



ugr

Universidad
de Granada

TRABAJO FIN DE MÁSTER
MÁSTER UNIVERSITARIO EN INGENIERÍA INFORMÁTICA

Sistema de Información Web en tiempo real para uso en ponencias y reuniones

Autor

Francisco Serrano Carmona

Tutores

Pedro Ángel Castillo Valdivieso

Pablo García Sánchez



Escuela Técnica Superior de Ingenierías Informática y de
Telecomunicación

—
Granada, Septiembre de 2015

Sistema de Información Web en tiempo real para uso en ponencias y reuniones

Francisco Serrano Carmona

Palabras Clave: Sistema de Información, Ponencias, Modelo-Vista-Controlador, Proyecto Web, Java Server Faces, Desarrollo, Websockets, MongoDB, NoSQL, Tiempo Real, Diseño, Análisis.

Resumen

En este documento se expone el análisis, desarrollo e implementación de un sistema de información basado en tecnologías web para la gestión en tiempo real de datos sobre estados de ánimo de los asistentes a una ponencia o presentación. Se explica: qué metodología se ha seguido para desarrollar el sistema, qué herramientas y tecnologías se han utilizado para afrontar los requisitos del sistema, cómo se han diseñado e implementado los diferentes componentes del sistema, cómo se usa el sistema implementado y qué beneficios ha supuesto el uso del sistema para sus usuarios.

En este trabajo se pretenden explorar diferentes técnicas de análisis, desarrollo e ingeniería de software, que uniéndolas darán como resultado un sistema robusto y adaptado a las necesidades de quienes van a usarlo. Se utilizarán tecnologías y metodologías eficaces que para ayudar al análisis, desarrollo e implementación del sistema de información, tales como: uso del patrón de arquitectura de software de Modelo-Vista-Controlador[23]; uso de modelos entidad-relación [7] y modelo relacional [6] para el diseño de la base de datos; uso de tecnologías avanzadas como WebSockets [27] y bases de datos NoSQL [16]; uso de Java Server Faces [9] como tecnología para el desarrollo.

También se explicará la implementación del proyecto: qué jerarquía de páginas tiene el sistema; qué agrupación de funcionalidades tiene el código del sistema; cómo puede instalarse el sistema en un entorno de producción; cómo se interactúa con el sistema.

Como resultado se pretende obtener un sistema que facilita y enriquece las presentaciones, reuniones o ponencias donde se utilice el programa. Permite a la persona que de la ponencia poder adaptar la presentación a los estados de ánimo del público, además de analizar dichos datos tras la ponencia. También permite responder a preguntas que lance el ponente en tiempo real, dando más herramientas al ponente para cumplir las expectativas de la ponencia con el público.

Web Information Systems to use in Presentations and meetings in real time

Francisco Serrano Carmona

Keywords: Information Systems, Presentations, Model-view-controller, Web project, Java Server Faces, Development, Websockets, MongoDB, NoSQL, Real time, Design, Analysis

Abstract

This paper presents the analysis, development and implementation of an Information system based in web technologies used for real time data management of the mood of those attending to a presentation or meeting. In it we expose: the methodology followed to develop the system, the tools and technologies used to meet the system requirements, how the different components of the System have been designed and implemented and which benefits has brought the use of the System for its users.

This paper has explored different analysis, development and software engineering techniques; and by joining them together they have resulted in a robust System, well adapted to the needs of its users. We have used efficient technologies and methodologies that helped the analysis, development and implementation of the information system, such as: use of standard software architecture *Model-View-Controller* [23], use of *entity-relation* models [7] and relational model [6] in order to design data bases; use of advanced technologies such as *WebSockets* [27] and *NoSQL* data bases [16], and use of Java Server Faces [9] as development technology.

Project implementation is also explained: the page hierarchy of the system, the group of features of the system code, how can a system be installed in a production environment and how to interact with the system.

As a result, we have obtained a system that helps and enrich presentations, meetings or talks where this program is used. It allows the speaker to adapt the presentation to the audience mood. In addition, it analyses these data after the presentation. It also helps to answer the questions the speaker makes in real time, providing him tools to meet the expectations of the meeting with the audience.

Índice de Contenido

1.- Introducción.....	5
1.1.- Objetivo del Proyecto.....	5
1.2.- Actores implicados en el sistema.....	5
1.2.1.- Ponente.....	5
1.2.2.- Público.....	6
1.3.- Relación entre actores y funcionamiento del sistema.....	6
2.- Metodología.....	8
2.1.- Metodología de obtención de requisitos.....	8
2.2.- Metodología de Ingeniería del Software.....	9
2.3.- Metodología de Diseño de la Base de Datos.....	10
2.4.- Objetivos y Requisitos Funcionales.....	11
2.5.- Tecnologías a utilizar.....	12
2.5.1.- Java.....	12
2.5.2.- Bibliotecas y frameworks usados en Java.....	13
2.5.3.- Software del servidor.....	14
2.5.4.- Modelo-vista-controlador.....	15
3.- Diseño del sistema.....	16
3.1.- Diseño de la base de datos.....	16
3.1.1.- Relación de Requisitos Funcionales con Entidades.....	16
3.1.2.- Modelo Entidad/Relación.....	17
3.1.3.- Modelo Relacional.....	17
3.1.4.- Modelo NoSQL.....	20
3.2.- Diseño del proyecto.....	21
3.2.1.- Paquetes del proyecto.....	21
3.2.2.- Jerarquía de archivos.....	23
3.2.3.- Mapa del sitio.....	24
3.2.4.- Diseño de Mensajes.....	24
3.2.5.- Diseño de Vistas.....	25
4.- Conclusión.....	31
4.1.- Uso del sistema en un entorno real.....	31
4.2.- Líneas futuras.....	32
5.- Bibliografía.....	33
Anexo I - Guía de Instalación.....	35
Configuración de GlassFish y MySQL en un servidor.....	35
Configuración de MongoDB en un servidor.....	37

1.- Introducción

En esta sección se expondrá un análisis de los elementos más relevantes del sistema: el ámbito del sistema (a qué está destinado el sistema), actores del sistema (qué entes interaccionarán con el sistema) y las relaciones de los actores en el sistema (cómo se comunica cada actor entre los otros actores para que queden estas comunicaciones reflejadas en el sistema).

1.1.- Objetivo del Proyecto

Se pretende diseñar un sistema de información basado en tecnologías web para la gestión en tiempo real y análisis posterior de los sentimientos y reacciones de un público frente a una ponencia o presentación, además de proporcionar herramientas entre quien da la ponencia y el público: gestión de preguntas, seguimiento de las secciones de la ponencia, etcétera.

Este sistema se puede generalizar como una forma de conseguir información de forma anónima e instantánea desde entidades relativamente pasivas de un evento informativo (como una presentación, una ponencia, un programa audiovisual en directo, una reunión laboral, etcétera) hacia el sujeto activo (la persona que presenta o expone el evento).

Con este proyecto la persona que imparte la ponencia puede captar de forma rápida cómo se siente el público al que se está dirigiendo (si están atendiendo, si se están aburriendo, etcétera) y así poder modificar la presentación para adaptarse a lo que pida el público. Además, al ser comunicación de forma anónima, el público puede sentirse más cómodo de comunicar sus sentimientos de forma sincera.

1.2.- Actores implicados en el sistema

A continuación se exponen los actores presentes en el sistema de información. Estos actores han sido identificados tras un análisis de los escenarios donde puede usarse la aplicación desarrollada, identificando las características y necesidades de cada tipo de actor.

1.2.1.- Ponente

Ponente es la persona que impartirá la presentación. Es la encargada de administrar la presentación (el tiempo que durará, las preguntas que se harán, con qué secciones contará la presentación, qué segmentos de público son interesantes para analizar, etcétera). Se espera que sea una persona con conocimiento a nivel usuario de internet y aplicaciones informáticas, que cuando imparta la ponencia disponga de un dispositivo de tamaño mediano con capacidad de conexión a internet (una *tablet* o portátil).

El ponente necesitará estar pendiente de dar la ponencia, por lo que la información a mostrar sobre la ponencia en el dispositivo deberá de ser clara y concisa, para que el ponente pueda analizar los datos de forma rápida mientras está exponiendo. Por lo que la información a mostrar al ponente durante la ponencia deberá de ser clara y concisa.

También se espera que la persona que imparta la ponencia tenga un acceso a los datos recogidos por el sistema para analizarlos a posteriori, por lo que será necesario un registro de dicha persona en el sistema.

Es posible que varios ponentes den una ponencia al mismo tiempo, por lo que el sistema debe de ser capaz de gestionar en tiempo real dicha situación.

Puede ser interesante para los ponentes conocer cierta información del público, por lo que el sistema debe de ser capaz de suministrar la información necesaria al ponente.

1.2.2.- Público

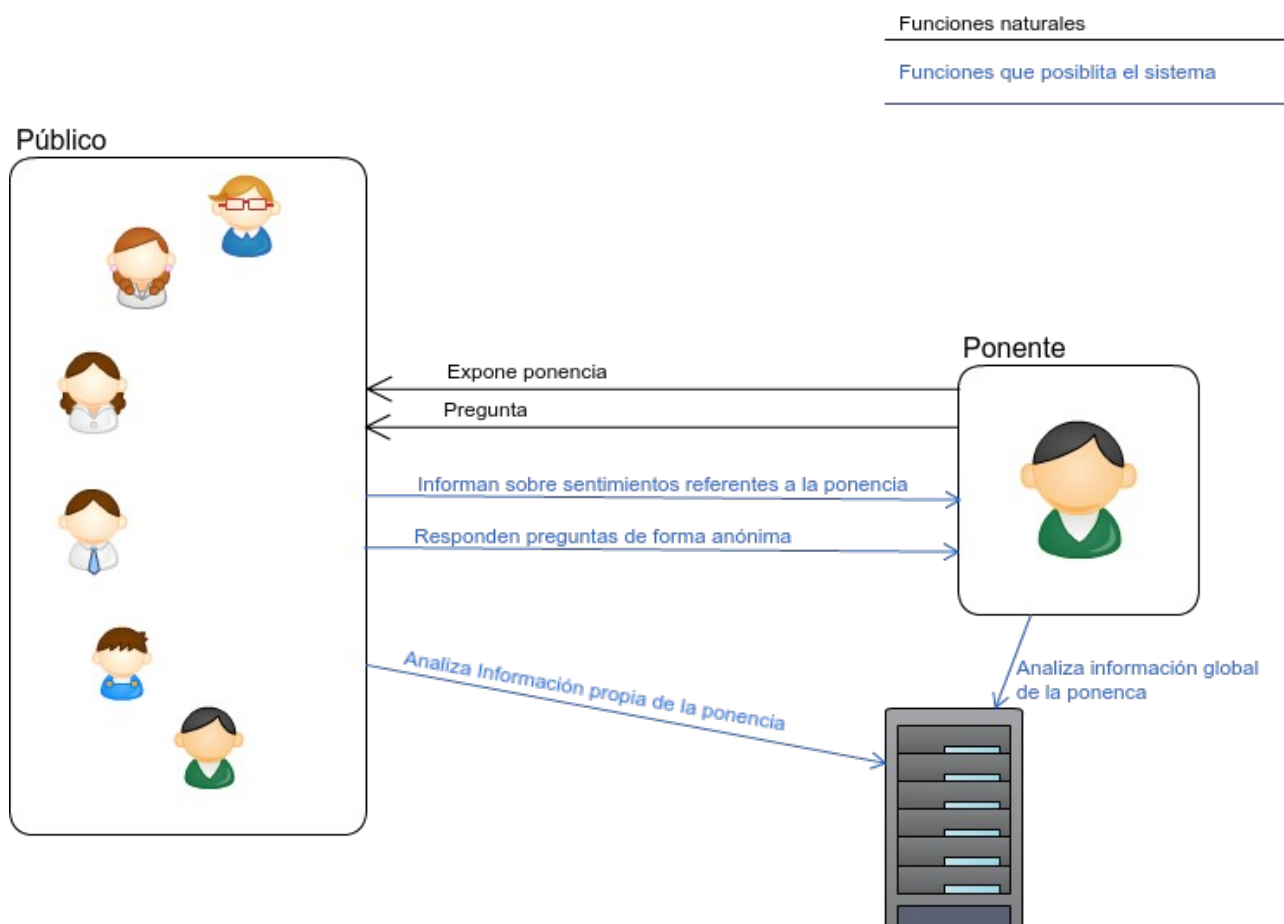
El público es una serie de actores en el sistema que actúan como sujetos relativamente pasivos dentro de una ponencia. Necesitarán atender a la ponencia y además, con este sistema, serán capaces de interactuar en tiempo real con el ponente, de forma anónima. Se espera que sean personas con conocimiento a nivel usuario de internet y aplicaciones informáticas, que cuando imparta la ponencia dispongan de un dispositivo de tamaño pequeño con capacidad de conexión a internet (una tablet o smartphone).

Los asistentes a la ponencia le deberán de dedicar gran parte de la atención a la ponencia, por lo que la interacción con el sistema a desarrollar deberá de ser sencilla y rápida (ya que no deseamos distraer a los asistentes con el sistema, si no que deben de estar pendientes de la ponencia).

A ciertos asistentes les puede ser de interés analizar a posteriori qué partes de la ponencia les resultaron más interesantes o qué respuestas marcaron a las preguntas que se hicieron en la ponencia. Para esta funcionalidad, se habilitará un registro de forma opcional de dicha persona en el sistema.

1.3.- Relación entre actores y funcionamiento del sistema

A continuación se presenta un diagrama de la relación simplificada entre los actores que intervienen en el sistema, detallando cómo interactúan con o sin usar el sistema:



Del diagrama presentado se detallan las siguientes propiedades:

- Sin usar el sistema, la información es en un único sentido: desde la persona que da la ponencia a la audiencia.
- El ponente puede lanzar preguntas, pero las respuestas no son anónimas por lo que pueden no ser respuestas sinceras.
- Usando el sistema el público tiene un canal de comunicación con el ponente, de forma anónima y en tiempo real, lo cual puede ser bueno para hacer presentaciones correctamente adaptadas a la audiencia, proporcionando una mejor experiencia a los asistentes.
- Usando el sistema, el público puede responder a preguntas de forma anónima, facilitando una mayor sinceridad (por ejemplo, si el ponente pregunta al público por el nivel de conocimiento de algún tema importante para la ponencia, un asistente podría mentir para no quedar en evidencia frente al resto de la audiencia).
- El sistema permite guardar los estados de ánimo del público de la ponencia, dando la posibilidad de un análisis posterior usando filtros por las características de cada asistente (por ejemplo, se podría filtrar por edad, género o por nivel de estudios, de forma que se podrían conocer los sentimientos de cada sector de la audiencia en cada punto de la ponencia).

2.- Metodología

En este apartado se desarrolla la metodología que se ha seguido para desarrollar el sistema. Se detallará la metodología según 3 perspectivas: metodología de obtención de requisitos, para explicar qué técnicas se han usado para la obtención de requisitos; metodología de Ingeniería del Software, donde se expondrá qué técnicas de Ingeniería del Software se han seguido para desarrollar el proyecto; y metodología de diseño de Base de Datos, donde se expondrá la metodología seguida para diseñar la estructura de la base de datos.

Una vez explicada la metodología, se expondrán los objetivos que tiene que cumplir el sistema, obteniendo de esta manera los requisitos funcionales. Por último en este apartado, se expondrán las tecnologías y técnicas que se utilizarán para la implementación del proyecto.

2.1.- Metodología de obtención de requisitos

La obtención de requisitos consiste en recoger la información de cómo se desea que funcione el sistema en función de los objetivos que debe de cumplir [5]. Para el desarrollo de este proyecto se han usado diversas técnicas de obtención de requisitos como metodología:

- Entrevistas. Para desarrollar este proyecto se ha contactado con ponentes y asistentes a ponencias para analizar cuales son las funcionalidades deseadas por ambos actores del sistema.
- Desarrollo de Prototipos. La obtención de requisitos se ha desarrollado con ayuda del desarrollo de prototipos: modelos de la interfaz, más o menos operativos (comenzando con un boceto en papel y terminando con la aplicación desarrollada). Las ventajas de esta metodología de obtención de requisitos son:
 - Al demostrar las funciones del sistema, se identifican las discrepancias entre desarrolladores y usuarios.
 - Durante el desarrollo del prototipo, se puede contrastar si los requisitos son inconsistentes y/o están incompletos.
 - Se dispone rápidamente de un sistema que demuestra usabilidad de la aplicación.
 - El prototipo se utiliza como base para escribir la especificación para el desarrollo.
- Diseño de *wireframes*. Un *wireframe*, también conocido como un esquema de página o plano de pantalla, es una guía visual que representa el esqueleto o estructura visual de un sitio web o aplicación. El wireframe esquematiza el diseño del contenido, incluyendo elementos de la interfaz y sistemas de navegación, y cómo funcionan en conjunto. Usualmente este esquema carece de estilo tipográfico, color o aplicaciones gráficas, ya que su principal objetivo reside en la funcionalidad, comportamiento y jerarquía de contenidos: se enfoca en qué hace la pantalla, en no cómo se ve. Los *wireframes* identifican:
 - Los tipos de información que será mostrada
 - La cantidad de las funciones disponibles
 - Las prioridades relativas de la información y las funciones
 - Las reglas para mostrar ciertos tipos de información
 - El efecto de los distintos escenarios en la pantalla

2.2.- Metodología de Ingeniería del Software

Metodología de ingeniería de software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. Una metodología de desarrollo de software es usada para estructurar, planear y controlar el proceso de desarrollo en sistemas de información.

En este proyecto se han usado diversas técnicas de Ingeniería del Software como metodología:

- Prototipado: El prototipado permite desarrollar modelos de aplicaciones de software que permiten ver la funcionalidad básica de la misma, sin necesariamente incluir toda la lógica o características del modelo terminado. El prototipado permite al cliente evaluar en forma temprana el producto, e interactuar con los diseñadores y desarrolladores para saber si se está cumpliendo con las expectativas y las funcionalidades acordadas.
- Diseño orientado a objetos [1]: enfoque de la ingeniería de software que modela un sistema como un grupo de objetos que interactúan entre sí. Se aplican técnicas de modelado de objetos para analizar los requisitos para un contexto - por ejemplo, un sistema de negocio, un conjunto de módulos de software - y para diseñar una solución para mejorar los procesos involucrados [2].
- Desarrollo ágil: El desarrollo ágil de software refiere a métodos de ingeniería del software basados en el desarrollo iterativo e incremental [3]. Los métodos ágiles enfatizan el código y diseños legibles en vez de la documentación técnica. Los métodos ágiles también enfatizan que el software funcional es la primera medida del progreso del desarrollo. Para el desarrollo de este proyecto se han utilizado diversas técnicas del desarrollo ágil, como la filosofía de Extreme Programming (XP).
 - La programación extrema (XP) es una metodología de desarrollo de la ingeniería de software [4]. Es el más destacado de los procesos ágiles de desarrollo de software. Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de la programación extrema consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos. Los valores de la programación extrema son: simplicidad, comunicación, retroalimentación y coraje.
 - Simplicidad: Se simplifica el diseño para agilizar el desarrollo y facilitar el mantenimiento.
 - Comunicación: En el desarrollo, el código comunica mejor cuanto más simple sea. Si el código es complejo hay que esforzarse para hacerlo inteligible. El cliente (la empresa de formación en el caso de este proyecto) decide qué características tienen prioridad y debe estar disponible para solucionar dudas.
 - Realimentación: Al realizarse ciclos muy cortos tras los cuales se muestran resultados, se minimiza el tener que rehacer partes que no cumplen con los requisitos y ayuda en el desarrollo a centrarse en lo que es más importante.
 - Coraje: Esto significa revisar el sistema existente y modificarlo, o si con ello los cambios futuros se implementarían más fácilmente. Otro ejemplo de coraje es saber cuando desechar una parte del sistema: quitar código fuente obsoleto, sin importar cuanto esfuerzo y tiempo se invirtió en crear ese código.
 - Aunque no se ha podido aplicar todo el sistema de la programación extrema en su conjunto (roles, desarrollo en parejas, pruebas unitarias...), sí se ha utilizado la filosofía ágil para el desarrollo del proyecto, consiguiendo unos resultados exitosos para los implicados (requisitos ajustados a la realidad de las necesidades, en el tiempo estimado).

2.3.- Metodología de Diseño de la Base de Datos

Para el diseño de la base de datos se ha utilizado, tras el análisis de los requisitos (explicados en el punto “Objetivos y Requisitos Funcionales”), la creación del modelo entidad-relación extendido que refleje los requisitos de información del proyecto.

- Un modelo entidad-relación [7] es una herramienta para el modelado de datos que permite representar las entidades relevantes de un sistema de información así como sus interrelaciones y propiedades. El modelo de datos entidad-relación está basado en una percepción del mundo real que consta de una colección de objetos básicos, llamados entidades, y de relaciones entre esos objetos. Una entidad representa una “cosa” u “objeto” del mundo real con existencia independiente, es decir, se diferencia unívocamente de otro objeto o cosa. Los atributos son las propiedades que describen a cada entidad en un conjunto de entidades. Las relaciones son enlaces entre las distintas entidades del modelo, con una cardinalidad establecida.

Una vez conseguido el modelo entidad-relación, se producirá un modelo relacional [6] para generar las relaciones a través del Sistema Gestor de Base de Datos. En este modelo todos los datos son almacenados en relaciones. La información puede ser recuperada o almacenada por medio de consultas que ofrecen una amplia flexibilidad y poder para administrar la información. El modelo relacional considera la base de datos como una colección de relaciones. De manera simple, una relación representa una tabla que no es más que un conjunto de filas, cada fila es un conjunto de campos y cada campo representa un valor que interpretado describe el mundo real. Cada fila también se puede denominar *tupla* o registro y a cada columna también se le puede llamar campo o atributo.

Además, en el proyecto se ha incluido sistema de datos “NoSQL” [16]. Las características comunes en estos sistemas, son principalmente: Ausencia de esquema relacional en los registros de datos, escalabilidad horizontal, y alta velocidad.

- La ausencia de esquema relacional significa que los datos no tienen una definición de atributos fija, es decir: Cada registro (o documento) puede contener una información con diferente forma en cada inserción, pudiendo así almacenar sólo los atributos que interesen en cada uno de ellos, facilitando el polimorfismo de datos bajo una misma colección de información. También se pueden almacenar estructuras de datos complejas en un sólo documento. Hacerlo así aumenta la claridad (al tener todos los datos relacionados en un mismo bloque de información) y el rendimiento.
- La escalabilidad horizontal es la posibilidad de aumentar el rendimiento del sistema instalando más nodo de forma sencilla. Muchos sistemas NoSQL permiten utilizar consultas del tipo *Map-Reduce* [17], las cuales pueden ejecutarse en varios nodos a la vez (cada uno operando sobre una porción de los datos) y reunir luego los resultados antes de devolverlos al cliente.
- Además, muchos de los sistemas NoSQL realizan operaciones directamente en memoria, y sólo vuelcan los datos a disco cada cierto tiempo. Esto permite operaciones de escritura rápidas.

2.4.- Objetivos y Requisitos Funcionales

Analizando (con las metodologías vistas en “Metodología de obtención de requisitos”) los requisitos del sistema, se ha llegado a una serie de objetivos generales que debe de cumplir el sistema, que dan lugar a varios sub-objetivos:

OBJ. 1. Guardar y administrar en el sistema los datos necesarios de los actores implicados para el uso del sistema. Poder tener un registro de los diferentes actores y datos que intervienen en el sistema.

Para ello se identifican las unidades de datos que entrarán en juego dentro del sistema. Analizando las entrevistas y las relaciones entre los actores, podemos encontrar los siguientes sub-objetivos:

OBJ. 1.1. Guardar y gestionar los datos de usuarios que se registren en el sistema: guardar datos para poder acceder al sistema y sobre las propiedades de los usuarios, de modo que permita a posteriori filtrar dichos datos en para el análisis de ponencias.

OBJ. 1.2. Guardar y gestionar los datos de las ponencias que usen el sistema: guardar información sobre las ponencias para que los asistentes a la ponencia puedan conectarse e interactuar con el sistema. Esto incluye:

OBJ. 1.2.1. Guardar y gestionar los estados de ánimo disponibles en la ponencia.

OBJ. 1.2.2. Guardar y gestionar las secciones en las que está dividida la ponencia.

OBJ. 1.2.3. Guardar y gestionar las propiedades de los usuarios que se usarán para analizar las ponencias.

OBJ. 1.2.4. Guardar y gestionar los filtros con los que se hará el análisis de la ponencia.

OBJ. 1.2.5. Guardar y gestionar las preguntas que el ponente hará durante la ponencia.

OBJ. 2. Guardar y para poder analizar en el sistema los datos sobre los estados de ánimo de los asistentes a las ponencias. Esta información puede ser de gran tamaño, puesto que se debe de guardar información de cada cambio de estado de ánimo que cada asistente realice en cada ponencia

A partir de los objetivos anteriormente descritos, se extraen los **Requisitos Funcionales** del sistema:

R.F. 1. Gestionar Usuarios.

R.F. 1.1. Dar de alta Usuarios.

R.F. 1.2. Consultar datos de Usuarios en el Sistema.

R.F. 1.3. Administrar información y propiedades de los Usuarios en el Sistema.

R.F. 2. Gestionar Presentaciones.

R.F. 2.1. Dar de alta Presentaciones

R.F. 2.2. Consultar datos de Presentaciones en el Sistema

R.F. 2.3. Administrar información de las Presentaciones en el Sistema.

R.F. 2.4. Borrar Presentaciones del Sistema.

R.F. 3. Gestionar Sentimientos disponibles en Presentaciones.

R.F. 3.1. Dar de alta Sentimientos.

R.F. 3.2. Consultar datos de Sentimientos en el Sistema.

R.F. 3.3. Administrar información de los Sentimientos en el Sistema.

R.F. 3.4. Borrar Sentimientos del Sistema.

- R.F. 4. Gestionar Secciones disponibles en Presentaciones.
 - R.F. 4.1. Dar de alta Secciones.
 - R.F. 4.2. Consultar datos de Secciones en el Sistema.
 - R.F. 4.3. Administrar información de las Secciones en el Sistema.
 - R.F. 4.4. Borrar Secciones del Sistema.
- R.F. 5. Gestionar Filtros disponibles en Presentaciones.
 - R.F. 5.1. Dar de alta Filtros.
 - R.F. 5.2. Consultar datos de Filtros en el Sistema.
 - R.F. 5.3. Administrar información de los Filtros en el Sistema.
 - R.F. 5.4. Borrar Filtros del Sistema.
- R.F. 6. Gestionar Preguntas disponibles en Presentaciones.
 - R.F. 6.1. Dar de alta Preguntas.
 - R.F. 6.2. Consultar datos de Preguntas en el Sistema.
 - R.F. 6.3. Administrar información de las Preguntas en el Sistema.
 - R.F. 6.4. Borrar Preguntas del Sistema.

2.5.- Tecnologías a utilizar

A continuación se especifican qué tecnologías se usarán en la implementación del proyecto. El lenguaje de programación utilizado para implementar el proyecto ha sido Java, con diversas bibliotecas y recursos para conseguir un sistema de información web estable, robusto, escalable, con posibilidad de funcionar en “la nube” y fácilmente administrable. Para el Sistema Gestor de Base de Datos se ha elegido MySQL, un SGBD libre y robusto. El proyecto se ha implementado usando el patrón de arquitectura software Modelo-Vista-Controlador, además de usarse en ciertos componentes el patrón de diseño Facade. Se han usado tecnologías de actualización de clientes web en tiempo real suministradas por HTML5, conocidas como “WebSockets”. Para la base de datos NoSQL se ha usado el sistema de base de datos MongoDB.

2.5.1.- Java

El lenguaje Java se creó con cinco objetivos principales:

- Debería usar el paradigma de la programación orientada a objetos.
- Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.
- Debería incluir por defecto soporte para trabajo en red.
- Debería diseñarse para ejecutar código en sistemas remotos de forma segura.
- Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos.

Una máquina virtual Java es una máquina virtual de proceso nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el bytecode Java), el cual es generado por el compilador del lenguaje Java.

Java Runtime Environment o JRE (Entorno de Ejecución de Java) es un conjunto de utilidades que permite la ejecución de programas Java. En su forma más simple, el entorno en tiempo de ejecución de Java está conformado por una Máquina Virtual de Java o JVM, un conjunto de bibliotecas Java y otros componentes

necesarios para que una aplicación escrita en lenguaje Java pueda ser ejecutada. El JRE actúa como un "intermediario" entre el sistema operativo y Java.

Por estas propiedades del lenguaje (Programa Orientado a Objetos, Multiplataforma y altamente extendido entre la comunidad de desarrolladores), se ha decidido que el sistema funcione bajo **Java SE 7**, usando el Entorno de Ejecución de Java (Java Runtime Environment) **1.7.0_80**.

2.5.2.- Bibliotecas y frameworks usados en Java

En el desarrollo de software, un framework, es una estructura conceptual y tecnológica de soporte definido, normalmente con módulos de software concretos, que puede servir de base para la organización y desarrollo de software.

Sobre las bibliotecas y frameworks usados en java, para este proyecto se han utilizado las que se detallan a continuación:

- **Java Server Faces (JSF) [8]:** Es un framework [10] para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE [9]. JSF usa JavaServer Pages (JSP) como la tecnología que permite hacer el despliegue de las páginas. Los siguientes objetivos de diseño representan el foco de desarrollo de JSF:
 - Definir un conjunto simple de clases base de Java para componentes de la interfaz de usuario, estado de los componentes y eventos de entrada.
 - Proporcionar un conjunto de componentes para la interfaz de usuario, incluyendo los elementos estándares de HTML para representar un formulario.
 - Definir APIs para la validación de entrada, incluyendo soporte para la validación en el lado del cliente.
 - Especificar un modelo para la internacionalización y localización de la interfaz de usuario.
 - Automatizar la generación de salidas apropiadas para el objetivo del cliente, teniendo en cuenta todos los datos de configuración disponibles del cliente, como versión del navegador.

En el proyecto se ha utilizado la versión de JSF 2.2.

- **PrimeFaces:** Es una biblioteca de componentes para Java Server Faces (JSF) de código abierto que cuenta con un conjunto de componentes enriquecidos que facilitan la creación de las aplicaciones web [11]. Es una biblioteca muy liviana, todas las decisiones hechas son basadas en mantener a PrimeFaces lo más liviano posible. El proyecto utiliza la versión de PrimeFaces 5.1. Dentro de esta biblioteca se hace un uso intensivo de los componentes "Primefaces Mobile", componentes especiales adaptados a dispositivos móviles con diseño responsivo.
- **Java Persistence API (JPA):** Es la API de persistencia desarrollada para la plataforma Java [12]. Sigue la filosofía de Mapeo objeto-relacional [14]. El mapeo objeto-relacional es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional, utilizando un motor de persistencia.
JPA posibilita usar Java Persistence Query Language [15], un lenguaje de consulta orientado a objetos. Es usado para hacer consultas contra las entidades almacenadas en una base de datos relacional. Está inspirado en gran medida por SQL, y sus consultas se asemejan a las consultas SQL en la sintaxis, pero opera con objetos entidad de JPA en lugar de hacerlo directamente con las tablas de la base de datos. Además, posibilita usar los Enterprise JavaBeans (EJB). El objetivo de los EJB es dotar al programador de un modelo que le permita abstraerse de los problemas generales de una aplicación empresarial (conurrencia, transacciones, persistencia, seguridad, etc.) para centrarse en el desarrollo de la lógica de negocio en sí. El hecho de estar basado en componentes permite que éstos sean flexibles y sobre todo reutilizables. En este proyecto se ha usado la implementación

EclipseLink de JPA 2.0 [13].

- **Atmosphere:** un framework Java y Javascript para el desarrollo de aplicaciones Web *Push* sobre el nuevo protocolo HTML5 Web Socket. Esto nos permite comunicarnos con navegadores web desde el contenedor de aplicaciones para conseguir comunicaciones en tiempo real, pudiendo llamar a rutinas de javascript en el cliente desde el servidor de forma automática. Se ha usado la version de Atmosphere 2.2.2. [19].
- **Conector de MySQL:** Biblioteca usada por JPA para conectarse a la base de datos MySQL.
- **Conector de MongoDB:** Biblioteca usada para administrar bases de datos MongoDB.
- **Chartist:** Potente y ligera biblioteca de Javascript usada para mostrar gráficos en el cliente.

2.5.3.- Software del servidor

A continuación se detallan las herramientas que se utilizan en el servidor del proyecto: MySQL como Sistema Gestor de Base de Datos, GlassFish como contenedor de aplicaciones web Java, etcétera.

- **GlassFish:** Es un contenedor de aplicaciones de software libre que implementa las tecnologías definidas en la plataforma Java y permite ejecutar aplicaciones que siguen esta especificación [21]. Se trata de un dispositivo de software que proporciona servicios de aplicación a los host cliente. El servidor de aplicaciones generalmente gestiona la mayor parte (o la totalidad) de las funciones de lógica de negocio y de acceso a los datos de la aplicación. El proyecto presentado utiliza la versión de GlassFish 4.1.
- **MySQL [24]:** Es un sistema de gestión de bases de datos [18] relacional orientado a objetos basado de Software Libre. Un sistema de gestión de bases de datos (SGBD) es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos, además de proporcionar herramientas para añadir, borrar, modificar y analizar los datos.

Al realizar el proyecto, se estudió una comparativa entre MySQL, PostgreSQL y SQLite [25]. Se pueden sacar las siguientes conclusiones:

- SQLite es un SGBD que puede ser embebido en la aplicación que estamos desarrollando (sin necesidad de tener un servidor de bases de datos funcionando fuera del proyecto). Al usar SQLite se hacen llamadas a ficheros de datos, en vez de usar puertos o sockets para comunicarse con el sistema, lo cual hace a SQLite extremadamente rápido y eficiente. Las desventajas de usar SQL son:
 - No existe gestión de usuarios y privilegios sobre tablas y bases de datos.
 - No permite comunicación de base de datos multi-usuario: al hacer escrituras y lecturas de un fichero, el fichero sólo puede ser accedido por un usuario al mismo tiempo, impidiendo la concurrencia.

Por estos motivos se descarta SQLite.

- PostgreSQL es un gestor de bases de datos objeto-relacional cuyo logro principal es ser extensible y tener un uso completo del estándar PL/SQL. La concurrencia dentro de este sistema es conseguida sin bloqueos de escritura gracias a un control avanzado de la concurrencia que además asegura cumplir el estándar ACID (Atomicidad, Consistencia, Aislamiento y Persistencia). Posee un gran número de tipos de datos y soporte para objetos, pero tiene las siguientes desventajas:
 - PostgreSQL es más lento que otros SGBD (como MySQL) en operaciones de lectura.
 - No es tan popular como otros SGBD, por lo que puede dar problemas en cuanto a soporte y proveedores de dicho SGBD.

- MySQL es el más popular de los SGBD libres. Es un producto de código abierto con gran uso en aplicaciones web. Soporta una gama básica de tipos de datos (no tantos como PostgreSQL), pero tiene varias ventajas:
 - Seguridad: MySQL implementa características avanzadas de seguridad (cifrado, gestión avanzada de usuarios).
 - MySQL es sencillo de escalar si es necesario.

Por todo ello fue elegido como SGBD para este proyecto. La versión utilizada es: MySQL 14.14.

- **CentOS:** Es el sistema operativo en el que está configurado el sistema. Es un sistema operativo basado en Linux y que se distribuye como software libre. CentOS está orientado a servidores, sin instalar una interfaz gráfica de usuario. En este proyecto se ha utilizado la versión de CentOS disponible en el Cloud de Amazon Web Services, consiguiendo un servidor web en la nube de alta escalabilidad.

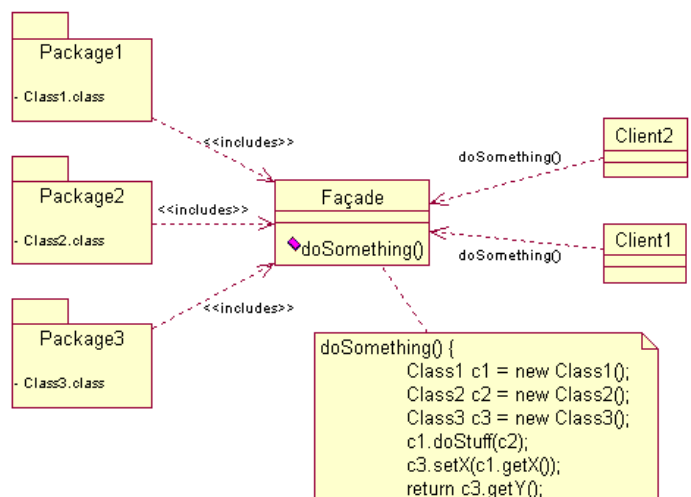
2.5.4.- Modelo–vista–controlador

El modelo–vista–controlador (MVC) es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones [23]. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario.

- **Modelo:** Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también la lógica de negocio de la aplicación.
- **Vista:** Presenta el 'modelo' (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario).
- **Controlador:** Responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos).

En el proyecto se aplica este patrón de arquitectura de software por medio del framework Java Server Faces, el cual posibilita la arquitectura MVC en el proyecto. Según el funcionamiento de JSF, la vista sería la página XHTML donde se muestra la información, la cual se enviará mediante el protocolo HTTP a la computadora cliente (tal y como funcionan las aplicaciones web); el controlador es la clase Java con la que se comunica la vista para interactuar con los eventos de la vista; y el modelo son las clases EJB soportadas por JPA. El MVC supone una arquitectura de desarrollo de software limpia y sencilla, que hace más sencillo el desarrollo del sistema.

Se aplica también el patrón “Facade” (fachada) cuando se necesita proporcionar una interfaz simple para un subsistema complejo, o cuando se quiere estructurar varios subsistemas en capas. Las fachadas son el punto de entrada a cada nivel del sistema complejo, aumentando la simplicidad y la reutilización de componentes para los consumidores de componentes de este diseño.



3.- Diseño del sistema

En este apartado se expondrá cómo se ha llevado a cabo el diseño, desarrollo e implementación del sistema. Se expone el diseño seguido para crear la base de datos del sistema y el diseño del código fuente e implementación del sistema.

3.1.- Diseño de la base de datos

Se procede al Diseño de la Base de Datos del proyecto. Como se ha explicado en el punto “Metodología de Diseño de la Base de Datos”, el diseño se desarrollará haciendo uso de un Modelo Entidad/Relación y un Modelo Relacional, que servirán para implementar mediante código SQL la base de datos en el sistema.

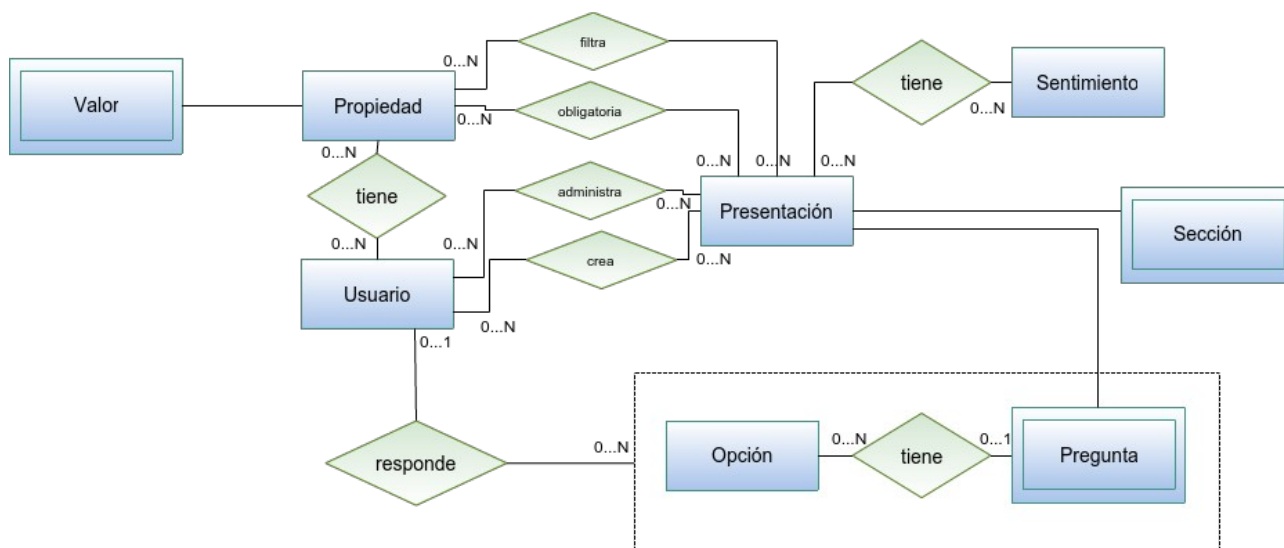
3.1.1.- Relación de Requisitos Funcionales con Entidades

En primer lugar se deben de identificar las entidades y sus relaciones. Para ello se van a utilizar los Requisitos Funcionales del punto “Objetivos y Requisitos Funcionales”:

- Se necesita una entidad “Usuario” para preservar los Requisitos Funcionales RF1, RF1.1, RF1.2 y RF1.3.
- Se necesita una entidad “Presentación” para preservar los Requisitos Funcionales RF2, RF2.1, RF2.2, RF2.3 y RF2.4.
- Se necesita una entidad “Sentimiento” relacionada con la entidad “Presentación” para preservar los Requisitos Funcionales RF3, RF3.1, RF3.2, RF3.3 y RF3.4.
- Se necesita una entidad “Sección” relacionada con la entidad “Presentación” para preservar los Requisitos Funcionales RF4, RF4.1, RF4.2, RF4.3 y RF4.4.
- Se necesita una entidad “Filtro” relacionada con la entidad “Presentación” para preservar los Requisitos Funcionales RF5, RF5.1, RF5.2, RF5.3 y RF5.4.
- Se necesita una entidad “Pregunta” relacionada con la entidad “Presentación” para preservar los Requisitos Funcionales RF6, RF6.1, RF6.2, RF6.3 y RF6.4.

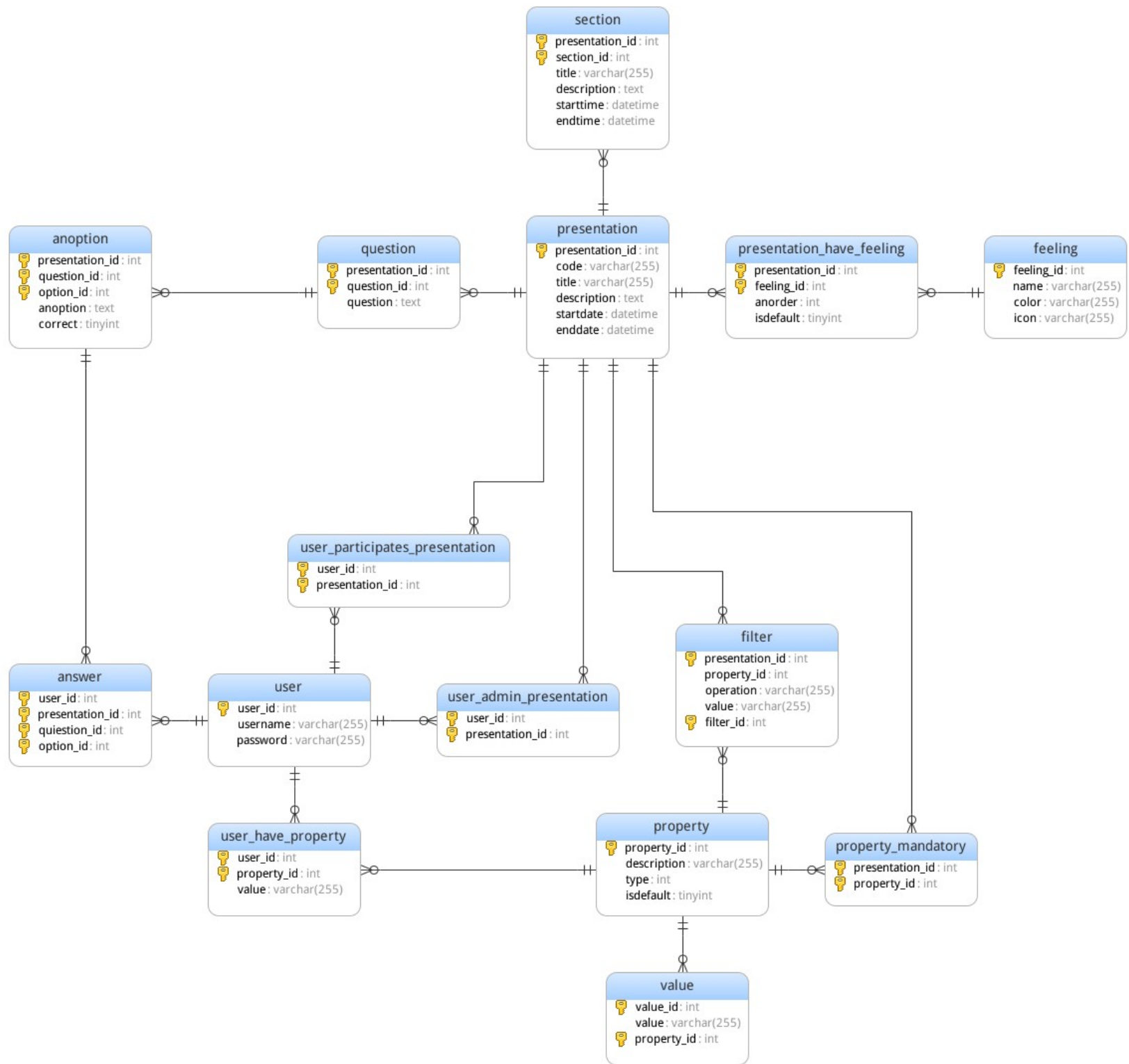
3.1.2.- Modelo Entidad/Relación

Una vez analizados los requisitos en el punto anterior, se crea el Modelo Entidad/Relación. A continuación se presenta el Modelo Entidad/Relación:



3.1.3.- Modelo Relacional

Tras obtener el modelo Entidad/Relación simplificado, se desarrolla dicho modelo hasta obtener el Modelo Relacional. El Modelo Relacional es un modelo que muestra las tablas (relaciones) y referencias (claves externas) que se implementará en la estructura de la base de datos, por lo que es una conceptualización del modelo físico de la base de datos. Se ha creado aplicando las Formas Normales [20] al Modelo Relacional extraído del modelo Entidad/Relación, hasta conseguir la Tercera Forma Normal en el diseño de la base de datos. Además, se le han añadido los atributos necesarios para cumplir todos los requisitos de información necesarios para el sistema. El Modelo Relacional resultante es el siguiente:



A continuación se presentan las relaciones más importantes que aparecen en el Modelo Relacional, comentando sus atributos más importantes:

- Presentación: Guarda los datos de la presentación
 - Título de la Presentación
 - Código (único entre presentaciones) de la presentación
 - Descripción de la presentación
 - Fecha y hora de inicio de la presentación
 - Fecha y hora de fin de la presentación
- Esta tabla tiene tablas relacionadas con ella, como son:
 - Sentimiento: Guarda los datos de sentimientos que los usuarios podrán elegir para la presentación
 - Nombre del sentimiento
 - Color con el que se representa el sentimiento
 - Icono con el que se representa el sentimiento
 - Relación de N a N con Presentación. A dicha relación se le añaden los atributos: orden (cuál es el sentimiento más importante y cual el que menos) y Default (cuál es el sentimiento que aparecerá marcado por defecto para el asistente a la presentación cuando se conecte).
 - Sección: Guarda los datos de las secciones de las que se compone una presentación
 - Título de la sección
 - Descripción de la sección
 - Fecha y hora de inicio de la sección
 - Fecha y hora de fin de la sección
 - Preguntas: Guarda las preguntas que se harán en la presentación
 - Texto de la pregunta
 - Guarda una relación con “Opción” (la cual se relaciona con “Respuesta”)
 - Texto de la opción
 - Marca si es o no es una opción correcta para la pregunta
 - Guarda una relación con “Respuesta”, que es la relación entre “Opción” y “Usuario”
 - Filtro: Guarda la propiedad por la que se puede filtrar la presentación al ser analizada
 - Operación de una propiedad: igual, desigual, mayor o menor que un valor
 - Valor por el que se filtra la respuesta
 - Obligatorio: Guarda las propiedades que deberán de ser introducidas obligatoriamente por los asistentes a una ponencia cuando interactúen con la presentación.
 - Administración y participación: indica qué usuarios pueden administrar y qué usuarios han participado en una presentación.

- Usuario: Guarda los datos de los usuarios del sistema
 - Nombre de usuario
 - Contraseña del usuario (Encpritada siguiendo el algoritmo MD5).
- Propiedad: Guarda las propiedades que los usuarios pueden rellenar y las presentaciones pueden utilizar para analizar los datos
 - Descripción de la propiedad
 - Tipo de propiedad: Lista, Fecha, Entero o Cadena de caracteres.
 - Guarda si es una propiedad por defecto (generada por el sistema y disponible para todos los usuarios) o no (generada por las presentaciones y disponible para usuarios que hayan participado en dichas presentaciones).
 - Si el tipo de propiedad es una Lista, existe una relación de valores para guardar los valores de las listas.
 - La propiedad se relaciona con el usuario con una relación de muchos a muchos aportando un campo “Valor” donde guarda el valor de la propiedad usada.

3.1.4.- Modelo NoSQL

Como se ha comentado en apartados anteriores, el proyecto utiliza una base de datos NoSQL (MongoDB) para guardar los datos relacionados con qué sentimientos tiene cada usuario en una presentación en cada instante de tiempo. Los datos guardados en esta base de datos consiste en una colección de documentos, en el que cada documento tiene la siguiente estructura:

- Fecha y hora en la que se ha guardado el sentimiento.
- ID de usuario del sentimiento.
- ID de la presentación.
- ID del sentimiento.
- Lista de propiedades asociadas a las propiedades obligatorias de la presentación, guardadas como:
 - ID de la propiedad
 - Valor de la propiedad

Al guardar estos datos en la base de datos NoSQL se mejora el rendimiento de análisis de los datos puesto que se hace de manera más eficiente por el funcionamiento descrito de estas bases de datos.

3.2.- Diseño del proyecto

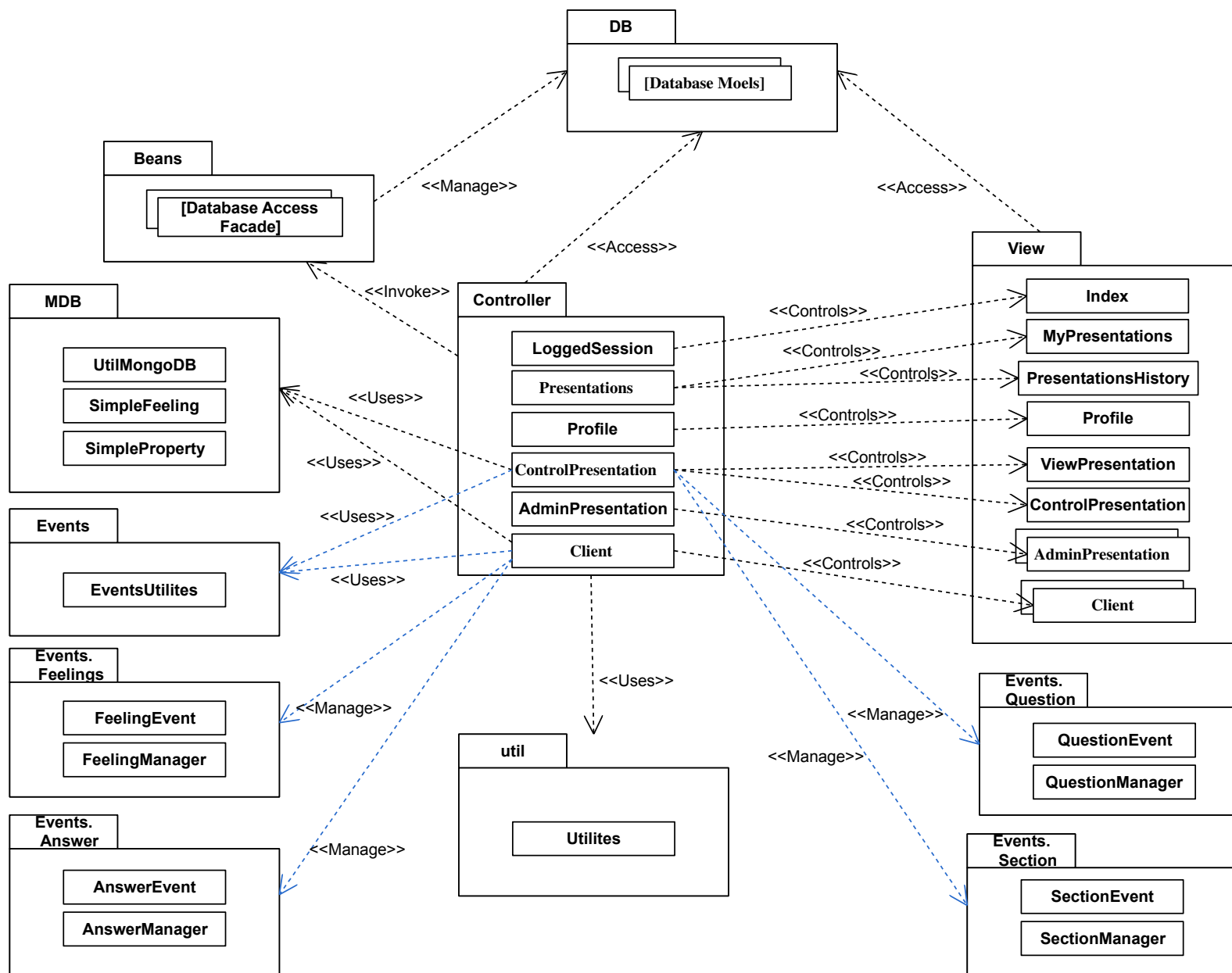
En esta sección se detalla el diseño usado para desarrollar el código fuente e implementar el sistema. Está dividido en las siguientes secciones: paquetes del proyecto, donde se indica qué paquetes de clases Java existen en el proyecto, sus contenidos, funcionalidades y agrupamientos; jerarquía de archivos, donde se indica cómo están organizadas las carpetas y archivos del sistema web; mapa del sitio, donde se detalla a qué vistas se puede llegar en el sistema; diseño de mensajes, donde se explica el sistema de mensajes usado con WebSockets para permitir análisis en tiempo real; diseño de vistas, donde se indica las vistas utilizadas y cómo se utilizan.

3.2.1.- Paquetes del proyecto

A continuación se detallan los paquetes de clases Java existentes en el proyecto. Un Paquete en Java es un contenedor de clases que permite agrupar las distintas partes de un programa cuya funcionalidad tienen elementos comunes. Se explicará el contenido del paquete, las funcionalidades de las clases del paquete y el por qué de dicho agrupamiento:

- Paquete de Beans de Fachada de Modelos de Entidades: Este paquete incluye las clases de control de acceso a la base de datos (consulta, edición, borrado) de los Modelos de las Entidades. En este paquete se encuentran las clases que dan la funcionalidad de recuperar, editar y borrar de la base de datos las distintas entidades que existen. Las clases son “Beans” de Fachada: Un Bean es un componente software que es reutilizable y evita reescribir el código para replicar la funcionalidad de un componente. “Fachada” hace referencia a que se ha usado el patrón de diseño “Fachada” para diseñar las clases de este paquete: el patrón de diseño “Fachada” significa que encapsula las llamadas a distintas clases (en este caso, llamadas a la biblioteca JPA) para que sea sencillo utilizar esta clase desde otras clases como interfaz.
- Paquete de Modelos de Entidades (DB): Este paquete guarda las clases EJB de los Modelos de las Entidades de la base de datos. Siguiendo el patrón de arquitectura software MVC, este paquete guarda las clases que están asociadas a las relaciones de la base de datos. Estas clases también incluyen el código de lógica de negocio para las entidades que lo necesitan. Es el paquete que incluye las “estructuras” de los objetos de la base de datos, para que puedan ser usadas en los controladores y las vistas.
- Paquete de Controladores: Según el patrón de arquitectura software MVC, este paquete guarda las clases de Controladores de las Vistas que interaccionan con los Modelos del sistema. Guarda las clases que tienen implementada la funcionalidad de la interfaz del sistema para las entidades que son accesibles para su gestión (la lógica de negocio). Por ejemplo: permiten crear, editar y borrar (llamando a las clases del Paquete de Beans de Fachada de Modelos de Entidades) Presentaciones, Usuarios, etcétera. En estas clases están implementadas las llamadas a las clases que cumplen los requisitos funcionales del sistema.
- Paquete de Utilidades: Este paquete incluye clases de utilidades diversas para el funcionamiento del sistema, tales como redireccionar páginas, mostrar mensajes, generar MD5, etcétera.
- Paquete de gestión de MongoDB: contiene las herramientas para trabajar con la base de datos NoSQL MongoDB: funciones para insertar y consultar registros y objetos simples para mapear los registros recibidos de las entidades.
- Paquete de utilidades JSF: contiene utilidades para extender la funcionalidad de JSF, tales como comparadores, las cuales extienden la posibilidad de operaciones de los modelos en las vistas.
- Paquete de eventos: este paquete y los sub-paquetes (respuestas, sentimientos, preguntas, secciones) contienen la lógica y los objetos auxiliares para la comunicación en tiempo real del sistema usando WebSockets de HTML5.

El Diagrama de Paquetes del proyecto, con sus relaciones entre ellos, es el siguiente:



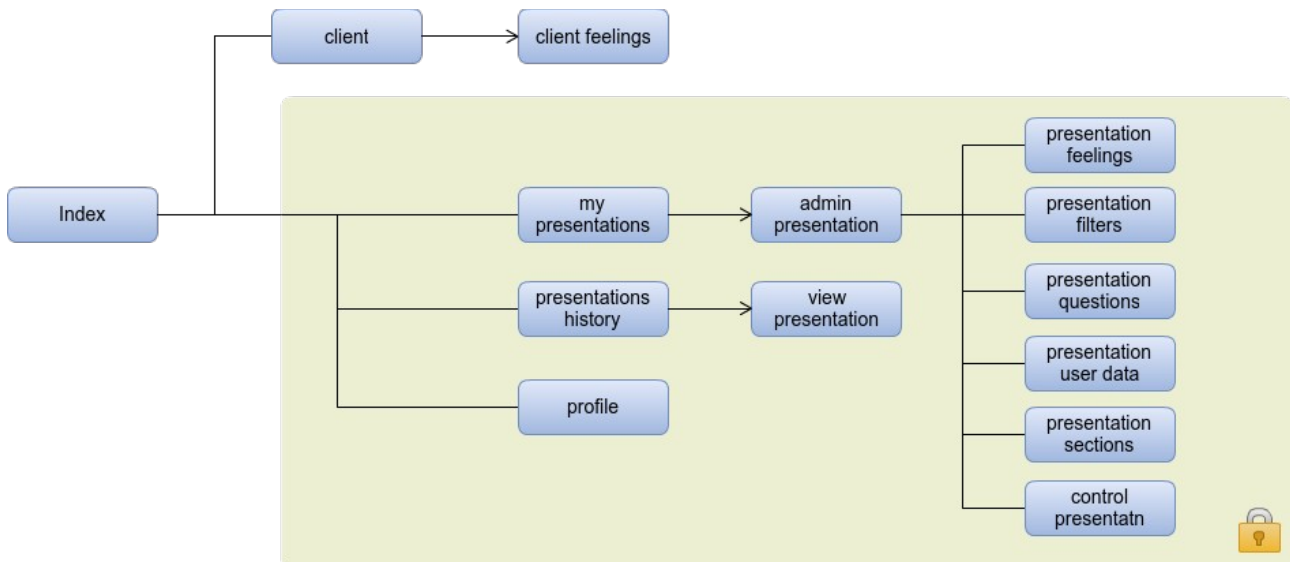
3.2.2.- Jerarquía de archivos

Los archivos del sistema están jerarquizados para un acceso sencillo que posibilite un acceso ordenado a los distintos recursos mediante peticiones HTTP. A continuación se indica la jerarquía usada en el sistema:

- **Resources:** contiene los recursos javascript, CSS e imágenes usados por el sistema.
 - **Functions.js:** Contiene la lógica del cliente web, en modo de funciones javascript.
- **Ping.html:** Página en blanco a la que hacer “Ping” para mantener activa la sesión del usuario.
- **Template.xhtml:** Vista plantilla la cual se exporta a todas las demás vistas, en la que se declaran las dependencias de JavaScript y la estructura básica de la aplicación (navegación, registro de usuario).
- **View:** contiene las vistas del sistema.
 - **Adminpresentation.xhtml:** Vista para la administración de presentaciones, contiene enlaces a las siguientes vistas:
 - **Adminpresentationfeelings.xhtml:** Vista que gestiona los sentimientos que pueden configurarse para la presentación.
 - **Adminpresentationfilters.xhtml:** Vista que gestiona los filtros que pueden configurarse para la presentación.
 - **Adminpresentationquestions.xhtml:** Vista que gestiona las preguntas que pueden configurarse para la presentación.
 - **Adminpresentationsections.xhtml:** Vista que gestiona las secciones que pueden configurarse para la presentación.
 - **Adminpresentaionuserdata.xhtml:** Vista que gestiona los datos de usuario que pueden configurarse para la presentación.
 - **Client.xhtml:** Contiene la vista a la que se conectan los asistentes a una ponencia, donde se gestionan las propiedades obligatorias asociadas a la ponencia. Esta vista direcciona a la vista de gestión de estados de ánimo de los asistentes, llamada **ClientsFeelings.xhtml**.
 - **Index.xhtml:** Página principal del sistema, desde la cual se puede acceder a presentaciones introduciendo el código de la presentación.
 - **Mypresentations.xhtml:** Vista que muestra el listado de las presentaciones que el usuario puede administrar.
 - **Presentationshistory.xhtml:** Vista que muestra el listado de las presentaciones en las que el usuario ha participado como asistente.
 - **Controlpresentation.xhtml:** Vista accesible para el ponente en la que se muestra la información en tiempo real de los asistentes a la presentación, según la configuración de la presentación.
 - **Profile.xhtml:** Vista en la que el usuario puede gestionar sus propiedades de usuario.
 - **Viewpresentation.xhtml:** Vista en la que el usuario accederá para ver una presentación en la que participó anteriormente. En esta vista se muestran los datos de sentimientos que el usuario generó para la ponencia, además de datos sobre las preguntas hechas en la ponencia.

3.2.3.- Mapa del sitio

A continuación se presenta un mapa de acceso al sitio, con las posibilidades de entrar y moverse por las diferentes vistas del sistema:



3.2.4.- Diseño de Mensajes

El sistema hace uso de mensajes entre la vista “*clientfeeling.xhtml*” y la vista “*controlpresentation.xhtml*” para gestionar la información en tiempo real entre el ponente y los asistentes. Los mensajes son enviados usando WebSocket (mediante la biblioteca Atmosphere).

WebSocket es una tecnología que proporciona un canal de comunicación bidireccional sobre un único socket TCP. Está diseñada para ser implementada en navegadores y servidores web, pero puede utilizarse por cualquier aplicación cliente/servidor. El protocolo WebSocket está normalizado por la IETF como el RFC 6455 [26]. Debido a que las conexiones TCP comunes sobre puertos diferentes al 80 son habitualmente bloqueadas por los administradores de redes, el uso de esta tecnología proporcionaría una solución a este tipo de limitaciones proveyendo una funcionalidad similar a la apertura de varias conexiones en distintos puertos, pero multiplexando diferentes servicios WebSocket sobre un único puerto TCP.

WebSocket está ya implementado en los siguientes navegadores (y posteriores versiones): Mozilla Firefox 8, Google Chrome 4 y Safari 5, así como la versión móvil de Safari en el iOS 4.2.1 y en Internet Explorer 10

La aplicación hace uso de 4 WebSockets:

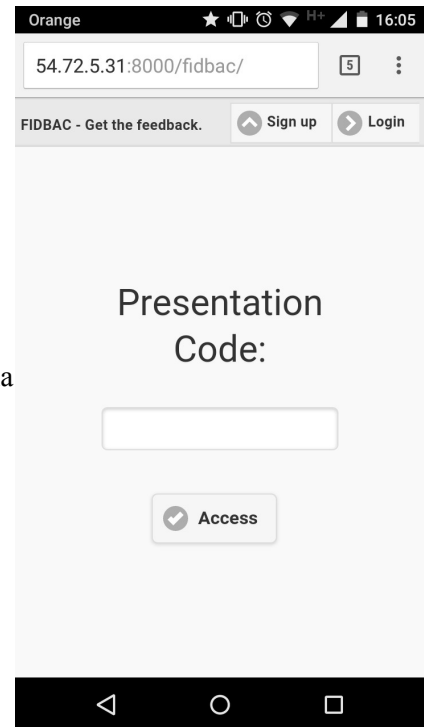
- ***controlpresentation.xhtml***
 - ***Answer***: Gestiona la recepción de las respuestas a las preguntas lanzadas por el presentador, además de permitir recibir de mensajes anónimos al ponente.
 - ***Feeling***: Gestiona la recepción de estados de ánimo de los asistentes, actualizando en tiempo real los datos del estado de los asistentes con la presentación.
- ***clientfeeling.xhtml***
 - ***Question***: Gestiona la recepción de las preguntas activadas por el presentador.
 - ***Section***: Gestiona la recepción de los cambios de secciones realizadas por el presentador, actualizando el estado de los asistentes. También controla la desconexión de los asistentes cuando finaliza la presentación.

3.2.5.- Diseño de Vistas

A continuación se mostrarán las vistas más interesantes de la aplicación, comentando su contenido y uso.

Index

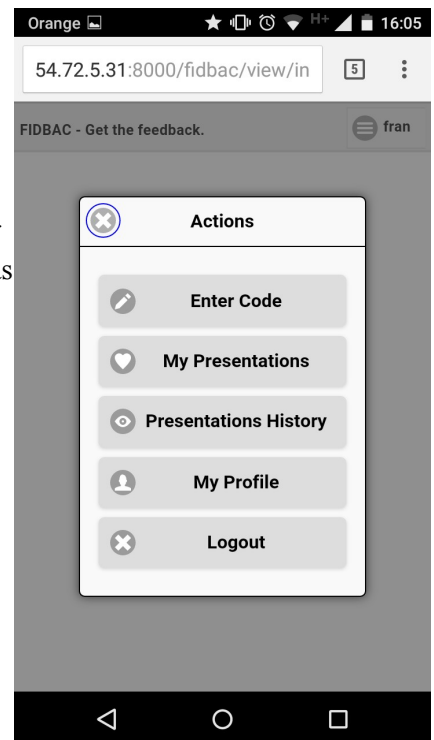
La página principal de la aplicación muestra tres funcionalidades: darse con un usuario y una contraseña (Sign Up), entrar en la aplicación con un usuario ya dado de alta anteriormente (Login) e introducir un código de presentación para interactuar con la presentación como oyente (Presentation Code). Se ha diseñado una vista sencilla para que sea sencilla de utilizar para el mayor número de personas posibles. Se puede acceder a una presentación si tener un usuario dado de alta, razón por la que esta opción se muestra en la página principal.



Menú de acciones

Una vez estamos conectados al sistema, tenemos disponibles las siguientes posibilidades:

- Introducir un código, lo cual nos llevará a la vista “index” mencionada anteriormente.
- Acceso a “Mis Presentaciones”, las presentaciones administradas.
- Acceso a “Historial de Presentaciones”, las presentaciones en las que se ha participado como asistente.
- “Mi Perfil” para gestionar datos del Perfil.
- “Logout” para salir del sistema.



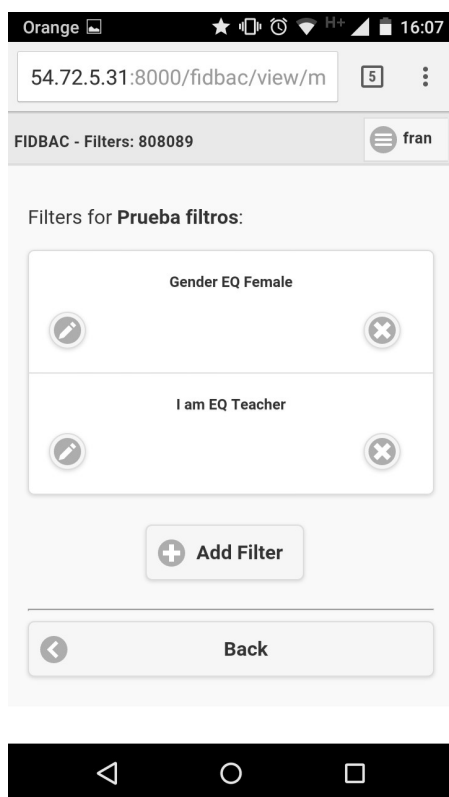
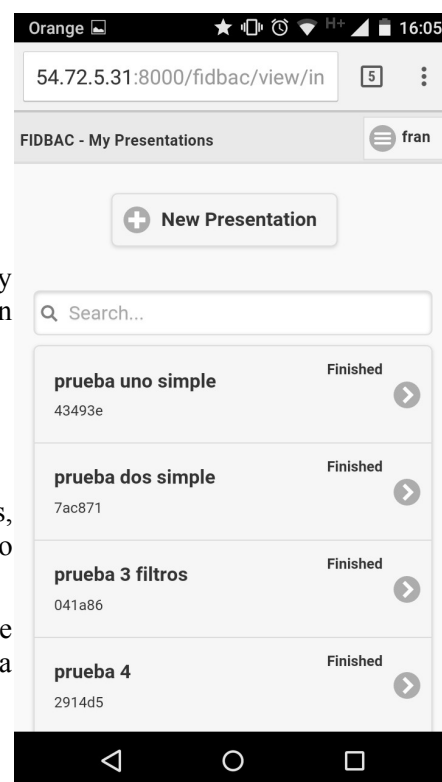
Mis Presentaciones

Muestra una lista de las presentaciones a las que tenemos acceso administrativo, mostrando: Título, código y estado (pendiente, iniciada y finalizada). También tenemos acceso a crear nuevas presentaciones (botón “New Presentation”) y a acceder a la administración de las presentaciones.

Dentro de cada presentación se muestra la funcionalidad para administrar los distintos aspectos de la ponencia: Sentimientos, Secciones, Preguntas, Datos de Usuario y Filtros.

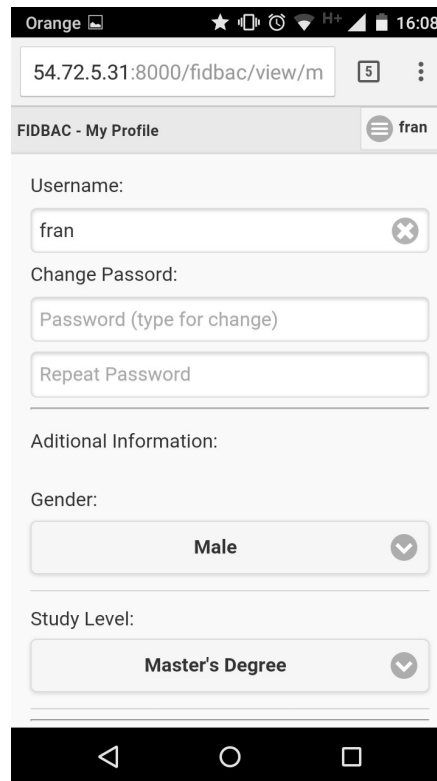
Las páginas tienen una estructura similar: se muestra una lista de los datos, pudiéndose editar dichos datos pulsando sobre ellos, o crearlos o eliminarlos pulsando los botones correspondientes.

Para editar los datos, se muestran ventanas “op-ups” en las que se puede editar la información del dato seleccionado. Se muestra un ejemplo para editar un filtro.



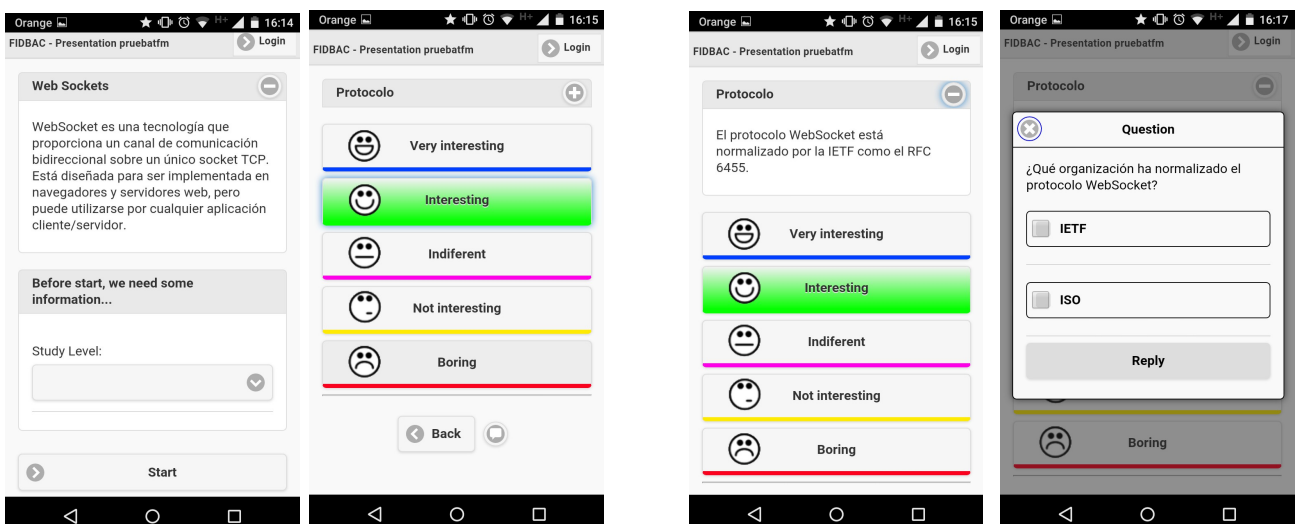
Edición de Perfil

En la vista de edición de perfil se pueden editar los datos referentes al usuario conectado: nombre de usuario y contraseña, además de los datos de usuario por defecto mas los datos de usuario que haya introducido al participar en alguna presentación.



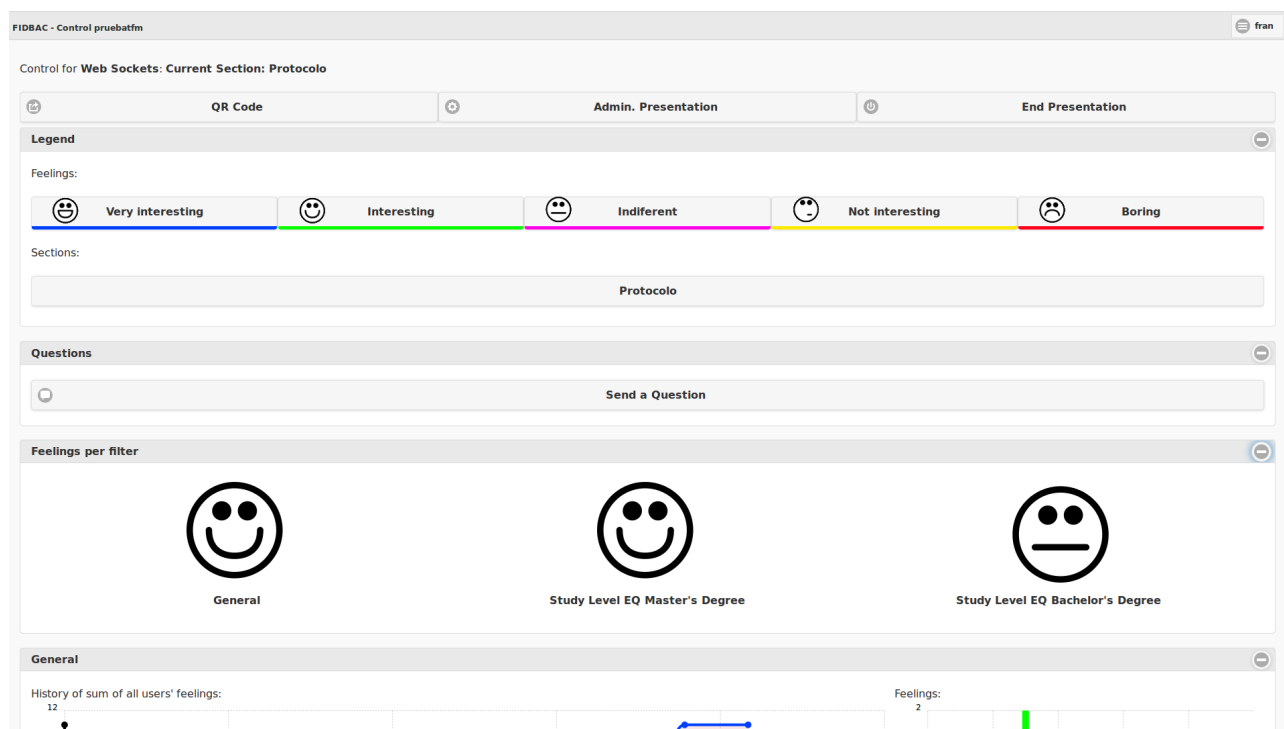
Asistente a una ponencia

La vista de un asistente a una ponencia es una vista en la que cada asistente podrá introducir los datos pedidos por la ponencia, seleccionar el estado de ánimo que le produce la ponencia y responder a preguntas que lance el ponente a través de la aplicación. Además, el asistente podrá realizar peticiones o preguntas a través del botón habilitado para ello (al lado del botón “Back”) y conocer información sobre la sección en la que se encuentra la ponencia.



Control de la ponencia

El ponente tiene acceso a la vista de control de la ponencia cuando está haciendo la presentación. Dicha vista es un “cuadro de mandos” completo para conocer el estado de los sentimientos de los asistentes (pudiendo analizar los datos usando los filtros) y para efectuar las operaciones de la presentación (pasar de sección y enviar preguntas). Esta vista está pensada para verse en una tablet ó portatil, por lo que no está adaptada a dispositivos de menor tamaño.



En la parte superior de la vista se muestran las siguientes opciones:

- Leyenda con las secciones y sentimientos (colores, iconos y nombres) configurados para la ponencia.
- Sección de preguntas para enviar las preguntas configuradas en la ponencia a los asistentes.
- Sección de “sentimientos por filtro”, el cual muestra, para cada filtro configurado, el sentimiento de moda (el sentimiento con más frecuencia entre los asistentes) en el momento actual de la presentación.



Bajo los paneles de control, se muestra el panel de datos general, el cual está separado en cuatro secciones:

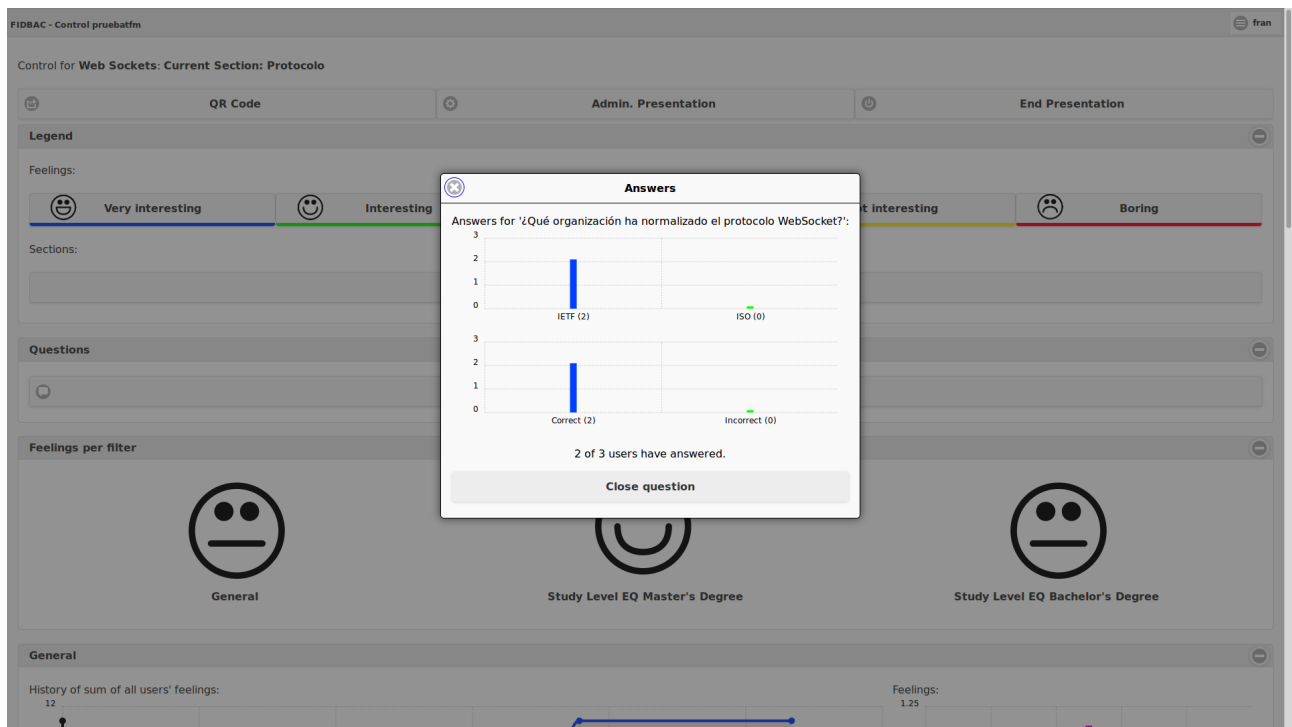
- Arriba a la izquierda se muestra un gráfico de líneas en el cual muestra, para cada instante de tiempo (el eje X del gráfico es el tiempo transcurrido de la presentación, medido en minutos) la suma de los estados de ánimo de todos los asistentes de la sala, en el que cada estado de ánimo tiene un valor de: 1 para el estado de ánimo más bajo y el tamaño de estados de ánimo para el estado de ánimo mas alto. Por ejemplo: 5: Muy Interesante; 4 Interesante; 3: indiferente; 2: No interesante; 1: Aburrido

Por lo tanto, para cada instante de tiempo el gráfico muestra la suma de los valores dichos para cada asistente a la presentación, por ejemplo: si existen 3 asistentes con estado de ánimo 5 y 2 asistentes con estado de ánimo 2, el valor final será 14. Este gráfico nos permite ver rápidamente qué secciones o tiempos de la ponencia fueron mas interesantes o menos interesantes de forma general para todos los usuarios.

- Abajo a la izquierda se muestra un gráfico de líneas en el cual muestra, para cada instante de tiempo (el eje X del gráfico es el tiempo transcurrido de la presentación, medido en minutos) el número de asistentes que ha seleccionado cada estado de ánimo. Cada estado de ánimo está representado por una línea en el gráfico, con el color de las líneas configurado al color que el presentador haya configurado para el estado de ánimo. Las líneas pueden mostrarse u ocultarse, para mayor claridad y capacidad de análisis, pulsando los botones que existen en la parte inferior del panel.
- Arriba a la derecha se muestra un gráfico de líneas en el cual está representado, para cada estado de ánimo, el número de asistentes que han seleccionado dicho estado de ánimo en el momento actual, con los colores seleccionados por los configurados en los sentimientos en la presentación. Este gráfico también puede representar la media de los asistentes en cada estado de ánimo si se pulsa el botón inferior “Mostrar media”.
- Abajo a la derecha se muestra un icono que representa el estado con mayor frecuencia en los asistentes en el estado actual, para conocer de forma rápida el estado predominante en la presentación. También puede mostrar el icono del estado con mayor valor de la media de los estados de ánimo, si se pulsa el botón inferior “Mostrar media”.

Tras el panel general, se muestran los paneles de información filtrados por los filtros configurados para la ponencia. Se muestra la misma información que en el panel anterior, pero filtrando la información sólo seleccionando los usuarios que cumplan los criterios del filtro configurado.

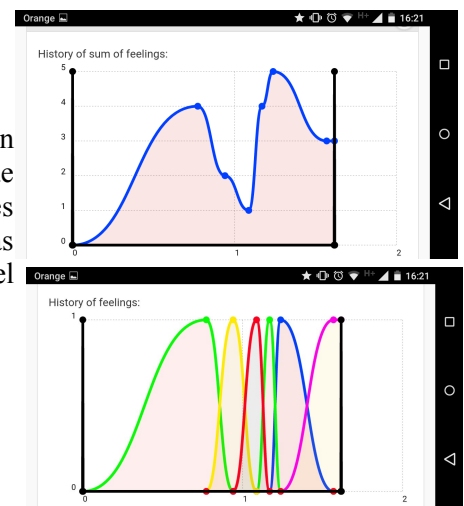
Estos datos se guardan en la base de datos NoSQL configurada en la aplicación, por lo que los datos estarán disponibles tanto en el transcurso de la ponencia como al finalizar la ponencia, haciendo posible variar los filtros usados una vez finalizada la ponencia para un análisis más profundo de la presentación realizada.



El presentador también puede lanzar al público las preguntas configuradas en la presentación, lo cual hará que a los asistentes se les active la vista de pregunta, y cuando respondan el ponente tendrá disponibles las gráficas de barras que le den información sobre las respuestas realizadas (actualizadas en tiempo real), además de conocer el número de asistentes que respondió correctamente a las preguntas y el número de asistentes que falló.

Análisis para asistentes

Una vez finalizada la ponencia, los asistentes (si tienen configurado un usuario para la aplicación) tendrán acceso al histórico del estado de ánimo que seleccionaron en la ponencia, para poder analizar qué partes de la ponencia fueron más interesantes y cuales menos. Las gráficas mostradas son las mismas que las secciones de la izquierda para el ponente (pero sólo mostrando datos del usuario conectado)



4.- Conclusión

Uniando las diferentes técnicas de ingeniería de software estudiadas en este documento se ha obtenido un sistema robusto y adaptado a las necesidades de quienes van a usarlo. Se han utilizado tecnologías y metodologías eficaces que para ayudar al análisis, desarrollo e implementación del sistema de información, lo cual ha aumentado la calidad del proyecto.

Como resultado se ha conseguido un sistema que facilita y enriquece las presentaciones, reuniones o ponencias donde se utilice el programa. Permite a la persona que de la ponencia poder adaptar la presentación a los estados de ánimo del público, además de analizar dichos datos tras la ponencia. También permite responder a preguntas que lance el ponente en tiempo real, dando más herramientas al ponente para cumplir las expectativas de la ponencia con el público. Por lo tanto se han conseguido los objetivos del proyecto planteado.

- **El sistema desarrollado ya terminado se ha dejado disponible para su uso en la URL: <http://bit.do/fidbac>**
- **Además, todo el código desarrollado se ofrece bajo licencia “GNU/GPL v3” en el repositorio abierto de GitHub: <http://github.com/frandai/fidbac>**

4.1.- Uso del sistema en un entorno real

A continuación se presentan las ventajas de implantar el sistema descrito en un entorno real. El sistema descrito en el proyecto ha sido usado en una reunión de trabajo de empresa, aportando los siguientes beneficios:

- El ponente ha podido adaptar la presentación a los estados de ánimo los asistentes, agilizando las secciones de la ponencia en la que los asistentes mostraban un menor estado de ánimo o haciendo paradas de 5 minutos si los asistentes expresaban cansancio a través de la aplicación.
- El ponente ha podido hacer una presentación adaptada a los conocimientos de los asistentes gracias a las preguntas hechas a través del sistema: al existir un desconocimiento de ciertas tecnologías, se ha centrado en explicar dichas tecnologías que otras que ya se conocían.
- El ponente ha podido analizar diferentes tipos de asistentes, pudiendo aprender para futuras ponencias qué contenidos son más interesantes para cada tipo de asistente (según su rol en la empresa).
- El ponente ha tenido un mayor control sobre su ponencia dado que ha podido controlar el tiempo y tener presente cuánto llevaba cada sección.
- Los asistentes han tenido información complementaria sobre las ponencias y las secciones gracias a la información mostrada en la aplicación.
- Los asistentes han podido enviar sugerencias o preguntas de forma anónima en el transcurso de la ponencia, permitiendo al ponente responderlas y enriqueciendo la formación del público.

4.2.- Líneas futuras

A continuación se presentan las líneas de trabajo futuras para este proyecto:

- Escalabilidad: Este sistema ha sido usado con un número relativamente reducido de usuarios de forma concurrente. Si se quisiera usar de forma masiva (para analizar programas de televisión u otros medios de comunicación masivos) debería de desarrollarse una arquitectura que soportara la escalabilidad del sistema. Tanto la base de datos NoSQL usada como la base de datos relacional están preparadas para utilizarse de forma escalable, pero el sistema debería de ser adaptado para este uso (gestión de memoria, duplicidad, consistencia, etcétera).
- Internacionalización: Para cumplir el objetivo de a la internacionalización en un futuro, el aspecto del portal web sigue un diseño claro, conciso y minimalista. De acuerdo con los estudios de Jan M.Pawlowsky [28] sobre las investigaciones de Geert Hostfede, los países con diferentes dimensiones culturales reflejan diferentes apariencias en sus páginas web. Aunque Europa posea gran diversidad de regiones que le da multitud de dimensiones culturales [29], debido a que se engloban dentro de un paradigma cultural común, la apariencia del sistema es favorablemente aceptada en Europa [30]. Para el proceso de internacionalización se pueden usar “Bundles” integrados en java [31].
- Funcionalidades futuras: Al ser un proyecto de Software Libre (liberado bajo licencia GPL 3.0) es posible que se genere una comunidad de desarrolladores para aumentar la funcionalidad del sistema. Como objetivos funcionales futuros del sistema se encuentran: aumentar la funcionalidad de los filtros, permitir realizar copias de información de presentaciones, aumentar la utilidad de la información para el ponente, etcétera.

5.- Bibliografía

1. Torres M. Diseño orientado a objetos. UNAD. http://datateca.unad.edu.co/contenidos/204023/Torres_M._Diseno_Orientado_a_Objeto.pdf. Accedido el 7 de Julio del 2015.
2. Meyer, Bertrand (1988). Object-Oriented Software Construction. Cambridge: Prentise Hall International Series in Computer Science. p. 23. ISBN 0-13-629049-3
3. Cockburn, Alistair. Agile Software Development. Highsmith Series.
4. Anónimo. Programación extrema. <http://www.extremeprogramming.org/>. Accedido el 7 de Julio del 2015.
5. César Arturo Guerra. Obtención de Requerimientos. Técnicas y Estrategia. SG. <http://sg.com.mx/revista/17/obtencion-requerimientos-tecnicas-y-estrategia>. Accedido el 7 de Julio del 2015.
6. E. F. Codd, The Relational Model for Database Management, Addison-Wesley Publishing Company, 1990, ISBN 0-201-14192-2
7. Barker, Richard (1990). CASE Method: Entity Relationship Modelling. Addison-Wesley. ISBN 0201416964.
8. Ricard Lou Torrijos. Introducción a la Tecnología JavaServer Faces. Programación en castellano. http://programacion.net/articulo/introduccion_a_la_tecnologia_javascript_faces_233. Accedido el 7 de Julio del 2015.
9. Ed Burns (2005). JavaServer Faces: The Complete Reference.
10. Riehle, Dirk (2000), Framework Design: A Role Modeling Approach. <http://www.riehle.org/computer-science/research/dissertation/diss-a4.pdf>. Accedido el 7 de Julio del 2015.
11. K. Siva Prasad Reddy (2013). PrimeFaces Beginner's Guide.
12. Christian Bauer (2006). Java Persistence with Hibernate
13. Doug Clarke. Introducing EclipseLink. ECLIPSE ZONE. <http://eclipse.dzone.com/articles/introducing-eclipselink>. Accedido el 7 de Julio del 2015.
14. Scott W. Ambler. Mapping Objects to Relational Databases: O/R Mapping In Detail. <http://www.agiledata.org/essays/mappingObjects.html>. Accedido el 7 de Julio del 2015.
15. Anónimo. JPA Queries. Objectdb. <http://www.objectdb.com/java/jpa/query>. Accedido el 7 de Julio del 2015.
16. MongoDB. What is NoSQL?. <https://www.mongodb.com/nosql-explained>. Accedido el 9 de Septiembre de 2015.
17. Jeffrey Dean and Sanjay Ghemawat (2004). MapReduce: Simplified Data Processing on Large Clusters. <http://research.google.com/archive/mapreduce.html>. Accedido el 7 de Julio del 2015.
18. Database Management System Concept. FK Publications
19. Mathieu Carbou (2011). IBM. Reverse Ajax, Part 4: Atmosphere and CometD. <http://www.ibm.com/developerworks/library/wa-reverseajax4/wa-reverseajax4-pdf.pdf>. Accedido el 7 de Julio del 2015.
20. Date, C. J. (1999), An Introduction to Database Systems (8th ed.). Addison-Wesley Longman. ISBN 0-321-19784-4.
21. Anónimo. Oracle GlassFish Server: Frequently Asked Questions.

- <http://www.oracle.com/us/products/middleware/application-server/oracle-glassfish-server-faq-071872.pdf> Accedido el 7 de Julio del 2015.
22. Install GlassFish on a CentOS 6 VPS. RoseHosting. <https://www.rosehosting.com/blog/install-glassfish-on-a-centos-6-vps/>. Accedido el 7 de Julio del 2015.
 23. Patrón de arquitectura Modelo Vista Controlador (MVC). Universidad Carlos 3 de Madrid. <http://www.lab.inf.uc3m.es/~a0080802/RAI/mvc.html>. Accedido el 7 de Julio del 2015.
 24. Luis Alberto Casillas. Bases de datos en MySQL. UOC. http://ocw.uoc.edu/computer-science-technology-and-multimedia/bases-de-datos/bases-de-datos/P06_M2109_02151.pdf. Accedido el 9 de Septiembre de 2015.
 25. Anónimo. About SQLite. <https://www.sqlite.org/about.html>. Accedido el 9 de Septiembre de 2015.
 26. IETF. The WebSocket Protocol. <https://tools.ietf.org/html/rfc6455>. Accedido el 9 de Septiembre de 2015.
 27. Malte Ubl. Introducción a los WebSockets: incorporación de sockets a la Web. <http://www.html5rocks.com/es/tutorials/websockets/basics/#toc-introduction-sockets>. Accedido el 9 de Septiembre de 2015.
 28. Dr. Jan M. Pawlowski (2012) Universidad de Jyväskylä. Global Information System: Localization and Internationalization.
 29. Geer Hofstede, Michael Minkov (2010.) Cultures and Organizations: Software of the Mind, Third Edition Paperback
 30. Eduardo José Corzo Peña (2014). UGR. Proyecto de fin de carrera: "Sistema de Información Adaptativo Basado en Realidad Aumentada".
 31. Javier Garbedo. Internacionalización con Bundles. <http://javiargarbedo.es/15-apuntes-java/ficheros/359-internacionalizacion-con-bundles>. Accedido el 9 de Septiembre de 2015.

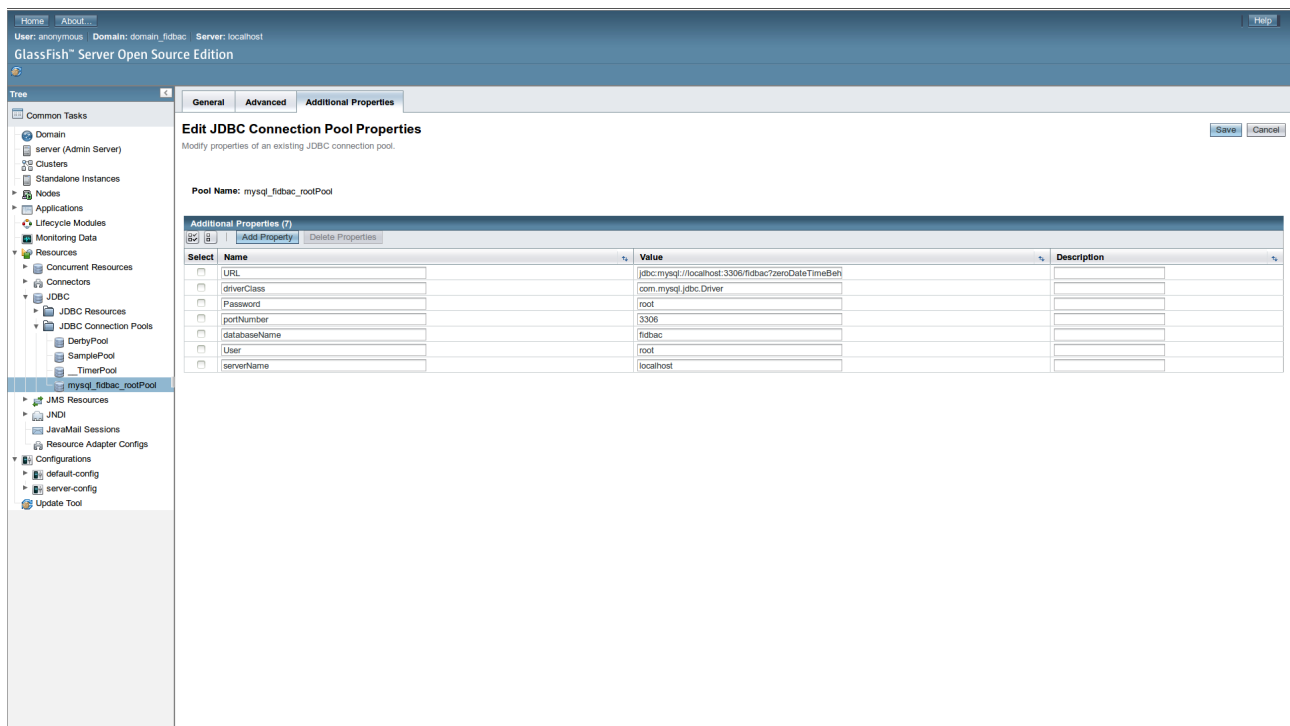
Anexo I - Guía de Instalación

A continuación se explican la forma de instalar el sistema en un entorno de producción.

Configuración de GlassFish y MySQL en un servidor

Si se desea instalar el sistema en un servidor, es necesario seguir los siguientes pasos en el sistema operativo en el que se desea instalar el sistema:

- Es necesario instalar el paquete de Java Development Kit versión 7 (JDK 7) de Oracle.
- Es necesario instalar el paquete de MySQL versión 14.14.
- Es necesario instalar GlassFish 4.1. Al instalarlo crearemos el dominio “domain1” [22].
- Una vez instalado el software necesario, se configura para poder ejecutar el sistema:
 - Para instalar la base de datos, se debe crear una base de datos llamada “fidbac” en el Sistema Gestor de Base de Datos de MySQL. En esta nueva base de datos se cargará el esquema relacional guardado en el archivo SQL “fidbac.sql” (incluido en este proyecto).
 - Para instalar la implementación Java del sistema, deberemos de entrar en la interfaz de administración de GlassFish. A esta interfaz se puede acceder desde el puerto 4848 del servidor (127.0.0.1:4848).
 - Primero se debe de crear la conexión a la base de datos creada. Para ello debe de crearse un nuevo Pool de conexiones JDBC desde *Recursos* → *JDBC* → *Pools de Conexiones JDBC*, pulsando el botón *Nuevo...* (*Pool JDBC*).
 - Nombre de Pool: **mysql_fidbac_rootPool**
 - Tipo de Recurso: **javax.sql.DataSource**
 - Proveedor de Controladores de la Base de Datos: **com.mysql.jdbc.jdbc2.optional.MysqlDataSource**
 - Para que reconozca esta Clase de Origen de Datos, es posible que se deba de incluir la biblioteca “*mysql-connector-java-5.1.23-bin.jar*” dentro de la ruta de glassfish: */glassfish4/glassfish/domains/domain1/lib/*.
 - Propiedades Adicionales:
 - URL: **jdbc:mysql://localhost:3306/fidbac?zeroDateTimeBehavior=convertToNull**
 - driverClass: **com.mysql.jdbc.Driver**
 - Password: **[password del usuario de la base de datos]**
 - portNumber: **3306**
 - databaseName: **fidbac**
 - User: **[usuario de la base de datos]**
 - serverName: **localhost**



- Tras esto, se añade la conexión de base de datos para que la aplicación pueda conectarse con la base de datos. Para ello debe de crearse un nuevo Recurso JDBC desde *Recursos* → *JDBC* → *Recursos JDBC*, pulsando el botón *Nuevo...* (*JDBC*).
- Debe de configurarse con las siguientes propiedades:
 - Nombre JNDI: **1**
 - Nombre de Pool: **mysql_fdbac_rootPool**
- Una vez creada la conexión con la base de datos, se procede a desplegar la aplicación en el sistema. Para ello se siguen los siguientes pasos:
 - En la interfaz de administración de GlassFish, se accede a *Aplicaciones* → *Desplegar...* (*Nueva Aplicación*).
 - En *ubicación*, se selecciona la aplicación a desplegar “**fdbac.war**” (incluida en este proyecto).
 - El atributo *Raíz de Contexto* podemos dejarlo vacío o seleccionar una ruta de raíz de contexto.
 - Pulsamos “Aceptar”, y ya estará desplegada la Aplicación en el servidor GlassFish.
- Para que funcione correctamente la gestión de presentaciones en tiempo real, es necesario activar “Websockets” y “Commet” en la configuración del conector “HTTP-LISTENER-1”, dentro del apartado “HTTP-Service”.

Configuración de MongoDB en un servidor

MongoDB es la base de datos NoSQL utilizada en la aplicación. A continuación se indican los pasos para instalar este sistema de base de datos en sistemas operativos CentOS:

Se añade el repositorio donde se aloja MongoDB a yum

```
printf "[mongodb]\nname=MongoDB Repository\nbaseurl=http://downloads-  
distro.mongodb.org/repo/redhat/os/x86_64/ \ngpgcheck=0\nenabled=1" >  
/etc/yum.repos.d/mongodb.repo
```

Se instalan MongoDB y MongoDB-server

```
yum -y install mongodb-org mongodb-org-server
```

Se crea el directorio necesario que utiliza por defecto MongoDB

```
mkdir /data  
mkdir /data/db
```

Se inicia por primera vez MongoDB

```
mongod
```