PROYECTO J2EE: Mi Menú de Hoy

Jose Antonio López Vicente
ETIG

Salvador Campo Mazarico

10/01/2016

Dedicatoria

Este proyecto está dedicado a mi mujer, Melina, que me apoyó a retomar los estudios y a mis hijas que me roban el tiempo necesario para su desarrollo, al igual que me dan fuerza para finalizarlo.

Resumen del proyecto

Este trabajo de fin de carrera, titulado MiMenuDeHoy, consistirá en realizar tanto el Análisis, Diseño como la Implementación mediante tecnología J2EE de una aplicación web que nos permita gestionar un negocio de venta de menús de diarios de alimentación por internet.

La aplicación se puede considerar dividida en dos partes, una parte de gestión o CMS donde los usuarios administradores gestionan los siguientes contenidos: Ingredientes, Platos, Recetas, Menús, Suscripciones, y otra parte pública, disponible para todos los usuarios, donde éstos pueden suscribirse al servicio, consultar sus menús asignados y modificar cierta información personal. Además la parte pública dispone de un buscador de recetas, visible para todos los usuarios, que puede usarse también con fines comerciales y de promoción de la plataforma.

El proyecto no contempla la lógica del reparto a domicilio, ni la venta, ni la gestión de stocks, pero está pensado como una base para ser extensible y adaptarse a estas y otras nuevas funcionalidades.

En cuanto a la implementación la mayor dificultad ha radicado en la integración del gran número de tecnologías utilizadas y que enumero a continuación: Java 7 como lenguaje, framework Spring MVC 3.2 como kernel de la aplicación, Hibernate 4.2.7 como motor ORM, Apache Tiles 3.0.5 como framework de vista para la gestión de plantillas, JSTL como lenguaje server side en las vistas, MySQL como Base de datos, el módulo Spring Security para gestionar la seguridad y permisos, Eclipse como herramienta de desarrollo y Maven como herramienta de empaquetado e integración. A esto hay que añadir HTML, CSS, Javascript y JQuery como lenguajes a nivel de cliente web.

Índice de Contenidos

1. Introducción

1.1 Justificación del TFC y contexto en el que se desarrolla: punto de partida y aportación del TFC

Este trabajo de fin de carrera pretende crear una plataforma que nos permita gestionar un negocio de venta de menús de diarios de alimentación por internet para dar servicio generalmente a las personas que realizan horario de oficina, y disponen habitualmente de entre una y dos horas para comer, durante las cuales no les da tiempo a ir a su domicilio y cocinar, o bien personas que no disponen de una oficina con cocina, o bien no se pueden planificar una dieta equilibrada sea por falta de tiempo o por habilidades, y no se pueden cocinar sus platos del día siguiente.

Para un comercio de este tipo le ayudaría a tener inventariado sus menús, gestionar sus usuarios y además le serviría como punto de entrada de captación de clientes ya que existirá una parte web accesible a todo el público por internet. Además de esta parte pública existirá una parte privada o de gestión, solo accesible para el comercio.

El funcionamiento a grandes rasgos seria el siguiente:

Los usuarios accederán vía web a la parte pública de Mi Menú de Hoy, podrán registrarse y suscribirse por un periodo de tiempo mensual al servicio de menús, deberán escoger el tipo de menú (existirán menús para personas con alguna incompatibilidad alimenticia por alergias). Una vez suscritos el aplicativo ofrecerá diariamente un menú al usuario (previamente asignado por un usuario con perfil Administrador o Nutricionista). Además todos los usuarios de la web tanto suscritos como no suscritos podrán acceder a la base de datos de recetas del aplicativo.

Los usuarios administradores dispondrán de un panel de administración donde podrán gestionar los Ingredientes, Platos, Recetas, de productos, los menús y las recetas publicadas.

En cuanto a los diferentes actores del sistema, existirán cuatro tipos:

- Usuario **Administrador**: administrador del negocio

- Usuario **Nutricionista**: gestor, creador de los menús en base a criterios de salud

- Usuario **Cliente**: usuario suscrito
- Usuario **Anónimo**

Permisos de cada Tipo de Usuario:

- **Administrador**: dispondrá de todos los permisos de la aplicación

- **Nutricionista**: Solo tendrá acceso a la parte del panel de administración de la confección de menús.

- Usuario **cliente**: solo dispondrá de acceso a la parte pública, en la que podrá consultar recetas, suscribirse por un periodo de tiempo y consultar su menú diario.

- Usuario **Anónimo**: dispondrá de la posibilidad de ver los menús semanales y buscar recetas publicadas.

1.2 Objetivos del TFC

El objetivo del trabajo consiste en realizar el Análisis, Diseño y Desarrollo del aplicativo MiMenu de Hoy con tecnología J2EE utilizando para ello el patrón de arquitectura de software MVC. Se escoge el framework Spring MVC por el nicho de mercado que dispone y sus expectativas de futuro, así como una preferencia personal para poder actualizar mis conocimientos sobre esta tecnología.

Para todo el desarrollo se utilizarán tecnologías actuales y vigentes en el mercado, como son: Java 7 como lenguaje, para la capa de Modelo se utilizará como ORM la tecnología Hibernate, para la capa de vista se utilizará Apache Tiles, Spring Security como gestor de la seguridad y Spring MVC como nexo de unión entre todos ellos en la capa de controlador. Para la capa de vista también se utilizará HTML, Javascript, Jquery, JQueryUI.

1.3 Enfoque y método utilizado

Para llevar a cabo el proyecto, se seguirá una metodología basada en Métrica 3 orientada a la Planificación, Desarrollo y Mantenimiento de Sistemas de Información y conocida como “Ciclo de Vida” que seguirá las siguientes etapas:

Durante la etapa de **Análisis** se recopilará la información necesaria para el desarrollo de la aplicación. A partir de los requerimientos iniciales y de las necesidades funcionales se estudia y se analiza para plantear la mejor solución y detectar posibles puntos críticos del desarrollo.

Transcurrida esta etapa se procede a **Diseñar** la solución determinando y resolviendo de forma técnica cada una de las funciones de forma general del aplicativo.

Se definirán los “Casos de Usos” para cubrir cada una de las funcionalidades y las acciones a realizar obteniendo un modelo cercano a la programación orientada a objetos.

En esta etapa también se define la arquitectura y el modelo de base de datos a implementar.

En la etapa de **Pruebas** se comprueba que el software realice correctamente cada una de las funciones implementadas.

Durante las pruebas surgen errores, o cambios de funcionalidad que deben solventarse durante esta etapa para garantizar la entrega del producto final.

Finalmente, una vez realizadas las pruebas se puede decir que el aplicativo está disponible para su puesta en marcha.

En todo el ciclo de vida hay que ir realizando documentación para que en la entrega final se disponga de la documentación de todo el sistema.

Los objetivos a nivel de hitos a desarrollar serán los siguientes:

- Panel de Administración
    - Mantenimiento de Usuarios
    - Mantenimiento de Suscripciones
    - Mantenimiento de Ingredientes
    - Mantenimiento de Recetas
    - Mantenimiento de Platos
    - Mantenimiento de Menús
    - Mantenimiento de Menús Semanales o Asignación de Menús

- Web
    - Home
    - Buscador de Recetas
    - Registro
    - Suscripción
    - Mis Menús Semanales
    - Menús del día (Recetas Destacadas)
    - Mi área privada

Descripción funcional de cada módulo:

- Mantenimiento de Usuarios: listado de usuarios con posibilidad de alta, modificación y borrado (CRUD). Solo disponible para Administradores del Aplicativo.

- Mantenimiento de Suscripciones: Un usuario solo podrá estar suscrito a una suscripción al mismo tiempo por lo que este mantenimiento irá incluido en el mantenimiento de usuarios.

- Mantenimiento de Ingredientes: Con el podremos gestionar los diferentes ingredientes que aparecen en los platos.

- Mantenimiento de Platos: gestión de los diferentes platos, esta sección será accesible para el Nutricionista, que podrá dar de alta nuevos platos y gestionar los ingredientes que contiene cada plato así como las posibles categorizaciones del plato (nombre, descripción, pasta, nº de calorías , contiene alérgenos , etc.)

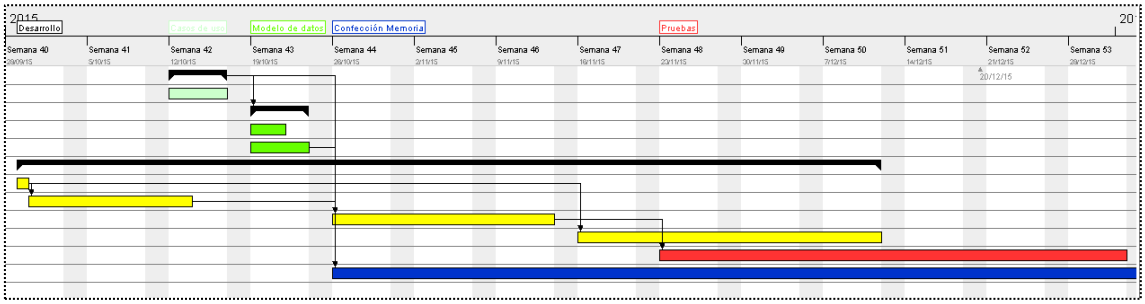
- Mantenimiento de Recetas: gestión de las diferentes recetas, ligadas a platos. En la misma se podrá incluir fotos, y destacar las mismas para la parte pública.

- Mantenimiento de Menús: El nutricionista podrá elegir los platos que forman parte de un Menú diario, categorizar este menú en función de sus consumidores.

- Asignación de menús: el nutricionista puede asignar diferentes menús al periodo que comprende una suscripción.

1.4 Planificación del proyecto.

Planificaremos el trabajo en las siguientes etapas:



- Etapa 1: Análisis funcional
  Se iniciará al finalizar la PAC1

- Etapa 2: Diseño Técnico
  Esta etapa se iniciará una vez finalizada la Fase de Análisis.

- Etapa 3: Desarrollo del aplicativo
  El desarrollo propiamente se inicia durante la PAC1, en la que se comienza a construir el core o núcleo del aplicativo. El desarrollo continúa una vez avanzado en el análisis funcional y diseño entregado en la PAC2.

- Etapa 4: Pruebas
  Esta etapa se irá realizando una vez iniciado el desarrollo.

- Etapa 5: Redacción de la memoria, y la presentación virtual esta etapa durará prácticamente la totalidad del proyecto, y se iniciará una vez finalizado el Diseño.

  Este es el detalle en cuanto a fechas de las tareas planificadas.

Nombre	Fecha de inicio	Fecha de fin
Análisis	12/10/15	16/10/15
Casos de uso	12/10/15	16/10/15
Diseño	19/10/15	23/10/15
Modelo de datos	19/10/15	21/10/15
Jerarquia Clases	19/10/15	23/10/15
Desarrollo	29/09/15	11/12/15
Preparación Entorno	29/09/15	29/09/15
Core Aplicación	30/09/15	13/10/15
Backend	26/10/15	13/11/15
FrontEnd	16/11/15	11/12/15
Pruebas	23/11/15	1/01/16
Confección Memoria	26/10/15	11/01/16

1.5 Productos obtenidos

Una vez realizado el Análisis, y el Diseño, la implementación dará como resultado un proyecto empaquetado en formato .war con sus instrucciones de instalación, así como esta memoria que incluye tanto el Análisis, Diseño, y método de implementación utilizado.

1.6 Breve descripción de los siguientes capítulos

En el capítulo siguiente, el número dos, describiremos el Análisis Funcional realizado, en el siguiente, el Diseño Técnico implementado, y en el cuarto el resultado de la implementación. En el quinto capítulo se incluirán unos pequeños manuales técnicos de instalación.
El sexto capítulo contendrá una estimación económica y por último el séptimo capítulo serán las conclusiones y algunas referencias de la bibliografía utilizada.

2. Análisis Funcional

2.1 Actores Principales

Tal como se expusieron en el documento de Plan de Trabajo, se considera que los usuarios factibles de esta aplicación se pueden agrupar en los siguientes tipos, enumerados de menor a mayor permiso:

- Usuario **Anónimo** o **Invitado**
- Usuario **Cliente**: usuario suscrito o registrado en el sistema.
- Usuario **Nutricionista**: gestor, creador de los menús y que asigna a las personas en base a criterios de salud y sus preferencias.
- Usuario **Administrador**: administrador del negocio

En el sistema en un futuro podría habilitarse un tercer tipo de Usuario Chef, que podría encargarse únicamente de la introducción de Recetas.

Permisos de cada tipo de Usuario:

- **Administrador**: dispondrá de todos los permisos de la aplicación

- **Nutricionista**: Tendrá acceso a la parte del panel de administración de la confección de menús, y a la de asignación de menús a usuarios.

- Usuario **cliente**: solo dispondrá de acceso a la parte pública, en la que podrá consultar recetas, suscribirse por un periodo de tiempo y consultar su menú diario.

- Usuario **Anónimo**: dispondrá de la posibilidad de ver los menús semanales y buscar recetas publicadas.

2.2 Casos de uso

Desde la parte pública de la aplicación se podrán realizar las siguientes acciones:

- Consulta de Recetas
- Mostrar Receta
- Suscripción
- Login / Logout
- Área Privada\Consulta de mis Menús
- Área Privada\Modificar datos usuario

Desde la parte privada o panel de administración tendremos la posibilidad dependiendo de nuestro perfil de realizar las siguientes acciones:

- Login / Logout Panel Administración

- Consulta Usuarios
  - Alta
  - Baja
  - Modificación

- Consulta Ingredientes
  - Alta
  - Baja
  - Modificación

- Consulta Platos
  - Alta
  - Baja
  - Modificación

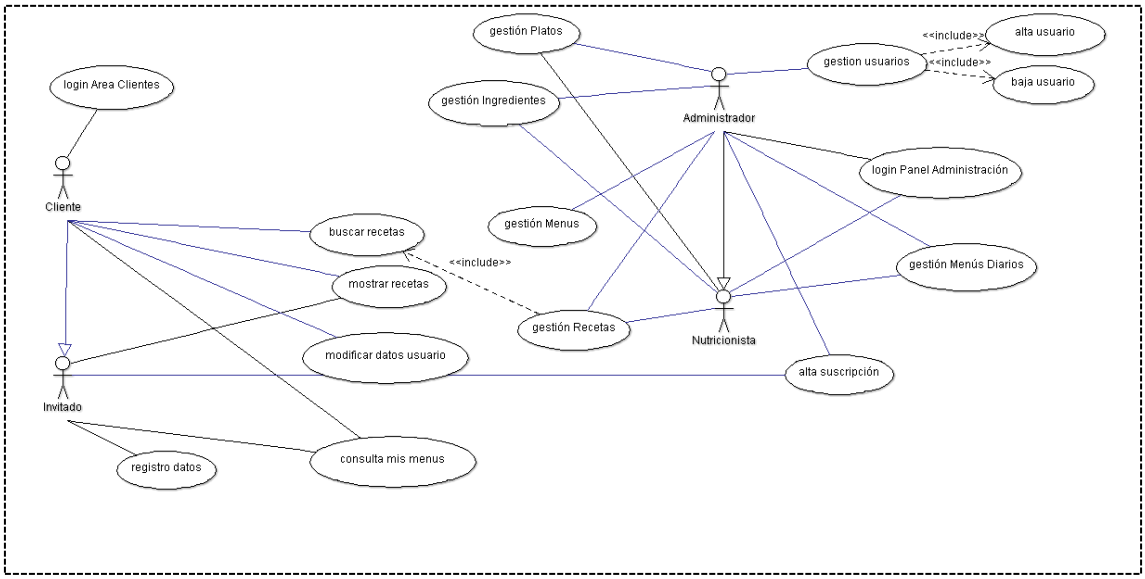
- Consulta Recetas
  - Alta
  - Baja
  - Modificación

- Consulta Menús
  - Alta
  - Baja
  - Modificación

- Consulta Menús Diarios
  - Alta
  - Baja
  - Modificación

- Consulta Suscripciones
    - Alta
    - Baja
    - Modificación

Para simplificar el modelo de casos de uso, se han agrupados en “gestión..” los casos de uso que implican alta, modificación y baja de un elemento del sistema en la sección de administración del aplicativo.



2.3 Descripción de Casos de uso

En cuanto a la parte pública del aplicativo tenemos los siguientes casos de uso.

Caso de uso	Consulta de Recetas
Actor Principal	Usuario Invitado y Usuario Cliente
Precondición	
Postcondición	El sistema muestra las recetas encontradas
Escenario Principal	1. El usuario se conecta a la web sección Recetas 2. El sistema muestra las ultimas recetas incorporadas 3. El usuario especifica tipo y/o clasificación del plato 4. El sistema busca la receta por los criterios
Flujos Alternativos	El usuario puede consultar las recetas encontradas Ejecutando el caso de uso Mostrar Receta

Caso de uso	Mostrar Receta
Actor Principal	Usuario Invitado y Usuario Cliente
Precondición	1. El usuario ha buscado la receta 2. El usuario está mostrando una de las ultimas recetas
Postcondición	
Escenario Principal	1. El usuario se conecta a la web sección Recetas 2. El usuario busca Recetas 3. El usuario hace click sobre cualquiera de las recetas encontradas
Flujos Alternativos	

Caso de uso	Registro de Cliente
Actor Principal	Usuario Invitado
Precondición	El usuario ha seleccionado un tipo de suscripción
Postcondición	El usuario queda registrado como Cliente
Escenario Principal	1. El usuario accede a la web 2. El usuario accede a la sección suscripción 3. El usuario hace click en un tipo de suscripción 4. El usuario rellena sus datos
Flujos Alternativos	

Caso de uso	Suscripción
Actor Principal	Usuario Anónimo y Usuario Cliente
Precondición	El usuario se acaba de registrar
Postcondición	El usuario queda suscrito
Escenario Principal	1. El usuario accede a suscribirse 2. El usuario introduce sus datos 3. El usuario selecciona el tipo de suscripción
Flujos Alternativos	1. El usuario ha hecho login en el sistema.

Caso de uso	Login
Actor Principal	Usuario Anónimo
Precondición	
Postcondición	El usuario queda autentificado en el sistema
Escenario Principal	1. El usuario accede a la sección de login 2. El usuario introduce usuario y password
Flujos Alternativos	1. Si el usuario introduce mal sus credenciales, se le muestra el error y la pantalla de login. 2. Si el usuario introduce bien sus datos puede acceder a la sección de Mis Menús

Caso de uso	Consulta Mis Menús
Actor Principal	Usuario Cliente
Precondición	
Postcondición	El usuario queda autentificado en el sistema
Escenario Principal	1. El usuario accede a la sección de login 2. El usuario introduce usuario y password
Flujos Alternativos	1. Si el usuario introduce mal sus credenciales, se le muestra el error y la pantalla de login. 2. Si el usuario introduce bien sus datos puede acceder a la sección de Mis Menús

Caso de uso	Modificar Datos Usuario
Actor Principal	Usuario Cliente
Precondición	El usuario ha hecho login
Postcondición	Los datos del usuario son modificados
Escenario Principal	1. El usuario modifica sus datos personales 2. El usuario guarda el formulario
Flujos Alternativos	

En cuanto a la parte de administración tenemos los siguientes casos de uso.

Caso de uso	Login AdminPanel
Actor Principal	Usuario Administrador o Usuario Nutricionista
Precondición	El usuario no está autenticado
Postcondición	El usuario queda autenticado en el sistema
Escenario Principal	1. El usuario se conecta a la web a la sección de administración /admin 2. El sistema muestra pantalla de login 3. El usuario introduce sus credencias 4. El sistema chequea que el usuario tiene el Rol Administrador 5. El sistema muestra la pantalla por defecto de administración
Flujos Alternativos	1. Si el usuario no introduce bien sus credenciales se muestra otra vez la pantalla de login. 2. Una vez introducido bien los datos puede acceder a las opciones disponibles para su perfil/rol.

Caso de uso	Listado Usuarios
Actor Principal	Usuario Administrador o Usuario Nutricionista
Precondición	El usuario está autenticado en el sistema
Postcondición	Se muestra el listado de usuarios
Escenario Principal	1. El usuario accede a la sección de usuarios
Flujos Alternativos	1. Si el usuario es Administrador puede acceder a Eliminar Usuario y Modificar Usuario.

Caso de uso	Alta Usuarios
Actor Principal	Usuario Administrador
Precondición	El usuario está autenticado en el sistema
Postcondición	Se da de alta un usuario
Escenario Principal	1. El usuario accede a la sección de usuarios 2. El usuario hace click en el botón de alta 3. El usuario introduce los datos del usuario 4. El usuario guarda el usuario
Flujos Alternativos	1. Si el usuario es Administrador no guarda el Usuario se accede al Listado de Usuarios.

Caso de uso	Modificación Usuarios
Actor Principal	Usuario Administrador
Precondición	El usuario está autenticado en el sistema
Postcondición	Se modifica un usuario
Escenario Principal	1. El usuario accede a la sección de usuarios 2. El usuario hace click en el botón de Editar 3. El usuario modifica los datos del usuario 4. El usuario guarda el usuario
Flujos Alternativos	1. Si el usuario es Administrador no guarda el Usuario se accede al Listado de Usuarios.

Caso de uso	Baja Usuarios
Actor Principal	Usuario Administrador
Precondición	El usuario está en el listado de usuarios
Postcondición	Se elimina un usuario
Escenario Principal	1. El usuario accede a la sección de usuarios 2. El usuario clica en el botón de Eliminar Usuario
Flujos Alternativos	

Caso de uso	Listado Ingredientes
Actor Principal	Usuario Administrador
Precondición	El usuario está autenticado
Postcondición	Se muestra el listado de ingredientes
Escenario Principal	1. El usuario accede a la sección de ingredientes
Flujos Alternativos	Se puede dar de Alta, Eliminar o Editar Ingrediente

Caso de uso	Alta Ingrediente
Actor Principal	Usuario Administrador
Precondición	El usuario está en el listado de ingredientes
Postcondición	Se da de alta el ingrediente
Escenario Principal	1. El usuario accede a la lista de ingredientes 2. El usuario clica en el botón de Alta 3. El usuario introduce los datos 4. El usuario clica en el botón guardar.
Flujos Alternativos	

Caso de uso	Modificación Ingrediente
Actor Principal	Usuario Administrador
Precondición	El usuario está en el listado de ingredientes
Postcondición	Se da de alta el ingrediente
Escenario Principal	1. El usuario accede a la lista de ingredientes 2. El usuario clica en el botón de Edición
Flujos Alternativos	

Caso de uso	Baja Ingrediente
Actor Principal	Usuario Administrador
Precondición	El usuario está en el listado de ingredientes
Postcondición	Se da de baja el ingrediente
Escenario Principal	1. El usuario accede a la lista de ingredientes 2. El usuario clica en el botón de Eliminar
Flujos Alternativos	

Caso de uso	Listado Platos
Actor Principal	Usuario Administrador
Precondición	El usuario esta autenticado
Postcondición	
Escenario Principal	1. El usuario accede a la lista de platos 2. Se muestra el listado de platos
Flujos Alternativos	Si el usuario hace click en el botón Editar se Mostrará el Plato para su modificación. Si el usuario hace click en el botón Eliminar se Eliminará el Plato. Si el usuario hace click en el botón Alta, se mostrará la modificación Plato vacío.

Caso de uso	Eliminar Plato
Actor Principal	Usuario Administrador
Precondición	El usuario está en el Listado de platos
Postcondición	Se elimina el plato y todas sus recetas
Escenario Principal	1. El usuario accede a la lista de platos 2. El usuario hace click en Eliminar Plato
Flujos Alternativos	

Caso de uso	Alta / Modificación Plato
Actor Principal	Usuario Administrador
Precondición	El usuario esta autenticado
Postcondición	Se modifica o crea el plato
Escenario Principal	1. El usuario accede a la lista de platos 2. El usuario hace click en Alta Plato o en Modificar el Plato. 3. Si es Alta Plato el formulario aparece vacio. 4. El usuario rellena los datos 5. El usuario clica en el botón Guardar
Flujos Alternativos	Si el usuario no guarda el plato se redirige al Listado de Platos.

Caso de uso	Listado Recetas
Actor Principal	Usuario Administrador
Precondición	El usuario esta autenticado
Postcondición	Se muestran las recetas del sistema
Escenario Principal	1. El usuario accede a la listado de rectas
Flujos Alternativos	Desde el menú se puede acceder al resto de Listados disponibles. Se puede Eliminar una Receta desde el Listado y se también se puede Modificar.

Caso de uso	Alta / Modificación Receta
Actor Principal	Usuario Administrador
Precondición	El usuario está en el listado de recetas
Postcondición	Se modifica o crea la Receta
Escenario Principal	1. El usuario hace click en el botón Alta o en Modificar. 2. Si es Alta el formulario aparece vacío, si es Modificar aparecen los datos de la Receta. 3. El usuario rellena los datos 4. El usuario hace click en el botón Guardar
Flujos Alternativos	Si el usuario no guarda se redirige al Listado de Recetas.

Caso de uso	Eliminar Receta
Actor Principal	Usuario Administrador
Precondición	El usuario está en el listado de recetas
Postcondición	Se elimina la Receta
Escenario Principal	1. El usuario hace click en el botón Eliminar
Flujos Alternativos	

Caso de uso	Listado Menús
Actor Principal	Usuario Administrador o Usuario Nutricionista
Precondición	El usuario está autenticado
Postcondición	Se muestra el listado de Menús
Escenario Principal	1. El usuario accede a la sección de Menús
Flujos Alternativos	

Caso de uso	Alta / Modificación Menú
Actor Principal	Usuario Administrador o Usuario Nutricionista
Precondición	El usuario está en el listado de menús
Postcondición	Se modifica o crea el Menú
Escenario Principal	1. El usuario hace click en el botón Alta o en Modificar. 2. Si es Alta el formulario aparece vacío, si es Modificar aparecen los datos del Menú. 3. El usuario rellena los datos 4. El usuario hace click en el botón Guardar
Flujos Alternativos	Si el usuario no guarda se redirige al Listado de Menús.

Caso de uso	Eliminar Menú
Actor Principal	Usuario Administrador
Precondición	El usuario está en el listado de menús
Postcondición	Se elimina el Menú y se elimina los Menús Diarios
Escenario Principal	1. El usuario hace click en el botón Eliminar 2. Se debe comprobar que no esté asignado el Menú a ninguna suscripción. 3. Si solo está asignado a un menú Diarios futuro se permite eliminar. 4. Si está asignado a Menús Diarios anteriores a la fecha actual se debe pedir confirmación.
Flujos Alternativos	

Caso de uso	Asignación Menus a Suscripción
Actor Principal	Usuario Administrador o Usuario Nutricionista
Precondición	El usuario está autenticado
Postcondición	Se asigna un Menú a una suscripción en un día
Escenario Principal	1. El usuario accede al Listado de Suscripciones sin menú asignado. 2. El usuario selecciona una suscripción 3. Se muestra el listado de Menús Diarios asignado y por asignar. 4. El usuario selecciona una fecha por asignar y un menú. 5. El usuario guarda la asignación.
Flujos Alternativos	

Caso de uso	Listado Suscripciones
Actor Principal	Usuario Administrador
Precondición	El usuario está autenticado
Postcondición	Se muestra el listado de Suscripciones
Escenario Principal	1. El usuario accede a la sección de Suscripciones
Flujos Alternativos	

Caso de uso	Alta / Modificación Suscripción
Actor Principal	Usuario Administrador
Precondición	El usuario está en el listado de suscripciones
Postcondición	Se modifica o crea la Suscripción
Escenario Principal	1. El usuario hace click en el botón Alta o en Modificar. 2. Si es Alta el formulario aparece vacío, si es Modificar aparecen los datos de la suscripción. 3. El usuario rellena los datos 4. El usuario hace click en el botón Guardar
Flujos Alternativos	Si el usuario no guarda se redirige al Listado de Suscripciones.

2.4 Glosario de términos

A continuación se describen los principales términos y reglas de negocio que ayudan a entender el funcionamiento de la aplicación.

- **Nutricionista**: Tipo de usuario con conocimientos en el mundo de la alimentación y salud, capaz de confeccionar Menús saludables para personas.

- **Administrador**: Administrador o Gestor es la persona propietaria del negocio, generalmente con un perfil más orientado a negocio.

- **Tipo de Suscripción**: Los tipos de suscripción representan ciertas características de la suscripción, para simplificar se ha ligado únicamente a un Tipo de Menú (standard, vegetariano, bajo en calorías, etc.). Pero este concepto podría ampliarse para abarcar por ejemplo el periodo contractual, o cierto extras como podrían ser, numero de menús de regalo, descuentos en el precio, etc.

- **Suscripción**: Es el vínculo contractual que une a un Cliente de Mi Menú de Hoy, y un tipo de Suscripción por un periodo determinado de tiempo.

- **Plato**: Es la representación abstracta de una Receta, que forma parte de un Menú. Pueden ser de tres tipos: primero, segundo o postre.

- **Receta**: Es una de las maneras en la que se puede cocinar un Plato.

- **Menú**: Representa la comida de medio día formada por varios platos, generalmente un primer plato, un segundo plato y un postre.

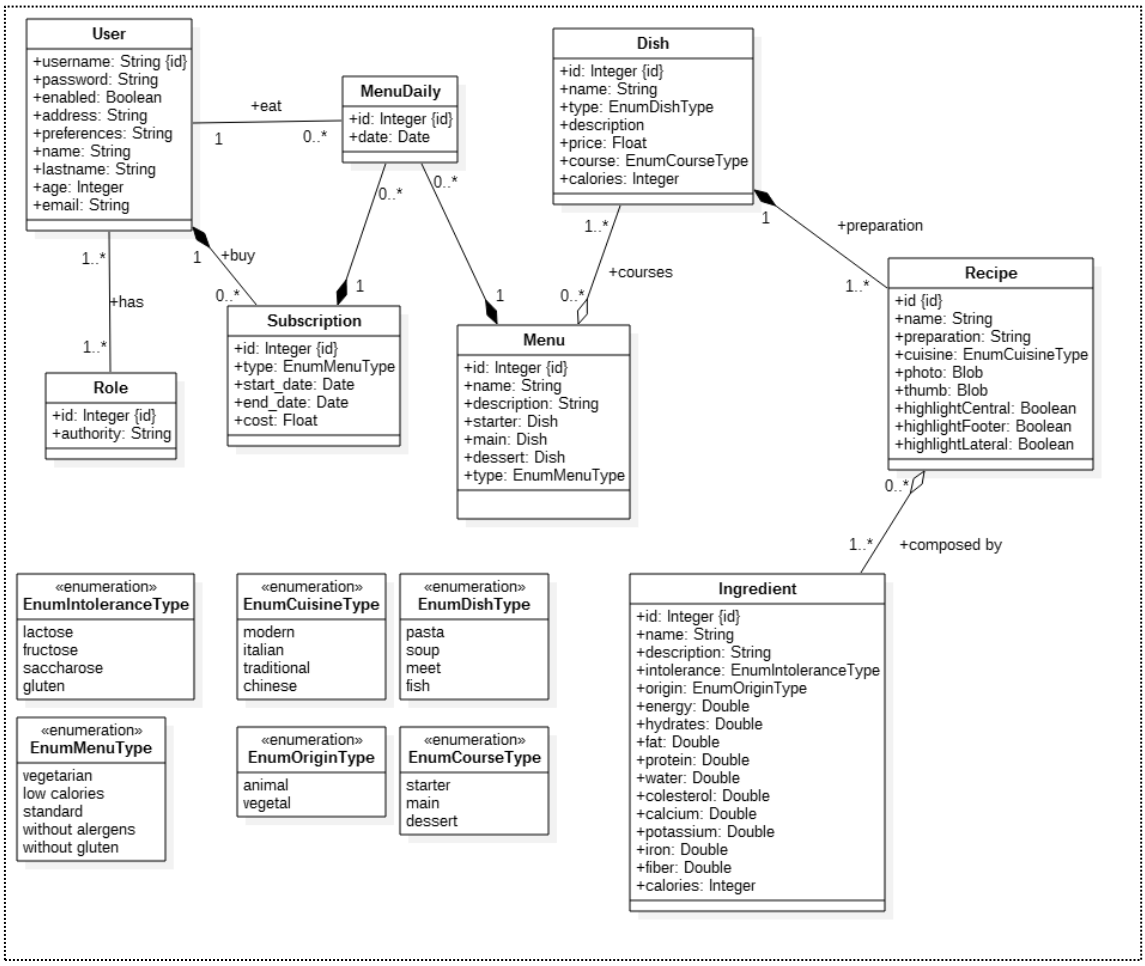
3. Diseño Técnico

Tras la fase de análisis se diseña la solución a implementar, para ello nos ayudará la realización de los siguientes diagramas: Diagrama de clases, Modelo de datos, Diagramas de estados, de actividades y de secuencia.

Se definirá el modelo conceptual en inglés así como todos los elementos del desarrollo para dar una mayor portabilidad del aplicativo en el extranjero.

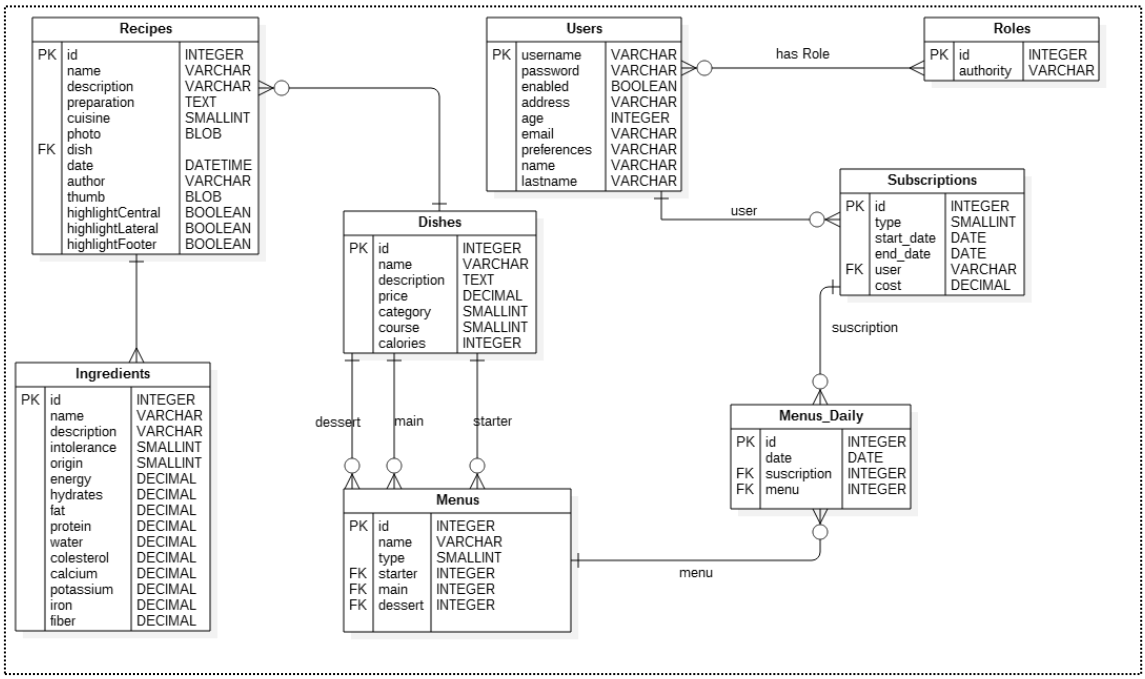
3.1 Diagrama de clases

A continuación se muestra el diagrama de clases definido para el proyecto.



3.2 Modelo de datos

A falta de pequeñas modificaciones o retoques en cuanto a atributos que pueden surgir durante el desarrollo, las entidades principales sobre las que se basará el modelo serán las siguientes.



NOTAS:

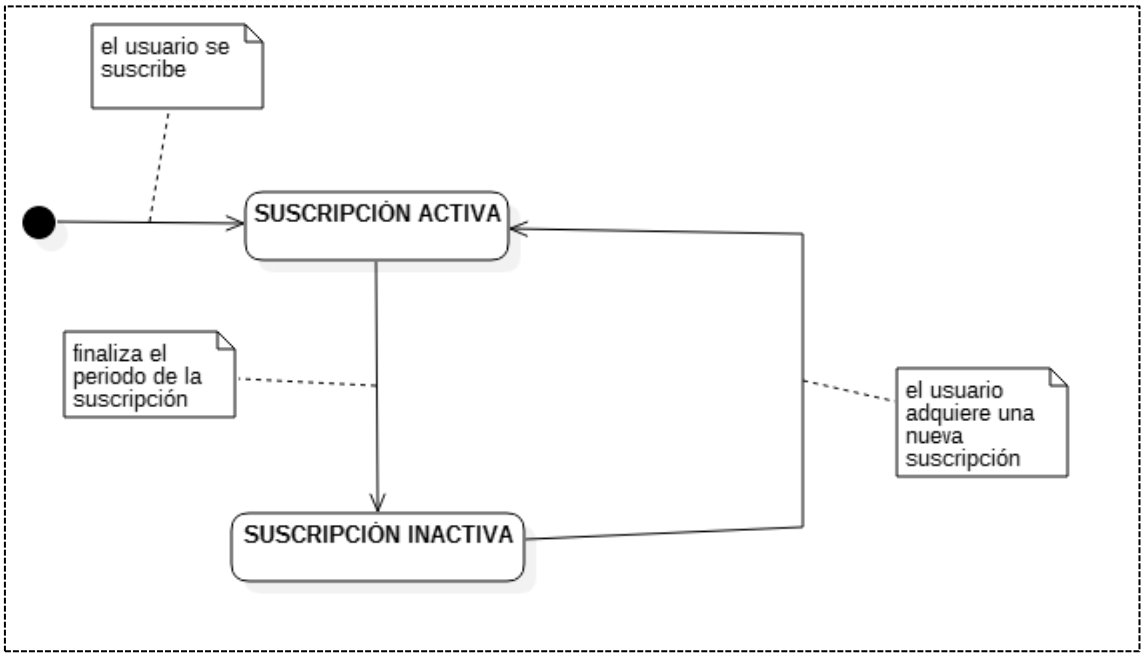
Todos los precios utilizados en el modelo son orientativos.

Los valores energéticos así como los de los componentes de los ingredientes son por unidad de 100 gramos.

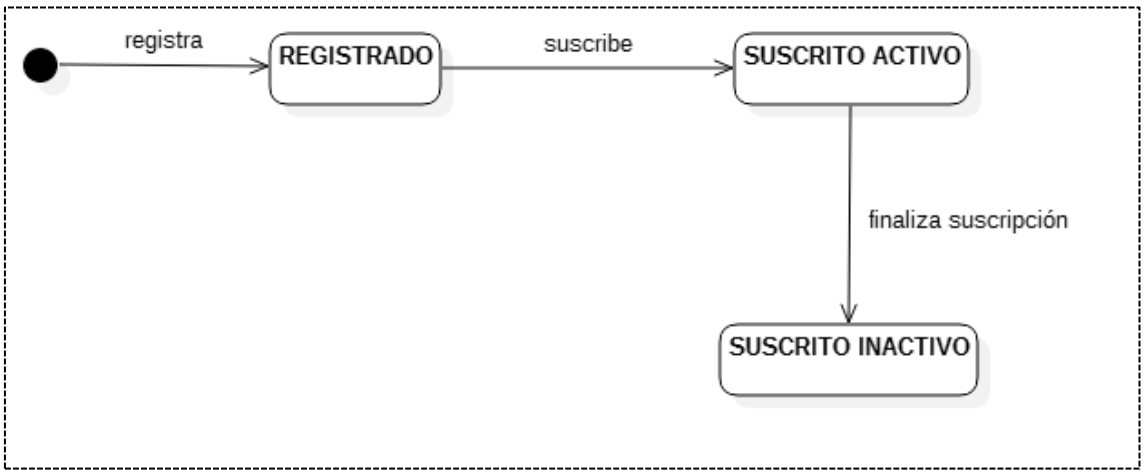
3.3 Diagrama de estado

Este aplicativo no posee un workflow complejo de aprobación por lo que no existen multitud de estados en las entidades principales. En principio se pueden modelar los siguientes cambios de estados:

- Cambios de estado de una suscripción



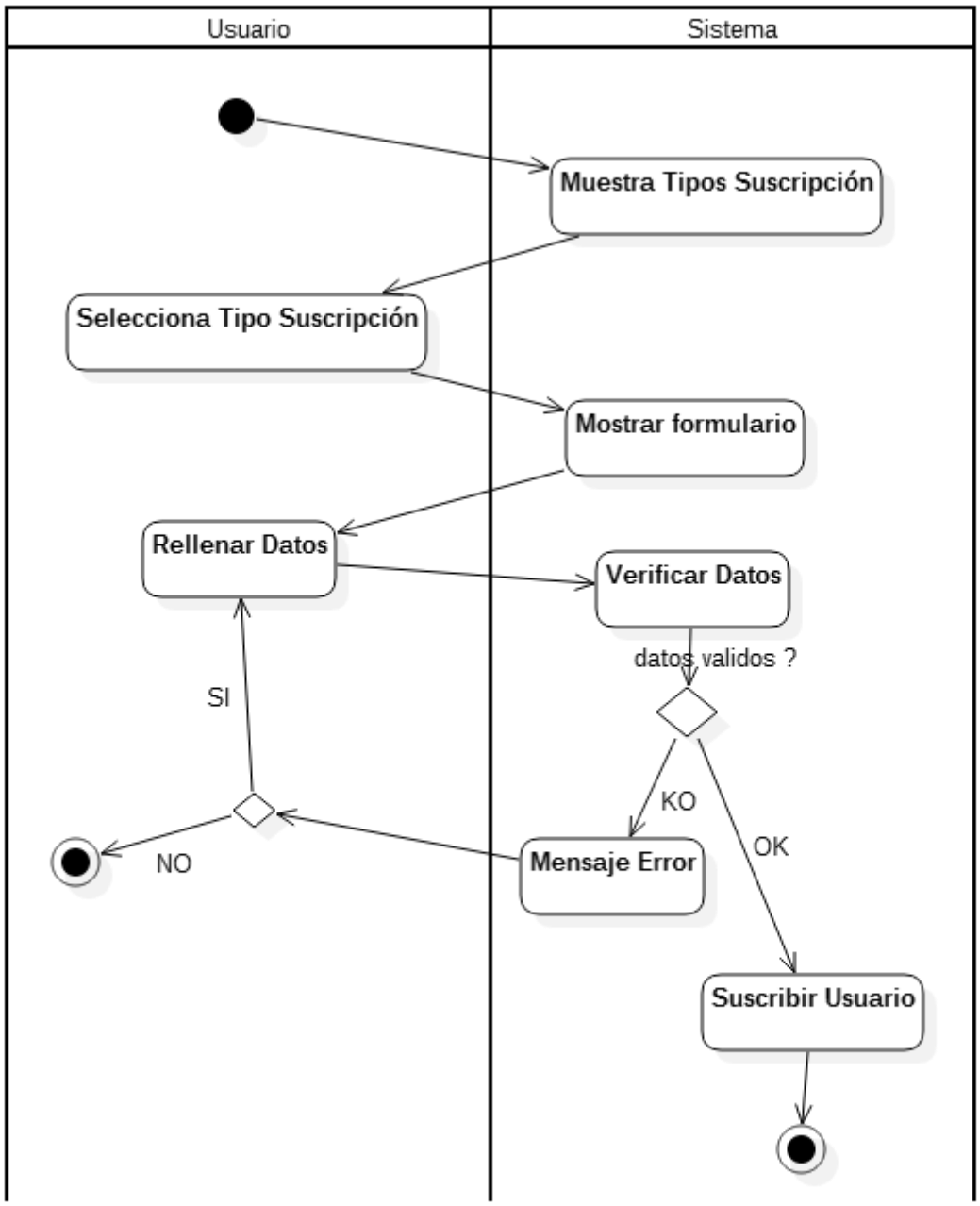
- Cambios de estado de un usuario



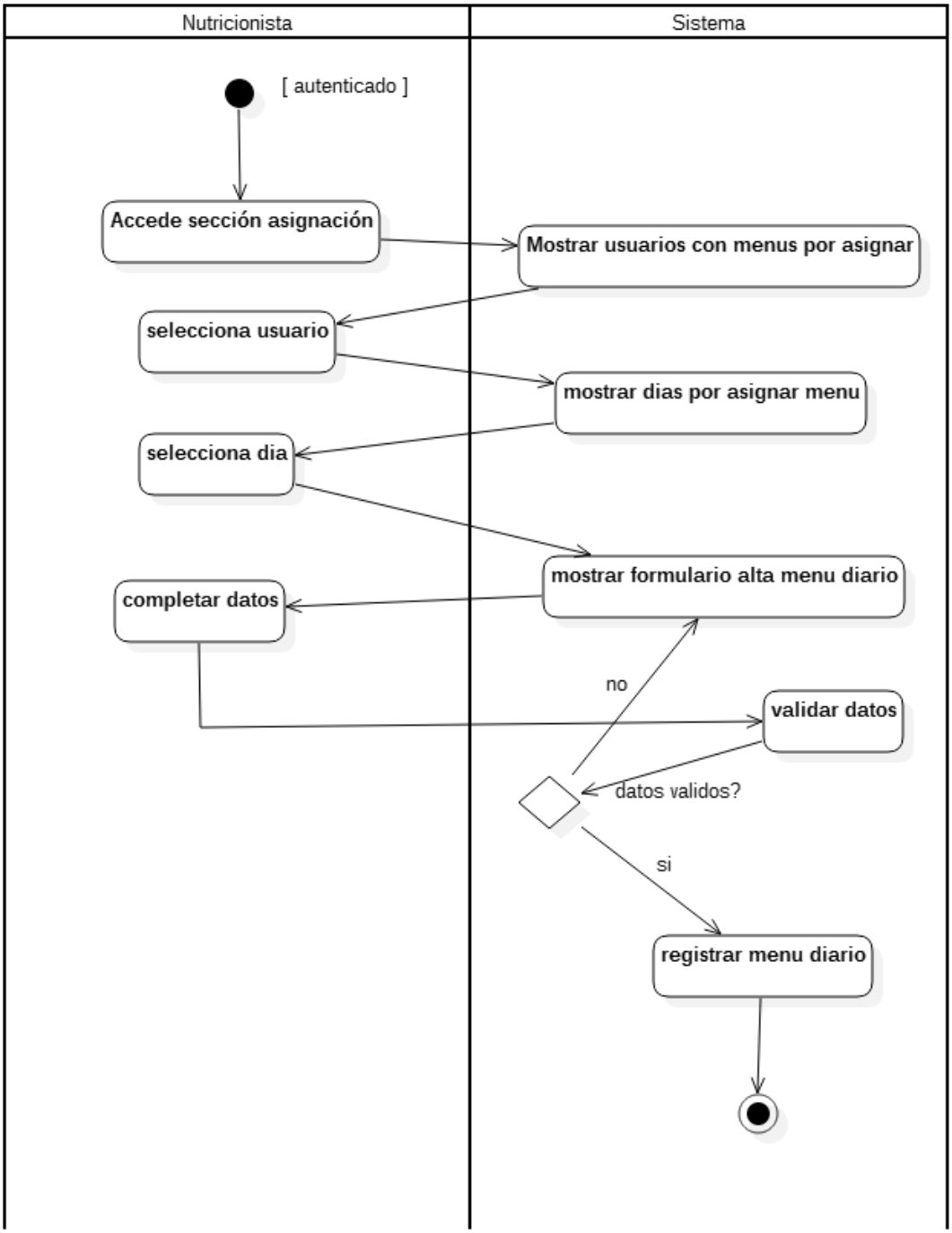
3.4 Diagrama de actividad

Las actividades principales en la aplicación son: Suscripción, y Asignación de Menú a Usuario.

- Suscripción



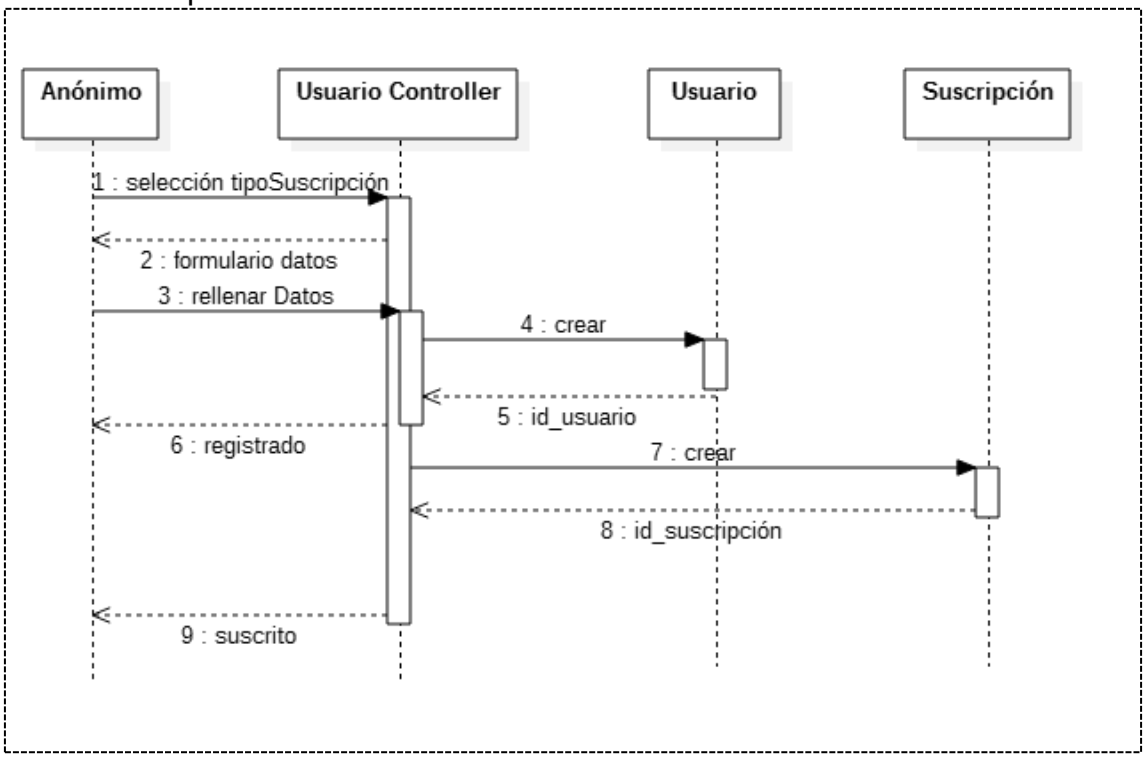
- Asignación de Menú a Usuario



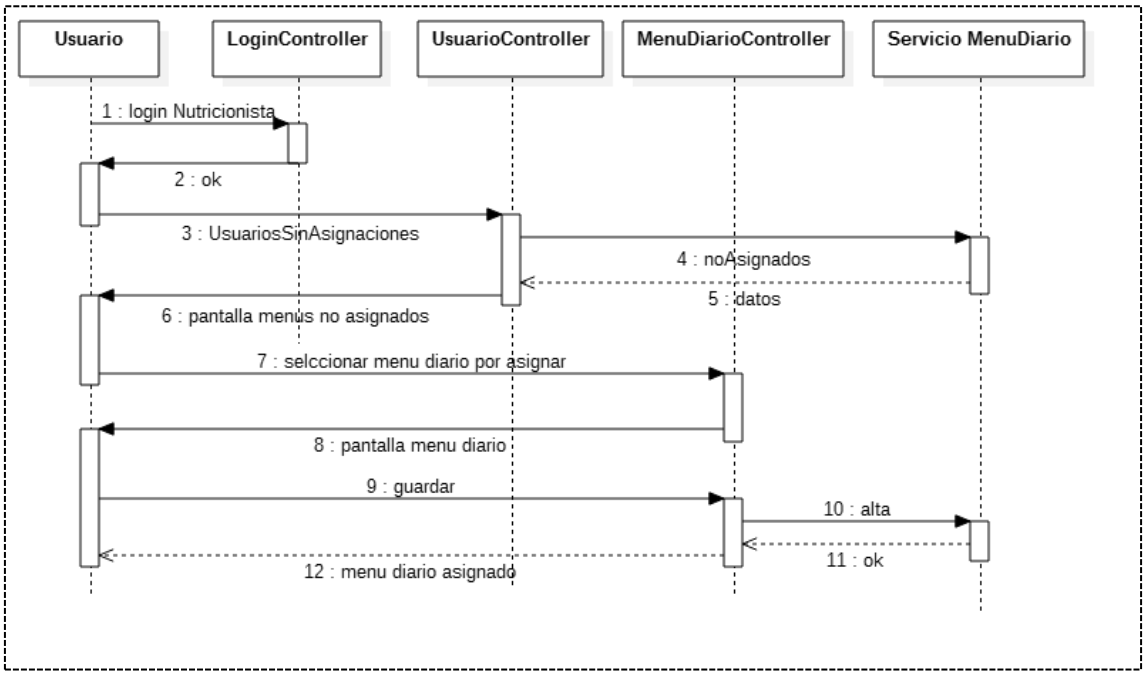
3.5 Diagrama de secuencia

Describimos a continuación los diagramas de secuencia más significativos: Suscripción, y Asignación de Menú a Usuario.

- Suscripción:

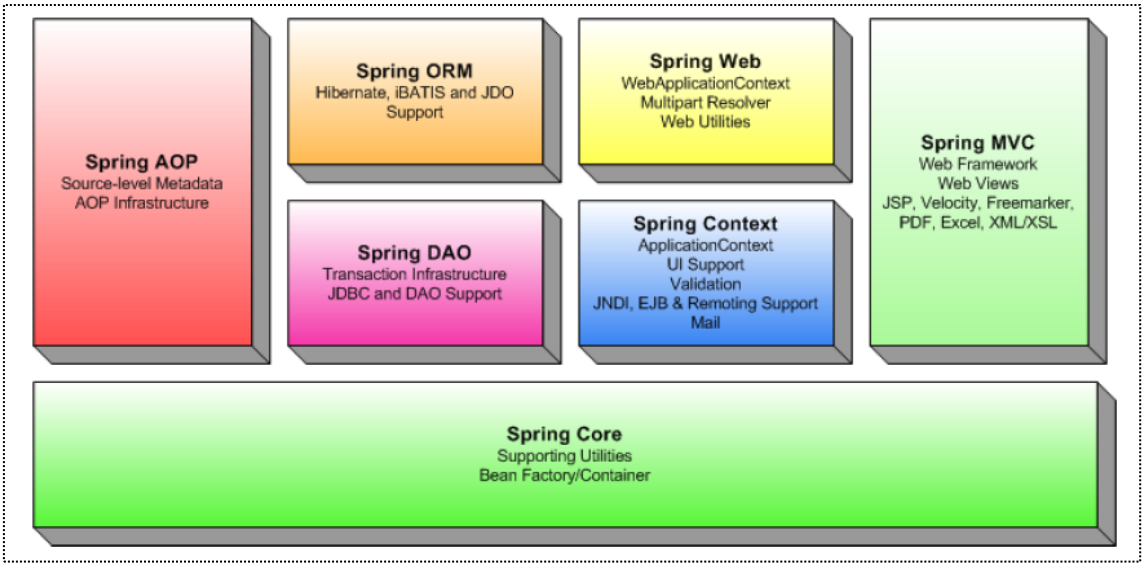


Asignación de Menú a Usuario:

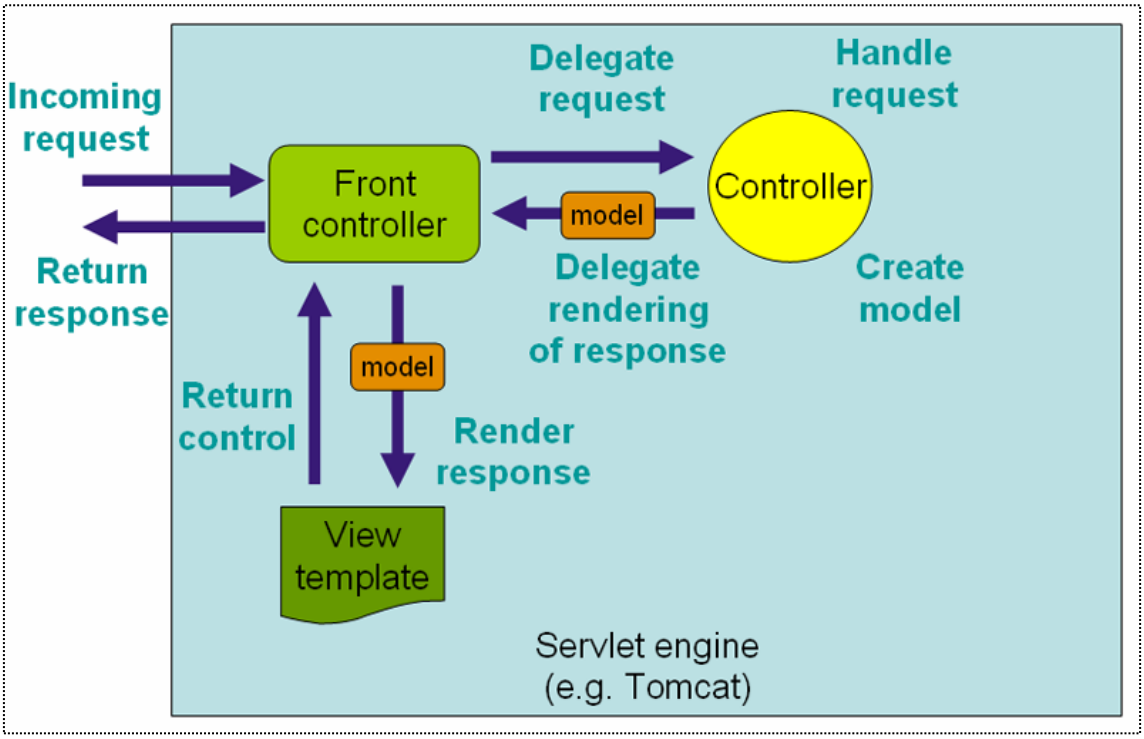


3.6 Arquitectura tecnológica

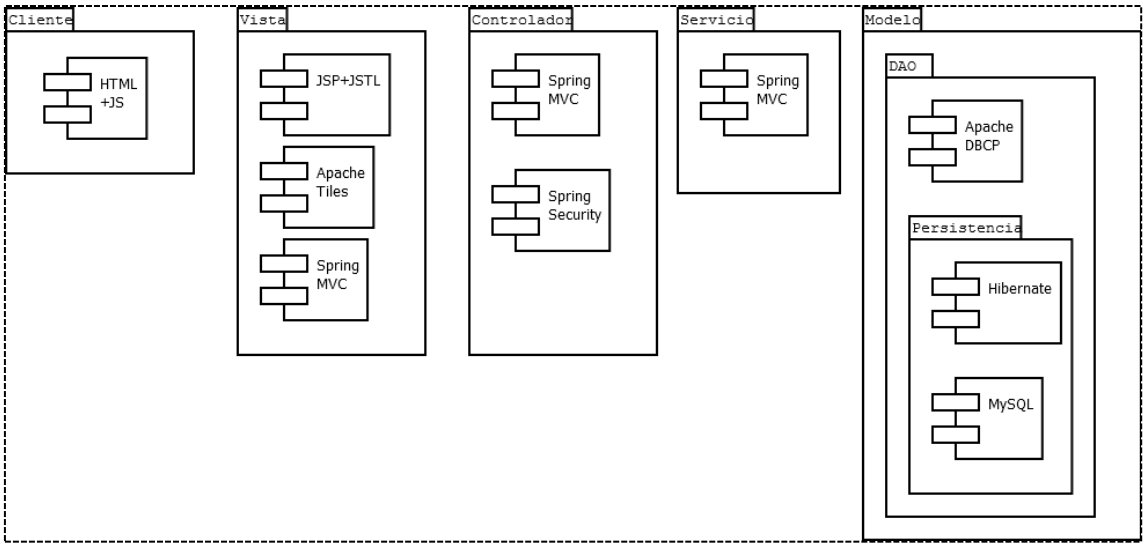
Para el desarrollo de este aplicativo web, y teniendo en cuenta los requisitos iniciales, se ha escogido la arquitectura J2EE para su desarrollo, ya que se adapta perfectamente al mismo, es una tecnología madura y muy extendida por el mercado y a su vez dispone de una gran cantidad de implementaciones del patrón MVC, que utilizaremos en el desarrollo. Una de estas implementaciones es el framework Spring MVC, uno de los más extendidos y modulares del mercado que nos dará mucha flexibilidad a la hora de ampliarlo en un futuro. Éstos son los componentes del ecosistema Spring.



El flujo de las peticiones en Spring MVC es el siguiente:



A continuación mostramos las capas lógicas en las que se compone nuestra aplicación:



- **Modelo**: contiene las reglas de negocio del sistema y conexión con BBDD.
- **Persistencia**: mapeo de objetos a BBDD
- **DAO**: almacena y recupera la información de un motor de base de datos.

- **Servicio**: capa de abstracción del DAO a casos de uso, es opcional, pero se ha desarrollado como previsión de futuras mejoras y para dotar de más flexibilidad en cuanto al acceso datos, como por ejemplo utilizar múltiples conexiones a base de datos con motores diferentes.

- **Vista**: representa el navegador web con el que interactuará el usuario.
- **Controlador**: representa la navegación y la integración entre las capas de vista y modelo.

➢ Cliente: se conecta a internet y accede desde un navegador a la aplicación, sobre éste se ejecuta tanto javascript como html y css.

Entre las principales ventajas de utilizar esta arquitectura multicapa utilizando el patrón MVC encontramos:

➢ Mantenimiento: una modificación en una capa no implica la modificación del resto de capas de la aplicación. Para facilitar esta parte se ha utilizado el sistema de paquetes MAVEN 3.

➢ Escalabilidad: el sistema al utilizar el framework Spring es independiente en cuanto a las tecnologías usadas y es flexible en cuanto a la utilización de otras tecnologías, vía configuración XML o mediante anotaciones JAVA.

- Cliente: No se necesita ningún cliente propietario a instalar en la parte de cliente, únicamente un navegador web ya presenta en la mayoría de las estaciones de trabajo de los usuarios.

- Vista: Al utilizar Apache Tiles, es posible gestionar diferentes tipos de plantillas, y crear diferentes layouts para poder reutilizar el máximo código posible en la parte web, se dividen las pantallas en módulos permitiendo la modificación de cada uno de ellos por separado.

- Modelo: se trabajará siempre sobre objetos (Entity Objects), y estos serán mapeados mediante configuración a Base de datos. La generación inicial de la Base de datos es automática, y tendremos flexibilidad en cuanto a cambiar de una Base de datos a otra, ya que Hibernate posee drivers compatibles con multitud de base de datos. Utilizaremos java Annotations para simplificar los ficheros de configuración del aplicativo.

En cuanto a las tecnologías utilizadas en el desarrollo hemos escogido:

- Java: se utiliza la versión oficial de **J2SE 7.0**, no hemos escogido la última versión para evitar problemas de compatibilidad.

- Controlador: Spring MVC 3.2 como framework principal de la aplicación Web, a éste debemos sumar el módulo de Spring Security que nos proporciona todo lo necesario para implementar la seguridad en nuestra aplicación.

- Vista: Para la generación de las páginas web y componentes se utilizará, en la parte Servidor: JSTL, Apache Tiles 3 y para la parte Cliente, para estandarizar el código para su funcionamiento en diferentes clientes/navegadores se utilizará los frameworks, JQuery 1.10, JQuery UI 1.10.

- Modelo: Se utilizará Hibernate 4 compatible con Java 7 en su versión estable, y además como previsión de uso del proyecto en el ámbito profesional, se ha integrado un pool de conexiones de base de datos, implementado con Apache DBCP v2.1.1.

4. Implementación

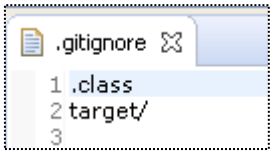
4.1 Consideraciones Iniciales

Como inicio del proyecto, lo primero que hicimos fue seleccionar la plataforma de desarrollo, en este caso utilizamos Eclipse sobre plataforma Windows ya que a nivel estético ofrece mejores características que los IDE de Linux.

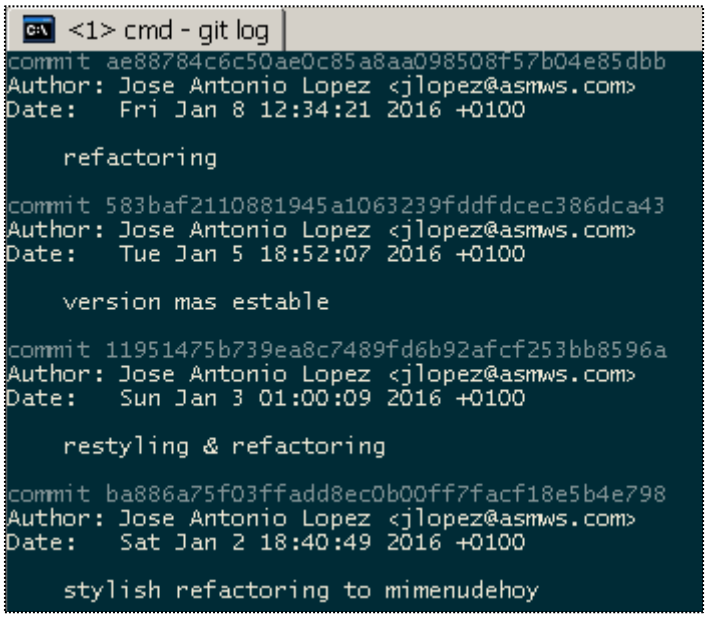
Para dotar de mayor flexibilidad a la hora de gestionar librerías, y empaquetar el proyecto utilizaremos Maven v3.1.

Consideraremos el charset UTF-8 el utilizado para almacenar tanto archivos de código, como de datos en base de datos, para así tener compatibilidad con otros idiomas. Además utilizaremos internacionalización i18n, con lo que dotaremos a nuestra aplicación de la capacidad de ser multiidioma, si bien, en el código solo estará presente el fichero para idioma castellano (messages_es.properties).

Opcionalmente, aunque no menos importante en cualquier desarrollo, hemos instalado un repositorio GIT para evitar tener algún contratiempo durante el desarrollo. Éste es el fichero de ignorar ficheros que hemos utilizado.



De esta manera tendremos controlado el histórico de modificaciones y podremos abordar con mayor fiabilidad futuras mejoras del proyecto.

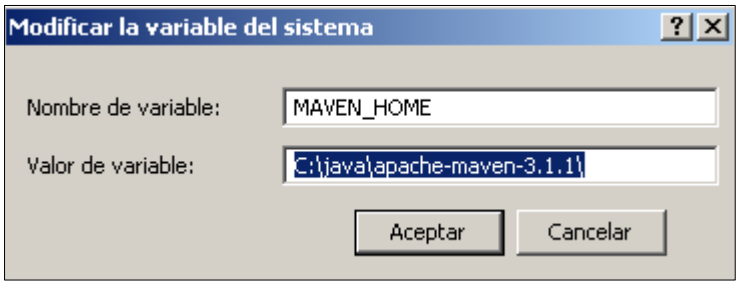


4.2 Preparación del Entorno de Desarrollo

Después de instalar java jdk, eclipse, un mysql local y un tomcat sobre eclipse, necesitamos instalar maven, para ello después descargar e instalar, desde esta ubicación:

http://archive.apache.org/dist/maven/maven-3/3.1.1/

Necesitaremos crear una variable de entorno MAVEN_HOME con el path utilizado para instalarlo, y deberemos incluir esa misma variable en la variable de entorno %PATH%.

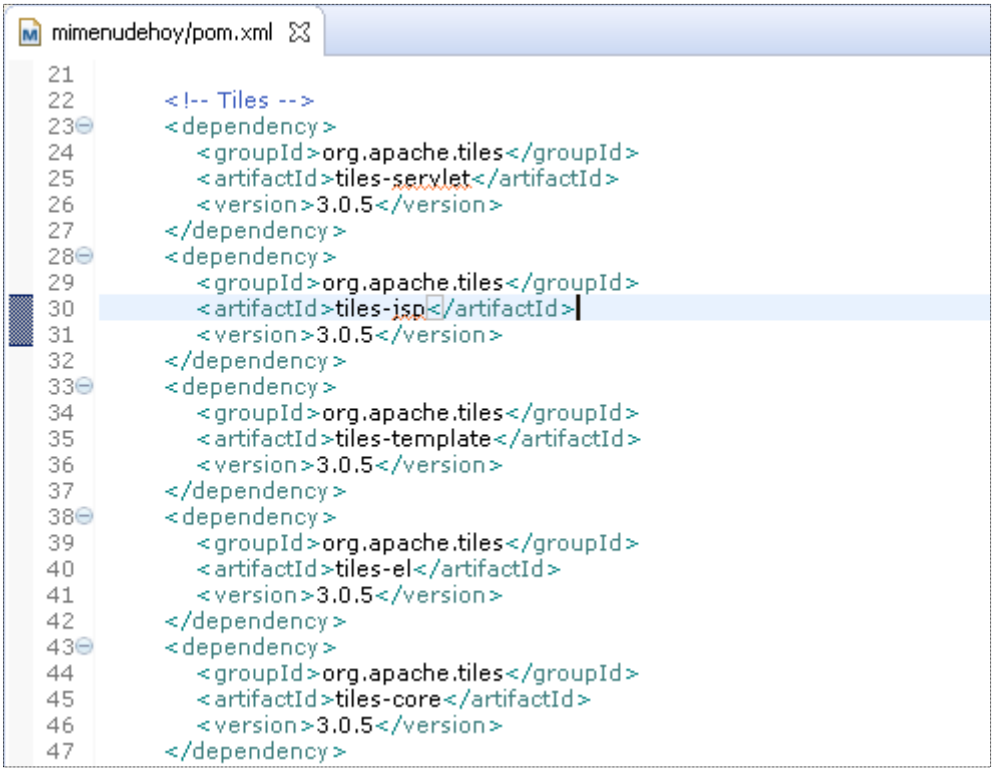


Definiremos nuestro pom.xml para Apache Maven.
Aquí detallo las características más importantes.

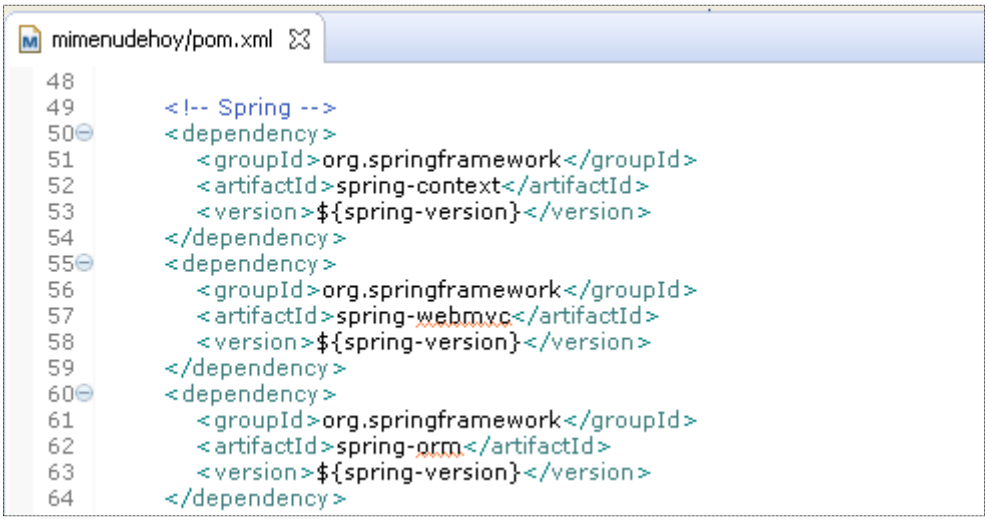
Definimos nombre de proyecto, versión y números de versión de algunos componentes que se repiten.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_1_0.xsd">

  <modelVersion>4.0.0</modelVersion>
  <groupId>uoc.com.mimenudehoy</groupId>
  <artifactId>mimenudehoy</artifactId>
  <name>mimenudehoy</name>
  <packaging>war</packaging>
  <version>1.0.0</version>
  <properties>
    <java-version>1.7</java-version>
    <spring-version>3.2.16.RELEASE</spring-version>
    <spring-security-version>3.1.7.RELEASE</spring-security-version>
    <hibernate-version>4.2.7.Final</hibernate-version>
    <mysql-version>5.1.27</mysql-version>
    <apache-connection-pooling-version2>2.1.1</apache-connection-pooling-version2>
    <log4j.version>1.2.17</log4j.version>
  </properties>
  <dependencies>
```

Para la capa de vista usaremos Apache Tiles 3, incluimos sus dependencias.



Como nuestra aplicación utiliza Spring MVC es indispensable definir sus dependencias.



La implementación de la seguridad la realizaremos con Spring Security, he aquí sus dependencias.

```
88      <!-- Spring security -->
89      <dependency>
90         <groupId>org.springframework.security</groupId>
91         <artifactId>spring-security-web</artifactId>
92         <version>${spring-security-version}</version>
93      </dependency>
94      <dependency>
95         <groupId>org.springframework.security</groupId>
96         <artifactId>spring-security-config</artifactId>
97         <version>${spring-security-version}</version>
98      </dependency>
99
100     <!-- Spring Security JSP Taglib -->
101     <dependency>
102        <groupId>org.springframework.security</groupId>
103        <artifactId>spring-security-taglibs</artifactId>
104        <version>${spring-security-version}</version>
105     </dependency>
106
```

Dependencias del modelo: Hibernate, driver de base de datos mySql, pool de conexiones Apache Database Connection Pooling.

```
<!-- Hibernate -->
<dependency>
   <groupId>org.hibernate</groupId>
   <artifactId>hibernate-core</artifactId>
   <version>${hibernate-version}</version>
</dependency>

<dependency>
   <groupId>org.hibernate</groupId>
   <artifactId>hibernate-entitymanager</artifactId>
   <version>${hibernate-version}</version>
</dependency>

<!-- Apache's Database Connection Pooling -->
<dependency>
   <groupId>org.apache.commons</groupId>
   <artifactId>commons-dbcp2</artifactId>
   <version>${apache-connection-pooling-version2}</version>
</dependency>


<!-- MySQL -->
<dependency>
   <groupId>mysql</groupId>
   <artifactId>mysql-connector-java</artifactId>
   <version>${mysql-version}</version>
</dependency>
```

Otras dependencias necesarias para poder utilizar FileUploads en la carga de imágenes en las recetas.

```
<!-- File Uploading -->
<dependency>
   <groupId>commons-fileupload</groupId>
   <artifactId>commons-fileupload</artifactId>
   <version>1.3.1</version>
</dependency>

<dependency>
   <groupId>commons-io</groupId>
   <artifactId>commons-io</artifactId>
   <version>2.4</version>
</dependency>

<dependency>
   <groupId>commons-codec</groupId>
   <artifactId>commons-codec</artifactId>
   <version>1.10</version>
</dependency>
```

4.3 Descriptor de despliegue (web.xml)

Una vez tenemos resuelto el tema de las dependencias, debemos configurar nuestra aplicación para que arranque, para ello modificamos nuestro web.xml, que nos describe como debe desplegarse la aplicación, de la siguiente manera:

Añadiendo el servlet principal de spring MVC, y especificando los ficheros de configuración que utilizará spring, en este caso hemos separado la configuración de la seguridad ubicada en el archivo spring-security

```
<servlet>
  <servlet-name>mvc-dispatcher</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/mvc-dispatcher-servlet.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>mvc-dispatcher</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>
<listener>
  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>
        /WEB-INF/spring/spring-security.xml
        /WEB-INF/spring/application-context.xml
        /WEB-INF/spring/tiles-context.xml
    </param-value>
</context-param>
```

Nombraremos al servlet principal mvc-dispatcher, y su configuración estará en el fichero mvc-dispatcher-servlet.xml

Para integrar el módulo de seguridad Spring Security, deberemos añadir el listener, HttpSessionEventPublisher e indicar que gestionará todas las urls en el SpringSecurityFilterChain.

```
<listener>
  <listener-class>org.springframework.security.web.session.HttpSessionEventPublisher</listener-class>
</listener>
<filter>
  <filter-name>springSecurityFilterChain</filter-name>
  <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
</filter>
<filter-mapping>
  <filter-name>springSecurityFilterChain</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

Para poder utilizar utf-8 en todo el proyecto deberemos añadir el siguiente filtro.

```
<filter>
  <filter-name>encodingFilter</filter-name>
  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
  <init-param>
    <param-name>encoding</param-name>
    <param-value>UTF-8</param-value>
  </init-param>
  <init-param>
    <param-name>forceEncoding</param-name>
    <param-value>true</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>encodingFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

4.4 Configuración de Controladores

Esta configuración es sencilla y está ubicada en el fichero: mvc-dispatcher-servlet.xml.

```
<annotation-driven />
<context:component-scan base-package="uoc.com.mimenudehoy.controller" />
```

En este fichero indicamos la ruta de paquete de los controllers, y que usaremos Java Annotations en los mismos para simplificar la configuración, ya que a simple vista en el propio controlador podremos mapear urls a métodos del mismo.

```
@RequestMapping(value = { "/recipes" })
public String list(Map<String, Object> map) {

    List<Recipe> recipes = service.getLastRecipes();
    map.put("list", recipes);

    map.put("allCuisineTypes", EnumCuisineType.values() );

    map.put("recipe", new SearchRecipeForm() );
    map.put("menu", "recipes");

    List<Recipe> lateral = service.getHighLightsLateral();
    map.put("recipesLateral", lateral);


    return "recipes";
}
```

El acceso a datos desde los controladores se realiza utilizando la capa de servicios, definida previamente en la arquitectura tecnológica.

```
@Controller
public class SearchRecipeController {

    @Autowired
    private RecipeService service;

    @InitBinder
    public void initBinder(WebDataBinder binder) {
        binder.registerCustomEditor(EnumCuisineType.class, new EnumCuisineTypePropertyEditor());
    }


    @RequestMapping(value = { "/recipes" })
    public String list(Map<String, Object> map) {

        List<Recipe> recipes = service.getLastRecipes();
        map.put("list", recipes);
```

4.5 Acceso a Base de Datos

La configuración se declara en el fichero application-context.xml típico de los frameworks de Spring, en su interior se declaran los siguientes componentes:

- dataSource: gestor de acceso a base de datos.
- sessionFactory: gestor de las sesiones en base de datos.
- transactionManager: gestor de las transacciones, utiliza

```
<bean id="propertyConfigurer" class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
   <property name="locations">
      <list>
         <value>classpath:database.properties</value>
      </list>
   </property>
</bean>

<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
   destroy-method="close" p:driverClassName="${jdbc.driverClassName}"
   p:url="${jdbc.url}" p:username="${jdbc.username}" p:password="${jdbc.password}" />


<bean id="sessionFactory" class="org.springframework.orm.hibernate4.LocalSessionFactoryBean">
   <property name="dataSource" ref="dataSource" />

   <property name="packagesToScan">
      <list>
         <value>uoc.com.mimenudehoy.model</value>
         <value>uoc.com.mimenudehoy.mappings</value>
      </list>
   </property>
   <property name="hibernateProperties">
      <props>
         <prop key="hibernate.hbm2ddl.auto">update</prop>
         <prop key="hibernate.dialect">${hibernate.dialect}</prop>
         <prop key="hibernate.show_sql">false</prop>
         <prop key="hibernate.connection.CharSet">utf8</prop>
         <prop key="hibernate.connection.characterEncoding">utf8</prop>
         <prop key="hibernate.connection.useUnicode">true</prop>
      </props>
   </property>
</bean>

<bean id="transactionManager" class="org.springframework.orm.hibernate4.HibernateTransactionManager">
   <property name="sessionFactory" ref="sessionFactory" />
</bean>

<tx:annotation-driven transaction-manager="transactionManager" />
```

La capa de datos se implementa a nivel de código mediante unas Interfaces DAO que a su vez extienden de una clase genérica GenericDAO que contiene los métodos básicos de obtener la sesión, guardar, borrar y realizar búsquedas por id.

```
package uoc.com.mimenudehoy.dao;

import java.util.List;

public interface GenericDao<T> {

   public T findById(String id);

   public void delete(String id);

   public List<T> list();

   public void save(T entity);

}
```

Con esta implementación simplificamos el mantenimiento y la implementación de los otros DAO.

```
package uoc.com.mimenudehoy.dao;

import java.util.List;

import uoc.com.mimenudehoy.model.Dish;
import uoc.com.mimenudehoy.model.EnumCourseType;

public interface DishDao extends GenericDao<Dish>{

   public Dish findByName(String name);

   public List<Dish> findByType(EnumCourseType tipo);

}
```

4.6 Persistencia en Base de Datos

En cuanto a la persistencia de datos se utiliza Hibernate con Java Annotations en las clases del modelo, de esta manera evitamos los ficheros con extensión .hbm de configuración de Hibernate. Se muestra un ejemplo a continuación.

```
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private Long id;

@Column(length = 255)
private String name;

@Column(length = 512)
private String description;

@NumberFormat(style=Style.NUMBER, pattern="###.##")
private Double price;

@Enumerated(EnumType.ORDINAL)
private EnumDishType category;

@Enumerated(EnumType.ORDINAL)
private EnumCourseType course;

private Integer calories;

 @ManyToMany(cascade=CascadeType.REFRESH)
 @JoinTable(name="dishes_ingredients", joinColumns=@JoinColumn(name="dish_id"), inverseJoinColumns=@JoinColumn(name="ingredient_id"))
private List<Ingredient> ingredients = new ArrayList<Ingredient>();
```

Todas las clases persistidas se encuentran en el package uoc.com.mimenudehoy.model, y tienen la anotación @Entity. Se corresponden con las entidades principales del sistema y son las siguientes:

- Dish / Plato
- Ingredient / Ingrediente
- Menu
- MenuDaily / Menú Diario de una suscripción
- Recipe / Receta
- Role / Rol del usuario
- Subscription
- User

En este mismo paquete también se han incluido objetos nos persistidos, pero con posibilidad en un futuro de persistirse como son:

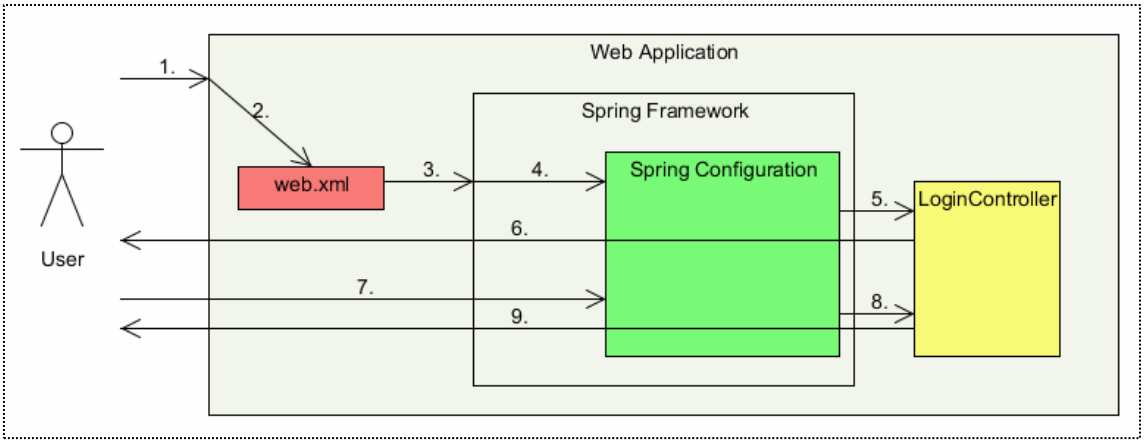
- EnumCourseType / Tipos de Plato (primero , segundo, postre)
- EnumCuisineType / Tipos de cocina
- EnumDishType / Categorización de Platos
- EnumIntoleranceType / Tipos de intolerancias Alimenticias
- EnumMenuType / Tipos de Menús
- EnumOriginType / Orígenes de los ingredientes

Estos objetos han sido definidos como Java5 Enums utilizando clave y valor.

```
package uoc.com.mimenudehoy.model;


public enum EnumOriginType {
   ANIMAL(0),
   VEGETAL(1);

   private int value;

   private EnumOriginType(int value) {
      this.value = value;
   }

   public int getValue() {
      return this.value;
   }

}
```

4.7 Implementación de la seguridad

La seguridad la implementa el módulo Spring Security que opcionalmente se puede integrar con Spring MVC. El flujo de la autenticación es el siguiente:



1. El Usuario accede a la URL de la aplicación
2. La aplicación consulta el web.xml
3. El fichero web.xml coincide con el patron de URL
4. El control es redirigido al *DispatcherServlet* de Spring framework
5. Spring framework detecta que la URL pedida es segura, busca la página de login que le corresponde y redirige la petición al *LoginController* que es un Controlador de Spring MVC
6. El LoginController muestra la página de Custom Login
7. El usuario introduce su usuario y contraseña y envía el formulario de login.
8. Spring realiza la autenticación y autorización de las credenciales del usuario utilizando la configuración de Spring Security, y redirige al *LoginController*
9. *LoginController* muestra la página originalmente solicitada una vez autenticado.

Como vimos en la sección 4.3 el fichero de configuración de la seguridad que hemos utilizado es spring-security.xml. Para la autorización hemos utilizado una implementación sencilla mediante configuración en xml, en él fichero es necesario indicar la query SQL que ejecutaremos para obtener los usuarios, y la query SQL necesaria para obtener los roles de un usuario.

A continuación describiremos sus aspectos principales.

```
<authentication-manager>
  <authentication-provider>
    <jdbc-user-service data-source-ref="dataSource" users-by-username-query="select username, password, enabled from users where username = ?"
      authorities-by-username-query="select user_username as username, roles.authority as authority from users_roles inner join roles on roles.id=roles_id where user_username = ?" />
  </authentication-provider>
</authentication-manager>
```

En este caso utilizamos la tabla Users para almacenar username y password, la tabla roles para almacenar los posibles roles de la aplicación como son: ROLE_CLIENT, ROLE_NUTRI, ROLE_ADMIN. Y la tabla user_roles como tabla de la relación N:N entre usuarios y roles.

Una vez dicho esto, los otros puntos destacables en cuanto a la seguridad en la aplicación, porque requieren de autenticación, serian: el acceso al panel de administración, y el acceso al área privada del usuario. Esto se define en el fichero de la siguiente forma:

```
<!-- LOGIN ADMIN PANEL -->
<http pattern="/admin/**" auto-config="false" use-expressions="true" disable-url-rewriting="true">

    <intercept-url pattern="/admin" access="hasAnyRole('ROLE_ADMIN','ROLE_NUTRI')" />
    <intercept-url pattern="/admin/menu/**" access="hasAnyRole('ROLE_ADMIN','ROLE_NUTRI')" />
    <intercept-url pattern="/admin/user/**" access="hasAnyRole('ROLE_ADMIN')" />
    <intercept-url pattern="/admin/subscription/**" access="hasAnyRole('ROLE_ADMIN')" />
    <intercept-url pattern="/admin/recipe/**" access="hasAnyRole('ROLE_ADMIN')" />
    <intercept-url pattern="/admin/dish/**" access="hasAnyRole('ROLE_ADMIN')" />
    <intercept-url pattern="/admin/ingredient/**" access="hasAnyRole('ROLE_ADMIN')" />

    <form-login
        login-page="/admin_login"
        authentication-failure-url="/admin_login?error"
        username-parameter="username"
        password-parameter="password" />

    <logout logout-success-url="/admin_login?logout" />

</http>
```

Login para el panel de administración que aparecerá siempre que no se tenga el ROL_ADMIN o se acceda a una url que comience por /admin, o bien si se tiene el Nutricionista, éste solo tendrá acceso a la parte de menús, para poder crear nuevos menús o bien asignarlos a suscripciones.

El acceso al área privada en la sección pública de la aplicación se detalla a continuación.

```
<!-- LOGIN WEB -->
 <http auto-config="false" use-expressions="true" disable-url-rewriting="true">

    <intercept-url pattern="/myProfile" access="hasRole('ROLE_CLIENT')" />
    <intercept-url pattern="/myMenus" access="hasRole('ROLE_CLIENT')" />
    <intercept-url pattern="/login" access="isAnonymous()"/>

    <!-- access denied page -->
    <access-denied-handler error-page="/403" />

    <form-login
       login-page="/login"
       authentication-failure-url="/login?error"
       username-parameter="username"
       password-parameter="password" />

    <logout logout-success-url="/login?logout"  />

 </http>
```

Se define como obligatorio tener el rol ROLE_CLIENT para poder acceder a las urls /myMenus y /myProfile, en su ausencia se mostrará la vista mapeada con la url /login.

4.8 Implementación de las vistas

La integración de Apache Tiles se realiza en el fichero tiles-context.xml

Su contenido es el siguiente:

```
<mvc:resources mapping="/res/**" location="/resources/" />
<mvc:default-servlet-handler />

<bean id="viewResolver" class="org.springframework.web.servlet.view.tiles3.TilesViewResolver"/>


<!-- Helper class to configure Tiles 3.x for the Spring Framework -->
<!-- See http://static.springsource.org/spring/docs/3.0.x/javadoc-api/org/springframework/web/servlet/view/tiles2/TilesConfigurer.html -->
<!-- The actual tiles templates are in the tiles-definitions.xml -->
<bean id="tilesConfigurer" class="org.springframework.web.servlet.view.tiles3.TilesConfigurer">
   <property name="definitions">
      <list>
         <value>/WEB-INF/tile-defs/templates.xml</value>
         <value>/WEB-INF/tile-defs/views.xml</value>
      </list>
   </property>
</bean>
```

Con Apache Tiles conseguimos definir una serie de plantillas, e independencia entre el nombre de la vista y su correspondiente archivo jsp, por lo que realizar algún cambio importante a nivel de presentación normalmente se resolvería sin modificar la parte de controlador, solo los ficheros de configuración de las vistas, y la propia vista.

En este fichero a su vez definimos dos configuraciones:

- Templates.xml que contiene la definición de las plantillas principales, una para el panel de administración y otra para la parte pública.
- Views.xml que contiene la definición de las diferentes vistas.

Las dos plantillas principales de la aplicación son:

- adminTemplate: para las pantallas de la parte de administración
- webTemplate: para la parte pública.

```
<definition name="adminTemplate" template="/WEB-INF/template/adminTemplate.jsp">
   <put-attribute name="menu" value="/WEB-INF/views/admin/admin-menu.jsp"/>
   <put-attribute name="content" value="/WEB-INF/template/defaultContent.jsp"/>
</definition>

<definition name="webTemplate" template="/WEB-INF/template/web.jsp">
</definition>
```

Ambas tienen una parte estática, y una parte dinámica, la parte dinámica es el atributo content o body, que varía en cada vista como se puede apreciar en este extracto del fichero de configuración de vistas views.xml.

```
<!-- ADMIN ACCESS -->

<definition name="listUsers" extends="adminTemplate" >
   <put-attribute name="content" value="/WEB-INF/views/admin/user/listUsers.jsp"/>
</definition>

<definition name="userFormDlg" extends="adminTemplate" >
   <put-attribute name="content" value="/WEB-INF/views/admin/user/userForm.jsp"/>
</definition>

<definition name="listRecipes" extends="adminTemplate" >
   <put-attribute name="content" value="/WEB-INF/views/admin/recipe/listRecipes.jsp"/>
</definition>
```

Ejemplo de vistas de la parte pública.

```
<!-- PUBLIC ACCESS -->

<definition name="index" extends="webTemplate">
   <put-attribute name="body" value="/WEB-INF/views/index.jsp"/>
</definition>

<definition name="403" template="/WEB-INF/views/403.jsp"/>

<definition name="recipes" extends="webTemplate" >
   <put-attribute name="body" value="/WEB-INF/views/recipes/listRecipes.jsp"/>
</definition>

<definition name="viewRecipe" extends="webTemplate" >
   <put-attribute name="body" value="/WEB-INF/views/recipes/recipe.jsp"/>
</definition>
```

Dentro de cada vista, hemos incluido siempre una serie de librerías disponibles para nuestra con

includes.jsp

```
1  <%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>
2  <%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
3  <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
4  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
5  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
6  <%@ taglib uri="http://tiles.apache.org/tags-tiles" prefix="tiles" %>
7  <%@ taglib prefix="sec" uri="http://www.springframework.org/security/tags" %>
8
9  <c:set var="baseURL" value="${pageContext.request.contextPath}"/>
10
```

5. Manuales

5.1 Requerimientos de Software

Los requerimientos de software necesarios para el correcto funcionamiento de la aplicación son los siguientes:

- Java 7 para la compilación del proyecto, si bien el proyecto es compatible con Java 1.5 o superior.

  http://www.oracle.com/technetwork/es/java/javase/downloads/jre7-downloads-1880261.html

- Apache Tomcat 7 como servidor de aplicaciones para el entorno de desarrollo. En el entorno de producción recomendaría utilizar Apache como servidor web y utilizando el módulo mod_jk conectar con el contenedor java Apache Tomcat.

  https://tomcat.apache.org/download-70.cgi

  Para los deploys seria de ayuda instalar el aplicativo tomcat-manager que viene por defecto.
  Nota: esto requiere de la configuración de un usuario con rol [manager] en Tomcat.

- El proyecto está desarrollado con IDE eclipse, por lo que se podría importar en él, aunque el montaje recomendado es utilizando Apache Maven 3.1.1.

  http://archive.apache.org/dist/maven/maven-3/3.1.1/binaries/

  Para realizar los deploys en producción, en principio, se debe generar un archivo .war generado con maven y se cargarían manualmente utilizando la aplicación tomcat manager. En su defecto se podría utilizar algún plugin de maven que hace el deploy automático.

- Base de datos: MySql 5.1 o superior.
  Inicialmente por el pequeño volumen de datos no haría falta instalar el servidor de BBDD en un host diferente, por lo que lo instalaría donde estuviera instalado el servidor Tomcat.

  http://downloads.mysql.com/archives/get/file/mysql-5.1.72-winx64.msi

- Sistema operativo Linux distribución Ubuntu 14.04 LTS recomendado para los servidores tanto web como de Base de datos.

  http://releases.ubuntu.com/14.04/

5.2 Procedimiento de instalación

La instalación de toda la plataforma se puede dividir en tres partes:
Instalación de la Base de datos, Instalación del servidor de aplicaciones e Instalación de la aplicación.

- Instalación de la Base de Datos MySQL

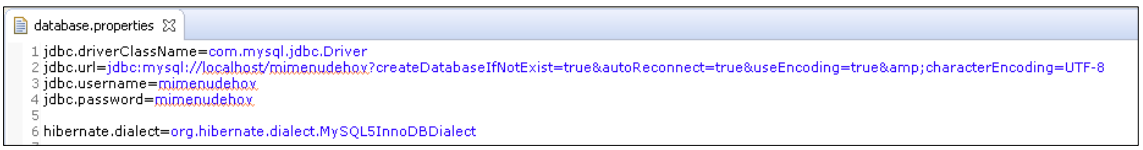
La instalación de la Base de datos resulta sencilla, ya que simplemente se trata de descargar la aplicación, e instalarla para servidores Windows

http://dev.mysql.com/downloads/mysql/

O bien para servidores Linux seria instalar el paquete mysql-server, como se detalla en esta web.

https://help.ubuntu.com/12.04/serverguide/mysql.html

Una vez instalada la BBDD, deberemos crear un usuario para la aplicación, en este caso, la aplicación por defecto viene configurada con el usuario: mimenudehoy password:mimenu de hoy, como se puede ver en el fichero database.properties.



El usuario utilizado debe tener permisos en la base de datos, para asignárselos debemos ejecutar el siguiente comando en una consola mySql:

```
CREATE USER mimenudehoy IDENTIFIED BY ‘mimenudehoy’;
GRANT ALL PRIVILEGES ON mimenudehoy.* TO mimenudehoy;
```

- Instalación del Servidor de Aplicaciones Apache Tomcat

Previamente necesitaremos tener instalado JDK7 ya que utilizamos Java7 como nuestro lenguaje, y Tomcat es compatible con esta versión.
Para ello deberemos instalarlo desde la página oficial de Oracle

http://download.oracle.com/otn-pub/java/jdk/7u79-b15/jdk-7u79-windows-x64.exe

Y deberemos configurar la variable de entorno JAVA_HOME:

Deberemos instalar el servidor de aplicaciones Tomcat7, para Windows descargable desde:
https://tomcat.apache.org/download-70.cgi

- Instalación de la aplicación

La aplicación se entregará empaquetada en un archivo **mimenudehoy.war**, generado con Maven mediante el comando: “mvn package” y un archivo **import.sql** inicial con la estructura básica de la base de datos. Este SQL es necesario simplemente para la importación básica de los Roles, ya que la configuración de hibernate recrea todo el modelo de datos.

Los pasos a seguir para instalar la aplicación:

1- Importar el fichero mimenudehoy.initial.sql en mysql, utilizar el schema que el mismo fichero define.

   Con esto tendremos generada la Base de Datos prácticamente vacía, y solo con la tabla ROLES, USER_ROLES y USERS rellena, así como un usuario admin con password:r00t con acceso al panel de administración.

2- Deployar en nuestro servidor Tomcat el war mimenudehoy.war, éste al arrancar verificará que exista la Base de datos.

   Con esto tendremos arrancada la aplicación, tanto la parte pública como la privada.

6. Valoración Económica

A partir de los casos de uso a desarrollar, y sin tener en cuenta el tiempo de aprendizaje o formación, considerando unas tarifas aproximadas al mercado, por perfil hemos estimado lo siguiente para este proyecto:

Perfil	**Tarifa**	**Horas**	**Precio**
Analista	50€	32	1.600€
Programador	40€	140	5.600€
Jefe de Proyecto	90€	24	2.160€
Maquetador	30€	40	1.200€
			10.560€

7. Conclusiones

Después de volver desarrollar con tecnología J2EE pasados unos años, en los que he estado alternando otras tecnologías (PHP, LotusNotes, Scala) he visto que se han mejorado algunos aspectos como la simplificación de las configuraciones, la velocidad, la recarga, la mejora de la api, si bien el principal problema en mi opinión, de este tipo de proyecto es la integración, el hacer que funcione todo. En ese sentido hay otras soluciones en PHP por ejemplo Symfony2 que son más sencillas de implementar y traen menos dolores de cabeza, pero la flexibilidad que te ofrece esta arquitectura utilizando Spring, su estabilidad, madurez, seguridad, y la amplia comunidad de desarrolladores no tiene comparación. Esta solución tiene una curva de aprendizaje más empinada que otras tecnologías, pero su versatilidad la hace ideal para proyectos de mediana y gran envergadura. Además de ser una solución escalable, modular, multiplataforma y OpenSource con lo que esto supone en aspectos económicos.

Personalmente no había tenido contacto con SpringMVC y no me ha resultado muy difícil empezar a utilizarlo, aunque seguramente no haya usado la totalidad de su potencial. Pero el uso de anotaciones simplifica y ayuda a la comprensión del código para posteriores modificaciones.

Una parte que me resultó compleja fue sobretodo la parte del mapeo de entidades con Hibernate, los mapeos de relaciones generalmente, con tablas intermedias y la configuración para que éste cuadre con el modelo de datos establecido. También la implementación de la seguridad me supuso un tiempo importante, que había subestimado en la valoración, pero al tener dos puntos de entrada diferentes me dio problemas. Otra parte que me hubiera gustado desarrollar son los tests unitarios pero por falta de tiempo no los he podido implementar, y las pruebas que se han realizado han sido pruebas funcionales a nivel de QA.

8. Bibliografía

- http://biblioteca.uoc.edu/ca/
- http://stackoverflow.com/
- http://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm
- http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html
- http://www.mkyong.com/tutorials/spring-mvc-tutorials/
- http://www.mkyong.com/tutorials/hibernate-tutorials/
- http://spring.io/docs
- https://maven.apache.org/guides/index.html
- http://howtodoinjava.com/2014/10/27/complete-hibernate-query-language-hql-tutorial/
- https://docs.jboss.org/hibernate/orm/3.3/reference/en/html/queryhql.html
- http://howtodoinjava.com/2012/11/17/hibernate-one-to-many-mapping-using-annotations/
- http://www.freewebsitetemplates.com/
- http://purecss.io/
- https://jquery.com/
- http://community.tinymce.com/forum/viewforum.php?id=1

- Rumbaugh, James; Jacobson, Ivar; Booch, Grady (1999). «The Unified Modeling Language Manual».
  Addison Wesley Longman, Inc.