



FUTSCORES

FutScores es una aplicación que le permite a aficionados del Fútbol recibir las últimas actualizaciones de partidos de los equipos que siguen.



*Juan Francisco Ramírez Escobar
Javier Alejandro Moyano Cipamocha
Esteban Salazar Arbeláez*

PATRÓN PUB/SUB

Juega un papel clave para gestionar la actualización de eventos deportivos en tiempo real.



RabbitMQ



redis

Redis



AstroJS



GoLang



ESTILO PUB/SUB

Modelo de comunicación asíncrona en el que los emisores (publishers) envían mensajes a un canal, y los receptores (subscribers) se suscriben a los **tópicos** para recibir los mensajes que les interesan.



PUBLISHER

- Cada vez que hay un evento importante en un partido el sistema publica un mensaje en un canal de eventos.
- El mensaje incluye detalles como el tipo de evento (gol, tarjeta), el equipo involucrado y la hora exacta del evento.

SUBSCRIBER

- Los usuarios de la plataforma que están interesados en seguir un partido específico se suscriben a ese canal de eventos.
- El sistema se asegura de que estos usuarios reciban una actualización instantánea cada vez que el publicador emite un mensaje sobre el partido que están siguiendo.



VENTAJAS



- **Desacoplamiento:** Los publishers y subscribers no necesitan conocerse, lo que permite que los componentes del sistema evolucionen independientemente.
- **Escalabilidad:** Es fácil añadir nuevos suscriptores sin afectar a los publishers, lo que permite que el sistema crezca sin problemas.
- **Distribución de mensajes en tiempo real:** Garantiza que los mensajes se entreguen a todos los suscriptores en tiempo real
- **Eficiencia y asincronía:** Permite el envío de mensajes de manera asincrónica, lo que mejora el rendimiento del sistema al no depender de respuestas inmediatas.





DESVANTAJAS

- **Complejidad de configuración:** La implementación del estilo puede requerir una infraestructura más compleja y la configuración de middleware para la gestión de los mensajes
- **Sobrecarga innecesaria en sistemas pequeños:** Para aplicaciones simples o de bajo tráfico, el estilo PUB/SUB puede ser innecesariamente complejo y generar una sobrecarga.
- **Dificultad para depurar:** El desacoplamiento entre el emisor y los suscriptores hace que sea más difícil rastrear errores y problemas de entrega de mensajes

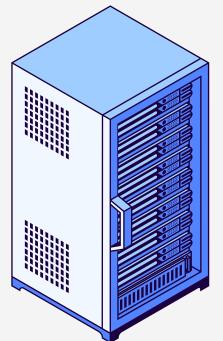


HISTORIA



1990's

El estilo PUB/SUB surge como solución para la comunicación asíncrona en sistemas distribuidos, especialmente en CORBA y sistemas basados en eventos



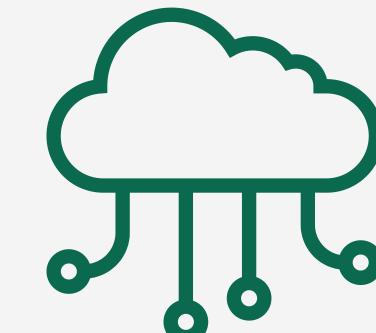
2000's

El estilo gana popularidad con el auge de arquitecturas orientadas a servicios (SOA) y mensajería empresarial, permitiendo escalabilidad en sistemas complejos.



2010's

Con la adopción de microservicios y cloud computing, PUB/SUB se convierte en un pilar para aplicaciones en tiempo real. Soluciones como RabbitMQ y Kafka implementan este patrón.



RABBIT MQ

RabbitMQ es una de las soluciones de mensajería más comunes en el mercado. Es ampliamente adoptado en arquitecturas de microservicios y sistemas distribuidos para manejar la comunicación entre aplicaciones y servicios.



INSTAGRAM

Utiliza RabbitMQ para gestionar la cola de notificaciones de los usuarios cuando reciben un comentario o un like en una foto. Esto permite que la aplicación procese grandes volúmenes de notificaciones en tiempo real.



PINTEREST

Usa RabbitMQ para distribuir eventos de "pinning" a través de diferentes servidores, lo que permite gestionar las actualizaciones de contenido de los usuarios y sus seguidores.





VENTAJAS

- **Desacoplamiento:** RabbitMQ permite que los productores (publishers) y consumidores (subscribers) estén completamente desacoplados, lo que facilita la evolución independiente de los componentes del sistema.
- **Manejo de múltiples protocolos:** RabbitMQ es compatible con múltiples protocolos de mensajería (como AMQP, MQTT, STOMP).
- **Enrutamiento avanzado:** RabbitMQ permite enrutamiento flexible mediante el uso de exchanges (direct, topic, fanout, etc.), lo que facilita la creación de patrones de mensajería complejos según las necesidades del sistema.





DESVANTAJAS

- **Complejidad de configuración:** La instalación y configuración de RabbitMQ puede ser compleja, especialmente al gestionar entornos de alta disponibilidad, clústeres y configuraciones avanzadas de seguridad.
- **Consumo de recursos:** RabbitMQ puede requerir un uso considerable de memoria y procesamiento, especialmente cuando maneja grandes cantidades de mensajes o persistencia.
- **Dificultad en la depuración:** Como en todo sistema PUB/SUB, el desacoplamiento entre los productores y los consumidores dificulta la trazabilidad de los mensajes.



HISTORIA



2007

RabbitMQ fue lanzado en 2007 por Rabbit Technologies, creado específicamente para implementar el protocolo AMQP (Advanced Message Queuing Protocol).



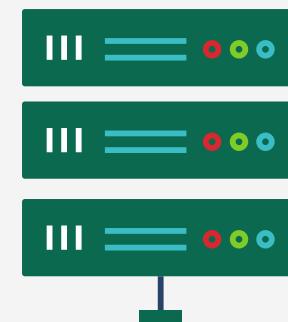
2010

En 2010, VMware adquirió Rabbit Technologies, incorporando RabbitMQ a su portafolio de soluciones empresariales para la integración de aplicaciones.



2015

Con el auge de las arquitecturas de microservicios y el cloud computing, RabbitMQ se consolida como una herramienta clave para la mensajería en sistemas distribuidos.



ASTRO JS

Comparado con frameworks más establecidos como React o Vue, Astro todavía está en una fase de crecimiento, pero su uso en proyectos de rendimiento crítico está aumentando.



SKYSCANNER

Un popular sitio web de búsqueda de vuelos que utiliza Astro para optimizar el rendimiento de su landing page, asegurando tiempos de carga rápidos y una mejor experiencia de usuario.



E-COMMERCE

Empresas de ecommerce de bajo volumen y sitios corporativos han implementado Astro para mejorar la experiencia de usuario, especialmente en dispositivos móviles, con la carga mínima de JavaScript.



VENTAJAS



- **Rendimiento optimizado:** Genera HTML estático de manera predeterminada, lo que reduce los tiempos de carga de la página.
- **Carga mínima de JavaScript:** Solo carga JavaScript cuando es necesario para la interactividad, lo que minimiza el tamaño y carga de la página y mejora la experiencia de usuario.
- **Integración con frameworks populares:** Astro se puede integrar con componentes de React, Vue, Svelte y más, lo que facilita la creación de sitios modernos y flexibles.





DESVANTAJAS

- **Limitaciones en aplicaciones dinámicas:** Astro está más orientado a sitios estáticos o con pocas interacciones dinámicas.
- **Ecosistema en crecimiento:** Aunque Astro está creciendo rápidamente, su ecosistema y la cantidad de plugins no es tan vasta como la de frameworks más consolidados.
- **Renderizado en tiempo de compilación:** Dado que Astro genera la mayor parte del contenido estáticamente, puede no ser la mejor opción para aplicaciones que requieren un renderizado dinámico en el servidor o TR.



HISTORIA



2021

Astro fue lanzado en 2021 por los creadores de Snowpack, con el objetivo de optimizar la creación de sitios estáticos y mejorar el rendimiento web.



2022

En 2022, Astro ganó tracción rápidamente gracias a su enfoque en la carga mínima de JavaScript y su capacidad para integrarse con frameworks populares.



Actualidad

Hoy en día, Astro se está adoptando cada vez más en sitios que buscan optimizar el rendimiento, SEO, y la experiencia del usuario.



GO LANG

Go es un lenguaje de programación muy popular en la actualidad, especialmente en el desarrollo de sistemas backend, microservicios, y plataformas de infraestructura en la nube. Su adopción ha crecido significativamente debido a su simplicidad y eficiencia.



UBER

Uber utiliza Go en varias de sus microservicios backend, particularmente aquellos que requieren alta concurrencia y baja latencia para manejar el tráfico en tiempo real de su plataforma de transporte.



SOUNDCLOUD

La plataforma de streaming SoundCloud eligió Go para construir APIs que soportan gran cantidad de usuarios y reproducen música en tiempo real, garantizando un rendimiento óptimo.



VENTAJAS



- **Alto rendimiento:** Go es un lenguaje compilado que produce ejecutables muy eficientes en cuanto a consumo de recursos y velocidad, lo que lo hace ideal para sistemas de alto rendimiento y aplicaciones a gran escala.
- **Simplicidad y facilidad de uso:** Go está diseñado con una sintaxis simple y un conjunto de características minimalistas, lo que reduce la curva de aprendizaje y facilita su uso. (*Es un mix entre C++ y Python*)
- **Excelente soporte para concurrencia:** Go tiene un modelo de concurrencia incorporado basado en goroutines y canales, lo que permite desarrollar aplicaciones altamente concurrentes y escalables.





DESVANTAJAS

- **Menor comunidad y ecosistema que lenguajes más maduros:** Aunque Go tiene una comunidad en crecimiento, no es tan amplia como la de lenguajes más antiguos como Java o Python.
- **Limitaciones en proyectos de frontend:** Go está principalmente orientado al desarrollo backend y de sistemas, por lo que no es común verlo en proyectos de frontend o en entornos más centrados en el cliente.
- **Rigidez en la estructura del código:** Go exige una estructura de proyectos muy específica y estricta, lo que puede ser limitante para los desarrolladores que están acostumbrados a tener más flexibilidad.



HISTORIA



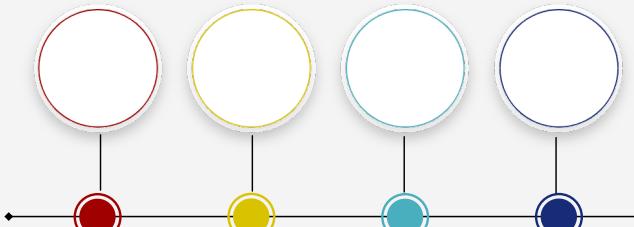
2009

Go fue desarrollado por Google en 2009 por ingenieros como Robert Griesemer, Rob Pike y Ken Thompson, con el objetivo de mejorar la gestión de concurrencia y simplificar el desarrollo de aplicaciones escalables



2013

En 2013, Go alcanzó la versión 1.0, marcando su adopción formal en proyectos de producción.



2016

Go comenzó a ser adoptado por grandes empresas como Docker, Kubernetes y Uber, debido a su capacidad para manejar servicios de alta concurrencia y sistemas distribuidos.



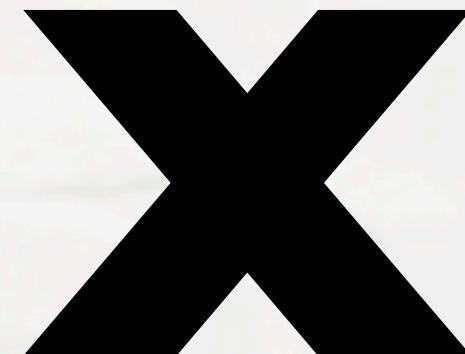
REDIS DB

Redis es extremadamente común en aplicaciones que requieren un acceso rápido a datos temporales o de alta frecuencia, siendo utilizado para almacenamiento en caché, gestión de sesiones, y colas de trabajo en tiempo real.



TWITTER / X

Twitter utiliza Redis para gestionar el almacenamiento en caché de las timelines de los usuarios, mejorando el rendimiento en la entrega de actualizaciones en tiempo real.



GITHUB

Redis se usa en GitHub para gestionar la cola de tareas de procesamiento en segundo plano, como la compilación de repositorios y la entrega de notificaciones a los usuarios.



VENTAJAS



- **Alto rendimiento:** Redis es extremadamente rápido debido a que almacena los datos en memoria, lo que lo convierte en una excelente opción para aplicaciones que requieren respuestas en tiempo real y baja latencia.
- **Soporte para múltiples estructuras de datos:** Redis soporta estructuras de datos avanzadas como listas, sets, hashes, sorted sets, bitmaps, y hyperloglogs, lo que lo hace muy versátil para diferentes tipos de casos de uso.
- **Facilidad de uso:** Redis tiene una interfaz simple y comandos intuitivos, lo que facilita su adopción y uso incluso por desarrolladores que no están familiarizados con bases de datos NoSQL.





DESVANTAJAS

- **Limitaciones de memoria:** Al ser una base de datos en memoria, Redis está limitado por la cantidad de RAM disponible en el servidor. Esto puede ser un problema para aplicaciones que manejan grandes volúmenes de datos, ya que la memoria RAM es más costosa que el almacenamiento en disco.
- **Persistencia no garantizada en tiempo real:** Aunque Redis ofrece persistencia de datos, su enfoque en la velocidad puede ocasionar una pérdida de datos en escenarios de fallo, ya que la persistencia no es continua y depende de configuraciones como snapshots periódicos o la escritura en el log AOF.



HISTORIA



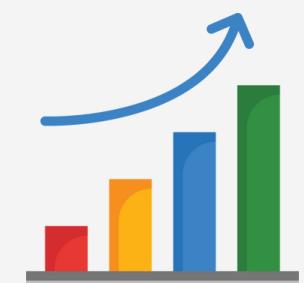
2009

Redis fue creado en 2009 por Salvatore Sanfilippo para mejorar la escalabilidad de una aplicación web de análisis de tráfico. Inicialmente, era una base de datos clave-valor simple que operaba en memoria.



2011-2015

Durante estos años, Redis evolucionó con el soporte para nuevos tipos de datos, replicación, y la introducción de la persistencia en disco.



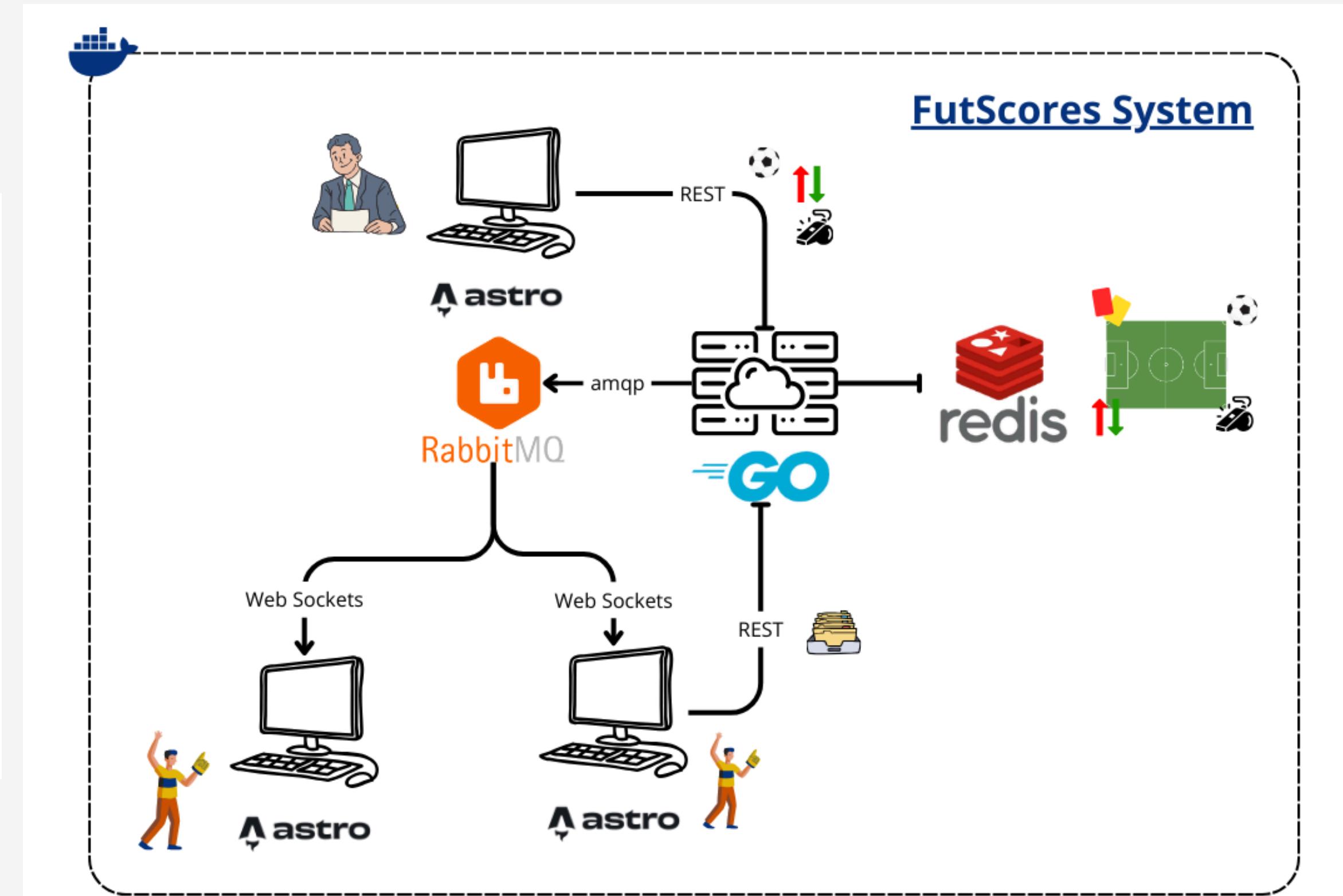
2016

Redis se convirtió en una herramienta fundamental para aplicaciones de alto rendimiento, siendo adoptado por empresas como Twitter, GitHub, y Snapchat.



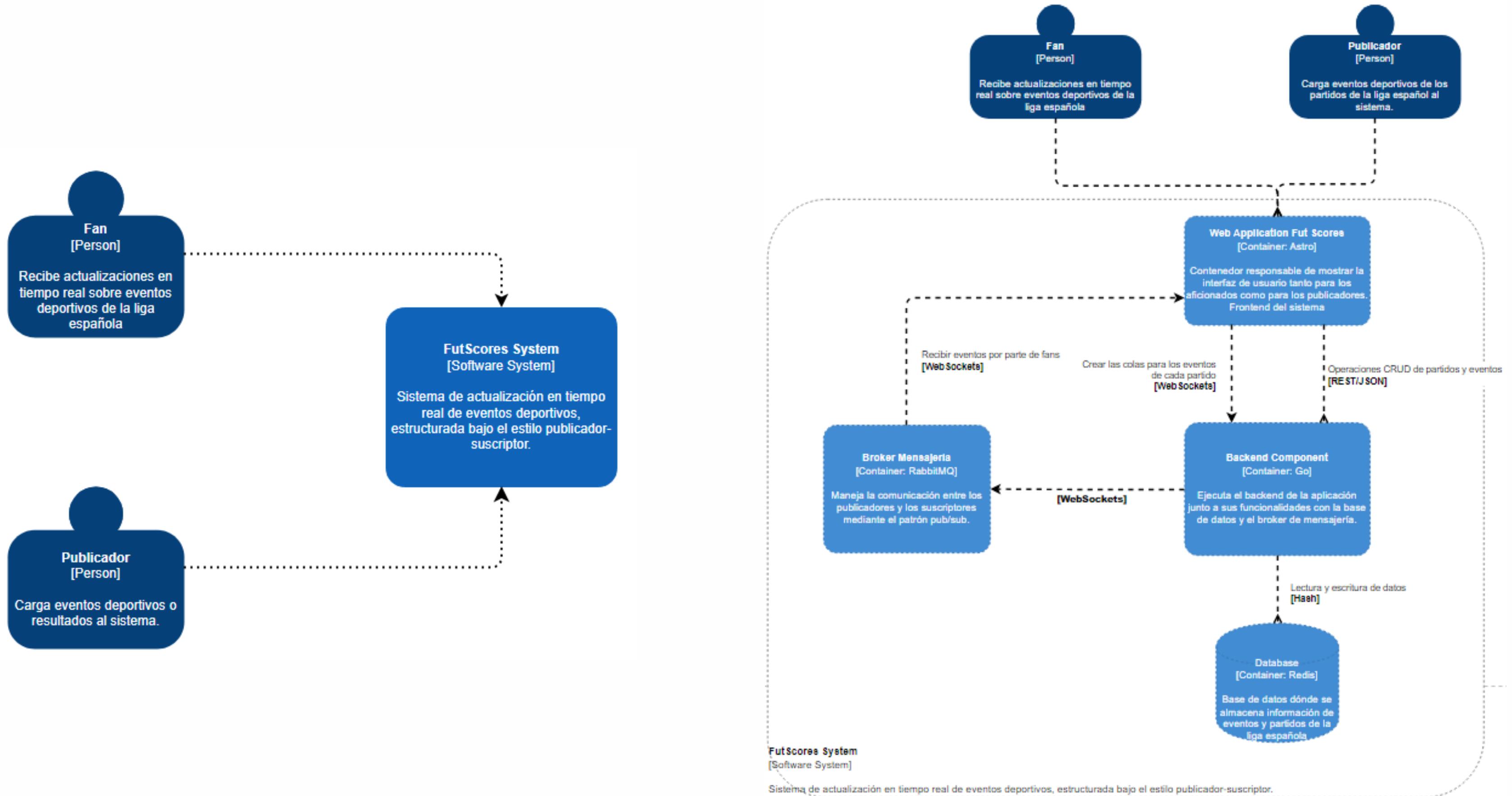
APLICACION

FUTSCORES



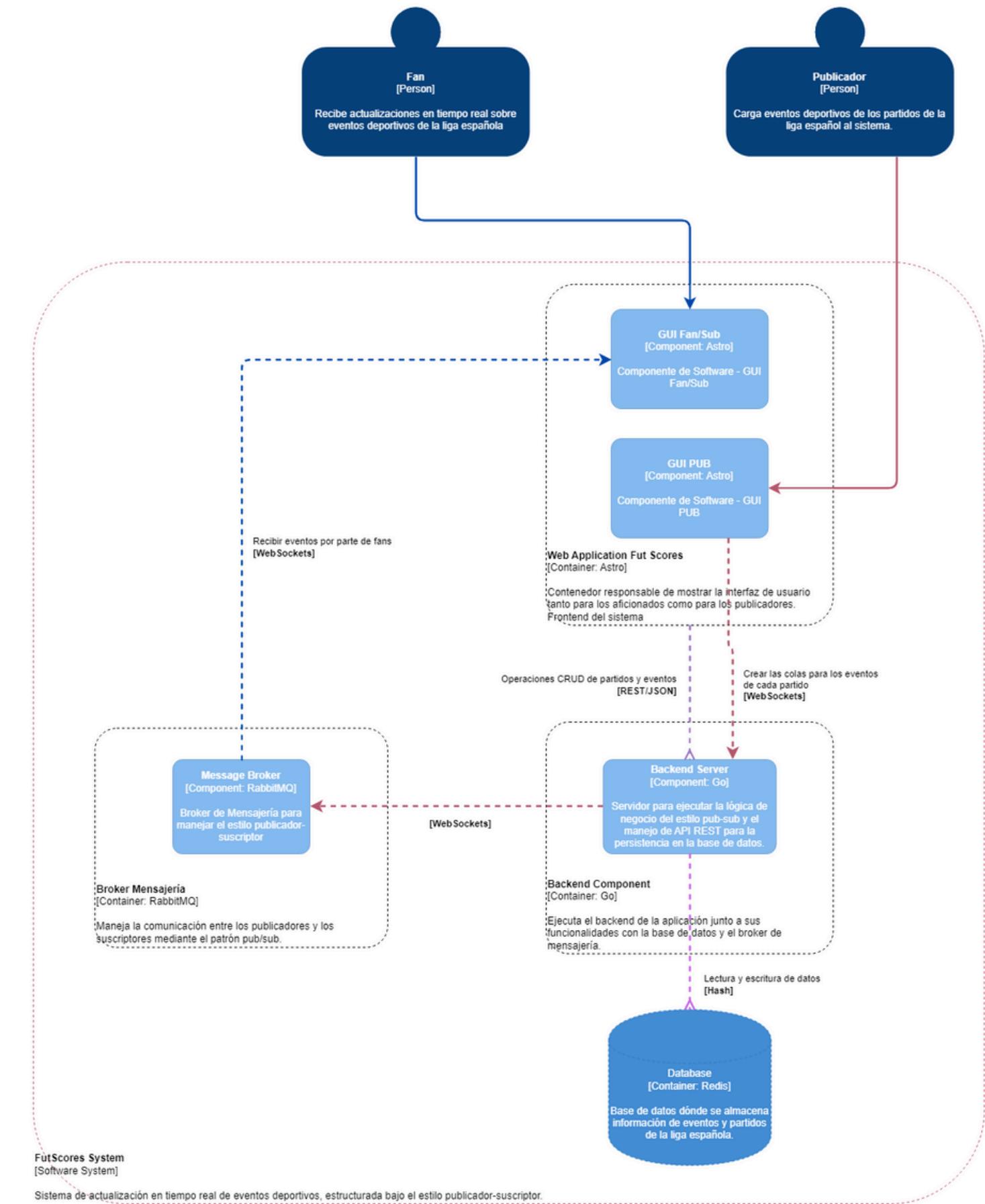
VISTAS DEL SISTEMA - C4

C4



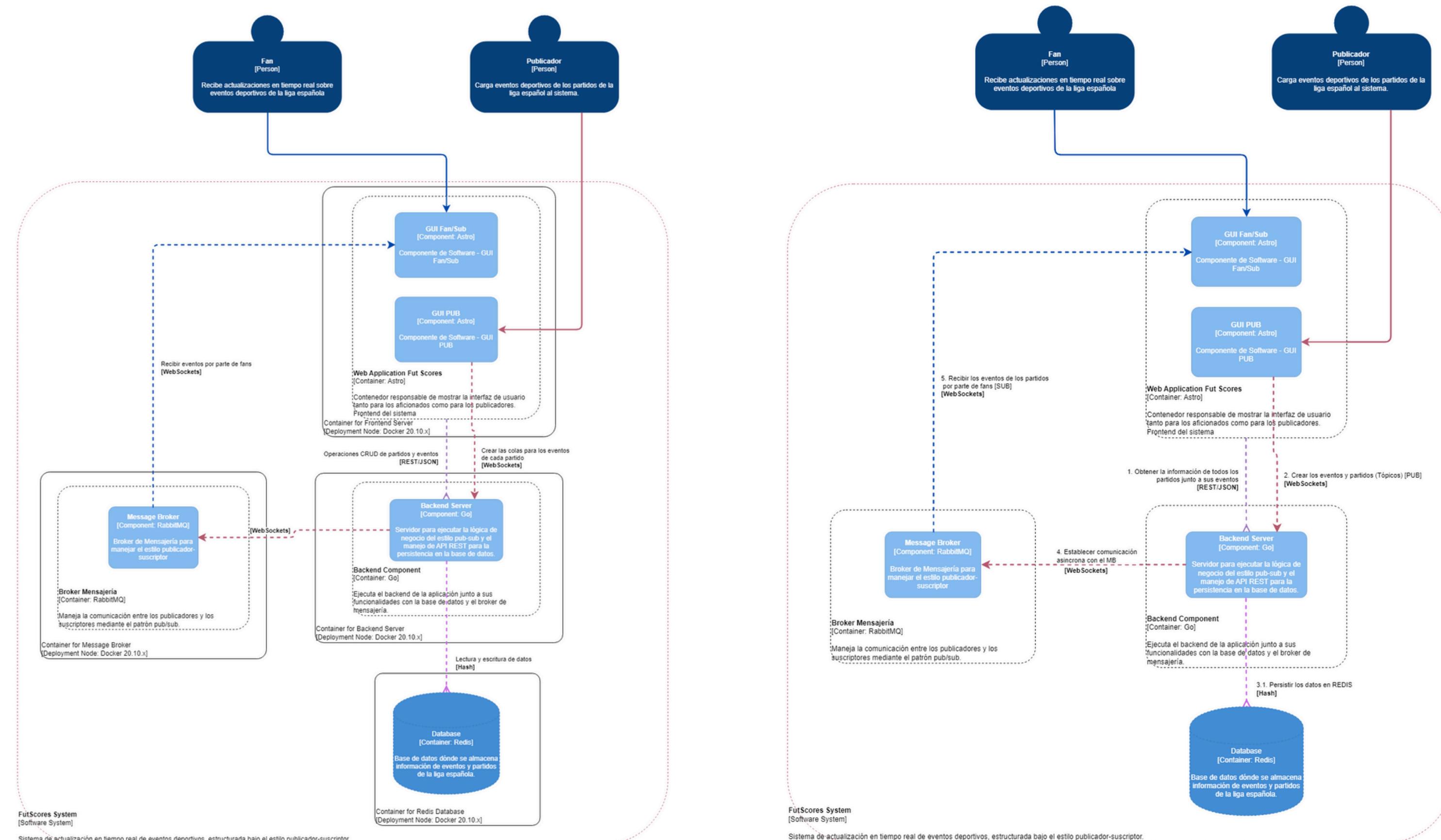
VISTAS DEL SISTEMA - C4

C4



VISTAS COMPLEMENTARIAS - C4

C4



MATRIZ MERCADO LABORAL VS TEMAS

Tecnología	Demanda de Empleo	Oportunidades de crecimiento profesional
Patron PUB/SUB	Alta, especialmente en sistemas distribuidos, IoT, y aplicaciones en tiempo real.	Oportunidades significativas para ingenieros de backend y desarrolladores especializados en sistemas distribuidos.
Astro JS	Moderada, debido a su reciente adopción, pero en aumento entre empresas que priorizan la optimización del rendimiento web.	Crecimiento moderado, con oportunidades en áreas de desarrollo web moderno y performance optimization.
RabbitMQ	Alta, con una demanda constante en empresas que utilizan arquitecturas de microservicios y sistemas de mensajería.	Oportunidades amplias para arquitectos de software y desarrolladores en sistemas distribuidos y mensajería.
Redis	Muy alta, Redis es ampliamente adoptado para el almacenamiento en caché y bases de datos en memoria.	Altas oportunidades en roles de backend, DevOps, y administración de bases de datos NoSQL.
GoLang	Muy alta, especialmente en el desarrollo backend y servicios en la nube.	Gran crecimiento profesional, con fuerte demanda de desarrolladores de Go en el desarrollo de microservicios, APIs y plataformas cloud.



**GRACIAS POR
SU ATENCIÓN**