|  |
| --- |
| **UNIVERSIDAD EUROPEA**    ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO  CICLO FORMATIVO DE GRADO SUPERIOR DESARROLLO DE APLICACIONES MULTIPLATAFORMA |
| PROYECTO FIN DE CICLO |
| **PÁDEL RAC** |
|  |
| **Rafaela Barrena**  **Carlos Dizy**  **Alfredo Rodríguez** |
| **CURSO 2021-22** |

**TÍTULO**: **PÁDEL RAC**

**AUTOR**: Rafaela Barrena Pajuelo, E. Alfredo Rodríguez Pozo, Carlos Dizy Herrero

**TUTOR DEL PROYECTO**: Carlos Elvira Gómez

**FECHA DE LECTURA**: 8 de Junio de 2022

**CALIFICACIÓN**:

Fdo: Rafaela Barrena Pajuelo,

E. Alfredo Rodríguez Pozo,

Carlos Dizy Herrero

Tutor/a del Proyecto : Carlos Elvira Gómez

**RESUMEN:**

El objetivo del presente proyecto es desarrollar una aplicación web para la gestión de un centro deportivo de pádel.

Para que el diseño de la aplicación sea lo más real posible y en un futuro poder producirla y venderla, para el uso en centros deportivos, hemos contactado con un centro deportivo ubicado en el polígono de Vallecas.

En dicho centro deportivo realizamos un estudio detallado de cuáles son las necesidades del centro, y el funcionamiento de este, y así poder describir todas las tareas que se quiere llevar a cabo a través de la aplicación.

**ABSTRACT:**

The objective of this project is to develop a web application for the management of a padel sports centre.

In order to make the design of the application as real as possible and in the future to be able to produce and sell it for use in sports centres, we have contacted a sports centre located in the Vallecas industrial estate.

In this sports centre we carried out a detailed study of the needs of the centre and how it works, in order to describe all the tasks to be carried out through the application.

**AGRADECIMIENTOS**

Es de bien nacidos el ser agradecidos”. Puede ir en algún orden (de más a

menos implicación directa en el proyecto: profesores y compañeros, amigos, familia) o no.  Esta obra se distribuye bajo una licencia Creative Commons.

Se permite la copia, distribución, uso y comunicación de la obra si se respetan las

siguientes condiciones:

* Se debe reconocer explícitamente la autoría de la obra incluyendo esta nota y su
* enlace.
* La copia será literal y completa
* No se podrá hacer uso de los derechos permitidos con fines comerciales, salvo permiso expreso de los autores.

El texto precedente no es la licencia completa sino una nota orientativa de la licencia

original completa(jurídicamente válida) que puede encontrarse en: http://creativecommons.

org/licenses/by-nc-nd/3.0/deed.es

**INDICE**

[1. Introducción 1](#_Toc104744578)

[1.1. Objetivos 1](#_Toc104744579)

[1.2. Motivaciones 1](#_Toc104744580)

[1.3. Antecedentes 1](#_Toc104744581)

[2. DESARROLLO DE LA PRÁCTICA 2](#_Toc104744582)

[2.1. Material 2](#_Toc104744583)

[2.2. Planificación 6](#_Toc104744584)

[**2. Ejecución: Sprint** 6](#_Toc104744585)

[**3. Control: Burn Down** 7](#_Toc104744586)

[2.3. Descripción del trabajo realizado 7](#_Toc104744587)

[2.4. Resultados y validación 16](#_Toc104744588)

[3. CONCLUSIONES 17](#_Toc104744589)

[3.1. Aportaciones 17](#_Toc104744590)

[3.2. Trabajo futuro 17](#_Toc104744591)

[4. BIBLIOGRAFÍA Y WEBGRAFÍA 19](#_Toc104744592)

[5. ANEXOS I](#_Toc104744593)

[5.1. Presentación general de un informe I](#_Toc104744594)

[5.2. Directrices particulares para el proyecto II](#_Toc104744595)

[5.3. Directrices particulares para los listados de programas II](#_Toc104744596)

[5.4. Algunas reglas mecanográficas IV](#_Toc104744597)

**INDICE DE LISTADOS (OPCIONAL)**

Listado 2: Hola mundo en ensamblador III

**INDICE DE FIGURAS (OPCIONAL)**

Figura 1: Captura de pantalla 2

# Introducción

**PÁDEL RAC** es una aplicación de gestion de pistas de padel, en formato web que esta orientada para centros públicos y privados.

Con esta aplicación se podrá realizar reservas, anulación y modificación de las pistas de Pádel

Contará con un diseño fácil y funcional.

## Objetivos

Los objetivos de **PÁDEL RAC**, hacer una apliacion que pueda ayudar a los usuarios de este deporte a realizar reservas y consultas con mas rapidez.

Aplicar todos los conocimientos adquiridos en el grado, aportando nuevas tecnoliagias y mejoras en los dispositivos

## Motivaciones

**PÁDEL RAC,** se crea ante la necesidad de utilizar tecnologias de vanguardia para poder adaptarlas a las necesidades de los usuarios y con ello mejorar la gestion en las entidades publicas y privades de centros deportivos.

La motivación del equipo de **PÁDEL RAC** es desarrollar por nuestros medios una aplicación funcional con tecnologías avanzadas de uso actual en las empresas.

## Antecedentes

El equipo PÁDEL RAC, como usuarios, aficionados y jugadores de pádel, hemos sufrido la mala gestión de las reservas de pistas que tienen algunos centros deportivos.

Ante la pasividad de estas instituciones y miedo al cambio, nos vemos obligados a la creación de una aplicación web, donde, tanto el usuario como el propio centro puedan interactuar de manera fácil y efectiva.

Para ello, después de un estudio pormenorizado y detallado de cuales son los aspectos a implementar en la aplicación, hemos decidido utilizar herramientas para el diseño de una aplicación que funcione vía web, y que ademas pueda albergar todos los datos necesarios de manera local en una base de datos.

En dichos datos va incluida la información de los propios usuarios, los detalles técnicos del centro y de las pistas y lo mas importante las reservas que cada usuario realizara, en función del horario, el tamaño de la pista o la cantidad de jugadores. Además el usuario tendrá la opción de crear su propio partido, a la cual podrá añadir jugadores, u otros jugadores podrán añadirse a ese partido.

# DESARROLLO DE LA PRÁCTICA

La metodologia de trabajo aplicada en este proyecto se basa una buena planificacion , con un desarrollo integrado de los lenguajes aprendidos y la utilizacion de las nuevas tecnologias aprendidas.

## Material

El desarrollo y codificación de la aplicación se realizará bajo las siguientes tecnologías:

* Como marco de trabajo utilizaremos el framework Spring para la inyección de dependencias, bajo la plataforma Java EE. Utilizaremos el IDE Eclipse Enterprise para desarrollo web.
* Usaremos el patrón MVC (Modelo – Vista - Controlador), para lograr separar la interfaz de usuario de la lógica de la aplicación.
* La gestión de los datos se realizará con: o MySQL como gestor de base de datos.
* Hibernate para el mapeo objeto-relacional, y la persistencia de datos. Técnica que se usa para mapear una representación de datos desde un modelo de objeto, que en un principio serán POJO, a un modelo de datos relacional.
* Apache Tomcat 9, como servidor de la aplicación.
* JSP, CSS y Javascript, para la resolución de las vistas, estilos y scripts del lado de cliente.

**Spring Framework**

* Spring es un framework de código abierto.
* Fué creado para simplificar el desarrollo de aplicaciones empresariales usando POJO’s para conseguir objetivos que antes sólo eran posibles con EJB.
* Simplificar el desarrollo en Java. Spring es un framework no intrusivo, esto quiere decir que nuestras clases serán simples clases Java (POJO´s) no teniendo que heredar de otras clases, ni implementar otras interfaces propias de Spring.
* Evitar el acoplamiento. Spring utiliza la inyección de dependencias. No crearemos el objeto sino que esperaremos a que nos lo faciliten.
* Reutilización de objetos. Mas adelante veremos como los beans creados por Spring se mantienen en un contenedor y de esta forma podremos reutilizarlos.
* Reducir el código reutilizable mediante aspectos y plantillas. Tomando como ejemplo JDBC vemos la cantidad de código que se repite para lanzar dos queries muy parecidas. Spring soluciona esto utilizando plantillas.

***Componentes de Spring***

Spring es un marco de trabajo modular que cuenta con una arquitectura organizada en 20 módulos diferentes, que se pueden separar en seis categorías de funcionalidad (véase la figura 2-1).

En conjunto, estos módulos proporcionan todo lo necesario para desarrollar una aplicación empresarial. Además, no es necesario basar la aplicación al completo en Spring, basta con hacer uso de aquellos módulos que se ajusten a ésta. (Spring incluso ofrece puntos de integración con otros marcos de trabajo y bibliotecas).



*Inyección de dependencias*

La inyección de dependencias (DI) es una forma de asociar objetos de aplicación, de forma que no tengan por qué saber de dónde proceden sus dependencias o la forma en que se implementan. En lugar de adquirir las dependencias por ellos mismos, se proporciona a los objetos dependientes aquellos de los que dependen.

Cualquier aplicación está formada por una serie de objetos que deben trabajar de forma conjunta para conseguir un objetivo de negocio.

Estos objetos deben ser conscientes de la existencia de los otros y comunicarse entre sí para llevar a cabo el trabajo. Sin embargo, el enfoque tradicional al crear asociaciones entre objetos de aplicaciones (mediante construcción o búsqueda) genera código complicado, difícil de reutilizar y de probar como unidades.

En Spring, los objetos (bean) no son responsables de encontrar o crear el resto de objetos que necesitan para llevar a cabo su trabajo. En su lugar, el contenedor les asigna referencias a los objetos con los que tienen que colaborar.

En una aplicación basada en Spring, sus objetos de aplicación van a residir dentro del contenedor de Spring. El contenedor va a crear los objetos, los va a conectar, a configurar y a administrar su ciclo de vida completo.

Como se indicó en el apartado anterior, el contenedor de Spring se encuentra en el núcleo del marco de trabajo. Utiliza inyección de dependencias (DI) para administrar los componentes que forman una aplicación. Esto incluye la creación de asociaciones entre componentes que colaboran entre sí. De esta forma, los objetos están más limpios y son más fáciles de comprender, permiten su reutilización y son más fáciles de probar.

*Ciclo de vida de un bean*

En una aplicación Java tradicional, el ciclo de vida de un bean es sencillo. La palabra clave new de Java se utiliza para instanciar el bean. A continuación, el bean está listo para utilizarlo. Una vez deje de utilizarse, puede eliminarse.

Frente a esto, el ciclo de vida de un bean en un contenedor Spring es más complejo, pasa por varios pasos entre su creación y su eliminación. Cada paso es una oportunidad para personalizar la forma en que el bean se administra en Spring.

Los pasos son:

1. Spring instancia el bean.
2. Spring inyecta valores y referencias de bean en las propiedades de éste.
3. Si el bean implementa BeanNameAware, Spring proporciona el ID del bean al método setBeanName().
4. Si el bean implementa BeanFactoryAware, Spring invoca el método setBeanFactory(), proporcionando él mismo la fábrica de *bean*.
5. Si el *bean* implementa *ApplicationContextAware*, Spring invoca el método *setApplicationContext*(*)*, proporcionándolo en una referencia al contexto de aplicación contenedor.
6. Si el *bean* implementa la interfaz *BeanPostProcessor*, Spring invoca su método *postProcessBeforeInitialization()*.
7. Si el *bean* implementa la interfaz *InitializingBean*, Spring invoca su método *afterPropertiesSet()*. De forma similar, si el *bean* se ha declarado con un método *init*, se invoca el método de inicialización especificado.
8. Si el *bean* implementa la interfaz *BeanPostProcessor*, Spring invoca su método *postProcessAfterInitialization()*.
9. Llegados a este punto, el *bean* estará listo para que la aplicación lo utilice, y va a permanecer en el contexto de la aplicación hasta que se elimine.
10. Si el *bean* implementa la interfaz *DisposableBean*, Spring invoca sus métodos *destroy()*. Del mismo modo, si se ha declarado con un método *destroy*, se invoca el método especificado.

*Contenedores de Beans*

La filosofía de Spring es la reutilización de objetos (en adelante beans). Se trata de crear un objeto y almacenarlo en un contenedor para su posterior uso.

En una aplicación desarrollada con Spring, los beans (objetos de aplicación) van a residir dentro del contenedor de beans.

Dicho contenedor se encuentra en el núcleo (Core).

El contenedor utiliza DI (Inyección de Dependencias) para crear los beans, conectarlos con otros, configurarlos y administrar su ciclo de vida.



En adelante, se utiliza indistintamente el nombre contenedor, core-container, Spring IoC container o contexto de Spring para referirse a éste.

**Acceso a datos**

Un requisito que tienen prácticamente todas las aplicaciones es la persistencia de datos. La persistencia es un atributo de los datos que asegura que estarán disponibles incluso más allá de la vida de una aplicación, y aquí entran en juego las bases de datos. Se puede decir que una base de datos es un conjunto de datos persistentes que es utilizado por los sistemas de aplicación de alguna empresa dada.

En este capítulo se va a tratar sobre el sistema de gestión de base de datos utilizado, MySQL; el mapeo objeto-relacional, y la persistencia en aplicaciones basadas en Spring.

Para la aplicación “PadelCAR” se ha hecho uso de MySQL. MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation y está considerada como la base datos open source más popular del mundo, y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web.



*Características de MySQL*

Entre las características de MySQL destacan:

* Escalabilidad y flexibilidad: Está disponible en gran cantidad de plataformas y sistemas. Ofrece la posibilidad de selección de mecanismos de almacenamiento que ofrecen diferentes velocidades de operación, soporte físico, capacidad, … Además, la naturaleza de código abierto de MySQL permite una personalización completa.
* Alto rendimiento
* Alta disponibilidad
* Apoyo transaccional robusto
* Fuerte protección de datos: Ofrece un sistema de contraseñas y privilegios seguro. Además, viene con soporte incorporado para SSL.
* Facilidad de gestión
* APIs disponibles para múltiples lenguajes de programación

*Configuración*

Para hacer uso de la base de datos MySQL, hay que incluir en el fichero POM la dependencia del conector de MySQL.

//TODO:

## Planificación

El equipo de Padel CAR se ha organizado con reuniones semanales todos los martes para plantear y diseñar y desarrollar la aplicación.

Para el desarrollo de la aplicación hemos trabajado sobre un repositorio remoto en GitHub, donde los integrantes del equipo han podido añadir, modificar y actualizar las partes de la aplicación para que esta funcione correctamente.

Padel Car hace uso de la metodología **Scrum** proceso para llevar a cabo el conjunto de tareas de forma regular con el objetivo principal de trabajar de manera colaborativa, para fomentar el trabajo en equipo. Con este método de trabajo lo que se pretende es alcanzar el mejor resultado de un proyecto determinado

La metodología Scrum pasa por diferentes fases que hacen posible que se lleve a cabo con éxito.

**1. Planificación: Product Backlog**

El Product Backlog es la fase en la que se establecen las tareas prioritarias y donde se obtiene información breve y detallada sobre el proyecto que se va a desarrollar.

El Product Backlog es necesario para poder arrancar con el primer sprint, tiene permitido cambiar y crecer tantas veces como sea necesario en función del aprendizaje adquirido en el desarrollo del producto.

**2. Ejecución: Sprint**

Dentro del método Scrum, **el Sprint es el corazón**, es el intervalo de tiempo que como máximo tiene una duración de un mes y en donde se produce el desarrollo de un producto que es entregable potencialmente.

**se puede definir el Sprint como un mini proyecto** en donde el equipo de trabajo sea focalizado en el desarrollo de tareas para alcanzar el objetivo que se ha definido previamente en el Sprint planning.

**3. Control: Burn Down**

El Burn Down es la fase en la que **se mide el progreso de un determinado proyecto** Scrum. En ella, el Scrum Master será el encargado de actualizar los gráficos cuando se finalice cada uno de los Sprint.

## Descripción del trabajo realizado

**Diseño del proyecto**

Actualmente Spring se ha convertido en el estándar para el desarrollo de

numerosos proyectos Java, por este motivo se utiliza para el desarrollo del presente

proyecto.

Spring nos permite desarrollar aplicaciones de manera más rápida y eficaz. Tiene

una amplia compatibilidad para la integración con otros frameworks y librerías de uso

común para la creación de aplicaciones web, como Tiles, APIs en capa de persistencia

como Hibernate y otros muchos recursos.

En nuestro caso, Spring se integra al proyecto como dependencia a través de

Maven, por lo tanto, los módulos de Spring que se deseen agregar en el proyecto se

añadirán en el pom.xml (Project Object Model) para que Maven lo integre. Las

dependencias se añaden entre las etiquetas <dependencies></dependencies>.

Por ejemplo:

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>4.3.0.RELEASE</version>

</dependency>

Con el objetivo de simplificar el desarrollo de aplicaciones Spring posee la

característica de Inyección de Dependencias (DI), que es una forma de asociar objetos

de la aplicación. Para explicarlo de forma sencilla y rápida, en Spring, los objetos

(bean) no son responsables de encontrar o crear el resto de los objetos que necesitan para llevar a cabo su trabajo, en su lugar, el contenedor del núcleo de Spring (fábricas de beans), les asigna referencias a los objetos con los que tienen que colaborar.

Spring ofrece tres mecanismos para configurar los beans y sus dependencias, que

no son excluyentes y se pueden mezclar:

* Configuración explícita de XML
* Configuración explícita en Java
* Detección implícita y conexión automática de bean.

Para este proyecto se opta por la configuración en Java, con el uso de

anotaciones, ya que es más potente, ofrece seguridad de tipos y permite

refactorización.

Las clases anotadas con @Component (del paquete org.springframework.stereotype) identifica a una clase como clase de componente y sirve para indicar a Spring que debe crear un bean para la clase. Por defecto, el identificador del bean creado va a ser el nombre de la clase anotada con la primera letra en minúscula: @Component ("idDelBean”).

Por ejemplo, con la anotación

@ComponentScan: @ComponentScan(basePackages = "padelcar")

La conexión automática es una forma de permitir que Spring satisfaga

automáticamente las dependencias de un bean buscando otros bean de la aplicación

que coincidan con sus necesidades.

Para indicar que debe realizarse la conexión automática se utiliza la anotación @Autowired del paquete org.springframework.beans.factor.annotation.

Por ejemplo, en el proyecto se conecta el bean de IClienteService en ClienteController.java:

@Autowired IClienteService clienteService

Esquema de la aplicación en base a la estructura del framework Spring 4:

Escala de tiempo

Descripción generada automáticamente con confianza media

**Arquitectura del sistema**

Para el desarrollo de la aplicación se ha usado el entorno Eclipse Java EE IDE

for Web Developers , la aplicación web se ha desarrollado en Java

EE 8. Las dependencias de librerías se resuelven con Apache Maven.

Se utiliza Spring MVC 4 para implementar el modelo MVC y la inyección de

dependencias. Para la base de datos se utiliza MySQL Server 5 y para el acceso a la

base de datos se usará JPA+Hibernate 4.

En la parte de la vista, se hará uso de las tecnologías Apache Tiles, para la

resolución de vistas; JSP, para las vistas; CSS y BootStrap, para el estilo; y Javascript

para los scripts del lado del cliente.

El servidor de aplicación es Apache Tomcat 9.

En la siguiente figura se muestran las diferentes tecnologías que se han

utilizado de forma esquemática:



**Modelo de datos**

El modelo de datos se ha implementado sobre una base de datos MySQL Server

Llamada padelcardb. El diseño del modelo de datos con las tablas que componen la

aplicación, los atributos de las tablas y las relaciones establecidas entre éstas, son

descritas en el siguiente diagrama:

Diagrama

Descripción generada automáticamente

A continuación, se muestra una breve descripción de las tablas utilizadas:

Tabla Cliente:

En esta tabla se van a almacenar los datos referentes a los clientes

Existentes:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Siguiente Tabla, se corresponde con un catalogo de las pistas existentes que se pueden seleccionar:

PISTAS:

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

La tabla Reserva:

Se almacenan las reservas que hacen los clientes con las informaciones para dicha reserva

Texto

Descripción generada automáticamente

**Modelo de capas**

El diseño de la aplicación va a seguir el siguiente modelo de capas:



**Estructura de paquetes**

Las clases Java van a ir dentro de la carpeta src/main/java en los siguientes

paquetes:

* Configuración:padelcar.config
* Capa de Dominio (Entidades): padelcar.model
* Capa de Persistencia (Repositorios): padelcar.dao
* Capa de Acciones (Controladores):padelcar.controller
* Capa de Negocio (Servicios): padelcar.service

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

***CONFIGURACIÓN****:* en este paquete van a ir las diferentes clases de

configuración, definidas mediante la anotación @Configuration:

Texto

Descripción generada automáticamente

HibernateConfig.java: Clase para configurar el contexto raíz, en concreto los componentes del nivel de datos. Se define la fábrica de administradores de entidades, las propiedades de Hibernate y la base de datos de la aplicación. Para conectar con la base de datos de la aplicación se añade la anotación @PropertySource(value =

{ "classpath:config.properties"}).

En el archivo config.properties se indica, entre otras cosas, el nombre de la base de datos y sus usuario y contraseña de acceso.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

WebConfig.java: Clase para configurar el contexto servlet.

WebInitializer.java: Clase para configurar el DispatcherServlet

***CAPA DE DOMINIO*:** En la capa de dominio se encuentran las entidades del sistema. A cada tabla del modelo de datos se le asigna una entidad.

Cuando se configura la fábrica de entidades, en la clase HibernateConfig se indica este paquete. Las clases de este paquete deben tener las siguientes características:

La clase va a llevar la anotación @Entity

El nombre de la clase coincidirá con el de la tabla, la primera letra en mayúscula y el resto en minúscula. La clase llevará la siguiente anotación con el nombre de la table correspondiente en cada caso: @Table(name="cliente")

Por cada propiedad de la clase se va a tener un get/set asociado.

***CAPA DE PERSISTENCIA:*** En esta capa se encuentran las interfaces y

clases usadas para el acceso a la base de datos, es decir lo repositorios, y

se anotarán mediante @Repository.

La convención seguida para nombrar cada repositorio es utilizar el nombre de la entidad con el sufijo “Dao”, por ejemplo, “ClienteDao”.

Las interfaces siempre irán nombradas con una I de prefijo y las clases que implementan una interfaz llevarán el sufijo Impl, por ejemplo “ClienteDaoImpl”, para el caso de las clases e “IClienteDao” para el caso de las interfaces.

***CAPA DE ACCIONES:*** En el paquete padelcar.controller van a ir las clases que realicen la función de controlador, es decir, las clases encargadas de interceptar las solicitudes HTTP del cliente y de traducir dichas solicitudes en operaciones específicas de negocio a ser realizadas.

Se sigue la siguiente convención:

* Utilizan el sufijo “Controller” en su nombre.
* Llevan la anotación @Controller
* En la medida de lo posible, no realizarán funciones de procesamiento, sino que se realizará la lógica del negocio en los objetos del servicio.

***CAPA DE NEGOCIO***: es una capa intermedia que está encargada de coger los datos de la capa de persistencia (los repositorios) y proporcionar la funcionalidad necesaria a la capa de acción (los controladores). Para cada entidad va a haber una clase de negocio que va a:

* Utilizar el sufijo “Service” en su nombre, por ejemplo“ClienteServiceImpl”, y llevará la anotación @Service
* Implementar una interfaz cuyo nombre será el mismo que el de la clase y con el prefijo “I”, por ejemplo “IClienteService”.

***Capa de presentación***

La capa de presentación la constituyen las vistas que se van a integrar con el

resto de capas.

A modo de ejemplo se añade una vista del proyecto

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Los ficheros correspondientes a la capa de presentación van a ir dentro

de la carpeta src/main/webapp.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Páginas JSP: En la aplicación se va a utilizar JSP para la tecnología de la vista. Para el diseño se va a hacer uso de Apache Tiles como motor de diseño para reducir la duplicación de elementos comunes. En la carpeta WEB-INF está ubicado:

En WEB-INF/tiles se encuentra el fichero tiles.xml que va a contener las especificaciones de mosaico.

En WEB-INF/tiles/layouts está la plantilla que va a usar el mosaico base.

En WEB-INF/tiles/template están las plantillas de cada apartado (encabezado, menú y pie de la web)

En WEB-INF/page se encuentran el resto de los ficheros .jsp que se corresponden con la sección central de las diferentes páginas de la aplicación.

Otros elementos de la capa de presentación: Dentro de la carpeta

webapp/static se encuentran el resto de los elementos como son los ficheros

css (static/css) para aplicar un estilo propio a la aplicación, los ficheros

javascript (static/js) y las imágenes ubicadas en la carpeta static/img.

## Resultados y validación

En esta última parte se deben presentar los resultados obtenidos del trabajo. Si el trabajo consistía en hacer un programa se comentará su funcionamiento, lo que tarda en ejecutarse, lo bien que va, los problemas que tiene, etc.

Sobre todo también hay que presentar los resultados de las simulaciones o ejecuciones que se hayan realizado. Si se ha hecho una red es normal probar que funciona, en este caso se comentarán las condiciones en las que se ha probado, la normativa que se ha utilizado, por qué creemos que funciona bien, etc. Esto mismo es aplicable a la instalación de un servicio de red o diseño de una base de datos.

Normalmente el diseño de la red, la configuración del servicio o la documentación de la base de datos se adjuntan en un apéndice al final, salvo que sea realmente breve y se desee comentar aquí mismo. Ver los apéndices al final de esta guía para ver cómo presentar correctamente los listados.

Cuando los resultados contengan variables numéricas o estadísticas es necesario hacer una análisis de los errores de los valores obtenidos. Este análisis debe incluir tanto la exactitud como la precisión de las medidas obtenidas.

En esta parte de resultados debe incluirse un estudio detallado de los resultados obtenidos junto con sus implicaciones. Un resumen de este estudio se debe incluir en la parte final de conclusiones.

# CONCLUSIONES

Son las conclusiones propiamente dichas del trabajo realizado. Es uno de los capítulos **más importantes.**

Comenzar con un resumen de los realizado destacando los aspectos más importantes, principales hitos conseguidos, principales problemas encontrados, etc. Se comenta si se han conseguido los objetivos, si se recomienda el método utilizado o cualquier otro, si el resultado es fiable o no, si son necesarias nuevas pruebas; es decir, se hace un resumen breve de los principales puntos del trabajo realizado y los resultados obtenidos. Este es un buen lugar para destacar la razón de retrasos, parones, o cualquier otro incidente que haya retrasado la realización del PFC.

## Aportaciones

En este apartado se pueden destacar aquellos aspectos novedosos que aporta la realización del proyecto. Resulta interesante evaluar el impacto que puede tener cada una de las aportaciones realizadas.

En las aportaciones resulta interesante también comentar la relación de los resultados obtenidos con los trabajos previos que existan sobre el tema.

## Trabajo futuro

En los trabajos de investigación extensos con objetivos ambiciosos conviene enumerar aquellos puntos del trabajo sobre los que se debe hacer énfasis. También se deben destacar aquellas líneas que el trabajo abre y que pueden dar resultados interesantes.

# BIBLIOGRAFÍA Y WEBGRAFÍA

Al final del informe es necesario añadir la bibliografía general con las referencias que se hayan hecho a lo largo del texto. Un trabajo de investigación y desarrollo sin referencias es un trabajo muy poco documentado y demuestra una falta del estudio de los antecedentes y entorno del trabajo, por lo tanto tiene poca credibilidad como tal.

La bibliografía debe seguir el formato A.P.A. como los siguientes:

EDUCAMADRID. (2011) www.educa2.madrid.org. Fecha de consulta: 22:17, febrero 14, 2011 de<http://www.educa2.madrid.org/educamadrid>

CLOUTIER, J. (1975). L’ére D’emerec ou la comunication audio-scrito-visuelle à L’ heure des self-media. (segunda ed.). Montréal: Les Press de L’ Université de Montréal.

CALLEJO GALLEGO, J. (2008). El esquema espaciotemporal en la sociedad digital. Madrid. Madrid: UNED.

O’REILLY, T. (2004, 05) de Octubre. Conferences.oreillynet.com. Fecha de consulta: 09:09, febrero 15, 2011 de<http://conferences.oreillynet.com/web2con>

KAPLÚN, M. (1998). Una pedagogía de la comunicación. Madrid: Ediciones de la Torre.

# ANEXOS

En los anexos se recoge información técnica subsidiaria al trabajo realizado y que no tiene cabida directa en el esquema presentado hasta ahora. Es el lugar ideal para los listados de programas, esquemas de circuitos, demostración de teoremas, etc.

Numerar con letras mayúsculas (pueden tener (sub)secciones: A.1, A.3.1, etc.). Incluir en el anexo todo aquello que ocupe muchas páginas y tenga una estructura repetitiva que pueda hacer tediosa la lectura (resumir en el texto principal y referenciar el anexo). El Manual de Usuario puede incluirse como un anexo o como un documento aparte (comenzando en la página 1 o no). Debería ser una guía de uso donde se explica cómo utilizar cada funcionalidad que ofrezca el sistema, explicando los pasos a dar y apoyándonos masivamente en pantallazos.

## Presentación general de un informe

Lo más importante de un trabajo es sin duda el trabajo en sí y sus resultados, sin embargo, un buen trabajo mal presentado arrojará dudas sobre lo allí expuesto y sobre la persona que ha realizado el trabajo y la memoria. Por lo tanto resulta conveniente cuidar la presentación del mismo:

* Hay que evitar en lo posible las faltas de ortografía y de mecanografía. Con los procesadores modernos de texto es cada vez más difícil cometer este tipo de faltas. Hay que aprender a usar y utilizar las herramientas de corrección ortográfica. Las herramientas de revisión sintáctica son también aconsejables, aunque todavía están en proceso de mejora. Un trabajo presentado con faltas de ortografía, aunque sea bueno, dará una impresión lamentable. Al final de los apéndices se dan unas reglas mecanográficas básicas.
* Los trabajos profesionales se deben presentar mecanografiados siempre.
* Es mejor utilizar un tipo de letra estándar y un tamaño también estándar. El tamaño de letra debe estar entre 10 y 12 pt., y la fuente puede ser Times, Arial o similar. Por lo general la letra Times de 12 pt. suele ser la más utilizada. Nunca hay que usar tipos raros o poco legibles y nunca tamaños inferiores a 10 pt. (10 pt. ya es realmente pequeño) ni superiores a 12 pt.
* Los listados de los programas se hacen con una fuente que tenga espaciado fijo (Courier por ejemplo). Estas fuentes suelen ser más grandes que las normales por lo que puede ser interesante cambiar también el tamaño de letra y dejarlo entre 8 y 10 pt.
* El interlineado debe ser simple y nunca menor de una línea. En algunos casos se puede agrandar algo pero sin sobrepasar la mitad del interlineado simple (1.5 líneas).
* En trabajos extensos resulta interesante incluir un índice con los capítulos, secciones y subsecciones que contenga, así como la página en la que se encuentra.
* Una buena presentación no significa que haya que utilizar todos los tipos de letra posibles, ni todos los tamaños, ni todos los colores; la utilización de estos recursos (color, enfatizado, tamaño, etc.) sólo es interesante siempre que quede justificado, siendo la regla general no alterar el texto estándar a no ser que se esté seguro de querer hacerlo.
* La inclusión de esquemas y figuras ayudan a la claridad y mejoran la presentación de un trabajo.
* Es aconsejable utilizar hojas blancas A4 (210 x 297 mm) por ser un formato estándar y el más utilizado en Europa. Se pueden incluir otro tipo de hojas (milimetradas, cuadriculadas,...) siempre que exista una buena razón para ello (gráficos, esquemas, etc.) y siempre que el tamaño no sobresalga del resto.
* El proyecto se debe encuadernar de forma que su consulta sea lo más cómoda posible.

## Directrices particulares para el proyecto

Los consejos anteriores son válidos para la realización de un informe en general. Para el proyecto fin de ciclo es aconsejable su aplicación

* La presentación sigue siendo muy importante, pero en este caso son más importantes los contenidos. Por tanto no hay que perder el tiempo en hacer rótulos bonitos a todo color ni buscar dibujos de ordenadores por la red.
* En la portada añadir el título del proyecto y los integrantes.
* No dejar hojas completas en blanco como separación, no sirven para nada y son un gasto inútil.
* A poder ser se presentarán las hojas escritas tanto por delante como por detrás.
* Abrir una nueva página al comenzar una nueva sección, salvo para evitar que un título quede suelto. Hacer un salto de página para empezar una nueva sección.
* El proyecto se encuaderno con una espiral simple.
* La extensión del proyecto debe ser entre 30 y 60 páginas incluyendo todo el trabajo realizado, las conclusiones, etc., sin extenderse en detalles superfluos. Esto depende de si el trabajo la presenten una o más personas y si está realizada en la empresa

.

## Directrices particulares para los listados de programas

En los proyectos es bastante común la realización de memorias sobre la creación de algún programa, da igual el lenguaje, que haga alguna cosa. En estos casos las directrices dadas en los puntos anteriores son de obligada aplicación, aunque resulta conveniente resaltar los siguientes aspectos:

* El listado del programa debe estar comentado siempre. Si un programa no tiene comentarios casi es mejor no incluirlo.
* Un listado del programa, por muy bien comentado que esté, no es por sí solo la memoria del proyecto, de hecho debería ser una de las partes menos importantes.
* Si se incluye el listado completo del programa se pondrá en los anexos al final de la memoria. Sólo en el caso de que el programa sea muy pequeño (apenas una rutina) se puede poner en la parte de desarrollo o resultados.
* Si el listado es muy grande (más de 3 ó 4 páginas) es mejor no incluirlo o ponerlo aparte en un cd. También se puede intentar reducir el tamaño de letra, los espaciados, etc.
* En la parte de desarrollo o resultados se describe el funcionamiento del programa, por qué se ha hecho así, se explicará si funciona bien o no, bajo qué casos, etc. En estos apartados se pueden incluir trozos del listado del programa, pero sólo aquellas partes significativas para la explicación. Si el trozo que se incluye ocupa una página o más entonces no es aconsejable ponerlo.
* Los listados, tanto si son fragmentos como si es el listado global, se deben hacer con un tipo de letra de tamaño fijo (por ejemplo Consolas). Por otro lado, el tamaño debe ser lo menor posible para que ocupe poco espacio.
* Es recomendables insertar los listados en una table e insertarse un título (Referencias -> Insertar Título), si hay muchos listados puede ser interesante añadir un Índice de listados.

|  |
| --- |
| ; Image base = 0x00400000  %define RVA(x) (x-0x00400000)  section .text  push dword hello  call dword [printf]  push byte +0  call dword [exit]  ret    section .data  hello db "Hello world!"    section .idata  dd RVA(msvcrt\_LookupTable)  dd -1  dd 0  dd RVA(msvcrt\_string)  dd RVA(msvcrt\_imports)  times 5 dd 0 ; ends the descriptor table    msvcrt\_string dd "msvcrt.dll", 0  msvcrt\_LookupTable:  dd RVA(msvcrt\_printf)  dd RVA(msvcrt\_exit)  dd 0    msvcrt\_imports:  printf dd RVA(msvcrt\_printf)  exit dd RVA(msvcrt\_exit)  dd 0    msvcrt\_printf:  dw 1  dw "printf", 0  msvcrt\_exit:  dw 2  dw "exit", 0  dd 0 |

Listado 2: Hola mundo en ensamblador

## Algunas reglas mecanográficas

Estas reglas son casi tan importantes como las ortográficas y su incumplimiento produce una mala impresión del trabajo que se está realizando. Se listan a continuación algunas de las más importantes o utilizadas:

* Los signos de puntuación como los puntos, comas, dos puntos, etc., van siempre unidos a la palabra que preceda; es decir, nunca hay que poner un espacio delante de cualquiera de estos signos.
* En cuanto a los signos que se abren y cierran como las comillas, los paréntesis, las llaves, etc., el signo que abre debe ir unido a la palabra que sigue, y el signo que cierra debe ir unido a la palabra precedente; es decir, detrás de un signo que abre nunca va un espacio al igual que delante de un símbolo de cierre donde tampoco se pone espacio.
* Los títulos de las secciones o subsecciones no deben ir sueltos. Si no hay espacio en la página para empezar una sección se debe empezar en página nueva.