ESCUELA SUPERIOR DE INGENIERÍA

INGENIERÍA INFORMÁTICA

GESTIÓN DE CENTRO DE MEJORA DEL RENDIMIENTO Y LA SALUD

Jesús Soriano Candón

12 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, 12 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 aplicacion web que permita la gestion de las actividades de CoreSport, un centro de mejora del rendimiento y la salud, así como de los usuarios del mismo. En concreto, se desarrollara la pagina web pública de la empresa, así como el area de clientes, donde cada usuario tendra acceso a la gestion de actividades, datos personales, mensajería interna, dietas, entrenamientos, etc. A su vez, los administradores del sitio accederán a una gestion mas 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 (JavaServer 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**

5. Diseño del Sistema **23**
5.1. Arquitectura del Sistema . . . 23
5.1.1. Arquitectura Física . . . 23
5.1.2. Arquitectura Lógica . . . 24
5.2. Diseño Físico de Datos . . . 27
5.3. Diseño de la Interfaz de Usuario . . . 27

6. Construcción del Sistema **33**
6.1. Entorno de Construcción . . . 33
6.1.1. Entorno para la Web Pública . . . 35
6.2. Código Fuente . . . 35
6.2.1. Módulo Web . . . 35
6.2.2. Módulo EJB . . . 44
6.3. Gestión de Base de Datos . . . 46
6.3.1. *Entities* . . . 46
6.3.2. *Facades* . . . 47
6.3.3. Sistema de Gestión de Base de Datos (SGBD) . . . 49

7. Pruebas del Sistema **51**
7.1. Entorno de Pruebas . . . 51
7.2. Roles . . . 51
7.3. Niveles de Pruebas . . . 51
7.3.1. Pruebas Unitarias . . . 52
7.3.2. Pruebas de Integración . . . 52
7.3.3. Pruebas de Sistema . . . 52
7.3.4. Pruebas de Aceptación . . . 54

III Epílogo **55**

8. Manual de implantación y explotación **59**
8.1. Introducción . . . 59
8.2. Requisitos previos . . . 59
8.3. Inventario de componentes . . . 60
8.4. Procedimientos de instalación . . . 60
8.5. Pruebas de implantación . . . 63
8.6. Procedimientos de operación y nivel de servicio . . . 64
8.7. Introducción . . . 64
8.8. Características . . . 64
8.9. Requisitos previos . . . 65
8.10. Uso del sistema . . . 65

9. Conclusiones **67**
9.1. Objetivos alcanzados . . . 67
9.2. Lecciones aprendidas . . . 67
9.3. Trabajo futuro . . . 68

GNU General Public License **71**

Índice de figuras

5.1. Representación de Arquitectura de 3 Capas 24
5.2. Proceso del Patrón MVC 25
5.3. Comparativa entre modelo MVC y arquitectura de 3 capas 26
5.4. Interfaz de usuario: Inicio de sesión 27
5.5. Interfaz de usuario: Registro 28
5.6. Interfaz de usuario: Pantalla general 28
5.7. Interfaz de usuario: Pantalla con tabla de datos 29
5.8. Interfaz de usuario para dispositivos móviles: Inicio de sesión 29
5.9. Interfaz de usuario para dispositivos móviles: Pantalla general 30
5.10. Interfaz de usuario para dispositivos móviles: Menú desplegable 30
5.11. Interfaz de usuario: Diagrama de navegación 31

6.1. Arquitectura de 3 Capas con Frameworks 34
6.2. Estructura de los ficheros 36
6.3. Directorio *Web Pages* 36
6.4. Directorio *Source Packages* 40
6.5. Directorio *Comparativa de Lenguajes* 43
6.6. Directorios *Libraries, Enterprise Beans* y *Configuration Files* 43
6.7. Directorios módulo *Booking-ejb* 44
6.8. Instancias de *Service* en la tabla *services* 50

8.1. *Creación de un nuevo proyecto en NetBeans* 61
8.2. *Creación del proyecto: Ubicación* 62
8.3. *Definición de los módulos del proyecto* 62
8.4. *Estableciendo el context root del archivo application.xml* 62

Índice de cuadros

Parte I

Prolegómeno

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, saco búlgaro... y otras actividades propias de un centro de estas características.
Este Proyecto Fin de Carrera (PFC) 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

[1] Para 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.
- Saco Búlgaro (Bulgarian Bag): 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 divido 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 multilenguaje, 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.
- **Git** 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 **??**, 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.
  - Gesión 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.
- Siguientes 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 relaciones 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 por 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

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 Aspire 5732z 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.

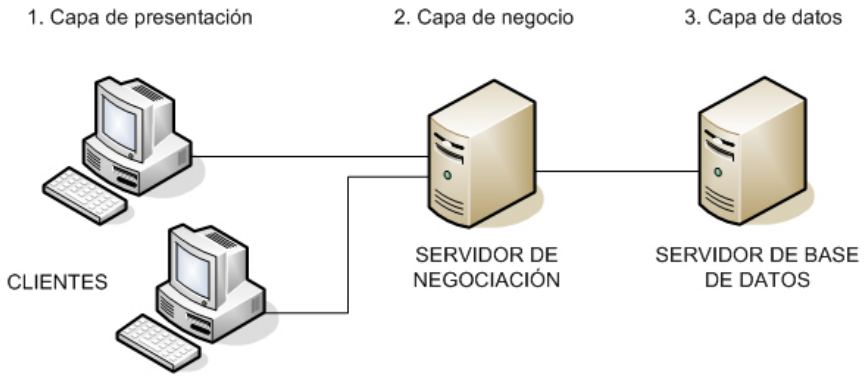

Figura 5.1: Representación de Arquitectura de 3 Capas

La programación por capas es un modelo de desarrollo software en el que el objetivo primordial es la separación (desacomplamiento) 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 mismos 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 ([**?**]).

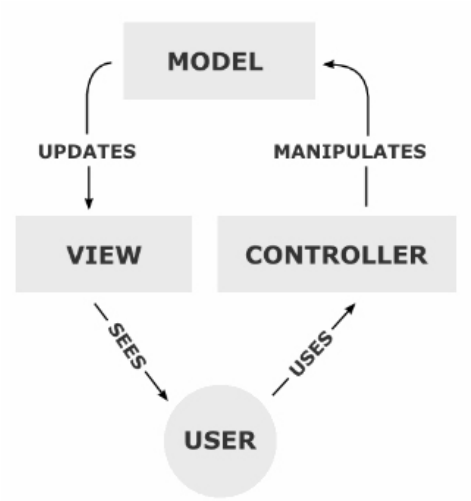

Figura 5.2: Proceso del Patrón MVC

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.

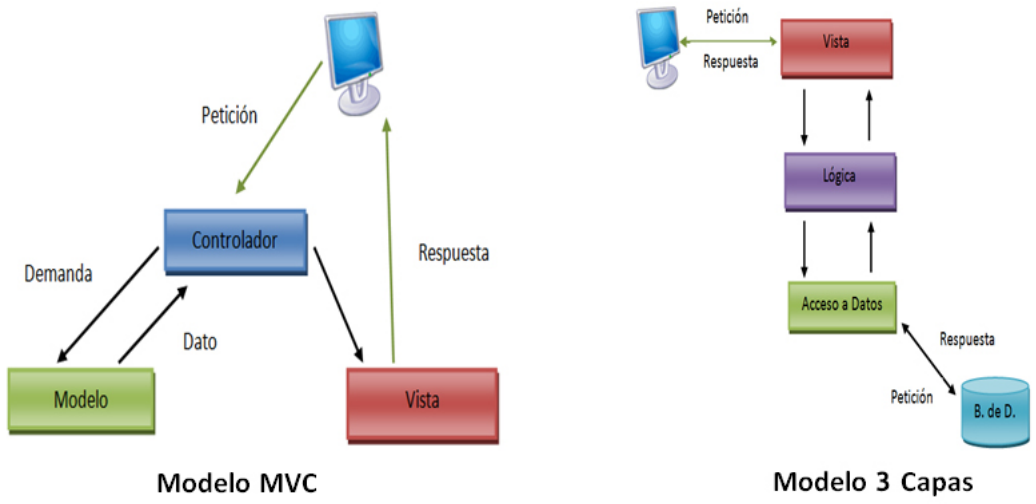

Figura 5.3: Comparativa entre modelo MVC y arquitectura de 3 capas

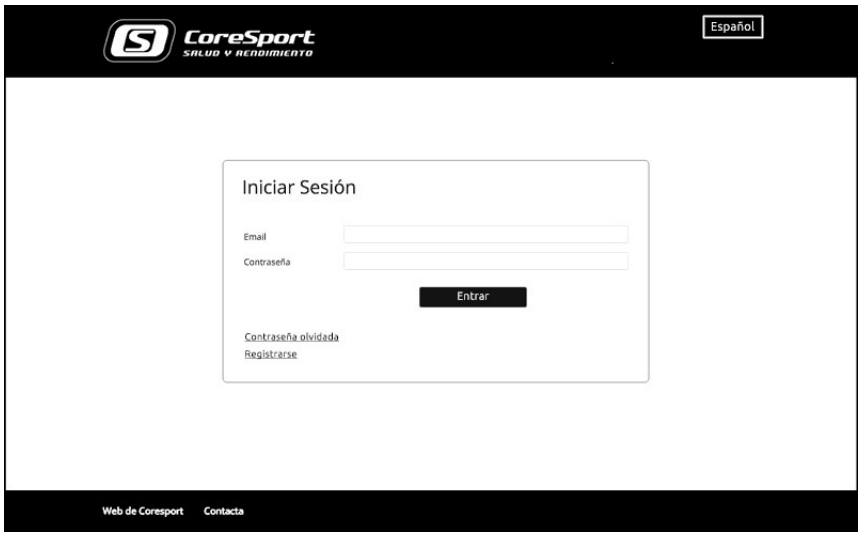

Figura 5.4: Interfaz de usuario: Inicio de sesión

5.2. Diseño Físico de Datos

Habiendo realizado previamente el modelo de conceptual de clases, detallado en la sección **??**, 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.).

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.

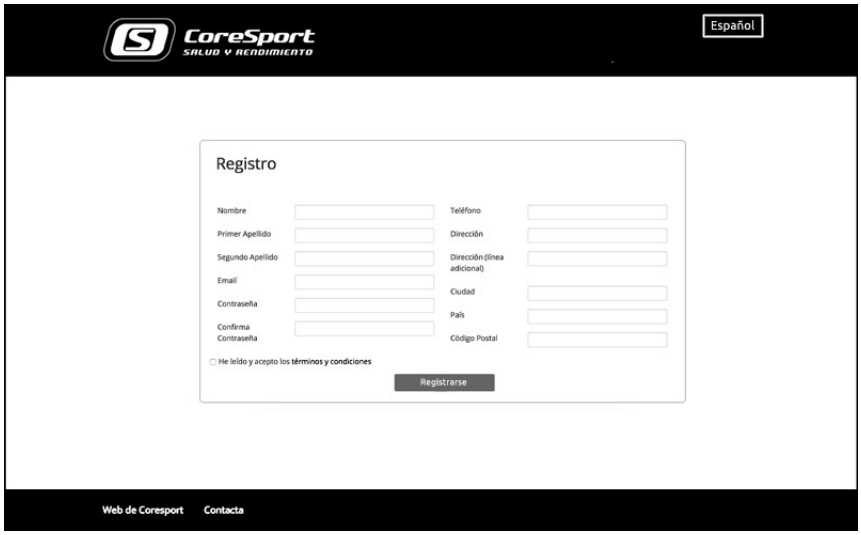

Figura 5.5: Interfaz de usuario: Registro

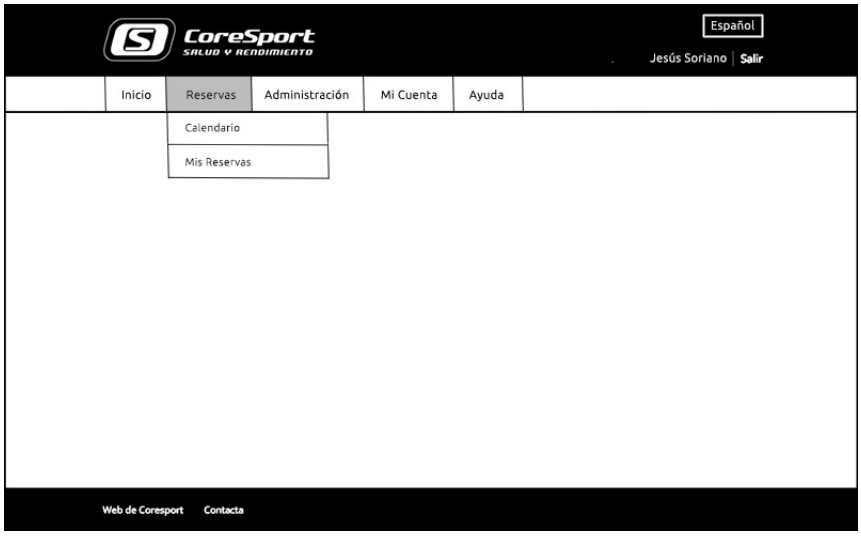

Figura 5.6: Interfaz de usuario: Pantalla general

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

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

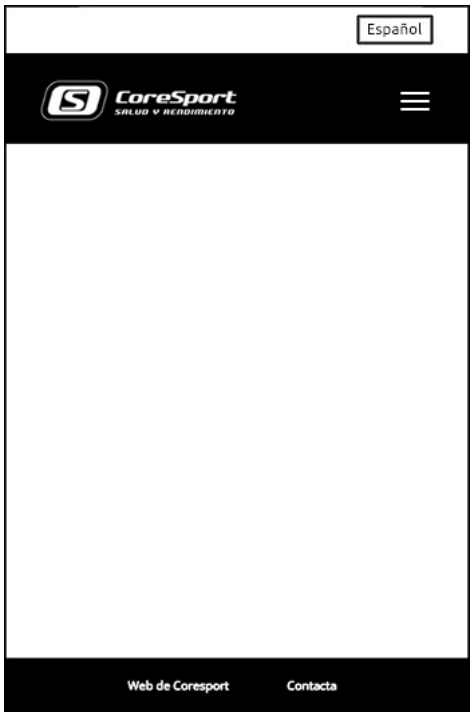

Figura 5.9: Interfaz de usuario para dispositivos móviles: Pantalla general

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.

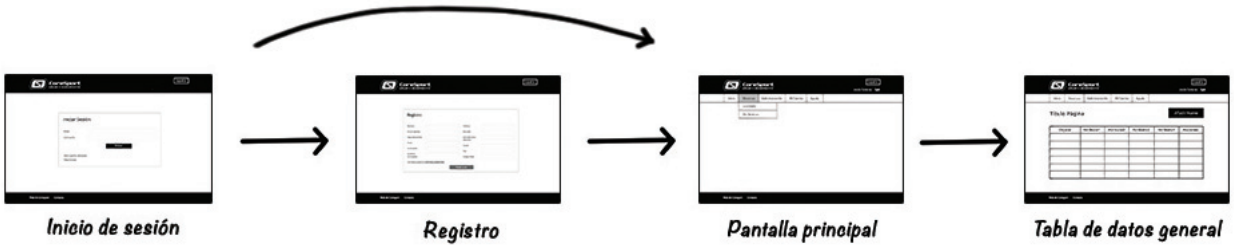

Figura 5.11: Interfaz de usuario: Diagrama de navegación

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.

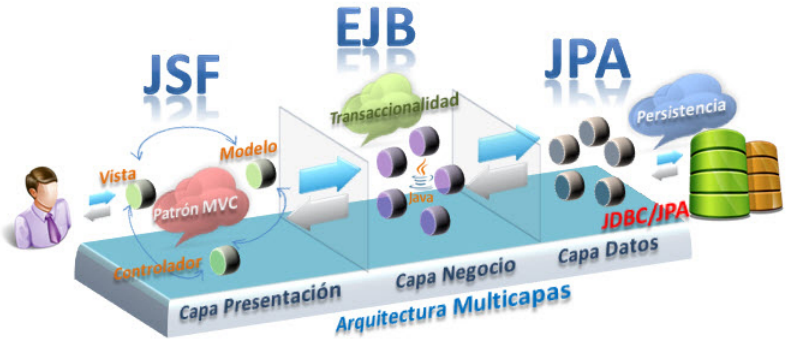

Figura 6.1: Arquitectura de 3 Capas con Frameworks

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

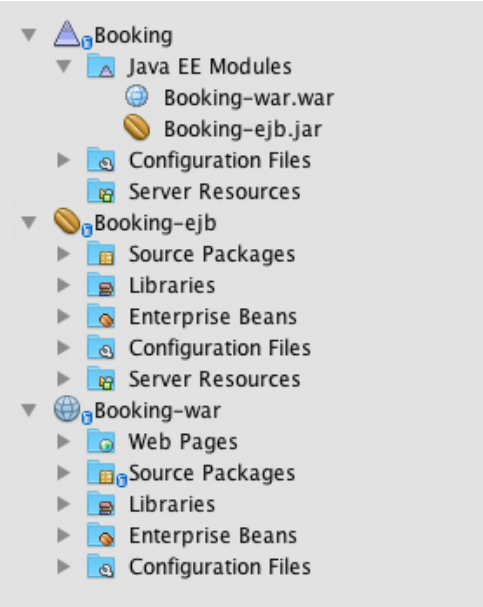

Figura 6.2: Estructura de los ficheros

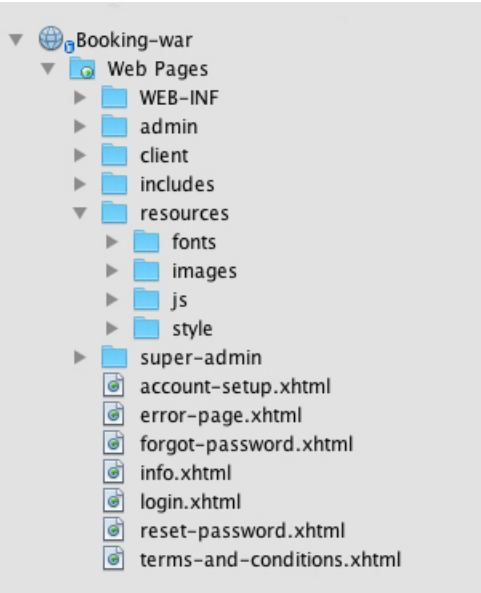

Figura 6.3: Directorio *Web Pages*

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:

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-facesconfig_2_0.xsd"
    version="2.0">
    <application>
        <locale-config>
            <default-locale>es</default-locale>
            <supported-locale>en</supported-locale>
        </locale-config>
        <resource-bundle>
            <base-name>com.booking.language.text</base-name>
            <var>txt</var>
        </resource-bundle>
    </application>
</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.

```
    <security-constraint>
        <display-name>NOT-ALLOWED-ANY</display-name>
        <web-resource-collection>
            <web-resource-name>NOT-ALLOWED-ANY</web-resource-name>
            <description/>
            <url-pattern>/includes/*</url-pattern>
        </web-resource-collection>
        <auth-constraint/>
    </security-constraint>
    <security-constraint>
        <display-name>SUPER_ADMIN</display-name>
        <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:

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
   www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets">

    <ui:composition template="../WEB-INF/client-template.xhtml">
        <ui:define name="content">

            <ui:include src="../includes/services-include.xhtml" />

        </ui:define>
    </ui:composition>

</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:

```
                    <h:panelGroup rendered="#{servicesController.
                       userRole eq 'ADMIN' or servicesController.
                       userRole eq 'SUPER_ADMIN'}">
                        <div class="col-sm-offset-3 col-md-offset-3 col-
                           xs-6 col-sm-3 col-md-3 text-right">
                            <p:commandLink action="#{servicesController.
                               prepareNewService()}" update="
                               :newServiceForm:newServiceClass"
                               oncomplete="PF('newServiceDialog').show()
                               " >
                                <span class="btn btn-primary btn-block
                                   btn-main">#{txt['b_new_service']}</
                                   span>
                            </p:commandLink>
                        </div>
                    </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 "*ADMINz* "*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

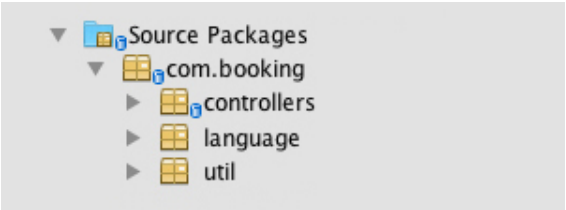

Figura 6.4: Directorio *Source Packages*

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.

```
import javax.ejb.EJB;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ViewScoped;
import org.primefaces.context.RequestContext;

@ManagedBean
@ViewScoped
public class ServicesController implements Serializable {

    @EJB
    private ServiceFacade serviceFacade;
    @EJB
    private ClassFacade classFacade;
    @EJB
    private AuditFacade auditFacade;
    @EJB
    private AppointmentFacade appointmentsFacade;

    private List<Service> services;
    private User loggedUser;
    private Organisation organisation;
    private Service selectedService;
    private String newServiceName;
    private String newServiceDescription;
    private boolean isNewService;

    public ServicesController() {
    }

    @PostConstruct
    public void init() {
        loggedUser = FacesUtil.getCurrentUser();
        organisation = FacesUtil.getCurrentOrganisation();

        isNewService = true;
        services = serviceFacade.findAllServicesOfOrganisation(
            organisation);
    }

    public String activateService(Service service) {
        serviceFacade.activateService(service);
        FacesUtil.addSuccessMessage("servicesForm:msg", "El servicio ha
            sido activado correctamente.");

        try {
            // Audit service activation
            String ipAddress = FacesUtil.getRequest().getRemoteAddr();
            auditFacade.createAudit(AuditType.ACTIVAR_SERVICIO,
                loggedUser, ipAddress, service.getId(), organisation);
        } catch (Exception e) {
            Logger.getLogger(ServicesController.class.getName()).log(
                Level.SEVERE, null, e);
```

```
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 extension *.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*

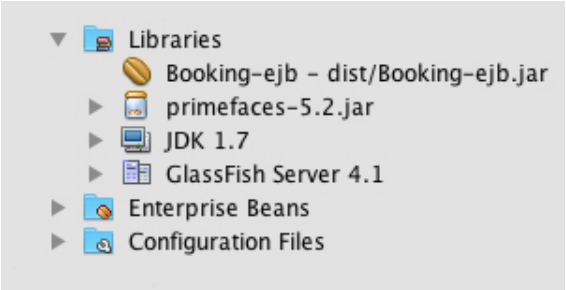

Figura 6.6: Directorios *Libraries, Enterprise Beans* y *Configuration Files*

util El directorio util se utiliza para almacenar todas aquellas clases que se han creado pensadas 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 completarían 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*).

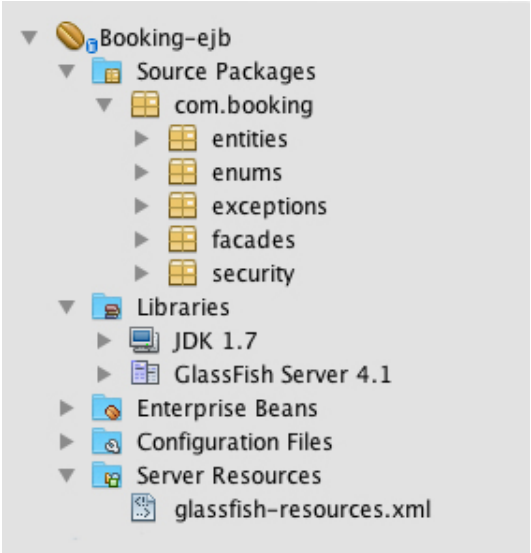

Figura 6.7: Directorios módulo *Booking-ejb*

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 **??**, 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="g9O3e0B_R2tw"/>
10         <property name="URL" value="jdbc:postgresql://localhost:5432/
               booking"/>
11         <property name="driverClass" value="org.postgresql.Driver"/>
```

```
    </jdbc-connection-pool>
    <jdbc-resource enabled="true" jndi-name="jdbc/booking" object-type="
       user" pool-name="postgre-sql_bookingPool"/>
</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 Persistenece 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*.

```
@Entity
@Table(name = "services")
public class Service implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @Basic(optional = false)
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private long id;
    @Column(name = "name")
    private String name;
    @Column(name = "description")
    private String description;
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "organisation", referencedColumnName = "id")
    private Organisation organisation;
    @Column(name = "status")
    @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 pequeños 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,
            Organisation organisation) throws ServiceAlreadyExistsException {
18
```

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

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

```
  CONSTRAINT fk_services_organisation FOREIGN KEY (organisation)
      REFERENCES schema_booking.organisations (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
ALTER TABLE schema_booking.services
  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*

Capítulo 7

Pruebas del Sistema

En este capítulo se presenta el plan de pruebas del sistema de información, incluyendo los diferentes tipos de pruebas que se han llevado a cabo.

7.1. Entorno de Pruebas

Como se mencionó en la sección *Arquitectura Física* 5.1.1, para la realización de las pruebas de este PFC se utilizarán dos ordenadores portátiles del propio alumno:

- El primero, un MacBook Pro de 15 pulgadas, con procesador Intel Core i7 de 2.2 GHz y 16GB de memoria RAM DDR3, usando, por tanto, Mac OS y todos los elementos software nombrados en la sección *Entorno de Construcción* 6.1.
- El segundo equipo, un Acer Aspire 5732z con procesador Dual Core, 4GB de memoria RAM y disco duro SSD y Windows 7, con los mismos programas y frameworks, los necesarios para la instalación y ejecución del sistema.

7.2. Roles

La mayor parte de las pruebas serán realizadas por el alumno en sus equipos, utilizando perfiles de los diferentes tipos de usuarios posibles en el sistema. Una vez estas hayan sido llevadas a cabo, se procederá a evaluar el sistema con los dos miembros que componen la empresa, para verificar requisitos y usabilidad.

7.3. Niveles de Pruebas

Las pruebas realizadas en el sistema han sido en su totalidad de forma manual.

Durante el desarrollo del mismo se han ido realizando pruebas unitarias para corroborar el funcionamiento de las funciones que se iban desarrollando, además de realizar pruebas de integración cuando se finalizaban módulos completos, como por ejemplo la creación y gestión de citas y su integración en el calendario de actividades, probándose la funcionalidad del entorno de citas y su interacción.

Una vez finalizado cada ciclo de desarrollo, con los objetivos propuestos en cada uno, se realizan también pruebas de sistema, comprobando que las nuevas funcionalidades no han afectado a la funcionalidad del resto del sistema y el funcionamiento del mismo es el adecuado.

En la finalización del sistema completo, se volverán a realizar todas las pruebas mencionadas siguiendo los mismos procesos, como se explica a continuación.

7.3.1. Pruebas Unitarias

Se realizan pruebas unitarias para cada funcionalidad del sistema, poniendo especial atención a los detalles y que cada una de ellas tenga el funcionamiento esperado, como por ejemplo la función de todos los elementos de cada formulario, que se muestren correctamente los mensajes de error o de información, los datos introducidos se comprueban correctamente, etc.

El sistema pasa con éxito las pruebas y se procede a las pruebas de integración.

7.3.2. Pruebas de Integración

Una vez comprobado que el sistema pasa correctamente las pruebas unitarias, se procede a las de integración.

Se comprueba que los conjuntos de funcionalidades o módulos funcionan correctamente e interactúan entre sí de forma esperada. Por ejemplo, se comprobará que todos los datos introducidos por el usuario se guardan correctamente en la BD pasando por las capas correspondientes o que el número de plazas disponibles en una clase se reduce cuando un usuario reserva.

De este modo, se comprueba que la funcionalidad de los módulos es correcta.

7.3.3. Pruebas de Sistema

Una vez el sistema parece funcionar de una forma adecuada y sin errores, se procede a realizar pruebas funcionales y no funcionales, de acuerdo a los requisitos establecidos en el catálogo de requistos 3.4.

Pruebas Funcionales

Con estas pruebas se analiza el buen funcionamiento de la implementación de los flujos normales y alternativos de los distintos casos de uso del sistema. Así, iremos probando cada requisito funcional:

- Se comprueba que la opción de **selección** de idioma está presente y realiza el cambio correctamente.
- Un nuevo usuario puede realizar su **registro** obteniendo acceso al sistema.
- El **acceso al sistema** para usuarios registrados se hace como se espera.
- La **sesión queda cerrada** de forma correcta, siendo obligatorio volver a identificarse para acceder al sistema.

- En la página de inicio se observa las **notificaciones** del usuario.
- Los usuarios tienen acceso a la bandeja de su **correo**.
- Aquí, es posible navegar a la edición de correos y **enviar uno nuevo** adecuadamente.
- La opción de **restablecer contraseña** se prueba y se recibe el correo correctamente, pudiéndose completar la acción.
- Asimismo, una vez identificado, el usuario puede **cambiar su contraseña** de forma adecuada.
- También sus **datos del perfil**, quedando los cambios reflejados en el sistema.
- Las clases disponibles son mostradas de forma correcta y el usuario puede **realizar reservas** en las mismas.
- Así como **cancelar dichas reservas**.
- En cuanto a **citas**, vemos que el sistema muestra también un correcto comportamiento y el usuario puede **solicitarlas**.
- El administrador, por su parte, **responde a las solicitudes de cita** de forma adecuada, llegándole la notificación al usuario.
- El propio usuario puede **cancelarla**, siendo el administrador quien recibe la solicitud en caso de estar aceptada.
- Todas las citas y clases, tanto reservadas como disponibles y pasadas, pueden consultarse en el **calendario de actividades** diponible.
- Otra opción sería **consultar las reservas** del usuario en la página correspondiente, donde se observa que se muestran correctamente.
- Se comprueba que, tanto el perfil de usuario como los de administración, pueden ver todas las acciones realizadas por él mismo o por otros usuarios (solo administradores), mediante la página de **auditorías**.
- Las opciones de administración de usuarios también son testadas, siendo posible **activar/-suspender usuarios, ver y editar sus perfiles, así como activar/suspender y ver los perfiles de otros administradores.**
- El superadministrador, además, puede editar el perfil de los administradores sin ningún tipo de error.
- Respecto a al gestión de servicios, los dos roles de administración pueden **dar de alta, activar, suspender y editar servicios.**
- De la misma forma, la **gestión de clases** se realiza correctamente, estando disponibles las mismas opciones.
- El **alta, activación y suspensión de cita** también se ejecutan con éxito.
- Por último, se comprueba la gestión de **archivos**, donde la **creación y edición** de los mismos parece correcta.

- Tanto el administrador como los usuarios a los que van dirigido son capaces de realizar su **descarga del archivo**.

Pruebas No Funcionales

Respecto a las pruebas de los requisitos no funcionales identificados en la subsección 3.4.2 los resultados han sido:

- **Disponibilidad:** Este requisito dependerá del servidor donde se aloje el producto final. Todavía no se han realizado pruebas de producción, solo de desarrollo. Una vez se contrate un servidor se realizarán las pruebas pertinentes. En principio, un servidor debe proporcionar el servicio adecuado para cumplir este requisito, siendo la aplicación alojada en él accesible 24 horas.
- **Fiabilidad:** Por una parte, se realizan pruebas de testeo para asegurarnos que el sistema no posee ningún error. Por otro, se utilizan sesiones Java para los usuarios y se encriptan las contraseñas para ofrecer más seguridad a la aplicación, quedando guardadas en base de datos de esta manera, con el objetivo de que este requisito se cumpla correctamente.
- **Internacionalizacion:** Se comprueba que la opción de traducción se realiza correctamente, eligiendo el idioma deseado en el desplegable que el sistema muestra en todo momento.
- **Usabilidad:** Se ha desarrollado una interfaz intuitiva de fácil acceso y uso, así como adaptada a distintos dispositivos, cumpliendo con este requisito. Aunque se recomiendo su uso en pantallas medianas o grandes, como tablets u ordenadores, debido al tamaño de algunas tablas de datos y mayor facilidad de uso por el espaciado.
- **Mantenibilidad:** Esta prueba se realizará a lo largo de la vida del sistema. En principio, hará falta poco mantenimiento y la opción de escalabilidad, ya sea para su uso con otras empresas o para añadir nuevas funcionalidades, ha sido tenida en cuenta en el desarrollo del proyecto para facilitarlo.

7.3.4. Pruebas de Aceptación

Una vez todas las pruebas han sido realizadas, se realizan a nivel general con los clientes finales, tanto administradores, que han ido interactuando con el sistema a lo largo de su desarrollo, como usuarios voluntarios del centro. De esta manera, se busca obtener un feedback, tanto en la facilidad de uso como sensaciones de los usuarios.

Estas pruebas de aceptación resultan exitosas. Si bien es cierto que han sido realizadas con una pequeña muestra en entorno de desarrollo. Las mismas se realizarán en entorno de producción con una muestra de testeadores mayor y por un periodo algo más prolongado, antes de su uso definitivo.

Parte III

Epílogo

En esta última parte quedarán recogidas las conclusiones y los manuales necesarios para el manejo de la aplicación resultado del desarrollo.

Capítulo 8

Manual de implantación y explotación

Las instrucciones de instalación y explotación del sistema se detallan a continuación.

8.1. Introducción

El presente software está destinado a la gestión de un centro de mejora de la salud y el rendimiento, en concreto, ha sido una personalización para el centro *CoreSport*, en Chiclana de la Frontera (Cádiz).

Cualquier centro similar que precise de un software de gestión puede hacer uso del mismo con o sin modificaciones, bajo la licencia GNU GPL (Licencia Pública General de GNU) en su versión 3 o superior (9.3).

8.2. Requisitos previos

Ha de diferenciarse dos tipos de usos o instalaciones del sistema:

- Sin modificaciones: Si se pretende hacer uso del software tal y como se entrega, sin necesidad de un desarrollo previo para su modificación o ampliación, habría que hacer uso de un servidor para la instalación de GlassFish o WildFly (JBoss) (recomendado), o cualquier otro servidor de aplicaciones compatible para, posteriormente, realizar la instalación del software. Para ello, bastaría con el archivo *Booking.ear* contenido en el directorio *dir*, el cual se puede cargar directamente en el servidor, junto con el backup de la base de datos con los datos básicos para poder iniciar el sistema. Se añadirá un manual de instalación para entornos de producción en un futuro cercano.

- También sería posible, una vez el sistema se encuentre en producción, utilizar el mismo servidor para varias empresas, lo que facilitaría la instalación -ya estaría hecha- y reduciría el precio del servicio al ser compartido. En este caso, solo habría que añadir la nueva organización y sus elementos para la interfaz (logo, icono, estilo...).

- Instalación para desarrollo: En caso que se requiera una instalación local para desarrollo, la instalación sería diferente. Para ello, cualquier equipo básico con un rendimiento aceptable sería suficiente para la instalación. Respecto al software, haría falta, aparte del código del propio proyecto, la instalación de *JDK, NetBeans, PostgreSQL* y *pgAdmin*, junto los

archivos *jar* de *Primefaces, PostgreSQL* y *Commons IO*, librerías de las que se beneficiará el PFC.

8.3. Inventario de componentes

Al descargar una copia del código fuente del PFC, se obtiene lo siguiente:

- **Código fuente del proyecto**: Código Java de la aplicación web.
- Últimas versiones disponibles -en septiembre de 2017- de los **archivos .jar** necesarios, contenidos en el directorio *lib: Primefaces, PostgreSQL* y *Commons IO*.
- Copia de una **base de datos** básica, con una organización, un superadministrador, un administrador y un cliente. Además de un **archivo explicativo** donde se facilitan los datos de estos usuarios y contraseñas y se explica cómo crear estos datos de forma personalizada.
- **Memoria del proyecto**, tanto en PDF como en LaTeX, para que pueda ser actualizada si se desea.

8.4. Procedimientos de instalación

A continuación se detallarán todos los pasos necesarios para la instalación del sistema en un equipo para su desarrollo o para pruebas locales.

Primeramente se procederá a la descarga de todos los elementos software necesarios:

- *JDK (Java Development Kit)* **??**, que es el software que proporciona las herramientas para el desarrollo de aplicaciones Java.
- *NetBeans*, IDE (Entorno de Desarrollo Integrado) para el desarrollo de aplicaciones. Pensado principalmente para aplicaciones Java.
- *Oracle* también ofrece la posibilidad de descargar NetBeans con JDK8, para que sea más cómodo y rápido **??**.
- Códifo fuente del proyecto, disponible en el repositorio *GitHub* **??** destinado a ello **??**.
- *PostgreSQL* **??**, sistema de gestión de bases de datos a utilizar.
- *pgAdmin* **??**, interfaz de usuario para la administración de bases de datos PostgreSQL.
- Librerías *.jar* indicadas, si no se encuentran en el directorio *lib*, o si existen nuevas versiones. Las librerías serían:
  - *Commons IO*
  - *Primefaces*

Una vez descargado todo lo necesario, se procederá a la instalación de cada uno de ellos, junto a la creación y configuración del proyecto:

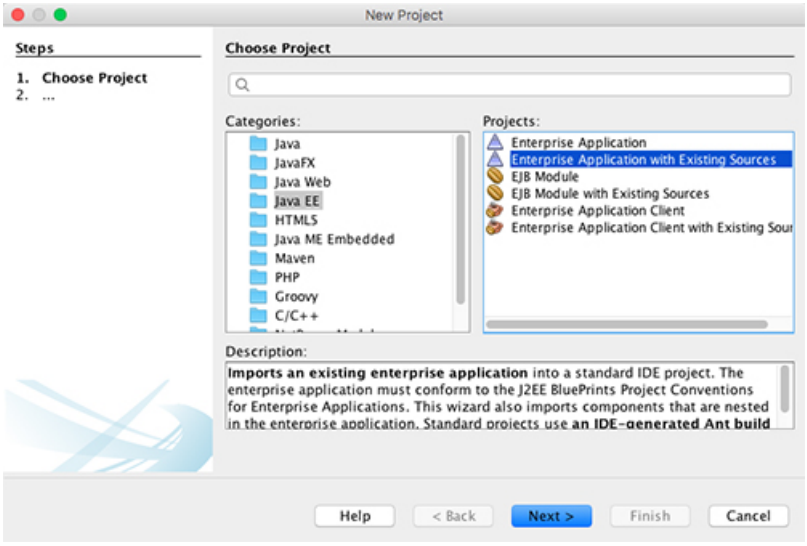

Figura 8.1: *Creación de un nuevo proyecto en NetBeans*

1. Se instalará *NetBeans* con *JDK* -actualmente JDK8-, ya sea en la misma instalación o por separados.

2. Seguidamente, se creará un nuevo proyecto en *NetBeans*, eligiendo la categoría *Java EE* y el tipo de proyecto *Enterprise Application with Existing Sources*, para crear el proyecto a partir del código fuente descargado.

3. En el siguiente paso de la creación del proyecto, se selecciona la ubicación del mismo. Podemos cambiar el nombre y ubicación del proyecto en este paso, así como indicarle a *NetBeans* cuál es el directorio que vamos a utilizar para almacenar las librerías que se van a usar activando la casilla *Use Dedicated Folder for Storing Libraries*, como vemos en la figura 8.1.

4. En el próximo paso elegiremos el servidor de aplicaciones a emplear, en este caso GlassFish, y la versión Java EE, eligiendo en ambas opciones su versión más reciente.

5. Por último en cuanto a la creación del proyecto se refiere, se observa que el módulo Web Booking-war se ha añadido automáticamente. Añadimos también el módulo EJB haciendo clic en la opción para ello y eligiendo el directorio Booking-ejb. Una vez añadido, seleccionamos que se trata del módulo EJB, como vemos en la imagen 8.3.

6. Una vez el proyecto ha sido creado, definiremos dónde se encuentra nuestro módulo web a través del *context root* del archivo *application.xml* que encontraremos en el directorio *Configuration Files* del proyecto. Cambiaremos la fila dedicada a ello, definiendo el *context root* como sigue: *<context-root/>*, de tal forma que le indicaremos al sistema que el módulo se encuentra en el directorio raíz, no haciendo falta ruta alguna para llegar a él.

7. En este momento se puede observar que en el módulo Web existen advertencias de errores, a través de iconos en los directorios donde estos aparecen; se trata de la falta de las librerías de las que el código hace uso. Por lo que seguidamente instalaremos estas dependencias del proyecto, es decir, las librerías *Primefaces* y *Commons IO*. Para ello, simplemente haremos

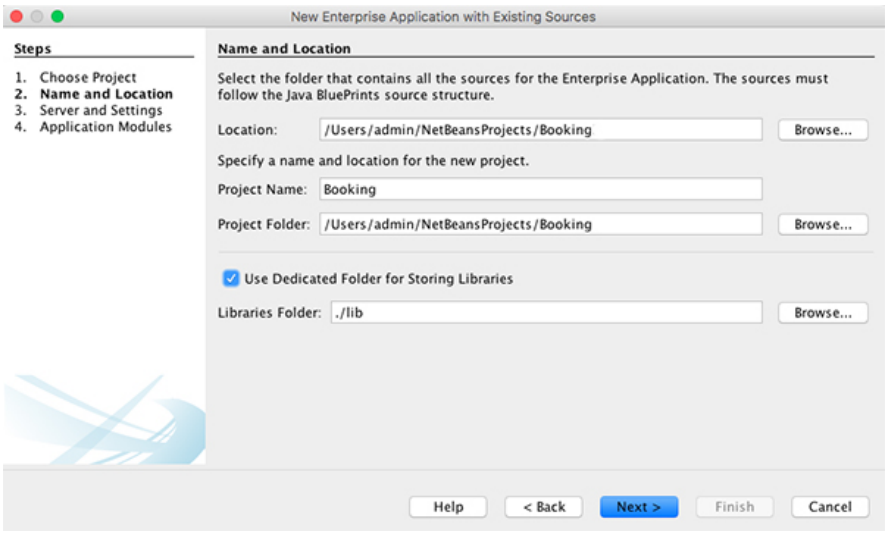

Figura 8.2: *Creación del proyecto: Ubicación*

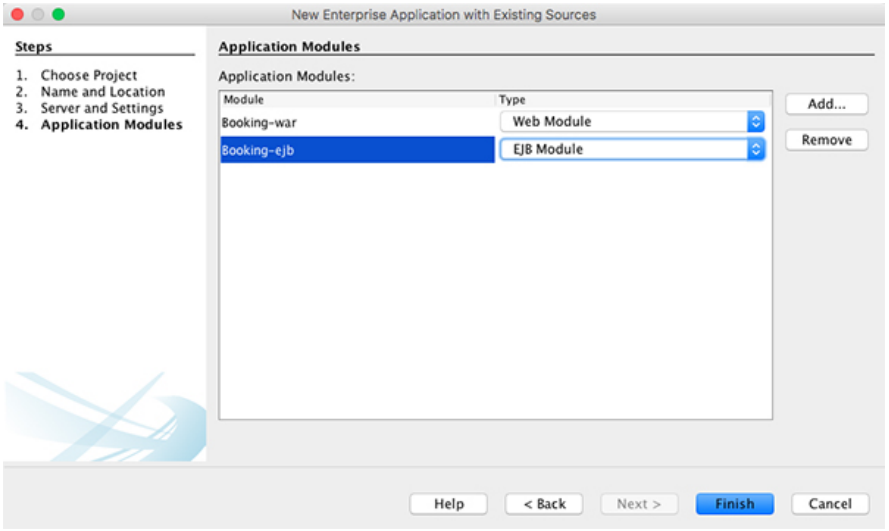

Figura 8.3: *Definición de los módulos del proyecto*

```
<?xml version="1.0" encoding="UTF-8"?>
<application version="6" xmlns="http://java.sun.com/xml/ns/
  <display-name>Booking</display-name>
  <module>
    <ejb>Booking-ejb.jar</ejb>
  </module>
  <module>
    <web>
      <web-uri>Booking-war.war</web-uri>
      <context-root/>
    </web>
  </module>
</application>
```

Figura 8.4: *Estableciendo el context root del archivo application.xml*

clic secundario en el directorio *Libraries* del módulo Web (*Booking-war*) y seleccionamos la opción para añadir un archivo JAR (*Add JAR/Folder...*). Seleccionamos una de las dos librerías y haremos seguidamente el mismo proceso para la otra. Observamos que los iconos de errores desaparecen.

8. Prodeceremos ahora a la instalación de *PostgreSQL* y *pgAdmin* para su uso. Si no se crea un servidor por defecto, crearemos uno usando "*localhost*"para el nombre y el *host* y el resto de campos por defecto, como *5432* para el puerto.

9. Seguidamente crearemos un usuario dándole el nombre de "*booking*".

10. Por último, crearemos nuestra base de datos "*booking*", seleccionando al usuario con mismo nombre como propietario de la misma.

11. A continuación realizaremos la carga de los datos de una base de datos básica. Para ello, haciendo clic secundario en la base de datos *booking*, seleccionamos *Restore...* y utilizamos la copia facilitada en el directorio *db*. En el mismo, podemos consultar los datos de acceso para el sistema, así como un pequeño manual para crear nuevos administradores.

12. Volviendo a *NetBeans*, procedemos a comprobar que todo funciona correctamente compilando el proyecto (clic secundario al proyecto y seleccionamos *Build*) y haciendo el *deploy* (mismo procedimiento, clicando *Deploy*).

13. Una vez finalizado el proceso con éxito, abriremos el navegador web que usemos en nuestro equipo y accederemos a la aplicación mediante la URL *localhost:8080*.

14. Si todo ha ido bien, nos aparecerá la página de inicio de sesión. En caso que se produzca algún error podemos consultarlo en la ventana de *GlassFish* del *Output* de nuestro *NetBeans*. Si se trata de un error que no se logre solucionar, puedes contactar con el autor, Jesús Soriano, a través de su web **??**, del repositorio de *GitHub* **??** o mediante correo electrónico: info@jesussoriano.com.

8.5. Pruebas de implantación

Para comprobar que el sistema funciona correctamente, realizaremos pruebas básicas.

- Para empezar, se procederá con el inicio de sesión de alguno de los usuarios que se facilitan.
- A continuación, podemos realizar el registro de nuevos usuarios.
- Una vez comprobado que esto se realiza correctamente, se puede hacer uso de las funcionalidades del sistema por parte tanto del superadministrador, como administrador y usuario. Como por ejemplo, creación de nuevos servicios o citas, subida de archivos, edición de perfiles, suspensión y activación de servicios y usuarios, reserva de plazas, comprobación de notificaciones, creación de noticias, etc.
- El proceso para realizar dichos ejemplos se puede consultar en el manual de usuario **??**.
- Se recomienda la creación de usuarios reales para la utilización del sistema y no usar los que se facilitan como ejemplo.

8.6. Procedimientos de operación y nivel de servicio

Si se realizan modificaciones o ampliaciones de requisitos del sistema, se recomienda que se trabaje siempre guardando una copia de seguridad tras la finalización de un nuevo componente o modificación de uno existente. En este PFC se ha trabajado usando *GitHub* como repositorio *Git*. También es recomendable guardar un backup de la base de datos cada cierto tiempo, o cuando los datos son sólidos, por si hubiese algún problema con la misma o se requiera una nueva instalación del sistema.

En caso de usarse el sistema en producción, los backups de base de datos serían casi de carácter obligatorio para evitar la pérdida de datos de usuarios reales.

Capítulo 9

Manual de usuario

Las instrucciones de uso del sistema se detallan a continuación.

9.1. Introducción

Este es un sistema de gestión para un centro deportivo, desarrollado concretamente para *CoreSport*, centro para la mejora de la salud y el rendimiento.

La aplicación web será accesible desde cualquier dispositivo con conexión a internet y un navegador web, distinguiéndose 3 tipos de usuarios: superadministrador, administrador y usuario. Los socios de la empresa, y trabajadores si se estima oportuno, serán los administradores, mientras que los usuarios serán los clientes del centro. El rol de superadministrador será llevado a cabo por la persona encargada del sistema, en este caso el propio alumno desarrollador del proyecto.

9.2. Características

Este sistema de gestión proporciona numerosas características que a continuación se detallan:

- Los usuarios, administradores y superadministradores podrán realizar las siguientes acciones:

    - Seleccionar idioma. Registrarse en el sistema[1]. Iniciar y cerrar sesión. Recuperar la contraseña en caso de olvido. Cambiar su contraseña y el resto de sus datos del perfil. Mandar y leer correo interno. Ver notificaciones del sistema. Reservar plaza en una clase y cancelar las reservas. Solicitar citas de algún servicio específico, así como cancelar la solicitud o reserva de cita. Consultar las reservas realizadas. Ver el calendario de actividades con todas las disponibles, las pasadas y las reservadas. Ver el histórico de acciones realizadas en el sistema. Ver los comunicados. Descargarse los archivos a los que tenga acceso.

- Respecto a administradores y superadministardores, además de estas funcionalidades, podrán:

[1]Todo nuevo registro se dará de alta como üsuario", si se tratase de un administrador, será asignado como tal por el superadministrador u otro administrador

- Responder a solicitudes de cita.
- Cancelar la cita de un usuario.
- Activar, suspender y editar usuarios.
- Activar o suspender a otro administrador.
- Ver el histórico de acciones de los usuarios del sistema y otros administradores.
- Dar de alta, editar, suspender o activar servicios.
- Dar de alta, editar, suspender o activar clase.
- Dar de alta, editar, suspender o activar cita.
- Crear nuevo, editar, asignar destinatarios o eliminar archivo.
- Crear nuevo, editar, suspender o activar comunicado.

- El superadministrador, además, podrá:
  - Activar, suspender o editar administradores.

9.3. Requisitos previos

Para la utilización del sistema no se requiere ningún elemento hardware o software fuera de lo común. Cualquier dispositivo con conexión a internet y navegador web puede hacer uso de ella.

9.4. Uso del sistema

El sistema no precisa de conocimiento fuera de lo común en un sistema de gestión. La interfaz es intuitiva y las funcionalidades están estructuradas de manera sencilla a través del menú. Se irá relatando cómo realizar las tareas disponibles, mostrando visualmente algunos ejemplos más representativos.

El primer paso para usar la aplicación web sería el registro de usuario. Para ello,
Describir todos los aspectos necesarios para una utilización efectiva y eficiente del sistema por parte de los usuarios.

Capítulo 10

Conclusiones

En este último capítulo se detallan las lecciones aprendidas tras el desarrollo del presente proyecto y se identifican las posibles oportunidades de mejora sobre el software desarrollado.

10.1. Objetivos alcanzados

Tras la finalización de este Proyecto Fin de Carrera se han alcanzado tanto objetivos previamente definidos como experiencias motivadoras de las que se hablará en la siguiente sección.

En pocas semanas desde el inicio del proyecto se alcanzó unos de los objetivos del mismo, la página web pública de la empresa **??**. Este primer objetivo fue a la vez un aporte de motivación, ya que se obtuvieron los primeros resultados visibles y un pequeño logro personal al finalizar mi segunda página web en activo en ese momento. Desde su realización, la web ha tenido un buen funcionamiento, con resultados visibles para la empresa.

Tras el desarrollo y finalización de la parte principal del proyecto el resultado obtenido es bastante más funcional y motivador. Se consigue exitosamente una solución al problema planteado en la introducción de esta memoria 1.1, obteniendo un software capaz de gestionar la totalidad de servicios del centro, con la opción de añadir actividades, clases, citas, archivos, usuarios, la comunicación entre ellos, seguimiento de acciones, etc.

Se obtiene, por tanto, un sistema que palia todos los objetivos definidos con la aprobación de los clientes finales y el incentivo de haber realizado un completo sistema de gestión que servirá de uso para, al menos, una empresa en expansión con entre 100 y 200 clientes hasta el momento.

10.2. Lecciones aprendidas

La realización de este proyecto me ha aportado muchas cosas interesantes, desde la adquisición de conocimiento respecto a las tecnologías usadas o la mejora en la codificación del software -tanto estructuralmente como en la programación en general- hasta el crecimiento personal a la hora de la resolución de problemas, autoaprendizaje y gestión del tiempo. Aunque no ha sido un camino fácil, claro está.

Desde el primer momento, la realización de este proyecto ha sido todo un reto. Para empezar, la decisión de qué lenguajes usar, frameworks, tecnologías, etc. Afortunadamente, mi experiencia

laboral e inquietud por el aprendizaje me ha facilitado el trabajo en muchos aspectos, ya que anteriormente al inicio del PFC había estado trabajando con aplicaciones webs usando JavaEE y los frameworks usados. Además, toda la parte de web pública, interfaz de usuario y estilos ha coincidido con, puede decirse que, mis inicios en el aprendizaje de diseño web más formalmente, variante en la que estoy centrando mi carrera profesional actualmente.

Aunque la planificación temporal de toda la realización del PFC se estimara para 8 meses, diversas circunstancias han influido en que el espacio temporal se haya alargado hasta los 33 meses, cuatro veces más de lo estimado. Esto no quiere decir que los requisitos hayan crecido en número o dificultad, o que la programación se haya complicado más de lo estimado, sino que, principalmente, el atraso se ha debido a compaginar el desarrollo del proyecto con diversos trabajos a tiempo parcial, desde programador Java hasta diseñador web, incluyendo otros trabajos esporádicos relacionados con el diseño o el arte. Aparte, claro está, de imprevistos que han surgido en el camino, vida social, voluntariado en un grupo Scout, práctica de deporte, etc. Por supuesto, han surgido dificultades de programación -no sería un proyecto real sin algún que otro quebradero de cabeza- y algún cambio en los requisitos por parte de los clientes.

Aún así, ha sido una gran satisfacción haber acabado este sistema de gestión, habiendo aportado y mejorado competencias en cuanto a este ámbito se refiere.

10.3. Trabajo futuro

Aunque el sistema de gestión obtenido cumpla con los objetivos y necesidades de la empresa en cuestión, a mi parecer, y en parte consecuencia de cambios de requisitos por parte de la propia empresa, existen diversas mejoras que se pueden aplicar de cara al futuro:

- Para empezar, en lo que la gestión de actividades y citas se refiere, una posible mejora, que seguramente se lleve a cabo en un futuro próximo, podría ser la opción de crear clases o citas recursivas. Es decir, que se repitan en el tiempo, ya sea mediante la elección de los días de la semana en las que se repetiría, realizando el calendario de clases o citas semanalmente, o hacerla recursiva para que se repita la misma clase o cita cada semana el mismo día a la misma hora, creando dicha clase o cita solo una vez, eligiendo fecha de finalización del bucle si fuese oportuno.

- Otra posible mejora sería la adaptación total del sistema para su uso por parte de distintas empresas. El desarrollo ha sido realizado teniendo en cuenta esta opción, pero habría que completarlo con una gestión completa de organizaciones, y que todo lo relacionado con cada una de ellas esté automatizado, añadiendo simplemente su logo, estilo, etc. Que, por otra parte, sería un trabajo de poco esfuerzo, ya que como se ha mencionado la programación se ha realizado de esta forma, por lo que habría que dedicarle tiempo a la realización de pruebas con varias organizaciones y mejora o ampliación de las funcionalidades que necesiten.

- También se ha tenido en cuenta una mejora en cuanto a opciones y permisos. Con esto quiero decir, añadir la opción de que cada usuario elija si desea que se le pueda contactar mediante el correo interno, si desea recibir notificaciones por correo, etc. Esta mejora se aplicará próximamente, seguramente antes de que pase a producción.

- Otra de las posibles tareas a implementar sería la segmentación de usuarios. Es decir, la agrupación de los mismos dependiendo de unas ciertas características para, por ejemplo, destinar archivos subidos o mostrar comunicados solo a un grupo de usuarios, dependiendo si hacen un tipo concreto de actividad, usen algún servicio específico o que cumplan unos criterios de edad o peso.
- Una mejora que se tiene en cuenta para futuras versiones del software es la de personalización de la interfaz. En este momento es posible personalizarla a través del archivo de la hoja de estilo, pero se podría añadir la opción de customizar los colores de elementos como el cabecero, botones, etc, o elección del color principal y secundario.
- Un requisito que no se ha citado por parte de los clientes ha sido la internacionalización de la web pública. Por motivos de marketing y usabilidad, esta característica también será tenida muy en cuenta para un futuro próximo.
- Añadir otros idiomas para la interfaz sería otra opción a introducir con relativa facilidad. Solo habría que traducir el archivo de propiedades del lenguaje al idioma deseado.
- Y por último, respecto a la misma web pública, se está barajando la posibilidad de realizar un nuevo diseño usando *WordPress* **??** un *CMS* **??** mundialmente conocido y cada vez más extendido en el diseño web. Este cambio daría acceso a los clientes a la gestión de la web para realizar acciones puntuales, como la escritura de entradas del blog, por ejemplo. También facilitaría el mantenimiento de la web, así como posibles extensiones de la web, como la creación de un módulo de ventas.

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

The GNU General Public License is a free, copyleft license for software and other kinds of works.
The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.
When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.
To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.
For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.
Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.
For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.
Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for

those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.
Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.
The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions

0. Definitions.

   "This License" refers to version 3 of the GNU General Public License.

   "Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

   "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

   To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

   A "covered work" means either the unmodified Program or a work based on the Program.

   To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

   To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

   An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

   The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

   A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

   All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

   You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

   Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

   When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights

under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

   You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

   You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

   a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

   b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

   c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

   d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

   A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

   You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified ob-

ject code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work.

These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent

license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

    If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

    Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

    The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

    THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

    If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

End of Terms and Conditions

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <textyear>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program>  Copyright (C) <year>  <name of author>

This program comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
This is free software, and you are welcome to redistribute it
under certain conditions; type ‘show c’ for details.
```

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see `http://www.gnu.org/licenses/`.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read `http://www.gnu.org/philosophy/why-not-lgpl.html`.