SGH: Sistema de Gestión de Horarios

Martin Stiben Narvez
Servicio Nacional de Aprendizaje (SENA), Regional Huila,
Centro de la Industria, la Empresa y los Servicios (CIES),
Ficha 2899747
Neiva, Colombia
msnarvaez21@soy.sena.edu.co

Jesus Ariel González Bonilla
Servicio Nacional de Aprendizaje (SENA), Regional Huila
Neiva, Colombia

Resumen

SGH es un proyecto para gestionar horarios escolares de forma sencilla y eficiente. Está pensado para simplificar la organización de los tiempos en instituciones educativas, aprovechando tecnologías modernas que garantizan un funcionamiento ágil y confiable. Cuenta con una aplicación web desarrollada con Next.js, un backend robusto creado en Java con Spring Boot, y una app móvil hecha en React Native para Android, para que estudiantes y profesores puedan consultar sus horarios en cualquier momento y desde dispositivos Android. Además, su arquitectura está diseñada de manera modular, lo que permite que el sistema crezca con facilidad y sea más fácil de mantener y mejorar con el tiempo.

Palabras clave: Gestión de horarios; Metodologías ágiles; Historias de usuario; Arquitectura modular; Spring Boot; Next.js; React Native; Aplicación web; Aplicación móvil.

Keywords

Gestión de horarios, Metodologías ágiles, Historias de usuario, Arquitectura modular, Spring Boot, Next.js, React Native, Aplicación web, Aplicación móvil

1. Introducción

Gestionar los horarios en un colegio o universidad es uno de esos desafíos que, si se resuelven bien, hacen que todo funcione mejor. Se trata de organizar las clases de manera inteligente: que no se solapen, que las aulas estén bien aprovechadas y que los profesores y estudiantes puedan concentrarse en lo importante, sin perder tiempo con planillas interminables. Los sistemas automatizados son clave aquí. No solo resuelven conflictos de horarios casi al instante, sino que permiten consultas rápidas y actualizaciones en tiempo real desde dispositivos Android o la web. Hoy en día, contar con una herramienta accesible desde el celular Android o la web no es un lujo, es una necesidad. Pero crear un sistema así no es sencillo. Hay que tener en cuenta muchos detalles: la disponibilidad de cada profesor, las preferencias de horario, además, que sea fácil de usar y seguro. Por eso SGH es una solución desarrollada para enfrentar estos retos de frente. Un sistema pensado para simplificar la gestión de horarios, usando tecnologías modernas que garantizan flexibilidad, seguridad y una experiencia intuitiva para todos.

2. Marco teórico y trabajos relacionados

2.1. Gestión de Horarios Académicos y Algoritmos de Optimización

Imagina que eres un director de colegio y tienes que armar el horario de clases para cientos de estudiantes y profesores. Antes, esto se hacía con papel y lápiz, anotando manualmente quién da qué clase, cuándo y dónde, lo que tomaba días y generaba errores como superposiciones o aulas ocupadas al mismo tiempo. Hoy, sistemas como SGH automatizan esto usando algoritmos que resuelven estos conflictos de manera inteligente.

Un sistema de gestión de horarios académicos, como SGH, combina herramientas simples para registrar cursos, asignaturas, profesores y sus horarios disponibles, con algoritmos que generan horarios evitando choques. Es como un rompecabezas gigante donde cada pieza (clase, profesor, aula) debe encajar sin solaparse. En la práctica, soluciones similares usan algoritmos de optimización para resolver estos problemas, demostrando que es posible hacer esto de forma eficiente en entornos educativos [**?**]. Comparado con métodos manuales tradicionales, que dependen de la intuición humana y son propensos a errores, SGH ofrece una alternativa automatizada que ahorra tiempo y reduce frustraciones.

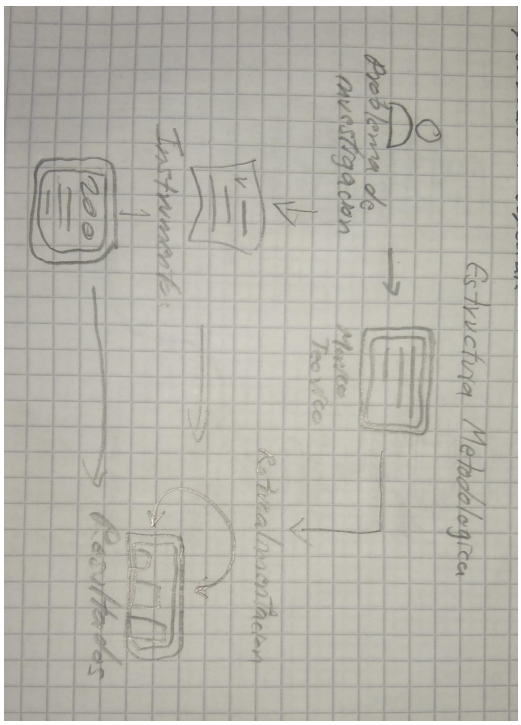

Figura 1: Estructura metodológica para la gestión de horarios académicos.

2.2. Especificación de Requerimientos en Entornos Ágiles

Cuando desarrollas software, necesitas saber exactamente qué quieres construir. En enfoques tradicionales, esto se hace con documentos largos y detallados que describen cada función paso a paso, como un manual de instrucciones. Pero en entornos ágiles, como el usado en SGH, se prefiere algo más flexible: historias de usuario.

Las historias de usuario son como pequeñas anécdotas que cuentan qué necesita el usuario. Según Izaurralde (2013), los requerimientos se dividen en funcionales (lo que hace el sistema) y no funcionales (como rapidez o seguridad). La ingeniería de requerimientos implica recopilar, analizar y validar estas necesidades, asegurando que sean correctas, completas y fáciles de cambiar si algo cambia.

En SGH, usamos historias de usuario porque permiten adaptar el sistema a medida que aprendemos más sobre las necesidades reales de los usuarios, en lugar de atarnos a un plan rígido desde el inicio. Comparado con especificaciones tradicionales, que pueden ser abrumadoras y difíciles de actualizar, las historias de usuario hacen el proceso más conversacional y humano, facilitando la colaboración entre desarrolladores y usuarios.

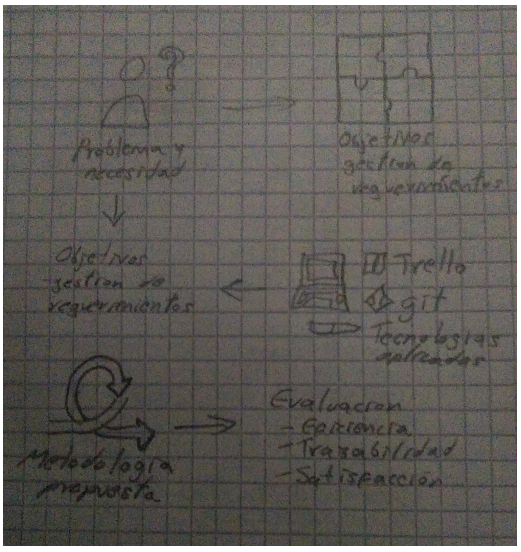

Figura 2: Ejemplo de historia de usuario en entornos ágiles.

2.3. Metodologías Ágiles: Scrum en SGH

Scrum es como un juego de equipo donde divides el trabajo en rondas cortas llamadas sprints, cada una de unas semanas, para construir el software poco a poco. En SGH, adoptamos Scrum porque permite responder rápidamente a cambios, como cuando un profesor pide ajustar su disponibilidad.

En Scrum, los requerimientos se especifican con historias de usuario, que son simples y enfocadas en el valor para el usuario [?]. A diferencia de métodos tradicionales que requieren planear todo al detalle antes de empezar, Scrum permite empezar con lo básico y refinar en cada sprint mediante reuniones diarias y retrospectivas. Esto hace que el desarrollo sea más adaptable y menos estresante.

Comparado con enfoques waterfall (donde todo se planea de una vez), Scrum en SGH es como construir una casa habitación por habitación en lugar de diseñar todo el plano primero: puedes ajustar si encuentras un problema, y el resultado final se siente más vivo y útil.

2.4. Tecnologías Backend: Java con Spring Boot

El corazón de SGH es su backend, construido con Java y Spring Boot. Java es como un lenguaje confiable y maduro, usado en muchas aplicaciones grandes porque es rápido, seguro y funciona en cualquier máquina. Spring Boot simplifica el desarrollo al proporcionar herramientas listas para usar, como manejar bases de datos o seguridad.

En SGH, usamos JPA para conectar el código con la base de datos de forma automática, y Spring Security con JWT para autenticar usuarios de manera segura. Comparado con otros frameworks como Node.js o Python Django, Spring Boot ofrece más estabilidad para aplicaciones complejas, aunque puede ser un poco más pesado al inicio. Pero para un sistema como SGH, que maneja datos sensibles de horarios, esta robustez es clave.

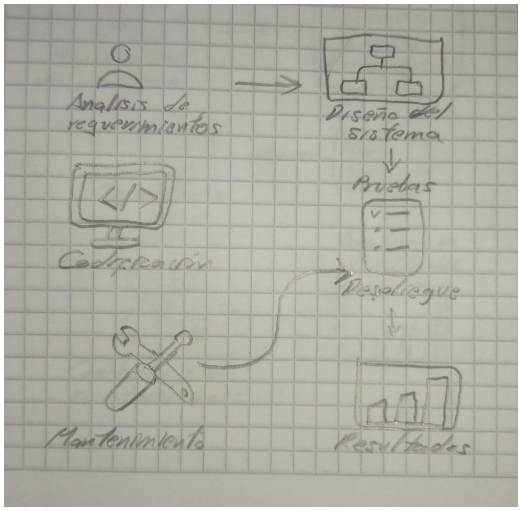

Figura 3: Tecnologías Java y nuevas innovaciones en desarrollo backend.

2.5. Bases de Datos: MySQL en SGH

Las bases de datos son como el archivador digital donde guardas toda la información. MySQL es una opción popular porque es gratuita, rápida y fácil de usar, ideal para proyectos como SGH que necesitan almacenar relaciones complejas, como qué profesor da qué clase en qué aula [?].

MySQL surgió en los 90 como una evolución de sistemas más antiguos, y hoy es una herramienta clave para datos estructurados. Comparado con bases de datos no relacionales como MongoDB, que son más flexibles para datos irregulares pero pueden ser menos

consistentes, MySQL garantiza que todo esté bien organizado con el modelo ACID [?]. En SGH, optamos por MySQL porque nuestros datos (cursos, profesores, horarios) tienen relaciones claras que necesitan integridad, y lo ejecutamos en XAMPP para desarrollo local, con phpMyAdmin para gestionar todo sin complicaciones.

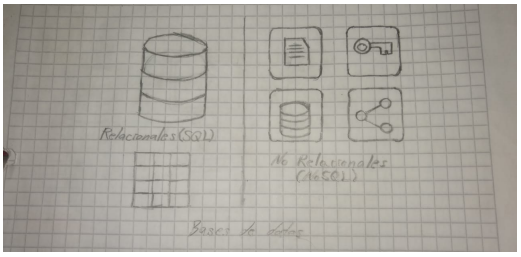

Figura 4: Comparación entre bases de datos SQL y NoSQL.

2.6. Interfaces Web y Móviles: Frameworks para Android

SGH no es solo un sitio web; también tiene una app móvil para Android. Para lograr esto, usamos React Native, que permite desarrollar una app nativa para Android con un solo código base. En el frontend web, Next.js con Tailwind CSS crea interfaces modernas y responsivas.

Comparado con desarrollo nativo puro (escribir código específico para Android), React Native ahorra tiempo y dinero, aunque a veces requiere ajustes para un rendimiento óptimo. En SGH, esto significa que estudiantes y profesores pueden acceder a sus horarios desde dispositivos Android, haciendo el sistema más accesible y práctico.

2.7. Arquitectura Modular y Microservicios

La arquitectura de SGH es modular, como construir con bloques Lego: cada parte (backend, frontend, móvil) es independiente, pero encaja perfectamente. Esto facilita mantener y expandir el sistema, como agregar nuevas funciones sin romper lo existente.

Comparado con arquitecturas monolíticas (todo en un bloque grande), la modularidad en SGH permite actualizaciones más seguras y escalabilidad. La coexistencia de requerimientos tradicionales con historias de usuario asegura que tengamos lo mejor de ambos mundos: precisión y flexibilidad [?].

2.8. APIs RESTful y Autenticación JWT

Las APIs son como puentes que conectan las partes de SGH. Usamos RESTful para intercambiar datos de forma segura y eficiente, con JWT para autenticar usuarios sin guardar sesiones en el servidor [?].

Comparado con APIs más antiguas, REST es simple y escalable, ideal para apps móviles. En SGH, documentamos todo con Swagger para que sea fácil probar y entender, mejorando la colaboración entre equipos.

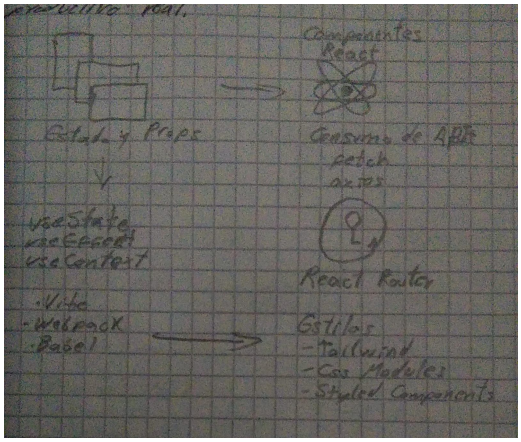

Figura 5: Ejemplo de APIs RESTful en sistemas distribuidos.

2.9. Docker: Contenerización para Despliegue

Docker es como empaquetar SGH en una caja portable que funciona igual en cualquier máquina. Resuelve problemas de compatibilidad, reduciendo tiempos de despliegue de días a horas (Chamú, 2025).

Comparado con máquinas virtuales, que simulan computadoras completas, Docker es más ligero y eficiente, perfecto para orquestar servicios en SGH sin complicaciones.

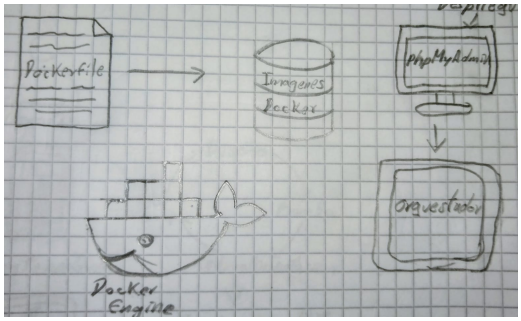

Figura 6: Contenerización con Docker para despliegue eficiente.

3. Metodología de investigación aplicada

3.1. Enfoque de desarrollo

Para la gestión del proyecto se utilizó una metodología ágil inspirada en Scrum, con iteraciones semanales y priorización de funcionalidades, similar a prácticas en desarrollo con Java y Spring Boot [?]. Los requerimientos funcionales incluyeron: registro de cursos, asignaturas, profesores y disponibilidades; generación automática de horarios; visualización y consulta de horarios; y exportación a PDF, Excel e imágenes. Estos requerimientos se especificaron mediante historias de usuario siguiendo el modelo INVEST

(Independientes, Negociables, Valiosas, Estimables, Pequeñas y Verificables), permitiendo una definición iterativa y adaptable [?]. Requerimientos no funcionales: seguridad en autenticación, usabilidad en interfaces y escalabilidad para múltiples usuarios.

La arquitectura seleccionada fue modular, con el backend en Spring Boot actuando como núcleo, el frontend web en Next.js para administración y el móvil en React Native para consultas. Esto permite separación de responsabilidades y facilidad de mantenimiento.

Desarrollo del backend (Spring Boot): Se implementó siguiendo el patrón MVC (Model-View-Controller), con controladores REST para CRUD de entidades (cursos, profesores, asignaturas, horarios), servicios de negocio y repositorios JPA. La generación automática de horarios utiliza algoritmos simples de asignación basados en disponibilidades. Se integró Spring Security con JWT para autenticación en el inicio de sesión. Servicios de exportación generan PDFs con iTextPDF, Excels con Apache POI y gráficos con JFreeChart.

Desarrollo de la aplicación web (Next.js): Construida con TypeScript, utiliza Axios para consumir la API REST. Incluye dashboards para gestión de entidades, generación de horarios y visualización con gráficos de Recharts. La interfaz es responsiva con Tailwind CSS y estilos personalizados en CSS.

Desarrollo de la aplicación móvil (React Native): Desarrollada con Expo para Android, permite login y consulta de horarios por curso o profesor. Utiliza navegación con React Navigation y almacenamiento local con AsyncStorage.

Pruebas e integración: Se realizaron pruebas unitarias en el backend con JUnit, pruebas manuales en interfaces y pruebas de integración end-to-end. Las APIs están documentadas con Swagger (OpenAPI) para facilitar el desarrollo y pruebas. La base de datos MySQL maneja persistencia, con Docker para orquestación en desarrollo.

4. Implementación del software

Describimos arquitectura, decisiones tecnológicas y *pipelines*. Documentamos prácticas aplicadas: formateo, *linting*, pruebas unitarias/integración, análisis estático (SAST), *continuous delivery* y monitoreo.

4.1. Arquitectura del sistema

La arquitectura implementada sigue las mejores prácticas de DevOps [?], integrando automatización en todo el ciclo de desarrollo.

4.2. Comparación de tecnologías

La selección de tecnologías se basó en criterios objetivos. La tabla 1 presenta una comparación detallada de los frameworks evaluados.

Tabla 1: Tecnologías Utilizadas en SGH

Tecnología	Lenguaje	Uso	Ventajas
Spring Boot	Java	Backend	Alta escalabilidad, seguridad robusta
Next.js	JS/TS	Frontend Web	Rendimiento optimizado, SSR
React Native	JavaScript	Móvil	Desarrollo cross-platform
MySQL	SQL	BD	Integridad de datos, relaciones complejas

5. Evaluación y resultados

Aplicación web funcional: Desarrollada en Next.js, permite gestión completa de entidades, generación de horarios y exportaciones. Incluye autenticación segura y dashboards intuitivos.

Backend robusto: En Spring Boot, maneja lógica de negocio, generación y exportaciones. Pruebas muestran rendimiento adecuado para instituciones medianas.

Aplicación móvil: En React Native para Android, facilita consultas rápidas de horarios, mejorando accesibilidad para estudiantes y profesores.

Generación automática: Algoritmos asignan horarios evitando conflictos, con historial de generaciones.

Exportaciones: Horarios exportables a PDF, Excel e imágenes, útiles para impresión y distribución.

Seguridad: JWT implementado, con roles de usuario (admin, coordinador).

6. Discusión

La elección de tecnologías resultó acertada para el alcance. Spring Boot proporcionó estabilidad, Next.js rapidez en desarrollo web y React Native para Android móvil. La arquitectura modular facilita expansiones futuras.

Comparado con soluciones existentes, SGH ofrece integración web-móvil y exportaciones variadas, diferenciándose de sistemas puramente desktop.

Limitaciones incluyen algoritmos de generación simples; futuras versiones podrían integrar optimización avanzada.

7. Conclusiones y trabajo futuro

SGH cumple objetivos de gestión eficiente de horarios, con integración web-móvil y tecnologías modernas. Facilita administración educativa, reduciendo tareas manuales.

Trabajos futuros: Mejorar algoritmos de generación, crear un tipo de foro para el colegio y escalar a más instituciones.

A. Descripción de componentes del sistema SGH

Aplicación web (Next.js): Interfaz para administración, generación y visualización de horarios.

Servidor backend (Spring Boot): API REST para lógica de negocio y persistencia.

Aplicación móvil (React Native): Consulta de horarios en dispositivos Android.

Base de datos (MySQL): Almacenamiento de datos de cursos, profesores, horarios.

Referencias

A. Y. Arciniegas. 2025. *Practicante en Lenguaje de Programación Java y Nuevas Tecnologías.* Universidad Tecnológica de Pereira.

J. Cein Villanueva, E. Gómez Domínguez, P. Zamora Reséndiz, A. Priego Clemente, and E. Landero García. 2025. APIs RESTful para interoperabilidad entre aplicación móvil y aplicación web. *CL-RCM* 9, 3 (2025). doi:10.37811/cl_rcm.v9i3.18325

M. P. Izaurralde. 2013. *Caracterización de Especificación de Requerimientos en entornos Ágiles: Historias de Usuario.* Universidad Tecnológica Nacional.

J. M. Lozano. 2018. *Creación y Gestión de una Base de Datos con MySQL y phpMyAdmin.* Universidad de Jaén.

L. A. Saltos. 2022. *Estudio Comparativo entre Bases de Datos Relacional y No Relacional.* Universidad Técnica de Babahoyo.