REPÚBLICA DEL ECUADOR

INSTITUTO SUPERIOR TECNOLÓGICO TENA

Tecnología, Innovación y Desarrollo

DESARROLLO DE SOFTWARE

IMPLEMENTAR UNA APLICACIÓN WEB QUE AUTOMATICE EL PROCESO DE ELABORACIÓN DEL HORARIO DOCENTE EN EL INSTITUTO SUPERIOR TECNOLÓGICO TENA

Trabajo de Integración Curricular, presentado como requisito parcial para optar por el título de Tecnólogo Superior en Desarrollo de Software

AUTOR: Alan Israel García Castro

TUTOR: Ing. Italo Marcelo Lara Pilco

TENA - ECUADOR

2025 - IIS

REPÚBLICA DEL ECUADOR

INSTITUTO SUPERIOR TECNOLÓGICO TENA
Tecnología, Innovación y Desarrollo

IMPLEMENTAR UNA APLICACIÓN WEB QUE AUTOMATICE EL PROCESO DE ELABORACIÓN DEL HORARIO DOCENTE EN EL INSTITUTO SUPERIOR TECNOLOGICO TENA

Trabajo de Integración Curricular, presentado como requisito parcial para optar por el título de Tecnólogo Superior en Desarrollo de Software.

AUTOR: Alan Israel García Castro

TUTOR: Ing. Italo Marcelo Lara Pilco

Tena - Ecuador

2025-IIS

APROBACIÓN DEL TUTOR

ING. ITALO MARCELO LARA PILCO

PROFESOR DEL INSTITUTO SUPERIOR TECNOLÓGICO TENA.

CERTIFICA:

En calidad de Tutor del Proyecto Integrador denominado: Trabajo de Integración Curricular, de autoría del señor Alan Israel García Castro, con CC. 1550197857 estudiante de la Carrera de Desarrollo de Software del Instituto Superior Tecnológico Tena, CERTIFICO que se ha realizado la revisión prolija del Trabajo antes citado, cumple con los requisitos de fondo y de forma que exigen los respectivos reglamentos e instituciones.

Tena, 06 de enero del 2026

Ing. Italo Lara

TUTOR DEL TIC

CERTIFICACIÓN DEL TRIBUNAL CALIFICADOR

Tena, 23 de enero del 2026

Los Miembros del Tribunal de Grado abajo firmantes, certificamos que el Trabajo de Titulación denominado: **IMPLEMENTAR UNA APLICACIÓN WEB QUE AUTOMATICE EL PROCESO DE ELABORACIÓN DEL HORARIO DOCENTE EN EL INSTITUTO SUPERIOR TECNOLOGICO TENA**, presentado por **Alan Israel García Castro**, con CC: **1550197857**, estudiante de la Carrera de Desarrollo de Software del Instituto Superior Tecnológico Tena, ha sido corregida y revisada; por lo que autorizamos su presentación.

Atentamente;

ING. JUAN ESPÍN
PRESIDENTE DEL TRIBUNAL

ING. OSWALDO BONIFAZ
MIEMBRO DEL TRIBUNAL

LIC. ANDRÉS VÉLEZ
MIEMBRO DEL TRIBUNAL

AUTORÍA

Yo, Alan Israel García Castro, con CC: 1550197857, declaro ser autor del presente Trabajo de Titulación denominado: **IMPLEMENTAR UNA APLICACIÓN WEB QUE AUTOMATICE EL PROCESO DE ELABORACIÓN DEL HORARIO DOCENTE EN EL INSTITUTO SUPERIOR TECNOLOGICO TENA** y absuelvo expresamente al Instituto Superior Tecnológico Tena, y a sus representantes jurídicos de posibles reclamos o acciones legales por el contenido de la misma.

Adicionalmente acepto y autorizo al Instituto Superior Tecnológico Tena, la publicación de mi trabajo de Titulación en el repositorio institucional- biblioteca Virtual.

AUTOR:

Alan Israel García Castro
CÉDULA: 1550197857

FECHA: Tena, 06 de enero del 2026

CARTA DE AUTORIZACIÓN POR PARTE DEL AUTOR

Yo, **Alan Israel García Castro**, declaro ser autor del Trabajo de Titulación titulado: **IMPLEMENTAR UNA APLICACIÓN WEB QUE AUTOMATICE EL PROCESO DE ELABORACIÓN DEL HORARIO DOCENTE EN EL INSTITUTO SUPERIOR TECNOLOGICO TENA**, como requisito para la obtención del Título de: TECNÓLOGO SUPERIOR EN DESARROLLO DE SOFTWARE: autorizo al Sistema Bibliotecario del Instituto Superior Tecnológico Tena, para que, con fines académicos, muestre al mundo la producción intelectual del Instituto, a través de la visualización de su contenido que constará en el Repositorio Digital Institucional.

Los usuarios pueden consultar el contenido de este trabajo en el RDI, en las redes de información del país y del exterior, con las cuales tenga convenio el Instituto. El Instituto Superior Tecnológico Tena, no se responsabiliza por el plagió o copia del presente trabajo que realice un tercero. Para constancia de esta autorización, en la ciudad de Tena, 05 de enero del 2026, firma el autor.

AUTOR: Alan Israel García Castro

FIRMA:

CÉDULA: 1550197857

DIRECCIÓN: Km 1 ½ vía (Tena-Archidona)

CORREO ELECTRÓNICO: alan.garcia@est.itstena.edu.ec

CELULAR: 098 458 9459

DATOS COMPLEMENTARIOS DEL TUTOR

MSC. Lara Pilco Italo Marcelo

TRIBUNAL DEL GRADO:

Ing. Juan Espín. (Presidente).

Ing. Oswaldo Bonifaz. (Miembro).

Lic. Andrés Vélez. (Miembro).

I. DEDICATORIA

Este trabajo va dedicado a mis padres, José García y Genny Castro. Ellos son mi mayor motivo para salir adelante. Hubo momentos en los que pensé que no iba a poder, pero siempre estuvieron ahí recordándome de lo que soy capaz. Cada vez que sentía que me rendía, sus palabras me levantaban. Sé todo lo que han tenido que sacrificar para que yo llegara hasta acá. Las desveladas, el trabajo duro, todo para que yo pudiera cumplir este sueño.

II. AGRADECIMIENTO

Expreso mi sincero agradecimiento al Instituto Superior Tecnológico Tena, por abrirme las puertas de esta prestigiosa institución y brindarme la oportunidad de formarme como profesional en el área de Desarrollo de Software.

Al Ing. Italo Marcelo Lara Pilco, tutor de este proyecto y docente de la institución, por su invaluable apoyo, guía y por compartir generosamente su experiencia y conocimientos sobre los procesos de elaboración de horarios, lo cual fue fundamental para el desarrollo exitoso de este sistema.

Al Ing. Oswaldo Bonifaz, profesor de investigación, quien como parte activa del proceso de planificación horaria me brindó información esencial sobre las necesidades reales y los procedimientos institucionales, permitiéndome comprender a profundidad la problemática a resolver.

III. ÍNDICES

Tabla de contenidos

IV. ÍNDICE DE ILUSTRACIONES

V. ÍNDICE DE TABLAS

1. TEMA

IMPLEMENTAR UNA APLICACIÓN WEB QUE AUTOMATICE EL PROCESO DE ELABORACIÓN DEL HORARIO DOCENTE EN EL INSTITUTO SUPERIOR TECNOLOGICO TENA

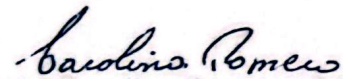
ABSTRACT

Currently, at the Instituto Superior Tecnológico Tena, the preparation of teacher schedules is carried out manually. This process requires considerable time and is susceptible to errors, as multiple variables must be considered simultaneously. Problems arise such as overlapping schedules, inefficient use of available resources, and complications when making on-the-fly changes.

This curricular integration project aims to create a web application that facilitates all processes related to teacher schedule management. As a pilot phase, the application will be tested in the Software Development program. The system will feature several components: a module for managing users with three different access levels (teacher, coordinator, and rectorate), a section where teachers can upload and modify their schedules in digital format, interactive visual tools that allow reviewing and making adjustments to activity distribution, a two-stage approval system to include observations and make necessary corrections, and finally the option to create official documents in PDF format. The objective is to reduce schedule conflicts and achieve better resource utilization, which represents a significant advancement in the modernization of the institution's administrative processes.

Keywords: Academic scheduling, administrative digitalization, educational management, resource optimization, web application, workflow automation.

Reviewed by



B.A. Carolina Romero, M.Ed.,

Professor Language Center

2. FUNDAMENTACIÓN DEL TEMA

2.1. Necesidad

Organizar los horarios de los profesores en el Instituto Superior Tecnológico Tena representa uno de los problemas más complejos de la gestión académica. Al inicio de cada nuevo período académico, es necesario coordinar múltiples factores de manera simultánea: disponibilidad de cada profesor, asignación de aulas libres, la distribución adecuada de la carga horaria para evitar agotamiento, el cumplimiento de las horas asignadas, cumplimiento de actividades complementarias (T/AES, tutorías, acompañamiento estudiantil), validación de períodos permitidos para cada actividad, y, además, tomar en consideración compromisos personales previamente informados por los docentes.

Hacer todo el proceso de manera manual consume tiempo considerable y, lo más problemático, es que incluso cuando se considera que el proceso está concluido, suelen surgir detalles que obligan a modificarlo. Estos ajustes generan un efecto en cadena que termina por rehacer gran parte del horario. El resultado son errores que ocasionan conflictos entre docentes, ya sea por la superposición de clases o por la falta de cumplimiento de acuerdos previamente establecidos, entre otros inconvenientes.

Se requiere implementar una solución tecnológica que optimice el tiempo invertido en la elaboración de horarios, permitiendo reasignar estos recursos hacia actividades de mayor valor académico e institucional. La automatización del proceso no solo representa un ahorro temporal, sino que garantiza una mejor organización institucional, considerando que una planificación estructurada desde las etapas iniciales repercute positivamente en el desarrollo de las actividades académicas subsecuentes.

2.2. Actualidad

Actualmente, el Instituto Superior Tecnológico Tena gestionaba el proceso de elaboración de horarios docentes completamente de forma manual. No existía ninguna plataforma digital especializada para este proceso; se utilizaban hojas de cálculo electrónicas propensas a errores humanos y el software aSc TimeTables (herramienta que permite crear horarios escolares de forma

automática y organizada) únicamente para la visualización inicial, sin integración con el resto del flujo de trabajo institucional.

Durante la etapa de levantamiento de requerimientos mediante entrevistas semiestructuradas con el coordinador de carrera y observación directa del proceso, se identificó un uso ineficiente de recursos institucionales. El proceso manual de elaboración de horarios requería varios días de trabajo continuo que podría destinarse a actividades de mayor impacto académico, tales como la mejora de programas de estudio o la coordinación de metodologías de enseñanza con el personal docente.

Con el desarrollo de esta aplicación web, se ha implementado una solución que integra las funcionalidades básicas del proceso en un único sistema centralizado, permitiendo automatizar el flujo desde la carga inicial hasta la aprobación final. Como proyecto piloto aplicado a la carrera de Desarrollo de Software, el sistema contempla los requerimientos fundamentales identificados, dejando las particularidades específicas de cada carrera para su implementación en versiones posteriores del sistema.

2.3. Importancia

La importancia del sistema radica en las funcionalidades que ofrece para optimizar el proceso de elaboración de horarios docentes. El aplicativo permite reducir significativamente el tiempo requerido para esta tarea administrativa, liberando recursos humanos calificados para que puedan enfocarse en actividades de planificación académica, diseño curricular y acompañamiento pedagógico, donde su experiencia profesional genera mayor valor agregado para la institución.

En el ámbito académico, el sistema facilita la elaboración de horarios más equilibrados y permite gestionar de manera ágil situaciones institucionales emergentes. La funcionalidad de edición y validación en tiempo real del aplicativo posibilita realizar ajustes y adecuaciones precisas ante imprevistos del semestre académico, reduciendo sustancialmente los tiempos de replanificación que anteriormente se extendían por varios días.

Los docentes también resultan beneficiados al eliminar la elaboración manual de sus horarios mediante formatos tradicionales. El sistema proporciona una interfaz estandarizada donde

pueden cargar sus archivos de horario en formato digital y agregar sus actividades complementarias de manera estructurada. Además, cada docente puede consultar su horario en cualquier momento a través de la plataforma, sin necesidad de solicitarlo o esperar la recepción de versiones actualizadas por correo electrónico, contando con acceso directo para gestionar y visualizar su información académica de forma autónoma.

Y aunque los estudiantes no van a tocar el sistema directamente, van a notar el cambio. Menos avisos de "se cambió el horario de esta materia", mejor aprovechamiento de los espacios, y en general una organización que funciona más fluida. Todo eso termina mejorando su experiencia educativa, que al final es lo que se busca.

2.4. Presentación del problema profesional a responder

El Instituto Superior Tecnológico Tena carece de un sistema automatizado para la elaboración y gestión de horarios docentes, lo que ha ocasionado conflictos de horarios, uso ineficiente de recursos y retrasos en la planificación académica.

Campo: Tecnologías de la Información y Comunicación

Área: Informática

Aspecto: Sistema web

Sector: Programación

Línea de investigación: Desarrollo de Software y sistemas informáticos.

2.5. Delimitación

2.5.1. Delimitación Espacial

El Trabajo de Integración Curricular se lo realizará en el Instituto Superior Tecnológico Tena, el mismo que está ubicado en la vía Tena-Archidona en el km 1,5.

2.5.2. Delimitación temporal

El proyecto se lo efectuará en el Periodo Académico 2025-IIS.

2.5.3. Delimitación técnica

El alcance técnico del proyecto se delimitó a la implementación de un sistema web monocapa sin integración con sistemas externos existentes en la institución. Se estableció como límite superior la arquitectura basada en el patrón Modelo-Vista-Controlador (MVC) que proporciona separación clara de responsabilidades entre la lógica de negocio, la presentación de datos y el manejo de peticiones del usuario, facilitando así futuras modificaciones o ampliaciones del sistema sin comprometer la estabilidad de componentes existentes (Gamma, 1994), excluyendo arquitecturas más complejas como microservicios o sistemas distribuidos que excederían las capacidades técnicas del personal institucional.

En el ámbito del backend, se delimitó el uso de Python 3.10.1 con Flask 3.0.0, estableciendo como frontera tecnológica frameworks minimalistas que permiten construir aplicaciones con la funcionalidad exacta requerida sin imposiciones arquitectónicas innecesarias, en contraste con alternativas más robustas pero complejas como Django que incluyen componentes que el proyecto no requiere (Grinberg, 2018). La persistencia de datos se delimitó exclusivamente a SQLite 3 como sistema de gestión de base de datos embebido que elimina la necesidad de configurar y mantener un servidor de base de datos independiente, reduciendo así los requisitos de infraestructura y simplificando los procedimientos de respaldo que se limitan a copiar un único archivo (Owens, 2006), descartando motores cliente-servidor como PostgreSQL o MySQL que requerirían infraestructura adicional no contemplada. El modelo de datos se restringió a ocho tablas principales: usuarios, horarios, observaciones, observaciones_especificas, asignacion_actividades, configuracion, configuracion_historial, y actividades_complementarias, estableciendo que cualquier entidad adicional queda fuera del alcance actual.

Para el frontend, la delimitación técnica excluyó el uso de frameworks o bibliotecas externas de terceros, restringiéndose únicamente a tecnologías web estándar: HTML5 semántico para garantizar accesibilidad y estructura lógica del contenido, CSS3 completamente personalizado para mantener control total sobre la apariencia visual y el rendimiento eliminando

código innecesario de frameworks genéricos, y JavaScript vanilla para toda la lógica del cliente evitando las dependencias y complejidad que introducen bibliotecas externas (Marcotte, 2011). Se estableció como límite funcional la compatibilidad únicamente con navegadores modernos que soporten Drag & Drop API nativa de HTML5 para la funcionalidad de arrastrar y soltar actividades, optimizada con Touch Events para dispositivos móviles, excluyendo explícitamente versiones antiguas de navegadores. La generación de documentos PDF se delimitó exclusivamente al procesamiento del lado del cliente utilizando bibliotecas especializadas, evitando así la carga en el servidor y permitiendo que usuarios con conexiones lentas puedan generar documentos sin depender del ancho de banda de subida.

El alcance de seguridad se delimitó mediante la implementación de múltiples capas de protección alineadas con las recomendaciones establecidas para aplicaciones web (OWASP Foundation, 2021), restringiéndose a hasheo SHA-256 para contraseñas por su balance entre seguridad computacional y rendimiento en el contexto de una aplicación educativa con carga moderada de usuarios, y autenticación mediante sesiones de Flask con decoradores personalizados que verifican tanto la existencia de sesión activa como el nivel de acceso según rol, estableciendo el principio de privilegio mínimo donde cada actor del sistema tiene acceso únicamente a las funcionalidades necesarias para sus responsabilidades específicas (Stallings, 2018). Se excluyeron mecanismos avanzados como autenticación multifactor o integración con servicios de directorio activo institucionales. La validación se delimitó a verificación doble de datos tanto en cliente como en servidor para prevenir manipulación maliciosa, estableciendo que ninguna validación crítica debe depender exclusivamente de código JavaScript que puede ser modificado por usuarios con conocimientos técnicos. La gestión de sesiones se restringió a expiración automática tras 12 horas de sesión permanente, balanceando seguridad con usabilidad para usuarios que trabajan durante jornadas laborales completas.

La delimitación de rendimiento estableció métricas específicas cuantificables que el sistema debe cumplir para garantizar una experiencia de usuario fluida: tiempo máximo de carga de página de 1 segundo para dashboards y vistas principales, tiempo de procesamiento de archivo .hored inferior a 2 segundos incluyendo validación y persistencia en base de datos, y tiempo de generación de documento PDF inferior a 3 segundos considerando la conversión HTML a imagen

y posterior inserción en PDF, descartando optimizaciones adicionales fuera del alcance inicial. El sistema se delimitó al soporte de al menos 20 usuarios simultáneos sin degradación perceptible de rendimiento mediante sincronización con mecanismos de control de acceso para base de datos y modo WAL (Write-Ahead Logging) de SQLite que permite lecturas concurrentes sin bloquear escrituras (Elmasri, 2015), excluyendo escenarios de carga masiva que requerirían soluciones de escalamiento horizontal.

La delimitación de compatibilidad y accesibilidad estableció restricción exclusiva a navegadores modernos Google Chrome y Mozilla Firefox en sus versiones actualizadas, excluyendo versiones antiguas que representan riesgos de seguridad y carecen de soporte para APIs modernas como Drag & Drop y Fetch. El diseño responsivo se delimitó mediante tres breakpoints principales: 768px para tablets donde se reorganizan elementos manteniendo funcionalidad completa, 480px para dispositivos móviles grandes donde la interfaz se adapta a orientación vertical, y 360px para dispositivos móviles pequeños donde se simplifican controles táctiles, descartando adaptaciones para dispositivos excepcionales fuera de estos rangos. La paleta de colores se restringió al uso de tonos institucionales del Instituto Superior Tecnológico Tena (#0097A7 y #00838F) para mantener coherencia con la identidad corporativa, estableciendo además que todos los elementos de interfaz deben cumplir con contraste mínimo de 4.5:1 según estándares de accesibilidad para garantizar legibilidad, excluyendo personalizaciones temáticas adicionales. La tipografía se delimitó al uso de Segoe UI como fuente principal con fallback a fuentes sans-serif genéricas, asegurando visualización correcta en cualquier sistema operativo sin requerir instalación de fuentes adicionales.

2.5.4. Unidades de observación

Las unidades de observación que se contemplan para este trabajo están enfocadas directamente a la Unidad de TIC del Instituto Superior Tecnológico Tena.

2.6. Beneficiarios

2.6.1. Directos

- Los docentes del Instituto Superior Tecnológico Tena.

2.6.2. Indirectos

- Estudiantes
- Vicerrectorado
- Coordinadores de Carrera
- Profesores
- IST Tena

3. OBJETIVOS

3.1. Objetivo General

Implementar una aplicación Web que automatice la elaboración del horario docente en el Instituto Superior Tecnológico Tena, con el fin de optimizar la planificación académica.

3.2 Objetivos Específicos

- Analizar los requerimientos funcionales y técnicos necesarios para la automatización del proceso de elaboración de horarios docentes.
- Diseñar una interfaz web intuitiva que permita a los usuarios gestionar, visualizar y modificar los horarios de manera eficiente.
- Desarrollar un aplicativo web que automatice la elaboración del horario docente en el Instituto Superior Tecnológico Tena

4. ASIGNATURAS INTEGRADORAS

Tabla 1 Asignaturas Integradoras

ASIGNATURAS INTEGRADORAS	
Asignaturas	**Resultados de Aprendizaje**
Programación de Aplicaciones Web	Utiliza lenguajes de programación tanto del lado del cliente como del servidor para realizar aplicaciones web.
Base de Datos	Elabora el modelamiento, ilustración y evaluación del proceso de base de datos.
Análisis y Diseño de Sistemas	Aplica metodologías y técnicas de investigación en la búsqueda, fundamentación y elaboración de soluciones informáticas.

Elaborado por: Alan García (2025)

5. FUNDAMENTACIÓN TEÓRICA

5.1. Sistemas de Gestión de Horarios Académicos

La organización de horarios en instituciones de educación superior es un proceso sumamente complejo. Es necesario coordinar la disponibilidad de decenas de docentes, gestionar aulas con capacidades diversas, asignar laboratorios especializados y, al mismo tiempo, evitar que los estudiantes enfrenten conflictos entre asignaturas. En años anteriores, esta tarea se realizaba mediante hojas de cálculo electrónicas, lo que generaba problemas en cadena ante cualquier modificación, muchas veces detectados únicamente cuando las clases ya habían comenzado.

Los sistemas modernos han transformado por completo este panorama. Actualmente, verifican de manera automática que ningún docente sea asignado a dos lugares distintos en el mismo horario (una situación que ocurría con frecuencia), revisan la capacidad de los salones y confirman la disponibilidad de los equipos. La investigación "A survey of approaches for university course timetabling problem" comparó instituciones de educación superior que implementaban sistemas automatizados contra aquellas que continuaban con procesos manuales; los resultados evidenciaron una reducción significativa en conflictos de horarios y menor tiempo invertido en correcciones de errores (Babaei, 2015).

Los componentes esenciales incluyen permisos diferenciados por roles, notificaciones automáticas cuando algo cambia, etc. Técnicamente el sistema debe escalar conforme crece la institución, mantener seguridad de información personal, y ser intuitivamente fácil de usar.

5.2. Programación Web

La programación web constituye el conjunto de prácticas, lenguajes y tecnologías empleadas para desarrollar aplicaciones que se ejecutan a través de navegadores web, operando mediante la arquitectura cliente-servidor donde el navegador solicita recursos a un servidor remoto que los procesa y devuelve (Olana & Tolasa, 2025). Esta modalidad revolucionó la distribución de software al eliminar instalaciones, actualizaciones manuales y problemas de compatibilidad con múltiples sistemas operativos, permitiendo que las aplicaciones funcionen en cualquier dispositivo con conexión a internet.

La evolución desde páginas estáticas simples en 1991 hasta aplicaciones complejas modernas incluyó hitos como la introducción de JavaScript en 1995 que permitió interactividad, AJAX en 2004 que mejoró la fluidez mediante actualizaciones parciales de páginas, y frameworks modernos desde 2010 que crearon aplicaciones de página única con navegación instantánea y rendimiento comparable a aplicaciones nativas. Las características distintivas incluyen accesibilidad multiplataforma sin requerir instalación de aplicaciones cliente con versiones específicas por sistema operativo, actualizaciones centralizadas que benefician a todos los usuarios simultáneamente, arquitectura distribuida que optimiza procesamiento entre cliente y servidor mediante componentes intermediarios independientes, escalabilidad de interacciones entre componentes, y despliegue independiente que facilita evolución del sistema sin afectar otros componentes (Fielding & Taylor, 2002).

5.3. Back-End

El backend constituye la parte de una aplicación web que opera en servidores remotos, invisible para usuarios, pero fundamental para procesamiento de solicitudes, ejecución de lógica de negocio compleja, gestión de acceso y manipulación de datos en bases de datos, comunicación con servicios externos y coordinación de recursos del sistema (Carmo, Ferreira, & Figueiredo, 2024). Actúa como intermediario inteligente entre frontend y bases de datos, donde cuando un usuario realiza una acción como enviar un formulario de registro, el backend recibe la petición HTTP, valida que los datos cumplan reglas de negocio, verifica que el usuario no exista previamente, hashea la contraseña para almacenamiento seguro, guarda el usuario en base de datos, posiblemente envía email de confirmación mediante servicio externo, y genera respuesta JSON que envía al frontend.

La arquitectura backend moderna enfatiza separación de responsabilidades dividiendo lógica en capas de controladores que manejan requests, servicios con lógica de negocio y modelos que representan datos, implementando seguridad mediante autenticación, autorización, validación exhaustiva y cifrado de datos sensibles (Nikum, y otros, 2024), optimizando rendimiento mediante caché, optimización de consultas, procesamiento asíncrono y uso eficiente de recursos, y garantizando escalabilidad para crecer horizontal o verticalmente según aumente demanda sin degradación significativa.

5.3.1. Lenguajes de Programación para Backend

Python 3.10.1 es el lenguaje interpretado dinámicamente tipado seleccionado para el desarrollo del backend del sistema, conocido por su sintaxis clara que prioriza la productividad del desarrollador y su versatilidad para desarrollo web, ciencia de datos, machine learning, automatización y scripting (Albesher, 2024). Esta versión específica garantiza compatibilidad con las bibliotecas seleccionadas y aprovecha las mejoras de rendimiento y seguridad introducidas en esta release.

Flask 3.0.0 es el microframework web seleccionado para el proyecto por su naturaleza minimalista que proporciona únicamente rutas, requests, responses y templates, permitiendo añadir todo lo demás mediante extensiones según necesidad, lo que posibilita construir aplicaciones con la funcionalidad exacta requerida sin imposiciones arquitectónicas innecesarias que caracterizan a frameworks full-stack como Django que incluyen ORM, migraciones, panel de administración, autenticación y templates integrados (Albesher, 2024). Esta elección permite mantener el sistema ligero y ajustado a los requerimientos específicos del Instituto Superior Tecnológico Tena sin componentes innecesarios que aumentarían la complejidad del mantenimiento.

5.3.2. Python y Framework Flask

Python ganó popularidad en desarrollo web principalmente por su sintaxis clara y accesible, junto con un ecosistema extenso de librerías. El lenguaje fue diseñado pensando en la legibilidad del código y la productividad del desarrollador, características especialmente útiles en proyectos donde los requerimientos evolucionan frecuentemente (Raschka, 2019).

Por otro lado, Flask es aquello que no incorpora de manera predeterminada. A diferencia de Django, que ofrece un entorno completamente preconfigurado, Flask proporciona únicamente los componentes esenciales. Implementa el estándar WSGI (fundamental para la compatibilidad con cualquier servidor web), incluye un sistema de rutas mediante decoradores, gestiona peticiones HTTP, integra Jinja2 para el renderizado de plantillas y ofrece soporte para sesiones. Esta filosofía minimalista es intencional: se empieza con poco y después se va conectando extensiones según lo que el proyecto requiera (Grinberg, 2018). Para una aplicación universitaria de gestión interna, este enfoque resulta bastante práctico.

Para proyectos universitarios o aplicaciones de uso interno, Flask representa una solución en el punto óptimo. Mientras que Django ofrece un entorno completamente integrado desde el inicio (lo cual puede ser adecuado en ciertos casos), en otros termina incorporando funcionalidades que no llegan a utilizarse. Flask, en cambio, proporciona únicamente lo esencial y permite crecer de manera orgánica conforme a las necesidades reales del proyecto.

5.4. Arquitectura Modelo-Vista-Controlador

El patrón MVC organiza el software en tres componentes principales. El Modelo gestiona los datos y las reglas de negocio; la Vista corresponde a la información que se presenta al usuario, es decir, la interfaz visible; y el Controlador coordina el flujo, recibiendo las peticiones, consultando al Modelo según lo requerido y determinando qué Vista mostrar. Este patrón surgió originalmente en aplicaciones de escritorio y fue adaptado para la web, donde hay diferencias importantes, principalmente que todo funciona bajo HTTP y que el navegador impone sus propias limitaciones sobre cómo se puede estructurar la interacción (Leff, 2001).

Al usar MVC mantenemos el código organizado y facilita la comprensión del proyecto incluso después de varios meses. En equipos, permite que diferentes personas trabajen en distintas capas sin generarse problemas mutuamente. El patrón permite reutilizar los Modelos para múltiples Vistas (Gamma, 1994). En el contexto de un sistema académico esto significa que los mismos datos y lógica pueden servir para una interfaz administrativa compleja, otra más simple para docentes, y quizás una app móvil para estudiantes. Las tres comparten el backend pero presentan la información de formas distintas.

5.4.1. Patrones de Arquitectura

Los patrones de arquitectura de software son soluciones reutilizables y probadas a problemas recurrentes en diseño de sistemas, proporcionando vocabulario común que facilita comunicación entre desarrolladores y estableciendo estructuras efectivas probadas en miles de proyectos (Majeed & Rauf, 2018). Definen estructura global del sistema especificando cómo se organizan componentes principales en módulos o capas, cómo interactúan mediante interfaces bien definidas, y qué responsabilidades tiene cada componente, influenciando profundamente atributos de calidad como mantenibilidad determinando facilidad de entender y modificar código,

escalabilidad definiendo cómo el sistema puede crecer, testabilidad facilitando testing de componentes aislados, y rendimiento mediante eficiencia de comunicación entre componentes.

Las características principales incluyen separación de responsabilidades dividiendo sistema en componentes con responsabilidades claramente definidas reduciendo acoplamiento, mejorando comprensión y facilitando testing y reutilización (Taibi & Lenarduzzi, 2018),independencia tecnológica permitiendo cambiar implementaciones sin afectar otros componentes siempre que se mantengan interfaces como cambiar de MySQL a PostgreSQL modificando solo capa de acceso a datos, testabilidad mejorada mediante aislamiento de lógica de negocio de infraestructura permitiendo testeo unitario sin dependencias externas con mocks ejecutándose rápidamente.

Los patrones principales incluyen arquitectura en capas organizando sistema en capas horizontales donde cada capa depende solo de inferiores típicamente con presentación, lógica de negocio, persistencia y base de datos proporcionando separación clara y facilidad de reemplazo pero añadiendo overhead de performance, microservicios descomponiendo aplicación en servicios pequeños independientemente desplegables comunicándose mediante APIs donde cada microservicio gestiona dominio específico encapsulando completamente datos y lógica con ventajas de escalabilidad granular, independencia tecnológica, despliegues independientes y fault isolation (Pahl & Jamshidi, 2016), y arquitectura hexagonal colocando lógica de negocio en núcleo completamente aislada de infraestructura donde núcleo define puertos como interfaces abstractas y adaptadores son implementaciones concretas conectando con tecnologías específicas con dependencias apuntando hacia adentro proporcionando testabilidad excepcional, independencia de frameworks, facilidad de cambios de infraestructura y claridad arquitectónica.

5.5. Gestión de Bases de Datos con SQLite

SQLite es una base de datos totalmente embebida, que no requiere instalación ni configuración de un servidor independiente. Todo el motor se almacena en un único archivo, lo que facilita su copia o respaldo sin complicaciones. A pesar de su portabilidad, cumple completamente con las propiedades ACID, lo cual garantiza que las transacciones sean confiables incluso si el sistema se cae en el momento menos esperado (Owens, 2006).

Para proyectos educativos o aplicaciones internas de alcance limitado, SQLite resulta prácticamente ideal. No requiere configuraciones innecesarias, funciona de manera uniforme en cualquier plataforma y ofrece soporte completo para SQL estándar (Elmasri, 2015). El módulo sqlite3 de Python permite trabajar directamente con la base de datos mediante SQL nativo, utilizando Row Factory (define cómo se devuelven las filas de una consulta) para acceder a los datos mediante nombres de columna en lugar de índices numéricos, lo que hace el código más legible y mantenible.

5.5.1. SQL (Structured Query Language)

SQL es el lenguaje estándar para interactuar con bases de datos relacionales, desarrollado en los años setenta en IBM y estandarizado por ANSI en 1986 evolucionando a través de revisiones como SQL-92 que estableció base moderna, SQL:1999 que añadió expresiones regulares y triggers, SQL:2003 con funciones de ventana y XML, SQL:2011 con datos temporales y SQL:2016 con soporte JSON. SQL se divide en sublanguajes especializados incluyendo DDL que define estructura mediante CREATE TABLE especificando columnas y restricciones, ALTER TABLE modificando tablas existentes, DROP TABLE eliminando tablas, CREATE INDEX creando índices y CREATE VIEW definiendo vistas como consultas guardadas, DML que manipula datos mediante SELECT consultando con filtrado WHERE, ordenamiento ORDER BY, agrupación GROUP BY, joins combinando tablas, funciones agregadas COUNT, SUM, AVG, MIN y MAX, INSERT añadiendo filas, UPDATE modificando filas y DELETE eliminando filas, DCL que controla permisos mediante GRANT otorgando permisos específicos y REVOKE revocándolos implementando principio de mínimo privilegio, y TCL que gestiona transacciones mediante BEGIN TRANSACTION iniciando transacciones, COMMIT confirmando cambios permanentemente, ROLLBACK deshaciendo cambios y SAVEPOINT creando puntos de restauración intermedios (Silva, 2016).

Los joins combinan datos de múltiples tablas donde INNER JOIN retorna solo coincidencias, LEFT JOIN retorna todas las filas izquierdas con coincidencias derechas rellenando NULL donde no hay coincidencia, RIGHT JOIN es el inverso y FULL OUTER JOIN retorna todas las filas de ambas tablas. Las subconsultas son consultas anidadas permitiendo lógica compleja en WHERE para filtrar basándose en otra consulta, en FROM como tablas derivadas, o en SELECT

para calcular valores, donde subconsultas correlacionadas referencian columnas de consulta externa ejecutándose para cada fila. Las funciones de ventana realizan cálculos a través de conjuntos de filas sin colapsar resultados como ROW_NUMBER asignando números secuenciales, RANK y DENSE_RANK asignando rankings, y LAG y LEAD accediendo a valores de filas anteriores o siguientes, permitiendo running totals, promedios móviles y rankings sin joins complejos.

Los CTEs o Common Table Expressions son consultas nombradas temporales definidas con WITH que mejoran legibilidad dividiendo consultas complejas en partes nombradas comprensibles, donde CTEs recursivos permiten consultas jerárquicas como organigramas o árboles de categorías. Los procedimientos almacenados son bloques de código SQL almacenados en servidor que encapsulan lógica de negocio compleja con múltiples consultas aceptando parámetros y retornando resultados, mejorando rendimiento al reducir tráfico de red y compilarse una vez.

Los triggers ejecutan automáticamente código en respuesta a eventos como INSERT, UPDATE o DELETE útiles para auditoría registrando automáticamente cambios, validaciones complejas verificando reglas de negocio, mantenimiento de datos desnormalizados y ejecución de lógica garantizando operaciones conjuntas. SQLite específicamente implementa mayoría del estándar SQL con tipado dinámico donde tipos son sugerencias permitiendo diferentes tipos en misma columna, careciendo de procedimientos almacenados complejos, triggers INSTEAD OF y sistema de usuarios, pero proporcionando extensiones útiles como funciones JSON, full-text search mediante FTS5 y R-Tree para consultas geoespaciales.

5.6. Metodología Kanban

Kanban tiene sus orígenes en las fábricas de Toyota, pero resultó funcionar muy bien para el desarrollo de software. Se describe usando tres principios fundamentales: visualizar el flujo de trabajo completo, limitar la cantidad de tareas que se hacen simultáneamente, y mejorar de forma continua mediante retroalimentación constante (Anderson, 2010). A diferencia de Scrum que utiliza sprints con estructura definida, Kanban fluye de manera más natural, usa un tablero visual donde se van moviendo tarjetas entre diferentes columnas conforme el trabajo va avanzando.

Los principios clave de la metodología incluyen: visualizar absolutamente todo el trabajo pendiente, limitar cuánto trabajo en progreso se tiene simultáneamente para forzar a concentrarnos y terminar las cosas en lugar de tener mil tareas a medias, gestionar el flujo general identificando dónde se están atorando las tareas, y definir criterios bien explícitos de cuándo exactamente se deben mover las tareas entre las diferentes columnas (Ahmad, 2013). Para proyectos individuales funciona perfectamente sin necesidad de todas las ceremonias complejas que tienen otras metodologías ágiles.

Aplicación de Kanban en el Proyecto

Para este proyecto se implementó un tablero Kanban con tres columnas principales: Pendiente, En Progreso y Completado. Se estableció un límite de trabajo en progreso (WIP limit) de máximo tres tareas simultáneas para mantener el foco y garantizar que cada funcionalidad se completara antes de iniciar nuevas características. El desarrollo se estructuró en cinco iteraciones semanales, cada una con entregables funcionales específicos que fueron validados antes de proceder a la siguiente fase.

Durante la primera semana se estableció la infraestructura base del proyecto, configurando el entorno de desarrollo con Python y Flask, implementando la estructura MVC y desarrollando el sistema de autenticación básico. La segunda semana expandió la funcionalidad con la gestión completa de usuarios y la creación de la estructura de base de datos. En la tercera semana se desarrolló la funcionalidad core del sistema: procesamiento de archivos .hored, interfaz de visualización con drag & drop, y validaciones automáticas de reglas de negocio. La cuarta iteración refinó la experiencia de usuario implementando fusión automática de actividades, sistema de observaciones y optimización móvil. La quinta semana completó el sistema con generación de PDF, funcionalidades administrativas y pruebas exhaustivas.

Cada tarea se movía del estado "Pendiente" a "En Progreso" solo cuando había capacidad disponible según el límite WIP establecido. Una vez completada y validada mediante pruebas funcionales con usuarios reales, la tarea pasaba a "Completado". Al finalizar cada iteración semanal se realizaban sesiones de revisión donde se demostraba la funcionalidad implementada a los usuarios clave del Instituto, recopilando retroalimentación que informaba ajustes en las

siguientes iteraciones. Este enfoque iterativo permitió detectar y corregir problemas tempranamente cuando su costo de corrección era menor.

5.7. Ingeniería de Requisitos

Antes de escribir las línea de código es imprescindible comprender qué necesitan realmente los usuarios. Si se cometen errores durante esta fase inicial, se arrastrará ese problema a lo largo de absolutamente todo el proyecto (Sommerville, 1997). Existen varias técnicas que funcionan mucho mejor cuando se las combina entre sí: entrevistas estructuradas con usuarios clave, observación directa donde literalmente se los ve trabajar con sus herramientas actuales, revisión detallada de la documentación que ya existe, talleres colaborativos donde se discuten y priorizan las necesidades reales, y prototipos rápidos que permiten validar las ideas antes de invertir demasiado tiempo.

Los requisitos generalmente se categorizan en dos tipos principales: funcionales que son concretos y medibles describiendo exactamente qué debe hacer el sistema, y no funcionales que describen los atributos más cualitativos del sistema. Estos suelen organizarse en categorías como rendimiento, seguridad, usabilidad, mantenibilidad y escalabilidad (Pressman, 2020). Lo más importante de todo este proceso es que se involucre de manera continua a los usuarios finales durante absolutamente todo el proceso, mostrándoles prototipos desde las etapas más tempranas y validando constantemente las suposiciones que se va haciendo.

5.8. Seguridad en Aplicaciones Web

La seguridad debe considerarse desde el inicio del proyecto y no añadirse posteriormente como un parche. La autenticación es el proceso que verifica quién es cada persona en el sistema, mientras que la autorización decide qué cosas específicas puede hacer cada usuario según el rol que tenga asignado. Un principio crítico que no se puede ignorar es nunca guardar las contraseñas de los usuarios en texto plano dentro de la base de datos, siempre se tiene que usar algoritmos de hash criptográficos (OWASP Foundation, 2021).

En este proyecto se implementará hashlib SHA-256 para el hasheo de contraseñas mediante la función hash_password() que utiliza hashlib.sha256(password.encode()).hexdigest(). Si bien

SHA-256 no es un algoritmo específicamente diseñado para contraseñas (como bcrypt o argon2 que incluyen salt automático y son deliberadamente lentos), proporciona un nivel de seguridad funcional para el contexto institucional del proyecto, combinándose con otras medidas como sesiones con expiración de 12 horas, cookies HTTPOnly y SameSite, y validación de credenciales en el servidor. Las mejores prácticas de seguridad implementadas incluyen la validación estricta de entradas tanto en cliente como servidor, la implementación de decoradores de autenticación personalizados, la gestión segura de sesiones con expiración automática y la restricción de acceso basada en roles mediante decoradores Python (Stallings, 2018).

5.9. Pruebas de Usabilidad

Las pruebas de usabilidad buscan responder una pregunta esencial: ¿qué tan sencillo resulta para una persona promedio completar las tareas previstas dentro del sistema? Aunque parece un concepto simple, su importancia es fundamental. Se ha demostrado que, con apenas cinco participantes, es posible identificar alrededor del 85% de los problemas graves de usabilidad (Nielsen, 1993). Este hallazgo resulta contraintuitivo, pues se esperaría necesitar un número mayor de usuarios, pero la evidencia confirma la eficacia de este enfoque.

Durante las pruebas de usabilidad se evalúan diversos aspectos: la tasa de éxito (es decir, si los usuarios lograron completar la tarea), el tiempo requerido en comparación con lo esperado, los errores cometidos durante el proceso y el nivel de satisfacción con la experiencia. Para esto último se usan cuestionarios ya validados como el System Usability Scale (SUS) (Tullis, 2013).

En el contexto universitario resulta especialmente importante realizar pruebas con todos los tipos de usuarios: docentes, personal administrativo y estudiantes. Cada grupo tiene sus propias expectativas de cómo debería funcionar todo, lo que resulta intuitivo para un profesor podría generar confusión en un estudiante, y viceversa.

5.10. Despliegue de Aplicaciones Web

Para poner en producción aplicaciones Flask hay varias opciones. La ruta tradicional es usar Gunicorn (un servidor WSGI) detrás de Nginx actuando como proxy inverso. Nginx se

encarga de manejar las conexiones que llegan y todo lo de SSL/TLS, mientras que Gunicorn ejecuta múltiples workers que procesan las peticiones de la aplicación de forma concurrente.

Pero también están las plataformas cloud tipo Heroku o PythonAnywhere que simplifican significativamente el proceso de despliegue. Básicamente se sube el código y ellos se encargan del resto. Docker representa otra alternativa interesante, empaqueta la aplicación con todas sus dependencias en contenedores ligeros. Estos contenedores garantizan portabilidad entre diferentes distribuciones de Linux, arrancan rápidamente y mantienen aislamiento entre procesos, asegurando que la aplicación funcione de manera consistente en cualquier entorno que soporte Docker (Merkel, 2014).

Para lograr un despliegue profesional es necesario considerar varios elementos adicionales. Entre ellos se incluyen: la utilización de scripts para automatizar el proceso de deployment y evitar tareas manuales, el uso de variables de entorno para almacenar información sensible como contraseñas y claves de API, evitando su inclusión directa en el código, la implementación de sistemas de monitoreo de logs que permitan detectar problemas antes de que se conviertan en fallas críticas, la ejecución periódica de respaldos automáticos y la capacidad de realizar un rollback rápido a una versión anterior en caso de que una actualización presente inconvenientes. Aunque todo esto puede parecer un esfuerzo adicional, resulta esencial para garantizar la estabilidad y confiabilidad del sistema, especialmente ante incidentes inesperados.

5.11. Marco Legal

Ley Orgánica de Educación Superior (LOES)

Según la Ley Orgánica de Educación Superior del Ecuador, en su Art. 93 sobre el principio de calidad, las instituciones de educación superior deben garantizar la organización eficiente de sus procesos académicos y administrativos. El Art. 117 establece que los institutos superiores técnicos y tecnológicos tienen autonomía académica para gestionar sus procesos de planificación curricular y asignación de recursos, lo cual incluye la organización de horarios docentes de manera eficiente y transparente.

Ley de comercio electrónico, firmas electrónicas y mensajes de datos

Que el uso de sistemas de información y redes electrónicas, incluida la Internet ha adquirido importancia para el desarrollo del comercio y la producción, permitiendo la realización y concreción de múltiples negocios de trascendental importancia, tanto para el sector público como para el sector privado; Que es necesario impulsar el acceso de la población a los servicios electrónicos que se generan por y a través de diferentes medios electrónicos; Que se debe generalizar la utilización de servicios de redes de información e Internet, de modo que estos se conviertan en un medio para el desarrollo del comercio, la educación y la cultura; Que a través del servicio de redes electrónicas, incluida la internet se establecen relaciones económicas y de comercio, y se realizan actos y contratos de carácter civil y mercantil que es necesario normarlos, regularlos y controlarlos, mediante la expedición de una especializada sobre la materia; Que es indispensable el uso de los servicios electrónicos, incluido el comercio electrónico y acceder con mayor facilidad a la cada vez más compleja red de los negocios internacionales.

Reglamento de Régimen Académico del Consejo de Educación Superior (CES)

El Reglamento de Régimen Académico establece en su Art. 16 los lineamientos para la organización del aprendizaje, especificando que las instituciones de educación superior deben planificar adecuadamente la distribución de actividades académicas, incluyendo horas de docencia, prácticas preprofesionales, tutorías y actividades de vinculación con la sociedad. Esta normativa fundamenta la necesidad de contar con sistemas tecnológicos que faciliten la gestión eficiente de estos componentes académicos.

COIP artículos referentes

Ley Orgánica de Educación Superior (LOES)

Según la Ley Orgánica de Educación Superior del Ecuador, en su Art. 93 sobre el principio de calidad, las instituciones de educación superior deben garantizar la organización eficiente de sus procesos académicos y administrativos (Asamblea Nacional del Ecuador, 2010). El Art. 117 establece que los institutos superiores técnicos y tecnológicos tienen autonomía académica para gestionar sus procesos de planificación curricular y asignación de recursos, lo cual incluye la organización de horarios docentes de manera eficiente y transparente (Asamblea Nacional del Ecuador, 2010).

Ley de comercio electrónico, firmas electrónicas y mensajes de datos

El uso de sistemas de información y redes electrónicas, incluida la Internet, ha adquirido importancia para el desarrollo del comercio y la producción, permitiendo la realización y concreción de múltiples negocios de trascendental importancia tanto para el sector público como para el sector privado (Congreso Nacional del Ecuador, 2002). Es necesario impulsar el acceso de la población a los servicios electrónicos que se generan por y a través de diferentes medios electrónicos, generalizando la utilización de servicios de redes de información e Internet de modo que estos se conviertan en un medio para el desarrollo del comercio, la educación y la cultura (Congreso Nacional del Ecuador, 2002). A través del servicio de redes electrónicas, incluida la internet, se establecen relaciones económicas y de comercio, y se realizan actos y contratos de carácter civil y mercantil que es necesario normarlos, regularlos y controlarlos mediante la expedición de una ley especializada sobre la materia (Congreso Nacional del Ecuador, 2002).

Reglamento de Régimen Académico del Consejo de Educación Superior (CES)

El Reglamento de Régimen Académico establece en su Art. 16 los lineamientos para la organización del aprendizaje, especificando que las instituciones de educación superior deben planificar adecuadamente la distribución de actividades académicas, incluyendo horas de docencia, prácticas preprofesionales, tutorías y actividades de vinculación con la sociedad (Consejo de Educación Superior, 2019). Esta normativa fundamenta la necesidad de contar con sistemas tecnológicos que faciliten la gestión eficiente de estos componentes académicos.

Código Orgánico Integral Penal (COIP) - Artículos Referentes

Artículo 229.- Revelación ilegal de base de datos. La persona que, en provecho propio o de un tercero, revele información registrada, contenida en ficheros, archivos, bases de datos o medios semejantes, a través o dirigidas a un sistema electrónico, informático, telemático o de telecomunicaciones; materializando voluntaria e intencionalmente la violación del secreto, la intimidad y la privacidad de las personas, será sancionada con pena privativa de libertad de uno a tres años (Asamblea Nacional del Ecuador, 2014). Si esta conducta se comete por una o un servidor público, empleadas o empleados bancarios internos o de instituciones de la economía popular y

solidaria que realicen intermediación financiera o contratista, será sancionada con pena privativa de libertad de tres a cinco años (Asamblea Nacional del Ecuador, 2014).

Artículo 230.- Intercepción ilegal de datos. Será sancionada con pena privativa de libertad de tres a cinco años la persona que, sin orden judicial previa, en provecho propio o de un tercero, intercepte, escuche, desvíe, grabe u observe, en cualquier forma un dato informático en su origen, destino o en el interior de un sistema informático, una señal o una transmisión de datos o señales con la finalidad de obtener información registrada o disponible (Asamblea Nacional del Ecuador, 2014). También será sancionada la persona que diseñe, desarrolle, venda, ejecute, programe o envíe mensajes, certificados de seguridad o páginas electrónicas, enlaces o ventanas emergentes o modifique el sistema de resolución de nombres de dominio de un servicio financiero o pago electrónico u otro sitio personal o de confianza, de tal manera que induzca a una persona a ingresar a una dirección o sitio de internet diferente a la que quiere acceder (Asamblea Nacional del Ecuador, 2014).

Artículo 232.- Ataque a la integridad de sistemas informáticos. La persona que destruya, dañe, borre, deteriore, altere, suspenda, trabe, cause mal funcionamiento, comportamiento no deseado o suprima datos informáticos, mensajes de correo electrónico, de sistemas de tratamiento de información, telemático o de telecomunicaciones a todo o partes de sus componentes lógicos que lo rigen será sancionada con pena privativa de libertad de tres a cinco años (Asamblea Nacional del Ecuador, 2014). Si la infracción se comete sobre bienes informáticos destinados a la prestación de un servicio público o vinculado con la seguridad ciudadana, la pena será de cinco a siete años de privación de libertad (Asamblea Nacional del Ecuador, 2014).

Artículo 234.- Acceso no consentido a un sistema informático, telemático o de telecomunicaciones. La persona que sin autorización acceda en todo o en parte a un sistema informático o sistema telemático o de telecomunicaciones o se mantenga dentro del mismo en contra de la voluntad de quien tenga el legítimo derecho, para explotar ilegítimamente el acceso logrado, modificar un portal web, desviar o redireccionar de tráfico de datos o voz u ofrecer servicios que estos sistemas proveen a terceros, sin pagarlos a los proveedores de servicios legítimos, será sancionado con la pena privativa de la libertad de tres a cinco años (Asamblea Nacional del Ecuador, 2014).

5.12. Marco conceptual

El marco conceptual define los términos y conceptos específicos utilizados en el desarrollo del sistema de gestión de horarios docentes del Instituto Superior Tecnológico Tena, estableciendo el significado preciso de los conceptos operativos en el contexto particular de esta investigación.

5.12.1. Conceptos Institucionales

Horario Docente: Distribución semanal de actividades académicas y complementarias asignadas a un profesor. Incluye asignaturas impartidas con sus respectivos códigos, aulas asignadas y actividades complementarias que debe cumplir durante el período académico.

Actividades Complementarias: Responsabilidades académicas adicionales a la docencia directa. Incluyen T/AES, Dirección de Proyectos Integradores, Coordinación de Área, Investigación y Vinculación con la Sociedad, cada una con códigos identificadores únicos y límites de horas asignables.

T/AES (Tutorías de Acompañamiento Estudiantil): Actividad complementaria de acompañamiento personalizado a estudiantes con dificultades académicas. Según normativa institucional, debe programarse exclusivamente en los períodos 6, 7, 9 y 10 del horario semanal, constituyendo una regla de negocio crítica del sistema.

Ciclo Académico: Período semestral durante el cual se desarrollan las actividades académicas, identificado mediante formato año-nivel (ejemplo: 2025-IIS). Aparece en los horarios oficiales generados por el sistema y sirve como parámetro de configuración global.

5.12.2. Conceptos Técnicos del Sistema

Formato .hored (Horarios Editables): Archivo JSON estructurado desarrollado específicamente para este proyecto. Contiene información del docente, configuración del período académico, matriz de clases organizadas por días y períodos.

Estado de Horario: Indicador del progreso del horario en el flujo de aprobación. Siete estados: borrador (permite edición y guardado automático), revisión_coordinador (enviado por

docente, bloqueado), rechazado coordinador (devuelto con observaciones), revisión rectorado (aprobado por coordinador), rechazado rectorado (devuelto por rectorado), aprobado (habilita descarga de documentos oficiales) y pendiente (cuando un horario está en proceso de revisión).

Fusión Automática de Actividades: Funcionalidad que combina visualmente actividades idénticas asignadas en períodos consecutivos del mismo día, mejorando la legibilidad del horario impreso. La fusión es visual; internamente el sistema mantiene el registro individual de cada período.

Observaciones de Revisión: Sistema dual de retroalimentación del flujo de aprobación: observaciones generales sobre el horario completo y observaciones específicas asociadas a celdas particulares del horario. Cada una registra el revisor responsable, fecha y contenido textual del comentario.

Validación en Tiempo Real: Verificación automática de reglas de negocio durante la edición del horario, con retroalimentación inmediata mediante indicadores visuales. Incluye: límite máximo de 8 horas por día, restricción de T/AES a períodos específicos, verificación de límites por actividad complementaria y prevención de asignación en celdas ocupadas.

5.12.3. Conceptos de Roles y Permisos

Rol Docente: Nivel básico de acceso con permisos para cargar archivos .hored, visualizar y editar su propio horario, agregar actividades complementarias según límites asignados, enviar horario a revisión, consultar observaciones y descargar documentos oficiales una vez aprobados. No tiene acceso a funcionalidades administrativas.

Rol Coordinador: Hereda permisos de docente y añade capacidades de revisión y gestión: revisar y aprobar/rechazar horarios con observaciones, gestionar actividades complementarias, administrar usuarios docentes, asignar límites de horas por actividad, configurar ciclo académico, y gestionar su propio perfil (datos personales y contraseña).

Rol Rectorado: Nivel máximo que hereda capacidades de docente y coordinador, añadiendo la aprobación final de horarios. Visualiza horarios pre-aprobados por coordinadores,

realiza aprobación definitiva o rechaza con observaciones independientes, configura parámetros institucionales globales, y gestiona su propio perfil (datos personales y contraseña).

5.12.4. Conceptos de Interacción del Usuario

Drag & Drop (Arrastrar y Soltar): Funcionalidad implementada mediante API HTML5 que permite asignar actividades complementarias arrastrándolas desde una barra lateral y soltándolas en la celda deseada. Incluye optimizaciones para dispositivos móviles mediante Touch Events, con delays para diferenciar scroll de arrastre y feedback háptico mediante vibración.

Dashboard (Panel de Control): Interfaz principal diferenciada según rol del usuario. Docente: muestra sus horarios con estados y opciones disponibles. Coordinador: presenta horarios pendientes de revisión ordenados por fecha de envío. Rectorado: muestra horarios pre-aprobados pendientes de aprobación final.

Guardado Automático: Funcionalidad que preserva el trabajo en progreso sin acción manual del usuario. El sistema ejecuta guardado automático cada 30 segundos mientras el horario está en estado borrador, enviando cambios al servidor mediante peticiones AJAX, garantizando recuperación ante interrupciones.

5.12.5. Conceptos de Integridad y Auditoría

Sistema de Logging: Mecanismo de registro de eventos organizado en seis categorías mediante archivos independientes: autenticación (login/logout/cambios de contraseña), horarios (creación/modificación/eliminación), administración (aprobaciones y rechazos), sistema (inicialización y operaciones generales), errores (excepciones con información para debugging) y concurrencia (acceso simultáneo). Cada entrada incluye timestamp, usuario responsable y acción realizada.

Timestamp (Marca de Tiempo): Registro preciso de fecha y hora en formato ISO 8601 zona horaria local (America/Guayaquil). El sistema implementa timestamps diferenciados para cada etapa del flujo: creación del horario, envío a coordinador, aprobación/rechazo por coordinador, envío a rectorado y aprobación/rechazo final, permitiendo auditoría completa.

Historial de Configuración: Tabla de auditoría que registra todos los cambios en parámetros institucionales del sistema. Cada modificación documenta: valor anterior, valor nuevo, usuario responsable, fecha y hora del cambio, y justificación textual, permitiendo rastrear la evolución de parámetros y responsabilizar las modificaciones.

6. METODOLOGÍA

6.1. Materiales

El desarrollo del sistema se realizó en una estación de trabajo con procesador AMD Ryzen 5 7535HS a 3.30 GHz, 16 GB de RAM y 477 GB de almacenamiento, capacidad suficiente para ejecutar el entorno de desarrollo, servidor local y pruebas simultáneas del sistema.

Los recursos de software incluyeron Python 3.10.1 como lenguaje base, Flask 3.0.0 como framework web, SQLite 3 mediante el módulo sqlite3 nativo para la gestión de base de datos, HTML5 y CSS3 completamente personalizados para el frontend sin frameworks CSS externos, y JavaScript vanilla para toda la lógica del cliente. El desarrollo se realizó en Visual Studio Code como editor de código. Las pruebas de compatibilidad se ejecutaron en navegadores web modernos como Google Chrome y Firefox.

6.2. Ubicación del Área de Estudio

El proyecto se desarrolló en el Instituto Superior Tecnológico Tena, fundado el 28 de julio de 2003, con RUC 1768193160001; ubicado en el Km 1½ vía Tena-Archidona, provincia de Napo, Ecuador.

6.3. Tipo de Investigación / Estudio

Este trabajo corresponde a una investigación aplicada de carácter tecnológico. A diferencia de la investigación pura, cuyo objetivo es generar conocimiento por sí mismo, aquí se abordó un problema concreto: el proceso manual y propenso a errores de elaboración de horarios en el Instituto Superior Tecnológico Tena.

La investigación se orientó a resolverlo mediante la creación de una herramienta funcional completamente automatizada que integra validaciones de negocio, gestión de roles, flujo de aprobación multi-etapa y generación de documentos oficiales.

Se empleó un enfoque mixto que combinó investigación descriptiva (análisis del proceso actual) con investigación propositiva (diseño e implementación de solución). El desarrollo se realizó mediante metodología Kanban, permitiendo entregas incrementales validadas con usuarios finales en cada etapa.

6.4. Marco metodológico: Kanban

6.4.1. Análisis de Requerimientos

El análisis de requerimientos se logrará mediante un enfoque cualitativo de investigación aplicada que combinará múltiples técnicas de recolección de información para garantizar comprensión integral del problema y las necesidades de los usuarios finales. Se aplicará la metodología de ingeniería de requisitos adaptada al contexto específico de una institución educativa de nivel superior (Sommerville, 1997). El proceso se estructurará en tres fases complementarias que permitirán triangular la información y validar los hallazgos desde diferentes perspectivas.

La primera fase consistirá en entrevistas semiestructuradas con actores clave del proceso de planificación académica, utilizando un protocolo de preguntas abiertas que permitirá explorar tanto los aspectos técnicos como los procedimentales del proceso actual. Esta técnica facilitará la identificación de necesidades explícitas manifestadas directamente por los usuarios, así como expectativas sobre la solución automatizada.

La segunda fase implementará observación directa no participante del proceso completo de elaboración de horarios durante un ciclo académico, registrando mediante diario de campo estructurado cada paso, decisión, dificultad y corrección realizada. Esta observación resultará fundamental para identificar necesidades implícitas que no surgirán en las entrevistas, revelando aspectos operativos que los usuarios realizan de manera automática sin reconocerlos como requisitos del sistema.

La tercera fase comprenderá análisis documental de horarios de semestres anteriores y normativa institucional vigente, permitiendo identificar patrones recurrentes, restricciones formales y formatos de salida esperados.

La integración de información de las tres fases metodológicas se realizará mediante técnicas de análisis de contenido cualitativo, codificando las transcripciones de entrevistas, registros de observación y documentos revisados para identificar temas recurrentes y establecer categorías de requerimientos. Este análisis permitirá organizar los hallazgos en requerimientos funcionales coherentes y estructurados, sin anticipar resultados específicos.

Adicionalmente, se definirán requerimientos no funcionales en las dimensiones de seguridad, usabilidad, rendimiento y mantenibilidad, estableciendo criterios cuantificables que permitirán posteriormente validar el cumplimiento del sistema desarrollado. La validación de los requerimientos identificados se realizará mediante sesiones de revisión con los informantes clave, presentándoles los hallazgos documentados y solicitando confirmación de completitud y precisión, lo que permitirá realizar ajustes antes de proceder a las etapas de diseño e implementación. Este enfoque metodológico riguroso garantizará que el sistema desarrollado responda efectivamente a las necesidades reales de la institución.

6.4.2. Diseño de Interfaz

El diseño de la interfaz web se logrará mediante un proceso iterativo centrado en el usuario que aplicará los principios de usabilidad establecidos (Nielsen, 1993) y las mejores prácticas de diseño responsivo documentadas (Marcotte, 2011). La metodología seguirá un enfoque de prototipado incremental que iniciará con esquemas conceptuales de baja fidelidad y progresará

hacia implementaciones funcionales de alta fidelidad, incorporando retroalimentación de usuarios reales en cada iteración.

El proceso comenzó con la técnica de wireframing en papel, creando bocetos simples de las pantallas principales del sistema que representaron la estructura y flujo de navegación sin invertir tiempo en detalles visuales prematuros. Estos wireframes fueron presentados a los actores clave del proceso de gestión de horarios en sesiones de revisión temprana, permitiendo validar la conceptualización inicial de la solución y realizar ajustes fundamentales en la organización de la información antes de avanzar a diseños más elaborados. Esta fase inicial de validación conceptual resultó crucial para garantizar que la propuesta de diseño alineara con los modelos mentales y expectativas de los usuarios finales.

La transición de wireframes a diseños de alta fidelidad se realizó mediante la elaboración de mockups detallados que incorporaron la paleta de colores institucionales, la jerarquía tipográfica definida y el sistema de iconografía consistente. Estos mockups fueron implementados como prototipos funcionales utilizando HTML y CSS, permitiendo a los usuarios interactuar con interfaces reales en lugar de imágenes estáticas. La metodología incluyó cinco sesiones de pruebas de usabilidad con diferentes perfiles de usuarios, aplicando la técnica de pensamiento en voz alta donde los participantes verbalizaron sus pensamientos mientras interactuaron con el sistema.

Durante estas sesiones se registraron sistemáticamente las dificultades de comprensión, los errores cometidos, el tiempo requerido para completar tareas específicas y las sugerencias explícitas de mejora. Los hallazgos de cada sesión fueron analizados mediante técnicas de análisis cualitativo, identificando patrones recurrentes de problemas de usabilidad y priorizándolos según frecuencia y severidad. Los ajustes identificados fueron implementados iterativamente, creando nuevas versiones del prototipo que fueron sometidas a ciclos adicionales de validación hasta alcanzar niveles satisfactorios de facilidad de uso.

La implementación técnica del diseño se realizó siguiendo un enfoque mobile-first que priorizó la experiencia en dispositivos móviles y posteriormente escaló a pantallas más grandes, utilizando CSS3 completamente personalizado sin dependencia de frameworks externos. Se aplicaron los principios de diseño responsivo mediante un sistema de grillas flexible basado en

CSS Grid y Flexbox, implementando tres breakpoints principales que adaptaron la interfaz a diferentes tamaños de pantalla manteniendo funcionalidad completa. La paleta de colores institucionales se aplicó consistentemente en todos los componentes de interfaz, estableciendo jerarquía visual mediante variaciones de saturación y luminosidad. El resultado del proceso metodológico iterativo y validado empíricamente fue un conjunto de interfaces diferenciadas por rol que fueron aprobadas por los usuarios finales, confirmando que el diseño es intuitivo, estéticamente coherente con la identidad institucional, y funcionalmente completo para soportar todas las operaciones requeridas en el proceso de gestión de horarios docentes.

6.4.3. Desarrollo de la Aplicación

El desarrollo del aplicativo web se logró mediante la aplicación de metodología Kanban que permitió gestionar el flujo de trabajo de manera visual y flexible, adaptándose a las necesidades cambiantes identificadas durante el proceso de implementación (Anderson, 2010). Esta metodología ágil fue seleccionada por su énfasis en la entrega continua de valor y su capacidad para adaptarse a equipos pequeños sin la sobrecarga de ceremonias complejas. El trabajo se organizó mediante un tablero visual con tres columnas que representaron el estado de cada funcionalidad: pendiente, en proceso y completado, limitando el trabajo en progreso a máximo tres tareas simultáneas para mantener foco y garantizar que cada componente fuera desarrollado completamente antes de iniciar el siguiente. El desarrollo se estructuró en cinco iteraciones semanales, cada una con objetivos específicos y entregables funcionales que fueron validados antes de proceder a la siguiente fase.

La primera iteración estableció la infraestructura fundamental del proyecto, configurando el entorno de desarrollo con Python y Flask, implementando la estructura MVC que separó responsabilidades, y desarrollando el sistema básico de autenticación con gestión de sesiones. La segunda iteración expandió la funcionalidad de gestión de usuarios, creando la estructura completa de base de datos con las siete tablas principales y desarrollando las operaciones CRUD necesarias para administrar docentes y sus asignaciones. La tercera iteración se enfocó en la funcionalidad core del sistema, implementando el procesamiento de archivos .hored, desarrollando la interfaz interactiva de visualización y edición de horarios con drag and drop, e incorporando las validaciones automáticas de reglas de negocio que garantizaron cumplimiento de normativa

institucional. La cuarta iteración refinó la experiencia de usuario mediante la implementación del algoritmo de fusión automática de actividades consecutivas, el desarrollo del sistema dual de observaciones a nivel de celda y general, y la optimización de la interfaz para dispositivos móviles con soporte táctil. La quinta iteración completó el sistema con la generación de documentos PDF oficiales, implementación de funcionalidades administrativas complementarias, y ejecución de pruebas exhaustivas de compatibilidad, seguridad y rendimiento.

Cada funcionalidad desarrollada fue sometida a un proceso de validación que incluyó pruebas unitarias de componentes individuales para verificar comportamiento correcto en casos normales y casos extremos, pruebas de integración para confirmar que los módulos interactuaran correctamente entre sí, y pruebas de aceptación con usuarios reales que validaron el cumplimiento de los requerimientos funcionales documentados en la primera etapa del proyecto. La metodología incluyó además sesiones de revisión al finalizar cada iteración donde se demostró la funcionalidad implementada a los usuarios clave, recopilando retroalimentación que informó ajustes en las iteraciones subsiguientes. Este enfoque iterativo e incremental permitió detectar y corregir problemas tempranamente cuando su costo de corrección fue menor, y garantizó que el producto final respondiera efectivamente a las necesidades reales de los usuarios.

El proceso de desarrollo se complementó con la implementación de buenas prácticas de ingeniería de software, incluyendo control de versiones para rastrear cambios en el código fuente, documentación técnica de la arquitectura y componentes principales, y sistema de logging categorizado que facilitó el diagnóstico de problemas en producción. Las pruebas finales de validación incluyeron verificación de compatibilidad en los navegadores definidos en la delimitación técnica, pruebas de responsividad en los breakpoints establecidos, validación de medidas de seguridad mediante intentos de ataques comunes como inyección SQL, y medición de rendimiento confirmando cumplimiento de las métricas establecidas. El resultado fue un sistema web completamente funcional, exhaustivamente testeado, y validado por usuarios finales, listo para su despliegue en el entorno de producción del Instituto Superior Tecnológico Tena, acompañado de documentación detallada que facilitó su mantenimiento y evolución futura por el equipo técnico institucional.

7. RESULTADOS

7.1. Análisis de Requerimientos Funcionales, No Funcionales y Técnicos

El análisis de requerimientos se desarrolló a través de un proceso sistemático de recolección de información cualitativa que combinó tres técnicas complementarias. En primer lugar, se llevaron a cabo entrevistas semiestructuradas con el coordinador de carrera y el responsable de la gestión de horarios, las cuales permitieron documentar el proceso actual en su totalidad e identificar las principales dificultades del sistema manual, tales como los conflictos de horarios recurrentes, el tiempo excesivo invertido en validaciones manuales, y los errores de asignación que frecuentemente se detectaban solo al inicio del período académico.

Ilustración 1 Formato Excel Horarios Docentes ISTT

HORARIO GENERAL DOCENTE

CARRERA/S: Desarrollo de Software

DOCENTE: Ing. Italo Marcelo Lara Pilco, MSc Cédula N°: 0603454364

PERÍODO: 2025 II S

INICIO	FIN	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES	SÁBADO
07:00	08:00	FDP-DS-(IA_Matutina)	FWA-DS-(I_A_Matutina)	FWA-DS-(I_A_Matutina)	FDP-DS-(IA_Matutina)	FWA-DS-(I_A_Matutina)	
08:00	09:00	FDP-DS-(IA_Matutina)	FWA-DS-(I_A_Matutina)	FWA-DS-(I_A_Matutina)	FDP-DS-(IA_Matutina)	FWA-DS-(I_A_Matutina)	
09:00	10:00	CED	FWA-DS-(I_A_Matutina)		Unidad de FPP	FDP-DS-(IA_Matutina)	
10:00	11:00	CED	PDC		Unidad de FPP	FDP-DS-(IA_Matutina)	
11:00	12:00	Unidad de FPP	Unidad de FPP		PDC	FDP-DS-(IA_Matutina)	
12:00	13:00		TIAES			TIAES	
ALMUERZO							
14:00	15:00			Unidad de FPP		Unidad de FPP	
15:00	16:00		COR_ESTRAT	CED		Unidad de FPP	
16:00	17:00	TIAES	COR_ESTRAT	CED	TIAES		
17:00	18:00	FDP-DS-(IA_Nocturna)		FDP-DS-(IA_Nocturna)	FDP-DS-(IA_Nocturna)		
18:00	19:00	FDP-DS-(IA_Nocturna)		FDP-DS-(IA_Nocturna)	FDP-DS-(IA_Nocturna)		
19:00	20:00			FDP-DS-(IA_Nocturna)			
20:00	21:00						
21:00	22:00						
	TOTAL	8	8	8	8	8	8

ACTIVIDADES COMPLEMENTARIAS	CODIGO
TUTORIAS, ACOMPAÑAMIENTO ESTUDIANTIL	TIAES
PRE PROFESIONALES	LEPP
COORDINACIÓN ESTRATÉGICA	COR ESTRAT
CALIFICACIÓN DE EVALUACIONES Y DEBERE	CED
PREPARACIÓN DE CLASES	PDC

Asignatura	Codigo	Curso/Paralelo
Fundamentos de Programación-DS-1A_Matutina	DDW-102	1A_Matutina
Fundamentos de Programación-DS-1_A_Nocturna	DDW-102	1A_Nocturna
Programación Web Avanzada	DDW-403	4A_Matutina

Nota. Formato manual de horarios de docentes realizados en Excel; Elaborado por Alan García (2025).

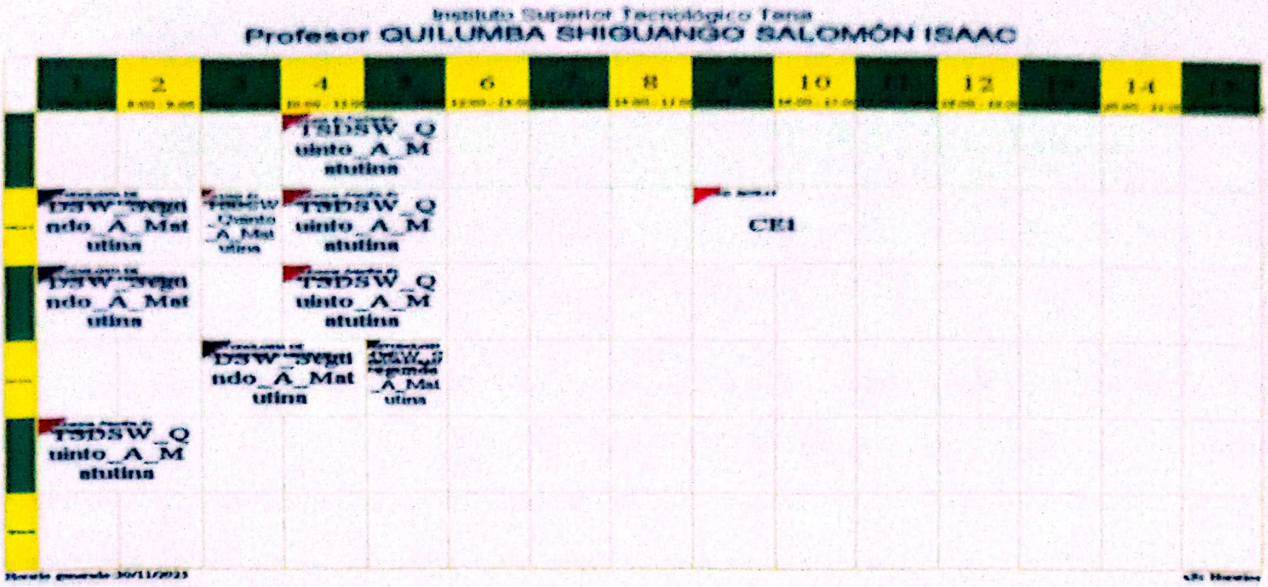
Estas conversaciones permitieron precisar los requisitos de validación críticos, entre ellos el límite máximo de ocho horas diarias de carga docente establecido por normativa institucional y las restricciones específicas para Tutorías de Acompañamiento Estudiantil (T/AES) que debían ubicarse únicamente en los períodos 6, 7, 9 y 10 del horario semanal. Además, se evidenció la necesidad de establecer múltiples niveles de aprobación que incluyeron revisión técnica del coordinador de carrera y aprobación final por parte de rectorado, así como la gestión adecuada de actividades complementarias con límites configurables por docente.

Posteriormente se realizó un proceso de observación directa durante tres semanas consecutivas del período de planificación académica. Durante este tiempo se analizó detalladamente cada paso del proceso de elaboración de horarios, registrando mediante diario de campo estructurado las decisiones tomadas, el tiempo invertido en cada actividad, las dificultades que surgían durante la asignación manual, y los procedimientos de corrección aplicados cuando se detectaban conflictos.

Esta etapa resultó particularmente valiosa porque permitió identificar numerosos aspectos operativos que no habían sido mencionados explícitamente en las entrevistas iniciales, como la necesidad de fusión visual automática de actividades consecutivas idénticas para mejorar la legibilidad del horario impreso, la importancia de mantener un estado de borrador que permitiera recuperar trabajo en progreso en caso de interrupciones, y la frecuencia con que se requerían ajustes menores que justificaron un sistema de observaciones granular tanto a nivel de celda específica como comentarios generales sobre el horario completo.

El análisis documental complementó las técnicas anteriores mediante la revisión sistemática de doce horarios correspondientes a los últimos cuatro semestres académicos, permitiendo identificar patrones recurrentes de distribución horaria, conflictos típicos que aparecían con mayor frecuencia, y formatos de salida generados por el software aSc TimeTables utilizado previamente por la institución. Se analizó la estructura de actividades complementarias y sus restricciones específicas, encontrando que actividades como tutorías, dirección de proyectos integradores, y coordinación de áreas tenían asignaciones variables que debían ser configurables por ciclo académico. Este trabajo permitió consolidar una visión integral de las exigencias normativas y operativas que debían ser consideradas en el diseño del sistema, resultando en la creación del formato .hored (horarios editables), un archivo JSON estructurado concebido específicamente para responder a las necesidades de intercambio de información entre el sistema de planificación institucional y la aplicación de edición individual de horarios docentes.

Ilustración 2 Formato Base Asc Time Tables



Nota. Visualización del formato base de aSc TimeTbales; Elaborado por Alan García (2025).

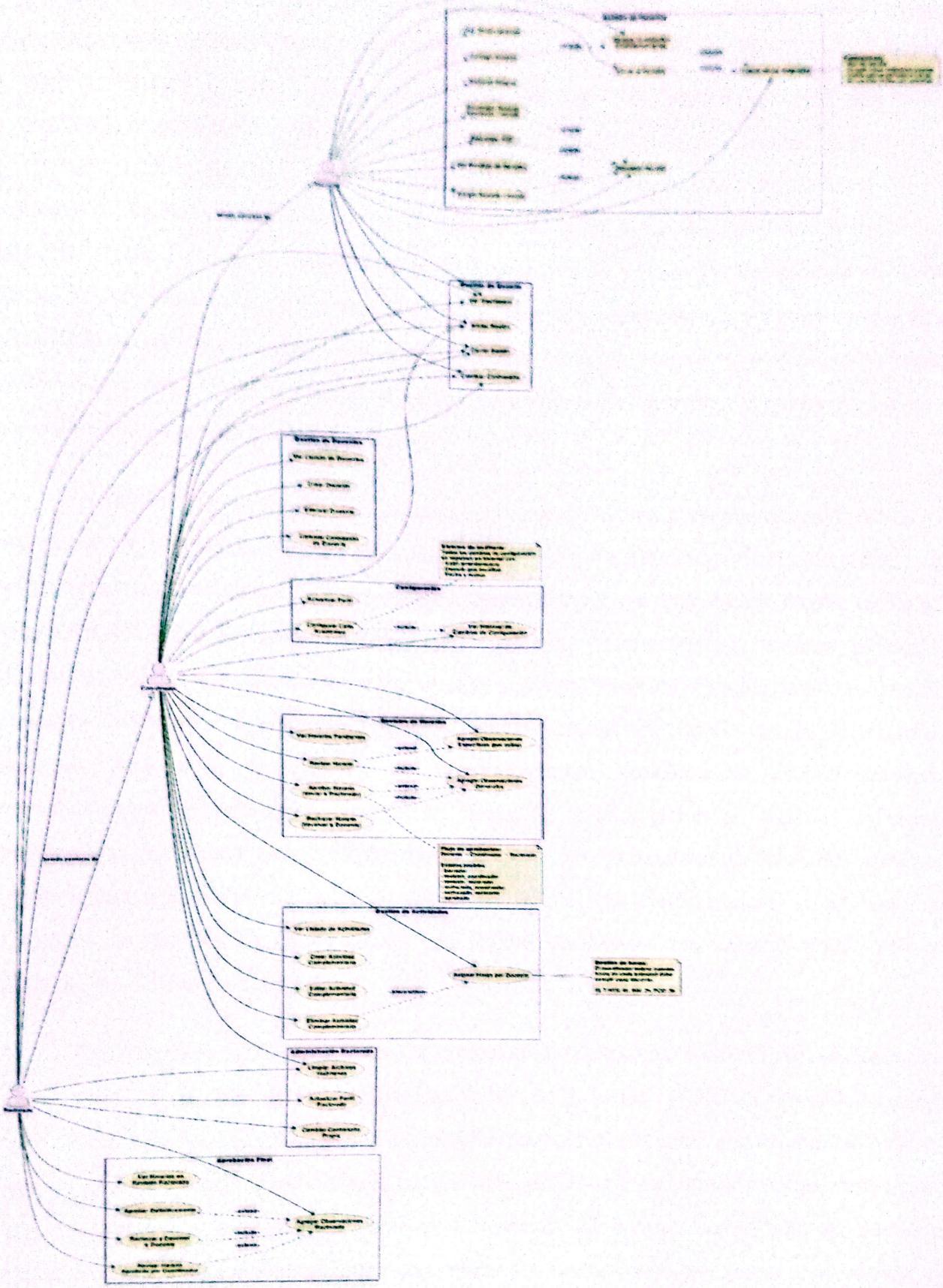
Como resultado del proceso metodológico de análisis, se consolidaron 33 requerimientos funcionales organizados en diez categorías principales que cubrieron el espectro completo de funcionalidades necesarias para automatizar el proceso de gestión de horarios docentes. Adicionalmente, se definieron requerimientos no funcionales en las dimensiones de seguridad, usabilidad, rendimiento y mantenibilidad, estableciendo criterios cuantificables que permitieron posteriormente validar el cumplimiento del sistema desarrollado.

7.1.1. Diagramas de Casos de Uso del Sistema

El diagrama de casos de uso constituyó una representación visual fundamental del sistema de gestión de horarios, donde se ilustraron de manera clara y estructurada las interacciones entre los diferentes actores y las funcionalidades principales del aplicativo web. Este diagrama se elaboró utilizando el lenguaje de modelado PlantUML, lo que permitió generar una documentación técnica precisa y actualizable que reflejó la arquitectura funcional del sistema implementado.

El sistema contempló tres tipos de actores principales con niveles de permisos jerárquicos claramente diferenciados. El actor Docente representó al usuario básico del sistema, quien tuvo la capacidad de cargar sus horarios en formato .hored, visualizarlos de manera interactiva, agregar actividades complementarias según las asignaciones establecidas, y enviar sus horarios para revisión cuando fueron validados correctamente. Por su parte, el actor Coordinador heredó todos los permisos del docente y además contó con capacidades administrativas extendidas que incluyeron la revisión y aprobación de horarios enviados por docentes, la gestión completa del catálogo de actividades complementarias, la administración de usuarios docentes, y la configuración de parámetros del ciclo académico vigente. Finalmente, el actor Rectorado se situó en el nivel más alto de la jerarquía de permisos, heredando las capacidades tanto de docente como de coordinador, y añadiendo la responsabilidad crítica de realizar la aprobación final y definitiva de los horarios académicos.

Ilustración 3 Diagrama de Casos de Uso del Sistema de Gestión de Horarios



Nota. Representación UML de actores, funcionalidades y flujos de aprobación del sistema realizado con PlantUML. Elaborado por Alan García (2025).

El diagrama organizó los casos de uso en paquetes funcionales coherentes que agruparon responsabilidades relacionadas. El paquete de Gestión de Horarios contuvo los casos de uso fundamentales para el trabajo diario del docente, incluyendo la carga de archivos hored con validación automática de estructura, la visualización interactiva del horario semanal, la incorporación de actividades complementarias mediante interacción drag-and-drop, y la validación continua de límites establecidos como el máximo de ocho horas diarias y las restricciones específicas para Tutorías de Acompañamiento Estudiantil. El paquete de Revisión de Horarios agrupó las funcionalidades exclusivas del coordinador relacionadas con el proceso de aprobación, destacando la capacidad de agregar observaciones tanto generales sobre el horario completo como observaciones específicas asociadas a celdas particulares del horario, permitiendo así una retroalimentación precisa y contextualizada.

La gestión administrativa del sistema se organizó en tres paquetes especializados. El paquete de Gestión de Actividades permitió al coordinador mantener actualizado el catálogo de actividades complementarias con sus códigos identificadores, descripciones y límites de horas asignables, así como configurar individualmente para cada docente las horas exactas que debía cumplir en cada tipo de actividad durante el ciclo académico vigente. El paquete de Gestión de Docentes proporcionó operaciones CRUD completas para administrar el registro de usuarios docentes, incluyendo la creación de nuevas cuentas, modificación de información, restablecimiento de contraseñas cuando fue necesario, y eliminación de registros en casos justificados. El paquete de Configuración centralizó los parámetros globales del sistema, particularmente la definición del ciclo académico activo y el mantenimiento de un historial completo de cambios de configuración con registro de timestamps, usuario responsable y justificación del cambio realizado.

Las relaciones entre casos de uso se expresaron mediante estereotipos UML estándar que documentaron dependencias funcionales. La relación de inclusión (<<include>>) indicó que un caso de uso invocaba obligatoriamente a otro como parte de su ejecución normal; por ejemplo, el caso "Cargar Horario (.hored)" incluyó necesariamente "Visualizar Horario" porque después de cargar un archivo el sistema debía mostrar su contenido al usuario. La relación de extensión (<<extend>>) señaló comportamientos opcionales que podían o no ejecutarse dependiendo de

condiciones específicas; así, "Aprobar Horario (enviar a Rectorado)" pudo extenderse opcionalmente con "Agregar Observaciones Generales" si el coordinador decidió incluir comentarios adicionales en su aprobación. Adicionalmente, se identificó una relación de precondición en "Eliminar Actividad Complementaria" que requirió verificar primero mediante "Asignar Horas por Docente" que ningún docente tuviera horas asignadas en esa actividad antes de permitir su eliminación del catálogo.

El flujo de aprobación de horarios siguió una secuencia estructurada de tres estados principales que garantizó múltiples niveles de revisión. El proceso inició cuando el docente completó su horario y lo envió a revisión, momento en el cual el horario transitó del estado "borrador" al estado "revisión_coordinador". El coordinador realizó la primera evaluación técnica y pudo tomar dos decisiones: rechazar el horario devolviéndolo al docente con observaciones específicas para corrección (estado "rechazado_coordinador"), o aprobarlo y enviarlo al siguiente nivel de revisión (estado "revisión_rectorado"). En esta segunda etapa, rectorado efectuó la revisión final pudiendo igualmente rechazar el horario para devolverlo directamente al docente (estado "rechazado_rectorado"), o aprobar definitivamente el documento (estado "aprobado"), momento a partir del cual el docente pudo descargar tanto el archivo .hored oficial como la versión PDF para archivo institucional.

El sistema implementó validaciones automáticas críticas que se ejecutaron en tiempo real durante la edición del horario, proporcionando retroalimentación inmediata al usuario mediante indicadores visuales. La validación del límite máximo de ocho horas por día se calculó sumando tanto las clases regulares como todas las actividades complementarias asignadas en cada jornada, mostrando alertas progresivas cuando se aproximaba al límite (color amarillo) y alertas de error cuando se excedía (color rojo). La restricción específica para Tutorías de Acompañamiento Estudiantil (T/AES) verificó que estas actividades únicamente se asignaran en los períodos académicos 6, 7, 9 y 10 del horario semanal, bloqueando la asignación en otros períodos mediante validación de arrastre. Adicionalmente, el sistema controló que cada docente no excediera los límites individuales configurados por el coordinador para cada tipo de actividad complementaria, calculando en tiempo real el total acumulado de horas asignadas y comparándolo contra el límite establecido.

Este diagrama de casos de uso fundamentó el diseño arquitectónico del sistema y sirvió como guía durante todo el proceso de desarrollo, asegurando que cada funcionalidad implementada respondiera efectivamente a una necesidad documentada de los usuarios y mantuviera coherencia con el modelo conceptual establecido durante la fase de análisis de requerimientos.

7.1.2. Requerimientos Funcionales del Sistema

Tabla 2 Requerimientos Funcionales del Sistema

Categoría	Requerimiento	Descripción
Gestión de Archivos	Carga de archivos .hored	Permitir la carga de archivos en formato .hored con validación de integridad y estructura
Gestión de Archivos	Descarga de archivos .hored	Permitir la descarga de archivos .hored únicamente para horarios con estado "aprobado"
Gestión de Usuarios	Autenticación multi-rol	Sistema de login con tres niveles de acceso: docente, coordinador y rectorado
Gestión de Usuarios	Gestión de sesiones	Mantener sesiones activas con expiración automática tras inactividad
Gestión de Usuarios	Cambio de contraseña	Permitir a los usuarios cambiar su contraseña de forma segura
Edición de Horarios	Visualización interactiva	Mostrar horarios en formato de tabla con información de períodos, días y actividades
Edición de Horarios	Drag & Drop de actividades	Permitir agregar actividades complementarias mediante arrastrar y soltar
Edición de Horarios	Fusión automática	Fusionar automáticamente actividades idénticas en períodos consecutivos
Edición de Horarios	Guardado de cambios	Guardar cambios realizados en el horario de forma automática mediante AJAX
Validaciones	Límite de 8 horas/día	Validar que la suma de clases y actividades no exceda 8 horas por día
Validaciones	Restricción T/AES	Validar que actividades T/AES solo se asignen en períodos 6, 7, 9 y 10

Categoría	Requerimiento	Descripción
Validaciones	Límites por actividad	Verificar que cada docente no exceda los límites asignados por tipo de actividad
Validaciones	Prevención de conflictos	Impedir la asignación de actividades en celdas ya ocupadas
Validaciones	Notificaciones en tiempo real	Mostrar alertas visuales cuando se aproxime o exceda límites establecidos
Flujo de Aprobación	Gestión de estados	Administrar estados: borrador, revisión_coordinador, rechazado_coordinador, revisión_rectorado, rechazado_rectorado, aprobado
Flujo de Aprobación	Envío a revisión	Permitir al docente enviar horario a revisión del coordinador
Flujo de Aprobación	Revisión coordinador	Permitir al coordinador aprobar (enviar a rectorado) o rechazar (devolver a docente)
Flujo de Aprobación	Revisión rectorada	Permitir al rectorado aprobar definitivamente o rechazar horarios
Sistema de Observaciones	Observaciones generales	Permitir agregar comentarios generales sobre el horario completo
Sistema de Observaciones	Observaciones específicas	Permitir agregar comentarios en celdas específicas del horario
Sistema de Observaciones	Visualización de observaciones	Mostrar al docente todas las observaciones realizadas por revisores
Sistema de Observaciones	Historial de observaciones	Mantener registro de todas las observaciones con timestamps y revisor
Generación de Documentos	Generación de documentos	Generar documentos PDF oficiales con formato institucional utilizando jsPDF y html2canvas en el lado del cliente, incluyendo logotipo, datos del docente, tabla de horarios y sección de firmas.
Generación de Documentos	Vista previa	Mostrar vista previa antes de la descarga
Gestión Administrativa	CRUD de docentes	Permitir al coordinador crear, leer, actualizar y eliminar docentes

Categoría	Requerimiento	Descripción
Gestión Administrativa	CRUD de actividades	Gestionar actividades complementarias con sus códigos y descripciones
Gestión Administrativa	Asignación de límites	Configurar límites de horas por actividad y por docente
Gestión Administrativa	Configuración de ciclo	Definir ciclo académico (ejemplo: 2025-IIS)
Auditoría	Registro de acciones	Mantener log de todas las acciones realizadas en el sistema
Auditoría	Timestamps	Registrar fecha y hora de cada cambio de estado y modificación
Dashboard	Panel docente	Mostrar al docente sus horarios con estados y opciones disponibles
Dashboard	Panel coordinador	Mostrar lista de horarios pendientes de revisión con información relevante
Dashboard	Panel rectorado	Mostrar horarios pre-aprobados pendientes de aprobación final

Elaborado por: Alan García (2025)

7.1.3. Requerimientos No Funcionales Implementados

Tabla 3 Requerimientos No Funcionales Implementados

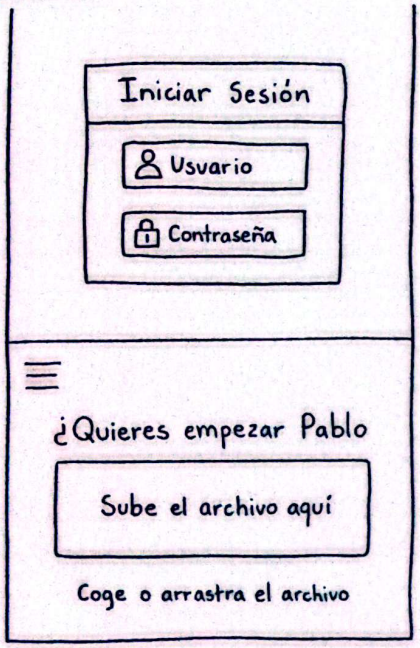
Dimensión	Requerimientos Implementados
Seguridad	Cookies seguras: HTTPOnly, SameSite=Lax, Secure; y validar sesión en cada petición con decoradores. Validación estricta doble: Python (servidor) y JavaScript (cliente). Contraseñas hasheadas con hashlib.sha256 antes de almacenar. Sesiones configuradas con HttpOnly y SameSite para mitigar ataques CSRF.
Usabilidad	Interfaz intuitiva validada con usuarios reales en diseño iterativo. Compatibilidad cross-browser (Google Chrome y Mozilla Firefox recientes). Indicadores visuales claros con CSS (códigos de color e iconografía consistente). Mensajes de error comprensibles, sin jerga técnica y con guías de corrección.
Rendimiento	Tiempo de respuesta < 2 segundos en operaciones comunes (profiling Python y Chrome DevTools). Generación de PDF en cliente, reduciendo carga en servidor y dependencia de ancho de banda. Índices en campos de búsqueda frecuente (CREATE INDEX en horarios, cédula y claves foráneas) para optimizar consultas JOIN.
Mantenibilidad	Código documentado con comentarios explicativos en español. Uso de SQL directo sin ORM para control y optimización. Separación clara de capas siguiendo arquitectura MVC, aislando cambios en módulos específicos.

Elaborado por: Alan García (2025)

7.2. Diseño de Interfaz Web Intuitiva

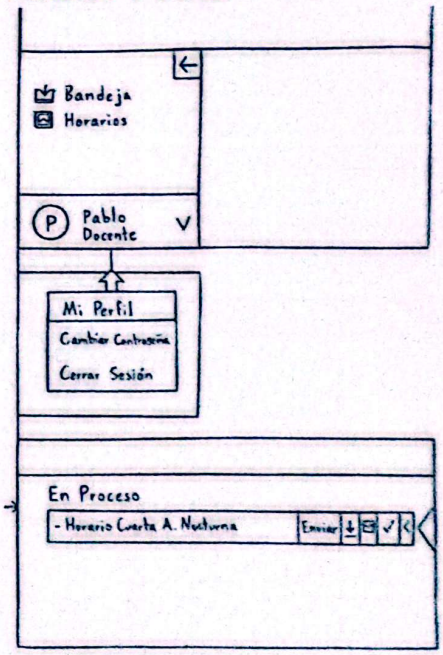
El diseño de la interfaz web se desarrolló aplicando un enfoque centrado en el usuario que situó las necesidades y capacidades de los usuarios finales como eje central del proceso creativo. El trabajo comenzó con la elaboración de bocetos simples en papel de las pantallas principales del sistema, utilizando esquemas compuestos únicamente por cajas rectangulares representando componentes de interfaz, flechas indicando flujo de navegación, y anotaciones textuales describiendo funcionalidad. Estos wireframes de baja fidelidad permitieron visualizar la conexión entre los distintos componentes del sistema: la interfaz para subir archivos .hored con indicadores de validación, la representación visual de la tabla de horarios semanal con períodos en filas y días en columnas, el mecanismo de asignación de actividades mediante acciones de arrastrar y soltar desde una barra lateral flotante, y el flujo correspondiente de revisión con estados diferenciados y sistema de observaciones. Este ejercicio inicial de diseño conceptual fue fundamental para garantizar que la solución propuesta respondiera de manera lógica e intuitiva a las expectativas de los usuarios antes de invertir recursos en diseños detallados o implementación.

Ilustración 5 Boceto De La Interfaz De Autenticación Y Carga De Horarios



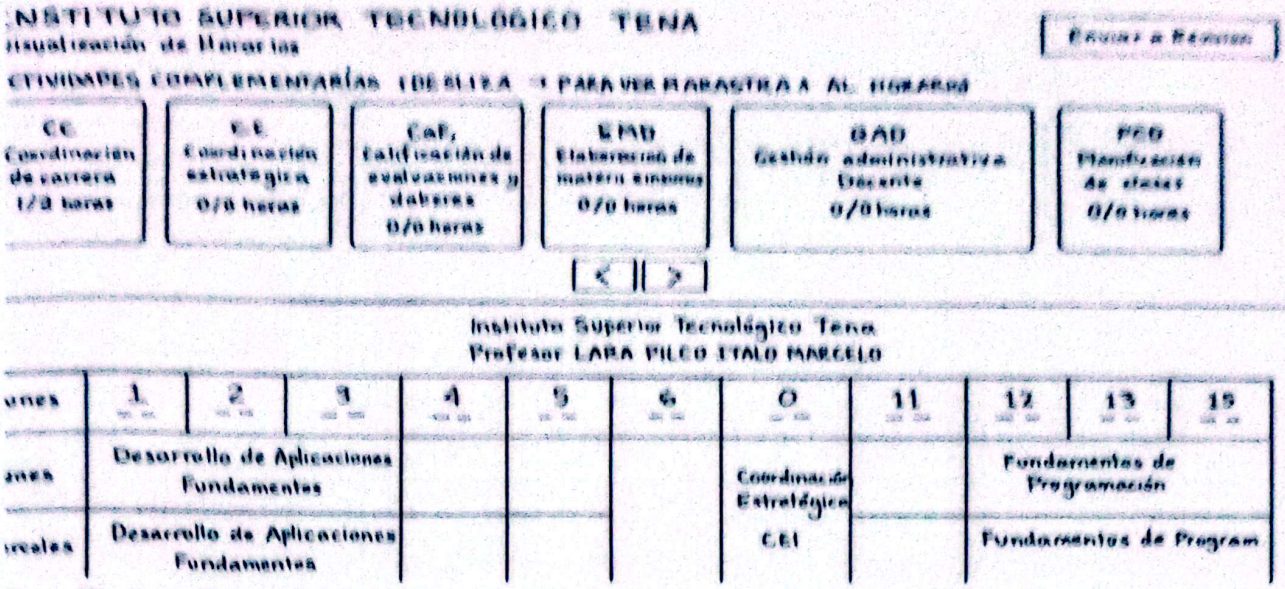
Nota. Diseño preliminar de login y módulo de carga de horarios; Elaborado por Alan García (2025).

Ilustración 4 Boceto De La Bandeja Docente Para La Gestión De Horarios Y Perfil



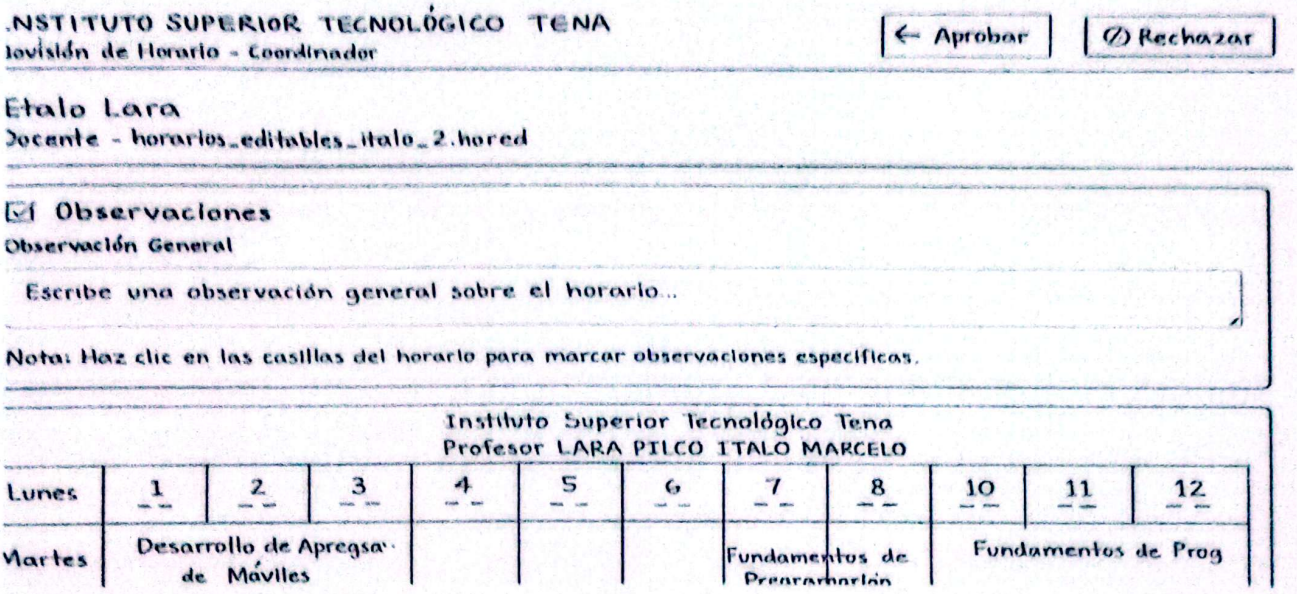
Nota. Diseño preliminar para gestionar horarios y perfil; Elaborado por Alan García (2025).

Ilustración 6 Boceto De La Interfaz Edición de Horarios Docente



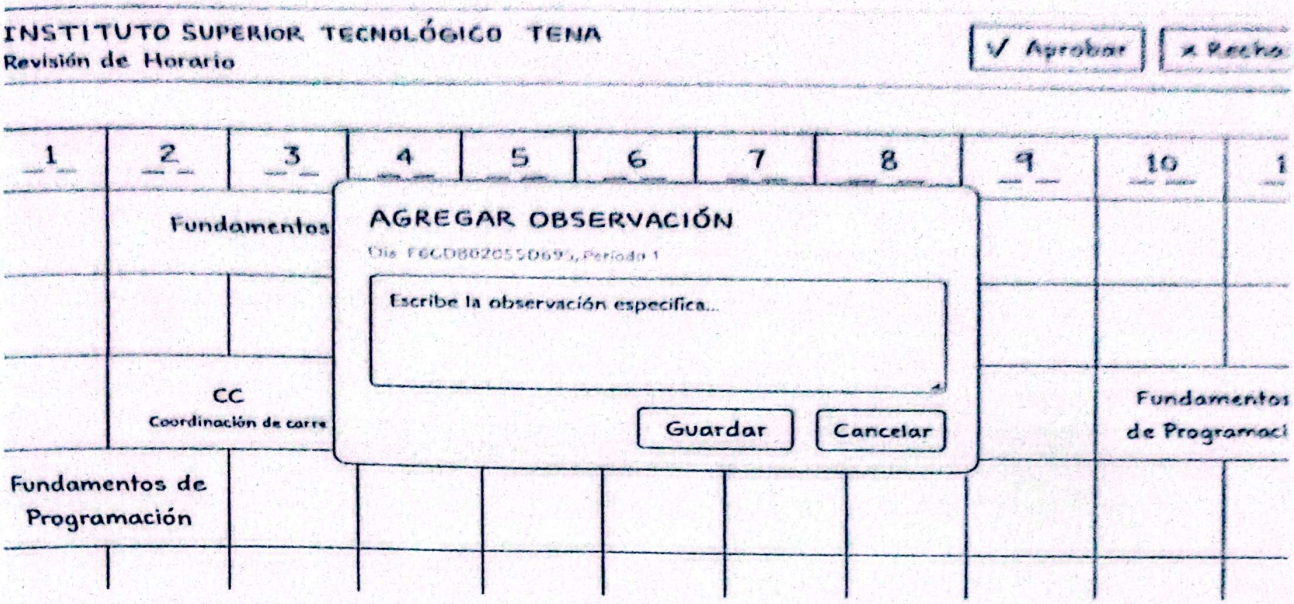
Nota. Diseño preliminar de la interfaz de edición de horarios para envió a revisión; Elaborado por Alan García (2025).

Ilustración 7 Boceto De La Interfaz Coordinación: Revisión De Horarios con Observaciones



Nota. Diseño preliminar de la interfaz de revisión de horarios docentes del Coordinador de Carrera; Elaborado por Alan García (2025).

Ilustración 8 Boceto De La Interfaz Coordinación: Observaciones Específicas Por Celdas



Nota. Diseño preliminar de la interfaz de revisión de horarios docentes del Coordinador de Carrera con observaciones especificas por celda; Elaborado por Alan García (2025).

Ilustración 9 Formato Hored (Horarios Editables)

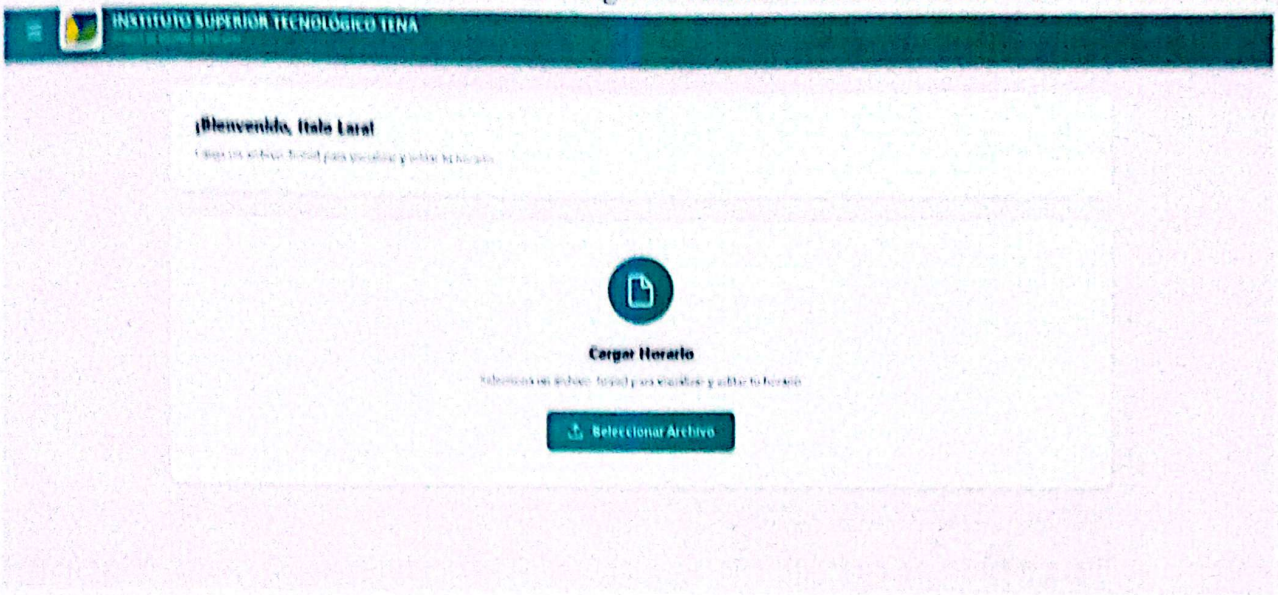
Instituto Superior Tecnológico Tena
Profesor QUILUMBA SHIGUANGO SALOMÓN ISAAC

	2 8:00 - 9:00	4 10:00 - 11:00	6 12:00 - 13:00	8 14:00 - 15:00	10 16:00 - 17:00	12 18:00 - 19:00	14 20:00 - 21:00
Lunes		Proyecto de Titulación TSDSW_Quinto_A_Matutina					
Martes	METODOLOGIA DE DESARROLLO DE SISTEMAS DSW_Segundo_A_Matutina	Proyecto de Titulación TSDSW_Quinto_A_Matutina / Tendencias Actuales de Programación TSDSW_Quinto_A_Matutina			CORD. ESTRAT CE1		
Miércoles	METODOLOGIA DE DESARROLLO DE SISTEMAS DSW_Segundo_A_Matutina	Tendencias Actuales de Programación TSDSW_Quinto_A_Matutina					
Jueves		METODOLOGIA DE DESARROLLO DE SISTEMAS DSW_Segundo_A_Matutina					
Viernes	Tendencias Actuales de Programación TSDSW_Quinto_A_Matutina						
Sábado							

Nota. Visualización del formato hored del proyecto web; Elaborado por Alan García (2025).

Con base en las especificaciones recibidas durante las sesiones de revisión de wireframes, se realizaron ajustes iterativos que dieron lugar a diseños de mayor fidelidad visual y funcional. En esta etapa se definieron los colores institucionales del Instituto Superior Tecnológico Tena, incorporando específicamente los tonos #0097A7 (cian principal) y #00838F (cian oscuro) como colores primarios que transmiten profesionalismo y coinciden con la identidad corporativa existente. Se añadió además un matiz de originalidad mediante el uso de degradados sutiles y sombras CSS para comunicar al mismo tiempo familiaridad institucional y renovación tecnológica.

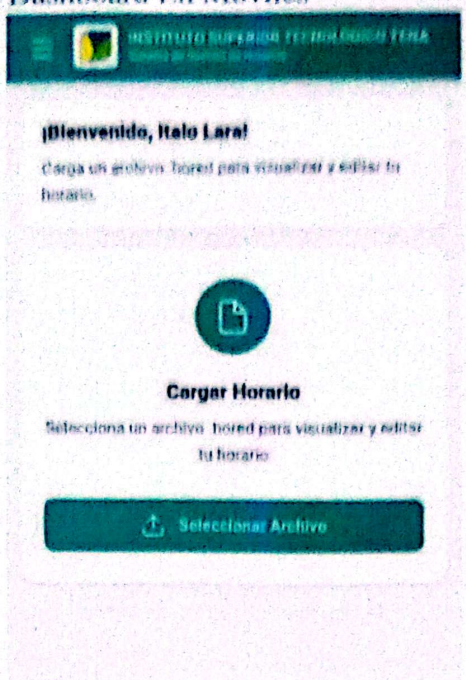
Ilustración 10 Vista Inicial Del Módulo De Carga De Horarios En El Sistema Web – Vista Escritorio



Nota. Interfaz docente de bienvenida para cargar archivos . hored; Elaborado por Alan García (2025).

Se seleccionaron tipografías que garantizaran legibilidad óptima tanto en pantallas de escritorio como en dispositivos móviles, eligiendo finalmente Segoe UI como familia principal por su diseño optimizado para lectura en pantalla y amplia disponibilidad en sistemas operativos modernos. Se diseñó un conjunto consistente de iconografía en formato SVG inline para las acciones más comunes del sistema, incluyendo representaciones visuales para subir archivo, editar horario, agregar actividad, eliminar asignación, aprobar, rechazar, descargar PDF, y cerrar sesión, con el objetivo de facilitar el reconocimiento visual de funcionalidades y reducir la dependencia de texto descriptivo extenso.

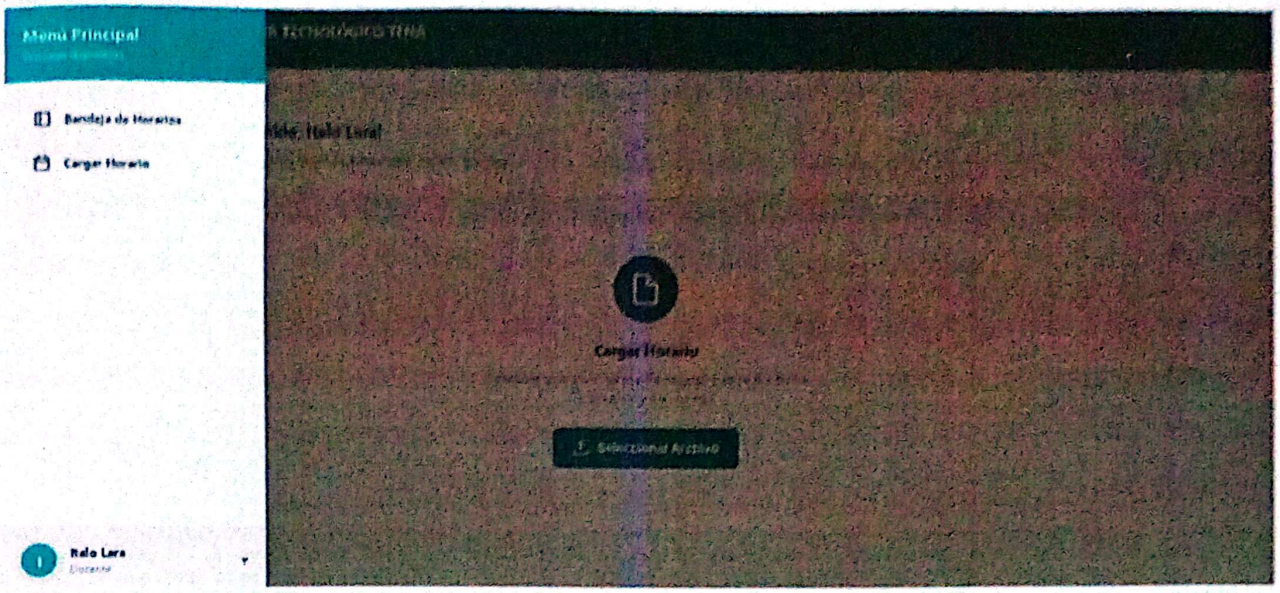
Ilustración 11 Vista HTML Dashboard En Móviles



Nota. Interfaz para cargar y editar el horario Docente; Elaborado por Alan García (2025).

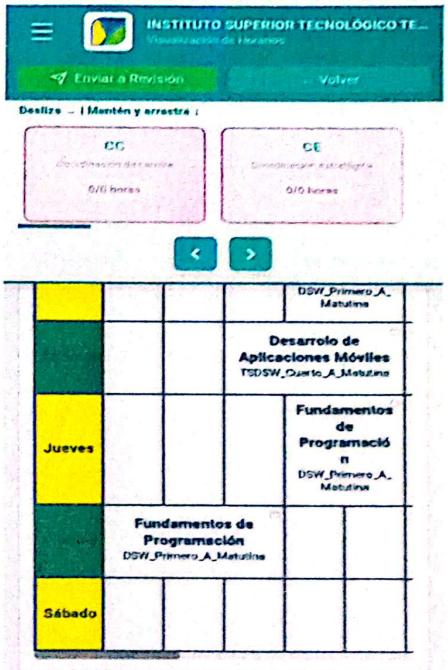
La implementación técnica del diseño se realizó utilizando CSS3 completamente personalizado, desarrollando desde cero todos los componentes visuales sin dependencia de frameworks como Bootstrap o Materialize, lo cual permitió garantizar control total sobre la apariencia visual, optimización del tamaño de carga eliminando código innecesario, y adaptabilidad precisa a diferentes tamaños de pantalla. Se implementaron tres breakpoints principales mediante media queries CSS: 768px para tablets donde los elementos de navegación se reorganizaron en disposición optimizada, 480px para smartphones grandes donde la barra lateral se convirtió en menú desplegable tipo hamburguesa y las tablas de horario se hicieron scrollables horizontalmente, y 360px para smartphones pequeños donde se priorizó diseño vertical y se simplificaron controles táctiles aumentando áreas de toque. El sistema de grillas responsivo se construyó mediante CSS Grid y Flexbox nativos, evitando dependencias externas y permitiendo ajustes precisos para cada contexto de visualización.

Ilustración 12 Vista Menú - Pantalla Principal Docente



Nota. Vista del menú de la pantalla principal docente; Elaborado por Alan García (2025).

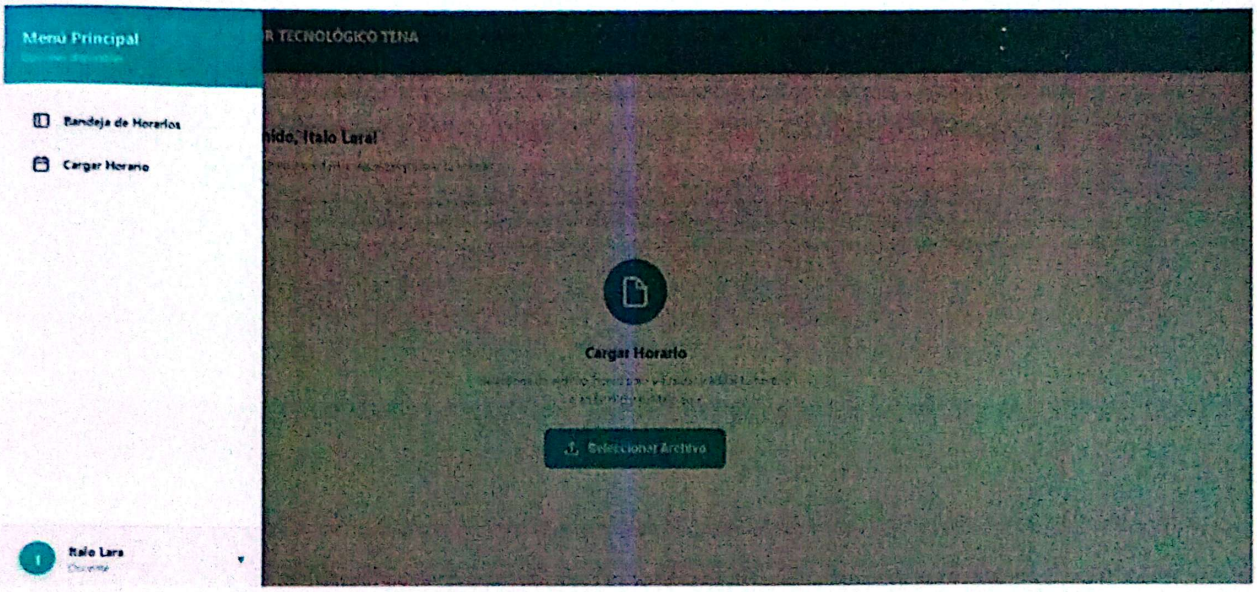
Ilustración 13 Vista Edición Horario Docente



Nota. Vista de edición de horarios scrollables horizontalmente; Elaborado por Alan García (2025).

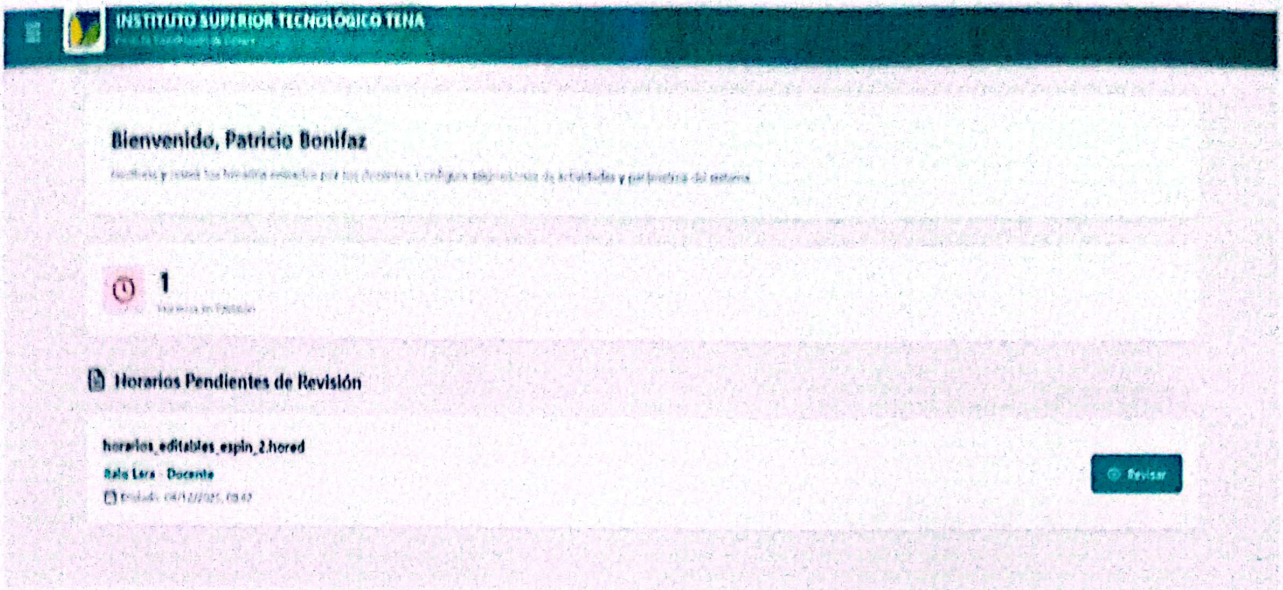
Se desarrollaron prototipos funcionales en HTML que se evaluaron con usuarios reales en cinco sesiones de pruebas de usabilidad realizadas durante el proceso de diseño. La primera sesión contó con la participación del coordinador de carrera y del responsable de la generación de horarios del ISTT; ambos interactuaron con la interfaz de carga de archivos y la visualización inicial de horarios, aplicando el método de pensamiento en voz alta para verbalizar sus expectativas y dudas durante la interacción. Las sesiones siguientes incluyeron a dos docentes sin experiencia técnica previa, quienes probaron la funcionalidad de edición mediante arrastrar y soltar. Posteriormente, el equipo de rectorado evaluó el flujo de revisión y aprobación. Finalmente, se realizó una sesión de validación del sistema completo con el coordinador de carrera y el encargado de horarios.

Ilustración 14 Visualización De Horarios



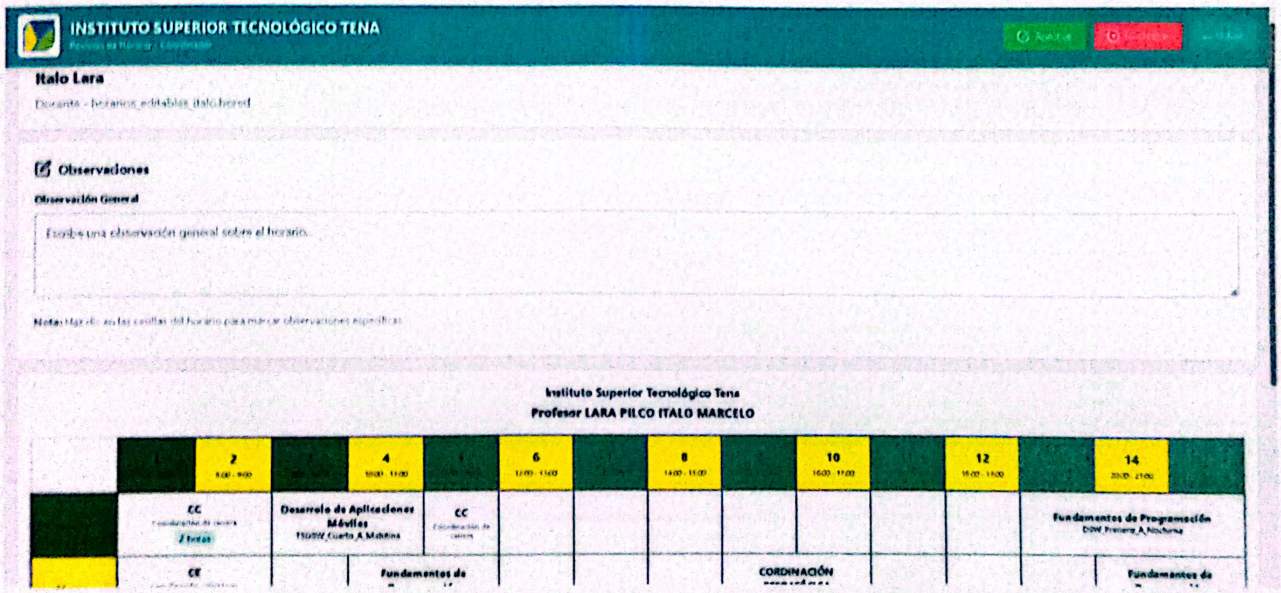
Nota. Vista docente de los horarios .hored cargados para su edición; Elaborado por Alan García (2025).

Ilustración 15 Bandeja de Horarios Pendientes de Revisión para Coordinación



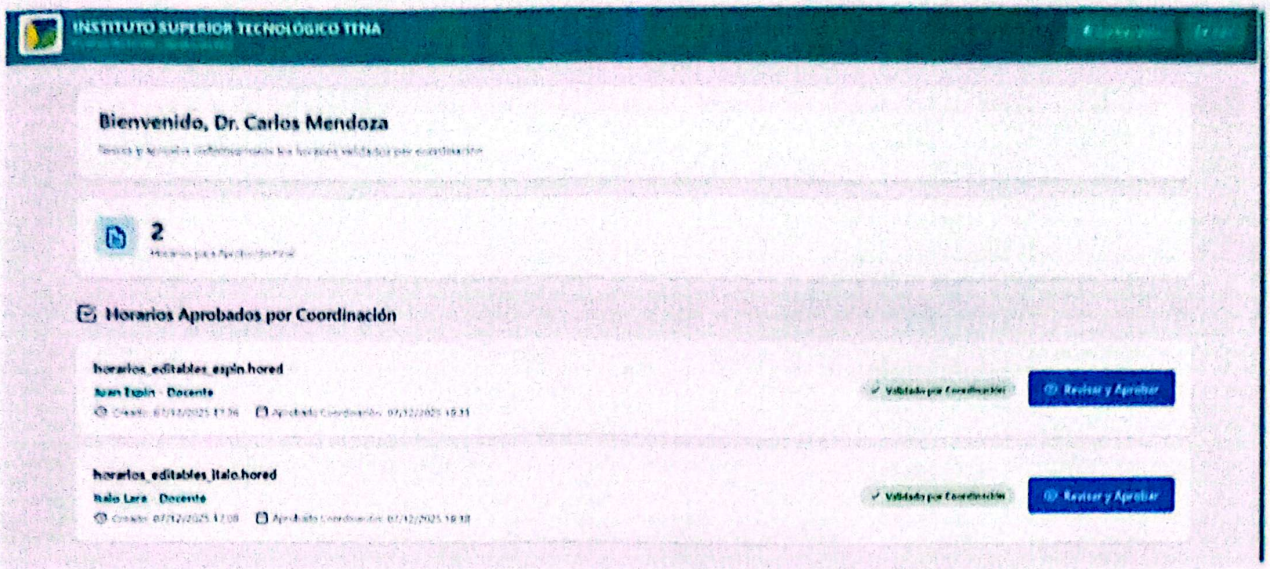
Nota. Bandeja del coordinador con horarios enviados por docentes para validar; Elaborado por Alan García (2025).

Ilustración 16 Vista Coordinador: Visualización de Horarios Docentes Para Revisión



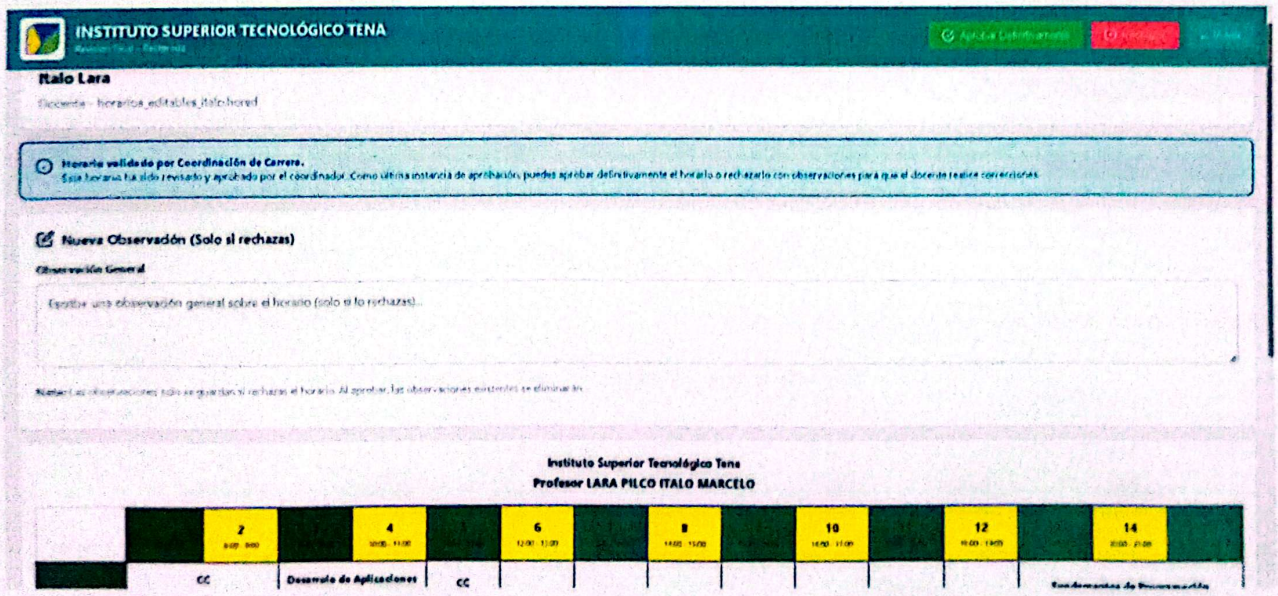
Nota. Vista del coordinador para la revisión de horarios docentes; Elaborado por Alan García (2025).

Ilustración 17 Bandeja de Horarios Pendientes de Revisión para Rectorado



Nota. Bandeja de Rectorado con horarios revisados y aprobados por el Coordinador de carrera para dar la aprobación final; Elaborado por Alan García (2025).

Ilustración 18 Vista Rectorado: Visualización de Horarios Docentes Para Revisión y Aprobación Final



Nota. Vista de rectorado para la revisión y aprobación final de horarios docentes; Elaborado por Alan García (2025).

Durante estas sesiones se registraron sistemáticamente las dificultades de comprensión, errores cometidos, tiempo requerido para completar tareas específicas, y sugerencias explícitas de mejora. Los hallazgos fueron analizados para identificar patrones recurrentes de problemas de usabilidad, priorizándolos según frecuencia y severidad, e implementando ajustes iterativos que crearon nuevas versiones del prototipo validadas en ciclos adicionales hasta alcanzar niveles satisfactorios de facilidad de uso.

El resultado del proceso de diseño iterativo y validado empíricamente fue el desarrollo de interfaces específicas diferenciadas para cada rol dentro del sistema. La interfaz del docente incluye los siguientes componentes principales organizados de manera intuitiva:

Tabla 4 Componentes Principales de la Interfaz según Rol

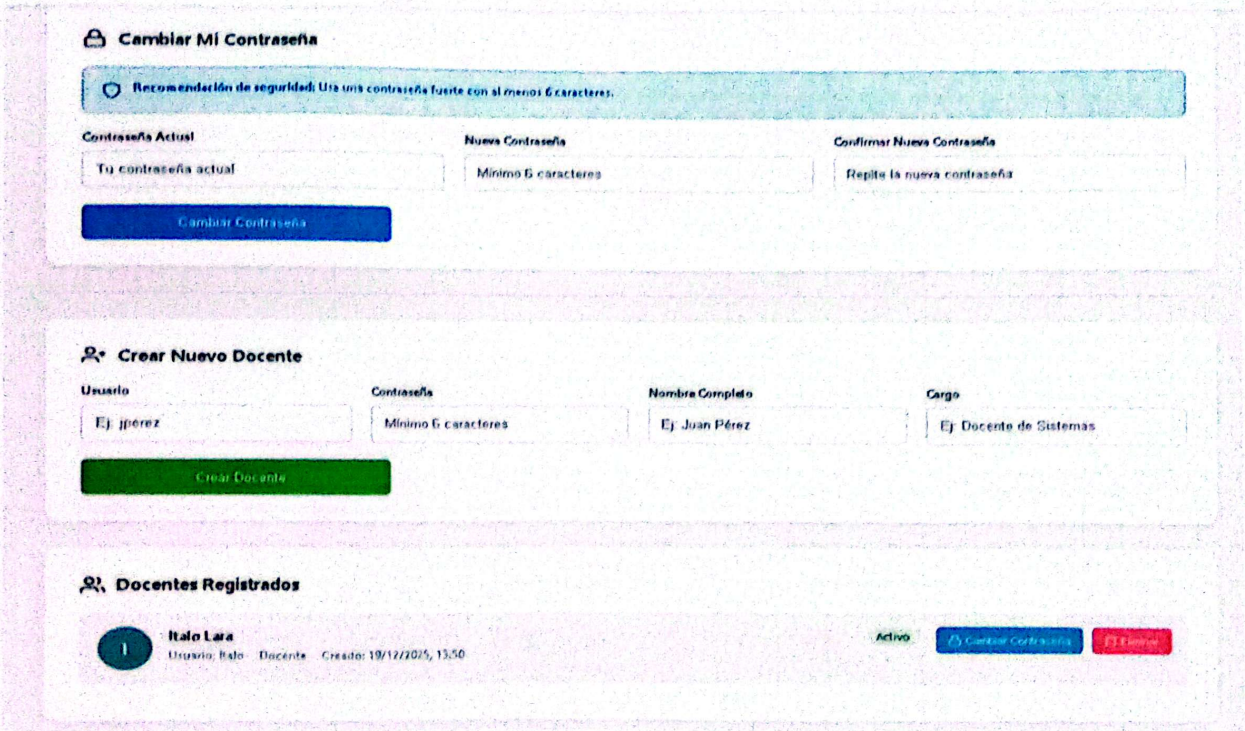
Rol	Componentes principales de la interfaz
Docente	• Panel de bienvenida personalizada con nombre completo y cargo • Cargador de archivos .hored con validación JSON y estructura "Horario Editable - aSc TimeTables" • Bandeja de horarios con estados: borrador; revision_coordinador; revision_rectorado; rechazado_coordinador; rechazado_rectorado; aprobado • Editor interactivo con Drag & Drop API HTML5 para arrastrar actividades complementarias • Barra lateral con actividades complementarias disponibles y control de horas asignadas/consumidas (X/límite horas) • Indicador de horas por día debajo de cada columna (X/8 horas) con código de color dinámico (ok; warning; exceeded) • Visualización de observaciones del coordinador/rectorado: generales y específicas por celda • Panel para cambiar la contraseña del docente con validaciones (mínimo 6 caracteres)

Rol	Componentes principales de la interfaz
	• Descarga de horarios aprobados en formato .hored.
Coordinador	• Dashboard con lista de horarios pendientes de revisión (estado revision_coordinador) ordenados por fecha de envío. • Vista de revisión individual con capacidad de agregar observaciones generales y específicas por celda (día, período). • Panel "Mi Perfil" para actualizar información personal: usuario (login), nombre completo y cargo institucional, con validaciones (mínimo 3 caracteres) y confirmación mediante contraseña actual. • Sección independiente para cambiar contraseña propia (contraseña actual + nueva + confirmación, mínimo 6 caracteres). • Panel CRUD de docentes: crear (validación de usuario único), listar, eliminar (eliminación en cascada de horarios y asignaciones), resetear contraseñas. • Módulo de gestión de actividades complementarias: crear, editar (código y descripción), eliminar. • Módulo de asignación de límites de horas por actividad a cada docente. • Configurador de ciclo académico con historial de cambios para auditoría. • Botones para aprobar (envía a rectorado) o rechazar (devuelve a borrador con observaciones).
Rectorado	• Dashboard de aprobación final que muestra solo horarios en revision_rectorado ordenados por fecha de revisión del coordinador • Vista de lectura del horario completo con información del docente (nombre, cargo)

Rol	Componentes principales de la interfaz
	• Sistema independiente de observaciones (comentarios generales y específicos separados de los del coordinador) para trazabilidad multi-etapa • Controles de aprobación definitiva: aprobar (cambia estado a aprobado y habilita descarga) o rechazar (cambia estado a rechazado_rectorado, docente debe corregir) • Panel de configuración del perfil de rectorado (usuario, nombre, cargo, contraseña).

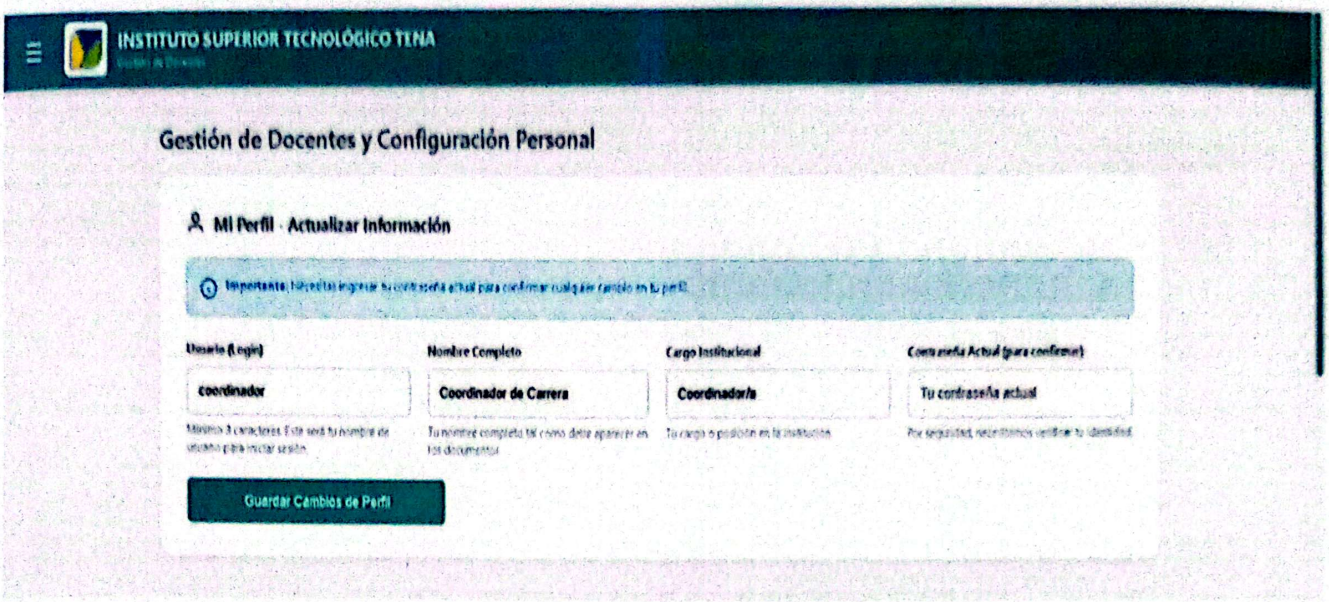
Elaborado por: Alan García (2025)

Ilustración 19 Vista Coordinador: Modificación Y Actualización De Contraseña Propia Y Creación de Cuentas Docentes



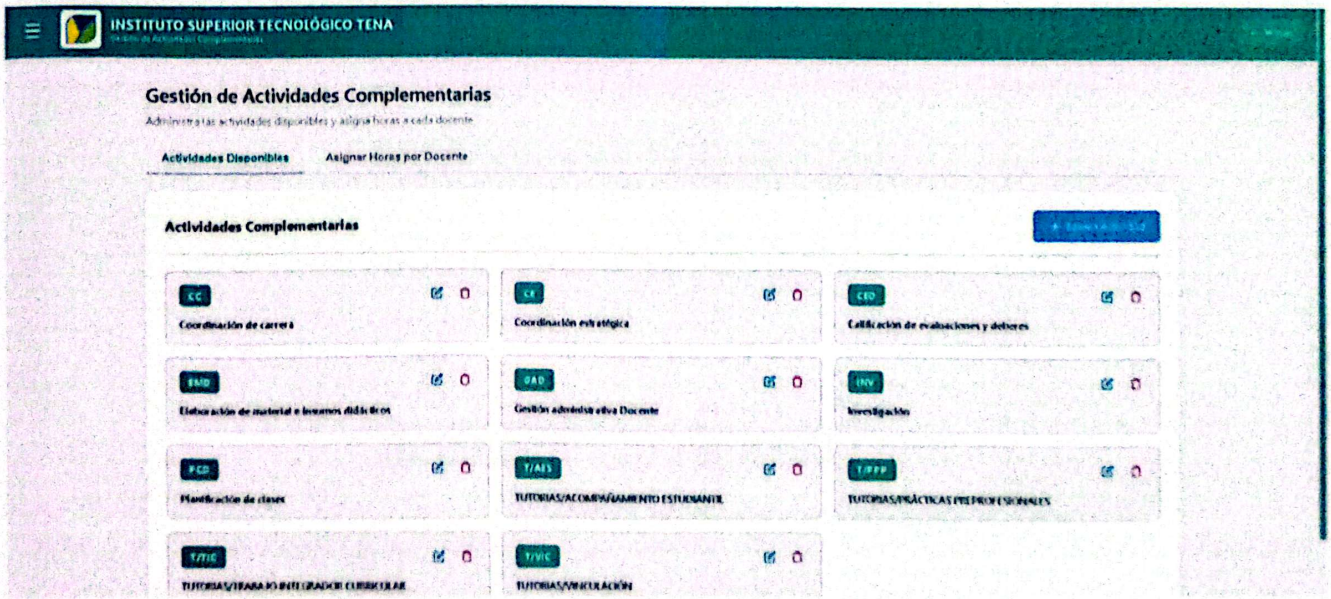
Nota. Esta vista facilita la administración y control de acceso a la plataforma para el coordinador; Elaborado por Alan García (2025).

Ilustración 20 Vista Coordinador: Actualización Y Modificación De Datos Personales



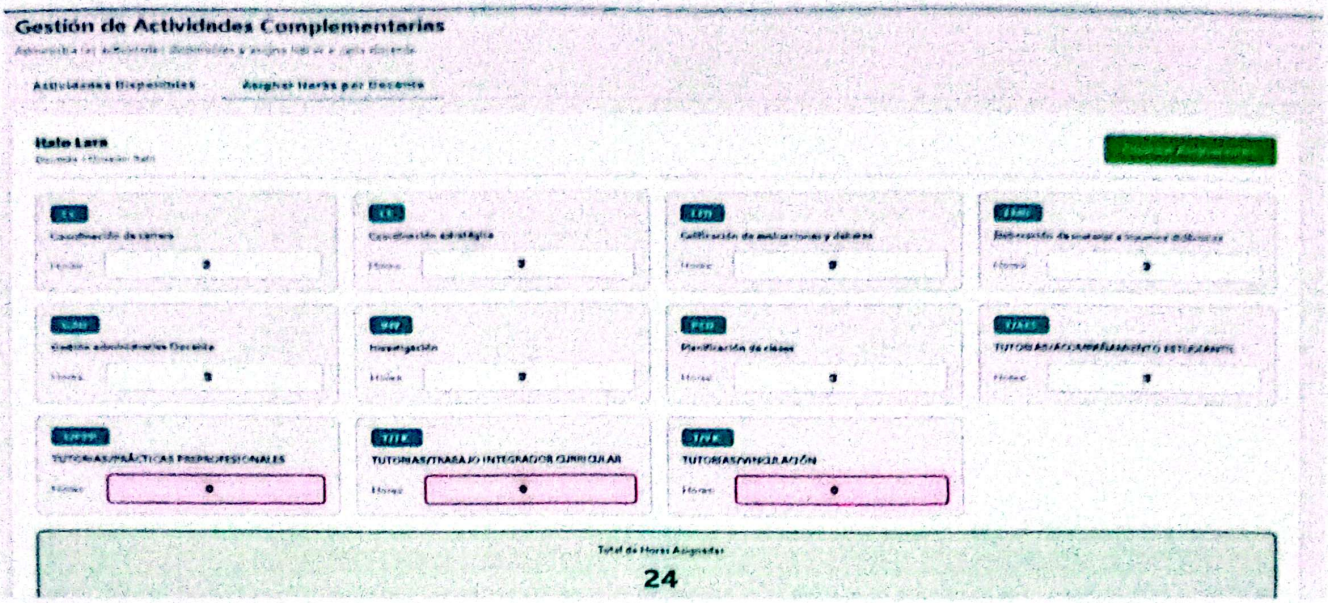
Nota. Vista del coordinador para actualizar y modificar los datos del perfil propio; Elaborado por Alan García (2025).

Ilustración 21 Vista Coordinador: Creación De Actividades Complementarias



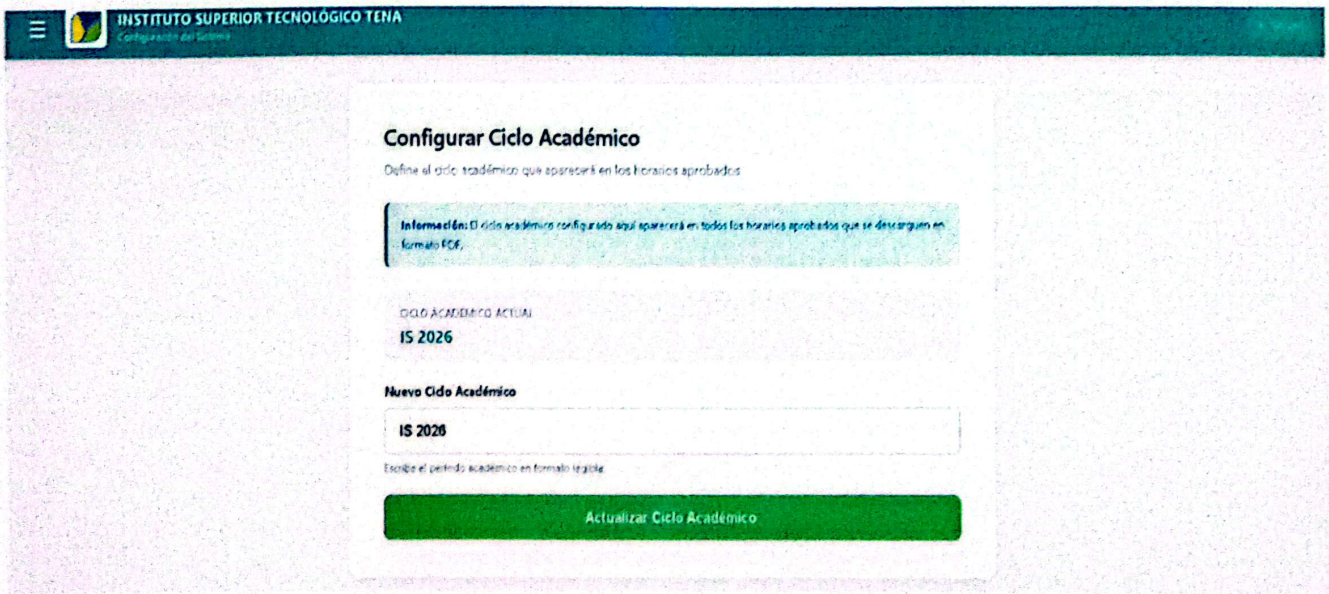
Nota. Vista del coordinador para crear actividades complementarias, además de poder editarlos y eliminarlos; Elaborado por Alan García (2025).

Ilustración 22 Vista Coordinador: Asignación De Actividades Complementarias Para Cada Docente



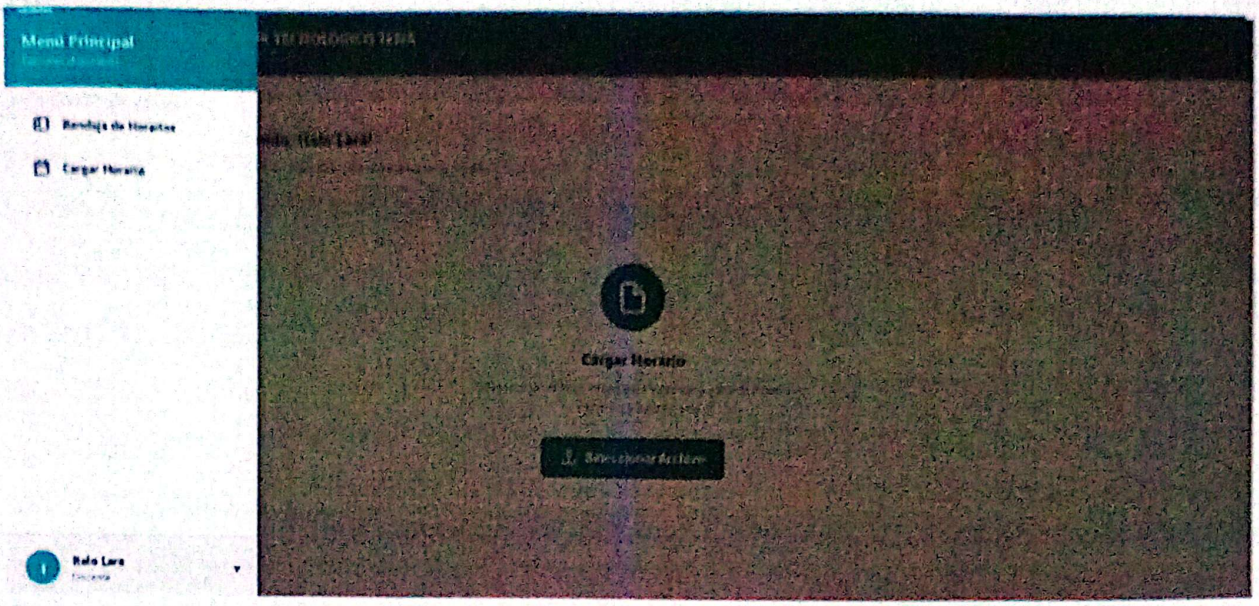
Nota. Vista del coordinador para asignar actividades complementarias para cada docente que exista y que este creado; Elaborado por Alan García (2025).

Ilustración 23 Vista Coordinador: Configuración Del Ciclo Académico



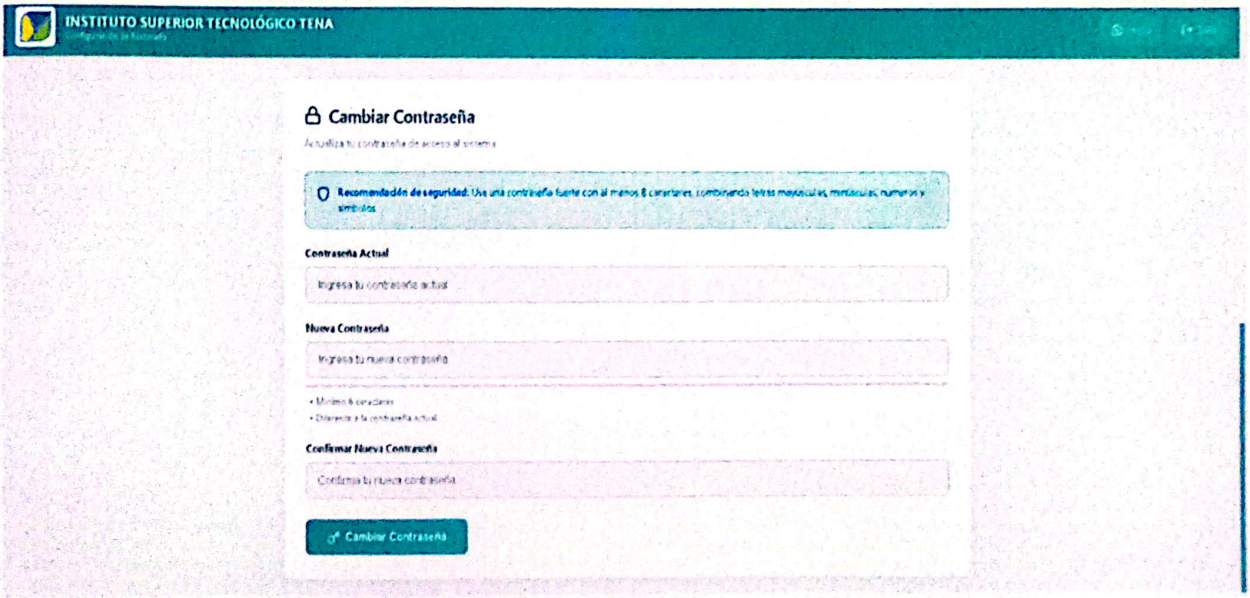
Nota. Vista del coordinador para configurar el ciclo académico de los horarios finales; Elaborado por Alan García (2025).

Ilustración 24 Vista Rectorado: Modificación Y Actualización Del Perfil Propio



Nota. Vista de rectorado para actualizar datos personales del perfil propio; Elaborado por Alan García (2025).

Ilustración 25 Vista Rectorado: Actualización Y Modificación De Contraseña Propia



Nota. Vista rectorado para actualizar la contraseña propia; Elaborado por Alan García (2025).

Los principios de diseño aplicados de manera consistente en todas las interfaces incluyen el uso coherente de los colores institucionales #0097A7 y #00838F en encabezados, botones de acción principal y elementos de navegación; iconografía SVG inline clara y consistente para representar acciones comunes; navegación intuitiva mediante sidebar CSS que se colapsa en dispositivos móviles; transiciones CSS suaves entre estados para proporcionar feedback visual de cambios; tipografía Segoe UI con jerarquía visual clara diferenciando títulos, encabezados y texto de cuerpo; y espaciado responsivo que se ajusta en los breakpoints 768px, 480px y 360px manteniendo proporciones armónicas y facilidad de interacción. La validación con usuarios reales involucró al coordinador de carrera quien confirmó intuitividad en gestión de docentes y actividades, dos docentes participantes quienes validaron facilidad en carga y edición de horarios, además de rectorado quien verificó fluidez del flujo de aprobación. El resultado fue una interfaz aprobada con ajustes mínimos, considerada funcional y amigable por todos los perfiles de usuarios evaluados.

7.2.1. Frontend y Experiencia de Usuario

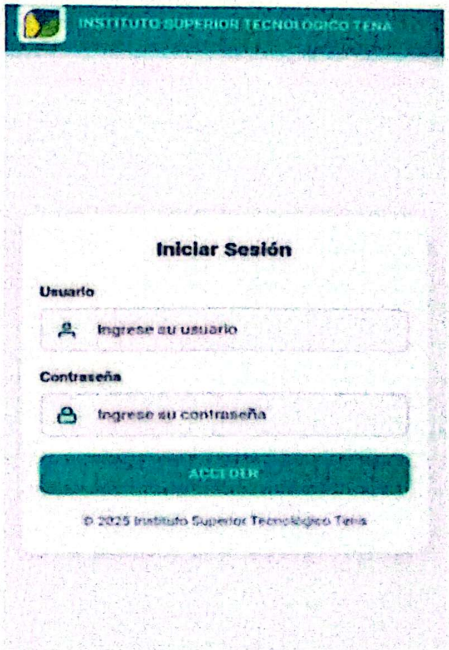
Se implementó una interfaz completamente responsiva utilizando HTML5 semántico con elementos como header, nav, main, section y footer para mejorar accesibilidad y estructura lógica; CSS3 completamente personalizado desarrollado desde cero sin frameworks CSS externos para mantener control total sobre diseño y optimizar tamaño de carga; y JavaScript vanilla puro sin librerías externas para toda la lógica del cliente.

Se desarrollaron quince plantillas HTML diferenciadas: login para autenticación, dashboard mostrando panel principal del docente, bandeja_docente para gestión de lista de horarios, visualizar proporcionando edición interactiva con drag & drop, descargar con vista previa y generación de PDF, cambiar_password para gestión de seguridad, coordinador_dashboard mostrando panel de revisor, coordinador_revisar para interfaz de revisión con observaciones, crear_docente para CRUD de docentes, gestionar_actividades para CRUD de actividades complementarias, configurar_ciclo para parámetros de ciclo académico, rectorado_dashboard mostrando panel de aprobación, rectorado_revisar para aprobación final, rectorado_configuracion para configuración institucional, y página 404 para manejo de errores.

Ilustración 26 Vista HTML 404

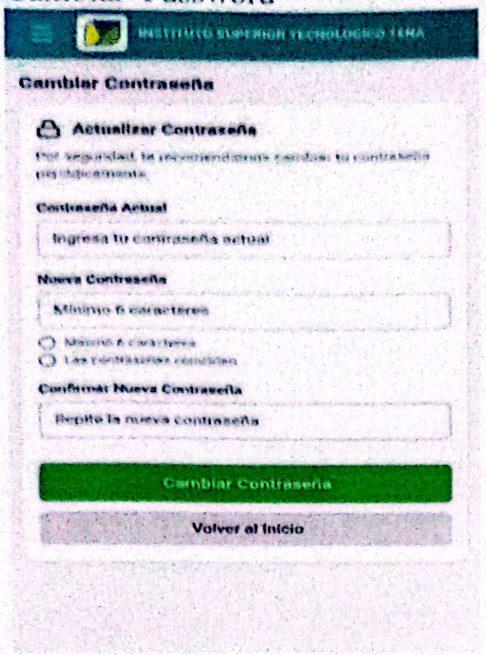
Nota. Interfaz para cuando la ruta solicitada no existe; Elaborado por Alan García (2025).

Ilustración 27 Vista HTML Login



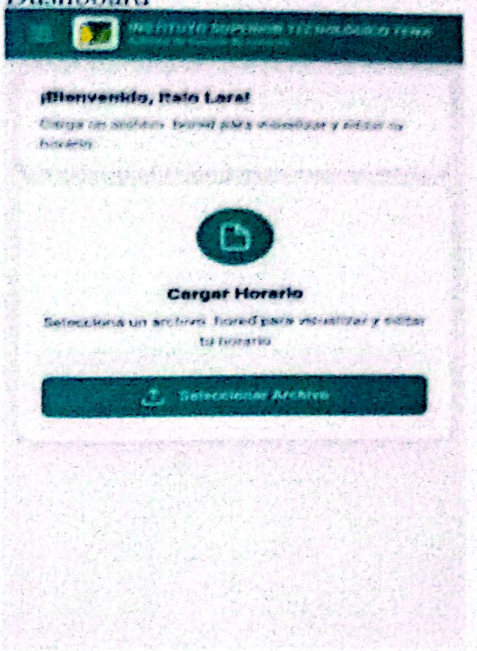
Nota. Interfaz para autenticación de usuarios; Elaborado por Alan García (2025).

Ilustración 29 Vista HTML Cambiar Password



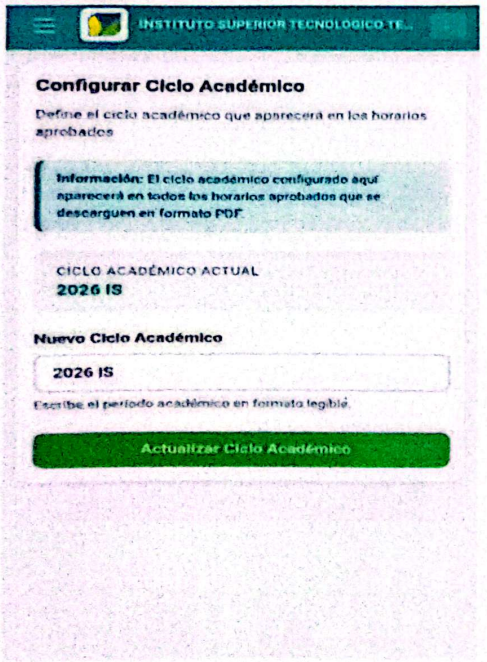
Nota. Formulario para actualizar la clave de acceso Del Docente; Elaborado por Alan García (2025).

Ilustración 28 Vista HTML Dashboard



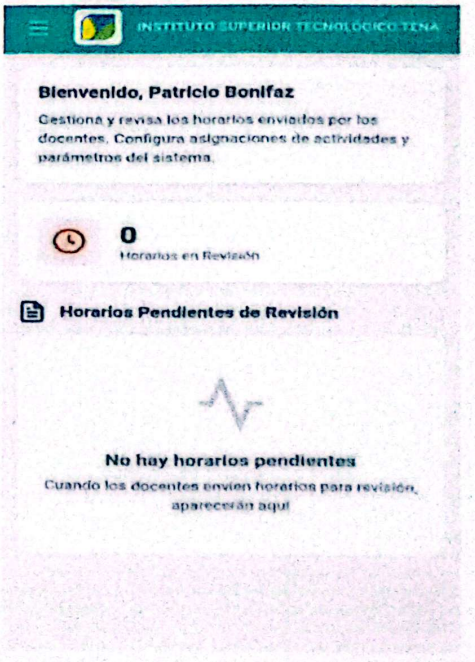
Nota. Interfaz para cargar y editar el horario Docente Elaborado por Alan García (2025).

Ilustración 31 Vista Html Coordinador Dashboard



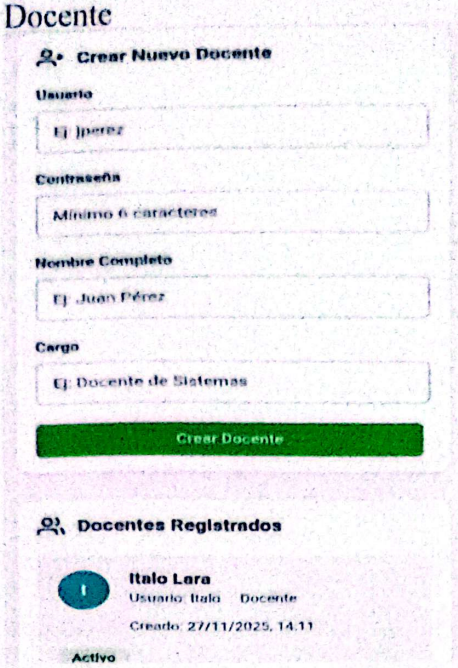
Nota. Interfaz para actualizar el período institucional; Elaborado por Alan García (2025).

Ilustración 30 Vista Html Coordinador Dashboard



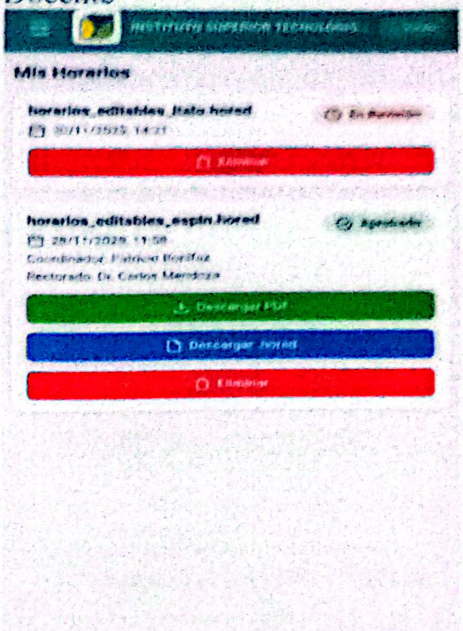
Nota. Interfaz del coordinador de carrera para revisar horarios; Elaborado por Alan García (2025).

Ilustración 32 Vista Html Crear Docente



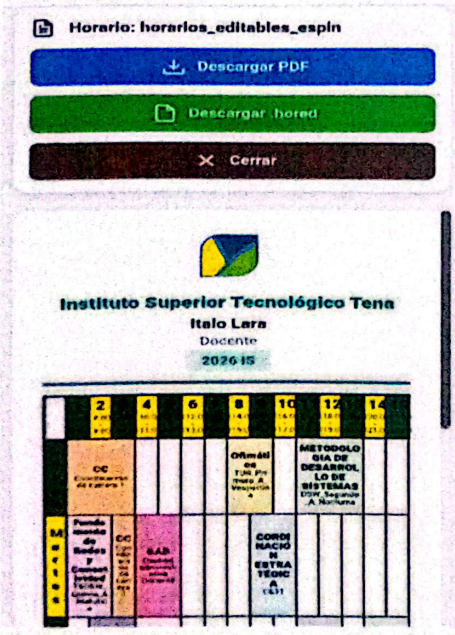
Nota. Formulario para registrar nuevos docentes en el sistema; Elaborado por Alan García (2025).

Ilustración 33 Vista Html Bandeja Docente



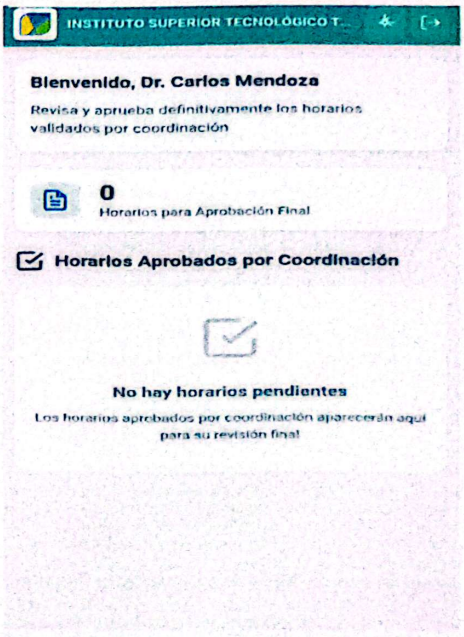
Nota. Interfaz para gestionar horarios enviados y aprobados; Elaborado por Alan García (2025).

Ilustración 35 Vista Html Descargar Pdf



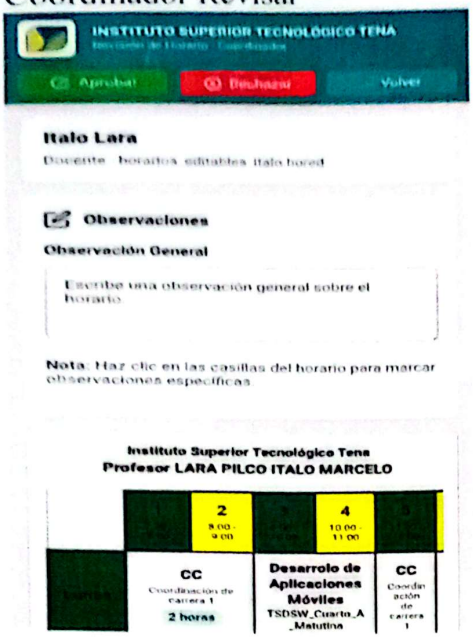
Nota. Interfaz para previsualizar y exportar el horario aprobado; Elaborado por Alan García (2025).

Ilustración 34 Vista HTML Descargar PDF



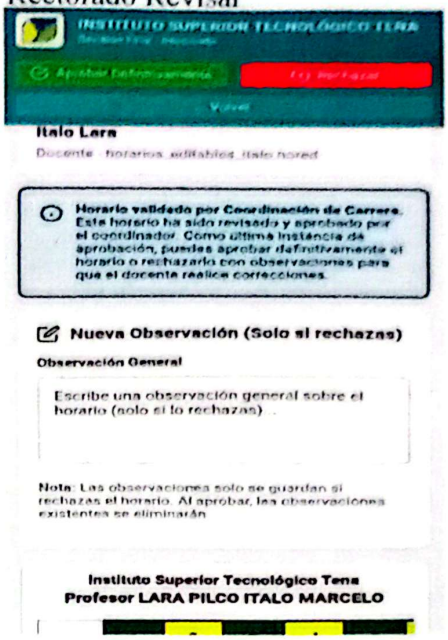
Nota. Panel para aprobar horarios validados por coordinación; Elaborado por Alan García (2025).

Ilustración 36 Vista Html Coordinador Revisar



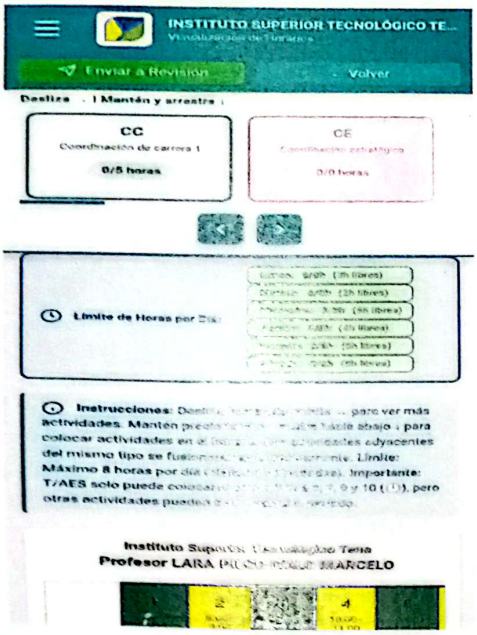
Nota. Interfaz para validar horarios y agregar observaciones; Elaborado por Alan García (2025).

Ilustración 37 Vista Html Rectorado Revisar



Nota. Interfaz para aprobar o rechazar el horario como última instancia; Elaborado por Alan García

Ilustración 38 Vista Html Visualizar

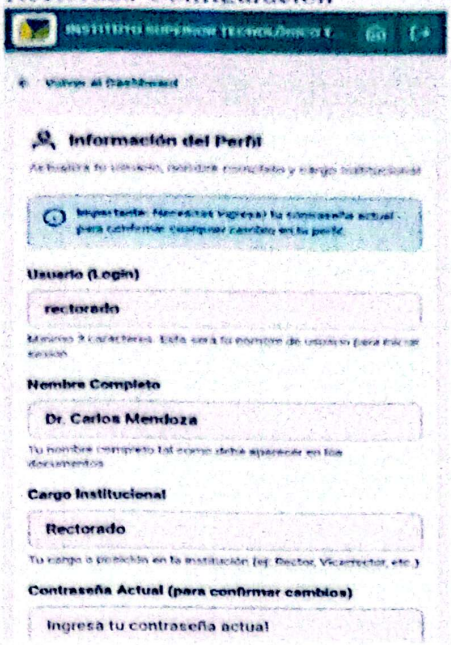


Nota. Interfaz interactiva para editar horarios con drag & drop; Elaborado por Alan García (2025).

Ilustración 39 Vista Html Gestionar Actividades

Nota. Panel para administrar actividades complementarias; Elaborado por Alan García (2025).

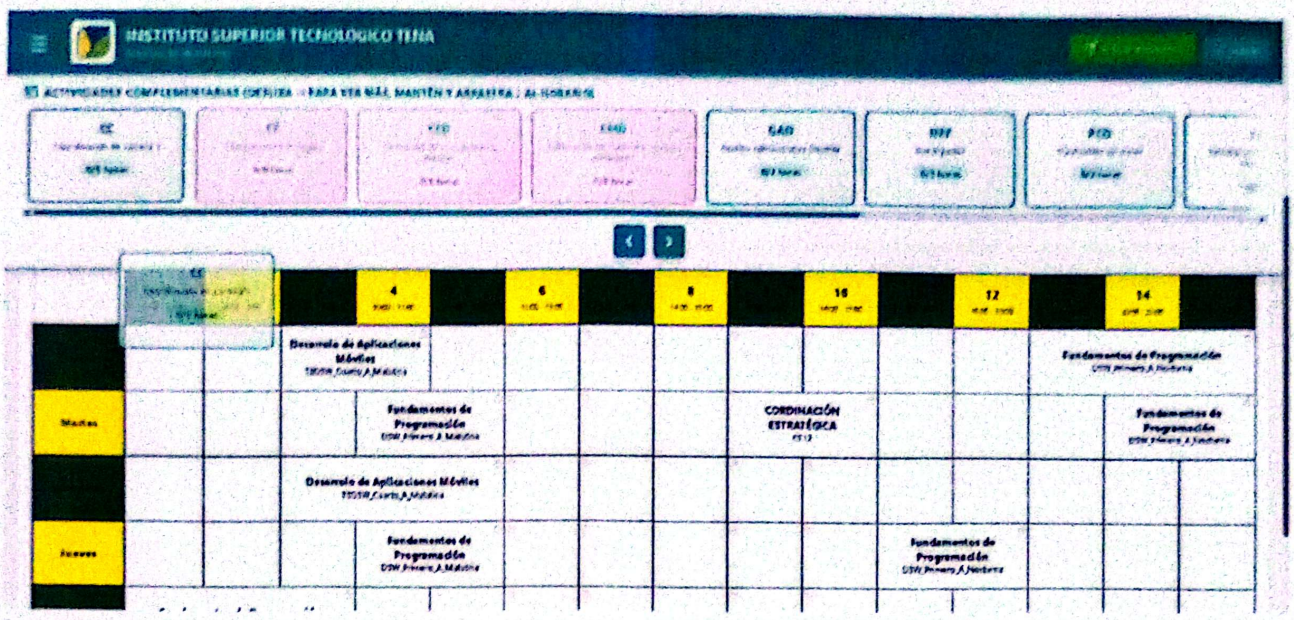
Ilustración 40 Vista Html
Rectorado Configuración



Nota. Formulario para actualizar perfil institucional;
Elaborado por Alan García (2025).

La implementación de drag & drop utilizó la API HTML5 nativa con eventos dragstart, dragover, drop y dragend, optimizada para dispositivos móviles mediante Touch Events (touchstart, touchmove, touchend) con delays de 200ms para diferenciar scroll de drag, vibración háptica mediante navigator.vibrate() cuando estuvo disponible para feedback táctil, y elemento fantasma que siguió el dedo del usuario durante el arrastre para compensar oclusión natural de la mano sobre pantalla.

Ilustración 41 Vista Horario Interactivo Escritorio



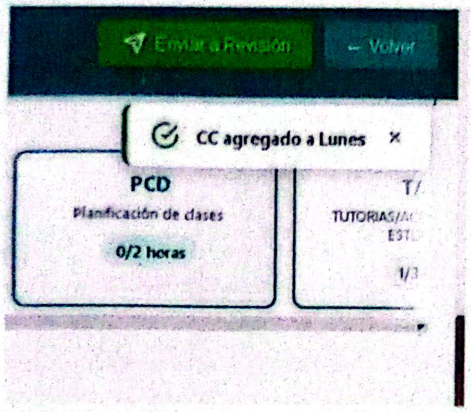
Nota. Vista Docente para Colocar Actividades y Visualizar el Horario Semanal; Elaborado por Alan García (2025).

Se implementó algoritmo JavaScript de fusión automática que detectó celdas adyacentes horizontalmente con código de actividad idéntico, combinó visualmente mediante modificación del atributo colspan incrementándolo por cada celda adicional fusionada, eliminó celdas redundantes del DOM, actualizó indicador visual mostrando rango de períodos como P3-P5, y mantuvo integridad de datos almacenando cada período individualmente en base de datos, aunque visualmente aparecieron fusionados.

Las validaciones en tiempo real incluyeron alerta visual cuando se aproximó al límite de 8 horas por día mostrando indicador de color dinámico, restricción de T/AES solo en períodos permitidos 6, 7, 9 y 10 bloqueando el drop en otros períodos, validación de límites individuales de actividades por docente consultando tabla asignacion_actividades, y prevención de colocación de actividades en celdas ya ocupadas verificando estado antes de permitir drop. Las APIs y librerías utilizadas incluyeron FileReader API para lectura de archivos .hored del lado del cliente con validación de estructura JSON, Fetch API para comunicación asíncrona con servidor mediante peticiones POST y GET, conversión UTC a zona horaria local (America/Guayaquil) mediante Date API nativo con biblioteca pytz en servidor, sistema de notificaciones mediante toasts con estilos CSS personalizados que aparecieron en esquina superior derecha con animaciones slide-in

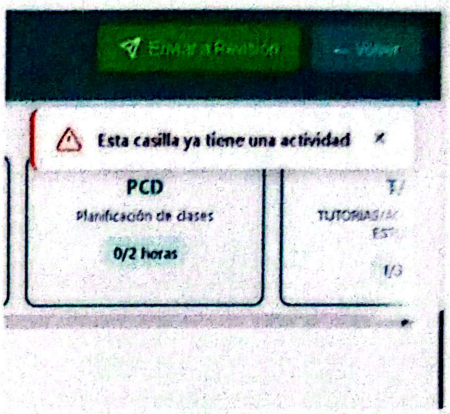
y fade-out, y modales de confirmación para acciones críticas implementados completamente con HTML/CSS/JS vanilla sin dependencias externas.

Ilustración 43 Validación Visual En Horarios Docentes



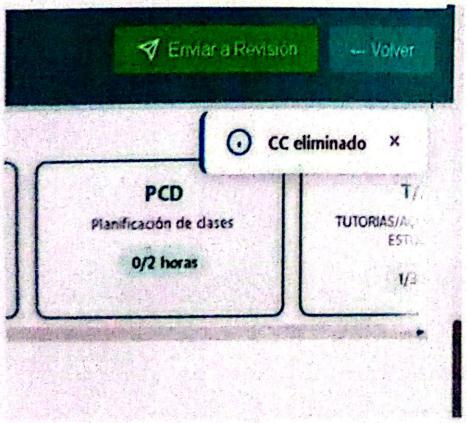
Nota. Confirmación visual al añadir una actividad; Elaborado por Alan García (2025).

Ilustración 42 Validación Visual En Horarios Docentes



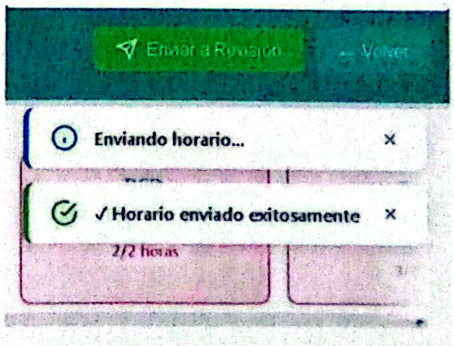
Nota. Mensaje de validación cuando una casilla ya tiene actividad; Elaborado por Alan García (2025).

Ilustración 45 Validaciones Visuales En Horarios Docentes



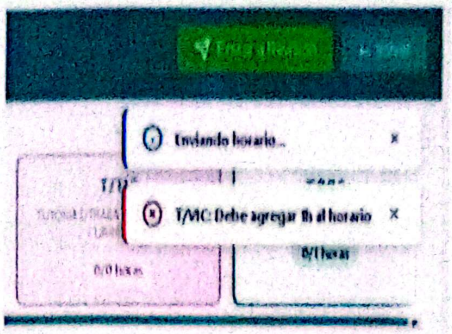
Nota. Notificación al borrar una actividad del horario; Elaborado por Alan García (2025).

Ilustración 44 Validaciones Visuales En Horarios Docentes



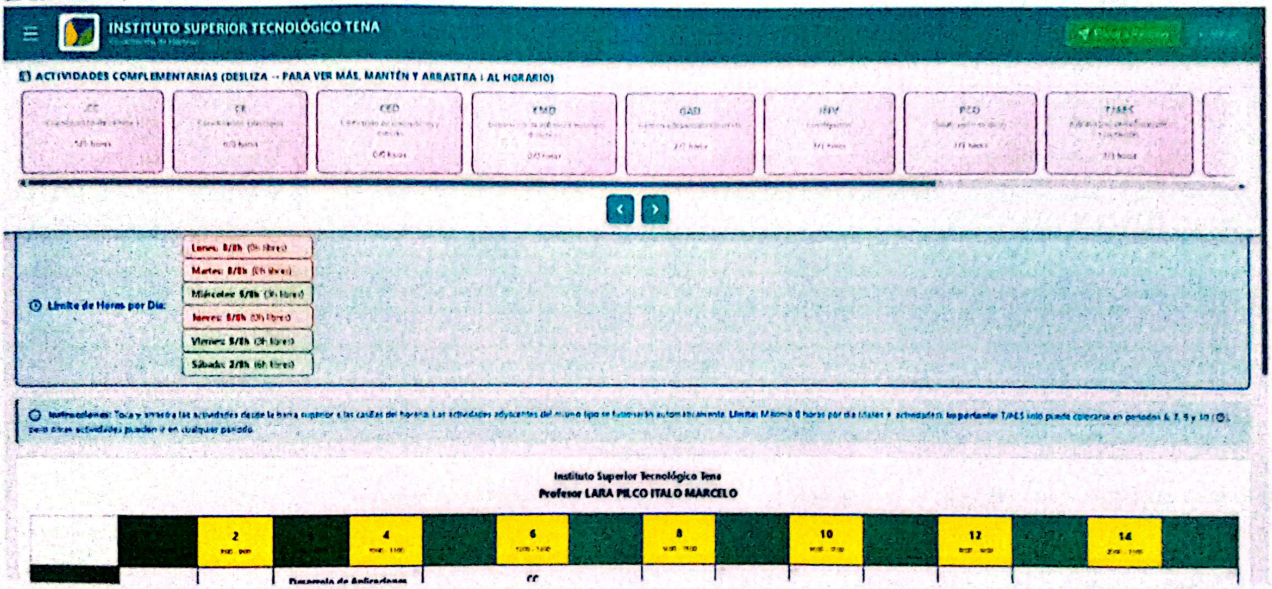
Nota. Mensaje de éxito tras enviar el horario; Elaborado por Alan García (2025).

Ilustración 46 Validaciones Visuales En Horarios Docentes



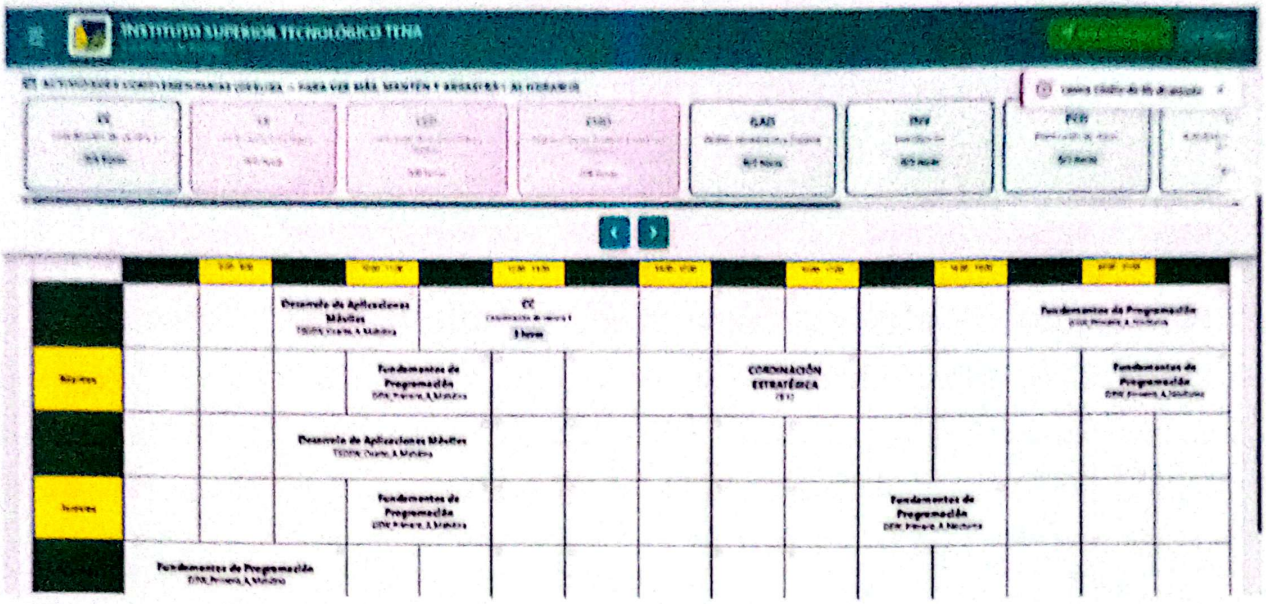
Nota. Advertencia cuando falta una actividad obligatoria al momento de enviar el horario a revisión; Elaborado por Alan García (2025).

Ilustración 47 Interfaz Horarios Docente Con Indicadores Visuales De Limites De 8 Horas Por Día



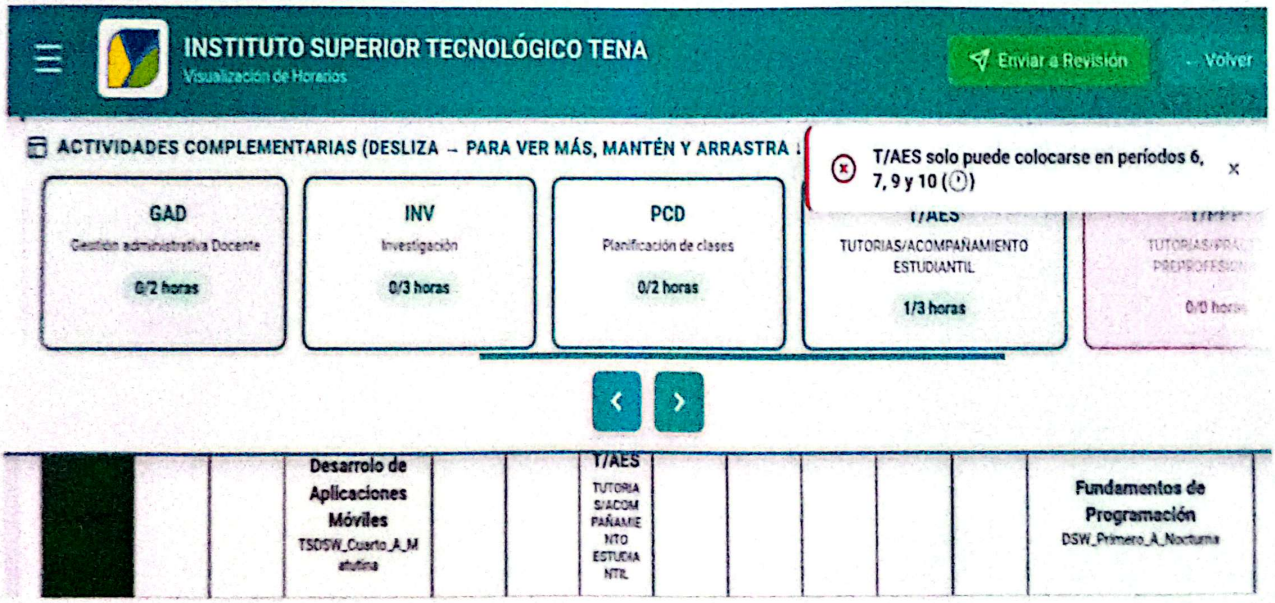
Nota. Vista que muestra cumplimiento de 8h por día en los indicadores visuales; Elaborado por Alan García (2025).

Ilustración 48 Vista Límite De 8 Horas De Horarios En Tiempo Real



Nota. Interfaz con alertas cuando se supera el límite diario permitido para los Docentes; Elaborado por Alan García (2025).

Ilustración 49 Restricción De Horarios En La Actividad Complementaria (T/Aes)



Nota. Interfaz que valida los períodos permitidos para la actividad tutorías de acompañamiento estudiantil (T/AES) : Elaborado por Alan García (2025).

7.3. Desarrollo del Aplicativo Web

El desarrollo del aplicativo web se ejecutó mediante metodología Kanban que permitió gestionar el flujo de trabajo de manera visual y flexible. El trabajo se organizó mediante un tablero con tres columnas representando el estado de cada funcionalidad: pendiente, en proceso y completado, limitando el trabajo en progreso a máximo tres tareas simultáneas para mantener foco y garantizar desarrollo completo de cada componente antes de iniciar el siguiente. El desarrollo se estructuró en cinco iteraciones semanales, cada una con objetivos específicos y entregables funcionales que fueron validados antes de proceder a la siguiente fase. A continuación, se presenta el detalle de las actividades realizadas en cada iteración, organizadas mediante tablas que muestran las tareas pendientes, en progreso y completadas para cada semana de desarrollo.

Semana 1: Del 20 al 23 de octubre de 2025

Tabla 5 Kamba Semana 1

Pendientes	En Progreso	Completadas
• **Implementar sistema de autenticación multi-rol** • **Crear base de datos con SQLite** • **Diseñar interfaz de login** • **Establecer roles (docente, coordinador, rectorado)**	• Configuración inicial del proyecto Flask • Estructura MVC • Instalación de Python y Flask	• Análisis de requerimientos completo • Diseño de arquitectura del sistema • Definición de flujo de trabajo (borrador → revisión coordinador → revisión rectorada → aprobado)

Elaborado por: Alan García (2025)

Semana 2: Del 27 al 31 de octubre de 2025

Tabla 6 Kamba Semana 2

Pendientes	En Progreso	Completadas
• **Crear interfaz para carga de archivos .hored** • **Implementar validación de límite de 8 horas/día** • **Diseñar dashboard docente**	• Sistema de login con hash SHA-256 • Gestión de sesiones con Flask session • Creación de tablas en base de datos usando sqlite3 • CRUD de usuarios por coordinador	• Sistema de autenticación funcional • Protección de rutas por rol • Base de datos relacional estructurada • Decoradores @login_required y @role_required implementados

Elaborado por: Alan García (2025)

Semana 3: Del 05 al 11 de noviembre de 2025

Tabla 7 Kamba Semana 3

Pendientes	En Progreso	Completadas
• **Implementar fusión automática de actividades consecutivas** • **Sistema de observaciones específicas por celda** • **Generación de documentos con formato institucional**	• Parser JSON para archivos .hored • Interfaz de visualización de horarios con drag & drop • Validaciones de negocio (8h/día, T/AES en períodos 6,7,9,10) • Barra flotante de actividades complementarias • Gestión de actividades complementarias (CRUD)	• Carga de archivos .hored funcional • Límite de 8 horas por día implementado • Dashboard docente con upload • Visualización interactiva de horarios • Estados del horario (borrador, revisión_coordinador, revisión_rectorado, aprobado, rechazado)

Elaborado por: Alan García (2025)

Semana 4: Del 17 al 21 de noviembre de 2025

Tabla 8 Kamba Semana 4

Pendientes	En Progreso	Completadas
• Optimización de rendimiento • Pruebas de usabilidad con usuarios reales • Documentación técnica	• Sistema de observaciones (generales y específicas) • Flujo completo de aprobación/rechazo • Dashboard coordinador con funcionalidades de revisión • Dashboard rectorado • Sistema de logging en 6 categorías • Generación de documentos oficiales	• Fusión automática de actividades adyacentes • Barra lateral flotante responsiva • Drag & drop optimizado para móvil (touch events) • Asignación de límites de actividades por docente • CRUD completo de docentes • Configuración de ciclo académico

Elaborado por: Alan García (2025)

Semana 5: Del 24 al 28 de noviembre de 2025

Tabla 9 Kamba Semana 5

Pendientes	En Progreso	Completadas
• **Manual Técnico** • **Manual de usuario**	• Pruebas de compatibilidad (Chrome, Firefox) • Pruebas de seguridad (validaciones) • Ajustes finales de interfaz según feedback	• Sistema de observaciones completo • Descarga de documentos oficiales • Descarga de .hored para horarios aprobados • Estado "borrador" para recuperación de progreso • Cambio de contraseña para docentes • Perfil editable para coordinador y rectorado • Sistema de logging completo para auditorías • Flujo de aprobación completo funcional • Validaciones en tiempo real • Responsividad completa • Manejo de concurrencia con RLock • Conversión UTC a zona horaria local • Sistema de notificaciones (toasts) • Pruebas completadas

Elaborado por: Alan García (2025)

Semana 6: Del 01 al 05 de diciembre de 2025 (Finalización del Proyecto)

Tabla 10 Kamba Semana 6

Pendientes	En Progreso	Completadas
-	-	• Manual técnico completo del sistema • Manual de usuario detallado con capturas de pantalla • Pruebas de compatibilidad finalizadas (Chrome, Firefox, Edge, Safari) • Pruebas de seguridad y validaciones completadas exitosamente • Ajustes finales de interfaz implementados según feedback de usuarios • Capacitación al personal administrativo del instituto completada • Sistema desplegado y funcionando en producción • Validación exitosa de todos los requerimientos funcionales y no funcionales • Proyecto finalizado exitosamente

Elaborado por: Alan García (2025)

7.3.1. Arquitectura e Implementación Técnica

Se implementó una arquitectura Modelo-Vista-Controlador (MVC) que proporcionó separación clara de responsabilidades facilitando el mantenimiento futuro del sistema. En el backend se utilizó Python 3.10.1 como lenguaje principal seleccionado por su sintaxis clara y excelentes librerías para desarrollo web. Flask 3.0.0 fue elegido como framework web por su naturaleza minimalista que permitió construir lógica de negocio específica sin imposiciones arquitectónicas innecesarias, en contraste con alternativas como Django que incluyeron componentes adicionales no requeridos para este proyecto.

La estructura del proyecto se implementó siguiendo el patrón MVC de manera integrada en un único archivo Python principal (app.py) que contiene: la lógica de acceso a datos mediante funciones que utilizan el módulo sqlite3 nativo con context managers para conexiones (`@contextmanager` con `get_db_connection`), las plantillas HTML externas en la carpeta templates utilizando motor Jinja2 con `render_template` para las vistas (login.html, dashboard.html, visualizar.html, bandeja_docente.html, coordinador_dashboard.html, rectorado_dashboard.html, etc.), y las rutas con decoradores Flask (`@app.route`) para los controladores. La carpeta static contiene recursos como el favicon e icono institucional, la carpeta uploads almacena los archivos .hored de horarios, la carpeta logs contiene los archivos de auditoría categorizados (sistema.log, errores.log, autenticacion.log, horarios.log, administracion.log, concurrencia.log), y el archivo de base de datos SQLite se ubica en la carpeta static como horarios.db. Esta arquitectura monolítica pero bien organizada con separación lógica de responsabilidades facilita el despliegue y mantenimiento del sistema. Para la autenticación se implementó un sistema basado en sesiones de Flask con decoradores personalizados @login_required que verificó existencia de sesión activa y @role_required que además validó nivel de acceso según rol, utilizando hashlib SHA-256 para hasheo irreversible de contraseñas antes de almacenarlas en base de datos.

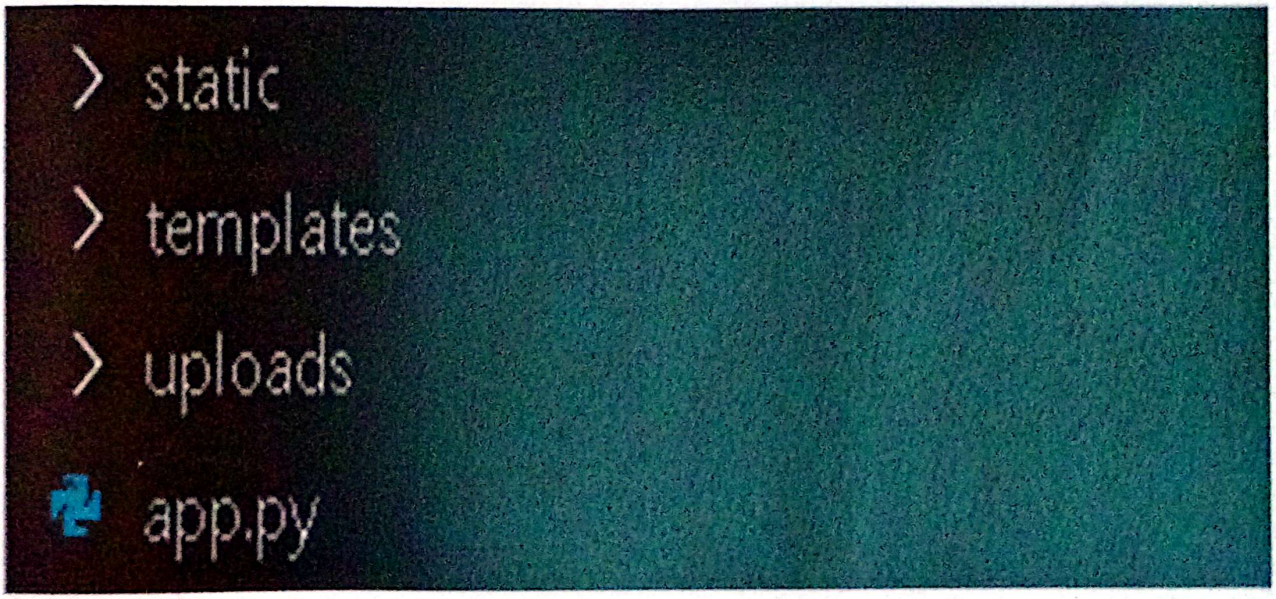
Tabla comparativa

Tabla 11 Comparación Entre Flask y Django

Aspecto	Flask 3.0.0	Django
Estructura	Simple y ligera	Completa y más compleja
Funcionalidades incluidas	Solo lo esencial: rutas, plantillas y manejo de solicitudes	Amplio conjunto de herramientas y módulos integrados
Flexibilidad	Alta: permite construir la lógica personalizada según necesidades	Menor: sigue una estructura más rígida y predefinida
Código inicial	Requiere escribir más código al inicio	Menos código inicial gracias a sus componentes integrados
Control sobre componentes	Total control sobre cada parte del sistema	Control limitado por la arquitectura del framework

Elaborado por: Alan García (2025)

Ilustración 50 Árbol Del Proyecto



Nota. Árbol del proyecto en base a la arquitectura MVC; Elaborado por Alan García (2025).

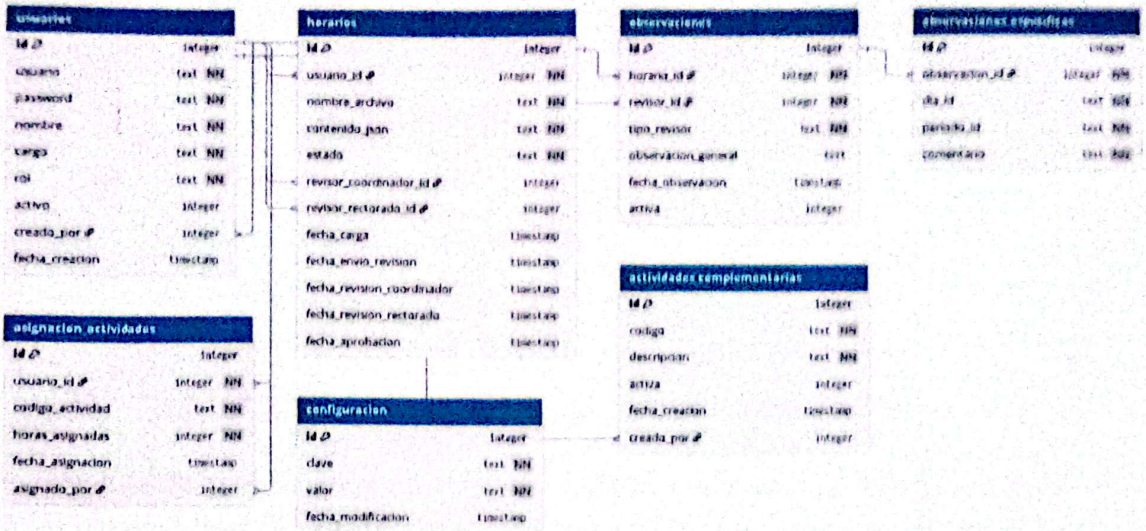
7.3.2. Base de Datos y Gestión de Información

Para la base de datos se implementó SQLite 3 mediante el módulo sqlite3 nativo de Python, seleccionado por tratarse de una base embebida que elimina la necesidad de configurar servidor externo, reduciendo requisitos de infraestructura y simplificando procedimientos de respaldo que se limitan a copiar un único archivo. Se habilitó modo WAL (Write-Ahead Logging) mediante instrucción PRAGMA journal_mode=WAL para mejorar concurrencia permitiendo lecturas simultáneas sin bloquear escrituras. Se utilizó Row Factory configurando connection.row_factory = sqlite3.Row para trabajar con diccionarios en lugar de tuplas, facilitando acceso a datos por nombre de columna en lugar de índice numérico.

La estructura de base de datos comprende ocho tablas principales: usuarios con campos para id autoincrementado como identificador único, usuario (nombre de usuario), password (contraseña hasheada), nombre completo, cargo, rol (docente/coordinador/rectorado), estado activo, creado_por como clave foránea, y fecha de creación; horarios con id autoincrementado, usuario_id como clave foránea al docente, nombre_archivo, contenido_json almacenando la estructura completa del horario, estado del flujo (borrador/revisión_coordinador/revisión_rectorado/rechazado_coordinador/rechazado_rectorado/aprobado), revisor_coordinador_id, revisor_rectorado_id, y múltiples timestamps para cada etapa del flujo; observaciones con id, horario_id, revisor_id, tipo_revisor, observacion_general, fecha_observacion y estado activa; observaciones_especificas para comentarios por celda con observacion_id, dia_id, periodo_id y comentario; asignacion_actividades relacionando usuarios con actividades mediante claves foráneas incluyendo código_actividad, horas_asignadas y asignado_por; configuracion para parámetros del sistema con clave, valor, descripción, creado_por, modificado_por y fechas; configuracion_historial para auditoría de cambios de configuración; y actividades_complementarias con código único y descripción. El sistema de logging se implementa mediante archivos de texto rotativos organizados en seis categorías (sistema.log, errores.log, autenticacion.log, horarios.log, administracion.log, concurrencia.log) utilizando RotatingFileHandler de Python, no mediante tabla de base de datos.

Se implementaron índices optimizados mediante CREATE INDEX en campos de búsqueda frecuente como estado de horarios, cédula de usuarios, y id de docente en asignaciones para garantizar tiempos de respuesta inferiores a 2 segundos en consultas complejas.

Ilustración 51 Diagrama Entidad-Relación de la Base de Datos

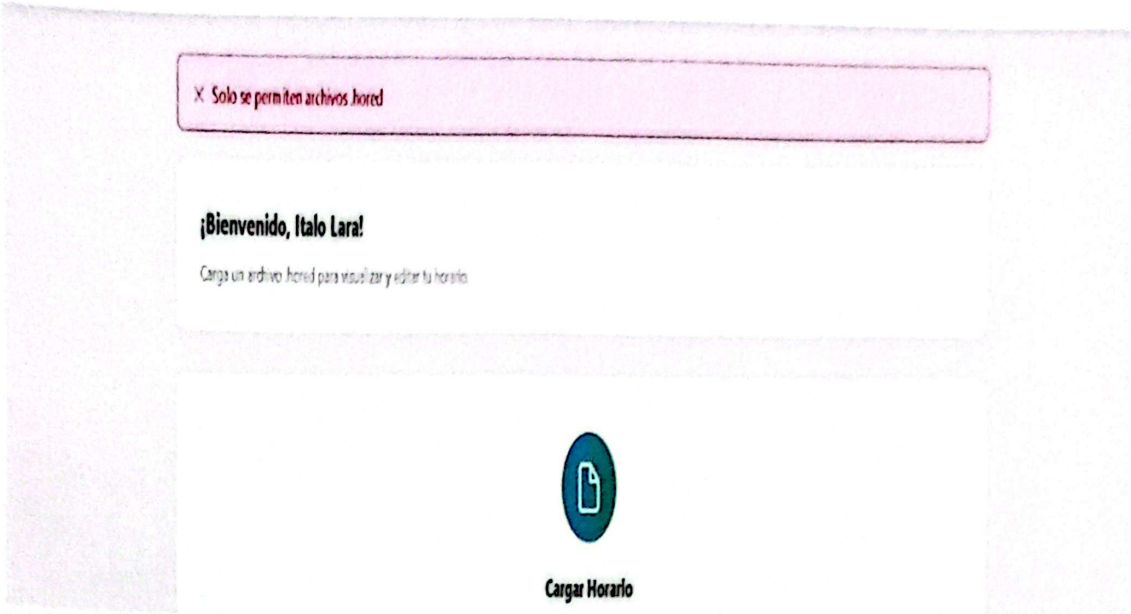


Nota. Representa las tablas principales y sus relaciones para la gestión automatizada de horarios docentes; Elaborado por Alan García (2025).

7.3.3. Funcionalidades Clave Implementadas

El procesamiento de archivos .hored se realizó mediante parser JSON con función json.loads() de Python que convirtió texto JSON a diccionarios y listas, implementando validación automática de estructura que verificó presencia de campos obligatorios (docente, periodo, horario, actividades_complementarias) y tipos de datos esperados, rechazando archivos que no cumplieron esquema definido mediante bloques try-except que capturaron excepciones JSONDecodeError. El manejo robusto de errores registró en sistema de logs cualquier intento de carga de archivo corrupto almacenando contenido problemático para análisis posterior, y notificó al usuario mediante mensaje amigable evitando terminología técnica confusa.

Ilustración 52 Carga de Archivo .hored con Validación



Nota. Interfaz de carga que valida la estructura del archivo .hored y notifica errores de formato al docente; Elaborado por Alan García (2025).

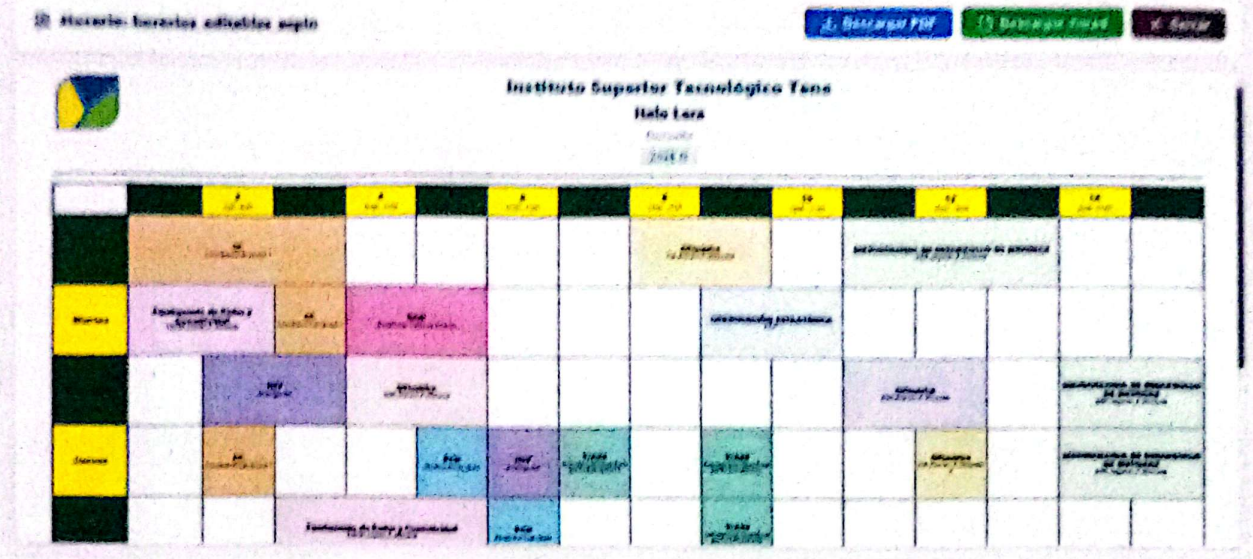
El flujo de aprobación multi-etapa administró seis estados diferenciados: borrador como estado inicial permitiendo edición sin restricciones y guardado automático cada 30 segundos, revisión_coordinador cuando el docente envió bloqueando edición posterior y notificando al coordinador, rechazado_coordinador cuando el coordinador devolvió el horario al docente con observaciones para corrección, revisión_rectorado cuando coordinador aprobó manteniendo bloqueo de edición, rechazado_rectorado cuando rectorado devolvió el horario al docente con observaciones para corrección, y aprobado como estado final habilitando descarga de PDF oficial y archivo .hored. Las transiciones se implementaron mediante consultas SQL UPDATE que modificaron campo estado, agregaron timestamp de transición en campos específicos, registraron en archivos de log la acción con usuario responsable, y dispararon notificaciones toast informando cambio de estado exitoso.

Ilustración 53 Flujo De Revisión Y Aprobación De Horarios

Nota. Diagrama del proceso de revisión y validación de horarios: Elaborado por Alan García (2025).

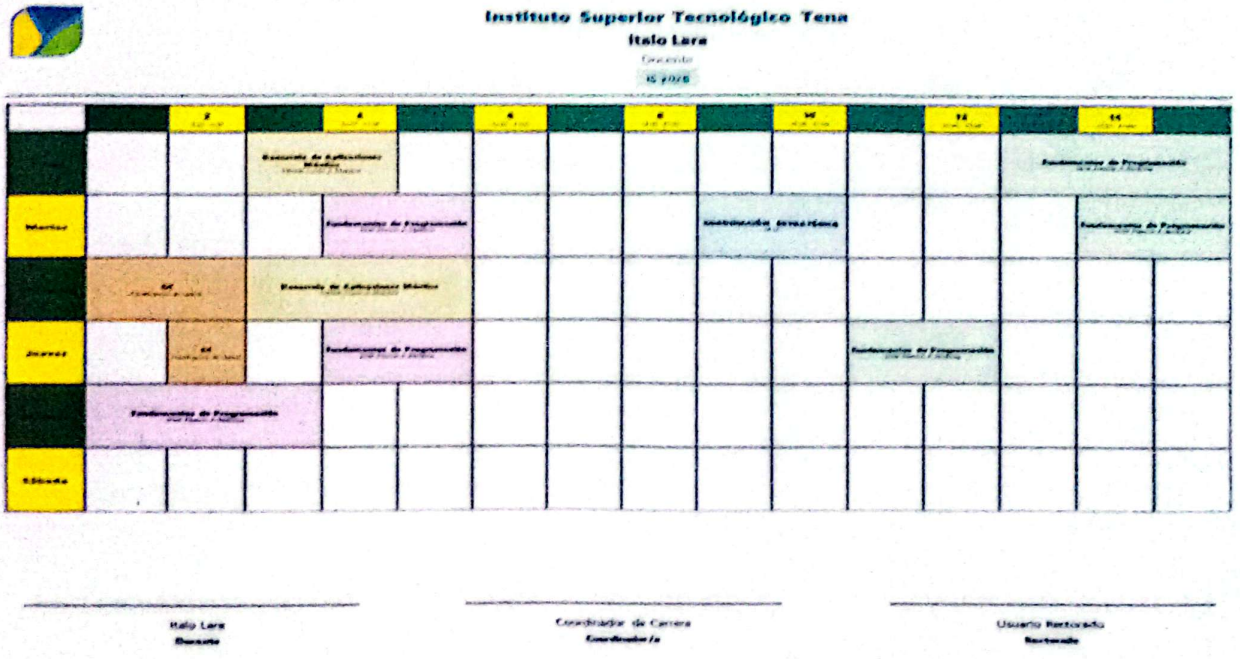
La generación de documentos PDF oficiales utiliza bibliotecas del lado del cliente para evitar procesamiento en servidor: jsPDF versión 2.5.1 para creación de archivo PDF y html2canvas versión 1.4.1 para convertir DOM HTML del horario a imagen rasterizada. El proceso inicia cuando usuario con horario aprobado presiona botón Descargar PDF, ejecutando función JavaScript que selecciona elemento DOM de tabla de horarios, aplica html2canvas para convertir a imagen con resolución optimizada de 2 para claridad en impresión, crea documento PDF con jsPDF configurando orientación paisajista landscape y tamaño A4, agrega encabezado institucional insertando logotipo del ISTT embebido como base64, agrega información del docente (nombre, cargo, período), inserta imagen del horario escalándola apropiadamente, agrega sección de firmas con tres campos predefinidos (docente, coordinador, rectorado) con líneas horizontales y etiquetas, y dispara descarga con nombre formateado.

Ilustración 54 Vista Previa Y Descarga Del Pdf



Nota. Interfaz del Horario final con opción de exportar en PDF o .hored; Elaborado por Alan García (2025).

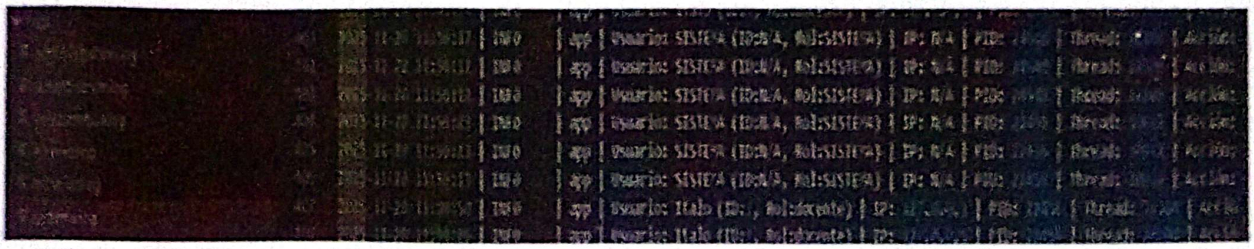
Ilustración 55 Visualización Del Exportado PDF: Horario Final



Nota. Visualización del horario final aprobado y descargado por parte del docente en PDF; Elaborado por Alan García (2025).

El sistema de logging completo implementó seis categorías diferenciadas (administración, autenticación, concurrencia, errores, horarios, sistema): autenticación registrando login exitoso/fallido, logout y cambios de contraseña; edición registrando creación, modificaciones y eliminaciones de horarios; aprobación registrando acciones de coordinador y rectorado con timestamps; configuración registrando cambios en parámetros institucionales; error registrando excepciones capturadas con información detallada para debugging; y sistema registrando eventos de inicialización y operaciones o auditorías.

Ilustración 56 Sistema de Logging y Auditoría por Categorías



Nota. Visualiza el registro de acciones del sistema, clasificado por tipo de evento y usuario; Elaborado por Alan García (2025).

7.3.4. Tecnologías Utilizadas

Tabla 12 Tecnologías Utilizadas

Componente	Tecnología	Versión
Lenguaje Backend	Python	3.10.1
Framework Web	Flask	3.0.0
Base de Datos	SQLite	3
Acceso a BD	sqlite3 (módulo nativo)	-
Frontend	HTML5, CSS3, JavaScript vanilla	-

Componente	Tecnología	Versión
Hasheo de contraseñas	hashlib (SHA-256)	Nativo Python
Gestión de concurrencia	threading.RLock	Nativo Python
Logging	RotatingFileHandler	Nativo Python
Modo de BD	WAL (Write-Ahead Logging)	SQLite nativo
Context Manager BD	@contextmanager personalizado	Nativo Python
Control de escritura BD	threading.RLock	Nativo Python
Retry automático	execute_with_retry	Implementación personalizada
Generación PDF	jsPDF, html2canvas	2.5.1, 1.4.1

Elaborado por: Alan García (2025)

8. CONCLUSIONES

Mediante entrevistas semiestructuradas con el coordinador, observación directa del proceso manual y revisión de horarios anteriores, identifiqué las necesidades del sistema. Documenté 33 requerimientos funcionales en 10 categorías: gestión de archivos, usuarios, edición de horarios, validaciones, flujo de aprobación, sistema de observaciones, generación de documentos, gestión administrativa, auditoría y dashboard. Establecí criterios no funcionales de seguridad (contraseñas hasheadas con SHA-256, cookies HttpOnly y SameSite), usabilidad (interfaz sin necesidad de entrenamiento), rendimiento (respuesta menor a 2 segundos) y mantenibilidad (arquitectura MVC documentada). La observación directa reveló restricciones no mencionadas en las entrevistas, como la fusión automática de actividades consecutivas y el estado "borrador" para recuperar trabajo en progreso.

Desarrollé la interfaz mediante pruebas iterativas con usuarios. El coordinador y los docentes validaron su facilidad de uso y confirmaron que no requiere entrenamiento. El diseño responsivo funciona correctamente en todos los dispositivos. El sistema drag & drop resultó intuitivo incluso para usuarios sin experiencia técnica. Los indicadores visuales de horas, límites y estados de aprobación informan claramente sobre las restricciones. Cada rol (docente, coordinador, rectorado) tiene su interfaz personalizada con acceso únicamente a sus funciones.

El sistema fue desarrollado con tecnologías de código abierto (Python/Flask, SQLite, CSS3 y JavaScript vanilla) que facilitan el mantenimiento futuro. Implementé medidas de seguridad estándar: hashlib SHA-256 para contraseñas, validación dual cliente-servidor, tokens de sesión Flask y expiración a 12 horas. Las pruebas de rendimiento confirmaron tiempos adecuados: carga < 1s, procesamiento < 2s, generación de PDF < 3s. El uso de SQL nativo (sin ORM) y la arquitectura MVC clara permiten modificaciones futuras sin comprometer la integridad de los datos.

9. RECOMENDACIONES

Se recomienda establecer un proceso de revisión semestral de los requerimientos funcionales del sistema mediante entrevistas con usuarios, análisis de métricas de uso del sistema de logging, y evaluación de solicitudes de mejora. Adicionalmente, realizar un estudio de viabilidad para expandir el sistema desde la fase piloto en Desarrollo de Software hacia las demás carreras del Instituto Superior Tecnológico Tena, adaptando las validaciones de negocio y actividades complementarias según las particularidades de cada programa académico. Esta expansión gradual permitirá validar la escalabilidad del sistema antes de su implementación institucional completa.

Desarrollar un programa de capacitación estructurado que incluya: video tutoriales de 2-3 minutos para cada funcionalidad principal, manual de usuario digital con capturas de pantalla y casos de uso comunes, y sesiones presenciales de inducción al inicio de cada período académico. Implementar un mecanismo de retroalimentación continua mediante un formulario dentro del sistema para reportar dificultades de usabilidad o sugerencias de mejora, analizando esta información trimestralmente para priorizar ajustes. Realizar pruebas de accesibilidad siguiendo estándares WCAG 2.1 nivel AA para garantizar que usuarios con discapacidades visuales o motrices puedan utilizar el sistema efectivamente.

Establecer un plan de mantenimiento preventivo y evolutivo que contemple: actualización trimestral de dependencias de Python y Flask para corregir vulnerabilidades de seguridad, evaluación de migración del algoritmo de hasheo de contraseñas de SHA-256 a bcrypt o Argon2 para mayor resistencia a ataques de fuerza bruta, implementación de respaldos automáticos diarios de la base de datos con almacenamiento externo, y monitoreo proactivo del sistema de logging para detectar patrones anómalos de acceso. En el mediano plazo, evaluar la migración de SQLite a PostgreSQL si el número de usuarios concurrentes supera los 20 previstos inicialmente, garantizando el rendimiento óptimo conforme crezca la adopción institucional. Documentar la arquitectura del sistema mediante diagramas UML actualizados y comentarios exhaustivos en el código.

10. BIBLIOGRAFÍA

Referencias

Ahmad, M. O. (2013). Kanban in software development: A systematic literature review. (IEEE, Ed.) *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*, 9-16. doi:10.1109/SEAA.2013.28

Albesher, A. (Diciembre de 2024). An Observational Study on Flask Web Framework Questions on Stack Overflow (SO). *IET Software*. doi:10.1049/sfw2/1905538

Anderson, D. J. (2010). *Kanban: Successful evolutionary change for your technology business.* (B. H. Press, Ed.)

Asamblea Nacional del Ecuador. (2010). Ley Orgánica de Educación Superior. *Registro Oficial Suplemento 298, 298.*

Asamblea Nacional del Ecuador. (2014). Código Orgánico Integral Penal. *Registro Oficial Suplemento 180, 180.*

Babaei, H. K. (2015). A survey of approaches for university course timetabling problem. *Computers & Industrial Engineering, 86*, 43-59. doi:10.1016/j.cie.2014.11.010

Carmo, K. X., Ferreira, F. J., & Figueiredo, E. (2024). Performance Evaluation of Back-End Frameworks: A Comparative Study. *Proceedings of the 20th Brazilian Symposium on Information Systems (SBSI '24)* (págs. 1-9). Juiz de Fora, Brasil: ACM. doi:10.1145/3658271.3658314

Congreso Nacional del Ecuador. (2002). Ley de Comercio Electrónico, Firmas Electrónicas y Mensajes de Datos. *Registro Oficial Suplemento 557, 557.*

Consejo de Educación Superior. (2019). Reglamento de Régimen Académico. *Resolución RPC-SO-08-No.111-2019.*

Elmasri, R. &. (2015). *Fundamentals of database systems* (7 ed.). (Pearson, Ed.)

Fielding, R. T., & Taylor, R. N. (2002). Principled design of the modern Web architecture. *ACM Transactions on Internet Technology, 2*(2), 115-150. doi:10.1145/514183.514185

Gamma, E. H. (1994). *Design patterns: Elements of reusable object-oriented software.* (Addison-Wesley, Ed.)

Grinberg, M. (2018). *Flask web development: Developing web applications with Python* (2 ed.). (O. Media, Ed.)

Leff, A. &. (2001). Web-application development using the model/view/controller design pattern. (IEEE, Ed.) *Proceedings Fifth IEEE International Enterprise Distributed Object Computing Conference*, 118-127. doi:10.1109/EDOC.2001.950428

Majeed, A., & Rauf, I. (2018). MVC Architecture: A Detailed Insight to the Modern Web Applications Development. *Peer Review Journal of Solar and Photoenergy Systems, 1*(1), PRSP.000505.

Marcotte, E. (2011). *Responsive web design.* (A. B. Apart, Ed.)

Merkel, D. (2014). Docker: Lightweight Linux containers for consistent development and deployment. *Linux Journal, 2014*(239), 2-9.

Nielsen, J. (1993). *Usability engineering.* (A. Press, Ed.)

Nikum, P., Padhal, I., Parge, S., Patange, G., Pande, O., & Ramtirthe, P. (2024). Exploration of the Future of Web Development and Emerging Technologies. *Proceedings of the 5th International Conference on Communication and Information Processing (ICCIP)-2023.* doi:10.2139/ssrn.4687203

Olana, B. K., & Tolasa, D. G. (2025). Adoption of web application framework to enhance web project result. *American Journal of Computer Science and Technology, 8*(2), 102-120. doi:10.11648/j.ajcst.20250802.15

OWASP Foundation. (2021). *OWASP top 10 - 2021: The ten most critical web application security risks.* Obtenido de OWASP: https://owasp.org/Top10/

OWASP Foundation. (2024). *OWASP guide - Validation of data on client and server side*. Obtenido de OWASP: https://owasp.org/

Owens, M. (2006). *The definitive guide to SQLite*. (Apress, Ed.)

Pahl, C., & Jamshidi, P. (2016). Microservices: A Systematic Mapping Study. *Proceedings of the 6th International Conference on Cloud Computing and Services Science (CLOSER). 1*, págs. 137-146. Rome, Italy: SciTePress. doi:10.5220/0005785501370146

Pallets Projects. (2023). *Flask documentation: A Python web framework*. Obtenido de Flask Documentation: https://flask.palletsprojects.com/

Pressman, R. S. (2020). *Software engineering: A practitioner's approach* (9 ed.). (M.-H. Education, Ed.)

Raschka, S. &. (2019). *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2* (3 ed.). (P. Publishing, Ed.)

Sharma, A., Jain, A., Bahuguna, A., & Dinkar, D. (Diciembre de 2020). Comparison and Evaluation of Web Development Technologies in Node.js and Django. *International Journal of Science and Research (IJSR), 9*(12), 1416-1420. Obtenido de https://www.ijsr.net/getabstract.php?paperid=SR201202223534

Silva, Y. N. (2016). SQL: From traditional databases to big data. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, 495-500. Nueva York, Estados Unidos: ACM. doi:10.1145/2839509.2844560

Sommerville, I. &. (1997). *Requirements engineering: A good practice guide*. (J. W. Sons, Ed.)

Stallings, W. &. (2018). *Computer security: Principles and practice* (4 ed.). (Pearson, Ed.)

Taibi, D., & Lenarduzzi, V. (2018). On the Definition of Microservice Bad Smells. *IEEE Software*(3), 56-62. doi:10.1109/MS.2018.2141031

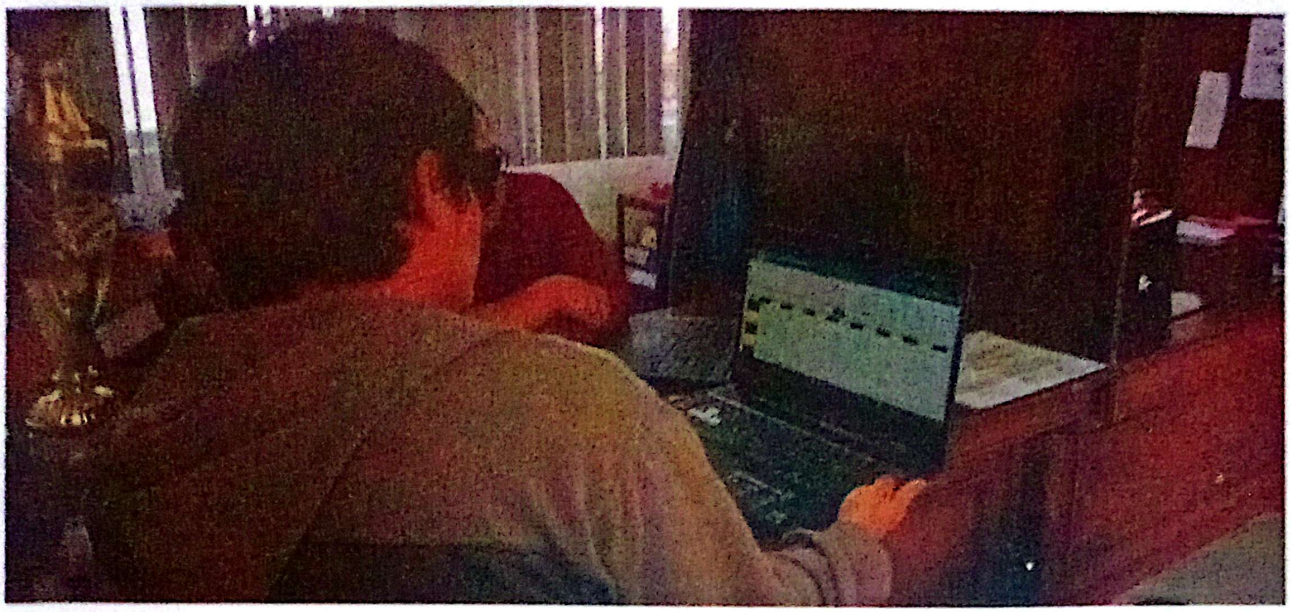
Tilkov, S., & Vinoski, S. (2010). Node.js: Using JavaScript to Build High-Performance Network Programs. *IEEE Internet Computing, 14*, 80-83. doi:10.1109/MIC.2010.145

Tullis, T. &. (2013). *Measuring the user experience: Collecting, analyzing, and presenting usability metrics* (2 ed.). (M. Kaufmann, Ed.)

11. ANEXOS

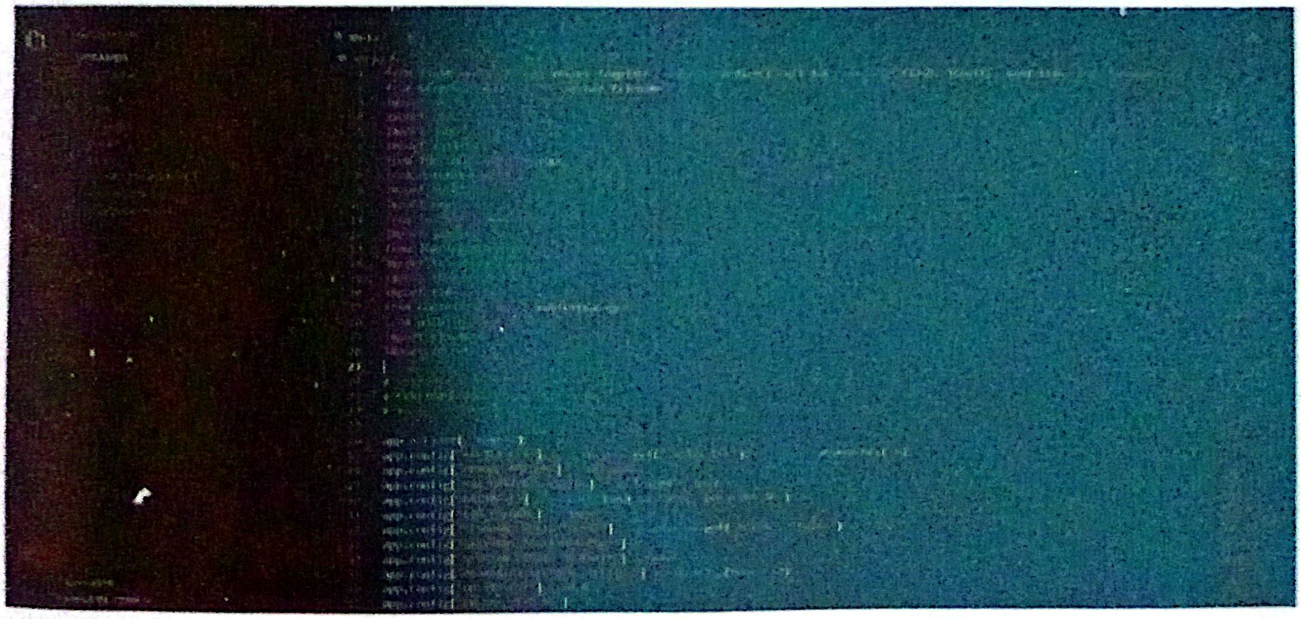
11.1. Imágenes

Ilustración 57 Demostración del Sistema y Recopilación de Cambios



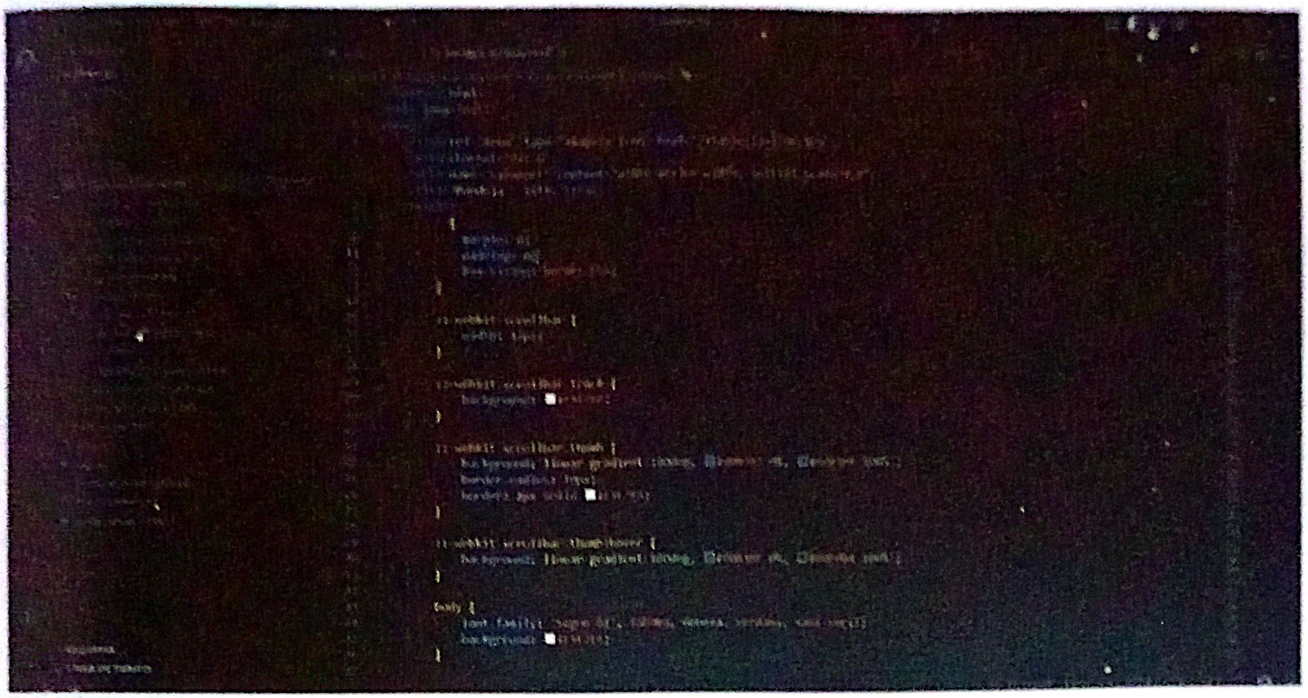
Nota. Presentación práctica del sistema y recopilación de observaciones con el docente; Elaborado por Alan García (2025).

Ilustración 58 Vista IDE (Visual Studio Code): Codificación del Back-End



Nota. Vista y codificación del Back-End en el editor de código (VS Code) del sistema web para automatizar horarios docentes en el IST Tena; Elaborado por Alan García (2025).

Ilustración 59 Vista IDE (Visual Studio Code): Codificación del Front-End



Nota. Vista y codificación del Front-End en el editor de código (VS Code) del sistema web para automatizar horarios docentes en el IST Tena; Elaborado por Alan García (2025).

Ilustración 60 Solicitud Espacio Servidor

OFICIO ISTT-DS-2025-661-O

Tena, 08 de diciembre de 2025

Msd. Lorena Yánez
Rectora Del Instituto Superior Tecnológico Tena
Presente,

Asunto: Solicitud de espacio en el servidor institucional para alojamiento web
De mis consideraciones:

Reciba un cordial saludo junto a los deseos de bienestar; yo, Alan Israel García Castro, estudiante de la Carrera de Tecnología en Desarrollo de Software del Instituto Superior Tecnológico Tena, autor del proyecto "Implementar una Aplicación Web que Automatice el Proceso de Elaboración del Horario Docente en el Instituto Superior Tecnológico Tena", desarrollado como parte del proceso de titulación en la carrera de Desarrollo de Software; solicito de la manera más comedida, autorice y delegue a quien corresponda, la asignación de un espacio dentro del servidor del IST Tena para alojar la aplicación antes mencionada, sistema con el que se busca automatizar el proceso de elaboración y gestión de horarios docentes.

Adicionalmente, solicito la posibilidad de contar con un subdominio institucional para el sistema de gestión de horarios docentes que permita el acceso adecuado a la aplicación.

Características Técnicas del Sistema

Tecnologías Empleadas

- **Backend:** Python 3.10.1, Flask 3.0.0
- **Frontend:** HTML5, CSS3, JavaScript
- **Base de datos:** SQLite 3 con modo WAL (Write-Ahead Logging)
- **Arquitectura:** Modelo-Vista-Controlador (MVC)
- **Sistema de logs:** Rotating logs con límites automáticos
- **Seguridad:** Sesiones seguras, hash SHA-256, tokens CSRF

Espacio Estimado Requerido

Se solicita una asignación aproximada de 20 GB para garantizar el correcto funcionamiento del sistema, el almacenamiento de archivos, base de datos y logs.

Requisitos del Servidor

- **Python 3.10 o superior**
- **Servidor web compatible con Flask (Gunicorn/Nginx)**
- **Soporte para SQLite 3**
- **Acceso HTTP/HTTPS**

Agradezco de antemano su atención a la presente solicitud, y quedo atento a cualquier requerimiento adicional o coordinación técnica necesaria para proceder con el alojamiento del sistema web.

Atentamente;

Ing. P. Guerrero.
Por favor revisar y ver
Ara si se cuenta con el
espacio y remitir un informe
si procediera

Alan Israel García Castro
Estudiante de DS
C.I. 1550197857

08-12-25 9:57

Nota. Solicitud de asignación de un espacio en el servidor del IST Tena para alojar la aplicación web; Elaborado por Alan García (2025).

Ilustración 61 Solicitud de Certificado de Aprobación de Implementación

Tena, 03 de febrero de 2026

Msd. Lorena Yánez
Rectora Del Instituto Superior Tecnológico Tena
Presente,

Asunto: Solicitud de Certificado de Aprobación – Aplicación Web Automatización Horario Docente

De mis consideraciones:

Reciba un cordial saludo junto a los deseos de bienestar; yo, Alan Israel García Castro, estudiante de la Carrera de Tecnología en Desarrollo de Software del Instituto Superior Tecnológico Tena, autor del proyecto **"Implementar una Aplicación Web que Automatice el Proceso de Elaboración del Horario Docente en el Instituto Superior Tecnológico Tena"**, desarrollado como parte del proceso de titulación en la carrera de Desarrollo de Software; solicito de manera formal la emisión y entrega de un certificado de aprobación que acredite la implementación exitosa de la aplicación web para automatización del horario docente en el IST Tena.

Como se recuerda, previamente gestioné la asignación de espacio en servidor institucional (Oficio ISTT-DS-2025-001-O) para alojar esta aplicación, la cual ha sido debidamente implementada y puesta en funcionamiento, cumpliendo con los objetivos planteados en el proyecto.

Agradezco de antemano su atención a la presente solicitud, y quedo atento a cualquier requerimiento adicional o coordinación necesaria para proceder con la emisión del certificado.

Atentamente;

Alan Israel García Castro
Estudiante de DS
C.I. 1550197857

03-02-26 10:42

Nota. Oficio de solicitud formal para la emisión del certificado que acredita la implementación exitosa de la aplicación web de automatización del horario docente en el IST Tena; Elaborado por Alan García (2025).

11.2. Encuestas y Entrevistas

11.2.1. Lenguajes de Programación para Backend

- ¿Cuáles son las principales dificultades que enfrenta actualmente al elaborar o recibir su horario docente?
- ¿Qué información considera indispensable que aparezca en su horario (materias, aulas, actividades complementarias, etc.)?
- ¿Con qué frecuencia necesita realizar cambios o ajustes en su horario durante el semestre?
- ¿Qué herramientas digitales utiliza actualmente para organizar sus clases y actividades?
- ¿Qué características le gustaría que tenga un sistema web para facilitar la gestión de su horario?

11.2.2. Entrevista a Rectorado

- ¿Cuáles son los principales problemas que observa en la planificación manual de horarios docentes?
- ¿Qué criterios institucionales deben cumplirse obligatoriamente en la elaboración de los horarios?
- ¿Cómo se gestionan actualmente las aprobaciones y correcciones de los horarios?
- ¿Qué beneficios espera obtener la institución al contar con un sistema automatizado de horarios?
- ¿Qué aspectos de seguridad y confiabilidad considera prioritarios en un sistema web de gestión académica?

11.2.3. Entrevista a Coordinador de Carrera

- ¿Qué pasos sigue actualmente para organizar y validar los horarios de los docentes de su carrera?
- ¿Cuáles son los problemas más frecuentes que enfrenta al coordinar la asignación de aulas y recursos?

- ¿Qué tipo de reportes o documentos necesita generar regularmente sobre los horarios?
- ¿Qué funcionalidades considera necesarias para que un sistema web le facilite la coordinación académica?
- ¿Cómo cree que un sistema automatizado podría mejorar la comunicación entre docentes y coordinación?