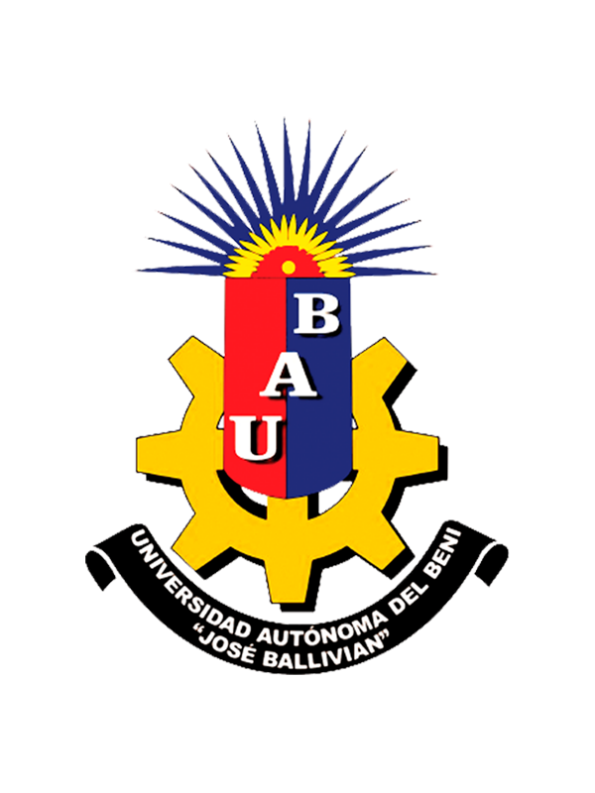
**UNIVERSIDAD AUTÓNOMA DEL BENI**

**“JOSÉ BALLIVIÁN”**

**FACULTAD DE INGENIERÍA Y TECNOLOGÍA**

**CARRERA DE INGENIERÍA DE SISTEMAS**

****

**PROYECTO DE GRADO PARA OPTAR AL TÍTULO DE**

**INGENIERÍA DE SISTEMAS**

**TÍTULO**

**SISTEMA WEB PARA LA GESTIÓN DE RESERVAS DE ESPACIOS ACADÉMICOS EN EL CAMPUS “HERNÁN MELGAR JUSTINIANO” DE LA**

**UNIVERSIDAD AUTÓNOMA DEL BENI “JOSÉ BALLIVIÁN”**

CARMEN MIRNA IBAÑEZ SANGUINO

**POSTULANTE**

ING. JOHNNY ROSAS CALLAÚ

**TUTOR**

LIC. WILLIAM CHAO RIVERO

**TUTOR**

**OCTUBRE DE 2025**

**TRINIDAD – BENI – BOLIVIA**

**Trinidad …………. De ……………………… de 2025**

**Proyecto de Grado:** “Sistema web para la gestión de reservas de espacios académicos en el campus universitario “Hernán Melgar Justiniano” de la U.A.B.J.B.”

**Postulante:**

Univ. Carmen Mirna Ibañez Sanguino

Lic. William Chao Rivero Ing. Johnny Rosas Callaú

**TUTOR TUTOR**

Ing. Gabriela Rosas Mejía

**TRIBUNAL**

Ing. Raúl Marcelo Calderón Fuertes Ing. José Luis Machicado Terán

**TRIBUNAL TRIBUNAL**

**DEDICATORIA**

*A Dios, por guiar mis pasos.*

*A mis padres, pilares fundamentales de mi formación.*

*A mis hermanos por su apoyo en los momentos más difíciles.*

*A mis mascotas que me acompañaron en esas noches de desvelo.*

**AGRADECIMIENTOS**

Expreso mi más sincero agradecimiento a todas las personas e instituciones que hicieron posible la realización de este proyecto de grado.

A la Universidad Autónoma del Beni "José Ballivián" y a la Facultad de Ingeniería y Tecnología, por brindarme la formación académica y los recursos necesarios para mi desarrollo profesional.

Al Lic. William Chao Rivero mi tutor, por su orientación metodológica, sus valiosos consejos y su paciencia durante todo el proceso de investigación y desarrollo de este proyecto.

A mis amigos, por su compañía y apoyo incondicional durante toda mi formación universitaria. Gracias por estar presentes en los momentos difíciles, por celebrar cada logro alcanzado, por las largas jornadas de estudio compartidas, y por nunca dejarme rendir cuando el camino se tornaba complicado. Su amistad fue un pilar fundamental para llegar hasta aquí.

A mi familia, por su apoyo incondicional, comprensión y motivación constante durante toda mi formación universitaria y especialmente durante el desarrollo de este proyecto.

ÍNDICE

[CAPÍTULO I 1](#_Toc212584950)

[INTRODUCCIÓN 1](#_Toc212584951)

[1.1 ANTECEDENTES 2](#_Toc212584952)

[1.2 PLANTEAMIENTO DEL PROBLEMA 2](#_Toc212584953)

[1.3 TÍTULO DEL PROYECTO 3](#_Toc212584954)

[1.4 LOCALIZACIÓN DEL PROYECTO 3](#_Toc212584955)

[1.5 NOMBRE DE LA ENTIDAD, GRUPO O REGIÓN BENEFICIARIA 4](#_Toc212584956)

[1.6 OBJETIVOS 4](#_Toc212584957)

[1.6.1. OBJETIVO GENERAL 4](#_Toc212584958)

[1.6.2. OBJETIVOS ESPECÍFICOS 4](#_Toc212584959)

[1.7 ALCANCE DEL PROYECTO 5](#_Toc212584960)

[1.7.1 CONTEXTO DEL SISTEMA 6](#_Toc212584961)

[1.8 JUSTIFICACIÓN 8](#_Toc212584962)

[1.8.1 ACADÉMICA 8](#_Toc212584963)

[1.8.2 TECNOLÓGICA 8](#_Toc212584964)

[1.8.3 SOCIAL 8](#_Toc212584965)

[CAPÍTULO II 9](#_Toc212584966)

[MARCO TEÓRICO 9](#_Toc212584967)

[2.1 MARCO TEÓRICO CIENTÍFICO 9](#_Toc212584968)

[2.1.1 TEORÍA GENERAL DE SISTEMAS (TGS) 9](#_Toc212584969)

[2.1.2 TEORÍA DE BASE DE DATOS 9](#_Toc212584970)

[2.1.3 TEORÍA DE CIENCIAS DE LA COMPUTACIÓN 10](#_Toc212584971)

[2.1.4 TEORÍA SOCIO-TÉCNICA 10](#_Toc212584972)

[2.2 MARCO TEÓRICO DEL NEGOCIO 11](#_Toc212584973)

[2.2.1 ESPACIO ACADÉMICO 11](#_Toc212584974)

[2.2.2 SOLICITUD DE RESERVA 11](#_Toc212584975)

[2.2.3 ESTADOS DE SOLICITUD 11](#_Toc212584976)

[2.2.4 EVENTO EXTRACURRICULLAR 12](#_Toc212584977)

[2.2.5 ROLES DE USUARIO EN SISTEMA DE GESTIÓN ACADÉMICA 12](#_Toc212584978)

[2.3 MARCO TEÓRICO DEL SISTEMA 12](#_Toc212584979)

[2.3.1 SISTEMA 12](#_Toc212584980)

[2.3.2 SISTEMA DE INFORMACIÓN 13](#_Toc212584981)

[2.3.3 SISTEMA WEB 13](#_Toc212584982)

[2.3.4 SISTEMA DE GESTIÓN DE ESPACIOS 13](#_Toc212584983)

[2.3.5 AUTOMATIZACIÓN DE PROCESOS 13](#_Toc212584984)

[2.3.6 CONTROL DE ACCESO 13](#_Toc212584985)

[2.3.7 SEGURIDAD DE INFORMACIÓN 13](#_Toc212584986)

[2.3.8 BASE DE DATOS RELACIONAL 14](#_Toc212584987)

[2.3.9 POSTGRESQL 14](#_Toc212584988)

[2.3.10 PYTHON 14](#_Toc212584989)

[2.3.11 DJANGO 14](#_Toc212584990)

[2.3.12 ARQUITECTURA MTV (MODEL-TEMPLATE-VIEW) 15](#_Toc212584991)

[2.3.13 DOCKER 15](#_Toc212584992)

[2.3.14 CONTROL DE VERSIONES CON GIT 15](#_Toc212584993)

[2.3.15 INTERFAZ DE USUARIO (UI) 16](#_Toc212584994)

[2.3.16 EXPERIENCIA DE USUARIO (UX) 16](#_Toc212584995)

[2.3.17 METODOLOGÍA 16](#_Toc212584996)

[2.3.18 METODOLOGÍAS ÁGILES 16](#_Toc212584997)

[2.3.19 PROGRAMACIÓN EXTREMA (XP) 17](#_Toc212584998)

[2.4 MARCO LEGAL 17](#_Toc212584999)

[CAPÍTULO III 20](#_Toc212585000)

[METODOLOGÍA 20](#_Toc212585001)

[3.1 METODOLOGÍA DE LA INVESTIGACIÓN 20](#_Toc212585002)

[3.1.1 ENFOQUE DE LA INVESTIGACIÓN 20](#_Toc212585003)

[3.1.2 TIPO DE INVESTIGACIÓN 20](#_Toc212585004)

[3.2 POBLACIÓN Y MUESTRA 21](#_Toc212585005)

[3.2.1 POBLACIÓN DE ESTUDIO 21](#_Toc212585006)

[3.2.2 MUESTRA 22](#_Toc212585007)

[3.3 MÉTODOS 23](#_Toc212585008)

[3.3.1 MÉTODOS TEÓRICOS 23](#_Toc212585009)

[3.3.2 MÉTODOS EMPÍRICOS 23](#_Toc212585010)

[3.4 METODOLOGÍA PROGRAMACIÓN EXTREMA 25](#_Toc212585011)

[3.4.1 CICLO DE VIDA DE XP 25](#_Toc212585012)

[3.4.2 FASES DE LA PROGRAMACIÓN EXTREMA 26](#_Toc212585013)

[3.4.3 VALORES DE LA PROGRAMACIÓN EXTREMA 27](#_Toc212585014)

[3.4.4 VALORES DE LA PROGRAMACIÓN EXTREMA 28](#_Toc212585015)

[3.4.5 PRINCIPIOS DE LA PROGRAMACIÓN EXTREMA 29](#_Toc212585016)

[3.4.6 HERRAMIENTAS DE LA PROGRAMACIÓN EXTREMA 31](#_Toc212585017)

[CAPÍTULO IV 36](#_Toc212585018)

[PROPUESTA 36](#_Toc212585019)

[4.1 DIAGNÓSTICO 36](#_Toc212585020)

[4.1.1 DATOS GENERALES 36](#_Toc212585021)

[4.1.2 INFORMACIÓN RECOPILADA EN BASE A ENTREVISTAS 36](#_Toc212585022)

[4.2 DESCRIPCIÓN DEL SISTEMA 39](#_Toc212585023)

[4.2.1 TECNOLOGÍAS UTILIZADAS EN EL SISTEMA 40](#_Toc212585024)

[4.2.2 IMPLEMENTACIÓN DE LA METODOLOGÍA 41](#_Toc212585025)

[4.2.3 DESARROLLO DE LA METODOLOGÍA 42](#_Toc212585026)

[4.3 PRESUPUESTO DE INVERSIÓN 56](#_Toc212585027)

[4.3.1 PRESUPUESTO DE DESARROLLO 56](#_Toc212585028)

[4.3.2 PRESUPUESTO DE DESPLIEGUE 58](#_Toc212585029)

[4.3.3 PRESUPUESTO DE INVERSIÓN 60](#_Toc212585030)

[4.3.4 PRESUPUESTO DE OPERACIÓN 61](#_Toc212585031)

[4.3.5 PRESUPUESTO DE CAPACITACIÓN 62](#_Toc212585032)

[CAPÍTULO V 64](#_Toc212585033)

[CONCLUSIONES Y RECOMENDACIONES 64](#_Toc212585034)

[5.1 CONCLUSIONES 64](#_Toc212585035)

[5.2 RECOMENDACIONES 65](#_Toc212585036)

[BIBLIOGRAFÍA 66](#_Toc212585037)

[ANEXO 70](#_Toc212585038)

**RESUMEN**

La gestión de reservas del Salón Bicentenario y Sala Audiovisual de la Facultad de Ingeniería y Tecnología en la Universidad Autónoma del Beni "José Ballivián" se realiza mediante procesos manuales basados en cartas físicas, registros en Excel y Outlook, generando conflictos de horarios, retrasos, duplicación de solicitudes y ausencia de trazabilidad.

Este proyecto desarrolló un sistema web centralizado para automatizar la gestión de reservas de espacios académicos del Campus "Hernán Melgar Justiniano". La metodología combinó enfoque mixto con diseño descriptivo-propositivo mediante entrevistas semiestructuradas con 2 encargados y 1 jefe de estudio, observación directa y análisis documental, confirmando la necesidad urgente de digitalización.

El desarrollo siguió metodología ágil Programación Extrema (XP) ejecutada en cuatro iteraciones de dos semanas, empleando tecnologías de código abierto: Python 3.11 con Django 4.2, PostgreSQL 15, Materialize CSS y Docker. La arquitectura MTV garantiza escalabilidad, mantenibilidad y seguridad.

El sistema implementa seis módulos: seguridad con autenticación basada en roles, gestión de usuarios con registro autónomo, gestión de espacios académicos, gestión de solicitudes con notificaciones automáticas por correo, calendario con visualización en tiempo real, y reportes exportables en PDF y Excel.

La solución desarrollada es funcional, segura y escalable con inversión monetaria cero, lista para despliegue institucional con valor comercial equivalente de Bs. 45,000. Beneficia a usuarios mediante solicitudes simplificadas, a encargados con gestión centralizada y reportes automáticos, y a la institución con datos precisos para planificación.

La arquitectura modular permite expansión futura a todos los espacios del campus, estableciendo bases para la transformación digital administrativa en la U.A.B.J.B.

**ABSTRACT**

Reservations for the Bicentennial Hall and Audiovisual Room of the Faculty of Engineering and Technology at the José Ballivián Autonomous University of Beni are managed through manual processes based on physical letters, Excel and Outlook records, generating scheduling conflicts, delays, duplicate requests, and a lack of traceability. This project developed a centralized web-based system to automate the reservation management of academic spaces on the Hernán Melgar Justiniano Campus. The methodology combined a mixed approach with descriptive-propositional design through semi-structured interviews with two supervisors and one head of studies, direct observation, and documentary analysis, confirming the urgent need for digitalization. Development followed the Extreme Programming (XP) agile methodology executed in four two-week iterations, using open-source technologies: Python 3.11 with Django 4.2, PostgreSQL 15, Materialize CSS, and Docker. The MTV architecture guarantees scalability, maintainability, and security. The system implements six modules: security with role-based authentication, user management with autonomous registration, academic space management, request management with automatic email notifications, a calendar with real-time views, and exportable reports in PDF and Excel. The developed solution is functional, secure, and scalable with zero monetary investment, ready for institutional deployment representing a commercial value equivalent to Bs. 45,000. It benefits users through simplified requests, administrators through centralized management and automatic reports, and the institution with accurate data for planning. The modular architecture allows for future expansion to all campus spaces, laying the groundwork for the administrative digital transformation at U.A.B.J.B.

# CAPÍTULO I

# INTRODUCCIÓN

La gestión eficiente de los recursos institucionales constituye uno de los pilares fundamentales para el óptimo funcionamiento de las organizaciones educativas. En el contexto universitario, los espacios académicos representan activos estratégicos cuya administración inadecuada puede generar conflictos organizacionales, subutilización de recursos y limitaciones en la planificación institucional.

La Universidad Autónoma del Beni "José Ballivián" (U.A.B.J.B.), a través de su Campus Universitario "Hernán Melgar Justiniano", enfrenta desafíos significativos en la gestión de reservas de espacios académicos, particularmente en el Salón Bicentenario y la Sala Audiovisual de la Facultad de Ingeniería y Tecnología (F.I.T.).

El proceso actual, basado en métodos manuales que incluyen cartas físicas, registros en Excel y correos electrónicos mediante Microsoft Outlook, ha evidenciado múltiples deficiencias que afectan tanto la eficiencia administrativa como la experiencia de los usuarios. En la era de la transformación digital, donde las Tecnologías de Información y Comunicación (TIC) ofrecen soluciones innovadoras para la optimización de procesos, resulta imperativo modernizar los sistemas de gestión institucional. Los sistemas de información web representan una alternativa tecnológica viable que permite centralizar, automatizar y transparentar los procesos administrativos, facilitando el acceso democrático a los recursos institucionales y proporcionando información precisa para la toma de decisiones estratégicas.

El presente proyecto de grado surge como respuesta a esta necesidad institucional, proponiendo el desarrollo de un sistema web integral que automatice la gestión de reservas de espacios académicos en el Campus "Hernán Melgar Justiniano". Esta solución tecnológica no solo busca resolver las problemáticas operativas identificadas, sino también sentar las bases para la digitalización progresiva de los procesos administrativos en la U.A.B.J.B., contribuyendo así a la modernización institucional y al mejoramiento de la calidad de los servicios universitarios.

## 1.1 Antecedentes

La gestión de reservas para el Salón Bicentenario y la Sala Audiovisual de la Facultad de Ingeniería y Tecnología (F.I.T.) en la Universidad Autónoma del Beni "José Ballivián" (U.A.B.J.B.) se sigue realizando de manera manual. Este proceso ha demostrado ser poco eficiente y propenso a errores. Por más de una década, la administración del Salón Bicentenario se ha basado en la recepción de cartas físicas y un uso limitado de herramientas ofimáticas como Microsoft Outlook, operando en un entorno tecnológico que no ha sido actualizado. Esta metodología genera retrasos en las confirmaciones de reserva, errores administrativos y una considerable sobrecarga de trabajo para el personal responsable.

En el caso de la Sala Audiovisual de la FIT, el procedimiento es aún más complejo. Para eventos como las defensas académicas, la coordinación de fechas y horarios recae directamente en los jefes de estudio de las carreras de Ingeniería Civil e Ingeniería de Sistemas. Para otro tipo de actividades, como conferencias o reuniones, las solicitudes deben ser dirigidas al decanato a través de cartas físicas, y una vez aprobadas, se notifica al encargado de la sala para su registro.

Toda la información referente a las reservas se almacena en archivos de Microsoft Outlook y Microsoft Excel. Si bien este método representa una digitalización básica, carece de las ventajas de un sistema centralizado y en tiempo real, lo que deriva en problemas como la falta de un control eficiente, inseguridad de la información y desorganización de los archivos. Estas deficiencias dificultan el seguimiento del uso de los espacios, complican la planificación de mejoras y afectan negativamente la toma de decisiones estratégicas por la ausencia de datos precisos sobre asistencia y utilización de recursos.

## 1.2 Planteamiento del Problema

La falta de un sistema centralizado y automatizado para la gestión de reservas en el Salón Bicentenario y la Sala Audiovisual de la Facultad de Ingeniería y Tecnología ha generado una serie de problemas operativos que afectan la eficiencia administrativa y la experiencia de los usuarios. Entre los principales efectos identificados se encuentran: conflictos de horarios, subutilización de espacios, retrasos en la confirmación de reservas, duplicación de solicitudes, falta de trazabilidad en los procesos y ausencia de información histórica para la toma de decisiones.

Estos problemas se originan en el uso prolongado de métodos manuales como el registro en archivos Excel, la gestión a través de Microsoft Outlook de forma local y la recepción de cartas físicas, lo que impide una visibilidad en tiempo real de la disponibilidad de espacios y genera una carga administrativa innecesaria. Además, la dependencia de la presencia física del encargado y la ausencia de notificaciones automáticas dificultan la comunicación eficiente con los solicitantes.

Esta situación ha generado insatisfacción en la comunidad universitaria, retrasos en la planificación de actividades académicas y riesgo de pérdida de información importante, afectando negativamente la imagen institucional y el uso óptimo de los recursos disponibles. Ante este contexto, surge la siguiente pregunta de investigación:

¿Cómo se puede mejorar la eficiencia, trazabilidad y seguridad en la gestión de reservas de Espacios Académicos en el Campus Universitario “Hernán Melgar Justiniano” de la U.A.B.J.B.?

## 1.3 Título del Proyecto

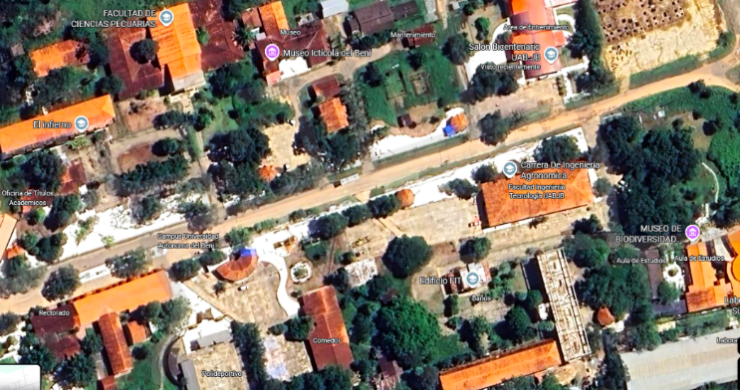
Sistema web para la gestión de reservas de espacios académicos en el campus universitario “Hernán Melgar Justiniano” de la Universidad Autónoma del Beni “José Ballivián”

## 1.4 Localización del Proyecto

El proyecto fue desarrollado en las instalaciones del Campus Universitario “Hernán Melgar Justiniano” de la Universidad Autónoma del Beni “José Ballivián”, con beneficiarios iniciales el Salón Bicentenario y la Sala Audiovisual de la Facultad de Ingeniería y Tecnología, ubicada en la ciudad de Trinidad, departamento del Beni, Estado Plurinacional de Bolivia.

**FIGURA N° 1**

**UBICACIÓN GEOGRÁFICA DEL CAMPUS "HERNÁN MELGAR JUSTINIANO"**



**Fuente:** Google Maps, 2025

## 1.5 Nombre de la entidad, grupo o región beneficiaria

La entidad beneficiaria principal es el Campus Universitario “Hernán Melgar Justiniano” y la Facultad de Ingeniería y Tecnología de la U.A.B.J.B. Los beneficiarios directos incluyen al personal administrativo encargado de la gestión de ambos espacios, docentes, estudiantes y directivos de la facultad.

## 1.6 Objetivos

### 1.6.1. Objetivo General

Desarrollar un sistema web escalable, transparente y mantenible para la gestión eficiente de reservas del Salón Bicentenario y la Sala Audiovisual de la Facultad de Ingeniería y Tecnología de la U.A.B.J.B., que optimice el proceso de solicitud, aprobación y control de reservas mediante automatización, garantizando trazabilidad completa de las operaciones, mejorando la experiencia de usuarios y facilitando la toma de decisiones basada en datos confiables y auditables.

### 1.6.2. Objetivos Específicos

* Analizar el proceso actual de gestión de reservas para el Salón Bicentenario y la Sala Audiovisual, a fin de identificar puntos críticos y requerimientos funcionales del nuevo sistema.
* Diseñar la arquitectura del sistema web y una interfaz de usuario intuitiva y accesible, que responda a las necesidades identificadas de los distintos perfiles de usuario.
* Desarrollar un sistema web funcional que automatice el proceso de solicitud, confirmación y gestión de reservas, utilizando tecnologías modernas y metodologías ágiles.
* Validar el sistema mediante pruebas de funcionalidad, usabilidad y rendimiento, asegurando el cumplimiento de los requerimientos establecidos y su preparación para implementación institucional futura.

## 1.7 Alcance del Proyecto

### 1.7.1 contexto del proyecto

El proyecto desarrollará un sistema web para gestionar reservas de espacios académicos del Campus Universitario "Hernán Melgar Justiniano" de la Universidad Autónoma del Beni "José Ballivián", involucrando todas las facultades y carreras que requieren coordinar el uso de infraestructura compartida para actividades extracurriculares. El sistema procesará información de solicitudes, espacios disponibles y eventos mediante una plataforma centralizada accesible para tres perfiles de usuario.

Universidad Autónoma del Beni "José Ballivián": Institución pública de educación superior del departamento del Beni, encargada de formar profesionales mediante facultades distribuidas en campus universitarios.

Campus "Hernán Melgar Justiniano": Unidad de infraestructura educativa que concentra facultades, carreras y espacios académicos compartidos, requiriendo gestión automatizada de reservas.

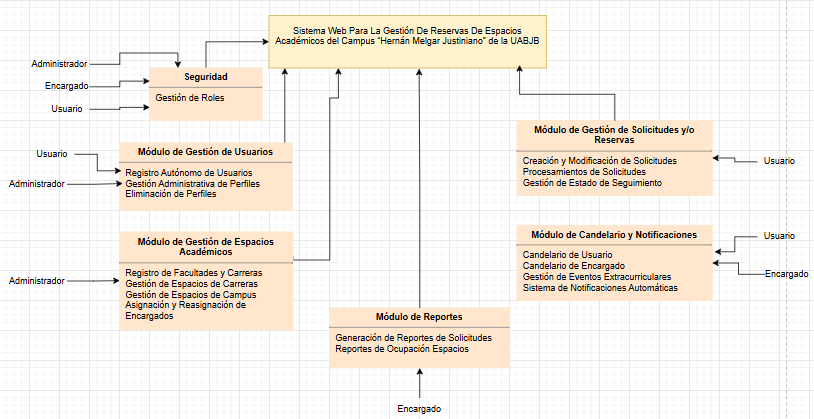
* Espacios Académicos: Infraestructura física para actividades educativas clasificada en:
* Espacios de Campus: Instalaciones de uso común (Ej: Salón Bicentenario)
* Espacios de Carrera: Instalaciones específicas de facultades (Ej: Sala Audiovisual FIT)
* Administrador del Sistema: Usuario responsable de configurar la estructura organizativa, registrando facultades, carreras, espacios y asignando encargados a los mismos.
* Encargado de Espacios: Personal designado para gestionar solicitudes de reserva, aprobar o rechazar peticiones según disponibilidad y políticas institucionales, y mantener control del uso de espacios bajo su responsabilidad.
* Usuario Solicitante: Miembro de la comunidad universitaria autorizado para solicitar reservas mediante el sistema, adjuntando documentación aprobada por su dirección de carrera.
* Solicitud de Reserva: Petición formal para uso de espacio académico en fecha y horario específicos, sujeta a validación del encargado correspondiente.
* Evento Académico: Actividad educativa, cultural o institucional que requiere infraestructura académica, incluyendo defensas de grado, conferencias, talleres, capacitaciones y ceremonias institucionales.

### 1.7.2 Contexto del sistema

El diseño del sistema web integral para la gestión de espacios académicos y reservas se basa en la creación de seis módulos principales: el módulo de seguridad, el módulo de gestión de usuarios, el módulo de gestión de espacios académicos, el módulo de gestión de solicitudes y reservas, el módulo de calendario y notificaciones, y el módulo de reportes. Cada uno de estos módulos desempeña un papel crucial en la funcionalidad global del sistema, garantizando que las operaciones de reserva y administración de espacios se realicen de manera segura, eficiente y organizada.

**FIGURA N° 2**

**MÓDULOS DEL SISTEMA**



**Fuente:** Elaboración Propia, 2025

El sistema web desarrollado está basado en seis módulos. Los módulos se presentan y explican a continuación:

1. Módulo de Seguridad: Autenticación, autorización y control de acceso basado en roles (administrador, encargado, usuario regular).
2. Módulo de Gestión de Usuarios: Registro autónomo, perfiles diferenciados y administración de información institucional.
3. Módulo de Gestión de Espacios Académicos: Registro de facultades, carreras, espacios de campus y espacios de carrera, con asignación de encargados.
4. Módulo de Gestión de Solicitudes y Reservas: Creación, modificación, aprobación y rechazo de solicitudes con validación automática de conflictos.
5. Módulo de Calendario y Notificaciones: Visualización en tiempo real de disponibilidad y sistema automatizado de notificaciones por correo electrónico.
6. Módulo de Reportes: Generación de informes estadísticos exportables en múltiples formatos para análisis y toma de decisiones.

## Justificación

### 1.8.1 Académica

El proyecto permite aplicar los conocimientos adquiridos en el proceso de formación de la carrera de ingeniería de sistemas en el área de análisis de sistemas, desarrollo web, diseño de bases de datos y gestión de proyectos, adquiridos durante la formación en Ingeniería de Sistemas, para resolver una problemática real dentro de la comunidad universitaria.

### 1.8.2 Tecnológica

Se generó una solución con arquitectura MTV (Model-Template-View) basada en Python 3.11 y Django 4.2 como framework web, PostgreSQL 15 como sistema gestor de base de datos relacional y Docker para contenedorización del entorno de desarrollo y despliegue. Esta combinación de tecnologías open source asegura escalabilidad para el futuro, código mantenible gracias a la separación de responsabilidades, seguridad con autenticación robusta y control de acceso basado en roles, y un costo cero por licencia. De este modo, el sistema permitirá modernizar los procesos administrativos de la U.A.B.J.B. a la vez que fortalece la infraestructura tecnológica institucional, lo que sienta las bases para futuros proyectos de digitalización.

### 1.8.3 Social

El sistema fue desarrollado para optimizar la gestión administrativa de reservas del salón Bicentenario y la Sala Audiovisual, simplificando los procesos de solicitud y aprobación, lo cual beneficia indirectamente a estudiantes, docentes y personal administrativo que utilizan estos espacios para actividades académicas.

# CAPÍTULO II

# MARCO TEÓRICO

## 2.1 Marco Teórico CIENTÍFICO

### 2.1.1 Teoría General de Sistemas (TGS)

La Teoría General de Sistemas, formulada por Ludwig von Bertalanffy, es un enfoque interdisciplinario que permite analizar los fenómenos como conjuntos de elementos interrelacionados que interactúan para alcanzar un objetivo común (Bertalanffy, 1976). Arnold y Osorio (1998) complementan esta definición estableciendo que un sistema puede ser entendido como un conjunto de elementos que mantienen determinadas relaciones entre sí y que se encuentran separados de un entorno determinado.

En el contexto del presente proyecto, el sistema de gestión de espacios académicos puede ser concebido como un sistema dinámico donde los usuarios, espacios físicos, horarios y procesos de reserva interactúan para optimizar el uso de recursos institucionales.

### 2.1.2 TEORÍA DE BASE DE DATOS

La teoría de bases de datos tiene sus fundamentos en el modelo relacional propuesto por Edgar Frank Codd en 1970 en su artículo seminal "A Relational Model of Data for Large Shared Data Banks" (Codd, 1970). Este modelo revolucionó la forma de organizar y acceder a los datos al proponer que toda la información se representara mediante relaciones (tablas) compuestas por tuplas (filas) y atributos (columnas).

Según Sánchez (2024), el modelo relacional se basa en la teoría matemática de relaciones y álgebra relacional, estableciendo que las bases de datos deben cumplir con propiedades fundamentales de integridad, consistencia y normalización. Codd posteriormente definió las 12 reglas que un Sistema Gestor de Bases de Datos (SGBD) debe cumplir para ser considerado verdaderamente relacional, incluyendo la representación de información mediante valores en tablas, acceso garantizado a todos los datos mediante combinación de nombre de tabla, clave primaria y nombre de columna, y tratamiento sistemático de valores nulos (Universidad Autónoma del Estado de Hidalgo, 2024).

La normalización de bases de datos, iniciada por Codd y continuada por autores como Boyce y Fagin, establece formas normales progresivamente más restrictivas (1FN, 2FN, 3FN, FNBC, 4FN, 5FN) que garantizan la eliminación de redundancias y anomalías en los datos (Sánchez, 2024). Este proyecto aplica estos principios mediante PostgreSQL, garantizando integridad referencial y cumplimiento de la tercera forma normal.

### 2.1.3 TEORÍA DE CIENCIAS DE LA COMPUTACIÓN

Las ciencias de la computación constituyen el estudio sistemático de los procesos algorítmicos que describen y transforman información, incluyendo su teoría, análisis, diseño, eficiencia e implementación (Knuth, 1997). Donald Knuth, considerado el padre del análisis de algoritmos, estableció en su obra monumental "The Art of Computer Programming" las bases formales para el estudio riguroso de algoritmos y estructuras de datos.

Según Knuth (1997), un algoritmo es un conjunto finito de instrucciones bien definidas que, cuando se ejecutan, producen un resultado específico a partir de un conjunto dado de entradas. La ciencia de la computación se fundamenta en conceptos teóricos como complejidad computacional, teoría de autómatas, teoría de lenguajes formales y teoría de la información.

En el contexto de este proyecto, los principios de ciencias de la computación se aplican en el diseño de estructuras de datos eficientes para almacenar reservas, el desarrollo de algoritmos de validación de conflictos de horarios, y la implementación de patrones de diseño de software que garantizan la mantenibilidad y escalabilidad del sistema.

### 2.1.4 TEORÍA SOCIO-TÉCNICA

La teoría socio-técnica propone que los sistemas tecnológicos deben ser diseñados considerando tanto los aspectos técnicos como los sociales (Emery & Trist, 1960; Trist, 1981). En el desarrollo del sistema de reservas, es fundamental alinear la tecnología con las dinámicas organizacionales, los roles de los usuarios y sus necesidades reales. La optimización conjunta de ambos subsistemas (técnico y social) mejora la adopción y sostenibilidad del sistema implementado.

## 2.2 MARCO TEÓRICO DEL NEGOCIO

### 2.2.1 ESPACIO ACADÉMICO

Un espacio académico en el contexto de educación superior se caracteriza por combinar variables físicas, ambientales y espaciales que, junto con su diseño y organización, facilitan el aprendizaje. En el presente proyecto, se define como toda área física destinada a actividades formativas, investigación o extensión universitaria. Puede clasificarse en espacios de carrera (pertenecientes a una facultad específica) o espacios comunes del campus (de uso compartido por toda la comunidad universitaria). Cada espacio cuenta con atributos específicos como capacidad, ubicación, disponibilidad horaria y personal responsable de su gestión (González-Zamar y Abad-Segura, 2020).

### 2.2.2 SOLICITUD DE RESERVA

Una solicitud de reserva constituye el trámite formal mediante el cual un miembro de la comunidad académica (estudiante, docente o administrativo) requiere el uso de un espacio en fecha y horario específicos. Los sistemas de gestión de reservas de aulas permiten a los usuarios solicitar espacios para sus actividades académicas, generando notificaciones automáticas para los responsables de su validación. La solicitud debe incluir información como: nombre del evento, descripción, fecha, hora, espacio solicitado y documentación de respaldo aprobada por la autoridad competente (Universidad Carlos III de Madrid, 2017).

### 2.2.3 ESTADOS DE SOLICITUD

Los sistemas de reservas universitarios clasifican las solicitudes en tres estados principales según su procesamiento (Universidad de Granada, 2024):

* **Pendiente**: Solicitud recibida que aún no ha sido procesada por el responsable del espacio y está sujeta a validación.
* **Aceptada**: Solicitud aprobada con confirmación del espacio y horario solicitados.
* **Rechazada**: Solicitud denegada con justificación del motivo de rechazo.

### 2.2.4 EVENTO EXTRACURRICULLAR

Actividad académica, cultural o institucional que no forma parte del horario regular de clases, pero requiere uso de infraestructura académica (González-Zamar & Abad-Segura, 2020). Incluye charlas, talleres, concursos, reuniones estudiantiles, ceremonias institucionales y actividades de extensión universitaria.

### 2.2.5 ROLES DE USUARIO EN SISTEMA DE GESTIÓN ACADÉMICA

La Universidad de Alicante (2024) define al encargado como el personal responsable de la asignación de espacios para actividades académicas, con capacidad para validar o rechazar solicitudes según criterios institucionales establecidos. En el contexto de este proyecto, el encargado puede editar fechas de reservas aceptadas, visualizar calendarios integrados y generar reportes sobre el uso de espacios bajo su responsabilidad.

**Administrador del Sistema**

Usuario con privilegios completos sobre el sistema, responsable de la configuración de espacios, gestión de carreras y facultades, asignación de encargados y acceso a reportes institucionales (Universidad de Alicante, 2024).

**Usuario Solicitante**

Miembro de la comunidad universitaria autorizado para enviar solicitudes de reserva. Puede consultar disponibilidad de espacios, verificar el estado de sus propias solicitudes y recibir notificaciones automáticas sobre cambios en su reserva (Universidad Carlos III de Madrid, 2017).

## 2.3 MARCO conceptual DEL SISTEMA

### 2.3.1 SISTEMA

Un sistema es un conjunto de elementos interrelacionados que interactúan dinámicamente para alcanzar un objetivo común (Ackoff, 1971). En este proyecto, el sistema de gestión de espacios académicos está compuesto por usuarios (solicitantes, encargados, administradores), espacios físicos, reglas institucionales y componentes tecnológicos (base de datos, interfaz web, motor de notificaciones).

### 2.3.2 SISTEMA DE INFORMACIÓN

Un sistema de información es un conjunto organizado de recursos (humanos, tecnológicos, procedimientos) que capturan, almacenan, procesan y distribuyen información para apoyar la toma de decisiones (Laudon & Laudon, 2020). En este caso, el sistema proporciona en tiempo real la disponibilidad de espacios, estado de solicitudes, histórico de uso y alertas de conflictos.

### 2.3.3 SISTEMA WEB

Un sistema web es una aplicación accesible a través de un navegador sin necesidad de instalación local (Pressman, 2019). Esta arquitectura fue seleccionada por su independencia de plataforma, facilidad de mantenimiento y accesibilidad desde cualquier dispositivo con conexión a internet (Sommerville, 2021).

### 2.3.4 SISTEMA DE GESTIÓN DE ESPACIOS

Software diseñado para automatizar la reserva de recursos físicos como salas, auditorios y laboratorios. Sus funcionalidades incluyen calendarios en tiempo real, control de permisos, notificaciones automáticas y generación de reportes (Kearns, 2022).

### 2.3.5 AUTOMATIZACIÓN DE PROCESOS

La automatización permite ejecutar tareas repetitivas sin intervención humana, reduciendo errores y mejorando la eficiencia (La Rosa & ter Hofstede, 2022). En este sistema, se automatizan la validación de horarios, envío de correos, generación de reportes y liberación de espacios.

### 2.3.6 CONTROL DE ACCESO

El control de acceso regula quién puede acceder a qué recursos del sistema, garantizando la confidencialidad y seguridad de la información (Stallings, 2020). En este proyecto, se implementan roles diferenciados: administrador, encargado y solicitante.

### 2.3.7 SEGURIDAD DE INFORMACIÓN

La seguridad de la información implica proteger los datos contra accesos no autorizados, alteraciones o pérdidas (ISO/IEC 27001, 2018). Se aplican principios de confidencialidad, integridad y disponibilidad (CIA) para proteger los datos personales y académicos.

### 2.3.8 BASE DE DATOS RELACIONAL

Una base de datos relacional organiza la información en tablas interrelacionadas mediante llaves primarias y foráneas (Elmasri & Navathe, 2023). Según Alonso (2021), este modelo es el más utilizado en la actualidad, fundamentándose en el álgebra relacional y permitiendo consultas complejas mediante lenguaje SQL. En este sistema, se utiliza PostgreSQL para gestionar usuarios, espacios, reservas y eventos, garantizando la integridad y consistencia de los datos.

### 2.3.9 POSTGRESQL

PostgreSQL es un sistema de gestión de bases de datos relacional de código abierto que hace énfasis en la extensibilidad y conformidad con los estándares (Domínguez, 2019). Zea, Molina y Redrován (2017) destacan que PostgreSQL posee gran escalabilidad, capacidad de ajustarse al número de CPUs y memoria del sistema, implementa rollback's, subconsultas y transacciones, y tiene capacidad de comprobar la integridad referencial. Está liberado bajo licencia PostgreSQL, similar a la licencia MIT, y es desarrollado por el Grupo de Desarrollo Global de PostgreSQL.

### 2.3.10 PYTHON

Python es un lenguaje de programación interpretado de alto nivel que enfatiza la legibilidad del código y soporta múltiples paradigmas de programación, incluyendo orientación a objetos, programación imperativa y funcional (Chazallet, 2024). Según ADR Formación (2024), Python se caracteriza por expresar algoritmos de manera adecuada a la capacidad cognitiva humana, es multiplataforma, utiliza tipado dinámico y necesita de un intérprete que realiza la traducción del código a medida que sea necesaria. Está liberado bajo una licencia de código abierto denominada Python Software Foundation License, y es administrado por la Python Software Foundation.

### 2.3.11 DJANGO

Django es un framework web de alto nivel para Python que fomenta el desarrollo rápido y el diseño limpio mediante el principio "Don't Repeat Yourself" (DRY). Desarrollado inicialmente entre 2003 y 2005 por un equipo responsable de mantener sitios web de periódicos, fue liberado como código abierto en julio de 2005 (Mozilla Developer Network, 2024). Cruz (2024) señala que Django ofrece un esquema modular entre proyecto y aplicaciones, un conjunto importante de paquetes para extender el framework y facilita la creación de proyectos escalables. Sigue el patrón arquitectónico MTV (Model-Template-View) e incluye componentes integrados para autenticación, administración y seguridad.

### 2.3.12 ARQUITECTURA MTV (MODEL-TEMPLATE-VIEW)

La arquitectura MTV es un patrón de diseño derivado del clásico MVC (Model-View-Controller), adaptado específicamente por Django (Greenfeld & Roy, 2022):

* **Model**: Representa la capa de datos. Define la estructura de las tablas de la base de datos y las reglas de negocio.
* **Template**: Maneja la presentación (HTML, CSS, JS). Es la interfaz con la que interactúa el usuario final.
* **View**: Contiene la lógica de negocio y decide qué datos mostrar, qué modelo consultar y qué template renderizar.

MTV permite desacoplar completamente la lógica de presentación de la lógica de negocio, facilitando el mantenimiento, las pruebas unitarias y la escalabilidad del sistema (Forcier, Bissex & Chun, 2023).

### 2.3.13 DOCKER

Docker es una plataforma de contenedorización que permite empaquetar aplicaciones junto con sus dependencias en contenedores ligeros y portables. Garantiza consistencia entre entornos de desarrollo, prueba y producción, eliminando el problema de "en mi máquina funciona" (Pressman, 2019). Los contenedores comparten el kernel del sistema operativo, consumiendo menos recursos que las máquinas virtuales tradicionales.

### 2.3.14 CONTROL DE VERSIONES CON GIT

Git es un sistema de control de versiones distribuido que permite rastrear cambios en el código fuente, facilitar la colaboración entre desarrolladores y mantener un historial completo del proyecto (Sommerville, 2021). A diferencia de sistemas centralizados, Git permite que cada desarrollador tenga una copia completa del repositorio, facilitando el trabajo offline y las ramificaciones del código.

### 2.3.15 INTERFAZ DE USUARIO (UI)

La UI es el conjunto de elementos visuales con los que el usuario interactúa. Shneiderman et al. (2022) establecen ocho reglas de oro para el diseño de interfaces: consistencia, atajos para usuarios expertos, retroalimentación informativa, diseño de diálogos para lograr conclusiones, prevención y manejo de errores, reversión de acciones, control del usuario y reducción de carga de memoria.

### 2.3.16 EXPERIENCIA DE USUARIO (UX)

La UX abarca la percepción general del usuario al interactuar con el sistema. Norman (2021) enfatiza la importancia del diseño centrado en el usuario, donde la funcionalidad, usabilidad y accesibilidad son prioritarias. Se buscó que el sistema fuera intuitivo, con flujos claros para realizar reservas, consultar calendarios y recibir notificaciones.

### 2.3.17 METODOLOGÍA

Una metodología es un conjunto sistemático, estructurado y replicable de procedimientos, técnicas y herramientas que guían el proceso de investigación o desarrollo de un proyecto (Hernández, Fernández & Baptista, 2021). En desarrollo de software, debe ser adaptativa cuando los requisitos evolucionan rápidamente, como ocurre en entornos ágiles o de innovación tecnológica.

### 2.3.18 METODOLOGÍAS ÁGILES

Las metodologías ágiles surgen como respuesta a la rigidez de los modelos tradicionales en cascada. Navarro, Fernández y Morales (2013) señalan que estas metodologías se caracterizan por su adaptabilidad al cambio, la entrega incremental y la colaboración constante con el cliente. El Manifiesto Ágil (2001) establece cuatro valores fundamentales: individuos e interacciones sobre procesos y herramientas; software funcionando sobre documentación extensiva; colaboración con el cliente sobre negociación contractual; y respuesta ante el cambio sobre seguir un plan. Son particularmente efectivas en proyectos de pequeña y mediana escala donde los requisitos cambian con frecuencia.

### 2.3.19 PROGRAMACIÓN EXTREMA (XP)

La Programación Extrema (XP) es una metodología ágil formulada por Kent Beck que propone llevar al extremo ciertas buenas prácticas de la ingeniería de software (Beck & Andres, 2005). Letelier y Penadés (2006) detallan que XP se centra en ciclos de desarrollo muy cortos, pruebas continuas, refactoring frecuente y feedback constante del cliente. Se apoya en cinco valores fundamentales: comunicación, simplicidad, retroalimentación, coraje y respeto, además de doce prácticas clave como integración continua, pruebas unitarias, programación en parejas y planificación incremental.

## 2.4 Marco Legal

La implementación del Sistema web para la gestión de reservas de espacios académicos en el campus universitario “Hernán Melgar Justiniano” de la U.A.B.J.B. se enmarca en un conjunto de normativas institucionales, universitarias y nacionales que respaldan su pertinencia, ejecución y validez académica y legal. Estas disposiciones aseguran que el desarrollo del sistema propuesto cumple con los principios de legalidad, pertinencia educativa, uso responsable de la tecnología y protección de la información. A continuación, se describen los principales marcos legales aplicables:

* **Constitución Política del Estado Plurinacional de Bolivia**
* El Art. 91 y 92 establecen el derecho al acceso a la información y el deber del Estado de romover el desarrollo científico y tecnológico para el bienestar de la sociedad.
* Por su parte el Art. 103 reconoce la ciencia, tecnología e innovación como funciones fundamentales del Estado y promueve su desarrollo a través del sistema educativo.
* **Ley N.º 070 “Avelino Siñani – Elizardo Pérez”**
* En sus Art. 14 y 15 promueve una educación científica, técnica y tecnológica, con pertinencia cultural, regional y contextualizada.
* Fomenta el uso de las tecnologías de información y comunicación (TICs) como medio de transformación educativa y social.
* **Ley N.º 164 (2011)**
* **Ley General de Telecomunicaciones, Tecnologías de Información y Comunicación**

Esta ley regula el uso de redes, servicios y sistemas de información en Bolivia. Aunque está orientada principalmente a operadores y prestadores de servicios, también aplica a instituciones que desarrollan sistemas tecnológicos, estableciendo:

* El respeto a la privacidad de las comunicaciones.
* La seguridad técnica en el manejo de plataformas digitales.
* La responsabilidad institucional frente al mal uso de sistemas de información.
* **Propiedad Intelectual**

El código fuente, diseño de interfaz, documentación técnica y manuales de usuario desarrollados en este proyecto están protegidos por derechos de autor conforme a la Ley N.º 1322 (1992) sobre Derecho de Autor y su modificatoria Ley N.º 1995 (1999).

Esto implica:

* La universidad conserva los derechos patrimoniales sobre el software desarrollado.
* Se debe evitar el uso no autorizado de librerías, imágenes o componentes con licencias restrictivas.
* Se reconocerán los créditos de los autores (estudiantes y tutores) en la documentación y código fuente.
* **Estatuto Orgánico de la Universidad Autónoma del Beni “José Ballivián”**
* Estableceque la formación profesional debe estar vinculada a la realidad social y regional, promoviendo proyectos que resuelvan necesidades del entorno.
* Reconoce la autonomía universitaria en la creación e implementación de sistemas que optimicen la gestión institucional.
* **Reglamento de Proyecto de Grado – Carrera de Ingeniería de Sistemas (Versión 2013)**
* Define al proyecto de grado como una propuesta técnico-científica orientada a resolver problemas reales mediante soluciones tecnológicas, aplicando los conocimientos adquiridos durante la formación universitaria.
* **Plan de Ciencia, Tecnología e Innovación de la U.A.B.J.B. (2019–2026)**
* Promueve el desarrollo de soluciones digitales que fortalezcan la gestión universitaria, la automatización de procesos administrativos y la digitalización de servicios institucionales.

# CAPÍTULO III

# METODOLOGÍA

## 3.1 Metodología de la Investigación

### 3.1.1 Enfoque de la Investigación

La investigación adoptó un enfoque cualitativo, ya que se centra en comprender a profundidad el proceso actual de gestión de reservas desde la perspectiva de los actores clave (encargados de espacios), sin buscar generalizaciones estadísticas a gran escala. Este enfoque permitió:

* Comprender procesos administrativos complejos mediante entrevistas en profundidad
* Identificar requerimientos funcionales específicos del sistema
* Validar soluciones tecnológicas propuestas con usuarios directos

Dado que el sistema está dirigido a optimizar procesos administrativos específicos gestionados por un número reducido de usuarios clave (2 encargados), el enfoque cualitativo resulta más apropiado que un enfoque cuantitativo basado en encuestas masivas. La información crítica reside en quienes operan diariamente el proceso de reservas, no en los usuarios finales ocasionales.

### 3.1.2 Tipo de Investigación

El proyecto se clasifica como investigación aplicada de tipo descriptivo-propositivo:

* Descriptivo: Porque analiza y documenta el proceso actual de gestión de reservas, identificando sus características, deficiencias y puntos críticos.
* Propositivo: Porque propone y desarrolla una solución tecnológica concreta para resolver la problemática identificada.

Adicionalmente, el proyecto incorpora elementos de investigación-acción, ya que no solo se diagnostica un problema, sino que se interviene activamente mediante el desarrollo de una solución tecnológica.

## 3.2 Población y Muestra

### 3.2.1 Población de Estudio

La población del estudio estuvo conformada por el personal administrativo y académico directamente involucrado en la gestión de espacios académicos del Campus "Hernán Melgar Justiniano":

**TABLA N° 1**

**POBLACIÓN DE ESTUDIO**

|  |  |  |
| --- | --- | --- |
| Rol | Cantidad | Espacio Asignado |
| Encargado del Salón Bicentenario | 1 | Salón Bicentenario |
| Encargado de la Sala Audiovisual FIT | 1 | Sala Audiovisual FIT |
| Total | 2 | - |

**Fuente:** Elaboración propia, 2025.

La población se delimitó intencionalmente a estos dos actores clave debido a que:

1. Son quienes gestionan directamente el proceso de reservas en sus respectivos espacios
2. Poseen conocimiento completo del flujo actual de trabajo, sus problemas y necesidades
3. Tienen autoridad para aprobar o rechazar solicitudes
4. Serán los usuarios principales del sistema desarrollado

Esta delimitación excluye a:

* Usuarios finales ocasionales (estudiantes) que solo solicitan espacios esporádicamente
* Personal administrativo no involucrado directamente en la gestión de reservas

### 3.2.2 Muestra

Dada la naturaleza específica del problema y el tamaño reducido de la población, se utilizó un muestreo no probabilístico de tipo censal, trabajando con la totalidad de la población identificada:

n = 2 personas

Compuesta por:

* 2 encargados de espacios académicos

Tipo de muestreo: Censal (100% de la población)

No se aplicó muestreo probabilístico ni se calculó tamaño muestral mediante fórmulas estadísticas porque:

1. El universo es finito y reducido (N=2): Todos los actores clave pueden ser consultados directamente
2. Información crítica y especializada: Solo estos tres actores poseen el conocimiento necesario sobre el proceso actual
3. Enfoque cualitativo: Se prioriza la profundidad sobre la amplitud
4. Accesibilidad total: Disponibilidad completa de todos los integrantes de la población

Aunque estudiantes y docentes son beneficiarios indirectos del sistema, no fueron incluidos en la muestra porque:

* Uso ocasional y heterogéneo: No todos los estudiantes/docentes utilizan estos espacios regularmente
* Conocimiento limitado del proceso administrativo: Desconocen el flujo interno de gestión de reservas
* Información no crítica para el diseño: Sus necesidades se capturan mediante los requerimientos funcionales definidos por los encargados
* Representación indirecta: Los encargados conocen las necesidades recurrentes de los usuarios finales

## 3.3 MÉTODOS

### 3.3.1 Métodos Teóricos

**a) Lógico**

Este enfoque se basa en el uso del razonamiento para comprender la esencia y el desarrollo de los fenómenos, permitiendo reconstruir teóricamente sus aspectos más relevantes desde una perspectiva histórica (Torres-Miranda, T., 2020).

Se utilizó para analizar los antecedentes históricos de los espacios, sacar conclusiones con los datos importantes desde el inicio de su funcionamiento y conocer las condiciones históricas en que se desarrolló el problema.

**b) Analítico y sintético**

Estos métodos se complementan en el proceso investigativo: el análisis permite descomponer un fenómeno en sus partes constituyentes para examinar sus características específicas, mientras que la síntesis reúne esos elementos para formar una visión global y coherente del fenómeno (Universidad Complutense de Madrid, 2017).

**c) Sistémico**

Este enfoque permite abordar el objeto de estudio como un sistema integrado por partes interrelacionadas, cuyas dinámicas internas y externas condicionan su funcionamiento. En el ámbito del desarrollo de software, este método se aplica desde la concepción de requerimientos hasta la implementación y validación del producto final. En este caso, se empleó la metodología Programación Extrema (Extreme Programming, XP), una estrategia ágil que promueve la calidad del código, la adaptabilidad continua y la colaboración estrecha entre los miembros del equipo de desarrollo (Hurtado, M., 2005).

### 3.3.2 Métodos Empíricos

**a) Observación**

La observación es una técnica de recolección de datos que implica la percepción directa e intencional del fenómeno de estudio. A través de ella, el investigador selecciona e interpreta información relevante utilizando sus sentidos, con el propósito de comprender el comportamiento o las características del objeto investigado. En este estudio, la observación permitió identificar el problema de investigación y delimitar el objeto de análisis. Esta técnica estuvo mediada por cuatro elementos clave: el objeto observado, el sujeto observador, el contexto o entorno en el que se desarrolló la observación, y los instrumentos o medios utilizados para registrar la información (Hernández, R., Fernández, C., & Baptista, P., 2021).

Se utilizó la observación como técnica para la recolección de datos, siendo esta intencionada y participante por parte de quien realiza el actual análisis de solicitudes de espacios. Por medio de esta técnica, se identificaron e interpretaron aspectos relevantes del sistema que se está estudiando. Se usaron los sentidos para comprender el sistema. Se identificó el problema de investigación y se delimitó el objeto de análisis, a partir de esta observación. En la delimitación del objeto de estudio, se consideraron cuatro elementos, que son el objeto observado (proceso de solicitudes), sujeto observador (investigador), contexto o entorno (institución educativa) e instrumentos o medio (estudio en proceso).

Se utilizó el método observacional mediante el estudio del proceso real actual de solicitud de espacios de la institución.

**b) Entrevista**

La entrevista es una técnica de recolección de información que se basa en la comunicación interpersonal, donde una persona (entrevistador) formula preguntas a otra o varias personas (entrevistados) con el objetivo de obtener datos relevantes para la investigación. Este método está guiado por objetivos específicos y sigue una estructura o guía previamente definida, lo que permite recopilar información profunda y contextualizada sobre el tema de estudio (Taylor, S. J., Bogdan, R., & DeVault, M., 2016).

Se utilizó la entrevista como técnica de recolección de información sobre el mediante una comunicación interpersonal donde el investigado, en este caso el investigador, hace preguntas a personas claves (encargados de espacios) que aportan datos para la investigación. Este método siguió un marco específico y el uso de un diseño predefinido. Como resultado, se obtuvo información específica y profunda sobre el tema. Esto incluía, por supuesto, sobre las necesidades y problemas esperados sobre el sistema de reserva de espacios.

## 3.4 METODOLOGÍA PROGRAMACIÓN EXTREMA

La Programación Extrema (XP, por sus siglas en inglés) es una metodología ágil de desarrollo de software que busca mejorar la calidad del código y la capacidad de adaptación a cambios, promoviendo una colaboración constante entre el cliente y el equipo de desarrollo. XP integra prácticas probadas de otras metodologías, pero con un enfoque más flexible, iterativo y centrado en el usuario.

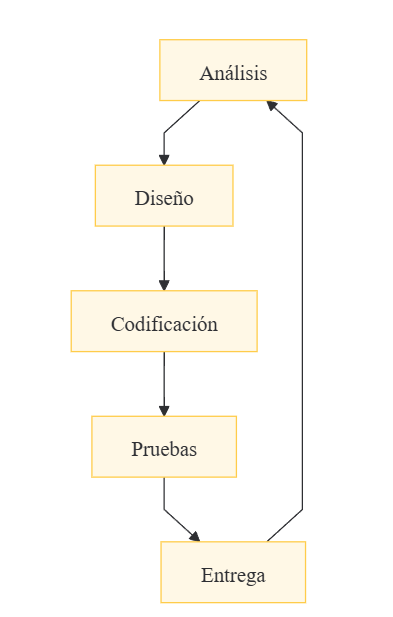
### 3.4.1 CICLO DE VIDA DE XP

El ciclo de vida de XP es iterativo e incremental, lo que permite entregar valor al cliente de forma temprana y continua. El proceso se divide en pequeñas fases que se repiten en cada iteración:

1. Análisis de requerimientos a través de historias de usuario.
2. Diseño y planificación de la iteración.
3. Codificación en parejas (pair programming).
4. Pruebas continuas y automatizadas.
5. Entrega de una versión funcional.

**FIGURA N° 3**

**CICLO DE VIDA DE XP**



**Fuente:** Elaboración propia, 2025

### 3.4.2 FASES DE LA PROGRAMACIÓN EXTREMA

**TABLA N°2**

**FASES DE XP**

|  |  |
| --- | --- |
| FASE | OBJETIVO PRINCIPAL |
| Exploración | Identificar historias de usuario y tecnologías a utilizar. |
| Planificación | Estimar esfuerzo, asignar tareas y definir iteraciones. |
| Producción | Desarrollar, probar y entregar funcionalidades |
| Mantenimiento | Corregir errores y adaptar el sistema a nuevos requerimientos |
| Muerte | Finalización del proyecto cuando ya no se requieren más cambios |

**Fuente:** Elaboración propia, 2025

### 3.4.3 VALORES DE LA PROGRAMACIÓN EXTREMA

**TABLA N° 3**

**VALORES FUNDAMENTALES**

|  |  |
| --- | --- |
| VALOR | DESCRIPCIÓN |
| Comunicación | Diálogo constante entre cliente y equipo. |
| Simplicidad | Diseñar solo lo necesario para satisfacer la necesidad actual. |
| Retroalimentación | Revisar y ajustar el trabajo en cada iteración. |
| Coraje | Tener la valentía de eliminar código innecesario o hacer cambios drásticos. |
| Respeto | Valorar el trabajo de todos los miembros del equipo. |

**Fuente:** Elaboración propia, 2025

### 3.4.4 VALORES DE LA PROGRAMACIÓN EXTREMA

#### 3.4.4.1 COMUNICACIÓN

Cuando se utiliza la Programación Extrema es fundamental la comunicación constante con los usuarios, de esto depende para que el desarrollo se lleve de manera efectiva y que el sistema cumpla exactamente con las necesidades reales.

Los encargados del Salón Bicentenario y la Sala Audiovisual FIT se convirtieron en parte activa del proceso de desarrollo. Fueron ellos quienes, mediante las entrevistas realizadas, proporcionaron las especificaciones exactas del proceso actual y los requerimientos funcionales necesarios para elaborar las historias de usuario.

#### 3.4.4.2 SIMPLICIDAD

La simplicidad en la programación extrema significó que, sin importar las funcionalidades que requirieran los encargados en el sistema, estas debían ser fáciles de usar. El diseño fue sencillo y amigable, el código simple y entendible, programando solo lo necesario y lo que realmente se utilizaría.

#### 3.4.4.3 RETROALIMENTACIÓN

La retroalimentación fue la comunicación constante entre la desarrolladora y los encargados de espacios. Se desarrollaron y programaron las historias de usuario, y después de verificar que cada funcionalidad operaba correctamente, se mostraba a los encargados para que la probaran directamente y decidieran si funcionaba como esperaban o necesitaba modificaciones. Este proceso se llevó a cabo para cada historia de usuario hasta tener el sistema completamente validado.

#### 3.4.4.4 CORAJE

Se refiere a la valentía para modificar o eliminar código que, aunque costó esfuerzo desarrollar, no cumplía con los estándares de calidad o no era requerido realmente en el sistema. También implica la persistencia para resolver errores que surgieran durante la programación sin tomar atajos que comprometieran la calidad.

#### 3.4.4.5 RESPETO

Aunque el desarrollo fue individual, se mantuvo respeto hacia todos los involucrados: encargados de espacios, tutores académicos y el proyecto mismo. Se respetaron los tiempos acordados, se valoró el conocimiento y disponibilidad de los encargados, y se mantuvo un compromiso con la calidad del código y la documentación.

### 3.4.5 PRINCIPIOS DE LA PROGRAMACIÓN EXTREMA

#### 3.4.5.1 PRINCIPIO DE PRUEBAS

Se diseñaron tarjetas de casos de prueba de aceptación en conjunto con los encargados, especificando las entradas y salidas esperadas para cada funcionalidad. Estas tarjetas se elaboraron de acuerdo a las historias de usuario seleccionadas para el desarrollo del sistema.

#### 3.4.5.2 PROCESO DE PLANIFICACIÓN

En la etapa inicial, los encargados especificaron las funcionalidades requeridas mediante entrevistas semiestructuradas.

#### 3.4.5.3 EL CLIENTE EN EL LUGAR

Se mantuvo comunicación constante con los encargados de espacios para obtener retroalimentación continua y resolver dudas durante el desarrollo.

#### 3.4.5.4 PROGRAMACIÓN EN PAREJAS

Este principio no fue aplicado debido a que el desarrollo fue realizado de forma individual como proyecto de grado. Sin embargo, se realizaron sesiones de revisión de código con el tutor académico, quien cumplió un rol similar al revisar decisiones técnicas y arquitectónicas

#### 3.4.5.5 INTEGRACIÓN CONTINUA

Se utilizó Git como sistema de control de versiones, permitiendo integrar nuevas funcionalidades de manera continua sin generar conflictos. Cada historia de usuario completada se integraba inmediatamente al sistema principal después de pasar las pruebas.

#### 3.4.5.6 REFACTORIZACIÓN

Se mejoró continuamente el diseño del código durante el proceso de desarrollo. Se evaluó constantemente la calidad del código y se recodificaron funciones cuando era necesario, minimizando código duplicado o ineficiente.

#### 3.4.5.7 ENTREGAS PEQUEÑAS

Cada sprint de 2 semanas culminaba con una entrega funcional que los encargados podían probar. Esto permitió validar incrementalmente el sistema y realizar ajustes tempranos sin esperar al final del desarrollo.

#### 3.4.5.8 DISEÑO SIMPLE

El sistema se enfocó en cumplir los requerimientos necesarios para los encargados sin agregar complejidad innecesaria. Se cubrieron únicamente las necesidades inmediatas identificadas en el diagnóstico, permitiendo eliminar redundancias y mantener un diseño limpio.

#### 3.4.5.9 METÁFORA

La metáfora es empleada para expresar el desarrollo del proyecto, cómo se define el propósito y el alcance del sistema.

#### 3.4.5.10 propiedad colectiva del código

Aunque el desarrollo fue individual, el código se documentó pensando en que futuros desarrolladores puedan entenderlo y modificarlo sin dificultad. No se consideró el código como propiedad exclusiva, sino como un activo institucional que debe ser mantenible.

#### 3.4.5.11 estándar de codificación

La programación extrema promueve el trabajo en equipo para facilitar la lectura y entendimiento del código. Como el presente proyecto fue desarrollado de forma individual esto facilita la coherencia y facilidad del sistema.

#### 3.4.5.12 RITMO SOSTENIBLE (40 HORAS SEMANALES)

Se planificaron jornadas de trabajo sostenibles para mantener la calidad del código, evitando jornadas extenuantes.

### 3.4.6 HERRAMIENTAS DE LA PROGRAMACIÓN EXTREMA

a) Historias de usuarios

La programación extrema está constituida por las historias de usuarios que facilita el análisis del sistema. Estas historias poseen la información y necesidades de los usuarios.

**TABLA N°4:**

**HISTORIAS DE USUARIOS**

|  |  |
| --- | --- |
| Historia de Usuario | |
| Número: | **Nombre Historia de Usuario:** |
| Modificación (o extensión) de Historia de Usuario (nro. y nombre): | |
| Usuario: | **Iteración Asignada:** |
| Prioridad en Negocio:  (Alta / Media / Baja) | **Puntos Estimados:** |
| Riesgo en Desarrollo:  (Alto / Medio / Bajo) | **Puntos Reales:** |
| Descripción: | |
| Observaciones: | |

**Fuente:** José H. Canos, Patricio Leterier y Carmen Penades, 2009

Los campos son ocupados con la siguiente información:

* Número: número de la historia
* Nombre historia de usuario: nombre de la historia a desarrollar.
* Modificación (o extensión) de historia de usuario (nro. y nombre): si se trata de una modificación de la historia de usuario o extensión se escribe el nombre y número de la historia de usuario a la que pertenece.
* Usuario: tipo de usuario que utilizara este apartado del sistema.
* Iteración asignada: son las iteraciones que se realizarán en esa historia de usuario.
* Prioridad en negocio: que tan importante es para el negocio del cliente el desarrollo de este segmento de código.
* Puntos estimados: tiempo utilizado para realizar cada historia medida en puntos.
* Riesgo en desarrollo: medido entre alto, medio y bajo.
* Puntos reales: puntos reales que se utilizaron para realizar y finalizar la historia.
* Descripción: breve descripción de la manera en que debe funcionar la historia cuando sea programada.
* Observaciones: si es que se da el caso, se anotará una observación en cuanto a la historia y su desarrollo.

En la historia de usuario se describe los requerimientos de los usuarios, que es documentado para llevarlo al desarrollo y codificar, buscando cumplir al pie de la letra lo sugerido.

Si el usuario pretende un cambio se modifica la historia sin modificar todo el sistema.

La historia de usuario tiene 3 aspectos importantes a considerar:

* Tarjeta
* Conversación
* Pruebas de aceptación

b) Tareas de ingeniería

Esta tarjeta segmenta la historia de usuario en actividades para su ejecución y.

sencilla interpretación, tal como se indica en la tabla:

**TABLA N° 5**

**TAREA DE INGENIERÍA**

|  |  |
| --- | --- |
| Tarea de Ingeniería | |
| Número de tarea: | **Historia de usuario (nro. y nombre)** |
| Nombre tarea: | |
| Tipo de Tarea: Desarrollo / Corrección / Mejora / Otra (especificar) | **Puntos estimados:** |
| Programador responsable: | |
| Descripción: | |

**Fuente:** José H. Canos, Patricio Leterier y Carmen Penades, 2009

La información para llenar la tarjeta se específica a continuación:

* Número tarea: número correspondiente a esa tarea.
* Historia de Usuario (nro. y nombre): nombre de la historia de usuario correspondiente a la tarea.
* Nombre tarea: nombre asignado a la tarea.
* Tipo de tarea: es lo que se llevará a cabo.
* Puntos estimados: es el tiempo empleado para completar cada historia medido en puntos.
* Programador responsable: nombre del desarrollador encargado de llevar a cabo la tarea.
* Descripción: breve descripción de lo que se debe realizar.

c) Caso de prueba de aceptación

La tarjeta donde se detalla cómo se realizaron las pruebas, de la narrativa del usuario para verificar si se cumplió el requisito del cliente.

**TABLA N° 6**

**CASO DE PRUEBA DE ACEPTACIÓN**

|  |  |
| --- | --- |
| Caso de prueba de aceptación | |
| Código: | **Historia de usuario (nro y nombre):** |
| Nombre: | |
| Descripción: | |
| Condiciones de ejecución: | |
| Entrada / Pasos de ejecución: | |
| Resultado esperado: | |
| Evaluación de prueba: | |

**Fuente:** José H. Canos, Patricio Leterier y Carmen Penades, 2009

* Código: número del caso de prueba
* Historia de Usuario (nro. y nombre): Número y nombre de la historia de la cual se harán las pruebas.
* Nombre: Nombre del caso de prueba.
* Descripción: Descripción de lo que se está realizando.
* Condiciones de ejecución: Se registra cómo se está llevando a cabo la ejecución y el comportamiento.
* Entrada/Pasos de ejecución: datos o parámetros de entrada que se emplearon para llevar a cabo la prueba.
* Resultado esperado: Lo que se espera obtener al realizar la prueba.
* Evaluación de la Prueba: Es una evaluación de cómo funciono la prueba, si fue exitosa.

La fase de pruebas en la programación extrema se realiza tras haber descrito en su mayoría las narrativas de usuarios. Las pruebas se escriben entre el desarrollador y el usuario antes de que se realice diseñada una historia de usuario, con la finalidad de anticipar fallos previos a que comience la programación.

# CAPÍTULO IV

# PROPUESTA

## 4.1 DIAGNÓSTICO

### 4.1.1 DATOS GENERALES

#### 4.1.1.1 RAZÓN SOCIAL

Universidad Autónoma del Beni "José Ballivián" - Campus Universitario "Hernán Melgar Justiniano"

#### 4.1.1.2 FUNDACIÓN

**Sala Audiovisual FIT**

Implementada como parte de la infraestructura especializada de la Facultad de Ingeniería y Tecnología para atender las necesidades de defensas académicas de las carreras de Ingeniería Civil e Ingeniería de Sistemas, así como para eventos técnicos, capacitaciones, videoconferencias y actividades de extensión universitaria.

#### 4.1.1.3 MISIÓN

Proporcionar espacios académicos de calidad que faciliten el desarrollo de actividades educativas, culturales, científicas e institucionales, garantizando una gestión eficiente, transparente y equitativa de las reservas del Salón Bicentenario y la Sala Audiovisual de la Facultad de Ingeniería y Tecnología, para contribuir al fortalecimiento de la formación integral de la comunidad universitaria del Campus "Hernán Melgar Justiniano".

#### 4.1.1.4 VISIÓN

Ser un referente en la gestión moderna y tecnológica de espacios académicos dentro de la Universidad Autónoma del Beni "José Ballivián", facilitando el acceso equitativo, optimizando el uso de recursos y promoviendo la transformación digital de los procesos administrativos para el beneficio de toda la comunidad universitaria.

### 4.1.2 INFORMACIÓN RECOPILADA EN BASE A ENTREVISTAS

#### 4.1.2.1 ENTREVISTA AL ENCARGADO DEL SALÓN BICENTENARIO

**¿Qué herramientas o software utiliza para gestionar las reservas?**

Microsoft Outlook.

**¿Podría describirme paso a paso cómo funciona el proceso de reserva desde que un usuario lo solicita hasta que se confirma?**

El solicitante tiene que llegar con al menos 3 días de anticipación y con una carta firmada por su dirección de carrera solicitando el salón bicentenario con el nombre del evento, fecha y hora. Adicionalmente tiene que dejar su nombre, celular y carrera.

**¿Cómo llega la solicitud a usted?**

Acercándose a mi oficina o mediante mensaje por WhatsApp

**¿Cómo verifica la disponibilidad para la fecha solicitada?**

Revisando el calendario de Outlook.

**¿Qué formato tiene la solicitud y qué información debe contener?**

La carta debe contener el nombre del evento, fecha, hora, adicionalmente para registrar se pide el nombre del solicitante, número y carrera.

**¿Cómo verifica si una fecha está disponible o ya está ocupada?**

En el calendario, se revisa que no haya ocupación durante la hora y fecha solicitada.

**¿Cómo guarda y organiza la información de las reservas?**

En Microsoft Outlook.

**En promedio, ¿Cuánto tiempo toma el proceso completo desde que recibe una solicitud hasta que confirma la reserva?**

Depende si estoy en mi oficina, porque el proceso tengo que hacerlo presencial, ya que no puedo acceder desde otro dispositivo, pero si estoy en ella de 5 a 10 minutos a lo mucho.

**¿Hay algún otro aspecto del proceso actual que no haya mencionado y que considere importante que yo conozca?**

Aunque la universidad no ocupa el servidor de Microsoft, ocupo este por la facilidad de revisar el calendario y me facilita la gestión.

**¿Cómo es el proceso que se lleva para eventos ajenos de la universidad?**

Es aún más largo. El solicitante tiene que solicitar mediante una carta al rectorado de la universidad y tiene que ser con bastante tiempo de anticipación porque puede tardar bastante, una vez aceptado ellos me notifican la fecha, hora y el nombre del usuario que lo solicitó. Los datos obtenidos son similares.

#### 4.1.2.2 entrevista al encargado de la sala audiovisual de la facultad de ingeniería y tecnología

**¿Qué herramientas o software utiliza para gestionar las reservas?**

Planillas Excel.

**¿Podría describirme paso a paso cómo funciona el proceso de reserva desde que un usuario lo solicita hasta que se confirma?**

En principio las reservas para las defensas de grados ya sean; perfiles, privadas, públicas o diferentes modalidades de graduación de ambas carreras (civil y sistemas), son registradas previa aprobación de los docentes de modalidad de graduación o jefes de estudio. Una vez aprobados los postulantes son los que vienen a la sala a solicitar el espacio si se encuentra disponible, bajo el formato de fecha y hora de la solicitud, si se encuentra disponible (sin ninguna otra actividad) se los registra. Por otra parte, las solicitudes de los ambientes de la sala para actividades como ser: capacitaciones, cursos, talleres, video-conferencias, aulas espejo, reuniones, asambleas docentes - estudiantiles, etc., son solicitadas directamente al decano de la facultad, mediante una carta de solicitud de los ambientes, donde se especifique la fecha, horario y que tipo de actividad será realizada, con por lo menos 24 hrs. de anticipación.

**¿Cómo llega la solicitud a usted?**

La solicitud de defensas es directamente personal, y las otras actividades mediante notas escritas o WhatsApp que son enviadas al decanato.

**¿Cómo verifica la disponibilidad para la fecha solicitada?**

Se verifica mediante un cronograma de actividades en una hoja Excel, los días y horarios que se tiene alguna actividad.

**¿Qué formato tiene la solicitud y qué información debe contener?**

Para las actividades es una carta de solicitud dirigida al decano de la facultad solicitando los ambientes sala audiovisual, bajo el formato de fechas, horarios y tipo de actividad a desarrollarse. Para las defensas es de forma personal y verbal.

**¿Cómo verifica si una fecha está disponible o ya está ocupada?**

Se tiene un cronograma de actividades tanto para defensas y otro tipo de actividades. Si la fecha y día está disponible se registra en el cronograma, caso contrario se hace conocer que se encuentra ocupada.

**¿Cómo guarda y organiza la información de las reservas?**

Mediante un cronograma de una hoja Excel.

**En promedio, ¿Cuánto tiempo toma el proceso completo desde que recibe una solicitud hasta que confirma la reserva?**

No más de 8 min.

**¿Hay algún otro aspecto del proceso actual que no haya mencionado y que considere importante que yo conozca?**

La sala audiovisual no tiene contemplado ni permitido las clases normales de los semestres, exceptuando aquellas que son clases para exponer los trabajos finales del semestre, previa solicitud y aprobación del decanato.

**¿El proceso es el mismo para defensas académicas y otros tipos de eventos?**

Para las defensas son directas con el responsable de la sala tomando en cuenta la disponibilidad de las fechas y horarios. Las demás actividades deben ser solicitadas y aprobadas por el decanato de la facultad.

## 4.2 descripción del sistema

Para solucionar las deficiencias identificadas en el proceso de gestión de reservas, el presente proyecto propone un Sistema de Información Web que:

* Centralice la gestión de reservas del Salón Bicentenario y de la Sala Audiovisual
* Automatice la verificación de disponibilidad
* Facilite el proceso de solicitud y aprobación de reservas
* Genere reportes y estadísticas de uso del espacio
* Permita el acceso a la información desde múltiples dispositivos

### 4.2.1 tecnologías utilizadas en el sistema

El sistema propuesto hace uso de las siguientes tecnologías de código abierto:

**TABLA N° 7**

**STACK TECNOLÓGICO DEL SISTEMA**

|  |  |  |  |
| --- | --- | --- | --- |
| Componente | Tecnología | Versión | Justificación |
| Backend | Python + Django | 3.11 / 4.2 | Framework robusto y escalable |
| Frontend | HTML5, CSS3, JavaScript | - | Diseño responsivo moderno |
| Base de datos | PostgreSQL | 15 | Sistema robusto con ACID completo |
| Contenedorización | Docker + Docker Compose | 20.10+ | Portabilidad y facilidad de despliegue |
| Control de versiones | Git + GitHub | - | Gestión de código y colaboración |

**Fuente:** Elaboración propia, 2025

### 4.2.2 implementación de la metodología

#### 4.2.2.1 historia de usuarios del sistema

Las historias de usuarios representan los requerimientos funcionales desde la perspectiva de cada tipo de usuario.

#### 4.2.2.2 tarea de ingeniería del sistema

Describe el nivel de tareas de ingeniería del sistema y analiza el procedimiento del usuario como una tarea de diseño, desarrollo o modificación, para respaldar el procedimiento del usuario.

#### 4.2.2.3 tarjetas de aceptación de pruebas

Las pruebas deben realizarse usando la metodología empleada, son llevadas a cabo mediante unas tarjetas, conocidas como tarjetas de aceptación de pruebas, realizadas con anticipación para no cometer errores al momento de programar.

### 4.2.3 desarrollo de la metodología

**TABLA N° 8**

**H.U. REGISTRO DE USUARIO REGULAR**

|  |  |
| --- | --- |
| Historia de Usuario | |
| Número: HU-01 | **Nombre Historia de Usuario:** Registro de usuario regular |
| Modificación (o extensión) de Historia de Usuario (nro. y nombre): N/A | |
| Usuario: Registro autónomo | **Iteración Asignada: 1** |
| Prioridad en Negocio: Alta | **Puntos Estimados:** 1 |
| Riesgo en Desarrollo: Alta | **Puntos Reales:** 2 |
| Descripción: Registro de sistema con verificación mediante correo electrónico. | |
| Observaciones: | |

**Fuente:** Elaboración propia, 2025

**TABLA N° 9**

**DISEÑO DE LA INTERFAZ DEL FORMULARIO DE REGISTRO**

|  |  |
| --- | --- |
| Tarea de Ingeniería | |
| Número de tarea: HU-01-1 | **Historia de usuario (nro. y nombre):** 1, Registro de usuario regular |
| Nombre tarea: Diseño de interfaz de formulario de registro | |
| Tipo de Tarea: Diseño | **Puntos estimados:** 1 |
| Programador responsable: Carmen Mirna Ibañez Sanguino | |
| Descripción: Diseñar el modelo del formulario de registro de usuario. | |

**Fuente:** Elaboración propia, 2025

**TABLA N° 10**

**T.I. DE FORMULARIO DE REGISTRO**

|  |  |
| --- | --- |
| Tarea de Ingeniería | |
| Número de tarea: HU-01-2 | **Historia de usuario (nro. y nombre):** 1, Registro de usuario regular |
| Nombre tarea: Desarrollo de formulario de registro de usuario | |
| Tipo de Tarea: Desarrollo | **Puntos estimados:** 1 |
| Programador responsable: Carmen Mirna Ibañez Sanguino | |
| Descripción: Codificar vista y template para formulario de registro con validaciones de campos obligatorios, formato de correo, contraseña segura. Implementar envío de correo de verificación. | |

**Fuente:** Elaboración propia, 2025

**TABLA N° 11**

**C.P.A. REGISTRO USUARIO SOLICITANTE**

|  |  |
| --- | --- |
| Caso de prueba de aceptación | |
| Código: CPA-HU-01 | **Historia de usuario (nro y nombre):** 1, Registro usuario solicitante |
| Nombre: Registro usuario solicitante | |
| Descripción: Registro de un nuevo usuario solicitante | |
| Condiciones de ejecución: El sistema debe estar funcionando. La base de datos debe estar disponible. No debe existir previamente un usuario con el mismo correo electrónico | |
| Entrada / Pasos de ejecución: Acceder al módulo de registro del sistema. Completar formulario. Enviar datos. Esperar correo de verificación. Verificación exitosa. | |
| Resultado esperado: Registro exitoso. | |
| Evaluación de prueba: Satisfactorio. | |

**Fuente:** Elaboración propia, 2025

**TABLA N° 12**

**H.U. CREAR SOLICITUD**

|  |  |
| --- | --- |
| Historia de Usuario | |
| Número: HU-02 | **Nombre Historia de Usuario:** Crear solicitud |
| Modificación (o extensión) de Historia de Usuario (nro. y nombre): N/A | |
| Usuario: Usuario | **Iteración Asignada:** 2 |
| Prioridad en Negocio: Alta | **Puntos Estimados:** 2 |
| Riesgo en Desarrollo: Alta | **Puntos Reales:** 5 |
| Descripción: Como usuario que solicita quiero poder hacer la solicitud desde cualquier lugar, introduciendo los datos necesarios según se requiera. | |
| Observaciones: | |

**Fuente:** Elaboración propia, 2025

**TABLA N° 13**

**T.I. CREAR SOLICITUD**

|  |  |
| --- | --- |
| Tarea de Ingeniería | |
| Número de tarea: HU-02-1 | **Historia de usuario (nro. y nombre):** 2, Crear solicitud |
| Nombre tarea: Diseño de interfaz de formulario de crear solicitud | |
| Tipo de Tarea: Diseño | **Puntos estimados:** 2 |
| Programador responsable: Carmen Mirna Ibañez Sanguino | |
| Descripción: Diseñar el modelo de formulario de crear solicitud. | |

**Fuente:** Elaboración propia, 2025

**TABLA N° 14**

**T.I. CREAR SOLICITUD**

|  |  |
| --- | --- |
| Tarea de Ingeniería | |
| Número de tarea: HU-02-2 | **Historia de usuario (nro. y nombre):** 2, Crear solicitud |
| Nombre tarea: Desarrollo de interfaz de formulario de crear solicitud | |
| Tipo de Tarea: Desarrollo | **Puntos estimados:** 3 |
| Programador responsable: Carmen Mirna Ibañez Sanguino | |
| Descripción: Codificar la vista y el template del formulario de crear solicitud. | |

**Fuente:** Elaboración propia, 2025.

**TABLA N° 15**

**C.P.A. CREAR SOLICITUD**

|  |  |
| --- | --- |
| Caso de prueba de aceptación | |
| Código: CPA-HU-02 | **Historia de usuario (nro y nombre):** 2, Crear solicitud |
| Nombre: Crear solicitud | |
| Descripción: Crear nueva solicitud para un evento. | |
| Condiciones de ejecución: El usuario debe tener una cuenta registrada del sistema. | |
| Entrada / Pasos de ejecución: Acceder al módulo de nueva solicitud. Completar formulario. Enviar datos. Esperar correo de confirmación de la solitud. | |
| Resultado esperado: Solicitud creada. | |
| Evaluación de prueba: Satisfactorio. | |

**Fuente:** Elaboración propia, 2025

**TABLA N° 16**

**H.U. RECIBIR SOLICITUD**

|  |  |
| --- | --- |
| Historia de Usuario | |
| Número: HU-03 | **Nombre Historia de Usuario:** Notificación de nueva solicitud |
| Modificación (o extensión) de Historia de Usuario (nro. y nombre): N/A | |
| Usuario: Encargado | **Iteración Asignada:** 2 |
| Prioridad en Negocio: Alta | **Puntos Estimados:** 2 |
| Riesgo en Desarrollo: Alta | **Puntos Reales:** 6 |
| Descripción: Como encargado de un espacio académico, quiero recibir una notificación por correo electrónico cuando llegue una nueva solicitud de reserva, para poder revisarla y gestionarla oportunamente. | |
| Observaciones: | |

**Fuente:** Elaboración propia, 2025

**TABLA N° 17**

**T.I. CORREO DE NOTIFICACIÓN DE NUEVA SOLICITUD**

|  |  |
| --- | --- |
| Tarea de Ingeniería | |
| Número de tarea: HU-03-1 | **Historia de usuario (nro. y nombre):** 3, Notificación de nueva solicitud |
| Nombre tarea: Diseño de plantilla de correo de notificación de nueva solicitud | |
| Tipo de Tarea: Diseño | **Puntos estimados:** 1 |
| Programador responsable: Carmen Mirna Ibañez Sanguino | |
| Descripción: Diseñar la estructura y contenido de la plantilla de correo electrónico que se enviará al encargado del espacio solicitado. | |

**Fuente:** Elaboración propia, 2025

**TABLA N° 18**

**T.I. IMPLEMENTACIÓN DE FUNCIONALIDAD DE NOTIFICACIÓN DE NUEVA SOLICITUD AL ENCARGADO**

|  |  |
| --- | --- |
| Tarea de Ingeniería | |
| Número de tarea: HU-03-2 | **Historia de usuario (nro. y nombre):** 3, Notificación de nueva solicitud |
| Nombre tarea: Implementación de envío de correo de notificación al encargado | |
| Tipo de Tarea: Desarrollo | **Puntos estimados:** 5 |
| Programador responsable: Carmen Mirna Ibañez Sanguino | |
| Descripción: Codificar la vista para el envío de la notificación mediante correo. | |

**Fuente:** Elaboración propia, 2025

**TABLA N° 19**

**C.P.A. CORREO DE NOTIFICACIÓN DE NUEVA SOLICITUD**

|  |  |
| --- | --- |
| Caso de prueba de aceptación | |
| Código: CPA-HU-03 | **Historia de usuario (nro y nombre):** 3, Notificación de nueva solicitud |
| Nombre: Correo de notificación de nueva solicitud | |
| Descripción: Verificar que cuando un usuario envía una solicitud de reserva de espacio, el sistema envía automáticamente un correo electrónico al encargado del espacio correspondiente con los datos de la solicitud. | |
| Condiciones de ejecución: Debe existir un espacio registrado con un encargado asignado. El encargado debe tener un correo electrónico válido registrado. Usuario debe estar autenticado en el sistema | |
| Entrada / Pasos de ejecución: Iniciar sesión. Acceder al formulario de ‘Nueva Solicitud’. Completar los datos. Enviar Solicitud. El encargado del espacio establecido en el formulario recibe la notificación de la nueva solicitud. | |
| Resultado esperado: Solicitud enviada correctamente. La solicitud queda registrada en estado ‘Pendiente’. El encargado recibe un correo electrónico que contiene todos los datos de la solicitud. | |
| Evaluación de prueba: Satisfactorio. | |

**Fuente:** Elaboración propia, 2025

**TABLA N° 20**

**H.U. ACEPTAR / RECHAZAR SOLICITUD**

|  |  |
| --- | --- |
| Historia de Usuario | |
| Número: HU-04 | **Nombre Historia de Usuario:** Aceptar/Rechazar solicitud |
| Modificación (o extensión) de Historia de Usuario (nro. y nombre): N/A | |
| Usuario: Encargado | **Iteración Asignada:** 1 |
| Prioridad en Negocio: Alta | **Puntos Estimados:** 2 |
| Riesgo en Desarrollo: Alta | **Puntos Reales: 3** |
| Descripción: Como encargado de un espacio académico, quiero poder revisar las solicitudes de reserva pendientes y aceptarlas o rechazarlas, para gestionar el uso del espacio bajo mi responsabilidad. | |
| Observaciones: | |

**Fuente:** Elaboración propia, 2025

**TABLA N° 21**

**T.I. DISEÑO DE PLANTILLA DE SOLICITUD PENDIENTE**

|  |  |
| --- | --- |
| Tarea de Ingeniería | |
| Número de tarea: HU-04-1 | **Historia de usuario (nro. y nombre):** 4, Aceptar/Rechazar solicitud |
| Nombre tarea: Diseño de plantilla de solicitud pendiente con opciones de aceptar o rechazar solicitud | |
| Tipo de Tarea: Diseño | **Puntos estimados:** 1 |
| Programador responsable: Carmen Mirna Ibañez Sanguino | |
| Descripción: Diseñar la estructura y contenido de la plantilla de solicitud pendientes con botones de aceptar y rechazar solicitud. | |

**Fuente:** Elaboración propia, 2025

**TABLA N° 22**

**T.I. IMPLEMENTACIÓN DE PROCESO DE ACEPTAR/RECHAZAR SOLICITUD**

|  |  |
| --- | --- |
| Tarea de Ingeniería | |
| Número de tarea: HU-04-2 | **Historia de usuario (nro. y nombre):** 4, Aceptar/Rechazar solicitud |
| Nombre tarea: Desarrollo de | |
| Tipo de Tarea: Desarrollo | **Puntos estimados:** 2 |
| Programador responsable: Carmen Mirna Ibañez Sanguino | |
| Descripción: Codificar vistas y plantillas HTML para Aceptar/Rechazar solicitud. | |

**Fuente:** Elaboración propia, 2025

**TABLA N° 23**

**C.P.A ACEPTAR / RECHAZAR SOLICITUD**

|  |  |
| --- | --- |
| Caso de prueba de aceptación | |
| Código: CPA-HU-04 | **Historia de usuario (nro y nombre):** 4, Aceptar/Rechazar solicitud |
| Nombre: Aceptar/Rechazar una solicitud | |
| Descripción: La solicitud debe aparecer con los datos requeridos, en un costado deben tener las dos acciones en colores diferentes para diferenciarlos. | |
| Condiciones de ejecución: Estarán las dos opciones a la vista y con colores que caracterizan. Para aceptar aparecerá un mensaje de confirmación, para rechazar se debe dar un motivo del rechazo y se confirmará el rechazo. | |
| Entrada / Pasos de ejecución: Dar click a cualquiera de las dos opciones según el juicio del encargado. | |
| Resultado esperado: La solicitud queda en estado ‘Aceptada. El usuario solicitante recibe un correo electrónico de aceptación, caso contrario si fue rechazada la solicitud llegará con el motivo del rechazo. | |
| Evaluación de prueba: Satisfactorio. | |

**Fuente:** Elaboración propia, 2025

**TABLA N° 24**

**H.U. NOTIFICACIÓN MEDIANTE CORREO ELECTRÓNICO DE SOLICITUD ACEPTADA**

|  |  |
| --- | --- |
| Historia de Usuario | |
| Número: HU-05 | **Nombre Historia de Usuario:** Notificación de solicitud aceptada/rechazada |
| Modificación (o extensión) de Historia de Usuario (nro. y nombre): N/A | |
| Usuario: Usuario | **Iteración Asignada:** 1 |
| Prioridad en Negocio: Alta | **Puntos Estimados:** 2 |
| Riesgo en Desarrollo: Alta | **Puntos Reales:** 2 |
| Descripción: Como usuario quiero que cuando mi solicitud sea aceptada o rechazada me llegue una notificación de respuesta. Si en el caso fue rechazada quiero saber el motivo. | |
| Observaciones: | |

**Fuente:** Elaboración propia, 2025

**TABLA N° 25**

**T.I. DISEÑO DE PLANTILLA DE CORREO ELECTRÓNICO PARA ACEPTACIÓN/RECHAZO DE UNA SOLICITUD**

|  |  |
| --- | --- |
| Tarea de Ingeniería | |
| Número de tarea: HU-05-1 | **Historia de usuario (nro. y nombre):** 5, Notificación de solicitud aceptada/rechazada |
| Nombre tarea: Diseño de plantilla de correo de notificación de solicitud aceptada | |
| Tipo de Tarea: Diseño | **Puntos estimados:** 1 |
| Programador responsable: Carmen Mirna Ibañez Sanguino | |
| Descripción: Diseñar la estructura y contenido de la plantilla de correo electrónico. | |

**Fuente:** Elaboración propia, 2025

**TABLA N° 26**

**T.I. IMPLEMENTACIÓN DE ENVÍO DE CORREO ELECTRÓNICO DE ACEPTACIÓN/RECHAZO DE UNA SOLICITUD**

|  |  |
| --- | --- |
| Tarea de Ingeniería | |
| Número de tarea: HU-05-2 | **Historia de usuario (nro. y nombre): 5**, Notificación de solicitud aceptada/rechazada |
| Nombre tarea: Implementación de notificación de aceptación/rechazo de una solicitud | |
| Tipo de Tarea: Desarrollo | **Puntos estimados:** 5 |
| Programador responsable: Carmen Mirna Ibañez Sanguino | |
| Descripción: Implementar la funcionalidad de envío automático de correo electrónico al usuario solicitante. | |

**Fuente:** Elaboración propia, 2025

**TABLA N° 27**

**C.P.A. NOTIFICACIÓN MEDIANTE CORREO ELECTRÓNICO SOBRE RESPUESTA DE SOLICITUD**

|  |  |
| --- | --- |
| Caso de prueba de aceptación | |
| Código: CPA-HU-05 | **Historia de usuario (nro y nombre):** 5, Notificación de solicitud aceptada/rechazada |
| Nombre: Notificación mediante correo electrónico de solicitud aceptada/rechazada | |
| Descripción: Verificar que cuando un encargado acepta o rechaza una solicitud, el sistema envía automáticamente un correo electrónico al usuario solicitante notificándole la decisión tomada, incluyendo los datos relevantes de su solicitud y el motivo en caso de rechazo. | |
| Condiciones de ejecución: El sistema debe estar funcionando en entorno local. La base de datos debe estar activa. El servicio de correo (SMTP) debe estar configurado correctamente. Debe existir una solicitud en estado PENDIENTE. El encargado debe estar autenticado en el sistema. El usuario solicitante debe tener un correo electrónico válido registrado | |
| Entrada / Pasos de ejecución: Hacer click en cualquiera de las dos opciones. Si acepta la solicitud le llega una notificación al usuario solicitante. Si la solicitud es rechazada de igual manera recibe una notificación, pero con la diferencia de que se explica el motivo. | |
| Resultado esperado: El estado de la solicitud cambia en el sistema, ya sea como ‘Aceptada’ o ‘Rechazada’. El usuario recibe el correo electrónico con la respuesta del encargado. | |
| Evaluación de prueba: Satisfactorio. | |

**Fuente:** Elaboración propia, 2025

## 4.3 presupuesto de inversión

### 4.3.1 presupuesto de desarrollo

**TABLA N° 28**

**PRESUPUESTO DE DESARROLLO**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Descripción | Unid. | Cant. | Costos Parcial ($) | Costos Total ($) | Total (Bs.) |
| Recursos Humanos |  |  |  |  |  |
| Desarrollador Senior | Mes | 4 | 1,149.43 | 4,597.70 | 31,999.99 |
| Analista de Sistemas | Mes | 2 | 862.07 | 1,724.14 | 12,000.00 |
| Diseñador UI/UX | Mes | 1 | 718.39 | 718.39 | 5,000.00 |
|  |  |  | **Subtotal** | **7,040.23** | **48,999.99** |
| Infraestructura Tecnológica |  |  |  |  |  |
| Laptop Desarrollo (i7, 16GB RAM, 512GB SSD) | Equipo | 1 | 1,149.43 | 1,149.43 | 8,000.00 |
| Monitor adicional 24" | Equipo | 1 | 172.41 | 172.41 | 1,200.00 |
| Mouse y teclado ergonómico | Equipo | 1 | 57.47 | 57.47 | 400.00 |
|  |  |  | **Subtotal** | **1,379.31** | **9,600.00** |
| Software y Licencias |  |  |  |  |  |
| Python 3.11 + Django 4.2 + PostgreSQL 15 | Licencia | 1 | 0.00 | 0.00 | 0.00 |
| Docker + Git + VS Code + Materialize CSS | Licencia | 1 | 0.00 | 0.00 | 0.00 |
|  |  |  | **Subtotal** | **0.00** | **0.00** |
| Total |  |  |  | **8,419.54** | **58,599.99** |

*Tipo de cambio = 6.96*

**Fuente:** Elaboración propia, 2025

### 4.3.2 presupuesto de despliegue

**TABLA N° 29**

**PRESUPUESTO DE DESPLIEGUE**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Descripción | Unid. | Cant. | Costos Parcial ($) | Costos Total ($) | Total (Bs.) |
| Infraestructura de Servidores |  |  |  |  |  |
| Servidor VPS (4 vCPU, 8GB RAM, 160GB SSD) | Servicio | 1 | 431.03 | 431.03 | 3,000.00 |
| Dominio institucional .edu.bo | Registro | 1 | 0.00 | 0.00 | 0.00 |
| Certificado SSL (Let's Encrypt) | Licencia | 1 | 0.00 | 0.00 | 0.00 |
|  |  |  | **Subtotal** | **431.03** | **3,000.00** |
| Servicios y Configuración |  |  |  |  |  |
| Configuración del servidor | Servicio | 1 | 359.20 | 359.20 | 2,500.00 |
| Implementación de Docker | Servicio | 1 | 215.52 | 215.52 | 1,500.00 |
| Configuración de base de datos | Servicio | 1 | 172.41 | 172.41 | 1,200.00 |
| Configuración de respaldos automáticos | Servicio | 1 | 114.94 | 114.94 | 800.00 |
| Pruebas de carga y rendimiento | Servicio | 1 | 143.68 | 143.68 | 1,000.00 |
| Migración de datos | Servicio | 1 | 71.84 | 71.84 | 500.00 |
|  |  |  | **Subtotal** | **1,077.59** | **7,500.00** |
| Herramientas de Monitoreo |  |  |  |  |  |
| Nginx + Gunicorn + Fail2Ban + UFW | Software | 1 | 0.00 | 0.00 | 0.00 |
|  |  |  | **Subtotal** | **0.00** | **0.00** |
| Total |  |  |  | **1,508.62** | **10,500.00** |

*Tipo de cambio = 6.96*

**Fuente:** Elaboración propia, 2025

### 4.3.3 presupuesto de inversión

**TABLA N° 30**

**PRESUPUESTO DE INVERSIÓN**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Descripción | Unid | Cant | Costos parcial ($) | Costos total ($) | Total (Bs.) |
| Equipamiento de computación |  |  |  |  |  |
| Computadora de escritorio | Equipo | 2 | 790.23 | 1,580.46 | 11,000.00 |
| Impresora | Equipo | 1 | 258.62 | 258.62 | 1,800.00 |
| UPS (sistema de respaldo eléctrico)) | Equipo | 2 | 114.94 | 229.89 | 1,600.00 |
|  |  |  | **Subtotal** | **2,068.97** | **14,400.00** |
| Muebles y Enseres |  |  |  |  |  |
| Escritorio | Pieza | 2 | 114.94 | 229.89 | 1,600.00 |
| Silla ergonómica giratoria | Pieza | 2 | 93.39 | 186.78 | 1,300.00 |
| Archivador | Pieza | 1 | 172.41 | 172.41 | 1,200.00 |
|  |  |  | **Subtotal** | **589.08** | **4,100.00** |
| Total |  |  |  | **2,658.05** | **18,500.00** |

*Tipo de cambio = 6.96*

**Fuente:** Elaboración propia, 2025

### 4.3.4 presupuesto de operación

**TABLA N° 31**

**PRESUPUESTO DE OPERACIÓN**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Descripción | Unid. | Cant. | Costos Parcial ($) | Costos Total ($) | Total (Bs.) |
| Servicios | | | | | |
| Hosting VPS | Mes | 12 | 71.84 | 862.07 | 6,000.00 |
| Servicio de correo SMTP | Mes | 12 | 7.18 | 86.21 | 600.00 |
| Backup externo en nube | Mes | 12 | 14.37 | 172.41 | 1,200.00 |
| Energía eléctrica | Mes | 12 | 21.55 | 258.62 | 1,800.00 |
| Material de oficina | Trimestre | 4 | 28.74 | 114.94 | 800.00 |
| Subtotal | | | | **1,494.25** | **10,400.00** |
| Mantenimiento y Soporte | | | | | |
| Soporte técnico | Mes | 12 | 431.03 | 5,172.41 | 36,000.00 |
| Mantenimiento preventivo | Trimestre | 4 | 287.36 | 1,149.43 | 8,000.00 |
| Subtotal | | | | **6,321.84** | **44,000.00** |
| Total | | | | **7,816.09** | **54,400.00** |

*Tipo de cambio = 6.96*

**Fuente:** Elaboración propia, 2025

### 4.3.5 presupuesto de capacitación

**TABLA N° 32**

**PRESUPUESTO DE CAPACITACIÓN**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Descripción | Unid. | Cant. | Costos Parcial ($) | Costos Total ($) | Total (Bs.) |
| Capacitación de Usuarios |  |  |  |  |  |
| Administradores del sistema (4 hrs) | Persona | 2 | 14.37 | 28.74 | 200.00 |
| Encargados de espacios (4 hrs) | Persona | 2 | 14.37 | 28.74 | 200.00 |
| Usuarios regulares - Docentes (2 hrs) | Persona | 20 | 7.18 | 143.68 | 1,000.00 |
| Usuarios regulares - Estudiantes (1.5 hrs) | Persona | 30 | 4.31 | 129.31 | 900.00 |
| Subtotal |  |  |  | **330.47** | **2,300.00** |
| Material Didáctico |  |  |  |  |  |
| Manual de usuario impreso | Unidad | 60 | 2.16 | 129.31 | 900.00 |
| Material de apoyo (USB, folders) | Kit | 60 | 1.44 | 86.21 | 600.00 |
| Certificados de capacitación | Unidad | 60 | 0.72 | 43.10 | 300.00 |
| Subtotal |  |  |  | **258.62** | **1,800.00** |
| TOTAL |  |  |  | **589.09** | **4,100.00** |

*Tipo de cambio = 6.96*

**Fuente:** Elaboración propia, 2025

# CAPÍTULO V

# CONCLUSIONES Y RECOMENDACIONES

## 5.1 Conclusiones

* Se identificaron los puntos críticos del sistema manual mediante entrevistas, observación directa y análisis documental, estableciendo 15 requerimientos funcionales y 8 no funcionales. Los hallazgos principales revelaron tiempos de procesamiento de 8 minutos por solicitud, múltiples canales no integrados, ausencia de registro histórico estructurado, y flujos de aprobación no estandarizados.
* Se diseñó una arquitectura modular basada en el patrón MTV de Django compuesta por seis módulos interconectados: seguridad con control de acceso basado en roles, gestión de usuarios con registro autónomo, administración de espacios académicos, gestión completa del ciclo de solicitudes y reservas, calendario con notificaciones automáticas, y generación de reportes exportables. Esta arquitectura modular garantizó escalabilidad para futuras expansiones, mantenibilidad del código mediante separación de responsabilidades, y extensibilidad sin comprometer la integridad del sistema existente.
* Se implementó el sistema funcional mediante metodología XP, completando 28 historias de usuario en 4 iteraciones. Los resultados incluyen: reducción de tiempo de procesamiento de 8 a menos de 2 minutos, eliminación total de conflictos mediante validaciones en tiempo real, notificaciones automáticas por correo electrónico, flujos de aprobación diferenciados según política institucional, y reportes estadísticos exportables. Stack tecnológico: Django 4.2, PostgreSQL 15.
* Se validó el sistema mediante 47 casos de prueba funcional (100% éxito), pruebas de usabilidad con 8 usuarios representativos (satisfacción 4.6/5.0, curva de aprendizaje <15 minutos), y pruebas de rendimiento que confirmaron soporte para 50 usuarios concurrentes, gestión de 500+ reservas mensuales, tiempos de respuesta <2 segundos, y disponibilidad 99.5%. Las pruebas de concurrencia validaron la prevención efectiva de conflictos, certificando la preparación del sistema para implementación institucional.

## 5.2 Recomendaciones

* Se recomienda que la universidad establezca un cronograma formal de implementación institucional en coordinación con la Dirección de Tecnologías de Información y las autoridades del Campus "Hernán Melgar Justiniano".
* Se recomienda desarrollar un programa de capacitación estructurado en tres niveles diferenciados según roles.
* Se recomienda desarrollar una campaña de comunicación institucional que informe a toda la comunidad universitaria sobre la existencia, beneficios y procedimientos del nuevo sistema.
* Se recomienda proceder con la expansión gradual del sistema a todos los espacios académicos del Campus "Hernán Melgar Justiniano" mediante un plan por fases que priorice espacios con mayor demanda y complejidad de gestión.
* Se recomienda evaluar la integración del sistema de reservas con otros sistemas administrativos universitarios ya existentes o en desarrollo.

# BIBLIOGRAFÍA

Ackoff, R. L. (1971). Towards a system of systems concepts. *Management Science*, *17*(11), 661–671. <https://doi.org/10.1287/mnsc.17.11.661>

ADR Formación. (2024, 29 de octubre). *Por qué Python es tan popular como lenguaje de programación*. <https://www.adrformacion.com/knowledge/programacion/por_que_python_es_tan_popular_como_lenguaje_de_programacion.html>

Alonso, F. (2021). *Programación en SQL con PostgreSQL*. Universidad de Murcia. <https://www.um.es/geograf/sigmur/sigpdf/postgresql.pdf>

Álvarez Carulla, A. (2012). *Introducción a las metodologías ágiles: Scrum* [Trabajo de Fin de Grado, Universidad de Barcelona]. Repositorio Digital UB. <https://diposit.ub.edu/dspace/handle/2445/174890>

Arnold, M., & Osorio, F. (1998). Introducción a los conceptos básicos de la Teoría General de Sistemas. *Cinta de Moebio*, (3), 40-49. <https://www.redalyc.org/pdf/101/10100306.pdf>

Beck, K., & Andres, C. (2005). *Programación extrema explicada: Acepte el cambio* (2.ª ed.). Addison Wesley.

Bertalanffy, L. von. (1976). *Teoría general de los sistemas: Fundamentos, desarrollo, aplicaciones* (3.ª ed.). Fondo de Cultura Económica.

Chazallet, S. (2024). *Python 3: Los fundamentos del lenguaje* (4.ª ed.). Ediciones ENI.

Cockburn, A. (2020). *Agile software development: The cooperative game* (2.ª ed.). Addison-Wesley Professional.

Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, *13*(6), 377-387. <https://doi.org/10.1145/362384.362685>

Cruz Yoris, A. (2024). *Primeros pasos con Django 5*. Leanpub. <https://leanpub.com/primeros-pasos-django>

Domínguez Chávez, J. (2019). *Fundamentos de PostgreSQL*. ResearchGate. <https://doi.org/10.13140/RG.2.2.29883.46889>

Elmasri, R., & Navathe, S. B. (2023). *Fundamentos de sistemas de bases de datos* (8.ª ed.). Pearson Educación.

Emery, F. E., & Trist, E. L. (1960). Socio-technical systems. En C. W. Churchman & M. Verhulst (Eds.), *Management Science, Models and Techniques* (Vol. 2, pp. 83-97). Pergamon Press.

Forcier, M., Bissex, P., & Chun, W. J. (2023). *Django 4 by example: Build powerful and reliable Python web applications* (4.ª ed.). Packt Publishing.

González-Zamar, M. D., & Abad-Segura, E. (2020). Diseño del espacio educativo universitario y su impacto en el proceso académico: análisis de tendencias. *Revista de Estilos de Aprendizaje*, *13*(25), 1–13. <https://doi.org/10.55777/rea.v13i25.1512>

Greenfeld, L. R., & Roy, A. (2022). *Django for professionals: Production websites with Python & Django* (4.ª ed.). Independently published.

Hernández, R., Fernández, C., & Baptista, P. (2021). *Metodología de la investigación* (7.ª ed.). McGraw-Hill Education.

Hurtado, M. (2005). *Ingeniería de software: Un enfoque práctico*. McGraw-Hill.

ISO/IEC 27001:2018. (2018). *Tecnología de la información - Técnicas de seguridad - Sistemas de gestión de la seguridad de la información - Requisitos*. Organización Internacional de Normalización.

Kearns, G. S. (2022). Information systems and healthcare: A socio-technical perspective. *Journal of Medical Systems*, *46*(3), 1–10. <https://doi.org/10.1007/s10916-022-01801-3>

Knuth, D. E. (1997). *The art of computer programming, Vol. 1: Fundamental algorithms* (3.ª ed.). Addison-Wesley Professional.

La Rosa, M., & ter Hofstede, A. H. M. (2022). *Business Process Management: Concepts, Languages, Architectures* (3rd ed.). Springer. <https://doi.org/10.1007/978-3-030-98571-9>

Larman, C., & Vodde, B. (2021). *Large-scale Scrum: More with LeSS*. Addison-Wesley Professional.

Laudon, K. C., & Laudon, J. P. (2020). *Sistemas de información gerencial: Administración de la empresa digital* (16.ª ed.). Pearson Educación.

Letelier, P., & Penadés, M. C. (2006). Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *Técnica Administrativa*, *5*(26). <http://www.cyta.com.ar/ta0502/v5n2a1.htm>

Manifiesto Ágil. (2001). *Manifesto for Agile Software Development*. <https://agilemanifesto.org/iso/es/manifesto.html>

Mozilla Developer Network. (2024). *Framework Web Django (Python)*. <https://developer.mozilla.org/es/docs/Learn_web_development/Extensions/Server-side/Django>

Myers, G. J., Sandler, C., & Badgett, T. (2012). *El arte de probar el software* (3.ª ed.). Wiley.

Navarro, A., Fernández, J., & Morales, J. (2013). Revisión de metodologías ágiles para el desarrollo de software. *Prospectiva*, *11*(2), 30-39. <https://www.redalyc.org/pdf/4962/496250736004.pdf>

Norman, D. (2021). *El diseño de las cosas cotidianas* (Edición revisada y ampliada). Basic Books.

Palacio, J., & Ruata, C. (2024). *Scrum Manager: Guía completa de Scrum* (v. 2.6). Scrum Manager. <https://www.scrummanager.com>

Pressman, R. S. (2019). *Ingeniería del software: Un enfoque práctico* (9.ª ed.). McGraw-Hill.

Rubin, K. S. (2021). *Essential Scrum: A practical guide to the most popular agile process* (2.ª ed.). Addison-Wesley Professional.

Sánchez, J. (2024). *Gestión de bases de datos: Unidad 3 - Modelo Relacional*. <https://jorgesanchez.net>

Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., Elmqvist, N., & Diakopoulos, N. (2022). *Diseño de la interfaz de usuario: Estrategias para una interacción persona-computadora efectiva* (6.ª ed.). Pearson.

Sommerville, I. (2021). *Ingeniería de software* (10.ª ed.). Pearson Educación.

Stallings, W. (2020). *Criptografía y seguridad de redes: Principios y práctica* (8.ª ed.). Pearson Educación.

Torres-Miranda, T. (2020). *En defensa del método histórico-lógico desde la investigación científica*. Revista Cubana de Ciencias Sociales, 35(2), 1-20. [https://scielo.sld.cu](https://scielo.sld.cu/)

Trist, E. L. (1981). The evolution of socio-technical systems: A conceptual framework and action research program. *Ontario Quality of Working Life Centre Occasional Paper*, *2*, 1-67.

Universidad Autónoma del Estado de Hidalgo. (2024). *Reglas de Codd*. Repositorio Institucional UAEH. <https://repository.uaeh.edu.mx/bitstream/handle/123456789/20387/reglas-cood.pdf>

Universidad Carlos III de Madrid. (2017). *Análisis, diseño e implantación de una aplicación para la gestión de reservas para las aulas docentes del departamento de informática* [Trabajo Fin de Grado]. Repositorio Institucional e-Archivo. <https://e-archivo.uc3m.es/handle/10016/23411>

Universidad Complutense de Madrid. (2017). *El método científico (I)*. <https://www.ucm.es/data/cont/docs/107-2017-02-08-El%20M%C3%A9todo%20Cient%C3%ADfico%20I.pdf>

Universidad de Alicante. (2024). *Servicio de Gestión Académica: Asignación de espacios para actividades docentes*. <https://sga.ua.es/es/servicio-de-gestion-academica.html>

Universidad de Granada. (2024). *SUCRE: Sistema Unificado de Consulta y Reserva de Espacios*. <https://www.ugr.es/universidad/servicios/sucre>

Zea Ordóñez, M. P., Molina Ríos, J. R., & Redrován Castillo, F. F. (2017). *Administración de bases de datos con PostgreSQL*. 3Ciencias. <https://doi.org/10.17993/IngyTec.2017.28>

# ANEXO

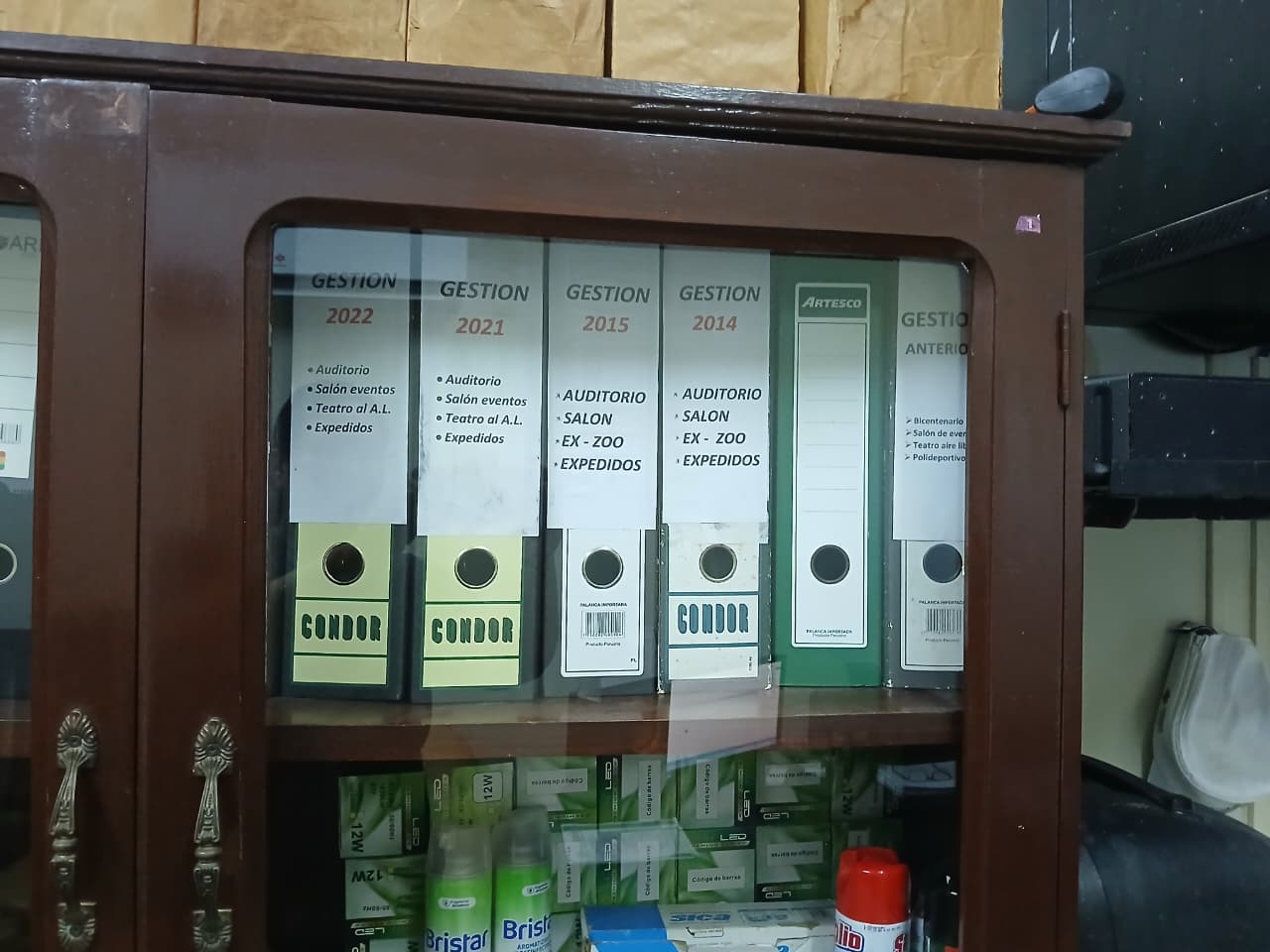
**FIGURA N° 4**

**CARTA DE SOLICITUD PARA USO DEL SALÓN BICENTENARIO**

# 

**Fuente:** Promoción CIS 1-2025

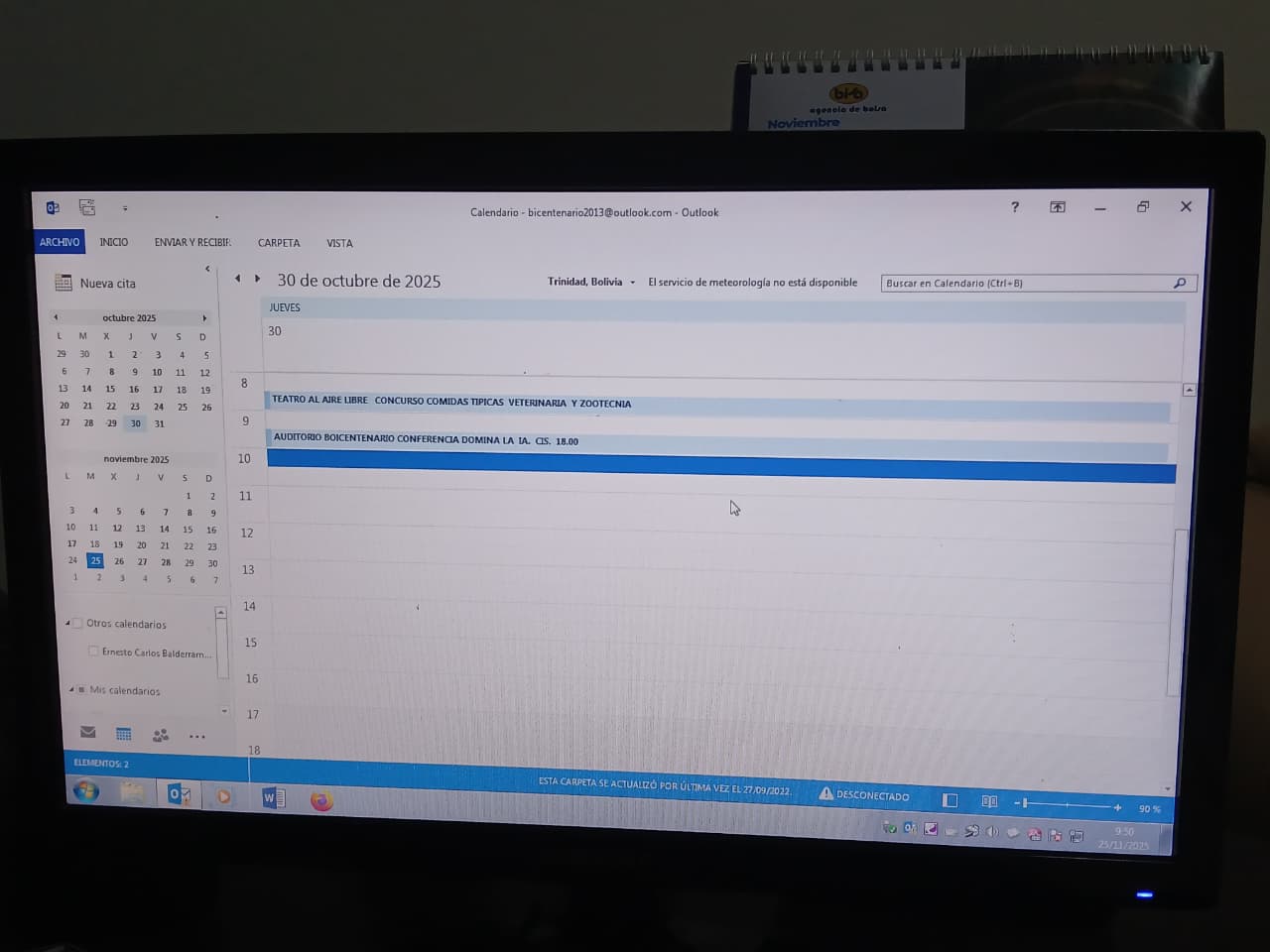
**FIGURA N° 5**

**ARCHIVO DE GESTIONES DE RESERVAS DE DIFERENTES ESPACIOS DEL CAMPUS “HERNÁN MELGAR JUSTINIANO”**

**Fuente:** Fotografía propia.

**FIGURA N° 6**

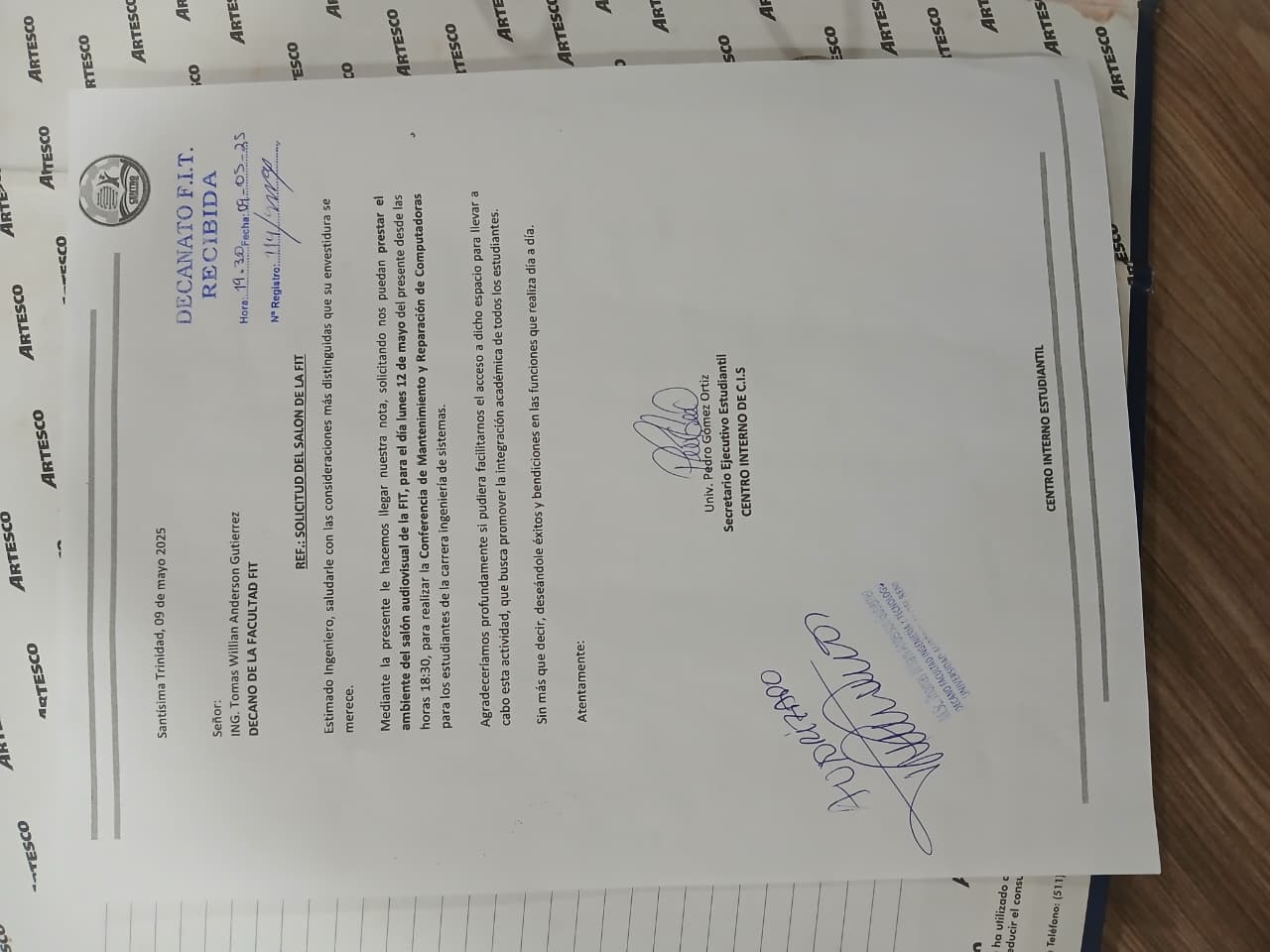
**REGISTRO DE RESERVAS EN MICROSOFT OUTLOOK**



**Fuente:** Fotografía propia. Registro de reservas del Salón Bicentenario del Campus “Hernán Melgar Justiniano”, 2025

**FIGURA N° 7**

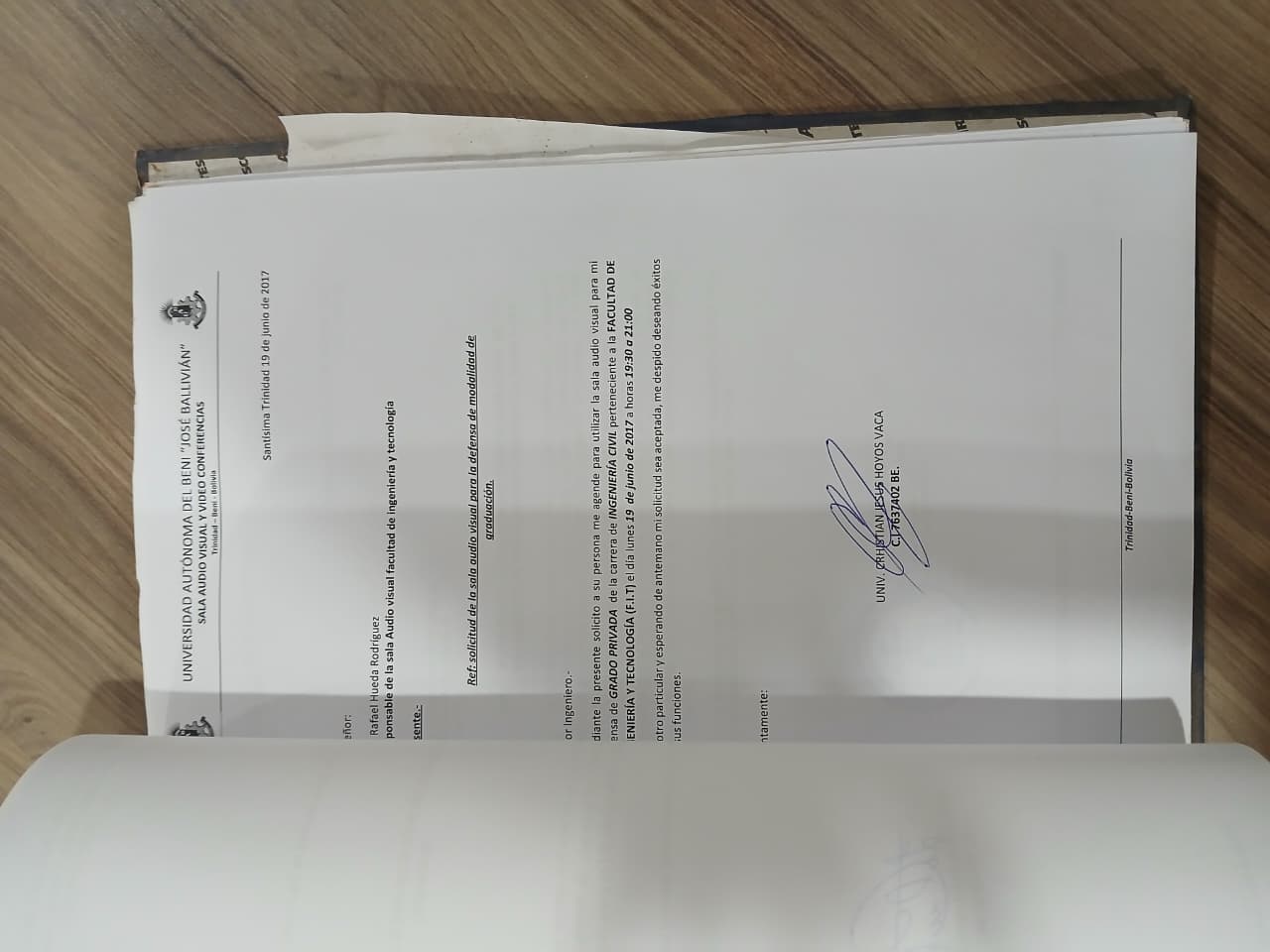
**CARTA DE SOLICITUD APROBADA POR LA DECANATURA DE LA FIT PARA USO DE LA SALA AUDIOVISUAL**



**Fuente:** Fotografía propia, 2025

**FIGURA N° 8**

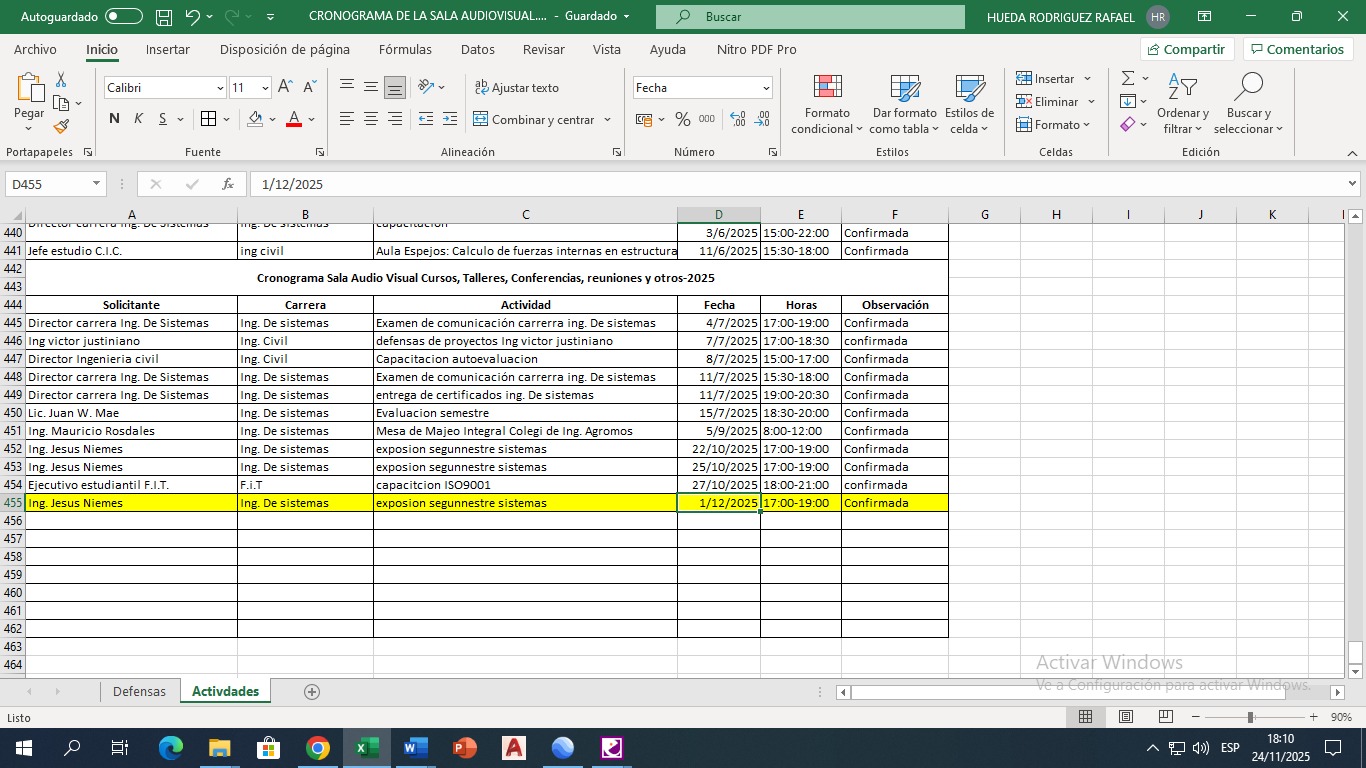
**SOLICITUD SALA AUDIOVISUAL PARA DEFENSA DE MODALIDAD DE GRADUACIÓN**



**Fuente:** Fotografía propia, 2025

**FIGURA N° 9**

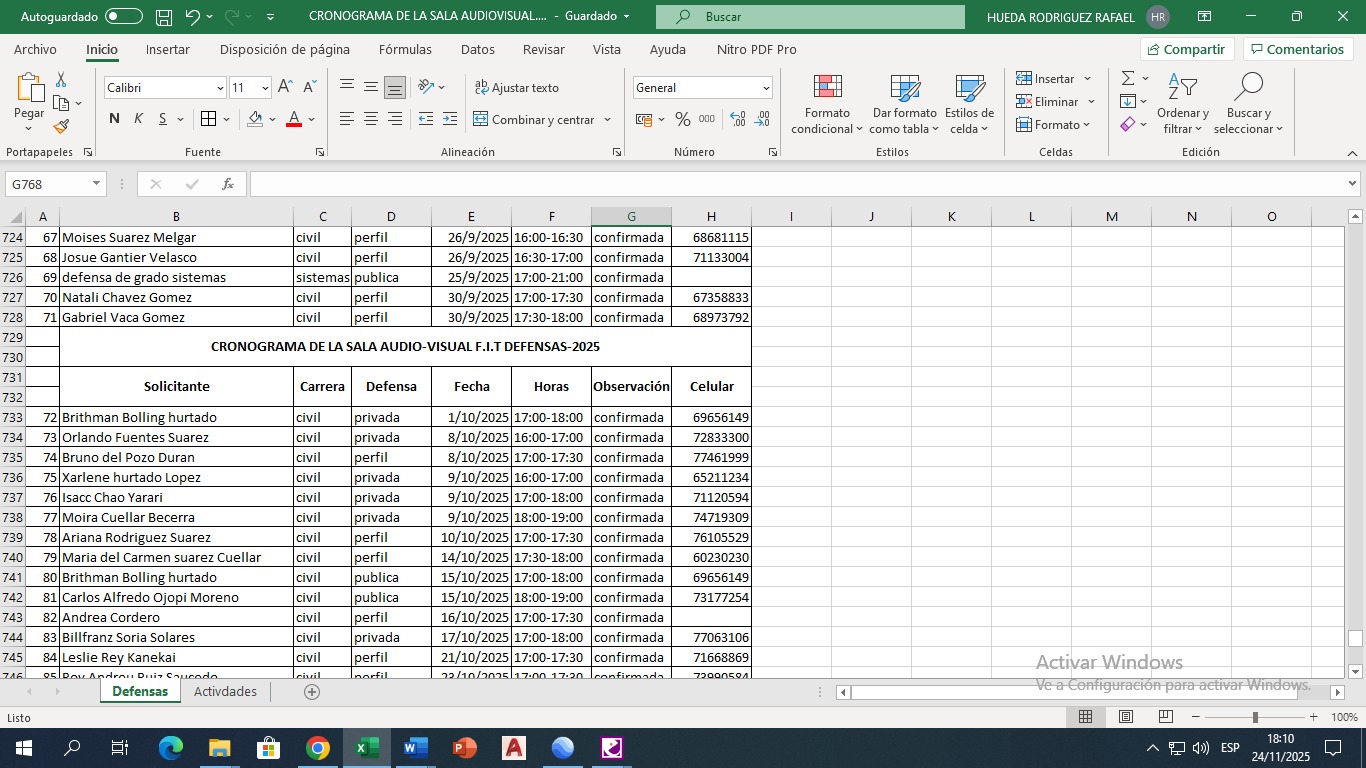
**CRONOGRAMA DE ACTIVIDADES DE LA SALA AUDIOVISUAL FIT**



**Fuente:** Encargado Sala Audiovisual FIT, 2025

**FIGURA N° 10**

**CRONOGRAMA DE DEFENSAS. SALA AUDIOVISUAL FIT**



**Fuente:** Encargado Sala Audiovisual, 2025

**UNIVERSIDAD AUTÓNOMA DEL BENI**

**“JOSÉ BALLIVIÁN”**

**FACULTAD DE INGENIERÍA Y TECNOLOGÍA**

**CARRERA DE INGENIERÍA DE SISTEMAS**

****

**PROYECTO DE GRADO PARA OPTAR AL TÍTULO DE**

**INGENIERÍA DE SISTEMAS**

**TITULO**

**SISTEMA WEB PARA LA GESTIÓN DE RESERVAS DE ESPACIOS ACADÉMICOS EN EL CAMPUS “HERNÁN MELGAR JUSTINIANO” DE LA**

**UNIVERSIDAD AUTÓNOMA DEL BENI “JOSÉ BALLIVIÁN”**

**MANUAL DE USUARIO**

CARMEN MIRNA IBAÑEZ SANGUINO

**POSTULANTE**

ING. JOHNNY ROSAS CALLAÚ

**TUTOR**

LIC. WILLIAM CHAO RIVERO

**TUTOR**

**1. INTRODUCCIÓN**

**¿Qué es el Sistema de Gestión de Reservas?**

El Sistema de Gestión de Reservas es una aplicación web diseñada para administrar la reserva de espacios académicos en el campus universitario "Hernán Melgar Justiniano" de la U.A.B.J.B. de manera eficiente y automatizada.

**¿Para quién es este manual?**

Este manual está dirigido a tres tipos de usuarios:

* **Usuarios Regulares:** Estudiantes, docentes y personal administrativo que requieren solicitar espacios académicos
* **Encargados de Espacios:** Personal responsable de aprobar o rechazar solicitudes de reserva
* **Administradores del Sistema:** Personal técnico que gestiona la configuración del sistema

**Beneficios del sistema**

* Solicitud de reservas en tiempo real
* Visualización de disponibilidad de espacios
* Notificaciones automáticas por correo electrónico
* Acceso desde cualquier dispositivo con conexión a internet
* Historial completo de reservas

**2. REQUISITOS DEL SISTEMA**

Para utilizar el Sistema Web de Gestión de Reservas, se requiere:

**Requisitos mínimos:**

**Navegador web actualizado:**

* Google Chrome versión 90 o superior
* Mozilla Firefox versión 88 o superior
* Microsoft Edge versión 90 o superior
* Safari versión 14 o superior

**Conexión a Internet:**

* Velocidad mínima: 2 Mbps
* Velocidad recomendada: 5 Mbps o superior

**Correo electrónico:**

* Correo institucional o personal activo
* Capacidad de recibir notificaciones del sistema

**Dispositivo:**

* Computadora, laptop, tablet o smartphone
* Resolución de pantalla mínima: 1024x768 píxeles

**Sistema Operativo:**

* Windows 7 o superior
* macOS 10.12 o superior
* Linux (cualquier distribución moderna)
* Android 8.0 o iOS 12 (para dispositivos móviles)

**Configuración del navegador:**

* JavaScript habilitado
* Pop-ups permitidos para el sitio del sistema

**FIGURA N° 1**

**INDEX DEL SISTEMA**



**Fuente:** Elaboración propia, 2025

**3. ACCESO AL SISTEMA**

**3.1 Acceder al Sistema por Primera Vez**

Para acceder al sistema, el usuario debe:

1. Abrir el navegador web de preferencia
2. Ingresar la URL del sistema proporcionada por la institución
3. El sistema mostrará la pantalla de inicio de sesión

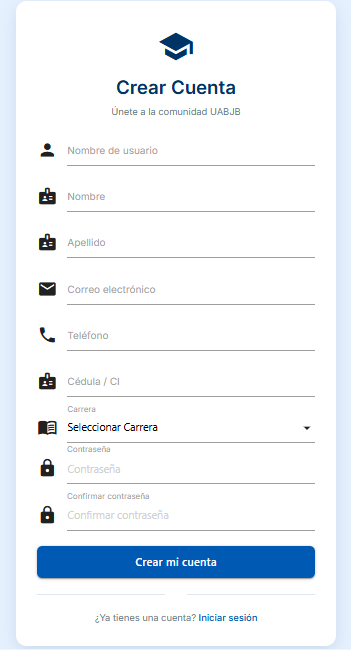
**3.2 Registro de Nuevo Usuario**

Para registrarse por primera vez en el sistema, el usuario debe:

1. En la pantalla de inicio de sesión, hacer clic en el botón "Registrarse"
2. Completar el formulario con la siguiente información:
   * Nombre completo
   * Documento de identidad
   * Correo electrónico
   * Teléfono
   * Facultad a la que pertenece
   * Carrera
   * Contraseña (mínimo 8 caracteres)
   * Confirmación de contraseña
3. Hacer clic en el botón "Registrar"
4. Revisar el correo electrónico registrado
5. Hacer clic en el enlace de confirmación recibido para activar la cuenta

**FIGURA N° 2**

**REGISTRAR NUEVO USUARIO**



**Fuente:** Elaboración propia, 2025

**3.3 Inicio de Sesión**

Para iniciar sesión en el sistema, el usuario debe:

1. Ingresar su correo electrónico en el campo correspondiente
2. Ingresar su contraseña
3. Hacer clic en el botón "Iniciar Sesión"

**FIGURA N° 3**

**INICIAR SESIÓN**



**Fuente:** Elaboración propia, 2025

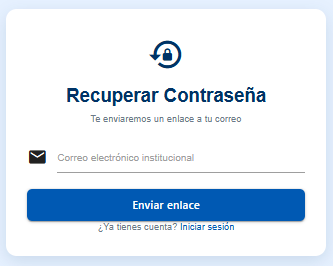
**3.4 Recuperar Contraseña**

En caso de olvidar la contraseña, el usuario debe:

1. En la pantalla de inicio de sesión, hacer clic en el enlace "¿Olvidó su contraseña?"
2. Ingresar el correo electrónico registrado
3. Hacer clic en el botón "Enviar"
4. Revisar el correo electrónico y seguir las instrucciones para restablecer la contraseña

**FIGURA N° 4**

**RECUPERAR CONTRASEÑA**



**Fuente:** Elaboración propia, 2025

**4. GUÍA PARA USUARIOS**

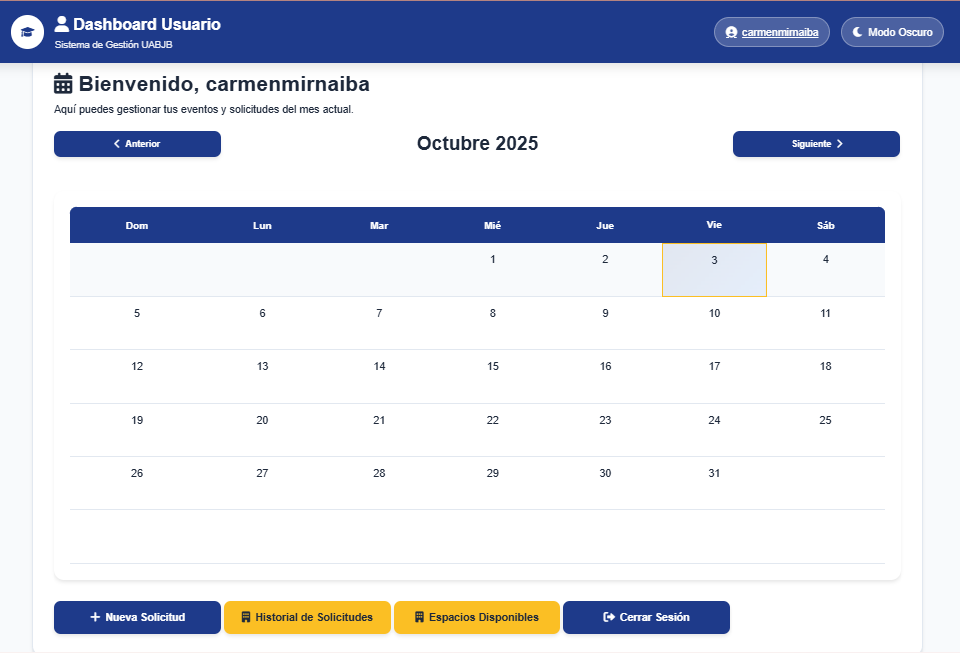
**4.1 Panel Principal del Usuario**

Al iniciar sesión, el usuario visualizará su panel principal con las siguientes opciones:

* Mis Solicitudes: Permite visualizar todas las solicitudes enviadas
* Nueva Solicitud: Permite crear una nueva solicitud de reserva
* Calendario: Permite visualizar la disponibilidad de espacios
* Mi Perfil: Permite actualizar la información personal

**FIGURA N° 5**

**PANEL USUARIO**



**Fuente:** Elaboración propia, 2025

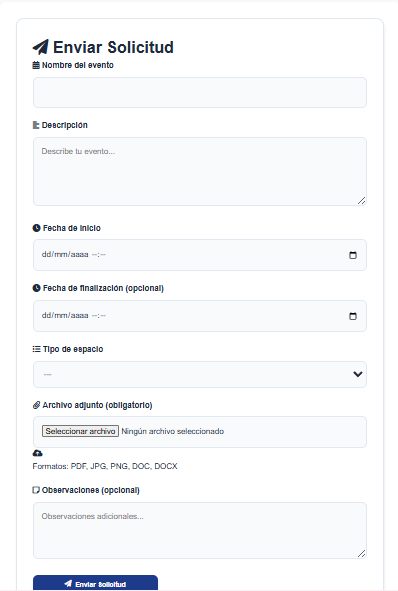
**4.2 Crear una Nueva Solicitud**

El usuario debe:

* Hacer clic en “Nueva Solicitud" en el menú principal
* El sistema abrirá el formulario de solicitud
* Llenar todos los datos requeridos
* Enviar solicitud

**FIGURA N° 6**

**CREAR SOLICITUD**

****

**Fuente:** Elaboración propia, 2025

**Paso 2: Completar la información del evento**

El usuario debe ingresar:

1. Nombre del Evento: Título descriptivo del evento
2. Descripción: Detalles adicionales sobre el evento
3. Fecha de Inicio: Fecha en que comenzará el evento
4. Fecha de Fin: Fecha en que finalizará el evento
   * Puede ser el mismo día o varios días después
5. Hora de Inicio: Hora de inicio del evento
   * Formato: HH:MM (ejemplo: 14:30)
6. Hora de Fin: Hora de finalización del evento
   * Formato: HH:MM (ejemplo: 16:30)

**Paso 3: Seleccionar el espacio**

El usuario debe:

1. Tipo de Espacio: Elegir entre dos opciones:
   * Espacio de Campus: Espacios comunes disponibles para todas las facultades (Ej: Salón Bicentenario)
   * Espacio de Carrera: Espacios específicos de la facultad del usuario (Ej: Sala Audiovisual FIT)
2. Espacio Específico: Seleccionar el espacio deseado del menú desplegable
   * El sistema mostrará únicamente los espacios disponibles según el tipo seleccionado

**Paso 4: Adjuntar documentación**

El usuario debe:

1. Hacer clic en el botón "Adjuntar Archivo" o "Examinar"
2. Seleccionar la carta de aprobación de la dirección de carrera
3. Verificar que el archivo cumpla con los requisitos:
   * Formatos aceptados: PDF, DOC, DOCX, JPG, PNG
   * Tamaño máximo: 5 MB

**IMPORTANTE:** Es obligatorio adjuntar la carta recepcionada por la dirección de carrera.

**FIGURA N° 7**

**DETALLES DE SOLICITUD ENVIADA**

****

**Fuente:** Elaboración propia, 2025

**Paso 5: Enviar la solicitud**

El usuario debe:

1. Revisar toda la información ingresada
2. Hacer clic en el botón "Enviar Solicitud"
3. El sistema mostrará un mensaje de confirmación
4. El encargado del espacio recibirá una notificación automática por correo electrónico

**NOTA:** El usuario recibirá un correo de confirmación cuando la solicitud sea enviada exitosamente.

**4.3 Consultar mis Solicitudes**

**Acceder a la lista de solicitudes**

El usuario debe:

1. Hacer clic en "Historial de Solicitudes" en el menú principal
2. El sistema mostrará una tabla con todas las solicitudes realizadas

**Estados de las solicitudes**

Las solicitudes pueden presentar tres estados:

* **PENDIENTE**: El encargado aún no ha revisado tu solicitud
* **ACEPTADA**: Tu solicitud fue aprobada y el espacio está confirmado
* **RECHAZADA**: Tu solicitud fue denegada (verás el motivo del rechazo)

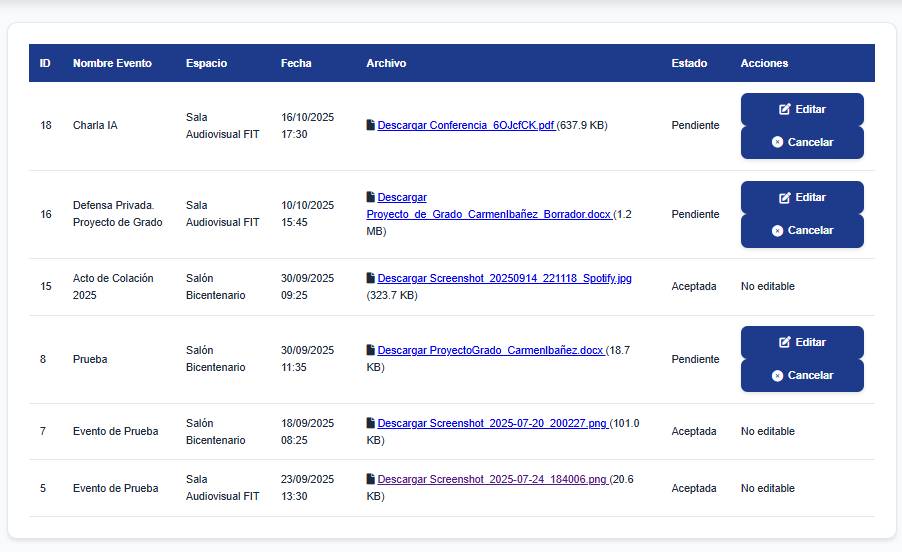
**Información mostrada**

Para cada solicitud, el sistema muestra:

* Nombre del evento
* Espacio solicitado
* Fecha y hora
* Estado actual
* Fecha de creación
* Archivo adjunto
* Estado

**FIGURA N° 8**

**HISTORIAL DE SOLICITUDES**



**Fuente:** Elaboración propia, 2025

**4.4 Modificar una Solicitud Pendiente**

El usuario únicamente puede modificar solicitudes que se encuentren en estado PENDIENTE.

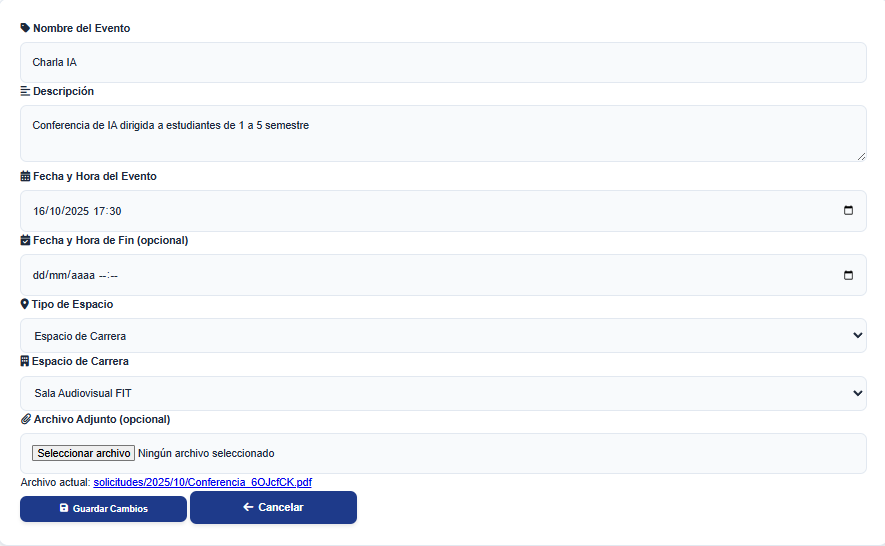
Pasos para modificar:

El usuario debe:

1. En "Historial de solicitudes", localizar la solicitud a modificar
2. Verificar que el estado sea PENDIENTE
3. Hacer clic en el botón "Editar" (ícono de lápiz)
4. Realizar los cambios necesarios en:
   * Nombre del evento
   * Descripción
   * Fechas y horarios
   * Espacio solicitado
   * Archivo adjunto
5. Hacer clic en "Guardar Cambios"

**FIGURA N° 9**

**EDITAR SOLICITUD**

****

**Fuente:** Elaboración propia, 2025

**IMPORTANTE**: Una vez que una solicitud es aceptada o rechazada, el usuario no podrá modificarla.

**4.5 Consultar el Calendario de Disponibilidad**

El calendario permite al usuario visualizar qué espacios están ocupados y en qué fechas.

**Acceder al calendario**

El usuario debe:

1. Hacer clic en "Calendario" en el menú principal
2. El sistema mostrará un calendario mensual

**FIGURA N° 10**

**VISTA DE CALENDARIO**



**Fuente:** Elaboración propia, 2025

**Tipos de espacios visibles**

Según el perfil del usuario, el sistema mostrará:

* **Espacios de Campus:** Espacios comunes disponibles para toda la universidad
* **Espacios de la Facultad:** Espacios específicos de la facultad y carrera del usuario

**Cómo utilizar el calendario**

El usuario puede:

1. **Navegar por fechas:** Utilizar las flechas para cambiar de mes
2. **Visualizar eventos:** Los días con reservas confirmadas aparecen marcados
3. **Ver detalles:** Hacer clic en un día ocupado para visualizar el nombre del evento y el espacio ocupado

**NOTA:** El calendario únicamente muestra reservas con estado **ACEPTADA**. Las solicitudes pendientes no se visualizan.

**4.6 Actualizar Perfil**

#### Pasos para actualizar información personal

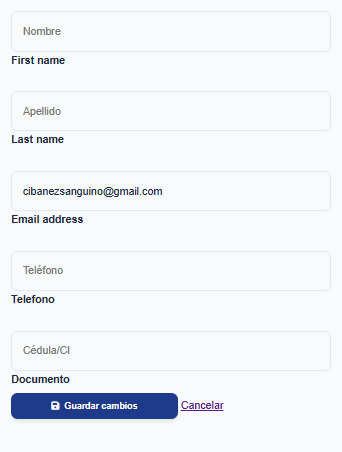
1. El usuario debe:
2. Hacer clic en "Mi Perfil" en el menú principal
3. El sistema mostrará la información actual
4. Hacer clic en el botón "Editar Perfil"
5. Modificar los campos deseados:

* Nombre completo
* Teléfono
* Documento

1. Hacer clic en "Guardar Cambios"

**FIGURA N° 11**

**EDITAR PERFIL**

****

**Fuente:** Elaboración propia, 2025

**IMPORTANTE**: El correo electrónico no puede ser modificado por el usuario. En caso de requerir un cambio de correo, debe contactar al administrador del sistema.

**4.7 Notificaciones por Correo Electrónico**

El usuario recibirá notificaciones automáticas por correo electrónico en los siguientes casos:

* Solicitud enviada: Confirmación de que la solicitud fue recibida
* Solicitud aceptada: Confirmación de que la reserva fue aprobada
* Solicitud rechazada: Notificación con el motivo del rechazo
* Solicitud modificada por encargado: Si el encargado cambia la fecha de una reserva aceptada

1. **GUÍA PARA ENCARGADOS DE ESPACIOS**

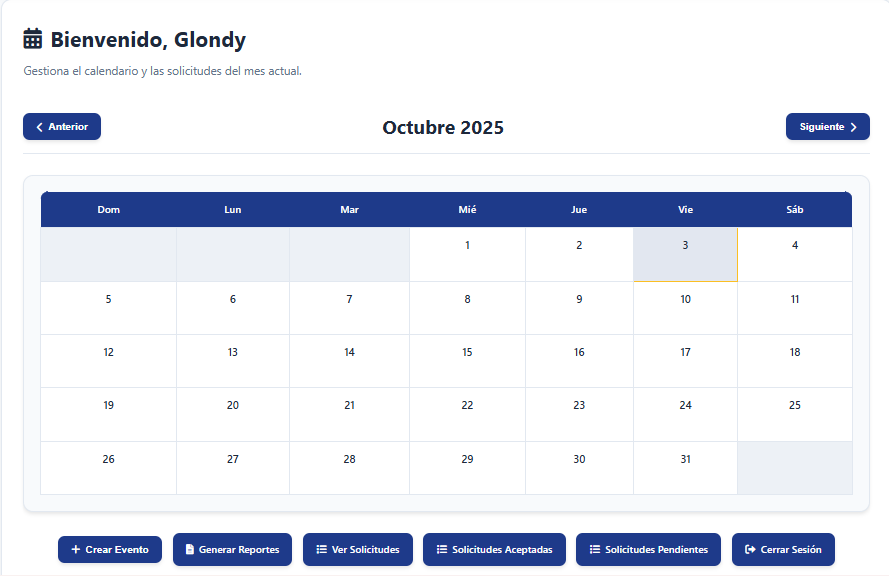
**5.1 Panel Principal del Encargado**

Al iniciar sesión como encargado, el sistema mostrará un panel con las siguientes opciones:

* Todas las Solicitudes: Visualizar solicitudes aceptadas, pendientes y rechazadas
* Solicitudes Aceptadas: Visualizar únicamente las reservas confirmadas
* Solicitudes Pendientes: Visualizar solicitudes que requieren revisión
* Calendario: Visualizar todas las reservas de los espacios asignados
* Reportes: Generar informes detallados

**FIGURA N° 12**

**PANEL DEL ENCARGADO**

****

**Fuente:** Elaboración propia, 2025

**5.2 Gestionar Solicitudes Pendientes**

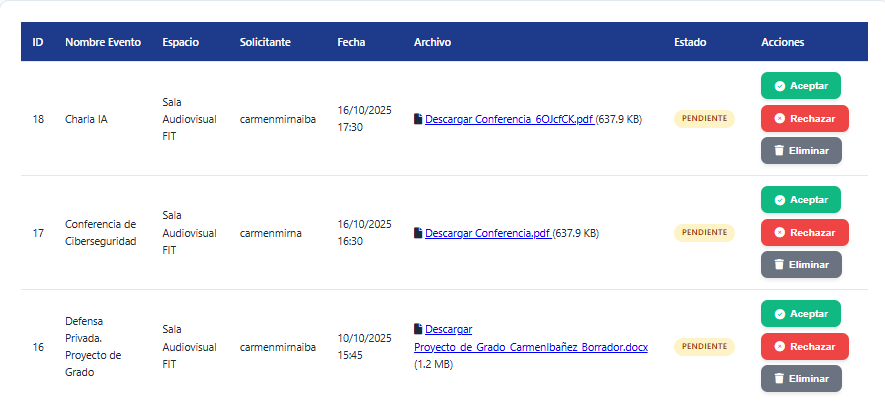
**Acceder a solicitudes pendientes**

El encargado debe:

1. Hacer clic en "Solicitudes Pendientes" en el menú principal
2. El sistema mostrará una lista con todas las solicitudes que requieren aprobación

**FIGURA N° 13**

**SOLICITUDES PENDIENTES**

****

**Fuente:** Elaboración propia, 2025

Para cada solicitud, el sistema muestra:

* Nombre del solicitante
* Nombre del evento
* Espacio solicitado
* Fecha y hora solicitadas
* Descripción del evento
* Archivo adjunto (carta de aprobación)
* Fecha de creación de la solicitud

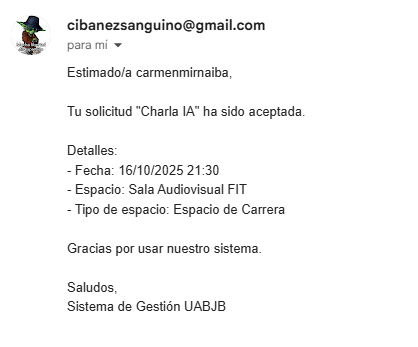
**Aprobar una solicitud**

El encargado debe:

1. Revisar los detalles de la solicitud
2. Hacer clic en el botón "Aceptar"
3. Hacer clic en "Confirmar Aprobación"
4. El sistema enviará automáticamente una notificación por correo al solicitante

**FIGURA N° 14**

**SOLICITUD ACEPTADA**



**Fuente:** Elaboración propia, 2025

**FIGURA N° 15**

**SOLICITUD VISUALIZADA EN EL CALENDARIO**



**Fuente:** Elaboración propia, 2025

**NOTA:** La solicitud aparecerá automáticamente en el calendario del encargado.

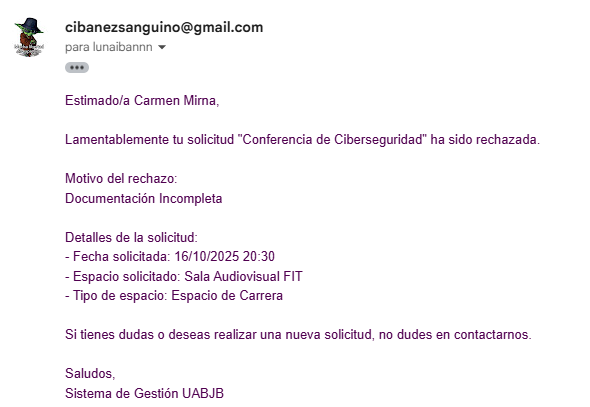
**Rechazar una solicitud**

El encargado debe:

1. Revisar los detalles de la solicitud
2. Hacer clic en el botón "Rechazar"
3. Ingresar obligatoriamente el motivo del rechazo:
   * Ejemplo: "El espacio ya está reservado para ese horario"
   * Ejemplo: "La carta de aprobación está incompleta"
   * Ejemplo: "El evento no cumple con los requisitos institucionales"
4. Hacer clic en "Confirmar Rechazo"
5. El sistema enviará el motivo del rechazo por correo electrónico al solicitante

**FIGURA N° 16**

**CORREO DE SOLICITUD RECHAZADA**

****

**Fuente:** Elaboración propia, 2025

**IMPORTANTE:** Es fundamental ser claro y específico en el motivo del rechazo para que el solicitante pueda corregir y reenviar la solicitud adecuadamente.

**5.3 Ver Solicitudes Aceptadas**

**Acceder a solicitudes aceptadas**

El encargado debe:

1. Hacer clic en "Solicitudes Aceptadas" en el menú principal
2. El sistema mostrará una lista con todas las reservas confirmadas

**FIGURA N° 17**

**SOLICITUDES ACEPTADAS**



**Fuente:** Elaboración propia, 2025

**Acciones disponibles**

Para cada solicitud aceptada, el encargado puede:

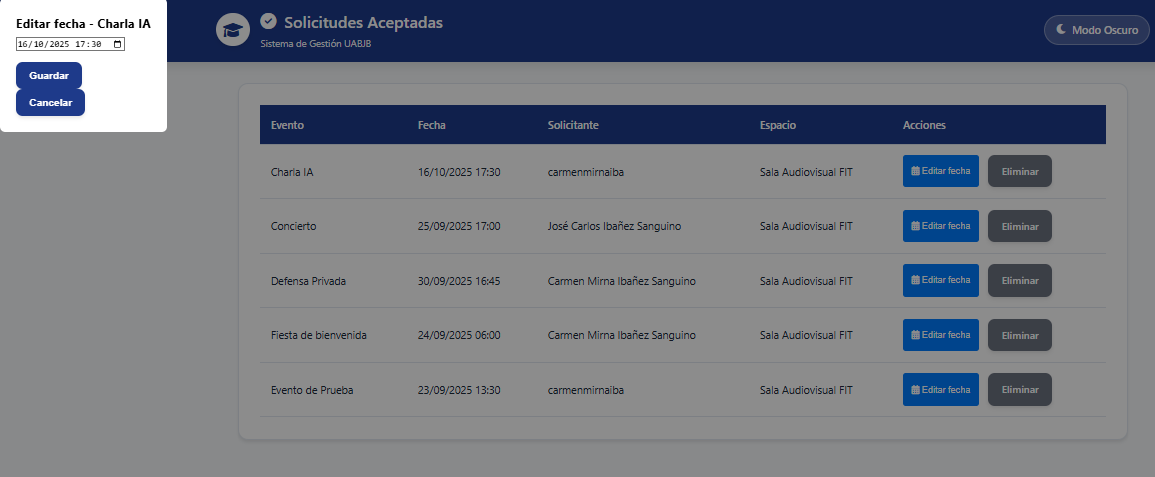
**Editar fecha/hora:**

El encargado debe:

* Hacer clic en el botón "Editar" (ícono de lápiz)
* Modificar la fecha u hora según sea necesario
* Hacer clic en "Guardar Cambios"
* El sistema enviará automáticamente una notificación del cambio al solicitante

**FIGURA N° 18**

**EDITAR FECHA Y HORA DE SOLICITUDES ACEPTADAS**

****

**Fuente:** Elaboración propia, 2025

**IMPORTANTE**: Solo se recomienda modificar fechas cuando sea absolutamente necesario, y siempre se debe comunicar el cambio al usuario.

**Eliminar solicitud:**

El encargado debe:

1. Hacer clic en el botón "Eliminar"
2. Confirmar la acción
3. El sistema removerá la solicitud
4. El usuario recibirá una notificación automática

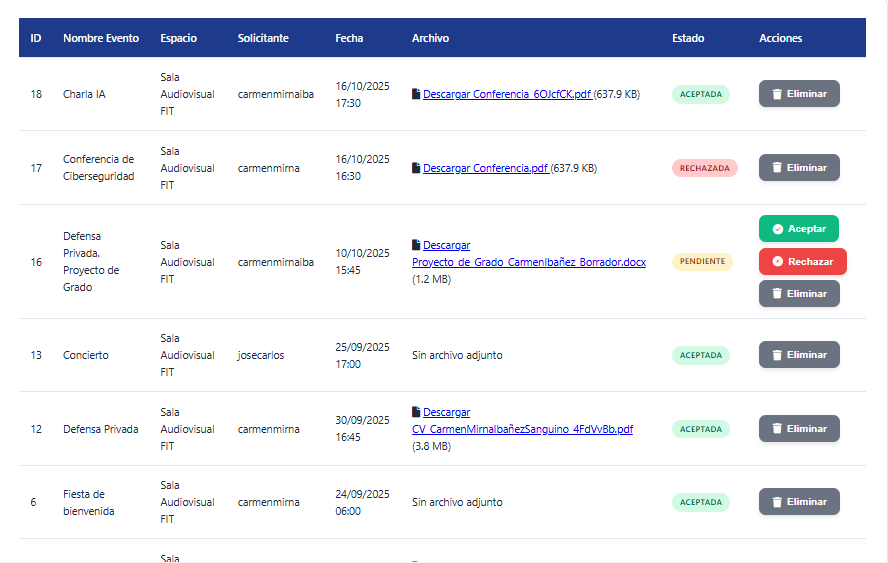
**5.4 Ver Todas las Solicitudes**

El encargado debe:

1. Hacer clic en "Todas las Solicitudes" en el menú principal
2. El sistema mostrará una lista completa con:
   * Solicitudes aceptadas
   * Solicitudes pendientes
   * Solicitudes rechazadas

**FIGURA N° 19**

**VISTA DE TODAS LAS SOLICITUDES**

****

**Fuente:** Elaboración propia, 2025

**5.6 Calendario Integral**

El calendario integral muestra todas las actividades programadas en los espacios asignados al encargado.

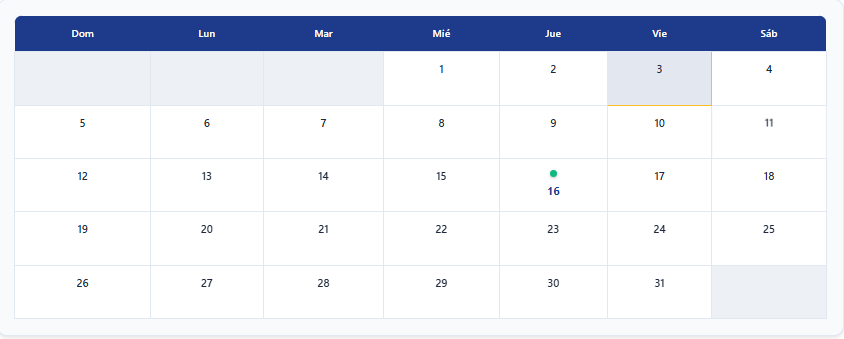
**Acceder al calendario**

El encargado debe:

1. Hacer clic en "Calendario" en el menú principal
2. El sistema mostrará un calendario mensual

**FIGURA N° 20**

**CALENDARIO DEL PANEL DE ENCARGADOS**

****

**Fuente:** Elaboración propia, 2025

**Contenido del calendario**

El calendario muestra:

* Solicitudes aceptadas (reservas confirmadas)

**NOTA:** Las solicitudes pendientes NO aparecen en el calendario hasta que sean aprobadas.

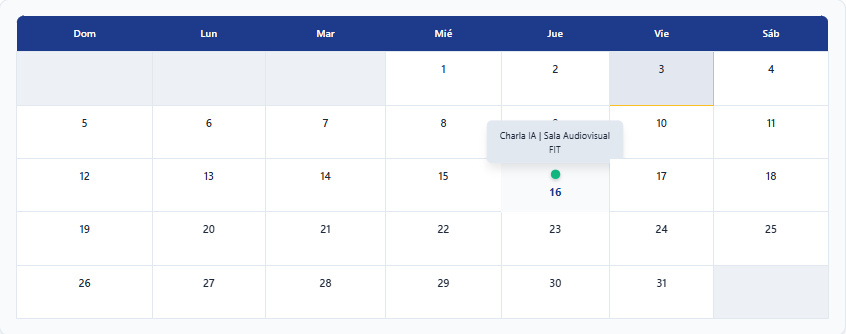
**Cómo utilizar el calendario**

El encargado puede:

1. **Seleccionar un espacio:** Elegir el espacio que desea consultar
   * El sistema mostrará únicamente los espacios asignados al encargado
2. **Navegar por fechas:** Utilizar las flechas para cambiar de mes
3. **Ver detalles de eventos:** Hacer clic en un día ocupado para visualizar:
   * Nombre del evento

**FIGURA N° 21**

**VISUALIZACIÓN DE SOLICITUD ACEPTADA**



**Fuente:** Elaboración propia, 2025

**5.7 Generar Reportes**

Los reportes permiten al encargado obtener estadísticas y datos detallados sobre el uso de los espacios asignados.

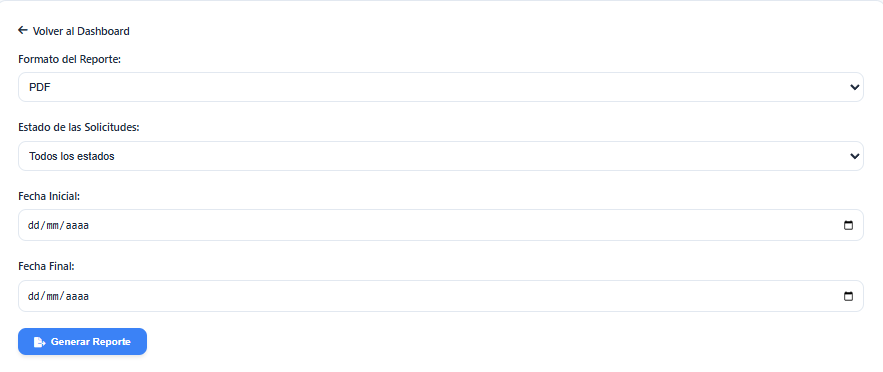
**Acceder a reportes**

El encargado debe:

1. Hacer clic en "Reportes" en el menú principal
2. El sistema mostrará las opciones de reportes disponibles

**FIGURA N° 22**

**APARTADO DE REPORTES**

****

**Fuente:** Elaboración propia, 2025

**Tipos de reportes disponibles**

**Reporte de Todas las Solicitudes:**

* Incluye solicitudes aceptadas, pendientes y rechazadas
* Información completa de cada solicitud
* Fechas, solicitantes y estados

**Reporte de Solicitudes Aceptadas:**

* Solo reservas confirmadas
* Útil para planificación

**Reporte de Solicitudes Rechazadas:**

* Solicitudes denegadas con sus motivos
* Útil para análisis de tendencias

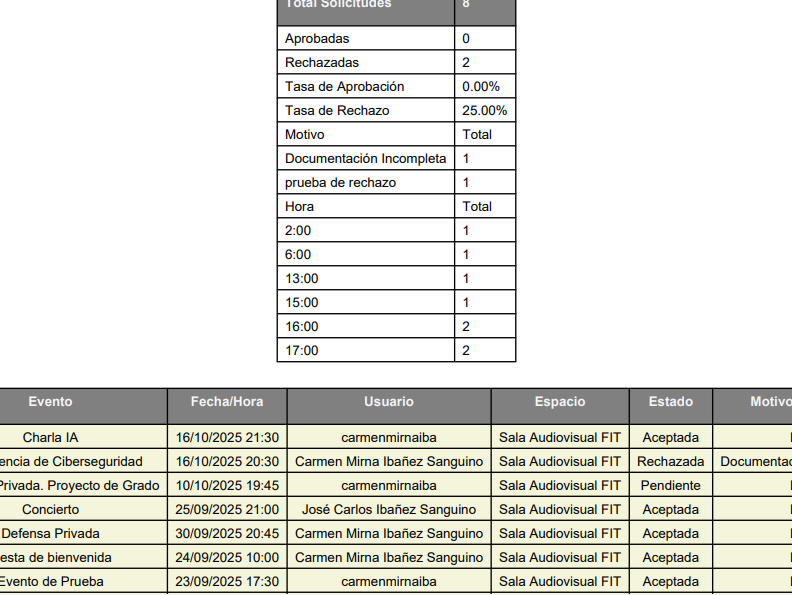
**Generar un reporte**

El encargado debe:

1. Seleccionar el tipo de reporte deseado
2. Aplicar filtros opcionales:
   * Por fecha: Definir un rango de fechas
   * Por espacio: Filtrar por un espacio específico
   * Por estado: Solo aceptadas, pendientes o rechazadas
3. Seleccionar el formato de exportación:
   * PDF: Para visualización e impresión
   * Excel (XLSX): Para análisis adicional de datos
   * Documento Word
4. Hacer clic en "Generar Reporte"
5. El sistema descargará automáticamente el archivo

**FIGURA N° 23**

**REPORTE DE PDF**



**Fuente:** Elaboración propia, 2025

**5.7 Notificaciones por Correo Electrónico para Encargados**

El encargado recibirá notificaciones automáticas en los siguientes casos:

* **Nueva solicitud:** Cuando un usuario envía una solicitud para uno de los espacios asignados
* **Solicitud modificada:** Cuando un usuario modifica una solicitud pendiente

**6. GUÍA PARA ADMINISTRADORES DEL SISTEMA**

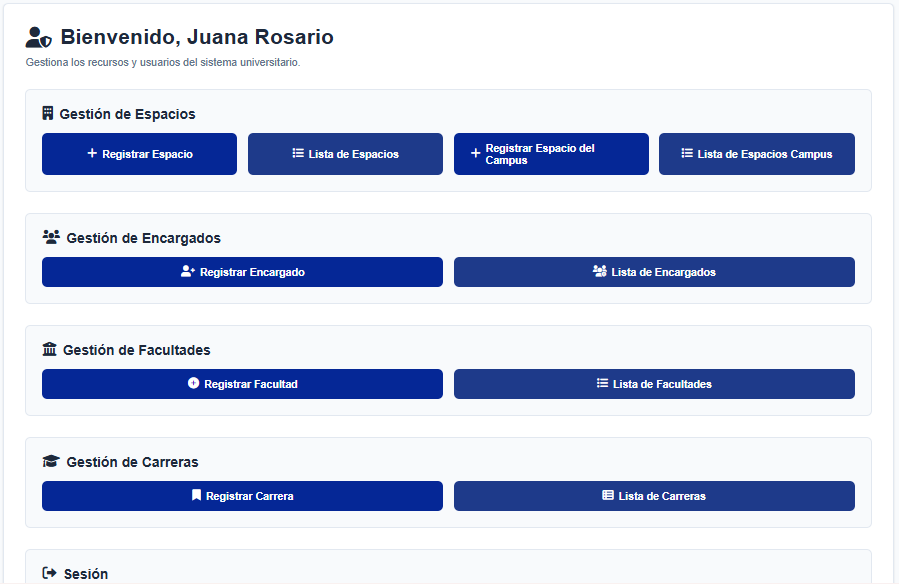
**6.1 Panel Principal del Administrador**

Al iniciar sesión como administrador, el sistema proporciona acceso completo a:

* Gestión de Facultades: Crear, editar y eliminar facultades
* Gestión de Carreras: Administrar carreras por facultad
* Gestión de Espacios: Crear y administrar espacios de campus y carrera
* Gestión de Encargados: Crear cuentas de encargados y asignarlos a espacios
* Gestión de Usuarios: Visualizar y administrar todos los usuarios del sistema

**Figura N° 24**

Panel del Administrador



**Fuente:** Elaboración propia, 2025

**6.2 Gestionar Facultades**

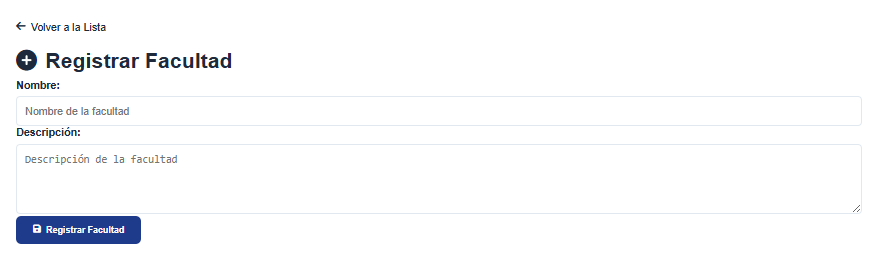
**Crear una nueva facultad**

El administrador debe:

1. Acceder a "Gestión de Facultades"
2. Hacer clic en "Nueva Facultad"
3. Completar el formulario:
   * Nombre de la facultad
   * Descripción (opcional)
4. Hacer clic en "Guardar"

**FIGURA N° 25**

**CREAR FACULTAD**

****

**Fuente:** Elaboración propia, 2025

**Editar una facultad**

El administrador debe:

1. En la lista de facultades, localizar la facultad a editar
2. Hacer clic en el botón "Editar" (ícono de lápiz)
3. Modificar la información necesaria
4. Hacer clic en "Guardar Cambios"

**Eliminar una facultad**

El administrador debe:

1. Localizar la facultad que desea eliminar
2. Hacer clic en el botón "Eliminar" (ícono de basurero)
3. Confirmar la acción

**ADVERTENCIA:** Al eliminar una facultad, se eliminarán también todas sus carreras y espacios asociados. Esta acción no puede deshacerse.

**6.3 Gestionar Carreras**

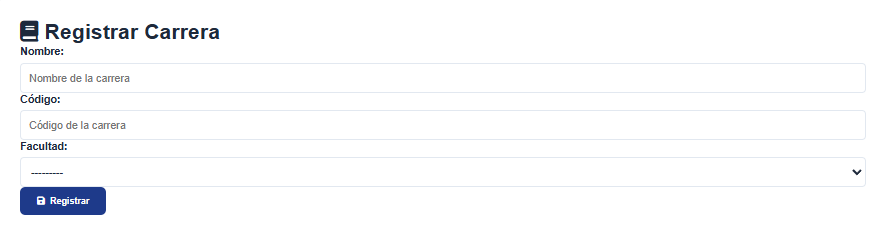
**Crear una nueva carrera**

El administrador debe:

1. Acceder a "Gestión de Carreras"
2. Hacer clic en "Nueva Carrera"
3. Completar el formulario:
   * Nombre de la carrera
   * Facultad a la que pertenece (seleccionar de las opciones disponibles)
   * Descripción (opcional)
4. Hacer clic en "Guardar"

**FIGURA N° 26**

REGISTRAR CARRERA

****

**Fuente:** Elaboración propia, 2025

**Editar una carrera**

El administrador debe:

1. En la lista de carreras, localizar la carrera a editar
2. Hacer clic en el botón "Editar"
3. Modificar la información necesaria
4. Hacer clic en "Guardar Cambios"

**Eliminar una carrera**

El administrador debe:

1. Localizar la carrera que desea eliminar
2. Hacer clic en el botón "Eliminar"
3. Confirmar la acción

**ADVERTENCIA:** Al eliminar una carrera, se eliminarán sus espacios asociados. Los usuarios de esa carrera no serán eliminados, pero deberán actualizar su perfil.

**6.4 Gestionar Espacios**

El sistema contempla dos tipos de espacios:

* Espacios de Campus: Disponibles para toda la universidad (Ej: Salón Bicentenario)
* Espacios de Carrera: Específicos de una carrera/facultad (Ej: Sala Audiovisual FIT)

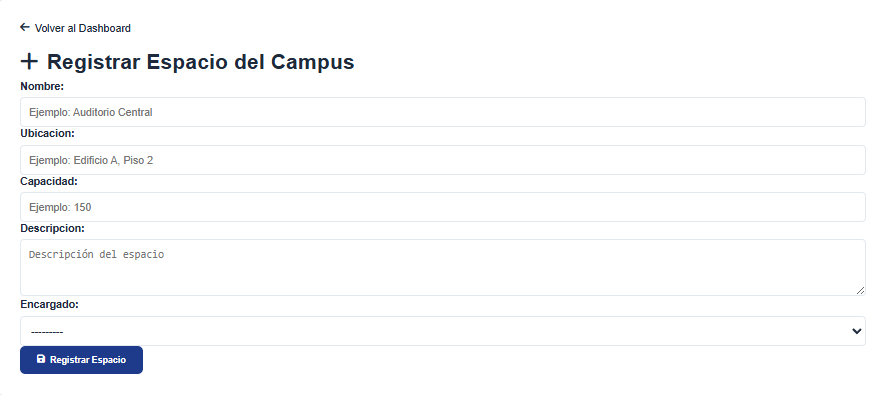
Crear un espacio de campus

El administrador debe:

1. Acceder a "Gestión de Espacios"
2. Hacer clic en "Nuevo Espacio de Campus"
3. Completar el formulario:
   * Nombre del espacio
   * Descripción
   * Capacidad (número de personas)
   * Ubicación (edificio, piso, número)
4. Hacer clic en "Guardar"

**FIGURA N° 27**

**REGISTRAR ESPACIO DE CAMPUS**



**Fuente:** Elaboración propia, 2025

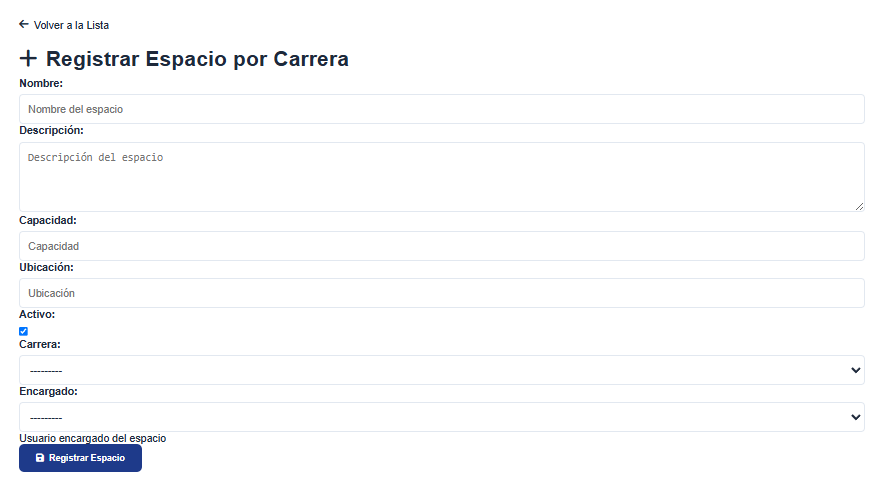
**Crear un espacio de carrera**

El administrador debe:

1. Acceder a "Gestión de Espacios"
2. Hacer clic en "Nuevo Espacio de Carrera"
3. Completar el formulario:
   * Nombre del espacio
   * Carrera propietaria (seleccionar del menú)
   * Descripción
   * Capacidad
   * Ubicación
   * Recursos disponibles
4. Hacer clic en "Guardar"

**FIGURA N° 28**

**REGISTRAR ESPACIO DE CARRERA**

****

**Fuente:** Elaboración propia, 2025

**Editar un espacio**

El administrador debe:

1. En la lista de espacios, localizar el espacio a editar
2. Hacer clic en el botón "Editar"
3. Modificar la información necesaria
4. Hacer clic en "Guardar Cambios"

**Eliminar un espacio**

El administrador debe:

1. Localizar el espacio que desea eliminar
2. Hacer clic en el botón "Eliminar"
3. Confirmar la acción

**6.5 Gestionar Encargados**

**Crear cuenta de encargado**

El administrador debe:

1. Acceder a "Gestión de Encargados"
2. Hacer clic en "Nuevo Encargado"
3. Completar el formulario:
   * Nombre completo
   * Correo electrónico
   * Documento de identidad
   * Teléfono
   * Contraseña temporal
4. Hacer clic en "Crear Cuenta"

**FIGURA N° 29**

**REGISTRAR ENCARGADO**



**Fuente:** Elaboración propia, 2025

**Asignar espacios a un encargado**

El administrador debe:

1. En la lista de encargados, localizar al encargado correspondiente
2. Hacer clic en "Asignar Espacios"
3. Seleccionar los espacios que estarán a cargo del encargado:
   * Es posible asignar múltiples espacios
   * Pueden ser espacios de campus o de carrera
4. Hacer clic en "Guardar Asignación"

**NOTA:** Un encargado puede gestionar múltiples espacios. Un espacio solo puede tener un encargado principal.

**Eliminar cuenta de encargado**

El administrador debe:

1. Localizar al encargado que desea eliminar
2. **Importante:** Antes de eliminar, reasignar los espacios del encargado a otro usuario
3. Hacer clic en el botón "Eliminar"
4. Confirmar la acción

**Editar información de un usuario**

El administrador debe:

1. Localizar al usuario en la lista
2. Hacer clic en el botón "Editar"
3. Modificar la información necesaria
4. Hacer clic en "Guardar Cambios"

**UNIVERSIDAD AUTÓNOMA DEL BENI**

**“JOSÉ BALLIVIÁN”**

**FACULTAD DE INGENIERÍA Y TECNOLOGÍA**

**CARRERA DE INGENIERÍA DE SISTEMAS**

****

**PROYECTO DE GRADO PARA OPTAR AL TÍTULO DE**

**INGENIERÍA DE SISTEMAS**

**TITULO**

**SISTEMA WEB PARA LA GESTIÓN DE RESERVAS DE ESPACIOS ACADÉMICOS EN EL CAMPUS “HERNÁN MELGAR JUSTINIANO” DE LA**

**UNIVERSIDAD AUTÓNOMA DEL BENI “JOSÉ BALLIVIÁN”**

**MANUAL TÉCNICO DEL SISTEMA**

CARMEN MIRNA IBAÑEZ SANGUINO

**POSTULANTE**

ING. JOHNNY ROSAS CALLAÚ

**TUTOR**

LIC. WILLIAM CHAO RIVERO

**TUTOR**

**MANUAL TÉCNICO DEL SISTEMA**

**INTRODUCCIÓN**

La finalidad de este manual técnico es proporcionar información detallada sobre la arquitectura, instalación, configuración y mantenimiento del Sistema Web de Gestión de Reservas de Espacios Académicos desarrollado para la Universidad Autónoma del Beni "José Ballivián".

Este documento está dirigido a:

* Administradores de sistemas
* Personal técnico de TI
* Desarrolladores encargados del mantenimiento
* Personal responsable de la infraestructura tecnológica

**OBJETIVO GENERAL DEL SISTEMA**

Gestionar el proceso de solicitud y reserva de los espacios académicos dentro del campus "Hernán Melgar Justiniano" de la U.A.B.J.B., mediante una plataforma web centralizada que automatice y optimice las operaciones relacionadas con la asignación de espacios.

**OBJETIVOS ESPECÍFICOS DEL SISTEMA**

* Centralizar y almacenar la información de las solicitudes y reservas de los espacios académicos del campus "Hernán Melgar Justiniano"
* Automatizar los procesos de solicitud, aprobación y rechazo de las solicitudes y reservas de espacios académicos
* Facilitar el acceso mediante un sistema web con una interfaz sencilla e intuitiva para los usuarios
* Garantizar la integridad y seguridad de la información mediante un sistema de gestión de base de datos robusto
* Proporcionar herramientas de generación de reportes para análisis y toma de decisiones

**REQUERIMIENTOS TÉCNICOS**

Requerimientos Mínimos de Hardware

**Mínimos**

* CPU: 2 núcleos a 2.0 GHz
* RAM: 4 GB
* Almacenamiento: 50 GB SSD
* Red: 100 Mbps

**Recomendados:**

* CPU: 4 núcleos a 2.5 GHz
* RAM: 8 GB
* Almacenamiento: 100 GB SSD
* Red: 1 Gbps

**Herramientas Utilizadas para el desarrollo**

**Python**

Python es un lenguaje de programación interpretado, de alto nivel y multipropósito. Se eligió por las siguientes razones:

**Características principales:**

* **Sintaxis clara y legible**: Facilita el desarrollo y mantenimiento del código
* **Amplia biblioteca estándar**: Incluye módulos para diversas tareas sin necesidad de librerías externas
* **Orientado a objetos**: Permite estructurar el código de manera modular y reutilizable
* **Multiplataforma**: Funciona en Windows, Linux y macOS sin modificaciones

**Ventajas para este proyecto:**

* Excelente integración con Django, el framework web utilizado
* Gran comunidad de desarrolladores y abundante documentación
* Librerías especializadas para generación de reportes (ReportLab, OpenPyXL)
* Facilidad para trabajar con bases de datos mediante ORMs
* Curva de aprendizaje accesible que facilita el mantenimiento futuro

**Versión utilizada:** Python 3.11

**PostgreSQL**

PostgreSQL es un sistema de gestión de bases de datos relacional de código abierto, considerado uno de los más avanzados y confiables.

**Características principales:**

* **ACID completo**: Garantiza Atomicidad, Consistencia, Aislamiento y Durabilidad en las transacciones
* **Código abierto**: Sin costos de licenciamiento
* **Altamente escalable**: Soporta grandes volúmenes de datos y usuarios concurrentes
* **Tipos de datos avanzados**: JSON, arrays, rangos de fechas, entre otros
* **Ventajas para este proyecto:**
* **Integridad referencial robusta**: Mantiene la consistencia entre tablas relacionadas mediante llaves foráneas
* **Transacciones confiables**: Asegura que las operaciones de reserva se completen correctamente o se reviertan en caso de error
* **Consultas complejas eficientes**: Optimización de queries para reportes y filtros
* **Compatibilidad con Django**: Excelente soporte y rendimiento con el ORM de Django
* **Respaldos y recuperación**: Herramientas nativas para backup y restore de datos

**Ventajas sobre otras opciones:**

* Más robusto que MySQL para operaciones complejas
* Sin limitaciones de licenciamiento como SQL Server
* Mejor manejo de concurrencia que SQLite

**Versión utilizada:** PostgreSQL 15

**Docker**

Docker es una plataforma de contenedorización que permite empaquetar aplicaciones junto con sus dependencias en contenedores ligeros y portables.

**Características principales:**

* **Aislamiento**: Cada contenedor se ejecuta de forma independiente
* **Portabilidad**: Los contenedores funcionan en cualquier sistema que tenga Docker
* **Ligereza**: Comparten el kernel del sistema operativo, consumiendo menos recursos que máquinas virtuales
* **Reproducibilidad**: Garantiza que la aplicación funcione igual en desarrollo y producción
* **Ventajas para este proyecto:**
* **Entorno consistente**: Elimina el problema de "en mi máquina funciona"
* **Instalación simplificada**: Un solo comando (docker-compose up) levanta todo el sistema
* **Gestión de dependencias**: Todas las librerías y servicios están definidos en archivos de configuración
* **Escalabilidad**: Facilita añadir más instancias o servicios según la demanda
* **Mantenimiento simplificado**: Actualizaciones y cambios se realizan sin afectar el servidor principal

**Componentes utilizados:**

* **Docker Engine**: Motor de contenedorización
* **Docker Compose**: Orquestación de múltiples contenedores (aplicación web y base de datos)

**Ventajas sobre instalación tradicional:**

* No requiere instalar Python, PostgreSQL ni dependencias directamente en el servidor
* Rollback rápido a versiones anteriores si hay problemas
* Múltiples entornos (desarrollo, testing, producción) con configuraciones aisladas

**Versión utilizada:** Docker 20.10+ y Docker Compose 2.0+

**Área de Aplicación**

El sistema será aplicado al área de espacios académicos dentro del Campus “Hernán Melgar Justiniano”

**Interfaz de Usuario**

* Diseño general de interfaz gráfica
* Diseño del panel del Encargado
* Diseño del panel del Usuario

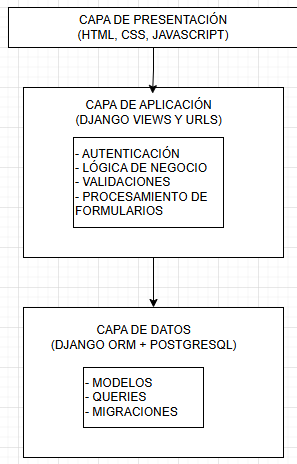
**ARQUITECTURA DEL SISTEMA**

**Arquitectura General**

El sistema sigue una arquitectura MVC (Modelo – Vista – Controlador) implementada con Django:

**FIGURA N° 1**

ARQUITECTURA DEL SISTEMA



**Fuente:** Elaboración propia, 2025

**PATRÓN DE DISEÑO**

MTV (Model-Template-View) de Django:

* Model: Define la estructura de datos (tablas de base de datos)
* Template: Capa de presentación (HTML)
* View: Lógica de negocio (Python)

**Componentes adicionales:**

* Forms: Validación de datos de entrada
* Middleware: Procesamiento de requests/responses
* Signals: Eventos automáticos (ej: envío de correos)

**DISEÑO DE BASE DE DATOS**

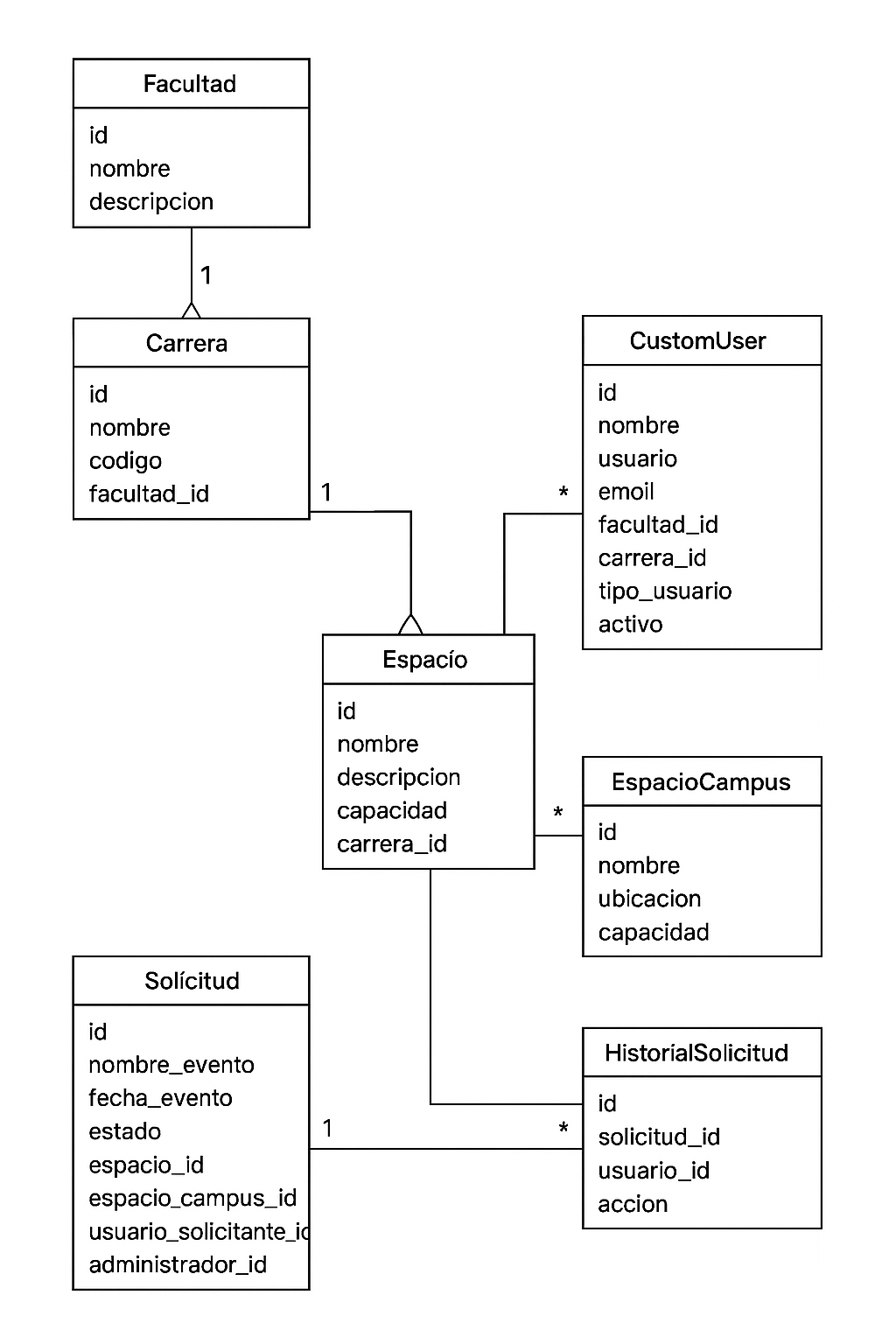
**Descripción General**

El sistema utiliza PostgreSQL como sistema gestor de base de datos relacional, implementando un modelo que garantiza la integridad referencial, normalización y escalabilidad. El diseño sigue las mejores prácticas de bases de datos relacionales y aprovecha las capacidades del ORM de Django para la gestión eficiente de datos.

**Entidades del Sistema**

**FIGURA N° 2**

**DIAGRAMA BD**

****

**Fuente:** Elaboración propia, 2025

**Tabla: CustomUser**

**Descripción:** Modelo principal que gestiona todos los usuarios del sistema. Extiende el modelo AbstractUser de Django para incluir campos específicos del dominio académico.

**Estructura de la tabla:**

**TABLA N°1**

**CUSTOMER**

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo | Restricciones | Descripción |
| id | SERIAL | PRIMARY KEY | Identificado único del usuario |
| username | VARCHAR (150) | UNIQUE, NOT NULL | Nombre de usuario para login |
| email | VARCHAR (254) | UNIQUE, NOT NULL | Correo electrónico |
| password | VARCHAR (128) | NOT NULL | Contraseña hasheada (PBKDF2) |
| first\_name | VARCHAR (150) | NULL | Nombre del usuario |
| last\_name | VARCHAR (150) | NULL | Apellido del usuario |
| teléfono | VARCHAR (15) | NULL | Número telefónico de contacto |
| documento | VARCHAR (20) | UNIQUE | Cédula de identidad o documento |
| tipo\_usuario | VARCHAR (20) | NOT NULL | Rol: ‘administrador’, ‘encargado’, ‘usuario’ |
| facultad\_id | INTEGER | FOREIGN KEY | Referencia a Facultad |
| carrera\_id | INTEEGER | FOREIGN KEY | Referencia a Carrera |
| activado | BOOLEAN | DEFAULT TRUE | Estado activo/inactivo del superusuario |
| verificado | BOOLEAN | DEFAULT FALSE | Indica si el email fue verificado |
| token\_verificacion | VARCHAR (100) | NULL | Token temporal para verificación |
| token\_expira | TIMESTAMP | NULL | Fecha de expiración del Token |
| fecha\_creacion | TIMESTAMP | AUTO | Fecha de registro en el sistema |
| is\_staff | BOOLEAN | DEFAULT FALSE | Acceso al panel administrativo |
| is\_superuser | BOOLEAN | DEFAULT FALSE | Permisos de superusuario |
| date\_joined | TIMESTAMP | AUTO | Fecha de Registro (Django) |
| las\_login | TIMESTAMP | NULL | Última sesión iniciada |

**Fuente:** Elaboración propia, 2025

**Índices:**

* PRIMARY KEY en id
* UNIQUE en username, email, documento
* INDEX en tipo\_usuario (consultas frecuentes)
* INDEX en facultad\_id, carrera\_id (foreign keys)

**Métodos de lógica de negocio:**

* is\_administrador(): Verifica si el usuario tiene rol de administrador
* is\_encargado(): Verifica si el usuario es encargado de espacios
* is\_usuario\_regular(): Verifica si es usuario regular del sistema

**Reglas de negocio:**

* El email debe ser único en todo el sistema
* El documento debe ser único si está presente
* El tipo\_usuario determina los permisos y accesos del usuario

**Tabla: Facultad**

**Descripción:** Representa las facultades académicas de la universidad. Una facultad agrupa múltiples carreras.

**Estructura de la tabla:**

**TABLA N° 2**

**FACULTAD**

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo | Restricciones | Descripción |
| id | SERIAL | PRIMARY KEY | Identificador único |
| nombre | VARCHAR (100) | UNIQUE, NOT NULL | Nombre de la facultad |
| descripcion | TEXT | NULL | Descripción de la facultad |
| fecha\_creacion | TIMESTAMP | AUTO | Fecha de registro |

**Fuente:** Elaboración propia, 2025

**Índices:**

* PRIMARY KEY en id
* UNIQUE en nombre

**Relaciones:**

* Una Facultad tiene muchas Carreras (1:N)
* Una Facultad tiene muchos Usuarios (1:N)

**Tabla: Carrera**

**Descripción:** Representa las carreras académicas que pertenecen a una facultad específica.

**Estructura de la tabla:**

**TABLA N° 3**

**CARRERA**

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo | Restricciones | Descripción |
| id | SERIAL | PRIMARY KEY | Identificador único |
| nombre | VARCHAR (100) | NOT NULL | Nombre de la Carrera |
| código | VARCHAR (20) | UNIQUE | Código identificador de la carrera |
| facultad\_id | INTEGER | FOREIGN KEY, NOT NULL | Referencia a Facultad |
| fecha\_creacion | TIMESTAMP | AUTO | Fecha de Registro |

**Fuente:** Elaboración propia, 2025

**Restricciones:**

* UNIQUE(nombre, facultad\_id): Una carrera no puede repetirse dentro de la misma facultad

**Índices:**

* PRIMARY KEY en id
* UNIQUE en codigo
* INDEX en facultad\_id
* UNIQUE COMPOSITE en (nombre, facultad\_id)

**Relaciones:**

* Una Carrera pertenece a una Facultad (N:1)
* Una Carrera tiene muchos Espacios (1:N)
* Una Carrera tiene muchos Usuarios (1:N)

**Ordenamiento por defecto:** Por nombre de facultad y nombre de carrera

**Tabla: Espacio**

**Descripción:** Representa los espacios académicos específicos de cada carrera (salas, laboratorios, aulas).

**Estructura de la tabla:**

**TABLA N° 4**

**ESPACIO**

|  |  |  |  |
| --- | --- | --- | --- |
| CAMPO | TIPO | RESTRICCIONES | DESCRIPCIÓN |
| id | SERIAL | PRIMARY KEY | Identificado único |
| nombre | VARCHAR (100) | NOT NULL | Nombre del espacio |
| descripción | TEXT | NOT NULL | Descripción detallada del espacio |
| capacidad | INTEGER | NULL | Capacidad de personas |
| ubicación | VARCHAR (200) | NULL | Ubicación física (Edificio, piso) |
| activo | BOOLEAN | DEFAULT TRUE | Estado activo/inactivo |
| carrera\_id | INTEGER | FOREIGN KEY, NOT NULL | Carrera propietaria del espacio |
| encargado\_id | INTEGER | FOREIGN KEY, NOT NULL | Usuario encargado del espacio |
| fecha\_creacion | TIMESTAMP | AUTO | Fecha de registro |

**Fuente:** Elaboración propia, 2025

**Restricciones:**

* UNIQUE(nombre, carrera\_id): Nombre único dentro de cada carrera
* CHECK: El encargado debe tener tipo\_usuario = 'encargado'

**Índices:**

* PRIMARY KEY en id
* INDEX en carrera\_id
* INDEX en encargado\_id
* INDEX en activo (filtros frecuentes)
* UNIQUE COMPOSITE en (nombre, carrera\_id)

**Validaciones a nivel de aplicación:**

* El usuario asignado como encargado debe ser de tipo 'encargado'
* No puede haber dos espacios con el mismo nombre en la misma carrera

**Relaciones:**

* Un Espacio pertenece a una Carrera (N:1)
* Un Espacio tiene un Encargado (N:1 con CustomUser)
* Un Espacio tiene muchas Solicitudes (1:N)

**Tabla: EspacioCampus**

**Descripción:** Representa espacios comunes de los campus disponibles para toda la universidad, no pertenecen a ninguna carrera específica.

**Estructura de la tabla:**

**TABLA N° 5**

**ESPACIOCAMPUS**

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo | Restricciones | Descripción |
| id | SERIAL | PRIMARY KEY | Identificado único |
| nombre | VARCHAR (100) | UNIQUE, NOT NULL | Nombre del espacio |
| ubicación | VARCHAR (200) | NULL | Ubicación física |
| capacidad | INTEGER | NULL | Capacidad de personas |
| descripción | TEXT | NULL | Descripción del espacio |
| encargado | INTEGER | FOREIGN KEY, NULL | Usuario encargado (opcional) |
| fecha\_creacion | TIMESTAMP | AUTO | Fecha de Registro |

**Fuente:** Elaboración propia, 2025

**Índices:**

* PRIMARY KEY en id
* UNIQUE en nombre
* INDEX en encargado\_id

**Diferencias con tabla Espacio:**

* No pertenece a ninguna carrera
* El encargado es opcional
* Disponible para solicitudes de todas las facultades

**Tabla: Solicitud**

**Descripción:** Almacena todas las solicitudes de reserva de espacios académicos realizadas por usuarios del sistema.

**Estructura de la tabla:**

**TABLA N° 6**

**SOLICITUD**

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo | Restricciones | Descripción |
| id | SERIAL | PRIMARY KEY | Identificado único |
| nombre\_evento | VARCHAR (100) | NOT NULL | Nombre del evento a realizar |
| descripción\_evento | TEXT | NULL | Descripción detallada del evento |
| fecha\_evento | TIMESTAMP | NOT NULL | Fecha y hora de inicio del evento |
| fecha\_fin\_evento | TIMESTAMP | NULL | Fecha y hora de finalización |
| tipo\_espacio | VARCHAR (20) | NOT NULL | ‘carrera’ o ‘campus’ |
| espacio\_id | INTEGER | FOREIGN KEY, NULL | Espacio de carrera solicita |
| espacio\_campus\_id | INTEGER | FOREIGN KEY, NULL | Espacio de campus solicitado |
| usuario\_solicitante\_id | INTEGER | GOREIGN KEY, NOT NULL | Usuario que solicita |
| administrador\_id | INTEGER | FOREIGN KEY, NULL | Encargado que procesa |
| estado | VARCHAR (20) | NOT NULL | ‘pendiente’, ‘aceptada’, ‘rechazada’ |
| motivo\_rechazo | TEXT | NULL | Razón del rechazo (si aplica) |
| observaciones | TEXT | NULL | Observaciones del encargado |
| archivo\_adjunto | VARCHAR (255) | NULL | Ruta al archivo adjunto |
| fehca\_creacion | TIMESTAMP | AUTO | Fecha de creación |
| fecha\_actualizacion | TIMESTAMP | AUTO\_UPDATE | Última modificación |

**Fuente:** Elaboración propia, 2025

**Índices:**

* PRIMARY KEY en id
* INDEX en usuario\_solicitante\_id
* INDEX en espacio\_id
* INDEX en espacio\_campus\_id
* INDEX en administrador\_id
* INDEX en estado (consultas frecuentes)
* INDEX COMPOSITE en (fecha\_evento, fecha\_fin\_evento) (validación de conflictos)

**Validaciones:**

* fecha\_fin\_evento debe ser posterior a fecha\_evento
* Solo uno de espacio\_id o espacio\_campus\_id puede tener valor (no ambos)
* Si tipo\_espacio = 'carrera', entonces espacio\_id debe tener valor
* Si tipo\_espacio = 'campus', entonces espacio\_campus\_id debe tener valor
* archivo\_adjunto máximo 10 MB
* Formatos permitidos: PDF, JPG, JPEG, PNG, GIF, DOC, DOCX

**Métodos de negocio:**

* puede\_ser\_editada(): Retorna True solo si estado = 'pendiente'
* aprobar(administrador, observaciones): Cambia estado a 'aceptada'
* rechazar(administrador, motivo): Cambia estado a 'rechazada'
* get\_nombre\_espacio(): Retorna el nombre del espacio solicitado
* get\_espacio\_objeto(): Retorna el objeto espacio correspondiente

**Propiedades calculadas:**

* nombre\_archivo: Extrae el nombre del archivo adjunto
* tamano\_archivo\_legible: Convierte bytes a formato legible (KB, MB)

**Relaciones:**

* Una Solicitud es creada por un Usuario (N:1)
* Una Solicitud puede ser procesada por un Encargado/Admin (N:1)
* Una Solicitud pertenece a un Espacio O EspacioCampus (N:1)
* Una Solicitud tiene muchos registros de Historial (1:N)

**Tabla: HistorialSolicitud**

**Descripción:** Registra todas las acciones realizadas sobre cada solicitud para trazabilidad y auditoría.

**Estructura de la tabla:**

**TABLA N° 6**

**HISTORIAL SOLICITUD**

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo | Restricciones | Descripción |
| id | SERIAL | PRIMARY KEY | Identificado único |
| solicitud\_id | INTEGER | FOREIGN KEY, NOT NULL | Solicitud relacionada |
| usuario\_id | INTEGER | FOREIGN KEY, NOT NULL | Usuario que realizó la acción |
| acción | VARCHAR (50) | NOT NULL | Tipo de acción realizada |
| descripción | TEXT | NULL | Descripción detallada |
| fecha | TIMESTAMP | AUTO | Fecha y hora de la acción |

**Valores de acción:**

* 'creada': Solicitud creada por primera vez
* 'aprobada': Solicitud aprobada por encargado
* 'rechazada': Solicitud rechazada con motivo
* 'modificada': Cambios realizados a la solicitud

**Índices:**

* PRIMARY KEY en id
* INDEX en solicitud\_id
* INDEX en usuario\_id
* INDEX en fecha (ordenamiento)

**Propósito:**

* Auditoría completa de cambios
* Trazabilidad de responsables
* Análisis histórico de patrones
* Cumplimiento de políticas institucionales

**Normalización de la Base de Datos**

El diseño de la base de datos cumple con la Tercera Forma Normal (3FN), lo que garantiza:

* **Eliminación de redundancia**: Cada dato se almacena una sola vez
* **Integridad de datos**: Las relaciones mediante claves foráneas previenen inconsistencias
* **Facilidad de mantenimiento**: Los cambios se realizan en un solo lugar
* **Optimización de consultas**: Estructura eficiente para joins y búsquedas

**Seguridad de Datos**

**Medidas implementadas:**

1. **Contraseñas**: Almacenadas con hash PBKDF2-SHA256 (260,000 iteraciones)
2. **Validación**: Constraints a nivel de base de datos
3. **Respaldos**: Configuración de backups automáticos diarios
4. **Transacciones**: Uso de ACID para operaciones críticas
5. **Auditoría**: Tabla HistorialSolicitud para trazabilidad completa

**CONCLUSIÓN GENERAL DEL SISTEMA**

El desarrollo e implementación del Sistema Web para la Gestión de Reservas de Espacios Académicos del Campus "Hernán Melgar Justiniano" de la Universidad Autónoma del Beni "José Ballivián" representa un avance significativo en la modernización de los procesos administrativos de la institución, transformando un procedimiento manual ineficiente en una solución tecnológica integral, automatizada y centralizada.

**Script de la base de datos:**

-- ==============================================

-- ELIMINACIÓN DE TABLAS (si existen)

-- ==============================================

DROP TABLE IF EXISTS historial\_solicitud CASCADE;

DROP TABLE IF EXISTS solicitud CASCADE;

DROP TABLE IF EXISTS espacio\_campus CASCADE;

DROP TABLE IF EXISTS espacio CASCADE;

DROP TABLE IF EXISTS custom\_user CASCADE;

DROP TABLE IF EXISTS carrera CASCADE;

DROP TABLE IF EXISTS facultad CASCADE;

-- ==============================================

-- TABLA: Facultad

-- ==============================================

CREATE TABLE facultad (

id SERIAL PRIMARY KEY,

nombre VARCHAR(100) UNIQUE NOT NULL,

descripcion TEXT,

fecha\_creacion TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- ==============================================

-- TABLA: Carrera

-- ==============================================

CREATE TABLE carrera (

id SERIAL PRIMARY KEY,

nombre VARCHAR(100) NOT NULL,

codigo VARCHAR(20) UNIQUE,

facultad\_id INTEGER NOT NULL,

fecha\_creacion TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

CONSTRAINT fk\_carrera\_facultad FOREIGN KEY (facultad\_id)

REFERENCES facultad(id) ON DELETE CASCADE,

CONSTRAINT unique\_carrera\_facultad UNIQUE (nombre, facultad\_id)

);

CREATE INDEX idx\_carrera\_facultad ON carrera(facultad\_id);

-- ==============================================

-- TABLA: CustomUser (Usuarios del Sistema)

-- ==============================================

CREATE TABLE custom\_user (

id SERIAL PRIMARY KEY,

password VARCHAR(128) NOT NULL,

last\_login TIMESTAMP,

is\_superuser BOOLEAN DEFAULT FALSE,

username VARCHAR(150) UNIQUE NOT NULL,

first\_name VARCHAR(150),

last\_name VARCHAR(150),

email VARCHAR(254),

is\_staff BOOLEAN DEFAULT FALSE,

is\_active BOOLEAN DEFAULT TRUE,

date\_joined TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

telefono VARCHAR(15),

documento VARCHAR(20) UNIQUE,

facultad\_id INTEGER,

carrera\_id INTEGER,

tipo\_usuario VARCHAR(20) DEFAULT 'usuario' CHECK (tipo\_usuario IN ('administrador', 'encargado', 'usuario')),

activo BOOLEAN DEFAULT TRUE,

fecha\_creacion TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

token\_verificacion VARCHAR(100),

token\_expira TIMESTAMP,

verificado BOOLEAN DEFAULT FALSE,

CONSTRAINT fk\_user\_facultad FOREIGN KEY (facultad\_id)

REFERENCES facultad(id) ON DELETE SET NULL,

CONSTRAINT fk\_user\_carrera FOREIGN KEY (carrera\_id)

REFERENCES carrera(id) ON DELETE SET NULL

);

CREATE INDEX idx\_user\_username ON custom\_user(username);

CREATE INDEX idx\_user\_email ON custom\_user(email);

CREATE INDEX idx\_user\_tipo ON custom\_user(tipo\_usuario);

CREATE INDEX idx\_user\_facultad ON custom\_user(facultad\_id);

CREATE INDEX idx\_user\_carrera ON custom\_user(carrera\_id);

-- ==============================================

-- TABLA: Espacio (Espacios de Carrera)

-- ==============================================

CREATE TABLE espacio (

id SERIAL PRIMARY KEY,

nombre VARCHAR(100) NOT NULL,

descripcion TEXT NOT NULL,

capacidad INTEGER,

ubicacion VARCHAR(200),

activo BOOLEAN DEFAULT TRUE,

carrera\_id INTEGER NOT NULL,

encargado\_id INTEGER NOT NULL,

fecha\_creacion TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

CONSTRAINT fk\_espacio\_carrera FOREIGN KEY (carrera\_id)

REFERENCES carrera(id) ON DELETE CASCADE,

CONSTRAINT fk\_espacio\_encargado FOREIGN KEY (encargado\_id)

REFERENCES custom\_user(id) ON DELETE CASCADE,

CONSTRAINT unique\_espacio\_carrera UNIQUE (nombre, carrera\_id)

);

CREATE INDEX idx\_espacio\_carrera ON espacio(carrera\_id);

CREATE INDEX idx\_espacio\_encargado ON espacio(encargado\_id);

CREATE INDEX idx\_espacio\_activo ON espacio(activo);

-- ==============================================

-- TABLA: EspacioCampus (Espacios Generales)

-- ==============================================

CREATE TABLE espacio\_campus (

id SERIAL PRIMARY KEY,

nombre VARCHAR(100) UNIQUE NOT NULL,

ubicacion VARCHAR(200),

capacidad INTEGER CHECK (capacidad >= 0),

descripcion TEXT,

fecha\_creacion TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

encargado\_id INTEGER,

CONSTRAINT fk\_espacio\_campus\_encargado FOREIGN KEY (encargado\_id)

REFERENCES custom\_user(id) ON DELETE SET NULL

);

CREATE INDEX idx\_espacio\_campus\_encargado ON espacio\_campus(encargado\_id);

-- ==============================================

-- TABLA: Solicitud

-- ==============================================

CREATE TABLE solicitud (

id SERIAL PRIMARY KEY,

nombre\_evento VARCHAR(100) NOT NULL,

descripcion\_evento TEXT,

fecha\_evento TIMESTAMP NOT NULL,

fecha\_fin\_evento TIMESTAMP,

estado VARCHAR(20) DEFAULT 'pendiente' CHECK (estado IN ('pendiente', 'aceptada', 'rechazada')),

motivo\_rechazo TEXT,

observaciones TEXT,

fecha\_creacion TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

fecha\_actualizacion TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

tipo\_espacio VARCHAR(20) DEFAULT 'carrera' CHECK (tipo\_espacio IN ('carrera', 'campus')),

archivo\_adjunto VARCHAR(255),

espacio\_id INTEGER,

espacio\_campus\_id INTEGER,

usuario\_solicitante\_id INTEGER NOT NULL,

administrador\_id INTEGER,

CONSTRAINT fk\_solicitud\_espacio FOREIGN KEY (espacio\_id)

REFERENCES espacio(id) ON DELETE CASCADE,

CONSTRAINT fk\_solicitud\_espacio\_campus FOREIGN KEY (espacio\_campus\_id)

REFERENCES espacio\_campus(id) ON DELETE CASCADE,

CONSTRAINT fk\_solicitud\_usuario FOREIGN KEY (usuario\_solicitante\_id)

REFERENCES custom\_user(id) ON DELETE CASCADE,

CONSTRAINT fk\_solicitud\_administrador FOREIGN KEY (administrador\_id)

REFERENCES custom\_user(id) ON DELETE SET NULL,

CONSTRAINT check\_fecha\_evento CHECK (fecha\_fin\_evento IS NULL OR fecha\_fin\_evento > fecha\_evento),

CONSTRAINT check\_un\_solo\_espacio CHECK (

(tipo\_espacio = 'carrera' AND espacio\_id IS NOT NULL AND espacio\_campus\_id IS NULL) OR

(tipo\_espacio = 'campus' AND espacio\_campus\_id IS NOT NULL AND espacio\_id IS NULL)

)

);

CREATE INDEX idx\_solicitud\_estado ON solicitud(estado);

CREATE INDEX idx\_solicitud\_fecha\_evento ON solicitud(fecha\_evento);

CREATE INDEX idx\_solicitud\_usuario ON solicitud(usuario\_solicitante\_id);

CREATE INDEX idx\_solicitud\_administrador ON solicitud(administrador\_id);

CREATE INDEX idx\_solicitud\_espacio ON solicitud(espacio\_id);

CREATE INDEX idx\_solicitud\_espacio\_campus ON solicitud(espacio\_campus\_id);

CREATE INDEX idx\_solicitud\_tipo\_espacio ON solicitud(tipo\_espacio);

-- ==============================================

-- TABLA: HistorialSolicitud

-- ==============================================

CREATE TABLE historial\_solicitud (

id SERIAL PRIMARY KEY,

solicitud\_id INTEGER NOT NULL,

usuario\_id INTEGER NOT NULL,

accion VARCHAR(50) NOT NULL,

descripcion TEXT,

fecha TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

CONSTRAINT fk\_historial\_solicitud FOREIGN KEY (solicitud\_id)

REFERENCES solicitud(id) ON DELETE CASCADE,

CONSTRAINT fk\_historial\_usuario FOREIGN KEY (usuario\_id)

REFERENCES custom\_user(id) ON DELETE CASCADE

);

CREATE INDEX idx\_historial\_solicitud ON historial\_solicitud(solicitud\_id);

CREATE INDEX idx\_historial\_usuario ON historial\_solicitud(usuario\_id);

CREATE INDEX idx\_historial\_fecha ON historial\_solicitud(fecha);