

---

# **CityQuest: Turismo Inteligente Gamificado para el Futuro Sostenible de GreenLake Village**

---

**Asier Aurre<sup>1</sup>, Jaime Etxebarria<sup>2</sup>, Martin Fernandez de Retana<sup>3</sup>,**  
Lingfeng Chen<sup>4</sup>, David Fernandez<sup>5</sup>

<sup>1</sup>[asier.aurre@opendeusto.es](mailto:asier.aurre@opendeusto.es), <sup>2</sup>[jaime.etxebarria@opendeusto.es](mailto:jaime.etxebarria@opendeusto.es),

<sup>3</sup>[martin.f@opendeusto.es](mailto:martin.f@opendeusto.es) <sup>4</sup>[lingfeng.chen@opendeusto.es](mailto:lingfeng.chen@opendeusto.es),

<sup>5</sup>[david.f@opendeusto.es](mailto:david.f@opendeusto.es)

## **Abstract**

En un mundo donde la sostenibilidad y la experiencia personalizada definen el turismo moderno, presentamos CityQuest: una plataforma interactiva que transforma a GreenLake Village en un destino turístico inteligente y gamificado. Nuestra solución, impulsada por la Inteligencia Artificial (LLM), convierte cada visita en una aventura única a través de retos geolocalizados, recompensas y un asistente virtual de última generación. Al mismo tiempo, proporciona a las autoridades herramientas en tiempo real para una gestión turística eficiente, alineada con indicadores de sostenibilidad y métricas económicas. CityQuest no solo optimiza el turismo. Lo reinventa.

## **Contents**

<b>1 Introducción</b>	<b>3</b>
1.1 Contexto y problemática . . . . .	3
<b>2 Descripción de la Solución</b>	<b>6</b>
2.1 Concepto general y experiencia del usuario . . . . .	6
2.2 Funcionalidades principales . . . . .	7
2.2.1 Sistema de Retos y Recompensas . . . . .	7
2.2.2 Asistente Virtual Inteligente . . . . .	7
2.2.3 Dashboard de Administración y Análisis de Datos . . . . .	7
2.3 Casos de Uso y Escenarios Prácticos . . . . .	7
2.3.1 Turismo Inteligente desde la Administración Pública . . . . .	7
2.3.2 Desafío Entre Amigos: La Carrera por el Ranking . . . . .	8
2.3.3 Carlos y su Aventura Gastronómica . . . . .	8
<b>3 Arquitectura del Sistema</b>	<b>8</b>
3.1 Visión general de la arquitectura . . . . .	8
<b>4 Tecnologías Utilizadas</b>	<b>9</b>
4.1 Backend y servidores . . . . .	9
4.1.1 API en Flask . . . . .	9
4.1.2 Agente de IA con LiveKit . . . . .	9

4.2	Frontend y experiencia de usuario . . . . .	10
4.2.1	Aplicación Móvil . . . . .	10
4.2.2	Panel de Administración . . . . .	11
<b>5</b>	<b>Asistente Virtual con IA: Magellan</b>	<b>11</b>
5.1	Descripción del asistente y sus capacidades . . . . .	11
5.2	Implementación con LiveKit . . . . .	12
5.3	Multimodal y Voice Pipeline . . . . .	14
5.4	Ejecución de funciones en la API y conexión con dispositivos móviles . . . . .	15
5.4.1	LLM Callables. . . . .	15
5.4.2	Comunicación entre el agente y el dispositivo móvil. . . . .	15
<b>6</b>	<b>Gamificación y Sistema de Retos</b>	<b>16</b>
6.1	Mecanismo de Retos y Validaciones . . . . .	16
6.2	Ranking y Recompensas . . . . .	16
6.3	Interacción Social y Engagement . . . . .	16
<b>7</b>	<b>Administración y Gestión de Datos</b>	<b>16</b>
7.1	Base de Datos Relacional en Supabase . . . . .	16
7.2	Estructura de la Base de Datos . . . . .	17
<b>8</b>	<b>Video Demostrativo del Funcionamiento</b>	<b>18</b>
<b>9</b>	<b>Instrucciones de Instalación y Despliegue</b>	<b>18</b>
9.1	Repositorio . . . . .	18
9.2	Archivos .env y Variables de Entorno . . . . .	18
9.3	Backend y Panel de Administración . . . . .	18
9.3.1	Comandos para ejecutar el proyecto . . . . .	19
9.4	Aplicación Móvil . . . . .	19
9.4.1	Notas sobre la Aplicación . . . . .	20
<b>10</b>	<b>Demo reproducible</b>	<b>20</b>
10.1	1 - Creación de un nuevo local . . . . .	20
10.2	2 - Aprobación del local creado y creación de retos . . . . .	20
10.2.1	Aprobación de un local . . . . .	21
10.2.2	Gestión y creación de retos . . . . .	21
10.2.3	Gestión de premios . . . . .	21
10.2.4	Inicio de sesión en el dispositivo móvil . . . . .	21
10.2.5	Aceptar un desafío . . . . .	21
10.3	Hablar con el asistente . . . . .	22
<b>11</b>	<b>Bibliografía y Referencias</b>	<b>22</b>

# 1 Introducción

El turismo tiene un impacto directo en la economía local y en el medio ambiente. En los últimos años los beneficios y las consecuencias del turismo se halla en el centro del debate social y económico en España, pues se trata de uno de los sectores más importantes del país. Desde Overfitters Anónimos presentamos nuestra solución tecnológica CityQuest que impulsa mediante una colaboración entre la administración local y las diferentes entidades privadas locales un turismo de calidad y una economía.

En este documento técnico se estudia la evolución de diferentes entidades de los datos de muestra que se proporciona en cuanto al impacto económico y las consecuencias sobre el medio ambiente. A partir de esos datos hemos diseñado una serie de KPIs que identifica de forma cuantitativa del grado de compromiso con el medio ambiente y el impacto económico que genera cada entidad.

Se ofrece una solución tecnológica que se divide en dos partes, una dirigida a la administración local donde se ofrece una visualización directa a los puestos de responsabilidad los efectos que cada entidad tiene sobre la economía y la densidad de actividad de los turistas de forma que posibilita estrategias de planificación a largo plazo basado en datos y una respuesta eficiente a las diferentes situaciones. Y otra a los turistas dedicados a fomentar actividades culturales y el respeto con el medio ambiente.

## 1.1 Contexto y problemática

En un mundo cada vez mas conectado, el turismo resulta fundamental en la vida diaria de los habitantes de GreenLake, este fenómeno genera voces reacias que cada vez cobra mas relevancia en la sociedad. El termino del "Turismo destructivo" esta cada vez mas presente en los debates de los que habitan este lugar, a menudo se asocia directamente a daños medioambientales a demás de obstrucción de la vida y cultura local. Sin embargo también se alza posturas a favor de un turismo sano, basado en el respeto a la naturaleza , cultura y respeto de la localidad. Es por eso que en esta sección presentamos nuestro estudio de la problemática con el fin de dar un vistazo objetivo a la realidad ontológica que nos rodea, para así ofrecer una solución efectiva y eficaz.

Presentamos un nuevo KPI llamado "Índice verde", este indicador implica directamente cuanto de implicado está una entidad privada con el medio ambiente en un momento instante, tomando en cuenta su tamaño y factores dependientes a la economía de escala. Se computa de la siguiente manera:

$$I(t) = \frac{E(t) \cdot PR(t)}{\sum_{w_i \in W} w_i(t)}$$

donde:

- $E(t)$ : Energía consumida (en kWh) en el instante  $t$ .
- $PR(t)$ : Porcentaje de reciclaje medio en el instante  $t$ .
- $\sum_{w_i \in W} w_i(t)$ : Suma de todos los residuos generados en el instante  $t$ .

Este valor nos indica cuanta energía consume una empresa para su actividad, con respecto a los residuos que genera y el porcentaje de material reciclado que emplean, en un instante concreto. En caso de emplear mucha energía y generar pocos residuos en consecuencia implica que se utiliza energía renovable o verde, en este caso el índice aumenta. En caso contrario, el indice se reduce pues, no se están empleando un uso eficiente de la energía o se utiliza energía de fuentes contaminantes.

En la Figura 1 se puede observar como la distribución de los indices verdes están desviados a la izquierda lo que implica que una gran mayoría de establecimientos no son óptimos a nivel energético o no emplean energías renovables, estando por debajo de 0.7 el mas del 90% del área. Por lo que podemos afirmar con seguridad que existe una falta clara de compromiso con el medio ambiente entre los diferentes actores de la economía

No obstante, no basta con identificar el problema sino que también es importante identificar la periodicidad del problema a lo largo del tiempo sus características y los culpables por lo que mediante deducciones sencillas podemos determinar estos dos indicadores:

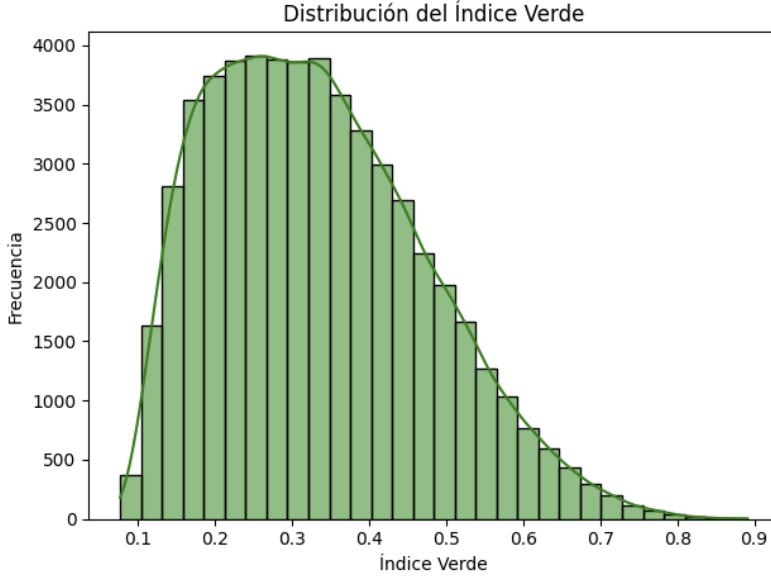


Figure 1: Distribución del índice verde

La derivada de  $I(t)$  respecto al tiempo  $t$  se obtiene aplicando la regla del cociente:

$$\frac{dI}{dt} = \frac{\left(\sum_{w_i \in W} w_i(t)\right) (E'(t) \cdot PR(t) + E(t) \cdot PR'(t)) - E(t) \cdot PR(t) \left(\sum_{w_i \in W} w'_i(t)\right)}{\left(\sum_{w_i \in W} w_i(t)\right)^2}$$

donde:

- $E'(t) = \frac{dE(t)}{dt}$
- $PR'(t) = \frac{dPR(t)}{dt}$
- $w'_i(t) = \frac{dw_i(t)}{dt}$

El resultado de la derivada es el pendiente que tiene este índice a lo largo del tiempo, que describe el nivel de compromiso que tiene la entidad con el medio ambiente, de ser positivo, eso quiere decir que está fomentando el uso eficiente de la energía o está fomentando el medio ambiente. De ser negativo implica que la empresa está reduciendo su compromiso con el medio ambiente, esto nos permite saber anomalías y malas prácticas en empresas.

Estudiando este indicador podemos saber sobre una línea de tiempo en qué épocas se genera una mayor y en qué época se genera una menor contaminación de cada empresa en particular, identificando los puntos donde  $\frac{dI}{dt} = 0$  y la periodicidad de estos, identificar el problema y tomar decisiones para solucionarlo.

La integral de  $I(t)$  desde  $t = a$  hasta  $t = B$  se expresa como:

$$\int_a^B I(t) dt = \int_a^B \frac{E(t) \cdot PR(t)}{\sum_{w_i \in W} w_i(t)} dt$$

Esta integral representa la acumulación del índice. Esto nos permite saber cuánto de responsable una entidad privada en relación a la contaminación acumulada en el presente. Por lo que es útil para identificar causas del problema y los culpables de esta, como se puede mostrar en la Figura 2 donde se muestra que hoteles han contribuido más a lo largo del tiempo con la contaminación existente.

Para determinar el impacto económico que tiene una compañía en el sector del turismo hemos desarrollado un índice determinado de forma relativa entre entidades la importancia que tiene cada una. Figura 3. Se calcula de la siguiente manera:

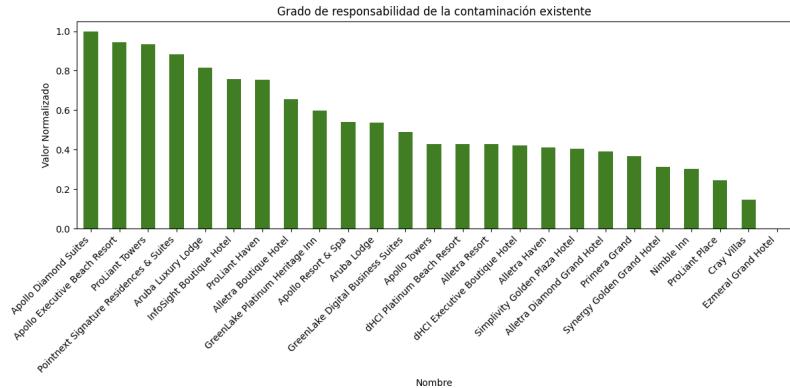


Figure 2: Distribución de responsabilidades

Definimos las siguientes variables:

- $R(t)$  = Ratio de Ingreso en el instante  $t$ ,
- $r_i(t)$  = Ingreso esperado de la fuente  $i$ ,
- $\theta_i(t)$  = Ingreso confirmado de la fuente  $i$ ,
- $c(t)$  = Cantidad de fuente de ingresos esperados cancelados,
- $p(t)$  = Precio Promedio por Noche,
- $I$  = Conjunto de fuentes de ingreso.

La fórmula se expresa como:

$$R(t) = \frac{\sum_{i \in I} (r_i(t) + \theta_i(t))}{c(t)} p(t).$$

Sea

$$F(t) = \sum_{i \in I} (r_i(t) + \theta_i(t)),$$

de modo que

$$R(t) = \frac{F(t)}{c(t)} p(t).$$

Para obtener la derivada de  $R(t)$  con respecto a  $t$ , se aplica la regla del producto y la del cociente. Con esto podemos estudiar como evoluciona los ingresos de la compañía y su impacto en la economía con respecto a años anteriores:

$$\frac{dR}{dt} = \frac{d}{dt} \left( \frac{F(t)}{c(t)} \right) p(t) + \frac{F(t)}{c(t)} p'(t),$$

donde

$$\frac{d}{dt} \left( \frac{F(t)}{c(t)} \right) = \frac{F'(t) c(t) - F(t) c'(t)}{[c(t)]^2},$$

y

$$F'(t) = \sum_{i \in I} (r'_i(t) + \theta'_i(t)).$$

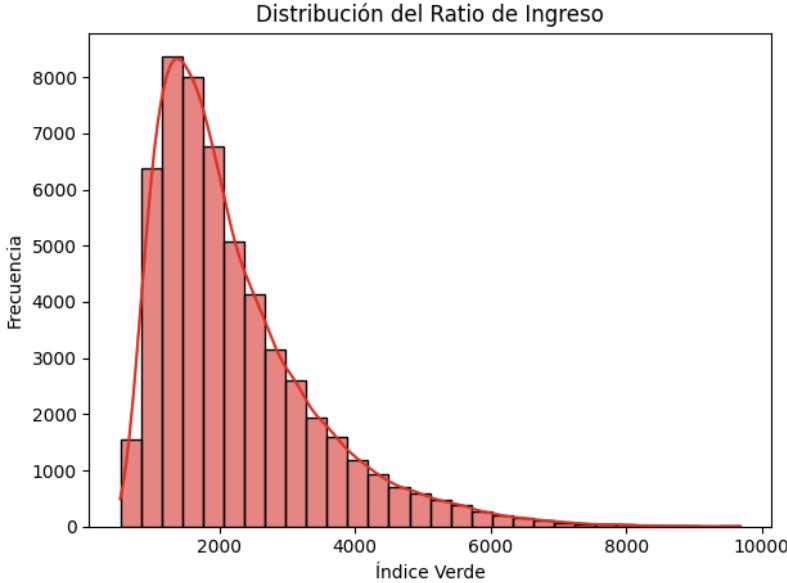


Figure 3: Distribución de responsabilidades

Por lo tanto, la derivada completa es:

$$\frac{dR}{dt} = \frac{F'(t)c(t) - F(t)c'(t)}{[c(t)]^2} p(t) + \frac{F(t)}{c(t)} p'(t).$$

La integral indefinida de  $R(t)$  respecto a  $t$  es, una vez calculada. Esto nos permite calcular el impacto generado por una compañía durante un periodo, es decir su impacto económico:

$$\int_a^b R(t) dt = \int_a^b \frac{F(t)}{c(t)} p(t) dt.$$

Tras el análisis detallado de los indicadores y la evolución temporal de los mismos, se evidencia de forma clara la necesidad de implementar medidas proactivas que impulse tanto la eficiencia energética como el impacto económico positivo. Estos resultados cuantitativos no solo permiten identificar áreas de mejora en el compromiso medioambiental y en la rentabilidad de las entidades, sino que también sientan las bases para desarrollar soluciones innovadoras que integren la gestión de recursos y la planificación estratégica. Con este panorama en mente, presentamos a continuación una propuesta integral que une tecnología, análisis de datos y experiencia de usuario para transformar y optimizar la interacción entre el turismo, la economía local y el medio ambiente.

## 2 Descripción de la Solución

### 2.1 Concepto general y experiencia del usuario

El sistema propuesto para GreenLake Village está diseñado para revolucionar la experiencia turística mediante la combinación de gamificación, inteligencia artificial y análisis de datos en tiempo real. Nuestra solución es una plataforma digital interactiva que incentiva a los visitantes a explorar la ciudad a través de desafíos geolocalizados, recompensas y asistencia personalizada basada en IA.

Los turistas accederán a la plataforma a través de una aplicación móvil intuitiva, donde podrán visualizar retos disponibles en diferentes categorías como gastronomía, cultura, naturaleza y

movilidad sostenible. La interacción con estos desafíos no solo permitirá a los usuarios descubrir GreenLake Village de manera dinámica, sino que también fomentará la participación en la economía local y la reducción del impacto ambiental mediante incentivos específicos establecidos por la administración en base a las problemáticas analizadas.

## 2.2 Funcionalidades principales

El sistema está compuesto por tres elementos clave:

### 2.2.1 Sistema de Retos y Recompensas

- Los visitantes podrán elegir entre múltiples retos que les llevarán a explorar la ciudad de forma interactiva.
- Los desafíos pueden completarse mediante diversas formas en función del tipo de desafío:
  - Escaneando un QR en ubicaciones clave.
  - Subiendo una foto de algo concreto.
  - Confirmando presencia mediante GPS en uno o múltiples puntos a lo largo de una ruta.
  - Realizando consumos en establecimientos locales (QR).
- Al completar retos, los usuarios ganan puntos que pueden ser usados para:
  - Obtener descuentos.
  - Acceder a experiencias exclusivas.
  - Participar en sorteos diarios.

### 2.2.2 Asistente Virtual Inteligente

- El asistente de IA actúa como un guía interactivo que:
  - Responde preguntas.
  - Sugiere retos y rutas
  - Proporciona información sobre la historia y cultura local.
  - Ofrece soporte y ayuda al usuario
- Además, el agente tiene la capacidad de ejecutar funciones en la API de forma independiente al usuario y activar acciones en el dispositivo móvil, tales como:
  - Mostrar mapas interactivos.
  - Analizar el entorno, incluyendo edificios y puntos de interés, mediante la cámara.
  - Obtener la ubicación del usuario para mejorar recomendaciones, asistencia y otros servicios.
  - Presentar datos de interés en tiempo real.

### 2.2.3 Dashboard de Administración y Análisis de Datos

- Proporciona a la administración pública herramientas avanzadas para monitorizar el turismo en tiempo real.
- Incluye métricas sobre:
  - Afluencia de visitantes.
  - Impacto económico en comercios locales.
  - Retos más populares.
  - Análisis predictivo de la demanda turística.

## 2.3 Casos de Uso y Escenarios Prácticos

### 2.3.1 Turismo Inteligente desde la Administración Pública

Desde el panel de control, el equipo de GreenLake observa que el centro de la ciudad está abarrotado de turistas mientras que algunas zonas culturales menos visitadas permanecen vacías. En

cuestión de segundos, activan un reto especial que otorga puntos extra a quienes exploren estos lugares olvidados.

Las notificaciones llegan a los dispositivos de los visitantes en tiempo real. Algunos deciden aceptar el desafío y se aventuran a descubrir rincones ocultos de la ciudad, conociendo pequeños cafés, museos alternativos y parques tranquilos. Horas más tarde, el equipo revisa las métricas: la distribución del turismo ha mejorado y los comercios locales han recibido un aumento en clientes.

### 2.3.2 Desafío Entre Amigos: La Carrera por el Ranking

Un grupo de cinco amigos llega a GreenLake y, al abrir la aplicación, descubre el ranking diario de exploradores. Otro equipo de turistas ya ha acumulado una gran cantidad de puntos. La competencia comienza.

Para maximizar su puntuación, se dividen estratégicamente: unos siguen una ruta cultural explorando monumentos históricos, mientras que otros aceptan desafíos gastronómicos probando platillos típicos. Uno de ellos, apasionado por la fotografía, se embarca en un reto de capturar imágenes creativas de la ciudad.

La aplicación actualiza el ranking en tiempo real. La diferencia de puntos entre los dos equipos se reduce conforme avanza el día. Cuando cae la noche, los amigos consiguen superar a sus rivales y alcanzan la primera posición. Como recompensa, desbloquean un premio exclusivo y celebran su victoria en un restaurante recomendado por la propia app.

### 2.3.3 Carlos y su Aventura Gastronómica

Carlos aterriza en GreenLake con una sola pregunta en mente: “¿Dónde puedo probar la mejor comida local?” Abre la aplicación y el asistente de IA le responde con una propuesta irresistible: un reto gastronómico personalizado.

Su misión es sencilla pero deliciosa: visitar tres restaurantes seleccionados según sus gustos, degustar un plato en cada uno y escanear los códigos QR proporcionados. Motivado por la promesa de una experiencia única, Carlos acepta el desafío.

A medida que avanza por la ciudad, descubre sabores y texturas que nunca había probado. Cuando completa el reto, la aplicación le otorga puntos adicionales y le sorprende con un premio inesperado: una clase de cocina privada con un chef local. Su viaje a GreenLake se ha convertido en una experiencia gastronómica inmersiva, todo gracias a la gamificación.

## 3 Arquitectura del Sistema

### 3.1 Visión general de la arquitectura

El sistema de CityQuest está diseñado con una arquitectura modular que integra diversos servicios y tecnologías. Se compone de una aplicación móvil desarrollada en *React Native* con *Expo*, que permite a los usuarios interactuar con los retos y el asistente virtual. Para la administración y gestión de datos, se ha desarrollado un dashboard web en *React* con *Vite*, proporcionando a los administradores un entorno ágil y eficiente para la supervisión del sistema.

En el **backend**, la API principal está construida en *Python* con *Flask*, actuando como el núcleo del sistema al gestionar todas las interacciones con el agente y las solicitudes de los usuarios. Además, se ha implementado un **Agente de IA**, también en *Python*, que facilita la comunicación con *LiveKit* y permite la ejecución de comandos en la aplicación móvil, brindando capacidades avanzadas de interacción con el asistente virtual.

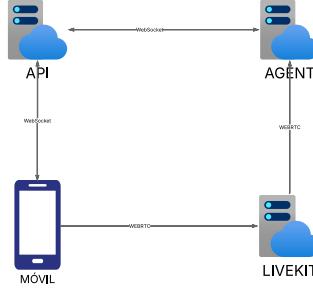


Figure 4: Diagrama de arquitectura de conexión de los servidores

Para la gestión de datos, **Supabase** funciona como la base de datos y sistema de autenticación, asegurando un almacenamiento seguro y escalable. La comunicación en tiempo real entre los distintos componentes se realiza a través de *WebSockets*, permitiendo una sincronización fluida de eventos, como la actualización de retos en la aplicación. Asimismo, **LiveKit** actúa como la infraestructura de comunicación en tiempo real, permitiendo que el asistente de IA pueda procesar comandos y responder de manera eficiente. En términos de inteligencia artificial, los modelos de procesamiento de lenguaje y voz se ejecutan en servidores de **OpenAI** y **Azure**, garantizando un rendimiento óptimo en la generación de respuestas y análisis de contenido.

Toda la infraestructura **backend**, incluyendo la API y el Agente de IA, ha sido empaquetada en *Docker* para facilitar el despliegue y escalabilidad. Estos servicios han sido desplegados en **Oracle Cloud**, aunque se ha decidido no utilizar los servicios en la nube durante la evaluación para garantizar que el sistema funcione correctamente en el momento de la prueba. Por otro lado, la aplicación móvil se distribuye como un *APK* descargable, sin necesidad de contenedorización, permitiendo a los usuarios acceder a todas las funcionalidades sin dependencias adicionales.

En conjunto, esta arquitectura modular y escalable permite una integración fluida entre los diferentes servicios, optimizando la experiencia tanto para los usuarios finales como para los administradores del sistema.

## 4 Tecnologías Utilizadas

### 4.1 Backend y servidores

El sistema se compone de múltiples servidores que trabajan en conjunto para garantizar una experiencia fluida y en tiempo real. A continuación, se detallan sus responsabilidades:

#### 4.1.1 API en Flask

El servidor desarrollado en **Flask** actúa como el núcleo del sistema, gestionando múltiples funcionalidades clave:

- Generación de tokens de autenticación para establecer sesiones seguras entre los agentes y los clientes.
- Mantenimiento de la conexión WebSocket con el agente de IA y la aplicación móvil, permitiendo una comunicación bidireccional en tiempo real.
- Generación de rutas personalizadas basadas en la ubicación y preferencias del usuario.
- Comunicación con un modelo de lenguaje (LLM) para analizar y describir el contenido de una imagen proporcionada por el usuario.
- Obtención de la dirección del usuario a partir de su ubicación.

#### 4.1.2 Agente de IA con LiveKit

El **Agent** es un servidor que se comunica con los servicios de **LiveKit** y gestiona la interacción con modelos de inteligencia artificial. Su función principal es la de actuar como intermediario entre el

usuario y los servicios de IA, permitiendo una experiencia multimodal en tiempo real. Algunas de sus características clave incluyen:

- Comunicación en tiempo real con la API y la aplicación móvil a través de WebRTC y WebSockets.
- Orquestación de modelos de inteligencia artificial para proporcionar respuestas personalizadas, ejecutar comandos y asistir en la navegación de la aplicación.
- Mantenimiento de sesiones de usuario activas, garantizando interacciones fluidas y contextuales.

## 4.2 Frontend y experiencia de usuario

### 4.2.1 Aplicación Móvil



Figure 5: Interfaz de aplicación con diferentes funciones de demostración

La aplicación móvil ha sido diseñada para proporcionar una experiencia fluida e intuitiva para los usuarios que visitan GreenLake Village. Su interfaz se basa en una navegación sencilla con un enfoque en la exploración de retos, la interacción social y la competencia gamificada. A continuación, se describen las principales pantallas:

**Explorar** La pantalla principal de la aplicación, presenta un mapa interactivo donde los usuarios pueden visualizar todos los retos activos representados por *pinpoints* en el mapa. Además, cuenta con un desplegable en la parte inferior que permite ver los desafíos en un formato de lista para facilitar la búsqueda y selección.

Para mejorar la accesibilidad y asistencia al usuario, durante toda la navegación en la aplicación se encuentra disponible un botón flotante azul que permite activar el asistente virtual mediante una pulsación prolongada. Este asistente facilita la orientación y brinda información relevante en tiempo real.

**Ranking** La pantalla de **Ranking** permite a los usuarios visualizar su posición en la clasificación general y compararse con otros participantes en tiempo real. En esta vista, se destaca la competitividad del sistema gamificado, motivando a los visitantes a completar más retos y escalar posiciones.

Además, incluye una sección de “**Ver premios**”, donde los jugadores pueden consultar las recompensas disponibles en función de los puntos acumulados.

**Feed** El Feed es el espacio donde se recopilan los retos completados por todos los usuarios, permitiendo visualizar imágenes y logros obtenidos. Esta funcionalidad fomenta la interacción social, ya que los participantes pueden ver las experiencias de otros viajeros, lo que incentiva la exploración y la participación activa en la comunidad del juego.

#### 4.2.2 Panel de Administración

Desde la administración pública existiría un selecto grupo de personas las cuales podrían acceder a el panel de administración donde se recopilarían todos los datos referentes a la aplicación en tiempo real y desde el cual se podría administrar cualquier tipo de cambio en cuanto a retos, locales y premios se refiere.

Para cumplir su prometido, el dashboard cuenta con una pestaña donde se puede ver los datos en diferentes tipos de gráficos para poder comprender la situación actual y así poder hacer los cambios necesarios.

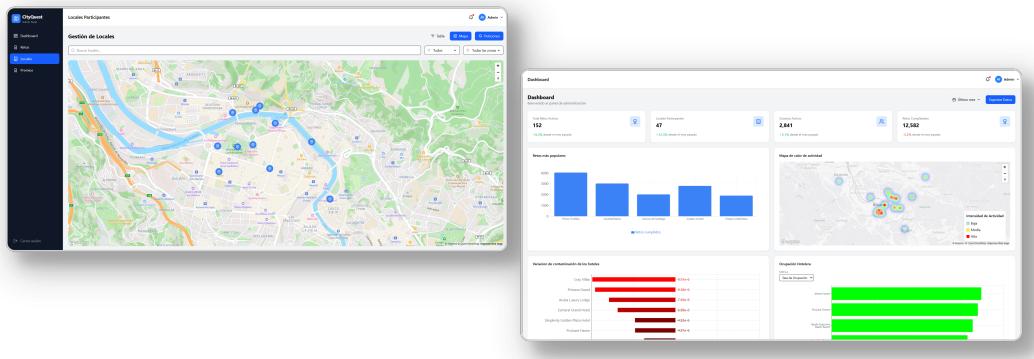


Figure 6: Pestaña de la administración pública con datos

## 5 Asistente Virtual con IA: Magellan

### 5.1 Descripción del asistente y sus capacidades

En el corazón de la experiencia CityQuest se encuentra un asistente virtual inteligente, diseñado para acompañar y potenciar la aventura de cada visitante. Más que una simple herramienta, este asistente actúa como un verdadero compañero de viaje, capaz de resolver dudas, sugerir rutas y recomendar desafíos adaptados a los intereses del usuario.

Hemos bautizado a este asistente como **Magellan**, en honor a *Fernando de Magallanes*, el explorador que lideró la primera circunnavegación del mundo. Al igual que él, Magellan está diseñado para guiar a los usuarios en su viaje, ofreciendo orientación contextualizada, recomendaciones personalizadas e información relevante en tiempo real.

Magellan es accesible tanto por voz como por chat, y puede activarse desde cualquier ventana de la aplicación mediante un botón flotante. Gracias a su integración profunda con el sistema, proporciona una experiencia de exploración fluida, interactiva y enriquecida.

Entre sus principales capacidades se incluyen:

- **Información turística interactiva:** El asistente tiene acceso a una base de datos actualizada sobre los puntos de interés de GreenLake Village, tales como monumentos, museos, parques y rutas turísticas. Los usuarios pueden solicitar información específica sobre cualquier lugar, y el asistente les ofrecerá descripciones detalladas, horarios, precios y recomendaciones adicionales.
- **Guía virtual de audio:** Cuando el usuario se encuentra en una ubicación específica, el asistente puede ofrecer explicaciones en formato de audio sobre el lugar, proporcionando información histórica y cultural de manera envolvente y educativa. Esta función se activará mediante el uso de la cámara del dispositivo, permitiendo que el asistente reconozca el lugar y brinde contenido contextual.
- **Cálculo y optimización de rutas:** El asistente virtual ayuda a los usuarios a planificar sus recorridos por la ciudad. Basándose en las preferencias del usuario y en los retos

que desee completar, el asistente sugiere las rutas más eficientes y personalizadas, optimizando el tiempo y la experiencia del recorrido. Además, podrá indicar el mejor uso del transporte público según la ubicación del usuario.

- **Seguimiento de retos y ranking:** El asistente ofrece un seguimiento en tiempo real de los retos en los que el usuario está participando, mostrando el progreso y proporcionando sugerencias de nuevos retos para completar. Además, mantiene a los usuarios informados sobre su posición en el ranking diario, motivándolos a continuar participando activamente.
- **Interacción y soporte en tiempo real:** El asistente actúa como un canal de comunicación para resolver dudas o problemas que los usuarios puedan tener durante su experiencia en GreenLake Village. Desde recomendaciones personalizadas hasta soporte en caso de emergencias, el asistente está disponible para guiar a los usuarios en cualquier momento del día.

## 5.2 Implementación con LiveKit

LiveKit es una plataforma de comunicación en tiempo real que permite integrar video, audio y datos en aplicaciones. Ofrece herramientas para crear experiencias de videollamadas, conferencias y colaboración en vivo. En este caso, hemos empleado LiveKit para conectar a los usuarios con nuestros AI Voice Agents, un sistema similar al utilizado por ChatGPT o plataformas como Character.ai.

La arquitectura de comunicación con LiveKit se representa en la Figura 7. En este esquema, los clientes (dispositivo móvil) establecen una conexión con LiveKit Cloud mediante WebRTC. A su vez, nuestro backend, que contiene los agentes de voz, se comunica con LiveKit utilizando WebRTC y recibe las transmisiones de audio del usuario. Luego, el audio es procesado mediante diferentes modelos de IA:

- **STT (Speech-to-Text):** Convierte la voz del usuario en texto.
- **LLM (Large Language Model):** Procesa el texto y genera una respuesta adecuada.
- **TTS (Text-to-Speech):** Convierte la respuesta del LLM nuevamente en audio.
- **Modelo multimodal:** Permite interacciones enriquecidas con imágenes y otros medios.

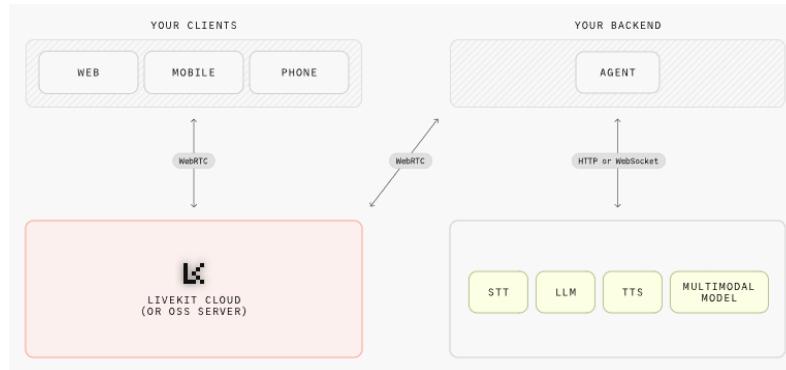


Figure 7: Arquitectura de comunicación con LiveKit

A continuación, se describe el flujo de interacción entre la aplicación móvil, el servidor y el servicio de LiveKit, siguiendo una secuencia de eventos que permiten la creación, el manejo y la finalización de una sesión de comunicación en vivo. Este proceso se ilustra visualmente en la Figura 8, que presenta un diagrama de secuencia que muestra cómo interactúan los diferentes componentes durante la transmisión en tiempo real.

1. **Creación de sesión:** La aplicación móvil solicita un token de sesión al servidor API, que lo devuelve para autenticar la comunicación.

2. **Inicialización:** La aplicación crea una nueva sesión, y el servidor responde con un ID de sesión y credenciales de LiveKit.
3. **Preparación de streaming:** La aplicación inicia el streaming y recibe confirmación cuando está listo.
4. **Conexión:** La aplicación se conecta a la sala en LiveKit y comienza a recibir video.
5. **Interacción:** Durante la comunicación, el usuario envía texto desde la aplicación al servidor, que genera respuestas de voz y actualiza el stream a través de LiveKit. Esta interacción ocurre en un ciclo continuo.
6. **Finalización:** Cuando el usuario termina, la aplicación cierra la sesión, el servidor confirma el cierre y se desconecta de LiveKit.

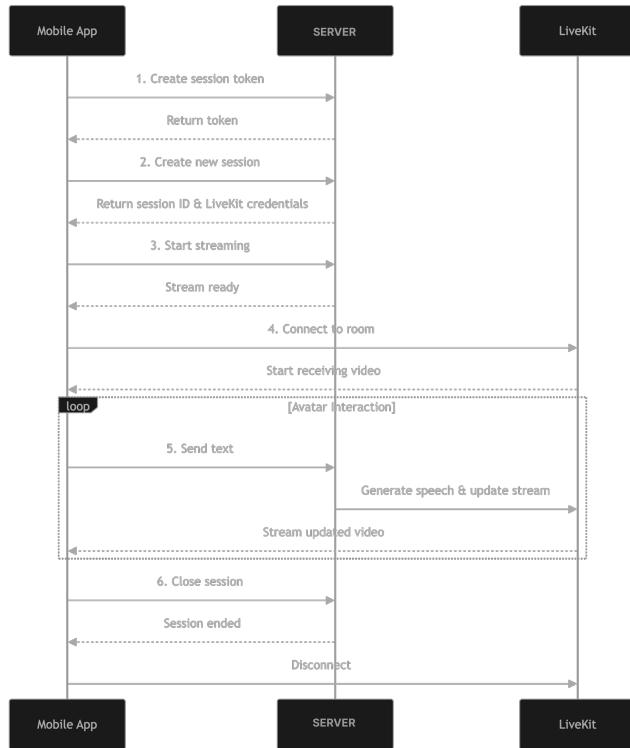


Figure 8: Diagrama de secuencia entre el usuario , el servidor y livekit

A continuación, se describe el flujo de trabajo entre el usuario y el agente en el contexto de LiveKit en la Figura 9.

1. **Registro del agente:** El agente de voz se registra en el servidor de LiveKit para estar disponible en futuras sesiones.
2. **Inicio de sesión del usuario:** Un usuario inicia una sesión en la aplicación y solicita asistencia de un agente.
3. **Asignación de trabajo:** El sistema asigna el trabajo al primer agente disponible.
4. **Conexión a la sala:** El agente se une a la sala de LiveKit junto con el usuario, estableciendo la comunicación en tiempo real.

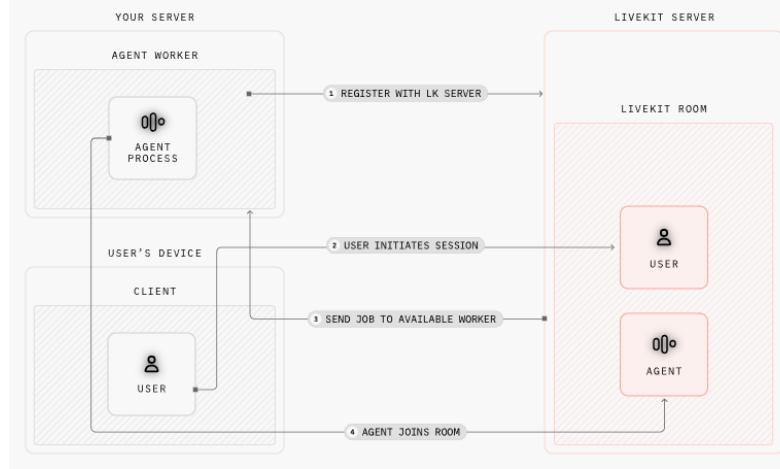


Figure 9: Flujo de trabajo entre usuario y agente en LiveKit

Gracias a este sistema, la interacción con los agentes de voz es rápida y fluida, permitiendo respuestas en tiempo real con baja latencia y una experiencia de usuario natural.

### 5.3 Multimodal y Voice Pipeline

LiveKit ofrece dos tipos de agentes de voz: `MultimodalAgent` y `VoicePipelineAgent`.

- `MultimodalAgent` utiliza el modelo multimodal de OpenAI y su API en tiempo real para procesar directamente el audio del usuario y generar respuestas en audio, similar al modo de voz avanzado de OpenAI, produciendo un habla más natural.
- `VoicePipelineAgent` utiliza una cadena de modelos STT, LLM y TTS, proporcionando mayor control sobre el flujo de la conversación, ya que permite modificar el texto devuelto por el LLM.

Característica	Multimodal	Voice Pipeline
<b>Tipo de Modelo</b>	single multimodal	STT, LLM, TTS
<b>Llamadas a funciones</b>	Si	Si
<b>RAG</b>	vía llamada a función	Si
<b>Habla natural</b>	más natural	
<b>Modificar respuesta del LLM</b>		Si
<b>Proveedores de modelo</b>	OpenAI	varios
<b>Detección de turnos</b>	VAD	VAD y modelo de detección de turnos

Table 1: Comparación entre `MultimodalAgent` y `VoicePipelineAgent`

Hemos implementado ambos modelos y, en el caso de `VoicePipelineAgent`, hemos utilizado Whisper para el reconocimiento de voz, GPT-4 para el procesamiento del lenguaje y Azure TTS para la síntesis de voz. Las conclusiones de esta comparación se presentan en la Tabla 2.

Tras evaluar ambas soluciones, hemos optado por utilizar `MultimodalAgent` debido a su notable rapidez en la generación de respuestas en tiempo real. Aunque `VoicePipelineAgent` ofrece mayor personalización en cuanto a modelos y síntesis de voz, la latencia introducida por el procesamiento en múltiples etapas lo hace menos adecuado para una experiencia fluida e inmediata.

Aunque hemos decidido utilizar `MultimodalAgent` por su velocidad, hemos mantenido ambas implementaciones. Esto permite cambiar entre ellas fácilmente ejecutando `multimodal.py` o `pipeline.py`, según las necesidades del proyecto.

Criterio	MultimodalAgent	VoicePipelineAgent
Velocidad de respuesta	Muy alta (en tiempo real)	Media (dependiente de modelos)
Facilidad de implementación	Alta (uso directo de OpenAI)	Media (requiere integración de modelos)
Personalización de la voz	Baja (voices predefinidas)	Alta (elige STT, LLM y TTS)
Control sobre conversación	Media (modificar no permitido, insertar sí)	Alto (modificar respuestas)
Consumo de recursos	Alto	Medio
Flexibilidad de integración	Media (depende de OpenAI)	Alta (adapta a diferentes proveedores)

Table 2: Conclusiones obtenidas tras comparar ambos modelos MultimodalAgent y VoicePipelineAgent

## 5.4 Ejecución de funciones en la API y conexión con dispositivos móviles

### 5.4.1 LLM Callables.

El agente puede ejecutar funciones en nuestro servidor mediante el uso de *LLM callables*. La implementación de esto se realiza de la siguiente manera: le describimos al agente en qué casos debe usar una función y cuál es su retorno.

El agente, ahora utilizando el contexto de la conversación, puede decidir si ejecutar funciones como `calculate_route`, `open_camera`, `ask_challenges` o `get_user_location`.

Esta implementación puede verse en la sección `Back/AGENT/api.py` del código.

### 5.4.2 Comunicación entre el agente y el dispositivo móvil.

Hemos implementado un sistema para la comunicación bidireccional entre el agente y el dispositivo móvil mediante el uso de *websockets*, usando el servidor API como intermediario. La implementación es la siguiente: cuando el usuario activa el asistente, además de las conexiones requeridas para la comunicación con LiveKit, el dispositivo móvil establece una nueva conexión *websocket* con el servidor API. El agente, de la misma manera, también establece una conexión. Se gestiona una *group session* para ambos usuarios, y a partir de este momento, el móvil podrá enviar y recibir datos del agente. Algunos de los ejemplos casos de uso que hemos implementado son los siguientes:

**Caso de uso 1:** El usuario pide al asistente una recomendación de rutas. El asistente analiza la situación y concluye que debe saber la localización del usuario, envía un mensaje por *websocket* con el comando `get_user_location` para obtener la dirección actual del usuario. El móvil recibe la petición por *websockets*, emite un nuevo *websocket* con la localización GPS y la API procesa esas coordenadas para obtener una dirección. Esta dirección es reenviada al agente nuevamente mediante la conexión *websocket*. Posteriormente, el agente ejecutará la función `ask_challenges` para obtener la información de los retos actuales de la base de datos. Cuando el usuario confirme que desea realizar un reto, el agente puede llamar a la función `accept_challenge` para aceptar el reto en nombre del usuario.

**Caso de uso 2:** El usuario pregunta por un edificio, monumento o sitio histórico que tiene frente a él. El asistente entiende que debe reconocer el lugar y ejecuta la función `open_camera`, que abrirá un modal con la cámara para el usuario e indicará que apunte hacia lo que está señalando. Tras esto, se tomará una imagen que el móvil enviará de nuevo a la API en formato *base64*. La API gestionará el procesamiento de la descripción del lugar y devolverá la información al agente a través de la conexión *websocket*. Finalmente, el agente insertará esta información en forma de mensaje `system`.

**Caso de uso 3:** El usuario solicita al asistente que calcule una ruta optimizada para completar el máximo número de retos antes de abandonar el área. El asistente analiza la petición y determina que debe calcular una ruta eficiente. Para ello, envía un mensaje por *websocket* con el comando `calculate_route`.

La API procesa la solicitud evaluando los retos disponibles en función de la ubicación del usuario y genera una ruta óptima. Una vez calculada, la API devuelve la ruta tanto al usuario como al asistente mediante la conexión *websocket*.

Para mostrar la ruta al usuario, se ejecuta la función `show_route`, lo que abre la vista del mapa con la ruta optimizada. Simultáneamente, el agente recibe la información de la ruta y la explica al usuario, detallando los retos incluidos en el trayecto y las mejores estrategias para completarlos.

## 6 Gamificación y Sistema de Retos

El sistema de gamificación es una de las piezas clave de nuestra aplicación, diseñado para incentivar la participación del usuario mediante retos, rankings y recompensas. La gamificación no solo motiva a los jugadores a explorar y completar desafíos, sino que también fomenta la interacción social y la competitividad sana.

### 6.1 Mecanismo de Retos y Validaciones

Cuando un usuario acepta un reto, este se registra en la base de datos en estado "pendiente". Una vez que cumple con los requisitos, la aplicación valida la acción y actualiza el estado a "completado". En el caso de los retos de GPS, la aplicación puede hacer uso de validaciones adicionales para evitar fraudes.

### 6.2 Ranking y Recompensas

Para fomentar la competitividad, los usuarios participan en un sistema de **ranking**, donde se acumulan puntos en función de los retos completados y la dificultad de los mismos. Los elementos clave de este sistema incluyen:

- **Puntuación:** Cada reto completado otorga una cantidad determinada de puntos.
- **Ranking Global y de Amigos:** Los usuarios pueden ver su posición en un ranking global o entre sus amigos.
- **Recompensas Virtuales:** Al alcanzar ciertos hitos, los usuarios pueden obtener premios en forma de insignias, monedas virtuales u objetos canjeables.

Los usuarios pueden canjear sus puntos por **premios** dentro de la aplicación, fomentando así la participación activa y el deseo de completar más retos.

### 6.3 Interacción Social y Engagement

La experiencia de usuario se potencia mediante elementos de interacción social, tales como:

- **Grupos:** Los jugadores pueden formar equipos para competir y compartir sus progresos.
- **Retos en Grupo:** Existen desafíos colaborativos donde múltiples usuarios deben contribuir para completar una tarea en conjunto.
- **Tablón de Actividad:** Un feed donde se muestran logros recientes de los usuarios, incentivando la participación.

Estos mecanismos refuerzan el compromiso del usuario y convierten la experiencia en algo más inmersivo y atractivo, promoviendo el uso continuo de la aplicación.

## 7 Administración y Gestión de Datos

Para la gestión y almacenamiento de datos, utilizamos **Supabase**, una plataforma basada en PostgreSQL que nos permite administrar una base de datos SQL con una integración nativa de autenticación, almacenamiento y sincronización en tiempo real.

### 7.1 Base de Datos Relacional en Supabase

Supabase se basa en PostgreSQL, una base de datos relacional que organiza los datos en tablas con relaciones bien definidas. En nuestro sistema, cada entidad clave (usuarios, retos, ubicaciones, etc.) se representa como una tabla, permitiendo consultas eficientes y escalabilidad.

Las principales características de nuestra base de datos son:

- **Almacenamiento SQL:** Todas las entidades del sistema se guardan en un esquema estructurado y relacional, garantizando integridad y consistencia.
- **Autenticación Nativa:** Utilizamos el sistema de autenticación de Supabase para gestionar usuarios de forma segura.
- **Almacenamiento de Imágenes:** Supabase Storage nos permite guardar imágenes de usuarios, retos y locales.
- **Sincronización en Tiempo Real:** Gracias al sistema de *realtime* de Supabase, los retos y otros elementos se sincronizan instantáneamente en la aplicación.

## 7.2 Estructura de la Base de Datos

Nuestra base de datos está compuesta por múltiples tablas interrelacionadas, cada una con un propósito específico. A continuación, se presenta el esquema general de la base de datos:

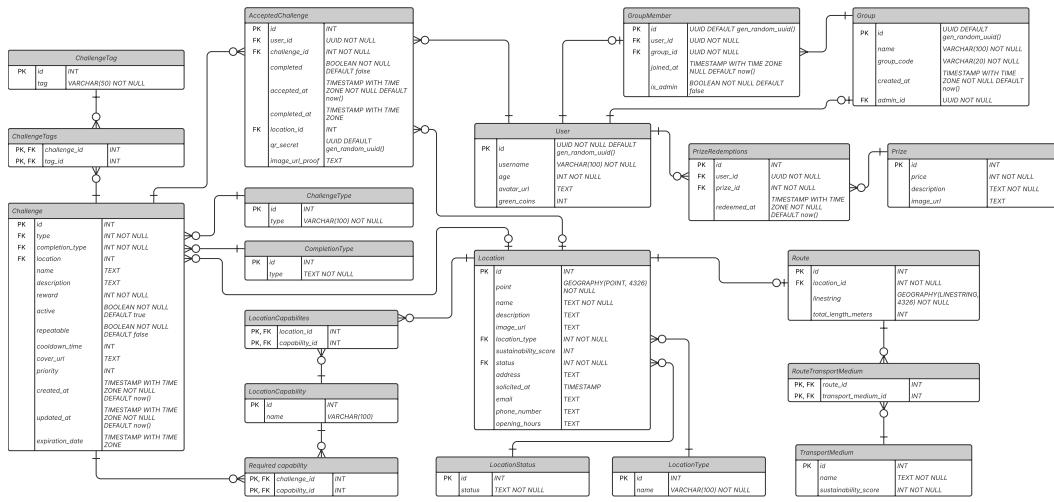


Figure 10: Esquema de la Base de Datos en Supabase

Las principales tablas y sus funciones son:

- **User:** Almacena la información de los usuarios, como nombre, avatar y monedas acumuladas.
- **Challenge:** Define los retos disponibles, su tipo de completado (QR, GPS, foto, etc.), descripción y condiciones.
- **Location:** Contiene información sobre las localizaciones,(locales y puntos de interés en los que se pueden completar retos).
- **AcceptedChallenge:** Registra los retos aceptados por los usuarios, junto con su estado de completado.
- **Group y GroupMember:** Gestionan grupos de usuarios y sus integrantes.
- **Prize:** Almacena los premios que los usuarios pueden obtener al completar retos.
- **Route:** Define rutas preestablecidas para completar retos basados en GPS.
- **TransportMedium:** Especifica medios de transporte que pueden ser utilizados en una ruta.
- **ChallengeTag y ChallengeType:** Permiten clasificar los retos según su tipo o etiquetas asignadas.

- **LocationType** y **LocationStatus**: Gestionan la categorización y estado de las ubicaciones registradas.

Esta estructura nos permite gestionar eficientemente las interacciones dentro de la aplicación, asegurando un flujo de datos ordenado y una experiencia de usuario fluida.

## 8 Video Demostrativo del Funcionamiento

A continuación se proporciona un enlace a la demostración en video de la solución CityQuest, en la que se muestra el funcionamiento general del sistema, la experiencia del usuario y las algunas de las funcionalidades implementadas (no todas):

- **Enlace al video:** <https://www.youtube.com/watch?v=ojqASRLULw8>

## 9 Intrucciones de Instalación y Despliegue

### 9.1 Repositorio

El código de la implementación de los dos servidores, el panel de administración y la aplicación móvil, así como los cuadernos de análisis de los datos proporcionados, se encuentran disponibles en el repositorio de GitHub: [https://github.com/hjasier/cds\\_app](https://github.com/hjasier/cds_app).

Puedes clonar el repositorio ejecutando el siguiente comando en tu terminal:

```
git clone https://github.com/hjasier/cds_app.git
```

La estructura del repositorio es la siguiente:

**Analysis** Cuadernos con los análisis de los datos.

**Back** Servidor de agente y API.

**Front** Aplicación móvil y panel de administración (dashboard).

### 9.2 Archivos .env y Variables de Entorno

Es importante destacar que los archivos `.env` que están presentes en el repositorio están vacíos. Para poder completar la configuración y que la aplicación funcione correctamente, deberás reemplazar estos archivos vacíos con los archivos `.env` correspondientes que hemos proporcionado en el archivo comprimido `ENVIRONMENT_VARIABLES.zip`.

Dentro del archivo zip encontrarás cuatro carpetas, cada una correspondiente a una parte diferente del sistema:

- **AdminDashboard**: Contiene el archivo `.env` que tendrás que introducir en `~/Front/AdminDashboard`.
- **ClientApp**: Contiene el archivo `.env` que tendrás que introducir en `~/Front/ClientApp`.
- **AGENT**: Contiene el archivo `.env` que tendrás que introducir en `~/Back/AGENT`.
- **API**: Contiene el archivo `.env` que tendrás que introducir en `~/Back/API`.

### 9.3 Backend y Panel de Administración

Hemos creado un archivo `Dockerfile` en cada uno de los proyectos (backend, panel de administración y aplicación móvil) para facilitar la creación de contenedores que contienen las aplicaciones y sus dependencias. Además, hemos configurado un archivo `docker-compose.yml` en la ruta raíz del repositorio (`~`) para gestionar la ejecución de todos los servicios necesarios de manera más sencilla.

Los requisitos previos para ejecutar el proyecto en contenedores Docker son los siguientes:

- Tener **Docker** instalado en tu máquina.
- Tener **Docker Compose** instalado.

### 9.3.1 Comandos para ejecutar el proyecto

#### 1. Construir las imágenes Docker:

Para construir las imágenes de los contenedores a partir de los `Dockerfile` de cada proyecto, puedes ejecutar el siguiente comando en la raíz del repositorio (~):

```
docker-compose build
```

Este comando construye las imágenes de todos los servicios definidos en el archivo `docker-compose.yml`, utilizando los respectivos `Dockerfile` de cada componente (backend, panel de administración, etc.).

#### 2. Levantar los contenedores:

Una vez que las imágenes estén construidas, puedes levantar los contenedores y ejecutar los servicios con el siguiente comando:

```
docker-compose up
```

Este comando inicia todos los contenedores en segundo plano (con la opción `-d` si deseas ejecutarlo en modo "detached").

#### 3. Detener los contenedores:

Para **detener** y **eliminar** los contenedores y redes creados por `docker-compose`, puedes usar el siguiente comando:

⚠ Solo ejecutar este comando si deseas terminar la demo ⚠

```
docker-compose down
```

Esto detendrá los contenedores y los eliminará, pero no eliminará las imágenes. Si quieres eliminar las imágenes también, puedes agregar la opción `-rmi all`.

#### Resumen de comandos

- `docker-compose build` – Construye las imágenes de los contenedores.
- `docker-compose up` – Levanta los contenedores
- `docker-compose down` – Detiene los contenedores.

### 9.4 Aplicación Móvil

Para la aplicación móvil, hemos dejado un release del APK en modo *preview* en el repositorio de GitHub. Este APK se puede instalar directamente en un dispositivo Android y no requiere configuración adicional. Sin embargo, en caso de querer ejecutar el código, los pasos son los siguientes:

1. Será necesario tener instalado **Node.js** y **npm** para poder ejecutarla localmente.
2. Descargar la versión de desarrollo publicada también en la sección de releases con el nombre `DEVELOPMENT.apk`.
3. En la ruta del proyecto del cliente `~/Front/ClientApp/`, ejecutar

```
npm i
```

seguido de

```
npx expo start
```

4. Se abrirá un servidor de desarrollo local. El dispositivo móvil debe estar en la misma red (sin restricciones) que el ordenador en el que se esté ejecutando el servidor.
5. Una vez ejecutado, se mostrará un código QR que deberá ser escaneado con la aplicación nativa de la cámara del móvil. Esto redirigirá al móvil a la versión de desarrollo.
6. Tras finalizar el proceso de *bundle*, la aplicación estará lista para probar.

Una vez que la aplicación esté funcionando, hemos establecido una página inicial que obliga a introducir la URL de la API antes de continuar. Al ingresar la URL, la aplicación intentará validar la conexión mediante una petición ping al servidor. Si el servidor no está abierto, no se permitirá avanzar.

Una vez validada la URL de la API, se te pedirá iniciar sesión o crear una cuenta. Puedes crear una cuenta utilizando información no verídica, como `test@test.com`, ya que hemos desactivado la autenticación por correo electrónico. Cualquier texto con formato de correo electrónico será aceptado.

Una vez iniciada la sesión, ¡eres libre de explorar la aplicación a tu antojo!

#### 9.4.1 Notas sobre la Aplicación

A continuación se detallan algunas notas importantes sobre la aplicación:

- **Pantalla de ranking:** No es funcional por el momento.
- **Creación de grupos:** Se realiza desde la pantalla de *Perfil*.
- **URL de la API:** Se puede cambiar también desde la pantalla de *Perfil*.
- **Mapa:** A veces, el mapa tarda mucho en cargar las coordenadas GPS. Sin embargo, al final las obtiene correctamente. Creemos que se debe a una petición asíncrona que se queda esperando en cola.
- **Error en la versión *preview*:** Hemos encontrado un error en la versión *preview* compilada, en el cual aparece una barra negra en la parte superior al pasar de la pantalla de mapa a otra ventana. Este error no es visible en la build de *development*
- **Activación del asistente:** El asistente se activa manteniendo pulsado el botón azul o, con una única pulsación, se accede a la ventana específica del asistente.
- **Indicadores de conexión:** En la ventana del asistente, el indicador rojo/verde derecho de la cara muestra si la conexión *websocket* está establecida y el izquierdo indica la conexión con LiveKit. Cuando ambos indicadores son correctos, se muestra una cara sonriente.

## 10 Demo reproducible

### 10.1 1 - Creación de un nuevo local

Dirígete al end point /form del panel de administración ,y rellena los datos

### 10.2 2 - Aprobación del local creado y creación de retos

Para comenzar, dirígete a la URL principal (<http://localhost:5173>) del panel de administración. Se solicitarán las credenciales de acceso:

- **Usuario:** admin
- **Contraseña:** admin123

Al iniciar sesión, serás redirigido a la sección de estadísticas del panel de administración. En esta sección podrás visualizar datos relevantes como estadísticas generales, gráficos y métricas de sostenibilidad relacionadas con hoteles, retos, locales, entre otros.

En la parte derecha de la interfaz se muestra un menú de navegación lateral, desde el cual puedes acceder a diferentes secciones del panel, tales como:

- **Retos**
- **Locales**
- **Premios**

#### 10.2.1 Aprobación de un local

Dentro de la sección **Locales**, aparecerá la solicitud del local que acabamos de crear. Desde esta pantalla, es posible aceptar la solicitud, así como editar los datos del establecimiento antes de su aprobación.

#### 10.2.2 Gestión y creación de retos

En la sección **Retos**, se muestra la lista de retos existentes, permitiendo editar, activar o desactivar cada uno de ellos. Para crear un nuevo reto, basta con pulsar el botón **Añadir Reto**, lo que desplegará un formulario donde se debe ingresar la información del nuevo desafío.

Dentro del formulario, en el apartado **Tipo de Completado**, se puede seleccionar la forma en que el usuario podrá completar el reto. Las opciones disponibles son:

- **QR:** Escaneando un código QR.
- **Punto GPS:** Llegando a una ubicación específica.
- **Ruta GPS:** Siguiendo una ruta determinada.
- **Foto:** Subiendo una imagen como prueba.

Si se selecciona la opción **Punto GPS**, aparecerá la opción de elegir una de las localizaciones ya creadas. De igual manera, si se selecciona **Ruta GPS**, será posible elegir una de las rutas disponibles. Cabe mencionar que, actualmente, el creador de rutas no está implementado, por lo que las dos rutas disponibles han sido añadidas manualmente a la base de datos.

#### 10.2.3 Gestión de premios

En la sección **Premios**, se puede visualizar la lista de premios disponibles, así como crear nuevos o editar los existentes según sea necesario.

#### 10.2.4 Inicio de sesión en el dispositivo móvil

Una vez descargada la aplicación, ya sea en modo *development* o *preview*, se mostrará una pantalla inicial en la que se solicitará la introducción de la URI de la API.

Es importante destacar que el dispositivo móvil debe estar conectado a la misma red que el servidor donde se ejecuta la API. Además, dicha red no debe tener restricciones que impidan el acceso en local, ya que esto podría interferir con la comunicación entre la aplicación y el servidor.

Una vez que la dirección de la API ha sido validada, será necesario reiniciar la aplicación. Tras el reinicio, aparecerá la ventana de inicio de sesión, donde el usuario podrá autenticarse o registrarse en la plataforma.

Para fines de prueba, se dispone de una cuenta preconfigurada con las siguientes credenciales. No obstante, la creación de una cuenta con datos ficticios proporcionará la misma experiencia y funcionalidad:

- **Usuario:** test@test.com
- **Contraseña:** testtest

#### 10.2.5 Aceptar un desafío

Para aceptar un reto, basta con seleccionar una de las localizaciones en el mapa. Al desplegar la lista de retos disponibles, podemos elegir el que más nos interese y pulsar en "Aceptar desafío" para los retos de tipo ruta, o en "Ruta" en los casos de retos basados en QR, GPS y foto.

Una vez aceptado, se mostrará la ruta hacia el reto, permitiéndonos completarlo según el método correspondiente. En el caso de los retos de tipo GPS, se puede utilizar una aplicación de simulación de ubicación para facilitar las pruebas.

Los retos cuya completación se basa en códigos QR podrán ser completados escaneando cualquier tipo de QR en la aplicación de desarrollo, sin restricciones.

### 10.3 Hablar con el asistente

Para iniciar una conversación con el asistente, basta con pulsar el botón azul flotante. Esta acción te llevará directamente a la pantalla principal del asistente, donde podrás comunicarte con él mediante voz o texto.

También es posible mantener pulsado el botón azul para hablar con el asistente sin abandonar la ventana actual. Esta funcionalidad permite recibir asistencia contextual sin interrumpir la navegación por la aplicación.

**Nota sobre un bug detectado:** Actualmente, existe un error intermitente que impide que la conexión con el asistente se establezca correctamente en algunos casos. Como solución temporal, se recomienda:

- Si se utiliza la opción de mantener pulsado y no funciona, se puede hacer una pulsación única, acceder a la pantalla del asistente, y luego regresar a la pantalla anterior. Tras esto, la funcionalidad de mantener pulsado suele restaurarse con normalidad.

## 11 Bibliografía y Referencias

### References

- [1] LiveKit, *Overview of LiveKit Agents*, <https://docs.livekit.io/agents/overview/>
- [2] TechWithTim, *LiveKit-AI-Car-Call-Centre*, GitHub, <https://github.com/techwithtim/LiveKit-AI-Car-Call-Centre/tree/main>, Accedido: Abril 2025.