TALLER 2

ISABELLA BLANCO  
JAIRO ANDRÉS SIERRA COMBARIZA

FACULTAD DE INGENIERIA

ARQUITECTURA DE SOFTWARE

PONTIFICIA UNIVERSIDAD JAVERIANA

BOGOTÁ D.C

2024

**Documento de investigación**

Los temas asignados se interrelacionan en la construcción de un sistema web completo. Next.js y React manejan la parte de la interfaz de usuario, mientras que Node.js gestiona la lógica de negocio en el backend. SOAP y XML permiten la integración segura y estructurada entre servicios, y PostgreSQL almacena y gestiona los datos del sistema.

1. **Definición**
   * **Estilo Web Service:** Es una tecnología que permite la comunicación y el intercambio de datos entre aplicaciones o sistemas a través de la web, usando protocolos estandarizados como SOAP y XML. Estos servicios son independientes de la plataforma y del lenguaje de programación, lo que facilita la interoperabilidad.
   * **Framework Frontend Next.js y React:** Next.js es un framework basado en React que facilita la creación de aplicaciones web del lado del servidor y del lado del cliente. Ofrece renderizado híbrido, generación de sitios estáticos y características como enrutamiento sencillo y optimización de rendimiento.
   * **Framework Backend Node.js:** Node.js es un entorno de ejecución para JavaScript que permite desarrollar aplicaciones del lado del servidor. Utiliza un modelo de E/S no bloqueante y orientado a eventos, lo que lo hace eficiente y escalable para aplicaciones en tiempo real.
   * **Base de Datos PostgreSQL:** PostgreSQL es un sistema de gestión de bases de datos relacional y orientado a objetos, conocido por su estabilidad, flexibilidad y capacidad para manejar grandes volúmenes de datos.
   * **Tecnologías y Protocolos SOAP y XML:** SOAP (Simple Object Access Protocol) es un protocolo de mensajería que utiliza XML para la comunicación entre aplicaciones. SOAP permite una comunicación estructurada y segura, ideal para entornos empresariales.
2. **Características**
   * **Web Service:**
     + Interoperabilidad entre aplicaciones.
     + Uso de protocolos estándar (HTTP, SOAP, XML).
     + Permite integración con diversas tecnologías y plataformas.
   * **Next.js y React:**
     + Renderizado del lado del servidor y cliente.
     + Enrutamiento automático y sencillo.
     + Soporte para JavaScript y TypeScript.
     + Optimización de rendimiento y SEO.
   * **Node.js:**
     + Basado en JavaScript para el backend.
     + Escalabilidad y alta concurrencia.
     + Amplia comunidad y ecosistema de paquetes (npm).
   * **PostgreSQL:**
     + Soporte para transacciones ACID.
     + Extensiones avanzadas y soporte para JSON.
     + Seguridad y robustez.
   * **SOAP y XML:**
     + Mensajería estructurada y segura.
     + Soporte para servicios web distribuidos.
     + Ideal para comunicaciones en entornos empresariales.
3. **Historia y evolución**
   * Web Service:
     + Surgieron a principios de los 2000 para facilitar la interoperabilidad entre aplicaciones, especialmente en un contexto donde diferentes sistemas necesitaban comunicarse de manera estándar.
     + El concepto de Web Services fue impulsado principalmente por la W3C y OASIS, quienes desarrollaron estándares clave como SOAP, WSDL (Web Services Description Language) y UDDI (Universal Description, Discovery, and Integration).
     + Inicialmente, SOAP fue el protocolo dominante, pero debido a su complejidad y a la rigidez de XML, muchos desarrolladores comenzaron a preferir servicios RESTful basados en HTTP y JSON por su simplicidad y rendimiento.
     + La evolución de los Web Services llevó al desarrollo de microservicios, que permitieron crear aplicaciones más modulares y escalables. Esta transición marcó un cambio hacia arquitecturas orientadas a servicios más ligeras y fácilmente integrables.
   * **Next.js y React:**
     + React (2013): Lanzado por Facebook, revolucionó el desarrollo frontend introduciendo un enfoque basado en componentes y un Virtual DOM que mejora significativamente el rendimiento en la renderización.
     + React se destacó rápidamente por su capacidad de construir interfaces de usuario dinámicas y de alta calidad, promoviendo el concepto de “aprende una vez, escribe en cualquier lugar” para crear aplicaciones web, móviles, y de escritorio.
     + Next.js (2016): Creado por Vercel, nació como un framework para extender las capacidades de React, permitiendo renderizado del lado del servidor (SSR) y generación de sitios estáticos (SSG), lo que mejoró el SEO y la velocidad de carga de las aplicaciones.
     + Next.js ha evolucionado continuamente, añadiendo soporte para TypeScript, API Routes, y funciones de prerenderización híbrida, posicionándose como una de las herramientas más poderosas para desarrolladores JavaScript.
   * **Node.js:**
     + Lanzado en 2009 por Ryan Dahl, Node.js fue innovador al permitir que JavaScript, un lenguaje originalmente diseñado para el frontend, pudiera ejecutarse del lado del servidor.
     + Su arquitectura basada en el motor V8 de Google permitió un alto rendimiento y el uso eficiente de recursos gracias a su modelo asíncrono y no bloqueante, ideal para aplicaciones en tiempo real.
     + Node.js popularizó el uso de microservicios y APIs RESTful debido a su capacidad para manejar múltiples conexiones simultáneamente sin sobrecargar el servidor.
     + Ha evolucionado con una comunidad robusta y un ecosistema extenso de paquetes (npm), que lo han consolidado como una de las opciones más versátiles para desarrollo backend.
   * **PostgreSQL:**
     + Surgió en la década de 1980 como una continuación del proyecto Ingres, bajo la dirección de Michael Stonebraker en la Universidad de California, Berkeley.
     + PostgreSQL, lanzado oficialmente en 1989 como Postgres y más tarde renombrado en 1996, fue uno de los primeros en ofrecer características avanzadas como transacciones ACID, integridad referencial, y soporte para lenguajes procedurales.
     + Ha sido constantemente actualizado para incluir soporte para JSON, replicación en streaming, y una extensibilidad sin precedentes, lo que le ha permitido competir con bases de datos NoSQL y mantenerse relevante en un mercado que cambia rápidamente.
     + Es ampliamente reconocido por su estabilidad, flexibilidad, y la capacidad de manejar operaciones complejas, siendo la elección preferida para aplicaciones críticas en finanzas, ciencia, y análisis de datos.
   * **SOAP y XML:**
     + SOAP (1998): Desarrollado por Microsoft, SOAP se convirtió en un protocolo estándar para la comunicación entre aplicaciones, utilizando XML como formato para la estructuración de datos.
     + XML (eXtensible Markup Language) se popularizó a finales de los 90 como un formato de datos estructurado, altamente legible por humanos y máquinas, facilitando la transferencia de datos entre sistemas heterogéneos.
     + SOAP se adoptó rápidamente en entornos corporativos debido a sus características de seguridad, como WSSecurity, que garantizan la integridad y confidencialidad de los mensajes.
     + Aunque SOAP y XML comenzaron a ser desplazados por tecnologías más ligeras como REST y JSON, siguen siendo indispensables en industrias como la financiera, donde la seguridad y la estructura rigurosa de los datos son esenciales.
4. **Ventajas y Desventajas**
   * **Web Service:**
     + Ventajas: Interoperabilidad, seguridad, escalabilidad.
     + Desventajas: Complejidad en la implementación, rendimiento comparado con REST.
   * **Next.js y React:**
     + Ventajas: SEO, rendimiento, flexibilidad.
     + Desventajas: Curva de aprendizaje, mayor configuración inicial.
   * **Node.js:**
     + Ventajas: Escalabilidad, rapidez en el desarrollo, reutilización de código.
     + Desventajas: No adecuado para tareas de CPU intensivas.
   * **PostgreSQL:**
     + Ventajas: Seguridad, extensibilidad, manejo de datos no estructurados.
     + Desventajas: Configuración compleja para altas cargas.
   * **SOAP y XML:**
     + Ventajas: Seguridad, estandarización.
     + Desventajas: Verbosidad, mayor latencia.
5. **Casos de uso**
   * **Web Service:** 
     + Integración de sistemas empresariales, como CRM y ERP.
     + Aplicaciones de banca y finanzas que requieren alta seguridad.
     + Empresas como Amazon y Google usan servicios web para integrar diferentes plataformas.
   * **Next.js y React:** 
     + Aplicaciones de comercio electrónico con SEO optimizado.
     + Dashboards y aplicaciones SPA (Single Page Application).
     + Lo usan en aplicaciones de Netflix y Hulu.
   * **Node.js:** 
     + Aplicaciones en tiempo real como chats y juegos en línea.
     + APIs RESTful y microservicios.
     + Plataformas como LinkedIn y Uber usan Node.js para sus servicios backend.
   * **PostgreSQL:** 
     + Aplicaciones que manejan grandes volúmenes de datos, como analíticas y big data.
     + Plataformas de comercio electrónico y sistemas de gestión de contenido.
     + Utilizado por empresas como Instagram y Apple.
   * **SOAP y XML:** 
     + Servicios empresariales donde la seguridad y la estructura del mensaje son cruciales, como en seguros y finanzas.
     + Microsoft y Oracle implementan SOAP en sus soluciones empresariales.
6. **¿Qué tan Común es nuestro Stack Designado?**

Este stack es relativamente común en aplicaciones empresariales que requieren integración segura y estructura de datos compleja. La combinación de Next.js, React y Node.js se ha vuelto popular en la construcción de aplicaciones modernas y escalables.

1. **Matrices de Análisis**
   * **Matriz de principios SOLID vs Temas asignados**

**Tabla

Descripción generada automáticamente con confianza media**

* + **Matriz de atributos de calidad vs Temas asignados**

**Tabla

Descripción generada automáticamente**

* + **Matriz de tácticas vs Temas asignados**

**Tabla

Descripción generada automáticamente**

* + **Matriz de mercado laboral vs temas asignados**

**Tabla

Descripción generada automáticamente**

* + **Matriz de patrones laborales vs Temas asignados**

**Tabla

Descripción generada automáticamente**

1. **Ejemplo práctico y funcional relacionando los temas:**

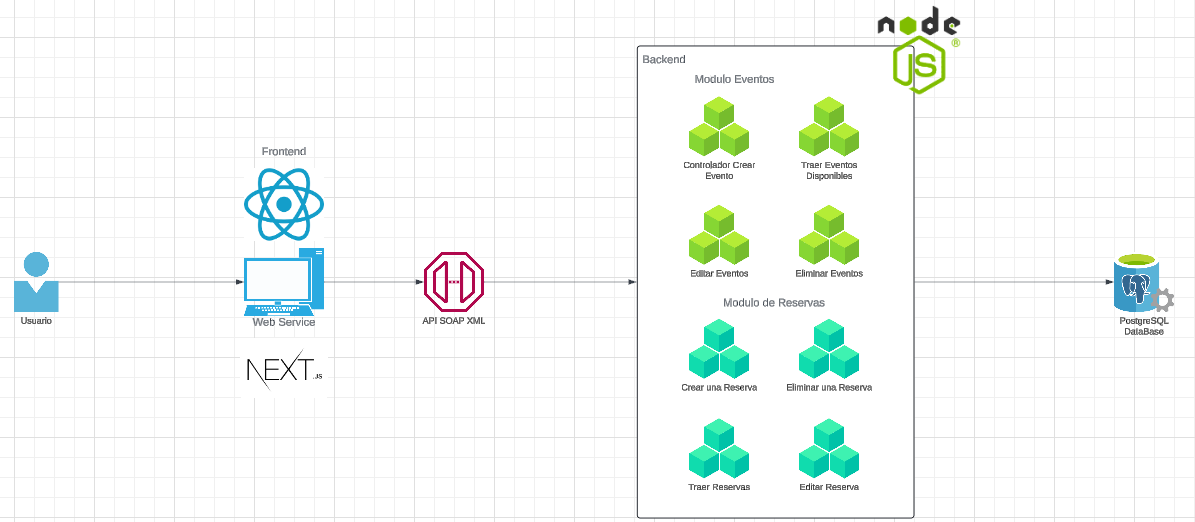
El código desarrollado es un sistema de Gestión de Reservas para Eventos es una plataforma diseñada para facilitar la organización, administración y gestión de eventos, permitiendo a los usuarios registrar sus reservas de manera eficiente. El sistema incluye un frontend desarrollado con Next.js y React, que proporciona una interfaz interactiva y fácil de usar, donde los usuarios pueden crear, editar y eliminar eventos, así como gestionar sus reservas, eliminarlas o actualizarlas. El backend está implementado en Node.js, utilizando un servicio SOAP con XML para la comunicación entre el frontend y el backend, manejando todas las solicitudes relacionadas con la creación de eventos, la gestión de reservas y la consulta de datos almacenados en la base de datos.

La base de datos utilizada es PostgreSQL, donde se almacena toda la información de los eventos, reservas y usuarios registrados. Esta arquitectura asegura una separación clara de responsabilidades entre la lógica de negocio, el manejo de eventos y la gestión de reservas, lo que mejora la escalabilidad y el mantenimiento del sistema.

El proyecto se centra en proporcionar un sistema robusto y escalable para la gestión de eventos y reservas, utilizando tecnologías modernas en el desarrollo web, aunque no tan comunes entre sí, asegurando una comunicación fluida entre el cliente y el servidor a través de servicios web estructurados con SOAP.

**Diagrama de Arquitectura de alto nivel:**

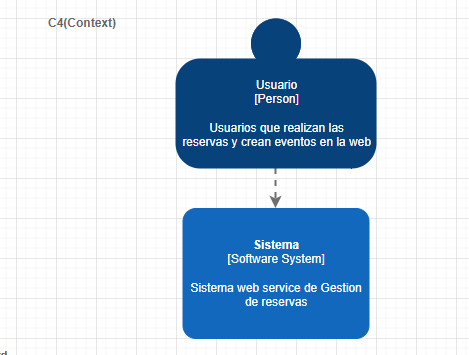
Representa una visión general del sistema, mostrando los principales componentes y su interacción, incluyendo el frontend, backend y base de datos, así como la integración para el web service con SOAP/XML.

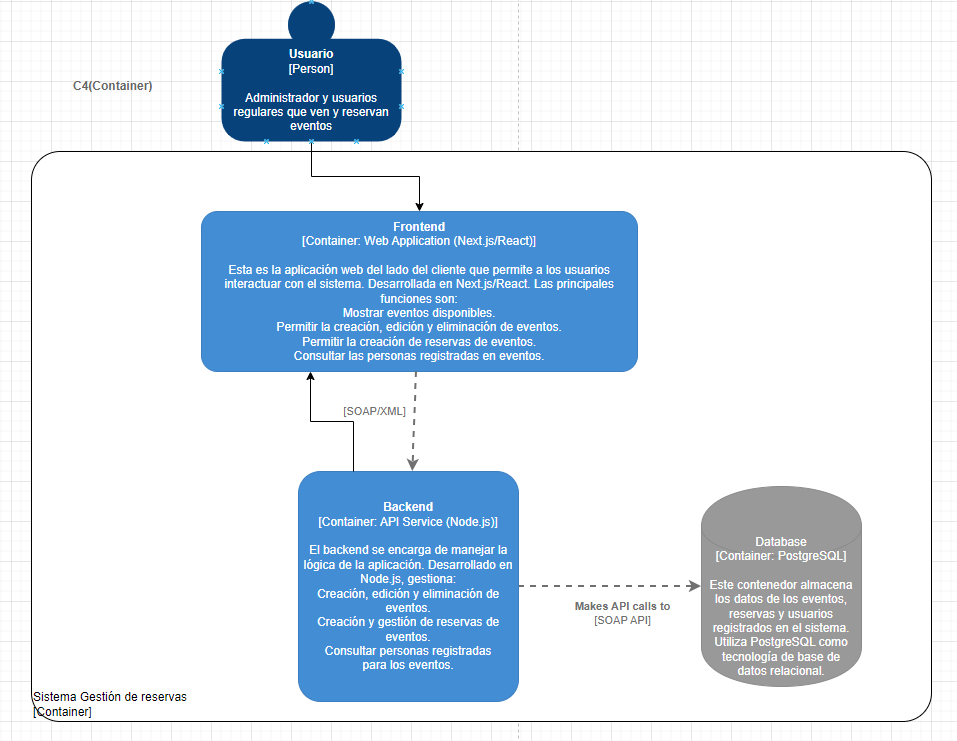


**C4 model:**

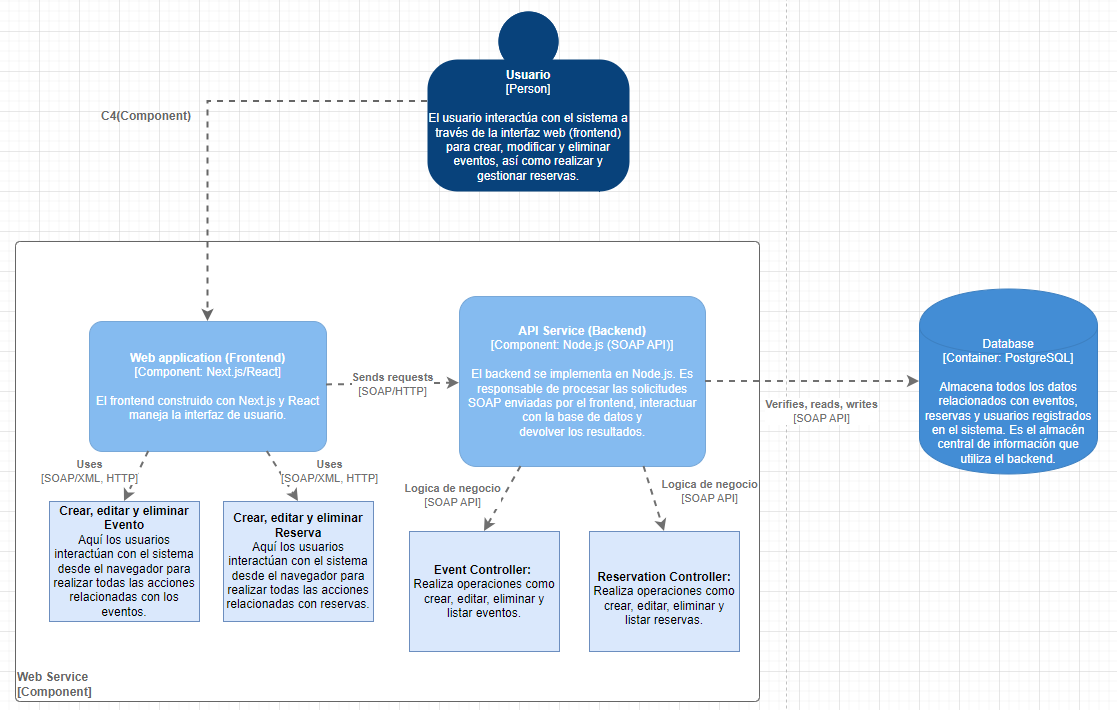
Es un marco que descompone el sistema en niveles de abstracción, desde el contexto general hasta los componentes detallados y sus interacciones en este caso de nuestro sistema de gestión de reservas.

**C4 Context:** Muestra una visión general del sistema dentro de su entorno, identificando los actores principales (usuario) y las interacciones de alto nivel con el sistema.

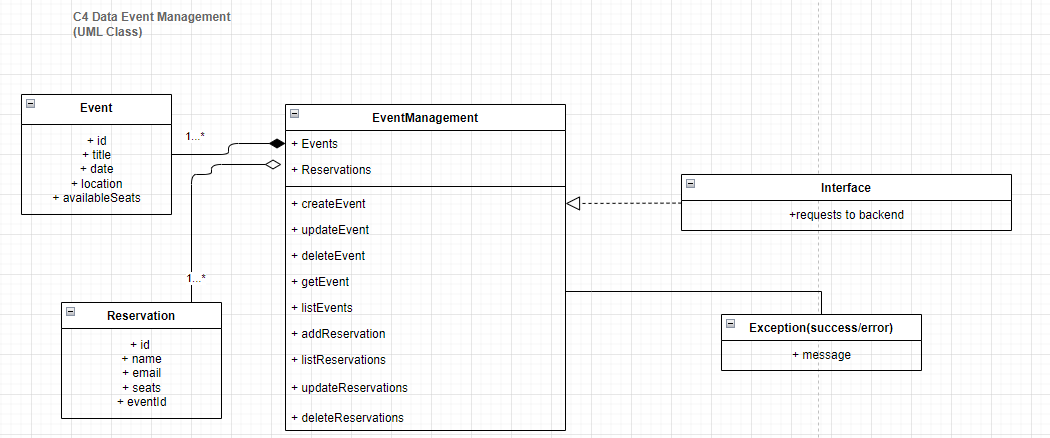


**C4 Container**: Representa los contenedores del sistema (frontend, backend, base de datos), mostrando cómo se comunican entre sí y qué tecnologías utilizan.

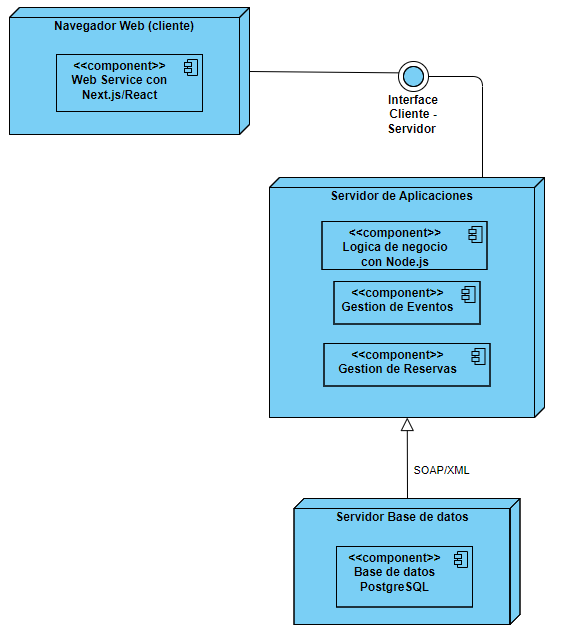
**C4 Component**: Detalla los componentes internos de los contenedores, incluyendo los módulos de lógica de negocio, controladores y puntos de integración dentro del backend.



**C4 Class UML**: Desglosa las relaciones entre clases y objetos dentro del sistema, mostrando los atributos y métodos principales para la gestión de eventos y reservas.



**C4 Despliegue**: Muestra la infraestructura física y lógica del sistema, detallando cómo los diferentes componentes (frontend, backend y base de datos) se despliegan en servidores o nodos específicos, y cómo interactúan entre sí a nivel de red y servicios.



1. **Ejemplo práctico y funcional**
   * **Código fuente en GitHub:** 
     + **Front :** <https://github.com/Jairo-Andres/Taller2_ArquisoftFront>
     + **Back:** <https://github.com/Jairo-Andres/Taller2_ArquisoftBack>
   * **Video del demo (con uso de postman al final):** <https://youtu.be/ykNY49uMCLs>