



ESCUELA SUPERIOR DE INGENIERÍA

INGENIERÍA INFORMÁTICA

GESTIÓN DE CENTRO DE MEJORA DEL RENDIMIENTO Y LA SALUD

Jesús Soriano Candón

3 de septiembre de 2017



INGENIERÍA INFORMÁTICA

GESTIÓN DE CENTRO DE MEJORA DEL RENDIMIENTO Y LA SALUD

- Departamento: Ingeniería Informática
- Directora del proyecto: Lorena Gutiérrez Madroñal
- Autor del proyecto: Jesús Soriano Candón

Cádiz, 3 de septiembre de 2017

Fdo: Jesús Soriano Candón

Agradecimientos

A mi madre, por su perseverancia y por tener más ilusión que yo en que acabara este proyecto.

Resumen

Este proyecto consiste en el desarrollo de una aplicación web que permita la gestión de las actividades de CoreSport, un centro de mejora del rendimiento y la salud, así como de los usuarios del mismo. En concreto, se desarrollará la página web pública de la empresa, así como el área de clientes, donde cada usuario tendrá acceso a la gestión de actividades, datos personales, mensajería interna, dietas, entrenamientos, etc. A su vez, los administradores del sitio accederán a una gestión más amplia sobre actividades, usuarios y demás características del sistema. Por tanto, este proyecto solventará la necesidad de informatizar la gestión del centro, ofreciendo un mayor control de la misma, a la vez de facilitar el mantenimiento de la información, la comunicación con los usuarios y ofrecerles una herramienta para gestionar cómodamente sus actividades y citas sin necesidad de contactar con el centro para tal fin.

El sistema se desarrollará bajo Java EE (Enterprise Edition), usando los frameworks JSF (Java-Server Faces), PrimeFaces, EJB (Enterprise JavaBeans) y JPA (Java Persistence API), aparte de los que cada uno de ellos haga uso.

La aplicación web se realizará de tal forma que permita una fácil adaptación al uso de la misma por parte de otras instituciones, pudiendo personalizar el estilo y el logo de la interfaz de usuario de acuerdo a las necesidades de cada una de ellas.

Palabras clave: Gestión de Centro de Mejora del Rendimiento y la Salud, Gestión de Centro Deportivo, Gestión de Gimnasio, CoreSport, Gestión Centro, Gestión Actividades.

Índice general

I	Prolegómeno	1
1.	Introducción	5
1.1.	Motivación	5
1.2.	Alcance	5
1.3.	Glosario de Términos	6
1.4.	Organización del documento	6
2.	Planificación	7
2.1.	Metodología de desarrollo	7
2.1.1.	Primer Ciclo	7
2.1.2.	Segundo Ciclo	7
2.1.3.	Tercer Ciclo	8
2.1.4.	Cuarto Ciclo	8
2.1.5.	Quinto Ciclo: Pruebas	8
2.2.	Planificación del proyecto	8
2.3.	Organización	8
2.4.	Costes	9
2.5.	Riesgos	9
II	Desarrollo	11
3.	Requisitos del Sistema	15
3.1.	Situación actual	15
3.1.1.	Procesos de Negocio	15
3.1.2.	Entorno Tecnológico	16
3.1.3.	Fortalezas y Debilidades	16
3.2.	Necesidades de Negocio	16
3.3.	Objetivos del Sistema	16
3.4.	Catálogo de Requisitos	17
3.4.1.	Requisitos funcionales.	17
3.4.2.	Requisitos no funcionales	17
3.4.3.	Reglas de negocio	18
3.4.4.	Requisitos de información	18
3.5.	Solución Propuesta	19

4. Análisis del Sistema	21
4.1. Modelo Conceptual	21
4.2. Modelo de Casos de Uso	22
4.2.1. Actores	46
4.3. Modelo de Comportamiento	47
5. Diseño del Sistema	101
5.1. Arquitectura del Sistema	101
5.1.1. Arquitectura Física	101
5.1.2. Arquitectura Lógica	102
5.2. Diseño Físico de Datos	105
5.3. Diseño de la Interfaz de Usuario	105
6. Construcción del Sistema	111
6.1. Entorno de Construcción	111
6.1.1. Entorno para la Web Pública	113
6.2. Código Fuente	113
6.2.1. Módulo Web	113
6.2.2. Módulo EJB	122
6.3. Gestión de Base de Datos	124
6.3.1. <i>Entities</i>	124
6.3.2. <i>Facades</i>	125
6.3.3. Sistema de Gestión de Base de Datos (SGBD)	127
III Epílogo	129
Bibliografía	133

Índice de figuras

4.1. Modelo Conceptual de Clases UML	22
4.2. Diagrama de secuencia: Seleccionar language	47
4.3. Diagrama de secuencia: Gestión de usuarios: Registro de nuevo usuario	48
4.4. Diagrama de secuencia: Iniciar sesión	50
4.5. Diagrama de secuencia: Restablecer contraseña	51
4.6. Diagrama de secuencia: Cerrar sesión	52
4.7. Diagrama de secuencia: Cambiar contraseña	53
4.8. Diagrama de secuencia: Gestión de usuarios: Editar perfil	54
4.9. Diagrama de secuencia: Leer correo interno	56
4.10. Diagrama de secuencia: Mandar correo interno	57
4.11. Diagrama de secuencia: Notificaciones	58
4.12. Diagrama de secuencia: Gestión de servicios: Reservar plaza en una clase	59
4.13. Diagrama de secuencia: Gestión de servicios: Cancelación de reserva de una clase	60
4.14. Diagrama de secuencia: Gestión de servicios: Solicitud de cita y respuesta	61
4.15. Diagrama de secuencia: Gestión de servicios: Cancelar Cita	63
4.16. Diagrama de secuencia: Gestión de servicios: Cancelar cita (Administrador)	64
4.17. Diagrama de secuencia: Gestión de servicios: Calendario de actividades	65
4.18. Diagrama de secuencia: Gestión de servicios: Consultar reservas	66
4.19. Diagrama de secuencia: Notificaciones	67
4.20. Diagrama de secuencia: Auditorías	68
4.21. Diagrama de secuencia: Auditorías de Administradores	69
4.22. Diagrama de secuencia: Gestión de usuarios: Activar/Suspender	70
4.23. Diagrama de secuencia: Gestión de usuarios: Ver perfil (Administrador)	71
4.24. Diagrama de secuencia: Gestión de usuarios: Editar usuario (Administrador)	72
4.25. Diagrama de secuencia: Gestión de administradores: Ver perfil de administrador	74
4.26. Diagrama de secuencia: Gestión de administradores: Activar/Suspender administrador	75
4.27. Diagrama de secuencia: Gestión de administradores: Editar administrador	76
4.28. Diagrama de secuencia: Gestión de servicios: Alta servicio	78
4.29. Diagrama de secuencia: Gestión de servicios: Activar/Suspender servicio	80
4.30. Diagrama de secuencia: Gestión de servicios: Editar servicio	81
4.31. Diagrama de secuencia: Gestión de servicios: Alta clase	83
4.32. Diagrama de secuencia: Gestión de servicios: Activar/Suspender clase	85
4.33. Diagrama de secuencia: Gestión de servicios: Editar clase	86
4.34. Diagrama de secuencia: Gestión de servicios: Alta cita	88
4.35. Diagrama de secuencia: Gestión de servicios: Activar/Suspender cita	90
4.36. Diagrama de secuencia: Gestión de servicios: Editar cita	91

4.37. Diagrama de secuencia: Gestión de servicios: Subir archivo	93
4.38. Diagrama de secuencia: Gestión de servicios: Descargar archivo	95
4.39. Diagrama de secuencia: Gestión de servicios: Editar archivo	96
4.40. Diagrama de secuencia: Gestión de servicios: Eliminar archivo	98
5.1. Representación de Arquitectura de 3 Capas	102
5.2. Proceso del Patrón MVC	103
5.3. Comparativa entre modelo MVC y arquitectura de 3 capas	104
5.4. Interfaz de usuario: Inicio de sesión	105
5.5. Interfaz de usuario: Registro	106
5.6. Interfaz de usuario: Pantalla general	106
5.7. Interfaz de usuario: Pantalla con tabla de datos	107
5.8. Interfaz de usuario para dispositivos móviles: Inicio de sesión	107
5.9. Interfaz de usuario para dispositivos móviles: Pantalla general	108
5.10. Interfaz de usuario para dispositivos móviles: Menú desplegable	108
5.11. Interfaz de usuario: Diagrama de navegación	109
6.1. Arquitectura de 3 Capas con Frameworks	112
6.2. Estructura de los ficheros	114
6.3. Directorio <i>Web Pages</i>	114
6.4. Directorio <i>Source Packages</i>	118
6.5. Directorio <i>Comparativa de Lenguajes</i>	121
6.6. Directorios <i>Libraries, Enterprise Beans y Configuration Files</i>	121
6.7. Directorios módulo <i>Booking-ejb</i>	122
6.8. Instancias de <i>Service</i> en la tabla <i>services</i>	128

Índice de cuadros

4.1. UC-01: Seleccionar idioma	23
4.2. UC-02: Gestión de usuarios: Registro	23
4.3. UC-03: Gestión de usuarios: Login	24
4.4. UC-04: Gestión de usuarios: Recuperar contraseña	25
4.5. UC-05: Gestión de usuarios: Cerrar sesión	26
4.6. UC-06: Gestión de usuarios: Cambiar contraseña	26
4.7. UC-07: Gestión de usuarios: Editar perfil	27
4.8. UC-08: Gestión de servicios: Leer correo interno	27
4.9. UC-09: Gestión de servicios: Mandar un correo	28
4.10. UC-10: Notificaciones	28
4.11. UC-11: Gestión de servicios: Reservar plaza en una clase	29
4.12. UC-12: Gestión de servicios: Cancelación de reserva de una clase	29
4.13. UC-13: Gestión de servicios: Solicitar cita	30
4.14. UC-14: Gestión de servicios: Responder a solicitud de cita	31
4.15. UC-15: Gestión de servicios: Cancelar cita	31
4.16. UC-16: Gestión de servicios: Cancelar cita de un usuario	32
4.17. UC-17: Gestión de servicios: Calendario de actividades	32
4.18. UC-18: Gestión de servicios: Consultar mis reservas	33
4.19. UC-19: Notificaciones	33
4.20. UC-20: Auditoría	34
4.21. UC-21: Auditoría para administradores	34
4.22. UC-22: Gestión de usuarios: Activar o suspender	35
4.23. UC-23: Gestión de usuarios: Ver perfil	35
4.24. UC-24: Gestión de usuarios: Editar usuario	36
4.25. UC-25: Gestión de administradores: Ver perfil de administradores	36
4.26. UC-26: Gestión de administradores: Activar o suspender administrador	37
4.27. UC-27: Gestión de administradores: Editar administrador	38
4.28. UC-28: Gestión de servicios: Alta de servicio	39
4.29. UC-29: Gestión de servicios: Activar o suspender servicio	39
4.30. UC-30: Gestión de servicios: Editar servicio	40
4.31. UC-31: Gestión de servicios: Alta de clase	41
4.32. UC-32: Gestión de servicios: Activar o suspender clase	41
4.33. UC-33: Gestión de servicios: Editar clase	42
4.34. UC-34: Gestión de servicios: Alta de cita	42
4.35. UC-35: Gestión de servicios: Activar o suspender cita	43
4.36. UC-36: Gestión de servicios: Editar cita	44
4.37. UC-37: Gestión de servicios: Subir archivo	45

4.38. UC-38: Gestión de servicios: Descargar archivo 45

4.39. UC-39: Gestión de servicios: Editar archivo 46

4.40. UC-40: Gestión de servicios: Eliminar archivo 46

Parte I

Prolegómeno

La primera parte de la memoria del PFC debe contener una introducción y una planificación del proyecto.

La introducción es un capítulo que, a modo de resumen, debe contener una breve descripción del contexto de la disciplina en la que el proyecto tiene aplicación y la motivación para su desarrollo, así como del alcance previsto.

El segundo capítulo debe incluir una planificación del proyecto. La planificación deberá ajustarse a las prácticas de ingeniería en general, y de la ingeniería del software en particular. Deberá tener en cuenta los plazos, los entregables (documentos y software), los recursos (humanos y de equipamiento inventariable) y el método de ingeniería de software a emplear.

Capítulo 1

Introducción

A continuación, se describe la motivación del presente proyecto y su alcance. También se incluye un glosario de términos y la organización del resto de la presente documentación.

1.1. Motivación

*CoreSport*¹ es un centro de mejora del rendimiento y la salud que ofrece, entre otros servicios, clases dirigidas de entrenamiento funcional, TRX, nutrición, fisioterapia... y otros tantos que se irán viendo a lo largo de la memoria del presente Proyecto Fin de Carrera (PFC en adelante). Este proyecto consiste en el desarrollo de una aplicación web que permita la gestión de las actividades del centro, así como de los usuarios del mismo. En concreto, se desarrollará la página web de la empresa, con acceso a área de clientes, donde cada usuario tendrá acceso a la gestión de actividades, así como los administradores a una gestión más amplia sobre actividades y usuarios.

Actualmente no existe en dicha organización ningún proceso telemático para realizar este tipo de gestión, por lo que todos los datos de actividades y citas quedan registrados en papel. Esto produce mayor esfuerzo para la gestión y el mantenimiento de la información, así como trabajo extra en la comunicación del usuario con el centro para la gestión de sus actividades o citas, ya sea vía telefónica o personalmente en el mismo centro.

Por lo tanto, con el desarrollo del sistema, se ofrecerá una herramienta para ambas partes, administradores y usuarios, que mejorará y facilitará la forma que hasta ahora han tenido para comunicarse y gestionar sus peticiones.

1.2. Alcance

La aplicación resultante se utilizará vía online por los administradores y usuarios del centro de salud y rendimiento *CoreSport*, situado en Chiclana de la Frontera. Por lo que será accesible desde cualquier dispositivo con acceso a Internet.

No obstante, aunque el proyecto se centre en los requisitos de esta empresa, se tendrá en cuenta la posibilidad de que otros centros similares hagan uso del software. Por tanto, aspectos claves

¹Para más información acerca de CoreSport, visita su página web www.coresport.es

como las actividades ofrecidas, interfaz de usuario o logotipo de la empresa serán fácilmente adaptables a nuevos posibles centros interesados en el uso de la aplicación.

1.3. Glosario de Términos

- **Entrenamiento funcional:** Este tipo de entrenamiento se centra en sesiones cortas, dinámicas, efectivas y entretenidas. Entre otras propiedades, podemos destacar la mejora de movilidad general, tanto articular como muscular, el gran gasto calórico que conlleva o la mejora de habilidades motrices: agilidad, coordinación y equilibrio.
- **TRX (Entrenamiento en suspensión):** Se considera *entrenamiento en suspensión* a los ejercicios funcionales que se desarrollan a través de un arnés sujeto por un punto de anclaje, ajustable no elástico fabricado de distintos materiales que permite realizar un entrenamiento completo para todo el cuerpo utilizando el propio peso corporal y la resistencia a la gravedad.
- **Bulgarian Bag (Saco Búlgaro):** Equipamiento de ejercicio en forma de luna creciente usado en entrenamiento de fuerza, pliometría, entrenamiento con pesas, ejercicio aeróbico, y fitness en general.

1.4. Organización del documento

El presente documento se divide en tres partes bien diferenciadas:

- **Prolegómeno:** Esta parte contiene una introducción al proyecto, en la cual se explica al lector en qué consistirá este de una forma general junto al contexto donde será usado, además de la planificación del mismo.
- La segunda sería la parte de **desarrollo**, donde se especifican los requisitos, análisis, diseño, construcción y pruebas del sistema. Es decir, se explica en detalle el proceso de desarrollo del proyecto, desde su planteamiento hasta las pruebas realizadas una vez finalizado, incluyendo toda la ingeniería del software. Es la parte más técnica de la documentación.
- **Epílogo:** Es la última parte del documento, donde encontraremos principalmente el manual de usuario, bibliografía e información sobre la licencia de la documentación y el software.
- **Software:** El producto final se divide en dos partes:
 - La **página web** pública de la organización con la que se trabaja, de acceso libre.
 - La **aplicación web** mediante la cual los administradores y usuarios tendrán la opción de realizar su gestión. Esta se podrá acceder desde la web pública, con la diferencia que se necesitará llevar a cabo un registro para su uso. Sería la parte principal del proyecto.

Capítulo 2

Planificación

2.1. Metodología de desarrollo

Previamente al desarrollo de la aplicación web, se ha llevado a cabo el de la web pública de la empresa. Para ello, se ha realizado un diseño inicial de forma orientativa para su posterior desarrollo y pruebas de funcionamiento en diferentes dispositivos. Una vez finalizada, y al poder ser usada independientemente, se ha puesto en producción mientras se implementaba el área de cliente, la parte principal en la que se centrará el proyecto.

Para el desarrollo de la mencionada aplicación web se ha usado un modelo incremental e iterativo. Primeramente se realizó un análisis de la aplicación en general y herramientas a utilizar para su desarrollo. A partir de ahí, se inicia el proceso de implementación del producto dividido en varios ciclos, en los cuales se han ido añadiendo distintas funciones a la aplicación, obteniendo así una versión más completa al final de cada una de las fases. En cada una de ellas se analiza, diseña, implementa y prueba estas funcionalidades a añadir.

A continuación, se describirá las tareas realizadas en cada uno de estos ciclos:

2.1.1. Primer Ciclo

Primero de todo, se ha estructurado la implementación del proyecto en distintos módulos. Una vez hecho esto, se ha configurado el servidor de aplicaciones y definido los distintos tipos de usuarios del sistema, implementando seguidamente el registro e identificación de usuarios, con manejo de sesiones y de errores.

A su vez, se ha realizado el diseño general de la interfaz del software, que se ha aplicado a las pantallas de esta fase del desarrollo, como registro de usuario, inicio de sesión o la página de inicio una vez realizado el login, teniendo en cuenta que existen diferentes vistas de la interfaz dependiendo del tipo de usuario.

Esta será multilinguaje, dando opción al usuario a elegir su lenguaje por defecto seleccionándolo en el menú desplegable para tal efecto.

2.1.2. Segundo Ciclo

En este segundo ciclo se ha implementado todo lo relacionado con la gestión de usuarios: vista y edición del perfil de los usuarios registrados, gestión de los mismos por parte de administradores,

gestión de administradores por el super-administrador, comunicación entre todo tipo de usuario, histórico de acciones en el sistema, etc.

2.1.3. Tercer Ciclo

Este ciclo comprende la principal funcionalidad del sistema, la gestión de actividades y citas. Se ha desarrollado la creación de servicios, actividades, citas, etc., junto a consulta de los mismos, edición, altas, bajas... Siempre desde una interfaz intuitiva que incluye un calendario donde ver toda la actividad del usuario.

Es la parte más compleja e interesante del producto, ya que cumple el principal requisito funcional por el cual se ha realizado este proyecto.

2.1.4. Cuarto Ciclo

En este último ciclo de implementación se han añadido las funcionalidades restantes para la finalización del producto, como notificaciones, contacto o contenido de la página de inicio, por ejemplo.

2.1.5. Quinto Ciclo: Pruebas

Finalmente, se han realizado las pruebas pertinentes del software y se ha procedido a subsanar los errores encontrados y llevar a cabo pequeñas mejoras. Aclarar que en cada ciclo se han realizado pruebas manuales de la parte correspondiente, por lo que esta fase final de pruebas se ha realizado sin mucha incidencia.

2.2. Planificación del proyecto

TODO: Estimación temporal y definición del calendario básico (hitos principales e iteraciones). Desarrollo de la planificación detallada, utilizando un diagrama de Gantt. Los diagramas de Gantt que se vean correctamente (girados y divididos si hace falta).

Se debe incluir una comparación cuantitativa del tiempo y el esfuerzo realmente invertido frente al estimado y planificado. Estos datos pueden recogerse del sistema de gestión de tareas empleado para el seguimiento del proyecto.

2.3. Organización

Respecto al desarrollo de la aplicación, he sido el único desarrollador, a la vez de testeador. La tutora del proyecto ha sido Lorena Gutiérrez Madroñal, guiándome en su proceso y asegurándose que cumplía los requisitos suficientes para que sea un proyecto completo.

En cuanto al cliente, los dos socios de *CoreSport* con los que he mantenido la comunicación durante el proceso de desarrollo, y posterior a este, han sido Ángel Soriano y Cristina Saucedo. Ellos serán los administradores del sistema, con opción de añadir algunos de los trabajadores restantes en el centro, como monitores de las clases, responsables de servicios externos o recepcionista.

El resto de usuarios del software serán los clientes de la empresa, que serán los usuarios finales del producto mediante previo registro.

El hardware utilizado para el desarrollo ha sido el propio ordenador portátil del alumno, un MacBook Pro, sirviendo así de entorno de programación y pruebas mediante el uso del siguiente software:

- **macOS Yosemite (10.10), El Capitán (10.11) y Sierra (10.12)** como sistemas operativos a lo largo de todo el desarrollo.
- **Glassfish** como servidor de aplicaciones local.
- **NetBeans** como entorno de desarrollo.
- **pgAdmin** como gestor y administrador de bases de datos, usando **PostgreSQL** como sistema de gestión.
- **Subversion** como sistema de control de versiones.
- **TeXShop** como editor de textos para la documentación en **LaTeX**.

2.4. Costes

TODO: Esta sección se realizará una vez finalizado el proyecto, para comprobar el tiempo empleado y hacer el coste acorde al mismo.

Estudio y presupuesto de los costes de los recursos (humanos y materiales) descritos anteriormente, necesarios para el proyecto.

Para el cálculo de costes de personal pueden consultarse las tablas salariales de la UCA para el personal técnico de apoyo contratado laboral [?], o bien otras más ajustadas a la realidad. El cálculo del coste del personal del proyecto debe hacerse en personas-mes, y luego hacer la correspondencia al coste monetario.

2.5. Riesgos

Primeramente, tendremos en cuenta el riesgo del hardware. No sabemos cuál va a ser la vida de nuestro equipo en el que estamos desarrollando una aplicación, por lo que siempre debemos tener una copia de nuestro código para evitar posibles pérdidas de datos, ya sea por avería o rotura. Para ello, se ha utilizado *Subversion*, un sistema de control de versiones que puede ser usado desde nuestro entorno de programación *Netbeans*, brindando una herramienta esencial y sencilla de usar. Tendremos así nuestra implementación en un lugar seguro, además de beneficiarnos de las opciones que un control de versiones ofrece.

Uno de los posibles riesgos que no podremos controlar es el cambio de requisitos durante el desarrollo del proyecto. Es común que los clientes cambien o añadan funcionalidades al sistema cuando van viendo algunos resultados, probando prototipos o simplemente puntos que se les haya pasado anteriormente.

Estos cambios conllevarán un tiempo extra en el desarrollo del producto. De aquí la importancia de la fase de elicitación de requisitos y la comunicación con el cliente, para tratar de obtener

unos requisitos bien definidos desde el principio y evitar así posibles cambios o nuevas funciones a añadir.

Otro riesgo potencial en la implementación del software es la actualización de las versiones de la tecnología que estemos usando. Es decir, cuando usamos librerías, frameworks, APIs, etc., estamos expuestos a que estos se actualicen, o incluso dejen de tener soporte. Al ser, de nuevo, un factor externo al desarrollo, no se podrán tomar grandes medidas de prevención, pero sí se puede mejorar con una buena decisión respecto qué tecnologías usar en nuestro desarrollo.

En el desarrollo del software, la inmensa mayoría de las veces existirán diversas formas de conseguir un objetivo. De aquí que tengamos la opción de decidir qué lenguaje de programación, librerías, frameworks, APIs... usar. Aún así, si nos vemos en el hecho de tener que realizar cambios en alguna de ellos durante o después del desarrollo, el impacto no debería ser muy grande con una implementación limpia y bien construida.

Un ejemplo clásico de riesgo podría ser la estimación del tiempo de desarrollo del proyecto. Es común que este sea menor al tiempo real de desarrollo y, como consecuencia, existe un incremento de recursos: el tiempo de entrega del proyecto, gastos extras -ya sea para el cliente o el equipo de desarrollo- o pérdida de beneficios para negocios que no obtienen su software a tiempo. En este caso, un retraso en la estimación del tiempo no traería grandes consecuencias, los usuarios y administradores del centro seguirán llevando la misma rutina de gestión hasta que el producto esté listo para pasar a fase de producción.

Una vez el producto esté en producción, su rendimiento dependerá de un servidor contratado, por lo que es un riesgo externo a tener en cuenta. Si el servidor bajo el que la aplicación esté funcionando falla, no se tendrá acceso a la misma. Este es un riesgo que no podemos controlar, y se confía en que la empresa encargada del mismo tome medidas de seguridad suficientes para que, en el caso de fallo, el servicio no sufra caída alguna.

Parte II

Desarrollo

Capítulo 3

Requisitos del Sistema

En esta sección se detalla la situación actual de la organización y las necesidades de la misma, que originan el desarrollo o mejora de un sistema informático. Seguidamente se presentan los objetivos y el catálogo de requisitos del nuevo sistema. Finalmente se describen las tecnologías a usar en la solución al problema.

3.1. Situación actual

Como se especificó en la sección 1.1, actualmente *CoreSport* no consta de proceso telemático alguno para la gestión de usuarios y actividades, por tanto, todos los datos de los servicios que se ofrecen, grupos, citas, etc. quedan registrados en papel con sus respectivas consecuencias, como pueden ser: dificultad para la gestión y el mantenimiento de la información, menor comodidad para el usuario a la hora de obtener información o gestionar sus servicios, mayor tiempo empleado tanto para los administradores del centro como para los usuarios del mismo a la hora de realizar este tipo de gestiones, etc.

La organización dispone, sin embargo, de un software de contabilidad para la gestión de pagos mensuales y puntuales de los usuarios. Este seguirá en uso una vez el producto resultante del PFC entre en producción, por lo que este no incluirá una sección destinada a tal efecto.

3.1.1. Procesos de Negocio

Asimilando que la organización no posee ningún proceso informático para la gestión deseada, analizaremos el proceso seguido para este fin.

Toda la información de los servicios, usuarios que van a usarlos, horarios, etc., se mantienen en papel. Básicamente, por cada mes se posee un listado de los grupos de actividades colectivas con todos los clientes que pertenecen a ellos, consultando así las plazas libres buscando el grupo correspondiente y apuntando o eliminando usuarios del mismo sobre el papel.

Respecto a las citas para el resto de servicios, se gestionan mediante agenda, donde se apunta cada una de ellas, incluyendo servicio, hora y usuario e, igualmente al caso anterior, se consulta si se puede dar cita a una determinada hora para un determinado servicio.

Como se ha mencionado anteriormente en este documento, la empresa posee un software específico para la gestión de pagos, siendo este el único medio telemático que almacena los datos de usuarios

del centro.

3.1.2. Entorno Tecnológico

Al ser un centro pequeño con escasos socios y trabajadores, el entorno tecnológico de la organización es bastante reducido. El ordenador principal está situado en la recepción del centro, el cual posee el software de gestión de pagos mencionado y periféricos básicos junto a un datáfono para cobro de cuotas y servicios. Aparte de esto, varios de los administradores hacen uso de sus propios portátiles como herramienta de trabajo, ya sea para seguimiento de los clientes en algunos servicios, realizar dietas, como herramienta de comunicación, investigación, etc.

El centro también posee conexión wifi para uso interno.

Asimismo, poseen también una página web realizada a través una plataforma de desarrollo de webs, mediante una de las plantillas que ofrecían.

3.1.3. Fortalezas y Debilidades

Como fortaleza, cabe destacar que se trata de una empresa en crecimiento, con buenos profesionales del sector. Tanto es así, que se han visto obligado a trasladarse a un centro más amplio y con mejores instalaciones que el anterior, debido al incremento de usuarios y de servicios ofertados.

Su principal debilidad podría recaer en la falta de un software para la gestión de usuarios y servicios, situación que genera la creación del producto de este PFC. Decir que la organización del centro y sus trabajadores, usando este tipo de procesos para la información, ha funcionado de manera eficaz hasta el momento.

3.2. Necesidades de Negocio

Los procesos de negocios que la aplicación reflejará son los explicados anteriormente en el punto 3.1.1, pero en este caso de una forma informatizada, donde los clientes toman también cierto protagonismo a la hora de la gestión tanto de los servicios, como de los propios usuarios.

3.3. Objetivos del Sistema

De acuerdo a lo tratado hasta el momento, los principales objetivos a cumplir serían los siguientes:

- Creación de una página web conteniendo información referente a la organización, sus servicios, contacto, etc.
- Desarrollo de un sistema online que ofrezca al menos:
 - Gestión de clientes.
 - Gestión de administradores.
 - Gestión de servicios por parte de clientes y administradores, incluyendo entrenamiento grupal, individual y citas, entre otros servicios.
 - Comunicación entre usuarios.

3.4. Catálogo de Requisitos

3.4.1. Requisitos funcionales.

Los requisitos funcionales que debe cumplir el sistema son los siguientes:

- Seleccionar idioma.
- Registro.
- Login.
- Restablecer contraseña.
- Logout.
- Cambiar contraseña.
- Modificar datos de usuario.
- Comunicarse con otros usuarios del sistema.
- Gestionar clases, citas y otros servicios (alta, baja y modificación).
- Calendario de actividades.
- Notificaciones, tales como nuevos emails.
- Registro de operaciones llevadas a cabo en el sistema.
- Siguiendo opciones para administradores:
 - Gestión de usuarios (suspender, activar y modificar).
 - Gestión de administradores (total para administrador del sistema y limitada para administradores de la organización).
 - Gestión de servicios (alta, baja y modificación).

3.4.2. Requisitos no funcionales

Los requisitos no funcionales que debe cumplir el producto final del PFC estarán relacionados con la calidad del software, seguridad, etc., especificándose cada uno de ellos a continuación:

- Disponibilidad: El sistema deberá estar disponible las 24 horas del día. Esto dependerá del servidor externo donde se aloje el producto, aunque en regla general será un requisito que podrá cumplirse.
Será accesible desde cualquier dispositivo con acceso a Internet y opción de navegación.
- Fiabilidad: Deberá ser una aplicación fiable para todo tipo de usuario, que no presente errores y contenga un mínimo de seguridad, tanto en posibles ataques como en la gestión de la base de datos. Para ello, las contraseñas se almacenarán encriptadas con un algoritmo adecuado y se gestionará el acceso de usuario mediante sesión y funciones habilitadas dependiendo de su rol.
- Internacionalización del producto, al menos disponible en español e inglés.

- Alto grado de usabilidad, ya que el software será utilizado por una gran variedad de perfiles de usuario. Se mantendrá una interfaz intuitiva y de fácil acceso y uso.
- Mantenibilidad: Poseerá un fácil mantenimiento del software, ya que prácticamente no requerirá acción alguna. Además, el código tiene en cuenta la escalabilidad del producto y se podrá modificar o añadir funcionalidades de forma cómoda e intuitiva para el desarrollador.

3.4.3. Reglas de negocio

Respecto a las reglas de negocio, la organización especifica simplemente:

- El producto final deberá ser de acceso online para estar disponible desde cualquier ubicación en cualquier momento.
- Hacer visible los términos y condiciones del uso de la aplicación, que pueden ser cambiantes.

3.4.4. Requisitos de información

El sistema gestionará datos de usuarios y de los servicios que ofrece la empresa.

De los usuarios, se recogerán los siguientes campos obligatorios:

- Nombre y apellidos.
- Correo electrónico.
- Contraseña.

Y opcionales:

- Teléfono.
- Dirección.
- Ciudad.
- País.
- Código Postal.

Respecto a los servicios, se guardarán los siguientes datos:

- Nombre del servicio.
- Descripción.
- Grupos de usuarios, en caso aplicable / Usuario individual en otros casos.
- Horarios en los que se ofrece el servicio.

3.5. Solución Propuesta

El producto que se desarrollará en este PFC consistirá en una aplicación web donde, tanto administradores como clientes, interactuarán para lograr una gestión óptima de usuarios y servicios. Estará basado en los procesos de negocios explicados en 3.1.1, con la diferencia que será un proceso informatizado accesible las 24 horas del día desde cualquier localización.

Cada usuario tendrá acceso a sus datos y a los servicios ofrecidos por la organización para su gestión, así como los administradores podrán realizar acciones similares con la ventaja de poder gestionar no solo sus datos y los servicios del centro, si no los de cada uno de los clientes, ofreciendo así una aplicación de gestión completa y personalizada, con posibilidad de alta, baja y modificación de usuarios y servicios en tiempo real.

Se hará uso de un servidor de aplicaciones para el alojamiento del producto, así como de una base de datos para el almacenamiento de la información. Ambos serán externos a la organización, contratando uno de tantos servicios ofrecidos en la red que posean las cualidades y seguridad deseada para tal fin.

Capítulo 4

Análisis del Sistema

Esta sección cubre el análisis del sistema de información a desarrollar, haciendo uso del lenguaje de modelado UML.

4.1. Modelo Conceptual

El diagrama conceptual de clases UML que muestra la figura 4.1 es la presentación gráfica de las clases necesarias para llegar a la solución del problema deseada, así como sus atributos, relaciones, etc. Nos basaremos en él para la posterior programación del software, donde cada clase del diagrama UML quedará representada por una clase Java.

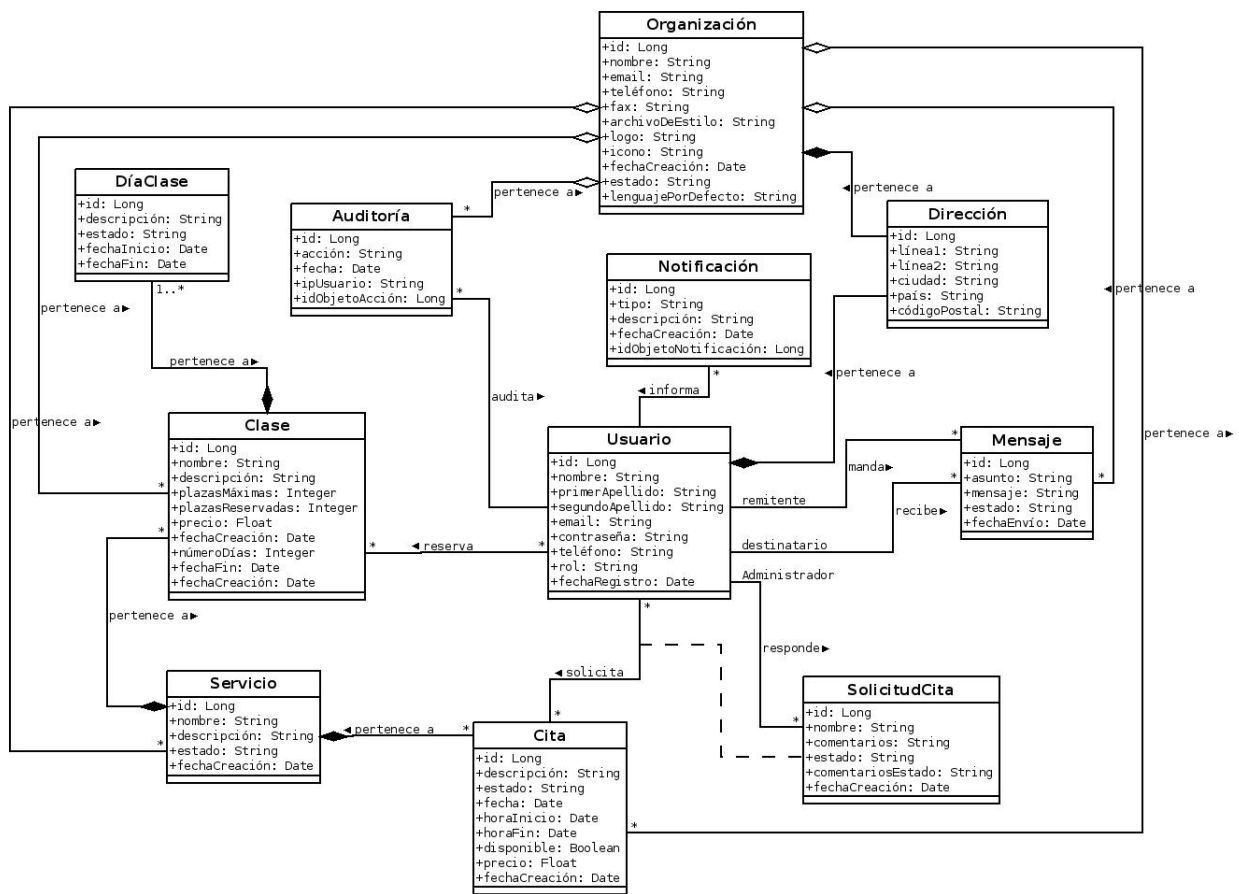


Figura 4.1: Modelo Conceptual de Clases UML

4.2. Modelo de Casos de Uso

A continuación se describirán los principales casos de uso que se llevarán a cabo en el sistema, correspondientes a los requisitos funcionales listados anteriormente en el apartado 3.4.1. Estos casos de uso se pueden emplear como mecanismo para representar las interacciones entre los actores y el sistema:

UC-01	Seleccionar idioma
Descripción	Cambiar el idioma en el que se muestra la aplicación.
Actores	Usuario, administrador o superadministrador.
Precondiciones	Ninguna.
Postcondiciones	El idioma de la aplicación se establecerá al que el usuario seleccione. Se establecerá este lenguaje por defecto para el usuario.
Escenario principal	<ol style="list-style-type: none"> 1. El usuario selecciona uno de los idiomas disponibles en la lista de selección del mismo, en cualquier pantalla de la aplicación. 2. La aplicación se mostrará en el idioma seleccionado. 3. El idioma se establece por defecto para el resto de veces que el usuario acceda a la aplicación.
Escenarios alternativos	

Cuadro 4.1: UC-01: Seleccionar idioma

UC-02	Registro
Descripción	Registro de usuario en el sistema.
Actores	Usuario.
Precondiciones	El usuario no puede haberse registrado previamente.
Postcondiciones	El usuario quedará registrado en el sistema y se mostrará la página de inicio del mismo.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra la página de registro con los campos correspondientes. 2. El usuario rellena al menos los campos obligatorios y marca la casilla de aceptación de términos y condiciones. 3. Una vez aceptado los términos y condiciones, el sistema habilita el botón de registro. 4. El usuario hace click en el botón de registro. 5. El sistema valida los datos y registra al usuario en la base de datos. 6. El sistema realiza el login del usuario y lo redirige automáticamente a la página de inicio de la aplicación.
Escenarios alternativos	<ol style="list-style-type: none"> 5.a. Los datos introducidos no son válidos: <ol style="list-style-type: none"> 5.a.1. El sistema muestra el mensaje de error correspondiente. 5.a.2. Vuelve al paso 2.

Cuadro 4.2: UC-02: Gestión de usuarios: Registro

UC-03	Login
Descripción	Inicio de sesión del usuario en el sistema.
Actores	Usuario, administrador o superadministrador.
Precondiciones	Ninguna.
Postcondiciones	El usuario se identificará en el sistema y se mostrará la página de inicio.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra la pantalla de inicio de sesión. 2. El usuario introduce su correo electrónico y contraseña. 3. El sistema valida los datos e inicia la sesión del usuario. 4. El sistema muestra la página de inicio con el menú principal.
Escenarios alternativos	<ol style="list-style-type: none"> 3.a. Los datos introducidos no son válidos: <ol style="list-style-type: none"> 3.a.1. El sistema muestra el mensaje de error correspondiente. 3.a.2. Vuelve al paso 2.

Cuadro 4.3: UC-03: Gestión de usuarios: Login

UC-04	Restablecer contraseña
Descripción	Restablecer la contraseña del usuario si esta ha sido olvidada.
Actores	Usuario, administrador o superadministrador.
Precondiciones	El usuario debe estar registrado en el sistema y su correo electrónico estar operativo.
Postcondiciones	El usuario establecerá una nueva contraseña para su login.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra la pantalla de restablecer contraseña. 2. El usuario introduce la dirección de correo electrónico que usó para su registro. 3. El sistema valida la dirección y envía un correo con el enlace para restablecer la contraseña. 4. El usuario accede al enlace recibido en el email. 5. El usuario introduce su nueva contraseña y la repite por seguridad. 6. El sistema valida los datos y la nueva contraseña para el usuario queda registrada en la base de datos.
Escenarios alternativos	<ol style="list-style-type: none"> 3.a. La dirección de correo introducida no es válida: <ol style="list-style-type: none"> 3.a.1. El sistema muestra el mensaje de error correspondiente. 3.a.2. Vuelve al paso 2. 4.a. El correo electrónico no ha sido recibido: <ol style="list-style-type: none"> 4.a.1. Vuelve al paso 1. 4.b. El enlace no es válido o ha expirado: <ol style="list-style-type: none"> 4.b.1. Vuelve al paso 1. 6.a. Las contraseñas introducidas no son válidas: <ol style="list-style-type: none"> 6.a.1. El sistema muestra el mensaje de error correspondiente. 6.a.2. Vuelve al paso 5.

Cuadro 4.4: UC-04: Gestión de usuarios: Recuperar contraseña

UC-05	Cerrar sesión
Descripción	Cerrar la sesión del usuario en el sistema.
Actores	Usuario, administrador o superadministrador.
Precondiciones	Debe ser un usuario previamente identificado.
Postcondiciones	Se terminará la sesión del usuario.
Escenario principal	<ol style="list-style-type: none"> 1. El usuario hará click en la opción para salir del sistema desde cualquier pantalla de la aplicación. 2. El sistema cerrará la sesión, quedando esta inhabilitada.
Escenarios alternativos	<ol style="list-style-type: none"> 1.a. El usuario hace click en salir del sistema a través del menú de opciones.

Cuadro 4.5: UC-05: Gestión de usuarios: Cerrar sesión

UC-06	Cambiar contraseña
Descripción	Establecer una nueva contraseña para el usuario.
Actores	Usuario, administrador o superadministrador.
Precondiciones	El usuario debe haberse identificado previamente.
Postcondiciones	El usuario establecerá una nueva contraseña para su login.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra la pantalla donde ingresar los datos para la nueva contraseña. 2. El usuario introduce los datos pedidos. 3. El sistema valida los datos y la nueva contraseña para el usuario queda registrada en la base de datos.
Escenarios alternativos	<ol style="list-style-type: none"> 3.a. Los datos introducidos no son válidos: <ol style="list-style-type: none"> 3.a.1. El sistema muestra el mensaje de error correspondiente. 3.a.2. Vuelve al paso 2.

Cuadro 4.6: UC-06: Gestión de usuarios: Cambiar contraseña

UC-07	Editar perfil
Descripción	Modificación de los datos del usuario.
Actores	Usuario, administrador o superadministrador..
Precondiciones	Debe ser un usuario previamente identificado en el sistema.
Postcondiciones	El usuario podrá modificar sus datos, excepto el email.
Escenario principal	<ol style="list-style-type: none"> 1. El usuario navega hasta su perfil. 2. El usuario hace click en la opción de editar. 3. El sistema muestra los datos del perfil en campos editables. 4. El usuario modifica o rellena los datos correspondientes. 5. El sistema modifica los datos y muestra el mensaje del proceso completado.
Escenarios alternativos	<ol style="list-style-type: none"> 5.a. Los datos introducidos no son válidos: <ol style="list-style-type: none"> 5.a.1. El sistema muestra el mensaje de error correspondiente. 5.a.2. Vuelve al paso 4.

Cuadro 4.7: UC-07: Gestión de usuarios: Editar perfil

UC-08	Leer el correo interno
Descripción	Leer alguno de los correos recibidos o enviados.
Actores	Usuario, administrador o superadministrador.
Precondiciones	El usuario debe haberse identificado previamente.
Postcondiciones	El sistema mostrará el mensaje seleccionado.
Escenario principal	<ol style="list-style-type: none"> 1. El usuario navega hasta la página deseada donde se encuentra el correo a leer, ya sea recibido o enviado. 2. El usuario hace click en el asunto del mensaje en cuestión. 3. El sistema muestra el mensaje y sus datos correspondientes. Si el mensaje no había sido previamente abierto, se marcará como mensaje leído.
Escenarios alternativos	

Cuadro 4.8: UC-08: Gestión de servicios: Leer correo interno

UC-09	Mandar un correo
Descripción	Mandar un correo interno a otro usuario de la aplicación.
Actores	Usuario, administrador o superadministrador.
Precondiciones	El usuario debe haberse identificado previamente.
Postcondiciones	Se mandará el mensaje redactado al usuario seleccionado.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra la página para redactar un nuevo email. 2. El usuario selecciona destinatario e introduce asunto y el mensaje a mandar. 3. El sistema valida los datos y manda el correo a la persona seleccionada.
Escenarios alternativos	<ol style="list-style-type: none"> 3.a. Los datos introducidos no son válidos: <ol style="list-style-type: none"> 3.a.1. El sistema muestra el mensaje de error correspondiente. 3.a.2. Vuelve al paso 2.

Cuadro 4.9: UC-09: Gestión de servicios: Mandar un correo

UC-10	Notificaciones
Descripción	Registro de notificaciones destinadas al usuario.
Actores	Usuario, administrador o superadministrador.
Precondiciones	Debe ser un usuario previamente identificado.
Postcondiciones	El sistema mostrará las notificaciones que se han generado para el usuario en las fechas indicadas.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema mostrará la página correspondiente al histórico de notificaciones. 2. El usuario seleccionará el periodo en el que desea ver las notificaciones y el tipo de notificación si lo desea. 3. El sistema muestra las notificaciones correspondientes.
Escenarios alternativos	

Cuadro 4.10: UC-10: Notificaciones

UC-11	Reservar plaza en una clase
Descripción	El usuario podrá reservar una plaza en una clase.
Actores	Usuario, administrador o superadministrador.
Precondiciones	El usuario debe haberse identificado previamente y debe haber alguna plaza libre en la clase.
Postcondiciones	El usuario tendrá una plaza reservada en una clase.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra las clases disponibles con su información principal. 2. El usuario hace clic en ver la información de la clase. 3. El sistema muestra la página con los detalles de la clase. 2. El usuario hace clic en la opción de reservar plaza. 3. El sistema realiza la reserva y muestra el mensaje.
Escenarios alternativos	<ol style="list-style-type: none"> 2.a. La clase a solicitar no dispone de plazas libres. <ol style="list-style-type: none"> 2.a.1. El icono para solicitar plaza no se mostrará.

Cuadro 4.11: UC-11: Gestión de servicios: Reservar plaza en una clase

UC-12	Cancelación de reserva de una clase
Descripción	El usuario cancelará la reserva de una clase, quedando así su plaza libre.
Actores	Usuario, administrador o superadministrador.
Precondiciones	El usuario debe haberse identificado previamente.
Postcondiciones	El usuario se dará de baja en una clase, quedando su plaza libre.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema mostrará las clases disponibles. 2. El usuario hace clic en ver la información de la clase. 3. El sistema muestra la información de la clase. 4. El usuario hará click en la opción correspondiente para darse de baja de esa clase. 5. El sistema dará de baja al usuario, quedando así su plaza libre en la clase.
Escenarios alternativos	

Cuadro 4.12: UC-12: Gestión de servicios: Cancelación de reserva de una clase

UC-13	Solicitar cita
Descripción	El usuario podrá solicitar cita de un determinado servicio a la hora seleccionada.
Actores	Usuario y administrador o superadministrador.
Precondiciones	El usuario y administrador deben haberse identificado previamente.
Postcondiciones	Se le asignará la cita seleccionada al usuario.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra las citas disponibles. 2. El usuario selecciona la cita y envía la solicitud. 3. El sistema marca la cita como pendiente y no estará disponible para el resto de usuarios. 4. El sistema hace llegar la solicitud a los administradores. 5. El administrador acepta la solicitud del usuario. 6. El sistema registra la cita y manda una notificación de aceptación al usuario.
Escenarios alternativos	<ol style="list-style-type: none"> 3.a. El usuario es administrador o superadministrador: <ol style="list-style-type: none"> 3.a.1. El sistema marca la cita como aceptada y no estará disponible para el resto de usuarios. 3.a.2. El escenario acabaría en este punto. 5.a. El administrador declina la solicitud de cita del usuario: <ol style="list-style-type: none"> 5.a.1. El sistema envía una notificación al usuario. 5.a.2. Vuelve al punto 1.

Cuadro 4.13: UC-13: Gestión de servicios: Solicitar cita

UC-14	Responder a solicitud de cita
Descripción	Respuesta de una solicitud de cita por parte del administrador o superadministrador.
Actores	Administrador o superadministrador.
Precondiciones	El administrador debe haberse identificado previamente.
Postcondiciones	La solicitud de cita del usuario quedará respondida.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra el listado de citas. 2. El administrador selecciona la cita que desea responder. 3. El sistema muestra la información de la cita y opciones disponibles. 4. El administrador selecciona la opción correspondiente (aceptar o rechazar la solicitud). 5. El sistema registra la respuesta, y realiza los cambios oportunos en la cita, y notifica al usuario.
Escenarios alternativos	

Cuadro 4.14: UC-14: Gestión de servicios: Responder a solicitud de cita

UC-15	Cancelar cita
Descripción	Cancelación de una cita o solicitud de cita por parte del propio usuario.
Actores	Usuario, administrador o superadministrador..
Precondiciones	El usuario debe haberse identificado previamente.
Postcondiciones	La cita o solicitud de cita del usuario quedará cancelada.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra el listado de citas. 2. El usuario selecciona la cita que desea cancelar. 3. El sistema muestra la información de la cita y opciones disponibles. 4. El usuario selecciona cancelar la cita. 5. El sistema cancela la cita dejándola libre nuevamente, y notifica a los administradores.
Escenarios alternativos	

Cuadro 4.15: UC-15: Gestión de servicios: Cancelar cita

UC-16	Cancelar cita de un usuario
Descripción	Cancelación de una cita por parte del administrador.
Actores	Administrador o superadministrador..
Precondiciones	El usuario debe haberse identificado previamente como administrador.
Postcondiciones	La cita del usuario quedará cancelada por parte del administrador.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra el listado de citas. 2. El administrador selecciona la cita que desea cancelar. 3. El sistema muestra la información de la cita y opciones disponibles. 4. El administrador selecciona cancelar la cita. 5. El sistema cancela la cita dejándola libre nuevamente y notifica al usuario.
Escenarios alternativos	

Cuadro 4.16: UC-16: Gestión de servicios: Cancelar cita de un usuario

UC-17	Calendario de actividades
Descripción	El usuario dispondrá de un calendario donde ver todas las clases y citas.
Actores	Usuario, administrador o superadministrador..
Precondiciones	El usuario debe haberse identificado previamente.
Postcondiciones	El sistema mostrará el calendario con las clases y citas del usuario.
Escenario principal	<ol style="list-style-type: none"> 1. El usuario navegará hasta la página correspondiente del calendario. 2. El sistema mostrará el calendario con los datos de actividades y citas de todos los servicios, distinguiendo por colores las pasadas, reservadas, disponibles o no disponibles.
Escenarios alternativos	

Cuadro 4.17: UC-17: Gestión de servicios: Calendario de actividades

UC-18	Consultar mis reservas
Descripción	El usuario, administrador o superadministrador podrá ver la lista de sus clases y citas reservadas o pendientes.
Actores	Usuario, administrador o superadministrador.
Precondiciones	El usuario debe haberse identificado previamente.
Postcondiciones	El sistema mostrará el todas las clases y citas del usuario.
Escenario principal	<ol style="list-style-type: none"> 1. El usuario navegará hasta la página correspondiente de reservas realizadas. 2. El sistema mostrará la lista de clases y citas de todos los servicios que el usuario haya reservado.
Escenarios alternativos	

Cuadro 4.18: UC-18: Gestión de servicios: Consultar mis reservas

UC-19	Notificaciones
Descripción	El sistema notificará acciones como nuevo correo recibido, cita o clase cancelada, etc.
Actores	Usuario, administrador o superadministrador..
Precondiciones	El usuario debe haberse identificado previamente.
Postcondiciones	El usuario dispondrá de notificaciones de determinadas acciones del sistema.
Escenario principal	<ol style="list-style-type: none"> 1. El usuario se dirigirá a la página correspondiente de notificaciones. 2. El sistema mostrará las últimas notificaciones por orden cronológico, marcando como vistas las nuevas si existiesen.
Escenarios alternativos	

Cuadro 4.19: UC-19: Notificaciones

UC-20	Auditoría
Descripción	Registro de operaciones llevadas a cabo en el sistema por el usuario.
Actores	Usuario.
Precondiciones	Debe ser un usuario previamente identificado.
Postcondiciones	El sistema mostrará las acciones llevadas a cabo por el usuario en las fechas indicadas.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema mostrará la página correspondiente al histórico de acciones. 2. El usuario seleccionará el periodo del que desea ver las acciones llevadas a cabo y el tipo de acción si lo desea. 3. El sistema muestra las acciones correspondientes.
Escenarios alternativos	

Cuadro 4.20: UC-20: Auditoría

UC-21	Auditoría para administradores
Descripción	Registro de operaciones llevadas a cabo en el sistema por el administrador o los usuarios.
Actores	Administrador o superadministrador.
Precondiciones	Debe ser un usuario previamente identificado como administrador o superadministrador.
Postcondiciones	El sistema mostrará las acciones llevadas a cabo por el el propio administrador o un usuario seleccionado en las fechas indicadas.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema mostrará la página correspondiente al histórico de acciones. 2. El administrador seleccionará el periodo del que desea ver las acciones llevadas a cabo, así como el usuario y el tipo de acción si lo desea. 3. El sistema muestra las acciones correspondientes.
Escenarios alternativos	

Cuadro 4.21: UC-21: Auditoría para administradores

UC-22	Activar o suspender usuario
Descripción	Activar o suspender a un usuario específico.
Actores	Administrador o superadministrador.
Precondiciones	Debe ser un usuario previamente identificado como administrador o superadministrador.
Postcondiciones	El administrador podrá suspender o activar el usuario seleccionado.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra el listado de usuarios del sistema. 2. El administrador navega por la lista y selecciona la acción correspondiente en la casilla del usuario en cuestión. 3. El sistema ejecuta la acción seleccionada (activar/suspender).
Escenarios alternativos	

Cuadro 4.22: UC-22: Gestión de usuarios: Activar o suspender

UC-23	Ver perfil
Descripción	Los administrador podrán ver el perfil de los usuarios del sistema.
Actores	Administrador o superadministrador.
Precondiciones	Debe ser un usuario previamente identificado como administrador o superadministrador.
Postcondiciones	El administrador podrá ver el perfil del usuario seleccionado.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra el listado de usuarios del sistema. 2. El administrador navega por la lista y selecciona la acción del usuario en cuestión. 3. El sistema redirecciona al administrador a la página del perfil del usuario.
Escenarios alternativos	

Cuadro 4.23: UC-23: Gestión de usuarios: Ver perfil

UC-24	Editar usuario
Descripción	Modificación de los datos de usuarios por parte del administrador.
Actores	Administrador o superadministrador.
Precondiciones	Debe ser un usuario previamente identificado como administrador o superadministrador.
Postcondiciones	El administrador podrá modificar los datos del usuario seleccionado.
Escenario principal	<ol style="list-style-type: none"> 1. El administrador se dirige al perfil del usuario deseado siguiendo los pasos de gestión de usuario detallados en el caso de uso anterior. 2. El administrador hace click en la opción de modificar usuario. 3. El sistema muestra los datos del usuario en campos editables. 4. El administrador modifica o rellena los datos correspondientes. 5. El sistema modifica los datos y notifica al usuario de los cambios realizados.
Escenarios alternativos	<ol style="list-style-type: none"> 5.a. Los datos introducidos no son válidos: <ol style="list-style-type: none"> 5.a.1. El sistema muestra el mensaje de error correspondiente. 5.a.2. Vuelve al paso 4.

Cuadro 4.24: UC-24: Gestión de usuarios: Editar usuario

UC-25	Ver administrador
Descripción	Ver perfil de administradores.
Actores	Administrador o superadministrador.
Precondiciones	Debe ser un usuario previamente identificado como administrador o superadministrador.
Postcondiciones	El administrador podrá ver los datos de otro administrador.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra el listado de administradores del sistema. 2. El administrador navega por la lista y selecciona la acción correspondiente en la casilla del administrador en cuestión. 3. El sistema redirige al administrador a la página correspondiente al perfil del usuario.
Escenarios alternativos	

Cuadro 4.25: UC-25: Gestión de administradores: Ver perfil de administradores

UC-26	Activar o suspender administrador
Descripción	Activar o suspender administrador por parte del superadministrador.
Actores	Superadministrador.
Precondiciones	Debe ser un usuario previamente identificado como superadministrador.
Postcondiciones	El superadministrador podrá activar/suspender al administrador seleccionado.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra el listado de administradores del sistema. 2. El superadministrador navega por la lista y selecciona la acción correspondiente en la casilla del administrador en cuestión. 3. El sistema ejecuta la acción (activar/suspender) y refleja el nuevo estado del administrador en la lista.
Escenarios alternativos	

Cuadro 4.26: UC-26: Gestión de administradores: Activar o suspender administrador

UC-27	Editar administrador
Descripción	Modificación de los datos de administradores.
Actores	Superadministrador.
Precondiciones	Debe ser un usuario previamente identificado como superadministrador.
Postcondiciones	El superadministrador podrá modificar los datos del administrador seleccionado.
Escenario principal	<ol style="list-style-type: none"> 1. El superadministrador se dirige al perfil del usuario deseado siguiendo los pasos de gestión de administrador detallados en el caso de uso correspondiente. 2. El superadministrador hace click en la opción de editar. 3. El sistema muestra los datos del administrador en campos editables. 4. El superadministrador modifica o rellena los datos correspondientes. 5. El sistema modifica los datos y notifica al administrador de los cambios realizados.
Escenarios alternativos	<ol style="list-style-type: none"> 5.a. Los datos introducidos no son válidos: <ol style="list-style-type: none"> 5.a.1. El sistema muestra el mensaje de error correspondiente. 5.a.2. Vuelve al paso 4.

Cuadro 4.27: UC-27: Gestión de administradores: Editar administrador

UC-28	Alta de servicio
Descripción	Añadir un servicio nuevo al sistema.
Actores	Administrador o superadministrador.
Precondiciones	Debe ser un usuario previamente identificado como administrador o superadministrador.
Postcondiciones	Se añadirá un nuevo servicio al sistema.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra la lista de servicios actuales. 2. El administrador hace click en la opción de añadir un nuevo servicio. 3. El sistema muestra una ventana con los datos necesarios para la creación del servicio. 4. El administrador introduce los datos necesarios. 5. El sistema valida los datos y registra el nuevo servicio, quedando reflejado en la lista.
Escenarios alternativos	<ol style="list-style-type: none"> 5.a. Los datos introducidos no son válidos: <ol style="list-style-type: none"> 5.a.1. El sistema muestra el mensaje de error correspondiente. 5.a.2. Vuelve al paso 4.

Cuadro 4.28: UC-28: Gestión de servicios: Alta de servicio

UC-29	Activar o suspender servicio
Descripción	Activar o suspender uno de los servicios del sistema.
Actores	Administrador o superadministrador.
Precondiciones	Debe ser un usuario previamente identificado como administrador o superadministrador.
Postcondiciones	El administrador activará o suspenderá unos de los servicios existentes en el sistema.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra la lista de servicios actuales. 2. El administrador hace click en la opción deseada (suspender/activar) de la casilla del servicio deseado. 3. El sistema suspende/activa el servicio.
Escenarios alternativos	<ol style="list-style-type: none"> 3.a. Hay clases o citas pertenecientes al servicio a suspender. <ol style="list-style-type: none"> 3.a.1. El sistema muestra el mensaje de error correspondiente. 3.a.2. Vuelve al paso 2.

Cuadro 4.29: UC-29: Gestión de servicios: Activar o suspender servicio

UC-30	Editar servicio
Descripción	Editar uno de los servicios del sistema.
Actores	Administrador o superadministrador.
Precondiciones	Debe ser un usuario previamente identificado como administrador o superadministrador.
Postcondiciones	El administrador editará los datos de unos de los servicios existentes en el sistema.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra la lista de servicios actuales. 2. El administrador hace click en la opción de editar de la casilla del servicio deseado. 3. El sistema muestra una ventana editable con los datos actuales del servicio. 4. El administrador edita los datos deseados. 5. El sistema valida los datos y registra los cambios, quedando reflejados en la lista de servicios.
Escenarios alternativos	<ol style="list-style-type: none"> 5.a. Los datos introducidos no son válidos: <ol style="list-style-type: none"> 5.a.1. El sistema muestra el mensaje de error correspondiente. 5.a.2. Vuelve al paso 4.

Cuadro 4.30: UC-30: Gestión de servicios: Editar servicio

UC-31	Alta de clase
Descripción	Añadir una clase nueva al sistema. A partir de este momento, los usuarios podrán reservar plaza en esta actividad.
Actores	Administrador o superadministrador.
Precondiciones	Debe ser un usuario previamente identificado como administrador o superadministrador.
Postcondiciones	Se añadirá una nueva clase del servicio seleccionado al sistema.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra la lista de clases actuales. 2. El administrador hace click en la opción de añadir una nueva clase. 3. El sistema muestra una nueva página con los datos necesarios para la creación de la clase. 4. El administrador introduce los datos necesarios. 5. El sistema valida los datos y registra la clase.
Escenarios alternativos	<ol style="list-style-type: none"> 5.a. Los datos introducidos no son válidos: <ol style="list-style-type: none"> 5.a.1. El sistema muestra el mensaje de error correspondiente. 5.a.2. Vuelve al paso 4.

Cuadro 4.31: UC-31: Gestión de servicios: Alta de clase

UC-32	Activar o suspender clase
Descripción	Suspender o activar uno de las clases de un determinado servicio del sistema.
Actores	Administrador o superadministrador.
Precondiciones	Debe ser un usuario previamente identificado como administrador o superadministrador.
Postcondiciones	El administrador activará o suspenderá una de las clase existentes en el sistema.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra la lista de clases actuales. 2. El administrador hace click en la opción deseada (suspender/activar) de la casilla de la clase deseada. 3. El sistema suspende/activa la clase y notifica a los usuarios que estén haciendo uso de la misma sobre la acción.
Escenarios alternativos	

Cuadro 4.32: UC-32: Gestión de servicios: Activar o suspender clase

UC-33	Editar clase
Descripción	Editar uno de las clases del sistema.
Actores	Administrador o superadministrador.
Precondiciones	Debe ser un usuario previamente identificado como administrador o superadministrador.
Postcondiciones	El administrador editará los datos de una de las clase existentes en el sistema.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra la lista de clases actuales. 2. El administrador hace click en la opción de editar de la casilla de la clase deseada. 3. El sistema muestra la página de edición de la clase con los datos actuales de la misma. 4. El administrador edita los datos deseados. 5. El sistema valida los datos y registra los cambios.
Escenarios alternativos	<ol style="list-style-type: none"> 5.a. Los datos introducidos no son válidos: <ol style="list-style-type: none"> 5.a.1. El sistema muestra el mensaje de error correspondiente. 5.a.2. Vuelve al paso 4.

Cuadro 4.33: UC-33: Gestión de servicios: Editar clase

UC-34	Alta de cita
Descripción	Añadir una cita al sistema para un determinado servicio.
Actores	Administrador o superadministrador.
Precondiciones	Debe ser un usuario previamente identificado como administrador o superadministrador.
Postcondiciones	Se añadirá al sistema una nueva cita del servicio seleccionado.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra la lista de citas actuales. 2. El administrador hace click en la opción de añadir una nueva cita. 3. El sistema navega hacia la página de creación de una cita nueva. 4. El administrador introduce los datos necesarios. 5. El sistema valida los datos y registra la nueva cita, quedando disponible para su solicitud.
Escenarios alternativos	<ol style="list-style-type: none"> 5.a. Los datos introducidos no son válidos: <ol style="list-style-type: none"> 5.a.1. El sistema muestra el mensaje de error correspondiente. 5.a.2. Vuelve al paso 4.

Cuadro 4.34: UC-34: Gestión de servicios: Alta de cita

UC-35	Activar o suspender cita
Descripción	Activar o suspender una de las citas de un determinado servicio del sistema.
Actores	Administrador o superadministrador.
Precondiciones	Debe ser un usuario previamente identificado como administrador o superadministrador.
Postcondiciones	El administrador activará o suspenderá una cita existente en el sistema.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra la lista de citas actuales. 2. El administrador hace click en la opción deseada (suspender/activar) de la cita deseada. 3. El sistema suspende/activa la cita, quedando esta inhabilitada/habilitada. 4. En caso de ser una cita con usuario asignado, se le notificará el cambio al mismo.
Escenarios alternativos	

Cuadro 4.35: UC-35: Gestión de servicios: Activar o suspender cita

UC-36	Editar cita
Descripción	Editar una cita del sistema.
Actores	Administrador o superadministrador.
Precondiciones	Debe ser un usuario previamente identificado como administrador o superadministrador.
Postcondiciones	El administrador editará los datos de una cita existente en el sistema.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra la lista de citas actuales. 2. El administrador hace click en la opción de editar en la cita deseada. 3. El sistema navega hacia la página de edición de la cita seleccionada. 4. El administrador edita los datos deseados. 5. El sistema valida los datos y registra los cambios. 6. En caso de ser una cita con usuario asignado, se le notificará el cambio al mismo.
Escenarios alternativos	<ol style="list-style-type: none"> 5.a. Los datos introducidos no son válidos: <ol style="list-style-type: none"> 5.a.1. El sistema muestra el mensaje de error correspondiente. 5.a.2. Vuelve al paso 4.

Cuadro 4.36: UC-36: Gestión de servicios: Editar cita

UC-37	Subir archivo
Descripción	Subir un archivo específico destinado a uno o varios usuarios, como entrenamientos personales, dietas, etc.
Actores	Administrador o superadministrador.
Precondiciones	Debe ser un usuario previamente identificado como administrador o superadministrador.
Postcondiciones	El administrador podrá subir un archivo especificando a qué usuario o usuarios va destinado el mismo, pero que este/estos puedan descargarlo.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra la lista de documentos actuales. 2. El administrador hace click en la opción de añadir un nuevo archivo. 3. El sistema navega hacia la página de subida de archivos. 4. El administrador introduce los datos necesarios, junto a los usuarios a los que van destinados el archivo, y selecciona el documento a subir de su dispositivo. 5. El sistema valida los datos y aloja el archivo, quedando disponible para su descarga por los usuarios especificados.
Escenarios alternativos	

Cuadro 4.37: UC-37: Gestión de servicios: Subir archivo

UC-38	Descargar archivo
Descripción	Descargar un archivo específico subido por un administrador, como entrenamientos personales, dietas, etc.
Actores	Usuario, administrador o superadministrador.
Precondiciones	<p>Debe ser un usuario previamente identificado.</p> <p>Debe haber algún documento subido destinado al usuario.</p>
Postcondiciones	El usuario podrá descargar un archivo previamente subido por un administrador y destinado al mismo.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra el listado de archivos disponibles para el usuario. 2. El usuario selecciona el documento a descargar. 3. El sistema realiza la descarga del mismo al dispositivo del usuario.
Escenarios alternativos	

Cuadro 4.38: UC-38: Gestión de servicios: Descargar archivo

UC-39	Editar archivo
Descripción	Editar los datos un archivo específico subido por un administrador, como entrenamientos personales, dietas, etc.
Actores	Administrador o superadministrador.
Precondiciones	Debe ser un usuario previamente identificado como administrador. Debe haber algún documento subido.
Postcondiciones	El administrador podrá editar un archivo previamente subido.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra el listado de archivos subidos. 2. El administrador hace click en la opción de editar en el archivo deseado. 3. El sistema navega hacia la página de edición del archivo seleccionado. 4. El administrador edita los datos deseados. 5. El sistema valida los datos y registra los cambios. 6. En caso de ser una archivo con usuario/s asignado/s, se le/s notificará la edición del documento.
Escenarios alternativos	

Cuadro 4.39: UC-39: Gestión de servicios: Editar archivo

UC-40	Eliminar archivo
Descripción	Eliminar un archivo específico subido por un administrador, como entrenamientos personales, dietas, etc.
Actores	Administrador o superadministrador.
Precondiciones	Debe ser un usuario previamente identificado como administrador. Debe haber algún documento subido.
Postcondiciones	El administrador podrá eliminar un archivo previamente subido.
Escenario principal	<ol style="list-style-type: none"> 1. El sistema muestra el listado de archivos subidos. 2. El administrador hace click en la opción de eliminar en el archivo deseado. 3. El sistema elimina el documento. 4. En caso de ser una archivo con usuario/s asignado/s, se le/s notificará la eliminación del documento.
Escenarios alternativos	

Cuadro 4.40: UC-40: Gestión de servicios: Eliminar archivo

4.2.1. Actores

Los actores que intervienen en el sistema son:

- **Usuario:** Es el usuario final del sistema. En este caso, un usuario de CoreSport, el centro de mejora de la salud y el rendimiento.
- **Administrador:** El administrador posee permisos extras, como gestión de servicios, clases, citas, etc. Personificándolo, se trataría de los socios del centro junto con la persona encargada de la recepción del mismo, que gestionará el sistema junto a los dueños.
- **Superadministrador:** El superadministrador tendrá todos los permisos del administrador añadiendo algunos extras, como la posibilidad de ver las distintas organizaciones que usan el sistema, si en un futuro se añaden nuevas. En principio, sería el alumno desarrollador del proyecto.

4.3. Modelo de Comportamiento

A continuación se describirá el modelo de comportamiento del sistema. Para ello, partir de los casos de uso redactados anteriormente, se realizarán los diagramas de secuencias correspondientes, donde se reflejarán las operaciones o servicios del sistema, detallándose los contratos de las mismas.

Nota: Todos los diagramas de secuencias realizados por el usuario son extrapolables a administrador y superadministrador. Asimismo, los realizados por los administradores podrían también generalizarse para superadministradores.

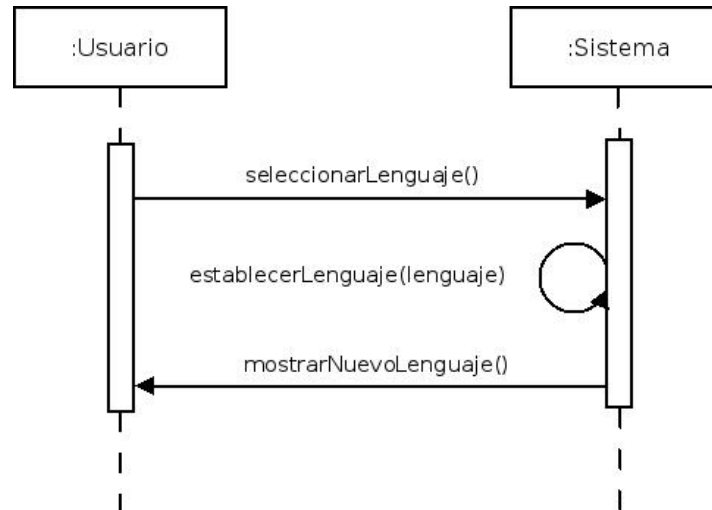


Figura 4.2: Diagrama de secuencia: Seleccionar lenguaje

Contrato de operación: seleccionarLenguaje()

- **Referencias cruzadas:** UC-01 (Cuadro 4.1).
- **Responsabilidades:** Seleccionar un nuevo lenguaje a establecer.
- **Precondiciones:** Ninguna

■ **Postcondiciones:**

- Se selecciona un nuevo lenguaje, que el sistema utilizará para la interfaz del programa.

Contrato de operación: establecerLenguaje(lenguaje)

■ **Referencias cruzadas:** UC-01 (Cuadro 4.1).

- **Responsabilidades:** Establecer un nuevo lenguaje para mostrar la interfaz. Si el usuario está registrado, se establecerá como su lenguaje por defecto.

■ **Precondiciones:**

- Se ha seleccionado un lenguaje.

■ **Postcondiciones:**

- Establecer el lenguaje recibido como lenguaje para la interfaz.
- Establecer el lenguaje por defecto para el usuario en caso de haberse identificado.

Contrato de operación: mostrarNuevoLenguaje()

■ **Referencias cruzadas:** UC-01 (Cuadro 4.1).

- **Responsabilidades:** Mostrar la interfaz con el nuevo lenguaje establecido.

■ **Precondiciones:**

- Se ha seleccionado y establecido un lenguaje.

■ **Postcondiciones:**

- La interfaz se muestra en el idioma seleccionado.

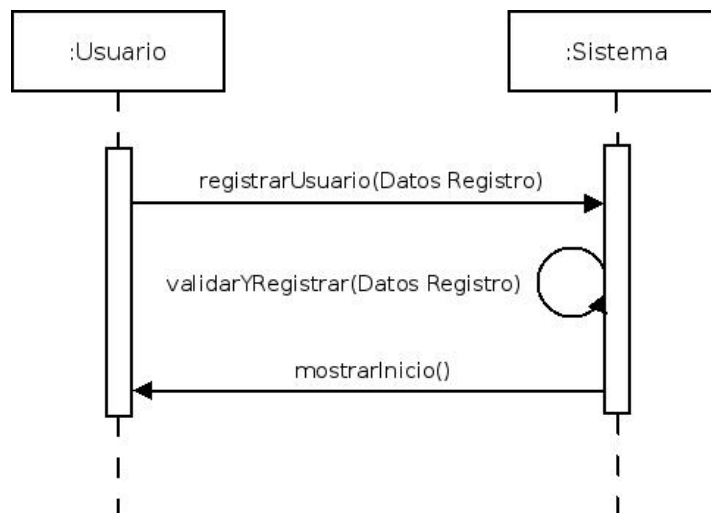


Figura 4.3: Diagrama de secuencia: Gestión de usuarios: Registro de nuevo usuario

Contrato de operación: registrarUsuario(Datos Registro)

- **Referencias cruzadas:** UC-02 (Cuadro 4.2).
- **Responsabilidades:** Se mandarán los datos de registro de un nuevo usuario al sistema.
- **Precondiciones:** Ninguna.
- **Postcondiciones:**
 - Se mandan los datos pedidos para registrar a un nuevo usuario al sistema.

Contrato de operación: validarYRegistrar(Datos Registro)

- **Referencias cruzadas:** UC-02 (Cuadro 4.2).
- **Responsabilidades:** Se validarán los datos recibidos y se realizará el registro del nuevo cliente.
- **Precondiciones:**
 - El usuario ha de haber enviado el formulario de registro con sus datos.
- **Postcondiciones:**
 - Se valida que los datos recibidos incluyen, al menos, todos los obligatorios.
 - Validación del email a registrar. No puede coincidir con el de algún usuario existente.
 - Registro de un nuevo usuario en el sistema.
 - Se realizará un inicio de sesión del nuevo usuario.

Contrato de operación: mostrarInicio()

- **Referencias cruzadas:** UC-02 (Cuadro 4.2).
- **Responsabilidades:** Se navegará hasta la página de inicio de la aplicación web.
- **Precondiciones:**
 - El usuario ha sido registrado correctamente.
 - El sistema ha iniciado sesión con los datos del usuario.
- **Postcondiciones:**
 - Se muestra la pantalla de inicio del sistema, con los datos del usuario.

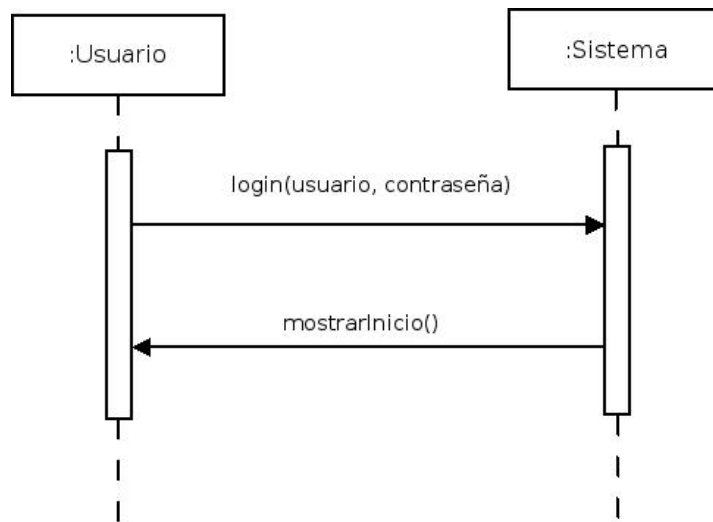


Figura 4.4: Diagrama de secuencia: Iniciar sesión

Contrato de operación: login(usuario, contraseña)

- **Referencias cruzadas:** UC-03 (Cuadro 4.3).
- **Responsabilidades:** Realizar el login del usuario en el sistema.
- **Precondiciones:**
 - El usuario no ha iniciado sesión previamente.
- **Postcondiciones:**
 - El sistema habrá comprobado que los datos de accesos son correctos.
 - Se realiza el inicio de la sesión del usuario, quedando este identificado.

Contrato de operación: mostrarInicio()

- **Referencias cruzadas:** UC-03 (Cuadro 4.3).
- **Responsabilidades:** Se navegará hasta la página de inicio de la aplicación web.
- **Precondiciones:**
 - El usuario ha sido registrado correctamente.
 - El sistema ha iniciado sesión con los datos del usuario.
- **Postcondiciones:**
 - Se muestra la pantalla de inicio del sistema, con los datos del usuario.

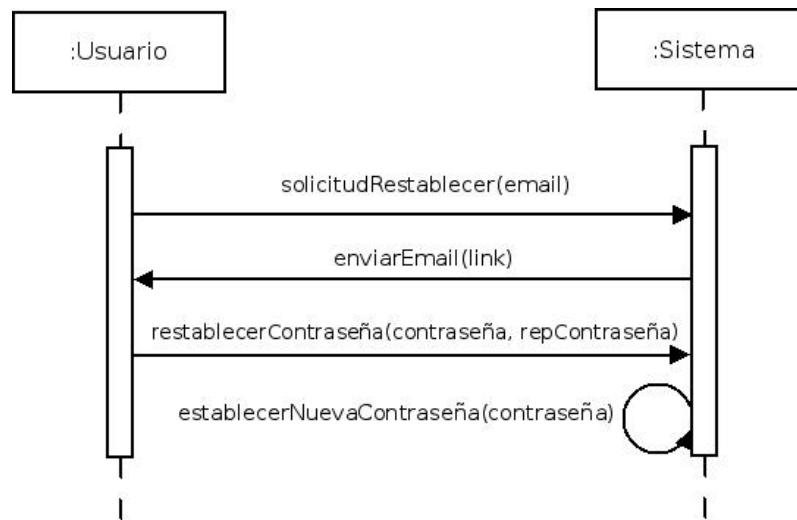


Figura 4.5: Diagrama de secuencia: Restablecer contraseña

Contrato de operación: solicitudRestablecer(email)

- **Referencias cruzadas:** UC-04 (Cuadro 4.4).
- **Responsabilidades:** Se solicitará al sistema restablecer la contraseña de un usuario que no ha iniciado sesión.
- **Precondiciones:**
 - El usuario no ha iniciado sesión previamente.
- **Postcondiciones:**
 - Se manda una solicitud de restablecer contraseña al sistema.

Contrato de operación: enviarEmail(link)

- **Referencias cruzadas:** UC-04 (Cuadro 4.4).
- **Responsabilidades:** Se enviará un email al usuario con el enlace para establecer una nueva contraseña.
- **Precondiciones:**
 - El usuario debe estar registrado en el sistema.
- **Postcondiciones:**
 - Se comprobará que el email pertenece a un usuario registrado.
 - Se envía un email a su correo con el enlace correspondiente para establecer la nueva contraseña.

Contrato de operación: restablecerContraseña(contraseña, repContraseña)

- **Referencias cruzadas:** UC-04 (Cuadro 4.4).

- **Responsabilidades:** Se mandará al sistema una nueva contraseña para el usuario.
- **Precondiciones:** Ninguna.
- **Postcondiciones:**
 - El sistema comprobará que las contraseñas introducidas coinciden.

Contrato de operación: establecerNuevaContraseña(contraseña)

- **Referencias cruzadas:** UC-04 (Cuadro 4.4).
- **Responsabilidades:** Se establecerá una nueva contraseña para el usuario.
- **Precondiciones:**
 - Se ha introducido la contraseña a establecer como nueva.
- **Postcondiciones:**
 - Se establece la contraseña introducida en la cuenta del usuario.
 - Se inicia sesión con los datos del usuario.

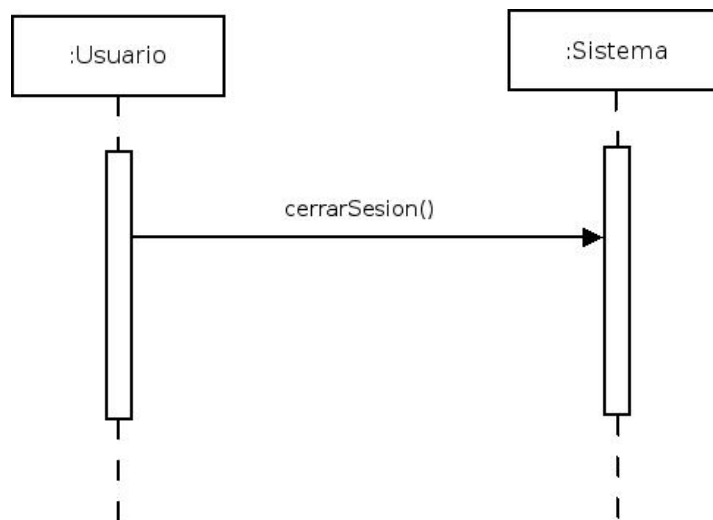


Figura 4.6: Diagrama de secuencia: Cerrar sesión

Contrato de operación: cerrarSesion()

- **Referencias cruzadas:** UC-05 (Cuadro 4.5).
- **Responsabilidades:** Se cerrará sesión del usuario actualmente identificado.
- **Precondiciones:**
 - El usuario se ha identificado en el sistema previamente.

■ **Postcondiciones:**

- El sistema terminará la sesión del usuario.
- La vista actual será redirigida a la página de login.

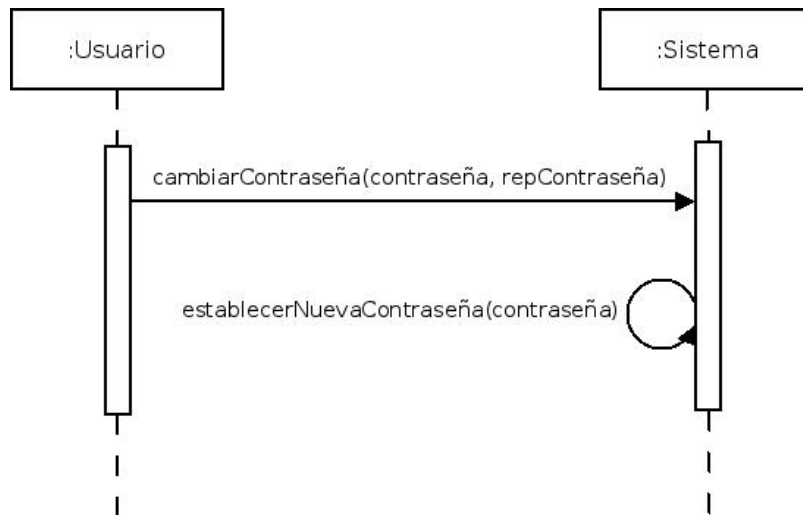


Figura 4.7: Diagrama de secuencia: Cambiar contraseña

Contrato de operación: `cambiarContraseña(contraseña, repContraseña)`

- **Referencias cruzadas:** UC-06 (Cuadro 4.6).
- **Responsabilidades:** Se mandará al sistema una nueva contraseña para el usuario.
- **Precondiciones:**
 - El usuario se ha identificado en el sistema previamente.
- **Postcondiciones:**
 - El sistema comprobará que las contraseñas introducidas coinciden.

Contrato de operación: `establecerNuevaContraseña(contraseña)`

- **Referencias cruzadas:** UC-06 (Cuadro 4.6).
- **Responsabilidades:** Se establecerá una nueva contraseña para el usuario.
- **Precondiciones:**
 - Se ha introducido la contraseña a establecer como nueva.
- **Postcondiciones:**
 - Se establece la contraseña introducida en la cuenta del usuario.

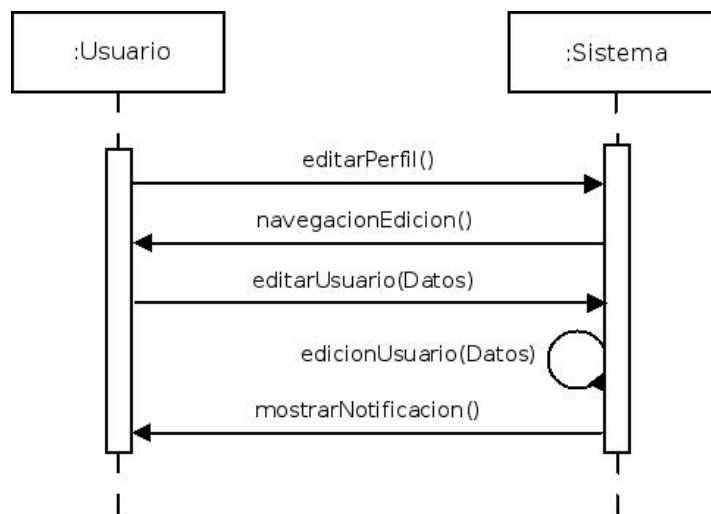


Figura 4.8: Diagrama de secuencia: Gestión de usuarios: Editar perfil

Contrato de operación: editarPerfil()

- **Referencias cruzadas:** UC-07 (Cuadro 4.7).
- **Responsabilidades:** El usuario enviará al sistema la opción de editar su perfil.
- **Precondiciones:**
 - El usuario se ha identificado en el sistema previamente.
 - El usuario ha navegado hasta la pantalla de su perfil.
- **Postcondiciones:**
 - Se manda la solicitud de edición del perfil.

Contrato de operación: navegacionEdicion()

- **Referencias cruzadas:** UC-07 (Cuadro 4.7).
- **Responsabilidades:** El sistema mostrará la página correspondiente a la edición de los datos del perfil del usuario.
- **Precondiciones:**
 - Se ha realizado la acción correspondiente para activar la navegación.
- **Postcondiciones:**
 - Se mostrará la página de edición del perfil del usuario.

Contrato de operación: editarUsuario(Datos)

- **Referencias cruzadas:** UC-07 (Cuadro 4.7).
- **Responsabilidades:** Se mandará al sistema los datos del usuario para que este sea editado.

- **Precondiciones:**

- El usuario se ha identificado en el sistema previamente.

- **Postcondiciones:**

- Se envía el formulario de datos del perfil al sistema.

Contrato de operación: `edicionUsuario(Datos)`

- **Referencias cruzadas:** UC-07 (Cuadro 4.7).

- **Responsabilidades:** Se realizará la edición de los datos del usuario, guardándolos en el sistema.

- **Precondiciones:**

- Se ha enviado el formulario con los datos a editar.

- **Postcondiciones:**

- El sistema guarda los datos recibidos del usuario en la base de datos.

Contrato de operación: `mostrarNotificacion()`

- **Referencias cruzadas:** UC-07 (Cuadro 4.7).

- **Responsabilidades:** Se mostrará un mensaje de acción por pantalla.

- **Precondiciones:**

- Se ha realizado la acción correspondiente para activar el mensaje.

- **Postcondiciones:**

- Se muestra el mensaje correspondiente a la acción en la pantalla, a modo de notificación para el usuario. Este puede ser confirmación de la acción o algún tipo de error en la ejecución de la misma.

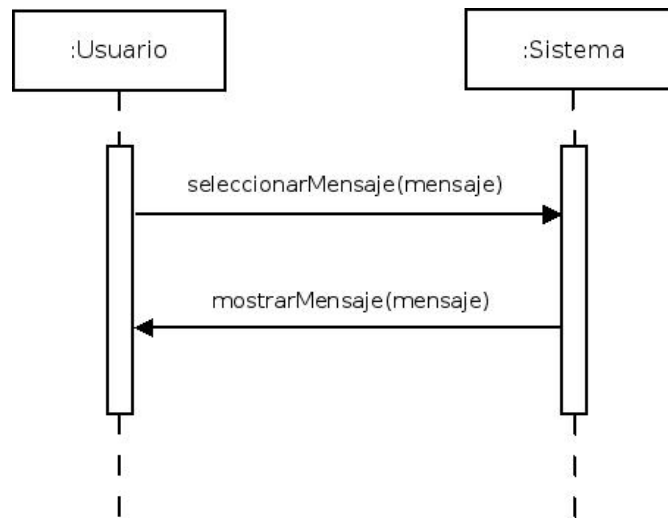


Figura 4.9: Diagrama de secuencia: Leer correo interno

Contrato de operación: seleccionarMensaje(mensaje)

- **Referencias cruzadas:** UC-08 (Cuadro 4.8).
- **Responsabilidades:** Se seleccionará el mensaje a mostrar.
- **Precondiciones:**
 - El usuario se ha identificado en el sistema previamente.
- **Postcondiciones:**
 - El sistema recibirá el mensaje concreto a mostrar.

Contrato de operación: mostrarMensaje(mensaje)

- **Referencias cruzadas:** UC-08 (Cuadro 4.8).
- **Responsabilidades:** Se mostrará el mensaje seleccionado al usuario.
- **Precondiciones:**
 - El mensaje habrá sido seleccionado por el usuario previamente.
- **Postcondiciones:**
 - El sistema muestra la página de visualización de mensajes con los datos detallados del mismo.

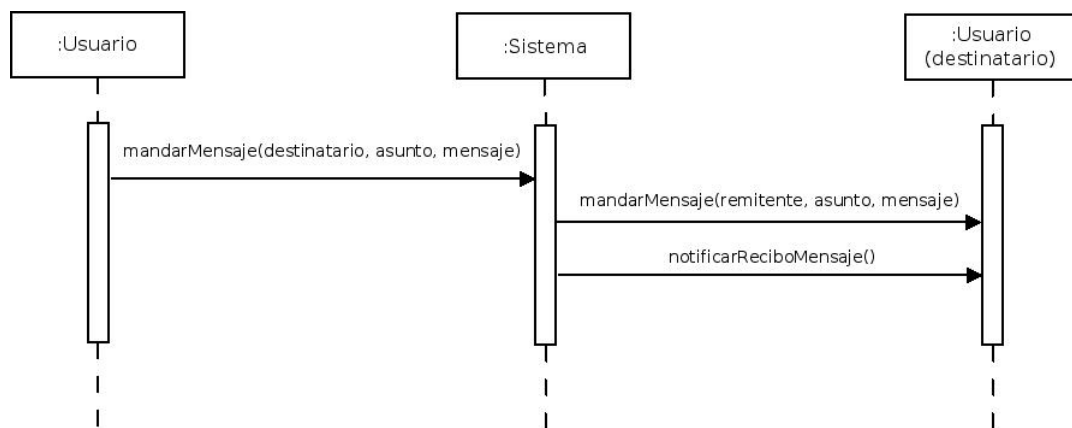


Figura 4.10: Diagrama de secuencia: Mandar correo interno

Contrato de operación: mandarMensaje(destinatario, asunto, mensaje)

- **Referencias cruzadas:** UC-09 (Cuadro 4.9).
- **Responsabilidades:** Se mandará al sistema el mensaje que se desea enviar, junto al asunto y el destinatario.
- **Precondiciones:**
 - El usuario se ha identificado en el sistema previamente.
- **Postcondiciones:**
 - El sistema recibirá los datos del mensaje: El cuerpo (mensaje en sí), asunto y destinatario).

Contrato de operación: mandarMensaje(remitente, asunto, mensaje)

- **Referencias cruzadas:** UC-09 (Cuadro 4.9).
- **Responsabilidades:** El sistema mandará el mensaje recibido a su destinatario.
- **Precondiciones:**
 - Se ha recibido el mensaje por parte del remitente.
- **Postcondiciones:**
 - El sistema manda al destinatario el mensaje recibido.

Contrato de operación: notificarReciboMensaje()

- **Referencias cruzadas:** UC-09 (Cuadro 4.9).
- **Responsabilidades:** Notificar al destinatario del mensaje que ha recibido un correo interno.
- **Precondiciones:**

- El usuario ha mandado un correo al usuario.
- **Postcondiciones:**
- Se creará una nueva notificación de recibo de mensaje para el destinatario.

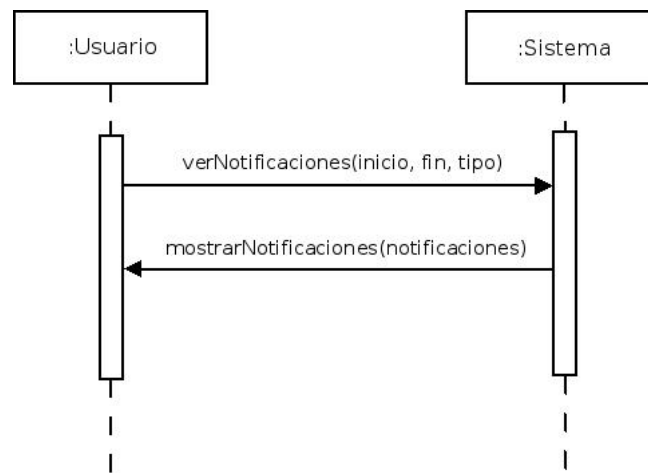


Figura 4.11: Diagrama de secuencia: Notificaciones

Contrato de operación: verNotificaciones(inicio, fin, tipo)

- **Referencias cruzadas:** UC-10 (Cuadro 4.19).
- **Responsabilidades:** Solicitar las notificaciones destinadas al usuario en unas determinadas fechas.
- **Precondiciones:**
 - El usuario se ha identificado en el sistema previamente.
- **Postcondiciones:**
 - Se envía al sistema el formulario con las fechas de inicio y fin elegidas para ver las notificaciones del usuario en el sistema. Se podrá mandar el tipo de notificación específica a mostrar.

Contrato de operación: mostrarNotificaciones(notificaciones)

- **Referencias cruzadas:** UC-10 (Cuadro 4.19).
- **Responsabilidades:** Mostrar las notificaciones destinadas al usuario.
- **Precondiciones:**
 - El usuario ha solicitado ver sus notificaciones.
- **Postcondiciones:**

- Se muestra el listado de notificaciones para el usuario entre las fechas seleccionadas.

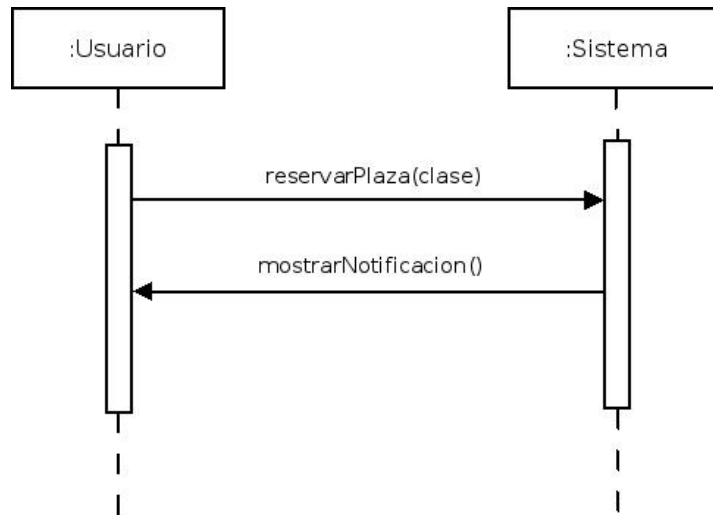


Figura 4.12: Diagrama de secuencia: Gestión de servicios: Reservar plaza en una clase

Contrato de operación: reservarLayout(clase)

- **Referencias cruzadas:** UC-11 (Cuadro 4.11).
- **Responsabilidades:** Se reservarLayout una plaza de la clase seleccionada al usuario.
- **Precondiciones:**
 - El usuario se ha identificado en el sistema previamente.
 - Ha de haber al menos una plaza libre en la clase.
 - El usuario no posee plaza en la clase seleccionada.
- **Postcondiciones:**
 - Se reservarLayout una plaza de la clase seleccionada al usuario.
 - Se reduce la plaza reservada de las totales disponibles de la clase.

Contrato de operación: mostrarNotificacion()

- **Referencias cruzadas:** UC-11 (Cuadro 4.11).
- **Responsabilidades:** Se mostrará un mensaje de acción por pantalla.
- **Precondiciones:**
 - Se ha realizado la acción correspondiente para activar el mensaje.
- **Postcondiciones:**

- Se muestra el mensaje correspondiente a la acción en la pantalla, a modo de notificación para el usuario. Este puede ser confirmación de la acción o algún tipo de error en la ejecución de la misma.

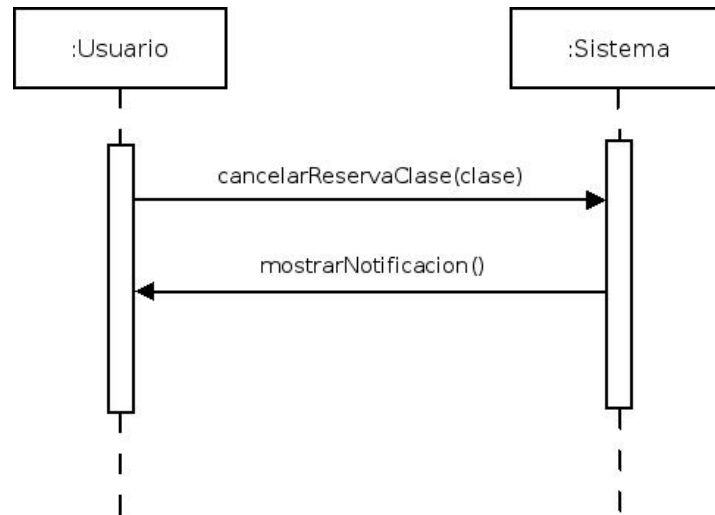


Figura 4.13: Diagrama de secuencia: Gestión de servicios: Cancelación de reserva de una clase

Contrato de operación: `cancelarReservaClase(clase)`

- **Referencias cruzadas:** UC-12 (Cuadro 4.12).
- **Responsabilidades:** Se cancelará la reserva del usuario previamente realizada de una clase específica.
- **Precondiciones:**
 - El usuario se ha identificado en el sistema previamente.
 - La clase ha sido reservada previamente.
- **Postcondiciones:**
 - Se cancela la reserva de la plaza del usuario en la clase específica.
 - Las plazas disponibles de la clase en concreto se incrementan en uno.

Contrato de operación: `mostrarNotificacion()`

- **Referencias cruzadas:** UC-12 (Cuadro 4.12).
- **Responsabilidades:** Se mostrará un mensaje de acción por pantalla.
- **Precondiciones:**
 - Se ha realizado la acción correspondiente para activar el mensaje.

■ **Postcondiciones:**

- Se muestra el mensaje correspondiente a la acción en la pantalla, a modo de notificación para el usuario. Este puede ser confirmación de la acción o algún tipo de error en la ejecución de la misma.

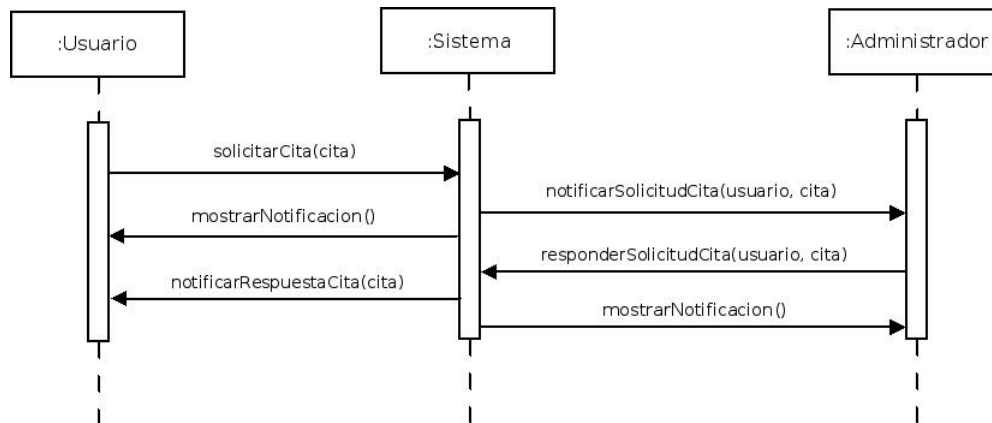


Figura 4.14: Diagrama de secuencia:: Gestión de servicios: Solicitud de cita y respuesta

Contrato de operación: solicitarCita(cita)

- **Referencias cruzadas:** UC-13 (Cuadro 4.13), UC-14 (Cuadro 4.14).
- **Responsabilidades:** Se enviará una solicitud de la cita seleccionada a los administradores.
- **Precondiciones:**
 - El usuario se ha identificado en el sistema previamente.
 - La cita debe estar disponible para su solicitud.
- **Postcondiciones:**
 - Se enviará una solicitud de la cita seleccionada a los administradores.
 - Se establece la cita como no disponible hasta que se resuelva la solicitud.

Contrato de operación: notificarSolicitudCita(usuario, cita)

- **Referencias cruzadas:** UC-13 (Cuadro 4.13), UC-14 (Cuadro 4.14).
- **Responsabilidades:** Mandar una notificación de solicitud de cita de un determinado usuario a los administradores.
- **Precondiciones:**
 - Se ha recibido una solicitud de cita en el sistema.
- **Postcondiciones:**

- Se creará una nueva notificación de solicitud de cita para los administradores.

Contrato de operación: `mostrarNotificacion()`

- **Referencias cruzadas:** UC-13 (Cuadro 4.13), UC-14 (Cuadro 4.14).
- **Responsabilidades:** Se mostrará un mensaje de acción por pantalla.
- **Precondiciones:**
 - Se ha realizado la acción correspondiente para activar el mensaje.
- **Postcondiciones:**
 - Se muestra el mensaje correspondiente a la acción en la pantalla, a modo de notificación para el usuario. Este puede ser confirmación de la acción o algún tipo de error en la ejecución de la misma.

Contrato de operación: `responderSolicitudCita(usuario, cita)`

- **Referencias cruzadas:** UC-13 (Cuadro 4.13), UC-14 (Cuadro 4.14).
- **Responsabilidades:** Responder a la solicitud de cita recibida por parte de un usuario específico.
- **Precondiciones:**
 - El administrador se ha identificado en el sistema previamente.
 - Se ha recibido una solicitud de cita por parte de un determinado usuario.
- **Postcondiciones:**
 - Se envía la respuesta de la solicitud al sistema.
 - Dependiendo de la respuesta (declinada o aceptada), se establece la disponibilidad de la cita (disponible o no disponible).

Contrato de operación: `notificarRespuestaCita(cita)`

- **Referencias cruzadas:** UC-13 (Cuadro 4.13), UC-14 (Cuadro 4.14).
- **Responsabilidades:** Mandar una notificación de respuesta de la solicitud de cita al usuario.
- **Precondiciones:**
 - Se ha recibido una respuesta de la solicitud de cita por parte de los administradores.
- **Postcondiciones:**
 - Se creará una nueva notificación de repuesta de solicitud de cita para el usuario.

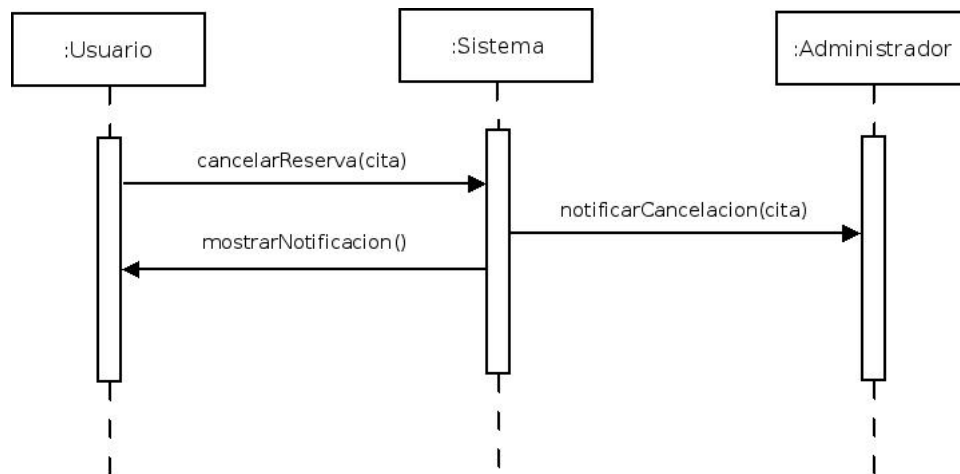


Figura 4.15: Diagrama de secuencia: Gestión de servicios: Cancelar Cita

Contrato de operación: cancelarReserva(cita)

- **Referencias cruzadas:** UC-15 (Cuadro 4.15).
- **Responsabilidades:** Se cancelará la cita o solicitud de cita del usuario, notificando a los administradores.
- **Precondiciones:**
 - El usuario se ha identificado en el sistema previamente.
 - El usuario ha solicitado la cita.
- **Postcondiciones:**
 - Se cancela la reserva o solicitud de la cita seleccionada.

Contrato de operación: notificarCancelacion(cita)

- **Referencias cruzadas:** UC-15 (Cuadro 4.15).
- **Responsabilidades:** Mandar una notificación de cancelación de la reserva o solicitud de cita a los administradores.
- **Precondiciones:**
 - Se ha cancelado una solicitud o reserva de cita.
- **Postcondiciones:**
 - Se creará una nueva notificación de cancelación de reserva o solicitud de cita para los administradores.

Contrato de operación: mostrarNotificacion()

- **Referencias cruzadas:** UC-15 (Cuadro 4.15).
- **Responsabilidades:** Se mostrará un mensaje de acción por pantalla.

■ **Precondiciones:**

- Se ha realizado la acción correspondiente para activar el mensaje.

■ **Postcondiciones:**

- Se muestra el mensaje correspondiente a la acción en la pantalla, a modo de notificación para el usuario. Este puede ser confirmación de la acción o algún tipo de error en la ejecución de la misma.

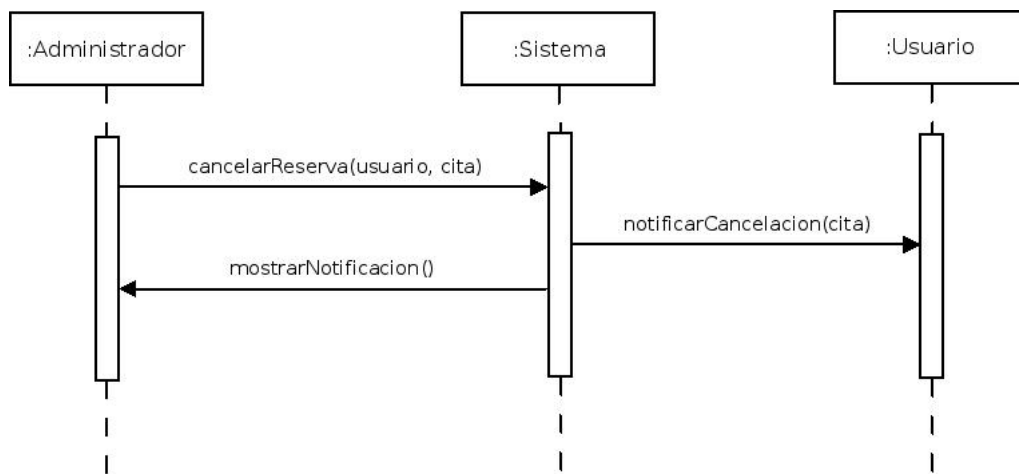


Figura 4.16: Diagrama de secuencia: Gestión de servicios: Cancelar cita (Administrador)

Contrato de operación: cancelarReserva(cita)

■ **Referencias cruzadas:** UC-16 (Cuadro 4.16).

- **Responsabilidades:** Un administrador cancelará la cita o solicitud de cita del usuario, notificando al mismo.

■ **Precondiciones:**

- El administrador se ha identificado en el sistema previamente.
- El usuario ha solicitado la cita.

■ **Postcondiciones:**

- Se cancela la reserva o solicitud de la cita seleccionada.

Contrato de operación: notificarCancelacion(cita)

■ **Referencias cruzadas:** UC-16 (Cuadro 4.16).

- **Responsabilidades:** Mandar una notificación de cancelación de la reserva o solicitud de cita al usuario.

■ **Precondiciones:**

- Se ha cancelado una solicitud o reserva de cita.

■ **Postcondiciones:**

- Se creará una nueva notificación de cancelación de reserva o solicitud de cita para el usuario.

Contrato de operación: mostrarNotificacion()

■ **Referencias cruzadas:** UC-16 (Cuadro 4.16).

■ **Responsabilidades:** Se mostrará un mensaje de acción por pantalla.

■ **Precondiciones:**

- Se ha realizado la acción correspondiente para activar el mensaje.

■ **Postcondiciones:**

- Se muestra el mensaje correspondiente a la acción en la pantalla, a modo de notificación para el administrador. Este puede ser confirmación de la acción o algún tipo de error en la ejecución de la misma.

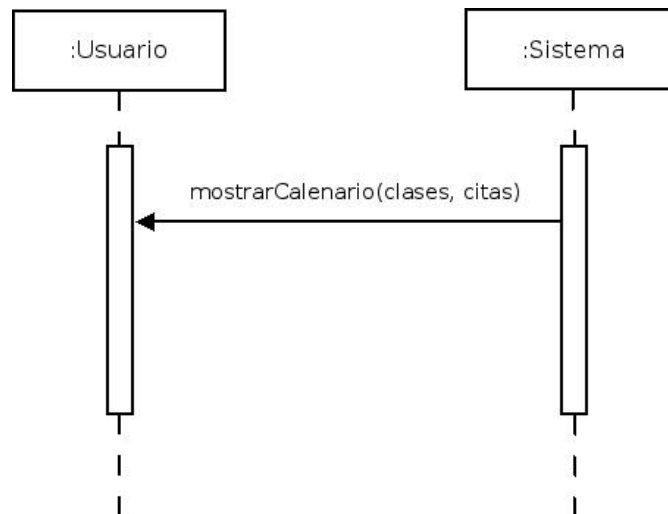


Figura 4.17: Diagrama de secuencia: Gestión de servicios: Calendario de actividades

Contrato de operación: mostrarCalendario(clases, citas)

■ **Referencias cruzadas:** UC-17 (Cuadro 4.17).

■ **Responsabilidades:** Mostrar el calendario mensual con las clases y citas disponibles en el sistema. Se distinguirá el estado de las mismas por su color de fondo o borde, para diferenciar entre clases o citas pasadas, disponibles, reservadas por el usuario...

■ **Precondiciones:**

- El usuario se ha identificado en el sistema previamente.
- El usuario ha accedido a la página de visualización del calendario.

■ **Postcondiciones:**

- El sistema muestra el calendario con todas las citas y clases disponibles, distinguiendo su estado por colores.

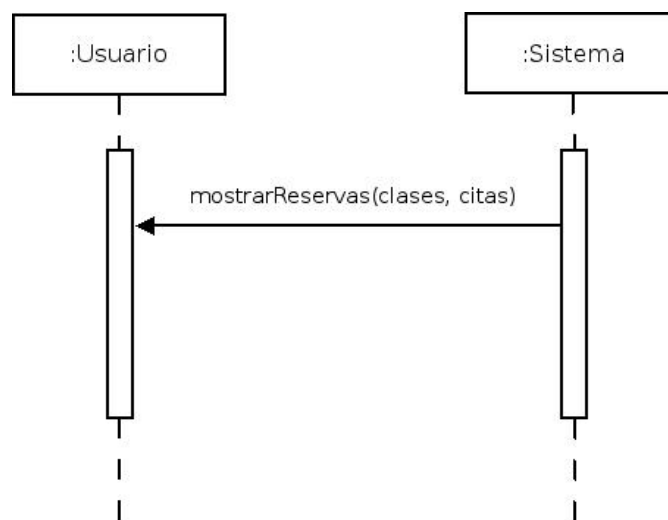


Figura 4.18: Diagrama de secuencia: Gestión de servicios: Consultar reservas

Contrato de operación: mostrarReservas(citas, clases)

■ **Referencias cruzadas:** UC-18 (Cuadro 4.18).

- **Responsabilidades:** Se mostrará una lista de las citas y reservas del usuario, tanto pasadas como actuales.

■ **Precondiciones:**

- El usuario se ha identificado en el sistema previamente.
- El usuario ha navegado hasta la página de vista de reservas.

■ **Postcondiciones:**

- Se listarán todas las citas y clases que el usuario haya reservado, distinguiendo entre pasadas y actuales.

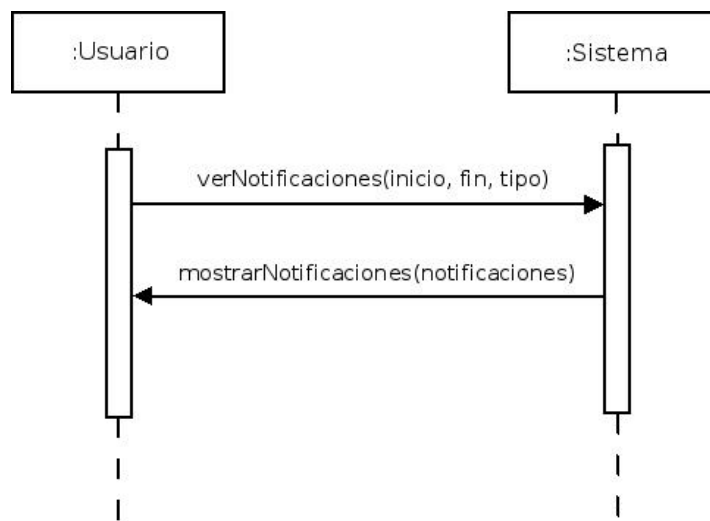


Figura 4.19: Diagrama de secuencia: Notificaciones

Contrato de operación: `verAuditorias(inicio, fin, tipo)`

- **Referencias cruzadas:** UC-19 (Cuadro 4.19).
- **Responsabilidades:** Solicitar las notificaciones que el sistema ha creado destinadas al usuario en unas determinadas fechas.
- **Precondiciones:**
 - El usuario se ha identificado en el sistema previamente.
- **Postcondiciones:**
 - Se envía al sistema el formulario con las fechas de inicio y fin elegidas para ver las notificaciones del usuario. Se podrá mandar el tipo de notificación específica a mostrar.

Contrato de operación: `mostrarNotificaciones(notificaciones)`

- **Referencias cruzadas:** UC-19 (Cuadro 4.19).
- **Responsabilidades:** Se mostrará la lista de notificaciones destinadas al usuario.
- **Precondiciones:**
 - El usuario se ha identificado en el sistema previamente.
 - El usuario ha navegado hasta la vista de notificaciones.
- **Postcondiciones:**
 - Se listarán todas las notificaciones que el sistema ha creado para el usuario.

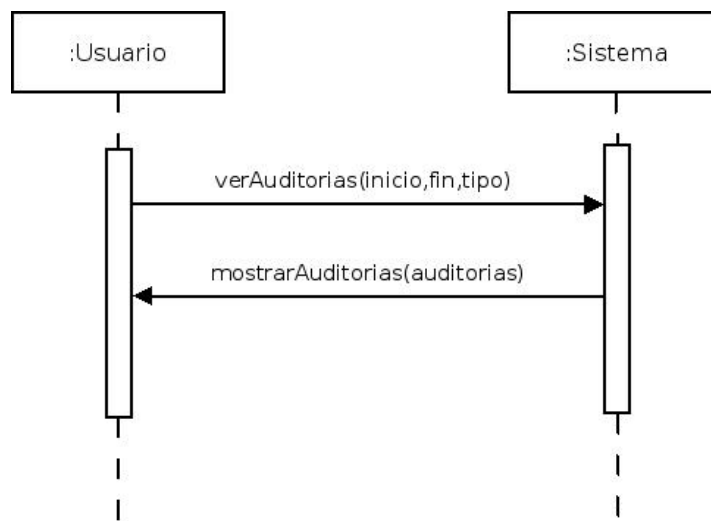


Figura 4.20: Diagrama de secuencia: Auditorías

Contrato de operación: `verAuditorias(inicio, fin, tipo)`

- **Referencias cruzadas:** UC-20 (Cuadro 4.20).
- **Responsabilidades:** Solicitar las acciones llevadas a cabo por el usuario en unas determinadas fechas.
- **Precondiciones:**
 - El usuario se ha identificado en el sistema previamente.
- **Postcondiciones:**
 - Se envía al sistema el formulario con las fechas de inicio y fin elegidas para ver las acciones del usuario en el sistema. Se podrá mandar el tipo de acción específica a mostrar.

Contrato de operación: `mostrarAuditorias(auditorias)`

- **Referencias cruzadas:** UC-20 (Cuadro 4.20).
- **Responsabilidades:** Mostrar las acciones llevadas a cabo por el usuario en el sistema.
- **Precondiciones:**
 - El usuario ha solicitado ver sus auditorías.
- **Postcondiciones:**
 - Se muestra el listado de acciones realizadas entre las fechas seleccionadas.

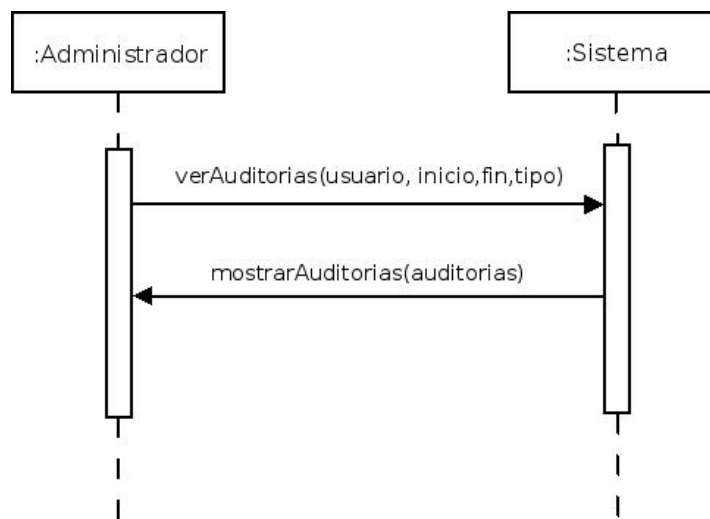


Figura 4.21: Diagrama de secuencia: Auditorías de Administradores

Contrato de operación: **verAuditorias(usuario, inicio, fin, tipo)**

- **Referencias cruzadas:** UC-21 (Cuadro 4.21).
- **Responsabilidades:** Solicitar las acciones llevadas a cabo por el usuario seleccionado en unas determinadas fechas.
- **Precondiciones:**
 - El administrador se ha identificado en el sistema previamente.
- **Postcondiciones:**
 - Se envía al sistema el formulario con el usuario específico y las fechas de inicio y fin elegidas para ver las acciones del usuario en el sistema. Se podrá mandar el tipo de acción específica a mostrar.

Contrato de operación: **mostrarAuditorias(auditorias)**

- **Referencias cruzadas:** UC-21 (Cuadro 4.21).
- **Responsabilidades:** Mostrar las acciones llevadas a cabo por el usuario seleccionado en el sistema.
- **Precondiciones:**
 - El administrador ha solicitado ver las auditorías seleccionadas.
- **Postcondiciones:**
 - Se muestra el listado de acciones realizadas entre las fechas seleccionadas para el usuario especificado.

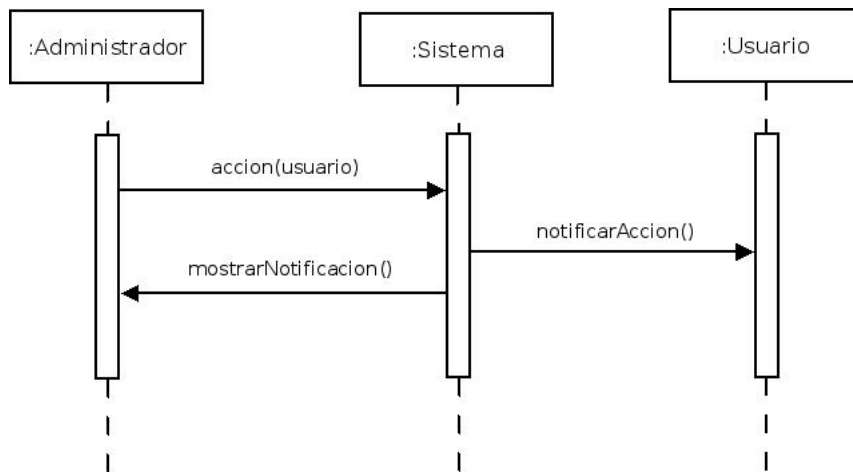


Figura 4.22: Diagrama de secuencia: Gestión de usuarios: Activar/Suspender

Contrato de operación: `accion(usuario)`

- **Referencias cruzadas:** UC-22 (Cuadro 4.22).
- **Responsabilidades:** El administrador podrá activar o suspender al usuario seleccionado.
- **Precondiciones:**
 - El administrador se ha identificado en el sistema previamente.
 - El usuario seleccionado debe estar activo para ser suspendido o viceversa.
- **Postcondiciones:**
 - Se realizará la acción seleccionada por el administrador (*activar, suspender*) referente a un usuario específico.

Contrato de operación: `notificarAccion()`

- **Referencias cruzadas:** UC-22 (Cuadro 4.22).
- **Responsabilidades:** Mandar una notificación de suspensión/activación de la cuenta al usuario.
- **Precondiciones:**
 - Se ha suspendido/activado la cuenta de un usuario.
- **Postcondiciones:**
 - Se creará una nueva notificación de activación o suspensión de cuenta para el usuario seleccionado.

Contrato de operación: `mostrarNotificacion()`

- **Referencias cruzadas:** UC-22 (Cuadro 4.22).
- **Responsabilidades:** Se mostrará un mensaje de acción por pantalla.

■ **Precondiciones:**

- Se ha realizado la acción correspondiente para activar el mensaje.

■ **Postcondiciones:**

- Se muestra el mensaje correspondiente a la acción en la pantalla, a modo de notificación para el administrador. Este puede ser confirmación de la acción o algún tipo de error en la ejecución de la misma.

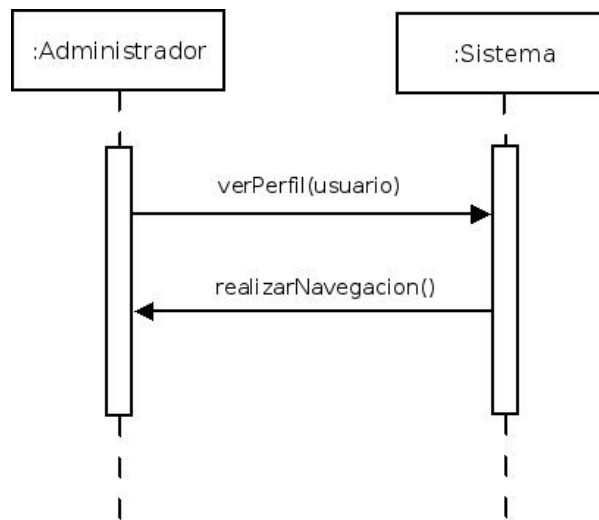


Figura 4.23: Diagrama de secuencia: Gestión de usuarios: Ver perfil (Administrador)

Contrato de operación: verPerfil(usuario)

- **Referencias cruzadas:** UC-23 (Cuadro 4.23).

- **Responsabilidades:** El administrador podrá ver el perfil del usuario seleccionado.

■ **Precondiciones:**

- El administrador se ha identificado en el sistema previamente.

■ **Postcondiciones:**

- Se navegará a la página correspondiente para ver los detalles del perfil de un usuario específico.

Contrato de operación: realizarNavegacion()

- **Referencias cruzadas:** UC-23 (Cuadro 4.23).

- **Responsabilidades:** El sistema mostrará la página correspondiente al perfil del usuario.

■ **Precondiciones:**

- Se ha realizado la acción correspondiente para activar la navegación.
- **Postcondiciones:**
- Se mostrará la página del perfil de usuario al administrador.

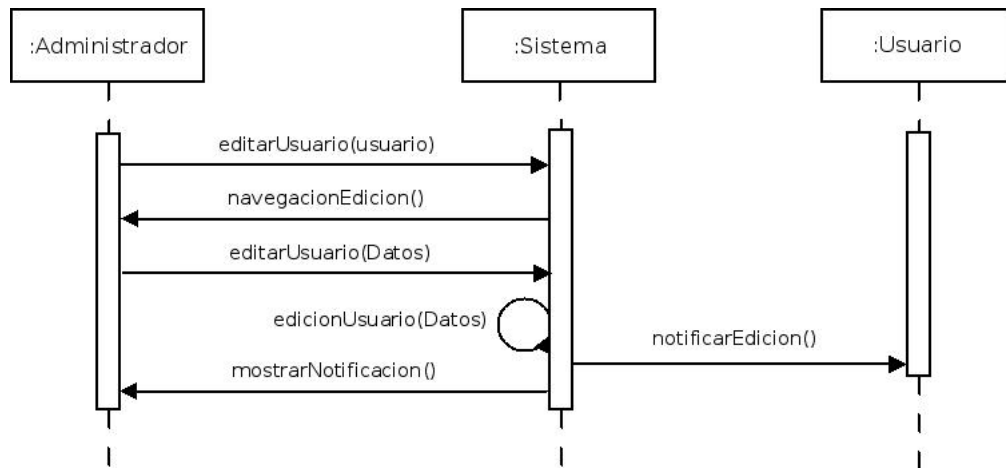


Figura 4.24: Diagrama de secuencia: Gestión de usuarios: Editar usuario (Administrador)

Contrato de operación: `editarUsuario(usuario)`

- **Referencias cruzadas:** UC-24 (Cuadro 4.24).
- **Responsabilidades:** Se informará al sistema de la acción de edición de un usuario específico.
- **Precondiciones:**
 - El administrador se ha identificado en el sistema previamente.
- **Postcondiciones:**
 - Se manda la solicitud de edición del usuario seleccionado.

Contrato de operación: `navigacionEdicion()`

- **Referencias cruzadas:** UC-24 (Cuadro 4.24).
- **Responsabilidades:** El sistema mostrará la página correspondiente a la edición de los datos del usuario.
- **Precondiciones:**
 - Se ha realizado la acción correspondiente para activar la navegación.
- **Postcondiciones:**

- Se mostrará la página de edición del perfil del usuario seleccionado.

Contrato de operación: editarUsuario(Datos)

- **Referencias cruzadas:** UC-24 (Cuadro 4.24).
- **Responsabilidades:** Se mandará al sistema los datos del usuario para que este sea editado.
- **Precondiciones:**
 - El administrador se ha identificado en el sistema previamente.
- **Postcondiciones:**
 - Se envía el formulario de datos del cliente a editar al sistema.

Contrato de operación: edicionUsuario(Datos)

- **Referencias cruzadas:** UC-24 (Cuadro 4.24).
- **Responsabilidades:** Se realizará la edición de los datos del usuario, guardándolos en el sistema.
- **Precondiciones:**
 - Se ha enviado el formulario con los datos a editar.
- **Postcondiciones:**
 - El sistema guarda los datos recibidos para el usuario específico en la base de datos.

Contrato de operación: notificarEdicion()

- **Referencias cruzadas:** UC-24 (Cuadro 4.24).
- **Responsabilidades:** Mandar una notificación de edición del perfil por parte del administrador al usuario.
- **Precondiciones:**
 - Se ha editado los datos del usuario por parte del administrador.
- **Postcondiciones:**
 - Se creará una nueva notificación de edición del perfil para el usuario seleccionado, indicando quién lo ha modificado.

Contrato de operación: mostrarNotificacion()

- **Referencias cruzadas:** UC-24 (Cuadro 4.24).
- **Responsabilidades:** Se mostrará un mensaje de acción por pantalla.
- **Precondiciones:**
 - Se ha realizado la acción correspondiente para activar el mensaje.

■ **Postcondiciones:**

- Se muestra el mensaje correspondiente a la acción en la pantalla, a modo de notificación para el administrador. Este puede ser confirmación de la acción o algún tipo de error en la ejecución de la misma.

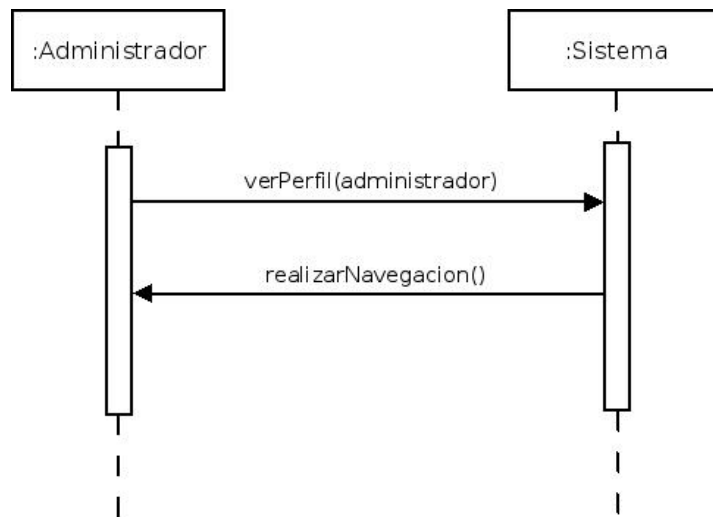


Figura 4.25: Diagrama de secuencia: Gestión de administradores: Ver perfil de administrador

Contrato de operación: verPerfil(administrador)

- **Referencias cruzadas:** UC-25 (Cuadro 4.25).
- **Responsabilidades:** El administrador podrá ver el perfil de otro administrador seleccionado.
- **Precondiciones:**
 - El administrador se ha identificado en el sistema previamente.
- **Postcondiciones:**
 - Se navegará a la página del perfil del administrador seleccionado para ver sus detalles en el sistema.

Contrato de operación: realizarNavegacion()

- **Referencias cruzadas:** UC-25 (Cuadro 4.25).
- **Responsabilidades:** El sistema mostrará la página correspondiente a la opción seleccionada para realizar la acción (ver administrador o mandar un mensaje al administrador).
- **Precondiciones:**
 - Se ha realizado la acción correspondiente para activar la navegación.

- **Postcondiciones:**

- Se mostrará la página correspondiente a la opción seleccionada por el administrador para realizar la acción (*ver perfil de administrador o mandar un correo al administrador*).

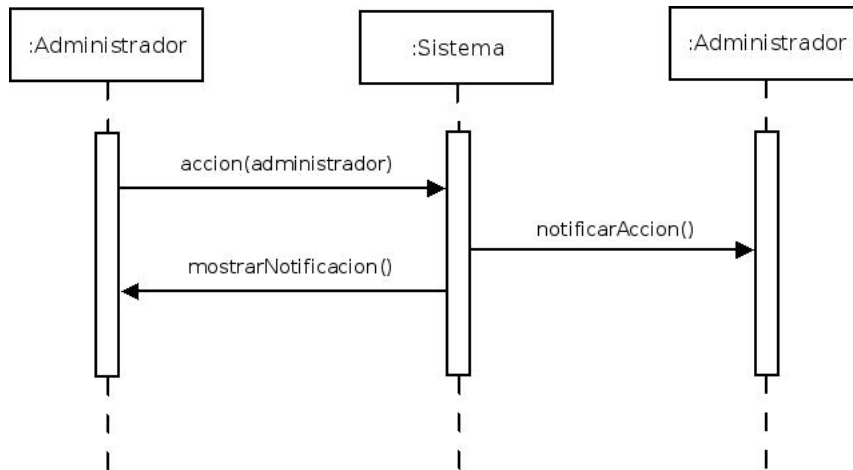


Figura 4.26: Diagrama de secuencia: Gestión de administradores: Activar/Suspender administrador

Contrato de operación: accion(administrador)

- **Referencias cruzadas:** UC-26 (Cuadro 4.26).
- **Responsabilidades:** El administrador podrá activar o suspender a otro administrador seleccionado.
- **Precondiciones:**
 - El administrador se ha identificado en el sistema previamente.
 - El administrador seleccionado debe estar activo para ser suspendido o viceversa.
- **Postcondiciones:**
 - Se realizará la acción seleccionada por el administrador (*activar, suspender*) referente a otro administrador específico.

Contrato de operación: notificarAccion()

- **Referencias cruzadas:** UC-26 (Cuadro 4.26).
- **Responsabilidades:** Mandar una notificación de suspensión/activación de la cuenta al administrador.
- **Precondiciones:**

- Se ha suspendido/activado la cuenta de un administrador.

■ **Postcondiciones:**

- Se creará una nueva notificación de activación o suspensión de cuenta para el administrador seleccionado.

Contrato de operación: mostrarNotificacion()

■ **Referencias cruzadas:** UC-26 (Cuadro 4.26).

■ **Responsabilidades:** Se mostrará un mensaje de acción por pantalla.

■ **Precondiciones:**

- Se ha realizado la acción correspondiente para activar el mensaje.

■ **Postcondiciones:**

- Se muestra el mensaje correspondiente a la acción en la pantalla, a modo de notificación para el administrador. Este puede ser confirmación de la acción o algún tipo de error en la ejecución de la misma.

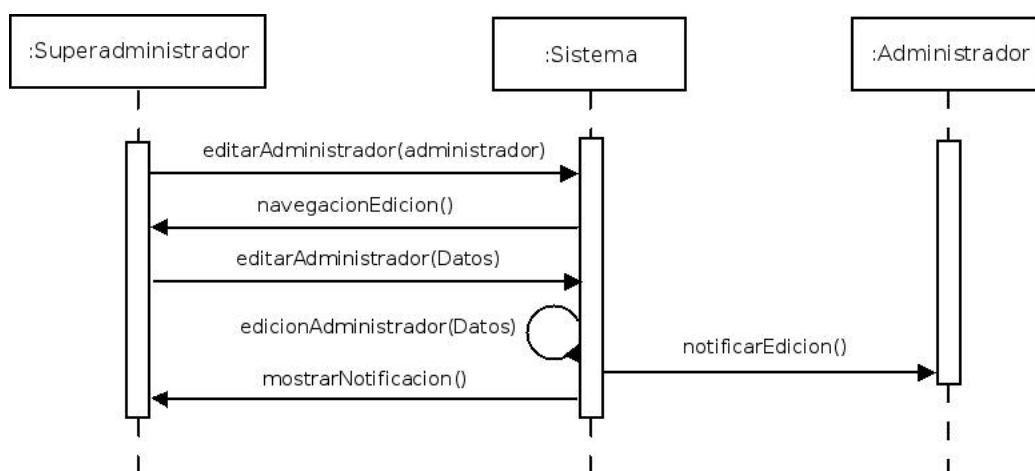


Figura 4.27: Diagrama de secuencia: Gestión de administradores: Editar administrador

Contrato de operación: editarAdministrador(administrador)

■ **Referencias cruzadas:** UC-27 (Cuadro 4.27).

■ **Responsabilidades:** Se informará al sistema de la acción de edición de un administrador específico por parte del superadministrador.

■ **Precondiciones:**

- El superadministrador se ha identificado en el sistema previamente.

- **Postcondiciones:**

- Se manda la solicitud de edición del administrador seleccionado.

Contrato de operación: navegacionEdicion()

- **Referencias cruzadas:** UC-27 (Cuadro 4.27).

- **Responsabilidades:** El sistema mostrará la página correspondiente a la edición de los datos del administrador.

- **Precondiciones:**

- Se ha realizado la acción correspondiente para activar la navegación.

- **Postcondiciones:**

- Se mostrará la página de edición del perfil del administrador seleccionado.

Contrato de operación: editarAdministrador(Datos)

- **Referencias cruzadas:** UC-27 (Cuadro 4.27).

- **Responsabilidades:** Se mandará al sistema los datos del administrador para que este sea editado.

- **Precondiciones:**

- El superadministrador se ha identificado en el sistema previamente.

- **Postcondiciones:**

- Se envía el formulario de datos del administrador a editar al sistema.

Contrato de operación: edicionAdministrador(Datos)

- **Referencias cruzadas:** UC-27 (Cuadro 4.27).

- **Responsabilidades:** Se realizará la edición de los datos del administrador, guardándolos en el sistema.

- **Precondiciones:**

- Se ha enviado el formulario con los datos a editar.

- **Postcondiciones:**

- El sistema guarda los datos recibidos para el administrador específico en la base de datos.

Contrato de operación: notificarEdicion()

- **Referencias cruzadas:** UC-27 (Cuadro 4.27).

- **Responsabilidades:** Mandar una notificación de edición del perfil por parte del superadministrador al administrador.

■ **Precondiciones:**

- Se ha editado los datos del administrador por parte del superadministrador.

■ **Postcondiciones:**

- Se creará una nueva notificación de edición del perfil para el administrador seleccionado, indicando quién lo ha modificado.

Contrato de operación: mostrarNotificacion()

■ **Referencias cruzadas:** UC-27 (Cuadro 4.27).

■ **Responsabilidades:** Se mostrará un mensaje de acción por pantalla.

■ **Precondiciones:**

- Se ha realizado la acción correspondiente para activar el mensaje.

■ **Postcondiciones:**

- Se muestra el mensaje correspondiente a la acción en la pantalla, a modo de notificación para el superadministrador. Este puede ser confirmación de la acción o algún tipo de error en la ejecución de la misma.

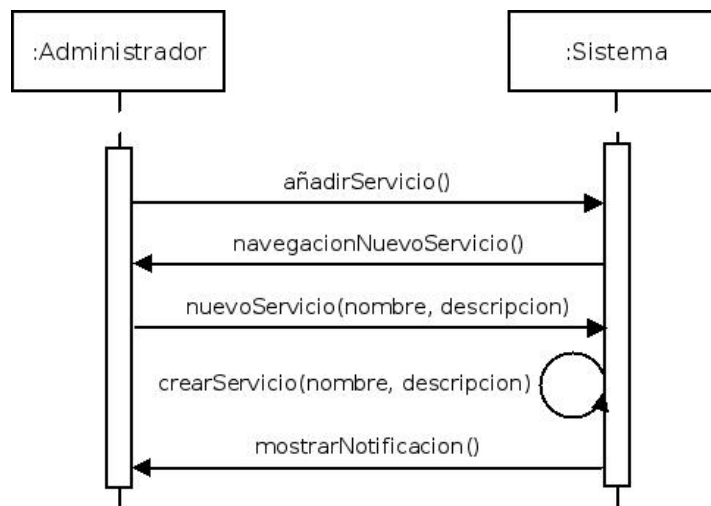


Figura 4.28: Diagrama de secuencia: Gestión de servicios: Alta servicio

Contrato de operación: añadirServicio()

■ **Referencias cruzadas:** UC-28 (Cuadro 4.28).

■ **Responsabilidades:** Solicitar al sistema crear un nuevo servicio.

■ **Precondiciones:**

- El administrador se ha identificado en el sistema previamente.

■ **Postcondiciones:**

- Se enviará al sistema la solicitud de crear un nuevo servicio por parte del administrador.

Contrato de operación: navegacionNuevoServicio()

■ **Referencias cruzadas:** UC-28 (Cuadro 4.28).

■ **Responsabilidades:** El sistema mostrará la ventana correspondiente a la creación de un nuevo servicio.

■ **Precondiciones:**

- Se ha realizado la acción correspondiente para activar la navegación.

■ **Postcondiciones:**

- Se mostrará la ventana de creación de un servicio nuevo en la interfaz del administrador.

Contrato de operación: nuevoServicio(nombre, descripcion)

■ **Referencias cruzadas:** UC-28 (Cuadro 4.28).

■ **Responsabilidades:** Se mandará al sistema los datos necesarios para la creación de un nuevo servicio.

■ **Precondiciones:**

- El administrador se ha identificado en el sistema previamente.

■ **Postcondiciones:**

- Se envía al sistema el nombre y la descripción del servicio a añadir.

Contrato de operación: crearServicio(nombre, descripcion)

■ **Referencias cruzadas:** UC-28 (Cuadro 4.28).

■ **Responsabilidades:** Creación de un nuevo servicio con los datos recibidos.

■ **Precondiciones:**

- Se han recibido los datos de creación de un nuevo servicio.

■ **Postcondiciones:**

- Se comprueban que el nombre del servicio es único.
- Se crea un nuevo servicio y se introducen los datos en la base de datos.

Contrato de operación: mostrarNotificacion()

■ **Referencias cruzadas:** UC-28 (Cuadro 4.28).

- **Responsabilidades:** Se mostrará un mensaje de acción por pantalla.
- **Precondiciones:**
 - Se ha realizado la acción correspondiente para activar el mensaje.
- **Postcondiciones:**
 - Se muestra el mensaje correspondiente a la acción en la pantalla, a modo de notificación para el administrador. Este puede ser confirmación de la acción o algún tipo de error en la ejecución de la misma.

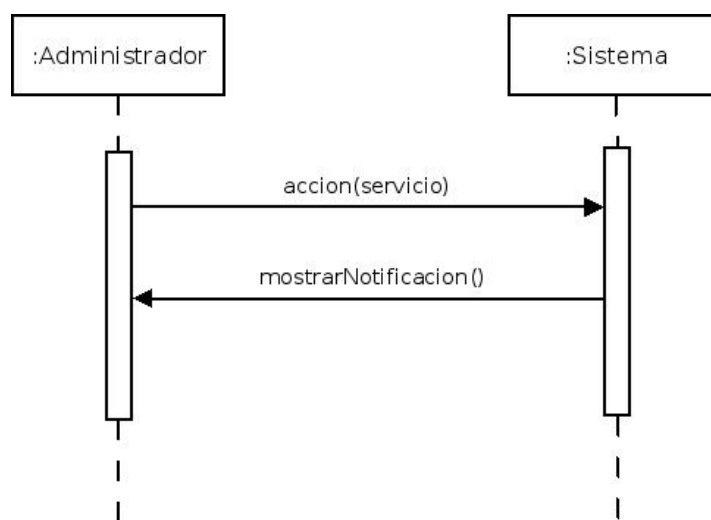


Figura 4.29: Diagrama de secuencia: Gestión de servicios: Activar/Suspender servicio

Contrato de operación: `accion(servicio)`

- **Referencias cruzadas:** UC-29 (Cuadro 4.29).
- **Responsabilidades:** El administrador podrá activar o suspender el servicio seleccionado.
- **Precondiciones:**
 - El administrador se ha identificado en el sistema previamente.
 - El servicio seleccionado debe estar activo para ser suspendido o viceversa.
- **Postcondiciones:**
 - Se realizará la acción seleccionada por el administrador (*activar*, *suspender*) referente a un servicio específico.

Contrato de operación: `mostrarNotificacion()`

- **Referencias cruzadas:** UC-29 (Cuadro 4.29).

- **Responsabilidades:** Se mostrará un mensaje de acción por pantalla.
- **Precondiciones:**
 - Se ha realizado la acción correspondiente para activar el mensaje.
- **Postcondiciones:**
 - Se muestra el mensaje correspondiente a la acción en la pantalla, a modo de notificación para el administrador. Este puede ser confirmación de la acción o algún tipo de error en la ejecución de la misma.

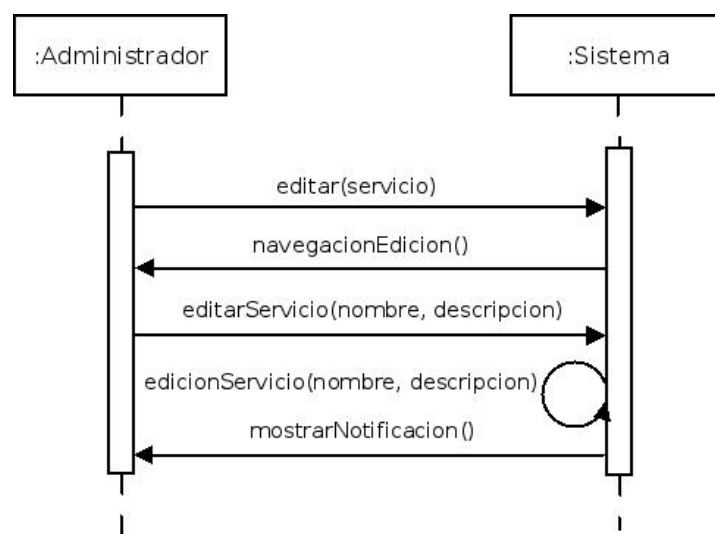


Figura 4.30: Diagrama de secuencia: Gestión de servicios: Editar servicio

Contrato de operación: editar(servicio)

- **Referencias cruzadas:** UC-30 (Cuadro 4.30).
- **Responsabilidades:** Se solicitará al sistema la edición de un servicio específico por parte del administrador.
- **Precondiciones:**
 - El administrador se ha identificado en el sistema previamente.
- **Postcondiciones:**
 - Se manda la solicitud de edición del servicio seleccionado.

Contrato de operación: navegacionEdicion()

- **Referencias cruzadas:** UC-30 (Cuadro 4.30).
- **Responsabilidades:** El sistema mostrará la página correspondiente a la edición del servicio.

- **Precondiciones:**

- Se ha realizado la acción correspondiente para activar la navegación.

- **Postcondiciones:**

- Se mostrará la ventana de edición del servicio seleccionado.

Contrato de operación: editarServicio(nombre, descripcion)

- **Referencias cruzadas:** UC-30 (Cuadro 4.30).

- **Responsabilidades:** Se mandará al sistema los datos del servicio para que este sea editado.

- **Precondiciones:**

- El administrador se ha identificado en el sistema previamente.

- **Postcondiciones:**

- Se envía el nombre y la descripción del servicio a editar al sistema.

Contrato de operación: edicionServicio(nombre, descripcion)

- **Referencias cruzadas:** UC-30 (Cuadro 4.30).

- **Responsabilidades:** Se realizará la edición de los datos del servicio, guardándolos en el sistema.

- **Precondiciones:**

- Se ha enviado el formulario con los datos a editar.

- **Postcondiciones:**

- Se comprueba que el nombre del servicio sea único.
- El sistema guarda los datos recibidos para el servicio específico en la base de datos.

Contrato de operación: mostrarNotificacion()

- **Referencias cruzadas:** UC-30 (Cuadro 4.30).

- **Responsabilidades:** Se mostrará un mensaje de acción por pantalla.

- **Precondiciones:**

- Se ha realizado la acción correspondiente para activar el mensaje.

- **Postcondiciones:**

- Se muestra el mensaje correspondiente a la acción en la pantalla, a modo de notificación para el administrador. Este puede ser confirmación de la acción o algún tipo de error en la ejecución de la misma.

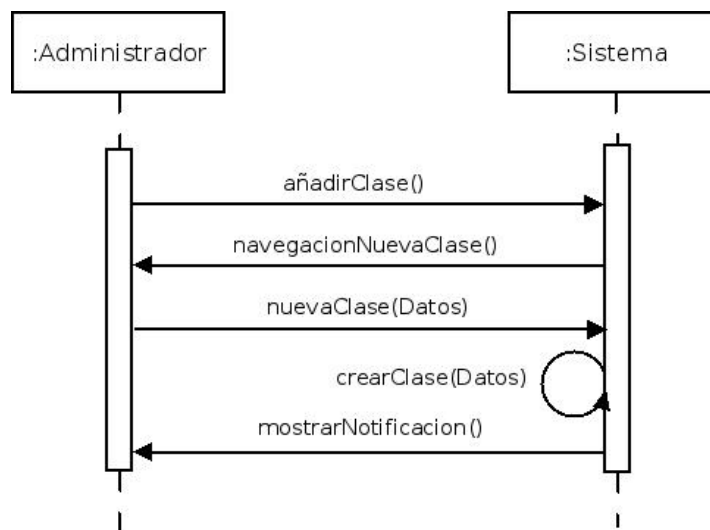


Figura 4.31: Diagrama de secuencia: Gestión de servicios: Alta clase

Contrato de operación: añadirClase()

- **Referencias cruzadas:** UC-31 (Cuadro 4.31).
- **Responsabilidades:** Solicitar al sistema crear una nueva clase.
- **Precondiciones:**
 - El administrador se ha identificado en el sistema previamente.
- **Postcondiciones:**
 - Se enviará al sistema la solicitud de crear una nueva clase por parte del administrador.

Contrato de operación: navegacionNuevaClase()

- **Referencias cruzadas:** UC-31 (Cuadro 4.31).
- **Responsabilidades:** El sistema mostrará la ventana correspondiente a la creación de una nueva clase.
- **Precondiciones:**
 - Se ha realizado la acción correspondiente para activar la navegación.
- **Postcondiciones:**
 - Se mostrará la ventana de creación de una clase nueva en la interfaz del administrador.

Contrato de operación: nuevaClase(Datos)

- **Referencias cruzadas:** UC-31 (Cuadro 4.31).
- **Responsabilidades:** Se mandará al sistema los datos necesarios para la creación de una nueva clase.

- **Precondiciones:**

- El administrador se ha identificado en el sistema previamente.

- **Postcondiciones:**

- Se envía al sistema los datos de la clase a añadir.

Contrato de operación: crearClase(Datos)

- **Referencias cruzadas:** UC-31 (Cuadro 4.31).

- **Responsabilidades:** Creación de una nueva clase con los datos recibidos.

- **Precondiciones:**

- Se han recibido los datos de creación de una nueva clase.

- **Postcondiciones:**

- Se comprueban que los datos recibidos son correctos.
- Se crea una nueva clase y se introducen los datos en la base de datos.

Contrato de operación: mostrarNotificacion()

- **Referencias cruzadas:** UC-31 (Cuadro 4.31).

- **Responsabilidades:** Se mostrará un mensaje de acción por pantalla.

- **Precondiciones:**

- Se ha realizado la acción correspondiente para activar el mensaje.

- **Postcondiciones:**

- Se muestra el mensaje correspondiente a la acción en la pantalla, a modo de notificación para el administrador. Este puede ser confirmación de la acción o algún tipo de error en la ejecución de la misma.

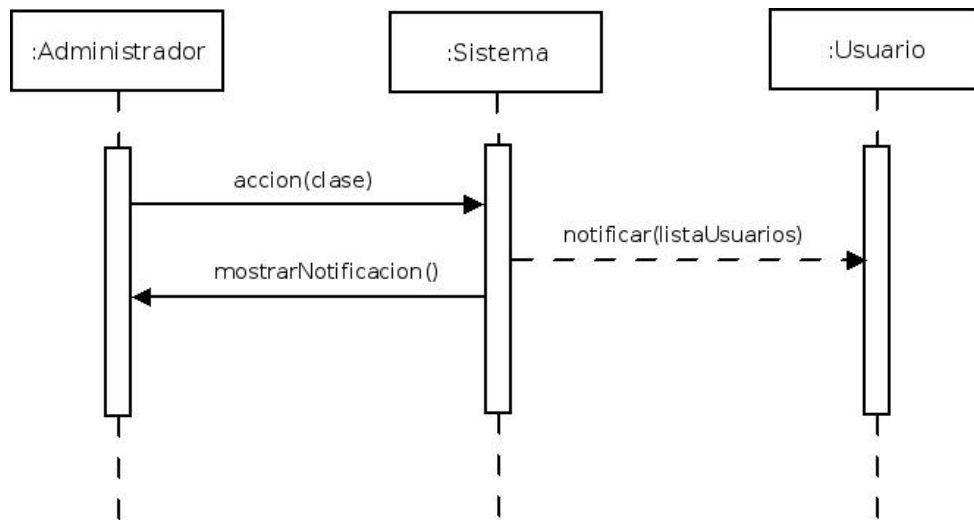


Figura 4.32: Diagrama de secuencia: Gestión de servicios: Activar/Suspender clase

Contrato de operación: `accion(clase)`

- **Referencias cruzadas:** UC-32 (Cuadro 4.32).
- **Responsabilidades:** El administrador podrá activar o suspender la clase seleccionada.
- **Precondiciones:**
 - El administrador se ha identificado en el sistema previamente.
 - La clase seleccionada debe estar activa para ser suspendida o viceversa.
- **Postcondiciones:**
 - Se realizará la acción seleccionada por el administrador (*activar, suspender*) referente a una clase específica.

Contrato de operación: `notificar(listaUsuarios)`

- **Referencias cruzadas:** UC-32 (Cuadro 4.32).
- **Responsabilidades:** Mandar una notificación de activación o suspensión de una clase específica por parte del administrador a todos los usuarios con reserva en la misma, en caso que existan.
- **Precondiciones:**
 - Se ha activado o suspendido una clase.
- **Postcondiciones:**
 - Se creará una nueva notificación de activación o suspensión de una determinada clase para cada uno de los usuarios con reserva en la misma.

Contrato de operación: `mostrarNotificacion()`

- **Referencias cruzadas:** UC-32 (Cuadro 4.32).
- **Responsabilidades:** Se mostrará un mensaje de acción por pantalla.
- **Precondiciones:**
 - Se ha realizado la acción correspondiente para activar el mensaje.
- **Postcondiciones:**
 - Se muestra el mensaje correspondiente a la acción en la pantalla, a modo de notificación para el administrador. Este puede ser confirmación de la acción o algún tipo de error en la ejecución de la misma.

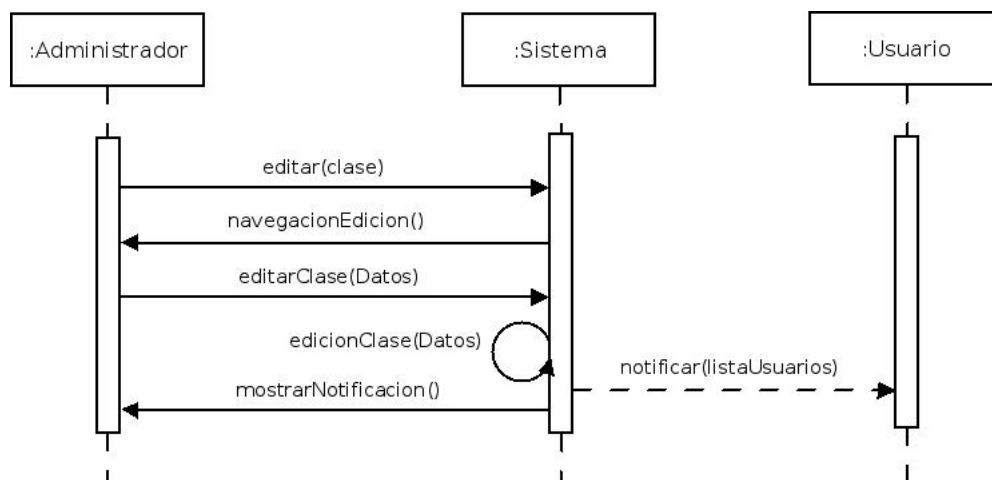


Figura 4.33: Diagrama de secuencia: Gestión de servicios: Editar clase

Contrato de operación: editar(clase)

- **Referencias cruzadas:** UC-33 (Cuadro 4.33).
- **Responsabilidades:** Se solicitará al sistema la edición de una clase específica por parte del administrador.
- **Precondiciones:**
 - El administrador se ha identificado en el sistema previamente.
- **Postcondiciones:**
 - Se manda la solicitud de edición de la clase seleccionada.

Contrato de operación: navegacionEdicion()

- **Referencias cruzadas:** UC-33 (Cuadro 4.33).

- **Responsabilidades:** El sistema mostrará la página correspondiente a la edición de la clase.
- **Precondiciones:**
 - Se ha realizado la acción correspondiente para activar la navegación.
- **Postcondiciones:**
 - Se mostrará la ventana de edición de la clase seleccionada.

Contrato de operación: `editarClase(Datos)`

- **Referencias cruzadas:** UC-33 (Cuadro 4.33).
- **Responsabilidades:** Se mandará al sistema los datos de la clase para que esta sea editada.
- **Precondiciones:**
 - El administrador se ha identificado en el sistema previamente.
- **Postcondiciones:**
 - Se envían todos los datos del formulario de edición de la clase al sistema.

Contrato de operación: `edicionClase(Datos)`

- **Referencias cruzadas:** UC-33 (Cuadro 4.33).
- **Responsabilidades:** Se realizará la edición de los datos de la clase, guardándolos en el sistema.
- **Precondiciones:**
 - Se ha enviado el formulario con los datos a editar.
- **Postcondiciones:**
 - Se comprueban que los datos recibidos son correctos.
 - El sistema guarda los datos recibidos para la clase específica en la base de datos.

Contrato de operación: `mostrarNotificacion()`

- **Referencias cruzadas:** UC-33 (Cuadro 4.33).
- **Responsabilidades:** Se mostrará un mensaje de acción por pantalla.
- **Precondiciones:**
 - Se ha realizado la acción correspondiente para activar el mensaje.
- **Postcondiciones:**
 - Se muestra el mensaje correspondiente a la acción en la pantalla, a modo de notificación para el administrador. Este puede ser confirmación de la acción o algún tipo de error en la ejecución de la misma.

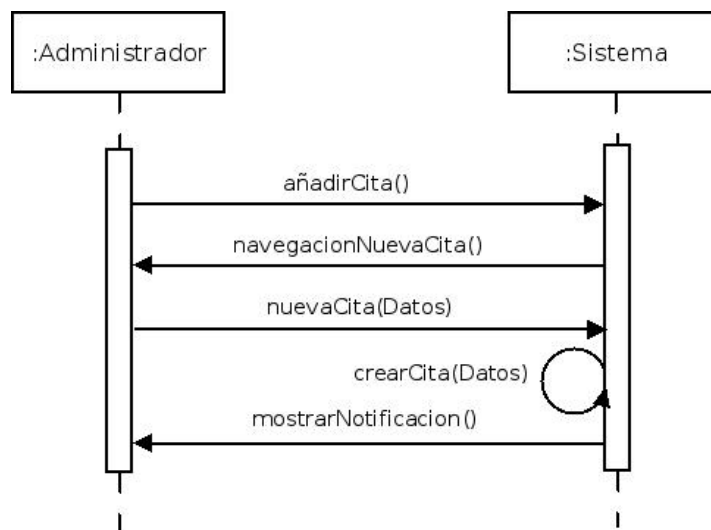


Figura 4.34: Diagrama de secuencia: Gestión de servicios: Alta cita

Contrato de operación: añadirCita()

- **Referencias cruzadas:** UC-34 (Cuadro 4.34).
- **Responsabilidades:** Solicitar al sistema crear una nueva cita.
- **Precondiciones:**
 - El administrador se ha identificado en el sistema previamente.
- **Postcondiciones:**
 - Se enviará al sistema la solicitud de crear una nueva cita por parte del administrador.

Contrato de operación: navegacionNuevaCita()

- **Referencias cruzadas:** UC-34 (Cuadro 4.34).
- **Responsabilidades:** El sistema mostrará la ventana correspondiente a la creación de una nueva cita.
- **Precondiciones:**
 - Se ha realizado la acción correspondiente para activar la navegación.
- **Postcondiciones:**
 - Se mostrará la ventana de creación de una cita nueva en la interfaz del administrador.

Contrato de operación: nuevaCita(Datos)

- **Referencias cruzadas:** UC-34 (Cuadro 4.34).
- **Responsabilidades:** Se mandará al sistema los datos necesarios para la creación de una nueva cita.

- **Precondiciones:**

- El administrador se ha identificado en el sistema previamente.

- **Postcondiciones:**

- Se envía al sistema los datos de la cita a añadir.

Contrato de operación: crearCita(Datos)

- **Referencias cruzadas:** UC-34 (Cuadro 4.34).

- **Responsabilidades:** Creación de una nueva cita con los datos recibidos.

- **Precondiciones:**

- Se han recibido los datos de creación de una nueva cita.

- **Postcondiciones:**

- Se comprueban que los datos recibidos son correctos.
- Se crea una nueva cita y se introducen los datos en la base de datos.

Contrato de operación: mostrarNotificacion()

- **Referencias cruzadas:** UC-34 (Cuadro 4.34).

- **Responsabilidades:** Se mostrará un mensaje de acción por pantalla.

- **Precondiciones:**

- Se ha realizado la acción correspondiente para activar el mensaje.

- **Postcondiciones:**

- Se muestra el mensaje correspondiente a la acción en la pantalla, a modo de notificación para el administrador. Este puede ser confirmación de la acción o algún tipo de error en la ejecución de la misma.

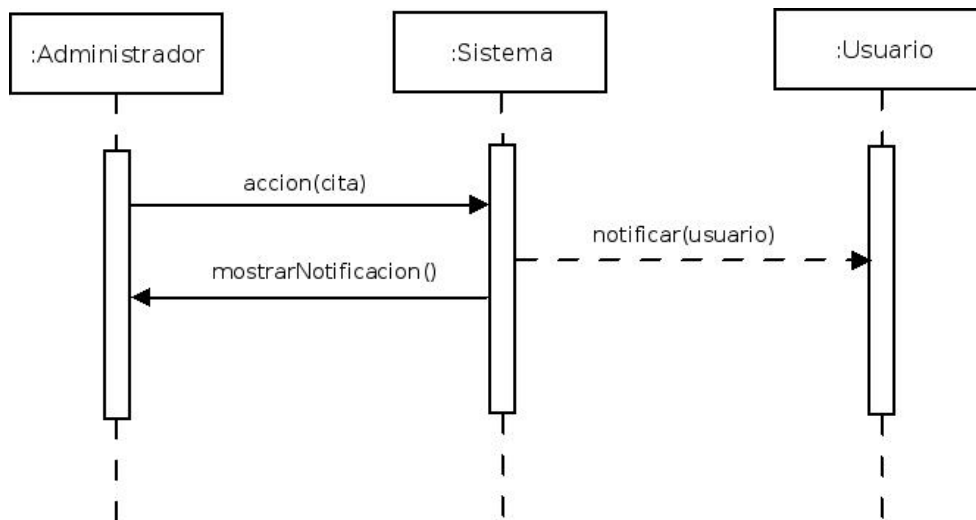


Figura 4.35: Diagrama de secuencia: Gestión de servicios: Activar/Suspender cita

Contrato de operación: `accion(cita)`

- **Referencias cruzadas:** UC-35 (Cuadro 4.35).
- **Responsabilidades:** El administrador podrá activar o suspender la cita seleccionada.
- **Precondiciones:**
 - El administrador se ha identificado en el sistema previamente.
 - La cita seleccionada debe estar activa para ser suspendida o viceversa.
- **Postcondiciones:**
 - Se realizará la acción seleccionada por el administrador (*activar, suspender*) referente a una cita específica.

Contrato de operación: `notificar(listaUsuarios)`

- **Referencias cruzadas:** UC-35 (Cuadro 4.35).
- **Responsabilidades:** Mandar una notificación de activación o suspensión de una cita específica por parte del administrador a todos los usuarios con reserva en la misma, en caso que existan.
- **Precondiciones:**
 - Se ha activado o suspendido una cita.
- **Postcondiciones:**
 - Se creará una nueva notificación de activación o suspensión de una determinada cita para cada uno de los usuarios con reserva en la misma.

Contrato de operación: `mostrarNotificacion()`

- **Referencias cruzadas:** UC-35 (Cuadro 4.35).
- **Responsabilidades:** Se mostrará un mensaje de acción por pantalla.
- **Precondiciones:**
 - Se ha realizado la acción correspondiente para activar el mensaje.
- **Postcondiciones:**
 - Se muestra el mensaje correspondiente a la acción en la pantalla, a modo de notificación para el administrador. Este puede ser confirmación de la acción o algún tipo de error en la ejecución de la misma.

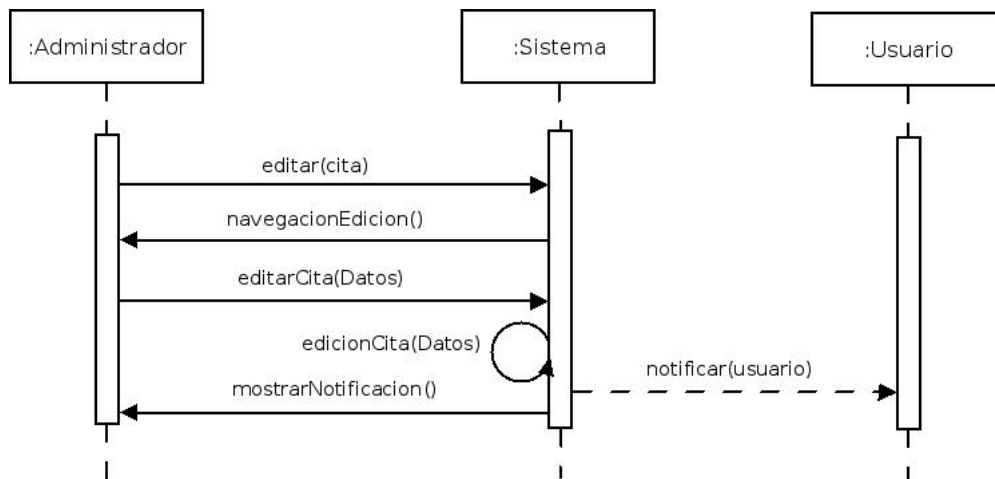


Figura 4.36: Diagrama de secuencia: Gestión de servicios: Editar cita

Contrato de operación: `editar(cita)`

- **Referencias cruzadas:** UC-36 (Cuadro 4.36).
- **Responsabilidades:** Se solicitará al sistema la edición de una cita específica por parte del administrador.
- **Precondiciones:**
 - El administrador se ha identificado en el sistema previamente.
- **Postcondiciones:**
 - Se manda la solicitud de edición de la cita seleccionada.

Contrato de operación: `navegacionEdicion()`

- **Referencias cruzadas:** UC-36 (Cuadro 4.36).
- **Responsabilidades:** El sistema mostrará la página correspondiente a la edición de la cita.

- **Precondiciones:**

- Se ha realizado la acción correspondiente para activar la navegación.

- **Postcondiciones:**

- Se mostrará la ventana de edición de la cita seleccionada.

Contrato de operación: editarCita(Datos)

- **Referencias cruzadas:** UC-36 (Cuadro 4.36).

- **Responsabilidades:** Se mandará al sistema los datos de la cita para que esta sea editada.

- **Precondiciones:**

- El administrador se ha identificado en el sistema previamente.

- **Postcondiciones:**

- Se envían todos los datos del formulario de edición de la cita al sistema.

Contrato de operación: edicionCita(Datos)

- **Referencias cruzadas:** UC-36 (Cuadro 4.36).

- **Responsabilidades:** Se realizará la edición de los datos de la cita, guardándolos en el sistema.

- **Precondiciones:**

- Se ha enviado el formulario con los datos a editar.

- **Postcondiciones:**

- Se comprueban que los datos recibidos son correctos.
- El sistema guarda los datos recibidos para la cita específica en la base de datos.

Contrato de operación: mostrarNotificacion()

- **Referencias cruzadas:** UC-36 (Cuadro 4.36).

- **Responsabilidades:** Se mostrará un mensaje de acción por pantalla.

- **Precondiciones:**

- Se ha realizado la acción correspondiente para activar el mensaje.

- **Postcondiciones:**

- Se muestra el mensaje correspondiente a la acción en la pantalla, a modo de notificación para el administrador. Este puede ser confirmación de la acción o algún tipo de error en la ejecución de la misma.

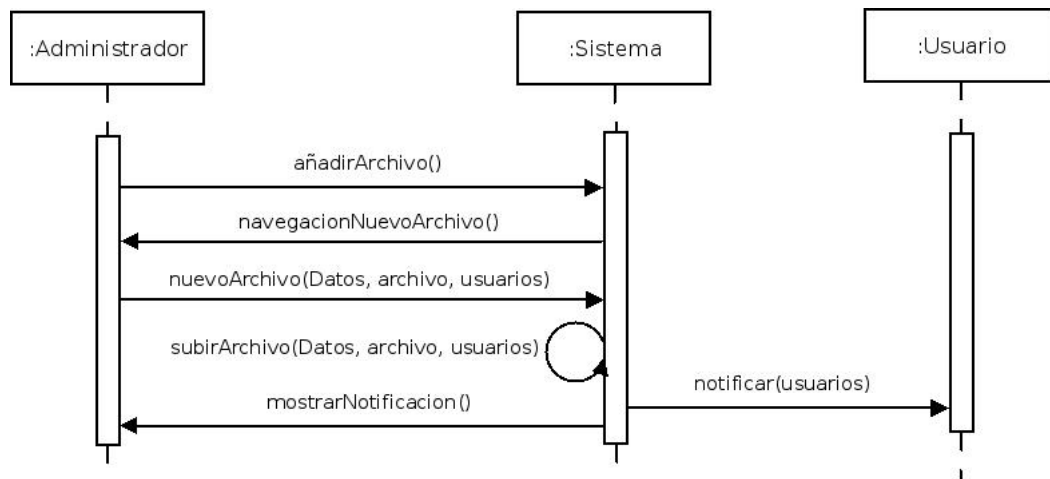


Figura 4.37: Diagrama de secuencia: Gestión de servicios: Subir archivo

Contrato de operación: `añadirArchivo()`

- **Referencias cruzadas:** UC-37 (Cuadro 4.37).
- **Responsabilidades:** Solicitar al sistema subir un archivo.
- **Precondiciones:**
 - El administrador se ha identificado en el sistema previamente.
- **Postcondiciones:**
 - Se enviará al sistema la solicitud de subir un documento por parte del administrador.

Contrato de operación: `navegacionNuevoArchivo()`

- **Referencias cruzadas:** UC-37 (Cuadro 4.37).
- **Responsabilidades:** El sistema mostrará la ventana correspondiente a la subida de un nuevo archivo.
- **Precondiciones:**
 - Se ha realizado la acción correspondiente para activar la navegación.
- **Postcondiciones:**
 - Se mostrará la ventana de subida de un nuevo archivo en la interfaz del administrador.

Contrato de operación: `nuevoArchivo(Datos, archivo, usuarios)`

- **Referencias cruzadas:** UC-37 (Cuadro 4.37).
- **Responsabilidades:** Se mandará al sistema los datos necesarios para subir el archivo.
- **Precondiciones:**
 - El administrador se ha identificado en el sistema previamente.

- **Postcondiciones:**

- Se envía al sistema los datos del archivo a subir, el propio archivo y el usuario o lista de usuarios a los que el documento va destinado.

Contrato de operación: subirArchivo(Datos, archivo, usuarios)

- **Referencias cruzadas:** UC-37 (Cuadro 4.37).

- **Responsabilidades:** Almacenamiento de una nuevo archivo, con los datos y archivo recibidos.

- **Precondiciones:**

- Se han recibido los datos y el archivo para la creación de un documento nuevo.

- **Postcondiciones:**

- Se comprueban que los datos recibidos y el archivo son correctos.
- Se crea y almacena un nuevo archivo y se introducen los datos en la base de datos.
- Se especifica qué usuarios tienen acceso al archivo para su posterior consulta o descarga.

Contrato de operación: notificar(usuarios)

- **Referencias cruzadas:** UC-37 (Cuadro 4.37).

- **Responsabilidades:** Mandar una notificación de documento disponible a los usuarios especificados al subir el archivo, en caso que existan.

- **Precondiciones:**

- Se ha subido al sistema un nuevo documento.

- **Postcondiciones:**

- Se creará una nueva notificación de archivo subido disponible para cada uno de los usuarios especificados por el administrador en el proceso de subida.

Contrato de operación: mostrarNotificacion()

- **Referencias cruzadas:** UC-37 (Cuadro 4.37).

- **Responsabilidades:** Se mostrará un mensaje de acción por pantalla.

- **Precondiciones:**

- Se ha realizado la acción correspondiente para activar el mensaje.

- **Postcondiciones:**

- Se muestra el mensaje correspondiente a la acción en la pantalla, a modo de notificación para el administrador. Este puede ser confirmación de la acción o algún tipo de error en la ejecución de la misma.

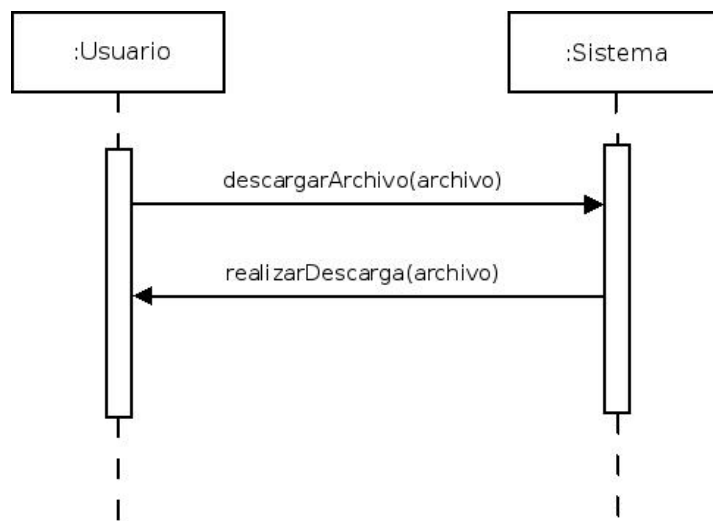


Figura 4.38: Diagrama de secuencia: Gestión de servicios: Descargar archivo

Contrato de operación: descargarArchivo(archivo)

- **Referencias cruzadas:** UC-38 (Cuadro 4.38).
- **Responsabilidades:** Solicitar al sistema la descarga del archivo seleccionado.
- **Precondiciones:**
 - El usuario se ha identificado en el sistema previamente.
 - Existen documentos subidos destinados al usuario.
- **Postcondiciones:**
 - El usuario manda al sistema la solicitud de descarga de un archivo específico.

Contrato de operación: realizarDescarga(archivo)

- **Referencias cruzadas:** UC-38 (Cuadro 4.38).
- **Responsabilidades:** Realizar la descarga del documento seleccionado al dispositivo del usuario.
- **Precondiciones:**
 - El usuario ha solicitado la descarga de un archivo específico.
- **Postcondiciones:**
 - El documento seleccionado se descargará al dispositivo del usuario.

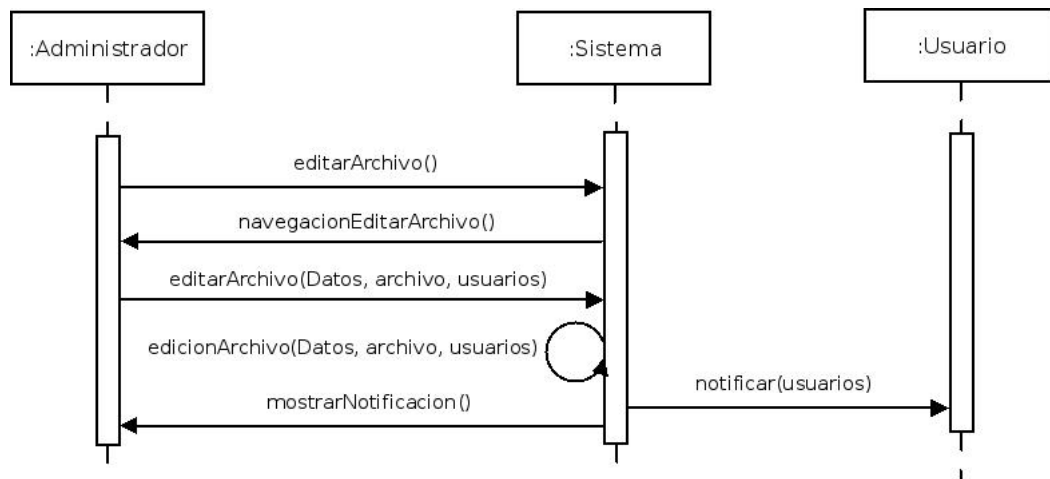


Figura 4.39: Diagrama de secuencia: Gestión de servicios: Editar archivo

Contrato de operación: editar(archivo)

- **Referencias cruzadas:** UC-39 (Cuadro 4.39).
- **Responsabilidades:** Se solicitará al sistema la edición de un archivo específico por parte del administrador.
- **Precondiciones:**
 - El administrador se ha identificado en el sistema previamente.
 - Existen documentos subidos.
- **Postcondiciones:**
 - Se manda la solicitud de edición del archivo seleccionado.

Contrato de operación: navegacionEdicion()

- **Referencias cruzadas:** UC-39 (Cuadro 4.39).
- **Responsabilidades:** El sistema mostrará la página correspondiente a la edición del archivo.
- **Precondiciones:**
 - Se ha realizado la acción correspondiente para activar la navegación.
- **Postcondiciones:**
 - Se mostrará la ventana de edición del archivo seleccionado.

Contrato de operación: editarArchivo(Datos, usuarios)

- **Referencias cruzadas:** UC-39 (Cuadro 4.39).
- **Responsabilidades:** Se mandará al sistema los datos del archivo y usuarios a los que va destinado para que sea editado.

- **Precondiciones:**

- El administrador se ha identificado en el sistema previamente.

- **Postcondiciones:**

- Se envían todos los datos del formulario de edición del archivo al sistema.
- Se especifican todos los usuarios que tendrán acceso al documento.

Contrato de operación: `edicionArchivo(Datos)`

- **Referencias cruzadas:** UC-39 (Cuadro 4.39).

- **Responsabilidades:** Se realizará la edición de los datos del archivo y la lista de usuarios con permiso para su acceso, guardándolos en el sistema.

- **Precondiciones:**

- Se ha enviado el formulario con los datos a editar y la lista de usuarios a los que el documento va destinado.

- **Postcondiciones:**

- Se comprueban que los datos recibidos son correctos.
- El sistema guarda los datos recibidos para el archivo específico en la base de datos.
- El sistema da autorización a los usuarios especificados para que tengan acceso al archivo.

Contrato de operación: `notificar(usuarios)`

- **Referencias cruzadas:** UC-39 (Cuadro 4.39).

- **Responsabilidades:** Mandar una notificación de documento editado a los usuarios especificados con acceso al archivo, en caso que existan.

- **Precondiciones:**

- Se ha editado un documento.

- **Postcondiciones:**

- Se creará una nueva notificación de archivo editado disponible para cada uno de los usuarios especificados por el administrador en el proceso de edición.

Contrato de operación: `mostrarNotificacion()`

- **Referencias cruzadas:** UC-39 (Cuadro 4.39).

- **Responsabilidades:** Se mostrará un mensaje de acción por pantalla.

- **Precondiciones:**

- Se ha realizado la acción correspondiente para activar el mensaje.

- **Postcondiciones:**

- Se muestra el mensaje correspondiente a la acción en la pantalla, a modo de notificación para el administrador. Este puede ser confirmación de la acción o algún tipo de error en la ejecución de la misma.

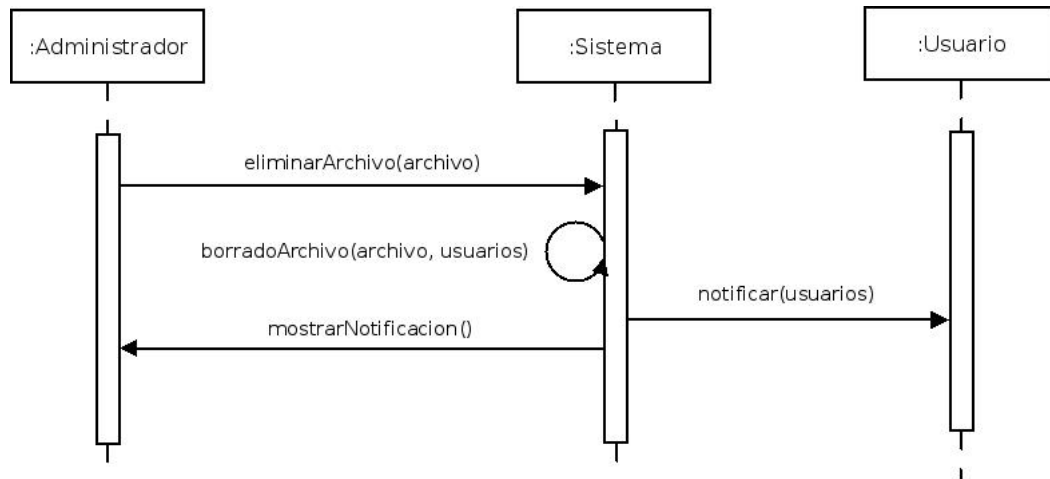


Figura 4.40: Diagrama de secuencia: Gestión de servicios: Eliminar archivo

Contrato de operación: **eliminarArchivo(archivo)**

- **Referencias cruzadas:** UC-40 (Cuadro 4.40).
- **Responsabilidades:** Solicitar al sistema la eliminación del archivo seleccionado.
- **Precondiciones:**
 - El administrador se ha identificado en el sistema previamente.
 - Existen documentos subidos.
- **Postcondiciones:**
 - El usuario manda al sistema la solicitud de eliminación de un archivo específico.

Contrato de operación: **borradoArchivo(archivo)**

- **Referencias cruzadas:** UC-40 (Cuadro 4.40).
- **Responsabilidades:** Realizar la eliminación del documento seleccionado.
- **Precondiciones:**
 - El administrador ha solicitado la eliminación de un archivo específico.
- **Postcondiciones:**
 - Los datos del documento seleccionado se eliminarán de la base de datos y el propio archivo de donde esté almacenado.

Contrato de operación: `notificar(usuarios)`

- **Referencias cruzadas:** UC-40 (Cuadro 4.40).
- **Responsabilidades:** Mandar una notificación de documento eliminado a los usuarios con permisos para ver el archivo, en caso que existan.
- **Precondiciones:**
 - Se ha eliminado del sistema un documento.
- **Postcondiciones:**
 - Se creará una nueva notificación de archivo eliminado para cada uno de los usuarios con permisos de acceso al archivo, especificados por el administrador.

Contrato de operación: `mostrarNotificacion()`

- **Referencias cruzadas:** UC-40 (Cuadro 4.40).
- **Responsabilidades:** Se mostrará un mensaje de acción por pantalla.
- **Precondiciones:**
 - Se ha realizado la acción correspondiente para activar el mensaje.
- **Postcondiciones:**
 - Se muestra el mensaje correspondiente a la acción en la pantalla, a modo de notificación para el administrador. Este puede ser confirmación de la acción o algún tipo de error en la ejecución de la misma.

Capítulo 5

Diseño del Sistema

A lo largo de este capítulo se detallará la arquitectura general del sistema de información, el diseño físico de datos, el diseño detallado de componentes software y el diseño de la interfaz de usuario:

5.1. Arquitectura del Sistema

En esta sección se define la arquitectura general del sistema de información, especificando la infraestructura tecnológica necesaria para dar soporte al software y la estructura de los componentes que lo forman.

5.1.1. Arquitectura Física

El desarrollo de este proyecto no precisa de ningún elemento hardware adicional al equipo de trabajo del alumno. Se ha utilizado un portátil MacBook Pro de 15 pulgadas, con procesador Intel Core i7 de 2.2 GHz y 16GB de memoria RAM DDR3. Para la realización de pruebas, se usará tanto este equipo como otro portátil del propio alumno, donde se instalará todo el software requerido para comprobar que la instalación y ejecución de la aplicación responde adecuadamente en un equipo diferente. En este caso será un portátil Acer con procesador Dual Core, 4GB de memoria RAM y disco duro SSD.

Respecto al software, el MacBook trabaja bajo el sistema operativo macOS Sierra. Todo el proyecto se ha desarrollado utilizando el IDE NetBeans 8.0.2 y usando el servidor de aplicaciones GlassFish en un entorno local. Para la documentación, se ha utilizado TeXShop, como herramienta de edición para \LaTeX . Para las pruebas, el portátil a utilizar correrá bajo Windows 7, usando el mismo IDE y servidor de aplicaciones.

Respecto al entorno de producción, la aplicación web se alojará en un servidor que se contratará para tal fin. Por lo tanto, solo se requerirá acceso al servidor para la instalación de, en este caso, el servidor de aplicaciones Wildfly (JBoss) -habiendo sido probado previamente en entorno local de desarrollo-, siendo este similar a GlassFish, por lo que la aplicación apenas requerirá cambio alguno y aportará algo más de robustez y calidad. Los usuarios del sistema podrán hacer uso del mismo utilizando cualquier dispositivo con acceso a internet a través de un navegador web, como sus propios móviles, tablets o PCs.

En el apartado 6.1 se describirá detalladamente todo el software, lenguaje, frameworks, etc. utilizados para el desarrollo del sistema.

5.1.2. Arquitectura Lógica

Para el desarrollo de esta aplicación web se ha utilizado una arquitectura de 3 capas, basada en el patrón de diseño Modelo-Vista-Controlador (MVC), donde la primera capa correspondería a la capa de usuario, la segunda la de negocio y por último la capa de datos.

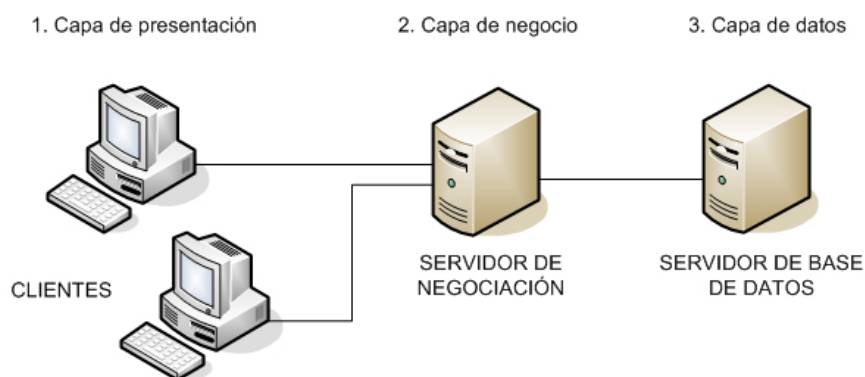


Figura 5.1: Representación de Arquitectura de 3 Capas

La programación por capas es un modelo de desarrollo software en el que el objetivo primordial es la separación (desacoplamiento) de las partes que componen un sistema software o también una arquitectura cliente-servidor: lógica de negocios, capa de presentación y capa de datos. De esta forma, por ejemplo, es sencillo y mantenible crear diferentes interfaces sobre un mismo sistema sin requerirse cambio alguno en la capa de datos o lógica.

La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, solo afectará al nivel requerido sin tener que revisar entre el código fuente de otros módulos ([?]).

Capa de presentación (frontend) Este grupo de artefactos software conforman la capa de presentación del sistema, incluyendo tanto los componentes de la vista como los elementos de control de la misma.

Es la capa que ve el usuario, denominada también *capa de usuario*. Presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). También es conocida como interfaz gráfica y debe tener la característica de ser amigable (entendible y fácil de usar) para el usuario. Esta capa se comunica únicamente con la capa de negocio.

Capa de negocio Esta capa recibe las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (o de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él.

Capa de persistencia Este grupo de artefactos software conforman la capa de integración del sistema, incluyendo las clases de abstracción para el acceso a datos.

Es donde residen los datos y es la encargada de acceder a los mismos. Está formada por un gestor de base de datos que realiza todo el almacenamiento de datos, recibe solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Es común que a la capa de negocio y de datos de los sistemas web se denomine conjuntamente como backend de la aplicación.

Como se ha comentado anteriormente, el patrón de diseño usado para el desarrollo del proyecto ha sido Modelo-Vista-Controlador, el cual separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento ([?]).

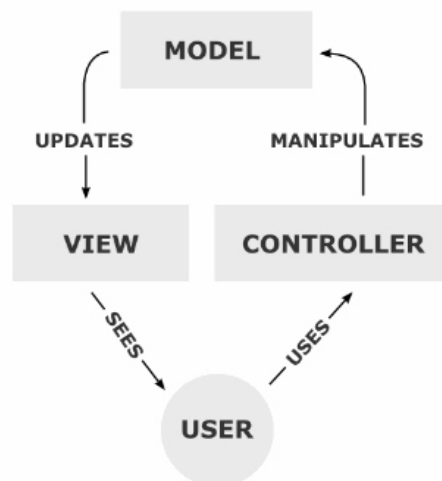


Figura 5.2: Proceso del Patrón MVC

Modelo Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la *vista* aquella parte de la información que en cada momento se le solicita para que sea mostrada al usuario. Las peticiones de acceso o manipulación de información llegan al *modelo* a través del *controlador*.

Controlador Responde a eventos (acciones del usuario) e invoca peticiones al *modelo* cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en la base de datos). También puede enviar comandos a su *vista* asociada si se solicita un cambio en la forma en que se presenta el *modelo* (por ejemplo, desplazamiento o scroll por un documento o por los diferentes registros de una base de datos), por tanto se podría decir que el *controlador* hace de intermediario entre la *vista* y el *modelo*.

Vista Presenta el *modelo* (información y lógica de negocio) en un formato adecuado para interactuar (la interfaz de usuario), por tanto requiere de dicho *modelo* la información que debe representar como salida.

Por tanto, aunque la arquitectura de 3 capas o niveles y el patrón MVC presenten sus similitudes y diferencias, cada uno tiene su función y son compatibles entre sí, de ahí el uso de ambos en el presente proyecto. A continuación, se presenta una gráfica comparativa de ambos modelos.

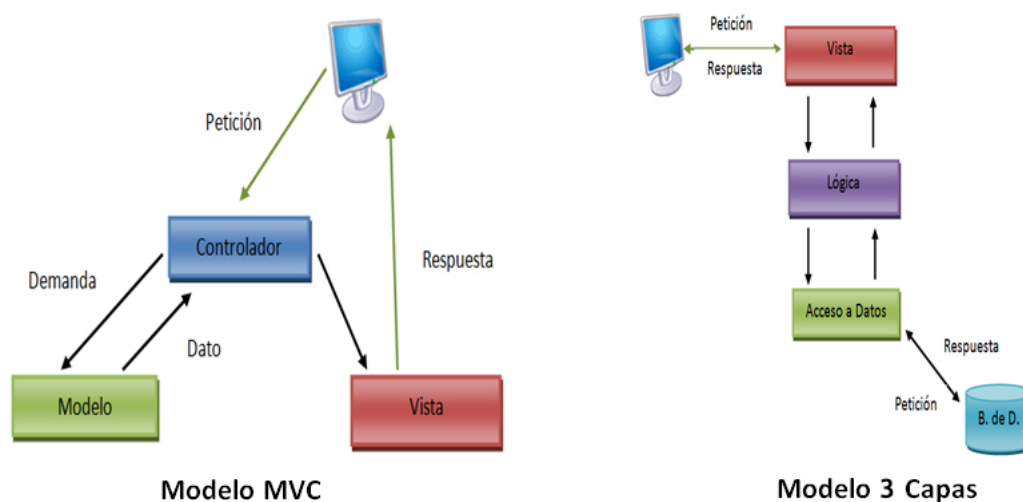


Figura 5.3: Comparativa entre modelo MVC y arquitectura de 3 capas

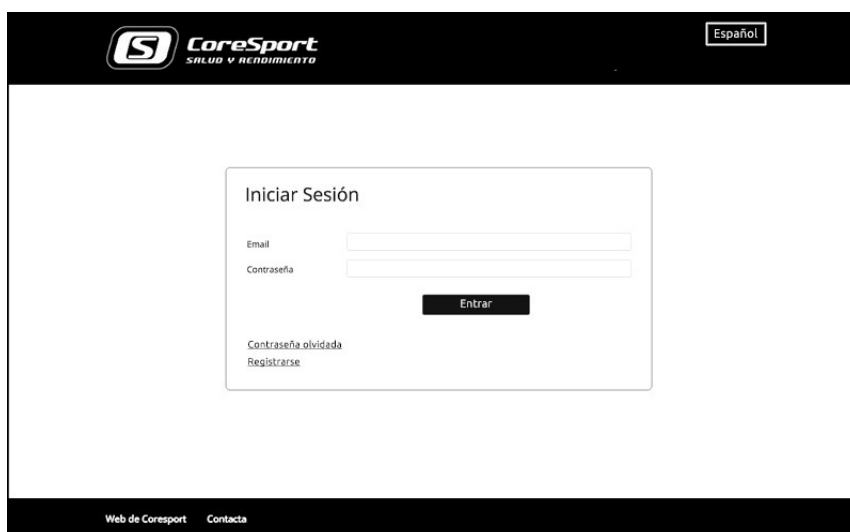
5.2. Diseño Físico de Datos

Habiendo realizado previamente el modelo de conceptual de clases, detallado en la sección 4.1, se puede tener una idea de la estructura física que tendrá los datos en el sistema de gestión de base de datos (SGBD) a utilizar, en este caso PostgreSQL, teniendo en cuenta que aparecerán nuevas tablas en la BD provenientes de las relaciones entre las clases. Pero, por supuesto, hay que tener en cuenta que el acceso a los mismos se realice de una forma eficaz e independiente al resto de la implementación.

Y es por ello por lo que la arquitectura lógica del sistema se divide en 3 capas bien diferenciadas. La tercera de las capas contendrá el DAO (*Data Access Object, Objeto de Acceso a Datos*), encargado del acceso a los datos físicos y única capa que realizará cambios en los mismos. Esto permite que si aflora la necesidad de cambios en la estructura de nuestros datos, o incluso cambiar de SGBD, las demás capas queden totalmente al margen de estos cambios, siendo un trámite independiente sin afectar -dependiendo del cambio, claro está- a la lógica de negocio o la interfaz de usuario.

5.3. Diseño de la Interfaz de Usuario

A continuación se muestra un prototipo de la interfaz de usuario del sistema para PC. Se mostrarán las páginas principales de la aplicación web de manera generalizada: pantalla de registro de usuario, inicio de sesión, pantalla generalizada de la aplicación una vez iniciada la sesión y página con prototipo de tabla de datos (para listado de usuarios, servicios, clases, etc.).



El prototipo muestra una interfaz web para el inicio de sesión. En la parte superior, hay una barra negra con el logo de CoreSport (un 'S' dentro de un círculo) y el texto 'CoreSport SALUD Y RENDIMIENTO' a la izquierda, y un botón 'Español' a la derecha. El contenido principal es un formulario centrado con el título 'Iniciar Sesión'. Dentro del formulario, hay dos campos de entrada: 'Email' y 'Contraseña'. Debajo de estos campos, hay un botón negro con el texto 'Entrar'. En la parte inferior del formulario, hay dos enlaces de texto: 'Contraseña olvidada' y 'Registrarse'. En la parte inferior de la página, hay una barra negra con el texto 'Web de Coresport' y 'Contacta'.

Figura 5.4: Interfaz de usuario: Inicio de sesión

CoreSport
SALUD Y RENDIMIENTO

Español

Registro

Nombre: Teléfono:

Primer Apellido: Dirección:

Segundo Apellido: Dirección (línea adicional):

Email: Ciudad:

Contraseña: País:

Confirmar Contraseña: Código Postal:

☐ He leído y acepto los términos y condiciones

Registrarse

Web de CoreSport Contacta

Figura 5.5: Interfaz de usuario: Registro

CoreSport
SALUD Y RENDIMIENTO

Español Jesús Soriano Salir

Inicio	Reservas	Administración	Mi Cuenta	Ayuda
--------	----------	----------------	-----------	-------

Calendario

Mis Reservas

Web de CoreSport Contacta

Figura 5.6: Interfaz de usuario: Pantalla general



Figura 5.7: Interfaz de usuario: Pantalla con tabla de datos

Asimismo, se ha realizado el diseño de las pantallas para dispositivos de menor tamaño, al ser un diseño adaptativo dependiendo del mismo. A continuación se podrán visualizar los mockups realizados para la interfaz de dispositivos móviles. En este caso, las pantalla de inicio de sesión, pantalla general de usuario y menú desplegado.



Figura 5.8: Interfaz de usuario para dispositivos móviles: Inicio de sesión

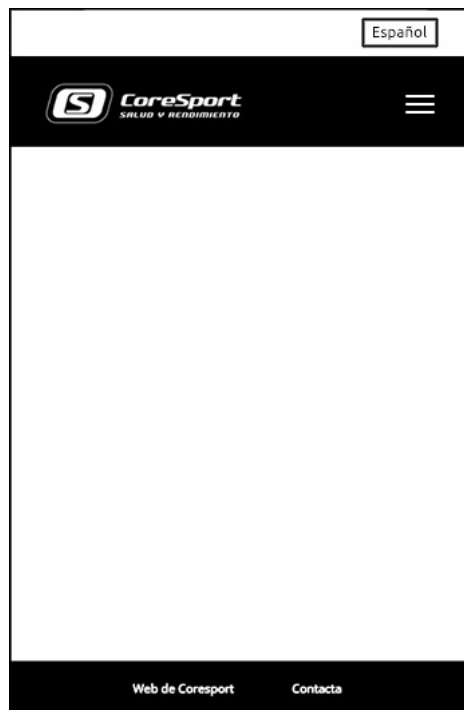


Figura 5.9: Interfaz de usuario para dispositivos móviles: Pantalla general



Figura 5.10: Interfaz de usuario para dispositivos móviles: Menú desplegable

Además, la figura 5.11 muestra el diagrama de navegación entre pantallas. Observamos que es una navegación sencilla; la página que se mostraría al acceder a la aplicación sería la de inicio de

sesión. Si se trata de un usuario registrado, podrá acceder directamente a la página principal de la aplicación (*Home*) a través del usuario y contraseña. En caso contrario, habría que navegar a la página de registro para que, una vez registrado, pueda acceder al sistema llegando a la página principal mencionada. Desde esta página de inicio (*Home*) se podrá navegar, a través del menú y/o enlaces disponibles, hasta las distintas vistas de la interfaz. En todo momento será posible cerrar la sesión del usuario, volviendo a la página de inicio de sesión, o cambiar el idioma de la interfaz mediante la opción destinada a ello en la parte superior derecha de la pantalla.

En el diagrama observamos que se navega hacia la vista de una tabla de datos. Este es un ejemplo de tantas vistas como hay en el sistema. Algunos ejemplos de tablas de datos pueden ser las página donde se listan los servicios, clases, citas, usuarios (para administradores) o reservas del usuario. Existen muchas más páginas en el sistema para navegar, como las páginas de perfil, cambio de contraseña, bandeja de entrada, calendario, creación/edición de servicios/clases/citas/usuarios para administradores, etc.

Esta navegación ocurrirá de la misma manera en todo tipo de dispositivos, donde el único cambio sería el diseño de la pantalla, como hemos observado en los prototipos para PC y móvil.

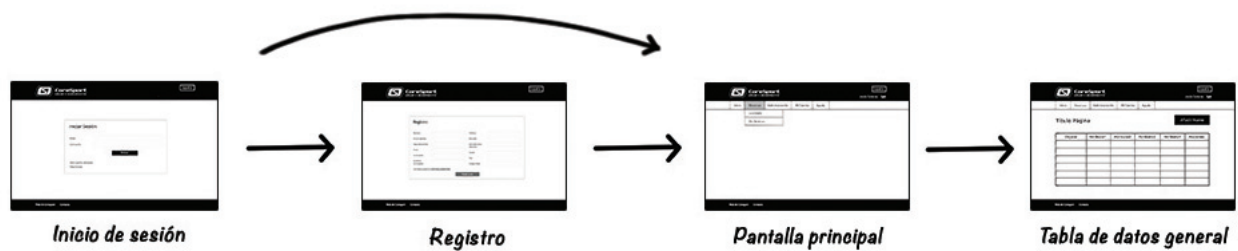


Figura 5.11: Interfaz de usuario: Diagrama de navegación

Capítulo 6

Construcción del Sistema

En este capítulo tratarán los aspectos relacionados con la implementación del sistema, así como del entorno tecnológico usado para el desarrollo del mismo.

6.1. Entorno de Construcción

Como se ha especificado en la sección 5.1.1, el desarrollo de este proyecto ha sido realizado haciendo uso del equipo del propio alumno, sin necesidad de alguna herramienta hardware extra. Para ello, se ha hecho uso de un marco tecnológico específico que se detallará a continuación:

Hardware Los elementos del hardware utilizados no son relevantes para el desarrollo del sistema, ya que no se requiere nada fuera de lo común en un equipo de trabajo convencional. En este caso, se ha utilizado un portátil MacBook Pro de 15 pulgadas, con procesador Intel Core i7 de 2.2 GHz y memoria RAM de 16GB 1333 MHz DDR3.

IDE (Entorno de Desarrollo Integrado) NetBeans [?] es un IDE libre y gratuito pensado especialmente en desarrollo de software bajo el uso del lenguaje de programación Java.

Lenguaje de Programación Para la realización de la aplicación web se ha utilizado el lenguaje de programación **Java**, en concreto la plataforma Java EE (Enterprise Edition), con la ayuda de varios frameworks para diferentes cometidos, como son JSF, PrimeFaces, EJB y JPA, que se describirán a continuación.

Frameworks

- **JSF (JavaServer Faces):** Framework MVC que proporciona un conjunto de componentes en forma de etiquetas definidas en páginas XHTML mediante el framework Facelets. Se utiliza para aplicaciones Java basadas en web simplificando el desarrollo de interfaces de usuario.
- **Facelets:** Framework basado que permite definir la estructura general de las páginas (su layout) mediante plantillas. Facelets se adapta perfectamente al enfoque de JSF y se incorpora a la especificación desde la revisión 2.1. La sustitución de JSP (JavaServer Pages) por Facelets como lenguaje básico para definir la disposición de las páginas permite separar perfectamente las responsabilidades de cada parte del framework. La estructura de la

página se define utilizando las etiquetas Facelets y los componentes específicos que deben presentar los datos de la aplicación utilizando etiquetas JSF. Para más información sobre JSF y/o Facelets véase el enlace [?] de la bibliografía.

- **PrimeFaces:** Este framework es una extensión de JSF de código abierto que cuenta con un conjunto de componentes enriquecidos para facilitar la creación de interfaces de usuario [?].
- **EJB (Enterprise JavaBeans):** Plataforma para construir aplicaciones empresariales portables, reusables y escalables, utilizando el lenguaje de programación java. EJB permite a los desarrolladores de aplicaciones enfocarse en construir la lógica de negocio sin la necesidad de gastar tiempo en la construcción de código de infraestructura [?].
- **JPA (Java Persistence API):** La persistencia dentro de EJB es administrada por JPA [?]. Este framework permite persistir automáticamente los objetos Java utilizando una técnica denominada object-relational mapping (ORM). ORM es esencialmente el proceso de mapear la información contenida en los objetos Java hacia las tablas de base de datos utilizando una configuración. JPA define un estándar para:
 - La creación de configuración metadata del ORM para mapear entidades hacia tablas relacionales.
 - La EntityManager API, una API estándar para realizar las operaciones CRUD (create, read, update y delete) de las entidades.
 - El lenguaje Java Persistence Query Language (JPQL), para realizar búsquedas y obtener información persistida de la aplicación.

En la siguiente figura podemos ver una representación de la integración de los frameworks descritos en la arquitectura de 3 capas vista en la sección 5.1.2.

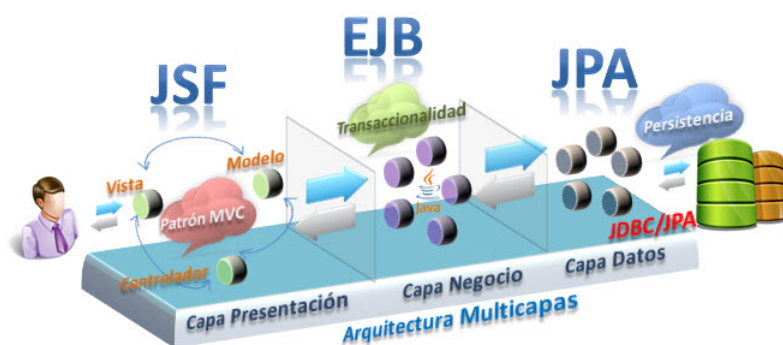


Figura 6.1: Arquitectura de 3 Capas con Frameworks

SGBD Se usará PostgreSQL, un sistema de gestión de bases de datos relacional orientado a objetos y libre. Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre o apoyados por organizaciones comerciales [?].

Para administrar la base de datos PostgreSQL se ha utilizado la herramienta pgAdmin [?].

Control de Versiones De sobra es conocido que para la realización de grandes proyectos, o aquellos que sean de carácter importante, es casi obligatorio el uso de copias de seguridad. Comúnmente, se utiliza un sistema de control de versiones: cómodo, seguro y fácil de usar.

Para la realización de este proyecto se ha usado Git [?], un sistema de control de versiones gratuito y de código abierto que garantiza confianza, eficacia y rapidez. Y en concreto, se ha usado la forja GitHub [?] para alojarlo.

6.1.1. Entorno para la Web Pública

No podemos olvidar que, aunque la documentación del proyecto se centre en la aplicación web del mismo, también se realiza un sitio web público para la empresa CoreSport [?].

Para la realización de esta web se ha utilizado el mismo equipo informático, pero distinto entorno software. En este caso, los IDE utilizados han sido Brackets [?] y Sublime [?], haciendo uso de HTML, CSS y JavaScript, como lenguajes para la realización de la web completa. Además, se ha usado el cliente FTP FileZilla [?] para alojar la misma.

6.2. Código Fuente

El código del proyecto, llamado Booking, se ha estructurado principalmente en 2 módulos: uno de ellos, Booking-war, contendría todo lo relativo a la interfaz gráfica, incluyendo los controladores de páginas xhtml (JSF), y el otro, Booking-ejb, toda la parte EJB, incluyendo la persistencia JPA. La figura 6.2 muestra la estructura general de los módulos y sus directorios principales.

6.2.1. Módulo Web

Veamos con algo más de detalle el módulo *Booking-war*. Observamos que se compone de 5 directorios bien diferenciados:

Web Pages

El primero de ellos hace referencia a los archivos de la interfaz de usuario, donde podemos distinguir por un lado el directorio *WEB-INF*, por otro *resources* y el resto de directorios que contienen todas las páginas XHTML divididas por carpetas dependientes del rol que el usuario posea.

WEB-INF Este directorio contiene, por una parte, todas las plantillas que se han creado para generar la interfaz de los distintos usuarios. Tendremos, por tanto, plantillas para los roles

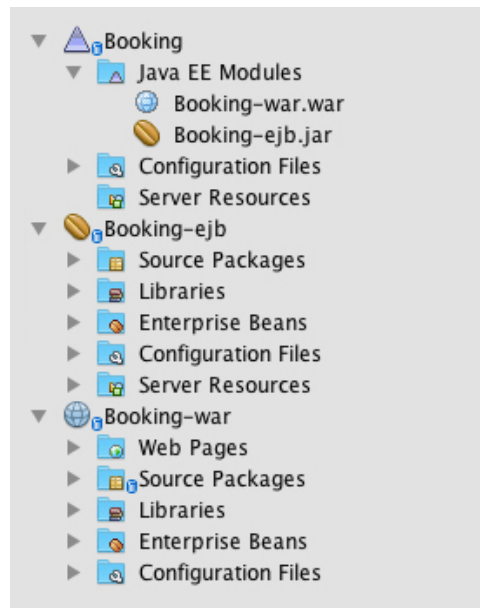


Figura 6.2: Estructura de los ficheros

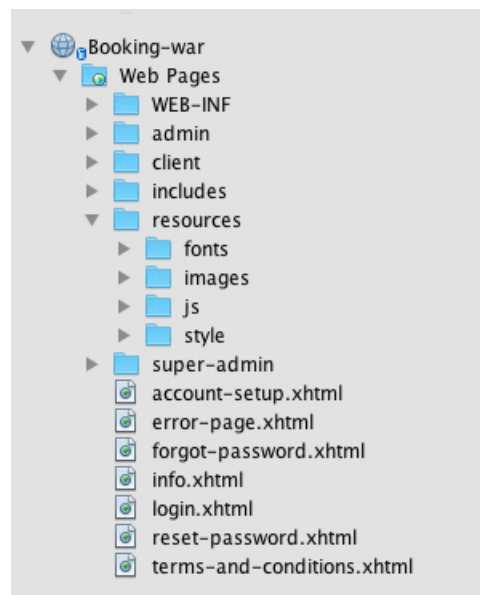


Figura 6.3: Directorio *Web Pages*

de superadministrador, administrador y usuario o cliente, además de las que estas usen por ser elementos en común, como pueden ser el footer (pie de página) o el selector de lenguaje.

Por otra parte, dentro de *WEB-INF* cabe destacar dos ficheros: *faces-config.xml* y *web.xml*, accesibles también desde el directorio *Configuration Files*. El primero de ellos se utiliza como fichero de configuración de idioma, donde se establece el idioma por defecto de la aplicación y aquellos que la misma soporta. En este caso se ha marcado el español como lenguaje por defecto, además de soportar el inglés, como vemos en su código:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <faces-config
3     xmlns="http://java.sun.com/xml/ns/javaee"
4     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5     xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
6     http://java.sun.com/xml/ns/javaee/web-facesconfig_2_0.xsd"
7     version="2.0">
8     <application>
9         <locale-config>
10             <default-locale>es</default-locale>
11             <supported-locale>en</supported-locale>
12         </locale-config>
13         <resource-bundle>
14             <base-name>com.booking.language.text</base-name>
15             <var>txt</var>
16         </resource-bundle>
17     </application>
18 </faces-config>
```

web.xml es un fichero de configuración donde podemos definir los parámetros principales de nuestra aplicación web, como pueden ser parámetros de autorización, redirecciones, tiempo máximo de sesión de usuario, gestión de errores, página de bienvenida por defecto, etc.

Así, podemos observar cómo se limita el acceso a los directorios dependiendo del rol del usuario: según la porción de código mostrada a continuación, ningún usuario estaría autorizado para acceder directamente a los archivos del directorio *include*, y el superadministrador solo accedería a los de la carpeta con su nombre. Existe la misma regla para el resto de roles (administrador y usuario). Los archivos que permanecen en el directorio *WEB-INF* sin incluirse en ningún subdirectorio sería accesible para todos los roles.

```
1     <security-constraint>
2         <display-name>NOT-ALLOWED-ANY</display-name>
3         <web-resource-collection>
4             <web-resource-name>NOT-ALLOWED-ANY</web-resource-name>
5             <description/>
6             <url-pattern>/includes/*</url-pattern>
7         </web-resource-collection>
8         <auth-constraint/>
9     </security-constraint>
10    <security-constraint>
11        <display-name>SUPER_ADMIN</display-name>
12        <web-resource-collection>
```

```

13         <web-resource-name>SUPER_ADMIN</web-resource-name>
14         <description/>
15         <url-pattern>/super-admin/*</url-pattern>
16         <!-- without using <http-method> will allow all http methods
            to be constrained : GET, PUT ETC... -->
17     </web-resource-collection>
18     <auth-constraint>
19         <description/>
20         <role-name>SUPER_ADMIN</role-name>
21     </auth-constraint>

```

Vemos también a continuación un ejemplo de cómo se tratarían los errores, habiendo creado previamente una página XHTML pensada para tal fin (*error-page.xhtml*):

```

1     <error-page>
2         <!-- Unauthorized -->
3         <error-code>401</error-code>
4         <location>/error-page.xhtml</location>
5     </error-page>
6     <error-page>
7         <!-- Forbidden -->
8         <error-code>403</error-code>
9         <location>/error-page.xhtml</location>
10    </error-page>
11    <error-page>
12        <!-- Not found -->
13        <error-code>404</error-code>
14        <location>/error-page.xhtml</location>
15    </error-page>

```

O cómo se establece la página de expiración de sesión:

```

1     <error-page>
2         <exception-type>javax.faces.application.ViewExpiredException</
            exception-type>
3         <location>/info.xhtml?info=session-expired</location>
4     </error-page>

```

En esta última porción de código vemos que el archivo *info.xhtml* acepta variables en la url, para indicar, en este caso, el tipo de información a mostrar. Así, el archivo nombrado mostrará información de sesión expirada, enlace expirado, o notificaciones del proceso para restablecer la contraseña olvidada. Este tipo de variables se usan en otras páginas de la aplicación, para consultar el perfil de algún usuario específico (siendo administrador para tener los permisos adecuados), ver los clases disponibles de un servicio o acciones de índole parecida donde se accede a los datos de una determinada instancia de una clase.

resources Es el directorio que contiene todas las fuentes, imágenes, archivos JavaScript y hojas de estilo CSS de la aplicación.

admin, client, includes y super-admin Directorios que contienen todas las páginas xhtml de cada uno de los roles que su nombre indica, es decir, las páginas principales de la interfaz de

usuario. La carpeta *includes* contiene todas las páginas que son comunes a varios roles, accediendo a ella a través de algún archivo de su propia carpeta, como podemos ver en el siguiente ejemplo:

```
1 <?xml version='1.0' encoding='UTF-8' ?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
  www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml"
4     xmlns:ui="http://java.sun.com/jsf/facelets">
5
6     <ui:composition template="../WEB-INF/client-template.xhtml">
7         <ui:define name="content">
8
9             <ui:include src="../includes/services-include.xhtml" />
10
11         </ui:define>
12     </ui:composition>
13
14 </html>
```

Este archivo simplemente incluye la plantilla del cliente (menús y footer) y el archivo *services-include.xhtml*, que contendrá el contenido del archivo para todos los roles.

Aquí podrían surgir dudas, ya que un superadministrador, un administrador y un cliente podrían tener distintas vistas en determinados ficheros, o simplemente tener más opciones disponibles en la vista de la tabla de clases de un determinado servicio, por ejemplo. Esto se gestionará dentro del archivo que se incluye limitando la vista de ciertos elementos a los roles que se autoricen:

```
1         <h:panelGroup rendered="#{servicesController.
2             userRole eq 'ADMIN' or servicesController.
3             userRole eq 'SUPER_ADMIN'}">
4             <div class="col-sm-offset-3 col-md-offset-3 col-
5                 xs-6 col-sm-3 col-md-3 text-right">
6                 <p:commandLink action="#{servicesController.
7                     prepareNewService()} update="
8                     :newServiceForm:newServiceClass"
9                     oncomplete="PF('newServiceDialog').show()"
10                    ">
11                     <span class="btn btn-primary btn-block
12                         btn-main">#{txt['b_new_service']}</
13                         span>
14                 </p:commandLink>
15             </div>
16         </h:panelGroup>
```

Siguiendo con el ejemplo de la vista de los servicios, el código anterior mostrará un botón para la creación de un nuevo servicio solo a los roles "ADMIN" "SUPER_ADMIN", por lo que estará oculto cuando la vista sea destinada a un usuario del centro.

Además, puede llamar la atención el texto a mostrar en el botón: *{txt['b_new_service']}*. Como vimos anteriormente, en el fichero *faces-config.xml* 6.2.1, en el sistema se admiten dos idiomas, español e inglés. Pues bien, aquí está la clave para poder generar la interfaz en ambos idiomas

-y los que se añadan en un futuro-. En la línea 14 del archivo mencionado se establece dónde encontrar los archivos de traducción: *com.booking.language.text*. Mientras que la línea 15, muestra la variable a usar para realizar las traducciones: *var*. Por lo que, cuando en los archivos XHTML encontramos código como el mostrado, el sistema tomará esa variable como una traducción e irá al directorio establecido en búsqueda del archivo del idioma seleccionado por el usuario, donde tomará el texto introducido para la variable en cuestión. En este caso, la variable sería *b_new_service* y el texto equivalente para el idioma español *Nuevo Servicio*, dado por la línea *b_new_service=Nuevo Servicio* del fichero del lenguaje español de la ruta. Veremos estos archivos de idioma en unas líneas (6.2.1).

Observamos en el código que el inicio de algunas etiquetas viene dado por una letra seguida de dos puntos. Esto indica qué tipo de elemento es el que le sigue a los dos puntos. Para ello, primeramente se incluye los paquetes necesarios:

```
1 <ui:component xmlns="http://www.w3.org/1999/xhtml"
2               xmlns:h="http://java.sun.com/jsf/html"
3               xmlns:ui="http://java.sun.com/jsf/facelets"
4               xmlns:f="http://java.sun.com/jsf/core"
5               xmlns:p="http://primefaces.org/ui">
```

Por tanto, las opciones que se utilizan son:

- h: elementos html convencionales.
- ui: elementos del framework Facelets.
- f: elementos propios de JSF.
- p: elementos de PrimeFaces.

Source Packages

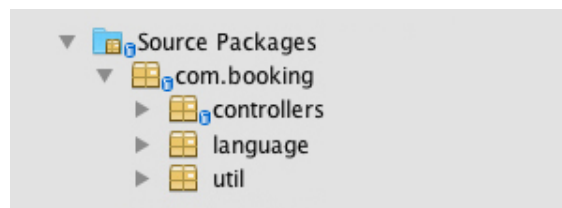


Figura 6.4: Directorio *Source Packages*

controllers Este directorio contiene los archivos Java principales asociados a la interfaz de usuario, los controladores. Cuando hablamos de JSF, el controlador Java asociado a cada página se denomina bean manejado o *managed bean*, como su nomenclatura muestra en el código. Por tanto, cada uno de esos archivos XHTML usará, al menos, un controlador, el cual transmite los datos necesarios a la interfaz, actuando de puente con la gestión de la base de datos a través de EJB.

```

1  import javax.ejb.EJB;
2  import javax.faces.bean.ManagedBean;
3  import javax.faces.bean.ViewScoped;
4  import org.primefaces.context.RequestContext;
5
6  @ManagedBean
7  @ViewScoped
8  public class ServicesController implements Serializable {
9
10     @EJB
11     private ServiceFacade serviceFacade;
12     @EJB
13     private ClassFacade classFacade;
14     @EJB
15     private AuditFacade auditFacade;
16     @EJB
17     private AppointmentFacade appointmentsFacade;
18
19     private List<Service> services;
20     private User loggedUser;
21     private Organisation organisation;
22     private Service selectedService;
23     private String newServiceName;
24     private String newServiceDescription;
25     private boolean isNewService;
26
27     public ServicesController() {
28     }
29
30     @PostConstruct
31     public void init() {
32         loggedUser = FacesUtil.getCurrentUser();
33         organisation = FacesUtil.getCurrentOrganisation();
34
35         isNewService = true;
36         services = serviceFacade.findAllServicesOfOrganisation(
37             organisation);
38     }
39
40     public String activateService(Service service) {
41         serviceFacade.activateService(service);
42         FacesUtil.addSuccessMessage("servicesForm:msg", "El servicio ha
43             sido activado correctamente.");
44
45         try {
46             // Audit service activation
47             String ipAddress = FacesUtil.getRequest().getRemoteAddr();
48             auditFacade.createAudit(AuditType.ACTIVAR_SERVICIO,
49                 loggedUser, ipAddress, service.getId(), organisation);
50         } catch (Exception e) {
51             Logger.getLogger(ServicesController.class.getName()).log(
52                 Level.SEVERE, null, e);
53         }
54     }
55 }

```

```

49         }
50
51         return "services.xhtml" + Constants.FACES_REDIRECT;
52     }

```

Podemos observar que este bean Java es un bean manejado de JSF por su especificación en la línea 6 de código. Y, justo en la siguiente línea, se especifica el alcance del mismo (scope). Hay distintas opciones de scope para los beans:

- **ApplicationScoped**: La información de este bean se guarda durante toda la vida de la aplicación web, desde que se ejecuta por primera vez hasta que se elimina del servidor.
- **SessionScoped**: Se puede intuir por el nombre que son beans de sesión, es decir, la información se mantiene desde que un usuario comienza una sesión en la aplicación hasta que esta acaba.
- **ViewScoped**: La información perdura el tiempo de vista de una página, o sea, desde que el usuario accede a la misma hasta que se navega a una distinta. Disponible desde JSF 2.0.
- **RequestScoped**: La vida del bean empieza cuando se produce una petición al servidor y acaba cuando el usuario recibe la respuesta con la información pedida. Por tanto, se creará una instancia del bean en cada petición al servidor.
- **NoneScoped**: Este bean se instancia cuando es invocado por otro bean, siendo eliminado cuando la necesidad acabe.
- **CustomScoped**: Desde JSF 2.0 también es posible la creación de scopes personalizados, donde se podrá configurar el tiempo del mismo.

Vemos también el uso de clases EJB en las líneas 10-17, clases que se encargarán de la consulta o edición de la información de nuestra base de datos. Por tanto, los controladores harán uso de las funciones de estas clases cada vez que se requiera hacer uso de la BD, como en la línea 36, donde se realiza una consulta de todos los servicios de la empresa, o en la 40, que se invoca a la función encargada de activar un servicio que estaba suspendido.

language Como su nombre indica, este directorio contendrá los archivos de los distintos lenguajes que el sistema soporta. Existirá un archivo de extensión *.properties* por cada lenguaje. Como se vio en 6.2.1, se establece un nombre de archivo base para los ficheros de texto:

```

1      <base-name>com.booking.language.text</base-name>

```

Así, existirá un archivo con el nombre base *text.properties* que el sistema consultará en caso de no encontrar la traducción en el archivo de idioma específico o algún otro error similar. En esta aplicación, por tanto, tendremos tres archivos en la carpeta: *text.properties*, *text_en.properties* y *text_es.properties*, donde los dos últimos serían los archivos de traducción para inglés y español, donde observamos que toman el nombre base especificado añadiendo el código del idioma que representan. En la figura 6.5 vemos un ejemplo de traducción para las mismas palabras, a las cuales se accederán en las vistas XHTML mediante la variable *var* como vimos anteriormente.

```
# login page
login=Login
email=Email
introduce_email=Introduce your email address
password=Password
introduce_password=Introduce password
forgot_password=Forgot your password?
```

```
login=Iniciar Sesión
email=Email
introduce_email=Introduce tu dirección email
password=Contraseña
introduce_password=Introduce tu contraseña
forgot_password=¿Has olvidado tu contraseña?
```

Figura 6.5: Directorio *Comparativa de Lenguajes*

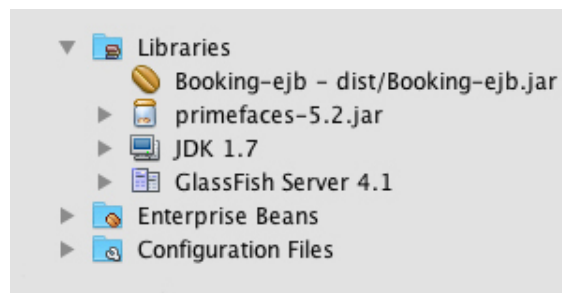


Figura 6.6: Directorios *Libraries*, *Enterprise Beans* y *Configuration Files*

util El directorio util se utiliza para almacenar todas aquellas clases que se han creado pensando en ser una ayuda en cualquier parte de la aplicación, como por ejemplo, *DateService*, mediante la cual se realizan gestiones de fechas (como devolver una fecha con hora 00:01, usada para búsquedas), *Constant*, que incluye algunas variables constantes usadas en toda la aplicación, o *FacesUtil*, que es la clase más usada de este directorio, mediante la cual podemos acceder a diversas funciones, como obtener o establecer atributos de sesión, obtener la dirección IP en uso, añadir mensajes en la vista actual, saber quién es el usuario haciendo uso de la aplicación o realizar redirecciones.

Libraries, Enterprise Beans y Configuration Files

Libraries Como su nombre indica, contiene todas las librerías/dependencias que este módulo usa, incluyendo el módulo *Booking-ejb*. JDK (Java), PrimeFaces y GlassFish completarán la lista.

Enterprise Beans Es simplemente una carpeta que contiene todas las clases EJB que utiliza el módulo *Booking-war*, es decir, todas las *Facades* de las que hace uso para acceso a la información de la BD.

Configuration Files Posee los archivos de configuración de este módulo, como los ya vistos *faces-config.xml* y *web.xml* u otros destinados al servidor (*glassfish-web.xml*).

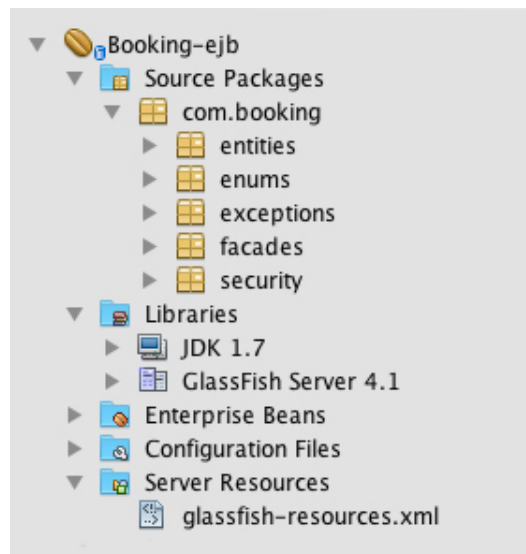


Figura 6.7: Directorios módulo *Booking-ejb*

6.2.2. Módulo EJB

Este módulo se centra en la gestión de las clases necesarias para la solución buscada para el proyecto, las cuales hemos visto en el diagrama conceptual de clases UML 4.1, así como su mapeo respecto a las clases creadas en la base de datos y la gestión de todo ello. La estructura del mismo es similar a la del módulo que acabamos de ver:

Source Packages

Este sería el directorio principal del módulo, donde se establecen los paquetes del mismo. Así, tendremos los siguientes paquetes que vamos a ir describiendo:

- **entities**: El cual contiene todas las clases utilizadas en el sistema, incluyendo las que surgen de las relaciones entre las clases del modelo conceptual UML, como puede ser el case de *Booking*, una clase que aparece de la relación entre las clases *ActivityClass* (Clase) y *User* (Usuario), conteniendo la instancia de ambos cada vez que se realiza una reserva de la clase de un servicio (entrenamiento funcional, TRX, pilates, etc.). Veremos un ejemplo de entidad en la sección 6.3.1.
- **enums**: Directorio con las clases *enum* del sistema, como los estados, tipos de notificaciones o roles.
- **exceptions**: Conjunto de excepciones creadas para el manejo del programa.
- **facades**: Este directorio, junto con *entities*, toma el protagonismo de los paquetes del módulo. Contiene todas las clases que realizan la gestión de los datos, tanto funciones CRUD (Crear, Leer, Actualizar y Borrar) como consultas. Por tanto, a cada una de las entidades creadas le corresponderá una clase *Facade* para su gestión en la base de datos, como veremos a continuación en la sección 6.3.2.

- **security**: Contiene el archivo de codificación de contraseñas con el algoritmo a usar en el momento de almacenar la misma en la BD y las funciones para las verificaciones oportunas en el inicio de sesión de los usuarios.

Libraries, Enterprise Beans, Configuration Files y Server Resources

El resto de directorios que componen el módulo EJB son los siguientes:

Libraries Librerías/dependencias que este módulo usa. En este caso, JDK (Java) y GlassFish.

Enterprise Beans Es simplemente una carpeta que contiene todas las *Facades* EJB generadas.

Configuration Files Posee archivos de configuración de este módulo, como por ejemplo *persistence.xml*, archivo de persistencia JDBC (framework usado por JPA).

Server Resources Podemos destacar el único fichero que posee este directorio, *glassfish-resources.xml*, a través del cual este módulo, y por tanto la aplicación, configura la conexión con el servidor GlassFish, indicando el puerto de conexión, la base de datos utilizada, datos de acceso, etc., como vemos en su código.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE resources PUBLIC "-//GlassFish.org//DTD GlassFish Application
   Server 3.1 Resource Definitions//EN" "http://glassfish.org/dtds/
   glassfish-resources_1_5.dtd">
3 <resources>
4   <jdbc-connection-pool allow-non-component-callers="false" associate-
     with-thread="false" connection-creation-retry-attempts="0"
     connection-creation-retry-interval-in-seconds="10" connection-
     leak-reclaim="false" connection-leak-timeout-in-seconds="0"
     connection-validation-method="auto-commit" datasource-classname="
     org.postgresql.ds.PGSimpleDataSource" fail-all-connections="false"
     " idle-timeout-in-seconds="300" is-connection-validation-required
     ="false" is-isolation-level-guaranteed="true" lazy-connection-
     association="false" lazy-connection-enlistment="false" match-
     connections="false" max-connection-usage-count="0" max-pool-size=
     "32" max-wait-time-in-millis="60000" name="postgre-
     sql_bookingPool" non-transactional-connections="false" pool-
     resize-quantity="2" res-type="javax.sql.DataSource" statement-
     timeout-in-seconds="-1" steady-pool-size="8" validate-atmost-once-
     -period-in-seconds="0" wrap-jdbc-objects="false">
5     <property name="serverName" value="localhost"/>
6     <property name="portNumber" value="5432"/>
7     <property name="databaseName" value="booking"/>
8     <property name="User" value="booking"/>
9     <property name="Password" value="g903e0B_R2tw"/>
10    <property name="URL" value="jdbc:postgresql://localhost:5432/
       booking"/>
11    <property name="driverClass" value="org.postgresql.Driver"/>

```

```

12     </jdbc-connection-pool>
13     <jdbc-resource enabled="true" jndi-name="jdbc/booking" object-type="
        user" pool-name="postgre-sql_bookingPool"/>
14 </resources>

```

6.3. Gestión de Base de Datos

Java Persistence API (JPA) es el acceso estándar a bases de datos relacionales en Java EE. Provee una forma simple y eficiente de gestionar el ORM (*object/relational mapping*) de objetos Java (*POJO*) respecto a los datos de la BD. Hablamos de las entidades JPA, o nuestras *entities* que acabamos de describir.

Cada entidad se asocia (aunque no en el 100% de los casos) a una tabla relacional de la BD, por lo que cada instancia de una entidad quedará representada por una fila de esa tabla. Estas entidades establecen diferentes relaciones entre ellas: *one-to-one*, *one-to-many*, *many-to-one* o *many-to-many*. Por tanto, las aplicaciones Java que gestionan estas entidades se ven en la necesidad de acceder y navegar por las instancias y sus relaciones. Y esta necesidad se satisface con JPQL (*Java Persistence Query Language*).

Vayamos por partes; centrándonos en nuestro código, hemos comprobado que el módulo EJB contiene los ficheros responsables de la gestión de base de datos. De acuerdo a lo afirmado en los párrafos iniciales de esta sección, podemos centrarnos en los paquetes *com.booking.entities* y *com.booking.facades* del mismo.

6.3.1. *Entities*

Empecemos con el ejemplo de la clase de java *Service*.

```

1  @Entity
2  @Table(name = "services")
3  public class Service implements Serializable {
4
5      private static final long serialVersionUID = 1L;
6
7      @Id
8      @Basic(optional = false)
9      @GeneratedValue(strategy = GenerationType.IDENTITY)
10     @Column(name = "id")
11     private long id;
12     @Column(name = "name")
13     private String name;
14     @Column(name = "description")
15     private String description;
16     @ManyToOne(fetch = FetchType.LAZY)
17     @JoinColumn(name = "organisation", referencedColumnName = "id")
18     private Organisation organisation;
19     @Column(name = "status")
20     @Enumerated(EnumType.STRING)

```



```

21     private Status status;
22     @Column(name = "created_date")
23     @Temporal(TemporalType.TIMESTAMP)
24     private Date createdDate;

```

Observamos cómo se establece la correspondencia de las entidades Java con las clases de la base de datos (ORM). De esta manera, la elección y uso del SGBD será independiente a nuestro código, simplemente habría que crear las clases y atributos con los nombres que definimos en las entidades. Así, para la clase *Service* tendremos una tabla *services* en la BD, con los atributos *id*, *name*, *description*, *organisation*, *status* y *created_date*, correspondientes a los atributos *id*, *name*, *description*, *organisation*, *status* y *createdDate* de nuestra clase Java *Service*.

Puede llamar la atención la estrategia de generación del atributo *id*: *@GeneratedValue(strategy = GenerationType.IDENTITY)*. Estos se crearán automáticamente en la BD, siguiendo un orden estándar, desde el número 1, por lo que no tendremos que gestionar los identificadores de las instancias de las clases.

Además, hemos observado que aparece una organización en el sistema. Puede parecer algo redundante al tratarse de una aplicación web para un centro de entrenamiento y empresa específicos. Aun siendo esto cierto, la programación del sistema se ha realizado pensando en la escalabilidad y teniendo en cuenta la posibilidad de que, en un futuro, otro centro similar puede hacer uso de la misma, incluso compartiendo el mismo servidor y la misma base de datos. De ahí que se gestione toda la información haciendo distinción de la organización a la que pertenece, así se podrán añadir otras empresas con pequeñas adaptaciones en el sistema, teniendo cada una de ellas sus propias características, personalizando el logo o el estilo de la interfaz de usuario.

6.3.2. *Facades*

Como se ha afirmado antes, a cada entidad le corresponderá un archivo *Facade* para las funciones pertinentes.

```

1  */
2  @Stateless
3  public class ServiceFacade extends AbstractFacade<Service> {
4
5      @PersistenceContext(unitName = "Booking-ejbPU")
6      private EntityManager em;
7
8      @Override
9      protected EntityManager getEntityManager() {
10         return em;
11     }
12
13     public ServiceFacade() {
14         super(Service.class);
15     }
16
17     public Service createNewService(String name, String description,
18         Organisation organisation) throws ServiceAlreadyExistsException {

```

```

19         if (findServiceByName(name, organisation) != null) {
20             throw new ServiceAlreadyExistsException("Lo sentimos, no ha
                sido posible crear el nuevo servicio: El nombre ya existe
                .");
21         }
22
23         Service service = new Service();
24         service.setName(name);
25         service.setDescription(description);
26         service.setOrganisation(organisation);
27         service.setCreatedDate(new Date());
28         service.setStatus(Status.ACTIVATED);
29         create(service);
30
31         return service;

```

Así, la "fachada" *ServiceFacada* contiene los métodos *createNewService*, *updateService*, *activateService* y *deactivateService* que los controladores (beans manejados) usarán para el acceso a la BD.

Aparte de estas funciones CRUD, también será la clase encargada de realizar las consultas SQL de cada entidad a través de JPQL, como vemos en los siguientes ejemplos:

```

1
2     public List<Service> findAllActiveServicesOfOrganisation(
3         Organisation organisation) {
4         return em.createQuery("SELECT s FROM Service s WHERE s.
            organisation = :organisation AND s.status = :statusActive
            ORDER BY s.name ASC").
5             setParameter("statusActive", Status.ACTIVATED).
            setParameter("organisation", organisation).getResultList
            ();

```

```

1
2     public Service findServiceByName(String serviceName, Organisation
3         organisation) {
4         return findUniqueResult(em.createQuery("SELECT s FROM Service s
            WHERE s.organisation = :organisation AND s.name =
            :serviceName ORDER BY s.name ASC").
5             setParameter("organisation", organisation).
            setParameter("serviceName", serviceName).getResultList()
            );

```

Gracias al uso de JPQL y al mapeo realizado, en la nomenclatura de las consultas se usa las clases y atributos Java, y no el nombre correspondiente de la base de datos. Este método de consulta facilita tanto la realización de las mismas por parte del programador como, de nuevo, la independencia del SGBD elegido o los cambios que puedan hacerse en el mismo.

6.3.3. Sistema de Gestión de Base de Datos (SGBD)

Como ya se ha informado, el SGBD elegido para llevar a cabo este proyecto ha sido *PostgreSQL*. *PostgreSQL*, o simplemente Postgres, es un sistema de gestión de bases de datos relacional orientado a objetos dirigido por una comunidad de desarrolladores que la gestionan de forma libre y altruista o mediante organizaciones comerciales. Está considerado el SGBD de código abierto más potente del mercado. *pgAdmin* es la herramienta oficial para administrar las bases de datos en PostgreSQL, y la que se ha usado en este caso.

Las características y ventajas del uso de PostgreSQL son las siguientes:

- Ahorros considerables de costos de operación: Diseñado con las características, estabilidad y rendimiento de grandes proveedores comerciales con un menor mantenimiento.
- Estabilidad y confiabilidad.
- Extensible: Debido a la disponibilidad de su código fuente aquel que lo requiera podría extender o personalizar el programa de acuerdo a sus necesidades.
- Multiplataforma, incluyendo Linux, Windows, Unix, Solaris y MacOS X.
- Estrategia de almacenamiento MVCC (Control de Concurrencias Multiversión): Consigue una mejor respuesta en grandes volúmenes de información, además de permitir acceso de solo lectura durante la edición de registros, dando la opción de realizar copias de seguridad en caliente.
- Herramientas gráficas de diseño y administración de bases de datos.
- Soporta tanto los tipos de datos, cláusulas, funciones y comandos estándar SQL92/SQL99 como los extendidos por el propio PostgreSQL.
- Buen sistema de seguridad.
- Gran capacidad de almacenamiento.
- Buena escalabilidad, soportando mayor cantidad de peticiones simultáneas a BD ajustándose a la CPU y cantidad de memoria disponible de forma óptima.
- Soporta claves ajenas (foreign keys), disparadores (triggers), vistas, integridad transaccional, herencia de tablas, tipos de datos y operaciones geométricas, transacciones distribuidas, afirmaciones(assertions), etc.

Volviendo a nuestro sistema, un ejemplo de script de creación de una tabla de la BD sería:

```
1 CREATE TABLE schema_booking.services
2 (
3     id serial NOT NULL,
4     created_date timestamp without time zone,
5     description character varying(255),
6     name character varying(255),
7     status character varying(255),
8     organisation bigint,
9     CONSTRAINT services_pkey PRIMARY KEY (id),
```

```

10     CONSTRAINT fk_services_organisation FOREIGN KEY (organisation)
11         REFERENCES schema_booking.organisations (id) MATCH SIMPLE
12         ON UPDATE NO ACTION ON DELETE NO ACTION
13 )
14 WITH (
15     OIDS=FALSE
16 );
17 ALTER TABLE schema_booking.services
18     OWNER TO booking;

```

Y la representación de dos instancias de la entidad Service en la tabla services:

	id [PK] serial	created_date timestamp without time zone	description character varying(255)	name character varying(255)	status character varying(255)	organisation bigint
1	1	2015-09-28 17:01:02.73	Entrenamiento Funcional en Suspensión	TRX	ACTIVATED	1
2	3	2015-09-28 17:03:36.149	Saco Búlgaro	Bulgarian Bag	ACTIVATED	1

Figura 6.8: Instancias de *Service* en la tabla *services*

Parte III

Epílogo

En esta última parte quedarán recogidas las conclusiones y los manuales necesarios para el manejo de la aplicación resultado del desarrollo. Si se ha realizado algún tipo de evaluación de la solución proporcionada, más allá de las pruebas del sistema, también deberá venir recogida en un capítulo separado dentro de esta parte. Pueden consultarse diversos tipos de evaluaciones sobre sistemas de información en [?]: casos de estudio, análisis estático, análisis dinámico, simulación, experimento controlado, etc.

Bibliografía

