interpretarse como acuerdos legales tradicionales que deben cumplirse, sino más bien como agentes descentralizados que reaccionan a transacciones según su programación [71].

3.1.2 Smart Contracts

Un *Smart Contract* es un programa que se ejecuta dentro de la *blockchain Ethereum*. Su tamaño máximo es de 24 KB. Contiene funciones y datos, y se referencia mediante una dirección en la *blockchain*. Se escriben en los lenguajes *Solidity* o *Vyper*. Una cuenta de usuarios interactúa con un *Smart Contracts* creando una transacción. Puede hacerlo con cualquier *Smart contract* porque todos son públicos. No pueden acceder a recursos fuera de la *blockchain Ethereum* por seguridad. Para acceder a recursos externos se utiliza un *oráculo*. Su utilidad está en: a) las finanzas descentralizadas (*DeFi*), que son préstamos, intercambios y *staking* automatizados, b) la gestión de activos digitales y sistemas de propiedad, y c) votaciones y gobernanza en sistemas transparentes y seguros para la toma de decisiones [68].

En la *Figura 3.2* se muestra un ejemplo de *Smart Contract* simple desarrollado en *Solidity*, llamado *WorldIDVerificationRegistry.sol*. Este almacena los registros de verificación relacionados con identidades validadas por *World ID* utilizando un *nullifierHash*

único por usuario, para demostrar cómo *World ID* podría registrar pruebas en una *blockchain Ethereum*. Tiene las siguientes características:

- Se define una estructura VerificationInfo con datos como el timestamp y un valor booleano que indica si fue verificado.
- Permite registrar una verificación con registerVerification().
- Expone funciones de lectura como isVerified() y getVerificationInfo() para consultar el estado de un hash.

Máquina Virtual de Ethereum

La *Máquina Virtual de Ethereum* (*EVM*) es el entorno computacional descentralizado en el que se ejecutan los *Smart Contracts*. Su objetivo es garantizar que cada nodo de la red se ejecute con el mismo código y de forma segura. Para operar utiliza *gas*. La *Máquina de Estados Distribuida* (*MED*) registra las cuentas, los saldos, almacena la información y el estado de los *Smart Contracts*. Evoluciona (cada estado se almacena en un bloque) en función de un conjunto de reglas definidas por la *EVM*. El *MED* se almacena en una estructura de datos llamada *Merkle Patricia Tree* (*MPT*) [76], enlazando las cuentas mediante un único *hash raíz*. El *MPT* permite verificar de manera eficiente cualquier cambio de *MED* sin la necesidad de almacenar todas las transacciones, ayudando a optimizar los recursos de la red.

La *EVM* funciona como una *stack machine* de 1024 elementos de 256 *b para hacerla compatible* con algoritmos criptográficos como *Keccak-256* [77] y *secp256k1* [78]. Durante la ejecución de un *Smart Contract*, la *EVM* mantiene una memoria temporal (array de *bytes*), que no se conserva entre transacciones. Sin embargo, los *Smart Contracts* almacenan datos persistentes en un *MPT* (vinculado a su cuenta y forma parte del *MED*). El mapeo de un *Smart Contract* en la *EVM* se puede hacer con los lenguajes: *Py-EVM* (*Python*)*, evmone* (*C++*)*, ethereumjs-vm* (*JavaScript*) *y revm* (*Rust*) [72].

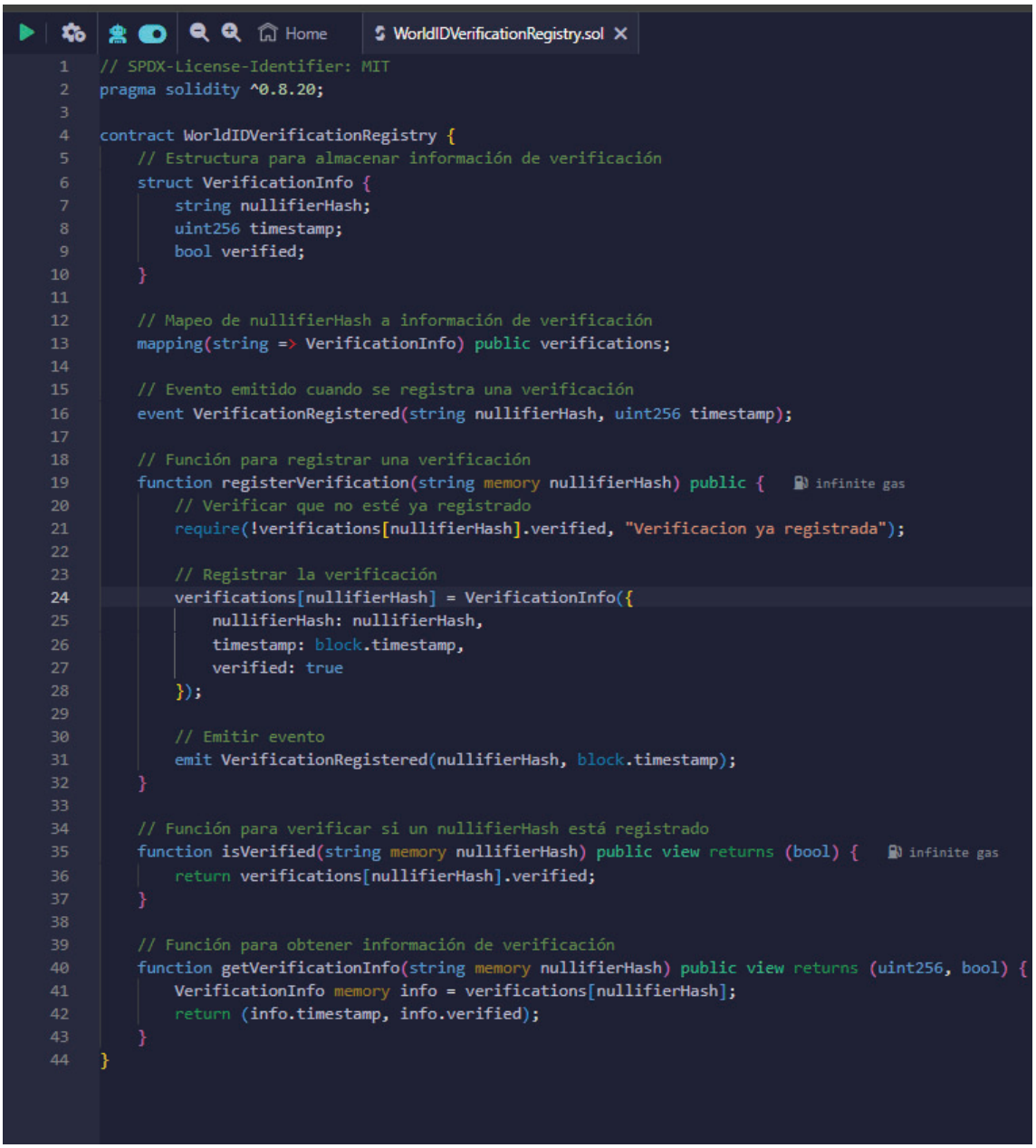

Figura 3.2 Ejemplo de un Smart Contract de Ethereum

Smart Contract multifirma

Exige la validación de varias firmas para poder autorizar una transacción. Este mecanismo se usa para grandes cantidades de *ETH* y así prevenir fallos críticos en la *blockchain Ethereum*.

Este método distribuye la responsabilidad para la ejecución de un *Smart Contract* y la administración de las claves entre varias partes, lo que reduce el riesgo de perder

información en caso de que se pierda una única clave privada. Por estas razones, un *Smart Contract* multifirma simplifica la gobernanza de las Organizaciones Autónomas Descentralizadas (*DAO*). Se implementa mediante un esquema en el que necesita *N* firmas válidas de un total de *M* firmas posibles, donde $N \leq M$ y $M > 1$, para poder ejecutar una transacción. Existen configuraciones por defecto que son $N = 3$, $M = 5$ o también $N = 4$, $M = 7$. Por lo tanto, se asegura que la mayoría de los titulares de las claves deben estar de acuerdo y firmar para que el *Smart Contract* se ejecute, lo que añade un nivel adicional de seguridad y consenso [73].

Token

Un *token* en el contexto de la tecnología *blockchain* es una unidad digital programable que representa un activo, un derecho o una utilidad dentro un sistema descentralizado. Son emitidos y gestionados por *Smart Contracts* que definen su comportamiento y reglas de uso.

Existen tres estándares para representar *tokens* en *Ethereum*, que son:

- ERC-20: estándar para *token* fungibles, que pueden ser intercambiados por otros del mismo tipo. Se utilizan para representar criptomonedas o recompensas [79].

- ERC-721: se define como *Non-Fungible Token* (*NFT*), cada uno es único e irrepetible. Se usan para representar obras de arte digitales, certificados, títulos de propiedad, entre otros [80].

- ERC-1155: gestiona los tokens fungibles y *NFT* en un único *Smart Contracts*, optimizando recursos [81].

Al estar gestionado por un *Smart Contract*, permite definir funciones como transferencias, asignación de permisos, emisión limitada, entre otras características. También pueden utilizarse como mecanismos de gobernanza, identidad digital, acceso a funcionalidades exclusivas o como instrumentos financieros en aplicaciones de *DeFi* [82].

3.1.3 Aplicación Descentralizada

Una Aplicación Descentralizada (*DApp*) utiliza un *backend* (*Smart Contract*) que se ejecuta en la *blockchain*, y el *frontend* (interfaz de usuario), que puede desarrollarse en lenguajes *web* tradicionales como *HyperText Markup Language* (*HTML*), *Cascading Style Sheets* (*CSS*) y *JavaScript*, y alojarse en un sistema descentralizado. Una *DApp* de *Ethereum* se ejecuta en la *EVM*, lo que asegura que los errores que puedan existir en un *Smart Contract* no afecten el funcionamiento general de la *blockchain*.

Entras las ventajas que presenta una *DApp* están que los datos se almacenan en la *blockchain*, garantizando su integridad y autenticidad. Sin embargo, las *DApp* tienen numerosas dificultadas como su mantenimiento, que una vez implementada, son difíciles de modificar debido a la naturaleza inmutable de la *blockchain*. Cada nodo debe guardar y ejecutar todos los intercambios, esto puede sobrecargar la red [27].

3.2 Escalabilidad y mejoras de Ethereum

Ethereum tuvo que mejorar su escalabilidad, para tener una mayor capacidad de procesamiento de las transacciones de forma rápida y eficiente sin comprometer la seguridad ni la descentralización. Aunque la red ha evolucionado con actualizaciones como *The Merge* [83] o la transición a *PoS*, el crecimiento de las *DApp* ha limitado su capacidad, especialmente en congestión de red y costes elevados de gas.

Para abordar estos problemas, *Ethereum* desarrolló su *Layer 2* para procesar las transacciones externamente. Después almacenan los resultados en la red principal [84]. Destacan las tecnologías como los *Optimistic Rollups*, los *zk-Rollups*, los *state channels* [85], *sidechains* [86] y *frameworks* de desarrollo como el *OP Stack* [24], que permiten construir nuevas *blockchains* compatibles con *Ethereum*.

Estas mejoras aumentan el rendimiento, facilitan la creación de sistemas descentralizados y accesibles para usuarios y desarrolladores.

3.2.1 Mecanismos de Rollups

Los usuarios firman las transacciones y las envían a los operadores de *Rollup*, que las procesan y las agrupan por lotes antes de ser registradas en *Ethereum* utilizando un *calldata* [87] (región de memoria que almacena los datos de las transacciones dentro de un *Smart Contract*, del *Layer 1,* sin alterar su estado) o *blobs* [88]. Permiten reducir los costes y aumentar la velocidad de las transacciones sin comprometer la seguridad y descentralización. Esto se debe a que los operadores de *rollups* agrupan múltiples transacciones en grandes lotes (pudiendo emplear técnicas de compresión de datos [20]) antes de enviarlas a la *blockchain Ethereum Layer 1*. Con esta agrupación se distribuye los costos fijos entre muchas transacciones, disminuyendo las tarifas para los usuarios.

En *Optimistic Rollups* todas las transacciones ejecutadas son válidas por defecto (sin necesidad de verificarlas individualmente antes de registrarlas en la *blockchain*) y solo se verifican si se representa una discrepancia. La resolución de una discrepancia (mediante un mecanismo de prueba de fraude) consiste en que cualquier usuario puede impugnar una transacción sospechosa. Si se demuestra que la transacción no es válida, el protocolo vuelve a ejecutar la operación y corrige el estado del sistema, además el *Operador* (*Aggregators* o *Validators*) que incluyó la transacción incorrecta es penalizado económicamente. Si no hay objeciones dentro del período de desafío, el lote se considera válido y pasa a formar parte de la *blockchain Ethereum Layer 1* definitivamente [26].

Los componentes de un *Optimistic Rollup* son*:*

- *Onchain Contracts*: se encargan de almacenar los bloques de *Optimistic Rollups*, de actualizar los estados y gestionar los depósitos de cada usuario.
- *Offchain Virtual Machine* (*VM*): compatible con la *EVM*, procesa las transacciones fuera de la *blockchain Ethereum Layer 1*.

En *Zero-Knowledge Rollups* los *operadores* publican un resumen de los cambios necesarios para así poder reflejar todas las transacciones de un lote, incluyendo las pruebas de validez criptográficas útiles para confirmar que las modificaciones realizadas sean correctas. Sólo se publican en la *blockchain Ethereum Layer 1* las pruebas criptográficas que certifican la

validez de un lote. Por tanto, los nodos de la *blockchain* que deseen alterar el estado del *Smart Contract* que contiene la raíz de un árbol de *Merkle* que describe las transacciones del lote, el cual se debe aportar la prueba de validez.

También incluye un contrato verificador, que valida las *ZKP* que son generadas por los productores de bloques. Este historial de transacciones es inmutable y constituye a la cadena de *ZK-Rollup*. Existen dos tipos de *ZKP*: a) *ZK-SNARKs*: generan pruebas concisas y rápidas de verificación, pero requieren una configuración de confianza inicial para establecer parámetros seguros y b) *ZK-STARKs*: no dependen de una configuración de confianza y son más escalables, pero generan pruebas más grandes, lo que aumenta los costos de verificación en *Ethereum* [20].

Una transacción en un *ZK-Rollup* solo se considera finalizada si el contrato en *Ethereum* acepta la prueba de validez. Esto impide que operadores maliciosos alteren la *blockchain Ethereum Layer 1*, ya que, si un operador intenta bloquearla, el usuario puede enviarla directamente al *Smart Contract* en *Ethereum*, asegurando la descentralización [20].

En algunos casos, el Operador es un secuenciador centralizado, siendo el responsable de ejecutar las transacciones y publicar los bloques en la *blockchain Ethereum Layer 1*. También rota un conjunto de validadores en prueba de participación, donde los participantes bloquean los fondos en el contrato del *rollup* y pueden perderlos si actúan de manera deshonesta.

3.2.2 State Channel

Se basa en un flujo de operaciones claramente definido y estructurado en tres fases:

- *Apertura del canal*: los usuarios que realizan transacciones despliegan un *Smart Contract* en la *blockchain Ethereum Layer 1* y depositan una cantidad de *tokens* como garantía. En él se define las reglas del canal, gestiona los depósitos y actúa como árbitro en caso de disputa, luego se firma como un estado inicial que marca el punto de partida del canal.

- *Transacciones offchain*: una vez abierto el canal, los usuarios intercambian transacciones firmadas entre sí, en la que cada transacción incluye:
    - Un *nonce*, que indica el orden de las actualizaciones y previene ataques de repetición.
    - El estado anterior del canal.
    - El nuevo estado actualizado de la transacción.
    - La operación ejecutada, por ejemplo, una transferencia de *tokens* entre las partes.

  Estas actualizaciones no se registran en *blockchain Ethereum Layer 1* hasta su cierre, reduciendo el coste de *gas*, mientras que ambas partes firman nuevos estados y se consideran válidos, igual que una transacción *onchain*.

- *Cierre del canal*: se publica en *Ethereum* el último estado firmado por todas las partes. El *Smart Contract* valida las firmas y distribuye los fondos en función del saldo final, en caso de que ambas partes estén de acuerdo, el cierre del canal es de manera inmediata. En caso de conflicto, se activa una ventana de desafío, durante la cual cualquier parte puede impugnar el estado propuesto presentando una versión más reciente [85].

Los *State Channels* dependen de *Ethereum* en los siguientes aspectos:

- *Apertura y cierre del canal*: para iniciar un canal, las partes implicadas deben desplegar un *Smart Contract* en *Ethereum* y depositar fondos, y para finalizar el canal, se publica el último estado firmado por todas las partes, permitiendo que el contrato distribuya los fondos según el acuerdo final.

- *Seguridad mediante pruebas de fraude*: en caso de desacuerdo, cualquier participante puede activar un proceso de disputa *onchain*, en el cual se verifica la validez del último estado enviado. En caso de que una de las partes intente cerrar el canal con un estado obsoleto o no válido, la otra parte puede responder con el estado más reciente firmado, lo que garantiza la protección de los fondos.

- *Garantía de disponibilidad* (*liveness*): mientras el contrato esté desplegado en *Ethereum*, el canal permanece operativo. En cambio, de otras soluciones como las *sidechains*, que pueden fallar independientemente, la disponibilidad del canal depende directamente del estado de la red principal.

- *Finalidad de las transacciones*: aunque las operaciones *offchain* tienen validez si están firmadas por todos los participantes, solo adquieren finalidad definitiva una vez que son registradas en la cadena principal al cerrar el canal [85].

3.2.3 Sidechains

Las *Sidechains* son *blockchains* independientes que se ejecutan en paralelo a *Ethereum* y que permiten la transferencia de activos entre ambas redes mediante mecanismos de interoperabilidad, conocidos como *bridges* (*Smart Contract* desplegados tanto en *blockchain Ethereum Layer 1*), aunque pueden compartir similitudes superficiales con *Ethereum*. Estas tienen su propio historial de bloques, lógica de ejecución, algoritmos de consenso y parámetros técnicos, sin depender directamente de *Ethereum* para su seguridad o validación. Esto implica riesgos y compromisos importantes en términos de descentralización.

Su seguridad recae directamente en su conjunto de *validadores*, lo que significa que no heredan las garantías de seguridad que ofrece el nivel principal de *Ethereum*. Estas redes que se ejecutan de manera paralela permiten ejecutar *DApp* compatibles con *EVM* de forma eficiente, con tarifas de transacción reducidas y tiempos de bloque más rápidos.

Los *bridges* transfieren activos entre ambas redes, sin mover realmente los *tokens*, sino bloqueándolos en una cadena y emitiendo un equivalente en la otra mediante mecanismos de *minting* y *burning*. Son el elemento crítico porque si se vulnera el *Smart Contract* en una de las cadenas, los activos de los usuarios pueden estar en peligro [86].

Una *sidechain* puede emplear tiempos de bloque más cortos y límites de gas más altos. Esto incrementa el *throughput*, reduce las tarifas y dificultan que cualquier usuario ejecute un nodo completo. Esto favorece la centralización al depender de *supernodos* con mayor

capacidad computacional. Puede ser compatibles con *EVM* facilitando la migración de *DApp*.

3.2.4 Framework OP Stack

Su objetivo es proporcionar una infraestructura software estandarizada, modular y de código abierto para la creación de *blockchains Layer 2* que puedan operar sobre *Ethereum Layer 1* mediante *Optimistic Rollups*. Conforma la *OP Mainnet* que es la *blockchain* de *Optimism* (componente principal de la iniciativa de *Optimism Superchain*). Una *Superchain* es una *blockchain* interconectada con múltiples *blockchains* (todas basadas en la infraestructura de *OP Stack*, formando un ecosistema denominado *OP chains* que eliminan los cuellos de botella de *Ethereum* para el despliegue de *DApp* [89]), que comparten *bridging*, gobernanzas descentralizadas, actualizaciones sincronizadas, un nivel de comunicación común, entre otras cosas.

Su arquitectura permite crear *blockchains* compatibles con *Ethereum* y formadas con el modelo de interoperabilidad de la *Superchain* [25] que es una mejora de escalabilidad en *OP Stack* del *Bedrock* [90]. Aunque aún no ha cumplido aún con su potencial para construir una *web* verdaderamente descentralizada. Cada *OP Chain* puede configurarse de manera individual según sus necesidades, incluyendo la modificación del secuenciador, definir los mecanismos de disponibilidad de los datos y pueden elegir el modelo de seguridad entre pruebas de fraude o pruebas de validez criptográfica.

Su desarrollo está basado en los siguientes principios:

- *Simplicidad*: utiliza el menor número de componentes posibles para garantizar un sistema seguro.
- *Pragmatismo*: se enfoca en responder a necesidades concretas de los usuarios y desarrolladores, teniendo como resultado una adopción mucho más eficiente y rápida.
- *Sostenibilidad*: está orientado a la viabilidad a largo plazo.

- *Optimismo*: permitir que *Ethereum* pueda escalar sin perder sus principios de seguridad y descentralización [91].

Los niveles de la arquitectura de *OP Stack* son:

- *Data Availability Layer*: define dónde se almacenan los datos de entrada de la *blockchain* basada en *OP Stack* mediante: *calldata*, *logs* de eventos o *blobs Proto-Danksharding* (*EIP-4844*) [92].

- *Sequencing Layer*: por defecto, *OP Stack* opera con un secuenciador centralizado, encargado de controlar como se recopilan y se publican las transacciones de los usuarios al nivel de disponibilidad de datos. Aunque en un futuro se espera implementar algunos modelos con múltiples secuenciadores descentralizados.

- *Derivation Layer*: procesa los datos crudos y los convierte en entradas estructuradas para así poder ser ejecutadas como *Smart Contracts* en la *EVM*.

- *Execution Layer*: define el manejo de los estados de la *blockchain* y ejecuta las transacciones dentro del entorno de la *EVM* para *Smat Contracts* y las *DApp*.

- *Settlement Layer*: permite que otras redes puedan verificar y registrar los estados de *OP Stack* en las *blockchains* externas, lo que garantiza una interoperabilidad y mecanismos de salida seguros [93].

Una característica fundamental del funcionamiento de la *Superchain* es la derivación de los estados de forma determinista desde *Ethereum*. Esto permite que cualquier nodo sincronice sin necesidad de validadores externos, lo que reduce la fragmentación y mejora la descentralización. Al trabajar todas las *OP Chains* sobre la *blockchain Ethereum Layer 1* heredan su la base de seguridad y validación. Esto es, *OP Stack* mantiene una elevada compatibilidad con *Ethereum*, aunque introduce algunas diferencias respecto a su funcionamiento:

- *El bridging entre Ethereum y OP Stack*: permite a los usuarios mover activos entre ambas *blockchains* y verificarlos con pruebas de fraude.

- *Introduce distintos estados de finalización de los bloques:* un bloque pasa por las fases: *Unsafe*, *Safe* y *Finalized*.

- *Las blockchains defindas con OP Stack no cuentan como un mempool público*: el secuenciador el único actor con acceso a las transacciones pendientes, lo que mejora su eficiencia; pero puede introducir algunos riesgos de centralización.

- Los desarrolladores pueden personalizar algunos componentes, como el secuenciador y los mecanismos de validez. Pero la modificación del código base puede hacer que la *blockchain* resultante sea incompatible con el *Optimism Superchain* [94].

Un aspecto importante del funcionamiento de la *Superchain* es su modularidad en la secuenciación de las transacciones. Aunque por defecto *OP Stack* opera con un secuenciador centralizado, la infraestructura de la *Superchain* permite la integración de múltiples secuenciadores descentralizados, como se muestra en la *Figura 3.3*. Lo que mejora la resiliencia y minimiza los riegos de censura.

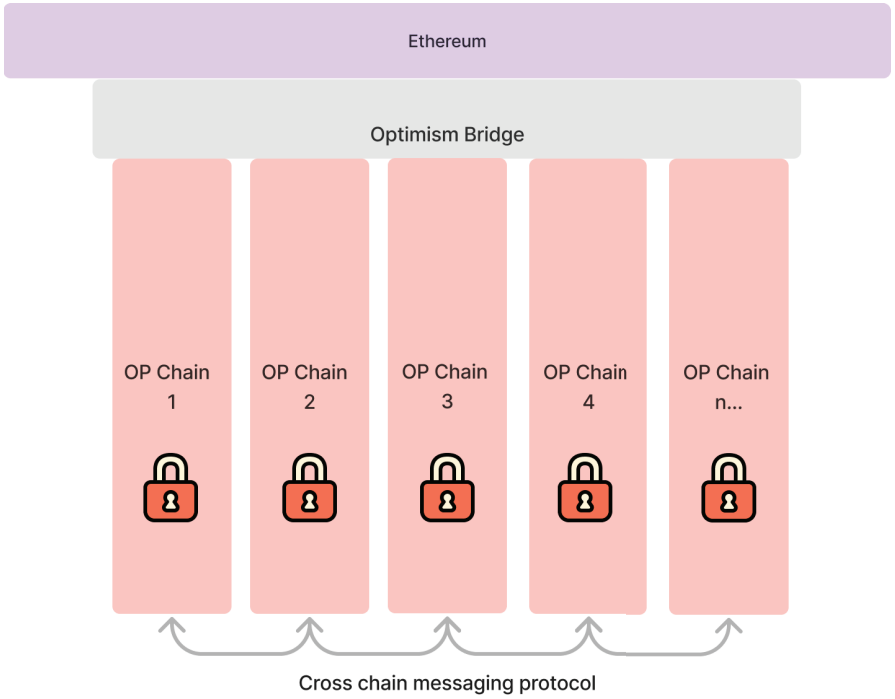

Figura 3.3 Infraestructura de la Superchain

Fuente: https://docs.optimism.io/_next/image?url=%2F_next%2Fstatic%2Fmedia%2Fsuperchain-diag.7fc0979f.png&w=1920&q=75

También introducen una mejora de los mecanismos del sistema de mensajería entre las *OP chains*. Esto permite ejecutar y validar transacciones con baja latencia. En caso de que existan fallos sistemáticos o riesgos de seguridad, el sistema tiene la capacidad de restaurar la funcionalidad de la *Superchain* mediante el mecanismo *soft forks* en *Ethereum* sin afectar su integridad operativa.

Otro aspecto por resolver del funcionamiento de la *Superchain* es la optimización de la disponibilidad de los datos, implementando un modelo hibrido en *OP Stack*, donde los datos pueden ser almacenados en *Ethereum* como *calldata*, *blobs* de *EIP-4844* o en soluciones externas de disponibilidad de datos (*Alt-DA*), lo que reduce de manera significativa los costos de almacenamiento sin comprometer la seguridad de las transacciones [89].

3.3 Tecnología World

La tecnología *World* es un proyecto criptográfico que nace como una *DApp*, cuenta con 3 elementos: la autenticación biométrica, su *token* nativo llamado *Worldcoin* (*WLD*) [10], y su propia *blockchain* descentralizada [95]. La interacción con la red se realiza principalmente a través de la *World App* (en la que los usuarios se registran con el *Orb* [13] y custodian sus credenciales [96]). Aunque la *World App* es el primer *frontend* oficial, su arquitectura de código abierto permite que terceros puedan desarrollar alternativas, generando ecosistemas diversos de las *DApp*.

3.3.1 World ID

World ID es la identidad humana de cada usuario registrado en esta *blockchain* [97]. Basado en un protocolo criptográfico-biométrico diseñado para garantizar que cada usuario registrado en la red de *World* sea único, se podría considerar un *pasaporte digital* dentro del sistema [98]. La asignación de credenciales a los usuarios se hace con el dispositivo biométrico *Orb* (Anexo A). Este sistema implementa varios niveles de seguridad y control

para poder garantizar su integridad y funcionalidad, entre los principales elementos de su arquitectura se encuentran:

- *La deduplicación*: garantiza que cada usuario solo pueda obtener un único *World ID*, dado que la verificación biométrica del iris es el método más preciso para comprobar la unicidad de un individuo. Con el diseño del *Orb* se puede realizar estas tareas con la mayor precisión posible, incluso en entornos donde podrían existir intentos de suplantación. Uno de los principales desafíos de cualquier sistema biométrico es la tasa de falsas aceptaciones, es decir, la posibilidad de que una persona no autorizada logre engañar al sistema, para disminuir este riesgo. El *Orb* incorpora múltiples sensores que ocupan el espectro electromagnético, complementado con un sistema de iluminación multiespectral que ayudan a diferenciar los intentos de fraude de interacciones ilegítimas. Además, dispone de un potente procesador que permite ejecutar múltiples redes neuronales en tiempo real para detectar posibles ataques de suplantación [99].

- *La autenticación*: permite que los usuarios puedan demostrar que son los propietarios de su credencial, sin necesidad de exponer sus claves privadas. Esto evita que los *World ID* se vendan o roben. Para el sistema de autenticación existen dos tipos de mecanismos de autenticación que son, el reconocimiento facial y la autenticación mediante el iris.

    - *La autenticación facial*: funciona de manera similar al sistema de *Apple Face ID* [100], comparando 1:1 el rostro del usuario con una plantilla previamente almacenada en su dispositivo, como muestra la *Figura 3.4*, para poder garantizar la seguridad de los usuarios, el *Orb* firma y cifra la información biométrica antes de ser enviada a la *wallet* del usuario, donde se almacena de manera segura. Este método extiende las garantías de seguridad del *Orb* al dispositivo móvil del usuario. Aunque esto presenta ciertas limitaciones debido a que los teléfonos personales no pueden considerarse dispositivos completamente seguros, para reforzar la seguridad, actualmente se están explorando soluciones de

Zero-Knowledge Machine Learning (*ZKML*) [101]. Aunque no ofrecen las mismas garantías que el sistema de autenticación mediante el *Orb* [102].

- *La autenticación por iris*: se usa en caso de que se requiera un nivel alto se seguridad, permitiendo verificar la identidad de usuario con el mismo *Orb*, con el que se emitió su *World ID*. Para este sistema, el usuario debe escanear un código *Quick Response* (*QR*) el cual es generado por su *wallet* y ser presentado físicamente ante el *Orb*. Así demostraría que sigue siendo el legítimo propietario de su credencial. Aunque este método es menos práctico debido a la necesidad de presentarse físicamente ante un dispositivo *Orb*, ofrece un nivel de seguridad adicional, que puede ser comparable a los procesos de verificación presencial utilizados en bancos y notarías [103].

- *La recuperación de credenciales*: permite a los usuarios recuperar su identidad en caso de pérdida de acceso o compromiso de su cuenta. La forma más sencilla de recuperar las credenciales de un *World ID*, es mediante una copia de seguridad. Si bien para el futuro están dispuestos a desarrollar un mecanismo de recuperación social. El método principal de recuperación es la reemisión de la credencial, lo que significa que el usuario debe volver a un *Orb* para verificar nuevamente su identidad. Esto elimina la necesidad de recordar contraseñas o credenciales adicionales, asegurando que solo el usuario pueda recuperar su *World ID*. Sin embargo, es importante resaltar que esta recuperación solo se aplica a la credencial de *World ID* y no a otros datos almacenados en la *wallet* del usuario, lo que refuerza la seguridad del sistema [104].

- *La revocación de World ID*: si se detecta que un *Orb* ha sido comprometido o que una entidad emisora esté actuando de manera fraudulenta, la Comunidad puede determinar la anulación de los *World ID* que fueron emitidos por dicho dispositivo. Para ello, la *World Foundation* tiene la capacidad de incluir en una lista de denegación a los emisores o dispositivos que representen un riesgo para la integridad del sistema. En caso de que un usuario se vea afectado por una

revocación injusta, puede acudir a otro *Orb* para obtener una nueva credencial sin inconvenientes [105].

- *La expiración de los World ID*: aunque un *Orb* no haya sido comprometido, es posible que, con el tiempo, se identifiquen vulnerabilidades en su hardware o software. Para poder mantener los altos estándares de seguridad, *World ID* puede establecer fechas de expiración para determinadas credenciales. Esto permite que los usuarios renueven su verificación en intervalos regulares. Esta estrategia no solo refuerza la integridad del sistema a largo plazo, sino que también evita la acumulación de credenciales antiguas que podrían volverse inseguras con el paso del tiempo [106].

Face Auth

Una de las últimas actualizaciones de *World* ha sido la introducción de *Face Auth*, es un nivel de seguridad adicional que actualmente se encuentra en fase de prueba, lo cual es un sistema de verificación biométrica 1:1 que asegura que solo la persona que verificó su *World ID* en un *Orb* pueda utilizarlo. Con este mecanismo añade un nivel extra de seguridad en acciones como compras en línea, transacciones financieras y accesos seguros a aplicaciones. Su funcionamiento consiste en comparar una *selfie* tomada en el dispositivo del usuario con las imágenes capturadas por el *Orb* durante la verificación inicial y manteniendo la privacidad del usuario. Esto es porque toda la información se almacena únicamente en el teléfono móvil del usuario, cifrada con su clave personal, garantizando que ni *World* ni terceros puedan acceder a ella [107].

Anonymized Multiparty Computation

Otras de las últimas actualizaciones más significativas en la protección de datos biométricos del sistema de *World* es el *Anonymized Multiparty Computation* (*AMPC*) [108]. El APMC es una tecnología de código abierto, que mejora la seguridad y privacidad de los códigos iris utilizados en la verificación de unicidad de los usuarios. Reemplaza el sistema anterior

Secure Multiparty Computation (*SMPC*) [109], proporcionando una solución más robusta basada en *GPU* de alto rendimiento. Específicamente el sistema usa el *NVIDIA H100*, que permite hasta 50 millones de comparaciones de unicidad por segundo.

Otra de las principales mejoras introducidas por *AMPC* incluyen la eliminación del uso de *Hamming Distance* (*HD*) en texto plano, lo que significa que solo se revela un resultado binario, sin exponer los datos biométricos. Además, introduce una protección avanzada mediante máscaras de iris y el anonimato de los datos biométricos, que nunca salen del dispositivo del usuario y solo se procesan criptográficamente dentro del *Orb*.

Para lograr estas mejoras, *AMPC* utiliza una infraestructura computacional avanzada. En ella cada nodo opera en instancias *AWS p5.48xlarge* con ocho *GPU NVIDIA H100*, creando una configuración proporciona de 3200 *Gbps* de ancho de banda y 20 *exaflops* de rendimiento computacional. Esto permitie gestionar la carga de casi 7 millones de usuarios verificados, el cual es gestionado por la alianza de *World Foundation* y *Nethermind*, de la Universidad de *Erlangen-Nuremberg* (*FAU*) y el *UC Berkeley Center for Responsible Decentralized Intelligence* (*RDI*) en los Estados Unidos, asegurando que ninguna entidad centralizada tenga acceso a los datos biométricos [108].

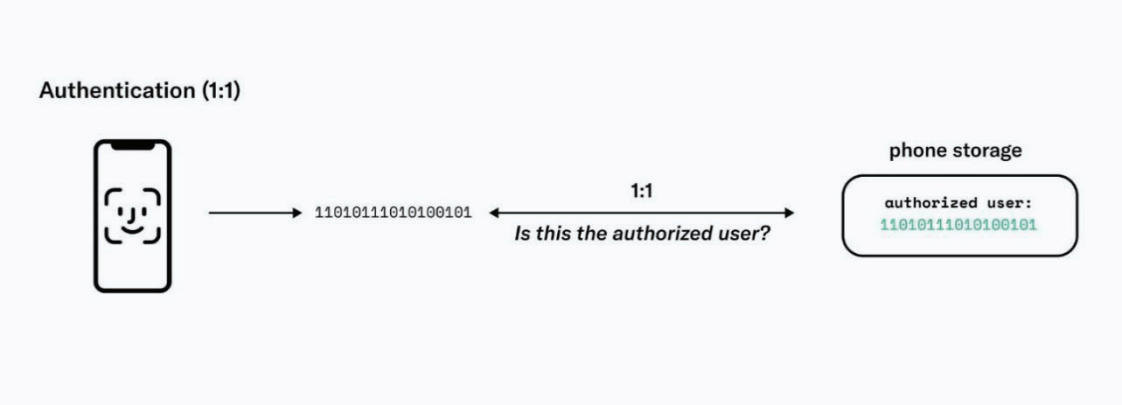

Figura 3.4 Esquema del sistema de autenticación 1:1

Fuente: https://whitepaper.world.org/images/whitepaper/notion2/image5.jpg

3.3.2 World Chain

World introduce su nueva *OP Stack blockchain* llamada *World Chain*. Está diseñada específicamente para poder gestionar las *DID* y las transacciones de manera segura, descentralizada y escalable [110]. Tiene como propósito servir como infraestructura tanto para el sistema *World ID*, como para la *World App*, asegurándose que las transacciones de los usuarios verificados tengan prioridad y que el acceso financiero descentralizado sea equitativo. Mientras que otros ecosistemas que trabajan sobre *Layer 2*, la *World Chain* introduce *Priority Blockspace for Humans* (*PBH*) [111], el cual se encarga de priorizar las transacciones de los usuarios que estén registrados con su *World ID* y así evitar que *bots* o sistemas automatizados monopolicen la red. Eso reduce los ataques *Miner Extractable Value* (*MEV*). El *MEV* consiste en que estos actores malintencionados alteran el orden de las transacciones para obtener beneficios y reducen la necesidad de pagar tarifas elevadas para asegurar inclusión entre bloques. También, se implementa el *Gas Allowance for Humans*, un subsidio de gas financiado, siendo una iniciativa de *World Foundation* para permitir transacciones sin costo, teniendo como objetivo eliminar las barreras económicas [110].

World Chain al integrar *World ID*, facilita la autenticación en plataformas digitales, el acceso seguro a servicios financieros y la prevención de fraudes, que puedan ocurrir *airdrops* [112] o votaciones descentralizadas. *World Chain* puede permitir que cualquier *DApp* pueda beneficiarse de este sistema sin necesidad de almacenar información personal de los usuarios, promoviendo un modelo más seguro y transparente para el usuario. La integración con la *PHB* garantiza que los usuarios verificados tengan prioridad en la red. Esto tiene como beneficio que los usuarios no sean desplazadas por algoritmos automatizados que intenten explotar la *blockchain* [110].

World Chain ofrece un tiempo por bloque de 2 s, en comparación con los 12 s de *Ethereum*. Mantiene un límite de gas por bloque de 30 millones, muy similar al de OP *Mainnet*. También, se ajustan los parámetros del mecanismo *EIP-1559*, reduciendo el incremento máximo de la tarifa base por bloque a un 0,8%, en comparación a los de *Ethereum* que son un 12,5%, ayudando a estabilizar los costos para usuarios con menor poder adquisitivo. Otro aspecto que garantiza la interoperabilidad es que los activos

pueden migrar desde *Ethereum* mediante el *bridging* nativo de *OP Stack*. Aunque los retiros requieren un tiempo de espera de 7 días, siendo esto una característica de los *Optimistic Rollups* [113] [114]. Para las empresas o instituciones universitarias, podrían integrarse al sistema de *World Chain*, ya que esta permite operar lo que ellos denominan *mini-App*. Es decir, una *DApp* dentro de su ecosistema, lo que podría facilitar la gestión de acceso seguro a las plataformas académicas. La verificación de *DID* en exámenes en línea o la implementación de sistemas de pago sin custodios centralizados.

World Chain está diseñado para responder a la necesidad de descentralización, una gobernanza abierta y la transparencia, mediante la monitorización de la actividad de la *blockchain* a través de herramientas como *World Dune Dashboard* y *L2BEAT*. Estas herramientas proporcionan métricas transparentes sobre las transacciones diarias, la distribución de *WLD* y la descentralización de la red. Además de todo esto, el código de *World Chain* es *open-source*, lo que permite auditorías independientes y refuerza la confianza en su infraestructura [110].

3.3.3 Proceso de Privacidad

El desarrollo para gestionar datos biométricos del sistema de *World* es innovador. Logra que la privacidad del usuario sea una prioridad fundamental. Utiliza un modelo en el que los datos biométricos no se guardan en la *blockchain* ni en servidores de Internet: se procesan de manera segura para generar un identificador único (*iris code*). El *iris code* permite verificar la *DID* de un usuario sin necesidad de revelar información personal, garantizando que cada persona en la red sea única sin comprometer su privacidad.

Como se muestra en la *Figura 3.5*, el proceso comienza con la captura de la imagen del iris a través del *Orb*. La imagen se procesa directamente en la memoria y no se almacena en el disco duro del *Orb*, a menos que el usuario opte explícitamente por hacer un respaldo de su imagen. Si el usuario no elige esta opción, la imagen se elimina inmediatamente después de su procesamiento. Si elige almacenarla, se cifra con estándares de seguridad avanzados como *AES-256* y se transmite a través de *Transport Layer Security* (*TLS*). Las

imágenes se cifran nuevamente con una clave pública del servidor, haciendo inaccesible en caso de que sea *hackeado*.

El iris *code* no es un simple *hash*, ya que dos imágenes del mismo iris pueden presentar variaciones menores debido a factores como iluminación, ángulo de captura u oclusiones parciales. Por eso se aplican filtros de *Gabor* bidimensionales sobre distintos puntos del iris, extrayendo únicamente la información de fase de la respuesta del filtro.

Este proceso implica una pérdida permanente de información: si alguien tuviera acceso al iris *code*, no podría reconstruir la imagen original del iris ni extraer datos biométricos utilizables. Esto es debido a que cada código generado es único para cada usuario y se compara mediante la *HD*, permitiendo reconocer una identidad sin necesidad de almacenar imágenes reales [115].

La privacidad de *World* esta reforzada, ya que utiliza el *ZKP* y medidas adicionales para garantizar que la recopilación de datos biométricos no comprometa la privacidad del usuario. Una de las medidas es el almacenamiento de las imágenes (opcional)que se pueden eliminar en cualquier momento desde la *World App*, como se muestra en la *Figura 3.6*. *World* se ha comprometido a no vender, ni comercializar, ni utilizar estos datos para ningún otro propósito que no sea la mejora del sistema de verificación de identidad. Este compromiso reflejado en su formulario de consentimiento de datos, que establece de manera explícita que la información biométrica no sería explotada con fines comerciales ni utilizada para publicidad de terceros [116].

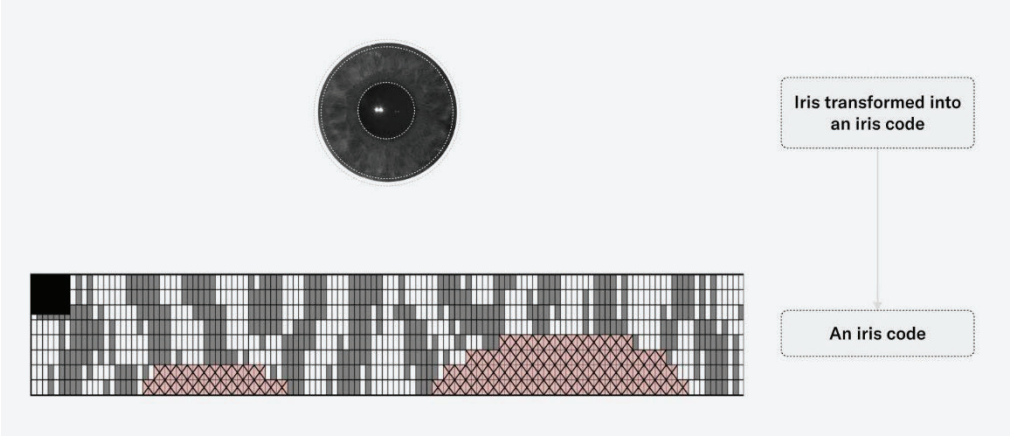

Figura 3.5 Generación del iris code a partir de la imagen del iris

Fuente: https://whitepaper.world.org/images/whitepaper/notion2/image57.jpg

En un futuro, el sistema de *World* planea permitir que los usuarios puedan gestionar sus propios datos biométricos de manera descentralizada mediante almacenamiento en custodia propia. Esto quiere decir que, en lugar de depender de un servidor central, cada usuario podría almacenar su *iris code* de forma cifrada en su propio dispositivo. También quieren que los usuarios puedan actualizar su *iris code* de manera autónoma sin necesidad de compartir datos sensibles.

Actualmente no existe ningún método que pueda reconstruir un *iris code* a la imagen original del iris. Aunque teóricamente se pudiera generar una imagen sintética que se produzca a partir de un *iris code* similar, esa imagen no sería una representación fiel del iris real, debido a la pérdida de información durante la conversión [115]. Además, cada sistema utiliza diferentes parámetros para los filtros de Gabor, lo que hace que incluso esta posibilidad sea poco práctica. Como resultado, el diseño de *World ID* garantiza que los datos biométricos de los usuarios no puedan ser explotados ni reutilizados sin su consentimiento [116].

3.2.4 World App

Los usuarios interactúan con el sistema mediante la *World App*, siendo la interfaz principal para interactuar con *World ID*. *Esta* aplicación guía a los nuevos usuarios a través del proceso de verificación utilizando el dispositivo *Orb*. Después de ser verificados la *World App* almacena de manera segura las credenciales de *World ID* y emplea protocolos criptográficos. El diseño de la *App* está orientado para facilitar el acceso sin fricciones a una infraestructura financiera descentralizada global, con la visión de que, en el futuro, múltiples *wallets* integren *World ID* como estándar [16].

Con el lanzamiento de *World App 3.0* en el último trimestre del año 2024 [117], que ha sido rediseñada desde cero para mejorar su utilidad y escalabilidad, lo que permite que la *World chain* llegue a más usuarios. Esto es porque introduce nuevas características y mejoras significativas en la experiencia de usuario, consolidado como una herramienta para el ecosistema. Entre las novedades más destacadas se encuentra la integración de la *World Chain*, también una *wallet* [118] rediseñada que incorpora funcionalidades

innovadoras: la capacidad de realizar pagos globales gratuitos en segundos, nuevas rampas de entrada y salida para depósitos y retiros, un *Vault* que permite generar rendimientos sobre los activos almacenados, y herramientas para conectar a los usuarios con sus comunidades.

Además, la aplicación ahora soporta el almacenamiento opcional de credenciales derivadas de pasaportes físicos con tecnología *NFC*, permitiendo a los usuarios demostrar atributos como edad o nacionalidad sin comprometer su identidad real. También quieren incorporar *World ID Deep Face*: una funcionalidad que se espera esté disponible en el otoño de 2025, siendo una herramienta que podría permitir a los usuarios verificar la autenticidad de interacciones humanas en tiempo real, combatiendo el uso de *deep fakes* en videollamadas o chats. Asimismo, la inclusión de nombres de usuario y una pestaña de contactos refuerza la dimensión social de la aplicación, facilitando la interacción dentro de la comunidad *World* [117].

3.2.5 MiniApp

Gracias a la actualización de la *World App* 3.*0*, se incorpora una nueva innovación: las *MiniApp*. Estas representan una evolución significativa al introducir una plataforma que permite a aplicaciones de terceros ejecutarse directamente dentro de la aplicación principal, que están diseñadas para integrarse de manera anónima y heredar las características de *World* como son las credenciales de *World ID*, la *wallet* [118] y la lista de contactos del usuario. De esta manera se ofrecen experiencias optimizadas exclusivamente para humanos verificados, es decir que tengan su *World ID*. A diferencia de las *DApp Ethereum*, las *MiniApp* funcionan dentro del entorno controlado en la *World App*, proporcionando un nivel adicional de usabilidad y seguridad basada en la verificación de DID.

Las *MiniApp* abarcan casos de uso cotidianos que se podrían aprovechar de la infraestructura descentralizada de *World*. Por ejemplo, los usuarios pueden chatear y enviar dinero a amigos, recargar saldo telefónico con activos digitales, crear encuestas exclusivas para humanos verificados o participar en juegos con otros miembros de la red.

Desde la perspectiva de los desarrolladores, las *MiniApp* son aplicaciones *web* que utilizan un *SDK* personalizado proporcionado por *World*, lo que simplifica su creación y lanzamiento, ya que este kit de desarrollo permite una comunicación de manera fluida con las capacidades nativas de la *World App*, como el acceso a la *wallet* o las credenciales de *World ID* [117].

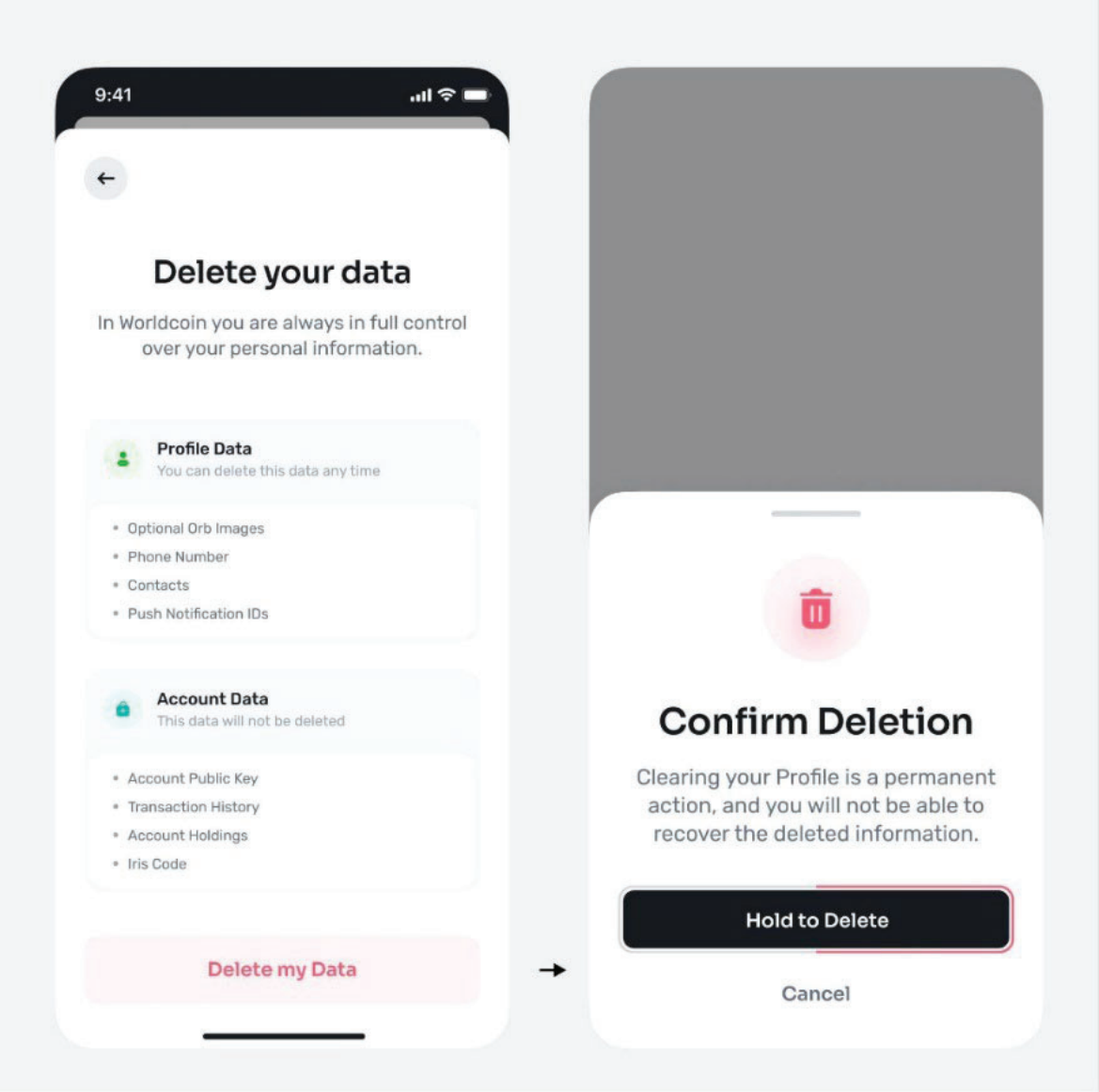

Figura 3.6 Borrar datos desde la World App

Fuente: https://whitepaper.world.org/images/whitepaper/notion2/image40.jpg

4. Herramientas utilizadas

Se describen las principales herramientas, lenguajes, *frameworks* y bibliotecas empleadas durante el desarrollo de la simulación *ULPGC Verify*, analizando tanto los entornos de desarrollo como los recursos proporcionados por *Worldcoin*, incluyendo el *SDK* de *World ID*, el *MiniKit* y las tecnologías auxiliares utilizadas para facilitar la conexión, almacenamiento y configuración de la aplicación.

4.1 Lenguajes y Frameworks

Para el desarrollo de *ULPGC Verify* se utilizaron principalmente lenguajes de programación y *frameworks* [119] enfocados al entorno *web*, seleccionados por su compatibilidad con el *SDK* de *World ID*, su eficiencia en la construcción de interfaces, y su integración con tecnologías actuales como *React* y *Next.js*. El lenguaje base es *TypeScript* [120] (extensión de *JavaScript*), que proporciona tipado estático, lo que permite desarrollar una aplicación más segura y escalable, especialmente al trabajar con respuestas estructuradas para los sistemas de verificación, que además se encuentra en la base de muchas de las herramientas utilizadas, como *React* [121]. Por ello, se emplea de forma implícita en diversos componentes y configuraciones. Estos lenguajes facilitan la construcción de una arquitectura orientada a componentes, integrando lógica de cliente y servidor en un único proyecto con herramientas como *Next.js* [122] *API Routes* y *React Hooks*.

4.1.1 JavaScript

JavaScript es un lenguaje de programación orientado a objetos, que permite implementar funciones para el desarrollo *web*. Se ejecuta directamente en el navegador, permitiendo crear aplicaciones interactivas y con capacidad de responder a eventos con el usuario, y es uno de los tres componentes fundamentales del desarrollo *web*, junto con *HTML* y *CSS*. Su modelo de ejecución basado en eventos y en un bucle de eventos no bloqueante, permite manejar múltiples operaciones asincrónicas mediante mecanismos como *callbacks*, *promesas* y la sintaxis *async*/*await*. Permite la manipulación del *Document Object Model* (*DOM*) [123], que es la estructura jerárquica de los elementos *HTML*, de esta manera puede actualizar de forma dinámica el contenido visual sin necesidad de recargar la página.

En *ULPGC Verify*, se implementa *JavaScript* para el entorno cliente como parte integral de la construcción de la interfaz mediante la biblioteca *React* [121], permitiendo una interacción fluida entre el usuario y el sistema. También se implementó en la definición de la lógica de eventos, gestionar el estado de la aplicación, validar entradas del usuario y comunicar con el *backend* a través de la *Fetch API*, utilizando solicitudes *HTTP* para enviar y recibir datos desde el servidor [124].

4.1.2 TypeScript

TypeScript permite detectar errores en el momento de su desarrollo y proporciona una mayor claridad en la definición en la estructuración de datos y funciones, declarando de forma explícitamente los tipos de datos, como *string*, *number* o estructuras complejas como *interfaces* y *enums*. Estas características aparte de prevenir errores comunes, también mejora la experiencia del desarrollador al integrar herramientas de autocompletado y documentación automática en los editores modernos.

En *ULPGC Verify*, se ha usado para definir de forma clara los tipos esperados en cada función y componente, por ejemplo, al recibir los resultados de verificación desde *World App*, se utiliza la interfaz *ISuccessResult* para asegurar que la estructura del dato retornado cumple con los campos esperados, de esta manera se reduce el riesgo de errores en tiempo de ejecución y facilita la depuración del sistema. La integración de *TypeScript* con *Next.js* es de forma directa, ya que el *framework* ofrece soporte nativo para este lenguaje, permitiendo de esta manera aprovechar sus ventajas sin necesidad de configuraciones adicionales, contribuyendo a una base de código más robusta y segura [120].

4.1.3 Next.js

Next.js es un *framework* de desarrollo *web* de código abierto basado en *React*, diseñado para facilitar la creación de aplicaciones rápidas y escalables, que es desarrollado por la empresa *Vercel*. Este *framework* proporciona funcionalidades como el *Server-Side Rendering* (*SSR*) [125], la *Static Site Generation* (*SSG*) [126], rutas de *API* integradas, optimización automática del rendimiento y compatibilidad con sistemas de diseño como *Tailwind CSS*. En *ULPGC Verify* se usa como la base de desarrollo, estructurando el sistema en diferentes niveles: la interfaz de usuario (*frontend*), la lógica de negocio, y los puntos de verificación del servidor (*API routes*), todo dentro de un único proyecto unificado.

Entre las principales ventajas de *Next.js* se encuentran su compatibilidad nativa con *TypeScript*, su sistema de rutas basado en archivos, y su enfoque híbrido que permite combinar *SSR* con *SSG* según las necesidades de cada componente. Esto facilita una carga rápida y mejora la indexación en buscadores. Su sistema de *hot reload* permite visualizar

en tiempo real los cambios durante el desarrollo. Para las necesidades específicas de *ULPGC Verify*, *Next.js* permite implementar un flujo seguro donde el usuario inicia una sesión de verificación desde el navegador, se redirige a *World App*, y posteriormente el sistema valida la prueba de identidad en el *backend* mediante una ruta como */api/verify* [122].

API Routes

Las *API Routes* son una funcionalidad integrada en *Next.js* que permite definir rutas del servidor y así poder manejar las peticiones *HTTP* dentro del mismo entorno de desarrollo, ya que con estas rutas actúan como funciones *serverless*. No requieren un servidor dedicado ni configuración adicional, y pueden procesar los datos, interactuando con la base de datos o comunicar a los servicios externos. Con cada archivo creado dentro de la carpeta */pages/api/* en un proyecto de *Next.js* se convierte automáticamente en un *endpoint* [127] que responde a las solicitudes *HTTP GET, POST*, entre otras, ya que estas funciones pueden exportarse de forma predeterminada utilizando la sintaxis de *Next.js*, y permiten construir fácilmente lógica del servidor sin salir del proyecto del *frontend*.

En *ULPGC Verify* se utiliza una *API Route* para definir la lógica de verificación de identidad, ya que con la ruta */api/verify* se recibe con una solicitud *POST* desde el cliente que contiene la prueba criptográfica generada por *World App*, y esta prueba se procesa en el servidor mediante la función *verifyCloudProof()*, y se devuelve una respuesta que indica si la verificación fue exitosa, de esta manera se divide la lógica permitiendo mantener de manera segura el proceso de validación, sin exponerlo en el código del cliente [128].

App Router

El *App Router* es el sistema de encaminamiento introducido a partir de la versión *Next.js* 13, que reemplaza al anterior sistema basado exclusivamente en la carpeta */pages/*. Con este nuevo enfoque se estructura a través de la carpeta */app/* y permitiendo definir rutas utilizando componentes de servidor (*Server Components*) y componentes de cliente (*Client Components*), de forma declarativa, flexible y orientado al control jerárquico de la interfaz. Su principal característica es la gestión de los *layouts* dinámicos, que permite definir estructuras persistentes entre páginas, como cabeceras o barras de navegación, sin la necesidad de repetir el código en cada componente. Introduce funciones como *loading.tsx*,

error.tsx y *not-found.tsx* de esta forma se gestionan los estados de carga y errores de manera nativa por ruta.

En *ULPGC Verify*, el uso del *App Router* permite organizar la aplicación en segmentos bien diferenciados, con un componente *Layout* general que integra el *MiniKitProvider*, y subcomponentes que gestionan vistas específicas como la página principal de verificación, facilitando esta manera la integración del flujo de autenticación, la reutilización de estructura visual y la separación de responsabilidades entre lógica de interfaz y lógica de servidor [129].

Server Components

Los *Server Components* son una funcionalidad introducida por *React* y adoptada en *Next.js* a través del *App Router*. Permite renderizar los componentes directamente en el servidor antes de enviarlos al cliente. A diferencia de los *Client Components*, estos no incluyen *JavaScript* adicional en el navegador, lo que reduce significativamente el tamaño del *bundle*, mejora el tiempo de carga y refuerza la seguridad, ya que la lógica sensible no se expone al cliente. Estos también pueden realizar operaciones como llamadas a bases de datos, lectura de archivos, consumo de las *API* o la ejecución de lógica que no requiere interacción directa con el usuario. De esta manera se ejecuta en el entorno del servidor, y el resultado renderizado se envía como *HTML* al navegador del usuario.

Para *ULPGC Verify* los *Server Components* se utilizan para cargar la estructura general como el *layout* principal, que envuelve toda la aplicación y proporciona contexto global, incluyendo la inicialización del *MiniKit* y también se aplican en la definición de rutas que no requieren interactividad inmediata, como aquellas que renderizan contenido estático o validado desde el *backend* [130].

Client Components

Los *Client Components* son componentes de *React*, que se ejecutan en el propio navegador del usuario, estos incluyen *JavaScript* en el cliente, lo que permite manejar interacciones dinámicas como formularios, eventos de click, gestión de estado, llamadas a las *API* desde el navegador y efectos secundarios mediante *hooks* como *useState* y *useEffect*. Para que

estos componentes sean reconocidos como cliente en el entorno de *Next.js* con *App Router* es necesario incluir al inicio del archivo la directiva *use client*. De esta manera se indica que el componente debe ser procesado y enviado como código ejecutable al navegador.

En *ULPGC Verify* los *Client Components* se implementan en la lógica interactiva del proceso de verificación, como el formulario de entrada del *ID* del estudiante, el botón de verificación y los mensajes de estado (*verificando*, *éxito*, *error*) definidos como *Client Components*, para reaccionar a la interacción del usuario y actualizar la interfaz en tiempo real según la respuesta del sistema, permitiendo aprovechar los *hooks de React*, facilitando la gestión del estado de la aplicación, la ejecución de efectos de los componentes y la comunicación con la *API Route* que valida las pruebas de identidad generadas por *World App* [131].

4.2 Bibliotecas de Interfaz de usuario

Las bibliotecas de interfaz de usuario utilizadas en el desarrollo de *ULPGC Verify* permiten construir y estructurar visualmente los componentes que forman la interfaz de usuario. Estas bibliotecas no solo aportan funcionalidad visual y diseño, sino que también permiten gestionar de forma eficiente la interactividad, la accesibilidad y el estilo de los elementos gráficos. La elección de estas herramientas se basa en su compatibilidad con *React*, su capacidad de integrarse con sistemas de diseño basados en utilidades como *Tailwind CSS*, y por la facilidad que tiene de construir componentes reutilizables, accesibles y personalizables [132].

4.2.1 React

React es una biblioteca de JavaScript de código abierto desarrollada por *Meta*, diseñada para construir interfaces de usuario dinámicas y basadas en componentes reutilizables. Está compuesta por una arquitectura basada en el *Virtual DOM* permitiendo actualizaciones eficientes en la interfaz de usuario, ya que solo se renderizan los componentes que han cambiado, lo que mejora significativamente el rendimiento en

aplicaciones interactivas y de gran escala. Uno de los principios fundamentales es el desarrollo mediante componentes, esto significa que la interfaz se divide en unidades lógicas independientes que pueden reutilizarse, gestionarse y probarse de forma aislada.

En *ULPGC Verify*, *React* se utiliza para construir toda la interfaz de usuario, ya que cada sección de la aplicación (como el ingreso del *Identifier* (*ID*) del estudiante, el botón de verificación o los mensajes de éxito/error) son componentes independientes, organizados jerárquicamente, con el uso de *hooks* como *useState* y *useEffect*, lo que es posible gestionar el estado de la aplicación y ejecutar efectos secundarios en función de las acciones del usuario y las respuestas del servidor [121].

React Hooks

Los React Hooks son una *API* introducida en la versión 16.8 de *React* que permite a los componentes funcionales acceder a las características avanzadas como el manejo de estado, el ciclo de vida del componente o la gestión de referencias, sin necesidad de utilizar componentes de clase. Esta funcionalidad representa un cambio significativo en la filosofía de desarrollo de *React*, ya que promueve una estructura más funcional, modular y fácil de mantener.

Ofrecen una forma coherente de reutilizar la lógica entre componentes. En lugar de depender de herencia o estructuras complejas, los desarrolladores pueden encapsular lógica en funciones independientes, mejorando la legibilidad del código y facilitando su testeo. Entre los *hooks* más utilizados se encuentran *useState* y *useEffect*, fundamentales en la mayoría de las aplicaciones modernas construidas con *React*.

- *useState*: permite declarar variables de estado dentro de componentes funcionales. Cada llamada a este *hook* retorna un par: el valor actual del estado y una función para actualizarlo. Es útil para almacenar datos dinámicos como formularios, autenticación, resultados de operaciones, entre otros.

- *useEffect*: se utiliza para manejar efectos secundarios, como llamadas a las *API*, suscripciones a servicios, temporizadores o inicialización de bibliotecas externas. Se ejecuta después del renderizado del componente y puede

configurarse para reaccionar a cambios en variables específicas o ejecutarse una sola vez al montar el componente [133].

4.2.2 shadcn/ui

shadcn/ui es una colección de componentes de interfaz de usuario reutilizables diseñada para ser utilizada con *React*, construida sobre la base de *Radix UI* y estilizada mediante clases utilitarias de *Tailwind CSS*. Proporciona los componentes prediseñados, como botones, formularios, pestañas, alertas, tarjetas, entre otros, todos ellos enfocados en la accesibilidad, la personalización y la coherencia visual. A diferencia de otras bibliotecas empaquetadas, *shadcn/ui* no se instala como una dependencia tradicional, sino que ofrece un sistema basado en la copia de componentes directamente al proyecto, permitiendo tener el control total sobre el código fuente de cada componente, lo que facilita su modificación y adaptación a las necesidades específicas de la aplicación que use esta biblioteca. Los desarrolladores pueden aplicar estilos personalizados directamente desde el *JavaScript XML* (*JSX*), sin necesidad de escribir *CSS* adicional.

En *ULPGC Verify*, *shadcn/ui* se utiliza para construir la mayor parte de la interfaz, incluyendo botones de acción, campos de entrada de datos, alertas de error, tarjetas de contenido y estructuras de navegación. Su uso permite mantener una estética coherente con el sistema visual, al mismo tiempo que se garantizaba accesibilidad y modularidad en cada componente [134].

4.2.3 Tailwind CSS

Tailwind CSS es un *framework* de diseño de código abierto basado en clases utilitarias, que permite aplicar estilos directamente en los elementos *HTML* o *JSX* sin necesidad de escribir hojas de estilo personalizadas. A diferencia de otros enfoques tradicionales de *CSS*, este promueve una metodología conocida como *utility-first*, en la que los elementos se construyen de manera que combina las clases específicas, definiendo sus propiedades individuales como colores, márgenes, tamaños de texto, bordes, entre otros. Tiene como principal ventaja la capacidad de acelerar el desarrollo de interfaces visuales, ya que

permite mantener el estilo visual directamente en el componente, de esta manera se reduce el tiempo de diseño, lo que evita el conflicto de los nombres o duplicación de reglas *CSS*. Su sistema de *responsive design* y su integración con *dark mode* están diseñados para funcionar de forma fluida en los proyectos actuales.

En *ULPGC Verify*, se utiliza *Tailwind CSS* para emplear el sistema principal de estilos y construir una interfaz limpia y coherente, facilitando de esta manera su desarrollo gracias a su integración con *Next.js* y *React*. Se aplican los estilos de forma personalizadas como los botones, formularios, alertas, iconos y tarjetas, sin la necesidad de escribir código *CSS* externo, de esta forma es bastante útil al trabajar con componentes de la biblioteca de *shadcn*/*ui*. Este ofrece compatibilidad con herramientas como *PostCSS* y sistemas de purgados automáticos que eliminan clases no utilizadas en producción, optimizando el rendimiento final de la aplicación y contribuye a reducir el tamaño del archivo *CSS* final, de esta forma se mejoran los tiempos de carga de la demo y su rendimiento en general [135].

4.2.4 Lucide React

Lucide React es una biblioteca de iconos *Scalable Vector Graphics* (*SVG*) desarrollada específicamente para ser utilizada con *React*, cuenta con el diseño de los iconos en *open-source* que ofrece un conjunto personalizable de símbolos y gráficos. Permite importar y utilizar iconos como componentes directamente dentro de archivos *JSX* o *TypeScript XML* (*TSX*), lo que simplifica su integración en aplicaciones *web*, tomando en cuenta que cada icono de *Lucide* es un componente individual de *React*, esto permite personalizar los atributos como el tamaño, el color, la opacidad o el estilo con las clases de *Tailwind CSS* u otras propiedades nativas.

Al ser una biblioteca de código abierto, se mantiene gracias a la comunidad de desarrolladores y ofrece una colección en constante evolución de iconos optimizados para el rendimiento *web*, asegurando compatibilidad con diferentes navegadores y dispositivos. Su uso se ha vuelto común en proyectos modernos por su facilidad de integración y la posibilidad de mantener la coherencia visual en la interfaz sin sacrificar rendimiento ni flexibilidad [136].

4.3 Componentes de World ID

El sistema de verificación de identidad descentralizado utilizado en *ULPGC Verify* se basa en los componentes proporcionados por el *SDK* oficial de *World ID*, concretamente a través del paquete *@worldcoin/minikit-js* [15]. Permite establecer una comunicación segura entre la aplicación *web* y *World App*, de esta manera se gestiona el proceso completo de verificación de forma criptográfica y privada con el usuario. Estos componentes que conforman este sistema abarcan tanto la inicialización del *SDK* como las órdenes para lanzar el flujo de verificación, los mecanismos para validar la prueba del lado del servidor y las estructuras de datos, que son necesarias para completar correctamente las respuestas. Cada componente cumple con una función específica dentro del flujo de autenticación sin custodia, garantizando de esta manera la identidad del usuario puede comprobarse sin necesidad de almacenar ni compartir información sensible.

4.3.1 World ID MiniKit

El *World ID MiniKit* es un *Software Development Kit* (*SDK*) oficial proporcionado por *Worldcoin*, el cual está diseñado para facilitar la integración de *MiniApp* dentro del sistema de *World App*, desarrollada por *Tools for Humanity*. Este kit permite que los desarrolladores puedan incorporar flujos de verificación de identidad que emplean pruebas criptográficas basadas en *ZKP*, diseñado para integrarse de forma eficiente con *frameworks* como *React* y *Next.js*. Desde el punto de vista funcional, proporciona una interfaz programable que permite invocar órdenes asíncronos como *verify()* [137], el cual inicia el proceso de autenticación del usuario a través de *World App*. El sistema devuelve una prueba firmada que puede ser verificada criptográficamente con el servidor mediante funciones como *verifyCloudProof()*. Con este mecanismo se comprueba que un usuario está verificado sin necesidad de conocer su identidad ni acceder a su información biométrica.

En *ULPGC Verify*, permite al usuario iniciar un proceso de verificación desde el navegador, continuar la autenticación en *World App* y retornar de forma segura con una prueba válida. Su inicialización se realiza mediante *MiniKit.install()*, y se puede gestionar

dentro del ciclo de vida de una aplicación mediante *hooks* como *useEffect*, permitiendo que el estado de verificación esté disponible en todos los componentes a través de un *context provider*. Todo ello optimizando tanto la experiencia del usuario como la arquitectura de la aplicación [15].

4.3.2 Software Development Kit oficial de World ID

Un *Software Development Kit* (*SDK*) es un conjunto de herramientas, bibliotecas, documentación y ejemplos que permite integrar una tecnología específica en sus aplicaciones. Simplifica el proceso de implementación de funcionalidades complejas al proporcionar una interfaz predefinida que abstrae los detalles técnicos subyacentes.

En *ULPGC Verify* se utilizó el *SDK* oficial de *World ID* proporcionado a través del paquete *@worldcoin/minikit-js*. Este contiene todos los elementos necesarios para conectar una aplicación *web* con *World App*, permitiendo al usuario iniciar un proceso de verificación de identidad, recibir una prueba criptográfica como respuesta y validarla de forma segura con el servidor[15].

4.3.3 Zero-Knowledge Proofs en World ID

Un usuario puede demostrar que está verificado en el sistema de *World ID* sin exponer su identidad, sus datos biométricos o cualquier otra información personal [21]. Las *ZKP* se generan automáticamente cuando el usuario completa el proceso de verificación en *World App*, ya que al estar integrada con el protocolo de *Worldcoin*, emite una prueba que contiene:

- Un *nullifier hash*: identificador único que impide el uso repetido de la misma prueba.
- Un *merkle root* y una *proof*: elementos criptográficos que permiten validar la pertenencia del usuario al conjunto de personas verificadas.

- Un *credential_type*: indica si la verificación fue realizada con el *Orb* o mediante un dispositivo.

Estos datos se envían al servidor a través de una *API Route*, que los valida mediante la función *verifyCloudProof()*. Para ello utiliza las claves públicas de *Worldcoin* y con el uso de *ZKP* se garantiza que el sistema de autenticación de las aplicaciones no almacene, ni transfieran la información personal de los usuarios, manteniendo la privacidad total en el proceso de acceso a servicios universitarios.

4.4 Herramientas y Utilidades

Se utiliza un conjunto de herramientas auxiliares para facilitar el despliegue, la persistencia de datos locales y la gestión de configuraciones sensibles. También permitió implementar un flujo de desarrollo ágil, así como cumplir con buenas prácticas de seguridad y mantenimiento de proyectos *web*. No forman parte del núcleo funcional de la aplicación; pero son esenciales para garantizar un entorno de desarrollo seguro, eficiente y bien estructurado.

4.4.1 ngrok

La *ngrok* es una herramienta que permite exponer un servidor local a Internet mediante la creación de un túnel seguro, es decir creando un subdominio mediante un *Domain Name System* (*DNS*) dinámico [138]. Es especialmente útil durante el desarrollo y las pruebas de aplicaciones *web* que necesitan ser accesibles externamente, sin necesidad de desplegar el proyecto en un entorno de producción [139].

En *ULPGC Verify* es necesario para generar una *Uniform Resource Locator* (*URL*) pública temporal, para poder enlazar de manera directa con el servidor local donde se ejecuta. Esto es ya que es necesario para que *World App* pueda comunicarse con el *backend* de la aplicación durante el proceso de verificación. La aplicación móvil requiere acceder a un *endpoint* accesible desde Internet para enviar la prueba generada por el usuario, permitiendo simular de forma realista el flujo de verificación desde un entorno de

desarrollo, sin necesidad de desplegar la aplicación en un servidor remoto. De esta manera se consigue hacer pruebas funcionales completas, comprobar la validez de las respuestas del servidor y ajustar el comportamiento del sistema antes de una posible implementación definitiva. La *URL* generada por *ngrok* fue almacenada localmente mediante *localStorage*. Esto permite su reutilización entre sesiones sin necesidad de reconfigurar manualmente cada vez que se reinicia el túnel [140].

4.4.2 localStorage

La *localStorage* es una *API* nativa del navegador que permite almacenar datos de forma persistente en el cliente, incluso después de cerrar o recargar la pestaña del navegador. Es decir, conservando la información hasta que el usuario la elimina manualmente o el código lo haga explícitamente. Los datos se almacenan en formato clave-valor y se accede a ellos mediante métodos como *setItem()*, *getItem()* y *removeItem()*. Esto permite usar estas funcionalidades para guardar configuraciones, *tokens*, preferencias o cualquier tipo de información que no sea crítica desde el punto de vista de la seguridad [141].

En *ULPGC Verify*, se utiliza para guardar la *URL* generada por *ngrok* durante la fase de desarrollo, permitiendo reutilizar automáticamente la misma *URL* al recargar la página o durante múltiples sesiones de prueba, evitando tener que copiarla manualmente cada vez que se reiniciaba el túnel. De esta manera facilita el flujo de verificación con *World App*, ya que garantiza que el *backend* sea accesible de forma continua mientras se hacen las pruebas a la aplicación [141].

4.4.3 Environment Variables

Las *Environment Variables* o variables de entorno se utilizan para almacenar configuraciones sensibles y parámetros que no deben estar codificados directamente en el repositorio del proyecto. En el desarrollo *web*, estas variables suelen guardarse en archivos con extensión (*.env*), que pueden contener datos como claves *API*, identificadores de cliente o las *URL* de servicios externos. Esto permite separar la configuración del código

fuente, mejorar la seguridad del proyecto y facilitar el despliegue en distintos entornos de desarrollo, pruebas o producción, sin necesidad de modificar el código [142].

En *ULPGC Verify* se usa para definir valores como el *action ID* de *World ID*, que se requiere para generar correctamente la señal de verificación desde *MiniKit*. Esto es ya que este valor se accede en tiempo de ejecución mediante funciones proporcionadas por *Next.js*, permitiendo inyectar variables de entorno en el código cliente o servidor según su prefijo. Lo que mantiene una arquitectura limpia y segura, evitando exponer configuraciones críticas en el repositorio o en el código compilado, facilitando la adaptación del sistema a distintos entornos de prueba sin necesidad de modificar manualmente el código fuente [143].

5. Análisis de la implementación de ULPGC Verify

En los ejemplos de implementación se describe el proceso de desarrollo e implementación de *ULPGC Verify*, explicando las fases de creación y configuración del proyecto, así como la estructura funcional de la aplicación, y detallando el proceso de ejecución y verificación del sistema en un entorno controlado, finalizando con una valoración técnica del rendimiento de la demo y su viabilidad en contextos reales.

5.1 Descripción, creación y configuración general del sistema

Como demostración práctica de la aplicación de la tecnología *World ID* en un entorno universitario, se desarrolló *ULPGC Verify*, con el objetivo de mostrar cómo funciona una *MiniApp*, mediante la implantación de un sistema de autenticación descentralizado y seguro, en este caso orientado a la *ULPGC*. *ULPGC Verify* reproduce como se iniciaría la sesión utilizando la identidad verificada a través de World ID. Todo ello para demostrar la viabilidad técnica de integrar este sistema a un servicio digital como seria el *Campus Virtual*, Biblioteca o gestión de tramites universitarios.

El desarrollo del proyecto comienza con la creación de una aplicación *Next.js* denominada *ulpgc-world-verify*, se realiza desde un terminal, utilizando la orden correspondiente para generar la estructura base del proyecto, como se muestra en la *Figura 5.1*, el orden crea el proyecto base utilizando *Next.js*.

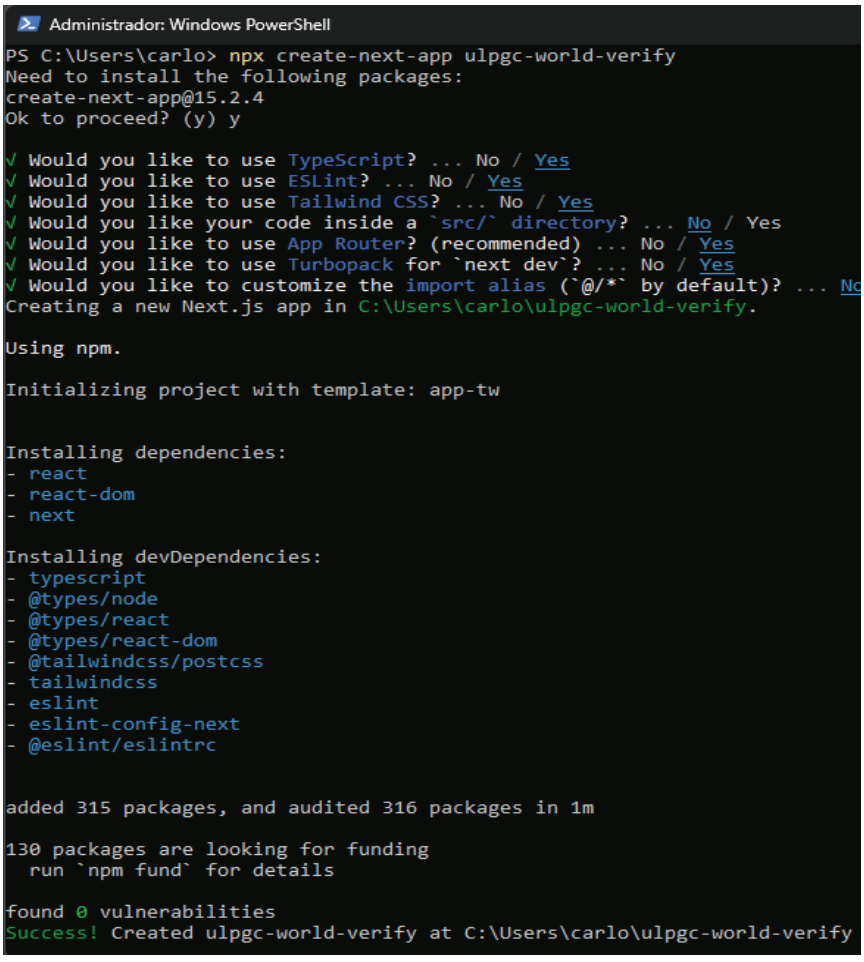

Figura 5.1. Orden de creación del proyecto Next.js ulpgc-world-verify

Figura 5.2. Orden de instalación MiniKit de SDK de World ID Mini-App

Una vez inicializado el entorno, el siguiente paso fue instalar el *SDK* de *World ID Mini-App* desde la terminal, el cual proporciona los componentes necesarios para habilitar la verificación de identidad desde aplicaciones *web*, utilizando la siguiente orden, como se muestra en la *Figura 5.2.*

A continuación, se instala la biblioteca de interfaz shadcn/ui desde la terminal, la cual facilita la incorporación de componentes reutilizables y estilizados en la aplicación, la orden correspondiente se muestra en la *Figura 5.3*.

```
PS C:\Users\carlo\ulpgc-world-verify> npx shadcn@latest init
√ Preflight checks.
√ Verifying framework. Found Next.js.
√ Validating Tailwind CSS config. Found v4.
√ Validating import alias.
√ Which color would you like to use as the base color? » Neutral
√ Writing components.json.
√ Checking registry.
√ Updating app\globals.css
  Installing dependencies.

It looks like you are using React 19.
Some packages may fail to install due to peer dependency issues in

√ How would you like to proceed? » Use --force
√ Installing dependencies.
√ Created 1 file:
  - lib\utils.ts

Success! Project initialization completed.
You may now add components.
```

Figura 5.3. Orden de instalación de componentes shadcn/ui

Posteriormente, también desde la terminal, se instalaron los componentes específicos necesarios para la interfaz de la *MiniApp*, tales como botones, tarjetas, campos de texto, etiquetas y alertas, las ordenes utilizadas para esta instalación se ilustran en la *Figura 5.4*.

```
PS C:\Users\carlo\ulpgc-world-verify> npx shadcn@latest add button card input label alert tabs
√ Checking registry.
  Installing dependencies.

It looks like you are using React 19.
Some packages may fail to install due to peer dependency issues in npm (see https://ui.shadcn.c

√ How would you like to proceed? » Use --force
√ Installing dependencies.
√ Created 6 files:
  - components\ui\button.tsx
  - components\ui\card.tsx
  - components\ui\input.tsx
  - components\ui\label.tsx
  - components\ui\alert.tsx
  - components\ui\tabs.tsx
```

Figura 5.4. Orden de instalación de componentes específicos shadcn/ui

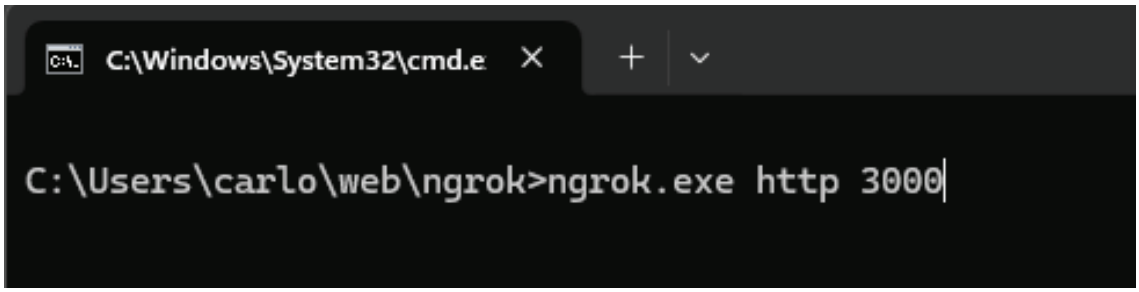

Figura 5.5. Orden de iniciación de ngrok al dominio localhost:3000

Con el entorno de desarrollo preparado, se procede a exponer localmente la aplicación mediante la herramienta *ngrok*, la cual se ejecuta desde la terminal y permite crear un túnel desde el entorno local hacia Internet. Esto es necesario porque el portal de desarrollo de *Worldcoin* exige que la aplicación esté accesible desde un dominio público para poder enlazarla con la *World App*, en la *Figura 5.5* se muestra la orden utilizada para iniciar *ngrok* sobre el *puerto 3000*.

Una vez iniciado *ngrok*, se genera un subdominio público temporal que permite acceder a la aplicación alojada localmente desde cualquier dispositivo con conexión a Internet. Se utiliza tanto para el registro de la *MiniApp* en el portal de desarrolladores de *Worldcoin* como para ejecutar la simulación desde un dispositivo móvil. La dirección generada se muestra en la *Figura 5.6.*

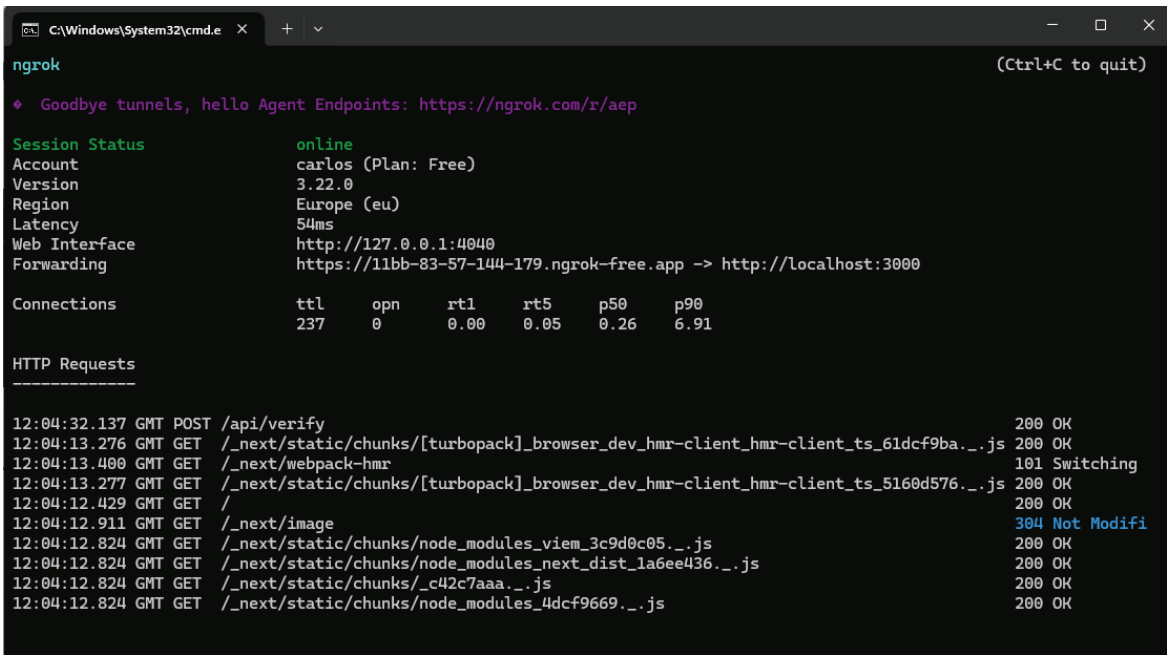

Figura 5.6. Resumen de ejecución de ngrok con el subdominio generado

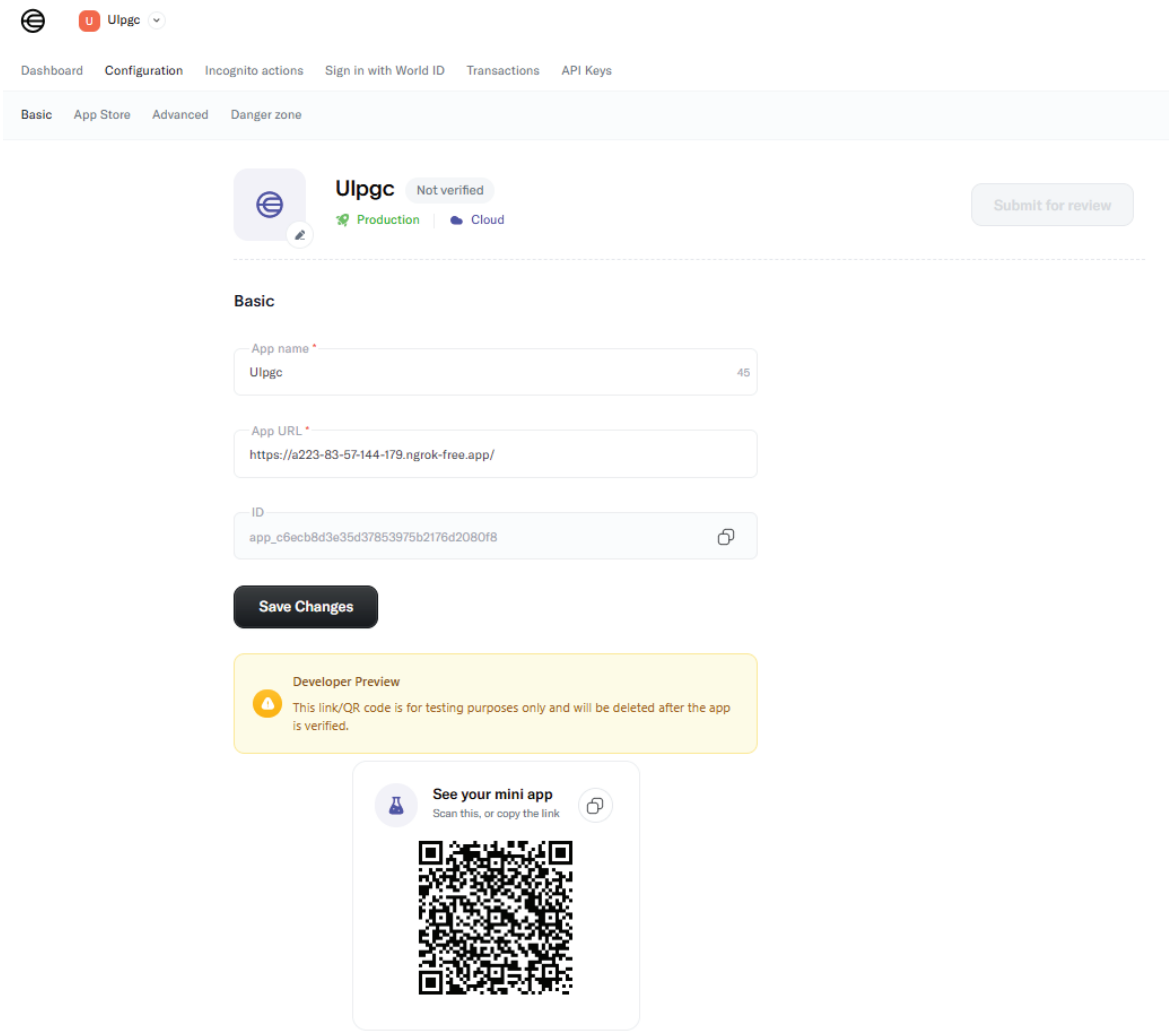

Figura 5.7. Portal web de World Developer con la App ULPGC

Desde el portal de desarrolladores de *Worldcoin* se procede al registro de la *MiniApp*. Se configuran parámetros esenciales como el nombre de la aplicación, una breve descripción y la *URL* de redirección generada por *ngrok*. Todo ello para poder enlazar la *MiniApp* con la *World App*, al finalizar el proceso, el sistema proporciona un identificador único de aplicación (*APP_ID*), el cual es necesario para habilitar la funcionalidad de verificación de usuarios, como se muestra en la *Figura 5.7*.

5.2 Estructura funcional

La aplicación se compone de distintos archivos, como se muestra en el repositorio de *Github* [144], cada uno con una funcionalidad específica dentro del proceso de verificación

de identidad. En la *Figura 5.8* se muestra cómo se configura la variable *APP_ID* en el archivo de entorno *.env.local* que contiene el identificador único de la aplicación proporcionado por el portal de desarrolladores de *Worldcoin*. Es necesario para poder enlazar correctamente la *MiniApp* con la *World App*.

El componente *MiniKitProvider.tsx*, como se muestra en la *Figura 5.9,* debe ser creado manualmente dentro del directorio *components*, y se encarga de inicializar el *SDK* del *MiniKit* de *Worldcoin*, permitiendo el uso de la *World ID* en la aplicación. Es importante señalar que, en esta fase de desarrollo, la aplicación funciona como una *demo* ya que el código contiene configuraciones propias de un entorno de pruebas. Esto es así ya que se utiliza el método *MiniKit.install()* sin especificar el *appId* directamente, lo cual es común en los desarrollos preliminares donde se espera que dicho valor sea recuperado desde una variable de entorno o que se utilice una configuración por defecto del *SDK*. Tampoco se implementa la lógica de gestión de sesiones ni se conecta a la base de datos institucional, permitiendo de esta manera, realizar pruebas reales de verificación con usuarios registrados en *World ID*, cumpliendo plenamente su propósito demostrativo.

En esta demo no se requiere que el usuario esté verificado mediante un dispositivo *Orb*, ya que al ser una *MiniApp* de prueba, el SDK permite hacer pruebas en modo desarrollo utilizando identificadores no verificados, siempre que se haya habilitado la opción desde el portal de desarrolladores. Al no requerirse esta verificación biométrica, cualquier usuario con la *World App* instalada puede acceder al proceso de autenticación de la *MiniApp*, lo cual simplifica el flujo de pruebas, pero también deja claro que no se trata de una aplicación final de producción.

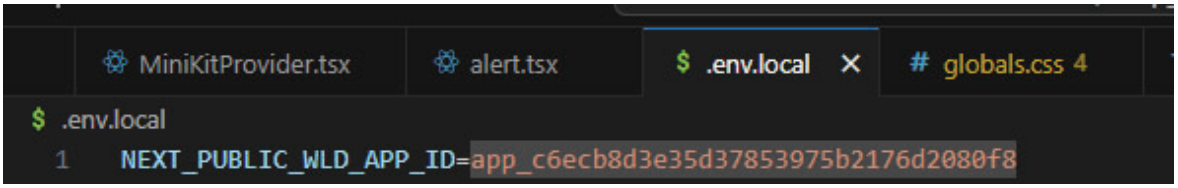

Figura 5.8. APP_ID en la variable de entorno .env.local

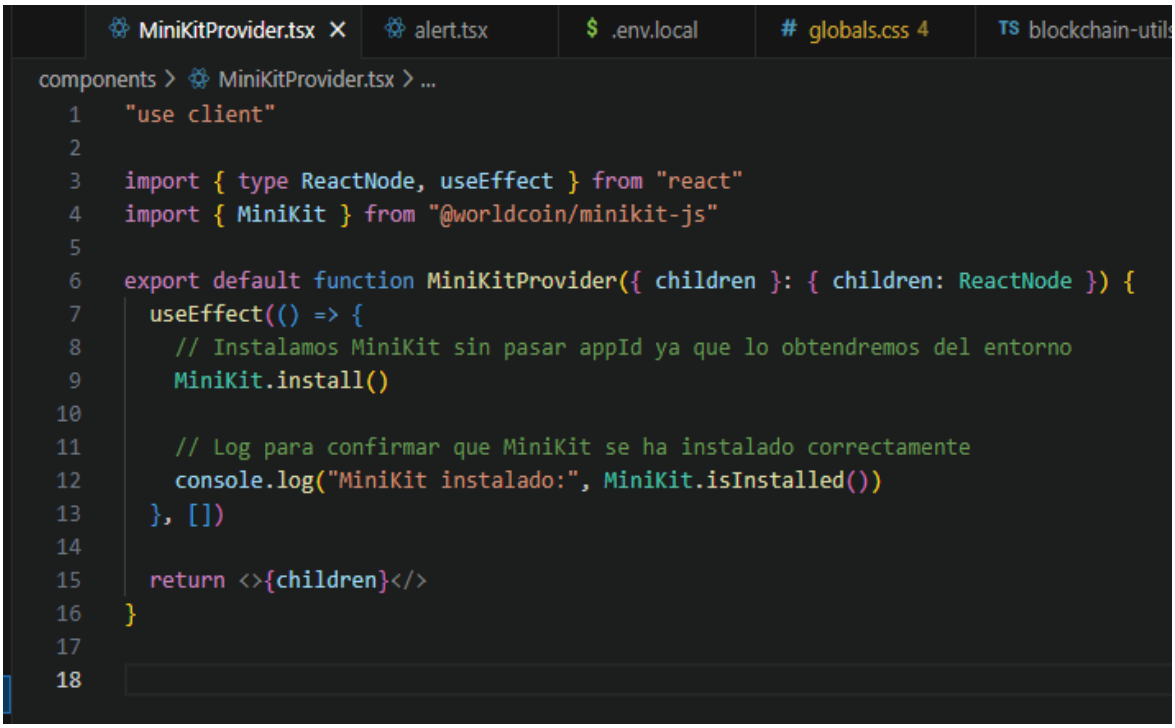

Figura 5.9. Código components/MiniKitProvider.tsx

Esta decisión se tomó de forma consciente con el objetivo de permitir la evaluación funcional de la aplicación sin depender del dispositivo *Orb*, aunque se quiera tener un usuario verificado con el *Orb*, no se pudiera porque gran parte de los países de la UE, incluyendo a España, tienen restringidos este dispositivo[145]. En el Anexo B, se describe el sistema biométrico de *Worldcoin*, aunque es muy relevante desde el punto de vista de seguridad, pero no forma parte del flujo de verificación implementado en esta simulación.

El archivo *layout.tsx*, como se muestra en la *Figura 5.10*, es una parte fundamental de la estructura de una aplicación creada con *Next.js* utilizando el sistema de encaminamiento basado en *App Router* para definir la estructura común que se aplica a todas las páginas del proyecto. Se incluyen metadatos básicos, hojas de estilo globales y proveedores de contexto. En este caso, toda la aplicación está compuesta por los componentes de *MiniKitProvider*. Esto permite que las funcionalidades del *SDK* de *World ID* estén disponibles de forma global en la interfaz, asegurando que cualquier componente hijo pueda acceder a las herramientas de verificación sin necesidad de repetir su configuración en cada página.

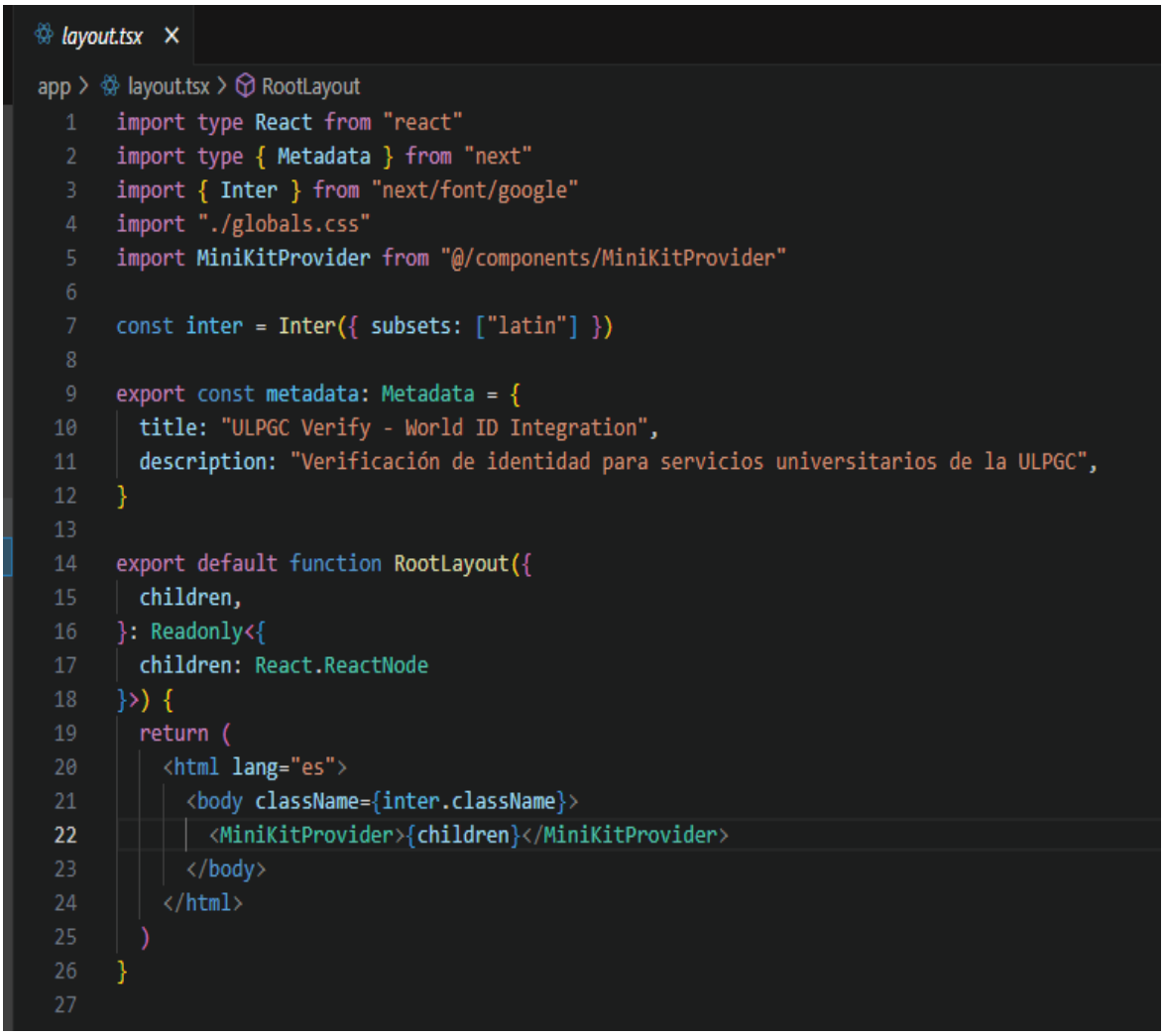

Figura 5.10. Código app/layout.tsx

El archivo *api/verify/route.ts*, como se muestra en la *Figuras* 5.*11*, la ruta *API* en Next.js. Funciona como el punto de entrada *backend* para procesar la verificación de identidad. Su función principal es recibir la prueba generada en el cliente tras la interacción con World *ID* y validarla mediante el método *verifyCloudProof* del *SDK* de *MiniKit*. Esta función verifica la autenticidad de la prueba criptográfica generada por la *World App*, asegurando que proviene de un usuario con una identidad registrada y verificada. Además, el archivo incluye una gestión básica de errores, retornando respuestas adecuadas en caso de que la prueba no sea válida o el proceso falle. Este *endpoint* es necesario para cerrar el ciclo de autenticación de forma segura en la simulación.

```typescript
import { type NextRequest, NextResponse } from "next/server"
import { verifyCloudProof, type IVerifyResponse, type ISuccessResult } from "@worldcoin/minikit-js"

interface IRequestPayload {
  payload: ISuccessResult
  action: string
  signal: string | undefined
}

export async function POST(req: NextRequest) {
  try {
    const { payload, action, signal } = (await req.json()) as IRequestPayload
    console.log("Verificando payload:", { action, signal })

    const app_id = process.env.NEXT_PUBLIC_WLD_APP_ID as `app_${string}`

    if (!app_id) {
      console.error("APP_ID no está configurado en las variables de entorno")
      return NextResponse.json({
        error: "Configuración de servidor incompleta",
        status: 500,
      })
    }

    try {
      // Verificar la prueba con la API de World ID
      const verifyRes = (await verifyCloudProof(payload, app_id, action, signal)) as IVerifyResponse

      console.log("Resultado de verificación:", verifyRes)

      if (verifyRes.success) {
        // Aquí puedes realizar acciones adicionales como registrar en blockchain
        return NextResponse.json({
```

(a)

```typescript
          verifyRes,
          status: 200,
          message: "Verificación exitosa",
        })
      } else {
        // Manejar errores de verificación
        return NextResponse.json({
          verifyRes,
          status: 400,
          message: verifyRes.error || "Error en la verificación",
        })
      }
    } catch (verifyError: any) {
      console.error("Error en verifyCloudProof:", verifyError)

      // Para propósitos de demostración, simulamos una verificación exitosa
      // En un entorno de producción, deberías manejar este error adecuadamente
      console.log("Simulando verificación exitosa para demostración")

      return NextResponse.json({
        status: 200,
        message: "Verificación simulada exitosa",
        simulatedVerification: true,
      })
    }
  } catch (error: any) {
    console.error("Error en la verificación:", error)
    return NextResponse.json({
      error: error.message || "Error interno del servidor",
      status: 500,
    })
  }
}
```

(b)

Figura 5.11. Código api/verify/route.ts: a) Validación con la API de World ID, b) Gestión de errores y verificaciones.

El archivo *page.tsx* es el componente principal de la interfaz de usuario, y su implementación completa se muestra en las *Figuras 5.12.a* a *Figura 5.12.d*, este componente es el responsable de renderizar las diferentes vistas de la aplicación según el estado del usuario, mostrando inicialmente un formulario donde se solicita al estudiante que introduzca su identificador, que sería el número de *DNI* sin letra y proceda a la verificación mediante *World ID*, una vez completado el proceso de autenticación, la interfaz cambia dinámicamente para dar la bienvenida al usuario verificado y mostrar un menú con accesos simulados a servicios institucionales como el *Campus Virtual*, la *Biblioteca* o los *Trámites*, ya que esta gestión condicional de la interfaz se realiza mediante estados locales.

En la *Figura 5.12.a* se muestra la declaración del componente *Home()* y la importación de múltiples elementos que forman parte de la interfaz y la lógica de *World ID*. Entre ellos el *useState*, que permite gestionar el estado local de variables como el *ID* del estudiante (*studentId*), el estado de verificación (*isVerified*), si se está procesando una verificación (*isVerifying*) y el mensaje de error (*error*). Estas variables se utilizan para implementar una lógica reactiva en la interfaz, mostrando las diferentes vistas dependiendo del progreso del usuario.

Se define también el método *handleVerify*, el cual encapsula el proceso de verificación mediante *World ID*. Este proceso comienza comprobando si la *World App* está instalada mediante *Minkit.isInstalled()*, antes de lanzar la solicitud de verificación.

A continuación, se construye el objeto *verifyPayload* con los parámetros de la verificación, especificando:

- *action*: nombre de la acción registrada (coherente con el backend).
- *signal*: el identificador del estudiante como dato vinculado a la verificación.
- *verification_level*: nivel de verificación permitido (en este caso, por dispositivo).

Este objeto llama al método *Minkit.commandsAsync.verify(),* que abre una ventana para que el usuario confirme la operación en la *App* de *World ID*.

En la *Figura 5.12.b* se controla la respuesta de la verificación. Si *finalPayload* es válido y contiene un mensaje de error. Si hay error, se muestra el error y se detiene el proceso. En caso contrario, se envía la verificación al *backend* mediante *fetch("/api/verify")*, utilizando una petición POST con la carga útil del resultado de *World ID*.

page.tsx

app > page.tsx > Home

```tsx
"use client"

import type React from "react"

import { useState } from "react"
import { MiniKit, type VerifyCommandInput, VerificationLevel, type ISuccessResult } from "@worldcoin/minikit-js"
import { Button } from "@/components/ui/button"
import { Card, CardContent, CardDescription, CardFooter, CardHeader, CardTitle } from "@/components/ui/card"
import { Alert, AlertDescription, AlertTitle } from "@/components/ui/alert"
import { Input } from "@/components/ui/input"
import { Label } from "@/components/ui/label"
import { CheckCircle2, Loader2 } from "lucide-react"
import Image from "next/image"

export default function Home() {
  const [studentId, setStudentId] = useState("")
  const [isVerified, setIsVerified] = useState(false)
  const [isVerifying, setIsVerifying] = useState(false)
  const [error, setError] = useState<string | null>(null)
  const [isLoggedIn, setIsLoggedIn] = useState(false)

  // Configuración para la verificación de World ID
  const handleVerify = async () => {
    if (!MiniKit.isInstalled()) {
      setError("World App no está instalada o no está disponible")
      return
    }

    try {
      setIsVerifying(true)
      setError(null)

      // Configuración de la verificación
      const verifyPayload: VerifyCommandInput = {
        action: "ulpgc-verification", // ID de acción registrado en el Portal de Desarrolladores
        signal: studentId, // Usamos el ID de estudiante como señal
        verification_level: VerificationLevel.Device, // Permitimos verificación por dispositivo
      }

      console.log("Iniciando verificación con:", verifyPayload)

      // World App abrirá un drawer para que el usuario confirme la operación
      const { finalPayload } = await MiniKit.commandsAsync.verify(verifyPayload)

      if (finalPayload.status === "error") {
        console.error("Error en la verificación:", finalPayload)
        // Accedemos a la información de error de manera segura
        const errorMessage =
```

(a)

```
            typeof finalPayload === "object" && finalPayload !== null && "message" in finalPayload
              ? finalPayload.message
              : "Error desconocido"
          setError(`Error en la verificación: ${errorMessage}`)
          setIsVerifying(false)
          return
        }

        console.log("Verificación exitosa, enviando al backend:", finalPayload)

        // Verificar la prueba en el backend
        const verifyResponse = await fetch("/api/verify", {
          method: "POST",
          headers: {
            "Content-Type": "application/json",
          },
          body: JSON.stringify({
            payload: finalPayload as ISuccessResult,
            action: "ulpgc-verification",
            signal: studentId,
          }),
        })

        const verifyResponseJson = await verifyResponse.json()

        console.log("Respuesta del backend:", verifyResponseJson)

        if (verifyResponseJson.status === 200 || verifyResponseJson.simulatedVerification) {
          console.log("Verificación confirmada por el backend")
          setIsVerified(true)
        } else {
          setError(`Error en la verificación: ${verifyResponseJson.message || "Respuesta inválida del servidor"}`)
        }
      } catch (err: any) {
        console.error("Error durante la verificación:", err)
        setError(err.message || "Error durante el proceso de verificación")
      } finally {
        setIsVerifying(false)
      }
    }

    // Manejar el inicio de sesión
    const handleLogin = (e: React.FormEvent) => {
      e.preventDefault()

      if (!isVerified) {
        handleVerify()
        return
      }

      // Si ya está verificado, permitir el acceso
      setIsLoggedIn(true)
    }

```

(b)

```
103   // Pantalla de bienvenida después de iniciar sesión
104   if (isLoggedIn) {
105     return (
106       <main className="flex min-h-screen flex-col items-center justify-center p-4 bg-gray-50">
107         <Card className="w-full max-w-md">
108           <CardHeader className="bg-gradient-to-r from-blue-700 to-blue-900 text-white rounded-t-lg">
109             <div className="flex justify-center mb-4">
110               <Image src="/ulpgc-logo.png" alt="ULPGC Logo" width={150} height={40} className="object-contain" />
111             </div>
112             <CardTitle className="text-center">Bienvenido al Campus Virtual</CardTitle>
113             <CardDescription className="text-blue-100 text-center">
114               Has accedido correctamente a los servicios de la ULPGC
115             </CardDescription>
116           </CardHeader>
117           <CardContent className="pt-6">
118             <div className="p-3 bg-green-50 border border-green-200 rounded-md">
119               <h3 className="text-sm font-medium text-green-800 flex items-center gap-1">
120                 <CheckCircle2 className="h-4 w-4" />
121                 Identidad verificada con World ID
122               </h3>
123               <p className="text-xs text-green-700 mt-1">ID de estudiante: {studentId}</p>
124             </div>
125
126             <div className="mt-6 space-y-4">
127               <h3 className="font-medium text-center">Servicios disponibles</h3>
128               <div className="grid grid-cols-2 gap-3">
129                 <Button variant="outline" className="h-20 flex flex-col">
130                   <span className="text-sm font-medium">Campus Virtual</span>
131                   <span className="text-xs text-gray-500">Accede a tus cursos</span>
132                 </Button>
133                 <Button variant="outline" className="h-20 flex flex-col">
134                   <span className="text-sm font-medium">Biblioteca</span>
135                   <span className="text-xs text-gray-500">Recursos digitales</span>
136                 </Button>
137                 <Button variant="outline" className="h-20 flex flex-col">
138                   <span className="text-sm font-medium">Expediente</span>
139                   <span className="text-xs text-gray-500">Consulta tus notas</span>
140                 </Button>
141                 <Button variant="outline" className="h-20 flex flex-col">
142                   <span className="text-sm font-medium">Trámites</span>
143                   <span className="text-xs text-gray-500">Gestiones administrativas</span>
144                 </Button>
145               </div>
146             </div>
147           </CardContent>
148           <CardFooter>
149             <Button className="w-full bg-blue-700 hover:bg-blue-800" onClick={() => setIsLoggedIn(false)}>
150               Cerrar sesión
151             </Button>
152           </CardFooter>
153         </Card>
154       </main>
155     )
156   }
157
```

(c)

```
// Pantalla de inicio de sesión
return (
  <main className="flex min-h-screen flex-col items-center justify-center p-4 bg-gray-50">
    <Card className="w-full max-w-md">
      <CardHeader className="bg-gradient-to-r from-blue-700 to-blue-900 text-white rounded-t-lg">
        <div className="flex justify-center mb-4">
          <Image src="/ulpgc-logo.png" alt="ULPGC Logo" width={150} height={40} className="object-contain" />
        </div>
        <CardTitle className="text-center">ULPGC Verify</CardTitle>
        <CardDescription className="text-blue-100 text-center">
          Accede a los servicios universitarios con World ID
        </CardDescription>
      </CardHeader>
      <CardContent className="pt-6">
        <form onSubmit={handleLogin} className="space-y-4">
          <div className="space-y-2">
            <Label htmlFor="studentId">ID de Estudiante</Label>
            <Input
              id="studentId"
              value={studentId}
              onChange={(e) => setStudentId(e.target.value)}
              placeholder="Ej: 42512345"
              required
            />
          </div>

          {error && (
            <Alert variant="destructive">
              <AlertTitle>Error</AlertTitle>
              <AlertDescription>{error}</AlertDescription>
            </Alert>
          )}

          {isVerified && (
            <div className="p-3 bg-green-50 border border-green-200 rounded-md">
              <h3 className="text-sm font-medium text-green-800 flex items-center gap-1">
                <CheckCircle2 className="h-4 w-4" />
                Identidad verificada
              </h3>
            </div>
          )}

          <Button type="submit" className="w-full bg-blue-700 hover:bg-blue-800" disabled={isVerifying || !studentId}>
            {isVerifying ? (
              <>
                <Loader2 className="mr-2 h-4 w-4 animate-spin" />
                Verificando...
              </>
            ) : isVerified ? (
              "Acceder a ULPGC"
            ) : (
              "Verificar con World ID"
            )}
          </Button>
        </form>
      </CardContent>
      <CardFooter className="flex flex-col text-center text-sm text-gray-500">
        <p>Esta mini-app utiliza World ID para verificar la identidad de los estudiantes y trabajadores de forma segura y privada.</p>
      </CardFooter>
    </Card>


  </main>
)
}
```

(d)

Figura 5.12. Código app/page.tsx: a) Declaración de componentes Home() y configuración inicial de estados, b) Proceso de verificación con World App y envío de datos al backend, c) Interfaz de bienvenida tras la verificación exitosa, d) Pantalla inicial login con formulario de introducción de ID de estudiante y mensaje de estado

El *backend* se encarga de validar nuevamente la prueba recibida. Si todo es correcto, se actualiza el estado *isVerified* a true. Esta lógica describe la teoría explicada en capítulos

anteriores sobre la verificación descentralizada mediante *ZKP*, donde el *frontend* genera la prueba y el *backend* es la que valida antes de conceder el acceso a los recursos.

En la *Figura 5.12.c se* muestra la vista que se renderiza cuando el usuario ya ha sido verificado exitosamente. La variable *isLoggedIn* pasa a true, lo que activa la visualización de una pantalla de bienvenida, mostrando un mensaje de confirmando la verificación con *World ID*. En esta vista se presentan los botones que simulan el acceso a los servicios de la ULPGC como el Campus Virtual, la Biblioteca o el Expediente. Utilizando las bibliotecas *shadcn/ui* y *Tailwind CSS* para componer la interfaz visual, con los elementos *Card*, *CardTitle*, *CardContent* y *Button*.

En la *Figura 5.12.d* se incluye la parte inicial de la aplicación, que es el formulario donde el usuario introduce su *ID de estudiante*. Este *ID* se utiliza como *signal* en la solicitud de verificación de *World ID*, funcionando como un identificador lógico para cada verificación. En el campo *Input* está enlazado mediante el *hook useState* al valor *studentId*, y cualquier cambio que el usuario haga se actualiza dinámicamente mediante *onChange*, lo que demuestra una aplicación correcta a *data binding*.

Además, se finaliza el componente *Home()* con la sección correspondiente a la gestión visual del estado de verificación del usuario. Se muestra un componente *<Alert>* en caso de que exista un error durante el proceso, lo cual está vinculado al estado error manejado previamente mediante el *hook useState*. Permitiendo notificar al usuario de manera visual algún fallo, como la imposibilidad de verificar el World ID o una respuesta inválida del backend.

El estado *isVerified* si es verdadero, se muestra un mensaje de validación. En el botón de envío *<button type="submit">* implementa una lógica condicional basada en el estado *isVerifying* y si el *studentId* está vacío. De esta forma se evita que el usuario pueda lanzar múltiples veces el proceso de verificación o lo intente sin haber introducido un *ID*. Dentro del botón se presenta una animación de carga *<Loader className="animate-spin"/>* que indica al usuario que la verificación está en curso.

Una vez verificado, el texto del botón cambia dinámicamente de *Verificar con World ID* a *Acceder a ULPGC*, demostrando el uso de renderizado condicional basado en el estado de la aplicación.

5.3 Ejecución y verificación de la simulación

La aplicación se ejecuta en un entorno local utilizando el servidor de desarrollo integrado de *Next.js*, el cual se lanza comúnmente mediante la orden *npm run dev* que inicia la aplicación en el *puerto 3000* del entorno local, compilando los archivos necesarios y habilitando la recarga automática. Para poder acceder a esta aplicación desde un dispositivo móvil, se utiliza el subdominio generado previamente por *ngrok*, que actúa como un túnel seguro desde Internet hacia el servidor local, permitiendo simular un entorno de producción sin necesidad de desplegar la aplicación en un hosting externo. En la *Figura 5.13* se muestra cómo se inicia el servidor de desarrollo y se prepara la aplicación para pruebas reales.

```
PS C:\Users\carlo\ulpgc-world-verify> npm run dev

> ulpgc-world-verify@0.1.0 dev
> next dev --turbopack

   ▲ Next.js 15.2.4 (Turbopack)
   - Local:        http://localhost:3000
   - Network:      http://192.168.1.38:3000
   - Environments: .env.local

 ✓ Starting...
 ✓ Ready in 2.1s
 o Compiling / ...
 ✓ Compiled / in 4.8s
 GET / 200 in 5444ms
 o Compiling /favicon.ico ...
 ✓ Compiled /favicon.ico in 847ms
 GET /favicon.ico?favicon.45db1c09.ico 200 in 983ms
 GET / 200 in 204ms
 GET /favicon.ico?favicon.45db1c09.ico 200 in 137ms
```

Figura 5.13. Orden de ejecución de la aplicación

Primero debemos escanear el código *QR* generado por el portal de desarrollo de *Worldcoin* utilizando la aplicación *World App* instalada en el dispositivo móvil, ya que al escanear el código *QR*, nos redirige hacia la *MiniApp* simulada *ULPGC Verify*, cargándola desde el subdominio proporcionado por *ngrok*, como se muestra en la *Figura 5.14.*

Una vez abierta la *MiniApp*, se presenta la interfaz de inicio, donde se invita al usuario a identificarse, donde se muestran los campos de entrada y un botón de verificación, como se muestra la pantalla principal en la *Figura 5.15*.

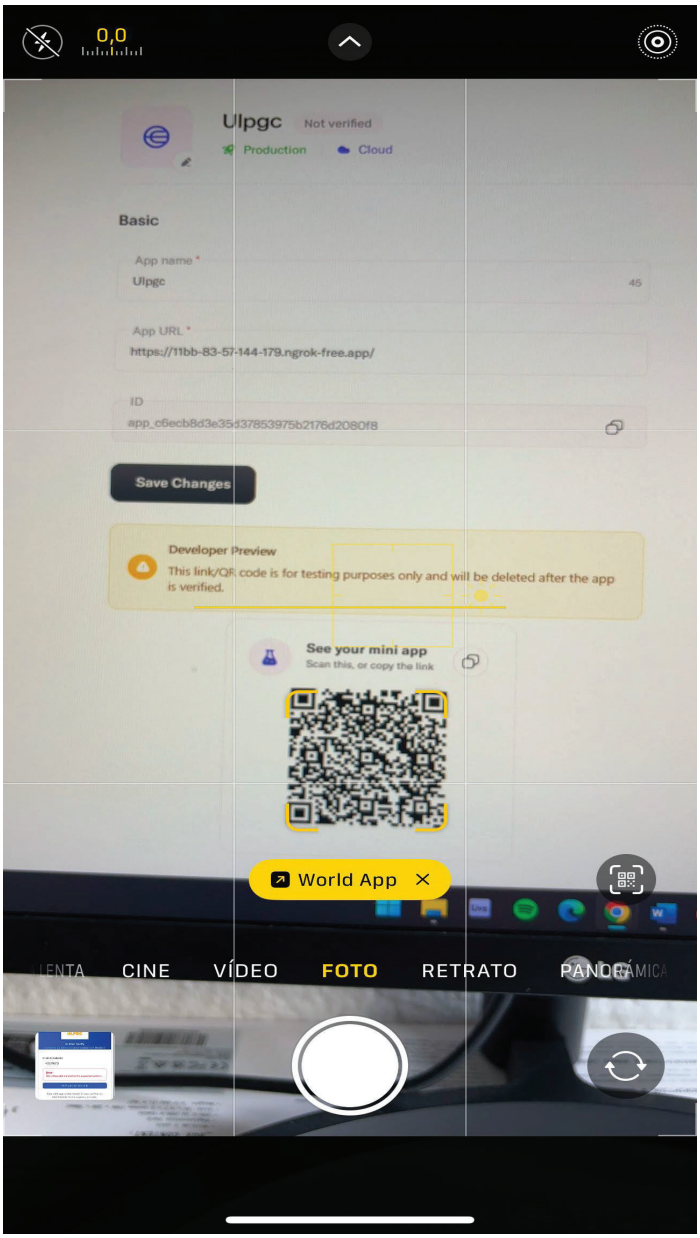

Figura 5.14. Escaneo de código QR del portal de nuestra aplicación para redirigir a la World APP descarga en nuestro móvil

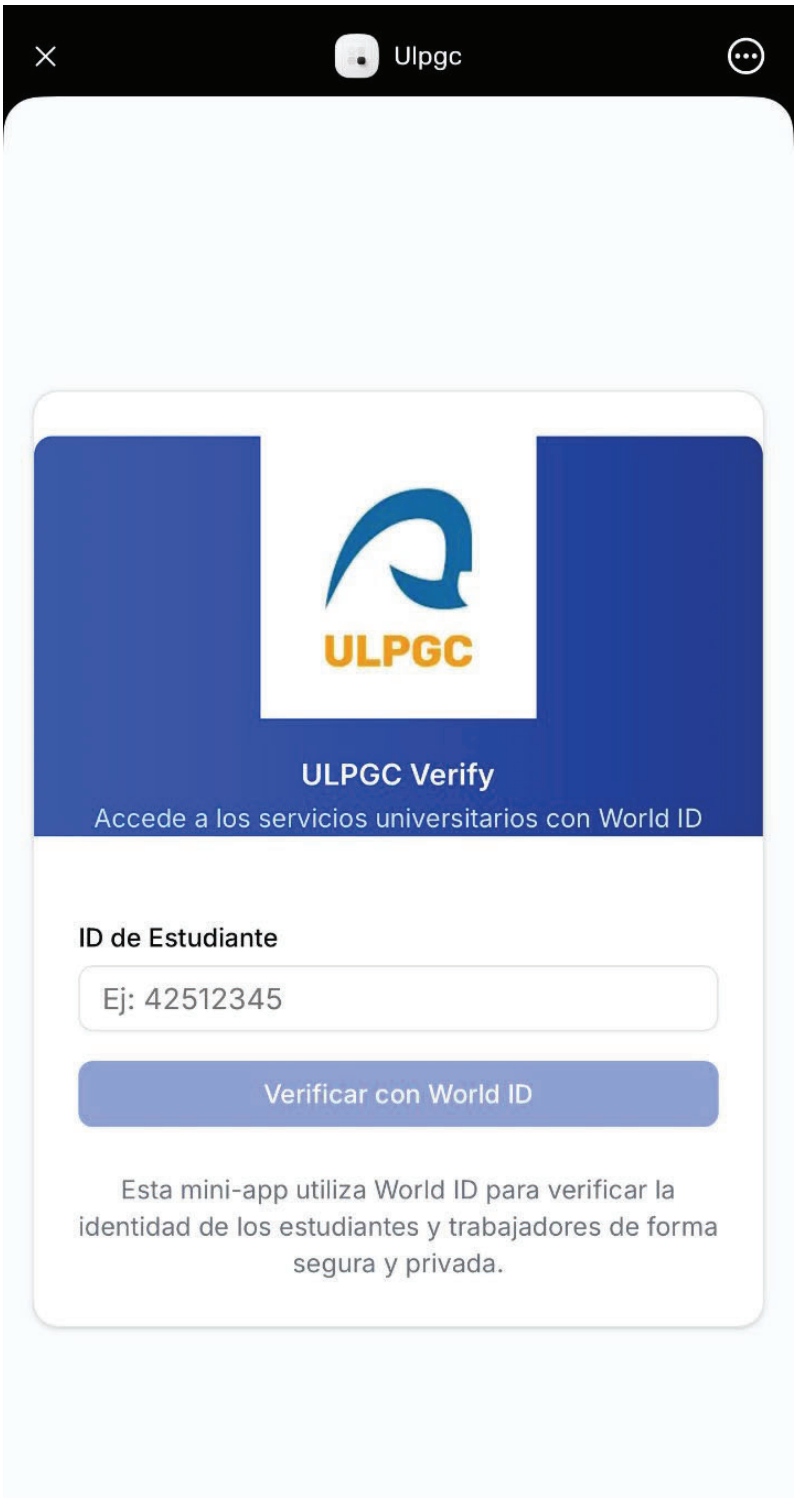

Figura 5.15. Interfaz de página de inicio de App ULPGC Verify

En esta fase, el usuario debe introducir su número de *DNI sin letra*, como se haría normalmente en el portal institucional de la *ULPGC* y posteriormente, hacer clic en el botón *Verificar con World ID*, que inicia el proceso de autenticación a través de la *World App*, como se muestra en la *Figura 5.16*.

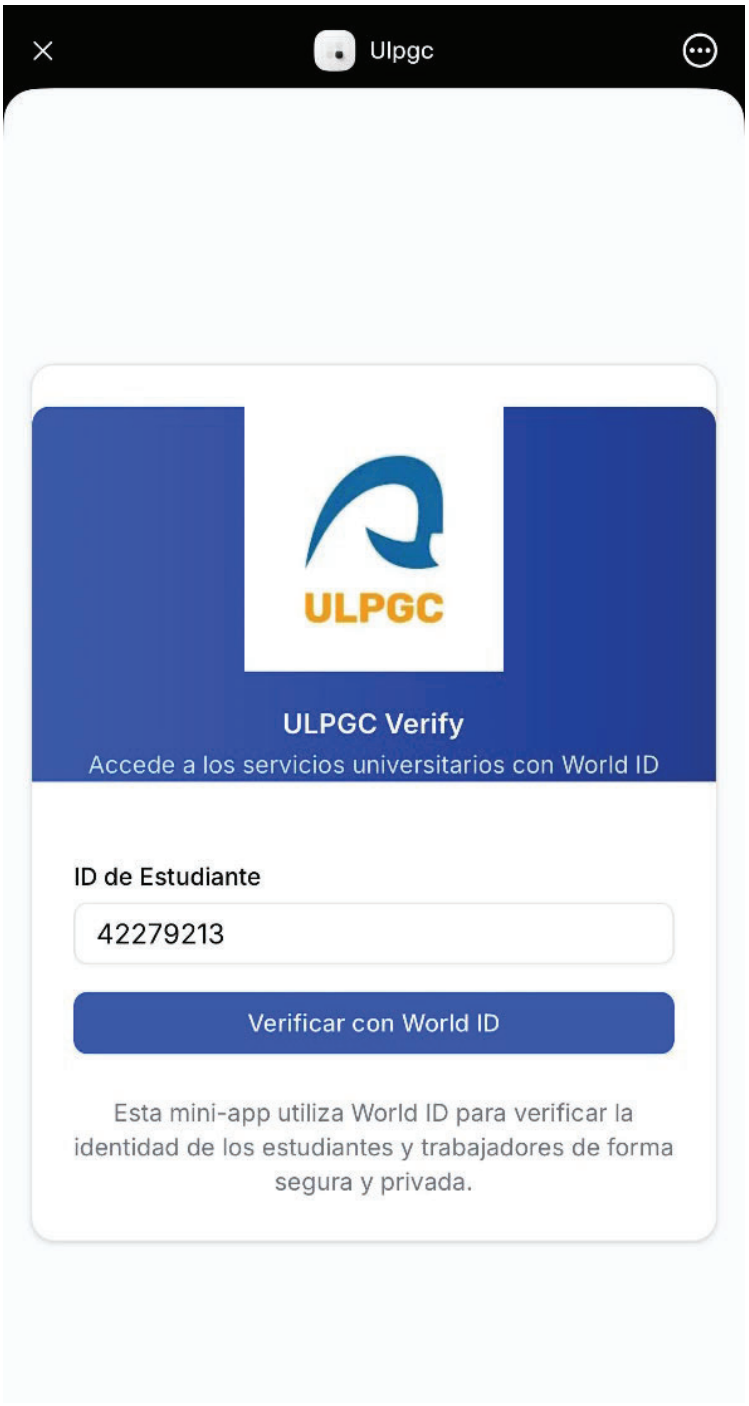

Figura 5.16. Interfaz de ingreso ID del estudiante y verificando con el botón verificar con World ID

Tras introducir el *DNI* y hacer clic en el botón de verificación, se solicita al usuario que confirme su identidad mediante el botón *Comprobación* dentro de la *World App*, de esta forma se activa la generación de una prueba criptográfica de identidad, utilizando los mecanismos de verificación de *Worldcoin*, como se muestra en la *Figura 5.17*.

Figura 5.17. Interfaz de solicitud de verificación mediante el botón Comprobación

Una vez iniciado el proceso, la aplicación entra en un estado de espera mientras se valida la prueba enviada por el dispositivo. Durante este tiempo, el sistema se comunica con los servidores de verificación de *Worldcoin* para confirmar que la identidad del usuario es válida y está registrada correctamente, como se muestra en la *Figura 5.18*.

Figura 18. Interfaz de espera mientras se verifica la World ID

Finalmente, si la prueba es válida, la aplicación muestra una pantalla de confirmación que indica que el usuario ha sido verificado correctamente mediante *World ID* y permite continuar con el flujo de navegación dentro de la *MiniApp*, como se muestra en la *Figura 5.19*.

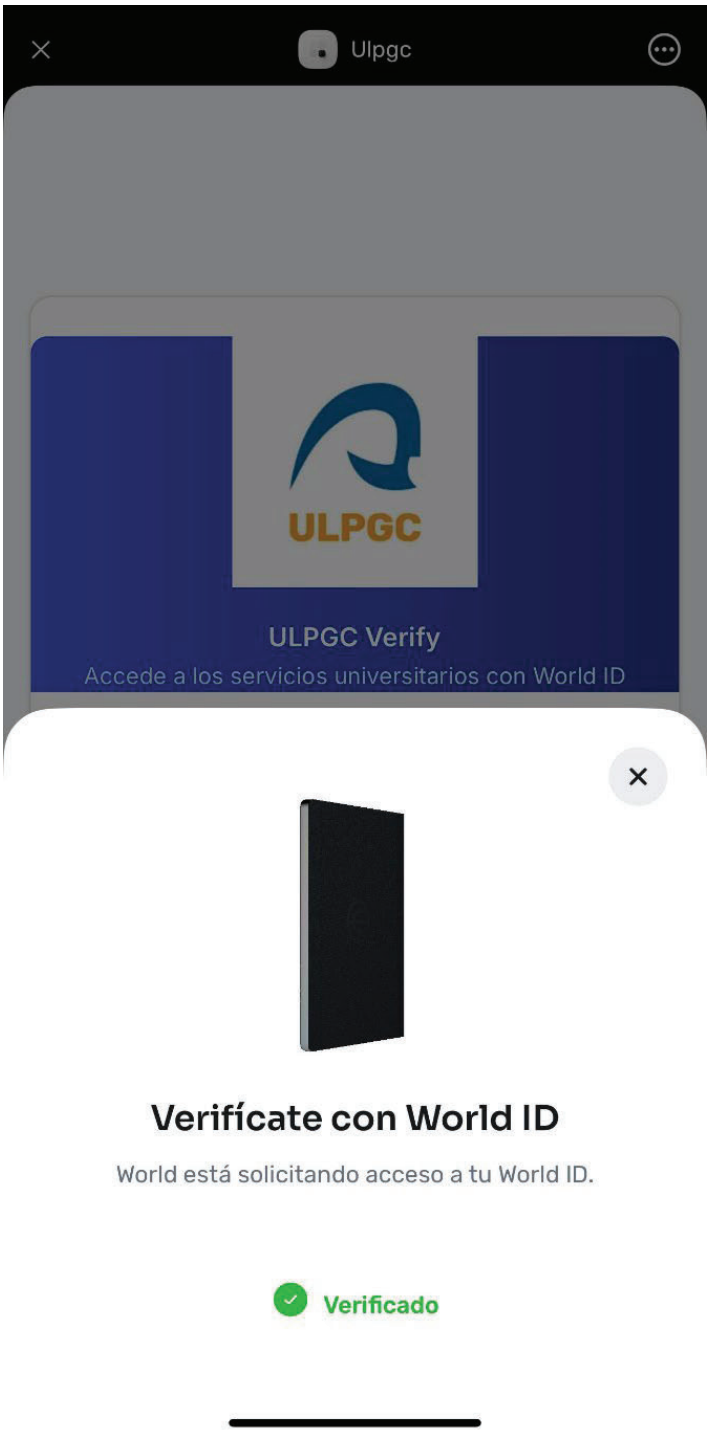

Figura 5.19. Interfaz del sistema que ya ha verificado la World ID

Una vez que el usuario ha sido verificado correctamente mediante *World ID*, la interfaz muestra un nuevo botón identificado como *Acceder a ULPGC*, que representa el paso final del proceso de autenticación, permitiendo al usuario acceder a los recursos simulados de la universidad como si estuviera dentro de un entorno institucional real, como se muestra en la *Figura 5.20.*

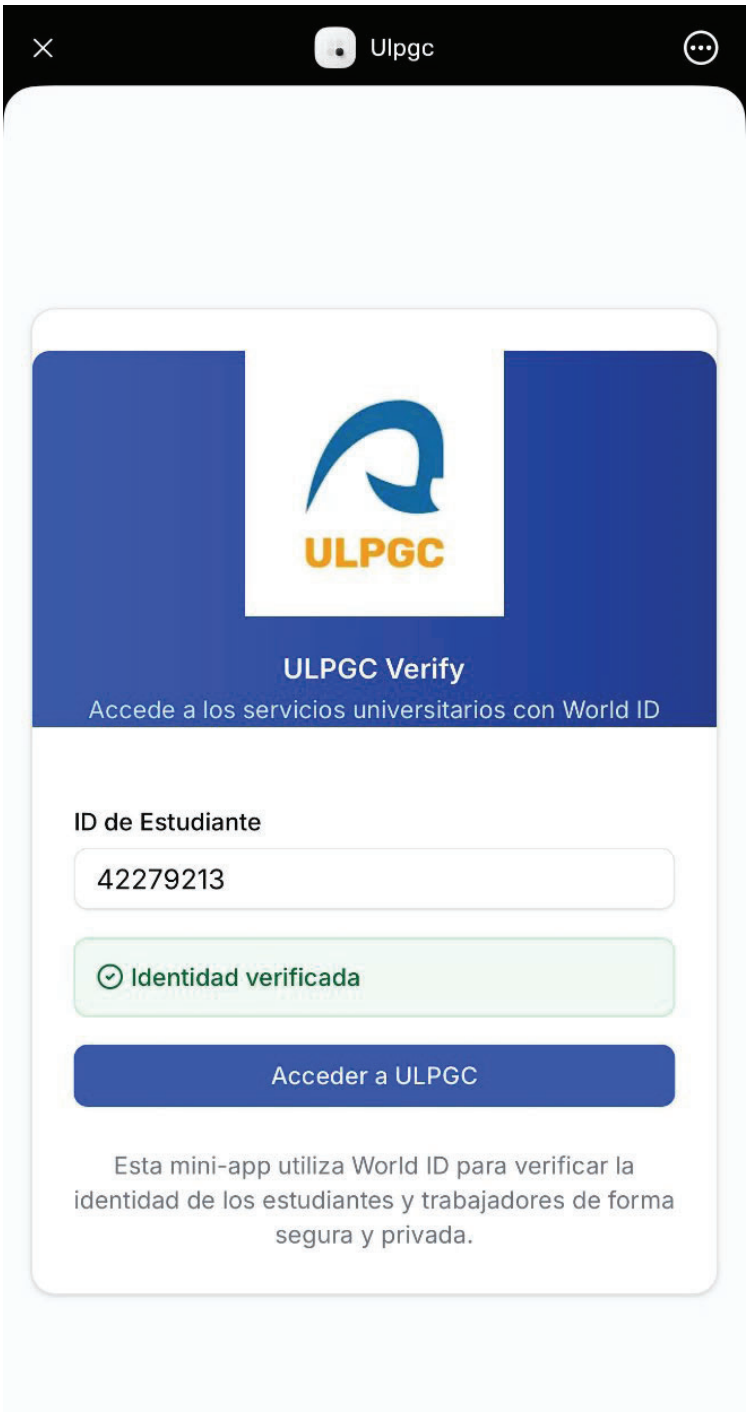

Figura 5.20. Interfaz del sistema de identidad verificada y botón de Acceder a ULPGC

Al pulsar el botón, se despliega el menú principal de la *MiniApp*, donde el usuario puede visualizar diferentes accesos simulados a servicios comunes dentro del sistema universitario, tales como el *Campus Virtual*, la Biblioteca, el Expediente Académico o el área de Trámites. Esta vista representa el estado final de la simulación, como se muestra en la *Figura 5.21.*

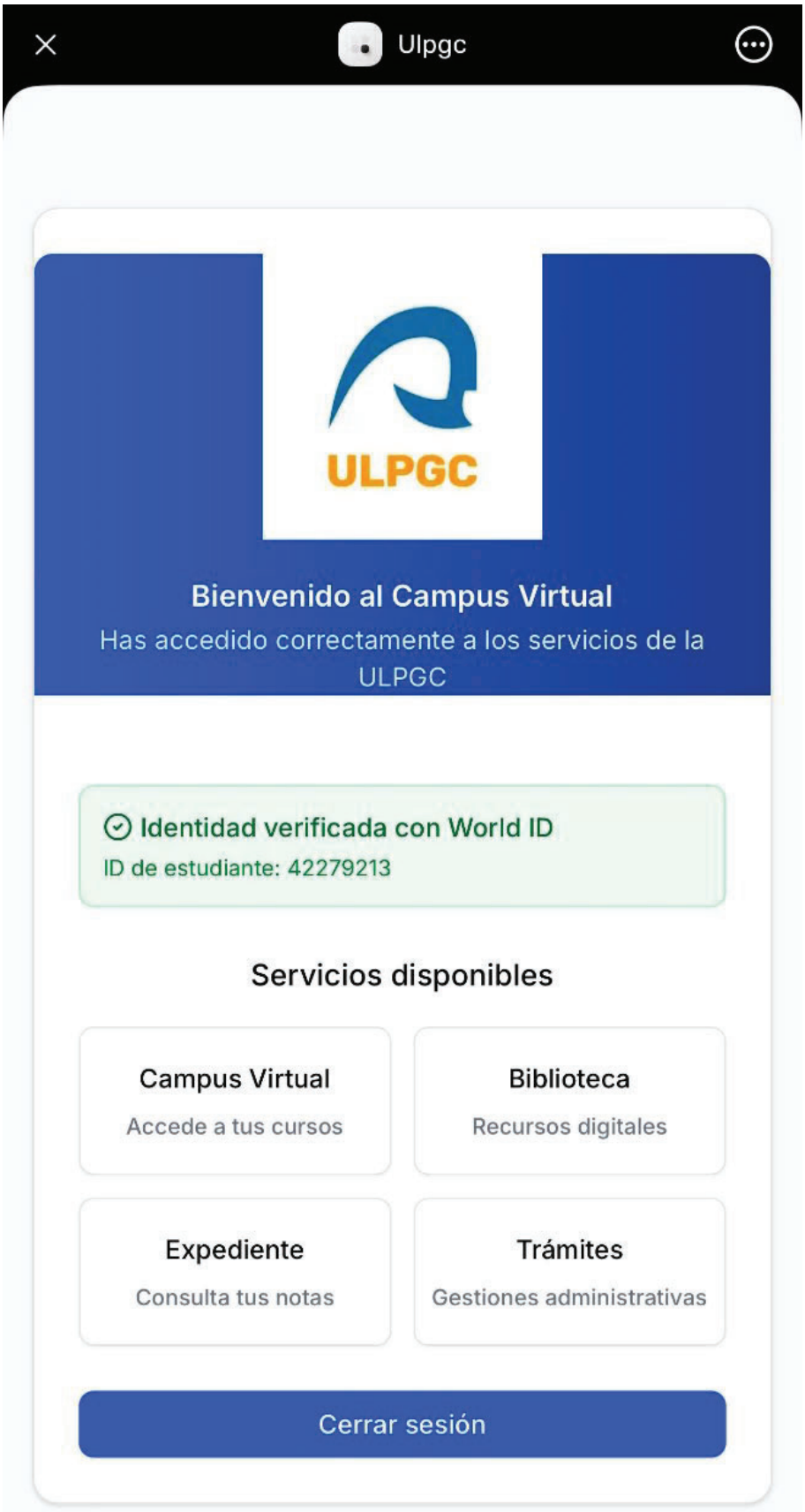

Figura 21. Interfaz del menú principal del Campus Virtual

5.4 Rendimiento

A partir de los registros obtenidos durante la ejecución de la simulación, se puede observar que la aplicación *ULPGC Verify* presenta tiempos de respuesta razonables para un entorno de pruebas expuesto mediante un túnel público, como se muestra en la *Figura 5.22*. La carga inicial de la interfaz mediante una petición *GET* se completa en aproximadamente *2,3 s*, mientras que el proceso de verificación de identidad mediante *World ID*, gestionado por el *endpoint /api/verify*, se ejecuta en *2,2 s*, incluyendo la validación del *ZKP*. Además, el *backend* local, implementado con *Next.js*, requiere *1,1 s* para compilar el *endpoint* antes de devolver una respuesta satisfactoria.

Por su parte, en la *Figura 5.23* se representa la terminal de *ngrok*, donde se registran todas las solicitudes *HTTP* externas, incluida la verificación, donde se reporta una latencia media de *63 ms* en la región europea, lo que garantiza una conexión estable para la utilización de la *World App* y el servidor local de desarrollo.

Los tiempos muestran un rendimiento adecuado para una *MiniApp* en entorno de simulación, también se evaluó el consumo de recursos del sistema durante la ejecución local de la *MiniApp*. Utilizando la herramienta *Administrador* de *tareas* de *Windows* mientras la aplicación se ejecutaba, justo en el momento de mayor carga, que sería la compilación inicial y validación del usuario.

Figura 5.22 Terminal de ejecución de la aplicación con npm run dev

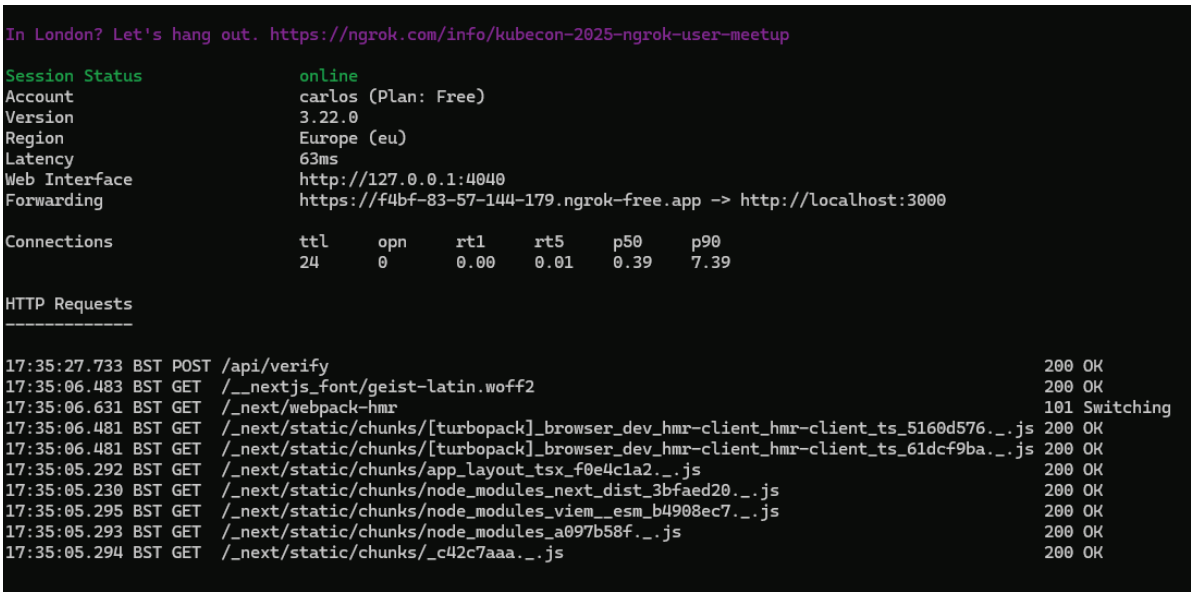

Figura 5.23 Consola de ngrok mostrando solicitudes HTTP y latencia

En la *Figura 5.24* se muestra el uso de CPU y memoria asociado a los procesos activos de Node.js, que corresponden tanto al servidor *Next.js* como al *SDK* de *Worldcoin* y la integración con *ngrok*, teniendo un uso de *CPU* aproximadamente de *4,4%,* mientras que la memoria ocupada por los procesos activos de Node.js aproximado de 530 MB, indicando una carga moderada para un entorno de desarrollo local. Las pruebas se realizaron en un equipo con procesador *Intel(R) Core (TM) i7-8565U a 1.80 GHz*, lo que permite contextualizar los resultados y confirmar que la demo es viable incluso en equipos de gama media-alta.

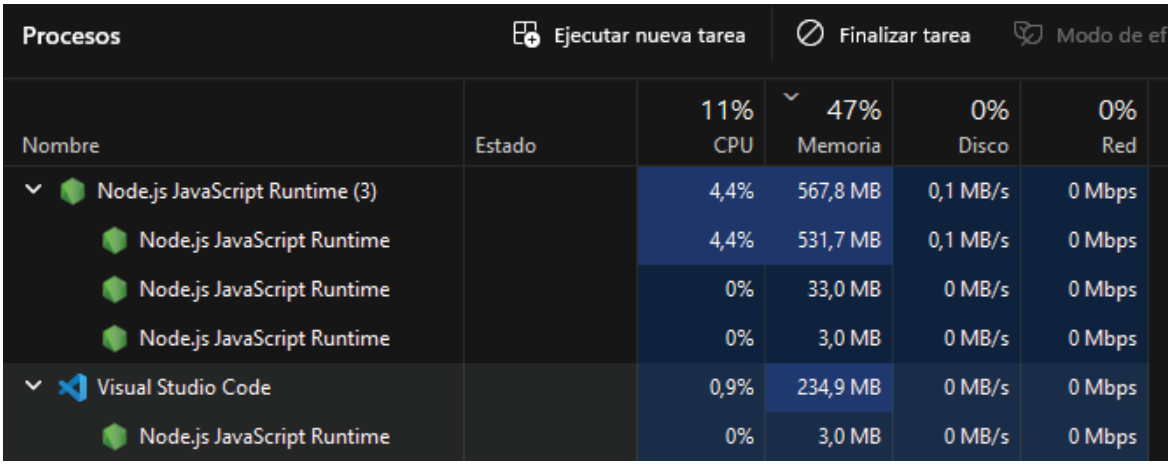

Figura 5.24 Uso de CPU y memoria de los procesos de Node.js durante la ejecución de la demo

6. Conclusiones y posibles ampliaciones

En este capítulo se exponen las conclusiones conseguidas tras el desarrollo, implementación y análisis del sistema *ULPGC Verify*, evaluando los resultados obtenidos en relación con los objetivos planteados inicialmente. Finalmente, se presentan posibles líneas de mejora y ampliaciones futuras que podrían incorporarse para evolucionar el proyecto hacia un entorno real de producción

6.1 Conclusiones

El desarrollo del presente TFG ha permitido analizar y demostrar la viabilidad de implementar un sistema de autenticación basado en tecnología *blockchain*, orientado a mejorar la seguridad, privacidad y descentralización en entornos donde se gestionan datos sensibles. Para ello hemos implementado el diseño de la aplicación *ULPGC Verify*, se ha investigado el uso de *Worldcoin* y su infraestructura como alternativa a los métodos de autenticación tradicionales, orientado al contexto académico.

En primer lugar, el análisis previo de los métodos de autenticación actualmente empleados en plataformas digitales se ha puesto de manifiesto que, si bien soluciones como las contraseñas, los tokens o la *MFA* son ampliamente utilizadas, presentan limitaciones significativas en cuanto a seguridad, experiencia de usuario y centralización. Esto es ya que los ataques por fuerza bruta, el robo de credenciales, el *phishing* o la dependencia de terceros son algunas de las vulnerabilidades detectadas. Esto tiene la necesidad de avanzar hacia modelos más robustos y sostenibles, capaces de ofrecer garantías tanto técnicas como éticas en la gestión de la identidad digital.

La elección de la tecnología *blockchain* responde a su capacidad para descentralizar la gestión de identidades, registrar información de forma inmutable y eliminar intermediarios. *Worldcoin* se presenta como una solución particularmente adecuada al integrar componentes avanzados como es su dispositivo *Orb*, su sistema de identificación *World ID* y *ZKP*. Tiene como base una arquitectura construida sobre *Ethereum Layer 1* y extendida mediante *OP Stack*, permitiendo escalar aplicaciones sin perder los beneficios de seguridad y descentralización de la red principal.

El proceso de implementación de *ULPGC Verify* ha demostrado que es posible desarrollar un sistema funcional que permita verificar la identidad del usuario sin almacenar datos personales ni depender de bases de datos internas. Con la ayuda de herramientas como el *World ID MiniKit*, *Next.js*, *Tailwind CSS* y *shadcn/ui*, se ha construido una *MiniApp* capaz de integrarse con la *World App*, recibir pruebas criptográficas y validar la autenticidad del usuario de forma segura. Durante su ejecución se ha validado el funcionamiento del sistema mediante pruebas locales, utilizando *ngrok* para habilitar

accesos remotos desde la *World App*, haciendo las respectivas medidas que muestran el tiempo de respuesta, siendo esta aceptable, con una carga inicial inferior a 2,5 segundos y un proceso de verificación completo que se ejecuta en torno a los 2 segundos. El consumo de recursos también ha sido moderado, con un uso de memoria cercano a los 530 *MB* y una carga de *CPU* del 4,4 % en momentos de mayor actividad, lo que demuestra la viabilidad del sistema incluso en equipos estándar.

A nivel conceptual, *ULPGC Verify* ha permitido reforzar la comprensión de los sistemas de identidad descentralizada. Ha mostrado cómo es posible crear una autenticación solida sin comprometer la privacidad del usuario y comprobando que el uso de *World ID* no requiere necesariamente el escaneo previo con *Orb* para propósitos de prueba. Esto facilita el desarrollo y validación del sistema sin necesidad de dispositivos biométricos especializados. La *MiniApp* ha sido diseñada para simular el acceso a servicios institucionales reales, como el campus virtual, la biblioteca o trámites académicos, lo que refuerza el carácter demostrativo del proyecto y permite fácilmente su integración en un entorno de producción.

Por último, el trabajo ha cumplido con los objetivos planteados, explorar una solución de autenticación innovadora basada en *blockchain*, implementando una *MiniApp* funcional y evaluar su comportamiento técnico.

6.2 Posibles ampliaciones

A partir de la experiencia obtenida durante el desarrollo de *ULPGC Verify*, se identifican diversas líneas de ampliación que permitirían llevar esta simulación a un entorno real de producción y mejorar sus funcionalidades.

- *La posibilidad de integración de una base de datos institucional*: que permita verificar que el *DNI* introducido pertenece efectivamente a un estudiante, docente o trabajador activo en la ULPGC, creando un sistema de doble validación, la primera sería la descentralización mediante *World ID* y la académica mediante la base de datos de la institución.

- *Vincular directamente los botones del menú de la MiniApp*: con los servicios reales de la ULPGC, como el acceso al campus virtual, trámites administrativos o expedientes, de esta forma se podría integrarse, pero requeriría establecer las *API* seguras y adaptar el sistema de roles según el perfil del usuario (estudiante, docente o personal administrativo).

- *Mecanismos de gestión de sesiones y registros de acceso*: de forma que se almacene un historial cifrado de entradas, cumpliendo con normativas de auditoría y seguras, y el uso de *Smart Contracts* específicos podría habilitar autorizaciones temporales o condicionales para ciertos servicios.

- *Disposición del dispositivo Orb*: tener un dispositivo *Orb* en cada escuela de la universidad y de esta manera sería posible realizar una validación biométrica real en el proceso de alta de cada usuario, elevando aún más el nivel de seguridad y eliminando cualquier posibilidad de duplicación de identidades.

Referencias bibliográficas

[1] "¿Qué es el ransomware? | IBM." Accessed: Apr. 06, 2025. [Online]. Available: https://www.ibm.com/es-es/topics/ransomware

[2] "Todo sobre el ciberataque que tiene en jaque a Barcelona: quién está detrás, cómo ha sido el 'hackeo' y por qué se eligió el Clínic." Accessed: Apr. 28, 2025. [Online]. Available: https://www.20minutos.es/tecnologia/ciberseguridad/todo-sobre-el-ciberataque-que-tiene-en-jaque-a-barcelona-quien-esta-detras-como-ha-sido-el-hackeo-y-por-que-se-eligio-el-clinic-5107297/

[3] "¿Qué es la ciberseguridad? | IBM." Accessed: Apr. 28, 2025. [Online]. Available: https://www.ibm.com/es-es/topics/cybersecurity

[4] "ENISA THREAT LANDSCAPE 2023." Accessed: Apr. 28, 2025. [Online]. Available: https://www.enisa.europa.eu/sites/default/files/publications/ENISA%20Threat%20Landscape%202023.pdf

[5] "¿Qué es Blockchain? | IBM." Accessed: Apr. 07, 2025. [Online]. Available: https://www.ibm.com/es-es/topics/blockchain

[6] "¿Qué Es Un Hash Y Cómo Funciona?" Accessed: May 15, 2025. [Online]. Available: https://latam.kaspersky.com/blog/que-es-un-hash-y-como-funciona/2806/?srsltid=AfmBOoo4EApdAXiG-agYT0VswQS4x4xVDVqqrlKPg3gH6uQ0hObf-zcv

[7] "¿Qué es Prueba de trabajo / Proof of Work (PoW)?" Accessed: Mar. 20, 2025. [Online]. Available: https://academy.bit2me.com/que-es-proof-of-work-pow/

[8] "¿Qué es Prueba de participación / Proof of Stake (PoS)?" Accessed: Apr. 07, 2025. [Online]. Available: https://academy.bit2me.com/que-es-proof-of-stake-pos/

[9] "Identidad digital descentralizada, ejemplos y conceptos clave." Accessed: Apr. 09, 2025. [Online]. Available: https://veridas.com/es/identidad-digital-descentralizada/

[10] "Introducing Worldcoin." Accessed: Mar. 25, 2025. [Online]. Available: https://whitepaper.world.org/#worldcoin-wld

[11] “¿Qué es Ethereum?” Accessed: Mar. 20, 2025. [Online]. Available: https://ethereum.org/es/what-is-ethereum/

[12] “Wallets - ¿Qué es una billetera de blockchain y cómo funciona?” Accessed: Apr. 09, 2025. [Online]. Available: https://kriptomat.io/es/blockchain/que-es-billetera-de-blockchaina/

[13] “The Orb.” Accessed: Apr. 04, 2025. [Online]. Available: https://whitepaper.world.org/#the-orb

[14] “Introducing World ID.” Accessed: Mar. 25, 2025. [Online]. Available: https://whitepaper.world.org/#world-id

[15] “MiniKit-JS SDK | World Docs.” Accessed: Apr. 01, 2025. [Online]. Available: https://docs.world.org/mini-apps/quick-start/installing

[16] “Introducing World App.” Accessed: Apr. 04, 2025. [Online]. Available: https://whitepaper.world.org/#world-app

[17] “Solana: A new architecture for a high performance blockchain v0.8.13.” Accessed: May 08, 2025. [Online]. Available: https://solana.com/solana-whitepaper.pdf

[18] “BNB Chain.” Accessed: May 08, 2025. [Online]. Available: https://docs.bnbchain.org/

[19] “Welcome to Cardano Docs | Cardano Docs.” Accessed: May 08, 2025. [Online]. Available: https://docs.cardano.org/

[20] “Zero-knowledge Rollups.” Accessed: Mar. 20, 2025. [Online]. Available: https://ethereum.org/es/developers/docs/scaling/zk-rollups/

[21] “Intro to zero-knowledge proofs, Semaphore and their application in World ID.” Accessed: Apr. 02, 2025. [Online]. Available: https://worldcoin.org/es-es/blog/world/intro-zero-knowledge-proofs-semaphore-application-world-id

[22] “¿Qué es Web3? - Explicación sobre Web3 - AWS.” Accessed: Apr. 07, 2025. [Online]. Available: https://aws.amazon.com/es/what-is/web3/

[23] “¿Qué es MetaMask y cómo empezar a usarlo? | Founderz.” Accessed: Apr. 07, 2025. [Online]. Available: https://founderz.com/es/blog/que-es-metamask-como-usarlo/

[24] “What is layer 2?” Accessed: Mar. 20, 2025. [Online]. Available: https://ethereum.org/en/layer-2/learn/

[25] “Getting started with the OP Stack | Optimism Docs.” Accessed: Mar. 24, 2025. [Online]. Available: https://docs.optimism.io/stack/getting-started

[26] “Rollups optimistas.” Accessed: Mar. 20, 2025. [Online]. Available: https://ethereum.org/es/developers/docs/scaling/optimistic-rollups/

[27] “Introducción a las DApps.” Accessed: Mar. 20, 2025. [Online]. Available: https://ethereum.org/es/developers/docs/dapps/

[28] “¿Qué es la autenticación? Definición y usos - Auth0.” Accessed: Mar. 31, 2025. [Online]. Available: https://auth0.com/es/intro-to-iam/what-is-authentication

[29] “El algoritmo SHA-256 explicado y visualizado paso a paso, bit a bit.” Accessed: Apr. 22, 2025. [Online]. Available: https://www.microsiervos.com/archivo/seguridad/algoritmo-sha-256-explicado-visualizado-paso-a-paso-bit-a-bit.html

[30] “Tabla arcoíris, Hash y Salts.” Accessed: Mar. 31, 2025. [Online]. Available: https://es.wikipedia.org/wiki/Tabla_arco%C3%ADris#Salts

[31] “GENERACIÓN DE CLAVE PERSONALIZADA DE AUTENTICACIÓN EN LA WEB DE LA ULPGC Y VÍAS DE CONTACTO. ACCESO A MIULPGC.” Accessed: Mar. 31, 2025. [Online]. Available: https://www.ulpgc.es/sites/default/files/ArchivosULPGC/cervantes/generacion_de_clave_de_autenticacion_m25-m45_v2.pdf

[32] “Interacción con la autenticación basada en formularios - Documentación de IBM.” Accessed: Apr. 01, 2025. [Online]. Available: https://www.ibm.com/docs/es/watson-explorer/12.0.x?topic=engine-interacting-form-based-authentication

[33] P. DE Privacidad Información Básica De Protección De Datos, “@ULPGC #ULPGC 5 | 25”, Accessed: Apr. 01, 2025. [Online]. Available: www.ulpgc.es,

[34] “Session Management - OWASP Cheat Sheet Series.” Accessed: Apr. 01, 2025. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html

[35] " ¿Qué es la autenticación basada en tokens?" Accessed: Mar. 31, 2025. [Online]. Available: https://www.entrust.com/es/resources/learn/what-is-token-based-authentication

[36] "JSON Web Tokens." Accessed: Mar. 31, 2025. [Online]. Available: https://auth0.com/docs/secure/tokens/json-web-tokens

[37] "RFC 6749 - The OAuth 2.0 Authorization Framework." Accessed: Mar. 31, 2025. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6749

[38] "RFC 7519 - JSON Web Token (JWT)." Accessed: Mar. 31, 2025. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7519

[39] "¿Qué es la autenticación multifactor (MFA)? | Cloudflare." Accessed: Mar. 31, 2025. [Online]. Available: https://www.cloudflare.com/es-es/learning/access-management/what-is-multi-factor-authentication/

[40] "¿Qué es la autenticación multifactor? - Soporte técnico de Microsoft." Accessed: Mar. 31, 2025. [Online]. Available: https://support.microsoft.com/es-es/topic/-qu%C3%A9-es-la-autenticaci%C3%B3n-multifactor-e5e39437-121c-be60-d123-eda06bddf661

[41] "Autenticación con doble factor | Servicio de Informática." Accessed: Mar. 31, 2025. [Online]. Available: https://si.ulpgc.es/multi-factor

[42] "¿Qué es el Credential Stuffing? | Akamai." Accessed: Apr. 10, 2025. [Online]. Available: https://www.akamai.com/es/glossary/what-is-credential-stuffing

[43] "¿Qué es el phishing? | IBM." Accessed: Apr. 10, 2025. [Online]. Available: https://www.ibm.com/es-es/topics/phishing

[44] "Qué es el SIM swapping | BBVA." Accessed: Apr. 10, 2025. [Online]. Available: https://www.bbva.es/finanzas-vistazo/ciberseguridad/ataques-informaticos/que-es-el-sim-swapping.html

[45] "Biometría para identificación y autenticación." Accessed: Apr. 01, 2025. [Online]. Available: https://www.thalesgroup.com/es/countries/americas/latin-america/dis/gobierno/inspiracion/biometria

[46] "Seguridad biométrica - Soporte técnico de Apple (ES)." Accessed: Apr. 01, 2025. [Online]. Available: https://support.apple.com/es-es/guide/security/sec067eb0c9e/web

[47] “¿Qué es la verificación biométrica y cómo funciona? | SEON.” Accessed: Apr. 01, 2025. [Online]. Available: https://seon.io/es/recursos/glosario/autenticacion-biometrica/

[48] “¿Qué son los certificados digitales? | Fortinet.” Accessed: Apr. 01, 2025. [Online]. Available: https://www.fortinet.com/lat/resources/cyberglossary/digital-certificates

[49] “Criptografía en certificados digitales: uso de claves públicas y privadas | Firma Digital con Certificado Digital Perú - Llama.pe.” Accessed: Apr. 01, 2025. [Online]. Available: https://llama.pe/blog/certificados-digitales/criptografia-en-certificados-digitales-uso-de-claves-publicas-y-privadas

[50] “Certificado Digital Funciones, Tipos y Uso.” Accessed: Apr. 01, 2025. [Online]. Available: https://blog.neteris.com/stepforward/seguridad-de-la-informacion-electronica.-el-certificado-digital

[51] “¿Qué es una red privada virtual o VPN? | Microsoft Azure.” Accessed: Apr. 10, 2025. [Online]. Available: https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-vpn

[52] “Cómo verificar un documento firmado electrónicamente por la ULPGC | ULPGC - Universidad de Las Palmas de Gran Canaria.” Accessed: Apr. 01, 2025. [Online]. Available: https://www.ulpgc.es/otea/como-verificar-documento-sede-electronica-ulpgc

[53] “¿Cuáles son las ventajas y desventajas de los certificados digitales para la autenticación?” Accessed: Apr. 01, 2025. [Online]. Available: https://www.linkedin.com/advice/0/what-advantages-disadvantages-digital-certificates-tbkre?lang=es&lang=es&originalSubdomain=es

[54] “FIDO2 - FIDO Alliance.” Accessed: Apr. 01, 2025. [Online]. Available: https://fidoalliance.org/fido2/

[55] “WebAuthn - ¿Qué es Web Authentication? - IONOS España.” Accessed: Apr. 10, 2025. [Online]. Available: https://www.ionos.es/digitalguide/servidores/seguridad/webauthn/

[56] “Web Authentication: An API for accessing Public Key Credentials - Level 2.” Accessed: Apr. 01, 2025. [Online]. Available: https://www.w3.org/TR/webauthn/

[57] “W3C and FIDO Alliance Finalize Web Standard for Secure, Passwordless Logins | 2019 | Press releases | W3C.” Accessed: Apr. 01, 2025. [Online]. Available: https://www.w3.org/press-releases/2019/webauthn/

[58] “¿Qué es el Single Sign on (SSO)? Características y ventajas.” Accessed: Apr. 10, 2025. [Online]. Available: https://chakray.com/es/que-es-el-single-sign-on-sso-definicion-caracteristicas-y-ventajas/

[59] “¿Qué es la identidad federada? - Artículo.” Accessed: Apr. 01, 2025. [Online]. Available: https://www.sailpoint.com/es/identity-library/what-is-federated-identity

[60] “eduroam ES - Pregunta Frecuentemnte Preguntadas (FAQ).” Accessed: Apr. 10, 2025. [Online]. Available: https://www.eduroam.es/faq/#toc_0

[61] “eduroam | UY | RAU, Funcionamiento.” Accessed: Apr. 10, 2025. [Online]. Available: https://www.eduroam.uy/funcionamiento.htm

[62] “802.1X-based solutions.” Accessed: Apr. 10, 2025. [Online]. Available: https://www.eduroam.uy/docs/DelD_v1.2-f.pdf

[63] “RFC 7593.” Accessed: Apr. 10, 2025. [Online]. Available: https://www.rfc-editor.org/rfc/pdfrfc/rfc7593.txt.pdf

[64] “¿Qué es OAuth y OpenID Connect? - ManageEngine Identity360.” Accessed: Apr. 01, 2025. [Online]. Available: https://www.manageengine.com/es/identity-360/que-es-ioa-y-como-funciona-oauth.html

[65] “Proof of personhood (prueba de condición de persona): qué es y por qué es necesaria.” Accessed: Apr. 10, 2025. [Online]. Available: https://world.org/es-es/blog/world/proof-of-personhood-what-it-is-why-its-needed

[66] “World ID: Implementing Proof of Human at Scale.” Accessed: Apr. 01, 2025. [Online]. Available: https://whitepaper.world.org/#world-id

[67] “Bitcoin: A Peer-to-Peer Electronic Cash System.” Accessed: Apr. 14, 2025. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[68] “Introducción a los contratos inteligentes.” Accessed: Mar. 20, 2025. [Online]. Available: https://ethereum.org/es/developers/docs/smart-contracts/

[69] “Ethereum - Wikipedia, la enciclopedia libre.” Accessed: Mar. 20, 2025. [Online]. Available: https://es.wikipedia.org/wiki/Ethereum

[70] “Introducción a ether.” Accessed: Mar. 20, 2025. [Online]. Available: https://ethereum.org/es/developers/docs/intro-to-ether/

[71] “Ethereum Accounts.” Accessed: Mar. 20, 2025. [Online]. Available: https://ethereum.org/en/whitepaper/#ethereum-accounts

[72] “Máquina virtual de Ethereum (EVM).” Accessed: Mar. 20, 2025. [Online]. Available: https://ethereum.org/es/developers/docs/evm/

[73] “Introducción a los contratos multifirma.” Accessed: Mar. 20, 2025. [Online]. Available: https://ethereum.org/es/developers/docs/smart-contracts/#multisig

[74] “Etherscan Information Center | Epoch in Ethereum.” Accessed: Apr. 25, 2025. [Online]. Available: https://info.etherscan.com/epoch-in-ethereum/

[75] “Gas y tarifas.” Accessed: Mar. 20, 2025. [Online]. Available: https://ethereum.org/es/developers/docs/gas/#what-is-gas

[76] “Merkle Patricia Trie.” Accessed: Apr. 14, 2025. [Online]. Available: https://ethereum.org/es/developers/docs/data-structures-and-encoding/patricia-merkle-trie/

[77] “¿Qué es Keccak256? Explora esta Función Hash Criptográfica y su Uso en Criptomonedas.” Accessed: Apr. 14, 2025. [Online]. Available: https://www.nervos.org/es/knowledge-base/what_is_keccak256_(explainCKBot)

[78] “Secp256k1: un Algoritmo Clave en Criptomonedas.” Accessed: Apr. 14, 2025. [Online]. Available: https://www.nervos.org/es/knowledge-base/secp256k1_a_key%20algorithm_(explainCKBot)

[79] “ERC-20 Token Standard.” Accessed: May 19, 2025. [Online]. Available: https://ethereum.org/en/developers/docs/standards/tokens/erc-20/

[80] “ERC-721 Non-Fungible Token Standard.” Accessed: May 19, 2025. [Online]. Available: https://ethereum.org/en/developers/docs/standards/tokens/erc-721/

[81] “ERC-1155 Multi-Token Standard.” Accessed: May 19, 2025. [Online]. Available: https://ethereum.org/en/developers/docs/standards/tokens/erc-1155/

[82] “¿Qué es un token, cómo funciona y para qué sirve? Tipos y ejemplos - Finect.” Accessed: May 12, 2025. [Online]. Available: https://www.finect.com/usuario/avillanuevae/articulos/que-es-un-token-como-funciona-y-para-que-sirve-tipos-y-ejemplos

[83] “The Merge.” Accessed: Apr. 14, 2025. [Online]. Available: https://ethereum.org/en/roadmap/merge/

[84] “¿Qué son las blockchains de Capa-2 de Ethereum y cómo funcionan? | Coinbase.” Accessed: Mar. 20, 2025. [Online]. Available: https://www.coinbase.com/es-es/learn/crypto-basics/what-are-ethereum-layer-2-blockchains-and-how-do-they-work

[85] “State Channels.” Accessed: Mar. 24, 2025. [Online]. Available: https://ethereum.org/en/developers/docs/scaling/state-channels/

[86] “Sidechains.” Accessed: Mar. 24, 2025. [Online]. Available: https://ethereum.org/en/developers/docs/scaling/sidechains/

[87] “Calldata .” Accessed: Apr. 18, 2025. [Online]. Available: https://www.quicknode.com/guides/ethereum-development/transactions/ethereum-transaction-calldata

[88] “¿Qué son los blobs en Ethereum?” Accessed: Apr. 18, 2025. [Online]. Available: https://www.nervos.org/knowledge-base/what_are_blobs_in_ethereum_(explainCKBot)

[89] “Superchain explainer | Optimism Docs.” Accessed: Mar. 24, 2025. [Online]. Available: https://docs.optimism.io/superchain/superchain-explainer

[90] “¿Qué son Bedrock?” Accessed: Apr. 18, 2025. [Online]. Available: https://solow.io/lecciones/que-es-el-op-stack

[91] “Design philosophy & design principles | Optimism Docs.” Accessed: Mar. 24, 2025. [Online]. Available: https://docs.optimism.io/stack/design-principles

[92] “¿Qué es EIP-4844 en Ethereum y cómo puede beneficiar a los usuarios? | Binance Academy.” Accessed: Mar. 24, 2025. [Online]. Available:

https://academy.binance.com/es/articles/what-is-eip-4844-in-ethereum-and-how-can-it-benefit-users

[93] “OP Stack components | Optimism Docs.” Accessed: Mar. 24, 2025. [Online]. Available: https://docs.optimism.io/stack/components

[94] “Differences between Ethereum and OP Stack Chains | Optimism Docs.” Accessed: Mar. 24, 2025. [Online]. Available: https://docs.optimism.io/stack/differences

[95] “Introducing World Network.” Accessed: Mar. 25, 2025. [Online]. Available: https://whitepaper.world.org/#introducing-world-network

[96] “How does World Work?” Accessed: Mar. 25, 2025. [Online]. Available: https://whitepaper.world.org/#how-does-world-work

[97] “World ID: Implementing Proof of Human at Scale.” Accessed: Apr. 03, 2025. [Online]. Available: https://whitepaper.world.org/#world-id-implementing-proof-of-human-at-scale

[98] “Introducción a la World ID.” Accessed: Apr. 03, 2025. [Online]. Available: https://world.org/es-es/blog/announcements/introducing-world-id-and-sdk

[99] “World ID Deduplication.” Accessed: Apr. 03, 2025. [Online]. Available: https://whitepaper.world.org/#deduplication-2

[100] “ Face ID - Apple.” Accessed: Apr. 03, 2025. [Online]. Available: https://support.apple.com/es-es/102381

[101] “Introducción al aprendizaje automático de conocimiento cero (ZKML).” Accessed: Apr. 03, 2025. [Online]. Available: https://world.org/es-es/blog/engineering/intro-to-zkml

[102] “World ID - Face Authentication.” Accessed: Apr. 03, 2025. [Online]. Available: https://whitepaper.world.org/#face-authentication

[103] “World ID - Iris Authentication.” Accessed: Apr. 03, 2025. [Online]. Available: https://whitepaper.world.org/#iris-authentication

[104] “World ID - Recovery.” Accessed: Apr. 03, 2025. [Online]. Available: https://whitepaper.world.org/#recovery-2

[105] “World ID - Revocation.” Accessed: Apr. 03, 2025. [Online]. Available: https://whitepaper.world.org/#revocation-2

[106] “World ID - Expiry.” Accessed: Apr. 03, 2025. [Online]. Available: https://whitepaper.world.org/#expiry-2

[107] “Presentamos Face Auth: la nueva tecnología de Worldcoin que mejora la seguridad privada.” Accessed: Apr. 03, 2025. [Online]. Available: https://world.org/es-es/blog/announcements/introducing-face-auth-worldcoin-newest-private-security-enhancing-tech

[108] “Presentamos AMPC: Otro salto en privacidad y rendimiento para World ID.” Accessed: Apr. 03, 2025. [Online]. Available: https://world.org/es-es/blog/engineering/introducing-ampc-another-leap-privacy-performance-world-id

[109] “World Foundation revela nuevo sistema SMPC, elimina códigos de iris antiguos.” Accessed: Apr. 04, 2025. [Online]. Available: https://world.org/es-es/blog/announcements/worldcoin-foundation-unveils-new-smpc-system-deletes-old-iris-codes

[110] “Why World Chain? | World Docs.” Accessed: Apr. 04, 2025. [Online]. Available: https://docs.world.org/world-chain/quick-start/why#what-is-different-about-world-chain

[111] “Presentamos PBH: Espacio de Bloque Prioritario para Humanos.” Accessed: Apr. 04, 2025. [Online]. Available: https://world.org/es-es/blog/engineering/introducing-pbh-priority-blockspace-for-humans

[112] “¿Qué es un airdrop de criptomonedas? | Coinbase.” Accessed: May 14, 2025. [Online]. Available: https://www.coinbase.com/es-es/learn/crypto-basics/what-is-a-crypto-airdrop

[113] “World Chain - Priority Blockspace for Humans.” Accessed: Apr. 04, 2025. [Online]. Available: https://docs.world.org/world-chain/quick-start/features#priority-blockspace-for-humans

[114] “OP Mainnet - L2BEAT.” Accessed: Apr. 04, 2025. [Online]. Available: https://l2beat.com/scaling/projects/op-mainnet

[115] “Privacy - The iris fragment.” Accessed: Apr. 04, 2025. [Online]. Available: https://whitepaper.world.org/#the-iris-fragment

[116] “World - Privacy.” Accessed: Apr. 04, 2025. [Online]. Available: https://whitepaper.world.org/#privacy

[117] “World App 3.0.” Accessed: Apr. 04, 2025. [Online]. Available: https://world.org/es-es/blog/announcements/introducing-world-app-3-super-app-humans

[118] “World - Wallets.” Accessed: Apr. 04, 2025. [Online]. Available: https://whitepaper.world.org/#wallets

[119] “¿Qué es un framework y para qué sirve? | Blog de Arsys.” Accessed: Apr. 02, 2025. [Online]. Available: https://www.arsys.es/blog/que-es-un-framework-en-programacion-y-para-que-sirve

[120] “¿Qué Es TypeScript? Guía Completa - Kinsta®.” Accessed: Apr. 02, 2025. [Online]. Available: https://kinsta.com/es/base-de-conocimiento/que-es-typescript/

[121] “React.” Accessed: Apr. 01, 2025. [Online]. Available: https://react.dev/

[122] “Introduction | Next.js.” Accessed: Apr. 01, 2025. [Online]. Available: https://nextjs.org/docs

[123] “Document Object Model (DOM) - Web APIs | MDN.” Accessed: Apr. 14, 2025. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

[124] “Introduction - JavaScript | MDN.” Accessed: Apr. 02, 2025. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction#where_to_find_javascript_information

[125] “Server Side Rendering | Modyo Docs.” Accessed: Apr. 14, 2025. [Online]. Available: https://docs.modyo.com/es/architecture/patterns/ssr.html

[126] “Static Site Generation | Modyo Docs.” Accessed: Apr. 14, 2025. [Online]. Available: https://docs.modyo.com/es/architecture/patterns/ssg.html

[127] “Qué es un endpoint y cómo funciona | Channel Partner.” Accessed: Apr. 14, 2025. [Online]. Available: https://www.channelpartner.es/seguridad/que-es-un-endpoint/

[128] “Routing: API Routes | Next.js.” Accessed: Apr. 02, 2025. [Online]. Available: https://nextjs.org/docs/pages/building-your-application/routing/api-routes

[129] “Introduction: App Router | Next.js.” Accessed: Apr. 02, 2025. [Online]. Available: https://nextjs.org/docs/app

[130] “Rendering: Server Components | Next.js.” Accessed: Apr. 02, 2025. [Online]. Available: https://nextjs.org/docs/app/building-your-application/rendering/server-components

[131] “Rendering: Client Components | Next.js.” Accessed: Apr. 02, 2025. [Online]. Available: https://nextjs.org/docs/app/building-your-application/rendering/client-components

[132] “Introducción a la biblioteca de interfaz de usuario - An Azure Communication Services concept document | Microsoft Learn.” Accessed: Apr. 02, 2025. [Online]. Available: https://learn.microsoft.com/es-es/azure/communication-services/concepts/ui-library/ui-library-overview?pivots=platform-web

[133] “Qué son los hooks y cómo utilizarlos en tu proyecto con React - Paradigma.” Accessed: Apr. 22, 2025. [Online]. Available: https://www.paradigmadigital.com/dev/hooks-como-utilizarlos-react/

[134] “Introduction - shadcn/ui.” Accessed: Apr. 02, 2025. [Online]. Available: https://ui.shadcn.com/docs

[135] “Qué es Tailwind CSS y por qué deberías usarlo | OpenWebinars.” Accessed: Apr. 02, 2025. [Online]. Available: https://openwebinars.net/blog/que-es-tailwind-css-y-por-que-deberias-usarlo/

[136] “What is Lucide? | Lucide.” Accessed: Apr. 02, 2025. [Online]. Available: https://lucide.dev/guide/

[137] “Commands | World MiniKit.” Accessed: Apr. 01, 2025. [Online]. Available: https://docs.world.org/mini-apps/quick-start/commands

[138] “¿Qué es el DDNS?: explicación del DNS dinámico - AWS.” Accessed: Apr. 14, 2025. [Online]. Available: https://aws.amazon.com/es/what-is/dynamic-dns/

[139] “Overview | ngrok documentation.” Accessed: Apr. 02, 2025. [Online]. Available: https://ngrok.com/docs

[140] “¿Qué es Ngrok y cómo utilizarlo? - El curso del Hacker.” Accessed: Apr. 02, 2025. [Online]. Available: https://www.elcursodelhacker.com/ngrok/

[141] “Window: localStorage property - Web APIs | MDN.” Accessed: Apr. 02, 2025. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage

[142] “Variables de Entorno: ¿Qué Son y Cómo Usarlas?” Accessed: Apr. 02, 2025. [Online]. Available: https://kinsta.com/es/base-de-conocimiento/variables-de-entorno/

[143] “Configuring: Environment Variables | Next.js.” Accessed: Apr. 02, 2025. [Online]. Available: https://nextjs.org/docs/pages/building-your-application/configuring/environment-variables

[144] “carlosortega108/ulpgc-world-verify.” Accessed: May 21, 2025. [Online]. Available: https://github.com/carlosortega108/ulpgc-world-verify

[145] “Worldcoin se compromete a paralizar su actividad en España | AEPD.” Accessed: Apr. 06, 2025. [Online]. Available: https://www.aepd.es/prensa-y-comunicacion/notas-de-prensa/worldcoin-se-compromete-paralizar-su-actividad-en-espana

[146] “Orb - Why Custom Hardware is Needed.” Accessed: Apr. 04, 2025. [Online]. Available: https://whitepaper.world.org/#why-custom-hardware-is-needed

[147] “Orb - Hardware.” Accessed: Apr. 04, 2025. [Online]. Available: https://whitepaper.world.org/#hardware

[148] “Orb - Designr.” Accessed: Apr. 04, 2025. [Online]. Available: https://whitepaper.world.org/#design

[149] “Orb - Optical System.” Accessed: Apr. 04, 2025. [Online]. Available: https://whitepaper.world.org/#optical-system

[150] “Orb - Battery.” Accessed: Apr. 04, 2025. [Online]. Available: https://whitepaper.world.org/#battery

[151] “Orb - Electronics.” Accessed: Apr. 04, 2025. [Online]. Available: https://whitepaper.world.org/#electronics

[152] “Biometrics.” Accessed: Apr. 04, 2025. [Online]. Available: https://whitepaper.world.org/#biometrics-2

[153] “Biometrics - False Non Matches.” Accessed: Apr. 04, 2025. [Online]. Available: https://whitepaper.world.org/#false-non-matches

[154] “Biometrics - How Iris Recognition Works.” Accessed: Apr. 04, 2025. [Online]. Available: https://www.robots.ox.ac.uk/~az/lectures/est/iris.pdf

[155] “IREX 10: Identification Track.” Accessed: Apr. 04, 2025. [Online]. Available: https://pages.nist.gov/IREX10/

[156] “Biometrics - Failure Cases.” Accessed: Apr. 04, 2025. [Online]. Available: https://whitepaper.world.org/#failure-cases

[157] “Biometrics - Effective Dual Eye Performance.” Accessed: Apr. 04, 2025. [Online]. Available: https://whitepaper.world.org/#effective-dual-eye-performance

[158] “Biometrics - Iris Feature Generation with Gabor Wavelets.” Accessed: Apr. 05, 2025. [Online]. Available: https://whitepaper.world.org/#iris-feature-generation-with-gabor-wavelets

[159] “Biometrics - Multi-scale Gabor filtering.” Accessed: Apr. 05, 2025. [Online]. Available: https://whitepaper.world.org/#multi-scale-gabor-filtering

[160] “Biometrics - Phase-quadrant demodulation and encoding.” Accessed: Apr. 05, 2025. [Online]. Available: https://whitepaper.world.org/#phase-quadrant-demodulation-and-encoding

[161] “Biometrics - Robustness of iris codes.” Accessed: Apr. 05, 2025. [Online]. Available: https://whitepaper.world.org/#robustness-of-iris-codes

[162] “Biometrics - High Confidence Visual Rocognition of Persons.” Accessed: Apr. 05, 2025. [Online]. Available: https://www.cl.cam.ac.uk/~jgd1000/PAMI93.pdf

[163] “Biometrics - Iris Inference System.” Accessed: Apr. 05, 2025. [Online]. Available: https://whitepaper.world.org/#iris-inference-system

[164] “Biometrics - Pipeline overview.” Accessed: Apr. 05, 2025. [Online]. Available: https://whitepaper.world.org/#pipeline-overview

[165] “Biometrics - Lazarski et al.” Accessed: Apr. 05, 2025. [Online]. Available: https://arxiv.org/pdf/2209.15471

[166] “ DeepLabV3+.” Accessed: Apr. 21, 2025. [Online]. Available: https://medium.com/@r1j1nghimire/semantic-segmentation-using-deeplabv3-from-scratch-b1ff57a27be

[167] “MobileNet V2.” Accessed: Apr. 21, 2025. [Online]. Available: https://www.eligeia.com/modelos/huggingface/google/mobilenet_v2_1.4_224

[168] “Biometrics - Segmentation.” Accessed: Apr. 05, 2025. [Online]. Available: https://whitepaper.world.org/#segmentation

[169] “Biometrics - Normalization.” Accessed: Apr. 05, 2025. [Online]. Available: https://whitepaper.world.org/#normalization

[170] “Biometrics - Feature generation.” Accessed: Apr. 05, 2025. [Online]. Available: https://whitepaper.world.org/#feature-generation

[171] “Biometrics - Comparación y coincidencia.” Accessed: Apr. 05, 2025. [Online]. Available: https://whitepaper.world.org/#matching

[172] “Biometrics - Iris code Upgrades.” Accessed: Apr. 05, 2025. [Online]. Available: https://whitepaper.world.org/#iris-code-upgrades

Pliego de condiciones

En este capítulo aborda los aspectos vinculados a las licencias, los derechos de autoría y las responsabilidades. También se detallan las condiciones en las que se ha llevado a cabo el desarrollo del TFG y los términos que deben aceptarse para el uso del software creado.

Pl.1. Condiciones hardware

A lo largo del desarrollo de este TFG, se han empleado los dispositivos hardware listados en la Tabla Pl.1.

Pl.2. Condiciones software

Las herramientas software empleadas en el desarrollo del TFG se organizan en dos apartados: las correspondientes al equipo principal (ordenador portátil MSI PS63 Modern 8RC) en la Tabla Pl.2 y las utilizadas en el dispositivo móvil (iPhone 15 Plus) en la Tabla PI.3.

Equipo	Modelo	Fabricante
Ordenador portátil	MSI PS63 Modern 8RC Intel Core i7-8565U, 16 GB RAM, 512 GB SSD, GTX 1050	MSI
Dispositivo móvil	iPhone 15 Plus (256 GB)	Apple

Tabla Pl.1. Condiciones Hardware.

Aplicación	Versión
Sistema Operativo Windows 11	24H2
Microsoft Office	Office 365
Visual Studio Code	1.100.0
JavaScript	22.14.0
TypeScript	5.8.2

Node.js	15.2.4
React	19.0.0
shadcn/ui	0.9.5
Tailwind CSS	4.0.15
Lucide React	0.483.0
@worldcoin/minikit-js	1.8.0
ngrok	3.22.0

Tabla Pl.2. Condiciones Software del equipo principal.

Aplicación	Versión
Sistema Operativo IOS	18.2.1
World App	2.8.8300

Tabla Pl.3. Condiciones Software del dispositivo móvil

Pl.3. Condiciones de uso por parte del usuario

Este apartado detalla las condiciones mínimas de hardware y software necesarias para que el usuario puede utilizar el sistema de verificación desarrollado. Para ejecutar la aplicación en fase de prueba, se recomienda el uso de *ngrok* como herramienta de túnel seguro, lo que permite establecer una conexión temporal entre el entorno local y la plataforma de

desarrolladores de *Worldcoin*, facilitando así la vinculación y prueba de la *MiniApp* dentro de la *World App*.

Pl.4. Condiciones de licencia

El código desarrollado en el presenta TFG son propiedad de la ULPGC. Cualquier persona o entidad que desee hacer uso total o parcial del mismo deberá aceptar las condiciones establecidas en esta licencia. El uso del sistema o su despliegue en plataformas externas requerirá autorización expresa por parte del autor, del tutor del TFG y de la Escuela de Ingeniería de Telecomunicación y Electrónica de la ULPGC.

Pl.5. Derechos de autor

Tanto el código fuente como la documentación asociada a la *MiniApp* están protegidos por la legislación vigente en materia de propiedad intelectual, así como por las disposiciones de los tratados internacionales aplicables. El sistema desarrollado se considera un producto sujeto a derechos de autor. No obstante, se permite su uso o reproducción bajo autorización expresa del autor, del tutor del TFG y de la Escuela de Ingeniería de Telecomunicación y Electrónica de la ULPGC.

Pl.6. Restricciones

No se permite el uso de ingeniería inversa. Sí se permite la transferencia del código a un tercero únicamente si no se conserve ninguna copia por parte del cedente, incluyendo modificadas, actualizadas o documentos complementarios.

Pl.7. Garantía

El autor del TFG lo presenta AS IS (tal cual), sin ningún tipo de garantía expresa o implícita. No se hace responsable de posibles daños directos o indirectos que pudieran derivarse del uso del código, la documentación o cualquier otro material asociado. No se garantiza la exactitud, fiabilidad ni idoneidad del sistema para fines específicos, ni se asegura que su funcionamiento esté libre de errores. El código no ha sido diseñado para su uso en entornos críticos o de alto riesgo que requieran tolerancia a fallos. Por tanto, se rechaza

expresamente cualquier garantía de adecuación del sistema para actividades que impliquen riesgos para equipos, sistemas o personas.

Pl.8. Limitación de responsabilidad

Ni el autor del TFG, ni los tutores, ni la Escuela de Ingeniería de Telecomunicación y Electrónica de la ULPGC serían responsables, en ninguna circunstancia, de daños directos, indirectos, incidentales, consecuentes o especiales, incluidos pérdidas económicas, interrupciones de actividad, pérdida de datos, beneficios no obtenidos o cualquier otra consecuencia derivada del uso o de la imposibilidad de uso del código y la documentación proporcionada. El usuario asume voluntariamente todos los riesgos asociados a la utilización del sistema y acepta las condiciones, cláusulas y restricciones expuestas. No se reconoce ninguna garantía adicional fuera de las ya expresamente indicadas en este documento.

Pl.9. Otras consideraciones

En caso de que alguna de las disposiciones recogidas en esta licencia sea declarada, total o parcialmente, inválida o inaplicable, dicha cláusula será modificada de forma adecuada para que resulte válida y ejecutable, sin que ello afecte a la validez del resto del contenido de la licencia. Este acuerdo se rige por la legislación vigente en España, y cualquier controversia derivada de su interpretación o aplicación estará sujeta a la jurisdicción exclusiva de los tribunales españoles. El usuario declara conocer y aceptar estas condiciones en su totalidad.

Presupuesto

Pr.1 Componentes del presupuesto

El presupuesto calculado se divide en las siguientes partes:

- Recursos materiales.
- Trabajo tarifado por tiempo empleado.
- Material Fungible.
- Redacción de la documentación.
- Derechos de visado del *Colegio Oficial de Ingenieros Técnicos de Telecomunicación* (*COITT*).
- Gastos de tramitación y envío.
- Aplicación de impuestos y coste total.

Pr.2 Recursos materiales

Para la ejecución y desarrollo de este TFG, se han necesitado diversos recursos materiales, tanto hardware como software, fundamentales para la implementación, simulación y documentación del proyecto. Para los recursos hardware, se ha utilizado un ordenador portátil, donde se desarrolló, probó y ejecutó la aplicación *ULPGC Verify*, además de documentar el desarrollo de la investigación. Por otro lado, se ha empleado un dispositivo móvil para realizar las pruebas de verificación mediante la *World App*, comprobando el funcionamiento de la *MiniApp* desde un entorno real. En cuanto al software, se ha utilizado una licencia de estudiante para el paquete *Microsoft Office 365*, por lo que no ha supuesto ningún coste adicional y las herramientas empleadas para el desarrollo, como *Next.js*, *Tailwind CSS*, *Node.js*, *World ID MiniKit*, *shadcn/ui* y *ngrok*, han sido utilizadas bajo planes gratuitos o de código abierto, sin implicar gastos económicos para el proyecto.

Para valorar económicamente el uso de los recursos materiales, se ha aplicado un sistema de amortización lineal, que considera una depreciación constante a lo largo de la vida útil de cada dispositivo, se calcula usando la *Ecuación Pr.1*.

$$Cuota\ anual = \frac{Valor\ de\ adquisición - Valor\ residual}{Número\ de\ años\ de\ vida\ útil}$$

Ecuación Pr.1

De esta manera, la amortización total aparece reflejada en la *Tabla Pr.1*, así como los diferentes valores y cuotas de los diferentes recursos empleados en este TFG.

(*) El valor residual, para los recursos *hardware*, ha sido calculado aplicando una depreciación del 25 % anual sobre su valor de adquisición, conforme a la LIS Art. 12.1 representando el valor teórico que tendría el bien tras su vida útil estimada.

El coste total de los materiales amortizables empleados en el desarrollo del presente TFG asciende a ciento cincuenta y nueve euros con ochenta y seis céntimos (159,86 €).

Recurso	Valor de adquisición (€)	Valor residual (*) (€)	Vida útil (años)	Cuota anual (€)	Uso (meses)	Cuota aplicable (€)
MSI Intel Core i7-8565U, 16 GB RAM, 512 GB SSD, GTX 1050	1.100,0	261,03	5	167,79	5	69,91
iPhone 15 Plus (256 GB)	1.119,0	472,07	3	215,64	5	89,85
MATERIALES AMORTIZABLES	**TOTAL**					**159,86**

Tabla Pr.1. Amortización total.

Pr.3 Trabajo tarificado por tiempo empleado

Este concepto contempla los gastos correspondientes a la mano de obra, calculados en función del número de horas dedicadas al desarrollo del TFG, conforme a las tarifas recomendadas por el COITT, donde el cálculo de los honorarios asociados a las horas de trabajo se realiza mediante la *Ecuación Pr.2:*

$$H = C_t \cdot 74{,}88 \cdot H_n + 96{,}72 \cdot H_e$$

Ecuación Pr. 2. Fórmula para el cálculo del trabajo tarificado por tiempo empleado

Donde:

- C_t Indica el coeficiente correcto en función de las horas dedicadas.
- H_n Indica las horas trabajadas dentro de la jornada laboral.
- H_e Indica las horas trabajadas fuera de la jornada laboral.

Según la Tabla Pr.2, el coeficiente corrector aplicable para un número de horas comprendido entre 180 y 360 horas es $C_t = 0{,}6$.

Para el desarrollo del TFG, se ha estimado un total de 300 horas de trabajo teórico, correspondientes al desarrollo de 12 créditos ECTS estipulado por el proyecto docente. A este esfuerzo se suman 10 horas adicionales consideradas fuera de jornada laboral, debido a tareas de resolución de incidencias técnicas externas, principalmente derivadas del uso de herramientas software de terceros. Sustituyendo en la *Ecuación Pr.2*:

$$H = 0{,}6 \cdot 74{,}88 \cdot 300 + 96{,}72 \cdot 10 = 14.445{,}6\ €$$

El coste total estimado en concepto de tiempo dedicado al desarrollo del TFG asciende a catorce mil cuatrocientos cuarenta y cinco euros con sesenta céntimos (14.445,2 €), libres de impuestos.

Horas empleadas (H_n)	Factor de corrección (C_t)
Hasta 36	1
36 – 72	0,9
72 – 108	0,8
108 – 144	0,7
144 – 180	0,65
180 – 360	0,6
360 – 540	0,55
540 – 720	0,5
720 – 1080	0,45
Más de 1080	0,4

Tabla Pr.2. Coeficientes para el cálculo de los honorarios

Pr.4 Material fungible

No se contempla ningún gasto por edición de documentos ni por gastos de material de oficina, por lo que el coste asociado al material fungible es de cero euros (0 €).

Pr.5 Redacción del trabajo

El importe correspondiente a la redacción del TFG se determina conforme a la fórmula establecida por el COITT, que se detalla en la siguiente expresión:

$$R = 0{,}07 \cdot P \cdot C_n$$

Ecuación Pr. 3 Cálculo del coste de redacción del trabajo

Donde:

- R son los honorarios por la redacción del TFG.
- P representa el presupuesto parcial acumulado hasta este punto: 14.605,46 €.
- C_n es el coeficiente de ponderación, que según el COITT tiene un valor de **1** para presupuestos menores de 30.050 €.

Sustituyendo:

$$R = 0{,}07 \cdot 14.605{,}46 \cdot 1 = 1.022{,}38$$

Por tanto, el coste estimado libre de impuestos correspondiente a la redacción del TFG asciende a mil veintidós euros con treinta y ocho céntimos (1.022,38 €).

Pr.6 Derechos de visado del COITT

Los gastos derivados del visado colegial del presente TFG se calculan conforme a las tarifas establecidas por el COITT, mediante la *Ecuación Pr. 4*:

$$V = 0{,}006 \cdot P \cdot C_v$$

Ecuación Pr.4 Cálculo de la tarifa de visado del COITT

Donde:

- V es el coste de visado del trabajo.
- P es el presupuesto del TFG
- C_v es el coeficiente reductor en función del presupuesto del trabajo.

Considerando los valores calculados anteriormente:

- Presupuesto parcial: 14.605,46 €
- Redacción del TFG: 1.022,38 €
- Presupuesto total acumulado (P): 15.627,84 €

Según lo establecido por el COITT, para presupuestos inferiores a **30.050 €**, el coeficiente C_v es igual a 1.

$$V = 0{,}006 \cdot 15.627{,}84 \cdot 1 = 93{,}77\ €$$

Por tanto, el coste libre de impuestos asociado a los derechos de visado del COITT asciende a noventa y tres euros con setenta y siete céntimos (93,77 €).

Pr.7 Gastos de tramitación y envío

Los gastos de tramitación y documentos están estipulados en seis euros (6,00 €).

Pr.8 Aplicación de impuestos y coste total

El presupuesto total estimado del proyecto se calculó en base a los diferentes recursos materiales y humanos empleados, asciende a 15.727,61 €, importe libre de impuestos. A esta cantidad se le aplica el Impuesto General Indirecto Canario (*IGIC*), correspondiente al 7 %, tal como establece el régimen fiscal vigente en la Comunidad Autónoma de Canarias para este tipo de servicios técnicos.

Descripción	Coste (€)
Recursos materiales	159,86
Trabajo tarifado por tiempo empleado	14.445,60
Costes de material fungible	0
Costes asociados a la redacción	1.022,38
Derechos de visado del COITT	93,77
Gastos de tramitación y envío	6,00
Suma	15.727,61
IGIC (7%)	1.100,93
TOTAL	**16.828,54**

Tabla Pr.3. Cálculo del presupuesto total

Por tanto, el presupuesto total del presente TFG asciende a dieciséis mil ochocientos veintiocho euros con cincuenta y cuatro céntimos (16.828,54 €), impuestos incluidos.

Las Palmas de Gran Canaria, a 21 de mayo de 2025

Fdo.: Carlos Manuel Ortega Negrín

ORTEGA NEGRIN CARLOS MANUEL - 42279213S

Firmado digitalmente por ORTEGA NEGRIN CARLOS MANUEL - 42279213S
Fecha: 2025.05.21 19:23:21 +01'00'

Anexos

Se realiza la descripción de ciertos conceptos que, si bien no son imprescindibles para comprender el desarrollo principal de este TFG, resultan relevantes para entender algunos aspectos subyacentes relacionados con la tecnología biométrica y el sistema Worldcoin.

Anexo A. Dispositivo biométrico Orb

Para realizar la verificación de identidad en el sistema *Worldcoin*, es necesario conocer el funcionamiento del dispositivo biométrico *Orb*. Se analiza su arquitectura, el diseño de *hardware* y las diferencias con otros dispositivos como los teléfonos inteligentes y su sistema para generar identificadores únicos a partir del escenario del iris.

A.1 Funcionamiento del Orb y limitaciones de los sistemas biométricos convencionales

El dispositivo biométrico *Orb*, fue desarrollado por *Tools for Humanity* de *World* para la verificación de unicidad de cada usuario que se registre en el sistema de *World ID*. Está diseñado para capturar imágenes del iris con la máxima precisión posible y evitar fraudes, siendo la pieza más importante dentro de la infraestructura de identificación digital descentralizada que propone *World*. A diferencia de otros sistemas de identificación, el hardware de *Orb* fue diseñado desde cero para garantizar la inclusividad, la resistencia al fraude y la privacidad. Esta tecnología permite verificar que un usuario es único sin necesidad de almacenar las imágenes biométricas en servidores centrales, asegurando que cada persona recibe exactamente un único *World ID* manteniendo la privacidad personal del usuario [13].

Actualmente los teléfonos inteligentes están equipados con sistemas avanzados de autenticación biométrica, pero presentan importantes limitaciones en cuanto la verificación de identidad a gran escala. En primer lugar, existen muchos modelos de teléfonos inteligentes que dependiendo del modelo tiene unas características de hardware. Por ejemplo, están equipadas con cámaras que carecen de la resolución necesaria para captar con detalle el iris de manera precisa, lo que afecta la fiabilidad del reconocimiento.

También estos dispositivos utilizan el espectro de luz visible, lo que genera reflejos y ruidos en la imagen especialmente en personas con ojos oscuros, disminuyendo la precisión del sistema al no distinguir el iris con la pupila. Por tanto, la captura las imágenes de alta resolución, ya que, utilizando cámaras diseñadas para operar en el espectro

infrarrojo cercano. Esto minimiza las interferencias de luz ambiental y mejora la calidad de la imagen capturada. Otra desventaja de los teléfonos inteligentes es su seguridad a la hora de verificar la identidad de un usuario es relativamente baja, ya que el objetivo del sistema no es solo autenticar la identidad de una persona, sino garantizar que un individuo no se registre de forma múltiple.

Un atacante no necesita suplantar la identidad de otra persona, sino simplemente aparentar ser un usuario distinto cada vez. Los teléfonos y cámaras de iris estándar carecen de sensores multiángulo y multiespectrales, por lo que pueden ser engañados con imágenes, vídeos o modelos tridimensionales. Además, la ausencia de un entorno de ejecución seguro en la mayoría de los teléfonos hace que sea sencillo para un usuario con los conocimientos adecuados pueda manipular la verificación y generar múltiples registros fraudulentos. Incluso los escáneres de iris existen en el mercado, están diseñados para aplicaciones de seguridad como el control de acceso o la verificación en aeropuertos, pero no cumplen los requisitos para una verificación de identidad descentralizada a gran escala.

Figura A.1 Dispositivo Orb

Fuente: https://soyhodler.com/wp-content/uploads/2024/10/New-Orb-1792x1280.jpg

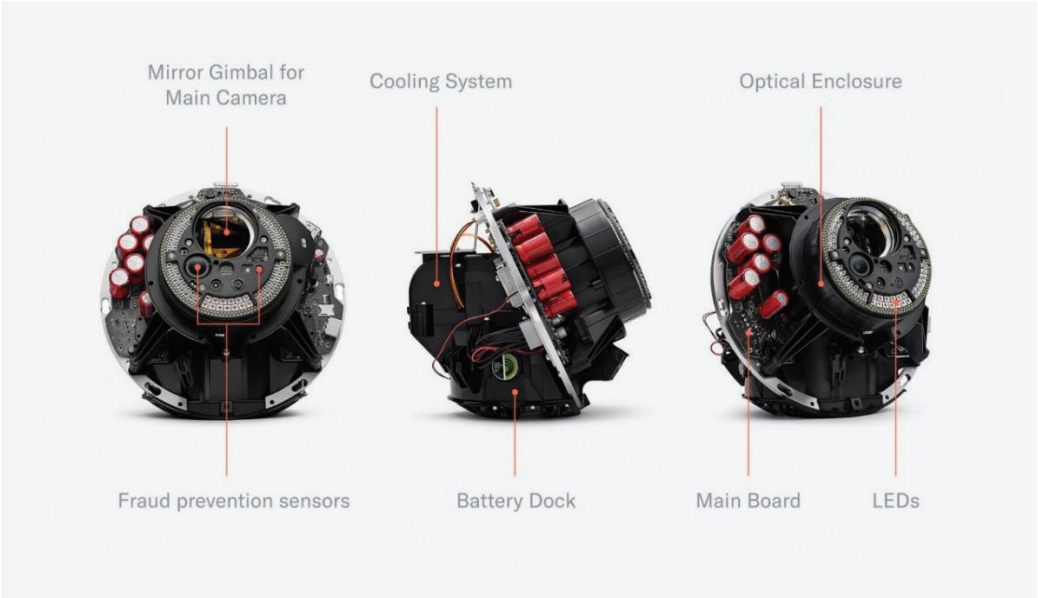

Figura A.2 Estructura general del Orb

Fuente: https://whitepaper.world.org/images/whitepaper/notion2/image55.jpg

Estos dispositivos están desarrollados para operar en entornos controlados y supervisados por el personal de seguridad, lo que reduce la necesidad de sistemas de protección contra ataques avanzados. En cambio, para el sistema de *World* esto no funciona, ya que el *World ID* debe funcionar en cualquier entorno de manera descentralizada y ser resistente a intentos de fraude sin necesidad de supervisión constante. El *Orb* incorpora un conjunto de cámaras infrarrojas, cámaras de profundidad, sensores térmicos y ópticas de precisión ser capaz de detectar cuando es un ser humano y cuando es una réplica artificial, en la *Figura A.1* se muestra el diseño final del dispositivo.

De manera teoría parece fácil que un dispositivo pueda ser capaz de ser lo suficientemente preciso solo con la ayuda de sensores, pero el hardware del *Orb* cuenta con una alta capacidad de procesamiento para así poder detectar en tiempo real si la imagen capturada es un fraude. El procesador que incorpora el *Orb* es un *NVIDIA Jetson Xavier NX*, capaz de ejecutar redes neuronales en el dispositivo sin necesidad de enviar datos a un servidor externo, lo que garantiza así una mayor privacidad y autonomía en el procesamiento de datos [146].

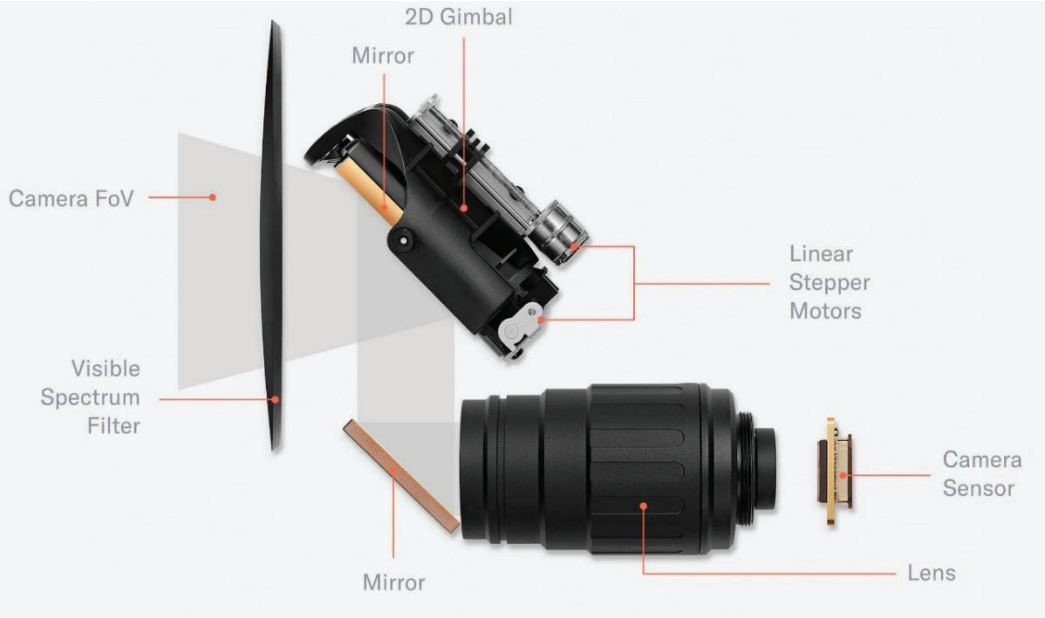

Figura A.3 El sistema óptico de Orb

Fuente: https://whitepaper.world.org/images/whitepaper/notion2/image66.jpg

La seguridad del *software* que utilizan los dispositivos disponibles en el mercado no garantiza un entorno descentralizado, lo que abre la posibilidad de que un atacante manipule el firmware de un escáner de iris convencional o altere los datos antes de que sean procesador. Esto representa una vulnerabilidad que contradice los principios de verificación de *World ID*, por este motivo, el *Orb* fue desarrollado por un firmware seguro. Se actualiza de manera remota a nivel del *bootloader*, lo que impide que cualquier tipo de manipulación no sea autorizada.

Entre las características del diseño del *Orb* es que incorpora un sistema de detección de manipulación que puede desactivar el dispositivo en caso de intento de manipulación, lo que evita cualquier modificación no autorizada. Además, todas las imágenes del iris se procesan de manera local en el dispositivo, es decir, sin necesidad de almacenamiento o envió a servidores externos, lo que garantiza la privacidad de los usuarios.

Partiendo de que, en el mundo de la tecnología, no existe un dispositivo que ofrezca una seguridad absoluta, el *Orb* tiene un diseño para establecer un nivel alto de seguridad. La implementación de múltiples capas de protección, incluyendo un equipo interno que trabaja permanentemente en identificar vulnerabilidades y optimizar los algoritmos de detección de intentos de suplantación [13].

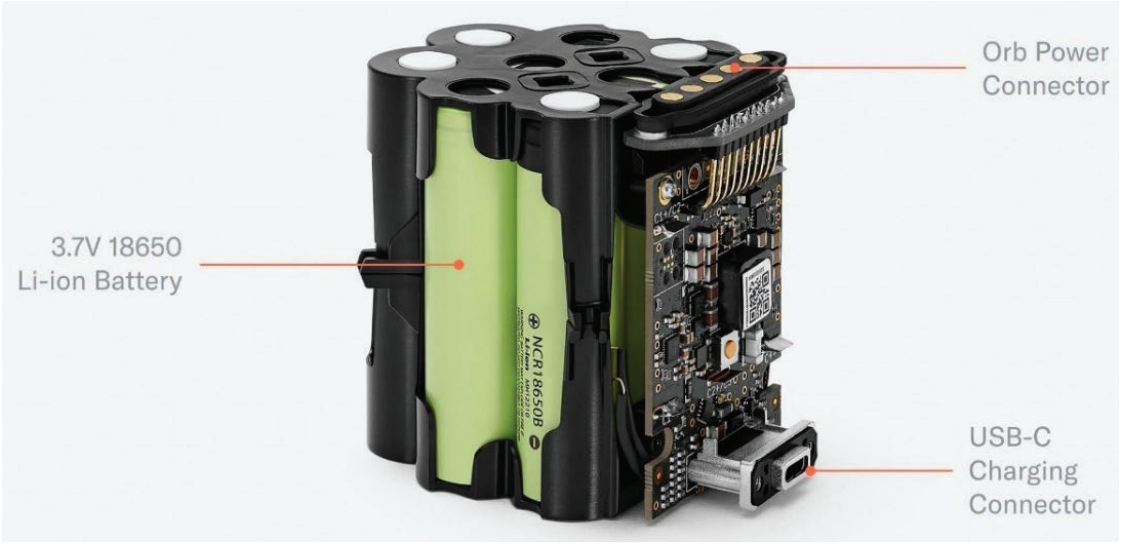

Figura A.4 Sistema de energía

Fuente: https://whitepaper.world.org/images/whitepaper/notion2/image61.jpg

A.2 Diseño y Hardware del Orb

El *Orb* está compuesto por varios módulos interconectados, donde su interior este compuesto por una placa base, el sistema óptico y el sistema de refrigeración. La mayor parte del sistema óptico se encuentra dentro de la carcasa sellada para evitar que entre el polvo y la humedad [147].

La estructura del dispositivo se divide en dos hemisferios, separados por la placa base, la cual esta inclinada unos 23,5º. La parte frontal está dedicada al sistema óptico, mientras que la posterior alberga la unidad de procesamiento principal y el sistema de refrigeración, también cuenta con una batería intercambiable en la parte superior, para garantizar el funcionamiento continuo en escenarios de movilidad [148], como se muestra en la *Figura A.2*.

El sistema óptico de *Orb* es uno de sus componente más complejos y avanzados, en el momento que empezaron hacer pruebas de campo, se dieron cuenta que el sistema tenía que ser de autoenfoque. El ojo humano siempre está en constante movimiento, por lo que se mejoró el hardware para integrar un sistema de autoenfoque y seguimiento ocular, facilitando de esta manera la alineación cuando el usuario se encuentra a una distancia aproximada de un metro.

Figura A.5 Comparativa entre la imagen capturada por el Orb y los estándares industriales

Fuente: https://whitepaper.world.org/images/whitepaper/notion2/image60.jpg

Para capturar las imágenes de alta calidad, el sistema óptico emplea una cámara de teleobjetivo estabilizada con dos ejes y sensores multiespectral que van verificando la autenticidad de la imagen. Además, un espejo con un campo de visión ajustable mediante un cardán, que permite ampliar significativamente el área de captura, como se muestra en la *Figura A.3*, lo que va optimizando el proceso de registro y mejorando la precisión del reconocimiento biométrico [149].

El sistema de energía, como se muestra en la *Figura A.4,* con una batería intercambiable basada en celdas de iones de *litio 18650*, es el mismo tipo que utilizan los vehículos eléctricos. Tiene una capacidad aproximada de unos *100Wh* y una configuración *4S2P* que proporciona *14’8V*. También incorpora un conector *USB-C*, el conector estándar que más se está usando en la actualizada por su eficiencia y rapidez de carga [150].

También se le añadieron al *Orb* diversos elementos visuales y táctiles para facilitar la interacción del usuario con el dispositivo. Por ejemplo, un anillo *Led Red, Green, Blue* (*RGB*) que rodea del dispositivo, que ayuda a indicar mediante señales visuales durante el proceso de registro, también incorpora un *Light Emitting Diode* (*LED*) de estado junto al único botón para indicar su estado operativo.

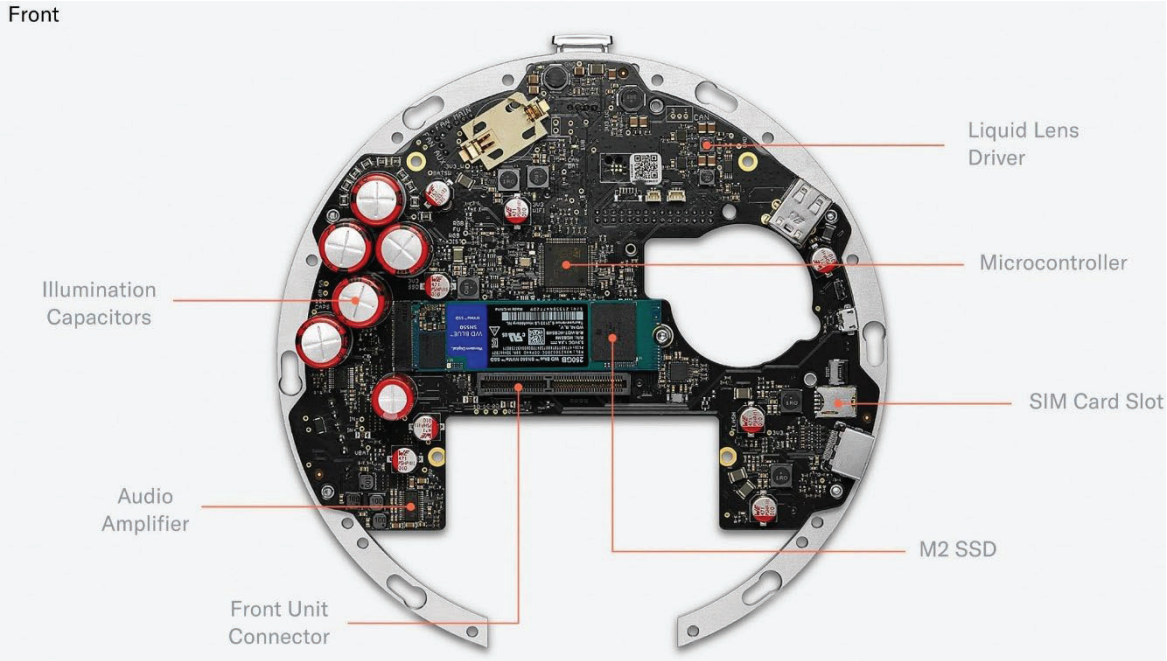

Figura A.6 Placa base frontal del Orb

Fuente: https://whitepaper.world.org/images/whitepaper/notion2/image68.jpg

El *Orb* contiene un lente optimizado para el espectro infrarrojo cercano, como se muestra en la *Figura A.5*. Cuenta con un sistema de autoenfoque está basado en una lente líquida que ajusta su posición en milisegundos, mediante el control de una red neuronal. El sistema de enfoque este combinado con un sensor de obturador global, un teleobjetivo, un sistema de espejos con cardan 2D y un filtro óptico, para poder garantizar que las imágenes sean capturadas de manera nítida y sin distorsión, incluso en condiciones de iluminación difíciles [149].

El sistema electrónico está compuesto por múltiples *Printed Circuit Board* (*PCB*), donde la placa base frontal, como se muestra en la *Figura A.6*. Contiene controladores de energía, circuitos de iluminación y un módulo de seguridad para la detección de manipuladores. También cuenta con una unidad de almacenamiento *M.2 Solid State Drive* (*SSD*) cifrada, la cual es necesaria para almacenar las imágenes de forma temporal y recopilación de datos, para poder garantizar que los datos del usuario estén protegidos y no ser manipulados por terceros.

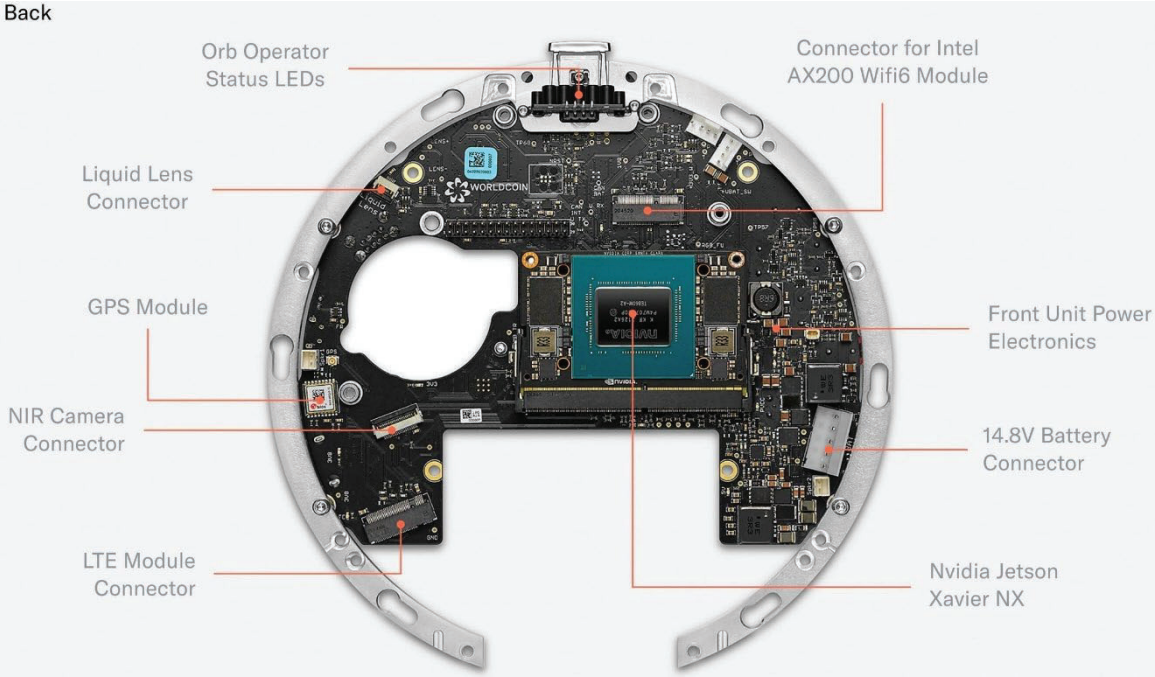

Figura A.7 Placa base trasera del Orb

Fuente: https://whitepaper.world.org/images/whitepaper/notion2/image64.jpg

Las imágenes almacenadas en el dispositivo están protegidas con un nivel adicional de cifrado asimétrico, lo que permite al *Orb* cifrar, pero no descifrar los datos de los usuarios. También el *Orb* cuenta con conexión *LTE* opcional, ya que cuenta con una ranura para la tarjeta *SIM*, permitiendo conectarse a la red en caso de no disponer de *WiFi* [151].

La placa trasera, como se muestra en la *Figura A.7*. Contiene el procesador *NVIDIA Jetson Xavier* NX*,* encargado de ejecutar múltiples redes neuronales en tiempo real. Gestiona la captura y procesamiento de las imágenes, detección de fraudes y la generación de código iris de manera completamente local. Garantizando la privacidad de los usuarios se mantengan en todo momento, eliminando el riesgo de exposición de datos sensibles. En la placa también integra un *GPS*, para rastrear la ubicación del *Orb* para evitar algún posible fraude y un módulo *WiFi 6*, que facilita la transferencia de datos de manera rápida [151].

El *Orb* también incorpora un sistema de prevención de suplantación, como se muestra en la *Figura A.8*. Ubicado en la PCB frontal que está equipada con 79 *LED* infrarrojos de diferentes longitudes de onda (740 nm, 850 nm y 940 nm). Permite capturar imágenes multiespectrales del iris y ser capaz de detectar intentos de falsificación. Esta parte de la placa también incorpora sensores adicionales, como una cámara gran angular de infrarrojo

cercano, una cámara de profundidad 3D y una cámara térmica. Este conjunto de cámaras trabaja de forma simultánea para ayudar a verificar que la imagen es capturada corresponde a un ser humano [151].

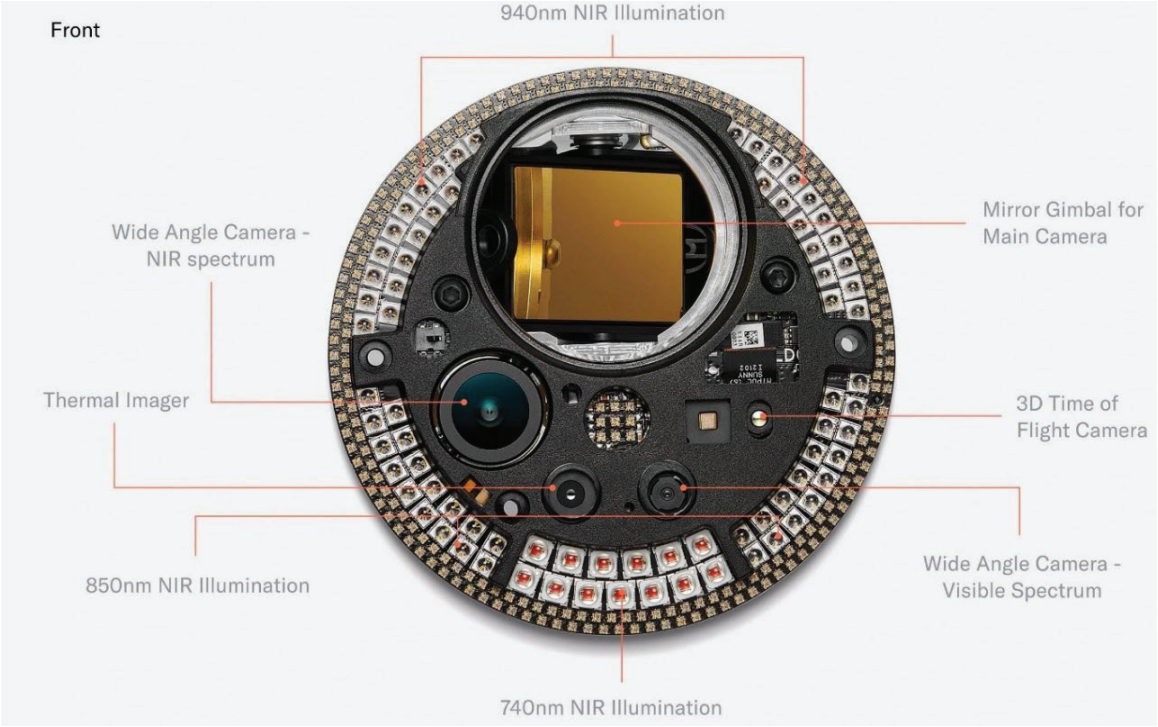

Figura A.8 Sistema de suplantación del Orb

Fuente: https://whitepaper.world.org/images/whitepaper/notion2/image67.jpg

Anexo B. Conceptos biométricos para el sistema Worldcoin

Si bien el sistema *Worldcoin* presenta una arquitectura descentralizada basada en tecnología *blockchain*, uno de sus componentes diferenciales es el uso de un sistema biométrico ocular para garantizar la unicidad de cada usuario y no almacena ni imágenes ni datos personales, sino que genera una codificación matemática a partir del iris del individuo, denominada código de iris.

B.1 Principios biométricos

En primer lugar, se analizan los principios biométricos fundamentales que permiten esta generación, incluyendo el uso de técnicas avanzadas como el filtrado de Gabor, la demodulación por cuadrantes de fase, y la posterior codificación binaria de las características extraídas. También proporciona una base teórica que ayude a comprender los fundamentos matemáticos y computacionales que permiten generar un código de iris único y no reversible, garantizando así la unicidad del usuario sin comprometer su privacidad.

Después del proceso de captura de la imagen del iris, ésta se procesa para confirmar la unicidad de cada imagen mediante un algoritmo biométrico. Está diseñado para operar a gran escala, considerando un escenario donde este modelo biométrico permanezca sin actualizaciones, de manera que su rendimiento se mantenga constante a lo largo del tiempo [152]. Un sistema biométrico puede cometer dos tipos de errores, uno es la identificación errónea de una persona con otra, conocido como falsa coincidencia y el otro error es no reconocer a un individuo registrado previamente, denominado falsa no coincidencia. Las métricas para evaluar el desempeño de cualquier tecnología biométrica son la *False Match Rate* (*FMR*) y la *False Non-Match Rate* (*FNMR*) [153].

Para analizar el sistema biométrico, hay que considerar estos tres sistemas con diferentes niveles de precisión:

- El sistema de John Daugman, que muestra una tasa de falsa coincidencias de 1.1×10^{-7} con una tasa de falsa no coincidencias de 0.00014 [154].

- Luego está el modelo desarrollado por *Nippon Electric Company* (*NEC*), estos resultados están reflejados en los informes *IREX IX* e *IREX X* [155] del Instituto Nacional de Estándares y Tecnología (*NIST*). Presentan una *FMR* de 10^{-8} y una *FNMR* de 0.045.

- Las primeras fases del desarrollo de *World*, diseñaron un prototipo conceptual que representaba una estimación del desempeño del reconocimiento de iris en un escenario ideal para la *FMR* de 10^{-6} y una *FNMR* de 0.005. Estos valores no son ideales, representaron que el reconocimiento del iris era la opción más viable para validar la identidad humana a gran escala [156].

B.2 El desempeño optimizado con ambos ojos

Los valores previamente mencionados se corresponden con el reconocimiento basado en un solo ojo; en cambio, el uso de ambos ojos puede mejorar la efectividad de un sistema biométrico, para evaluar su rendimiento, se consideran dos escenarios extremos:

- *Regla AND*: se acepta una coincidencia solo si ambos iris coinciden, permitiendo una mayor cantidad de usuarios en el sistema, pero a costa de una menor seguridad, ya que la tasa de coincidencias falsas disminuye mientras que la *FNMR* aumenta. Los valores de rendimiento para este enfoque son los siguientes [157]:

$$\mathrm{FMR}_{\mathrm{AND}} = \mathrm{FMR}^2$$

$$\mathrm{FNMR}_{\mathrm{AND}} = 2\mathrm{FNMR}(1 - \mathrm{FNMR}) + \mathrm{FNMR}^2$$

- *Regla OR*: se considera que un usuario está identificado si al menos uno de sus iris coincide con un registro existente, siendo más permisiva. Esto solo requiere una coincidencia parcial para validar la identidad, lo que reduce la *FNMR*, pero a medida que el número de personas registradas en el sistema crece, esta estrategia puede dificultar la inscripción de nuevos usuarios legítimos debido a

una mayor *FMR*. Los valores de rendimiento efectivos bajo estas condiciones se presentan en los siguientes estudios [157]:

$$\mathrm{FMR}_{\mathrm{OR}} = 2\mathrm{FMR}(1 - \mathrm{FMR}) + \mathrm{FMR}^2$$

$$\mathrm{FNMR}_{\mathrm{OR}} = \mathrm{FNMR}^2$$

B.3 Coincidencias falsas

La probabilidad de que el usuario legítimo número i experimente un error de coincidencia falsa [153] se puede calcular mediante la siguiente ecuación:

$$P_{\mathrm{FM}}(i) = 1 - P_{\text{no match with } i-1 \text{ users in DB}} = 1 - (1-p)^{i-1}$$

Donde $p = FMR$ representa la tasa de coincidencias falsas y sumando estos valores, se obtiene el número esperado de coincidencias falsas que han ocurrido después de que el usuario i haya sido registrado, es decir, el número de usuarios rechazados incorrectamente.

$$N_{\mathrm{FM}}(i) = \sum_{j=1}^{i} P_{\mathrm{FM}}(j) = \frac{(1-p)^i + i \cdot p - 1}{p}$$

Una tasa de coincidencias falsas elevada afecta considerablemente la usabilidad del sistema. Esto es porque la probabilidad de coincidencias erróneas aumenta a medida que se incrementa la cantidad de usuarios en la base de datos y la probabilidad de que un nuevo usuario sea rechazado de manera errónea podría ser del 100%, lo que hace que la aceptación de nuevos registros sea prácticamente imposible.

En la *Figura B.1* se muestra el rendimiento del sistema biométrico aplicando tanto para la regla *OR* como la regla *AND*. Esta gráfica está dividida en dos partes, la parte izquierda representa la regla *OR*, mientras que la derecha muestra la regla *AND*. En la primera fila de gráficos, se representa la probabilidad $P_{\mathrm{FM}}(i)$ del usuario número i sea rechazado erróneamente, mientras que en la segunda fila se muestra el número esperado $N_{\mathrm{FM}}(i)$ de usuarios que han sido rechazados de manera incorrecta después de que el usuario número i haya completado su registro.

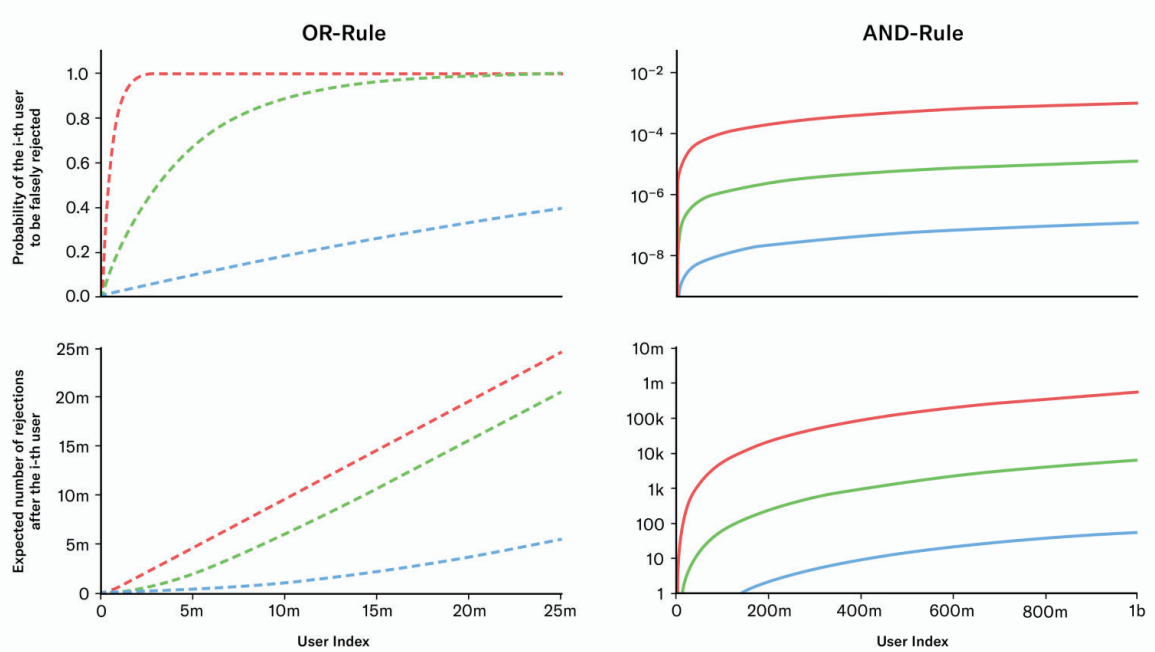

Figura B.1 Rendimiento de los sistemas OR y AND en tres escenarios distintos

Fuente: https://whitepaper.world.org/images/whitepaper/notion2/image49.png

Cada color en el gráfico corresponde, en la *Figura B.1* a los tres sistemas que hemos mencionado anteriormente: Verde representa el sistema de Daugman, Azul indica el sistema de *NEC* y Rojo representa la estimación más pesimista realizada en la fase inicial del estudio.

Al analizar las gráficas encontramos que al emplear la regla *OR*, la eficacia del sistema comienza a disminuir cuando el número de usuarios va alcanzando el millón. La probabilidad de que nuevos usuarios sean rechazados erróneamente aumenta de manera considerable, en el caso de la regla *AND* que ofrece una solución más escalable para una base de usuarios en crecimiento.

Cabe señalar que, para un gran número de usuarios $(i \gg 1)$ y un sistema biométrico de alta precisión $(p \ll 1)$. En la ecuación anterior puede volverse numéricamente inestable y para estimar el número de usuarios rechazados en este caso. Es posible aplicar una expansión de Taylor a la parte crítica de la ecuación considerando valores pequeños de p.

$$(1-p)^i = 1 - ip + \frac{1}{2}(i-1)ip^2 + O(i^3p^3)$$

Sustituyendo este resultado en la ecuación previa, se obtiene que:

$$N_{\mathrm{FM}}(i) = \frac{1}{2}(i-1)ip + O(i^3p^2) \approx \frac{1}{2}(i-1)ip$$

Esta es una aproximación válida siempre que se cumpla la condición:

$$i^2p \gg i^3p^2 \leftrightarrow ip \ll 1.$$

Para los usuarios fraudulentos, la probabilidad de que no ocurra una coincidencia permanece constante y no se ve afectada por el aumento en el número de usuarios del sistema. Solo existe un iris que pueda generar una falsa no coincidencia, cuando el propio iris del usuario que ya está registrado previamente. Por lo tanto, la probabilidad de que se produzca una falsa no coincidencia está determinada por:

$$\mathrm{FMR}_{\mathrm{AND}} = \mathrm{FMR}^2$$

El cálculo del número esperado de *FMR* se obtiene mediante:

$$N_{\mathrm{FNM}}(j) = \mathrm{j} \cdot P_{FNM} = j \cdot \mathrm{FNMR}$$

donde j representa al usuario número j no confiable que intenta engañar al sistema [153].

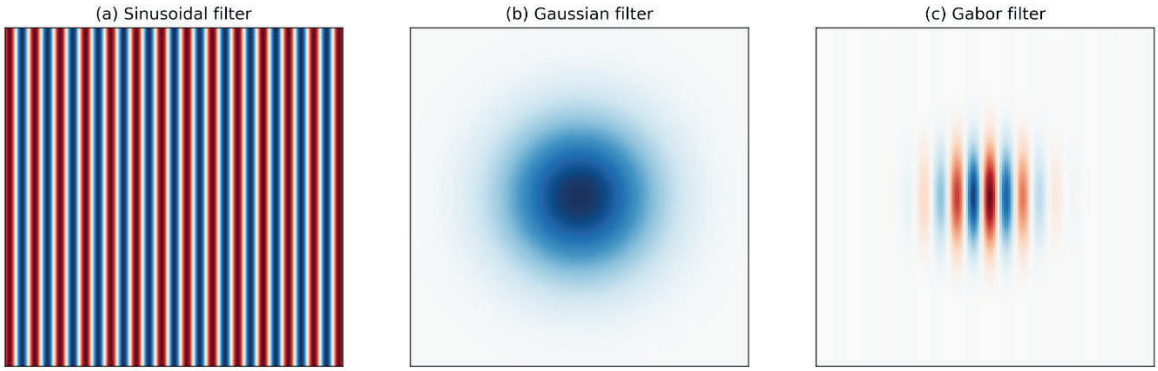

Figura B.2 Etapas de construcción de un filtro de Gabor

Fuente: https://whitepaper.world.org/images/whitepaper/notion2/image63.png

B.4 Filtrado de Gabor

Un *filtro de Gabor* [158] representa de manera muy similar la corteza visual que el ser humano percibe y distingue las texturas. Permite analizar el contenido representados en frecuencias en una dirección determinada dentro de la región de trabajo de una imagen. Es utilizada por el alto nivel de procesamiento de señales e imágenes debido a su capacidad de compresión en el dominio espacial y de frecuencia.

Como se muestra en la *Figura B.2*, un *filtro de Gabor* puede interpretarse como una señal sinusoidal de una frecuencia y orientación específicas, modulada por una onda gaussiana y matemáticamente, se define como:

$$G_{\lambda,\theta,\phi,\sigma,\gamma}(x,y) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right)\exp\left(j\left(2\pi\frac{x'}{\lambda} + \phi\right)\right)$$

Donde

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Entre los parámetros, σ y γ representan la desviación estándar y la relación de aspecto espacial de la envoltura gaussiana. Los parámetros λ y ϕ corresponden a la longitud de onda y el desfase del componente sinusoidal, mientras que θ determina la orientación de la función de *Gabor*. Según su configuración, un filtro de *Gabor* puede captar mejor las relaciones entre píxeles descritas por bandas espectrales estrechas y al mismo tiempo, tiene una compacidad espacial que permite adaptarse a irregularidades en la imagen.

Dado que un filtro de *Gabor* es complejo, sus componentes real e imaginarios funcionan como dos filtros en cuadratura, como se muestra en la *Figura B.3*. Donde (a) es la parte real del filtro que tiene simetría par y genera una alta respuesta en estructuras de líneas, mientras que (b) es la parte imaginaria del filtro que tiene una simetría impar y muestra una alta sensibilidad a características como bordes. En el filtro de simetría par, se tiene que garantizar que tenga un componente de corriente continua igual a cero, mientras que el filtro de simetría impar ya cumple con esta condición de forma natural. Esto asegura que

el filtro no genere respuestas uniformes en la imagen, sin importar la intensidad de esta [158].

B.5 Filtrado de Gabor en múltiples escalas

El iris se distribuye en múltiples escalas, que son reguladas por el parámetro σ, que de manera natural se representan utilizando filtros de diferentes tamaños. En la mayoría de los sistemas de filtrado de múltiples escalas siguen el principio de construcción de wavelets. Los filtros de cada nivel son versiones escaladas de los del nivel anterior y, a su vez, versiones escaladas de una wavelet base, reduciendo la redundancia y permite una representación más compacta. Estas pueden ajustarse en distintas orientaciones, definidas por θ, en la *Figura B.4,* se muestra la parte real de 28 wavelets de Gabor distribuidos en cuatro escalas y siete orientaciones [159].

B.6 Demodulación y codificación por cuadrantes de fase

Después de que se aplique el filtro de Gabor a una imagen del iris, se somete a un proceso de demodulación por cada zona analizada para poder extraer su información de fase, este procedimiento se representa en la *Figura B.5*.

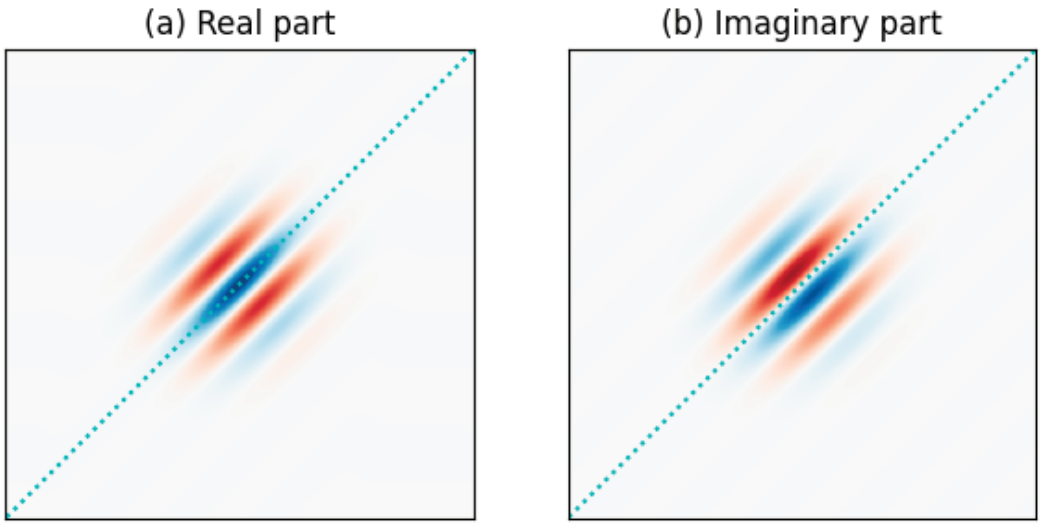

Figura B.3 Parte real e imaginaria de un filtro de Gabor

Fuente: https://whitepaper.world.org/images/whitepaper/notion2/image47.png

Es importante destacar que solo se registra la información de fase, ya que es más robusta que la magnitud y puede verse afectado por elementos externos como el contraste de imagen, la ganancia de la cámara o la iluminación.

Una de las ventajas de la demodulación por cuadrantes de fase es que genera un código cíclico, siendo más eficiente que un código binario tradicional. Puede cambiar dos bits, lo que hace que ciertos errores, mientras que el código cíclico solo permite una variación de un bit en la rotación entre cuadrantes de fase adyacentes y en caso de que una respuesta este cerca del límite entre dos cuadrantes.

El bit resultante se considera frágil porque suelen ser menos estables y pueden cambiar de valor debido a variaciones en la iluminación, desenfoque o ruido. Existen diversas estrategias para procesar los bits frágiles, y una de ellas es asignarles un peso menor durante el proceso de comparación.

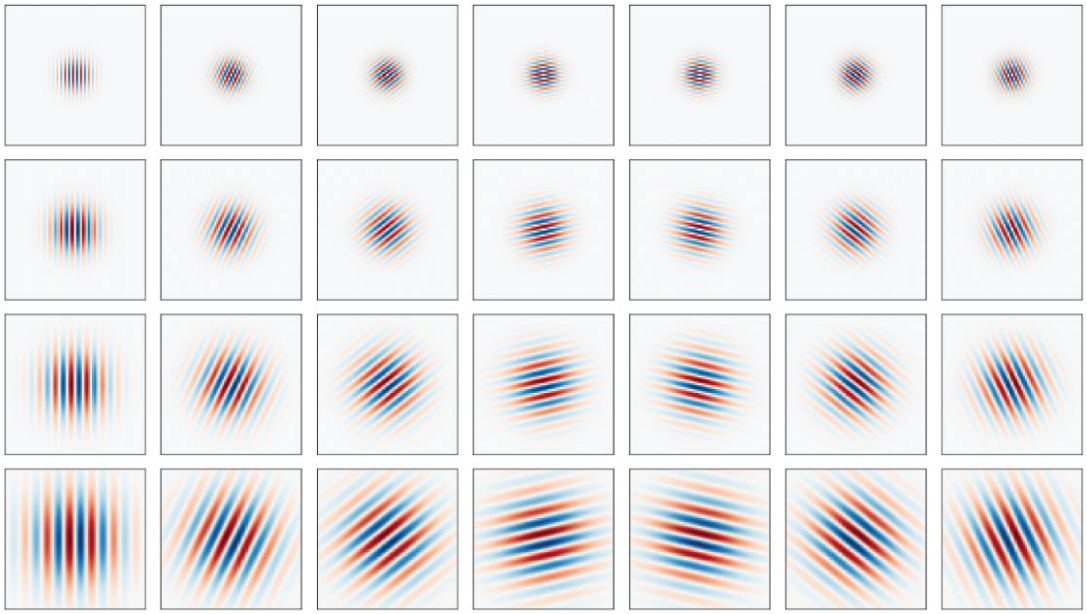

Figura B.4 Construcción de 28 wavelets de Gabor

Fuente: https://whitepaper.world.org/images/whitepaper/notion2/image69.png

Cuando se aplica el filtrado de Gabor en múltiples escalas a una imagen del iris, se generan varios códigos de iris que luego se concatenan para formar la plantilla final.

Dependiendo del número de filtros utilizados y sus factores de desplazamiento, la plantilla del iris puede ser de varios órdenes de magnitud más pequeña que la imagen original [160].

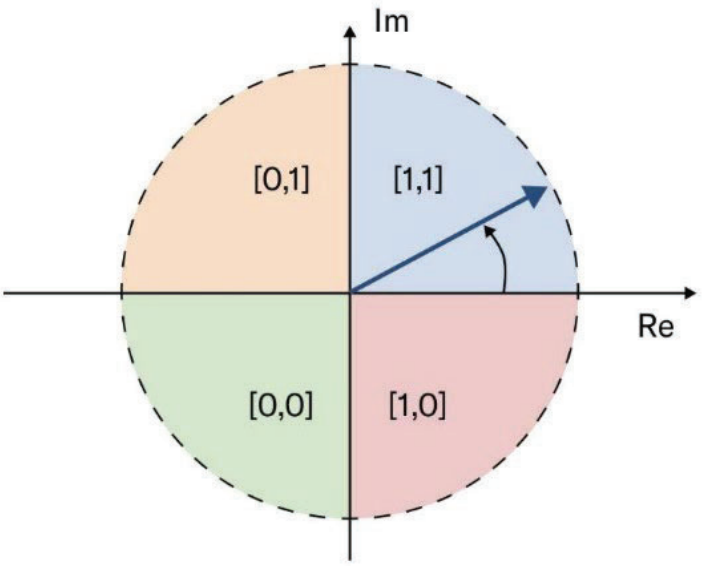

Figura B.5 Demodulación de la información por fase

Fuente: https://whitepaper.world.org/images/whitepaper/notion2/image30.jpg

B.7 Código del iris

Como ya sabemos los códigos iris se generan a partir de las respuestas de fase obtenidas mediante el filtrado de Gabor. Esto son altamente resistentes a las variaciones externas, como la iluminación, desenfoque y el ruido. Esta resistencia se tiene que medir de manera cuantitativa para saber la desviación del código iris resultante comparándolo con el original. Este se aplican progresivamente tres tipos de alteraciones a la imagen del iris que son, la corrección gamma para modificar la iluminación, filtrado gaussiano para introducir desenfoque y ruido gaussiano [161].

El nivel de alteración introducido en la imagen se mide utilizando el *Root Mean Square Error* (*RMSE*). Para calcula la diferencia entre los valores de los píxeles de la imagen original I y la imagen modificada I', donde *RMSE* se define, siendo N el número total de píxeles en ambas imágenes como:

$$\mathrm{RMSE} = \sqrt{\frac{1}{\mathrm{N}}\sum_{p=1}^{N}(I'_p - \mathrm{I}_p)^2}$$

Luego para medir el grado de cambio en el código de iris, se emplea la *HD*, comparando los bits de los códigos de iris original C y modificado C'. Se expresa como:

$$HD = \frac{1}{K}\sum_{P=1}^{K}\left|C'_p - C_p\right|$$

Donde K es la cantidad total de bits del código iris, siendo el valor *0* la *HD* que indica una coincidencia perfecta, mientras que el valor *1* significa que códigos del iris son completamente opuestos. La *HD* tiene un promedio entre los dos códigos generados aleatoriamente es de aproximadamente *0.5*.

Figura B.6 Niveles cambiantes de iluminación de los códigos iris

Fuente: https://whitepaper.world.org/images/whitepaper/notion2/image16.gif

La *Figura B.6* muestra cómo afecta la iluminación, el desenfoque y el ruido a la robustez de los códigos iris generados, tanto de manera visual como cuantitativa, siendo estos resultados no fueron generados utilizados los filtros en un sistema real. Se reflejan las propiedades generales del filtrado de *Gabor*, la imagen del iris ha sido normalizada, transformando su forma original de anillos en coordenadas cartesianas a una representación rectangular de tamaño fijo en coordenadas polares. Este paso es necesario para poder estandarizar el formato, eliminar oclusiones y resaltar la textura del iris.

Como se muestra en la *Figura B.7*, los códigos de iris son altamente resistentes a transformaciones de niveles de gris asociadas a cambios en la iluminación, ya que la *HD* apenas varía con el aumento del *RMSE*. Esto se debe a que un incremento en el brillo de los píxeles, que reduce el rango dinámico de los valores de intensidad, pero no afecta significativamente las características espaciales o de frecuencia de la textura del iris.

Figura B.7 Desenfoque de los códigos iris

Fuente: https://whitepaper.world.org/images/whitepaper/notion2/image31.gif

El desenfoque, puede reducir el contraste de la imagen y afectar la textura del iris, aunque como se muestra en la *Figura B.8*. Los códigos de iris mantienen su estabilidad incluso cuando el desenfoque es tan fuerte que la textura del iris se vuelve imperceptible a simple vista.

Esto se debe a que la información de fase capturada mediante el filtrado de *Gabor* registra la ubicación y presencia de la textura, en lugar de su intensidad, siempre y cuando que las propiedades espaciales o de frecuencia de la textura del iris permanezcan. Aunque estén significativamente atenuadas, los códigos de iris seguirían siendo consistentes.

Es importante resaltar que el desenfoque afecta principalmente las texturas de alta frecuencia, lo que impacta más a los filtros de Gabor de alta frecuencia. Por esta razón que se emplea un conjunto de filtros de Gabor en múltiples escalas para mejorar la robustez del sistema.

Figura B.8 Ruido en los códigos iris

Fuente: https://whitepaper.world.org/images/whitepaper/notion2/image29.gif

Por otro lado, cuando se introduce ruido gaussiano, se puede observar cambios más significativos en los códigos de iris, ya que tanto los componentes espaciales como los de frecuencia de la textura se ven alterados. De esta manera se genera un mayor número de bits frágiles a medida que el ruido aumenta y la textura del iris se vuelve irreconocible. La desviación en los códigos de iris sigue siendo relativamente baja, con una *HD* inferior a 0.2, estando muy por debajo del promedio. Esto resultados demuestran la eficacia del uso de filtros de *Gabor* para la generación de características del iris, incluso en condiciones de ruido significativo [161].

B.8 Sistema de inferencia del iris

En esta etapa del proceso es donde transforma las imágenes infrarrojas de alta resolución de los ojos izquierdo y derecho en un código de iris, representando de forma matemática y abstracta la entropía del iris, permitiendo verificar la unicidad a gran escala. La investigación de John Daugman [162], sigue siendo el método más utilizado para representar la textura del iris en el campo del reconocimiento biométrico. El sistema desarrollado por *World*, siguen las cuatro etapas principales del método de Daugman que serían, segmentación, normalización, generación de características y comparación [163].

En la *Figura B.9* se muestra un ejemplo de un iris capturado en el espectro cercano al infrarrojo. El lado derecho de la imagen, se muestra el código de iris correspondiente, el cual se compone de $n_f = 2$ mapas de respuesta generados por dos wavelets de Gabor bidimensionales.

Los mapas de respuesta se cuantizan en dos bits, lo que da como resultado un código de iris con dimensiones $n_h \times n_w \times n_{\mathcal{F}} \times 2$, donde n_h y n_w son los que representan el número de posiciones radiales y angulares en las que se aplican estos filtros. Aunque en la imagen solo se muestra el código de iris de un ojo, es importante destacar que una plantilla biométrica completa incluye los códigos de iris de ambos ojos [164].

Segmentación

La segmentación de imágenes de iris en alta resolución capturadas en el espectro infrarrojo ha sido propuesta por *Tools for Humanity* en el estudio de *Lazarski et al* [165]. Utilización de este modelo, que cuenta con un codificador compartido por dos decodificadores. El primero se encarga de estimar la geometría del ojo (pupila, iris y globo ocular) y otro enfocado en el ruido (elementos ajenos a la estructura ocular que pueden superponerse y afectar la visibilidad de la textura del iris, como pestañas o mechones de cabello).

Al estar separada por dos decodificadores permite un procesamiento más eficiente de los elementos superpuestos y otorga una mayor flexibilidad en el entrenamiento de los detectores. Esta arquitectura basada en *DeepLabv3+* [166], con un modelo *MobileNet v2* [167] como columna vertebral. Distinguir los elementos de ruido es un proceso mucho más complicado que etiquetar la geometría del ojo, ya que requiere un alto nivel de precisión para identificar cada detalle. Dependiendo del grado de desenfoque y la estructura ocular del usuario, el etiquetado de pestañas en una sola imagen puede tardar entre 20 y 80 minutos, mientras que el etiquetado de la geometría del ojo con la precisión requerida toma aproximadamente 4 minutos.

Para poder optimizar este proceso, los elementos de ruido se separan de los elementos geométricos, lo que permite reducir significativamente los costos y el tiempo de procesamiento sin comprometer la calidad de los datos.

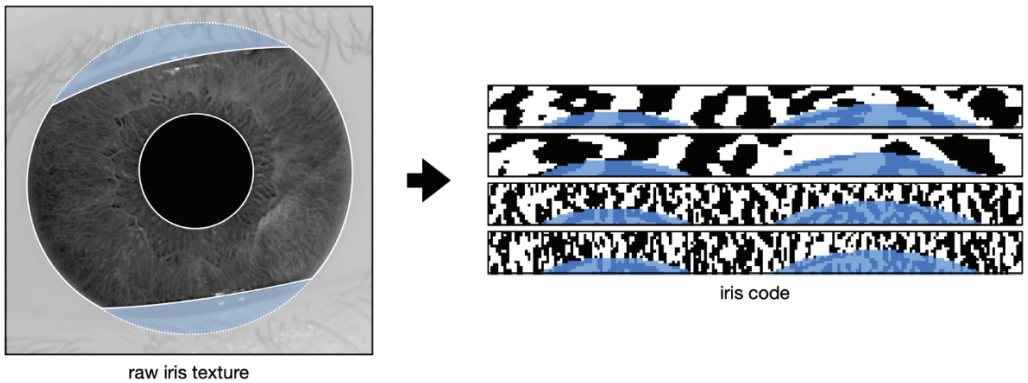

Figura B.9 Entrada y salida del proceso biométrico

Fuente: https://whitepaper.world.org/images/whitepaper/notion2/image11.png

Este modelo fue entrenado mediante la combinación de *Dice Loss* y *Boundary Loss*, donde la función de perdida *Dice Loss* se expresa de la siguiente manera:

$$L_D = \sum_k \left(1 - \frac{2\sum_{i,j} y_{i,j,k} \cdot p_{i,j,k}}{\sum_{i,j} y_{i,j,k}^2 \cdot \sum_{i,j} p_{i,j,k}^2}\right)$$

Donde los valores $\gamma_{i,j,k} \epsilon \{0,1\}$ representan la verdad fundamental codificada en *one-hot encoding*, mientras que $P_{i,j,k} \epsilon [0,1]$ corresponde a la salida del modelo para el píxel (i,j). El tercer índice k es la que indica la categoría a la que pertenece el píxel, como pupila, iris, globo ocular, pestañas o fondo.

La función *Dice Loss* mide esencialmente la similitud entre dos elementos, que son las etiquetas reales y la predicción del modelo. La detección precisa de los límites del iris es fundamental para un reconocimiento confiable. Incluso una pequeña distorsión en el contorno puede provocar una deformación en la imagen normalizada a lo largo de la dirección radial. Para solucionar este problema, se introdujo una función de pérdida adicional basada en una entropía cruzada ponderada, para enfatizar las zonas en los límites entre las clases, para fomentar contornos más definido, se expresada como:

$$L_B = \sum_{ij} \sum_k b_{i,j,k} \cdot \mathcal{Y}_{i,j,k} \cdot \log\left(P_{i,j,k}\right)$$

Donde $b_{i,j,k}$ representa el peso del límite, indicando qué tan cerca está el píxel (i,j) de la frontera entre la clase k y para mejorar la precisión del modelo en la detección exacta de los bordes sin perder estabilidad en la región. Se aplica un desenfoque gaussiano al contorno.

$$b_{i,j,k} = G(d(i,j,S_k))$$

Aquí, $d(i,j,S_k)$ denota la distancia entre el punto (i,j) y la superficie S_k. Definida como la distancia euclidiana mínima entre (i,j) y todos los puntos de S_k. Representada la frontera entre la clase k y las demás, siendo la función G correspondiente a una distribución gaussiana centrada en cero con una varianza finita.

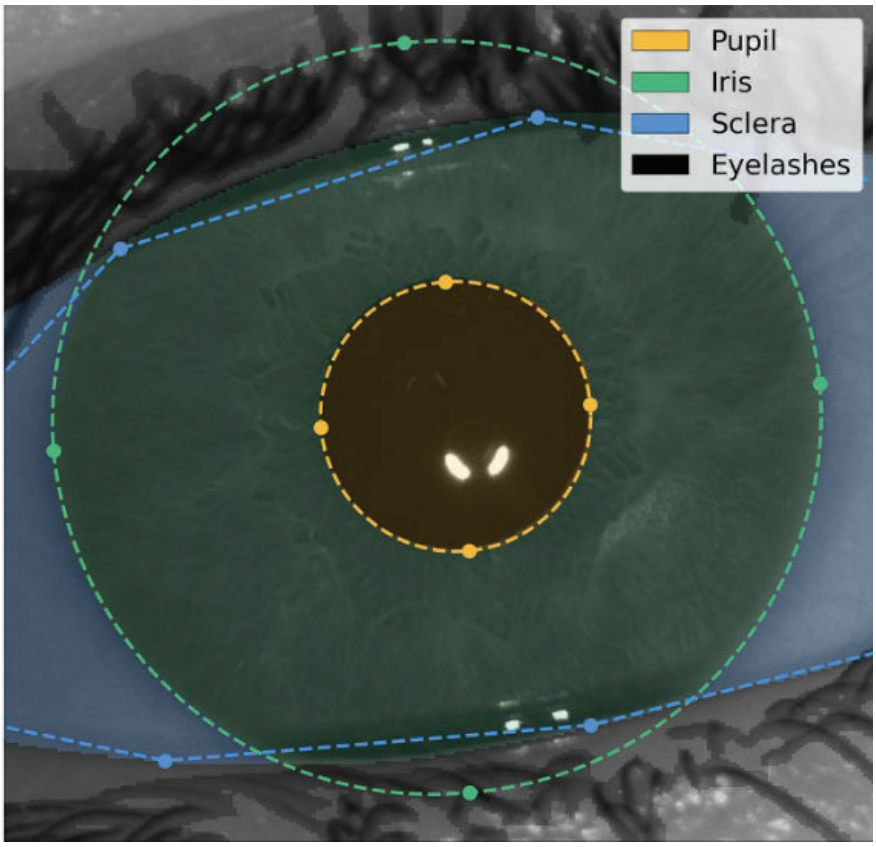

Figura B.10 Segmentación de una imagen del iris

Fuente: https://whitepaper.world.org/images/whitepaper/notion2/image65.png

En siguiente *Figura B.10* se muestra una imagen del iris superpuesta con los mapas de segmentación del modelo. Se incluyen puntos de referencia generados por un modelo de inteligencia artificial encargado de evaluar la calidad de la imagen durante la captura. Proporciona métricas de calidad que garantizan que solo se utilicen imágenes óptimas en la fase de segmentación, para poder asegurar que el código de iris se genere con una precisión necesaria para poder verificar la unicidad del usuario [168].

Normalización

La función de la normalización es aislar la textura relevante del iris, separándola de la piel, pestañas y esclerótica. Esto lo hace transformando la textura del iris en un sistema de coordenadas cartesianas a un sistema de coordenadas polares, como se muestra en la *Figura B.11*. Esta orientación del iris se define como un vector que conecta al centro de la pupila de un ojo con el centro de la pupila del otro ojo opuesto.

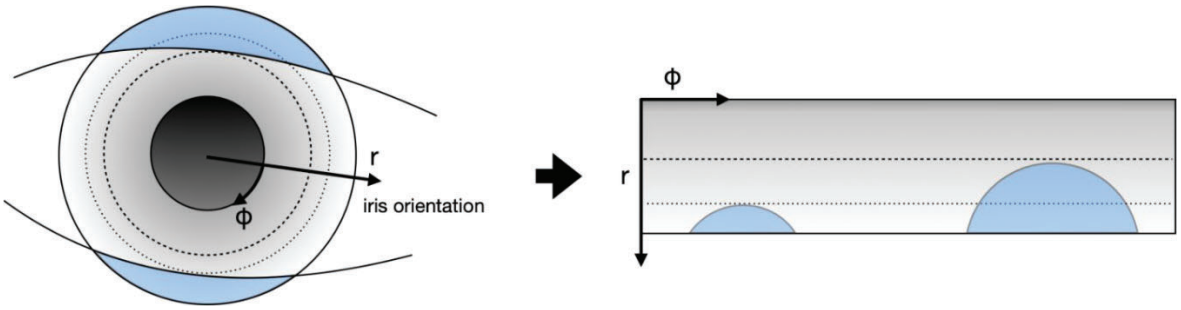

Figura B.11 Proceso de normalización

Fuente: https://whitepaper.world.org/images/whitepaper/notion2/image2.png

Esto reduce la variabilidad de la imagen al compensar los factores que existen a la hora de capturar la imagen. Puede ser la distancia de la persona a la cámara, la contracción o dilatación de la pupila en respuesta a la iluminación del entorno y la inclinación de la cabeza. En la *Figura B.12,* se muestra la versión normalizada del iris representado en la *Figura B.11*. Estos dos arcos de circunferencia visible en la imagen corresponden a los parpados, cuya forma original se ha distorsionado durante el proceso de normalización [169].

Generación de características

Una vez obtenida la textura del iris segmentada y normalizada, es posible generar un código de iris, como se muestra en la *Figura B.13*. Se aplican múltiples filtros de Gabor para obtener una respuesta compleja de dos bits por cada punto seleccionado y tener finalmente el código iris [170].

Comparación y coincidencia

Con la textura del iris ya convertida en un código iris, esta se compara con otros códigos para determinar si existe alguna coincidencia. De esta manera se utiliza el método de *HD* fraccional enmascarada. Se encarga de medir la proporción de bits del código de iris que coinciden en ambas muestras, sin considerar los bits enmascarados.

Debido a la parametrización de las wavelets de Gabor, cada bit tiene la misma probabilidad de ser 0 o 1, ya que los códigos de iris contienen más de 10,000 bits. Se generan dos códigos a partir de ojos diferentes, teniendo como promedio una *HD* de 0.5,

con el 99.95% de los códigos variando en menos de 0.05 *HD* con respecto a este valor, y el 99.9994% variando en menos de 0.07 *HD*.

Para mejorar la precisión, se comparan varias rotaciones del código de iris hasta encontrar la alineación con mayor probabilidad de coincidencia. Reduciendo el promedio de 0.5 *HD* a 0.45 *HD*, con una probabilidad de 1.6×10^{-7} de que sea inferior a 0.38 *HD*. Esta reducción del promedio hace extremadamente improbable que dos ojos distintos generen códigos de iris con una distancia inferior a 0.38 *HD*. Mientras que dos imágenes del mismo ojo producirían códigos con una distancia generalmente menor a 0.3 *HD*. Estableciendo un umbral entre estos valores, la posibilidad de distinguir de manera confiable entre identidades diferentes y coincidentes [171].

Luego para poder validad el rendimiento del algoritmo a gran escala, evaluaron su precisión utilizando un conjunto de 2.5 millones de pares de imágenes de iris en alta resolución obtenidas por 303 personas. Cada uno se distinguía por su variedad en términos de color de ojos, tono de piel, edad, presencia de maquillaje y condiciones oculares, tomando esta variedad de imagen y verificar sus identidades. Se mide la *FMR* y la *FNMR*, permitiendo evaluar el desempeño del sistema de manera objetiva y cuantificable, como se muestra en la *Figura B.14*.

La distribución de coincidencias presenta dos picos bien definidos. El primer pico ubicado en *HD* ≈ 0.08, corresponde a la *HD* media entre imágenes del mismo individuo capturadas en el mismo proceso de registro. Esto indica un alto grado de similitud.

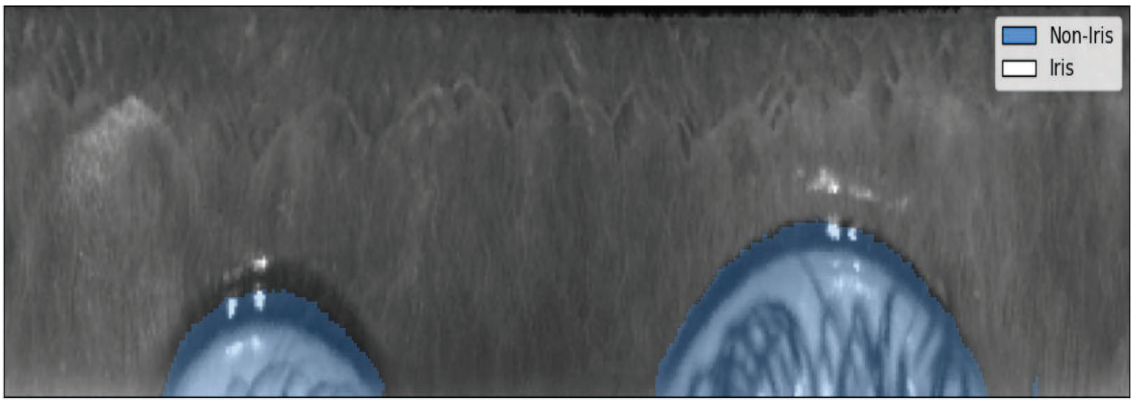

Figura B.12 Textura del iris normalizada

Fuente: https://whitepaper.world.org/images/whitepaper/notion2/image26.png

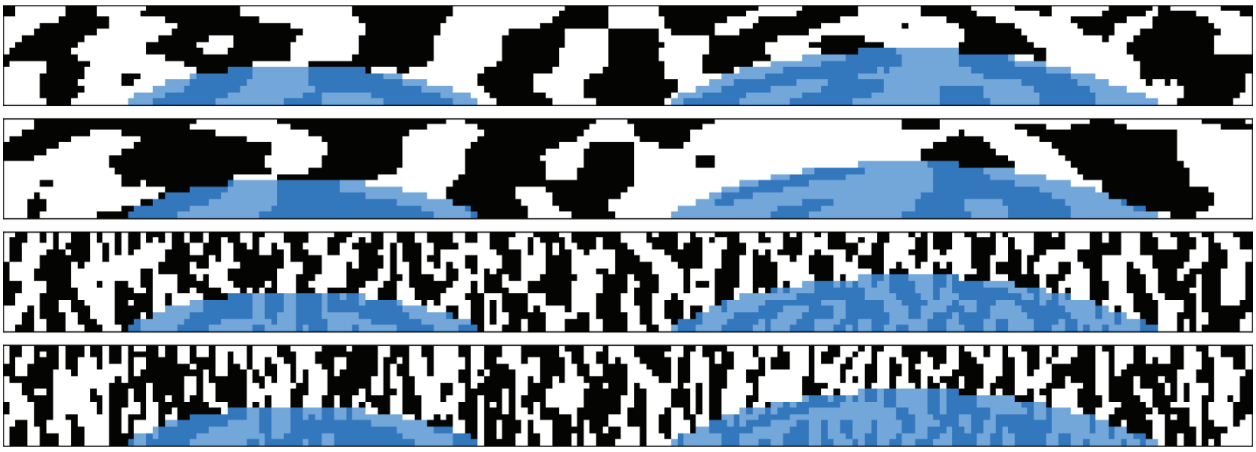

Figura B.13 El código iris final

Fuente: https://whitepaper.world.org/images/whitepaper/notion2/image23.png

Como es de esperarse en imágenes tomadas en condiciones casi idénticas y para el segundo pico ubicado en *HD* ≈ 0.2. Representa la *HD* media entre imágenes del mismo individuo tomadas en procesos de registro diferentes, con un intervalo de tiempo entre ellos que puede ser de varias semanas. Demostrando que estas imágenes tienen una menos similitud debido a la variación natural como la dilatación de la pupila, oclusiones parciales y la presencia de pestañas.

Este sistema sigue mejorando, ya que están implementando continuamente mejoras desde el sistema de autoenfoque más avanzados, optimizando la interacción entre hardware y modeles de inteligencia artificial, filtros de calidad en tiempo real más efectivos y la reducción de ruido en la imagen.

Para poder reducir la dispersión en la distribución de coincidencias y mejorar la precisión del sistema, se están implementando continuamente mejoras, incluyendo: sistemas de autoenfoque más avanzados, optimización de la interacción entre hardware y modelos de inteligencia artificial, filtros de calidad en tiempo real más efectivos, generación de características mediante aprendizaje profundo y reducción de ruido en la imagen.

Dado que no se encontraron pares de iris clasificados incorrectamente, no es posible calcular con precisión las *FMR* y *FNMR*. Pero si es posible estimar un límite superior para ambas tasas:

$$FMR = \frac{n_{FM}}{n_{TM} + n_{FM}} < \frac{1}{2.4 \cdot 10^{7}} = 4.1 \cdot 10^{-8}$$

$$FNMR = \frac{n_{FNM}}{n_{TNM} + n_{FNM}} < \frac{1}{2.4 \cdot 10^{4}} = 2.4 \cdot 10^{-5}$$

Con estos resultados, se verifica la unicidad a escala de mil millones de personas que puede lograrse con una precisión extremadamente alta. También se reconoce que el conjunto de datos utilizado en esta evaluación podría ampliarse. Es necesario un mayor esfuerzo para desarrollar bases de datos más extensas y diversas que permitan estimar el rendimiento biométrico de manera precisa [171].

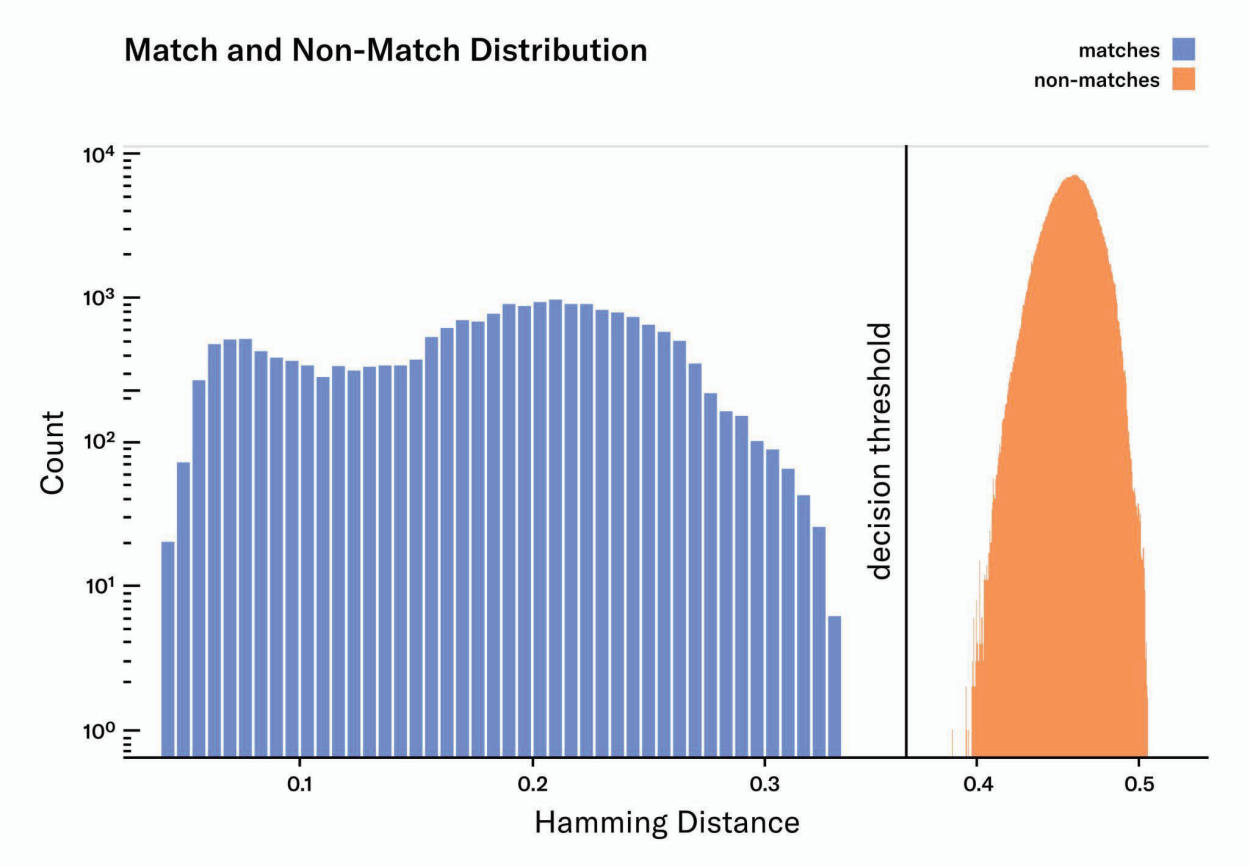

Figura B.14 Distribución de coincidencias y no coincidencias

Fuente: https://whitepaper.world.org/images/whitepaper/notion2/image39.jpg

Mejoras en los códigos de iris

Aunque la precisión del algoritmo actual de verificación de unicidad del *Orb* ya es extremadamente alta, con una tasa de coincidencias falsas de 1 en 40 billones en comparaciones 1:1. Para lograr una precisión aún mayor sería beneficioso a escala de mil millones de personas, por eso el algoritmo biométrico se encuentra en constante desarrollo y se actualizaría con el tiempo. Existen tres tipos principales de mejoras que pueden aumentar aún más la precisión del sistema:

- *Preprocesamiento de imágenes*: los elementos que se pueden optimizarse es la red de segmentación y los umbrales de calidad de las imágenes. Estas actualizaciones son compatibles con versiones anteriores, ya que modifican todo el proceso excepto el paso final de generación del código de iris. Estas mejoras las suelen implementar varias veces al año.
- *Generación de códigos de iris para futuras verificaciones*: también compatibles con versiones anteriores, estas actualizaciones modifican el algoritmo de generación de códigos de iris sin necesidad de recalcular los códigos previamente almacenados. Al introducir nuevos códigos *v2*, los cuales no son compatibles con los códigos *v1* que son más antiguos, entonces lo que hacen es comparar los códigos por separado, y si no se detecta ninguna coincidencia en ambos, el nuevo código *v2* se agrega a su correspondiente conjunto. De esta manera los grupos de código *v1* deja de aumentar, garantizando que ningún usuario con un *World ID* basado en *v1* pueda registrarse con una segunda identidad. La tasa de coincidencia evoluciona cuando se actualiza el algoritmo de generación de características de la siguiente manera:

$$P_{FM}(i) = 1 - P_{no\ match\ at\ all} = 1 - P_{no\ match\ in\ v1} \cdot P_{no\ match\ in\ v2}$$
$$P_{FM}(i) = 1 - (1 - FMR_1)^{n1} \cdot (1 - FMR_2)^{i-1-n1}$$

A partir de la ecuación correspondiente en la sección de rendimiento biométrico a escala de mil millones de personas, se deduce que la probabilidad de que un usuario legítimo número i experimente una coincidencia falsa sigue aumentando a medida que el conjunto *v2* se expande. En caso de que se cumple

la condición $FMR_2 < FMR_1$, la tasa de crecimiento de este error se reduce significativamente. Para los usuarios que forman parte del conjunto *v1*, la *FNMR* no se ve afectada, pero en cambio, para los nuevos registros, se aplica la *FNMR* correspondientes al algoritmo *v2*. Este tipo de actualización se espera que ocurra aproximadamente una vez al año.

- *Recálculo de códigos de iris existentes*: dependiendo de si la imagen original del iris aún está disponible, estas actualizaciones pueden o no ser compatibles con versiones anteriores, la idea es disminuir la frecuencia de actualización del código iris con el tiempo. Para determinar cuándo es necesario un recálculo, definimos n_1 como el número de códigos en el conjunto v1 y n_2 para el conjunto *v2*. En caso de que los códigos *v1* tengan una tasa de error significativamente mayores que los *v2* y afecten de manera crítica la tasa de coincidencias falsas incluso cuando $n_2 \gg n_1$. Entonces eventualmente debería recalcularse el conjunto *v1*, siendo necesario que las imágenes del iris sigan disponibles, lo cual puede lograrse de varias maneras:
    - *Recaptura de imágenes*: los usuarios pueden volver a un *Orb* para generar un nuevo código de iris, siendo este método poco práctico ya que esto dependería de la accesibilidad a un *Orb* y de las preferencias individuales.
    - *Almacenamiento de imágenes en custodia propia*: este método permite a los usuarios almacenar sus imágenes firmadas y cifradas de extremo a extremo en sus propios dispositivos, donde los usuarios en caso de que haga falta recalcular el código iris se recalculan de manera local en su teléfono móvil. Este método requiere que la actualización del código iris se realice mediante un *ZKML* en el dispositivo del usuario [172].