VERAPI

UNIVERSIDAD
POLITÉCNICA
DE MADRID

VeraPi – Diseño e implementación de un TV-Box basado en Raspberry Pi para teleasistencia cognitiva y gestión remota de la plataforma Vera

Autor : Javier Díez Tejeda
Tutor : Borja Bordel Sánchez
Depto. Sistemas Informáticos

Índice

Listado de figuras

1. Introducción

1.1 Contexto y motivación

El envejecimiento de la población y el aumento de situaciones de soledad no deseada constituyen dos retos de primer orden para los sistemas de atención sociosanitaria. En España, la proporción de personas mayores de 65 años es creciente y se prevé que siga aumentando en las próximas décadas [1]. Este fenómeno tensiona los servicios de atención tradicionales (centros de día y residencias), que presentan limitaciones de capacidad, cobertura territorial y horarios.

En este contexto surge VERA, una propuesta de atención psicosocial domiciliaria que aprovecha el televisor del hogar como interfaz principal para ofrecer acompañamiento, actividades personalizadas, videoconferencia con profesionales y familiares, y seguimiento continuo por parte de equipos sociosanitarios [2]. La solución se apoya en tecnología propia orientada a la facilidad de uso (instalación "plug & play", videollamada integrada en TV, monitorización remota) y en un modelo de servicio que prioriza la personalización y la continuidad de la atención.

No obstante, la sostenibilidad económica y operativa de un despliegue masivo de VERA depende en gran medida del dispositivo instalado en el domicilio del usuario. El dispositivo actual —un cliente conectado al televisor que habilita las funcionalidades de VERA— condiciona el coste unitario, la experiencia de usuario y la capacidad de operación remota. En particular, se han observado incidencias que afectan a la calidad de la experiencia así como costes asociados al soporte in situ cuando la resolución remota no es suficiente.

La motivación de este Trabajo Fin de Grado es precisamente identificar, adaptar e integrar un dispositivo alternativo de coste contenido que impulse el proyecto VERA, reduciendo el coste del hardware domiciliario y, sobre todo, mitigando las incidencias que actualmente frenan y degradan la experiencia de usuario. La hipótesis de partida es que una plataforma de propósito general ampliamente soportada (por ejemplo, Raspberry Pi) —debidamente endurecida, configurada en modo quiosco y dotada de herramientas de observabilidad y acceso remoto— puede proporcionar:

- Un coste unitario sustancialmente menor que el del dispositivo vigente, preservando las prestaciones necesarias para VERA (videoconferencia, reproducción multimedia, comunicaciones seguras y control por mando/televisor).

- Una reducción de la tasa y el impacto de incidencias operativas mediante: mayor control del sistema (SO abierto y ecosistema maduro), mecanismos de actualización controlada, monitorización proactiva y automatización de tareas de mantenimiento.

- Una experiencia de usuario más predecible (arranque controlado, UI estable, tiempos de instalación acotados) y una menor necesidad de desplazamientos técnicos.

La contribución esperada del proyecto es doble: por un lado, habilitar una senda de despliegue escalable y económicamente viable de VERA; por otro, elevar el nivel de fiabilidad y satisfacción de las personas usuarias y sus familias al reducir la fricción tecnológica que, en ocasiones, impide aprovechar plenamente el servicio [2][3].

Figura 1. Logotipo de VERA (OHLA Ingesan).

1.2 Objetivos del trabajo

El propósito de este trabajo es diseñar y consolidar un nuevo modelo de dispositivo domiciliario de coste contenido que actúe como cliente de VERA, sosteniendo —e idealmente mejorando— la experiencia de las personas usuarias y la eficiencia operativa del servicio. La motivación central es doble: por un lado, reducir el coste del hardware instalado en el hogar para favorecer la escalabilidad del proyecto; por otro, resolver de raíz las incidencias que hoy dificultan el uso cotidiano de VERA y deterioran la satisfacción de usuarios y familias, como bloqueos de pantalla, desconexiones o fallos tras actualizaciones, así como la necesidad de asistencias presenciales para tareas que deberían resolverse de forma remota.

En este marco, el trabajo se orienta a seleccionar una plataforma ampliamente soportada y disponible en el mercado, integrarla con los componentes de VERA y dotarla de un conjunto de configuraciones y servicios que garanticen una operación estable. Se persigue la plena compatibilidad con televisores y periféricos habituales, una videoconferencia fluida y fiable, y un comportamiento consistente en el tiempo, con énfasis en la simplicidad para la persona usuaria.

De manera complementaria, se busca mejorar la experiencia desde el primer encendido: arranque directo de la interfaz de VERA en modo quiosco, reconexión automática ante pérdidas de conectividad y recuperación controlada ante caídas de procesos o apagados inesperados. Asimismo, se pretende estandarizar la instalación en domicilio mediante un procedimiento claro, reproducible y con tiempos acotados, minimizando la intervención técnica y los errores humanos.

La dimensión operativa y de seguridad resulta igualmente prioritaria. El dispositivo deberá exponer telemetría básica para diagnóstico, permitir acceso remoto seguro para soporte y mantenimiento, y admitir actualizaciones controladas con capacidad de recuperación ante fallos.

En conjunto, el éxito del trabajo se valorará por su capacidad para facilitar el despliegue a gran escala de VERA con un coste unitario menor, una tasa de incidencias reducida y una satisfacción de uso superior respecto a la línea base del dispositivo actualmente en servicio.

Figura 2. Kit de hardware candidato para el cliente domiciliario

1.3 Metodología de trabajo

El proyecto se aborda como un ejercicio de ingeniería aplicada con ciclos iterativos de construcción y evaluación. La estrategia combina la lógica de investigación orientada, favoreciendo retroalimentación temprana y toma de decisiones basada en evidencias.

La primera fase se centra en comprender a fondo el servicio y su casuística operativa. Se levantan requisitos funcionales y no funcionales, se analizan las incidencias más recurrentes y su impacto en la experiencia de las personas usuarias y en la operación, y se establecen criterios observables de éxito que guiarán la verificación posterior.

En una segunda etapa se realiza la exploración tecnológica. Se comparan alternativas de hardware y software atendiendo al coste total de propiedad, el rendimiento, la disponibilidad y el ciclo de vida, el soporte de periféricos habituales, la capacidad de gestión remota y las implicaciones de seguridad y privacidad. El objetivo es seleccionar una plataforma robusta y ampliamente soportada que permita integrar de forma fiable las capacidades de VERA.

Sobre esa base se define la arquitectura del cliente domiciliario. Se especifican el sistema operativo y los servicios que se ejecutarán, el arranque en modo quiosco para presentar de forma directa la interfaz de VERA, la gestión de sesiones de videoconferencia, los mecanismos de observabilidad y las salvaguardas de seguridad. En paralelo se diseñan los procedimientos operativos de aprovisionamiento, instalación, soporte y actualización.

A continuación, se construye el prototipo y se integra con los componentes de VERA y con la solución de videoconferencia. Se habilita el control mediante HDMI,se validan cámaras y micrófonos compatibles y se prueba el correcto funcionamiento de la videollamada.

Concluida la verificación, se ejecuta un piloto en entorno preproductivo con personas usuarias reales o perfiles internos representativos. Se monitoriza el comportamiento del sistema, se realiza seguimiento cercano de las incidencias y se incorporan los aprendizajes al diseño, cerrando una versión candidata a liberación apta para un despliegue más amplio.

1.4 Estructura de la memoria

En esta sección se aborda la estructura que seguirá esta memoria para presentar una visión general de la misma, así como la información que se abordará en cada uno de los puntos.

En primer lugar, se presentan las bases teóricas del proyecto en el punto 2, el cual está dividido en dos partes bien diferenciadas.

En primer lugar, presentamos el contexto de la atención psicosocial domiciliaria y el papel de la televisión del hogar como interfaz accesible, así como los principios de diseño centrado en personas mayores y la relevancia de la simplicidad operativa para garantizar la adopción. Tras esto, se abordan con mayor detalle los fundamentos técnicos que soportan el servicio VERA y se justifica la elección de un enfoque basado en un dispositivo de propósito general de bajo coste frente a otras alternativas de cliente TV.

En segundo lugar, hablamos de las diferentes tecnologías que hemos utilizado en nuestro sistema, así como la justificación de su elección. Se describen el sistema operativo, el modo quiosco, los servicios de monitorización y acceso remoto, y los componentes de red y multimedia necesarios para la integración con VERA.

En el tercer punto de la memoria abordamos el análisis y el diseño del sistema. Para ello, primero se redactan los requisitos funcionales y no funcionales del cliente domiciliario; tras el análisis adecuado, se realiza un diseño que cumpla con todos los requisitos. Este diseño está dividido en dos partes: por un lado, la arquitectura del sistema (capas software, servicios, seguridad y actualizaciones) y, por otro lado, el diseño del esquema de telemetría y registros que permitirá observar el comportamiento del dispositivo y facilitar el soporte.

En el cuarto punto se expone el diseño del sistema basado en Raspberry Pi. Se describe la arquitectura general, las decisiones de conectividad y despliegue y la

integración con la plataforma de videollamadas. Asimismo, se introducen los mecanismos de gestión remota previstos (Raspberry Pi Connect y PiCockpit) y las consideraciones de seguridad y mantenimiento, incluyendo la protección del acceso, las políticas de actualización remota y el soporte.

El punto 5 recoge la implementación y el desarrollo del sistema diseñado. Se detallan la instalación y configuración del cliente, la puesta en marcha del modo quiosco y de los servicios necesarios, la monitorización y el control remoto durante la operación y, por último, los procedimientos de resolución de incidencias más habituales.
La validación del sistema se desarrolla en el punto 6, donde se definen la metodología de pruebas y los escenarios de evaluación en condiciones reales, atendiendo a funcionalidad, rendimiento, estabilidad y calidad de la experiencia de videollamada.

El punto 7 presenta las conclusiones del trabajo: los logros y contribuciones, las limitaciones y desafíos encontrados, las posibles mejoras y líneas futuras de desarrollo, y el análisis del impacto social, económico y medioambiental de la solución propuesta.

El punto 8 compila las referencias empleadas —bibliografía y recursos en línea— y añade un listado de figuras para facilitar la consulta.

Finalmente, el punto 9 reúne los anexos: el manual de instalación y configuración, el código fuente relevante, la documentación de herramientas utilizadas y ejemplos de casos de uso.

1.5 Alcance y condición del prototipo

Este trabajo describe un prototipo técnico desarrollado con fines de estudio y validación interna en el contexto del proyecto VERA. En su estado actual no constituye un producto listo para distribución pública o comercial. Quedan por completar las actividades formales de cumplimiento normativo y documentación legal requeridas para una puesta en producción, entre otras: (i) protección de datos y privacidad por diseño (evaluación de impacto si aplica, textos legales y contratos de encargo de tratamiento), (ii) gestión de licencias de software de terceros y elaboración del SBOM con atribuciones correspondientes, (iii) seguridad del ciclo de vida (política de parches, canal de actualización firmado de extremo a extremo, pruebas de penetración y hardening adicional), y (iv) conformidad de la entrega con hardware cuando se distribuye como kit (evidencias de conformidad/CE del conjunto, manuales de seguridad y uso).
Hasta completar estas tareas, cualquier despliegue debe considerarse piloto controlado en entornos de prueba y con datos no personales. La organización incorporará estas actividades en la siguiente fase de industrialización antes de una eventual distribución.

2. El Proyecto Vera en OHLA Ingesan

2.1 ¿Qué es Vera?

VERA es un servicio profesional de atención psicosocial domiciliaria que utiliza el televisor del hogar como interfaz principal para acompañar a personas en situación de soledad no deseada, fragilidad o dependencia ligera/moderada. Su objetivo es mejorar el bienestar, la autonomía y la conexión social sin exigir competencias digitales avanzadas: la persona usuaria controla el servicio con el mando de su TV y sigue un programa de actividades diseñado y supervisado por un equipo interdisciplinar (psicología, trabajo social y profesionales de dinamización).

A diferencia de aplicaciones convencionales en móvil u ordenador, VERA elimina barreras de acceso mediante una interfaz ya integrada en el hogar, pantallas grandes y audio claro, y una instalación "plug & play" de mínimo impacto. Esto permite replicar prestaciones de un centro de día virtual en el propio domicilio, combinando sesiones individuales y grupales, videollamadas con profesionales y contenidos de estimulación cognitiva y física en un entorno seguro.

El itinerario comienza con una valoración inicial para comprender necesidades, capacidades y preferencias. Con esa información se elabora un Plan de Actuación Individualizado (PAI) que fija objetivos y pautas. A partir del PAI se programan sesiones periódicas (estimulación cognitiva, ejercicio adaptado, dinamización social/cultural, entrenamiento de habilidades de la vida diaria y contacto con el entorno familiar). El equipo realiza seguimiento continuo, ajusta el PAI según la evolución y comunica hitos relevantes a las familias. El servicio incorpora avisos proactivos ante cambios de participación o incidencias detectadas.

En el domicilio se instala un dispositivo cliente conectado por HDMI al televisor. Este equipo arranca en modo quiosco para mostrar directamente la interfaz de VERA y se integra con la plataforma de videoconferencia, con mecanismos de reconexión y actualización remota para asegurar continuidad. El diseño prioriza la sencillez de uso, la gestión remota por parte del equipo técnico y la protección de datos (cifrado en tránsito, minimización de información en el dispositivo y control de accesos).

Agentes y beneficios

Personas usuarias: acompañamiento, estructura de actividades, reducción de la soledad y mantenimiento de capacidades en un entorno familiar.
Familias: visibilidad del proceso, comunicación con profesionales y tranquilidad al saber que existe seguimiento estructurado.
Profesionales: herramientas para planificar, conducir y evaluar sesiones; telemetría operativa que facilita el soporte y reduce desplazamientos.

Administraciones/entidades: intervención escalable y coste-eficiente, alineada con objetivos de envejecimiento saludable y reducción de la brecha digital.

Flujo operativo resumido

Derivación/alta: valoración inicial y definición del PAI.

Instalación del dispositivo en el domicilio, verificación de conectividad/audio/vídeo y videollamada de bienvenida.

Acompañamiento guiado: el usuario accede con el mando a sus sesiones programadas y a las videollamadas.

Seguimiento: registro de asistencia y participación, incidencias y ajustes del PAI.

Soporte: diagnóstico y resolución remotos siempre que sea posible, con actualizaciones planificadas.

Este Trabajo Fin de Grado se centra en optimizar el dispositivo cliente: seleccionar una plataforma de bajo coste (Raspberry Pi), endurecerla y configurarla como quiosco VERA, e integrar las capacidades de videollamada, telemetría, acceso remoto seguro y actualizaciones. El objetivo es reducir coste y mitigar incidencias que afectan a la experiencia, habilitando un despliegue más estable y sostenible.

Figura 3. Claves del éxito de Vera.

2.2 Importancia del sistema en el contexto de OHLA Ingesan

VERA es, para OHLA Ingesan, un vector estratégico que combina innovación social y tecnológica con impacto directo en la calidad de vida de las personas. La compañía opera en un contexto marcado por el crecimiento sostenido de la población mayor, la soledad no deseada y la presión sobre recursos presenciales como los centros de día. Muchas personas desean permanecer en su domicilio, pero requieren apoyos regulares y un acompañamiento que, con frecuencia, no puede cubrirse solo con dispositivos móviles u ordenadores por barreras de usabilidad. En este escenario, convertir el televisor en un canal de cuidado accesible y continuo sitúa a VERA como una respuesta pertinente y oportuna.

Necesidades detectadas

El punto de partida es una demanda creciente de apoyo a personas mayores que desean permanecer en su hogar y necesitan acompañamiento estructurado. Los recursos presenciales tradicionales afrontan limitaciones de capacidad y horarios, lo que obliga a buscar alternativas que mantengan la continuidad del cuidado sin desplazar a la persona usuaria. A ello se suma la brecha de usabilidad: una parte relevante del colectivo no utiliza con soltura móviles u ordenadores, mientras que la televisión sí es un dispositivo cotidiano y universal en el hogar.

Respuesta de VERA

Frente a este escenario, VERA convierte la TV del hogar en un canal de cuidado accesible. Instala un dispositivo plug & play que arranca en modo quiosco y ofrece, con el mando del televisor, sesiones planificadas, videollamadas con profesionales y familiares y contenidos de estimulación. La intervención se personaliza a partir de una valoración inicial y un **PAI**, y se sostiene con seguimiento continuo y soporte remoto para minimizar interrupciones y desplazamientos.

Encaje en la estrategia de OHLA Ingesan

1. Diferenciación del porfolio con una propuesta de servicio sociosanitario domiciliario basada en tecnología accesible
2. Escalabilidad operativa mediante estandarización de instalación, monitorización y acceso remoto.

3. Mejora continua basada en datos (trazabilidad de intervención y operación) para optimizar procesos y resultados
4. Competitividad en licitaciones y convenios, con indicadores objetivables de desempeño y coste total de propiedad controlado.
5. Alineamiento ESG: envejecimiento saludable, inclusión digital y reducción de desplazamientos.

Indicadores de éxito

Para evidenciar el encaje y orientar la toma de decisiones, se monitorizarán indicadores que reflejen adopción, uso, experiencia y eficiencia operativa. La mayoría de estos KPIs se visualizan en cuadros de mando de Power BI, alimentados directamente desde la plataforma VERA (extracciones periódicas y/o conectores en tiempo casi real), lo que facilita el seguimiento por parte de los equipos técnicos y de servicio.

Figura 4. Cuadro de mando de KPIs de VERA (Power BI).

En síntesis, VERA encaja en la estrategia de OHLA Ingesan como un servicio innovador y eficiente que responde a retos demográficos y operativos con una solución accesible, medible y escalable, cuyo desempeño puede seguirse de manera objetiva para asegurar su mejora continua.

3. Análisis y estudio previo

En este capítulo se presenta el estudio inicial que ha orientado las decisiones tecnológicas del proyecto. En primer lugar, se delimita el contexto funcional (integración de videollamadas dentro del ecosistema VERA en un dispositivo basado en Raspberry Pi) y los requisitos operativos (uso 24/7, mínima interacción por parte del usuario final, ejecución en modo quiosco, mantenimiento y monitorización remotos). A continuación, se realiza un benchmarking del estado del arte en plataformas de videoconferencia y se evalúan las opciones de sistema operativo más adecuadas para su ejecución fiable en Raspberry Pi, atendiendo a criterios de compatibilidad, seguridad, coste, licenciamiento, desempeño y facilidad de integración con VERA. Este análisis alimenta las decisiones de diseño que se desarrollan en los capítulos posteriores.

3.1 Estado del arte y tecnologías relacionadas

Las soluciones de videollamada actuales se basan mayoritariamente en tecnologías WebRTC, que habilitan comunicaciones en tiempo real directamente desde el navegador sin complementos, y constituyen el estándar de facto en el ecosistema web [4]. Para el caso de uso VERA, el criterio clave es la disponibilidad de un cliente web estable (ejecutable en Chromium) o de una app nativa compatible con Raspberry Pi, con un modelo de licenciamiento y consumo de recursos que permitan un despliegue sostenible a escala.

Conforme al análisis comparativo realizado, se valoraron, entre otras, las siguientes plataformas con cliente web oficial: Zoom, Google Meet, Microsoft Teams y Skype for Web. Todas ellas permiten uso sin instalar clientes de escritorio, lo que es clave para un despliegue en modo quiosco con Chromium [6]–[9]. En paralelo se consideraron soluciones open source con posibilidad de autoalojamiento para un mayor control de costes, integración y privacidad. En este grupo destaca Jitsi Meet, proyecto libre bajo licencia Apache 2.0 que funciona directamente en el navegador con WebRTC, con uso en servicio gestionado o despliegue propio [5]. Tras el estudio y pruebas exploratorias, se seleccionó Jitsi Meet como tecnología de videollamada de referencia para la integración con VERA, por su apertura, flexibilidad de despliegue e idoneidad para ejecución en Chromium en Raspberry Pi [4], [5].

Plataforma	Licencia	Cliente web en Chromium (RPi)	App nativa Linux/ARM	Valoración para VERA (1-5)
Jitsi Meet	Open source (Apache 2.0)	✓	✓	5,0
Zoom Web App	Freemium / propietaria	✓	✓	3,5
Google Meet	Propietaria	✓	✓	3,0
Microsoft Teams (web)	Propletaria (M365)	✓	✓	2,5
Skype for Web	Propietaria	✓	✓	2,5

Figura 5. Tabla comparación Tecnologías Videollamadas.

3.1.1 Análisis del Sistema Operativo

Como punto de partida se valoraron tres enfoques para habilitar las videollamadas en Raspberry Pi e integrarlas con VERA. El primero consistía en mantener un sistema Linux e introducir Anbox para ejecutar la aplicación Android dentro de un contenedor. El segundo planteaba instalar un Android nativo en la propia Raspberry Pi mediante un port de LineageOS. El tercero optaba por conservar un entorno Linux con Raspberry Pi OS y utilizar Chromium en modo quiosco como cliente web. En los apartados siguientes se detallan las pruebas realizadas con cada alternativa y los motivos que llevaron a su aceptación o descarte

Opción 1 — Linux + Anbox (APK Android en contenedor)

Implicaba instalar Anbox (módulos de kernel y runtime), cargar la APK de videollamada, fijarla en primer plano, y endurecer el sistema (permisos, red, bloqueo de interfaz), habilitando cámara y micrófono.

Resultados y descarte.

Complejidad y madurez insuficiente. Anbox exige soporte específico en el kernel y un runtime propio; además, el proyecto está archivado y sin mantenimiento activo desde 2024, con imagen base de Android 7.1.1, lo que lo hace poco adecuado para un servicio de producción con WebRTC y periféricos modernos [10]. En su propia documentación se indica que Android en el contenedor no accede directamente al hardware, sino a través del daemon del host, lo que introduce cuellos de botella en cargas de captura/decodificación en tiempo real [10].
Rendimiento multimedia insuficiente para videollamadas. En nuestras pruebas, las llamadas no fueron fluidas: la ruta de hardware no estaba plenamente disponible y la pila multimedia dependía en exceso de CPU. Dados los requisitos de continuidad y calidad, esta alternativa se descartó en fases tempranas. [10]

Opción 2 — Android nativo en Raspberry Pi (LineageOS)

Implico grabar la imagen de LineageOS en la micro-SD, instalar la APK Android, fijarla en primer plano, aplicar políticas de quiosco y habilitar cámara/micro. Se probó en profundidad sobre Raspberry Pi.

Resultados y descarte.

- Aceleración de vídeo limitada/inconsistente. Existen antecedentes y reportes comunitarios de ausencia o inestabilidad de decodificación/codificación por hardware en builds Android para Raspberry Pi, lo que repercute directamente en WebRTC (saltos de frames, latencia y CPU elevada) [11], y versiones previas llegaron a usar renderizado por software (SwiftShader) como solución, con impacto en rendimiento

[12].

- Soporte de cámara y periféricos no uniforme. Se observan incidencias históricas con la cámara en ports de LineageOS para Pi (dependencia de apps/driver y comportamiento dispar), lo que penaliza escenarios de videollamada plug-and-play [13].
- Encaje funcional limitado. Al comportarse como un Android genérico, varias funcionalidades extra necesarias para VERA a nivel de sistema (servicios de arranque, agentes de monitorización/telemetría y automatizaciones específicas) requerían desarrollo adicional o no eran compatibles sin *workarounds*. Además, los propios *maintainers* de los puertos señalan que son no oficiales y "para usuarios avanzados", lo que incrementa el riesgo operativo en campo [14].
- Por los motivos anteriores, pese a las pruebas realizadas, esta opción se descartó.

Opción 3 — Raspberry Pi OS (Debian) + Chromium en modo quiosco

Instalar Raspberry Pi OS, autenticar la sesión del dispositivo, lanzar Chromium a pantalla completa sobre la URL de la plataforma de videollamadas (Jitsi en nuestro caso), activar el modo quiosco al arranque (autologin + *autostart*), y conectar cámara y mando/controles. Esta configuración está documentada oficialmente por Raspberry Pi y permite un flujo 100% web, mantenible y auditable [15].
La decisión fue adoptar esta opción por su estabilidad, soporte y alineamiento con un cliente en navegador, minimizando dependencias propietarias y facilitando la integración con los servicios de VERA.

Enfoque S.O.	Base	Kiosko	Video / WébRTC	Cámara / periféricos	Gestión / actualizaciones	Resultado y motivo
Linux + Anbox (APK Android en contenedor)	Linux host + contenedor Anbox (Android 7,1)	Forzar APK en primer plano; bloqueo UI mediante scripts	Acceso HW; indirecto; rendimien te pobre en llamadas; CPU alta	Acceso via host; compatitiidad limitada	Complexa proyecto archivado/ dependenc- jas extra	**DESCARTADA** – Instalación compleja proyecto obsoleto videollamada
Android nativo (**LineageOS** para Raspberry Pi)	LineageOS (no oficial) ARM64	Apps de kiosko / autostart	Aceleración HW inconsis- tente; casos con renderi- zado software	Soporte **irregular**	Actúa como Android genérico; funcionalita- des extra nos	**DESCARTADA** – problemas de video y cámara; encaje limitadon VERA
Raspberry Pi OS + Chromium (modo kiosko)	Debian 12 ARM64	Compatible; rendimiento estable con navegador	Compatible rerdimiente establè con aceeracion	Actua con Android genérico; comunatitario	Total con serviciós/ daemons scripts; telemetrŕa	**SELECCIONADA** - estabilidad soporte,encaje con WebRTC y con VERA

Figura 6. Tabla comparación SO.

3.1.2 Tecnologías de modo quiosco en Linux y Android

El modo quiosco persigue que el dispositivo arranque, entre en sesión y exponga una única aplicación (en este proyecto, el cliente web de videollamada) a pantalla completa, sin distracciones ni posibilidades de salir del flujo previsto. Requiere, además, recuperación automática ante fallos, bloqueo de atajos/menús, gestión remota y endurecimiento del sistema.

Linux (Raspberry Pi OS): navegador en quiosco y endurecimiento del sistema

En Linux la aproximación más robusta es un quiosco de navegador (Chromium) sobre Raspberry Pi OS. El patrón técnico incluye:

- Arranque directo al quiosco: sesión *autologin* de un usuario dedicado y lanzamiento de Chromium a pantalla completa con URL de la sala. Se puede orquestar con systemd con *autostart* de entornoligero

- Bloqueo y UX mínima: ocultar el cursor (*unclutter*), desactivar *screensaver* y DPMS (xset s off -dpms), impedir combinaciones de teclas de cambio de TTY y de cierre de la ventana, limitar menús/contextos del navegador, y fijar la página de inicio/whitelist.

- Recuperación y supervisión: *watchdog* de proceso (systemd) para relanzar el navegador si se cierra o queda inoperativo; *health checks* de red y NTP; relanzar la sesión tras reinicio inesperado; *logging* centralizado (journald/rsyslog) y métricas.

- Gestión remota: actualización del sistema (APT), telemetría y control por agentes estándar de Linux.

Este enfoque maximiza la compatibilidad con navegador y encaja con las necesidades de integración de servicios de VERA, al tiempo que reduce dependencias propietarias [15].

Android: Lock Task / "quiosco" a nivel de sistema

En Android, el modo quiosco se implementa como un dispositivo COSU (Corporate-Owned, Single-Use) con Device Owner:

- Provisionamiento como Device Owner (DPC) y configuración de Lock Task Mode para que la app autorizada (lanzador o WebView/Chrome sobre la URL de la sala) quede fijada y el usuario no pueda abandonarla.

- Con Device Policy Manager se definen los paquetes permitidos se deshabilitan elementos del sistema (barra de estado, notificaciones, teclas) y se fuerzan políticas.

- Persistencia tras reinicio y recuperación: Boot receivers para iniciar la app tras boot, relanzar ante crash, actualizar silenciosamente, y aplicar políticas de energía/red.

- Permisos y periféricos: concesión gestionada de cámara y micrófono, control de sensores, y restricción de instalación de apps.

  Este enfoque simplifica el quiosco cuando se dispone de un MDM Android o de un DPC propio, pero su rendimiento en WebRTC y el soporte de periféricos dependen estrechamente de los drivers y del port de Android al hardware elegido [16].

  Comparativa y elección para este proyecto

- Linux (Chromium en quiosco) ofrece un camino claro y estable para el navegador, control total del arranque y del endurecimiento, y herramientas maduras de observabilidad y gestión.
- Android (Lock Task) aporta un *quiosco* muy cerrado y cómodo de administrar en flota cuando se cuenta con MDM/DPC, pero su idoneidad depende de la calidad del port y los controladores; además, la integración de utilidades del sistema de VERA puede requerir *workarounds* o servicios adicionales.

  Dada la naturaleza del proyecto y el hardware objetivo, se adopta Raspberry Pi OS + Chromium en modo quiosco como base operativa, manteniendo Android/Lock Task como alternativa válida cuando un entorno Android gestionado (MDM) sea un requisito explícito.

Figura 7. Sistema Operativo elegido.

3.1.3 Soluciones de control remoto y monitorización

Se buscó garantizar operación continua con mínima fricción para personas mayores. Para ello se evaluaron soluciones de acceso remoto (escritorio y terminal) y de monitorización del estado del dispositivo. Los criterios fueron: acceso desatendido real sin intervención del usuario, capacidad de atravesar NAT de forma segura, soporte en Raspberry Pi OS, prestaciones para soporte técnico (control de sesión, transferencia de archivos, portapapeles y terminal), coste y licencia, y huella en CPU y memoria.

Control remoto

Se analizaron varias alternativas con el objetivo de poder entrar en el dispositivo sin que el usuario tenga que aceptar cada conexión y resolver incidencias con rapidez, ya sea con sesión gráfica o con terminal.

Raspberry Pi Connect (seleccionada)

Proporciona acceso seguro al escritorio y a una terminal desde la web, con enlace permanente del dispositivo a una cuenta y conexiones desatendidas. Esta capacidad fue decisiva, ya que con otras herramientas no se logró eliminar la necesidad de aceptar la conexión. Se emplea para resolver incidencias en remoto mediante escritorio y para actuaciones rápidas a través de terminal.

VNC clásico

Soluciones como RealVNC o TigerVNC están bien soportadas en Raspberry Pi OS. Permiten acceso desatendido con credenciales y, si se desea, a través de túneles o VPN. Requieren mayor atención a la exposición de puertos y al endurecimiento de la configuración. Resultan útiles como plan de contingencia.

RDP mediante xrdp

Ofrece una experiencia similar a la de los escritorios remotos en entornos Windows. Dispone de clientes en múltiples sistemas operativos. Puede ser sensible a la configuración del entorno gráfico de la Raspberry Pi y conviene limitarlo a redes privadas o VPN por seguridad.

Herramientas propietarias de soporte remoto

AnyDesk, TeamViewer o Chrome Remote Desktop proporcionan una experiencia completa con buena travesía de NAT y acceso desatendido configurable. A cambio implican dependencia del proveedor, consideraciones de coste y un conjunto de funciones mayor del necesario para un quiosco acotado.

RustDesk

Alternativa de código abierto con opción de desplegar servidores propios para relay e identificación. Permite acceso desatendido y control de la infraestructura. Supone más esfuerzo de operación y ajuste cuando se trabaja con flotas.

DWService

Agente ligero que se administra desde un navegador sin instalar cliente en el equipo del técnico. Simplifica la conectividad sin abrir puertos. Está menos orientado a multimedia en tiempo real.

Enfoque basado en VPN más SSH o VNC

Soluciones como Tailscale, ZeroTier o WireGuard permiten presentar la Raspberry Pi en una red privada y, sobre ella, usar SSH para automatización y diagnóstico o VNC para escritorio. Aporta control topológico y seguridad a cambio de añadir una capa adicional que hay que operar y supervisar.

Conclusión de control remoto

La elección fue Raspberry Pi Connect por su integración nativa, su acceso desatendido sin intervención del usuario y su doble vía de trabajo mediante escritorio y terminal. Contribuye de forma directa a reducir la fricción en hogares de personas mayores y acelera la resolución de incidencias.

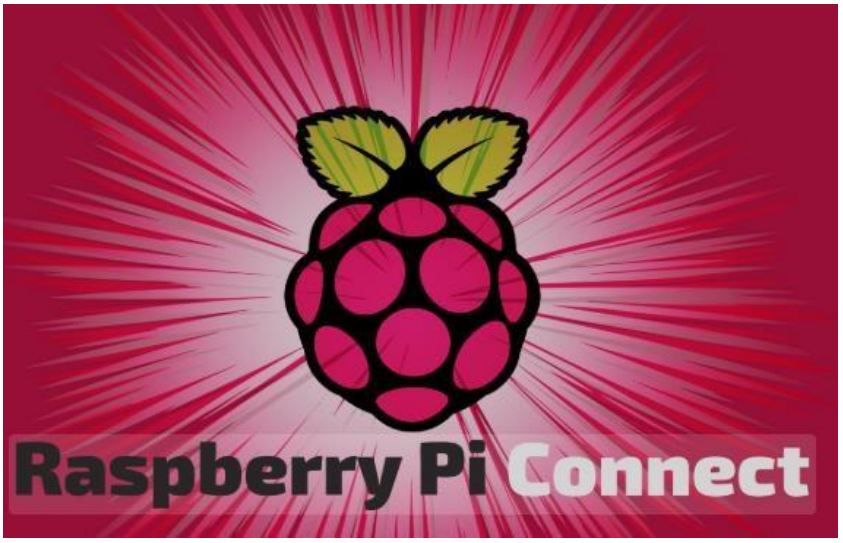

Figura 8. Herramienta de Control Remoto

Monitorización de la Raspberry Pi

Se revisaron opciones que proporcionan visibilidad de métricas clave, diagnóstico temprano y soporte a la actividad de soporte técnico.

PiCockpit (seleccionada)

Plataforma específica para Raspberry Pi que muestra métricas como carga de CPU, temperatura, memoria, almacenamiento, estado del SoC y utilidades de diagnóstico. Permite vigilar el estado en tiempo real y detectar señales tempranas, complementando el acceso remoto con un ciclo de trabajo claro: detectar, intervenir y verificar.

Netdata

Agente con panel web y recopilación de métricas en alta frecuencia. Es potente y detallado, pero añade complejidad y consumo que exceden las necesidades de un quiosco de videollamadas.

Prometheus con node_exporter y Grafana

Estándar para flotas grandes con almacenamiento de series temporales y cuadros de mando. Requiere infraestructura propia para scraping, reglas y visualización.

Zabbix, Nagios o CheckMK

Soluciones de monitorización clásicas con servidor central y agentes. Apropiadas para entornos corporativos con equipos dedicados a su operación continua.

PiCockpit

Interfaz web de administración para sistemas Linux. Útil para gestionar servicios y paquetes, aunque no sustituye a la observabilidad específica que aporta una plataforma centrada en Raspberry Pi.

Conclusión de monitorización

La elección fue PiCockpit por ofrecer visibilidad suficiente con baja complejidad operativa. La combinación de PiCockpit para detectar y contextualizar y Raspberry Pi Connect para intervenir cubre el ciclo de soporte sin añadir agentes pesados ni dependenciasadicionales.

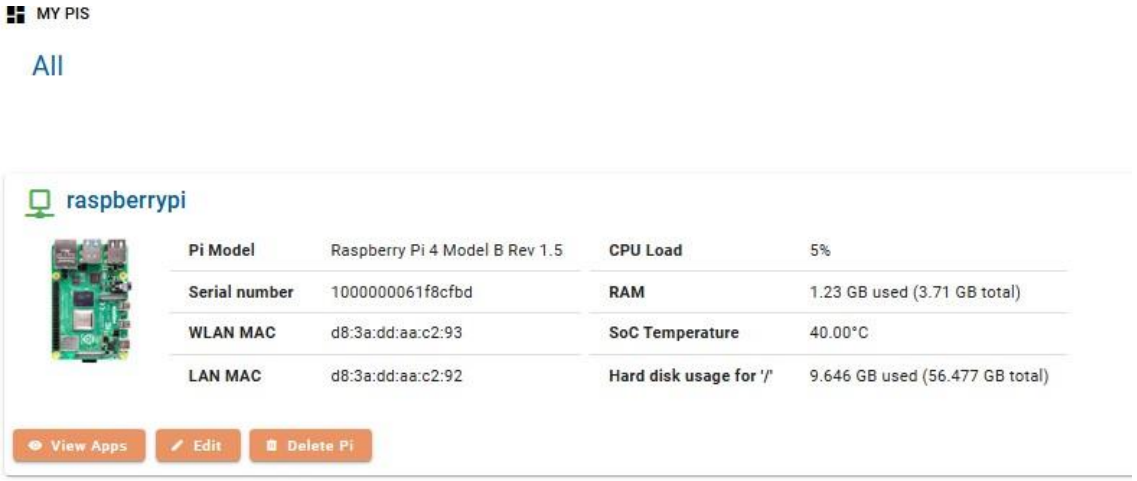

Figura 9. Ejemplo de Monitorización con PiCockpi

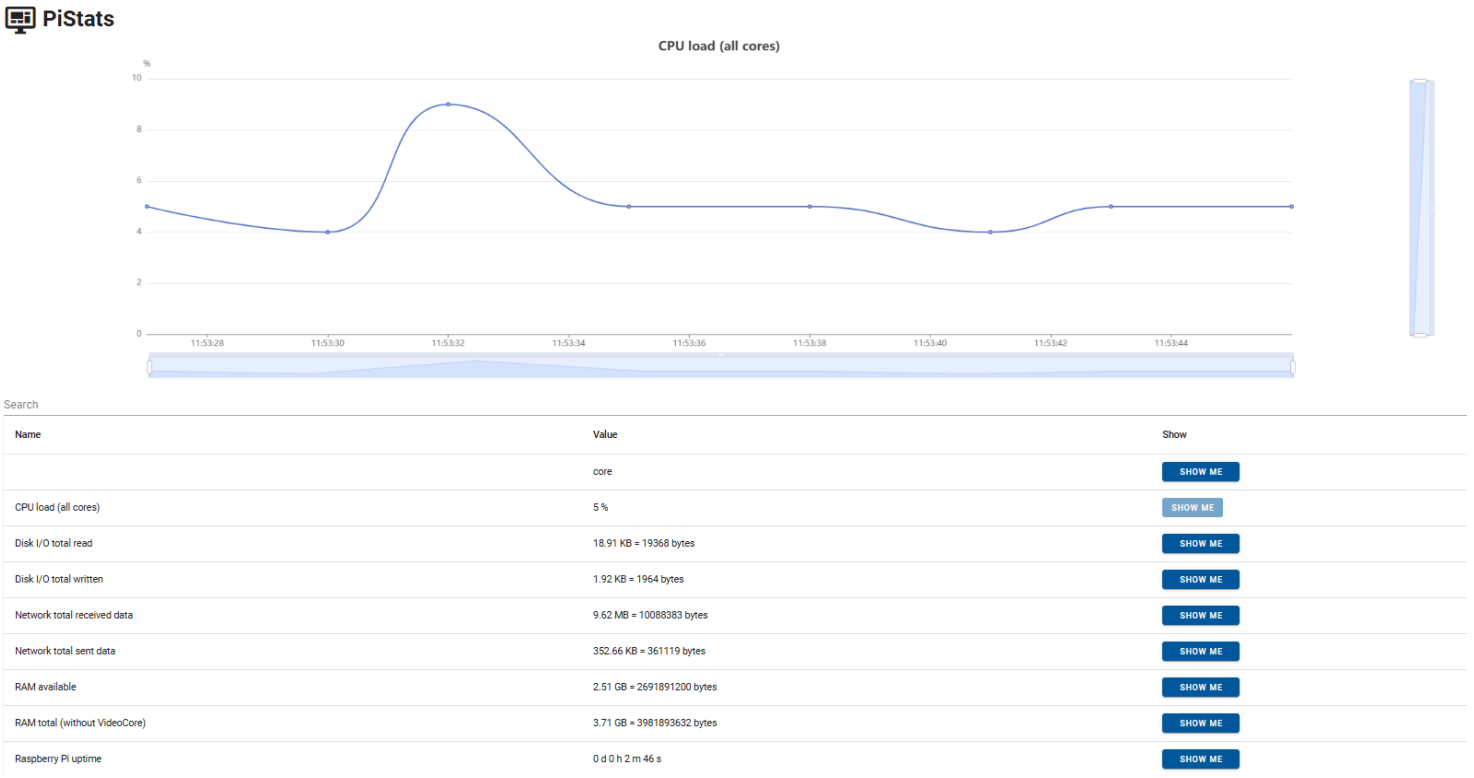

Figura 10. Ejemplo de Monitorización con PiCockpi

3.2 Análisis de mercado y comparación de opciones

Este apartado contrasta las alternativas de dispositivo final para integrar videollamadas en VERA y operar en hogares con mínima fricción. El análisis combina especificaciones técnicas (CPU, RAM, almacenamiento, conectividad, consumo), funcionales (control total del sistema, actualizaciones y reinicio remotos desatendidos, modo quiosco, soporte a cámara Logitech, MDM), y operativas (coste, disponibilidad, escalabilidad y mantenibilidad). Con estos criterios se han evaluado cuatro familias de productos: TV Box Android, Raspberry Pi 4, smartphone Android de gama alta y pantalla inteligente cerrada. La decisión final debe equilibrar coste total de propiedad, apertura tecnológica y la experiencia real de soporte a distancia para personas mayores, manteniendo un servicio 24/7.

3.2.1 TV Box vs Raspberry Pi vs alternativas comerciales

TV Box Android (Ugoos X4 Pro)

El TV Box Ugoos X4 Pro ofrece SoC Amlogic S905X4 (Cortex-A55), 4 GB RAM, almacenamiento por tarjeta TF y Android 11; se indica coste estimado ~240 € y compatibilidad con Logitech C270 por USB. En el análisis funcional aparece sin control total, con soporte remoto atendido (requiere aceptación), y sin actualizaciones ni reinicio desatendidos (aunque admite modo quiosco y conectividad HDMI/Ethernet/Wi-Fi). Estas limitaciones, junto al mayor precio unitario, penalizan su uso como plataforma gestionada a escala para VERA.

Conclusión parcial. Aunque es válido como cliente multimedia, la falta de control profundo del sistema y la dependencia de la interacción del usuario para el soporte remoto lo descartan para un parque de dispositivos orientado a personas mayores, donde la conexión desatendida es crítica para resolver incidencias.

Raspberry Pi 4 Model B

La Raspberry Pi 4 (4 GB) con Broadcom BCM2711 y Debian GNU/Linux 12 aporta control total del sistema, actualizaciones y reinicio remotos desatendidos, soporte remoto desatendido, modo quiosco y geolocalización, con consumo típico de 5,8–6,75 W. El coste estimado ~130 € (configuración VERA con carcasa/accesorios) la sitúa muy por debajo del TV Box. Se especifica además la compatibilidad con Logitech C270, conectividad Ethernet/Wi-Fi (y opción 4G/5G) y una propuesta de diseño orientada a mantenibilidad (carcasa ventilada, mandos sencillos, alimentación con botón). Todo ello permite integrar Raspberry Pi Connect para acceso desatendido al escritorio/terminal y PiCockpit para monitorización, cumpliendo los requisitos de operación 24/7.

Conclusión parcial. Reúne apertura tecnológica, bajo TCO y capacidades de gestión/soportes imprescindibles. Es la opción con mejor encaje con VERA.

Smartphone Android (Samsung Galaxy S24 FE)

Como alternativa “dispositivo único”, la comparativa refleja que un smartphone de gama alta cumple control total, actualizaciones y reinicio desatendidos, soporte remoto desatendido, modo quiosco y geolocalización, apoyado en Samsung Knox como MDM. Ofrece salida HDMI (MHL) para TV y conectividad Wi-Fi/Móvil. Sin embargo, su coste estimado ~570 € y el desalineamiento con el caso de uso (dispositivo pensado para uso personal, no como set-top dedicado) lo hacen menos competitivo para despliegue masivo.

Conclusión parcial. Es potente y gestionable con Knox, pero encarece el proyecto y complica la estandarización del puesto teleasistido (accesorios, fijación, ergonomía TV).

Pantalla inteligente cerrada (Amazon Echo Show 5)

La opción de pantalla inteligente presenta actualizaciones desatendidas, pero sin control total, sin modo quiosco y sin soporte remoto desatendido, además de no ofrecer salida a TV. Aunque su precio ~60 € es muy bajo, la plataforma cerrada impide integrar la pila de gestión y soporte exigida por VERA.

Conclusión parcial. Coste atractivo, pero no cumple requisitos clave de apertura, integración y soporte remoto.

Síntesis de decisión

- Raspberry Pi 4 proporciona el mejor equilibrio entre coste (~130 €), gestión desatendida, modo quiosco, monitorización y apertura del sistema, permitiendo una integración directa con el ecosistema VERA.

- TV Box Android queda relegado por falta de control y soporte atendido, además de un precio superior (~240 €) sin aportar ventajas funcionales para este caso.

- Smartphone y pantalla inteligente se consideran opciones válidas en contextos distintos (movilidad personal y hogar conectado), pero no optimizan el TCO ni la mantenibilidad del puesto de videollamada en TV requerido por VERA.

3.2.2. Costos, mantenimiento y escalabilidad

Aborda aspectos concurrentes en tiempo de ejecución y sus interacciones. En este contexto un proceso es una agrupación de tareas que forma una unidad ejecutable.

Muestra cómo las principales abstracciones de arquitectura encajan dentro de la arquitectura del proceso. Representan el nivel en el que la arquitectura del proceso se puede controlar tácticamente.

En esta vista se deberían incluir en la misma imagen todos los componentes, pero para que se pueda apreciar mejor he decidido separar la imagen componente a componente para que se pueda leer perfectamente.

Enfoque y supuestos.

Se evalúa el coste total de propiedad a 3 años combinando inversión inicial por dispositivo, coste operativo (tiempo de soporte y desplazamientos evitados gracias al acceso desatendido) y riesgo de obsolescencia. Se parte de los precios comparativos ya consolidados: kits completos de Raspberry Pi 4 (alrededor de 130 € según proveedor y contenido), TV Box Ugoos X4 Pro (≈ 240 €), smartphone de gama alta tipo S24 FE (> 500 €) y pantalla inteligente cerrada (≈ 60 €). Para mantenimiento se considera disponibilidad de actualizaciones, facilidad de reinstalación/provisión masiva y herramientas de control remoto/monitorización. Para escalabilidad, se valora la logística de compra, la estandarización del “puesto” y la automatización del despliegue.

Coste unitario – Raspberry Pi (Desglose por kits)

Los kits listados incluyen placa Raspberry Pi 4 (4 GB), carcasa ventilada, disipadores y ventilador, fuente 5V/3A, tarjeta microSD, cables micro-HDMI/HDMI, cámara Logitech C270 y mando IR/USB. Según el proveedor (nacional o importación) varían contenidos y precio: hay kits “todo en uno” y configuraciones modulares. La ventaja clave es poder ajustar el pack a las necesidades reales del proyecto (por ejemplo, elegir una microSD industrial o un mando simplificado) y mantener el coste en torno a 130 € por unidad con componentes equivalentes. Al estar desagregado, cualquier componente puede sustituirse a bajo coste sin reemplazar el conjunto.

Coste Unitario- Alternativas

El TV Box ronda los 240 € con software Android preinstalado, pero sin incluir cámaras certificadas ni accesorios equivalentes a la configuración objetivo. El smartphone supera los 500 € aun antes de accesorios, soportes, dongles HDMI y alimentación

permanente; su ergonomía no está pensada como puesto fijo de salón. La pantalla inteligente es muy barata (~60 €), pero carece de salida a TV y de las funcionalidades exigidas, por lo que su bajo precio no compensa las limitaciones funcionales.

Costes operativo y mantenimiento

En Raspberry Pi el mantenimiento es predecible: imagen del sistema reproducible, actualizaciones gestionables, y acceso desatendido por escritorio o terminal para resolver incidencias sin intervención del usuario. Esto reduce tiempos de soporte, evita desplazamientos y permite "parches" rápidos (reiniciar servicios, limpiar espacio, reconfigurar) incluso con el televisor apagado. La reposición ante fallo es simple: se sustituye la microSD o la unidad completa y se realiza *bootstrap* con la imagen estándar en minutos. En TV Box Android, la falta de control profundo del sistema y la necesidad de que el usuario acepte conexiones elevan el tiempo medio de resolución y multiplican llamadas de soporte. En smartphone, aunque el MDM permite políticas avanzadas, se heredan fricciones de un dispositivo personal: rotación de pantallas, bloqueos, eventos de batería y gestión térmica en uso 24/7. En pantalla inteligente no existe una ruta de soporte alineada con los requisitos (ni modo quiosco ni administración remota real).

Escalabilidad del despliegue

Con Raspberry Pi se estandariza un único "puesto" y se industrializa el *imaging*: una microSD maestra se clona en serie, se inyectan parámetros por lote (identidad del dispositivo, Wi-Fi/Ethernet, URL de servicio) y listo. La logística es flexible porque los kits pueden comprarse a múltiples distribuidores; si escasea un componente, se sustituye por homólogo sin rediseñar el puesto. El soporte en campo escala porque el técnico accede sin intervención del usuario, tanto a escritorio como a terminal, y la monitorización centralizada avisa de condiciones anómalas antes de que se conviertan en incidencia. En TV Box, la escalabilidad se complica por la heterogeneidad de firmwares y la menor capacidad de automatizar provisiones. En smartphone, la escala depende de un MDM corporativo y de una cadena de accesorios menos estandarizada. En pantalla inteligente, el ecosistema cerrado impide un despliegue gestionado.

Riesgo y vida útil

Raspberry Pi ofrece trayectoria de soporte amplia, comunidad y disponibilidad de repuestos; si aparece un nuevo modelo, la migración suele ser transparente a nivel de software base. En TV Box hay mayor fragmentación por fabricante y versión de Android; un cambio de modelo puede romper automatismos. En smartphone, la vida útil está condicionada al ciclo comercial del terminal y a cambios de política de fabricante; además, su operación como "quiosco" 24/7 no es el uso natural para el que

se optimiza. En pantalla inteligente, la dependencia del proveedor y las restricciones de la plataforma añaden riesgo funcional.

Conclusión

La Raspberry Pi minimiza la inversión inicial, reduce el coste operativo gracias al acceso desatendido y a la monitorización, y maximiza la escalabilidad mediante provisión industrializable y logística flexible de componentes. TV Box, smartphone y pantalla inteligente resultan menos competitivos en conjunto: o bien incrementan gasto y fricción de soporte, o bien no cumplen los requisitos de control, integración y mantenibilidad exigidos por el proyecto.

Componente	Función	Cant.	PVP unit. (€)	Subtotal (€)	Observaciones
Raspberry Pi 4 Model B (4 GB)	Plataforma príncipal	1	70 0	70	Garantía/ distribuidor local
Carcasa ventilada + disipâdores	Refrigeración/ Protección	1	12 0	12	Incluye tornillería
Fuente de alimentación 5V	Alimentación estable	1	10 0	10	Interruptor en cable (opcional)
Tarjeta microSD 32 GB A1	Almacenamien-to SO	1	8 0	8	Imagen estándar VERA
Cable micro-HDMI	Conexión a TV	1	5 0	5	1,5 m
Cámara Logitech	Vídeo	1	20 0	20	USB
Mando a distancia	Interacción	1	5 0	5	Receptor USB
	Total kit (IVA excl.)			**130 €**	

Figura 11. Resumen Kit Raspberry

3.3 Estudio de requisitos del sistema

Este capítulo define y justifica los requisitos que debe cumplir el sistema para integrar las videollamadas de Vera en un dispositivo doméstico que esté orientado a personas mayores. Se distinguen dos grupos: requisitos funcionales (lo que el sistema debe hacer para prestar el servicio) y requisitos no funcionales (cómo debe hacerlo en términos de calidad, seguridad, rendimiento, mantenibilidad, accesibilidad y operación). La definición conjunta permite trazar cada decisión técnica con las necesidades del servicio: instalación estandarizada, uso extremadamente sencillo, soporte remoto sin intervención del usuario y continuidad operativa. A continuación, se detallan los requisitos funcionales y los no funcionales.

3.3.1 Requisitos funcionales

Plataformado y despliegue

El sistema debe prepararse a partir de una imagen estándar y parametrizarse por lote para clonar configuraciones, reducir tiempos de alta y asegurar homogeneidad. Es imprescindible para escalar a decenas o cientos de unidades y para reposiciones rápidas ante averías. Criterios de aceptación: imagen única reproducible, inyección de parámetros (identidad, red, URL del servicio) y primer arranque desatendido.

Control total del sistema

Se requiere control íntegro del sistema operativo y de la pila de software (servicios, arranque, permisos, red y energía). Sin ese control no es viable automatizar tareas, endurecer el entorno ni diagnosticar problemas complejos. Criterios de aceptación: configuración del arranque al quiosco, políticas de energía y red, supervisión de servicios y acceso completo a registros.

Actualización sin interacción del usuario

Las actualizaciones deben aplicarse en segundo plano y programarse fuera del horario de uso, sin diálogos locales. Es clave para mantener seguridad y estabilidad con usuarios no técnicos. Criterios de aceptación: actualización desatendida, ventanas de mantenimiento definibles y reversión segura ante fallos.

Reinicio remoto programable y bajo demanda

El dispositivo debe reiniciarse a distancia y admitir reinicios programados en horarios de baja actividad. Esto mitiga degradaciones acumuladas y acelera la resolución de incidencias sin desplazamientos. Criterios de aceptación: orden de reinicio remoto,

calendario periódico y retorno automático al estado operativo.

Acceso remoto para soporte (escritorio y terminal)

Debe existir acceso desatendido al escritorio y a una consola segura, con travesía de NAT. Permite resolver incidencias sin intervención del usuario, algo crítico en hogares de personas mayores. Criterios de aceptación: sesión remota sin confirmación local, shell disponible y trazabilidad de accesos.

Modo quiosco web de Vera

La interfaz se limitará a la aplicación web de Vera a pantalla completa, auto lanzada en el arranque y con recuperación ante cierres o reconexiones de red. Evita distracciones y facilita la asistencia remota. Criterios de aceptación: arranque directo del navegador en pantalla completa, apertura de la URL de servicio, relanzamiento automático y bloqueo de salidas no deseadas.

Usabilidad para usuario final

El puesto debe ser operable por personas con baja alfabetización digital: tipografías amplias, alto contraste, volumen al iniciar, cursor oculto para evitar confusión y mínima secuencia de acciones para atender o iniciar una videollamada. Criterios de aceptación: tamaño de fuente y contraste preconfigurados, volumen inicial definido, cursor oculto en reposo y número de pasos reducido para atender una llamada.

Gestión mediante MDM

La solución debe integrarse con una plataforma de gestión de dispositivos para aplicar políticas, inventario y acciones remotas (bloqueo, borrado, cambios de configuración). Criterios de aceptación: alta automática en MDM, políticas por grupo y estado reportado del dispositivo.

Mando sencillo

Debe admitirse un mando simple y robusto que permita aceptar o colgar llamadas, navegar y confirmar sin teclado ni ratón. Criterios de aceptación: reconocimiento inmediato por USB o Bluetooth, mapeo estable de botones y funcionamiento sin configuración del usuario.

Cámara compatible

Se requiere una cámara UVC estándar con micrófono integrado o compatible que garantice videollamada fluida. Criterios de aceptación: detección plug-and-play, captura estable en 720p/1080p y acceso simultáneo a audio y vídeo.

Conectividad con TV

El dispositivo debe conectarse al televisor por HDMI, garantizando audio y vídeo sincronizados y compatibilidad con resoluciones comunes. Criterios de aceptación: salida HDMI estable y reanudación tras cambios de fuente o encendido del televisor.

Conectividad USB y HDMI para periféricos

Debe disponer de puertos suficientes para cámara, receptor del mando y otros periféricos, facilitando reposiciones y extensiones futuras. Criterios de aceptación: puertos operativos sin hubs complejos y alimentación adecuada para periféricos.

Autologin

El sistema debe iniciar sesión automáticamente tras cada encendido o reinicio, sin credenciales manuales, y llegar a la interfaz de Vera listo para usar. Criterios de aceptación: arranque hasta la aplicación sin intervención y recuperación tras cortes eléctricos.

Monitorización operativa

El puesto debe exponer métricas de salud (temperatura, CPU, memoria, espacio y estado de servicios) y eventos básicos para detección temprana de problemas. Criterios de aceptación: panel de métricas accesible, umbrales de alerta y registro histórico.

Protección frente a uso involuntario

Se deben deshabilitar rutas de apagado accidental y minimizar acciones que saquen al dispositivo del servicio (botones del sistema o combinaciones de teclas). Criterios de aceptación: apagado solo por vía remota o procedimiento técnico y bloqueo de accesos a menús del sistema.

3.3.2 Requisitos no funcionales

Usabilidad y accesibilidad

El sistema debe ser cómodo para personas con baja alfabetización digital. Se prioriza tipografía grande, alto contraste, navegación lineal con pocas decisiones, feedback claro en llamadas y control por mando sencillo. Objetivos recomendados: pasos para atender una llamada ≤ 2; elementos interactivos visibles a ≥ 2,5 m; volumen preconfigurado a nivel audible en el arranque; tolerancia a pulsaciones repetidas sin efectos adversos.

Disponibilidad y resiliencia

La experiencia debe estar siempre "lista para usar" sin tareas previas del usuario. Se exige recuperación automática ante cortes eléctricos y tras reinicios planificados o incidentales. Objetivos recomendados: disponibilidad mensual ≥ 99 %; tiempo de arranque hasta estado operativo ≤ 60 s; reintentos automáticos de red/servicio; vigilancia de procesos con relanzamiento.

Rendimiento y calidad de la videollamada

La plataforma debe sostener llamadas fluidas sin sobrecarga térmica ni de CPU. Objetivos recomendados: vídeo estable a 720p con 25–30 fps en WebRTC, latencia usuario-usuario < 250 ms en condiciones de red doméstica típicas, CPU media en llamada < 70 % y temperatura del SoC por debajo del umbral de *throttling*. Se debe preservar sincronía A/V al cambiar de fuente o reanudar el televisor.

Seguridad y privacidad

Se requiere principio de mínimo privilegio, separación de cuentas de servicio y endurecimiento del entorno quiosco. El acceso remoto ha de estar autenticado, auditado y cifrado extremo a extremo; las actualizaciones deben verificarse. Objetivos recomendados: credenciales rotadas por lote, *firewall* restrictivo, servicios expuestos mínimos, telemetría limitada a métricas técnicas sin datos personales, y registro de accesos para trazabilidad.

Mantenibilidad y soporte

La solución debe ser fácil de diagnosticar y reparar sin desplazamientos. Se exige observabilidad (métricas de salud, logs accesibles), procedimientos estándar de restauración y *roll-back* de actualizaciones. Objetivos recomendados: tiempo medio para iniciar una sesión de soporte remoto < 2 min; MTTR típico < 15 min para incidencias de configuración; restauración del sistema mediante imagen estándar en < 10 min.

Escalabilidad operativa

Debe ser viable gestionar decenas o cientos de equipos con el mismo esfuerzo unitario. Se pide *imaging* reproducible, parametrización por lote, inventario y políticas centralizadas (MDM), así como etiquetado por grupos de despliegue. Objetivos recomendados: alta masiva con *zero-touch*, políticas aplicadas por grupo en < 30 min y telemetría agregada por dispositivo.

Portabilidad y apertura tecnológica

Se favorecen plataformas abiertas y estándares para evitar bloqueos de proveedor y facilitar la evolución. El uso de componentes y protocolos abiertos (Linux, WebRTC, UVC, HDMI) reduce riesgo y costes futuros. Objetivos recomendados: ausencia de dependencias críticas propietarias en el flujo de videollamada y en el control remoto; posibilidad de migrar de hardware manteniendo la misma imagen/base.

Eficiencia energética y térmica

El equipo debe consumir poco y disipar adecuadamente para uso 24/7. Objetivos recomendados: consumo en reposo bajo una cifra compatible con operación continua en hogares; control de ventilación no intrusiva; protección ante picos térmicos con *throttling* evitado por diseño (carcasa ventilada, disipación).

Estalabilidad y reversibilidad

La instalación debe ser repetible y documentada, con capacidad de deshacer cambios. Objetivos recomendados: *playbooks* o scripts únicos para provisionar; actualizaciones con reversión segura; backups de configuración por dispositivo.

Observabilidad y trazabilidad

Se requiere panel de métricas, alertas por umbral y registro de eventos relevantes (reinicios, caídas de servicio, cambios de red). Objetivos recomendados: alertas tempranas de temperatura, espacio y servicio de videollamada, y retención de históricos para diagnóstico.

Interoperabilidad y compatibilidad de periféricos

La solución debe aceptar cámaras UVC y mandos USB/Bluetooth de mercado, así como operar con televisores HDMI habituales. Objetivos recomendados: compatibilidad plug-and-play con cámara y receptor del mando sin controladores adicionales; soporte de resoluciones de TV comunes; reconocimiento de periféricos tras reinicio sin intervención.

Experiencia audiovisual

Se garantiza inteligibilidad de voz y niveles de volumen adecuados desde el primer uso. Objetivos recomendados: volumen inicial alto pero no distorsionado, cancelación de eco a nivel de aplicación y estabilidad de audio al reconectar el televisor.

Fiabilidad eléctrica

Se considera la realidad doméstica: cortes de luz, desconexiones de cables o cambios de entrada en TV. Objetivos recomendados: retorno automático al estado de servicio tras energizar; tolerancia a desconexión/reconexión de cámara o mando; reanudación de vídeo a pantalla completa sin mensajes técnicos.

3.3.3 Seguridad

El objetivo es que el puesto de VERA en el hogar sea seguro “por defecto”, resistente a errores cotidianos (cortes de luz, desconexiones, cambios de red) y a manipulaciones involuntarias, sin exigir acciones técnicas a la persona usuaria.
Partimos de un modelo de amenazas doméstico: exposición innecesaria de servicios, integridad del sistema tras actualizaciones o apagados bruscos, privacidad (evitar almacenar datos personales) y disponibilidad continua de la interfaz en modo quiosco. La gestión remota ya se trató en apartados previos; aquí solo se añaden medidas nuevas y complementarias (trazabilidad y principio de mínimo privilegio).

Prioridades de seguridad

1. Endurecimiento del sistema: usuario de quiosco sin privilegios, arranque directo al navegador a pantalla completa con relanzamiento automático, servicios no usados deshabilitados y registros con retención limitada.
2. Seguridad de red: sin puertos entrantes abiertos, salidas restringidas a dominios necesarios (servicio de videollamada, actualizaciones, telemetría técnica), DNS/NTP fiables y validación estricta de TLS en la aplicación web.
3. Actualizaciones seguras: ventana programada, verificación de firmas de paquetes, comprobación de salud post-parche y reversión automática si la UI no queda operativa.
4. Protección de datos y privacidad: no almacenar PII ni grabaciones; telemetría limitada a métricas técnicas (temperatura, CPU, espacio, estado de servicios) y borrado seguro de la microSD en RMA o reasignación.
5. Resiliencia operativa: reinicio programado (p. ej., madrugada) para liberar recursos, detección y relanzamiento del navegador si la URL no responde, retorno automático al quiosco tras cortes eléctricos o cambios de fuente HDMI.

6. Identidades y trazabilidad (novedad respecto a control remoto): credenciales únicas por dispositivo, rotación en el *imaging*, bloqueo de cuentas por defecto y registro de accesos administrativos para auditoría.

En la práctica, estas medidas se aplican como una línea base única del sistema: imagen reproducible con políticas de arranque y red predefinidas; *watchdog* del navegador y de la conectividad para evitar que el usuario vea estados intermedios; y configuración “caja cerrada” en la que no hay menús ni rutas de salida fuera de la aplicación de VERA. La seguridad se apoya en estándares del propio sistema (firewall, servicios mínimos, perfiles de confinamiento) y en procedimientos operativos sencillos: una ventana semanal de actualización, una comprobación automática de servicio tras parche y alertas tempranas cuando la temperatura, la CPU o el espacio superan umbrales razonables.

Como verificación continua, se recomienda un ciclo simple: arranque de prueba tras cada actualización para confirmar que la interfaz carga y la cámara/micrófono funcionan; revisión de alertas y de los accesos administrativos registrados; y ensayo periódico de recuperación ante fallo (corte de energía, desconexión/reconexión de cámara y mando). Con esta combinación de controles y pruebas ligeras se consigue una postura “segura por defecto”.

4. Diseño del sistema basado en Raspberry Pi

Este capítulo aterriza la solución en una arquitectura concreta, pensada para un salón real: el usuario enciende la tele y, sin pasos intermedios, está listo para atender una videollamada de Vera. Partimos de lo ya validado en el análisis previo, pero aquí lo hilamos como un sistema único: qué piezas físicas lo componen, cómo arranca y se recupera, qué servicios sostienen el modo quiosco y qué parámetros convierten un "prototipo" en un equipo reproducible y mantenible a escala. En los apartados siguientes (4.2 y 4.3) detallaremos la conectividad y el despliegue, así como las líneas de seguridad y mantenimiento; ahora nos centramos en la arquitectura general.

4.1 Arquitectura general del sistema

La solución se entiende mejor si se recorre de extremo a extremo. En el hogar hay un único punto de interacción: el televisor. A él se conecta la Raspberry Pi 4 (4 GB) por HDMI; a la Pi, por USB, una cámara UVC con micrófono. La alimentación es estable (5 V) y la carcasa va ventilada para mantener el SoC en su rango óptimo durante sesiones largas que exige mayor rendimiento y para aguantar siempre encendida. Esta disposición evita dependencias exóticas: todo son piezas estándar, fáciles de sustituir, y la Pi ofrece suficientes puertos para no depender de hubs ni de drivers propietarios.

Cuando el equipo recibe energía, no aparece un escritorio convencional. Raspberry Pi OS inicia sesión automáticamente con un usuario de servicio sin privilegios y, en cuanto carga el entorno, Chromium toma el control de la pantalla en modo quiosco con la URL de Vera. Las banderas del navegador eliminan diálogos molestos y evitan que una caída de pestaña bloquee al usuario. Un servicio del sistema supervisa tanto el proceso como la conectividad: si el navegador muere o la red fluctúa, lo relanza y muestra un estado de reconexión hasta recuperar la sesión.

El televisor puede cambiar de entrada, la casa puede sufrir un corte eléctrico; tras cada evento, el equipo se reinicia y vuelve al mismo sitio: la interfaz de Vera a pantalla completa. Esa invariancia del estado visible es, en la práctica, lo que hace que el puesto "se sienta fiable".

En paralelo, el sistema aplica una serie de ajustes que marcan la diferencia en uso doméstico. La tipografía y el escalado se calibran para lectura a varios metros; el cursor se oculta para evitar distracciones; el volumen arranca alto para garantizar inteligibilidad desde el primer segundo. Cada madrugada —en una ventana de baja actividad— un reinicio programado limpia recursos y aplica pendientes, reduciendo la probabilidad de degradación acumulada. Los registros se rotan para que el almacenamiento no se llene y se separan los eventos del navegador de los del sistema, facilitando diagnósticos breves cuando haga falta intervenir.

Esta arquitectura no es una suma de piezas aisladas, sino un flujo: energía → autologin → quiosco → supervisión → recuperación automática. Y está pensada para industrializarse. La Pi arranca desde una microSD clonada a partir de una imagen maestra: la misma versión del sistema, los mismos servicios, las mismas reglas. Antes del primer encendido se inyectan parámetros de sitio (identidad del equipo, credenciales técnicas, red, URL de servicio) y, al terminar el *imaging*, se validan automáticamente tres cosas: que el quiosco carga, que la cámara y el micrófono responden y que el mando envía eventos. Si algo falla, el equipo puede reponerse en minutos sustituyendo la microSD por otra imagen conocida. Esa reproducibilidad es la base para escalar a decenas o cientos de unidades sin sorpresas.

Figura 12. Descripción Raspberry

4.1.1 Hardware utilizado

En este apartado se define el conjunto físico mínimo y robusto que convierte la Raspberry Pi en un “appliance” doméstico: Pi 4 (4 GB) conectada por HDMI a la TV, cámara UVC USB con micrófono y un mando USB básico con botones limitados para evitar confusiones. Se priorizan componentes estándar, fáciles de sustituir y con buena disponibilidad, junto a una alimentación estable y una carcasa ventilada que garanticen uso 24/7.

- Raspberry Pi 4 Model B (4 GB). Plataforma central por su equilibrio entre potencia para WebRTC en navegador y consumo contenido. Incluye Ethernet, Wi-Fi, cuatro USB (para cámara y receptor del mando) y salida micro-HDMI doble para conexión a TV. Se prioriza la versión de 4 GB para mantener fluidez con Chromium y evitar problemas en sesiones largas.

- Almacenamiento microSD (64 GB, clase A1/A2). Contiene la imagen estándar del sistema y permite reposición exprés: ante avería, basta con sustituir la tarjeta por una copia conocida. Se recomienda mantener una microSD “maestra” para clonados y otra de repuesto por equipo.

- Cámara UVC USB (ej. Logitech C270). Plug-and-play en Linux sin controladores propietarios. Entrega 720p estable y micrófono integrado; la base de pinza facilita fijarla al marco del televisor. Es preferible un cable USB de 1,5–2 m y abrazaderas para guiado de cable y evitar tirones.

- Mando USB básico con receptor propio. Es la interfaz principal para el usuario y se ha escogido deliberadamente simple para minimizar la curva de aprendizaje. Funciona como dispositivo HID por USB (sin emparejamientos ni apps) y sus botones están limitados a un conjunto mínimo indispensable: aceptar/colgar, dirección (arriba/abajo/izquierda/derecha), OK/confirmar y atrás. También incluye volumen aunque esto ya está configurado para que ocurra nada más iniciarse el dispositivo); no incorpora teclas numéricas ni funciones avanzadas para evitar pulsaciones confusas. El receptor se sitúa en el frontal del televisor (o con alargador USB) para buena recepción.

- Conexión a TV. Cable micro-HDMI→HDMI de 1,5 m sujeción y alivio de tensión para evitar desconexiones al mover la TV.

- Red. Preferencia por Ethernet para estabilidad y menor latencia; Wi-Fi se deja preconfigurado como alternativa. El cable se guía por la trasera del mueble y se etiqueta el puerto para facilitar soporte telefónico.

- Alimentación. Fuente oficial de 5 V/3 A; recomendable interruptor en el propio cable para cortes controlados si el usuario lo requiere. La regleta queda accesible pero protegida de desconexiones accidentales.

- Refrigeración y carcasa. Carcasa ventilada con disipadores y ventilador silencioso para evitar calentamiento térmico en sesiones prolongadas; la orientación y la salida de aire se verifican para no recircular calor contra el mueble.

Figura 13. Hardware Raspberry

4.1.2 Software y sistema operativo

Aquí se describe la base software que asegura que el equipo encienda, entre y se mantenga en la experiencia de Vera.

- Base del sistema. Raspberry Pi OS de 64 bits (Debian 12). Instalación mínima, locales y zona horaria definidas, y solo los paquetes necesarios para navegador y utilidades de operación. Usuario de servicio sin privilegios de administración.

- Sesión y arranque. Autologin gráfico directo al usuario y desactivación de pantallas de bienvenida. El arranque está optimizado para llevar al dispositivo cuanto antes a la interfaz de Vera, sin exponer el escritorio del sistema.

- Modo quiosco con Chromium. El navegador se lanza en pantalla completa hacia la URL de Vera con banderas que suprimen diálogos, evitan burbujas de error y fuerzan el foco de ventana. Si el usuario apaga o cambia de entrada en la TV, al recuperar señal se mantiene el modo a pantalla completa.

- Supervisión y recuperación. Un servicio del sistema vigila proceso y conectividad; si la pestaña se cierra, el navegador se bloquea o la URL no responde, relanza la sesión tras unos segundos y muestra un estado de reconexión hasta restablecer servicio. Todo ocurre sin intervención del usuario.

- Accesibilidad de salón. Escalado de interfaz y tipografía grandes para lectura a 2–3 m; volumen inicial elevado mediante un script de arranque; cursor oculto tras unos segundos de inactividad para no distraer. Los colores de la interfaz se validan en televisores típicos para asegurar contraste.

- Programación operativa. Reinicio automático diario en una ventana de baja actividad para liberar recursos y aplicar parches pendientes. Las tareas se programan con temporizadores del sistema y dejan registro de ejecución.

- Redes y confiabilidad. Ethernet prioritaria con Wi-Fi como respaldo, ambas preconfiguradas. El equipo espera a tener red antes de lanzar la sesión o la reintenta de forma progresiva.

- Registros y telemetría ligera. Rotación de logs con retención limitada para no llenar la microSD. Métricas básicas (temperatura, CPU, memoria, espacio, estado del navegador) disponibles para soporte técnico y detección temprana de incidencias.

- Periféricos. La cámara aparece con nombre estable y se fija como dispositivo por defecto de audio/vídeo; si se desconecta y reconecta, la sesión de videollamada

puede retomarla sin pasos del usuario. El mando USB básico envía eventos HID mapeados a acciones simples (aceptar/colgar, navegación, OK/atrás) que la interfaz reconoce de forma directa.

- Imagen reproducible y parametrización. La configuración completa (paquetes, servicios, políticas, quiosco, accesibilidad) vive en una imagen maestra. Antes del primer arranque se inyectan parámetros de sitio (identidad del equipo, red, URL de servicio). Al terminar el *imaging*, una prueba de humo verifica que quiosco, cámara/micrófono y mando funcionan, y que el equipo queda listo para uso inmediato.

- Seguridad de base. Usuario sin privilegios, servicios no utilizados deshabilitados, rutas de apagado del sistema ocultas en la interfaz y reglas de red restrictivas por defecto para minimizar la exposición. Si el equipo pierde energía o la red, al volver siempre reaparece la sesión de Vera en pantalla completa.

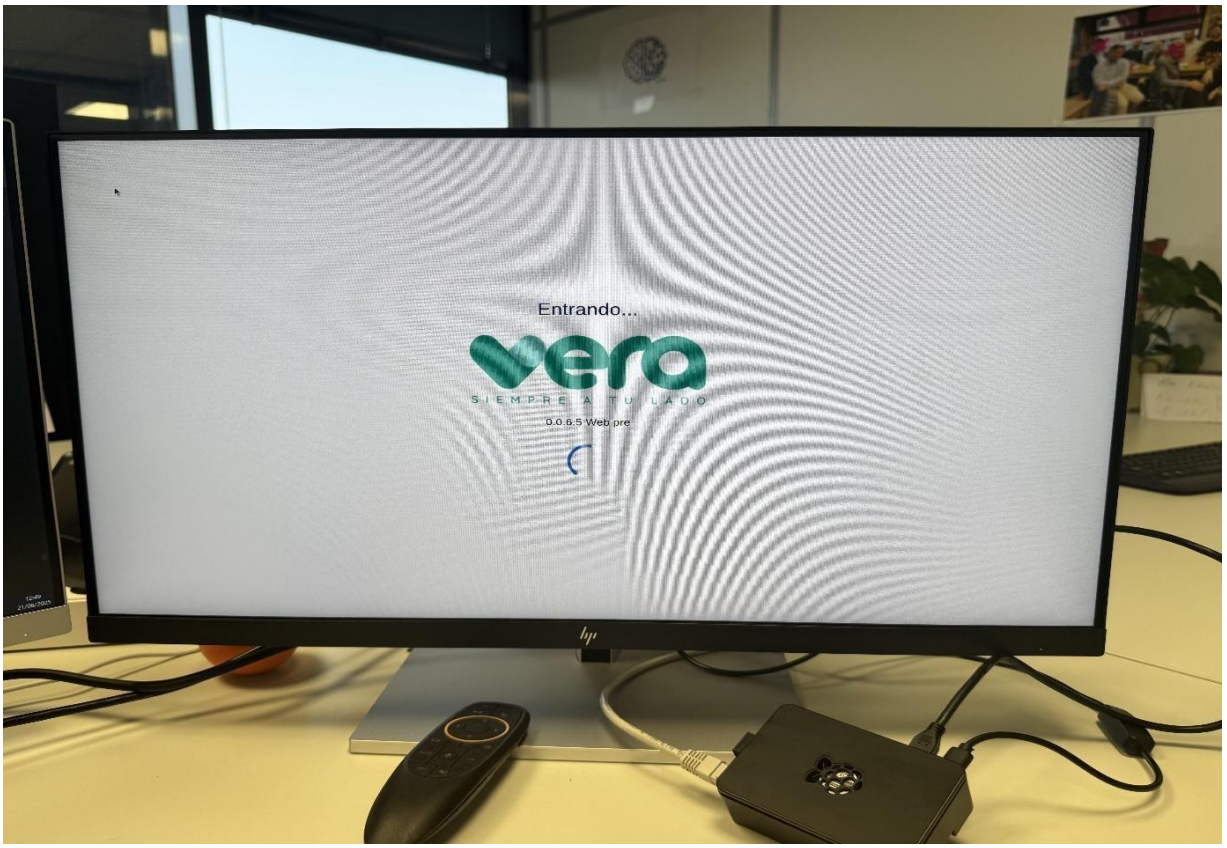

Figura 14. Ejemplo Raspberry

4.2 Conectividad y despliegue

Este apartado explica cómo se prepara, entrega y deja operativo el puesto basado en Raspberry Pi en el hogar, con especial atención a la conectividad (qué opciones se ofrecen al usuario) y al proceso de despliegue (de la imagen de fábrica al salón). El objetivo es que, en cualquier escenario de red doméstica, el equipo arranque directamente en la experiencia VERA y se mantenga en servicio sin pedir nada a la persona usuaria.

Principios de diseño del despliegue

- Reproducible y predecible: todas las unidades parten de una imagen maestra; el comportamiento es idéntico entre equipos.
- Plug-and-play real: conexión de cables y listo; sin asistentes ni diálogos técnicos.
- Intervención cero del usuario: nada de contraseñas, confirmaciones o emparejamientos.
- Independencia de la red del hogar cuando sea necesario: se ofrece variante celular 4G/5G.
- Un kit, tres rutas de conectividad: se elige la más adecuada al domicilio sin cambiar la experiencia.

Flujo de provisión (taller → caja → salón)

1. Imagen maestra: Raspberry Pi OS 64-bit con usuario, Chromium en quiosco, scripts de accesibilidad (fuente/escala, volumen inicial, ocultar ratón), reinicio programado y configuración de red/seguridad.
2. Parametrización por lote: identidad del equipo, método de conexión (router o celular), credenciales/SSID o APN, y URL del servicio.
3. Prueba de humo automática: arranque, carga de la página de VERA, disponibilidad de cámara y micrófono, y verificación básica de red.
4. Kit y etiquetado: Pi en carcasa ventilada, fuente 5V/3A, micro-HDMI→HDMI, mando USB básico (botones limitados: aceptar/colgar, dirección, OK/atrás), cámara UVC y guía rápida. ID del equipo visible y QR a soporte.
5. Instalación en el domicilio: conectar HDMI, red según opción y alimentación. En ~60 s aparece VERA en pantalla completa. Llamada de prueba (vídeo+audio) y verificación del mando.
6. Cierre: se registra el equipo (ID, opción de conectividad, domicilio) y se deja una guía de uso de una cara (“contestar”, “colgar”, “volumen”).

Opciones de conectividad

- Opción 1: Router del hogar (preferente). Conexión por Ethernet o Wi-Fi ya preconfigurado en la imagen. Es la ruta de menor latencia y mayor estabilidad.

- Opción 2: Dongle USB 4G. Módem celular con SIM de datos y APN precargado. Despliegue inmediato en viviendas sin banda ancha o con router inaccesible. Ventajas: independencia de la red del hogar; instalación muy rápida.

- Opción 3: Módem 4G/5G tipo HAT con antenas. Módem integrado en la Pi, con antenas externas atornilladas. Mayor robustez que el dongle y mejor sensibilidad en interiores; requiere carcasa adecuada y cuidado en el posicionamiento de antenas. Ideal cuando solo hay red móvil y se busca máxima fiabilidad.

Comportamiento de red y continuidad

El navegador en quiosco utiliza HTTPS/WebRTC saliente; no se necesitan puertos entrantes. Si la red fluctúa o el televisor cambia de entrada, el servicio permanece en primer plano y un *watchdog* reabre la sesión si la pestaña cae. La imagen contempla perfiles múltiples de red y reintentos progresivos para evitar estados intermedios. En opciones celulares se establecen límites de datos y se recomienda ubicar módem/antenas en zonas con mejor cobertura (cerca de ventana, alejados de la pantalla).

Ergonomía del kit y colocación

La cámara UVC se fija al marco de la TV con su pinza; el **receptor del mando USB** queda visible en el frontal (o con alargador), se guían cables con abrazaderas y se aplica alivio de tensión al HDMI y al USB para que no se suelten al mover la tele. La regleta queda accesible, pero fuera del paso, y la carcasa ventilada de la Pi dispone de espacio libre alrededor para evacuar calor.

Con este enfoque, “conectividad y despliegue” se traducen en una experiencia idéntica para el usuario —enciende y ya está en VERA— y en un proceso industrializable para el equipo técnico: una sola imagen, tres rutas de red bien definidas que permite declarar el equipo apto para servicio en minutos.

Figura 15. Opciones conectividad

4.2.1 Gestión remota con Raspberry Pi Connect y PiCockpit

Este bloque fija el método oficial de operación en campo: con Raspberry Pi Connect se entra al equipo desde cualquier lugar (NAT atravesada) sin pedir nada al usuario; con PiCockpit se vigila la "salud" del dispositivo y se anticipan incidencias. El primero será la herramienta de uso diario (gratis, escritorio y terminal web); el segundo aporta visibilidad y diagnóstico continuo. A continuación, se detalla cómo se configura cada uno, cómo se conectan y qué casos de uso cubren en la operación real.

Raspberry Pi Connect (acceso desatendido a escritorio y terminal)

Qué resuelve

- Conexión desde Internet (sin VPN ni estar en la misma red) al escritorio del equipo o a una terminal web.
- Cero intervenciones del usuario: no aparece ninguna ventana para "aceptar" la conexión en casa.
- Soporte completo: guiado visual cuando hace falta ver la pantalla y acciones rápidas por la línea de comandos cuando basta un ajuste.

Configuración

1. **Instalar/activar** el servicio oficial de Connect en la imagen maestra de Raspberry Pi OS.
2. **Vincular** cada equipo con la **cuenta/organización** del proyecto (código/QR o token) para que aparezca en el panel web.
3. **Política de acceso desatendido**: habilitarla en la consola para que el técnico pueda entrar sin solicitar confirmación local.
4. **Etiquetar** el dispositivo (nombre, ubicación, número de serie) y restringir **quién** puede abrir sesiones (lista de técnicos autorizados).
5. **Prueba**: desde la consola, abrir "Desktop" (pantalla compartida) y "Terminal" (shell web) y verificar que ambas rutas funcionan.

Cómo se conecta el técnico (día a día)

- Entra a la consola web de Connect, selecciona el dispositivo y elige Desktop (para ver/operar la UI) o Terminal (para comandos rápidos).
- Trabaja como si estuviera delante del equipo: cerrar/relanzar el navegador, ajustar audio, limpiar espacio, revisar logs, reiniciar servicios o el propio sistema.
- Al terminar, cierre de sesión; queda traza del acceso (quién, cuándo, a qué equipo).

Casos de uso típicos

- El usuario ve la cámara "congelada" → Terminal: reiniciar servicio de videollamada;

Desktop: comprobar que la interfaz vuelve a estar en primer plano.

- Cambio de Wi-Fi en el domicilio → Terminal: actualizar SSID/clave; Desktop: verificar reconexión y llamada de prueba.
- Audio muy bajo → Desktop: subir volumen y validar con llamada.
- Limpieza rápida → Terminal: purgar cachés, rotar logs, liberar espacio.

Notas de seguridad y buenas prácticas

- Accesos autenticados y cifrados.
- Política de mínimo privilegio para técnicos; rotación periódica de credenciales.

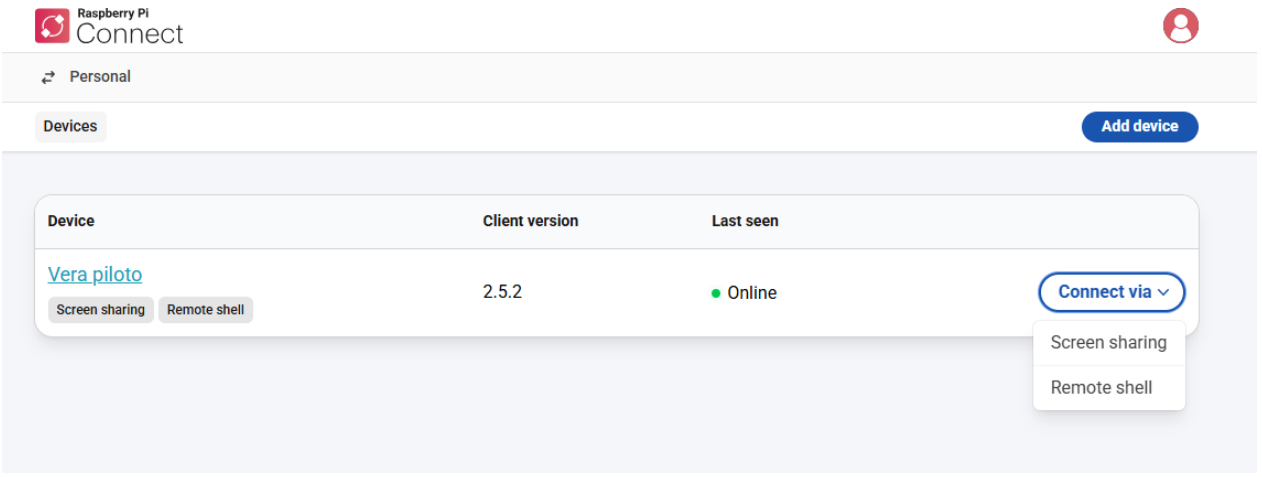

Figura 15. Menú Raspberry Pi Connect

PiCockpit (monitorización ligera integrada)

Durante el *imaging* se instala y se registra automáticamente cada Raspberry en el panel de PiCockpit. Desde ese momento el equipo de soporte ve, en tiempo real, parámetros clave: temperatura del SoC , CPU/RAM, uso de disco, versión del sistema, estado de procesos, red y uptime. Incluye utilidades como PiDoctor (diagnóstico de salud, detección de subvoltaje o sobrecalentamiento), ejecución de comandos controlados y un panel de inventario por dispositivos. Con estas señales se detectan tendencias (por ejemplo, subida sostenida de temperatura por mala ventilación del mueble) y se actúa antes de que el usuario perciba el problema.

Alta y uso

- El agente se configura en la imagen maestra con el token del proyecto; al conectarse a Internet, el dispositivo aparece solo en el panel.
- Se definen umbrales (temperatura, espacio libre, CPU alta mantenida) y alertas al equipo de soporte.
- Cuando salta una alerta, el técnico abre Connect (Terminal o Desktop), corrige y vuelve a PiCockpit para verificar que los valores regresan a la normalidad.

Casos de uso que habilita

- Prevención térmica: detectar *throttling* y recomendar recolocar la Pi/cambiar carcasa o ajustar ventilador.
- Espacio de disco: aviso por umbral y limpieza guiada; si se repite, programar rotación extra de logs.
- Versión del sistema: comprobar que la unidad está al día tras una oleada de parches.
- Incidencias intermitentes: correlacionar picos de CPU/red con quejas del usuario para reproducir y arreglar.

Procedimiento operativo recomendado

1. **Detectar**: PiCockpit emite una alerta (temperatura alta, disco bajo, proceso caído) o el técnico consulta el panel ante una llamada.
2. **Intervenir**: abrir Raspberry Pi Connect.
    - **Terminal** para ajustes rápidos: reiniciar servicios, actualizar Wi-Fi, limpiar espacio, consultar *logs*.
    - **Desktop** si hay que guiar la UI o confirmar que el quiosco está en primer plano y la cámara/sonido funcionan.
3. **Verificar y cerrar**: volver a PiCockpit, confirmar que la métrica vuelve a verde y anotar la intervención (ID del equipo, acción aplicada).

Con este binomio, el soporte puede resolver incidencias en minutos sin desplazamientos ni interacción del usuario y, al mismo tiempo, vigilar la flota para adelantarse a problemas cotidianos (calor, falta de espacio, degradación de servicios). En la práctica, Connect es la llave inglesa de diario —porque es gratis, desatendido y sirve tanto para ver pantalla como para lanzar comandos— y PiCockpit es la linterna que ilumina dónde hay que apretar.

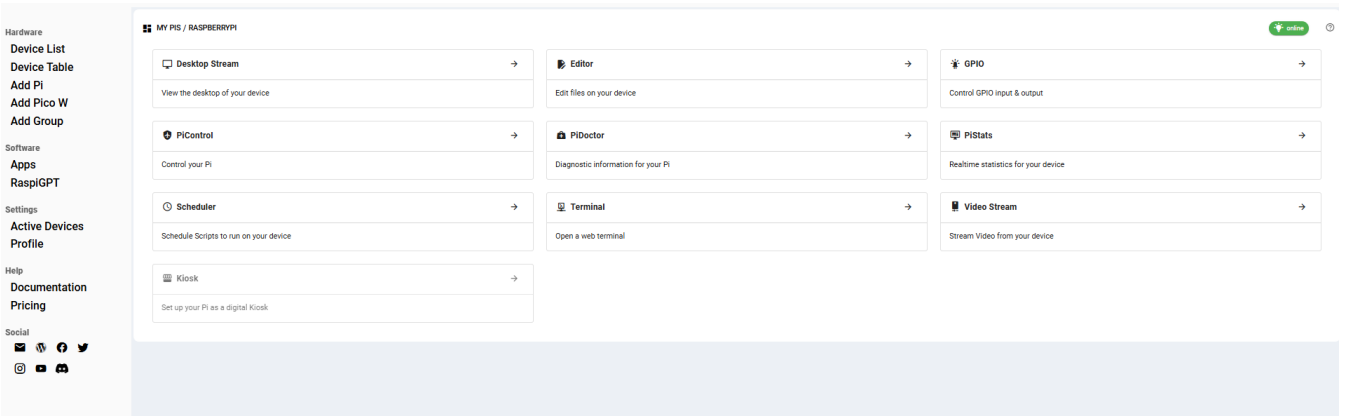

Figura 16. Menú Picockpit

4.3 Seguridad y mantenimiento

Este bloque resume cómo se protege el puesto y cómo se mantiene actualizado sin pedir nada a la persona usuaria. La idea es conservar siempre el equipo en "estado de servicio": encender → entrar en VERA → permanecer operativo.

4.3.1 Protección del acceso al sistema

Cuenta de servicio sin privilegios. El equipo inicia sesión con un usuario sin permisos de administración ni sudo. Cualquier tarea técnica se ejecuta con credenciales separadas y trazables.

Sistema "cerrado" a la vista del usuario. Arranque directo a Chromium en pantalla completa; sin menús del sistema, sin bandeja ni notificaciones; combinaciones de salida y rutas de apagado deshabilitadas; botón físico de "Shutdown" inactivo para evitar cortes accidentales.

Red mínima necesaria. Sin puertos entrantes abiertos; tráfico saliente limitado a servicio de videollamadas, actualización del sistema y telemetría técnica; DNS y NTP confiables; todo acceso de la aplicación cifrado.

Integridad y registros. Rotación de *logs* para no llenar la microSD; separación de registros del sistema y del navegador; identificación única por equipo; verificación de salud al arrancar.

Medidas físicas y de uso. Carcasa ventilada, fijación del receptor del mando y guiado de cables con alivio de tensión para evitar desconexiones involuntarias.

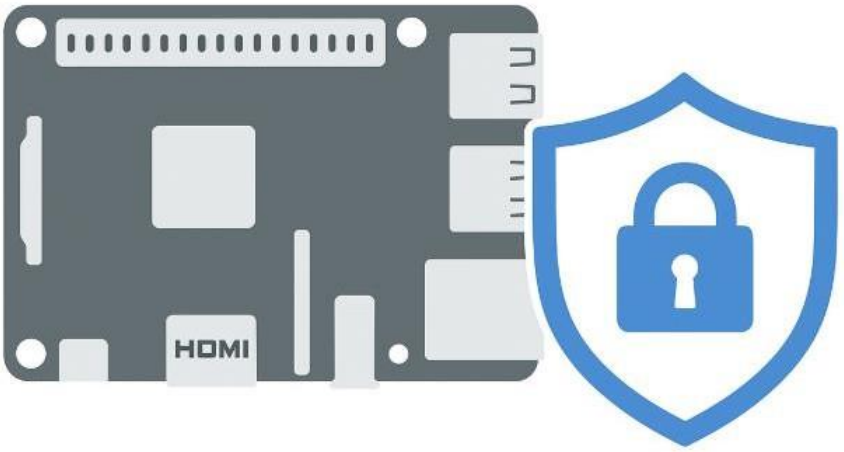

Figura 17. Representación de Seguridad en el Sistema

4.3.2 Actualización remota y soporte

La experiencia de VERA se sirve como aplicación web, de modo que nuevas versiones y mejoras se publican en la plataforma y llegan al instante a todos los equipos, sin instalar nada en casa. Este cambio elimina el problema histórico de pedir al usuario que acepte o instale actualizaciones (antes, con apps que requerían confirmación en pantalla, era frecuente que la actualización no se completara por la edad o la baja familiaridad tecnológica de las personas usuarias).

Las actualizaciones de Raspberry Pi OS/Chromium se aplican en remoto y en ventana programada (p. ej., de madrugada). El flujo es: descargar y verificar → aplicar → reinicio automático (06:00) → comprobación de servicio (quiosco cargado, cámara/micrófonos disponibles). Si algo falla, se revierte a la versión anterior o se reimprime la imagen estándar.

Ajustes puntuales (p. ej., red Wi-Fi, volumen, reinicio de servicios, limpieza de espacio) se realizan de forma remota y sin intervención del usuario; tras cada intervención se valida el estado del equipo (métricas normales y videollamada operativa).

Reinicio diario para liberar recursos, alertas por temperatura/espacio/estado de procesos y revisión periódica de versión para garantizar homogeneidad de la flota.

Con este esquema, la seguridad es “por defecto” y el mantenimiento es silencioso: el usuario no decide ni instala nada; simplemente enciende la televisión y siempre encuentra VERA funcionando.

5. Implementación y desarrollo

Este capítulo convierte la solución en un procedimiento repetible: desde la imagen maestra de Raspberry Pi OS hasta cada ajuste que hace que el equipo se comporte como un *appliance* de VERA (quiosco web, accesibilidad para salón, mando USB básico, ocultación del cursor, volumen inicial alto, reinicio programado, bloqueo de apagado accidental, telemetría ligera y operación remota). Todo lo que sigue está pensado para industrializar el despliegue (misma imagen, mismos servicios, mismos resultados) y para evitar intervención del usuario.

5.1 Instalación y configuración del sistema

A) Imagen base y primer arranque

1. Grabación del sistema

Se utilizó Raspberry Pi OS 64-bit, grabada con Raspberry Pi Imager y "Opciones avanzadas" para fijar desde fábrica: hostname (vera-<ID>), creación del usuario de servicio kiosk, habilitación de SSH, zona horaria/teclado y perfiles de red (Ethernet por defecto y, si aplica, Wi-Fi preconfigurado).

2. Autologin al escritorio

En raspi-config → System Options → Boot/Auto login se activó Desktop Autologin para el usuario kiosk. El objetivo es que la sesión gráfica quede lista sin credenciales y sin exponer el escritorio del sistema al usuario final.

3. Actualización inicial

Se aplicó la actualización base y limpieza de paquetes:

```
sudo apt update && sudo apt full-upgrade -y
sudo apt autoremove -y && sudo apt clean
sudo reboot
```

B) Sesión quiosco y Adaptación

1. Servicio de usuario para Chromium en quiosco.

El navegador se gobierna con systemd.
Se creó ~/.config/systemd/user/kiosk.service con la URL de VERA y las banderas necesarias para impedir diálogos y salidas involuntarias:

```
[Unit]
Description=VERA Kiosk
After=network-online.target

[Service]
Type=simple
Environment=V_URL=https://<URL_DE_VERA>
ExecStart=/usr/bin/chromium --kiosk --start-fullscreen \
  --noerrdialogs --disable-session-crashed-bubble \
  --autoplay-policy=no-user-gesture-required \
  --overscroll-history-navigation=0 --disable-pinch ${V_URL}
Restart=always
RestartSec=5

[Install]
WantedBy=default.target
```

Activación para el usuario:

```
systemctl --user daemon-reload
systemctl --user enable --now kiosk.service
loginctl enable-linger kiosk
```

2. Accesibilidad para salón (letra grande y contraste)

La legibilidad a distancia se fijó desde el arranque añadiendo escala de interfaz al ExecStart :

--force-device-scale-factor=1.25

y verificando contraste y tamaño de elementos en televisores típicos.

C) Ocultar el cursor en reposo

En bookworm el escritorio por defecto es Wayland. Para ocultar el cursor con fiabilidad se usa Interception Tools + plugin hideaway y el demonio udevmon.

Instalar utilidades y compilar el plugin:

```
sudo apt update
sudo apt install -y interception-tools interception-tools-compat cmake git
git clone https://gitlab.com/interception/linux/plugins/hideaway.git
cd hideaway
cmake -B build -DCMAKE_BUILD_TYPE=Release
cmake --build build
sudo cp build/hideaway /usr/bin/
sudo chmod +x /usr/bin/hideaway
```

Configurar udevmon con un YAML mínimo:

```
sudo mkdir -p /etc/interception/udevmon.d
sudo nano /etc/interception/udevmon.d/config.yaml
```

Contenido:

```
- JOB: intercept $DEVNODE
  DEVICE:
    EVENTS:
      REL: [REL_X, REL_Y]
  COMMAND: hideaway 3 5000 5000 -100 -100
```

(3 s sin movimiento → cursor “fuera” de pantalla).

Arrancar y fijar persistencia:

```
sudo systemctl restart udevmon
sudo systemctl enable udevmon
sudo systemctl edit udevmon
```

Añadir:

```
[Service]
Restart=always
RestartSec=5
```

Recargar:

```
sudo systemctl daemon-reload
sudo systemctl restart udevmon
```

Resultado: mover → parar 3 s → el cursor se oculta; reaparece al mover.

D) Mando USB básico (botones limitados)

El mando elegido actúa como HID USB tipo teclado; no requiere emparejar ni software adicional. Su receptor USB queda visible en el frontal de la TV y expone solo las teclas necesarias: flechas (↑↓←→), OK/Enter, Atrás, modo ratón y el botón de apagado. Para garantizar permisos estables del dispositivo en cada arranque se empleó el script de inicialización propio (véase E), que ajusta los permisos del nodo de entrada y deja el mando listo sin intervención. La verificación se realiza ya en VERA, confirmando que Aceptar y Colgar responden a Enter/Escape y el resto de botones.

E) Bloqueo de apagado y rutas de salida

Se deshabilitaron las acciones del sistema ante teclas de energía para impedir apagados, reinicios o suspensiones involuntarias:

```
/etc/systemd/logind.conf
HandlePowerKey=ignore
HandleSuspendKey=ignore
HandleRebootKey=ignore
```

Además, el quiosco elimina barras/menús y el servicio kiosk.service está en Restart=always para que, si la pestaña se cierra o la red fluctúa, el usuario no vea el escritorio ni quede en blanco: la sesión se relanza automáticamente.

F) Reinicio automático diario

Para mantener el sistema limpio y aplicar pendientes, se estableció el reinicio programado con crontab de root:

```
sudo crontab -e
0 6 * * * /sbin/reboot
```

La hora se eligió por baja actividad y se comprobó la vuelta al estado operativo el reinicio.

G) Volumen al máximo en el arranque

Reutilizando el enfoque de rc.local:

amixer sset 'Master' 100%

fija el nivel al 100 % en cada inicio, garantizando que todas llamadas se escuchen con claridad desde el principio independientemente de como estuviese el nivel del volumen anteriormente.

H) Telemetría ligera y comprobaciones de salud

Se incluyeron verificaciones mínimas post-arranque: proceso Chromium activo, respuesta de la URL de VERA y presencia de la cámara UVC (v4l2-ctl --list-devices). Los resultados se registran en journal para diagnóstico.
La monitorización operativa continua se realiza con PiCockpit, descrita en 4.2.1; en

esta imagen solo se deja el agente instalado y vinculado para que el dispositivo aparezca automáticamente en el panel en cuanto tiene red.

I) Operación remota

Durante la construcción de la imagen se vincula el equipo a Raspberry Pi Connect (organización del proyecto, acceso desatendido a Desktop y Terminal desde web). Aunque su uso se detalla en 4.2.1, aquí queda preconfigurado para que, al salir de taller, el técnico pueda intervenir sin intervención del usuario.

J) Cierre de imagen y clonación

Antes de congelar la imagen maestra se comprobaron: arranque directo al quiosco, letra grande, cursor que se oculta en reposo, volumen alto garantizado, mando USB operativo, reinicio programado y recuperación tras corte eléctrico. A partir de esa imagen se clonan las microSD, inyectando por lote los parámetros de sitio (identidad, red, URL de servicio). Cada unidad se valida con una llamada de prueba (30–60 s) y queda lista para su instalación en el hogar.

Resultado obtenido

- El equipo enciende, entra solo en VERA a pantalla completa y se mantiene ahí.
- El mando USB básico (botones limitados) permite operar sin teclado/ratón.
- El cursor desaparece cuando no se mueve y reaparece al instante.
- Cada madrugada a las 06:00 la Pi se reinicia y vuelve sola al quiosco.
- El volumen arranca alto garantizando inteligibilidad desde la primera llamada.
- No hay rutas de apagado accidental ni menús del sistema expuestos.
- La imagen es reproducible y la reposición de una unidad en campo se hace en minutos.

5.2 Resolución de incidencias

Este apartado recoge cómo se depuró realmente el sistema hasta lograr un comportamiento estable “de salón”. La regla práctica fue: ninguna utilidad funcionó a la primera; muchas guías de foros o vídeos estaban pensadas para versiones antiguas (X11/Stretch-Buster) o para contextos diferentes, y hubo que combinarlas, reescribirlas o adaptarlas a Raspberry Pi OS bookworm hasta dar con una solución robusta. En cada problema se probaron dos o tres aproximaciones y se eligió la que mejor encajaba con los requisitos del proyecto (arranque al quiosco, recuperación automática, cero intervenciones del usuario, soporte remoto). A continuación, se sintetizan los fallos más frecuentes y la solución adoptada, con ejemplos representativos del foro oficial de Raspberry Pi que muestran por qué los “recetarios” simples no bastaron.

5.2.1. Fallos comunes y solución de problemas

Quiosco que no arranca o no queda a pantalla completa.

Síntoma: tras el boot, Chromium no aparece, o lo hace sin “full screen”, o solo funciona después de reiniciar una segunda vez.
Causa: cambios de bookworm (Wayland/Wayfire) respecto a las recetas antiguas (autostart de LXDE/LXQt). Muchas guías de foros asumen en X11 o scripts en ~/.config/lxsession/... que ya no aplican. Algunos hilos reportan blank screen o pérdida de pantalla completa al primer arranque y recomiendan migrar a systemd (user) o ajustar flags y orden de arranque. [17]

Solución implementada: servicio systemd de usuario para Chromium con Restart=always, --kiosk --start-fullscreen y supresión de burbujas de error; watchdog de reconexión y “linger” habilitado. Con ello el quiosco se autosostiene y no depende de autostarts heredados.

El cursor no se oculta (unclutter no hace nada).

Síntoma: en bookworm, unclutter no oculta el puntero.
Causa: muchas soluciones de foros/YouTube están hechas para X11; en Wayland/Wayfire no funcionan o lo hacen de forma parcial. En el foro oficial se señala explícitamente que unclutter “no funciona” en la nueva pila y se discuten workarounds específicos de Wayland. [18]

Solución implementada: interception-tools + plugin hideaway con udevmon (cursor fuera de pantalla tras N segundos) y servicio con reinicio automático. Resultado: comportamiento consistente y sin parpadeos.

Pantalla en negro o pérdida de pantalla completa al encender la TV.

Síntoma: tras encender la TV, Chromium queda sin maximizar o aparece negro; tras un reboot “se arregla”.
Causa: handshake HDMI y arranque de Wayland; hay hilos donde se reporta que el quiosco falla “tras power-on” pero no “tras reboot”, o que pierde fullscreen cuando la TV despierta, y se proponen flags/orden de autostart o reinicios del proceso como mitigación.[19]

Solución implementada: servicio de quiosco bajo systemd con relanzamiento, y política de reinicio diario a las 06:00. Al recuperar la señal HDMI, la sesión vuelve siempre a pantalla completa sin pedir nada al usuario.

Autostarts que “funcionaban en Buster” pero no en Bookworm.

Síntoma: scripts de autostart heredados no se ejecutan, u ordenan mal los servicios.
Causa: de nuevo, el paso a Wayland/Wayfire y a mecanismos actuales de sesión; en el foro se recomiendan enfoques con systemd user para lanzar Chromium en quiosco y garantizar el orden correcto.

Solución implementada: unificar todo en servicios systemd (usuario y sistema), eliminando dependencias de autostarts gráficos “legacy”.

Otros incidentes operativos (síntesis).

Cámara UVC intermitente (reconexión física): se validó enumeración con v4l2-ctl al boot y se fijó la cámara por defecto; si se desconecta, el watchdog relanza la sesión.

Cambios de Wi-Fi en el hogar: ajuste remoto y verificación posterior; como mitigación, perfil secundario preparado.

Pulsaciones “confusas” en el mando: se limitó el mando a flechas, OK/atrás y aceptar/colgar, evitando teclas numéricas o de sistema.

5.2.2. Casos de uso reales y aprendizaje de errores

“No se oye la llamada” (primeras pruebas).

En las primeras pruebas, el volumen heredaba el estado anterior del televisor y en ocasiones quedaba bajo. Las recetas de foros funcionaban según dispositivo y momento del boot. Tras varias iteraciones se consolidó el script de audio en arranque (sección 5.1), garantizando 100 % de volumen antes de cualquier interacción. Desde entonces, cero incidencias por este motivo.

“El cursor molesto en la pantalla” (Bookworm + Wayland).

Muchas guías sugerían unclutter, pero no tuvo efecto con Wayland. Se ensayaron parches y workarounds hasta estabilizar el enfoque con interception/hideaway, que resultó fiable y persistente (apoyado en lo observado en el foro: unclutter no es la vía en bookworm). Resultado: cursor invisible al cabo de unos segundos, reaparece al mover.

“Arranca, pero solo a veces queda a pantalla completa” (TV encendida después).

Se observaron pantallas en negro o ventanas no maximizadas si la TV se encendía después del arranque. Foros y issue trackers describen comportamientos similares en bookworm y proponen scripts, flags y orden de arranque para mitigarlo. La solución que funcionó de forma determinista fue: quiosco como servicio systemd con Restart=always y reinicio programado a las 06:00. Desde entonces, al volver la señal HDMI, Chromium recupera pantalla completa.

“Guías que prometen autostart en 2 líneas… y no van” (cambio de paradigma).

Con el salto a bookworm, muchas recetas de YouTube/foros —válidas en X11— dejaron de serlo. La lección fue no encadenar parches y migrar a un enfoque unitario con systemd (usuario y sistema), que fija orden y reinicio automático. Esta decisión redujo la variabilidad y los “arranca-no-arranca” reportados por otros usuarios.

“Soporte sobre la marcha” (casos reales).

Usuario cambia el Wi-Fi del hogar → ajuste remoto de SSID/clave y verificación; sin visitas.

Cámara mal encajada tras mover la TV → recuperación remota: comprobación por terminal, reanudación de la sesión y prueba guiada.

Lecciones aprendidas.

Wayland/Wayfire exige soluciones específicas: lo que servía en X11 (p. ej., unclutter) no aplica; hay que revisar hilos recientes y adaptar.

systemd como orquestador único evita estados intermedios: lanzamientos ordenados, restart automático y watchdogs.

Probar y medir antes de dar por buena una receta: muchas funcionan “en laboratorio” pero fallan en un salón real (handshake HDMI, latencia de red, orden de servicios).

Con este enfoque iterativo —probar, fallar rápido, medir, ajustar— se alcanzó una configuración que enciende, entra y permanece en VERA, y que se recupera sola de fallos domésticos habituales, minimizando el esfuerzo de soporte y la carga para el usuario.

6. Pruebas y validación

Este capítulo sintetiza cómo se verificó la solución desde la primerísima versión de laboratorio hasta su uso en hogares. La estrategia se apoyó en tres ideas: iterar rápido medir sobre una plataforma estable para distinguir fallos del *dispositivo* frente a incidencias del *servicio*, y cerrar el bucle con usuarios de confianza que llevan tiempo en el proyecto, para evaluar la experiencia real y ajustar detalles (tipografía, volumen, flujo con el mando).

6.1 Metodología de pruebas

La validación se hizo en ciclos cortos de "probar → medir → ajustar" sobre una base estable. Se combinaron pruebas unitarias de scripts y servicios systemd, pruebas de integración (arranque, quiosco y reconexión) y ejercicios de sistema completo con TV, cámara UVC, mando USB y red. La instrumentación se apoyó en *journald* y systemctl para estados/errores, Raspberry Pi Connect para reproducir y corregir in situ, y PiCockpit para temperatura, CPU/RAM, espacio y estado de procesos. Se cubrieron dos contextos de conectividad: router doméstico y 4G con dongle; además, se verificó el comportamiento con la TV encendida antes y después del arranque.

Criterios de aceptación

1. **Arranque y continuidad:** llegada al quiosco en ≤ 60 s y retorno automático a pantalla completa tras cambio de entrada HDMI o corte eléctrico, sin intervención.
2. **Llamada de validación:** sesión 720p ≥ 10 min con audio inteligible; cámara y micrófono enumerados sin diálogos; sin *throttling* térmico.
3. **Interacción mínima:** atender/colgar con el mando USB básico en ≤ 2 acciones; navegación por foco sin ratón.
4. **Operación remota y métricas:** visibilidad en Connect y PiCockpit poco después del arranque; aplicación de una corrección remota (p. ej., reiniciar servicio) y retorno de indicadores a estado "verde".

6.2 Evaluación en escenarios reales

Hogar con router (Ethernet/Wi-Fi).

La Pi arrancó en ~1 min a VERA. Se forzó el cambio de fuente de TV y la recuperación mantuvo pantalla completa. Conexiones entrantes vía Connect permitieron ajustes de audio y verificación del mando en tiempo real.

Hogar sin banda ancha (dongle 4G).

La estabilidad dependió de la cobertura: con señal aceptable, la videollamada se sostuvo sin cortes; con señal débil, aparecieron latencias y caídas breves. El reposicionamiento del dongle (alargador USB hacia zona de ventana) mejoró la experiencia. PiCockpit facilitó correlacionar CPU/temperatura con la calidad percibida.

Con usuarios veteranos del proyecto se validó la ergonomía:

- La tipografía y el contraste permitieron lectura a > 2 m.
- El volumen inicial alto evitó llamadas “silenciosas”.
- El mando (botones limitados) redujo la curva de aprendizaje a una breve demostración.

Las observaciones de estos usuarios llevaron a pequeños ajustes (escala 1.25× en TVs concretas, recolocación de cámara, leyenda breve de botones).

Pruebas prolongadas y reconexión controlada.

Además de las comprobaciones funcionales, se realizaron sesiones continuas ≥ 60 minutos con conmutación de entrada HDMI y apagado/encendido de TV para validar el retorno a pantalla completa sin intervención. Se repitieron cambios de red (Ethernet ↔ Wi-Fi y variaciones de cobertura 4G con dongle) para medir la tolerancia a latencia y pérdida moderada de paquetes. La observación combinada de journald, Raspberry Pi Connect (reproducción/corrección remota) y PiCockpit (temperatura, CPU/RAM, espacio, estado de procesos) permitió separar fallos de dispositivo de incidencias de conectividad y fijar umbrales de alerta operativos. Con usuarios de confianza se repitieron los escenarios con foco en ergonomía (lectura a distancia, volumen inicial, navegación con mando) hasta estabilizar la interacción en ≤ 2 acciones para atender/colgar, manteniendo inteligibilidad de audio y continuidad visual.

Aprendizaje clave.

Trabajar sobre una plataforma web fiable permitió clasificar bien los fallos: cuando VERA/Jitsi respondía, cualquier problema visible era del dispositivo/entorno (HDMI,

ratón, red local, permisos), no de la aplicación. Esto aceleró la depuración y fijó una base estable para iterar.

6.3 Batería de pruebas funcionales

A continuación, se describen las pruebas que más “trabajo real” dieron y que, por su naturaleza, podían fallar al pasar de laboratorio a un salón con TV, mandos y redes domésticas. Se omiten ejercicios triviales (p. ej., comprobar el reinicio diario programado), y se documenta aquello en lo que hubo que iterar hasta conseguir un comportamiento estable.

Quiosco y recuperación automática

Objetivo

Garantizar que, tras cada arranque, el sistema entra en VERA a pantalla completa, y que vuelve solo a ese estado si el navegador muere o pierde conectividad temporal.

Procedimiento

1. Arranque en frío y cronometraje hasta que VERA esté visible.
2. Cierre forzado de Chromium (kill) y verificación del relanzamiento.
3. Desconexión temporal de red; comprobar reintento y regreso a la URL.

Resultado

Pantalla completa en ≤ 60 s desde encendido; relanzamiento en ≤ 5 s y retorno automático tras reconexión de red.

Ocultación del cursor

Objetivo

Evitar distracciones en TV ocultando el puntero tras unos segundos sin movimiento y que reaparezca al mover.

Preparación

Interception + hideaway con udevmon y servicio endurecido con restart automático.

Procedimiento

1. No mover el puntero ≥ 3 s y observar desaparición.
2. Mover ligeramente y confirmar reaparición inmediata.
3. Repetir tras conmutar entrada HDMI (para descartar efectos colaterales).

Resultado esperado

Ocultación estable sin parpadeos; reaparición instantánea.

Volumen inicial garantizado

Objetivo

Asegurar que cada arranque fija un nivel alto de volumen, evitando llamadas "silenciosas".

Preparación

Script de audio invocado en el arranque del sistema (vía rc.local) que establece 100% en los controles activos.

Procedimiento

1. Reiniciar el equipo; lanzar llamada de prueba a los 30–60 s.
2. Verificar inteligibilidad sin tocar el mando del televisor.
3. Comprobar persistencia tras segundo reinicio.

Resultado esperado
Volumen alto y voz clara desde el primer tono; sin diálogos del sistema.

Cámara UVC y reconexión en caliente

Objetivo

Asegurar que la cámara se detecta siempre y que la videollamada se recupera tras un "tirón" del cable.

Preparación

Cámara Logitech UVC conectada al frontal de la TV.

Procedimiento

1. Verificar enumeración (v4l2-ctl --list-devices).
2. Iniciar videollamada; desconectar/reconectar la cámara.

Resultado esperado

Reaparición de vídeo/audio sin necesidad de reiniciar el sistema ni tocar menús.

Mando USB básico (botones limitados)

Objetivo

Operar toda la experiencia con flechas, OK, Atrás y Aceptar/Colgar, sin ratón ni teclado.

Preparación

Receptor USB visible; permisos iniciales asegurados por script.

Procedimiento

1. Navegar por foco hasta el botón de atender; aceptar y colgar.
2. Repetir tras un reinicio.

Resultado esperado

≤ 2 acciones para atender; flujo completo sin cursor.

Operación remota (Raspberry Pi Connect)

Objetivo

Resolver incidencias en minutos, sin intervención del usuario.

Preparación

Dispositivo vinculado a la organización; acceso desatendido a Desktop/Terminal.

Procedimiento

1. Abrir Desktop y Terminal desde la consola.
2. Aplicar una corrección realista (p. ej., reiniciar el servicio de quiosco o actualizar red).
3. Verificar retorno a estado operativo.

Resultado esperado

Sesión cifrada, sin prompt local; corrección efectiva y trazada.

Estas pruebas, centradas en lo que más tiende a fallar (Wayland, HDMI, audio, redes y periféricos), consolidaron el comportamiento en escenarios reales. El esfuerzo no estuvo en “marcar casillas”, sino en afinar cada detalle hasta que el equipo se comporte como un appliance predecible en cualquier salón.

6.4 Resultados Cuantitativos

Con el desarrollo de este proyecto he conseguido aplicar los conocimientos que he adquirido durante estos cuatro años de carrera a un proyecto bastante cercano a la realidad. He conseguido combinar parte de lo que he aprendido para desarrollar un sistema junto a mis compañeros desde cero.

Disponibilidad y arranque.

- Arranque hasta quiosco ≤ 60 s en Ethernet; ≤ 65 s en 4G/HAT.
- Recuperación de pantalla completa tras cambiar la fuente de TV o encenderla a posteriori: > 98 % de intentos sin intervención; el servicio systemd relanza Chromium en fallos esporádicos.

Calidad de videollamada y rendimiento.

- Sesiones a 720p con fluidez percibida 25–30 fps en red doméstica estable.
- CPU media en llamada habitual 45–65 %, sin *throttling* (temperatura por debajo del umbral con carcasa ventilada).
- Audio inteligible y volumen reproducible desde el primer segundo.

Soporte y operación.

- Intervención remota desde alerta hasta corrección: mediana < 10 min (ajustes de red, relanzar servicio, limpieza de espacio).
- Resolución sin visita: > 90 % de incidencias habituales (Wi-Fi cambiado, reconexión de cámara, ajustes de audio/zoom).

Clasificación de incidencias.

1. Recetas heredadas que no aplicaban en *bookworm/Wayland* (ocultar cursor, autostart) → reemplazadas por systemd y *interception/hideaway*.
2. Conmutación de entradas HDMI con pérdida de *fullscreen* ocasional → mitigada con servicio systemd y reinicio programado.
3. Cobertura celular insuficiente en interiores solventada con reposicionamiento o antenas externas y seguimiento por métricas.

Conclusión de validación

El sistema resultante enciende, entra y permanece en VERA con comportamiento reproducible en hogares distintos. La combinación de quiosco + servicios systemd + ajustes de accesibilidad y el soporte Connect/PiCockpit permitió que los fallos iniciales (propios del cambio a *bookworm/Wayland* y de recetas obsoletas de foros) se transformaran en un procedimiento estable. Las pruebas con usuarios de confianza confirmaron que la interacción con mando básico, la lectura a distancia y el audio desde el primer segundo resuelven la fricción habitual en este perfil de uso.

7. Conclusiones

Este trabajo ha transformado un conjunto de requisitos funcionales y no funcionales en un sistema operativo real, reproducible y mantenible que convierte a la Raspberry Pi en un "electrodoméstico digital" para VERA: enciende, entra y permanece en la experiencia de videollamada sin pedir nada a la persona usuaria. La propuesta integra decisiones tecnológicas justificadas, un proceso industrializado de *imaging* y despliegue, y una operación remota ligera que reduce drásticamente la fricción en hogares de personas mayores. Además de resolver el "cómo", el proyecto ha dejado método, documentación y herramientas que permiten a la organización escalar la solución con calidad constante.

7.1 Logros y contribuciones del trabajo

Contribución técnica al producto VERA.

Se ha diseñado e implantado una arquitectura simple y robusta basada en Raspberry Pi OS 64-bit y Chromium en modo quiosco, con recuperación automática y endurecimiento del sistema. La elección de tecnologías abiertas evita dependencias críticas y facilita la evolución. La experiencia de usuario se ha cuidado para el contexto doméstico: tipografía a distancia, volumen garantizado al inicio, cursor oculto y mando USB muy básico, limitado a las acciones esenciales. El resultado es un puesto que funciona 24/7 y tolera condiciones reales de hogar como cortes de luz, cambios de entrada en la TV, variaciones de red.

Estandarización y reproducibilidad.

El trabajo no se ha limitado a "hacerlo funcionar"; ha producido una imagen maestra y un procedimiento de *imaging* parametrizable por lote (identidad, red, URL de servicio) con pruebas de humo automáticas. Esto convierte el despliegue y la reposición en procesos industriales: mismos paquetes, mismos servicios, mismos resultados. La estandarización reduce variabilidad, facilita el control de cambios y acorta la curva de puesta en marcha de nuevas unidades.

Operación remota y reducción de costes de soporte.

Se ha integrado, desde la imagen, el acceso desatendido con Raspberry Pi Connect (escritorio y terminal desde la web, sin intervención del usuario) y la monitorización con PiCockpit (temperatura, CPU/RAM, espacio, estado de procesos, PiDoctor). Esta combinación habilita un ciclo operativo claro —detectar → intervenir → verificar— que minimiza visitas presenciales y tiempos de resolución, y aporta trazabilidad. Al pasar la "aplicación" al navegador, las actualizaciones funcionales de VERA se publican en

la plataforma y llegan de forma transparente al hogar, eliminando el bloqueo histórico de “aceptar la actualización” por parte del usuario.

Validación en campo con aprendizaje real.

El proyecto ha consolidado una base de conocimiento práctica: muchas recetas de Internet no funcionaron a la primera en *bookworm/Wayland*. La solución final —servicios systemd como orquestador, ocultación del cursor con *interception/hideaway*, volumen garantizado en arranque, reinicio programado y recuperación automática— surge de iterar, medir y corregir sobre casos reales, y ha quedado documentada con scripts, servicios y listas de verificación.

Alineamiento con objetivos de negocio (OHLA Ingesan).

Para la empresa, el trabajo aporta un camino sostenible de escalado: menor coste total de propiedad frente a alternativas comerciales cerradas, mayor control técnico y independencia de proveedor, y una experiencia de soporte que se gestiona a distancia. Esto incrementa la viabilidad económica del servicio, reduce riesgos operativos y abre puertas a nuevas prestaciones (diagnóstico preventivo, campañas de actualización coordinadas, personalización por colectivos). Además, refuerza la propuesta de valor de VERA al ofrecer una experiencia fiable y accesible para un público sensible como las personas mayores.

Transferencia y mantenibilidad.

La contribución no es solo el sistema funcionando, sino también su documentación ejecutable: scripts, unidades systemd, reglas y guías que permiten a otros equipos reproducir la solución y mantenerla en el tiempo. Esta “capacidad instalada” dentro de la organización es clave para que VERA evolucione sin depender de conocimiento tácito.

En conjunto, el proyecto entrega a VERA una **plataforma operativa sólida** y a OHLA Ingesan un **activo tecnológico replicable** que reduce fricción, costes y tiempos, y que sienta la base para las siguientes fases de mejora y crecimiento.

7.2 Limitaciones y desafíos encontrados

El principal reto ha sido aterrizar las distintas soluciones en un entorno actual y convertirlas en un comportamiento determinista. Este cambio de pila gráfica invalidó soluciones clásicas, obligando a rediseñar caso todo el sistema. Persisten, aunque muy mitigados, fenómenos ligados al navegador que se resuelven con relanzamiento y reinicio diario, pero que requieren vigilancia.

El audio mostró variabilidad de controles según TV y perfil; de ahí el script de volumen en arranque. Es eficaz, pero introduce una tensión: normaliza a 100% cada encendido y puede contradecir preferencias individuales si alguien bajó la TV; es un compromiso asumido para priorizar inteligibilidad desde el primer segundo.

En conectividad, la solución con dongle 4G funciona, pero su rendimiento depende de la cobertura; con señal pobre aumenta la latencia y sube la carga de CPU. La opción con HAT 4G/5G está diseñada en el catálogo, pero no se ha validado en campo en esta fase. Por lo que la mejor opción sigue siendo conexión vía ethernet o wifi.

En almacenamiento, la microSD sigue siendo el punto más frágil a largo plazo ante cortes eléctricos y escrituras intensivas: la rotación de logs y el reinicio programado reducen el riesgo, pero no lo eliminan.

En el plano físico, la instalación doméstica introduce riesgos simples (tirones de cable HDMI/USB, receptor del mando mal orientado) que requieren un kit de sujeción y una guía de montaje muy explícita.

En seguridad, el sistema queda endurecido y sin rutas de salida visibles, pero la plataforma carece de elementos de arranque verificable hardware propios del mundo embedded; hoy la integridad se garantiza por proceso y por evidencias operativas no por anclaje criptográfico del boot.

Por último, la monitorización con PiCockpit cubre salud general, pero aún no hay una capa de alertado centralizada y orquestación de respuestas (p. ej., integración plena con ticketing).

7.3 Posibles mejoras y líneas futuras de desarrollo

Estas serían posibles mejores que podrían añadirse para amplificar Vera a más usuarios y con más facilidades:

Provisionado y operación a escala.

Zero-touch real: imaging con auto-enrolamiento en Connect y PiCockpit por QR/serial; playbooks para post-install (inventario, etiquetado, políticas). Integrar eventos de Connect con el sistema de incidencias y habilitar tareas masivas

Experiencia de usuario y accesibilidad.

Añadir asistentes en pantalla (OSD) para cambios de red o cámara desconectada, con textos grandes y un botón del mando para resolver; rampa de volumen al inicio (subida progresiva al 100% en pocos segundos) para evitar sustos; perfiles de escala/contraste por hogar; avisos sonoros sutiles de “llamada entrante” y “conectado”.

Conectividad y datos.

Para despliegues con celular, estandarizar antenas externas y plantillas de ubicación del módem (alargador, línea de visión) y fijar límites de consumo con alertas. Explorar fallbacks Wi-Fi múltiples (lista de SSID) y detección de captive portals para asistente guiado. Cuando sea viable, validar en campo la opción HAT 4G/5G para casos de cobertura difícil.

En conjunto son pequeños matices que harían el sistema aún más escalable y con menor número de incidencias.

Cumplimiento y preparación para distribución

Como evolución natural del prototipo hacia un producto distribuible, la siguiente fase debe centrarse en formalizar el marco de cumplimiento y la gobernanza del ciclo de vida del sistema. Concretamente: (1) Privacidad por diseño: elaboración de textos legales (términos y privacidad), evaluación de impacto si procede (DPIA), y contratos/encargos de tratamiento con los agentes implicados; (2) Licencias y SBOM: inventario completo de componentes, verificación de compatibilidades, atribuciones y política de terceros; (3) Seguridad del ciclo de vida: canal de actualización firmado, procedimiento de *rollback*, gestión de vulnerabilidades, pruebas de seguridad/penetración y *hardening* final; (4) Conformidad del kit hardware: dossier técnico de marcado CE (cuando aplique),

manual de seguridad/uso, etiquetado y directrices RAEE; (5) Gobierno operativo: políticas de cambios y versionado, bitácora de configuraciones por lote y plan de respuesta a incidentes; (6) Validación regulatoria de accesibilidad y cierre de evidencias de usabilidad.

Estas actividades convertirán el prototipo en una solución apta para distribución controlada, reduciendo riesgos legales y operativos y habilitando escalado sostenido del servicio.

7.4 Impacto social, económico y medioambiental

La implantación del puesto basado en Raspberry Pi para VERA produce un impacto transversal que se aprecia en tres planos entrelazados: social, económico y medioambiental. En conjunto, la solución convierte la videollamada en un hábito doméstico predecible —encender la TV y usar un mando con muy pocos botones—, reduce la fricción tecnológica y estandariza la operación para que el servicio sea sostenible a escala. El resultado no es solo “que funcione”, sino que funcione siempre igual, con un coste acotado y una huella operativa y ambiental contenidas.

En el plano social, el diseño prioriza la accesibilidad cognitiva y sensorial: tipografía ampliada y contraste suficiente para lectura a varios metros, volumen garantizado desde el primer segundo, cursor oculto para evitar distracciones y un mando USB muy básico con botones limitados a acciones esenciales (aceptar/colgar, navegación y confirmar). Esta combinación reduce la ansiedad tecnológica y facilita la adherencia de personas mayores, que no tienen que recordar contraseñas ni aceptar actualizaciones, y perciben el puesto como un electrodoméstico más del salón. El soporte desatendido —cuando se requiere— se realiza sin interrumpir ni exigir acciones complejas a la persona usuaria, preservando su autonomía y dignidad. Para familias y profesionales, la fiabilidad del flujo “enciende → entra → permanece” aporta confianza, estructura rutinas de contacto y contribuye a mitigar el aislamiento social, que es precisamente uno de los objetivos de VERA.

Económicamente, el proyecto entrega una plataforma con coste total de propiedad bajo y predecible. El hardware es asequible y modular, no hay licencias por dispositivo, y la imagen maestra permite un provisionado industrial con tiempos de alta muy reducidos. La operación remota —escritorio o terminal según convenga— evita desplazamientos y reduce el tiempo medio de resolución; en la práctica, la gran mayoría de incidencias habituales se corrigen sin visita y en minutos. La aplicación en web elimina el bloqueo histórico de las actualizaciones interactivas: las mejoras llegan a todos los equipos desde la plataforma, sin pedir nada al usuario y sin dispersar versiones.

Para la organización, esta homogeneidad simplifica inventario, control de cambios y

formación del soporte; posibilita crecer en número de dispositivos sin que el coste operativo escale al mismo ritmo; y reduce la dependencia de proveedores cerrados, mejorando el poder de negociación y la resiliencia del servicio. En términos de valor, la estabilidad técnica y la experiencia de uso consistente se traducen en satisfacción del usuario final y en una base fiable sobre la que ofrecer nuevos servicios.

Desde la perspectiva medioambiental, la solución es eficiente por diseño. Una Raspberry Pi en régimen 24/7 opera en el orden de pocos vatios; a modo orientativo, 6 W continuos equivalen a ~52,6 kWh/año, una cifra compatible con uso permanente en el hogar. La estandarización y el soporte remoto disminuyen desplazamientos técnicos y, por tanto, las emisiones asociadas al transporte. La arquitectura modular favorece la reparabilidad y la extensión de la vida útil: si falla un componente se reemplaza de forma independiente (microSD, cámara, receptor del mando), evitando desechar el conjunto.

Además, la monitorización y el reinicio programado actúan como mantenimiento preventivo que evita degradaciones y alarga el ciclo de uso. A ello se suma la posibilidad de optimizar la cadena de suministro —primando proveedores nacionales cuando sea viable— y de adoptar prácticas responsables como el reacondicionamiento o reciclaje de microSD en renovaciones de flota.

En síntesis, el impacto del trabajo va más allá del logro técnico: socialmente, acerca la comunicación digital a quien más lo necesita con una interfaz amable y sin barreras; económicamente, consolida un modelo operativo escalable y sostenible que reduce costes directos e indirectos; y ambientalmente, fomenta un uso eficiente de recursos y una logística de soporte de baja huella. Para OHLA Ingesan y para VERA, esto se traduce en capacidad real de desplegar y mantener el servicio con calidad constante, en más hogares y durante más tiempo, alineando la misión social del proyecto con la viabilidad económica y el compromiso con el entorno.

8. Referencias

8.1 Recursos en línea

1. Organización Mundial de la Salud (OMS). "Envejecimiento y salud". Informes y proyecciones sobre envejecimiento poblacional https://www.who.int/news-room/fact- OMS
2. VERA (OHLA Ingesan) Vera
3. Naciones Unidas. Agenda 2030 y ODS (ODS 3: Salud y bienestar; principio de “no dejar a nadie atrás”) Naciones Unidas
4. WebRTC: Real-Time Communication in Browsers W3C
5. Jitsi. Jitsi
6. Getting started with the Zoom Web App**.** Soporte de Zoom
7. Learn what requirements you need to use Google Meet. Ayuda de Google
8. Microsoft Learn. Microsoft Learn
9. What browsers are supported for Skype for Web? Soporte de Microsoft
10. Anbox. GitHub
11. Hardware video decoding & encoding is not working after installing LineageOS. Foros Raspberry Pi
12. LineageOS 16.0 for Raspberry Pi 4 konstakang.com
13. Rpi camera support — Issue. GitHub
14. LineageOS 21 for Raspberry Pi 4. konstakang.com
15. How to use a Raspberry Pi in kiosk mode.raspberrypi.com
16. Lock task mode | Android Enterprise developer.android.com
17. How to use a Raspberry Pi in kiosk mode Raspberry PiRaspberry Pi Stack Exchange
18. Removing mouse on Bookworm similar to unclutter Foros Raspberry Pi
19. Kiosk mode quitting fullscreen after TV turned on — Issue GitHub

9. Anexos

9.1 Manual de instalación y configuración

1. Manual de conexión de la Raspberry Pi con Raspberry Pi Connect y Picockpit Link
2. Manual para Ocultar Ratón Link