UACM

Universidad Autónoma
de la Ciudad de México

NADA HUMANO ME ES AJENO

COLEGIO DE CIENCIA Y TECNOLOGÍA

LICENCIATURA EN INGENIERÍA DE SOFTWARE

Sistema Web para la Gestión de Reservación de Sitios

TRABAJO RECEPCIONAL

QUE PARA OPTAR POR EL TÍTULO DE

LICENCIADO EN INGENIERÍA DE SOFTWARE

P R E S E N T A

JOB HERNÁNDEZ RODRÍGUEZ

D I R E C T O R

LIC. MARCOS LÓPEZ CHIMIL

Ciudad de México, mayo de 2025.

UACM
Universidad Autónoma de la Ciudad de México
Nada humano me es ajeno

CA Coordinación Académica

SISTEMA BIBLIOTECARIO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD AUTÓNOMA DE LA CIUDAD DE MÉXICO
COORDINACIÓN ACADÉMICA

RESTRICCIONES DE USO PARA LAS TESIS DIGITALES

DERECHOS RESERVADOS©

Índice

Índice **2**
Agradecimientos **5**
Capítulo 1 INTRODUCCIÓN **6**
Antecedentes 7
Objetivos 10
Objetivos Generales 10
Objetivos Específicos 10
Capítulo 2 MARCO TEORICO **11**
Proceso Unificado 13
Casos de uso 16
Spring Boot Framework 16
El mapeo objeto relacional 19
Herramientas de gestión de dependencias y construcción de proyectos 20
Capítulo 3 Análisis de Requisitos **21**
Información preliminar 21
Actores 25
Flujo principal 26
Reglas de negocio 26
Requisitos funcionales 26
Requisitos no funcionales 29
Diagrama de Casos de Uso 30
Casos de uso 31
Caso de uso Gestión de Reservaciones 32
Información general para este caso de uso. 32
Precondiciones. 32
Postcondiciones del caso de uso Gestión de Reservaciones 33
Restricciones / Problemas / Riesgos 33
Eventos desencadenantes 33
Listado de cursos de los casos de uso 34
Cursos detallados 34
3.1 Curso Principal Nueva Reservación Usuario Profesor 35
Capítulo 4 Análisis **40**

Diagrama de Objetos ... 40
Diagrama de secuencia ... 42
Diagrama de clases ... 44
Capítulo 5 Diseño ... **46**
Arquitectura de capas ... 46
Diagrama de despliegue ... 47
Sincronización de las reservaciones ... 49
Capítulo 6 Implementación ... **51**
Creación del proyecto ... 51
Configuración del proyecto ... 51
Gradle build ... 51
Dependencias preliminares ... 53
Archivo de configuración Application.properties ... 55
Desarrollo del módulo para realizar una reservación ... 57
Capa de persistencia ... 58
Capa de negocio ... 59
Capa de presentación ... 62
Ejecución de una reservación ... 71
Accediendo al sistema ... 71
Seleccionar del menú reservación nueva reservación ... 72
Fase 1 Información preliminar ... 72
Fase 2 Selección de horarios ... 77
Fase 3 Confirmación de reservación ... 78
Capítulo 7 Pruebas ... **80**
Pruebas unitarias ... 80
Pruebas del sistema utilizando complejidad ciclomática Fase 1 ... 83
Pruebas fase 1 ... 86
Complejidad Ciclomática Fase 1. Procesamiento del primer formulario ... 88
Pruebas fase 1 primer formulario ... 89
Pruebas fase 2 ... 95
Conclusiones ... **101**
Listado de archivos de configuración ... **103**
Listado de código fuente ... **103**
Referencias ... **104**

Anexo 105

Anexo 1 Casos de uso 105

1. Información general Gestión de Usuarios. 105

2. Información general Gestión de Reportes. 110

3. Información general Gestión de Reservaciones. 115

4. Información general Gestión de catálogos. 123

Agradecimientos

Quiero expresar mi más sincero agradecimiento, en primer lugar, a mi madre, por su amor incondicional, apoyo constante y por ser mi mayor fuente de fortaleza é inspiración a lo largo de la carrera.

Mi profundo agradecimiento a la Universidad Autónoma de la Ciudad de México, por ser el pilar fundamental en mi formación académica y personal. A lo largo de estos años, la cual brindó no solo el conocimiento si no las herramientas necesarias para desarrollarme de manera profesional. Gracias por fomentar en mí el compromiso, la disciplina.

A mi director de trabajo recepcional Marcos López Chimil, gracias por su dedicación, orientación y apoyo constante durante este proyecto.

Agradezco también a todos los profesores que formaron parte de mi formación académica, por compartir sus conocimientos y guiarme con dedicación durante toda la carrera. A los compañeros que se fueron cruzando en este trayecto, gracias por los aprendizajes compartidos y por ser parte de esta experiencia.

Finalmente, extiendo un especial agradecimiento a José Ernesto Ortiz Bautista que amablemente nos brindó la información necesaria para el desarrollo del sistema, cuya colaboración fue fundamental para la realización de este trabajo.

Capítulo 1 INTRODUCCIÓN

En la Universidad Autónoma de la Ciudad de México (UACM) en el área de recursos materiales del plantel San Lorenzo Tezonco (SLT), se presenta una problemática al reservar sitios para la realización de actividades académicas o administrativas, estos sitios pueden ser un aula audio visual, sala de cómputo o algún lugar idóneo para la realización de dichas actividades dentro del plantel (Ágora, Umbral etc.)

Actualmente se cuenta con un sistema que abordó este problema, que realicé durante mi prestación de servicio social, el cual se denomina “Sistema de Reservación de Espacios”, dicho sistema evita los traslapes y cuenta con las funciones: gestión de sitios, eventos, equipos y disponibilidad de horarios, sin embargo, percibimos que el usuario del área de Coordinación De Servicios Administrativos, no se familiarizó fácilmente con el sistema; por lo cual inferimos que el ambiente del sistema es poco amigable para nuestro usuario. Por otra parte, surgieron nuevas necesidades que el sistema puede cubrir de manera efectiva, que no se tenían contempladas; entre estas necesidades se encuentran: que los solicitantes puedan reservar los sitios desde internet, esto ayuda a los profesores, órganos de gobierno de la institución y administrativos, a que lo puedan acceder las 24 horas, 7 días de la semana al sistema, además de contar con diferentes roles, proporcionará distintos niveles de seguridad, e incluso ayudará a liberar al administrador de una sobrecarga de trabajo.

Como estudiante de la UACM en el plantel SLT, pude ver como ésta tiene necesidades de software. Deseaba realizar este sistema como ayuda o retribución, además este ejercicio me ha ayudado a adquirir experiencia para enfrentar problemas reales y tener un enfoque más profundo del mundo laboral.

La necesidad de desarrollar el sistema web surge con el propósito de simplificar y agilizar el proceso de reservación, además de hacer una interfaz más amigable para los usuarios, ya que como se mencionó, el usuario del sistema de escritorio no se pudo acoplar al sistema.

Con esta iniciativa los profesores podrán realizar reservaciones de manera conveniente desde una computadora personal o incluso desde un dispositivo móvil.

En otras palabras, se busca eliminar la necesidad de acudir directamente al área de recursos materiales y completar una papeleta física para llevar a cabo una reservación, brindando así una solución más eficiente y accesible.

La aplicación desarrollada durante mi servicio social es una versión de escritorio, y la transición hacia una versión web conlleva diversas ventajas significativas, entre las cuales destacan:

- El acceso para múltiples usuarios a la vez.
- La información de las reservaciones puede ser monitoreada las veinticuatro horas, los siete días de la semana.
- Menor uso de recursos de papelería (papel, plumones, pluma, copias, etc.)
- Compatibilidad multiplataforma.
- Diferentes tipos de acceso al sistema.
- Automatización de reportes.
- Menor propensión a errores.
- Certeza en la realización de reservaciones.

Antecedentes

La Coordinación De Servicios Administrativos cuenta con el sistema denominado “Sistema de Reservación de Sitios” con el cual se administran los sitios del plantel. Éste permite realizar las siguientes actividades:

Gestión de Reservaciones

- Agregar reservación. El sistema le permite al usuario agregar una reservación siempre y cuando esté disponible, el sitio en la fecha y horario deseado. En caso contrario, al usuario se le envía un mensaje indicándole que el sitio no está disponible y en ese caso podrá hacer uso de un botón de ayuda, ubicado en esa sección, él cual le mostrará los sitios disponibles para esa fecha y ese horario. Puede haber otras causas por las que no se logré agregar una reservación, y en ese caso el sistema enviará el mensaje de error correspondiente, por ejemplo: capacidad del sitio rebasada, error en captura de información, etc.

El sistema tiene dos formas de agregar reservaciones:

  - Agregar reservación por fechas: Esta característica permite seleccionar un día y agregarlo.
  - Agregar reservación por bloques. Selecciona un bloque de días, referido con un día inicial y un día final; periodo de días que puede abarcar, hasta un semestre. Por poner un ejemplo: el día inicial sería 1 enero y el final 31 mayo. También se deben indicar los días de la semana en los que se llevará a cabo el evento es decir; lunes, martes, miércoles, jueves, viernes, sábado, domingo. Al finalizar el registro de una reservación, el sistema envía un correo electrónico y muestra el identificador de la reservación.

- Modificar reservación. El sistema permite modificar la información de una reservación. Mediante el identificador de la reservación, el sistema mostrará la información actual y ahí mismo se podrán realizar los cambios.
- Eliminar reservación. Esta es por medio del identificador, el sistema realiza la búsqueda y elimina la reservación

Gestión de Catálogos

En el sistema existen los siguientes catálogos, cada uno independiente de otro: equipos, eventos, sitios, áreas administrativas. Los catálogos al tener una funcionalidad similar se agruparon en un solo conjunto de información esto para no tener tanta documentación, cabe resaltar que cada uno de los catálogos es independiente uno de otro y su funcionalidad es la siguiente:

- Agregar catálogo. Permite al usuario agregar a los catálogos.
- Modifica catálogo. Permite al usuario modificar información de los catálogos
- Eliminar catálogo. El sistema muestra la lista de todos los elementos de un catálogo y el usuario selecciona el que desea borrar.

Gestión de Usuarios

Agregar Usuario. Permite al administrador agregar otros usuarios para poder usar el sistema. El administrador es el único que puede realizar esta acción.

Modificar datos del Usuario. Los usuarios pueden modificar su información.

Eliminar Usuario. Permite al administrador eliminar un usuario, esta acción sólo la puede realizar el administrador del sistema.

Gestión de Reportes

El usuario administrador puede generar los siguientes reportes:

- Reporte de eventos por edificio. Es la representación de una gráfica de barras con el número de eventos realizados por edificio.
- Reporte de eventos. Es la representación de una gráfica de barras con el número de veces que se repitió un evento.
- Reporte de actividades. Es el reporte anual de cuantos eventos se realizaron de forma anual en una gráfica de barras.
- Reporte de edificios. Es el reporte anual de cuantos eventos se realizaron por edificio en una gráfica de barras.
- Reporte de eventos y material requerido. Es el reporte que se genera diariamente para ver los eventos que corresponden.
- Ver Disponibilidad
    - El sistema permite ver en bloques de treinta minutos la disponibilidad de los sitios dependiendo del día seleccionado.

Pros y contras de la situación actual

- Se automatizó la tarea, sin embargo, puede mejorar.

Fortalezas de la situación actual

- Evita empalmes entre horarios y sitios.
- Facilita la creación de reportes.

Debilidades de la situación actual

- Sólo puede ser utilizada por el administrador del sistema.
- Sólo se puede usar en la máquina en la que se realizó la instalación.
- El diseño visual puede mejorar.

Objetivos

Objetivos Generales

Implementación de una aplicación web que permita gestionar las reservaciones de los espacios disponibles dentro del plantel SLT de la UACM. Todo el código desarrollado estará debidamente documentado y sujeto a licencia de código abierto y será donado a la universidad, fomentando así la colaboración y el acceso abierto al software.

Objetivos Específicos

- Utilizar las tecnologías del Framework de Spring Boot y Bootstrap; esto con el fin de tener más experiencia con estas tecnologías que se utilizan en el mundo laboral.
- Creación de una interfaz amigable e intuitiva.
- Tener diferentes niveles de seguridad según el tipo de usuario.
- Conseguir un buen rendimiento en todas las funcionalidades del sistema.
- Hacer uso de las metodologías para el proceso de software y las buenas prácticas estudiadas en la carrera de Ingeniería de Software.

Capítulo 2 MARCO TEORICO

Siempre han existido fallos de software como el reportado por la Secretaría de la Función Pública al dejar expuestos datos de 830 mil funcionarios, La información vulnerada estaba disponible en Internet sin protección de contraseñas, los datos que se podían extraer eran: datos personales, cuentas bancarias, etc., como lo menciona el artículo del Economista “La base de datos estuvo expuesta por lo menos desde el 6 de mayo y hasta el 30 de junio de 2020 a través del motor de búsqueda en internet Shodan, que indexó los registros con el certificado de seguridad (SSL, Secure Sockets Layer) de funcionpublica.gob.mx el primer miércoles de mayo desde la dirección IP 200.33.31.87.”. Como menciona Soto Galindo, el sistema tiene una puerta trasera, para acceder a la base de datos sin la necesidad de una contraseña, con lo que la información es muy fácil de vulnerar; lo que quiere decir que se tuvo un error en el diseño de software. Los errores de software, configuración y hardware como en el ejemplo anteriormente mencionado, pueden provocar más que solo pérdida de información, dinero, o retrasos en entregas del producto. Estos fallos pueden ocasionar incluso la pérdida de vidas humanas, como fue el caso de un avión Boeing 737 Max 8 en Indonesia el cual cayó precipitadamente dejando 189 muertos. En este caso “Los pilotos de pruebas del gobierno que comprobaron el software actualizado de Boeing MAX en un simulador de vuelo la semana pasada encontraron una falla que podría resultar en la caída de la nariz del avión, según fuentes cercanas al proceso de certificación. Un fallo similar estuvo involucrado en ambos accidentes registrados por el MAX, el software de control de vuelo del avión presionó la nariz hacia abajo basándose en lecturas erróneas de un sensor” (Hosteltur, s. f.).

A través de la historia del software podemos observar que en sus inicios era realizado de una forma artesanal, puesto que no había ningún tipo de guía o estrategia que indicara qué pasos se podrían seguir para la correcta realización de un software de calidad. Por esta razón se han generado metodologías que ayudan a seguir pasos para el desarrollo de software.

Existen varios tipos de metodologías de desarrollo de software, cada una con distintas técnicas y métodos, todas ellas enfocadas en un tipo de proyecto donde

éstas pueden ser más eficaces en ciertos aspectos. Algunas metodologías pueden tener riesgos, ya que realizar algún cambio en su implantación, análisis, diseño o requerimientos pueden costar mucho trabajo lo que implica un mayor costo y tiempos de entrega mayor a los propuestos.

El usar la metodología correcta que cumpla con las características del proyecto, implica un menor riesgo de retrasos y un menor número de errores. A continuación, se mencionarán algunas metodologías de desarrollo de software, a fin de diferenciar los enfoques que manejan.

Modelo en Cascada: esta se centra en terminar una actividad antes de pasar a la siguiente. Lo que implica que realizar cambios es muy riesgoso. Por eso es importante definir bien todos los requisitos para evitar modificaciones. Las etapas principales de este proceso son las siguientes:

- Requisitos
- Diseño
- Implementación
- Verificaciones
- Mantenimiento

Metodologías Ágiles: se basan en entregas frecuentes y rápidas del sistema a desarrollar. Muchas de ellas no cuentan con algún modelo sistematizado o requieren más documentación. Las metodologías ágiles en la actualidad son muy utilizadas, pero pueden tener muchos fallos si no se tratan con cuidado desde su inicio, ya que realizar algún cambio cuesta mucho tiempo de elaboración y lleva muchos riesgos en el diseño, por tener poca o nula documentación.

Es cierto que las dos metodologías anteriormente mencionadas pueden tener riesgos, pero, aun así, el buen manejo de estas puede tener beneficio en algún tipo de proyecto en específico. Todo dependerá del entendimiento de la metodología, equipo de trabajo, uso de tecnologías y otros factores que pueden hacer que estas metodologías fallen.

Proceso Unificado

Para la elección de la metodología de desarrollo de software como vimos anteriormente, existen varias elecciones posibles pero la que más se adecua al proyecto es el Proceso Unificado ya que se fueron realizando distintos cambios durante todas las etapas. Cabe aclarar que esta también es una metodología ágil, pero en comparación con otras, requiere documentación detallada para sustentar el proyecto, como veremos más adelante. Se caracteriza por ser iterativo e incremental, lo cual nos permite reducir riegos en posibles cambios al proyecto.

El Proceso Unificado es una metodología enfocada en casos de uso, desarrollado principalmente por Ivar Jacobson, Grady Booch y James Rumbaugh. Éste surge con el objetivo de mejorar tanto la calidad como la eficiencia en el desarrollo de software. A medida que la complejidad de los proyectos ha ido en aumento, las metodologías tradicionales han demostrado no ser viables para garantizar que los proyectos se completen dentro del tiempo y los parámetros establecidos. El Proceso Unificado está conformado por varias técnicas, como el modelamiento de un sistema usando el Lenguaje Unificado de Modelado (UML) y la especificación de requisitos usando los Casos de Uso. Uno de sus principios fundamentales es ser "iterativo e incremental", lo que implica la elaboración progresiva del proyecto a través de múltiples iteraciones. En lugar de entregar el proyecto en su totalidad de una sola vez, se van entregando módulos del sistema de manera gradual. Esto permite al cliente evaluar si el producto cumple con sus expectativas a lo largo del proceso de desarrollo. Además, brinda al desarrollador la flexibilidad necesaria para realizar ajustes y mejoras conforme avanza el proyecto, reduciendo así los riesgos asociados a cambios drásticos y costosos.

El uso de UML en el Proceso Unificado facilita la comunicación, la planificación y la documentación, asegurando que los interesados tengan una visión clara del sistema en cada una de las etapas del proceso.
UML es un lenguaje de modelado el cual nos permite especificar, estandarizar, construir y documentar sistemas de software. Está basado en el modelo de Ericsson de 1967 el cual, para modelar sistemas lo hacía a través de subsistemas o componentes que se comunicaban entre sí, además, los subsistemas de más bajo

nivel se agrupaban en subsistemas de más alto nivel. “Estos diagramas de bloques corresponden directamente a una versión simplificada de los diagramas de clases o paquetes UML simplificados en lo que sólo mostraban asociaciones y que utilizaban para comunicaciones” (Ivar Jacobson et al., s. f.), aunque ya existía este tipo de modelado, los clientes no estaban familiarizados con los modelos presentados en su época.

El levantamiento de requerimientos consiste en extraer todos los procesos e ideas del usuario que permitan realizar sus tareas, reglas, dentro de un sistema ya hecho o uno nuevo. Esto se hace con el fin de que el usuario y el desarrollador puedan entenderse llevando estas ideas a un documento denominado caso de uso. “Los casos de uso son fragmentos de funcionalidad del sistema que proporciona al usuario un resultado importante. Los casos de uso representan los requisitos funcionales.”(Ivar Jacobson et al., s. f.), es decir dividen la funcionalidad en varios subsistemas, los cuales se van a regir por sus propias reglas, usuarios y condiciones, y pueden afectar otros casos de uso. Finalmente, un caso de uso da una descripción detallada de todo el flujo de su fragmento de funcionalidad.

Haciendo uso del Proceso Unificado, el cual propone un modelo iterativo e incremental, podemos observar todas las etapas listadas a continuación:

- Modelamiento de la organización (Flujo de trabajo)
- Especificación de requisitos (Análisis de requisitos)
- Análisis
- Diseño del sistema y de objetos
- Implantación
- Pruebas
- Liberación o despliegue
- Administración de configuración y cambios
- Administración del proyecto
- Ambiente

Cada etapa se revisa con un distinto nivel de prioridad. Esto ayudará a que, si surge nueva información que provoque modificaciones en el sistema, estas se puedan realizar de tal forma que no existan riesgos de afectar otras partes del sistema.

El proceso unificado propone las siguientes cuatro fases:

- Concepción (inicio): Se presenta el modelo, visión, metas, requisitos del usuario, costos y viabilidad del proyecto. Esta fase además se enfoca y dedica más tiempo al modelo del negocio y los requisitos.
- Elaboración: Se revisa y se dá prioridad a los requisitos del usuario, se identifican los riegos que puede tener la implementación desarrollando estrategias para mitigarlos o eliminarlos y se diseña la arquitectura identificando los compontes y su interacción con otros.

  En esta etapa se dará un mayor enfoque al análisis y al diseño de software. Estas tareas son de total relevancia para la elaboración del sistema, pues nos indican el comportamiento que debe de tener.
- Construcción: En esta etapa se afinan detalles, como lo son los diferentes tipos de casos alternos y los riesgos menores, también se realizará la implementación del sistema. Una de las ventajas que se obtienen al seguir el modelo de Proceso Unificado es que se puede retroceder a la parte del diseño, de ser necesario, o a cualquier otra de las fases o, en su defecto, se pueden realizar varias iteraciones. Por otra parte, se realizan múltiples pruebas para garantizar el correcto funcionamiento del sistema.
- Transición: El sistema debe de estar listo para ser probado, instalado y utilizado por los clientes sin ningún problema.

  Siguiendo estas fases se garantizará un desarrollo efectivo y con el menor número de errores.

  El Proceso Unificado, tiene muchas ventajas al ser iterativo e incremental, como son:
    - Adaptabilidad para nuevos requisitos o cambios en estos
    - Poco riesgo de tener un sistema que no cumpla los requisitos

Con la metodología de software se podrán seleccionar las tecnologías para desarrollar y el lenguaje en el cual se elabora el sistema, así como el entorno de ejecución que satisfaga todo el proyecto.

Casos de uso

La elaboración de casos de uso constituye una técnica fundamental en la ingeniería de software utilizada para describir las interacciones entre un actor y el sistema informático. Estos proporcionan un enfoque estructurado para identificar, visualizar y comprender las diversas funcionalidades que el sistema debe ofrecer a sus usuarios. “La idea de caso de uso permite la identificación del software que cumple con los objetivos del usuario. Los casos de uso son las funciones que proporciona un sistema para añadir valor a los usuarios.” (Ivar Jacobson et al., s. f.) La identificación de los objetivos del software a través de los objetivos del usuario es crucial, tanto para el sistema como para el usuario. Al clarificar las ideas y objetivos del usuario, el caso de uso contribuye significativamente a la creación de un sistema de software más efectivo y alineado con las necesidades y expectativas del usuario.

Spring Boot Framework

La elección del Framework de Spring Boot para el desarrollo se basó en la formación académica recibida, la facilidad del lenguaje (Java) y la reducción en la carga de archivos de configuración, además de contar una mucha información de respaldo.

Java es un lenguaje orientado a objetos, el cual es muy versátil, por lo cual se pueden desarrollar aplicaciones móviles de Android, de escritorio, un Servicio Rest, API, sistemas web, etc. Esto permite que las aplicaciones sean multiplataforma sin necesidad de realizar alguna configuración especial, además de contar con un amplio catálogo de ayuda y documentación.

Actualmente, no basta con tener un leguaje sólido, además los desarrolladores deben tener herramientas, reglas y patrones de diseños que, junto al lenguaje, puedan tener una mejor interacción entre sí. A estas herramientas se les denomina Framework, aunque usar uno tiene desventajas como la curva de aprendizaje. Aun así, son mayores los beneficios al usar un Framework ya que agiliza el desarrollo y usa estándares probados.

Spring Framework fue creado en el 2003 por Rod Jhonson en búsqueda de solucionar la complejidad de las tecnologías más pesadas de esa época, con

técnicas como reducción de configuración y simplificación de desarrollo entre otras. Originalmente se basó en su libro "Expert One-on-One J2EE Design and Development". Nace de simplificar los archivos de configuración que tiene Spring y ofrece elementos de desarrollo facilitando el realizar ciertas tareas como:

1. Conexión fácil a la base de datos. Se tiene un archivo de configuración donde se ponen los parámetros para hacer la conexión, además no está ligado a un solo manejador de base de datos y la migración es muy limpia, basta con cambiar la dependencia del manejador. En comparación, los sistemas anteriores a este tienen que crear archivos muy largos donde se tiene que crear la conexión, hacer las consultas y cerrar la conexión.
2. No tiene tantos archivos de configuración. Para la elaboración de un proyecto de Spring en comparación Java EE, este tiene un número mayor de archivos de configuración, Spring simplifica la carga de estos archivos.
3. Spring simplifica el periodo de pruebas ya que al desarrollar todo en componentes es mucho más sencillo probar cada uno de estos.

Spring Boot tiene implementado una serie de patrones de diseño "Un patrón de diseño es un conjunto de estándares industriales probados y cuenta con las mejores prácticas para resolver problemas recurrentes. Los patrones de diseño no son listas de soluciones para usar; más bien son una plantilla para implementar y aplicar la mejor solución posible para un problema." (Patel & Patel, 2018). Los patrones de diseño no son soluciones aplicadas sino una serie de pasos probados para llegar a una solución. De igual manera, si no se sigue esa serie de pasos, se puede llegar a una mala solución. "Es igualmente cierto que, si un patrón de diseño no se implementa de la manera correcta, crea muchos problemas en lugar de resolver el que esperaba resolver."(Patel & Patel, 2018, p.4). Los patrones más conocidos que maneja Spring Boot son los siguientes:

- Vista controlador
- Singleton
- Observer
- Factory Method

- Inyección de Dependencia

Después se mencionará más detalladamente cada uno de ellos, dependiendo en qué fase del proceso unificado nos encontremos.

El utilizar un patrón de diseño tiene ventajas como las siguientes:

- Mejora la reutilización del software
- El ciclo de desarrollo se vuelve más rápido
- Simplifica la mantenibilidad de código
- Mejora la eficiencia y desarrollo en general del software

El patrón de diseño más importante y conocido es el de Inyección de Dependencias (DI), el cual ayuda a disminuir el acoplamiento entre los componentes dentro de una aplicación, sirve para proporcionar objetos dependientes de las clases que lo necesitan. Es decir, en lugar de que un objeto instancia busque sus dependencias, las dependencias son proporcionadas desde el exterior.

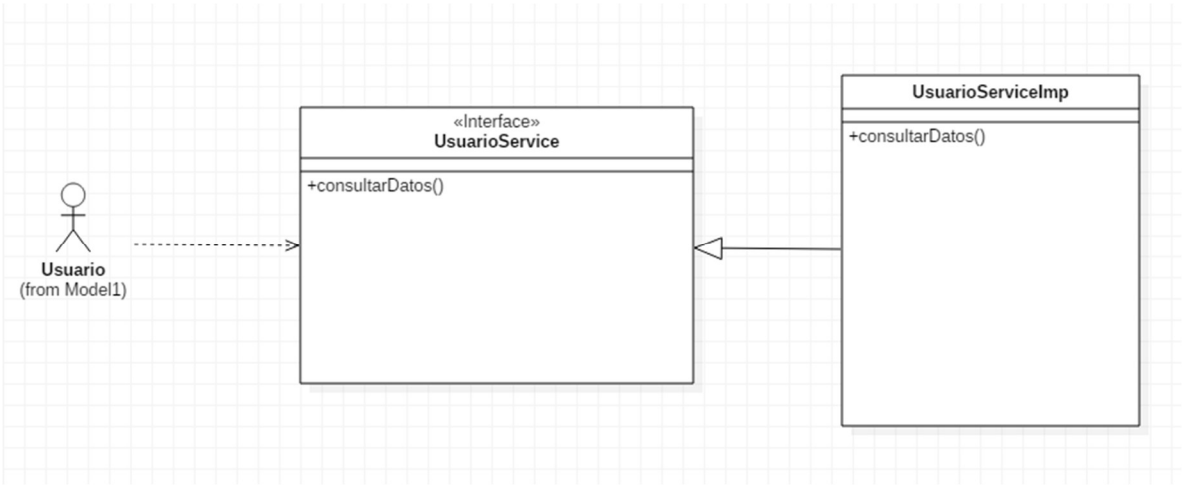


Imagen2.1 Modelo UML inyección de dependencias.

La utilización de interfaces crea una suerte de barrera que limita el conocimiento del método sobre la manera en que está siendo implementado internamente. Este enfoque contribuye a reducir el acoplamiento entre los diversos componentes del sistema, resultando en un código que es más mantenible y modificable a largo plazo. Como consecuencia, el sistema se adapta con mayor facilidad a los cambios tanto en las etapas de desarrollo como en las pruebas unitarias.

El mapeo objeto relacional

El mapeo objeto relacional como lo menciona Hinkula “El mapeo objeto relacional (ORM) es una técnica que le permite recuperar y manipular una base de datos utilizando un paradigma de programación orientada a objetos. ORM es realmente bueno para programadores porque se basa en conceptos orientados a objetos en lugar de bases de datos. También hace que el desarrollo sea mucho más rápido y reduce la cantidad de código fuente. ORM es en su mayor parte independiente de las bases de datos y los desarrolladores no tienen que preocuparse por declaraciones SQL específicas del proveedor.” (Patel & Patel, 2018). Aunque es cierto que se acorta el código durante el desarrollo también hay que tener en cuenta que el utilizar una técnica de mapeo adecuada requiere cierta experiencia y conocimiento de la estructura de la base de datos deseada.

Spring Boot proporciona anotaciones para que la configuración se haga de manera adecuada; estas son las más utilizadas:

- @Entity. Nos indica que la clase a utilizar será una entidad es decir una tabla en la base de datos.
- @Id. Establece que ese atributo será un identificador.
- @GeneratedValue. Esta es una estrategia para que los id sean consecutivos sin necesidad de estar agregando un contador.
- @OneToMany. Es una relación a nivel de base de datos. Nos indica que un objeto tiene de uno a muchos objetos.
- @ManyToMany. Indica una relación de muchos a muchos.
- @ManyToOne. Nos indica una relación de muchos a uno.
- @JoinColumn nos indica como se llamará la llave foránea en nuestra tabla.
- “*CascadeType.ALL*” Implica que todas las acciones llevadas a cabo con el objeto principal deben ser replicadas en sus objetos secundarios.
- "*CascadeType.PERSIST*" Indica que las operaciones de persistencia (como “guardar”) realizadas en la entidad principal se propagarán automáticamente a las entidades relacionadas.

- "*CascadeType.MERGE”* Indica que las operaciones de combinación (actualización) realizadas en la entidad principal se propagarán a las entidades relacionadas.

Herramientas de gestión de dependencias y construcción de proyectos

Las principales herramientas para gestionar el marco de trabajo de Spring Boot son Maven y Gradle, estas tecnologías están enfocadas en resolver cierto tipo de proyectos en específico. Maven se enfoca en proyectos muy grandes, ya que permite tener configuraciones más específicas con el servidor. Por otro lado, Gradle está más diseñado para proyectos más rápidos y móviles.

Utilizaremos Gradle en lugar de Maven debido a su versatilidad, “Gradle es una herramienta de automatización de compilación que simplifica y unifica el proceso de desarrollo de software. También gestiona las dependencias de nuestro proyecto y maneja el proceso de compilación.(Hinkula, 2023ª,p.4)”. Esta integración proporciona un mejor soporte para todas las dependencias y una gestión más eficiente del proyecto. Además, Gradle ofrece una construcción incremental superior; lo que implica que al compilar el proyecto solo se procesan los módulos modificados, lo que acelera el tiempo de desarrollo. Otra ventaja importante es que toda la configuración se realiza mediante un DSL (Lenguaje Específico del Dominio), lo que reduce la curva de aprendizaje en comparación con Maven, que utiliza XML.

Capítulo 3 Análisis de Requisitos

Información preliminar

Nuestro usuario principal, “Ernesto”, nos proporcionó varios formatos que contienen información crucial para las peticiones de reservaciones. Se convocaron varias reuniones con él para comprender por completo los pasos necesarios para solicitar una reservación. Estas reuniones fueron fundamentales para aclarar las reglas de negocio y las condiciones para obtener una visión detallada del proceso que él realiza diariamente como parte de sus responsabilidades laborales.

Como consecuencia del trabajo anteriormente mencionado se obtuvo el siguiente flujo principal para solicitar una reservación:

- El Profesor solicitante llena un formulario denominado “VALE DE APARTADO DE EQUIPO DE CÓMPUTO, AUDIO Y VIDEO”.

UACM Universidad Autónoma de la Ciudad de México

Nada humano me es ajeno

ENLACE ADMINISTRATIVO PLANTEL SAN LORENZO TEZONCO

VALE DE APARTADO DE EQUIPO DE COMPUTO, AUDIO Y VIDEO

NOMBRE DEL SOLICITANTE: **FECHA:**

COLEGIO, ACADEMIA Ó AREA DEL SOLICITANTE:

EQUIPO SOLICITADO:

FECHA (S) A REALIZAR EL EVENTO:			
HORARIO (S) DEL EVENTO:			
No. DE PERSONAS EN EL EVENTO:		**AULA (S) ASIGNADA:**	

EXT.

FIRMA DEL SOLICITANTE **FIRMA DE AUTORIZACIÓN**

NOTA: ESTA SOLICITUD, SERA VALIDA SI NO PRESENTA TACHADURA NI ENMENDEDURAS

EL SOLICITANTE TENDRA SOLAMENTE 15 MIN. DE TOLERANCIA PARA PODER LLEGAR A SU EVENTO Y EN CASO DE NO SER ASÍ, SE CERRARA LA AULA HASTA QUE LLEGUE EL SOLICITANTE, POR LO CUAL TENDRA QUE RESPETAR EL HORARIO ASIGNADO Y NO TENDRA MAS TIEMPO, YA QUE EL AULA SE OCUPARA DESPÚES Y YA ESTA ASIGNADA A OTRO EVENTO.

LA ENTREGA DEL EQUIPO , IMPLICA QUE ESTE DEBERA SER DEVUELTO EN LAS MISMAS CONDICIONES EN QUE LO RECIBE, SI SE PRESENTA ALGUN ANOMALIA DEBERA NOTIFICARLO INMEDIATAMENTE, DE LO CONTRARIO SE ASUMIRA QUE LA MISMA FUE PROVOCADA POR EL SOLICITANTE.

Imagen 3.1 VALE DE APARTADO DE EQUIPO DE CÓMPUTO, AUDIO Y VIDEO

- El encargado verifica en un archivo de Excel denominado “REPORTE DE EVENTOS Y MATERIAL REQUERIDO” si tiene libre algún sitio que cumpla con la solicitud, en caso de ser así, se le aparta el lugar deseado y se guardan los datos de la solicitud en este archivo.

Universidad Autónoma de la Ciudad de México UACM

Nada humano me es ajeno

NOVIEMBRE .2016

REPORTE DE EVENTOS Y MATERIAL REQUERIDO

FECHA	HORA	LUGAR	SOLICITANTE		EVENTO	MATERIAL EN PRESTAMO
			AREA	NOMBRE		
03-nov-16	8:30-11:30	B-216	ESTUDIOS SOCIALES E HISTORICOS		CLASE	CAÑON-CPU
03-nov-16	10:00-11:30	A-206	CPYAU		CLASE	CAÑON-CPU
03-nov-16	13:00-14:30	A-215	CPYAU		CLASE	CAÑON-CPU
03-nov-16	16:00-17:30	A-215	CPYAU		CLASE	CAÑON-CPU
03-nov-16	8:30-11:30	B-307	LENGUAJE Y PENSAMIENTO		CLASE	CAÑON-CPU
03-nov-16	11:30-13:00	B-307	LENGUAJE Y PENSAMIENTO		CLASE	CAÑON-CPU
03-nov-16	8:30-11:30	A-215	CREACION LITERARIA		CLASE	CAÑON-CPU
03-nov-16	11:30-14:30	B-216	COMUNICACIONY CULTURA		CLASE	CAÑON-CPU
03-nov-16	16:00-19:00	B-216	COMUNICACIONY CULTURA		CLASE	CAÑON-CPU
03-nov-16	10:00-13:00	C-216	PROMOCION DE LA SALUD		CLASE	CAÑON-CPU-DVD
03-nov-16	13:00-14:30	C-216	PROMOCION DE LA SALUD		CLASE	CAÑON-CPU-DVD
03-nov-16	11:30-14:30	C-207	ESYH		CLASE	CAÑON-CPU-TV-DVD
03-nov-16	8:30-10:00	C-207	ARTE Y PATRIMONIO		CLASE	CAÑON-CPU
03-nov-16	10:00-11:30	B-115	ARTE Y PATRIMONIO		CLASE	CAÑON-CPU
03-nov-16	16:00-19:00	C-216	BIOLOGIA		CLASE	CAÑON-CPU
03-nov-16	7:00-10:00	C-107	HISTORIA Y SOCIEDAD CONTEMPORANEA		CLASE	CAÑON-CPU
03-nov-16	17:30-19:00	C-207	HISTORIA Y SOCIEDAD CONTEMPORANEA		CLASE	CAÑON-CPU
03-nov-16	16:00-17:30	A-206	PROMOCION DE LA SALUD		CLASE	CAÑON-CPU
03-nov-16	8:30-10:00	C-216			CLASE	CAÑON
03-nov-16	17:30-19:00	C-012			CLASE	CAÑON-CPU
03-nov-16	11:30-14:30	A-206			CLASE	CAÑON-CPU
03-nov-16	13:00-16:00	B-115			CLASE	CAÑON-CPU
03-nov-16	11:30-13:00	C-012			CLASE	CAÑON
03-nov-16	11:30-13:00	B-115			CLASE	CAÑON-CPU
03-nov-16	8:30-11:00	B-101			CLASE	CAÑON-CPU
03-nov-16	9:00-11:00	A-201			CLASE	DVD-TV
03-nov-16	16:00-19:00	A-201			CLASE	DVD-TV
03-nov-16	13:00-15:00	A-201			CLASE	TV-DVD
03-nov-16	15:0-17:00	B-101			CLASE	DVD-TV
03-nov-16	11:30-16:00	C-101	PENSIONISSTE		PLATICA	SOLO ESPACIO
03-nov-16	17:30-20:30	A-206			CLASE	CAÑON-CPU-DVD-TV
03-nov-16	16:00-17:30	B-115			CLASE	DVD-TV
03-nov-16	16:00-17.30	C-107			CLASE	CPU-CAÑON
04-nov-16	11:30-14:30	C-101	CIENCIAS SOCIALES		CLASE	CAÑON-CPU-DVD

Imagen 3.2 REPORTE DE EVENTOS Y MATERIA REQUERIDO con datos reales.

Además de llevar a cabo el proceso de registro de papeletas para reservaciones se cuentan con escritos sellados y firmados por las autoridades de la universidad, en los cuales se solicita la reservación de un sitio por un periodo que puede extenderse a más de un mes. Estos documentos son enviados con anticipación antes del inicio del semestre, asegurando la disponibilidad de los espacios requeridos cuando comienzan las clases. En los documentos aquí presentados, para preservar la confidencialidad de los datos personales, se han eliminado los nombres del solicitante y del enlace administrativo.

UACM Universidad Autónoma de la Ciudad de México

Nada humano me es ajeno

ENLACE ADMINISTRATIVO
PLANTEL SAN LORENZO TEZONCO

Oficio No. UACM/ADMON/SLT/O-007/22
CDMX., a 10 de Enero de 2022.

Academia de Comunicación y Cultura
Plantel San Lorenzo Tezonco
Presente

En respuesta a su oficio de fecha **10** de **diciembre** de 2021, me permito informarle que ya fueron programados los espacios, mobiliario y materiales solicitados por usted, para el apoyo a los cursos que se impartirán durante el semestre 2022-I en las siguientes fechas.

Fechas	Horarios	Aulas asignadas
Enero		
25,27	8:30-11.30 Hrs.	A-201
Febrero		
1,8,15,22	8:30-11.30 Hrs.	A-201
3,10,17,24	8:30-11.30 Hrs.	A-201
Marzo		
1,8,15,22,29	8:30-11.30 Hrs.	A-201
3,10,17,24,31	8:30-11.30 Hrs.	A-201
Abril		
5,19,26	8:30-11.30 Hrs.	A-201
7,21,28	8:30-11.30 Hrs.	A-201
Mayo		
3,17,24	8:30-11.30 Hrs.	A-201
5,12,19,26	8:30-11.30 Hrs.	A-201

Sin otro particular, aprovecho para enviarle un cordial saludo

Atentamente

Enlace Administrativo
Plantel S.L.T

C.c.p. ..... AHJ/lcob

Prolongación San Isidro No.151, Col. San Lorenzo T. C.P. 09790 Del. Iztapalapa, México D.F.
Tel. 58 50 19 01 Ext. 13120, 13121, 13122, 13125, 13128 Fax. 30 92 05 74

Imagen 3.3 del formato de escrito para reservar sitios

El flujo principal del sistema que desarrollamos se centra en el proceso de reservación con la papeleta, el cual es fundamental para entender la experiencia de un usuario promedio al realizar esta acción. Aunque los escritos sellados y firmados por las autoridades de la universidad también son elementos importantes, la papeleta proporciona un panorama más detallado del proceso, brindando información valiosa sobre la perspectiva y las necesidades del usuario promedio durante la realización de una reservación.

La agenda, gestionada a través de un archivo en Excel, sirve como repositorio de información para todas las reservaciones. Este método, sin embargo, presenta un riesgo potencial de traslape, ya que el administrador realiza una revisión visual de la información, lo que podría resultar en la omisión involuntaria de algunas reservaciones, fenómeno que ya ha ocurrido. Además, cabe destacar que este formato se imprime diariamente, permitiendo al responsable de la apertura de espacios verificar quién ha realizado la reservación, proporcionar el equipo solicitado y supervisar la duración de la reservación.

Además de los formatos mencionados, se tuvo acceso a una serie de informes que la administración envía a las autoridades. Estos informes comprenden una recopilación de datos estadísticos sobre los sitios y el número de reservaciones. Los aspectos analizados incluyen:

- Reporte por Edificios
- Reporte por Eventos
- Reporte de Trabajo por día
- Reporte de Actividades

Cada uno de los reportes presentan información detallada y específica, proporcionando un análisis exhaustivo de los datos relacionados con los sitios y el número de reservaciones. Estos informes se han convertido en herramientas clave para comprender en profundidad el rendimiento y la utilización de los espacios,

contribuyendo así a la toma de decisiones por parte de las autoridades administrativas de la universidad.

Con relación al Reporte de trabajo por día, éste maneja toda la información de las reservaciones que se realizaron en un día, no importando la fecha en las que realizarán los eventos.

El Reporte por edificios proporciona el número de eventos realizados en cada sitio, con un enfoque particular en los edificios A, B y C, así como en el Umbral, el Domo, el Ágora y otros.

El Reporte de eventos muestra la cuenta de las reservaciones que se hicieron con un tipo específico de evento, por ejemplo, clases, titulación, Consejo Universitario, etc.

El Reporte de actividades plantea todas las actividades hechas, es decir; clases, titulaciones, entre otros; por el periodo de un semestre.

Actores

“No todos los actores representan a las personas. Pueden ser otros sistemas o hardware externo que interactúan con el sistema. Cada actor asume un conjunto coherente de papeles cuando interactúa con un sistema. Un usuario físico puede actuar como uno o varios actores, desempeñando los papeles de esos actores en su interacción con el sistema.” (Ivar Jacobson et al., s. f.), Los actores son individuos o sistemas externos que interactúan con el sistema. En nuestro caso, los actores identificados son:

- Administrador (“Ernesto”), encargado de la administración de los sitios de la UACM San Lorenzo Tezonco.
- Profesor de la UACM este puede ser un profesor de asignatura o de tiempo completo.
- Agente externo, estas son instituciones públicas o privadas que no son parte de la universidad, pero requieren un espacio dentro las instalaciones de la universidad para la realización de alguna actividad, por ejemplo: campañas de vacunación por parte de la Secretaría de Salud, capacitaciones del INE, entre otros.

Flujo principal

Cuando un profesor de la UACM desea organizar un evento en algún sitio, completa el 'Vale de Apartado de Equipo de Cómputo, Audio y Video' proporcionando sus datos. Posteriormente, el administrador verifica la disponibilidad de un sitio que cumpla con las especificaciones requeridas por el usuario y en caso de ser así, se anota en ‘Reporte De Eventos Y Material Requerido’.

Reglas de negocio

Las reglas de negocio constituyen los pilares fundamentales que definen el comportamiento, las restricciones y las políticas que rigen una organización. En nuestro sistema, las reglas de negocio identificadas son:

- **RN1.-** Toda la información guardada en la base de datos será almacenada con letras mayúsculas.
- **RN2**.- Las reservaciones se tienen que hacer con 3 días de anticipación, si no se deberá producir un mensaje de error para el usuario Profesor.
- **RN3**.- Sólo el Administrador puede registrar fechas anteriores al día actual.
- **RN4.-** El Administrador es el único usuario que puede eliminar bloques de horario.

Requisitos funcionales

“Los requisitos funcionales capturan las características de comportamiento deseadas centrales del sistema de software.” (Walkinshaw, 2017, p. 52). Como lo menciona Walkinshaw, son características del sistema, es decir, su funcionalidad.

Requisito	Descripción
R1	Usuarios • El sistema tendrá un perfil de Administrador y uno de Profesor. • El sistema permitirá crear nuevos usuarios con diferentes tipos de

	acceso a la información almacenada en el mismo. • El sistema identificará a los usuarios Profesor, por medio de su correo institucional. • El sistema mandará un correo electrónico con una contraseña provisional, una vez que se haya registrado un Profesor. • El sistema permitirá cambiar la información personal del Profesor: esto solo puede realizarlo el Administrador. • Una vez hecho el registro de usuario se enviará un correo electrónico con su contraseña.
R2	Sitios • El sistema deberá almacenar sitios que están a disposición para la realización de eventos: clases, salones con proyectores, eventos culturales, eventos escolares, eventos gubernamentales etc. Estos solo los puede registrar el Administrador. • El sistema permitirá dar de baja los sitios. Esto solo lo puede realizar el administrador. • El sistema permitirá Modificar los sitios: capacidad, descripción, etc. Esto solo los puede realizar el administrador
R3	Reservaciones • El sistema permitirá crear nuevas reservaciones. • El sistema permitirá modificar una reservación, siempre y cuando aún no se esté en la fecha de realización y solo la podrá realizar el Administrador. • El sistema permitirá eliminar reservaciones y solo las podrá realizar el Administrador. • El sistema enviará correos electrónicos al Profesor y Administrador en caso de que se haya cambiado algo que afecte la reservación.

R4	Equipos • El sistema permitirá guardar el nombre y la descripción de los equipos disponibles en préstamo para la realización de los eventos. Esto solo lo puede realizar el Administrador. • El sistema permitirá modificar la descripción de los equipos. Esto solo lo puede realizar el Administrador. • El sistema permitirá eliminar equipos. Esto solo lo puede realizar el Administrador.
R5	Reportes • El sistema permitirá crear reportes diarios. Esto solo lo puede realizar el Administrador. • El sistema permitirá crear reportes de los eventos que se realizaron en una fecha determinada, con la información de cada evento. Esto solo lo puede realizar el Administrador. • El sistema permitirá crear reportes semestrales: por tipo de evento, sitio, sede y por total de eventos realizados. Esto solo lo puede realizar el Administrador.
R6	Sedes • El sistema permitirá dar de alta una nueva sede. Esto solo lo puede realizar el Administrador. • El sistema permitirá modificar una sede. Esto solo lo puede realizar el Administrador. • El sistema permitirá dar de baja una sede. Esto solo lo puede realizar el Administrador.
R7	Eventos • El sistema permitirá dar de alta un nuevo evento. Esto solo lo

	puede realizar el Administrador. • El sistema permitirá modificar un evento. Esto solo lo puede realizar el Administrador. • El sistema permitirá dar de baja un evento. Esto solo lo puede realizar el Administrador.

Requisitos no funcionales

"Nos referiremos a un requisito no funcional simplemente como una característica de naturaleza cualitativa." (Walkinshaw, 2017, p. 52). Como lo menciona Walkinshaw, los requisitos no funcionales son una característica de naturaleza cualitativa. Estos no se refieren directamente a las funciones que el sistema debe realizar, sino a las características que definen la calidad y el rendimiento del sistema.

Requisito	Descripción
Rendimiento	Se emplearán las dependencias más actualizadas compatibles con el servidor de aplicaciones, con el objetivo de optimizar el rendimiento del sistema.
Usabilidad	El sistema ha sido diseñado con el objetivo de ofrecer una experiencia de usuario intuitiva y fácil de usar. Se llevarán a cabo pruebas de aceptación para garantizar que cumpla con las expectativas y necesidades del usuario final.
Seguridad	Al requerir el uso del correo institucional, se prevé que únicamente los profesores y el administrador tengan acceso al sistema. Además, que todas las vistas estarán protegidas para que no se permita el ingreso al sistema a estudiantes o gente externa a la universidad.

Mantenibilidad	El sistema está construido aplicando principios de diseño que permiten extender una funcionalidad sin modificar la que ya existe. Además, tiene bastante documentación para que el mantenimiento y posibles actualizaciones sean lo más transparentes posibles. Así mismo, el código cuenta con los suficientes comentarios para que los desarrolladores sigan realizándole posibles cambios.
Adaptabilidad	Al ser una aplicación web ésta llegará a un mayor número de dispositivos los cuales son los siguientes: PC, celulares, tabletas. Estos, al tener un navegador, podrán acceder al sistema sin ningún problema, ya que este es adaptable a estos dispositivos.
Portabilidad	El sistema al ser desarrollado en Java permitirá usar los archivos JAR y WAR lo que permite instalar el ejecutable en cualquier sistema operativo.

Diagrama de Casos de Uso

“Un diagrama de casos de uso describe parte del modelo de casos de uso y muestra un conjunto de casos de uso y actores con una asociación entre cada par actor/caso de uso que interactúan.” (Ivar Jacobson et al., s. f.). En el diagrama de casos de uso del sistema, se han identificado cuatro casos principales: Gestión de catálogos (Gestión de sitios, equipos, áreas administrativas, eventos), Gestión de reportes, Gestión de usuarios y Gestión de reservaciones. Cada uno de estos casos se ha diseñado de manera concisa, abarcando operaciones de altas, bajas y cambios en sus respectivas áreas de gestión. Cabe destacar que el usuario Agente Externo no está incluido en el sistema, ya que se espera que las reservaciones sean manejadas por el administrador.

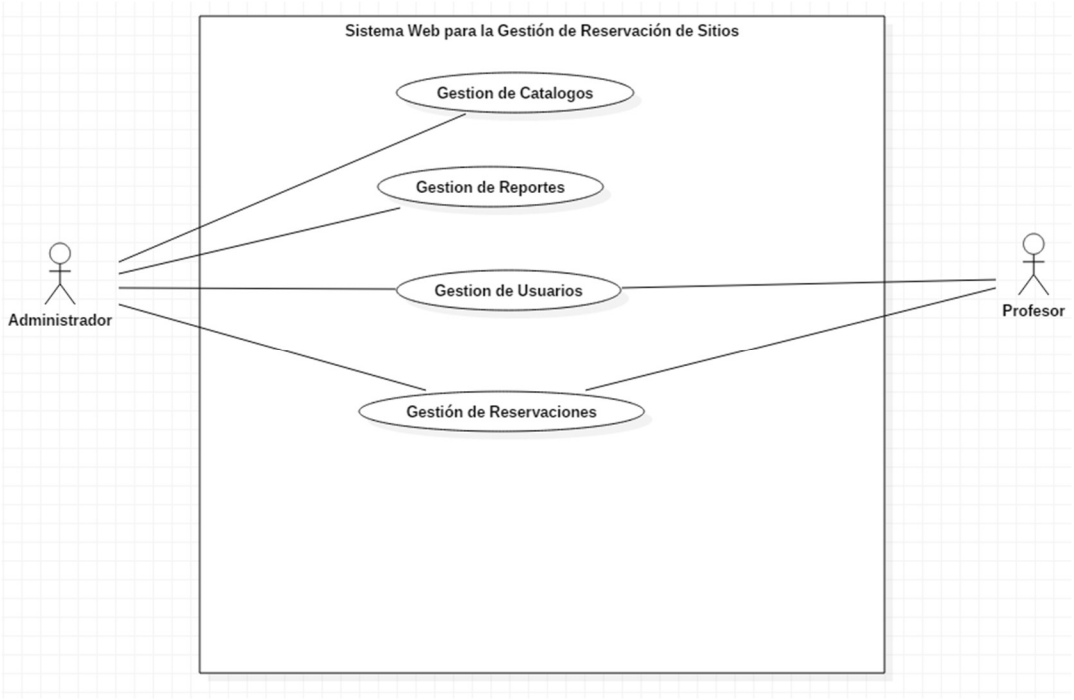


Imagen 3.4 Diagrama de caso de uso

Casos de uso

Como se mencionó anteriormente los casos de usos son un pilar para el proceso unificado. "Los casos de uso han sido adoptados universalmente para capturar requisitos de sistemas de software en general, y de sistemas basados en componentes en particular, pero los casos de uso son mucho más que una herramienta para capturar requisitos. Dirigen el proceso de desarrollo en su totalidad. Los casos de uso son la entrada fundamental cuando se identifican y especifican clases, subsistemas e interfaces, cuando se identifican y especifican casos de prueba y cuando se planifican las interacciones del desarrollo y la integración del sistema." (Ivar Jacobson et al., s. f.). Entonces se tiene que llevar a cabo una buena interpretación de estos. Los casos de uso son plantillas que nos permiten capturar la información relevante; que posteriormente se utilizarán para todas las demás fases.

En la primera iteración correspondiente al caso de uso de Gestión de usuarios, se consideró la posibilidad de aplicar una API o servicio REST para obtener la información de los profesores. El objetivo era evitar que los profesores completaran un formulario, ya que la información de éstos está disponible en el sistema de

profesores. Sin embargo, en una segunda iteración, al no lograr establecer comunicación con el área encargada de Sistemas, se propuso la alternativa de implementar un registro para recolectar la información necesaria de los profesores. A lo largo de las distintas iteraciones con el usuario, las plantillas de casos de uso fueron modificándose, estas alteraciones no tuvieron un impacto significativo en el sistema durante todo el desarrollo.

De los cuatro casos de uso que se identificaron, la prioridad más alta es “Gestión de Reservaciones” ya que representa el flujo principal del sistema. Esto fue con el fin de asegurarme de que este caso de uso no presente ninguna falla o bug, al ser crucial para el correcto funcionamiento del sistema. “Los mejores casos de uso son aquellos que añaden valor al negocio que implanta el sistema.” (Ivar Jacobson et al., s. f.). Si bien nuestro caso de uso se destaca como nuestra función principal, es importante destacar que esto no implica que los otros casos de uso carezcan de importancia; cada uno de ellos desempeña un papel significativo en el conjunto del sistema.

Caso de uso Gestión de Reservaciones

Aquí se desglosará el caso de uso principal “Gestión de Reservaciones”, detallando todos los aspectos y cambios que han surgido a lo largo del proceso.
Los actores principales en este caso de uso son el Administrador y el Profesor, ambos capacitados para llevar a cabo este proceso. Sin embargo, es importante destacar que cada uno opera bajo reglas de negocio distintas, las cuales mencionaremos a medida que avancemos. Durante todas las iteraciones del proceso unificado los requisitos fueron cambiando para este caso de uso.

Información general para este caso de uso.

Precondiciones.

El usuario deberá acceder con su correo institucional al sistema de Reservación de Sitios. Para cumplir con esta regla de negocio, se consideró utilizar el sistema de profesores como una API o servicio REST, para obtener la información básica de los

profesores. Sin embargo, como mencionamos anteriormente, no pudimos contar con el apoyo del área de Sistemas para implementar esta solución. En consecuencia, en el caso de uso de Gestión de usuarios, se diseñó un formulario de registro que solicita el correo institucional del profesor, donde se enviará una contraseña provisional.

El usuario deberá seleccionar del menú la opción de Reservaciones.
El Profesor o el Administrador disponen de esta funcionalidad. Dependiendo del flujo requerido por el usuario, seleccionarán algunas de las opciones correspondientes.

Postcondiciones del caso de uso Gestión de Reservaciones

Se mandará un correo electrónico con la información de la reservación hecha o eliminada. El sistema está configurado para enviar notificaciones por correo electrónico al usuario afectado ante el más mínimo cambio en las reservaciones.

Restricciones / Problemas / Riesgos

Si alguna organización de gobierno o educativa ajena a la UACM requiere de un sitio para la realización de eventos; se tendrá que solicitar directamente con el administrador del sistema. Esta restricción considera que, al no tener un correo electrónico institucional, no puede registrarse; por lo que el administrador tendrá que registrarlo.

Eventos desencadenantes

Los eventos desencadenantes en los casos de uso son las situaciones que inician la ejecución del caso de uso principal.

Eventos Internos

- El usuario o administrador requiere Reservación.

Eventos Externos

- Un usuario requiere de un sitio específico para la realización de un evento cultural o académico.

Listado de cursos de los casos de uso

Nueva Reservación - Usuario Profesor. Este será nuestro flujo principal, ya que es el más utilizado, aunque puede variar según las reglas de negocio.

Nueva Reservación - Usuario Administrador. Aunque es muy similar al flujo principal, presenta variaciones diseñadas para facilitar la tarea al administrador.

Eliminar Bloque de Horario - Usuario Administrador. Curso alternativo destinado a la eliminación de bloques de horarios por parte del administrador.

Mis Reservaciones - Curso alternativo donde los usuarios pueden revisar sus reservaciones realizadas, pendientes y aquellas a las que no asistieron.

Disponibilidad - Curso alternativo que permite a los usuarios verificar la disponibilidad de bloques de horarios para su uso.

Curso de excepción

Registro o eliminación de una reservación para un órgano de gobierno, institución externa de la universidad u otra dependencia, será atendida por el administrador. Como se mencionó al no tener un correo electrónico institucional no podrán ingresar al sistema.

Cursos detallados

Nueva reservación usuario Profesor. Para realizar una nueva reservación el profesor accede al menú correspondiente, en el cual el sistema despliega un formulario. En este formulario, el profesor podrá ingresará las fechas deseadas, seleccionará el sitio de los cual solo tiene acceso a las “aulas” en caso de que solicite un espacio tendrá que hacer la reservación mediante el Administrador, después especificará el número de asistentes y solicitará material a préstamo. Una vez que el profesor completa estos campos, presionara el botón de búsqueda. El sistema responderá mostrando una serie de bloques de horarios, marcados en verde para indicar disponibilidad y en

rojo para indicar ocupación. El profesor seleccionará los bloques deseados y, al finalizar, presionará el botón 'Reservar' para confirmar la reservación.

3.1 Curso Principal Nueva Reservación Usuario Profesor

#	Evento desencadenante	Respuesta del sistema
P1	El usuario a selecciona del menú Reservaciones “Nueva reservación”	Muestra un formulario con los siguientes campos: • Fechas de registro • Sitio • Número de asistentes • Equipos a préstamo
P2	Llenar el formulario y presiona el botón de “Buscar”	**Verificación de RN2 y RN3.** En caso de no haber errores el sistema muestra una serie de botones que tienen la hora. Si están color rojo quiere decir que esos horarios están ocupados, si están en verde, están disponibles. Así mismo, están divididos por días, de acuerdo con los que seleccionó el usuario.
P3	El usuario selecciona las horas donde desea realizar su reservación. Una vez finalizada la opción, presiona el botón de “Reservar”.	El sistema deberá mostrar un mensaje de confirmación de la reservación exitosa, junto con todos los detalles de esta. Esto incluye: • Usuario • Sitio • Evento • Número de asistentes • Fecha de solicitud • Comentario • Fecha y horarios solicitados

Nuestro curso principal se fue puliendo a lo largo de toda la fase del proceso de desarrollo, considerando diversas ideas, conceptos e interfaces todo en conjunto con

nuestro usuario principal. Al final, se encontró el curso que mejor se alineaba con las necesidades del usuario. Como podemos observar, se dispone de un formulario que recoge los datos necesarios para realizar una reservación. En un paso posterior, se muestran los lugares disponibles, permitiendo al usuario seleccionar entre las opciones de disponibilidad. Para consultar más flujos, estarán disponibles en el anexo 1.

Reglas de negocio utilizadas en este caso de uso con ejemplos concretos

RN1.- Toda la información guardada en la base de datos será guardada con letras mayúsculas, al completar toda la información de las reservaciones

RN2.- En caso de que las fechas a reservar estén próximas, a no más de 3 días del día actual, saldrá un mensaje de error. Esto quiere decir que, si nos encontramos en un 15 septiembre, el usuario no podrá reservar si no a partir del día 18.

RN3.- Solo el administrador puede registrar fechas anteriores al día actual. Esto se permite para que el Administrador pueda tener todo su histórico de las reservaciones anteriores que no alcanzo a registrar a tiempo. Por ejemplo, podrá registrar una reservación en el pasado para el 10-10-2020 sin problema.

RN4.- El administrador es el único que puede eliminar algún bloque de horario. Para evitar que los profesores estuvieran apartando y desapartando lugares se planteó que solo el administrador pueda dar de bajar estos registros.

Restricciones de sincronización

En caso de que dos usuarios necesiten el mismo espacio simultáneamente, se dará prioridad al usuario que haya ingresado primero. El otro usuario deberá esperar hasta que el espacio esté disponible nuevamente. La sincronización tiene una restricción de 2 minutos para terminar una reservación.

Información táctica del caso de uso

Prioridad: Alta. Se refiere con una prioridad alta a la importancia o urgencia que se asigna a lo largo del ciclo de vida de software.

Objetivo (s) de rendimiento

Caso de uso	Estímulo	Tiempo de respuesta esperado
Nueva Reservación	Nueva Reservación	1 segundo
Eliminar Reservación	Eliminar una Reservación	1 segundo
Disponibilidad	Ver disponibilidad	1 segundo
Mis reservaciones	Ver mis reservaciones	1 segundo

Para lograr que se establezcan estos objetivos de rendimiento, es necesario tener en cuenta dónde se alojará la aplicación y la velocidad de internet del servidor. Se llevarán a cabo pruebas exhaustivas para verificar si los objetivos se cumplen tanto a nivel de código como en la infraestructura del servidor donde se despliegue la aplicación.

Frecuencia

En la siguiente tabla se refiere a qué tan frecuentemente se ejecuta cada módulo.

Caso de uso	Frecuencia
Nueva reservación	Alta
Disponibilidad	Media alta
Eliminar Reservación	Baja
Mis reservaciones	Baja

Interfaz de usuario

Durante el levantamiento de requisitos se pensaron en diversas propuestas para la interfaz de usuario, la primera retomando la interface del sistema de escritorio, pero al no ser tan amigable con el usuario se pensaron en opciones llegando a la que se mostraran a continuación.

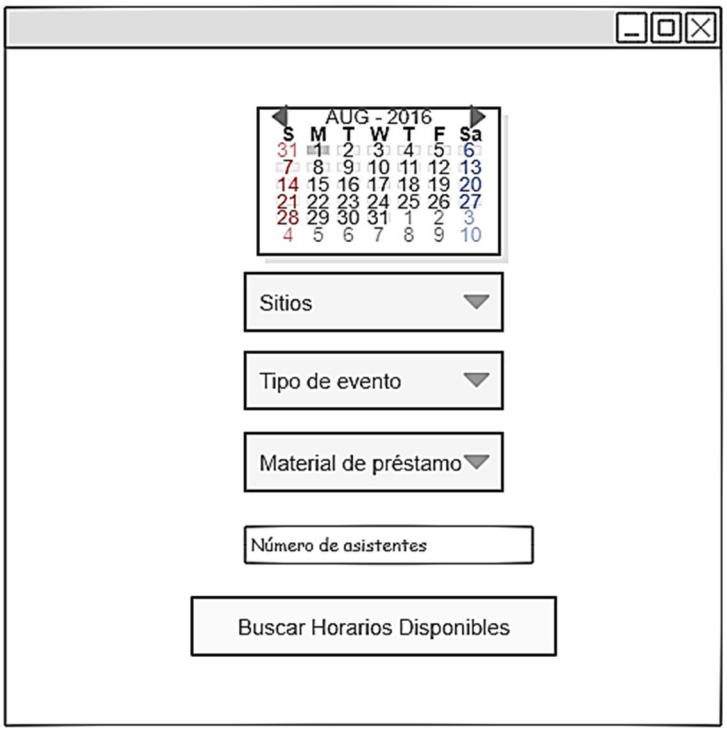


Imagen 3.5 Proceso de reservación en la primera fase de la reservación

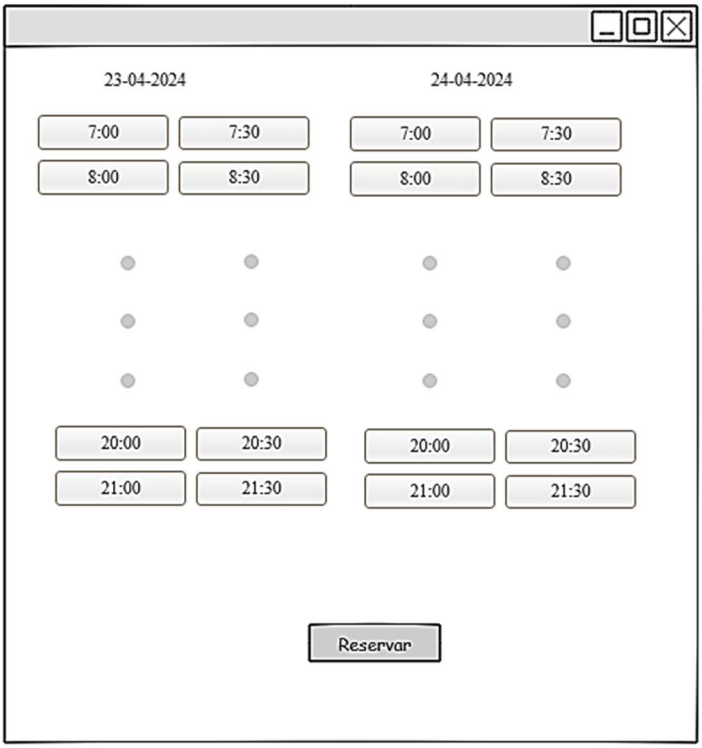


Imagen 3.6 Proceso de reservación en la fase de selección de horarios

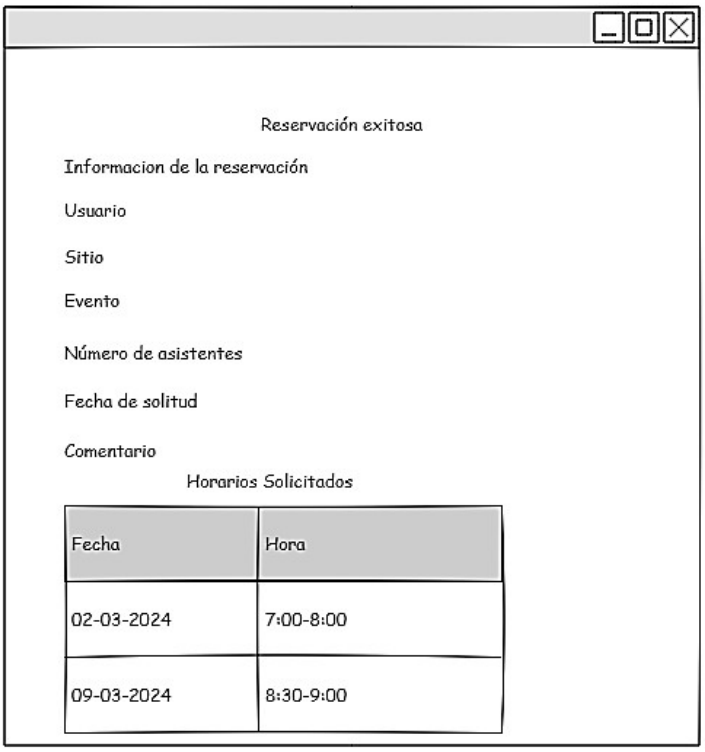


Imagen 3.7 Proceso de retroalimentación para el usuario una vez terminó su reservación.

Capítulo 4 Análisis

En este capítulo, se aborda la importancia de la elaboración del modelo a partir del análisis de requisitos, con el objetivo de mejorar las funcionalidades ya sistematizadas en la versión de escritorio; al mismo tiempo que se identificaran algunas nuevas funciones que mejoran el proceso al solicitar una reservación, ahora vía web.

Antes de comenzar el análisis, es crucial entender en qué consiste este proceso y no confundirlo con el diseño, ya que ambos están estrechamente relacionados, pero tienen objetivos distintos. El análisis se centra en comprender y documentar los requisitos mediante diagramas, asegurando que todas las necesidades y expectativas del sistema estén claramente definidas en un modelo generalmente construido en UML. Por otra parte, el diseño se enfoca en cómo el sistema cumplirá con esos requisitos, detallando la arquitectura y los componentes necesarios para implementar las funcionalidades identificadas durante el análisis.

Diagrama de Objetos

En este diagrama es la primera cercanía con los casos de uso con UML, se parte de tomar los casos de usos y obtener todos los sustantivos, ver cuál de ellos se repiten, si son sinónimos unos de otros ya que aquellos que son mayormente repetidos serán nuestras posibles clases o atributos en el diagrama de clases. También se trata de ver las posibles relaciones que tienen unos objetos con otros. “ Los objetos son las entidades básicas del modelo de objetos”(Weitzenfeld, s. f., p. 71).

Como se muestra en el siguiente diagrama los objetos son los círculos con una raya en la parte inferior, el controlador gestión horarios éste está representado por un círculo con flecha y la pantalla con la cual interactúa el usuario que se representa con un círculo con una ‘T’ invertida de lado izquierdo.

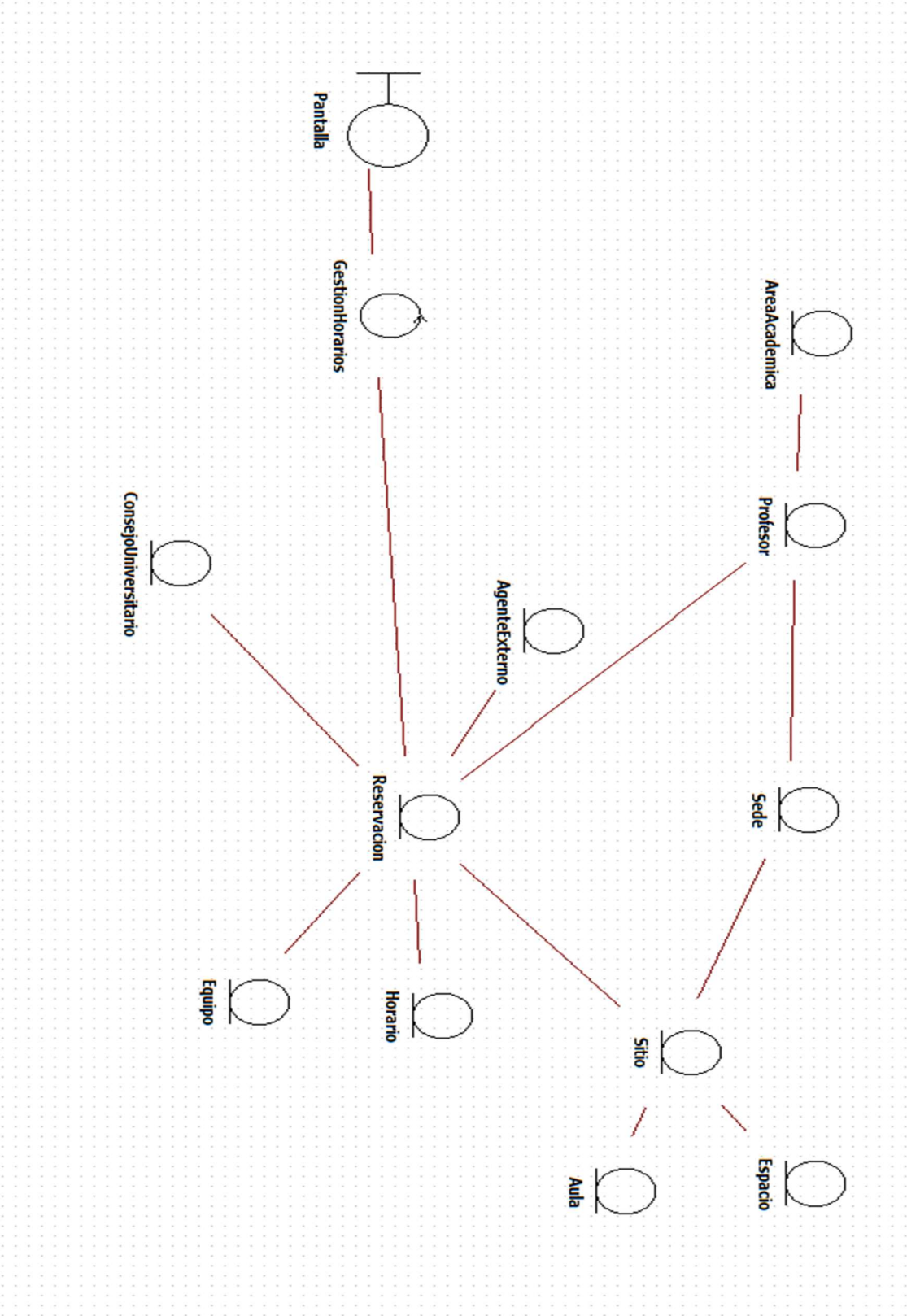


Imagen 4.1 Diagrama de objetos

Diagrama de secuencia

Se han pensado otras formas de optimizar este proceso para evitar la validación repetitiva de la disponibilidad de horarios. Se consideraron varias soluciones para realizar el proceso de reservación, muchas de ellas inspiradas en ejemplos de la vida cotidiana como una agenda o un calendario escolar. Sin embargo, en cada una de estas soluciones era necesario validar continuamente las disponibilidades de horarios. Observando otros sistemas con funcionalidades similares, se encontró el sistema de reservación de asientos de cine, al usuario se le muestran las distintas películas que se tienen en cartelera en nuestro caso serían los sitios y los asientos serian nuestros horarios, siguiendo esta misma lógica se evita la validación excesiva de los distintos horarios.

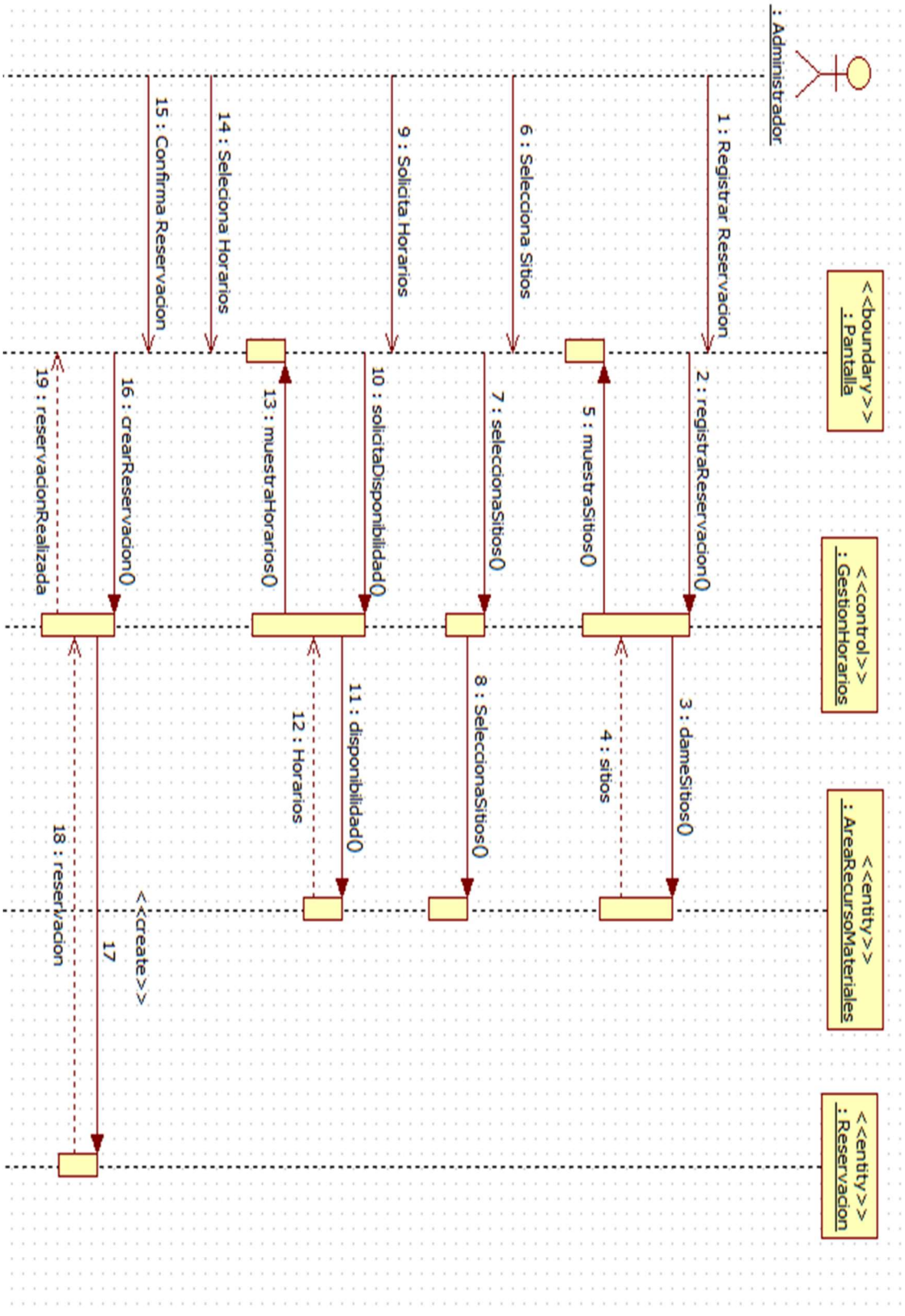


Imagen 4.2 Diagrama de secuencia realizar una reservación

Diagrama de clases

El presente diagrama muestra las relaciones entre las clases. La multiplicidad indica el número de instancias asociadas a otra instancia. Las agregaciones son relaciones donde las partes no dependen del todo y pueden existir independientemente del objeto que las contiene. En cambio, las composiciones son relaciones en las que las partes no pueden existir sin el objeto que las contiene.

Tomando en cuenta que la clase principal es la "Reservación", la cual está asociada con casi todas nuestras demás clases. Por lo tanto, es muy importante verificar que las asociaciones y agregaciones estén correctamente definidas. Esto garantizará que, al borrar un objeto de alguna clase, la operación se realice adecuadamente sin perder información en el proceso. Por ejemplo, los usuarios pueden tener una reservación, pero al borrar la reservación, los usuarios deben seguir existiendo. La única clase que tiene una relación de composición con "Reservación" es "Horario", ya que, si se borra la reservación también se deben liberar horarios asociados.

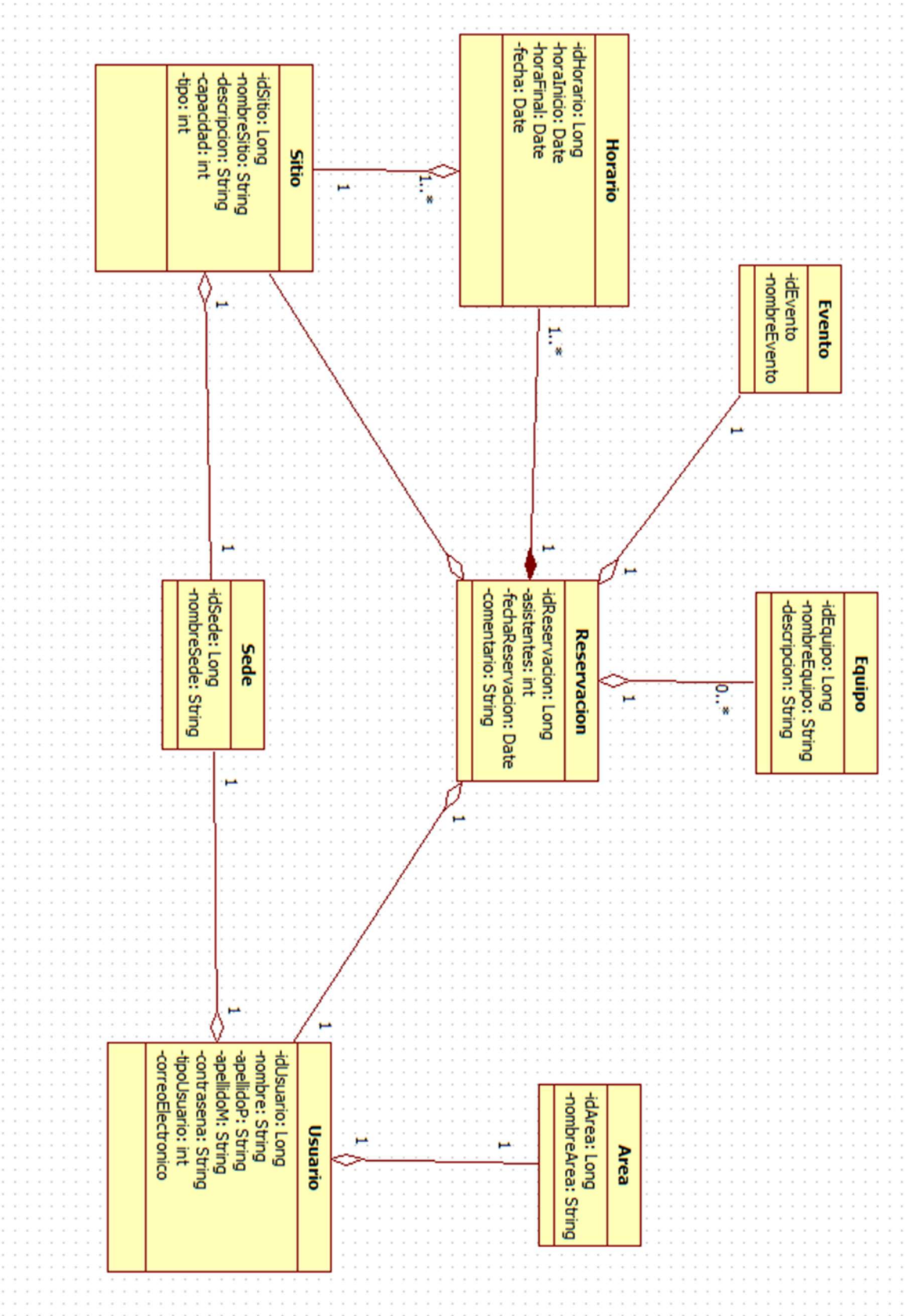


Imagen 4.3 Diagrama de clases

Capítulo 5 Diseño

“El diseño del sistema consiste en actividades de ingeniería para describir la arquitectura o estructura del sistema, así como actividades para describir los algoritmos y funciones necesarias para implementar los requisitos del sistema.” (O’Regan, 2019). Como menciona Regan, en esta fase veremos las actividades, algoritmos y funciones vitales para el sistema. De la misma manera que en el capítulo de requisitos, sólo nos enfocaremos en la solución para el requerimiento de reservar un sitio, ya que este es el núcleo de nuestro sistema.

Arquitectura de capas

La arquitectura de capas ilustra cómo se dividirán las responsabilidades de un conjunto de tareas específicas por cada capa. Dado que existen diversas formas de organizar estas capas, se decidió implantar la siguiente arquitectura, la cual es la más común en Spring Boot y en Frameworks de última generación:

- Capa de presentación: Controladores y vistas.
- Capa de negocio: Servicios.
- Capa de persistencia: Repositorios y DAO.

Este enfoque permite un desarrollo paralelo, cuando se trabaja en equipo, aunque no sea aplicable en este proyecto en particular. Sin embargo, facilita las fases posteriores del desarrollo, especialmente en la fase de pruebas, ya que el código aislado por capas permite probar únicamente el módulo en cuestión. Además, mejora la facilidad de desarrollo, permitiendo avanzar capa por capa o centrarse en funcionalidades específicas sin necesidad de desarrollar más allá de las otras capas.

Esto permite explotar el uso de las interfases ya que como menciono en capítulos anteriores el sistema utilizará el patrón de diseño Inyección de Dependencias, el cual hace que brinde un mejor modularidad del código, y facilitará la reutilización de éste en cada uno de los componentes de las capas superiores, sin necesidad de conocer los detalles de implementación de las capas inferiores.

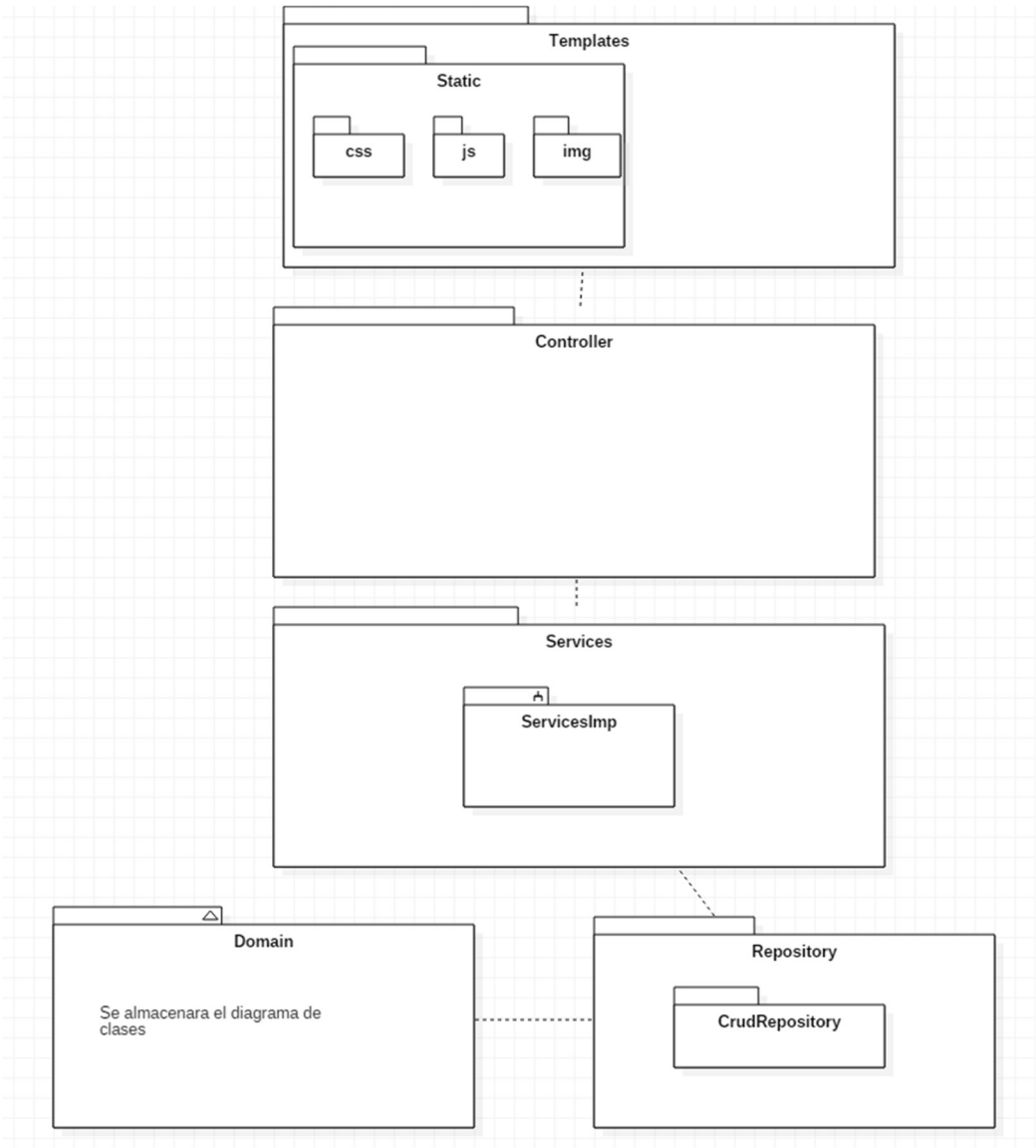


Imagen 5.1 Diagrama de capas para el Sistema

Diagrama de despliegue

“El diagrama de despliegue define la arquitectura física del sistema por medio de nodos interconectados. Estos nodos son elementos de hardware sobre los cuales pueden ejecutarse elementos de software.”(Ivar Jacobson et al., s. f.,p. 76). Aunque el servidor, versión de la base de datos no son las definitivas, podemos establecer los parámetros mínimos, para que el sistema funcione de manera adecuada. Al final, será el área encargada la que decidirá el servidor y la versión de la base de datos.

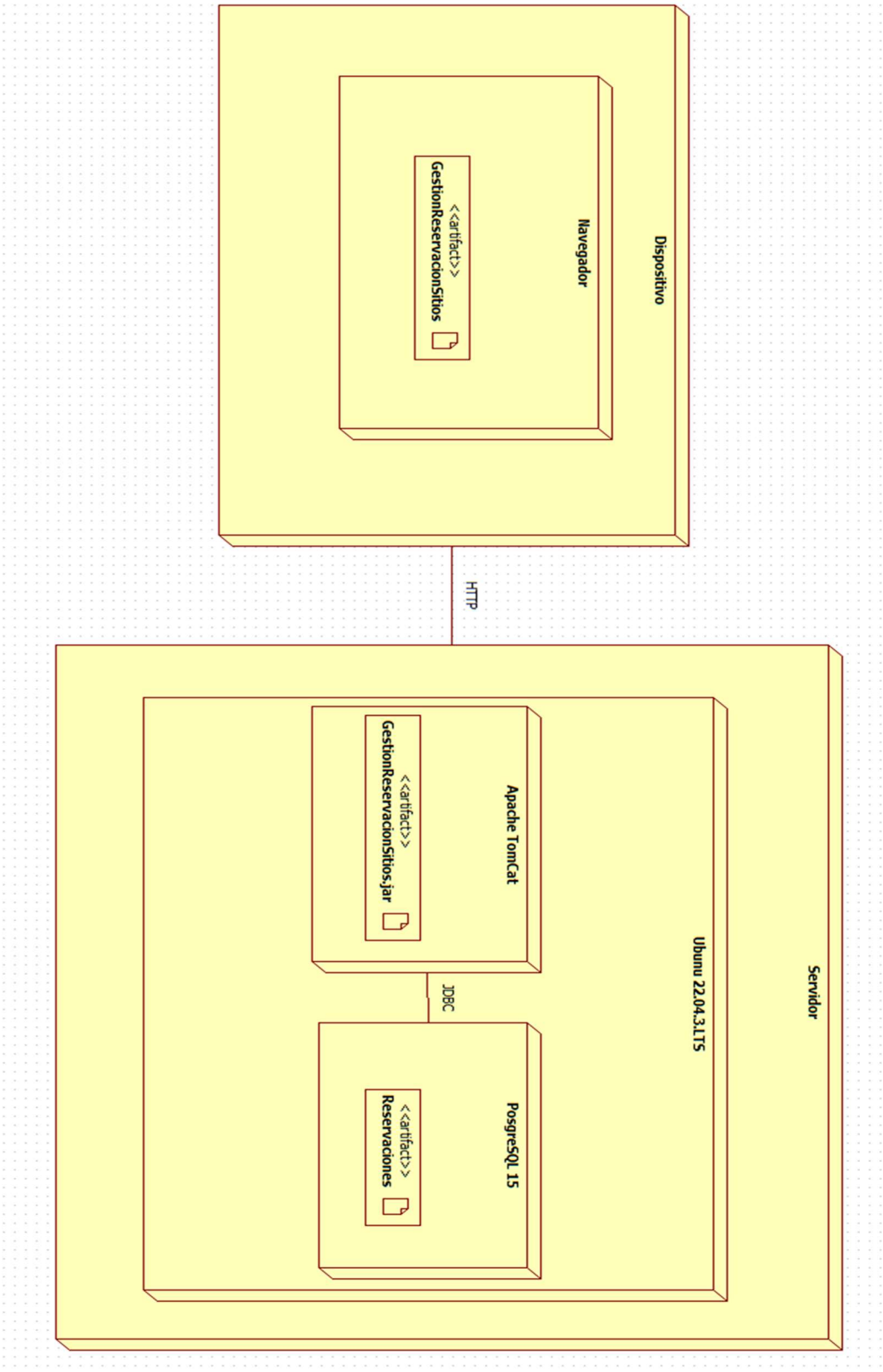


Imagen 5.2 Diagrama despliegue tentativo

Sincronización de las reservaciones

En comparación con el sistema de escritorio en el web se le planteó una nueva necesidad. Si dos usuarios deseaban reservar el mismo sitio al mismo tiempo, esto podría ocasionar empalmes en los bloques de horarios, lo cual compromete la integridad y la consistencia.

Una vez analizado el problema, se propuso la solución más sencilla: implementar una cola de espera. Si un usuario selecciona un sitio y pasa a la selección de horarios, se coloca en una cola de espera, los demás usuarios deberán esperar a que termine de realizar su reservación antes de poder proceder al sitio. Como se muestra en la siguiente imagen, la prioridad de la cola será al usuario que llegue primero a seleccionar el sitio.

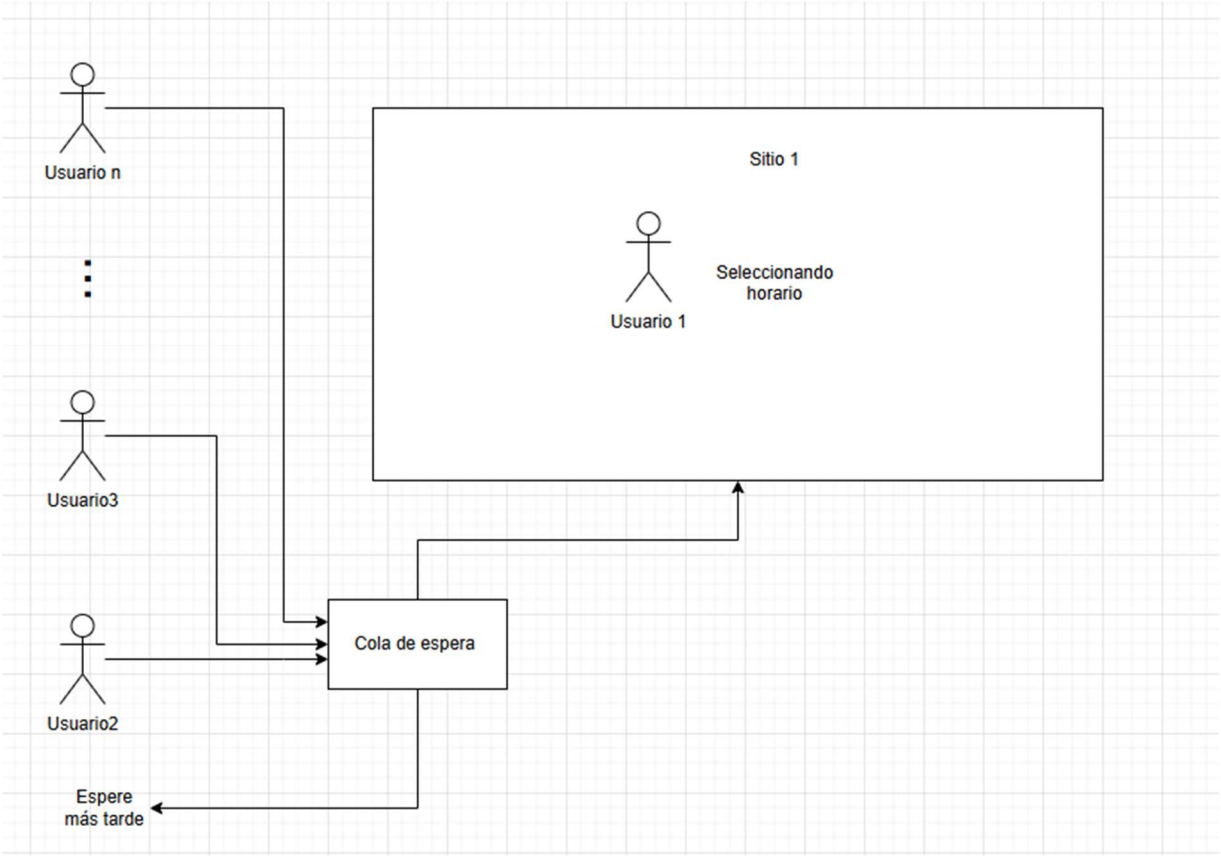


Imagen 5.3 Ejemplo de sincronización.

Es cierto que esto resuelve el problema, pero durante las pruebas con el sistema salieron más detalles como son los siguientes:

- ¿Qué pasa con los usuarios que interrumpen sin concluir el proceso?

- ¿Qué pasa con los usuarios que dejan abierta la ventana de selección de horarios?

En ambos casos, los sitios quedaban en la cola de espera y no se podían volver a seleccionar. Para resolver esto, se optó por establecer un tiempo de espera de 2 minutos. Si el usuario no completa su reservación dentro de ese tiempo, pierde la oportunidad de realizarla y tendrá que empezar el proceso nuevamente. De este modo, el sistema revisa periódicamente la cola de espera para liberar los sitios y que los usuarios pueden seleccionarlos.

Capítulo 6 Implementación

En este capítulo, se abordan tanto las configuraciones del proyecto como el desarrollo del proceso para hacer una reservación. Se detalla el proceso capa por capa, desde la solicitud inicial hasta la confirmación de la reservación, cubriendo cada paso del flujo de trabajo.

Creación del proyecto

El análisis inicial para comenzar el desarrollo involucró la exploración de diferentes tipos de proyectos en Spring y Spring Boot. Si bien ninguno es inherentemente superior al otro, cada uno sirve a propósitos distintos. Spring ofrece una mayor capacidad de personalización y control sobre la configuración, lo que conlleva una curva de aprendizaje más pronunciada debido a la complejidad de los archivos que maneja. Por otro lado, Spring Boot presenta una configuración más automatizada, lo que acelera significativamente el proceso de desarrollo. “Spring Boot es un moderno marco backend basado en Java que facilita el desarrollo más rápido que los marcos tradicionales basados en Java. Con Spring Boot, puedes crear un sistema de aplicación web independiente que tiene un servidor de aplicaciones integrado.” (Hinkula, 2023b,p.1). Su característica distintiva es la inclusión de un servidor de aplicaciones integrado, lo que simplifica enormemente el proceso de desarrollo.

Configuración del proyecto

Gradle build

El archivo `build.gradle` indica cómo se va a estructurar el proyecto y las versiones de las dependencias que se utilizarán. Para este proyecto, se decidió usar Java 11 ya que es una versión estable y ampliamente utilizada en el mercado, lo que proporciona una gran cantidad de documentación y soporte. Esta versión de Java es muy empleada en diversos proyectos a nivel mundial; un ejemplo destacado es su uso para firmar aplicaciones móviles en las tiendas de Google Play.

La versión de Spring Boot, utilizada fue la 2.1.18, que es compatible con Java 8 y 11. En las primeras versiones del proyecto, se planteó utilizar Java 8, pero dado que ya no recibe actualizaciones, se optó por Java 11, aunque muchas empresas y desarrolladores aún utilizan Java 8 debido a su estabilidad y amplio uso en la industria. Como podemos ver en la siguiente imagen 6.1 se muestra la versión del Spring Boot además de ver qué repositorio se utilizará para obtener las dependencias y quien se encargará de bajarlas.

```
buildscript {
      ext {
            springBootVersion = '2.1.18.RELEASE'
      }
      repositories {
            mavenCentral()
      }
      dependencies {
            classpath("org.springframework.boot:spring-boot-gradle-
plugin:${springBootVersion}")
      }
}
```

Archivo de configuración 6. 1 `Gradle.Build` versión de Spring Boot

Para la compilación en el entorno de desarrollo, se estableció la creación de un archivo `.jar`. Aunque aún no se han definido todos los detalles sobre cómo se implementará el aplicativo en los servidores de la UACM donde se alojará la aplicación. El empaquetado `.jar` brinda mejor facilidad de portabilidad y ejecución. Como se muestra en el archivo de configuración siguiente.

```
bootJar {
      baseName = 'reservacion'
      version =  '0.1.0'
}
group = 'mx.edu.uacm'

compileJava.options.encoding = 'UTF-8'
tasks.withType(JavaCompile) {
    options.encoding = 'UTF-8'
}
```

Archivo de configuración 6. 2 `Gradle.Build` configuración jar

Dependencias preliminares

A continuación, se muestran las principales herramientas que se utilizan a lo largo del desarrollo.

- `Bootstrap`. Es una herramienta de desarrollo front-end que proporciona estilos predefinidos para la creación de sitios web. Además de su amplia documentación de soporte, cuenta con una comunidad activa y una variedad de foros donde los usuarios pueden obtener ayuda y compartir conocimientos.
- `Spring Boot Data JPA`. Simplifica la configuración y el uso de JPA (Java Persistence API), una especificación estándar para mapear objetos a tablas en bases de datos (Mapeo Objeto Relacional). Entre sus conceptos clave se encuentran las entidades, las relaciones entre entidades y el EntityManager. Además, proporciona operaciones CRUD (Crear, Leer, Actualizar y Eliminar), lo que agiliza el desarrollo al simplificar la manipulación de datos. En la mayoría de los casos, sólo es necesario validar los datos antes de insertarlos, lo que permite un desarrollo más eficiente y rápido.
- `Spring Boot Mail`. Facilita la configuración y el uso del correo electrónico en las aplicaciones.
- `Spring Boot Starter Thymeleaf`. Thymeleaf es un motor de plantillas para la creación de vistas HTML, lo que permite integrar fácilmente datos dinámicos en páginas web. Spring Boot Starter Thymeleaf es una dependencia de Spring Boot que simplifica el uso del motor de plantillas Thymeleaf en aplicaciones web.
- `Spring Boot Starter Web`. Proporciona un conjunto de funcionalidades esenciales, como el manejo de solicitudes HTTP, la gestión de sesiones, el enrutamiento de URL y la configuración de vistas.
- `Junit`. Es un marco de pruebas unitarias para el lenguaje de programación Java. Permite escribir y ejecutar pruebas automáticas para verificar el comportamiento de los componentes de su código.

Durante el proceso de implantación se agregaron más dependencias para facilitar el desarrollo. Las más destacadas son las siguientes:

- `org.postgresql`. Sirve para realizar la conexión con la base de datos
- `javax.validation`. Su función es para validar ciertos parámetros en el mapeo objeto relacional
- `org.webjars`. En esencia es la configuración de Bootstrap

Estas dependencias fueron seleccionadas para garantizar un desarrollo más ágil y eficiente, permitiendo centrarse en la lógica de negocio. Las demás dependencias se muestran en el archivo de configuración 6.3.

```
implementation group: 'org.springframework.boot', name: 'spring-boot-
starter-web', version: '2.0.5.RELEASE'

implementation group: 'org.springframework.boot', name: 'spring-boot-
starter-data-jpa', version: '2.0.5.RELEASE'

testImplementation group: 'org.springframework.boot', name: 'spring-
boot-starter-test', version: '2.0.5.RELEASE'

implementation 'org.springframework.boot:spring-boot-starter-
thymeleaf:2.7.1'

implementation group: 'org.postgresql', name: 'postgresql', version:
'42.3.5'

implementation group: 'javax.validation', name: 'validation-api',
version: '2.0.1.Final'

implementation group: 'org.webjars', name: 'bootstrap', version:
'5.1.3'

implementation group: 'org.springframework.boot', name: 'spring-boot-
starter-mail', version: '2.2.0.RELEASE'

implementation group: 'com.sun.mail', name: 'javax.mail', version:
'1.6.2'

testImplementation group: 'junit', name: 'junit', version: '4.13.2'
```

Archivo de configuración 6. 3 `Gradle.Build` versión de `Spring Boot` Dependencias

Archivo de configuración Application.properties

Este archivo de configuración debe establecer los parámetros de la base de datos y la ubicación donde se desplegará la aplicación, además de otras configuraciones que iremos desglosando poco a poco.

Para empezar, mencionaremos los parámetros de la base de datos. Una de las ventajas de trabajar con Spring Boot es su flexibilidad para integrarse con diferentes gestores de bases de datos relacionales. Esto se logra configurando el driver de conexión adecuado mediante el parámetro `spring.datasource.driver-class-name`. Este parámetro puede variar según la versión y el gestor de la base de datos que esté utilizando. Para nuestro proyecto, hemos seleccionado PostgreSQL como gestor de base de datos y configurado el controlador correspondiente. Los siguientes parámetros son el nombre de usuario y la contraseña de la base de datos. Aunque en el desarrollo utilizamos el usuario administrador, es recomendable utilizar un usuario con los permisos mínimos necesarios para garantizar un mayor nivel de seguridad. Los parámetros correspondientes son `spring.datasource.username` y `spring.datasource.password`. Es fundamental seguir las mejores prácticas de seguridad al configurar estos datos de autenticación, como asignar permisos específicos a este usuario y utilizar contraseñas seguras y robustas. Para el parámetro de asignación de la base de datos es *`spring.datasource.url`* donde se establece el nombre de la base de datos, puerto y dirección IP.

Durante el desarrollo, otros cuatro parámetros de configuración fueron cambiados con el fin de hacer el desarrollo más ágil:

- `spring.jpa.show-sql`. Su fin es ver las consultas que se están haciendo durante la ejecución del sistema
- `spring.jpa.hibernate.ddl-auto`. Esta tiene varias configuraciones posibles pero las dos más utilizadas fueron las de `update` la cual deja la base de datos tal como está y la de `create` la cual crea, desde cero, toda la base de datos.
- `spring.datasource.continue-on-error`. Se utiliza para controlar el comportamiento del arranque de la aplicación en caso de errores relacionados

con la configuración de la fuente de datos. En `true` la aplicación continuará su ejecución y de en caso contrario, `false` parará la ejecución.

A continuación, aparece un fragmento del archivo `application.properties` donde aparecen dichos parámetros.

```
#spring.datasource.url=jdbc:h2:mem:AZ;DB_CLOSE_DELAY=-
1;DB_CLOSE_ON_EXIT=FALSE

spring.datasource.url=jdbc:postgresql://127.0.0.1:5432/Reservaciones
#spring.datasource.url=jdbc:postgresql://localhost:5432/reservaciones

#spring.datasource.username=postgres
spring.datasource.username=postgres

spring.datasource.password=
#spring.datasource.password=

#spring.datasource.driver-class-name=org.h2.Driver
spring.datasource.driver-class-name=org.postgresql.Driver
spring.messages.encoding=UTF-8

# Show or not log for each sql query
spring.jpa.show-sql = false
```

Archivo de configuración 6. 4 `application.properties` configuración de la base de datos

Los siguientes parámetros son los de configuración del puerto de salida y configuración de mail como se muestra en la imagen 6.5. Para el puerto de salida se estableció el 8181 porque estaba desplegada una aplicación en el puerto por default 8080 en nuestro servidor de desarrollo.

```
#Encender modo debug de mi app
logging.level.mx.edu.uacm=DEBUG

server.port:8181
spring.servlet.multipart.max-file-size=30MB
spring.servlet.multipart.max-request-size=30MB

#email Properpies
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.protocol = smtp
spring.mail.username=recursosMaterialesSlt@gmail.com
spring.mail.password=pkprduwrtcnstrcl
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
spring.mail.properties.mail.smtp.starttls.required=true
spring.mail.properties.mail.smtp.ssl.trust=smtp.gmail.co
```

Archivo de configuración 6. 5 `application.properties` configuración mail, puerto de salida y logs

“Leer los registros de la consola es una habilidad crucial cuando se desarrollan aplicaciones Spring”(Hinkula, 2023a) Es cierto que leer los registros de consola es muy útil y facilita el mantenimiento y desarrollo del sistema, sin embargo, el uso de estos registros en lugares específicos y estratégicos ayuda a maximizar su efectividad y asegurar una interacción adecuada. Esto implica colocar los `logs` en puntos clave del código donde puedan proporcionar información relevante sobre el estado y el comportamiento del sistema, ayudando así a identificar y resolver problemas de manera más eficiente. Por eso mismo se habilitó la configuración de *`logging.level.mx.edu.uacm`*

Desarrollo del módulo para realizar una reservación

A lo largo de la mayoría de los capítulos, nos centraremos exclusivamente en el proceso de reservación, dado que es el aspecto más complejo y crítico dentro del sistema, ésto nos permite abordar en detalle todos los aspectos relevantes. Para estos iremos desglosando capa por capa hasta llegar a su funcionamiento.

Capa de persistencia

En esta capa es donde se implementan nuestras clases y nuestros ORM, como ya se mencionó en el marco teórico, se debe tener mucho cuidado con el mapeo de éstas. Nuestra clase `Reservacion` utiliza las notaciones indispensables para el manejo de todo el mapeo. Como podemos ver, también se tiene más parámetros que no tienen alguna anotación asociada dentro de la reservación como `asistentes, fechaReservacion, comentario` estos al estar dentro de la entidad se pasan como columnas dentro de ella. En el código fuente 6.1 muestra el acomodo de las anotaciones y sus parámetros.

```
@Entity
public class Reservacion implements java.io.Serializable{

	@Id
	@GeneratedValue
	private Long idReservacion;

	private int asistentes;

	private Date fecharReservacion;

	private String comentario;

	@OneToMany(cascade = CascadeType.ALL,orphanRemoval = true)
	private List<Horario> horarios = new ArrayList<Horario>();



	@ManyToMany(cascade = {CascadeType.PERSIST,CascadeType.MERGE})
	@JoinTable(name = "reservacion_equipos")
	private List<Equipo> equipos= new ArrayList<Equipo>();

	@ManyToOne
	@JoinColumn(name = "id_evento")
	private Evento evento;

	@ManyToOne
	@JoinColumn(name = "id_usuario")
	private Usuario usuario;
```

Código fuente 6. 1ORM de la clase `Reservacion`

Esta capa se relaciona con todas las operaciones que se harán en la base de datos. Esta interfaz hereda a `CrudRepository` como se muestra en el código fuente 6.2, ésta nos ayuda en los métodos CRUD (Create ”crear”, Read “leer”,Update

“actualizar” y Delete “eliminar”) en dicha interfaz solo se pide el nombre de la clase y el atributo que se puso como id. Los métodos CRUD facilitan el desarrollo ya que la mayoría las operaciones SQL ya están definidas dentro de la interface y solo basta con ejecutarlas.

```
public interface ReservacionRepository
extends CrudRepository<Reservacion, Long>
```

Código fuente 6. 2 `ReservacionRepository` heredando los métodos CRUD

Capa de negocio

En la interfaz de capa de negocio se encuentran los métodos que se proporcionarán para el consumo de la capa superior. Solo nos enfocaremos en el proceso de agregar una reservación. El cual ocupa otros métodos de otras interfaces del sistema que las iremos explicando conforme las vayamos ocupando. En el siguiente código fuente se muestran los métodos proporcionados por la interfaz.

```
public interface ReservacionService {
	public String agregarReservacion(Reservacion r);

	public String modificarReservacion(Reservacion r);

	public String eliminarReservacion(Reservacion r);

	public ArrayList<ReservacionContenedor>
misReservaciones(Usuario u);

	public ArrayList<ReservacionContenedor>
misReservacionesAsistencia(Usuario u);

	public ArrayList<ReservacionContenedor>
misReservacionesFalta(Usuario u);

	public ArrayList<SeleccionEliminar>
dameReservacionEliminar(String dia) throws ParseException;

	public String eliminarHorarios(long id);
}
```

Código fuente 6. 3Interfase de los servicios de `ReservacionService`

La clase `ReservacionServiceImp` implementa la interfaz `ReservacionService`, proporcionando todos los métodos necesarios para gestionar las reservaciones. En esta implementación, aprovechamos el núcleo de

Spring mediante la inyección de dependencias de varias interfaces de la capa de persistencia, recordando que es un patrón de diseño que promueve la separación de responsabilidades lo que nos ayuda al desacoplamiento de clases. Además de que utiliza la notación `@Service` se usa para marcar una clase como un "servicio", esto le permite ser detectada y gestionada por Spring, lo cual facilita la gestión de las dependencias y promueve un diseño más modular y fácil de mantener. En nuestra clase, utilizamos cuatro interfaces clave que encapsulan diferentes aspectos del proceso de reservación. Las cuatro interfaces para gestionar diferentes aspectos del proceso de reservación son las siguientes además se muestra en el código fuente 6.4:

- `ReservacionRepository`: Implementa todos los métodos CRUD necesarios para gestionar las reservaciones.
- `SitioRepository`: Aunque tiene métodos CRUD, se utiliza principalmente para consultas muy específicas sobre los sitios.
- `HorarioRepository`: Se utiliza para guardar la asociación de los horarios con la reservación.
- `MessageSource` se utiliza para leer un archivo donde se almacenan todos los mensajes de texto que se tiene en el sistema.

```
@Service
public class ReservacionServiceImp implements ReservacionService {

	@Autowired
	private ReservacionRepository reservacionRespository;

	@Autowired
	private SitioRepository sitosRepository;

	@Autowired
	private HorarioRepository horarioRepository;

	@Autowired
	private MessageSource messageSource;

	private Locale locale= LocaleContextHolder.getLocale();

	private static final Logger log=LogManager.getLogger();
```

Código fuente 6. 4 `ReservacionServiceImp` inyección dependencias

Para nuestro método principal de creación de reservaciones, se implementan dos validaciones. Aunque hay varias reglas de negocio a considerar, estas se filtran en las capas superiores del sistema. Las dos validaciones específicas en esta capa son:

- Validación del Comentario: El comentario asociado a la reservación no debe exceder los 300 caracteres.
- Validación de Capacidad del Sitio: La cantidad de personas no debe sobrepasar la capacidad del sitio.

Una vez que se pasan estas dos validaciones, el sistema procede a guardar los horarios relacionados con la reservación. Se obtiene la fecha actual del servidor para registrar la reservación y finalmente se guarda la misma. A continuación, se muestra el código 6.5 fuente de agregar una reservación.

```
public String agregarReservacion(Reservacion r) {
     if(r.getComentario().length()>300) {
                return
messageSource.getMessage("message.reservacion.comentario",null
,locale) ;
          }else {
                int
capacidadSitio=r.getHorarios().get(0).getSitio().getCapacidad();
                if(r.getAsistentes()>capacidadSitio) {
                     return
messageSource.getMessage("message.reservacion.capaciodad",null
,locale) ;
                }
                for(int i =0;i<r.getHorarios().size();i++) {
                     horarioRepository.save(r.getHorarios().get(i));
                }
                long miliseconds = System.currentTimeMillis();
               Date date = new Date(miliseconds);
                r.setFecharReservacion(date);
                reservacionRespository.save(r);
          }
          return "guardado";
     }
```

Código fuente 6. 5 método `agregarReservacion()`

A continuación, se describen los demás servicios dentro del sistema:

- `SitioService`: se utiliza para consultar, mostrar la información de los sitios además de validar reglas de negocio y ser asignada a una reservación.

- `EquipoService`, `EventoService`, `AreaService`, `SedeService`: se utilizan para consultar, mostrar la información de cada uno de estos catálogos. Esta información será asignada a una reservación.
- `HorarioService`; tiene varios métodos para transformar una matriz de disponibilidad a un bloque de horarios, además se encarga de traer de la base de datos todos los horarios ocupados, este mismo los transforma para posteriormente mandarlos al controlador.
- `UsuarioService`: sirve para determinar los datos del usuario, aunque el objeto usuario se tiene en sesión, existe la posibilidad de que el Administrador pueda dar de alta un registro.
- `ColaDeEsperaService`: encargada de la concurrencia.
- `ReservacionService`: se utiliza para guarda información, mostrar información, eliminar información de una reservación.
- `EmailService`: se encarga de mandar correos electrónicos.

Capa de presentación

La capa de presentación se encarga de gestionar la interacción directa con el usuario y se divide en dos componentes principales: los controladores y las vistas (o templates). Esta división permite una separación clara de responsabilidades y facilita el desarrollo y mantenimiento del sistema. Los controladores actúan como intermediarios entre el modelo (lógica de negocio y datos) y las vistas. Su función principal es manejar las solicitudes del usuario, procesar los datos necesarios a través de los servicios correspondientes y devolver las respuestas adecuadas a las vistas. Las vistas son responsables de la presentación visual de la información al usuario. Utilizan plantillas HTML y tecnologías como Thymeleaf para renderizar las páginas web dinámicamente.

El `ReservacionController` utiliza la anotación `@Controller` para definir que esta clase es un controlador en el patrón de diseño Modelo-Vista-Controlador (MVC). El controlador principal de reservaciones (`ReservacionController`) utiliza varios servicios, cada uno con responsabilidades específicas y únicas. Estos servicios manejan diversas operaciones relacionadas con el proceso de reservaciones,

asegurando que todas las funcionalidades necesarias estén cubiertas de manera eficiente.

En el código fuente 6.6 se muestran todos los servicios utilizados dentro del código para el controlador `ReservacionController.`

```
@Controller
public class ReservacionController {

	@Autowired
	private SitioService sitioSevices;

	@Autowired
	private EquipoService equipoService;

	@Autowired
	private EventoService eventoService;

	@Autowired
	private HorarioService horarioService;

	@Autowired
	private UsuarioService usuariosService;

	@Autowired
	private ColaDeEsperaService colaEspera;

	@Autowired
	private AreaService areaService;


	@Autowired
	private SedeService sedeServices;

	@Autowired
	private ReservacionService reservacionService;
```

Código fuente 6. 6 `ReservacionController` servicios utilizados

A continuación, se verá toda la implementación del código para cada una de las fases del proceso de reservación.

Fase 1 Información preliminar

Para preparar todos los catálogos para la primera fase de reservación que el usuario necesita, se utiliza el método `prepaparReservacion()`. Este método realiza

varias tareas esenciales para garantizar que el usuario pueda acceder y utilizar el sistema de manera efectiva. A continuación, se describen estas tareas:

1. Verificación de Acceso del Usuario: El método reservación primero comprueba si el usuario ha iniciado sesión en el sistema. Si el usuario no ha accedido, es redirigido al formulario de acceso.
2. Determinación del Tipo de Usuario: Una vez que se verifica el acceso, el método determina el tipo de usuario que está intentando hacer la reservación. Esto es importante porque los profesores no pueden reservar todos los sitios disponibles en el sistema, el administrador sí.
3. Carga de Catálogos: El método carga todos los catálogos necesarios para el primer formulario de reservación del usuario. Estos catálogos incluyen información relevante como sitios, equipos, sedes, áreas y eventos según el tipo de usuario.
4. Verificación de Tiempos Muertos: Se verifica que no existan tiempos muertos en la cola de espera, los tiempos muertos son reservaciones que no fueron concluidas o cerradas de manera adecuada. Esto asegura que el proceso de reservación sea fluido.

En el código fuente 6.7 se muestra como esta implementado el método `prepararReservacion()`.

```
@GetMapping("/reservacionAgregar")
public String prepararReservacion(Map<String, Object> model) throws
IOException{
          Usuario usuario  = new Usuario();
          ArrayList<Sitio> sitios;

          if(httpSession.getAttribute("userLoggedIn")==null) {
               return "acceso";
          }
          usuario = (Usuario)
httpSession.getAttribute("userLoggedIn");
          if(usuario.getTipoUsuario()==0) {
                sitios= (ArrayList<Sitio>)
sitioSevices.dameTodosSitio();
          }else {
               sitios= (ArrayList<Sitio>)
sitioSevices.dameTodosLosSitioAulas();
          }
          ArrayList<Area> areas;
          areas = areaService.listaArea();
```

```
            ArrayList<Sede> sedes;
            sedes = sedeServices.listarSede();
            model.put("sedes", sedes);
            model.put("areas", areas);
            ArrayList<Evento> eventos = eventoService.listarEventos();
            ArrayList<Equipo> equipos = equipoService.listarEquipo();
            ArrayList<Usuario> usuarios =
usuariosService.dameTodosUsuarios();
            colaEspera.hayTiemposMuertos();
            model.put("usuarios",usuarios);
            model.put("equipos",equipos);
            model.put("eventos",eventos);
            model.put("sitios",sitios);
            return "reservacionAgregar";
      }
```

Código fuente 6. 7 método prepararReservacion() (Carga de datos para la vista)

Una vez que el usuario completa la primera fase de la reservación, se procede al siguiente método denominado `verSitio` el cual prepara para la fase 2. Este método es fundamental, ya que se encarga de generar los bloques de horarios necesarios para la segunda fase del proceso de reservación. A continuación, se desglosa cómo funciona este método:

1. Primero valida si la capacidad es la adecuada si no es así se regresa a la fase 1. Como se muestra en el código fuente 6.8
2. Verifica las reglas de negocio R2, R3 y R4. Como se muestra en el código fuente 6.9.
3. Se obtiene la disponibilidad de los horarios dependiendo de los días de seleccionados y los sitios, además se hace la transformación de horarios a la matriz de disponibilidad.
4. Se verifica que la cola de espera esté vacía en caso contrario se manda un mensaje al usuario. Como se muestra en el código fuente 6.9.
5. Se añade el usuario a la cola de espera. Como se muestra en el código fuente 6.9.
6. Se mandan los parámetros necesarios para la fase 2.
7. Se pasa a la fase 2 de la reservación de horarios.

```
@PostMapping("/reservacionAgregar")
```

```
public String verSitio(Map<String, Object> model,
                int idSitio,int idEvento,String comentario,
                String datepicker,String equipos,
                int asistentes,int usuarios,final RedirectAttributes
ra) throws Throwable {

          Usuario usuario  = new Usuario();
     usuario = (Usuario) httpSession.getAttribute("userLoggedIn");
          String[] parts = datepicker.split(",");
          ArrayList<String> fechas = new ArrayList<>();
          ArrayList<HorarioPorDia> horarios = new ArrayList<>();
          Sitio sitio = sitioSevices.dameSitio(idSitio);
          /*Validaciones*/
          if(asistentes>sitio.getCapacidad()) {
               ra.addFlashAttribute("message", "capacidad");
               ra.addFlashAttribute("sitio", sitio);
               return "redirect:/reservacionAgregar";
          }
```

Código fuente 6. 8 método `verSitio()` (prepara la selección de horarios)

```
/*Sacar el dia actual*/
/*Validaccion para no meter dias anteriores ala fecha*/
DateTimeFormatter dtf = DateTimeFormatter.ofPattern("dd/MM/yyyy");
String fechaActual=dtf.format(LocalDateTime.now());
for (int i = 0; i < parts.length; i++) {
     if(compararFechasAntesDeLDiaActual(parts[i],fechaActual)) {
          if(usuario.getTipoUsuario()!=0) {
               ra.addFlashAttribute("sitio", sitio);
               ra.addFlashAttribute("message", "fechasMenor");
               return "redirect:/reservacionAgregar";
          }

        }
      }
/*Regla del negocio Validacion Para 3 dias despeus de la actual*/
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
java.util.Date date = sdf.parse(fechaActual);
Calendar cal = Calendar.getInstance();
cal.setTime(date);
cal.add(Calendar.DAY_OF_YEAR, 3);
java.util.Date date2 = cal.getTime();
String fechaPosterior = sdf.format(date2);
for (int i = 0; i < parts.length; i++) {
    if(compararFechasAntesDeLDiaActual(parts[i],fechaPosterior )) {
        if(usuario.getTipoUsuario()!=0) {
            ra.addFlashAttribute("message", "fechasPosterior");
            ra.addFlashAttribute("sitio", sitio);
           return "redirect:/reservacionAgregar";
        }
    }
}
```

```
/*Preparando los horarios*/
for (int i = 0; i < parts.length; i++) {
      fechas.add(parts[i]);
      HorarioPorDia horario = new HorarioPorDia();
      horario = horarioService.dameHorarioPorDia(parts[i], idSitio);
      horarios.add(horario);
}
ra.addFlashAttribute("usuarios", usuarios);
ra.addFlashAttribute("comentario", comentario);
ra.addFlashAttribute("equipos", equipos);
ra.addFlashAttribute("idEvento", idEvento);
ra.addFlashAttribute("idSitio", idSitio);
ra.addFlashAttribute("nombreSitio", sitio.getNombreSitio());
ra.addFlashAttribute("asistentes", asistentes);
ra.addFlashAttribute("datepicker",datepicker);
ra.addFlashAttribute("horarios", horarios);
if(colaEspera.estaEnCola(sitio)) {
      ra.addFlashAttribute("message", "espera");
      return "redirect:/reservacionAgregar";
}
colaEspera.agregarAlaCola(sitio);
      return "redirect:/reservacionAgregarDias";
}
```

Código fuente 6. 9 método `verSitio` (reglas de negocio)

Fase 2 Selección de horarios

Una vez que el usuario selecciona sus horarios deseados el sistema verifica si el usuario ha seleccionado o no un bloque horario, si no es el caso lo regresa a la pantalla de selección para que siga haciendo el proceso. Como se muestra en el código fuente 6.10.

```
@PostMapping("/reservacionAgregarDias")
      public String guardandoReservacion(Map<String, Object> model,
                  int idSitio,String equipos,String datepicker,int
asistentes,int idEvento,RerservacionesContenedor reservacion,
                  final RedirectAttributes ra,String comentario,int
usuarios) throws Throwable {
log.info("Sitio= " + idSitio + "  Equipos"+ equipos + "  tamanioRe" +
reservacion.getReservaciones().size()+"Asistentes"+ asistentes);

Reservacion reservacionFinal = new Reservacion();
Evento evento = eventoService.dameEventoPorID(idEvento);
if(usuarios == 0) {
      Usuario usuario = (Usuario)
      httpSession.getAttribute("userLoggedIn");
      reservacionFinal.setUsuario(usuario);
}else {
      Usuario usuario = usuariosService.dameUsuario(usuarios);
      reservacionFinal.setUsuario(usuario);
}
```

```
log.info("reservacion"+reservacion.getReservaciones().get(1));

if(reservacion.getReservaciones().size()==0
      ||reservacion.getReservaciones().size()==1) {

      /*Preparando los horarios*/
      String[] parts = datepicker.split(",");
      ArrayList<String> fechas = new ArrayList<>();
      ArrayList<HorarioPorDia> horarios = new ArrayList<>();
      for (int i = 0; i < parts.length; i++) {
            fechas.add(parts[i]);
            HorarioPorDia horario = new HorarioPorDia();
            horario = horarioService.dameHorarioPorDia(parts[i],
      idSitio);
            horarios.add(horario);
      }
      ra.addFlashAttribute("usuarios", usuarios);
      ra.addFlashAttribute("message", "noseleccion");
      ra.addFlashAttribute("datepicker", datepicker);
      ra.addFlashAttribute("comentario", comentario);
      ra.addFlashAttribute("equipos", equipos);
      ra.addFlashAttribute("idEvento", idEvento);
      ra.addFlashAttribute("idSitio", idSitio);
      ra.addFlashAttribute("asistentes", asistentes);
      ra.addFlashAttribute("horarios", horarios);
      return "redirect:/reservacionAgregarDias";
}
```

Código fuente 6. 10 método `guardandoReservación` (Validando que se haya seleccionado un bloque)

Si el usuario ya seleccionó un bloque de horario, se procede a la transformación de bloques de matriz a datos de tipo date. Para optimizar el proceso de reservación, se han definido las siguientes condiciones que deben cumplirse para garantizar un flujo eficiente y rápido, este proceso se muestra en el código fuente 6.11:

1. Selección de un Solo Día:

Si el usuario selecciona únicamente una fecha (por ejemplo, 01-04-2023), el sistema realizará operaciones mínimas las cuales son indicar bloque inicial, final y dia de evento esto acelerará el proceso.

2. Selección de un Solo Bloque de Horario:

Si el usuario selecciona solo un bloque de horario (por ejemplo, el bloque 0, que corresponde a 7:00 a 7:30), el sistema ejecutará las operaciones necesarias para ese único intervalo, agilizando el procesamiento.

3. El caso general de reservación:

Se realiza un ciclo que itera a través de todos los días solicitados por el usuario. Durante este proceso, los bloques seleccionados se transforman y almacenan en un arreglo de horarios. A continuación, se detalla el proceso:

- Iteración de los Días Seleccionados. El sistema recorre cada día que el usuario ha seleccionado para la reservación.
- Transformación de Fechas y Horarios. Para cada día, el sistema transforma las fechas y horarios seleccionados por el usuario.
- Almacenamiento en el Arreglo de Horarios. Los horarios transformados se almacenan en un arreglo para su posterior procesamiento.

```
int condicion = 0; /*dia igual*/
String dia = ""; /*dia igual*/
int bloqueI = -1;
int bloqueF = -1;
for(int i=0;i < horariosFormatoDia.size();i++) {
if(dia.equals("")||dia.equals(horariosFormatoDia.get(i).getDia())) {
      if(condicion == 0) {
            bloqueI= horariosFormatoDia.get(i).getHora();
            bloqueF = horariosFormatoDia.get(i).getHora();
            condicion = 1;
            dia = horariosFormatoDia.get(i).getDia();
            }if(i>=1){
                  if(horariosFormatoDia.get(i).getHora()-
            horariosFormatoDia.get(i-1).getHora()!=1) {
                  bloqueF = horariosFormatoDia.get(i-1).getHora();
                  /*Seleccion de dias*/
                  Horario horario = new Horario();
                  String[] partsHorarios =dia.split("/");
                  String diaFormato = partsHorarios[2]+"-
                  "+partsHorarios[1]+"-"+partsHorarios[0];
                  Date date=Date.valueOf(diaFormato);
                  horario.setSitio(sitio);
                  horario.setFecha(date);
                  horario.setHoraInicio(horarioService.dameHoraString(
            bloque
                  horario.setHoraFinal(horarioService.dameHoraString(b
            loqueF+1));
                  horarios.add(horario);
                  bloqueI= horariosFormatoDia.get(i).getHora();
                  dia = horariosFormatoDia.get(i).getDia();
            }
      }else if(horariosFormatoDia.size()==1) {
            /*Caso que es un solo bloque*/
            bloqueI= horariosFormatoDia.get(i).getHora();
            bloqueF = horariosFormatoDia.get(i).getHora();
```

```
            dia = horariosFormatoDia.get(i).getDia();
            Horario horario = new Horario();
            String[] partsHorarios =dia.split("/");
            String diaFormato = partsHorarios[2]+"-" +
            partsHorarios[1]+"-"+partsHorarios[0];
            Date date=Date.valueOf(diaFormato);
            horario.setSitio(sitio);
            horario.setFecha(date);
      horario.setHoraInicio(horarioService.dameHoraString(bloqueI);
      horario.setHoraFinal(horarioService.dameHoraString(bloqueF+1));
            horarios.add(horario);
      }
}else {
      condicion = 1;
      bloqueF = horariosFormatoDia.get(i-1).getHora();
      dia = horariosFormatoDia.get(i-1).getDia();
      Horario horario = new Horario();
      String[] partsHorarios =dia.split("/");
      String diaFormato = partsHorarios[2]+"-"+
      partsHorarios[1]+"-"+partsHorarios[0];
      Date date=Date.valueOf(diaFormato);
      horario.setSitio(sitio);
      horario.setFecha(date);
      horario.setHoraInicio(horarioService.dameHoraString(bloqueI));
      horario.setHoraFinal(horarioService.dameHoraString(bloqueF+1));
      horarios.add(horario);
      bloqueI= horariosFormatoDia.get(i).getHora();
      dia = horariosFormatoDia.get(i).getDia();
}
if(i==horariosFormatoDia.size()-1) {
      bloqueF = horariosFormatoDia.get(i).getHora();
      Horario horario = new Horario();
      String[] partsHorarios =dia.split("/");
      String diaFormato = partsHorarios[2]+"-
      "+partsHorarios[1]+"-"+partsHorarios[0];
      Date date=Date.valueOf(diaFormato);
      horario.setSitio(sitio);
      horario.setFecha(date);
      horario.setHoraInicio(horarioService.dameHoraString(bloqueI));
      horario.setHoraFinal(horarioService.dameHoraString(bloqueF+1));
      horarios.add(horario);

      }
}
```

Código fuente 6. 11 transformación de horarios

Una vez terminado el proceso de transformación de horarios se recopila toda la información de la reservación, una vez guardada se sale de la cola de espera y comienza la fase 3, como se muestran en el código fuente 6.12.

```
reservacionFinal.setHorarios(horarios);
```

```
        reservacionFinal.setComentario(comentario);
        reservacionFinal.setEvento(evento);
        reservacionFinal.setAsistentes(asistentes);
        reservacionService.agregarReservacion(reservacionFinal);
        /*Cola Espera salir*/
        colaEspera.eliminarDeLaCola(sitio);
        ra.addFlashAttribute("reservacionFinal",
        reservacionFinal);
        ra.addFlashAttribute("horarios",
        reservacionFinal.getHorarios());
        ra.addFlashAttribute("sitio", sitio);
        return "redirect:/confirmacionReservacion";
}
```

Código fuente 6. 12 método `guardarReservación` (Guardando la reservación)

Fase 3 Confirmación de la reservación

Una vez finalizada la fase 2, se envía la información de la reservación al usuario para que esté al tanto del estado de su reservación, con esto culmina el proceso de realizar una reservación.

Ejecución de una reservación

Los pasos para realizar una reservación son los siguientes:

1. Acceder al sistema
2. Seleccionar del menú reservación nueva reservación
3. Fase 1 Información preliminar
4. Fase 2 Selección de horarios
5. Fase 3 Confirmación de reservación

Con los pasos anteriormente mencionados y visto a grandes rasgos la programación detrás, se presentan, mediante capturas de pantalla, el proceso que el usuario realiza con un enfoque menos técnico.

Accediendo al sistema

En este punto, se presenta un formulario que recoge la información necesaria para identificar al usuario y tener acceso a todas las funcionalidades que tiene su perfil. A continuación, se muestra una imagen de formulario de acceso.

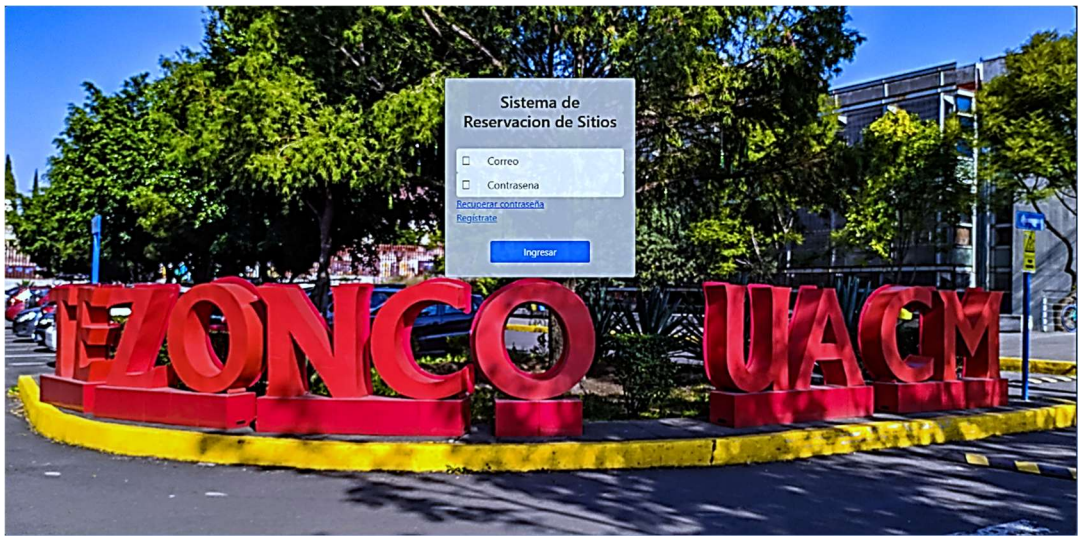


Imagen 6.1 Formulario de acceso.

Seleccionar del menú reservación nueva reservación

Una vez accedido al sistema se muestra el menú Reservación para seleccionar Nueva Reservación. En la siguiente imagen se puede apreciar el menú de Reservación.

Imagen 6.2 Menú Reservación

Aunque se puede ver la disponibilidad de distintas aulas antes de realizar una reservación nos enfocaremos en el camino feliz.

Fase 1 Información preliminar

El siguiente formulario recopila la información básica para realizar una reservación. Este formulario cambia dependiendo del tipo de usuario, el que se muestra continuación es el del perfil de Profesor.

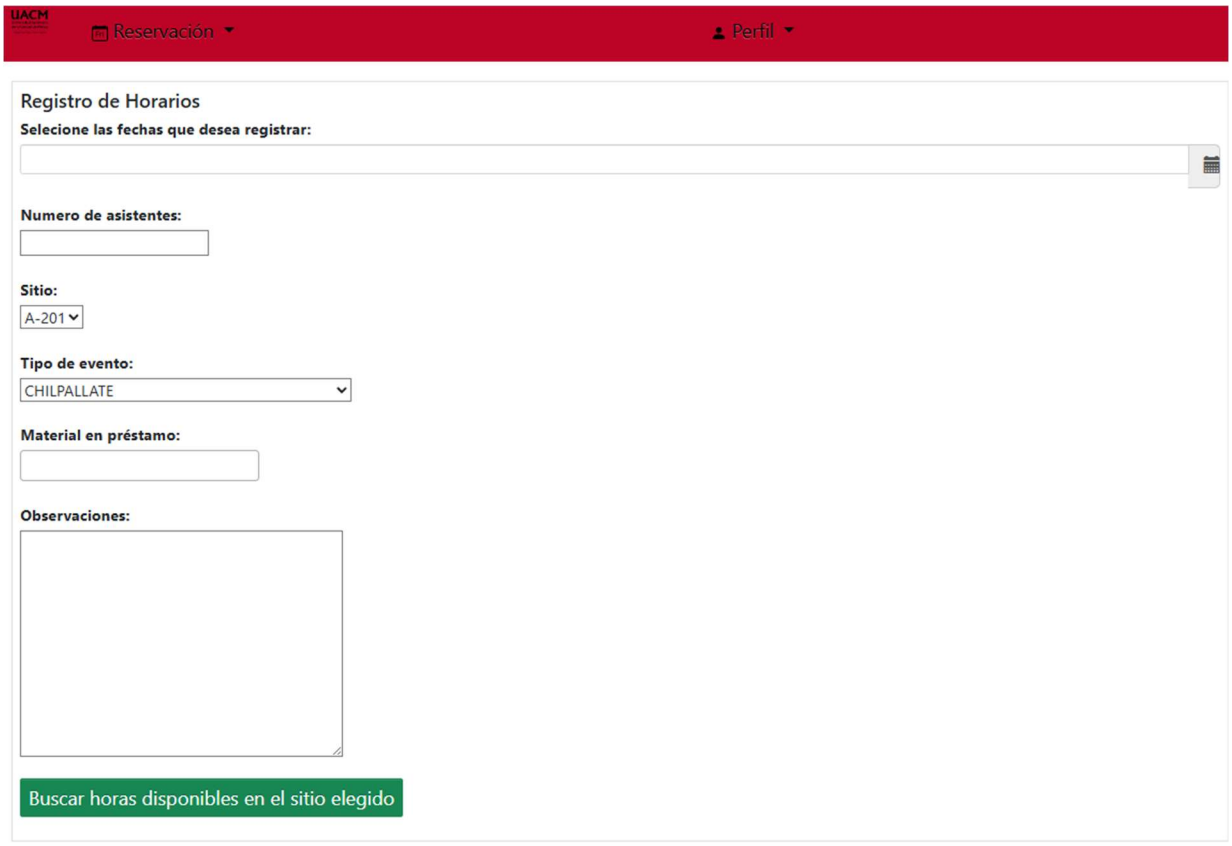


Imagen 6.3 Formulario de Información preliminar.

Para solucionar el problema existente en la aplicación de escritorio relacionado con la sección de fechas, se optó por implementar un calendario con selección múltiple, eliminando así dicha dificultad. Como se muestra en la siguiente imagen, el calendario de selección múltiple va marcando los días seleccionados de color azul y en color amarillo el día actual. A continuación, se muestra una imagen de cómo funciona el multi-calendario.

Imagen 6.4 Ejemplo de multi-calendario.

Para la selección de los equipos, se optó por un seleccionador múltiple, lo que facilita el proceso para el usuario y aumenta la velocidad de selección. Como lo podemos ver en la siguiente imagen, se van marcando los equipos seleccionados con opción de poder quitarlos.

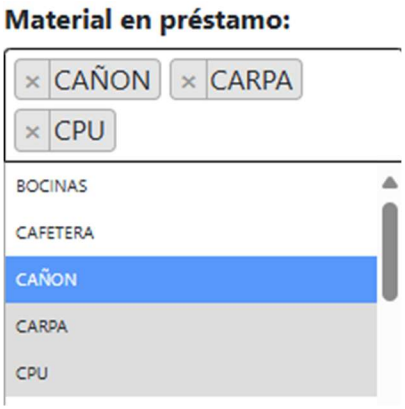


Imagen 6.5 Ejemplo de selección de equipos.

A continuación, se muestra un ejemplo de cómo quedaría la primera fase de la reservación cuando culmina, el usuario debe presionar el botón “Buscar hora disponibles en el sitio elegido”. Si la cola de espera está llena se le notificará al usuario con un mensaje de error. De ser correcta la información se pasa a la

segunda fase de reservación, además el sitio seleccionado pasa a la cola de espera para que ningún otro usuario pueda seleccionarla.

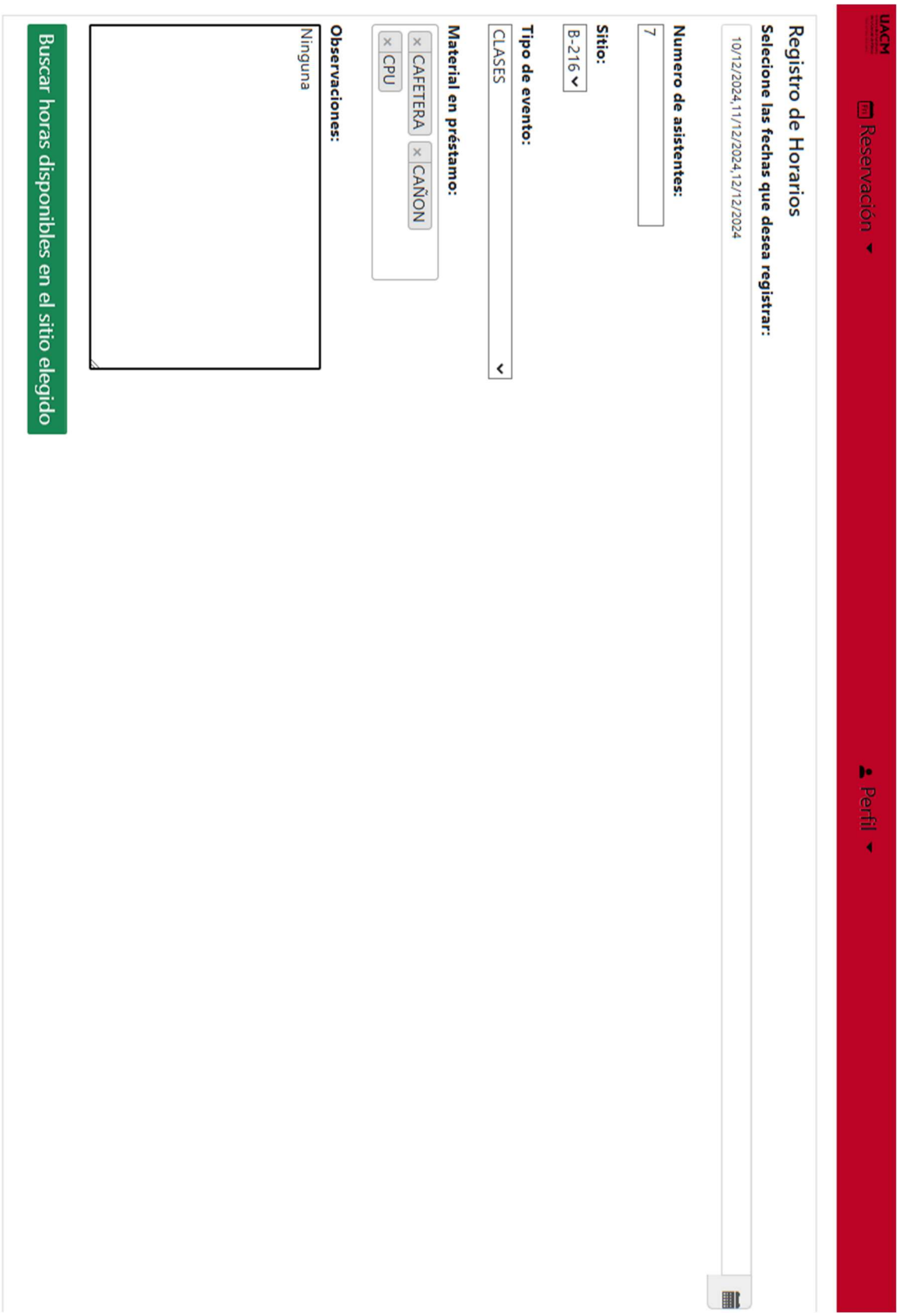


Imagen 6.6 Ejemplo de primera fase de la reservación terminada

Fase 2 Selección de horarios

En esta fase el usuario elige los bloques de horarios que desea seleccionar, una vez terminada su selección presionará el botón “Reservar”, como se muestra en la siguiente imagen.

Imagen 6.7 Ejemplo de segunda fase selección de horarios

Fase 3 Confirmación de reservación

Cuando se presiona el botón de “Reservar” el sistema muestra todos los datos de la Reservación para que el usuario verifique la información como se muestra en la siguiente imagen.

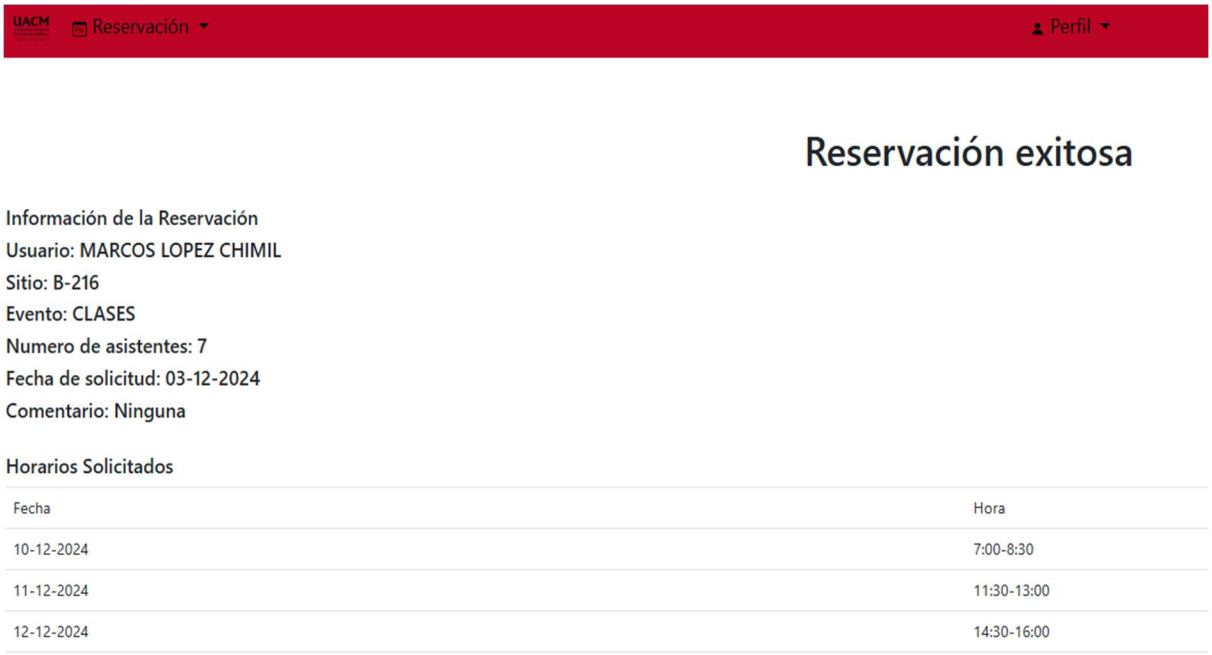


Imagen 6.8 Ejemplo de tercera fase Confirmación de Reservación

Por último, los bloques de horarios seleccionados se marcarán en color rojo la siguiente ves que algún usuario realice una reservación como se muestra en la siguiente imagen.

Imagen 6.9 Ejemplo de horarios ocupados

Capítulo 7 Pruebas

Las pruebas son de vital importancia para verificar si todo lo desarrollado funciona de manera adecuada. Durante este período, se llevaron a cabo diversos tipos de pruebas.

Pruebas unitarias

Las pruebas unitarias son pruebas de software que se enfocan en validar de forma aislada el funcionamiento de las unidades más pequeñas como funciones, métodos o clases. Estas pruebas tienen como objetivo garantizar que cada unidad del código funcione según lo esperado.

JUnit es un Framework de código abierto escrito en Java que se utiliza para realizar pruebas unitarias. Además, te permite una configuración muy sencilla y permite ejecutar tus pruebas con un solo clic.

La realización de pruebas unitarias es de vital importancia, ya que permite evaluar y validar partes específicas del código de manera sistemática. Estas pruebas contribuyen significativamente a garantizar la calidad y funcionalidad de cada componente del sistema. Con el objetivo de asegurar un 100% de cobertura en las pruebas unitarias del sistema, se realizó la división de las pruebas según las capas existentes. Como lo venimos haciendo a lo largo del escrito, solo nos enfocaremos en las pruebas unitarias relacionadas con la reservación, y todas estas pruebas unitarias se fueron realizando durante el desarrollo del sistema.

Para la primera prueba verificamos el número de caracteres del comentario de la reservación, si ésta sobrepasa 300 caracteres para provocar el siguiente mensaje de error “La longitud del comentario tiene que ser menor a 300 caracteres”. Así JUnit compara lo esperado con el mensaje que retorna el método que se está probando, si los dos textos son iguales la prueba será exitosa.

```
@Test
```

```
    public void reservacionServicesComentarioMayorCaracter() {
    Reservacion reservacion = new Reservacion();

    reservacion.setComentario("12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890");

    String respuesta

    = reservacionServices.agregarReservacion(reservacion);
    assertEquals
    (respuesta, "La longitud del comentario tiene que ser que menor
    a 300 caracteres");
}
```

Código fuente 7. 1 Prueba de comentario mayor a 300 caracteres.

La siguiente prueba es para verificar la capacidad de un sitio, para ello selecciono el sitio C-207 el cual tiene una capacidad de treinta y cinco personas. La prueba pretende sobrepasar este número para que Junit lo valide con un mensaje como lo podemos ver en el siguiente fragmento de código.

```
Reservacion reservacion = new Reservacion();
          Sitio sitio  = sitioServices.dameSitio(43);
          Date fecha = Date.valueOf("2024-08-14");
          Horario h1 = new Horario("21:00", fecha, "21:30");
          h1.setSitio(sitio);
          reservacion.setComentario("");
          reservacion.setAsistentes(36);
          ArrayList<Horario> horarios = new ArrayList<>();
          horarios.add(h1);
          reservacion.setHorarios(horarios);
          String respuesta
=reservacionServices.agregarReservacion(reservacion);
          assertEquals(respuesta, "El numero de asistenes es mayor a
la capacidad del sitio");
```

Código fuente 7. 2 Prueba de capacidad mayor a 36 personas

A continuación, se muestra una prueba de integración y consiste en que la reservación sea exitosa. Para esto, creamos una reservación y posteriormente seleccionamos el sitio 43, que corresponde al C-207. A continuación, seleccionaremos un usuario al azar. Luego, almacenarnos los demás datos de la reservación, como los horarios, equipos y el evento, posteriormente asignaremos

esta información a la reservación, ya con todos datos de la reservación cubiertos, la guardamos, como se muestra en el siguiente fragmento de código.

```
@Test
@Transactional
public void agregarReservacion() {
      Reservacion reservacion = new Reservacion();
      /*Sitio*/
      Sitio sitio  = sitioServices.dameSitio(43);
      //Usuario
      ArrayList<Usuario> usuarios =
      usuarioService.dameTodosUsuarios();
                reservacion.setUsuario(usuarios.get(0));
      /*Horarios*/
      Date fecha = Date.valueOf("2024-06-14");
      Horario h1 = new Horario("10:00", fecha, "18:30");
      ArrayList<Horario> horarios = new ArrayList<>();
      h1.setSitio(sitio);
      horarios.add(h1);
      reservacion.setHorarios(horarios);
      /*Equipos*/
      List<Equipo> equipos  = equipoService.listarEquipo();
      reservacion.getEquipos().addAll(equipos);
      /*Evento*/
      ArrayList<Evento> eventos = eventoService.listarEventos();
      Evento evento = eventos.get(0);
      reservacion.setEvento(evento);
      reservacion.setComentario("");
      reservacion.setAsistentes(34);
      String respuesta =
      reservacionServices.agregarReservacion(reservacion);
      assertEquals(respuesta, "guardado");
}
```

Código fuente 7. 3 Prueba de reservación exitosa.

Complejidad ciclomática

Existen diversos métodos y enfoques para identificar los casos mínimos de prueba, cada uno utilizando técnicas y herramientas específicas adaptadas a distintos contextos y necesidades. En este sistema, se empleó la complejidad ciclomática para determinar los casos mínimos de prueba debido a su enfoque claro y su facilidad de cálculo. Esta métrica, desarrollada por Thomas McCabe, permite analizar el flujo de control de un programa para identificar puntos clave en los que se deben realizar pruebas. Al reducir los casos a un número mínimo pero representativo, se optimiza el esfuerzo de prueba sin comprometer la cobertura y la calidad del sistema. Además, esta técnica es ampliamente utilizada en ingeniería de software porque facilita la

detección casos de pruebas importes y ayuda a garantizar que cada posible camino lógico en el código haya sido evaluado correctamente.

“Métrica ciclomática de McCabe, $V(G)$ de un gráfico G con n vértices y a aristas viene dada por la fórmula:

$$V(G) = a – n + 2$$

Nodos de entrada y salida. Cada nodo del gráfico, G, corresponde a un bloque de declaraciones en el programa donde el flujo es secuencial y los arcos corresponden a las ramas tomadas en el programa. Este gráfico se conoce como gráfico de flujo.
La complejidad ciclomática, V(G) nos proporciona dos cosas:

1. Encontrar el número de caminos independientes a través del programa.
2. Proporcionar un límite superior para el número de casos de prueba que deben ejecutarse para probar el programa a fondo. La complejidad es la medida que se define en términos de caminos independientes.” Chopra, R. (2018). *Software testing: Principles and Practices*. (p. 147).

Como nos menciona Chopra la complejidad ciclomática nos proporcionara el numero casos que deben de probarse, aunque es cierto que existen más formas para determinar los casos de pruebas esta es una de las más visuales ya que, con el grafo. nos podemos dar idea cuales serán estos. “Si V(G) es excesivamente alta el código que tiene un mayor riesgo debido a la dificultad de prueba. El valor umbral es 10. Cuando $V(G) > 10$, entonces la probabilidad de que el código no sea confiable es mucho mayor.” (Chopra, 2018, p.142) para Chopra el valor límite de la complejidad es de 9 pero esto dependerá del problema así mismo de la complejidad del módulo que se quiera probar. Entonces la complejidad ciclomática es una buena métrica para calcular el esfuerzo en pruebas y ver el costo que será en los posibles mantenimientos.

Pruebas del sistema utilizando complejidad ciclomática Fase 1

La complejidad ciclomática es una medida para estimar el costo de mantenimiento y obtener los números de caso que se tienen que probar a fondo dentro del sistema.
A continuación, veremos el grafo de complejidad ciclomática de la primera fase, en concreto es el método que se encarga de precargar de los catálogos. El numero en

el nodo representan el orden en el que van apareciendo en el código y los nodos pintados son las condiciones dentro del método. Otra forma de calcular la complejidad ciclomática es el número de condiciones más uno, como podemos ver, en el grafo tenemos una forma más de calcular esta que es el número, de regiones que se forman en el grafo. Para realizar el grafo de complejidad ciclomática se hace la comparación del código fuente y se van determinando los nodos del grafo. Para el nodo 1, el grafo abarcará la inicialización de las variables `usuario` y `sitios`. Para el nodo 2 se tiene una condicional la cual se utiliza para ver si el usuario tiene datos en sesión, en el grafo se pintará para distinguirla de los otros nodos.

El nodo 3 representa traer la información de la sesión para ser utilizada. El nodo 4 es una condicional `if/else` la primera se marcará con 4.1 que utiliza el servicio para traer todos los sitios y el 4.2 que solo trae los sitios para profesores. El 5 son todas las demás sentencias que se utilizan en el método y el nodo F es la culminación de la función. En el siguiente fragmento de código viene dividido como se representan los nodos en el diagrama de complejidad ciclomática.

```
public String prepaparReservacion(Map<String, Object> model)
throws IOException{
_____________________________________ 1
Usuario usuario = new Usuario();
ArrayList<Sitio> sitios;
_____________________________________

_____________________________________ 2
if(httpSession.getAttribute("userLoggedIn")==null) {
      return "acceso";
}
_____________________________________ 3
usuario = (Usuario) httpSession.getAttribute("userLoggedIn");
_____________________________________


_____________________________________ 4

if(usuario.getTipoUsuario()==0) {
       sitios= (ArrayList<Sitio>) sitioSevices.dameTodosSitio(); 4.1
}
_____________________________
else {
      sitios= (ArrayList<Sitio>)            4.2
      sitioSevices.dameTodosLosSitioAulas();
}
```

```
_____________________________________________

_____________________________________________ 5
ArrayList<Area> areas;
areas = areaService.listaArea();
ArrayList<Sede> sedes;
sedes = sedeServices.listarSede();
model.put("sedes", sedes);
model.put("areas", areas);
ArrayList<Evento> eventos = eventoService.listarEventos();
ArrayList<Equipo> equipos =
equipoService.listarEquipo();ArrayList<Usuario> usuarios =
usuariosService.dameTodosUsuarios();
colaEspera.hayTiemposMuertos();
model.put("usuarios",usuarios);
model.put("equipos",equipos);
model.put("eventos",eventos);
model.put("sitios",sitios);
return "reservacionAgregar";

} F
```

Código fuente 7. 4 Código divido para sacar la complejidad ciclomática

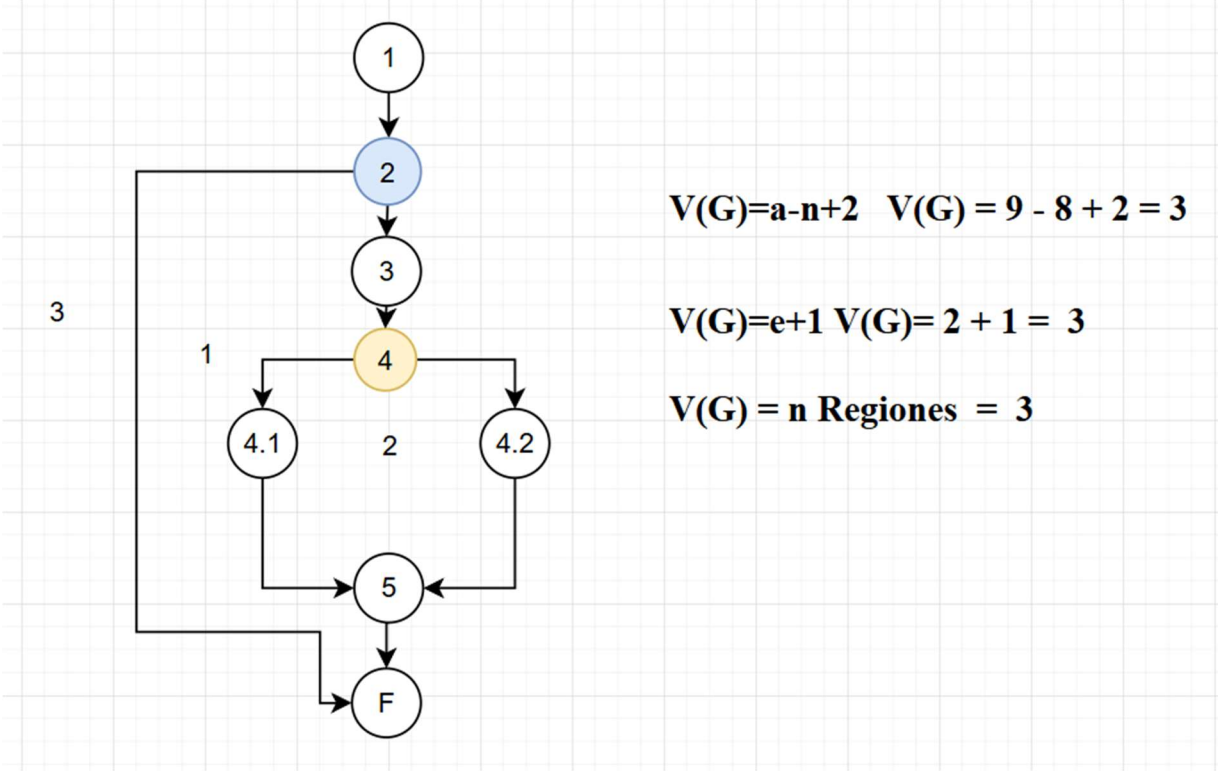


Imagen 7.1 Diagrama de complejidad ciclomática fase 1

Si determinamos que la complejidad ciclomática tiene un valor de 3 por las tres formas de calcularlo, aunque lo más formal de hacerlo es con la formula V(G)=a-n+2

las otras dos también nos dieron los mismos resultados. Los casos de pruebas mínimos que se tiene que realizar son siguientes:

1. Usuario con acceso al sistema.
2. Usuario Profesor, se cargan catálogos establecidos para él.
3. Usuario Administrador, el sistema además de cargar los mismos catálogos que el Profesor, también se le carga un catálogo de usuarios y más sitios para poder registrar.

Pruebas fase 1

1.- Usuario con acceso en el sistema. En caso de no haber accedido al sistema y tratar de hacer la reservación, el sistema redirecciona hacia el formulario de acceso, como se muestra en la siguiente imagen, la dirección del navegador indica hacia el registro de reservación, pero el sistema le muestra el formulario de acceso.

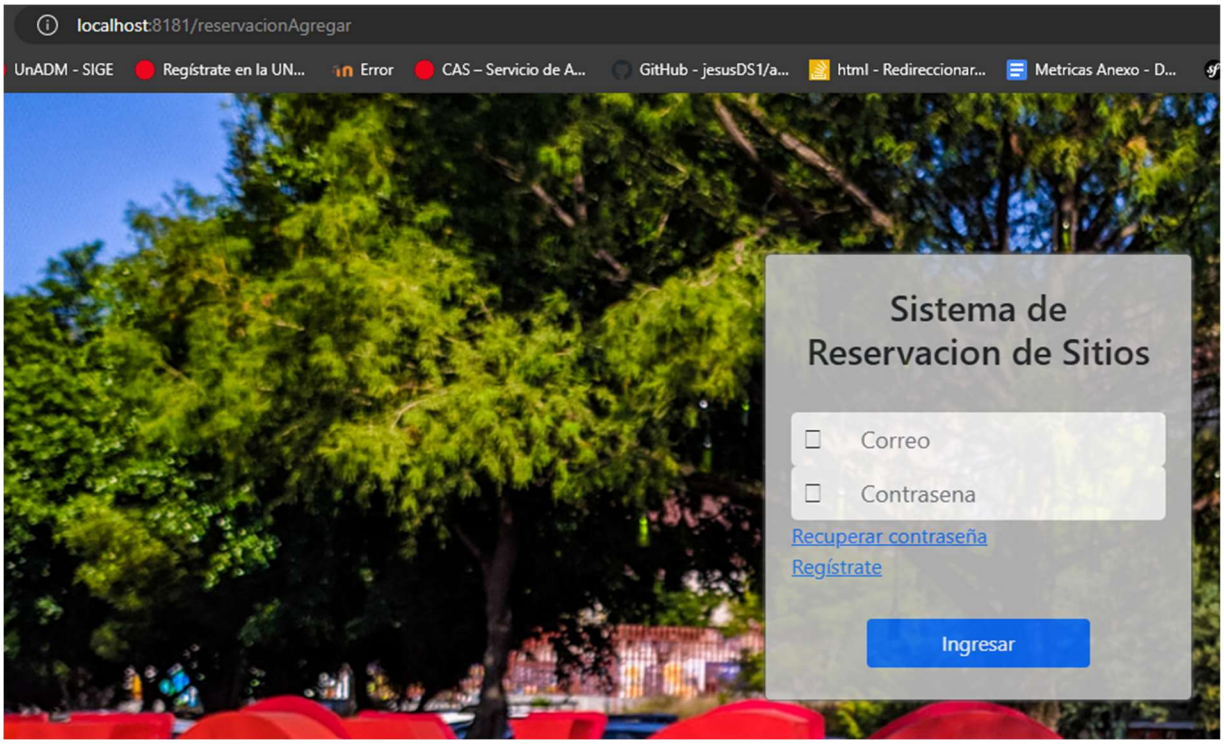


Imagen 7.2 Validación de Usuario con acceso en el sistema.

2.- Catálogo para profesores. Se precargan los sitios que solo el Profesor tiene permiso de visualizar. Como se puede apreciar en la imagen.

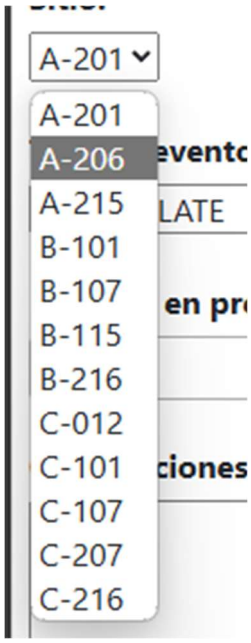


Imagen 7.3 Visualización del Catálogo de Sitios para Profesores

3.- Catálogos para administrador. Él puede ver todos los sitios además que se le precarga un catálogo con todos los profesores dados de alta en el sistema.

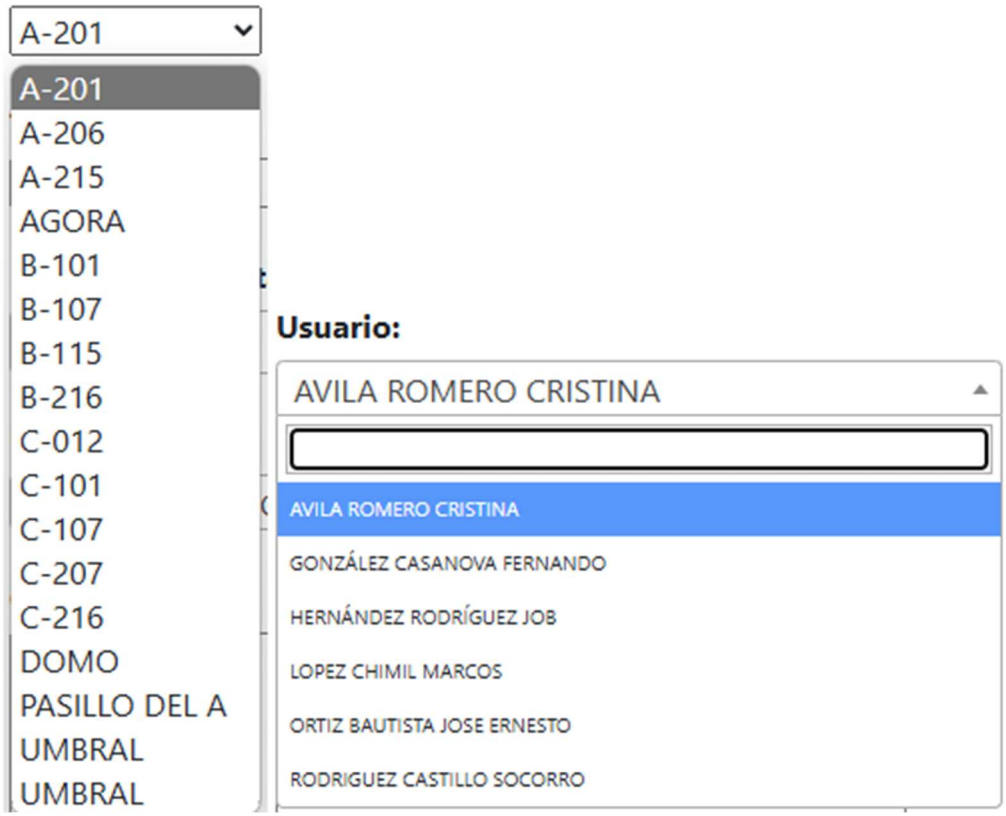


Imagen 7.4 Visualización de los Catálogos para el Administrador

Complejidad Ciclomática Fase 1. Procesamiento del primer formulario

En el siguiente grafo veremos el procesamiento que se tiene para pasar del primer formulario a la selección de horarios. Este método es muy importante ya que con esto hacemos más fácil la selección de horarios para el usuario. Como podemos ver en la siguiente imagen, tenemos $V(G) = 10$ lo que quiere decir que tenemos una complejidad ciclomática alta, además que tenemos 10 casos principales de pruebas de los cuales son los siguientes:

1. Verificar la capacidad del sitio seleccionado.
2. Condicional de ciclo.
3. Verificar que los Profesores no seleccionen días anteriores a la fecha actual.
4. Verificar que el administrador si pueda registrar fechas anteriores a la actual.
5. Condicional de ciclo
6. Verificar que los días seleccionados sean posterior a 3 días de la fecha actual solo para los Profesores.
7. Verificar que el Administrador puede reservar sin días de anticipación reservación.
8. Condicional de ciclo.
9. Preparación de horarios
10. Verificar que la Cola de espera esté disponible para utilizar el sitio.

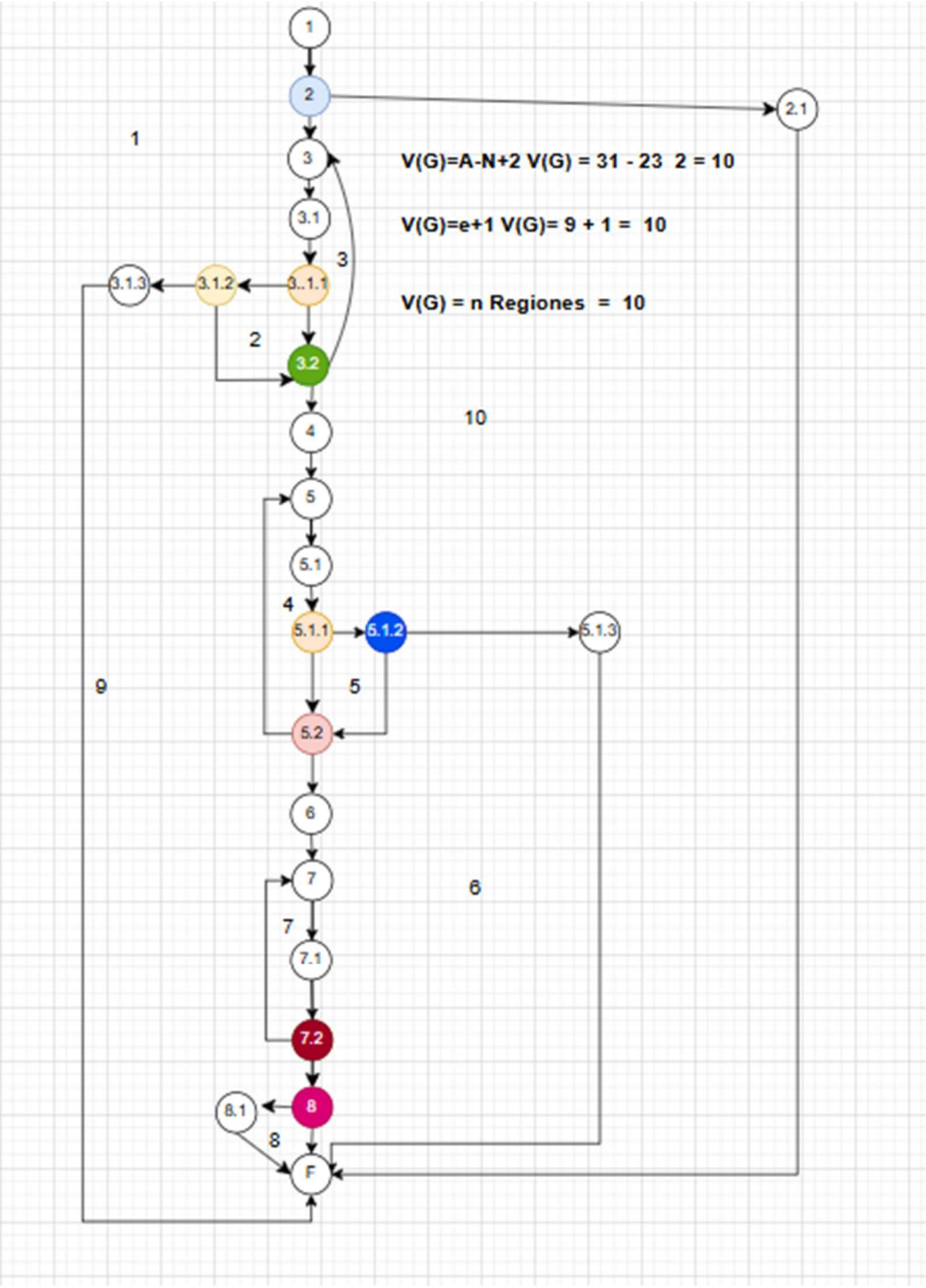


Imagen 7.5 Diagrama de complejidad ciclomática fase 1 procesamiento del primer formulario

Pruebas fase 1 primer formulario

1.- Tanto para el Administrador como para los Profesores se tiene que cumplir esta condición que los asistentes tiene que ser menor o igual a la capacidad de caso contrario le mandara un mensaje de error como se muestra en la siguiente imagen.

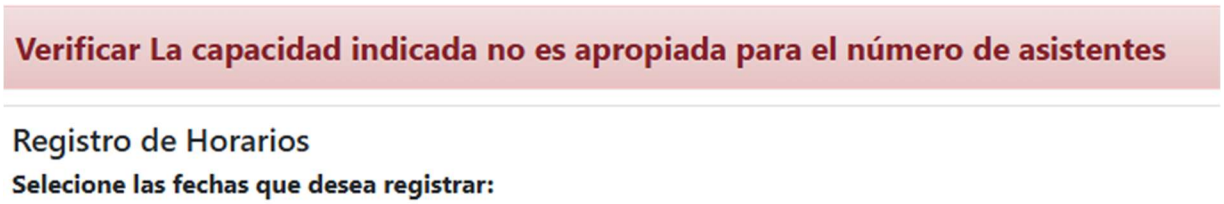


Imagen 7.6 Mensaje de la validación de la capacidad sobre pasada.

2.- Arreglo de fechas es igual a cero (inicio del ciclo).” el formulario no te deja avanzar si no seleccionas una fecha, por lo que esto no pasa, como se muestra en la siguiente imagen”.

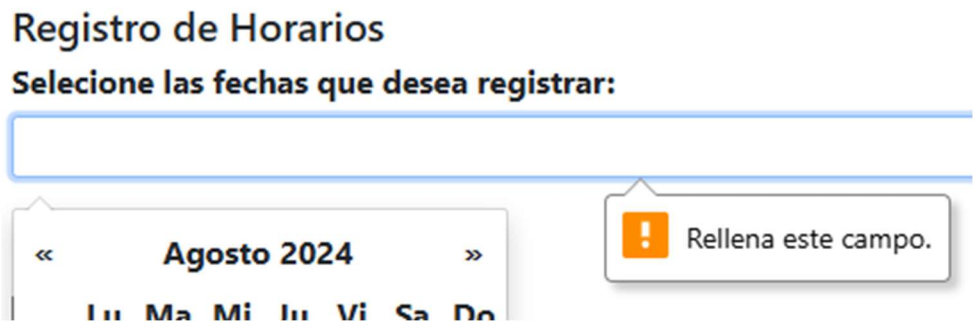


Imagen 7.7 Mensaje de la validación de selección de fechas.

3.- Para el perfil de Profesor no podrán realizar esto, en este caso el sistema de mostrará un mensaje de error como el que muestra en la siguiente imagen.

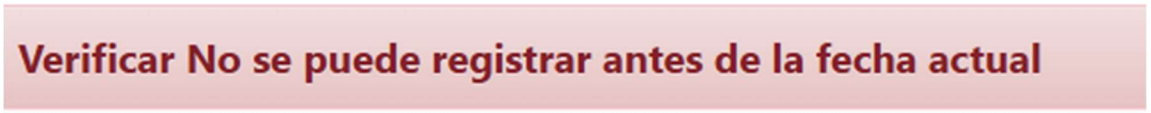


Imagen 7.8 Mensaje de la validación para perfil de profesor en caso de seleccionar días anteriores a la fecha actual.

4.- Verificar que el Administrador sí pueda registrar fechas anteriores a la actual. Al contrario del perfil de Profesor el Administrador podrá realizar estos registros para tener todo su histórico. Como se muestra en la siguiente imagen la fecha actual está marcada con color amarrillo y se seleccionan dos fechas antes por lo que en la siguiente fase se tienen que mostrar en la selección de horarios.

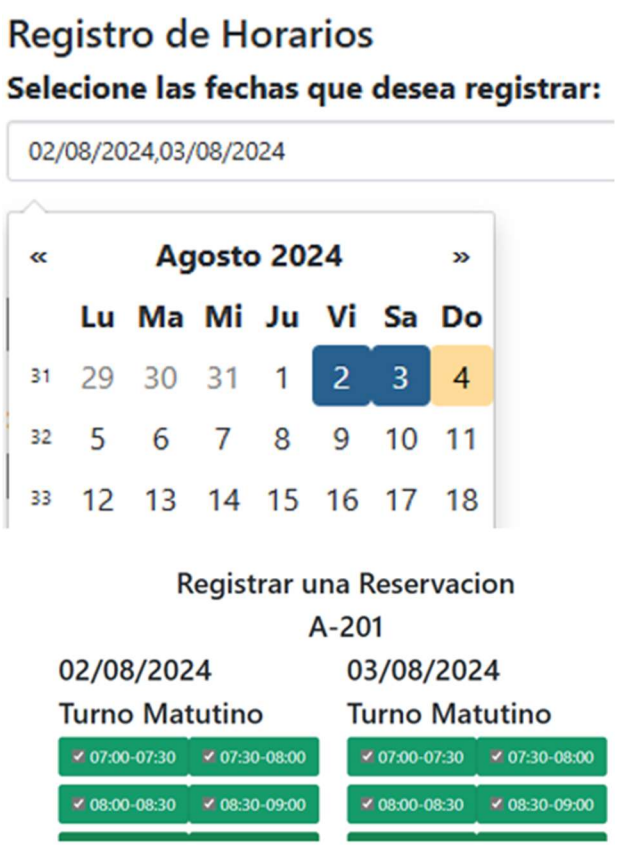


Imagen 7.9 Selección de fechas con días antes de la fecha actual para el Administrador

5.- Condicional del ciclo iterando las fechas.

6.- Verificar que los días seleccionados sean posteriores a 3 días de la fecha actual solo para el Profesor, en caso de cometerlo, se les mostrará el siguiente mensaje de error.

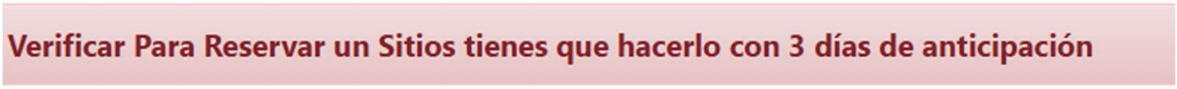


Imagen 7.10 Mensaje de la validación para perfil de Profesor en caso de no seleccionar fechas con 3 días de anticipación.

7.- Igual que en el caso de la prueba número 4, el administrador es libre de realizar todas estas reglas de negocio. Como se muestra en la siguiente imagen, las fechas seleccionas no cumplen con los 3 días de anticipación como lo tendría los Profesores.

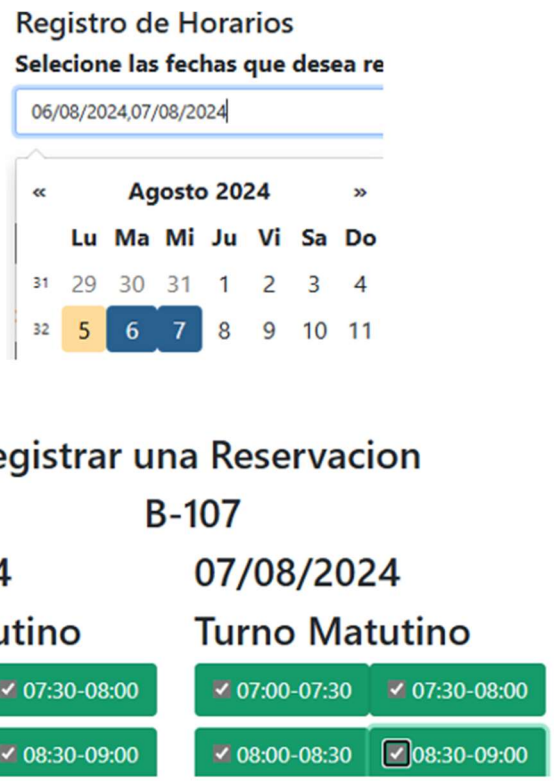


Imagen 7.11 Selección de fechas para el Administrador sin los 3 días de anticipación.

8.- Condicional del ciclo para preparar todos los horarios.

9.- Preparación de horarios, son los bloques que se mostrarán en la siguiente fase.

10.- Cola de espera, en caso de que el Sitio esté ocupado se mostrará el siguiente mensaje que podemos ver con la siguiente imagen.

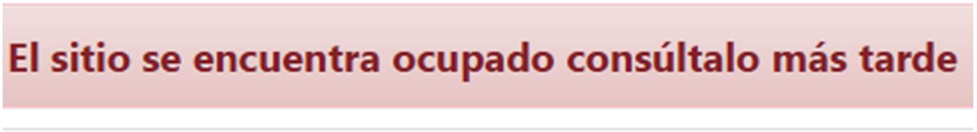


Imagen 7.12 Mensaje de la validación para la cola de espera.

Pruebas de integración utilizando complejidad ciclomática Fase 2 y 3

Para la fase 2 del proceso de reservación se calculó la complejidad ciclomática la cual es V(G) = 10, como se muestra en la siguiente imagen y para la Fase 3 es de 1, ya que solo se encarga de imprimir la información de la reservación exitosa, los casos de prueba que surgen son los siguientes:

1. Si la reservación la realizó un Profesor se traen sus datos de la sesión.
2. Si la reservación la realizó un Administrador los datos del Profesor o agente externo quien reservó la reservación se traen de la base de datos.

3. Si el usuario no seleccionó algún bloque de horario se le marcará mensaje de error.
4. Preparación de los horarios para volver a seleccionar.
5. Inicio del ciclo para transformar, separación de bloques por día.
6. Se verifica si solo selecciono un día.
7. Se verifica si son varios días.
8. Se verifica si hay más de un bloque de horario.
9. Se verifica si solo tengo uno o varios bloques de media hora.
10. Se verifica en el último bloque de horario.

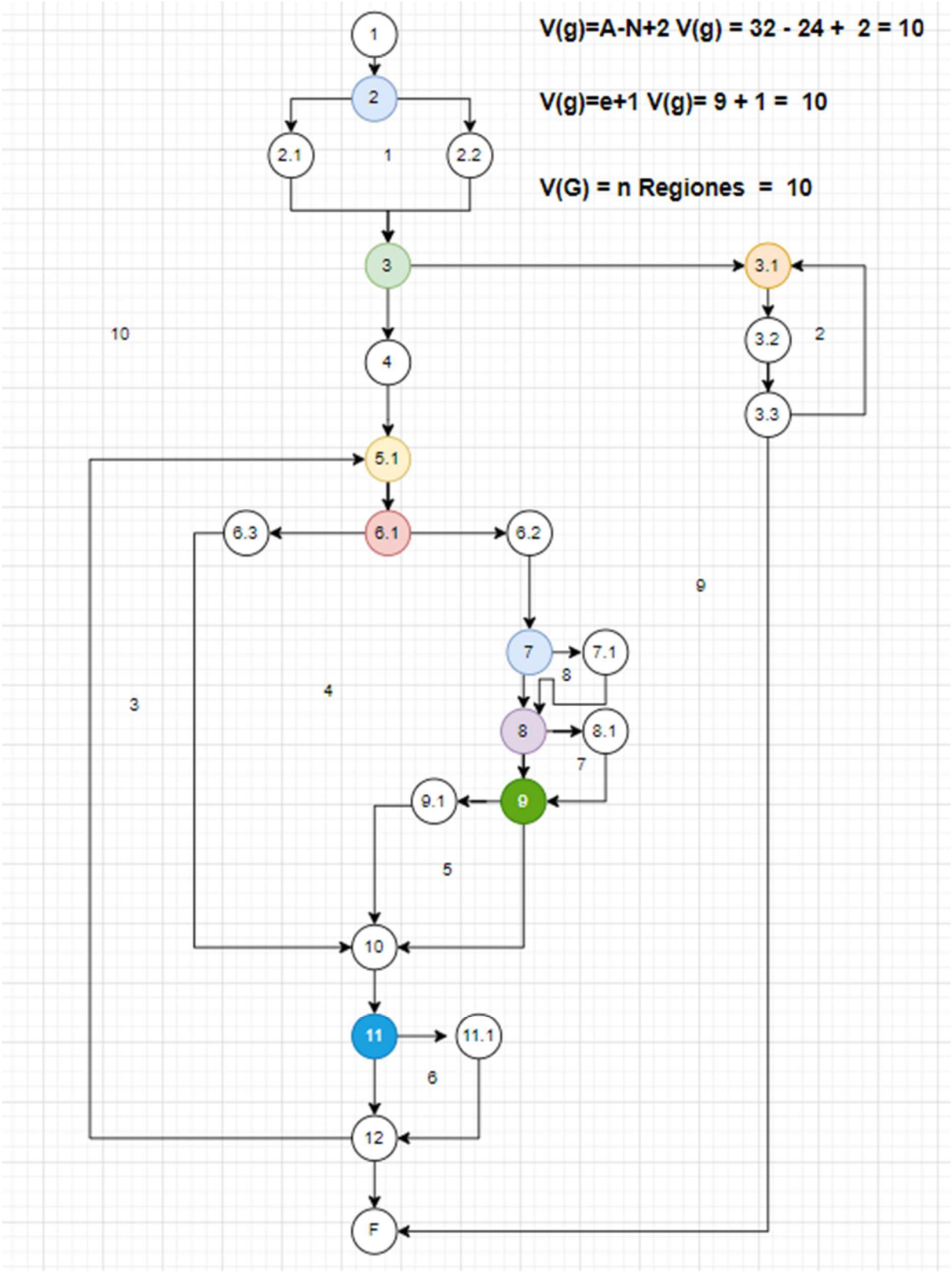


Imagen 7.13 Diagrama de complejidad ciclomática fase 2

Pruebas fase 2

Las pruebas que se realizaron durante esta fase son un poco más complicadas, ya que se debe conocer como se hace la trasformación de bloques de horarios a horas. Los casos pueden seleccionar uno o muchos bloques de horarios, así mismo si es uno o varios días, y así es como vamos avanzando en las pruebas. Se dará una breve explicación del caso en el que estamos situados para comprender mejor las pruebas.

Casos de prueba 1 y 2. Si la reservación la realizó un Profesor o un Administrador, la información se trae de manera distinta, esto es para llenar los huecos de información para la reservación. Hay que recordar que en la Fase 1 el Administrador tiene un formulario distinto para que él pueda registrar a los profesores y otras entidades fuera de la universidad.

Casos de prueba 3 y 4. Si el usuario no seleccionó algún bloque de horario se le marcará mensaje de error, como se muestra en la siguiente imagen.

Imagen 7.14 Mensaje de error al no seleccionar ningún horario

Esto se hace así para no tener reservaciones en blanco y que usuario de termine el proceso de reservación.

Casos de prueba 5 a 10. Corresponde a la transformación de bloques de horarios a fechas y días que seleccionó el usuario, para empezar a probarlo se debe tener un bloque de horario, teniendo una hora inicial, hora final y su fecha. Los casos son los siguientes:

- Se verifica si solo selecciono un día.
- Se verifica si son varios días.
- Se verifica si hay más de un bloque de horario.
- Se verifica si solo tengo uno o varios bloques de media hora.
- Se verifica en el último bloque de horario.

A continuación, se mostrarán las pruebas que validan los puntos anteriores, empezando con el bloque de media hora, aunque es uno de los casos más raros que se tienen el sistema, se adecúa para que pueda realizar esta selección. En la siguiente imagen se muestra como se seleccionan bloques de media hora salteados.

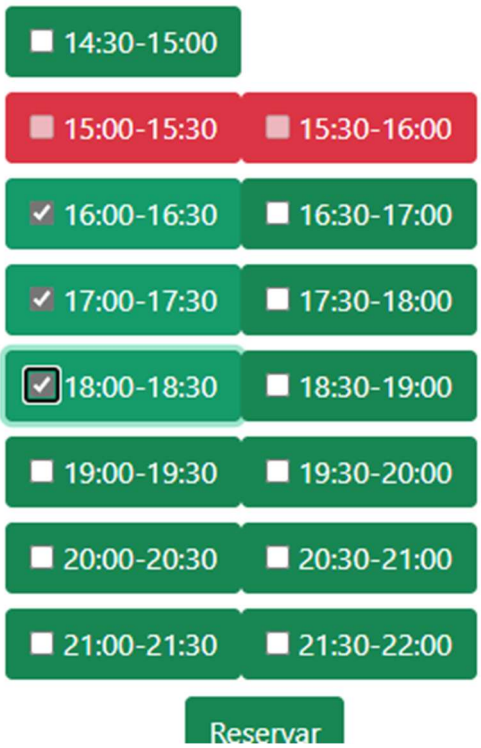
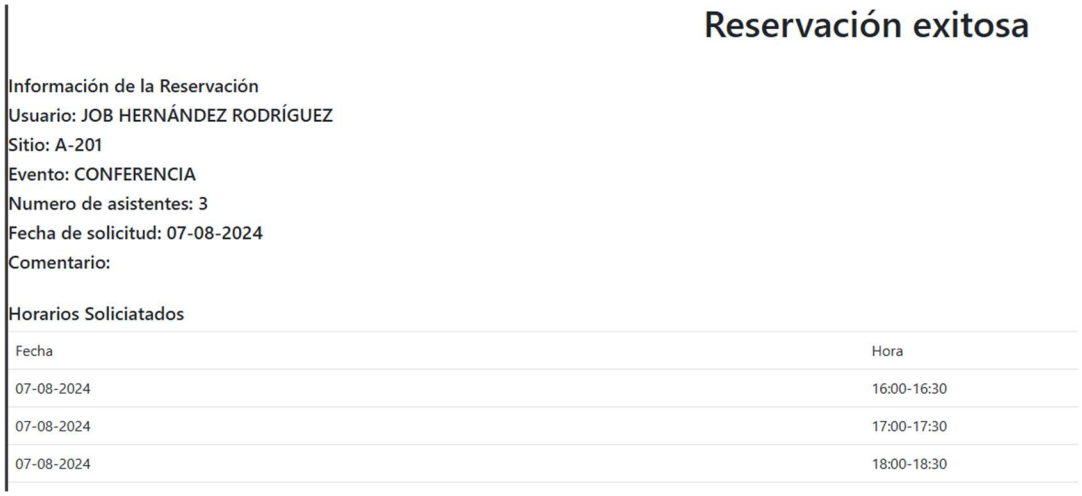


Imagen 7.14 Selección de bloques salteados de media hora

Después de presionar el botón de *Reservar* el sistema guarda la reservación de como muestra a continuación.



Reservación exitosa

Información de la Reservación
Usuario: JOB HERNÁNDEZ RODRÍGUEZ
Sitio: A-201
Evento: CONFERENCIA
Numero de asistentes: 3
Fecha de solicitud: 07-08-2024
Comentario:

Horarios Soliciatados

Fecha	Hora
07-08-2024	16:00-16:30
07-08-2024	17:00-17:30
07-08-2024	18:00-18:30

Imagen 7.15 Pantalla de registro guardado caso de bloques de media hora

El siguiente caso sería uno de los más comunes que se puede llevar a cabo y por lo mismo, es de las pruebas más importantes: cuando un usuario quiere reservar un sitio para una clase determinada se considera que el tiempo mínimo de clase en la universidad es de una hora con treinta minutos y a lo más de tres horas, así que plantearemos los dos escenarios. Por dar un ejemplo: si un profesor tiene una clase de hora y media y después una de 3 horas. Para el primer bloque de hora y media será de las 7:00 a 8:30 y el siguiente de 10:00 a 13:00. Como se muestra en la siguiente imagen.

Imagen 7.16 Pantalla de selección de horarios caso de selección de 7:00 a 8:30 y el 10:00 a 13:00

Para evitar que se reserve el horario de 9:00 a 10:00 los de bloques individualmente esta numerados del 0 al 30, cuando se va recorriendo los horarios seleccionados si existe un salto de número se considera que es otro horario. Como en este ejemplo se tienen seleccionados los bloques 0,1,2 el bloque 3,4,5 no lo están, a partir de que se

encuentra otro bloque seleccionado se vuelve a contar como otro horario distinto, en este caso 6,7,8,9,10,11.

Una vez guardada la reservación se mostrará la pantalla con los datos recopilados de la reservación, como se muestra en la siguiente imagen.

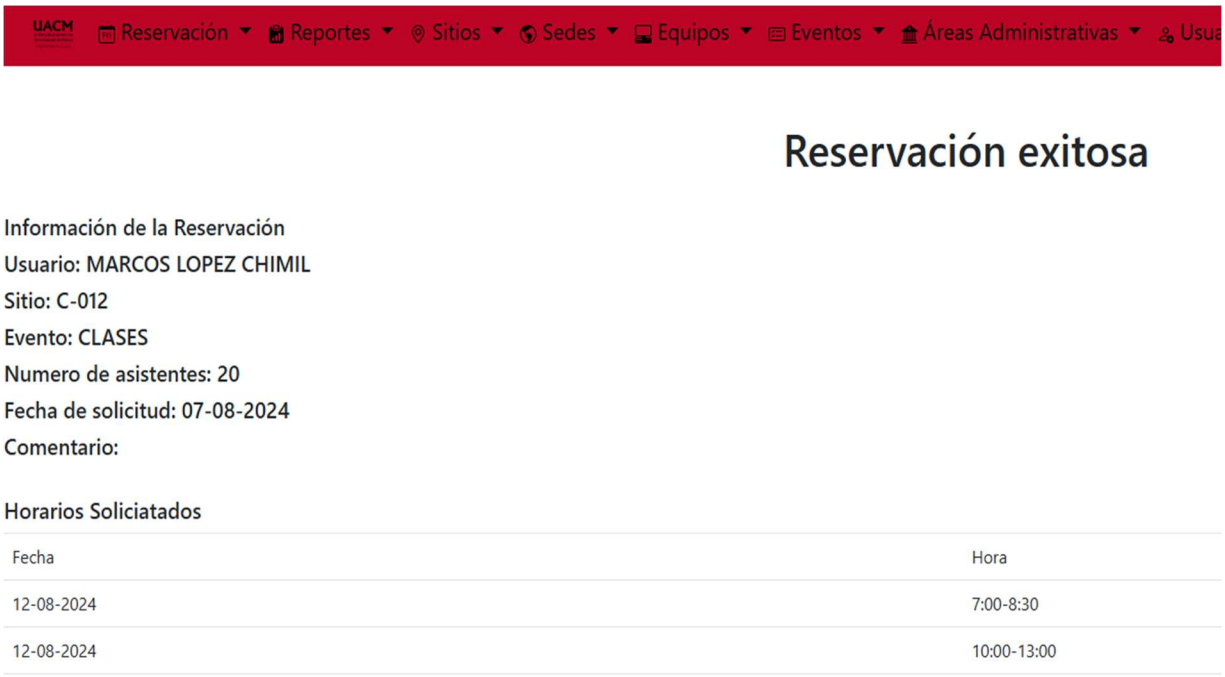


Imagen 7.16 Pantalla de confirmación de horarios caso de selección de 7:00 a 8:30 y el 10:00 a 13:00 registro guardado.

Para nuestro último caso probaremos dos de los puntos a la vez, con esto, el primer caso, es la selección de varios días y distintos horarios, el otro caso es tener una hora en el último bloque de horario en un día distinto. Como se muestra en la siguiente imagen, se seleccionaron el 28-08-2024 y 29-08-2024 los mismos bloques de 19:00 a 22:00, el último día 30-08-2024 en el horario de 7:00-8:30.

Imagen 7.17 Pantalla de selección de horarios caso de selección con tres días seleccionados y distintos bloques de horarios.

Para terminar con las pruebas, como en los otros casos, el sistema guardará la información y mostrará los datos de dicha información.

Imagen 7.18 Pantalla de confirmación de reservación exitosa de los tres días distintos.

Conclusiones

A lo largo de todo el desarrollo del sistema fue importante el seleccionar una metodología idónea ya que en las fases tempranas se perdió la comunicación con el usuario Administrador debido a la pandemia. Durante este desfase de tiempo se continuó el desarrollo del sistema implementando los catálogos, además de versiones preliminares de como realizar una reservación. Una vez que se restableció la comunicación con el usuario, fueron cambiando los requerimientos, el diseño, los procesos e interfaces graficas, pero debido a la flexibilidad del Proceso Unificado, el sistema pudo salir adelante. Aunque otras metodologías prevén estos riegos, el proceso unificado con sus distintas iteraciones permitió realizar dichos ajustes. La documentación generada, el levantamiento de requerimientos y los diagramas UML soy muy importantes para evaluar los cambios. Cabe mencionar que en el diagrama de clases del sistema, los cambios fueron mínimos, con lo que se mantuvo un buen diseño, muy escalable a cambios. Esto quiere decir que los patrones de diseño utilizados cumplieron sus propósitos.

La selección de lenguaje y el Framework fueron muy importantes para la realización del sistema, si el desarrollador no tiene buen conocimiento del lenguaje no podrá automatizar y realizar tareas mínimas, ya que la curva de aprendizaje es alta; así mismo, si conoce el lenguaje, pero no el Framework; no podrá explotar el cien por ciento de éste y posiblemente esté desarrollando software con malas prácticas. El utilizar un Framework para el desarrollo de sistemas tiene grandes ventajas, por mi parte el conocer el Framework de Spring Boot me dió una gran facilidad para el desarrollo del sistema. Actualmente conozco varios Frameworks, muchos de ellos, aunque ya tienen implementadas tecnologías y varios patrones de diseño como los que tiene Spring Boot, no tienen la flexibilidad que tiene Spring Boot. Por ejemplo el ORM de Laravel no es tan visual como el que tiene Spring, además que para realizarlo hay que tener una serie de sentencias en orden para que se pueda crea de manera automática la base de datos.

Por otra parte, existen mejoras que se pueden realizar en el sistema, muchas de ellas son más estéticas que de funcionalidad. Del funcionamiento del sistema, se

puede mejorar la cola de espera; esto para ir indicando al usuario el tiempo que tiene para realizar una reservación.

Actualmente el sistema está en un proceso de pruebas por el Administrador para ver si existe una nueva necesidad o requerimiento, además de encontrar posibles fallas para su corrección. Aún no se han realizado pruebas con el usuario Profesor. Lo ideal sería invitar a un profesor de los que más realice eventos en la Universidad para que realice pruebas en el sistema.

En concreto se tiene que elegir tanto la metodología como las tecnologías, si alguna de las dos fallas, se pueden tener tantas pérdidas monetarias y, en otros casos, vidas humanas, como mencionamos al principio del marco teórico.

Listado de archivos de configuración

Archivo de configuración 6. 1 `Gradle.Build` versión de Spring Boot 52
Archivo de configuración 6. 2 `Gradle.Build` configuración jar 52
Archivo de configuración 6. 3 `Gradle.Build` versión de Spring Boot" Dependencias"..... 54
Archivo de configuración 6. 4 `application.properties` configuración de la base de datos 56
Archivo de configuración 6. 5 `application.properties` configuración mail, puerto de salida y logs 57

Listado de código fuente

Código fuente 6. 1ORM de la clase Reservacion.....58
Código fuente 6. 2 ReservacionRepository heredando los métodos CRUD 59
Código fuente 6. 3Interfase de los servicios de ReservacionService.....59
Código fuente 6. 4 ReservacionServiceImp inyección dependencias.....61
Código fuente 6. 5 método agregarReservacion() 61
Código fuente 6. 6 ReservacionController 63
Código fuente 6. 7 método prepaparReservacion() (Carga de datos para la vista) ...65
Código fuente 6. 8 método verSitio() (prepara la selección de horarios) 66
Código fuente 6. 9 método verSitio (reglas de negocio) 67
Código fuente 6. 10 método guardandoReservación (Validando que se halla seleccionado un bloque) 68
Código fuente 6. 11 transformación de horarios 70
Código fuente 6. 12 método guardarReservación (Guardando la reservación) 71

Código fuente 7. 1 Prueba de comentario mayor a 300 caracteres. 81
Código fuente 7. 2 Prueba de capacidad mayor a 36 personas 81
Código fuente 7. 3 Prueba de reservación exitosa. 82
Código fuente 7. 4 Código divido para sacar la complejidad ciclomática 85

Referencias

Chopra, R. (2018). *Software testing: A self-teaching introduction*. Mercury Learning and Information.

Hinkula, J. (2023a). *Full stack development with Spring Boot 3 and React: Build modern web applications using the power of Java, React, and TypeScript* (Fourth edition). Packt Publishing Ltd.

Hinkula, J. (2023b). *Full stack development with Spring Boot 3 and React: Build modern web applications using the power of Java, React, and TypeScript* (Fourth edition). Packt Publishing Ltd.

Hosteltur. (s. f.). *Nuevo fallo de alto riesgo en el software del Boeing 737 MAX*. Hosteltur: Toda la información de turismo. Recuperado 14 de marzo de 2023, de https://www.hosteltur.com/129649_nuevo-fallo-de-alto-riesgo-en-el-software-del-boeing-737-max.html

Hosteltur. (2023, marzo 15). *Nuevo fallo de alto riesgo en el software del Boeing 737 MAX*. Hosteltur: Toda la información de turismo. https://www.hosteltur.com/129649_nuevo-fallo-de-alto-riesgo-en-el-software-del-boeing-737-max.html

Ivar Jacobson, Grady Booch, & Jamen Rumbaugh. (s. f.). *El proceso unificado de desarrollo de software* (Primera). Addisom Wesley.

O'Regan, G. (2019). *Concise Guide to Software Testing*. Springer International Publishing. https://doi.org/10.1007/978-3-030-28494-7

Patel, N., & Patel, K. (2018). *Java 9 Dependency Injection: Write loosely coupled code with Spring 5 and Guice*. Packt Publishing.

Soto Galindo, J. (2020, julio 4). *Función Pública expuso la declaración patrimonial de 830,000 funcionarios públicos*. El Economista. https://www.eleconomista.com.mx/politica/Funcion-Publica-expuso-la-declaracion-patrimonial-de-830000-funcionarios-publicos-20200704-0009.html

Walkinshaw, N. (2017). *Software Quality Assurance*. Springer International Publishing. https://doi.org/10.1007/978-3-319-64822-4

Weitzenfeld, A. (s. f.). *Ingenieria de software orientada a objetos con UML, Java e Internet/ Software Engineering applied to UML, Java and Internet Object*. Thomson.

Anexo

Anexo 1 Casos de uso

1. Información general Gestión de Usuarios.

1.1 Caso de uso 01

1.2 Diagrama

1.3 Resumen y Objetivos

El administrador podrá dar de alta a nuevos usuarios y administradores además los usuarios podrán modificar sus datos.

1.4 Líder / miembros del equipo de casos de uso

Job Hernández Rodríguez

1.5 Precondiciones

El usuario deberá estar registrado en el sistema institucional de la UACM.

El usuario es requisito que el usuario haya accedido previamente al sistema de reservación de sitios.

El usuario tendrá que acceder al menú de Reservaciones.

1.6 Postcondiciones

Si usuario se dió de alta se guardó su registro.

Si el usuario modificó su información se almacenaron dichos cambios. Si el usuario se dio de baja, los datos del usuario se eliminaron del sistema.

1.7 Restricciones / Problemas / Riesgos

Restricciones

Si un órgano de gobierno requiere acceso al sistema de reservación de sitios, deberá solicitar que se le haga una cuenta directamente con el administrador del sistema

Problema

Colaboración con el área de sistemas. Esto incluye la validación de ciertos campos que es determinante para el sistema.

Riesgos

Colaboración con el área de sistemas. Que el área de sistemas no incluya el Sistema de Reservación de Sitios en el menú institucional.

1.8 Eventos desencadenantes

1.9 Actor principal

Administrador del sistema

1.10 Actores secundarios

Profesor de la UACM

2. Lista de cursos de los casos de uso.

Gestión de Usuario

2.1 Curso principal (Camino feliz)

Agregar usuario

2.2 Curso(s) alternativos

Modificar Usuario

Dar de baja Usuario

2.3 Curso (s) de excepción:

Ninguno.

3. Cursos detallados

3.1 Curso Principal Agregar Usuario

#	Evento desencadenante	Respuesta del sistema
P1	Ingresa por primera vez al sistema Reservación de sitios, si no ver CA1	Se despliega un formulario para que introduzca los datos de su perfil: Nombre Apellido Paterno Apellido Materno Correo institucional Área Sede
P2	Usuario introduce los datos de su perfil y opcionalmente sube su foto	El sistema guarda la información y envía un correo al usuario con una contraseña temporal, para que este

	y presiona el botón de “Guardar”.	cree una nueva contraseña. (ver RN1), en caso de haber error el sistema manda mensaje de error.

Cursos Alternativo 1 Modificar Usuario

#	Causa o acción del Actor	Respuesta del sistema
A1	El Usuario ingresa a el apartado de modificar información. Si no ver CA2	Se muestra un formulario con la información actual del usuario: Nombre Apellido Paterno Apellido Materno Área Sede
A2	Usuario modifica los campos y presiona el botón de “Guardar”.	Sistema guarda la información en caso de que sea correcta, de lo contrario, se muestra un mensaje de error en el campo que generó el error.

Cursos Alternativo 2 Dar de baja Usuario

#	Causa o acción del Actor	Respuesta del sistema
A1	El administrador ingresa a el apartado de dar de baja usuario. Si no ver CA3	Sistema muestra la lista de todos los usuarios
A2	El administrador selecciona el usuario a borrar	Sistema muestra la información del usuario y el botón de “Eliminar”
A3	El administrador presiona el botón de “Eliminar “	Sistema manda mensaje de éxito.

3.2 Cursos Excepcionales: Ninguno.

3.3 Reglas de negocio

RN1.- Toda la información guardada en la base de datos será guardada con letras mayúsculas.

3.4 Restricciones /P*roblemas* /Riesgos: Ninguno

3.5 Casos de uso utilizados: Ninguno.

3.6 Restricciones de sincronización: Ninguno

4. Información táctica del caso de uso

4.1 Prioridad

Media

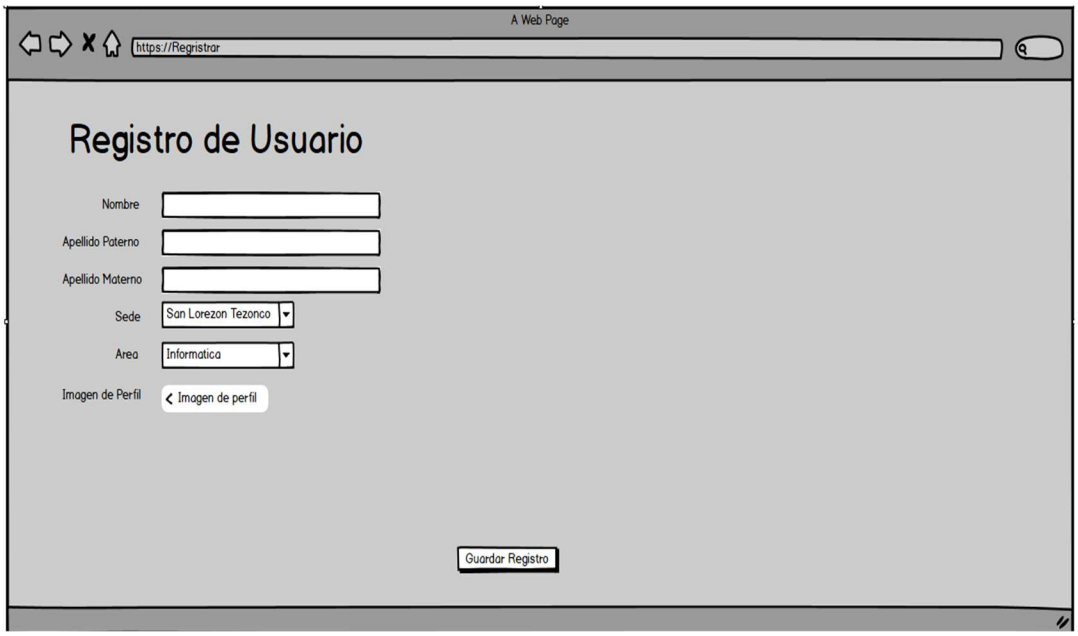
4.2 Objetivo (s) de rendimiento

Casos de uso	Estímulo	Tiempo de respuesta esperado
Agregar Usuario	Se presiona el botón de guardar nuevo usuario	0.05 segundos
Modificar Usuario	Se presiona el botón de guardar cambios	0.01 segundos
Dar de baja Usuario	Se presiona el botón de eliminar usuario	0.05 segundos

4.3 Frecuencia

Caso de uso	Frecuencia
Agregar usuario	Una vez por usuario, se esperan más de 800 registros.
Modificar usuario	Muy baja
Dar de baja Usuario	Muy baja

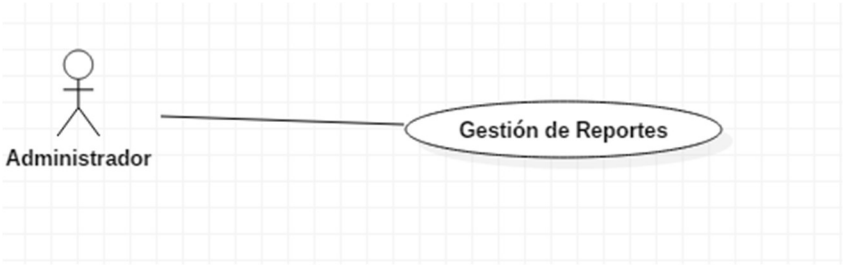
4.4 Interfaz de usuario



2. Información general Gestión de Reportes.

1.1 Caso de uso 02

1.2 Diagrama



1.3 Resumen y Objetivos

El sistema permitirá al administrador realizar diversos tipos de reportes de forma diaria y semestral.

1.4 Líder / miembros del equipo de casos de uso

Job Hernández Rodríguez

1.5 Precondiciones

El usuario administrador debe de estar en el sistema de Reservación de sitios

Los reportes generados deberán de ser posteriores a la fecha “26 de abril del 2001”

Debe de existir al menos una reservación en el sistema.

1.6 Postcondiciones

El sistema proporcionó un archivo en PDF con el formato del reporte solicitado.

1.7 Restricciones / Problemas / Riesgos

Ninguno

1.8 Eventos desencadenantes

El administrador desea Generar un reporte.

1.9 Actor principal

Administrador del sistema

1.10 Actores secundarios:

Ninguno

2. Lista de cursos de los casos de uso.

Gestión de reportes

2.1 Curso principal (Camino feliz)

Reporte de Eventos y material requerido

2.2 Curso(s) alternativos

Reporte Eventos

Reporte Por edificio

Reporte Trabajo por día

Reporte Actividades

2.3 Curso (s) de excepción

Ninguno.

3. Cursos detallados

3.1 Curso Principal Reporte de Eventos y material requerido

#	Evento desencadenante	Respuesta del sistema
P1	Administrador selecciona la opción de reporte de eventos y material requerido. De lo contrario ver CA1.	Muestra un calendario y el botón “Generar”.
P2	Selecciona una fecha del calendario y da clic en el botón “Generar Reporte”.	Si en la fecha escogida hay reservaciones, el sistema muestra el reporte, de lo contrario manda un mensaje de que no existen Reservaciones.

Curso Alternativo 1 Reporte de Eventos

#	Causa o acción del Actor	Respuesta del sistema
A1	El Administrador ingresa a reporte de Eventos. De lo contrario ver CA2	Muestra un calendario para fecha inicial y un calendario para fecha final y el botón “Generar Reporte”
A2	Selecciona una fecha inicial y una fecha final en los calendarios y da clic en el botón “Generar Reporte”	Si el rango de fechas es correcto muestra el reporte, de lo contrario manda un mensaje de error.

Curso Alternativo 2 Reporte Por edificio

#	Causa o acción del Actor	Respuesta del sistema
A1	El Administrador ingresa a reporte Por edificio. De lo contrario ver CA3	Muestra dos calendarios, la descripción de reporte botón de “Generar Reporte”.
A2	Selecciona una fecha inicial y una fecha final en los calendarios y da clic en el botón “Generar Reporte”	Selecciona una fecha inicial y una fecha final en los calendarios y da clic en el botón “Generar Reporte”

Curso Alternativo 3 Reporte Actividades

#	Causa o acción del Actor	Respuesta del sistema
A1	El Administrador ingresa a reporte de Actividades.	Muestra dos calendarios, la descripción de reporte botón de “Generar Reporte”.
A2	Selecciona una fecha inicial y una fecha final en los calendarios y da clic en el botón “Generar Reporte”.	Selecciona una fecha inicial y una fecha final en los calendarios y da clic en el botón “Generar Reporte”

Curso Alternativo 4 Reporte Trabajo por día

#	Causa o acción del Actor	Respuesta del sistema
A1	Administrador selecciona la opción de reporte de eventos y material requerido.	Muestra un calendario y el botón “Generar”
A2	Selecciona una fecha inicial en los calendarios y da clic en el botón “Generar”.	Si el rango de fechas es correcto muestra el reporte, de lo contrario manda un mensaje de error.

3.2 Cursos Excepcionales: Ninguno

3.3 Reglas de negocio: Ninguno

3.4 Restricciones /P*roblemas* /Riesgos: Ninguno

3.5 Casos de uso utilizados: Ninguno

3.6 Restricciones de sincronización: Ninguno

4. Información táctica del caso de uso

4.1 Prioridad

Baja

4.2 Objetivo (s) de rendimiento

Casos de uso	Estímulo	Tiempo de respuesta esperado
Reporte de Eventos y material requerido	Se generó el reporte de eventos y material requerido	1.5 segundos
Reporte Eventos	Se generó el reporte de eventos	1.5 segundos
Reporte Trabajo por día	Se generó el reporte de trabajo por día	0.5 segundos
Reporte Actividades	Se generó el reporte de actividades	3 segundos
Reporte Por edificio	Se generó el Reporte Por edificio	1 segundo

4.3 Frecuencia

Caso de Uso	Frecuencia
Reporte de Eventos y material requerido	1 a 2 veces por día
Reporte de Eventos	3 a 4 veces al año

Reporte Por edificio	2 a 3 al año
Reporte Actividades	2 a 3 al año
Reporte Trabajo por día	1 a 2 veces por día

4.4 Interfaz de usuario

3. Información general Gestión de Reservaciones.

1.1 Caso de uso 03

1.2 Diagrama

1.3 Resumen y Objetivos

Los usuarios podrán realizar nuevas reservaciones y eliminarlas.

1.4 Líder / miembros del equipo de casos de uso

Job Hernández Rodríguez

1.5 Precondiciones

El usuario deberá acceder con su correo institucional, al sistema de Reservación de Sitios

El usuario deberá seleccionar del menú la opción de Reservaciones

1.6 Postcondiciones

Se mandará un correo electrónico con la información de la reservación hecha o eliminada.

Si el horario a borrar es el último de la reservación, también se borrará la reservación.

1.7 Restricciones / Problemas / Riesgos

Restricciones

Si alguna organización de gobierno o educativa ajena a la UACM requiere de un sitio para la realización de evento, se tendrá que solicitar directamente con el administrador del sistema.

1.8 Eventos desencadenantes

Eventos Internos

- El usuario o administrador requiere Reservación.

Eventos Externos

Un usuario requiere de un sitio específico para la realización de un evento cultural o académico.

1.9 Actor principal

Profesor de la UACM

1.10 Actores secundarios

Administrador del sistema

2. Lista de cursos de los casos de uso.

Gestión de Reservaciones

2.1 Curso principal (Camino feliz)

Nueva Reservación Usuario Profesor

2.2 Curso(s) alternativos

Nueva Reservación Usuario Administrador

Eliminar Bloque de horario Usuario Administrador

Ver disponibilidad de horarios

Ver disponibilidad de horarios por aula

Ver mi historial de Reservaciones

2.3 Curso (s) de excepción:

Registro o eliminación de una reservación para un órgano de gobierno, institución externa de la universidad u otra dependencia, será atendida por el administrador.

3. Cursos detallados

3.1 Curso Principal Nueva Reservación Usuario Profesor

#	Evento desencadenante	Respuesta del sistema
P1	El usuario a selecciona del menú Reservaciones “Nueva reservación”	Muestra un formulario con los siguientes campos: Fechas de registro Sitio Número de asistentes Material a préstamo
P2	Llenar el formulario y presiona el botón de “Buscar”	RN2. En caso de no haber errores el

		sistema muestra una serie de botones que tienen la hora y en caso de estar de color rojo quiere decir que está ocupado, en el caso de verde está disponible. Así mimos están divididos por días, dependiendo los que selecciono el usuario.
P3	El usuario selecciona las horas donde desea realizar su reservación. Una vez finalizado la opción presiona el botón de "Reservar".	

3.2 Cursos Alternativo 1 Nueva Reservación Usuario Administrador

#	Causa o acción del Actor	Respuesta del sistema
A1	El usuario a selecciona del menú Reservaciones "Nueva reservación"	Muestra un formulario con los siguientes campos: Fechas de registro Sitio Número de asistentes Material a préstamo Usuario
A2	Llenar el formulario y presiona el botón de "Buscar".	Sistema muestra una serie de botones que tienen la hora, en caso de estar de color rojo quiere decir que está ocupado, en el caso de verde está disponible. Así mimos están divididos por días que selección el usuario. Nota el usuario administrador no tiene la restricción de 3 días de anticipación y la regla de registrar a alguien antes de la fecha actual.

A3	El usuario selecciona las horas donde desea realizar su reservación. Una vez finalizado la sección presiona el botón de “Reservar”.	Guarda la reservación

Cursos Alternativo 2 El administrador solo puede eliminar bloques de horarios

#	Causa o acción del Actor	Respuesta del sistema
A1	El usuario a selecciona del menú Reservaciones “Eliminar reservación”	Se muestran todas las reservaciones pendientes. En caso de que las reservaciones tengan más de un día éstas se muestran por separado. Al final de la información va un botón de eliminar.
A2	Usuario selecciona el día a eliminar	Se borra el día de la reservación.

Cursos Alternativo 3 Ver disponibilidad de horarios

#	Causa o acción del Actor	Respuesta del sistema
A1	El usuario a selecciona del menú Reservaciones “Disponibilidad”	Se muestran un formulario con un calendario para poder seleccionar las fechas para la revisión de la disponibilidad.
A2	Usuario selecciona “Consultar”	Sistema muestra todos los bloques de horario para todos sitios y fechas seleccionadas.

Cursos Alternativo 4 Ver disponibilidad de horarios por aula

#	Causa o acción del Actor	Respuesta del sistema
A1	El usuario a selecciona del menú Reservaciones “Disponibilidad por aulas”	Se muestran un formulario con un calendario para poder seleccionar las fechas para la revisión de la disponibilidad y una barra para seleccionar los sitios a consultar.
A2	Usuario selecciona “Consultar”	El sistema muestra todos los bloques de horario disponibles para todos los sitios y fechas seleccionados

Cursos Alternativo 5 Ver mi historial de Reservaciones

#	Causa o acción del Actor	Respuesta del sistema
A1	El usuario a selecciona del menú Reservaciones “Disponibilidad por aulas”	Se muestran un formulario con un calendario para poder seleccionar las fechas para la revisión de la disponibilidad y una barra para seleccionar los sitios a consultar.
A2	Usuario selecciona “Consultar”	El sistema muestra todos los bloques de horario disponibles para todos los sitios y fechas seleccionados

3.2 Cursos Excepcionales

Ninguno.

3.3 Reglas de negocio

RN2.- En caso de que las fechas de registro a reservar estén próximas a no más de 3 días del día actual, saldrá un mensaje de error.

RN3.- Solo el administrador puede registrar fechas anteriores al día actual.

RN4.- Solo se pueden registrar fechas en el mes actual, solo el administrador puede registrar en meses posteriores.

RN5.- Se establece la restricción de que los profesores no podrán registrar más de 5 fechas de clases a lo largo de todo el semestre.

RN6.- El administrador es el único que puede eliminar algún bloque de horario.

3.4 Restricciones /P*roblemas* /Riesgos: Ninguno

3.5 Casos de uso utilizados: Ninguno

3.6 Restricciones de sincronización:

En situaciones donde dos usuarios requieran el mismo espacio, se otorgará prioridad al usuario que haya ingresado primero. El otro usuario quedará en espera hasta que el espacio esté disponible nuevamente.

4. Información táctica del caso de uso

4.1 Prioridad

Alta

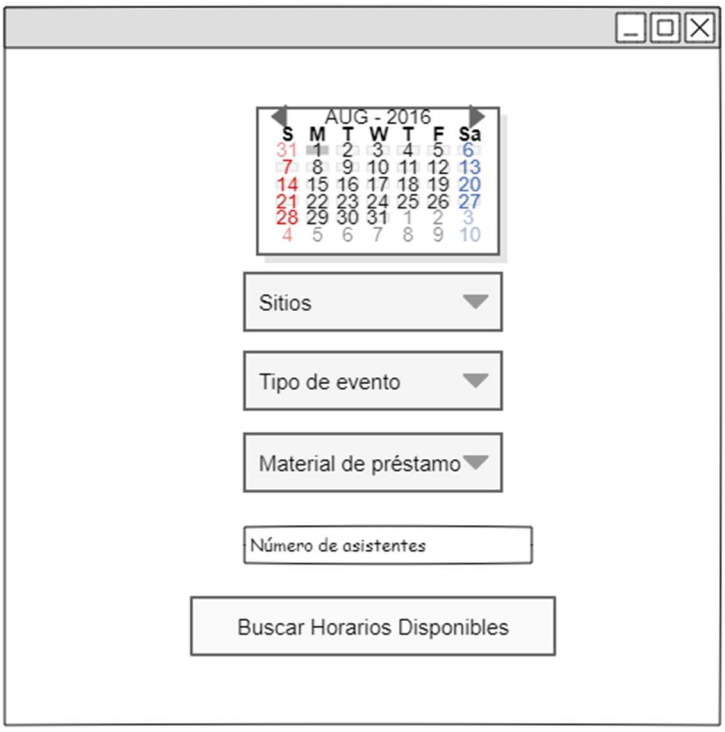
4.2 Objetivo (s) de rendimiento

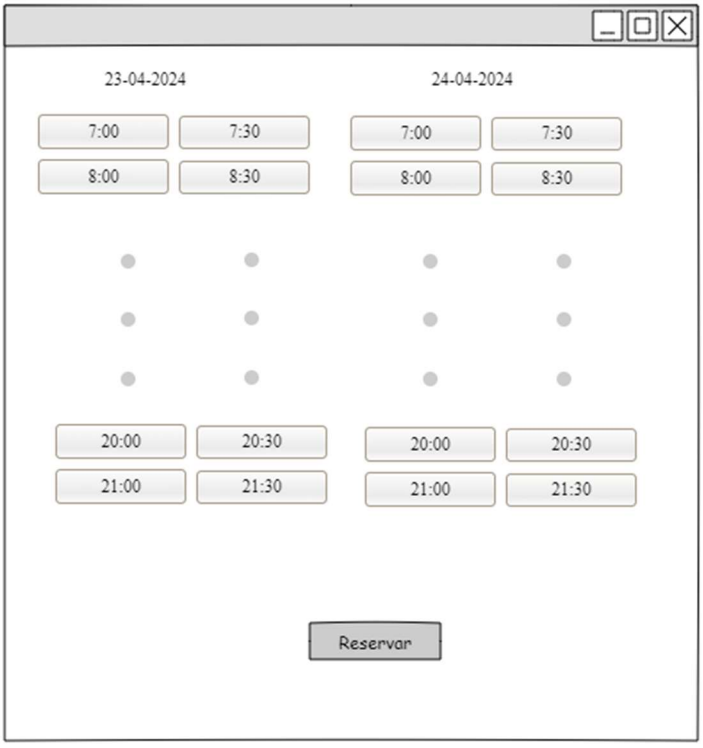
Casos de uso	Estímulo	Tiempo de respuesta esperado
Nueva Reservación Usuario Profesor	Generar una nueva reservación.	1.5 segundos
Nueva Reservación Usuario Administrador	Generar una nueva reservación.	1.5 segundos
Eliminar bloques de horarios	Eliminar un bloque de horario.	1.5 segundos
Ver disponibilidad	Ver la disponibilidad total	1.5 segundos
Ver disponibilidad de horarios por aula	Ver disponibilidad por sitios	1.5 segundos
Mis Reservaciones	Ver mi historial de reservaciones	1.5 segundos

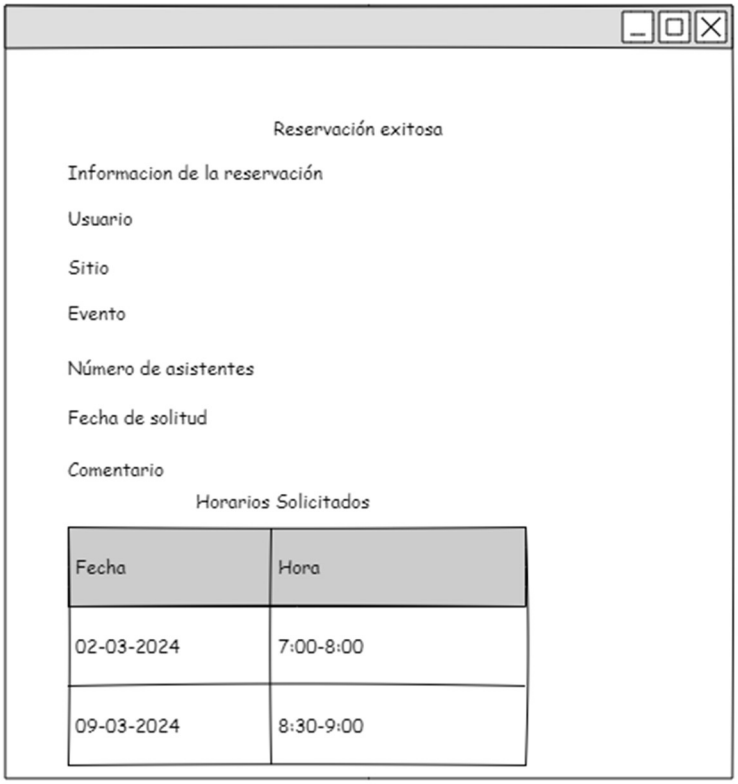
Frecuencia

Caso de uso	Frecuencia
Nueva reservación	Alta
Eliminar Reservación	Baja
Disponibilidad	Media alta
Mis reservaciones	Baja

4.4 Interfaz de usuario


AUG - 2016
S M T W T F Sa
Sitios
Tipo de evento
Material de préstamo
Número de asistentes
Buscar Horarios Disponibles


23-04-2024
24-04-2024
7:00
7:30
8:00
8:30
20:00
20:30
21:00
21:30
Reservar


Reservación exitosa
Informacion de la reservación
Usuario
Sitio
Evento
Número de asistentes
Fecha de solitud
Comentario
Horarios Solicitados
Fecha
Hora
02-03-2024
7:00-8:00
09-03-2024
8:30-9:00

4. Información general Gestión de catálogos.

Caso de uso 04

Diagrama

Resumen y Objetivos

El administrador podrá agregar, modificar y eliminar diferentes opciones de los catálogos de Sitios, Sedes, Equipos, Áreas Administrativas y Eventos del sistema.

1.4 Líder / miembros del equipo de casos de uso

Job Hernández Rodríguez

1.5 Precondiciones.

El Administrador debe de haber ingresado al Sistema de Reservación de Sitios

El Administrador deberá seleccionar alguna opción del menú de alguno de los respectivos menús de los catálogos del sistema

1.6 Postcondiciones

Los cambios se almacenaron

1.7 Restricciones / Problemas / Riesgos

Ninguna

1.8 Eventos desencadenantes

Administrador solicita gestionar catálogos.

1.9 Actor principal

Administrador

1.10 Actores secundarios

Ninguno

2. Lista de cursos de los casos de uso.

Gestión de Catálogos

2.1 Curso principal (Camino feliz)

Agregar Sitio

2.2 Curso(s) alternativos

Modificar Sitio

Agrega Sede,Equipo,Área Administrativa,Evento.

Modificar Sede,Equipo,Área Administrativa,Evento.

Eliminar Sitio,Sede,Equipo,Área Administrativa,Evento

2.3 Curso (s) de excepción:

Ninguno

3. Cursos detallados

3.1 Curso Principal Agregar Sitio

#	Evento desencadenante	Respuesta del sistema
P1	El Administrador selecciona la opción Agregar Sitio. De lo contrario CA1	Muestra los campos Nombre sitio Sede Tipo Capacidad Descripción Equipos que contiene
P2	El administrador introduce los datos correspondientes y da clic en el botón “Agregar”.	Si los datos introducidos son correctos, el sistema guarda la información, de lo contrario manda un mensaje de error.

3.2 Cursos Alternativo 1 Modificar Sitio

#	Causa o acción del Actor	Respuesta del sistema
A1	El administrador selecciona la opción de modificar sitio. De lo contrario CA2	Muestra una lista de los sitios registrados en el sistema
A2	Selecciona el sitio que desea modificar	Muestra los datos del sitio seleccionado
A3	Realiza los cambios y da clic en el botón “Guardar”.	Verifica que los datos sean correctos, si es así guarda los cambios, de lo contrario manda un mensaje de error y la causa de este.

Cursos Alternativo 2 Agrega Sede, Equipo, Área Administrativa o Evento.

#	Causa o acción del Actor	Respuesta del sistema
A1	El administrador quiere dar de alta una de las siguientes opciones Sede, Equipo, Área Administrativa, Evento, y selecciona la que desea dar de alta. De lo contrarios CA3	Sistema muestra un formulario pidiendo el nombre de la opción que haya solicitado (Sede, Equipo, Área Administrativa, Evento) y un botón de guardar.
A2	El administrador llena el formulario y presiona el botón de “Guardar”.	Si los datos introducidos son correctos el sistema guarda la información, de lo contrario manda un mensaje de error y la causa de este.

Cursos Alternativo 3 Modificar Sede, Equipo, Área Administrativa o Evento.

#	Causa o acción del Actor	Respuesta del sistema
A1	El administrador quiere modificar una de las siguientes opciones Sede, Equipo, Área Administrativa, Evento, y selecciona la que desea Modificar. De lo contrarios CA4	Sistema muestra la lista de los elementos que están disponibles a cambios ya sean de Sede, Equipo, Área Administrativa o Evento. Junto con el nombre
A2	El usuario selecciona uno de los	Si los datos introducidos son

	elementos mostrados por el sistema y escribe el nombre de la Sede, Equipo, Área Administrativa o Evento a modificar y presiona el botón de “Guardar”.	correctos el sistema guarda la información en la base de datos, de lo contrario manda un mensaje de error y la causa de este.

Cursos Alternativo 4 Eliminar Sitio, Sede, Equipo, Área Administrativa o Eventos.

#	Causa o acción del Actor	Respuesta del sistema
A1	El administrador quiere dar de bajar una de las siguientes opciones Sede, Equipo, Área Administrativa, Evento, y selecciona la que desea Modificar.	Sistema muestra la lista de los elementos que están disponibles a cambios ya sean de Sede, Equipo, Área Administrativa o Evento y un botón de “Eliminar”.
A2	El administrador selecciona qué elemento quiere dar de baja y presiona el botón de “Eliminar”.	Sistema elimina el equipo y muestra mensaje de éxito.

Cursos Alternativo 5 Mostrar información de Sitio, Sede, Equipo, Área Administrativa o Eventos.

#	Causa o acción del Actor	Respuesta del sistema
A1	Algún caso de uso requiere información de alguno de los catálogos.	Sistema muestra una lista o un elemento dependiendo del caso de uso Sede, Equipo, Área Administrativa o Evento

3.3 Cursos Excepcionales

Ninguno.

3.4 Reglas de negocio (opcional)

RN1.- Toda la información guardada en la base de datos será guardada con letras mayúsculas.

3.5 Restricciones /P*roblemas* /Riesgos: Ninguno

3.6 Casos de uso utilizados:Ninguno.

3.7 Restricciones de sincronización: Ninguno

4. Información táctica del caso de uso

4.1 Prioridad

Media

4.2 Objetivo (s) de rendimiento

Caso de uso	Estímulo	Tiempo de respuesta esperado
Agregar Sitio	Agregar un nuevo sitio	0.50 segundos
Modificar Sitio	Modificar un sitio	0.05 segundos
Agrega Sede, Equipo, Área Administrativa, Evento.	Agregar una sede, equipo, área administrativa o evento	0.50 segundos
Modificar Sede, Equipo, Área Administrativa, Evento	Modificar una sede, equipo, área administrativa o evento	0.50 segundos
Eliminar Sitio, Sede, Equipo, Área Administrativa, Evento	Eliminar un sitio, sede, equipo, área administrativa o evento	0.05 segundos

4.3 Frecuencia

Caso de uso	Frecuencia
Agregar Sitio	Baja
Modificar Sitio	Baja
Agrega Sede, Equipo, Área Administrativa, Evento.	Baja
Modificar Sede, Equipo, Área Administrativa, Evento	Baja
Eliminar Sitio, Sede, Equipo, Área Administrativa, Evento	Baja

4.4 Interfaz de usuario