

SGH: Sistema de Gestión de Horarios

Autor: Juan Pablo Saavedra Chambo

Afiliaciones: Servicio Nacional de Aprendizaje (SENA), Regional Huila,
Centro de la Industria, la Empresa y los Servicios (CIES), Ficha 2899747.

Contacto: jpsaavedra32@soy.sena.edu.co

Fecha: 2025

Resumen

El Sistema de Gestión de Horarios (SGH) es una plataforma integral desarrollada para optimizar la administración de horarios en instituciones educativas, empresariales y gubernamentales. El sistema ofrece interfaces web y móviles, permitiendo la gestión eficiente de maestros, cursos, asignaturas y horarios mediante tecnologías modernas como Java y frameworks asociados. Este proyecto aborda la necesidad de una planificación organizada de horarios, reduciendo conflictos y mejorando la eficiencia operativa. La arquitectura se basa en principios de desarrollo ágil, con énfasis en la seguridad, compatibilidad multiplataforma y tiempos de respuesta rápidos, conforme a los requerimientos especificados en el SRS.

Palabras clave: 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

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.

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.

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 (Izaurre, 2013). 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.

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 (Lozano, 2018).

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 (Saltos, 2022). 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.

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), el 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 (Izaurre, 2013).

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 (Cein Villanueva et al., 2025).

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.

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.

3 Métodos

A continuación, se detalla cómo se diseñó e implementó el proyecto SGH, incluyendo las herramientas utilizadas, la arquitectura seleccionada y las pruebas realizadas. Se buscó aplicar un enfoque práctico, siguiendo metodologías ágiles para la gestión del proyecto.

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. Los requerimientos funcionales del SRS incluyeron: inicio de sesión (RF1), visualización de horarios de maestros o cursos (RF2), gestión de maestros (RF3), gestión de cursos (RF4), gestión de asignaturas (RF5) y gestión de horarios (RF6). Estos se especificaron mediante historias de usuario siguiendo el modelo INVEST. Requerimientos no funcionales: seguridad en el acceso al sistema (RNF1), cifrado de datos sensibles (RNF2), tiempos de respuesta del sistema (RNF3), compatibilidad con múltiples dispositivos (RNF4), soporte técnico y documentación (RNF5), y cumplimiento de normativas legales (RNF6).

La arquitectura seleccionada fue modular, con backend en Java con Spring Boot, frontend web y móvil en React Native. Esto permite separación de responsabilidades.

Desarrollo del backend: Patrón MVC, controladores REST para CRUD, servicios de negocio y repositorios JPA. Autenticación con JWT. Gestión de entidades como maestros, cursos, asignaturas y horarios.

Desarrollo de la aplicación web: Interfaz para gestión de entidades y horarios, responsiva.

Desarrollo de la aplicación móvil: Consulta de horarios en Android.

Pruebas: Unitarias, manuales y de integración. Base de datos MySQL.

4 Resultados

- **Inicio de sesión:** Implementado con autenticación segura, validando credenciales y permitiendo acceso a funcionalidades según roles.
- **Visualización de horarios:** Interfaz para consultar horarios de maestros o cursos, con filtros y búsqueda.
- **Gestión de maestros:** CRUD para registrar y administrar maestros, incluyendo asignaturas y emails.
- **Gestión de cursos:** Registro y visualización de cursos disponibles.
- **Gestión de asignaturas:** Administración de asignaturas necesarias en la institución.

- **Gestión de horarios:** Asignación y visualización de horarios, evitando conflictos de disponibilidad.
- **Requerimientos no funcionales:** Seguridad con cifrado, tiempos de respuesta <3 segundos, compatibilidad multiplataforma, soporte técnico y cumplimiento normativo.

5 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.

6 Conclusiones

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.

7 Anexos

Anexo A. Descripción de componentes del sistema SGH

- **Aplicación web:** Interfaz para gestión de maestros, cursos, asignaturas y horarios, con funcionalidades de registro y visualización.
- **Aplicación móvil:** Acceso móvil para consulta de horarios por curso o profesor.
- **Backend (Java con Spring Boot):** Lógica de negocio, autenticación JWT, APIs REST para CRUD de entidades.
- **Base de datos (MySQL):** Almacenamiento de usuarios, maestros, cursos, asignaturas y horarios.

8 Referencias

1. Tecnologías Front-end y Back-end en Tendencia. (2017). Recuperado de <https://repository.usta.edu.co/handle/11362/40000/1/ae8f-4b01-a066-849fcb70e15f>
2. Especificando una arquitectura de software. (2020). Recuperado de <https://revistas.udistrital.edu.co/index.php/tia/article/view/13364>
3. Practicante en lenguaje de programación JAVA y nuevas tecnologías. (2025). Recuperado de <https://repositorio.utp.edu.co/entities/publication/192d53ab-1ac8-4829-ab8f-2639b995d120>
4. Análisis prospectivo de la industria de desarrollo de software en Colombia. (2020). Recuperado de <https://revistas.poligran.edu.co/index.php/puntodevista/article/view/1415>
5. Una revisión comparativa de la literatura acerca de metodologías tradicionales y modernas de desarrollo de software. (2019). Recuperado de <https://revistas.pascualbravo.edu.co/index.php/tia/article/view/13364>
6. Estudio sobre metodologías de desarrollo y su impacto en la productividad. (2020). Recuperado de <https://revistas.udistrital.edu.co/index.php/tia/article/view/13364>