Universidad Nacional de Costa Rica

Sede Regional Chorotega, campus Nicoya

Curso:

Paradigmas de programación

Trabajo de investigación final

F1 Data Analytics Dashboard

Tema:

Dashboard web para monitoreo de datos usando programación reactiva (React.js) y backend multiparadigma (Node.js).

Grupo:

1

Profesora:

Marianella Villafuerte

Estudiantes:

Dixon Gaitán Martínez	33.33%
Cristian Altamirano Montes	33.33%
Danny Orlando Abarca Alvarado	33.33%
Total	100%

Contenido

Introducción

En la actualidad, la analítica de datos en tiempo real se ha convertido en un componente esencial dentro del deporte de alto rendimiento. La Fórmula 1, al ser una de las disciplinas tecnológicamente más avanzadas del mundo, depende directamente del análisis instantáneo de variables como tiempos de vuelta, posiciones, estrategias de neumáticos y puntos obtenidos. Cada decisión en la pista se apoya en datos y la capacidad de procesarlos y visualizarlos de manera reactiva determina la eficacia con que un equipo puede adaptarse a las condiciones cambiantes de una carrera. En este contexto, surge el proyecto "F1 Data Analytics Dashboard", cuyo propósito es desarrollar una plataforma web capaz de monitorear y analizar en tiempo real el rendimiento de pilotos y equipos, aplicando paradigmas de programación combinados.

El sistema integra un frontend basado en React.js, construido bajo los principios de la programación reactiva, y un backend desarrollado en Node.js que combina los enfoques orientados a objetos y funcional. Esta arquitectura multiparadigma permite manejar el flujo de información de forma continua mediante Server-Sent Events (SSE), garantizando que los datos como la posición actual de un piloto, su tiempo de vuelta más reciente o los puntos acumulados se actualicen de forma automática en la interfaz, sin necesidad de recargar la página. De este modo, el proyecto reproduce el comportamiento de un centro de telemetría de F1, donde cada milisegundo cuenta y la capacidad de respuesta del sistema es fundamental.

Además del módulo principal de Leaderboard, el dashboard incorpora una sección de Metrics, dedicada al análisis del rendimiento interno del sistema, que mide parámetros como la latencia promedio, el número de actualizaciones por segundo (TPS) y las conexiones activas al flujo SSE. Esta característica permite validar la estabilidad y eficiencia del canal de datos, agregando una capa de autoevaluación que refuerza el valor técnico del proyecto. Así, el sistema no solo analiza a los pilotos y su desempeño en la pista, sino también su propio comportamiento como aplicación reactiva en tiempo real.

En términos académicos, el F1 Data Analytics Dashboard demuestra la aplicación práctica de paradigmas de programación en un entorno complejo, donde concurren asincronía, transmisión continua de información y visualización inmediata. Más allá de su

alcance funcional, el proyecto evidencia un enfoque profesional en la separación de capas, la tipificación de datos y la observabilidad del sistema, reflejando competencias clave en desarrollo de software moderno. En conjunto, esta propuesta materializa los objetivos del curso de Paradigmas de Programación al fusionar teoría, técnica y ejecución en una solución coherente, escalable y alineada con los estándares actuales de la ingeniería de software.

Planteamiento del problema

Necesidades de monitoreo en F1

La Fórmula 1 demanda una lectura continua de indicadores que evolucionan segundo a segundo. El tiempo de vuelta no solo determina el ritmo instantáneo, también habilita comparativas contra el mejor tiempo y deltas por piloto; la posición en pista cambia por rebases, paradas en boxes y eventos de safety car, por lo que la clasificación debe reflejar variaciones de forma inmediata y orden estable; finalmente, los puntos condicionan las estrategias a nivel de campeonato y requieren una actualización coherente con el reglamento de puntuación por sesión. En conjunto, estos tres ejes vueltos, posición y puntos, definen el estado de carrera y exigen un dashboard capaz de integrarlos en una vista unificada, con jerarquía visual clara y sin inconsistencias entre lo calculado y lo mostrado.

Requisitos de latencia, consistencia y UX

Para que el dashboard sea útil en un entorno de alta variabilidad, la latencia extrema a extremo del backend a la interfaz debe mantenerse baja y estable, de modo que la percepción del usuario no quede desfasada respecto al estado real. La consistencia implica que el orden y la integridad de los datos se preserven: si cambia el tiempo de vuelta de un piloto, la posición y los puntos asociados deben reaccionar en el mismo ciclo de actualización, evitando “parpadeos” o estados intermedios contradictorios. En términos de experiencia de usuario (UX), la interfaz debe comunicar de manera explícita estados de carga, reconexión y vacío de datos; priorizar legibilidad de la tipografía, contraste, densidad informativa y sostener una interacción reactiva donde las variaciones se reflejen sin recargas, con transiciones sutiles que mantengan el contexto.

Justificación de streaming (SSE)

El streaming de eventos es el mecanismo apropiado para sincronizar un flujo de datos que nunca termina durante una sesión de F1. Server-Sent Events (SSE) ofrece un canal unidireccional, eficiente y simple de mantener desde el servidor hacia el navegador, ideal para emisiones periódicas de estado por ejemplo Leaderboard y métricas con bajo overhead y reconexión automática del lado del cliente. A diferencia de sondeos por intervalo polling, el SSE reduce latencia percibida y tráfico innecesario; y frente a WebSockets, aporta una implementación más ligera cuando no se requiere comunicación

bidireccional. En este proyecto, SSE permite propagar cambios de vueltas, posición y puntos de forma continua y coherente con la UI reactiva, a la vez que habilita un módulo de métricas operativas como TPS, latencia observada, conexiones activas para validar en tiempo real la salud del propio sistema y garantizar que lo visto por el usuario corresponde al estado actual del flujo de datos.

Objetivo general

Diseñar e implementar un dashboard web de Fórmula 1 para monitoreo y análisis en tiempo real, aplicando programación reactiva en React.js y un backend multiparadigma en Node.js funcional y orientado a objetos, a fin de visualizar con baja latencia los tiempos de vuelta, posiciones y puntos de los pilotos.

Objetivos específicos

1. Implementar en el frontend un flujo reactivo que se suscriba a eventos de datos SSE y actualice sin recarga el Leaderboard y los indicadores clave.
2. Modelar en el backend el dominio OO (piloto, vuelta, clasificación) y aplicar transformaciones funcionales (map/filter/reduce) para calcular KPIs mejor vuelta, deltas, promedios.
3. Garantizar la consistencia y usabilidad de la interfaz, gestionando estados de carga, vacío y reconexión, y presentando la información con orden estable y legibilidad.
4. Instrumentar la observabilidad del sistema mediante métricas operativas por ejemplo latencia y TPS y definir criterios de aceptación medibles para validar el desempeño en la demo.

Justificación de paradigmas utilizados

El desarrollo del F1 Data Analytics Dashboard se fundamenta en la combinación de tres paradigmas de programación: reactivo, funcional y orientado a objetos, aplicados de forma complementaria para resolver la necesidad de monitorear datos de Fórmula 1 en tiempo real. Esta integración permite aprovechar las fortalezas de cada enfoque, garantizando una arquitectura moderna, eficiente y mantenible.

En el frontend, se implementó la programación reactiva mediante React.js, un entorno que facilita la suscripción a flujos de datos continuos y la actualización automática de la interfaz sin recargar la página. Cada componente del dashboard responde a cambios en el estado global provocados por eventos emitidos desde el servidor, permitiendo que la información sobre pilotos, tiempos de vuelta, posiciones y puntos se muestre de forma inmediata y coherente.

En el backend, desarrollado con Node.js, se aplicó un enfoque multiparadigma. Por un lado, la programación orientada a objetos se utilizó para modelar las entidades principales del dominio como el piloto, vuelta, clasificación y organizar los servicios responsables de la lógica del sistema, lo que favorece la modularidad y el mantenimiento del código. Por otro lado, la programación funcional se empleó en la transformación y análisis de los datos, aprovechando funciones puras y operaciones como map, filter y reduce para calcular métricas, promedios y deltas de tiempo entre pilotos, asegurando resultados predecibles y fáciles de probar.

Para la transmisión de información entre el backend y el frontend se seleccionó Server-Sent Events (SSE) como tecnología de comunicación en tiempo real. Esta elección se justifica por su simplicidad y eficiencia en escenarios donde los datos fluyen principalmente del servidor hacia el cliente. A diferencia de los WebSockets, SSE ofrece un canal unidireccional con reconexión automática, menor consumo de recursos y compatibilidad nativa con navegadores modernos, lo que lo hace ideal para la actualización continua del Leaderboard y el módulo de Metrics sin sobrecargar el sistema.

En conjunto, la combinación de estos paradigmas permitió construir un sistema reactivo, modular y funcionalmente expresivo, capaz de reflejar en tiempo real el desempeño de los pilotos y al mismo tiempo medir la estabilidad del propio flujo de datos.

Esta integración representa la aplicación práctica de los principios teóricos del curso, demostrando cómo paradigmas distintos pueden coexistir de manera armoniosa para resolver un problema técnico real.

Diseño técnico

El diseño técnico del F1 Data Analytics Dashboard se basa en una arquitectura cliente servidor reactiva, donde el flujo de información parte del backend, se transmite mediante Server-Sent Events (SSE) y se refleja de forma inmediata en la interfaz desarrollada con React.js. La estructura general se divide en tres capas principales: backend multiparadigma, canal de comunicación en tiempo real y frontend reactivo.

En la capa del servidor (backend), implementada en Node.js, se definieron los componentes esenciales del dominio siguiendo el paradigma orientado a objetos, con clases que representan Piloto, Vuelta y Clasificación. Los servicios asociados se encargan de manejar la lógica de negocio, como la actualización del orden de los pilotos, el cálculo de los puntos y la generación de métricas de desempeño. Paralelamente, se aplicaron principios de programación funcional para el procesamiento de datos, utilizando funciones puras que calculan promedios, tiempos más rápidos y deltas de rendimiento entre pilotos, reduciendo el acoplamiento y mejorando la claridad del código.

El canal de comunicación en tiempo real se implementó con SSE (Server-Sent Events), que permite al backend emitir eventos de manera continua hacia el frontend sin requerir peticiones repetidas. Cada evento incluye la información más reciente de la carrera con la posición, tiempo de vuelta y puntos, el cual es interpretado por la interfaz como una actualización del estado global. Además, el sistema incorpora un módulo de Metrics, encargado de analizar el rendimiento operativo del propio flujo de datos (latencia, TPS y conexiones activas), lo cual refuerza la fiabilidad y observabilidad del sistema durante la ejecución.

Por último, el frontend se construyó con React.js, aplicando el paradigma reactivo para garantizar una experiencia de usuario fluida. A través de *hooks* personalizados y un contexto de estado global, la interfaz se mantiene suscrita al flujo de eventos SSE. Esto permite que los componentes del Leaderboard y del módulo Metrics se actualicen

automáticamente al recibir nueva información, sin necesidad de recargar la página. Se priorizó una presentación visual clara, con enfoque en la legibilidad, contraste adecuado y estados definidos para conexión, carga y reconexión.

En conjunto, el diseño técnico integra la reactividad del frontend con la modularidad y el procesamiento funcional del backend, alcanzando un equilibrio entre simplicidad, eficiencia y escalabilidad. Este enfoque permite mantener un flujo continuo de datos de Fórmula 1, con baja latencia y visualización instantánea, cumpliendo los requerimientos del proyecto tanto a nivel técnico como académico.

Descripción de la implementación

La implementación del F1 Data Analytics Dashboard se llevó a cabo siguiendo una estructura modular que separa las responsabilidades entre el backend multiparadigma, el canal de comunicación en tiempo real (SSE) y el frontend reactivo. Cada componente cumple un rol específico para garantizar que los datos de Fórmula 1 pilotos, vueltas, posiciones y puntos se transmitan y visualicen con baja latencia y de forma coherente.

En el backend, desarrollado con Node.js y TypeScript, se aplicó un enfoque orientado a objetos para representar las entidades del dominio, como Piloto y Vuelta, y un modelo de datos temporal que almacena las clasificaciones en memoria. La lógica de negocio se organiza en servicios especializados que gestionan la simulación de carreras, el cálculo de puntuaciones y la actualización de posiciones. Paralelamente, se incorporaron elementos de programación funcional para realizar cálculos analíticos, como el promedio de tiempos de vuelta o la identificación de la mejor vuelta por piloto, empleando funciones puras y operaciones inmutables que garantizan resultados deterministas y fáciles de mantener.

La comunicación entre el servidor y la interfaz se implementó mediante Server-Sent Events (SSE), un canal unidireccional que transmite eventos continuos desde el backend hacia el frontend. Este flujo se mantiene activo durante toda la sesión y se reactiva automáticamente en caso de desconexión. A través de este mecanismo, el servidor emite actualizaciones periódicas con la información más reciente del Leaderboard, las métricas del sistema y el estado de la conexión, permitiendo una experiencia verdaderamente en

tiempo real. Adicionalmente, se desarrolló un módulo de Metrics que evalúa el rendimiento del sistema en ejecución, calculando la latencia promedio y la tasa de mensajes por segundo (TPS), con el fin de verificar la estabilidad del flujo de datos durante la demostración.

El frontend, construido en React.js, implementa la programación reactiva mediante hooks personalizados que se suscriben al canal SSE. Cada actualización emitida por el servidor se refleja automáticamente en la interfaz, modificando el estado global y disparando un render inmediato de los componentes afectados. El módulo Leaderboard muestra la clasificación en vivo con la posición, el piloto, el equipo, el tiempo de vuelta y los puntos, mientras que el módulo Metrics visualiza la latencia y el rendimiento operativo del sistema. Asimismo, se desarrolló una página de Admin Ingest, que permite crear pilotos y simular eventos de carrera, facilitando las pruebas y la presentación del proyecto.

Durante el desarrollo, se priorizó la consistencia visual y funcional del sistema. La interfaz gestiona explícitamente estados de carga, error, reconexión y ausencia de datos, garantizando una experiencia estable y comprensible. Todo el código se documentó y versionó mediante GitHub, manteniendo control de cambios y asegurando trazabilidad. El resultado final es una aplicación funcional, fluida y técnicamente sólida, que demuestra la integración exitosa de paradigmas de programación en un entorno de streaming de datos en tiempo real.

Lecciones aprendidas y dificultades encontradas

Durante el desarrollo del F1 Data Analytics Dashboard se comprendió de forma práctica cómo los distintos paradigmas de programación pueden integrarse en un mismo proyecto para resolver un problema real. La programación reactiva demostró su eficacia para manejar interfaces dinámicas y actualizaciones instantáneas, mientras que los enfoques funcional y orientado a objetos permitieron estructurar la lógica del backend de manera clara, modular y fácil de mantener. Esta combinación facilitó la creación de un sistema fluido y confiable para la visualización de datos en tiempo real.

Entre las principales dificultades se encontraron los retos técnicos asociados a la comunicación en tiempo real, como el manejo de latencia, la reconexión del canal SSE y la sincronización de los datos entre el servidor y la interfaz. También fue necesario ajustar la

frecuencia de emisión de eventos y controlar estados visuales como “sin datos” o “reconectando” para evitar inconsistencias en la experiencia del usuario. A pesar de ello, la experiencia permitió fortalecer el dominio de tecnologías modernas y comprender la importancia de una arquitectura bien definida y adaptable.

Repositorio del proyecto

El código fuente completo del proyecto F1 Data Analytics Dashboard se encuentra disponible públicamente en GitHub, donde se puede acceder a los archivos del frontend, backend y documentación técnica. En este repositorio se incluyen las instrucciones para ejecutar el sistema localmente, las dependencias necesarias y una guía de configuración para la demostración del flujo de datos en tiempo real.

Repositorio: https://github.com/crissXcoder/f1-dashboard.git

Conclusión

El desarrollo del F1 Data Analytics Dashboard permitió demostrar la aplicación práctica de paradigmas de programación en un entorno de datos en tiempo real. A través de la combinación de programación reactiva en el frontend y un backend multiparadigma en Node.js que integra los enfoques funcional y orientado a objetos, se logró construir una plataforma capaz de mostrar de forma continua los tiempos de vuelta, posiciones y puntos de los pilotos, con una experiencia de usuario fluida y confiable.

La implementación del canal Server-Sent Events (SSE) resultó clave para mantener una comunicación estable y eficiente entre el servidor y la interfaz, garantizando baja latencia y actualizaciones automáticas sin recarga. Además, el módulo Metrics aportó un nivel adicional de observabilidad al medir la latencia y el rendimiento del sistema durante la ejecución, reforzando la solidez técnica del proyecto.

En conjunto, este trabajo refleja cómo la integración de paradigmas distintos puede producir soluciones modernas, escalables y orientadas a la eficiencia. Más allá de cumplir con los objetivos académicos, el proyecto representa un ejemplo funcional de ingeniería de software aplicada a la analítica de datos deportivos, evidenciando la importancia del diseño modular, la asincronía controlada y la reactividad como pilares del desarrollo contemporáneo.

Bibliografía

- Fowler, M. (2019). *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional.
- Mozilla Developer Network. (s. f.). *Event loops and microtasks*. *MDN Web Docs*. https://developer.mozilla.org/en-US/docs/Web/API/HTML_DOM_API/Microtask_guide
- Node.js Foundation. (s. f.). *Introduction to Node.js*. *Node.js Documentation*. https://nodejs.org/en/learn/getting-started/introduction-to-nodejs
- React. (s. f.). *React – A JavaScript library for building user interfaces*. Meta Open Source. https://react.dev
- WHATWG. (s. f.). *Server-Sent Events – HTML Living Standard*. *WHATWG Specification*. https://html.spec.whatwg.org/multipage/server-sent-events.html
- Wampler, D. (2014). *Functional Programming for the Object-Oriented Programmer*. Pragmatic Bookshelf.
- Richards, M. (2020). *Software Architecture: The Hard Parts*. O'Reilly Media.
- Mozilla Developer Network. (s. f.). *Using Fetch*. *MDN Web Docs*. https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch
- Google Developers. (s. f.). *Web Performance Fundamentals*. *Web.dev*. https://web.dev/performance