UNIVERSITAT DE BARCELONA

Treball final de grau

GRAU DE INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

Slash and Pair móvil: Arquitectura de una aplicación web

Autor: Carlos Arenas Ferriz

Tutors:	Guillermo Blasco
	Guillem Palomar
	Pablo Martínez
Realitzat a:	Departament de Matemàtiques
	i Informàtica

Barcelona, 21 de juny de 2017

Abstract

Internet is one of the biggest inventions of the human kind, from its beginning it has moved forward in many directions until it has grown into something quite unmanageable. Initially it was just for a very few because devices that had network access were not available for the vast majority. This starts changing and becoming popular with time, until reaching the point that nowadays pretty much everyone has several devices from which to access the Internet, each of these with different features.

For this exact reason, Slash and Pair is born, a web application committed to unify the strengths between the most used devices, such as computers and smartphones.

A computer has a large screen which is handier when viewing content and a smartphone has a series of sensors such as gyroscope or touch screen that make the access more agile. Slash and Pair pairs these two devices by sending sensor data from one to another in order to create a new usability concept. To achieve this, a study of quality standards and architectural patterns has been carried out. The project is divided in two parts, dedicated to the mobile performance and another to the computer operation. Throughout the document the whole development of the mobile part and an analysis of the system architecture is shown concluding in a software that achieves its purpose of matching and accomplishes the interaction between devices.

Resum

Internet és un dels grans invents de la humanitat, des dels seus inicis ha avançat en moltes direccions fins a arribar a ser gairebé inabastable. Al principi només era per a uns pocs ja que no tothom disposava d'ordinadors que tinguessin accés a la xarxa. Això va canviant amb el temps i es va popularitzant, fins al punt que avui en dia tothom té diversos dispositius des dels quals accedir a Internet cada un amb característiques diferents.

Per aquest motiu neix *Slash and Pair*, una aplicació web dedicada a unificar els punts forts entre els dispositius més usats com són els ordinadors i els mòbils. En un ordinador es disposa d'una gran pantalla en què és més còmode el visionat de contingut i des d'un *smartphone* es compta amb una sèrie de sensors com giroscopi o pantalla tàctil que fan l'accés més àgil.

Slash and Pair aparella aquests dos dispositius enviant les dades dels sensors d'un a un altre amb l'objectiu de crear un nou concepte d'usabilitat. Per aconseguir-ho s'ha realitzat un estudi dels estàndards de qualitat i patrons d'arquitectura. El projecte està dividit en dues parts, una dedicada al funcionament del mòbil i una altra a l'ordinador.

Al llarg del document es mostra tot el desenvolupament de la part mòbil i una anàlisi de l'arquitectura del sistema concloent en un programari que aconsegueix el seu propòsit d'aparellar i aconsegueix la interacció entre dispositius.

Resumen

Internet es uno de los grandes inventos de la humanidad, desde sus inicios ha avanzado en muchas direcciones hasta llegar a ser casi inabarcable. Al principio solo era para unos pocos ya que no todo el mundo disponía de ordenadores que tuvieran acceso a la red. Esto va cambiando con el tiempo y se va popularizando, hasta el punto que hoy en día todo el mundo tiene varios dispositivos desde los que acceder a Internet cada uno con características diferentes.

Por este motivo nace Slash and Pair, una aplicación web dedicada a unificar los puntos fuertes entre los dispositivos más usados como son los ordenadores y los móviles. En un ordenador se dispone de una gran pantalla en la que es más cómodo el visionado de contenido y desde un smartphone se cuenta con una serie de sensores como giroscopio o pantalla táctil que hacen el acceso más ágil.

Slash and Pair empareja estos dos dispositivos enviando los datos de los sensores de uno a otro con el objetivo de crear un nuevo concepto de usabilidad. Para conseguirlo se ha realizado un estudio de los estándares de calidad y patrones de arquitectura. El proyecto está dividido en dos partes, una dedicada al funcionamiento del móvil y otra al ordenador.

A lo largo del documento se muestra todo el desarrollo de la parte móvil y un análisis de la arquitectura del sistema concluyendo en un software que consigue su propósito de emparejar y logra la interacción entre dispositivos.

Agradecimientos

Quiero agraceder a Guillermo Blasco por ofrecernos la oportunidad de realizar este proyecto con el que he aprendido tanto y por toda su ayuda ya que sin él *Slash and Pair* no hubiera sido posible.

Agradecer a Guillem Palomar y Pablo Martínez por toda su ayuda y consejo.

A Victor mi compañero durante este trabajo, que nos ha costado tantas horas y esfuerzo, y a Sergio, mi compañero a lo largo de toda la carrera, sin ellos se hubiera hecho mucho más duro todo el trayecto.

A mi padre, madre y hermana por haberme apoyado incondicionalmente en cualquier situación, sin ellos no hubiera llegado donde estoy.

Quiero agradecer a Helena por estar ahí siempre apoyándome, ayudándome cuando siempre que ha hecho falta y dándome fuerzas cuando no me quedaban.

Tabla de contenido

Lista de figuras

Lista de tablas

1 Introducción

1.1 Contexto

En los inicios de internet se desarrolló la tecnología HTML(*HyperText Markup Language*) , cuyo creador es considerado Tim Bernes-Lee[1] a quien también se le atribuyen HTTP(*Hypertext Transfer Protocol*) y URL(*Uniform Resource Locator*).

La Web ha sido considerada un gran invento para la humanidad ya que ha conseguido hacer accesibles multitud de conocimiento y contenidos además de revolucionar la forma que se tenía de entender el mundo.

WorldWideWeb, el primer navegador para ordenadores *Next*, se publica en 1991 hecho que impulsa el desarrollo, como punto de partida para Internet tal como se conoce a día de hoy.

Se crearon las primeras páginas web estáticas entre 1992 y 1994, en las que la velocidad de internet era muy limitada (módems de 2,4 Kbps). En esta primigenia generación de la Web la navegación era poco estructurada y contaba con largos textos como si de una hoja de papel escrita se tratase.

Después de esta primeriza versión, se fueron añadiendo nuevas tecnologías con las que se empezaron a añadir imágenes, hecho que ayudó a hacer una web mucho más esquematizada. La aparición a posteriori de CSS(*Cascading Style Sheets*) y *JavaScript* supuso un gran avance ya que el usuario tenía mucha más interacción con la web.

Aproximadamente sobre 2003 apareció la web 2.0[2] en la que el usuario se convierte en lo más importante y pasa de ser un espectador pasivo a lograr una interacción activa donde sus acciones en la web tienen una repercusión dinámica.

Esto supuso una gran revolución ya que fueron creadas plataformas como por ejemplo *Youtube*, *Wikipedia*, *Flickr* o *Google*.

La aparición ulterior en 2007 de los teléfonos móviles[3] marcó el inicio de otra manera de acceder a la web desde dispositivos menos convencionales.

Actualmente su acceso desde el móvil ha crecido tanto que hoy en día en España se accede más desde smartphones que desde ordenadores clásicos [4]. A finales de 2016 el 51,3% de los accesos se realizan desde dispositivos móviles o tablets y el 48,7% desde ordenadores.

Observando la trayectoria de internet y las múltiples direcciones en las que se puede expandir nace *Slash and Pair*; un sistema que pretende aprovechar las ventajas de ambos dispositivos con la finalidad de crear una interacción novedosa.

El uso de un ordenador conlleva un seguido de ventajas contra las que un *smartphone* no puede competir y viceversa; para ejemplificar, en un ordenador se dispone de una pantalla de mayor tamaño en la que se hace más cómodo el visionado de

contenidos y en un dispositivo móvil se cuenta con nuevos métodos más ágiles para acceder a este, como son una pantalla táctil y diversos sensores de los que dispone cualquier *smartphone*.

Slash and Pair se propone unir los dos mundos combinando las ventajas de ambos buscando un óptimo a la vez que ergonómico fin.

Para ello se ha creado un sistema que genera una interacción entre ambos dispositivos, pudiendo ver las acciones que realiza un dispositivo en el otro, construyendo así una nueva forma de acceder a la web.

Pese a que se hable de ordenadores en el documento, actualmente se puede aplicar este término a cualquier dispositivo que tenga acceso a la web, ya sea una consola, una televisión, una tablet, etc.

1.2 Motivación

Establecido el contexto anterior, la motivación de este trabajo es clara; poder crear un sistema desde el cual se genere esta interacción de datos entre dispositivos.

Independientemente de la intención de crear una nueva forma de entender internet, esta propuesta supone un gran reto a nivel tecnológico ya que no es un asunto trivial y se requiere de multitud de tecnologías diferentes que aplicar, como se verá a lo largo del documento.

Puesto que es una nueva tecnología se debe realizar poniendo especial cuidado en la calidad de la implementación, motivo por el cual se realiza un análisis de la arquitectura y de los estándares de calidad en software, por si en un hipotético caso en un futuro se sigue con el proyecto no surjan problemas no contemplados.

1.3 Objetivos

Este trabajo tiene como objetivos investigar, entender y aplicar la mejor arquitectura posible para la implementación de un proyecto del software, así como los mejores *frameworks* disponibles y los patrones de diseño más adecuados.

El desarrollo del trabajo ha sido realizado entre dos personas ya que es un proyecto de gran envergadura y la colaboración entre ambas partes ha hecho posible un resultado final más ambicioso.

A lo largo del documento se tratará la arquitectura del proyecto y el desarrollo de la parte destinada al funcionamiento de dispositivos móviles dentro del sistema.

1.4 Estructura del documento

En este apartado se resume la estructura del documento comentando brevemente el contenido de cada capítulo.

Previo a este apartado está la introducción en la que se pone en contexto al lector, para un mejor entendimiento del proyecto, también se comentan las motivaciones

que han llevado a la elección del tema que se ha desarrollado y los objetivos a conseguir.

El segundo capítulo Especificaciones y Requisitos de Software define según un estándar IEEE 830-1998[5], las especificaciones y requisitos con los que ha de cumplir el software desarrollado. Con su lectura se puede hacer una idea del funcionamiento de *Slash and Pair*.

En el siguiente apartado se explican las metodologías de desarrollo que se han seguido y la planificación. Ya que ha sido un desarrollo en equipo, se comenta la forma en la que se ha trabajado para llevar el proyecto adelante.

En el capítulo de análisis se ha realizado una descripción del proyecto mediante diagramas de casos de uso, explicando las funciones a nivel general, centrándose sobretodo en el ámbito del móvil ya que es la parte implementada en este trabajo.

Seguidamente se detalla el diseño del sistema, en el se da una descripción del mismo sistema y se realiza un análisis de la arquitectura, proponiendo las dudas que han ido surgiendo durante el desarrollo y explicando las soluciones que se han tomado y por qué motivo. Se hace un repaso de las tecnologías utilizadas así como el porqué se han utilizado y por último en el capítulo se da una breve explicación sobre cómo están integradas todas ellas para funcionar en el mismo proyecto.

A continuación, en el capítulo de ejemplos de ejecución se describe la adaptación del software desarrollado a otro software externo. En cualquier sistema externo en el que se implemente *Slash and Pair* se hace un uso de los datos dando sentido y demostrando las capacidades del proyecto realizado.

Llegando al final del documento se encuentran el resumen y las conclusiones a las que se ha llegado al final de todo el desarrollo del proyecto.

En el último apartado se comentan las líneas de futuro que se pueden seguir a partir de la finalización de *Slash and Pair*.

2 ERS

En el apartado anterior se ha introducido el proyecto comentando el contexto en el que se está en el momento de la realización del mismo, dando los motivos por los cuales se ha hecho la elección de este tema y comentando los objetivos a cumplir

En este capítulo se detallan las Especificaciones y Requisitos del Software según el estándar *IEEE 830-1998*, donde se da una visión del funcionamiento de *Slash and Pair* y los requisitos que ha de cumplir.

2.1 Descripción general

En está sección se presenta una descripción a alto nivel del software del proyecto. Se presentan las principales funciones así como las restricciones, dependencias y otros factores que afectan al desarrollo del sistema.

2.1.1 Perspectiva de producto

Este software está diseñado para estar alojado en un servidor web, desde el cual se accede desde un dispositivo que hace las funciones de monitor, normalmente un ordenador convencional en el que disponemos de una pantalla mayor, y desde un dispositivo móvil como puede ser cualquier *smartphone*, en el que se utilizarán los diferentes sensores de este.

El sistema depende de un navegador web capaz de ejecutar código *Javascript* en ambos dispositivos.

La interacción con el usuario debe ser con acciones realizadas a través del *smartphone* que tendrán un resultado en el dispositivo de mayor pantalla. Dichas acciones serán diferentes dependiendo del software que se ejecute en último término.

2.1.2 Funciones del sistema

El sistema tiene como funcionalidades el emparejamiento entre dos dispositivos, la recogida de datos de los sensores de un *smartphone* y el envío de datos del dispositivo móvil al ordenador emparejado.

El emparejamiento tendrá la función de asegurar el envío de datos entre dispositivos de un mismo usuario, evitando que sus datos se envíen a otros o recibir datos no deseados. Dicha función se iniciará ofreciendo una interfaz simple en la que se le darán instrucciones al usuario de los pasos a seguir desde ambos dispositivos.

La recogida de datos tendrá la función de extraer la información proporcionada por los diferentes sensores del *smartphone*.

El envío de datos transmite la información del dispositivo móvil al ordenador del usuario.

2.1.3 Características de los usuarios

El sistema deberá ofrecer una interfaz sencilla para que todo usuario sin conocimientos técnicos pueda hacer uso de él y logre realizar el emparejamiento de dispositivos.

Sería ideal que en todos los casos, cualquier tipo de usuario pueda acceder a los beneficios del sistema desarrollado, ya que se basa en la intuición.

2.1.4 Suposiciones y dependencias

Se asume que el dispositivo móvil que utiliza el usuario tiene acceso a los sensores necesarios para el correcto funcionamiento y estos son accesibles desde el navegador, como por ejemplo el giroscopio.

Se asume que el dispositivo monitor dispone de un navegador capaz de ejecutar *Javascript*.

Los dos dispositivos deben tener conexión estable a internet para el correcto funcionamiento del software.

2.2 Requisitos específicos

En este apartado se presentan los requisitos funcionales que deberán ser satisfechos.

2.2.1 Interfaces externas

En ambos dispositivos se requiere de un navegador con soporte para *Javascript*.

En caso del móvil se requiere que sean accesibles por el navegador los sensores que se utilicen.

En el funcionamiento ideal del emparejamiento se necesita de acceso a una cámara, aunque esto no es totalmente necesario ya que si no se tiene acceso, se ha de ofrecer una vía alternativa.

En caso de no disponer de la cámara ya sea porque el dispositivo no sea compatible o suceda algún error, se mostrará una manera alternativa de realizar el emparejamiento.

2.2.2 Restricciones de diseño

El sistema tiene unas restricciones de diseño por ser una aplicación web.

No se tiene acceso a manipular eventos como el ratón desde el escritorio ya que por seguridad que implementa el navegador no se tiene acceso a los eventos del sistema.

En el dispositivo móvil se puede acceder solamente a los sensores que estén disponibles a través de las *API JavaScript* disponibles para cada navegador.

Dependiendo del dispositivo no se tiene acceso a la cámara por lo cual se ha de ofrecer una alternativa para realizar el emparejamiento de dispositivos, esto puede suponer perder en seguridad.

2.2.3 Requisitos funcionales

Emparejamiento

REQ1	Inicio de la aplicación dispositivo ordenador: en primer lugar el usuario introduce el enlace en el navegador. Se inicia el proceso de emparejamiento en el servidor de escritorio, este genera una sesión y un código de emparejamiento.
REQ2	El servidor con el código de sincronización genera un código QR que se le muestra al usuario en el navegador del ordenador.
REQ3	Inicio de la aplicación dispositivo móvil: en primer lugar el usuario introduce el enlace el navegador.
REQ4	Se inicia el sistema el cual genera una sesión para móvil.
REQ5	El sistema enciende la cámara del smartphone con la cual se leerá el código qr.
REQ6	El usuario con la cámara encendida enfoca al código qr que se muestra en el ordenador, éste reconoce el código automáticamente sin ninguna acción por parte del usuario, y se envía al servidor.
REQ7	El servidor valida el código QR leído y realiza el emparejamiento de dispositivos.
REQ8	Una vez realizado el emparejamiento correctamente entre los dos dispositivos, el dispositivo móvil se redirecciona a una página nueva desde la cual se enviarán los datos de los sensores.
REQ9	Se envía un mensaje desde el servidor al ordenador informando de que un dispositivo móvil se ha conectado.
REQ10	En el navegador del ordenador será redireccionado a una nueva página en la que se podrán mostrar unas pequeñas instrucciones de funcionamiento. Si esto no resulta necesario se iniciará la aplicación externa que aprovecha los datos enviados para tener una funcionalidad.
REQ11	Fracaso en el emparejamiento de dispositivos. En el dispositivo móvil se muestra un mensaje de error y se vuelve a preparar la cámara para realizar una nueva lectura del código qr mostrado en el ordenador. Si el error viene dado por la falta acceso a la cámara por parte del dispositivo móvil, se le ofrecerá un camino alternativo para realizar el emparejamiento.

Tabla 1: Requisitos de emparejamiento

Comunicación

REQ12	El dispositivo móvil envía los datos de los sensores al servidor para que este los convierta en objetos y pueda enviarlos al ordenador de forma continuada.
REQ13	El servidor del ordenador recibe los datos y los interpreta.
REQ14	Los datos son enviados del servidor al navegador del ordenador.
REQ15	Un software externo debe poder hacer uso de los datos de los sensores proporcionados para realizar acciones.

Tabla 2: Requisitos de interacción.

2.2.4 Requisitos de rendimiento

El sistema debe estar preparado para una gran transmisión de datos, ya que el envío de la información de los sensores del dispositivo móvil, dependiendo de cuáles se utilicen, será en una alta frecuencia.

El sistema debe poder ser escalado si el número de usuarios crece en gran medida.

El sistema debe de dar respuesta en tiempo real. Si se realiza una acción desde el *smartphone*, inmediatamente deberá verse reflejada en el ordenador.

3 Metodología y planificación

En el apartado anterior se han detallado las especificaciones del estándar *IEEE 830-1998*, para dar una visión del proyecto al completo sin entrar en aspectos técnicos complejos.

En este capítulo se explica la metodología y planificación del proyecto de manera que se pueda ver por todas las fases por las que se ha pasado.

3.1 Tipo de metodología

El desarrollo del proyecto se basa en metodología agile, concretamente en *Desarrollo iterativo e incremental*[6]. Este proceso de desarrollo de software consiste en un conjunto de tareas agrupadas en pequeñas iteraciones.

Esta metodología permite realizar una evolución sobre el producto de una manera incremental de modo que al final de cada iteración se obtiene un software funcional con el que se puede realizar una demostración. Cada iteración se puede considerar un pequeño proyecto.

El desarrollo iterativo e incremental dispone de diferentes etapas, la etapa de planificación, etapa de iteración y etapa de despliegue, dentro de la segunda etapa existen las fases de análisis de la iteración, desarrollo y evaluación como se puede ver en la figura (1).

Figura 1: Página Desktop con código.

La etapa de preparación conlleva la aceptación del proyecto, se revisan los objetivos a alcanzar y se realiza un calendario de iteraciones en las que los miembros del equipo se reunirán al final de cada una de ellas. Se definen los roles de los miembros en el equipo. Esta primera etapa se realiza solo al inicio del proyecto.

Las etapas de planificación, desarrollo y evaluación forman parte de cada una de las iteraciones.

En la etapa de planificación se definen los objetivos de la iteración y se analizan las posibles mejoras sobre la anterior(si la hubiera), se define el trabajo a realizar durante la etapa de desarrollo y se distribuyen las funciones entre los miembros del equipo. En cada una se van añadiendo funcionalidades nuevas de manera que cada vez se hace un software más completo.

Durante la etapa de desarrollo, se implementan todas las funcionalidades definidas en la fase previa. Cada miembro del equipo ha de cumplir con lo que se ha analizado previamente.

En la etapa de evaluación, se analizan las nuevas funciones implementadas, se revisa el trabajo realizado durante la iteración y se anotan los posibles problemas y dificultades surgidas para tenerlo en cuenta en la siguiente iteración.

Los objetivos que se deben cumplir en cada iteración han de ser a corto plazo lo que ayuda positivamente en la productividad del equipo.

Cuando se llega a la etapa de despliegue, ya se ha terminado todo el desarrollo de la aplicación, por lo tanto se dispone de un producto final el cual ya está preparado para entrar en un entorno de producción.

Se ha elegido esta metodología ya que slash and pair es un proyecto que puede tener un gran crecimiento y un desarrollo de una duración superior a la que se disponía, por lo tanto teniendo un producto al final de cada iteración se aseguraba de disponer en todo momento de un software funcional.

En el caso del desarrollo de *Slash and Pair*, se ha seguido este proceso, las iteraciones han tenido una duración de dos semanas cada una, en las que se realiza una reunión en la que se evalúa el trabajo realizado y se analiza los objetivos a cumplir en la siguiente iteración.

Como ayuda para la organización del equipo, se ha contado con la herramienta *Trello*[7] que ha sido la plataforma utilizada para definir las tareas mediante tarjetas en las que se introduce una descripción de la tarea y se asigna a un miembro del equipo. Mediante esta herramienta se puede tener un seguimiento de cómo avanza el proyecto y cada iteración o si surgen problemas ya que según avanza en sus tareas cada miembro, puede ir asignando la tarjeta a una columna con un estado, estos estados son, por hacer, en progreso, acabado, archivado.

Para el control de versiones de *Slash and Pair* se ha utilizado la herramienta Git[8] bajo la plataforma Github[9], mediante la cual se realizan subidas del código que se va implementando, cosa muy útil cuando existen varios miembros en el equipo de desarrollo.

3.2 Fases del proyecto

En este apartado se explican las fases por las que ha pasado el proyecto. Como se ha comentado en el apartado anterior, se ha desarrollado en tres fases: preparación,

iteraciones y despliegue. A continuación se detallan las tres fases por las que ha pasado el proyecto.

3.2.1 Fase de planificación

En esta primera fase, se presenta la idea del proyecto. Aquí empieza la distribución de tareas para la construcción del proyecto *Slash and Pair*.

Una vez planteada, se comienza con el análisis funcional de cómo debería de ser el producto a realizar, para ello se preparan unos *storyboard* iniciales en los que se detalla cómo sería la interacción del usuario con el sistema definiendo todas las acciones que ha de realizar para cada paso que se ha de dar en el sistema para hacer uso de el.

Después de definir la funcionalidad, se pasan a analizar las tecnologías necesarias para llevar a cabo el desarrollo, de plantean opciones que pueden ayudar en la implementación aunque después, durante las iteraciones se van definiendo mejor.

En esta fase se plantea separar la implementación del software en dos proyectos cada cual funciona por separado y el reto a cumplir es la comunicación entre estos dos sistemas.

3.2.2 Fase de iteración

Una vez superada la fase de planificación en la que se da por cerrada la definición del proyecto, toca dar inicio a la fase de iteración, en la que comienza la implementación de *Slash and Pair*.

En cada iteración se ha obtenido un producto más funcional que en la anterior.

A continuación se pasa a detallar las iteraciones por las que ha pasado *Slash and Pair*.

Primera iteración En la iteración inicial, se propone comenzar con la creación del proyecto base a partir del cual se irá implementando todas las funciones.

Se crea un proyecto *Java* basado en lo que será la base del proyecto, usando *Spring Framework*[10]. El primer objetivo es seguir el tutorial de inicio de *Spring*. El segundo objetivo a cumplir sobre el proyecto creado es implementar *WebSockets*[11] siguiendo también los tutoriales proporcionados por la documentación oficial.

Al final de la iteración se tiene un software basado en *Spring* en el que se ha implementado *WebSockets*. La funcionalidad de la que dispone es una comunicación desde el navegador de un dispositivo, en la que enviar un mensaje de texto al servidor y este contesta pintando en el cliente una respuesta definida a este mensaje. Esto no tiene en cuenta a usuarios ya que el servidor realiza un broadcast a todo dispositivo conectado.

Segunda iteración En esta segunda iteración, se parte de los resultados obtenidos de la anterior. En la fase de análisis, se llega a la conclusión de que el siguiente paso a implementar es la comunicación entre dispositivo, enfocándose a la funcionalidad de *Slash and Pair*, por lo tanto, se deben extraer los datos de sensores, y enviarlos a servidor para que este los muestre por consola.

En el desarrollo, se busca información de las API disponibles para *JavaScript* que permiten obtener los datos de los sensores.

Una vez se dispone de ellos, se envían al servidor mediante los *WebSockets*.

Al final de la iteración se tiene un software desde el cual realizando acciones con el dispositivo, se obtiene una respuesta desde el servidor.

Tercera iteración En la tercera iteración se propone dar un avance grande al desarrollo del software. Se reestructura todo el código y se crea un proyecto intermedio entre el enfocado al escritorio y el enfocado a móvil. Desde este, se crean clases que tendrán en común los dos proyectos.

Se crea un proyecto padre, en el cual se incluyen los tres que se han citado. Desde este se definen todas las dependencias comunes, para así poder desarrollar a partir de las mismas versiones de software.

Una vez distribuido todo el código, se crea la forma en la que distinguir a los usuarios desde el servidor, por lo tanto se crean las sesiones de usuario.

Se implementa un sistema en el que guardar toda la información que se ha de compartir entre los dos proyectos(móvil y escritorio), con lo cual para comenzar en este avance, se siguen los tutoriales de *RabbitMQ*[12] y *Redis*[13].

Al final de esta iteración se tiene un software desde el cual se puede acceder desde un dispositivo y enviar información de los sensores.

Este dispositivo conectado tiene una sesión creada, aunque de momento el servidor no está preparado para contestar a este usuario en concreto.

Se implementa la cola *RabbitMQ* en la cual se envían los datos y una base de datos *Redis* en la cual se guarda el identificador de sesión de cada usuario y el código de emparejamiento creado por el escritorio.

Cuarta iteración En la cuarta iteración se ha de definir la estructura de los datos. Para realizar la comunicación entre los proyectos, se ha de realizar el desarrollo para que los datos enviados desde un dispositivo móvil solo lleguen a un ordenador y se comunique a todos como hasta ahora.

Para preparar los datos para el envío, se define una estructura de *JSON* que será como los datos viajen de un servidor a otro.

Con la sesiones de usuario que ya se implementaron en anteriores iteraciones, se implementa la cola de *RabbitMQ* para que solo recoja los datos que van dirigidos al usuario.

Se corrige la implementación de los *WebSockets* para que no hagan *broadcast* y

solo se comuniquen desde el servidor al cliente que toca.

Al final de esta iteración se dispone de un software con el que se puede realizar una comunicación entre dispositivos,y enviar los datos del dispositivo móvil al servidor móvil mediante la cola. El servidor de escritorio escucha la cola y envía los datos leídos al ordenador. El software es capaz de realizar el emparejamiento entre los dispositivos.

Quinta iteración En la quinta iteración se realizan pruebas de la aplicación tanto de funcionalidad como de rendimiento. Los dos servidores están desplegados localmente en un ordenador donde se conectan los dispositivos.

Se identifica que hay problemas en el envío de datos desde el móvil dependiendo de la red a la que esté conectado el sistema: se observan unas limitaciones en conexiones lentas y se para la conexión.

También se identifica que en el dispositivo móvil si se apaga la pantalla, se corta la conexión.

En esta iteración se identifican los problemas y se buscan soluciones a implementar en siguientes iteraciones.

Sexta iteración En la sexta iteración se decide implementar medidas de seguridad en el emparejamiento.

Para ello se implementa un lector de códigos QR en el dispositivo móvil y un generador en el lado del ordenador.

Detrás de los códigos QR hay un código de 32 caracteres de longitud.

También se implementa HTTPS, para tener una conexión más segura. Además, en dispositivos en los que se utiliza *Google Chrome* de navegador, para poder iniciar la cámara se requiere de HTTPS.

Se detecta que en algunos dispositivos no se puede acceder a la cámara, como es el caso de dispositivos IOS, en el que con el sistema implementado no es compatible con esta opción, por lo tanto se implementa una forma alternativa de introducir el código para el emparejamiento. La solución pasa por la creación de un código de X dígitos para realizar el emparejamiento. Esta opción la escogerá el usuario al apretar el botón correspondiente indicando que no tiene acceso a la cámara.

Al final de está iteración se dispone de un software desde el cual se puede realizar una conexión entre dispositivos a través de código QR mediante conexión HTTPS, y se envían los datos de un dispositivo a otro.

Séptima iteración En esta séptima y última iteración el sistema *Slash and Pair* ya funciona completamente, a falta de la implementación de un juego para la demostración.

Este es el objetivo de esta iteración, en la que se busca un software candidato para adaptar el sistema.

Una vez encontrado, se implementan las funciones necesarias para que el juego haga uso de los datos proporcionados por *Slash and Pair*.

3.2.3 Fase de despliegue

Como última fase del proyecto, está el despliegue de la aplicación en un entorno final. Una vez acabadas todas la iteraciones se da por acabado el desarrollo de la aplicación, por lo tanto el siguiente paso es poner *Slash and Pair* en producción.

3.3 Diagrama de Gantt

En la figura (2) se puede observar en un diagrama de *Gantt* como se ha distribuido la carga de trabajo a lo largo de la implementación del proyecto.

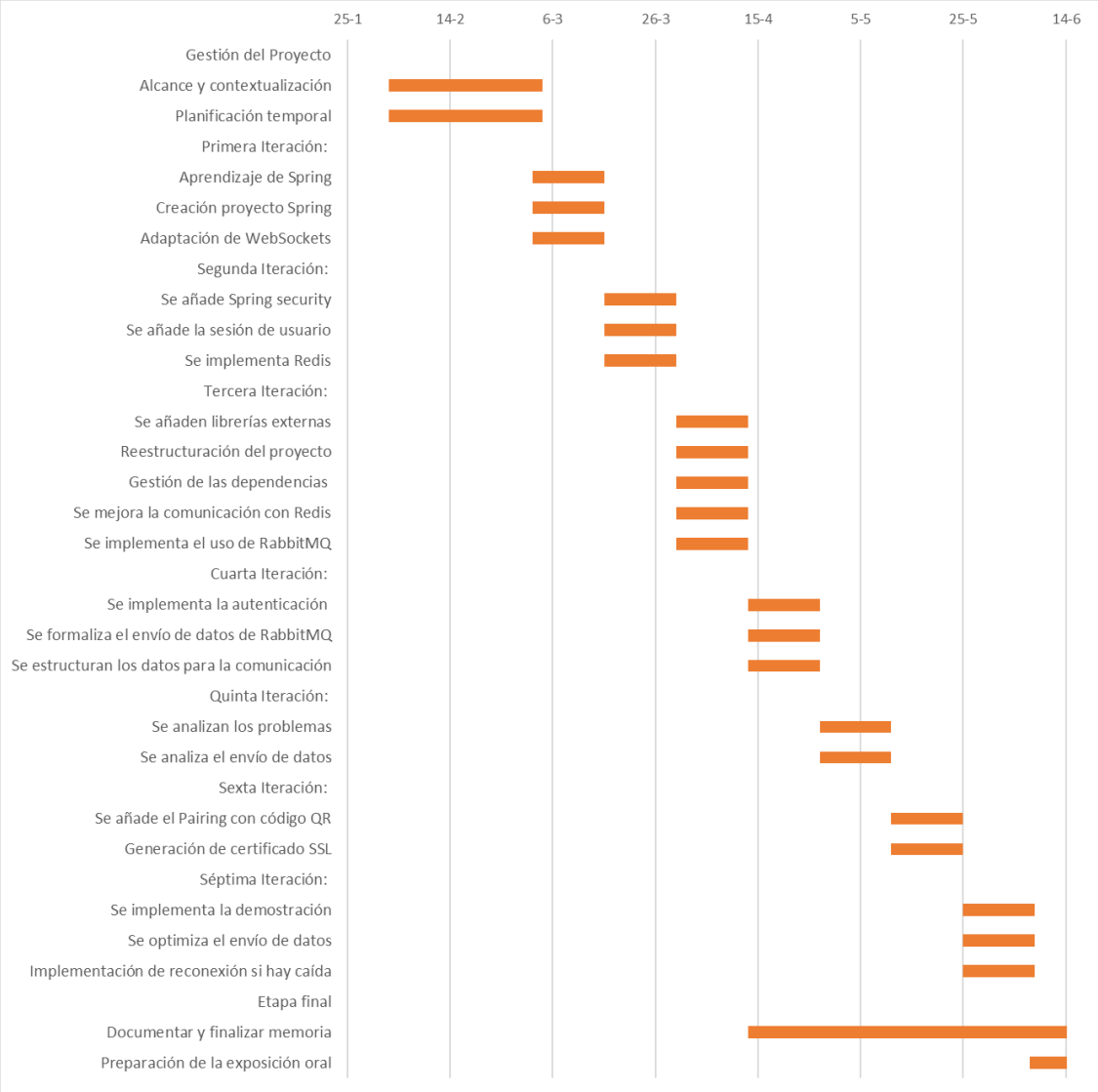

Figura 2: Diagrama de Gantt

4 Analisis

En el capítulo anterior se ha descrito la metodología que se ha seguido en el desarrollo de la aplicación *Slash and Pair* y también se han detallado todas las fases del proyecto especificado las funcionalidades que se han ido incluyendo, los problemas encontrados y la toma de decisiones.

En este apartado se describe el funcionamiento del sistema en cuanto las acciones de un usuario. Para un mejor entendimiento se redactan casos de uso generales de la aplicación y específicos de la parte de la aplicación tratada en este documento.

4.1 Diagrama de casos de uso general

Se describen las acciones que realiza un usuario en la aplicación *Slash and Pair*. No se entra en detalle en el funcionamiento de cada una de ellas ya que esa información se encuentra en el caso de uso específico que se explicará más adelante.

Tenemos un solo actor que es el usuario, se considera que dispone de dos dispositivos, en los cuales se realizan las mismas acciones ya que para realizar el emparejamiento se requiere de la autentificación de los dos dispositivos y la interacción entre ellos requiere de el dispositivo móvil desde el que se realizan acciones y el ordenador desde el que se visualizan los resultados de estas.

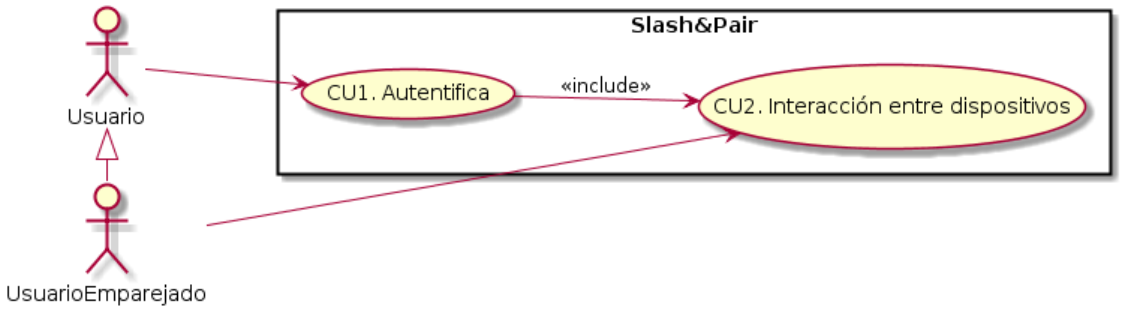

Figura 3: Casos de uso Generales

CU1: Autentificación		
Precondición	El usuario dispone de dos dispositivos, un ordenador y un smartphone. En ellos ha introducido en el navegador el enlace a la aplicación en ambos dispositivos El usuario es no dispone de otra instancia abierta del sistema en ninguno de los dispositivos.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario se conecte a la aplicación desde sus dispositivos.	
Secuencia normal	Paso	Acción
	1	El usuario inicia una instancia de la aplicación en cada uno de los dispositivos.
	2	Se muestra un código QR en el navegador del ordenador.
	3	El usuario escanea el código QR desde el dispositivo móvil, el usuario dispone de un botón con el que iniciar la cámara.
	4	El usuario usuario visualiza cómo se realiza una redirección automática en el navegador de sus dos dispositivos.
Postcondición	El usuario tiene ambos dispositivos emparejados en la aplicación.	
Excepciones	Paso	Acción
	3	El dispositivo móvil no reconoce el código QR del ordenador
		E1: El usuario dispone de un mensaje en el dispositivo móvil desde el inicio de la aplicación informando que en caso de que el dispositivo móvil no reconozca ningún código QR refresque la página del navegador en el ordenador.
	3	El código QR escaneado no es válido
		E2: Se alerta al usuario con un mensaje de error informando de que el código no es válido, se le pide que refresque la página del navegador.
Flujo alternativo	Paso	Acción
	3	El usuario no dispone de acceso a la cámara de su dispositivo móvil.
	4	El usuario ha de hacer clic en un botón en su ordenador, en el que está especificado que su función es cambiar el qr por un código
	5	El usuario introduce en su dispositivo móvil el código mostrado en el ordenador y presiona el botón de sincronizar para enviar el código.
		Se vuelve al paso 4 del flujo normal

Tabla 3: CU1: Autentificación

CU2: Interacción entre dispositivos		
Precondición	El usuario tiene ambos dispositivos emparejados en la aplicación.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario realice una interacción con el dispositivo móvil	
Secuencia normal	Paso	Acción
	1	Se redireccionan las páginas en los navegadores de ambos dispositivos
	2	El usuario interactúa con el smartphone realizando una acción que active los sensores del dispositivo
	3	El sistema recoge los datos del dispositivo móvil.
	4	El sistema envía los datos recogidos al ordenador
	5	El usuario ve reflejada su acción con el smartphone en el ordenador mediante una serie de cambios en el navegador de este.
Postcondición	El usuario interactúa con los dispositivos y ve reflejadas sus acciones del smartphone en el ordenador.	
Excepciones	Paso	Acción
	1	El usuario no visualiza correctamente la página
		E1: El usuario refresca la página, en caso de seguir con el mismo problema, uno de sus dispositivos no son compatibles con la aplicación.
	2	El usuario no ve reflejada correctamente su interacción en el ordenador, o no ve interacción alguna.
		E1: El dispositivo móvil puede no disponer de los sensores necesarios para el correcto funcionamiento de la aplicación. Se da un aviso de que su dispositivo no es compatible.

Tabla 4: CU2: Interacción entre dispositivos

4.2 Diagrama de casos de uso específico

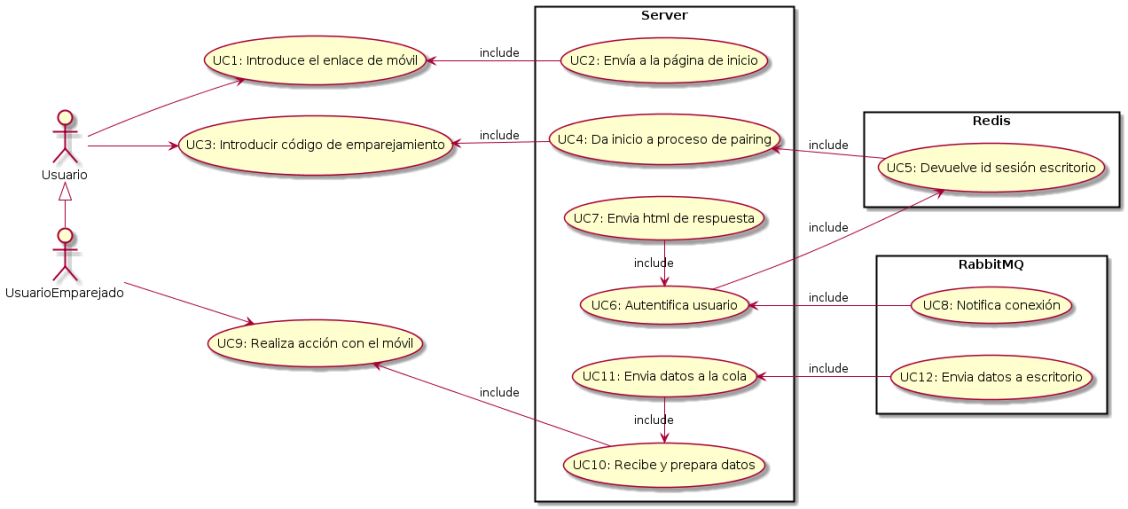

Figura 4: Casos de uso específicos

CU1 Introduce el enlace de móvil		
Precondición	El usuario dispone de internet y dispositivo móvil	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario introduzca la url de la aplicación en su dispositivo móvil.	
Secuencia normal	Paso	Acción
	1	El usuario dispone de un dispositivo móvil desde el cual abre la aplicación de navegador.
	2	El usuario introduce la url de la aplicación web
Postcondición	El usuario tendrá abierta la aplicación en su navegador con la página de web cargada.	

Tabla 5: CU1 Introduce el enlace de móvil

CU2: Envía a la página de inicio		
Precondición	El usuario ha introducido la url en su navegador en el dispositivo móvil.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el servidor reciba una petición en la url /mobile	
Secuencia normal	Paso	Acción
	1	El servidor recibe una petición en la url del móvil
	2	El servidor envía la página web al usuario
Postcondición	Se ha creado la página html del index la cual se enviará al navegador del dispositivo móvil donde se iniciará la cámara.	

Tabla 6: CU2: Envía a la página de inicio

CU3: Introducir código de emparejamiento		
Precondición	El usuario tiene iniciada la sesión en el escritorio donde se mostrará el código QR. El usuario tiene cargado el html que se le ha enviado desde servidor.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario realice el escaneo del código QR mostrado por el ordenador.	
Secuencia normal	Paso	Acción
	1	El usuario inicia la cámara de su dispositivo
	2	El usuario enfoca el código QR
	3	Se detecta automáticamente el código QR
Postcondición	El navegador del usuario tendrá la información del código de emparejamiento.	
Excepciones	Paso	Acción
	1	Si el usuario no enfoca bien el código qr o el dispositivo no lo detecta.
		E.1 El sistema sigue con la cámara encendida buscando un código QR
		E.2 Si no se reconoce en ningún momento el qr, el usuario tendrá instrucciones de refrescar la página en el ordenador.
	1	El usuario no tiene acceso a la cámara de su móvil
		E.1 El usuario dispone de un botón, propiamente indicado, en el ordenador para el caso de que no pueda encender su cámara, clicando en él se le muestra el código de emparejamiento.
		E.1 El usuario introduce dicho código en su dispositivo móvil.
Comentarios	Al acceder a este punto, el usuario dependiendo dependiendo del sistema operativo de su dispositivo móvil deberá dar permisos para abrir la cámara.	

Tabla 7: CU3: Introducir código de emparejamiento

CU4: Da inicio proceso de emparejamiento		
Precondición	El servidor ha recibido el código escaneado por el dispositivo móvil.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando reciba el código de sincronización desde el dispositivo móvil.	
Secuencia normal	Paso	Acción
	1	El servidor recibe el dato con el código de sincronización
	2	El servidor realiza consulta a base de datos con el código recibido.
Postcondición	El servidor ha consultado a base de datos si existen el código enviado asociado a una sesión de escritorio	
Comentarios	La consulta a base de datos es una consulta en Redis, donde se guarda el código y id de sesión desde el servidor de escritorio.	

Tabla 8: CU4: Da inicio proceso de emparejamiento

CU5: Devuelve id sesión a escritorio		
Precondición	Se ha consultado el código a base de datos.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando se realice la consulta del código en base de datos	
Secuencia normal	Paso	Acción
	1	La base de datos recibe una consulta con el código de sincronización
	2	Se comprueba si el dato existe en la base de datos
	3	Se devuelve la id de sesión de escritorio para realizar el emparejamiento de dispositivos
Postcondición	El servidor tiene la información necesaria para realizar el emparejamiento	
Excepciones	Paso	Acción
	2	No se encuentra en base de datos el código enviado
		E1. Se vuelve al caso de uso UC2 mostrando mensaje de error en el nuevo html
Comentarios	La base de datos utilizada en este punto es Redis en la que se consulta el código de sincronización.	

Tabla 9: CU5: Devuelve id sesión a escritorio

CU6: Autentificar usuario		
Precondición	El servidor ha comprobado el código de emparejamiento en la base de datos y tiene la id de sesión del ordenador del usuario.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el servidor autentifique el dispositivo móvil del usuario.	
Secuencia normal	Paso	Acción
	1	El servidor ha enviado una petición a la base de datos con el código de emparejamiento.
	2	El servidor recibe respuesta de la base de datos con la id del usuario.
	3	El servidor autentifica el dispositivo móvil con la misma id de sesión que en el escritorio.
Postcondición	El usuario tiene autentificado el dispositivo móvil emparejado con el ordenador, se ha enviado un aviso a la cola que notificará al servidor de escritorio.	

Tabla 10: CU6: Autentificar usuario

CU7: Envía html de respuesta		
Precondición	El usuario ha realizado el emparejamiento de dispositivos.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el servidor devuelve el html de respuesta cuando el usuario tenga los dispositivos emparejados.	
Secuencia normal	Paso	Acción
	1	El servidor tiene realizada la autentificación
	2	El servidor busca en templates el html de respuesta
	3	Se construye en servidor el html de respuesta en caso de ser necesario.
Postcondición	El servidor realiza una redirección de url al dispositivo móvil del usuario.	
Excepciones	Paso	Acción
	1	No se ha realizado correctamente la autentificación.
		E.1: Se realizan las mismas acciones pero con un html de error en vez de uno que inicie la lectura de los datos de los sensores.
Comentarios	En este html irán las funciones que recogen los datos de los sensores.	

Tabla 11: CU7: Envía html de respuesta

CU8: Notifica conexión		
Precondición	El usuario ha realizado el emparejamiento con el dispositivo móvil, el servidor ha encontrado el código de emparejamiento junto a una sesión de escritorio.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando se haya realizado el emparejamiento y se tenga que notificar al escritorio que todo el proceso se ha realizado con éxito.	
Secuencia normal	Paso	Acción
	1	El servidor notifica mediante una cola externa a este que el dispositivo móvil con la id encontrada desde el proceso anterior ha realizado la conexión.
	2	La cola notifica al servidor del ordenador que se ha emparejado un dispositivo móvil.
Postcondición	El servidor del ordenador queda notificado de que se ha realizado el emparejamiento con éxito de un dispositivo móvil.	
Comentarios	La cola que se encarga de notificar al servidor del ordenador que el dispositivo móvil ha realizado la conexión es RabbitMQ.	

Tabla 12: CU8: Notifica conexión

CU9: Realiza acción con el móvil		
Precondición	El usuario ya ha realizado el emparejamiento de dispositivos con éxito, tiene cargado el html que se encarga de la lectura de los datos de los sensores.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario realice una acción en la que se activen los sensores.	
Secuencia normal	Paso	Acción
	1	El usuario realiza una acción que tiene interacción con los sensores del dispositivo móvil.
	2	Se envían estos datos al servidor
Postcondición	El servidor ha recibido los datos de los sensores accionados.	
Comentarios	El envío de datos al servidor se realiza desde el navegador del dispositivo móvil.	

Tabla 13: CU9: Realiza acción con el móvil

CU10: Recibe y prepara datos		
Precondición	El servidor ha recibido datos de los sensores del dispositivo móvil que tiene una sesión emparejada previamente.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el servidor reciba datos de los sensores del dispositivo móvil.	
Secuencia normal	Paso	Acción
	1	El usuario ha realizado una acción desde el dispositivo móvil
	2	Se han enviado los datos a servidor
	3	El servidor procesa los datos
Postcondición	Los usuarios han sido procesados por el servidor, para darles formato y prepararlos para ser enviados a la cola	
Comentarios	Los datos dependiendo del uso que se les vaya a dar se les puede dar un procesamiento u otro.	

Tabla 14: CU10: Recibe y prepara datos

CU11: Envía datos a la cola		
Precondición	El servidor ha recibido los datos de los sensores móviles, este los ha tratado y están disponibles para ser enviados.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el servidor envía datos de sensores del dispositivo móvil a la cola.	
Secuencia normal	Paso	Acción
	1	Los datos están en servidor y han sido tratados por este
	2	Los datos son correctos y se envían a la cola
Postcondición	Los datos han sido enviados a la cola que se encargará de hacerlos llegar al servidor de escritorio.	
Comentarios	La cola utilizada en este paso es RabbitMQ	

Tabla 15: CU11: Envía datos a la cola

CU12: Envía datos a escritorio		
Precondición	Los datos de los sensores han sido enviados desde el servidor a la cola	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando la cola ha de enviar los datos al servidor del escritorio.	
Secuencia normal	Paso	Acción
	1	La cola recibe datos de los sensores
	2	La cola envía los datos al servidor de escritorio
Postcondición	Los datos han sido enviados al servidor de escritorio.	
Comentarios	La cola es la salida de datos del servidor del dispositivo móvil, a su vez es la entrada en el servidor por el lado escritorio.	

Tabla 16: CU12: Envía datos a escritorio

5 Diseño

5.1 Conocimientos previos

En el apartado anterior se detalla un análisis de las funciones de *Slash and Pair* mediante un caso de uso general para explicar a groso modo las dos funciones principales y un caso de uso específico para entrar en detalle del funcionamiento interno del sistema.

En este apartado se recoge toda la información que se ha obtenido previamente a la implementación del software.

5.1.1 Calidad de software

En fases previas al desarrollo del sistema se realizó un análisis de cómo debe ser un proyecto de software para tener una calidad y en que hacer énfasis a la hora de desarrollar para conseguir el mejor resultado.

5.1.2 Modelo McCall

El primer estándar presentado para evaluar la calidad del software fue de Jim McCall[14]. El modelo, desarrollado inicialmente para *Air Forcé y Dod*. En él se proponen una serie de factores de calidad, estos son organizados en tres ejes o puntos de vista: Operación, revisión y transición.

Dentro de cada uno de estos ejes se determinan los factores de calidad(5)

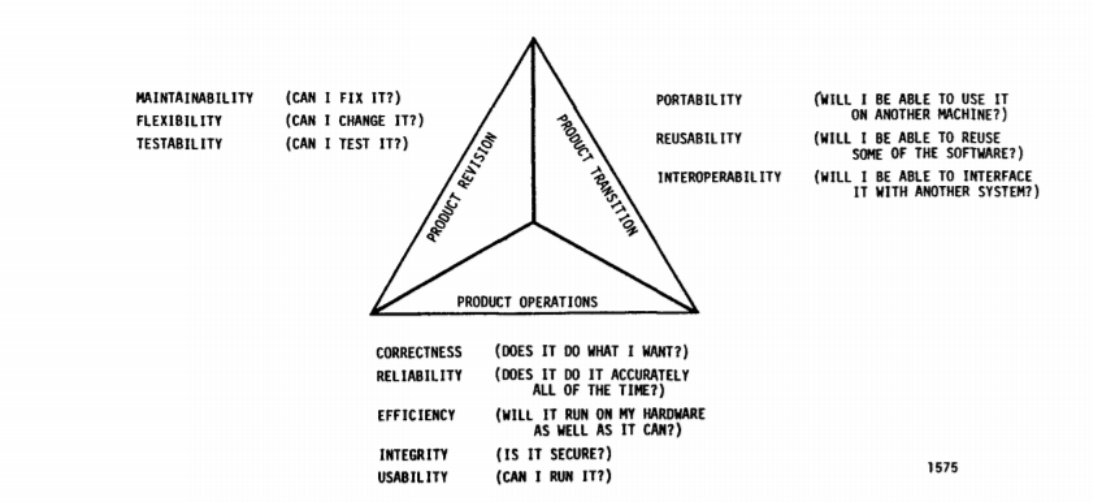

Figura 5: Factores de calidad de software según el modelo McCall

Operación: Facilidad de uso, integridad, eficiencia, corrección o exactitud, fiabilidad. Revisión: Facilidad de prueba, facilidad de mantenimiento, flexibilidad. Transición: Reusabilidad, portabilidad, interoperabilidad.

Facilidad de uso	Esfuerzo que se requiere por parte del usuario en aprender cómo funciona el software y en entender las entradas y salidas de este.
Integridad	Grado en el que el acceso al software o datos esté controlado para que no sea accesible a personas no autorizadas.
Eficiencia	La cantidad de recursos informáticos y el código requerido por un programa para realizar una función.
Corrección	Grado en el que un software cumple con sus especificaciones y cumple con los objetivos del usuario.
Fiabilidad	Medida en que se puede esperar que un programa lleve a cabo su función con la precisión requerida.
Facilidad de prueba	Esfuerzo requerido para probar un software para asegurar que cumple su función.
Facilidad de mantenimiento	Esfuerzo requerido para localizar y arreglar un error en el software
Flexibilidad	Esfuerzo requerido para modificar el software añadiendo funcionalidad o cambiandola.
Reusabilidad	Grado en que un software puede ser utilizado en otras aplicaciones, que sea modular, independencia entre sistema y software, atributos del software que determinan su dependencia del entorno operativo.
Portabilidad	Esfuerzo necesario para transferir un programa de una configuración de hardware y/o un entorno de sistema a otro.
Portabilidad	Esfuerzo necesario para transferir un programa de una configuración de hardware y/o un entorno de sistema a otro.
Interoperabilidad	Esfuerzo requerido para acoplar un sistema con otro.

Tabla 17: Factores de calidad según el modelo de McCall.

5.1.3 Modelo ISO 9126

El modelo ISO 9126[15] es una variante del modelo McCall, fue propuesto como estándar internacional para la medición de calidad en el software, el nombre completo es ISO 2196 es ?*Software Product Evaluation: Quality Characteristics and Guidline for Their Use*?.

Este modelo clasifica la calidad del software en un conjunto estructurado de características y subcaracterísticas.

Funcionalidad	Idoneidad Exactitud Interoperabilidad Seguridad Cumplimiento de normas
Fiabilidad	Madurez Recuperabilidad Tolerancia a fallos
Usabilidad	Aprendizaje Comprensión Operatividad Atractividad
Eficiencia	Comportamiento en el tiempo Comportamiento de recursos
Mantenibilidad	Estabilidad Facilidad de análisis Facilidad de cambio Facilidad de pruebas
Portabilidad	Capacidad de instalación Capacidad de reemplazamiento Adaptabilidad Co-existencia

Tabla 18: Factores de calidad según el ISO 9126.

El modelo mencionado distingue entre fallo y no conformidad. Un fallo es el incumplimiento de los requisitos previos, mientras que la no conformidad es el incumplmiento de los requisitos especificados.

5.1.4 Modelo de ISO 25000

El modelo de iso 25000[16] es el actual estándar en calidad de software, reemplaza a ISO 9126 e *ISO 145898(Software Product Evaluation).*

Sus mediciones pueden ser útiles no solamente para evaluar el producto software sino también para definir requerimientos de calidad. Su objetivo principal es guiar el desarrollo de los productos de software con la especificación y evaluación de requisitos de calidad.

El ISO 25000 dispone de diferentes ISO que en conjunto definen el estándar por el que se evaluarán los productos de software, los más importantes son ISO 25010, ISO 25012 e ISO 25040.

En ISO 25000 tenemos la definición del modelo de calidad del producto, se encuentra compuesto por las ocho características de calidad que se muestran en la siguiente figura.

Adecuación funcional	Completitud funcional Corrección funcional Pertinencia funcional
Eficiencia de desempeño	Coportamiento Temporal Utilización de recursos Capacidad
Compatibilidad	Coexistencia Interoperabilidad
Usabilidad	nteligibilidad Aprendizaje Protección frente a errores de usuario Estética Accesibilidad
Fiabilidad	Madurez Disponibilidad Tolerancia a fallos Capacidad de recuperación
Seguridad	Confidencialidad Integridad Norepudio Autenticidad Responsabilidad
Mantenibilidad	Modularidad Reusabilidad Analizabilidad Capacidad de ser modificado Capacidad de ser probado
Portabilidad	Adaptabilidad Facilidad de instalación Capacidad de ser reemplazado

Tabla 19: Factores de calidad según el ISO 9126.

El ISO 25012 es el modelo de calidad de datos, representa los cimientos sobre los cuales se construye un sistema de evaluación de un producto de datos. Se encuentra compuesto por las 15 características que se muestran en la siguiente figura(6).

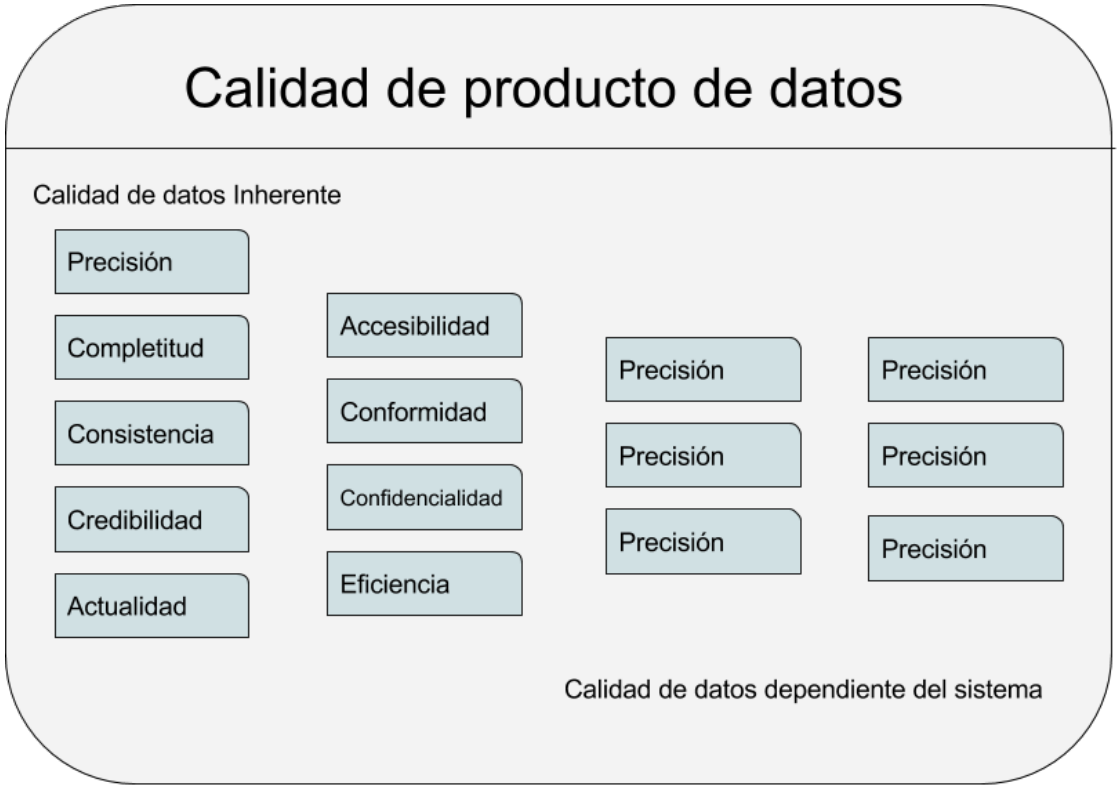

Figura 6: Características de calidad de datos según ISO25000.

ISO 25040 define el proceso para llevar a cabo la evaluación del producto software. DIcho proceso consta de un total de cinco actividades como podemos comprobar en la siguiente tabla(20).

1	Establecer los requisitos de la evaluación
2	Especificar la evaluación
3	Diseñar la evaluación
4	Ejecutar la evaluación
5	Concluir la evaluación

Tabla 20: Actividades que definen el proceso de evaluación según ISO 25040

5.1.5 Conclusiones de estándares de calidad

Según se ha visto en los estándares de calidad comentados previamente podemos sacar unas conclusiones para la toma de decisiones del proyecto.

Slash and Pair se ha diseña con la intención de cumplir con los estándares de calidad, para ello se ha desarrollado poniendo atención a los detalles mencionados en los diferentes modelos de evaluación calidad.

El sistema se desarrolla cumpliendo con los requisitos funcionales descritos anteriormente, tiene dos funciones claras, el emparejamiento de dispositivos de un

mismo usuario y la interacción entre ellos.

Debe ser un software eficiente ya que ha de ser capaz de procesar una gran cantidad de datos proveniente de los sensores del *smartphone*, apartado muy importante en el apartado de la interacción entre dispositivos.

La compatibilidad es un punto muy a tener en cuenta en el desarrollo ya que en un futuro *Slash and Pair* podría convertirse en un *framework* que poder integrar en otros sistemas. Para el desarrollo del sistema presentado existe una coexistencia con otro software desarrollado por terceros con el cual se realiza la demo, por lo tanto es un punto muy importante debido a la naturaleza del sistema.

En temas de usabilidad, según los requisitos propuestos, ha de ser accesible para cualquier usuario, por lo tanto ha de ser un sistema con capacidad para un rápido aprendizaje. Por este motivo se diseña pensando en que la interacción con el usuario sea lo más intuitiva posible, por este motivo se ha utilizado el sistema de aplicación web y no una nativa ya que para un usuario es menos molesto introducirse en una web y utilizar el sistema que no instalarse una aplicación en diferentes dispositivos, con lo que ello conlleva en los usuarios con menos conocimientos que han de instalar en diferentes sitios una aplicación con la dificultad que esto puede suponer.

El sistema ha de mostrar una fiabilidad, ha de ser tolerante a errores ya que se mueve con datos de sensores de dispositivos que realmente no se tiene la certeza de que todos sean igual. Al ser una aplicación web los errores con conexión han de afectar lo menos posible.

La seguridad es otro punto muy importante ya que tanto para el emparejamiento de dispositivos como para la interacción entre estos, el sistema ha de asegurar que un usuario no pueda emparejar su dispositivo con los de otros usuarios y que los datos que se reciben en el dispositivo móvil, en el camino hacia el ordenador no se pierdan o puedan ser modificables.

El sistema ha de tener una buena mantenibilidad ya que puede estár en crecimiento constante, puede tener que adaptarse para una escalabilidad mayor, o para nuevos sensores que sean necesarios incorporar.

La portabilidad del software se ha tenido en cuenta en la toma de decisiones, por eso es una aplicación web, para poder ejecutarse en los diferentes sistemas operativos o *hardware*, en este ámbito se ha de seguir trabajando para compatibilidad mayor entre diferentes navegadores y sistemas móviles que pueden no tener disponibles todos los sensores.

En cuestión de la calidad de datos se han tenido en cuenta los diferentes puntos expuestos anteriormente para que el sistema cumpla con los requisitos planteados.

Los datos son muy importantes ya que el funcionamiento de la aplicación se basa en transportar datos entre dispositivos por este motivo se ha de poner especial atención en cómo son tratados y que sean fiables.

Estos son los puntos que se han tenido más en cuenta para el diseño de *Slash and Pair*.

5.1.6 Principios de diseño

En este apartado se analizan los principios de diseño y patrones que se han estudiado previamente al desarrollo del código. Este punto es importante para la implementación de un software de calidad.

5.1.7 Inversión de control IoC

Inversión de control IoC[17] es un principio de diseño que afecta directamente al desarrollo del software *Slash and Pair*.

Lo que propone IoC es un diseño de software en el que el flujo de ejecución no es lineal como en los métodos de programación tradicionales, en los que se disponía de un flujo en el que se tenía conocimiento de cuándo y quién podía llamar a una función, de esta manera no había problemas de que una función se ejecutará en tiempos que no le corresponden.

En el software actual no siempre se tiene una ejecución lineal del código ya que las funciones pueden ser llamadas desde agentes externos al software como pueden ser librerías o una arquitectura externa. Esto quita el control de ejecución al sistema ya que una función puede ser llamada en un momento no esperado.

En *Slash and Pair* el flujo de ejecución depende de cuando el usuario realiza una serie de acciones, por lo tanto el sistema no tiene control en la ejecución lineal.

5.1.8 Patrón Model View Controller

Model View Controller[18], de ahora en adelante MVC, es un patrón de arquitectura del software, en el cual se separan los datos y la lógica de negocio de una aplicación de la interfaz de usuario. El patrón fue introducido por Trygve Reenskaug en los años 70.

Propone tres componentes distintos, modelo vista y controlador.

- Modelo: Representación de la información con la cual el sistema opera, se gestionan tanto consultas como actualizaciones de los datos implementando también los privilegios de acceso descritos en las especificaciones del software. Las peticiones de acceso o manipulación de los datos llegan a través del controlador.
- Controlador: Responde a eventos e invoca peticiones al modelo, envía comandos a la vista si se solicitan cambios en la forma que se representa el modelo. Se encarga de la lógica de negocio.
- Vista: Presenta los datos del modelo en un formato adecuado para interactuar con el usuario.

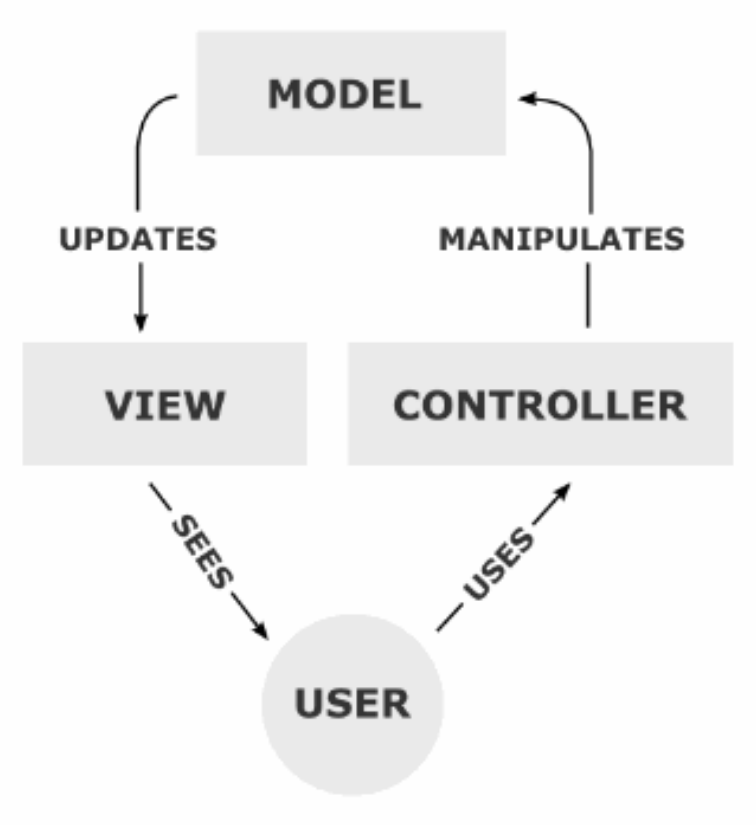

Figura 7: Colaboración entre los componentes de un MVC.

5.1.9 Patrón observer

El patrón observador (*observer*)[19] define una dependencia del tipo uno a muchos entre objetos, de forma que si uno de los objetos cambia de estado, se notifica este cambio a todos los objetos que tengan una dependencia de este.

Los objetos en este patrón pueden suscribirse o anular la suscripción de los demás objetos.

El objetivo consiste en desacoplar la clase de los objetos clientes del objeto, aumentando la modularidad del lenguaje, creando las mínimas dependencias y evitando bucles de actualización.

La estructura de este patrón es la mostrada en la figura

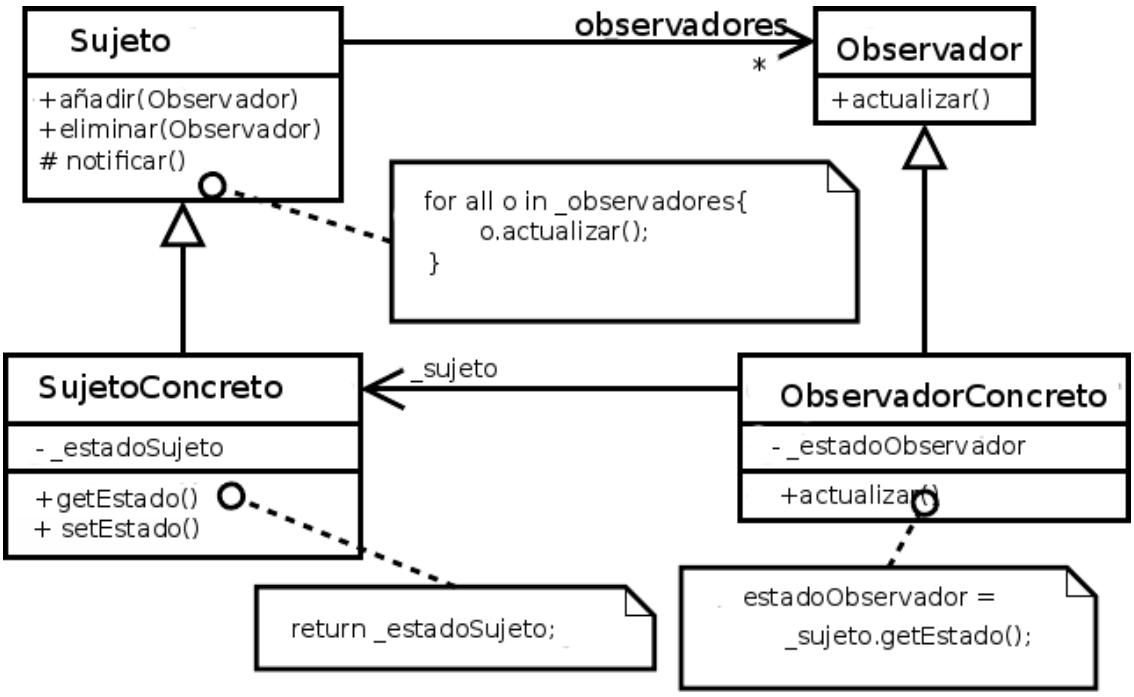

Figura 8: Estructura del patrón observador.

El sujeto proporciona una interfaz para agregar y eliminar observadores. El objeto conoce a todos sus observadores.

El observador define el método que usa el sujeto para notificar cambios en su estado.

El sujeto concreto mantiene el estado de interés para los observadores concretos y los notifica cuando cambia su estado. No tienen porque ser elementos de la misma jerarquía.

El observador concreto mantiene una referencia al sujeto concreto e implementa la interfaz de actualización, guarda la referencia del objeto que observan, así en caso de ser notificados de algún cambio, pueden preguntar sobre este cambio.

El patrón observador proporciona algunas propiedades de los sistemas que se comunican mediante mensajes.

- No hay que estar monitorizando un objeto en búsqueda de cambios, se realiza la notificación cuando esto ocurre.
- Permite agregar nuevos observadores para proporcionar otro tipo de funcionalidad sin cambiar el objeto observador.
- Bajo acoplamiento entre observable y observador, un cambio en uno no afecta al otro.

5.2 Desarrollo

En el apartado anterior se detallan los conocimientos adquiridos previamente a la implementación de *Slash and Pair* para asegurar un código de calidad.

En este capítulo se reseña el diseño de la aplicación a nivel de software. Se da una descripción del sistema, se describe la arquitectura de la aplicación, se detalla la elección de tecnologías así como la explicación del porqué de la decisión de utilizarlas y se explica brevemente la implementación de todas ellas en conjunto para conseguir realizar el proyecto.

5.2.1 Descripción del sistema

Slash and Pair es una aplicación diseñada para ejecutarse en servidores web. El proyecto consta de tres proyectos más pequeños, cada uno de ellos es un servidor.

Se puede hacer una gran división en el desarrollo. Por un lado tenemos la parte que concierne a los servicios que utiliza el ordenador y por otra la que concierne a los servicios para los dispositivos móviles. En medio de estas dos se ha desarrollado un servidor en el que están las declaraciones de los objetos comunes en ambos.

A parte de la división en el desarrollo en cuanto a dispositivos, el sistema tiene dos grandes funciones que engloban otras más específicas. Una de ellas es la sincronización entre los dispositivos, para ello se requiere de un proceso de emparejamiento que sucede tanto en la parte que se encarga del ordenador como la del *smartphone*. La otra es la interacción entre los dispositivos, el usuario realiza acciones con el *smartphone* y ve el resultado en el ordenador, esto requiere de una comunicación entre los dispositivos y servidores, y entre servidores.

Por la parte del proyecto de ordenador tenemos un servidor el cual se encarga de dar inicio a este proceso de emparejamiento de los dispositivos, para ello lo primero es crear una sesión para el ordenador que se conecta, junto con esta sesión se genera un código de emparejamiento que servirá para introducirlo en el dispositivo móvil. Estos dos elementos son guardados en una base de datos desde la cual se tiene acceso desde la parte móvil y la de escritorio.

Una vez generado el código y la sesión el servidor de escritorio genera un código HTML para mostrar en el navegador del usuario, en este se tienen unas pequeñas instrucciones para realizar el emparejamiento junto a un código QR, el cual se será escaneado desde el *smartphone*.

Por la parte del dispositivo móvil se tiene otro servidor desde el cual se sigue con el proceso de emparejamiento desde el punto que se ha quedado por parte del escritorio. Se genera un html plano desde el cual se iniciará la cámara del *smartphone* a través de *JavaScript*. En caso de que no sea posible iniciar la cámara se ofrece la posibilidad de seguir otro método para que aun así el usuario pueda llegara realizar el emparejamiento. Este proceso alternativo requiere que el usuario marque esta opción en su ordenador en el cual encontrará un botón que se encargará de reemplazar el código qr por un código que el usuario pueda introducir en el móvil en un campo de texto que habrá habilitado para tal funcionamiento.

Una vez se ha introducido el código de emparejamiento este se envía al servidor del móvil, desde aquí se va a buscar este código a una base de datos, en caso de encontrarlo viene con una id de sesión que es la que se le ha asignado al ordenador.

Una vez encontrada la id de sesión se genera la sesión por la parte móvil y se envía una notificación mediante una cola al servidor del escritorio, el mensaje en la cola se manda con la id del usuario para que solo le llegue al dispositivo del usuario.

El servidor móvil monta un HTML en el que se inicia el canal de *WebSocket* por el cual se realizará la comunicación de los datos de los sensores entre el móvil y su servidor.

En este punto empieza el otro proceso importante del que consta el proyecto, la interacción entre los dos dispositivos.

Como ya se ha comentado, en el dispositivo móvil tenemos cargado un HTML el cual iniciará la comunicación mediante *WebSocket*, con lo cual se puede asegurar que se tiene una forma de comunicación para cada usuario conectado sin que los datos puedan mezclarse con los de otros usuarios, este proceso será explicado con más detalles en este capítulo.

Este HTML también se encarga de la recogida de datos de los sensores haciendo uso de una API, estos datos se envían a través del mencionado *WebSocket*. Con lo cual los datos llegan al servidor de móvil donde se tratan y se envían al servidor de escritorio a través de otra cola.

En este punto se hace uso del proyecto intermedio, en el que está la estructura de los objetos, con los datos recogidos por los sensores se guardan en un objeto definido desde el cual se disponen de métodos que interpretan desde el objeto.

Desde el servidor de móvil se hace uso de estos objetos comunes para dar forma al objeto y construirlo en un formato válido para su serialización y así poder guardarlo en la cola con el id de usuario, lo cual permite que solo el usuario al que tiene que ir dirigido haga uso de este dato.

En el servidor del ordenador se recoge el dato de la cola mediante un *listener*, se interpreta y se envía mediante *WebSocket* al navegador del usuario. Una vez aquí se recoge el dato en el navegador y se le da un uso realizando una interacción con el HTML.

5.3 Arquitectura

5.3.1 Introducción

Slash and Pair es una aplicación de comunicación entre dispositivos. El objetivo es poder dar uso de los diferentes sensores de un dispositivo móvil con la comodidad de una pantalla grande como puede ser la de un ordenador.

Para conseguir este objetivo se han de tomar una serie de decisiones que marcan el funcionamiento del sistema.

Lo primera decisión a tomar es el medio por el que se va a realizar la comunicación entre los dispositivos. En este punto hay diferentes opciones, por un lado se puede realizar mediante un cable para conectar los dispositivos y se comuniquen, por otro lado una conexión inalámbrica.

Para dar usabilidad y comodidad al usuario se ha optado por la segunda opción, dentro de esta decisión hay más opciones por las que decantarse. Se puede realizar la comunicación mediante *bluetooth*, mediante una conexión *wifi* en la que los datos se envían a través de un *router*, otra forma de conseguir el objetivo de la comunicación es mediante un servidor web en el que se conectarán los dispositivos a través de internet y este envíe los datos de uno al otro.

Otra cuestión que se ha tenido en cuenta es la plataforma para la que hacer la aplicación, las opciones son:, una aplicación nativa para un sistema operativo de escritorio y otra para un sistema de un *smartphone*. Otra alternativa a esto es una aplicación web que pueda correr en todos los dispositivos a costa de perder algunas características que ofrece una aplicación nativa. Una vez planteadas las opciones más generales por las que se ha de tomar una decisión surgen nuevas cuestiones, de carácter más específico, a las que dar respuesta.

Por ejemplo, de qué recursos se disponen para el desarrollo del software? Se hace un desarrollo de zero o se utilizan *frameworks*, qué *frameworks* pueden dar más ayuda para la implementación?

En el siguiente apartado se darán respuestas a todas las cuestiones planteadas contando con la toma de decisiones que se ha seguido en el proyecto ofreciendo motivos por los que decantarse por una y no otra de las opciones. Para ello se detalla la arquitectura de la aplicación centrando la explicación en las dos funciones más importantes.

5.3.2 Diseño general

Como se ha visto en el apartado de conocimientos previos, *Slash and Pair* utiliza el patrón MVC.

Por un lado se dispone de un controlador en cada proyecto, uno por la parte de móvil y otro por la del ordenador. El cual gestiona la interacción que puede tener el usuario con el sistema y obtiene los datos desde el modelo.

La vista del proyecto son las diferentes plantillas HTML, las cuales son las que

el usuario tiene acceso y con las que puede interactuar. Estas no tienen conexión directa con el modelo de datos en el sistema, desde la vista se recogen los datos de los sensores con los que el usuario interactúa y mediante el controlador pasan al modelo.

El modelo de la aplicación dispone de una base de datos en la que están guardados los datos para el emparejamiento de la aplicación y una cola desde la que el controlador va guardando y leyendo los datos de los sensores.

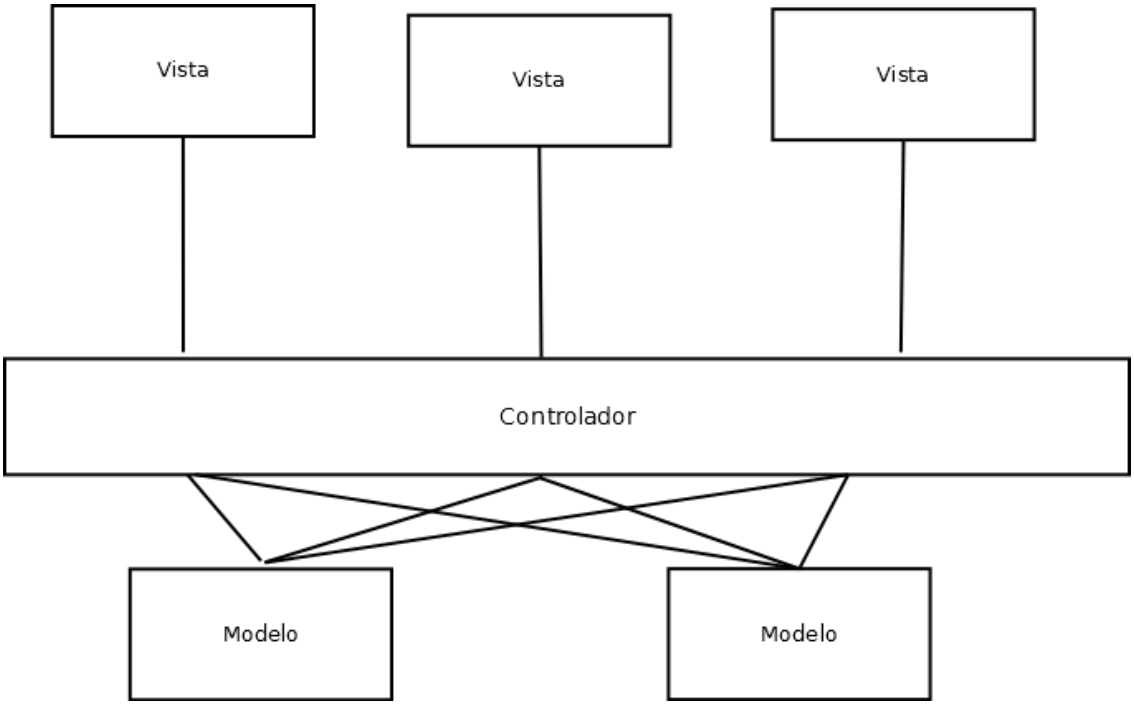

Figura 9: Modelo MVC

Según se puede observar en la figura (9), como se ha explicado, existe un único controlador en el proyecto el cual conecta las diferentes vistas con los diferentes modelos, en el caso de la aplicación las vistas son la página desde la cual se realiza el emparejamiento, otra desde donde se recogen los datos de los sensores y por último una en el ordenador a la cual le llegan dichos datos.

5.3.3 Diseño del emparejamiento

En este apartado se define el diseño de una de las dos funciones más importantes de *Slash and Pair*. Está función es la de emparejamiento de los dos dispositivos del usuario.

La primera decisión a tener en cuenta es cómo se realizará la comunicación entre los dos proyectos de móvil y de ordenador. Cada dispositivo está conectado a un servidor y entre estos servidores ha de haber una comunicación.

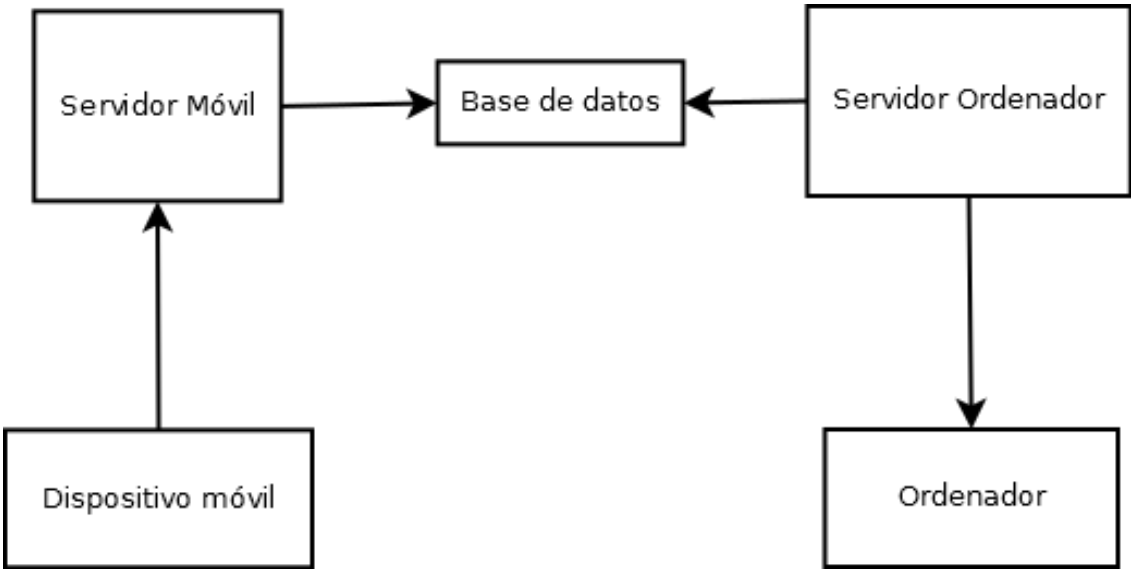

Figura 10: Diagrama de emparejamiento de dispositivos

Como podemos ver en la figura (11), el flujo de ejecución en el emparejamiento empieza en el servidor del ordenador, el cual genera el código necesario para la sincronización y genera una sesión para el dispositivo. Esto se envía al ordenador para que el usuario pueda introducir o leer el código desde su dispositivo móvil. A parte de enviar al ordenador la información, se guarda el código e id identificador de sesión en una base de datos.

Una vez el dispositivo móvil lee el código de sincronización, este envía el dato al servidor, desde el cual se busca en la base de datos si existe una entrada con el dato de sincronización recibido. En caso afirmativo se lee el id de usuario y se crea una sesión para el dispositivo móvil, a su vez se envía una notificación al ordenador para avisar al usuario de que se ha realizado el emparejamiento de dispositivos correctamente. Esta última parte se realiza mediante el mismo método que el envío de datos de los sensores que se explica en el siguiente apartado.

En medio de los dos servidores existe una base de datos. Como hemos comentado esto es debido a que es necesario guardar el dato de alguna manera ya que no sabemos en qué momento el usuario realizará el emparejamiento. Otro motivo de la utilización de una base de datos es la escalabilidad del proyecto, como se explica en siguientes apartados, el sistema está diseñado para ser escalable, lo cual requiere de un sistema aislado para la comunicación de los datos entre proyectos.

5.3.4 Diseño de interacción entre dispositivos

En este apartado se define el diseño de la otra gran función del sistema *Slash and Pair*, que es la interacción entre dispositivos.

El dispositivo móvil genera una serie de datos con los sensores, los cuales han de ser enviados al ordenador del usuario. Para ello han de pasar por el servidor de móvil, el cual guarda los datos en una cola donde el servidor del ordenador los lee

y se los envía al ordenador.

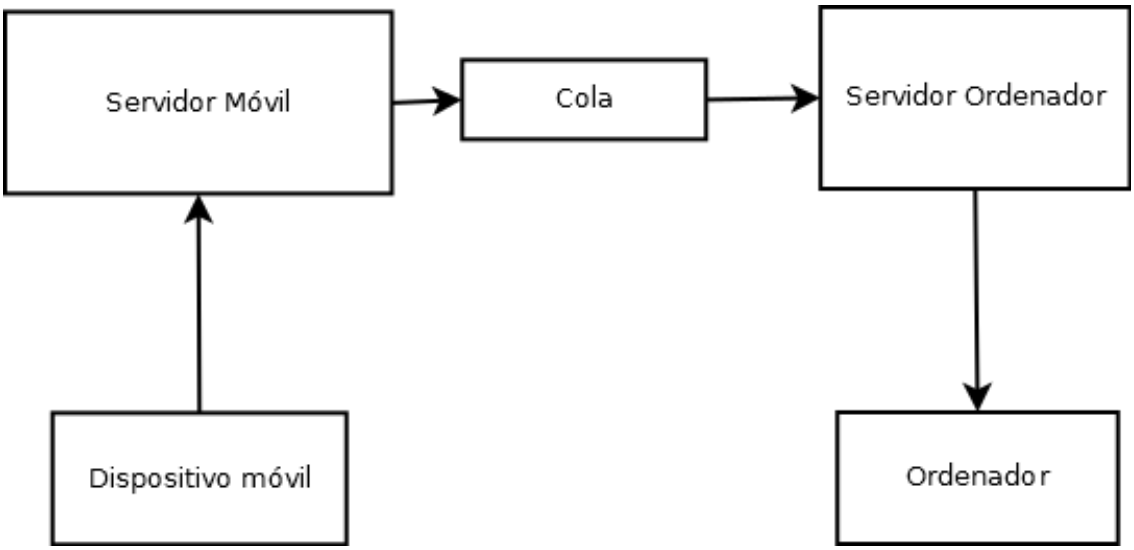

Figura 11: Diagrama de función de emparejamiento de dispositivos.

Como se puede ver en la figura (11), el flujo de datos va solo en una dirección. Como hemos comentado se generan los datos en el *smartphone* desde donde se envían al servidor de móvil, el servidor los recibe y los guarda en la cola.

Para que se pueda realizar todo el flujo de los datos de los sensores, estos han de ser serializados para que puedan ser enviados de un sitio a otro, esta serialización se realiza en el servidor de móvil donde los datos se preparan para ser enviados a la cola.

El servidor del ordenador hace uso del patrón *observer* comentado en el apartado de conocimientos previos, esto significa que tiene un objeto el cual observa mediante una referencia a la cola de datos, como se ha explicado cuando un objeto mantiene una referencia al sujeto concreto, en este caso la cola, le llega una notificación cuando el sujeto tiene una actualización. Esto pasa cuando a la cola le llega un dato para el usuario que está haciendo uso del sistema. El objeto observador tiene una referencia solo a los mensajes que lleguen para este usuario.

5.3.5 Estudio de la escalabilidad del proyecto

Slash and Pair se ha diseñado teniendo muy en cuenta la escalabilidad, por esto se ha separado en diferentes proyectos dividiéndolo para dispositivos móviles y para ordenadores.

Por ello se ha hecho un estudio de las posibles opciones que se tienen para llegado el momento en el que sea necesaria una expansión de los servidores desplegados.

El sistema tiene un reto, el cual es emparejar dos dispositivos de un mismo usuario. Esto si solo se tiene un servidor y un solo proyecto, se podría hacer mediante sesiones dentro del servidor. El problema llega cuando el número de usuarios crece,

y no se puede dar soporte a todos con el mismo servidor y se tiene que ampliar el número de estos.

Llegados este caso, no se puede controlar que el usuario conecte sus dos dispositivos a un mismo servidor.

Para solucionar este problema se ha decidido separar el proyecto en dos partes, crear un sistema para que gestione los dispositivos móviles y otro para los ordenadores, cada sistema está preparado para poder compartir un mismo servidor, o estar en servidores diferentes.

Estos dos sistemas han de tener una forma de comunicación, la cual se ha explicado en apartados anteriores, con la base de datos y la cola.

El sistema tal cual está ahora, que se encuentra en un solo servidor es como podemos ver en la figura (12).

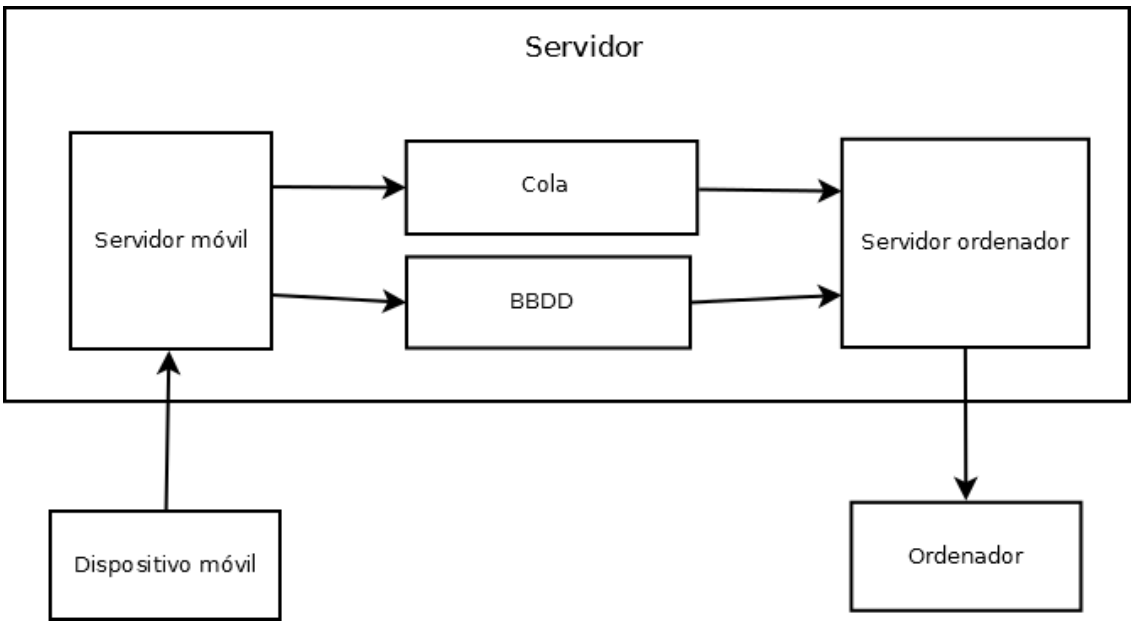

Figura 12: Diagrama de Slash and pair en un servidor.

En caso de crecimiento en número de servidores tenemos diferentes posibilidades. Se pueden tener diferentes servidores y cada uno de ellos con los dos sistemas para que pueda gestionar cualquier dispositivo que realice conexión con el (13)

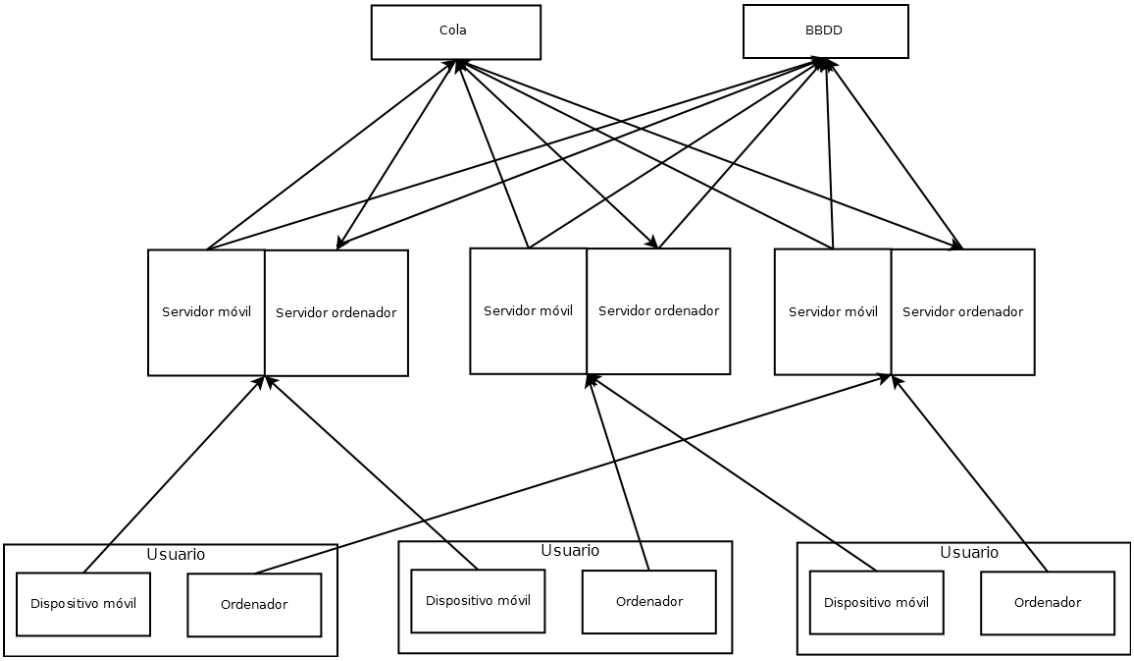

Figura 13: Diagrama de posible solución del sistema Slash and pair en varios servidores.

Otra posibilidad que se podría implementar para una posible expansión es que el usuario al realizar la conexión con sus dispositivos, estos se conecten a unos servidores de emparejamiento como se puede ver en la figura (14). La función de estos sería emparejar a los dispositivos de un mismo usuario y una vez realizado este proceso se redireccione donde apunta el dispositivo a otro servidor, que realice la interacción entre los dispositivos ya conociendo las id de sesión por las que se comunican estos dispositivos como se puede ver en la figura (15).

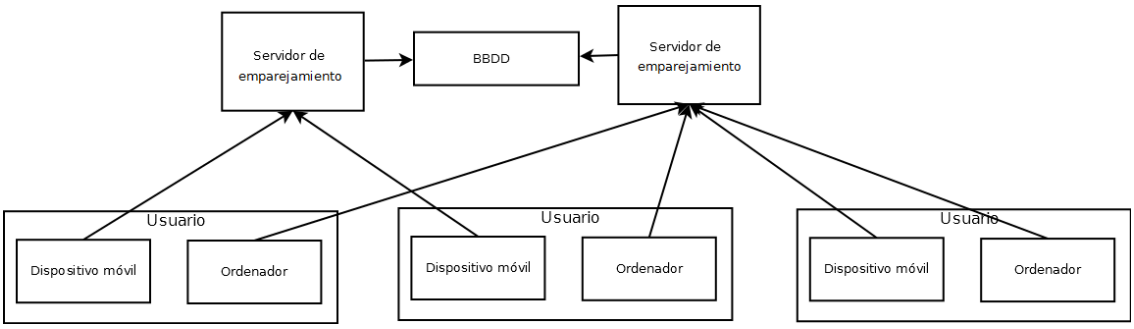

Figura 14: Diagrama de solución de expansión de Slash and pair con servidores de emparejamiento.

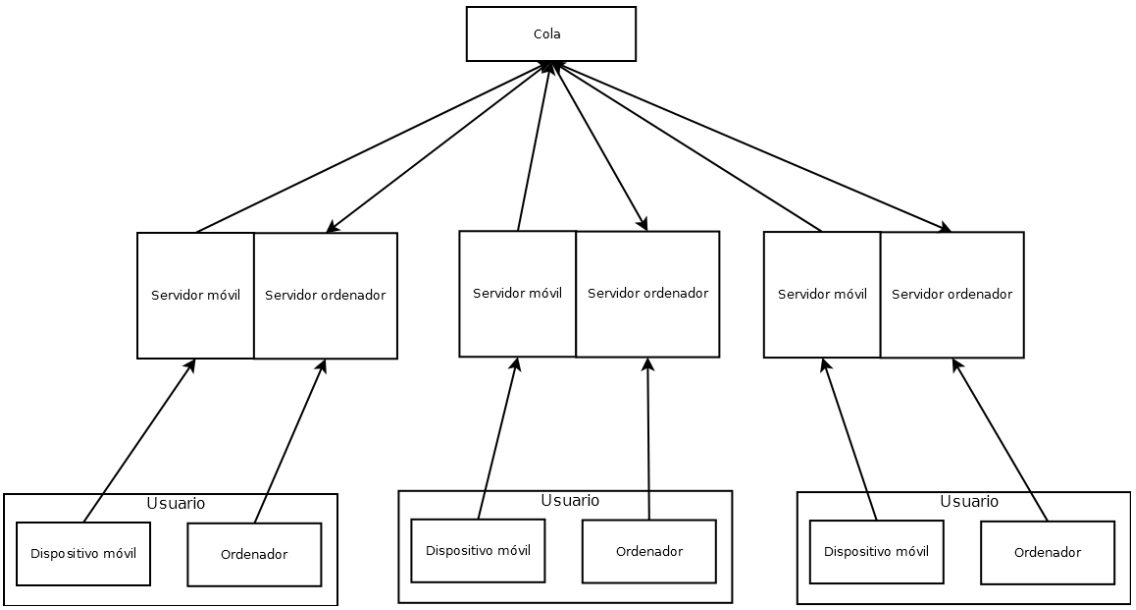

Figura 15: Diagrama de solución de interacción de Slash and pair en varios servidores.

5.4 Elección de tecnologías

5.4.1 Tecnologías generales

Para la implementación de *Slash and Pair*, se han utilizado una serie de tecnologías que se van a detallar a lo largo de este capítulo, tanto una breve explicación de lo que es como los motivos por los que se ha utilizado en el proyecto.

La aplicación ha sido escrita en Java[20] en su versión 1.8, la elección del lenguaje de programación viene dada al uso de otras tecnologías que se han utilizado que se comentarán más adelante.

Para la implementación de las diferentes vistas de la aplicación, como han de ser páginas web se ha utilizado HTML, CSS y *JavaScript*, concretamente en este último se ha utilizado la librería *jQuery*. La elección de HTML y CSS es trivial ya que es lo más útil para hacer páginas web hoy en día, *JavaScript* se ha utilizado sobre todo como herramienta para extraer los datos de los sensores y recibir estos datos en el ordenador y realizar acciones.

5.4.2 Spring Framework

La primera versión de *Spring*[21] fue escrita por Rod Johnson[22], actualmente pertenece a la empresa *Pivotal Software* y es de código libre.

Spring Framework es un conjunto de módulos completamente independientes los cuales se utilizan según las necesidades de cada proyecto. Proporciona un modelo de programación y configuración exhaustiva de aplicaciones basadas en Java.

En un software se suelen utilizar diferentes *frameworks* o librerías, las cuales requieren de unas instancias o ejecuciones en las que se crean atributos. Normalmente el desarrollador ha de lidiar con esto para que todos los *frameworks* y tecnologías utilizados en la aplicación puedan convivir.

Spring ayuda a solventar este problema ya que basándose en ficheros XML o anotaciones se encarga de construir todos los objetos necesarios para que el conjunto de *frameworks* funcione en armonía.

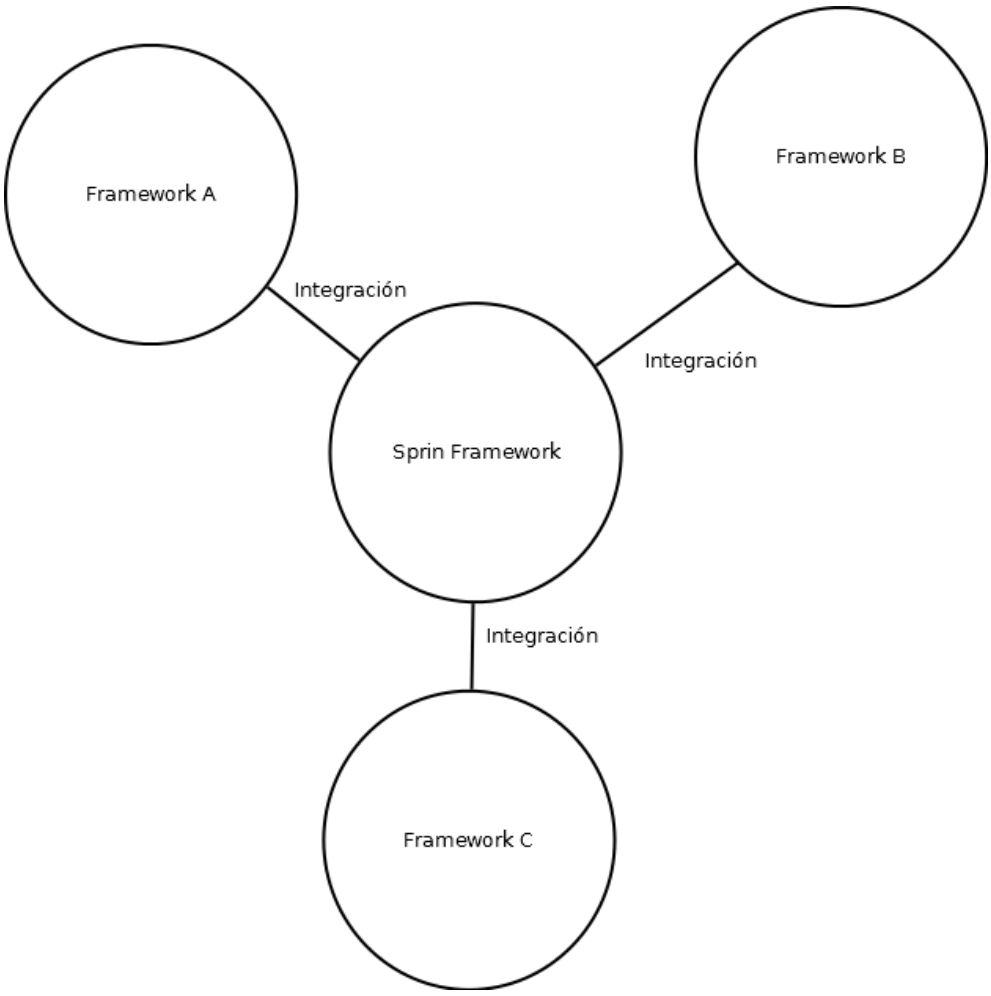

Figura 16: *Spring* integra los diferentes *frameworks*.

Esta armonía entre los *frameworks* se consigue mediante el diseño IoC el cual se ha comentado previamente. *Spring* implementa el contenedor de IoC el cual inyecta a cada objeto los atributos necesarios según las relaciones descritas en el fichero de configuración XML o en las anotaciones.

Esta es una de las razones por las que se ha elegido *Spring* como *framework*, ya que en *Slash and Pair* por su funcionalidad, varias tecnologías han de convivir y el flujo de ejecución que se tiene no es lineal.

Dentro de *Spring* existen multitud de complementos, uno de los cuales se utiliza para aplicaciones web, en el caso de *Slash and Pair* se ha tomado la decisión de utilizar *Spring MVC*[23], el cual ofrece una serie de ventajas en la aplicación del patrón MVC que se ha visto anteriormente que es necesario en la aplicación.

5.4.3 Maven

Slash and Pair es un proyecto que requiere de librerías para su implementación. La gestión de éstas sería un trabajo muy costoso por parte del desarrollador. Por este motivo hemos incluido *Maven* [24].

Es un gestor de dependencias, se encarga de gestionar todas las librerías, las importa y las descarga en una caché local, de esta manera incluyendo de la manera correcta la librería en el archivo *pom.xml*, que es el de configuración, *Maven* gestionará todo de forma transparente al desarrollador, evitando mucho trabajo.

A parte de la gestión de librerías tiene otras muchas tareas útiles.

- Puede construir el código del proyecto *Java*.
- Puede ejecutar tareas de *testing*.
- Realiza control de versiones.
- Sirve para desplegar el proyecto en un servidor.
- Permite la descarga de arquetipos que se usan como plantilla para la creación de nuevos proyectos.
- Crea y despliega la documentación en formato HTML.
- Se puede integrar con gestores de repositorios como *Git*

Como *Slash and Pair* está dividido en diferentes proyectos, los cuales tienen muchas dependencias en común, se requiere de un gestor de dependencias como *Maven*, en el cual se define la versión y la librería necesaria y las descarga para que todos los proyectos utilicen lo mismo.

A parte de este motivo, también ayuda en la compilación del código, cosa muy a tener en cuenta ya que al tener tantas dependencias externas y también internas entre los proyectos, es una forma fácil de conseguirlo.

5.4.4 Elección de Base de datos Redis

Redis[25] es un motor de base de datos, de código libre, está escrito en ANSI C por Salvatore Sanfilippo, se creó en el año 2009.

La base de datos de *Redis* es del tipo clave-valor persistentes que residen en memoria RAM y posteriormente vuelca el conjunto de datos almacenados al disco duro.

Redis es cliente/servidor por lo que levanta su servicio en el servidor y responde peticiones. Cuenta con interfaz de red, lo que hace posible la conexión de clientes o nodos desde otro *host*.

Esta función es muy interesante para el proyecto pensando en términos de escalabilidad, en la que los dispositivos de un usuario se conectan a servidores separados, entraremos en más detalle de la escalabilidad en siguientes apartados.

Uno de los puntos fuertes de *Redis* es su forma sencilla y eficiente de resolver problemas que no necesitan la complejidad de las bases de datos relacionales. Es mayormente utilizado para incorporar soluciones de cache o como *backend* de operaciones en línea en escenarios de alta demanda.

La eficiencia viene dada a que su único trabajo es establecer y recuperar(*set y get*) datos sobre estructuras con las que cuenta.

Debido a su simplicidad y eficiencia, se ha tomado la decisión de utilizar este sistema para el emparejamiento de dispositivos, ya que no requiere de una base de datos más compleja que guardar y recuperar datos con clave valor, que son código de emparejamiento e id de sesión. De esta manera tenemos una solución eficiente a este problema.

5.4.5 QR codificador y decodificador

En la funcionalidad de emparejamiento de la aplicación, como método principal para conseguirla, se ha de generar un código QR que mostrar en el navegador del ordenador y este ha de poder ser leído desde el dispositivo móvil.

Para codificar el código de emparejamiento desde el servidor del ordenador se utiliza una librería la cual se llama *ZXing*[26], con esta librería se realiza esta tarea.

ZXing es una librería de código abierto y es capaz de codificar y decodificar en la mayoría de formatos disponibles en la actualidad para códigos QR.

Desde la parte móvil se ha hecho uso del proyecto *WebCodeCamJS*[27] el cual ha sido desarrollado por Tóth András. Adaptando este proyecto a *Slash and Pair*, se tiene un método por el cual se puede disponer de la cámara del dispositivo y enfocando a un código QR, decodificarlo y enviar la información al servidor de móvil.

5.4.6 Elección de la cola RabbitMQ

RabbitMQ[28] [29] es un *bróker* de mensajería de código abierto que implementa un protocolo avanzado de mensajes por colas. Está escrito en *Erlang* y utiliza el *framework Open Telecom Platform* para construir sus capacidades de ejecución distribuida y conmutación ante errores.

El proyecto consta de diferentes partes, un servidor de intercambio, pasarelas para los protocolos HTTP, XMPP y STOMP, bibliotecas de clientes para Java y el *plugin Shovel* que se encarga de copiar mensajes desde un corredor de mensajes a otro.

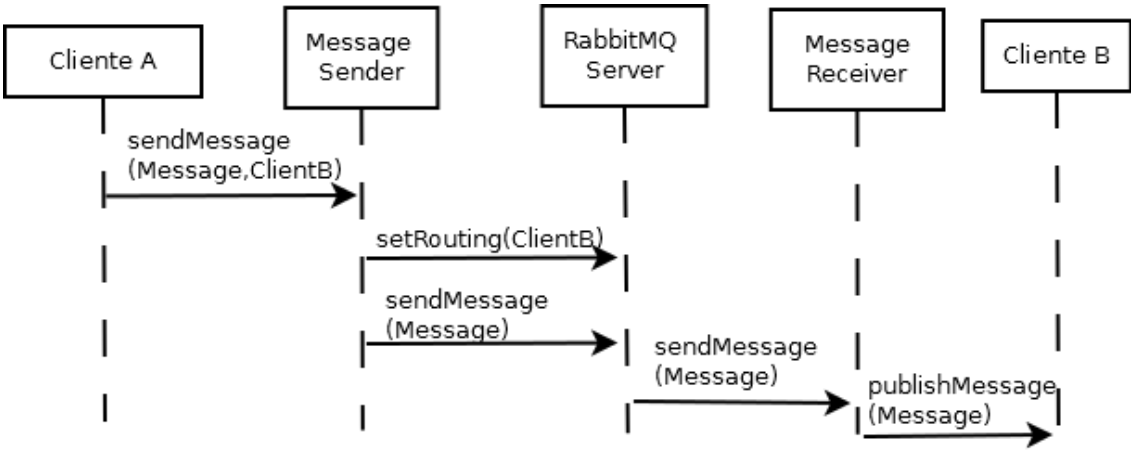

Figura 17: Diagrama de clases de *RabbitMQ*.

Como se ve en la figura (17), el funcionamiento de *RabbitMQ* es el siguiente: el Cliente A manda un mensaje a la cola, el mensaje se manda a través de una clase *Message Sender*, la cual guarda el mensaje en el servidor de *RabbitMQ*, el cual envía el mensaje a *Message Receiver* que publica el mensaje en el cliente B.

Uniendo el patrón de observador y la cola *RabbitMQ* se tiene una parte de la funcionalidad de la interacción de los dispositivos, el envío de datos entre servidores.

Se ha elegido *RabbitMQ* para el proyecto ya que suple el problema que se tiene de la comunicación de los datos de los sensores entre los proyectos, se necesita una tecnología la cual corresponde al patrón observador comentado anteriormente, y *RabbitMQ* cumple con este objetivo.

En el proyecto, desde el servidor de móvil se envían los datos a la cola, la cual notifica al servidor del ordenador mediante la estructura explicada.

Con la cola tenemos solucionado la parte de la comunicación de los datos entre servidores, en el siguiente apartado se comentará como se ha resuelto el problema de comunicar los dispositivos con sus respectivos servidores.

5.4.7 Elección de WebSockets

Un *WebSocket*[30] es una conexión bidireccional punto a punto entre navegador y su servidor, ambas partes pueden iniciar una comunicación enviando mensajes el uno al otro. Para la parte cliente no resulta ninguna innovación pero para la parte servidor si lo es, ya que se permite el envío de información al cliente sin obligar a éste a realizar ninguna petición.

WebSocket viene como estándar a partir de Java EE7 en la especificación JSW 356. Aunque muchos servidores ya contaban con una implementación propietaria de esta tecnología.

El funcionamiento de los *WebSockets* en un servidor es el siguiente, para cada una de las conexiones del *WebSocket* que generemos, el servidor asigna un hilo de ejecución (*thread*).

El diagrama sería cercano al mostrado en la figura (18).

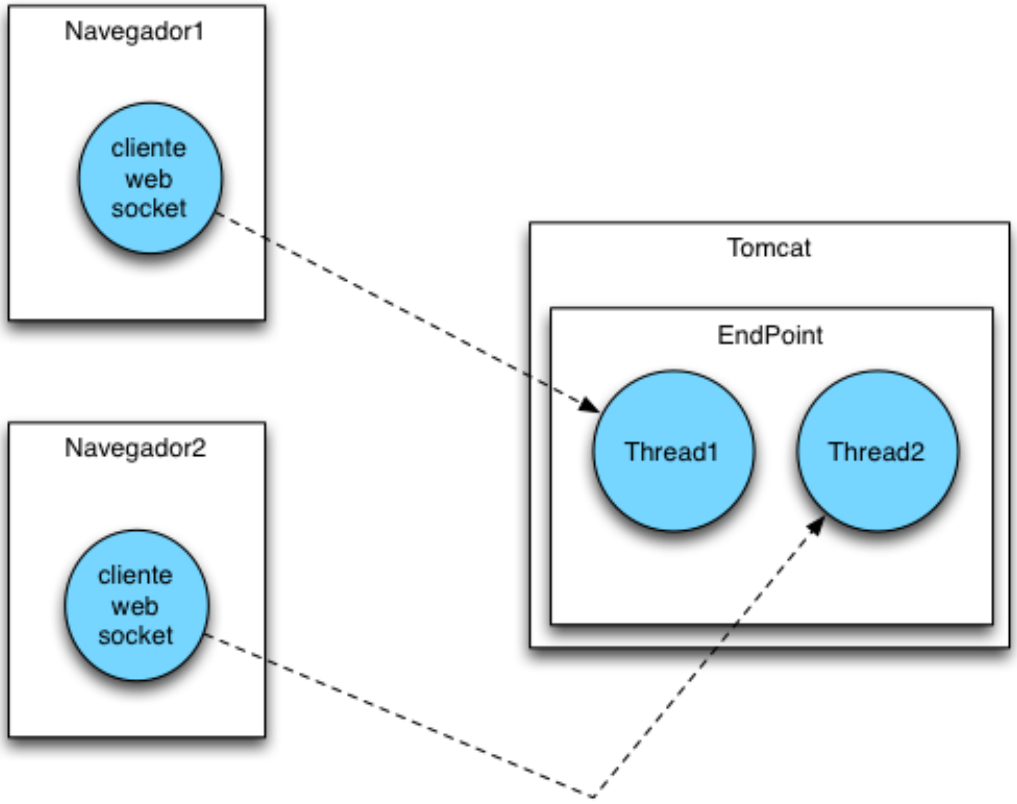

Figura 18: Diagrama de funcionamiento de *WebSocket*.

En el navegador del cliente *Slash and Pair* utiliza una librería de *JavaScript* llamada *SockJS*[31]. Mediante esta librería se realiza la conexión del *WebSocket* por parte del navegador en ambos dispositivos, por el cual se enviarán los datos de los sensores al servidor en caso del móvil y del servidor al ordenador en el otro caso.

Esta tecnología es necesaria en el proyecto ya que existe una comunicación permanente entre el cliente y el servidor, en todo momento se están enviando datos por lo tanto se necesita de un canal abierto para satisfacer esta necesidad.

Como se ha comentado por *WebSocket*, el servidor puede enviar información al cliente en cualquier momento, este es el otro motivo de la elección de esta tecnología ya que en la parte del ordenador, el servidor le ha de comunicar todos los datos de los sensores que le están llegando para que el usuario vea la reacción de sus acciones en el dispositivo móvil sin tener que hacer nada en el ordenador.

La conexión con el *WebSocket* por parte del cliente móvil se realiza justo después del proceso de emparejamiento, una vez se ha confirmado que se ha realizado con éxito, se redirige al navegador del *smartphone* a otro HTML desde el cual se realiza la conexión del *WebSocket*.

Por parte del escritorio la conexión se realiza justo cuando el navegador del

cliente hace la petición a la primera página.

5.4.8 Sensores accesibles de un smartphone

La interacción de los dos dispositivos requiere los datos de los sensores de los móviles, para ello se ha realizado un análisis de todos los que son accesibles mediante API para poder recoger los datos desde JavaScript en el navegador del dispositivo y enviarlos al servidor.

Se detallan en una lista todos los sensores que se podrían utilizar:

- Geolocalización[32]: La API de geolocalización sigue el estándar marcado por W3, permite localizar la ubicación del usuario utilizando las capacidades de posicionamiento de su dispositivo.
- Giroscopio [33]: Los eventos de giroscopio son eventos de orientación, los cuales permiten saber la orientación del dispositivo según sus tres ejes.
- Cámara [34]: La API permite acceder a la cámara del dispositivo.
- Vibración [35]: Permite a los desarrolladores proporcionar información no audible al usuario aprovechando el motor de vibración de los dispositivos.
- Sensor de luminosidad [36]: Permite acceder al sensor de luz del dispositivo, se puede utilizar para añadir un modo nocturno por ejemplo.
- Sensor de proximidad [37]: Proporciona a los desarrolladores acceso a los datos sobre objetos que están cerca del teléfono con el sensor incorporado.
- Sensor de batería [38]: Permite acceder al nivel de batería del dispositivo

Para la demostración que se ha implementado en este proyecto, el único sensor necesario es el giroscopio ya que es suficiente para mostrar el potencial de la aplicación ya que da mucho juego para el usuario en cuanto a que ve cómo su móvil puede convertirse en un mando de consola accediendo a la aplicación, por lo tanto es el que está implementado.

En el momento en el que se requiera del uso de cualquiera de los sensores comentados se podría añadir sin dificultad.

De esta manera se tiene solucionado el problema de comunicación de los dispositivos con sus respectivos servidores cerrando así la funcionalidad de interacción entre dispositivos. En siguientes capítulos se detalla cómo interactúan estas tecnologías entre sí.

5.5 Implementación

En el apartado anterior se ha detallado la elección de las tecnologías utilizadas en el proyecto.

A lo largo de este apartado se explica el funcionamiento de todas las tecnologías en el proyecto, comentando tanto su integración en el proyecto como el funcionamiento en conjunto entre todas ellas.

5.5.1 Servicios de Spring y separación de responsabilidades

Spring cómo se ha comentado anteriormente, es un *framework* el cual una de sus funciones es poder integrar una serie de tecnologías diferentes y que todas puedan funcionar en conjunto.

Existe una serie de servicios que se pueden integrar dentro de *Spring*, lo cual facilita mucho el uso de dichas tecnologías. Para la instalación de estos servicios en el proyecto es suficiente con hacer uso de *Maven*.

En cada proyecto existe un archivo para declarar todas las dependencias que ha de gestionar el proyecto, las cuales se descargan y se integran mediante *Maven*, este archivo se llama ?*pom.xml*?

5.5.2 Integrando de tecnologías en Spring

Ya se ha hablado de las diferentes tecnologías utilizadas y del funcionamiento en conjunto de todas en la aplicación mediante el uso de *Spring*, en este apartado se explica la integración de estas en el proyecto.

Integrando Redis en Spring *Redis* es uno de los servicios que están integrados en *Spring*, ya existen unas clases específicas las cuales añadiendo las dependencias pertinentes y teniendo una instancia de *Redis* ejecutándose, se puede hacer uso de esta base de datos.[39]

Una vez está integrado, *Spring* automáticamente, al arrancar se conecta a la base de datos para poder realizar las acciones pertinentes en cada momento. En este caso se realiza una escritura de datos con el código de emparejamiento y el identificador de sesión desde el servidor del escritorio, y una lectura desde el servidor de móvil con la que se busca si existe el código introducido en el *smartphone* para realizar el emparejamiento. Una vez se encuentra se lee el id de sesión del ordenador del usuario.

Integrando RabbitMQ en Spring Con *RabbitMQ* pasa lo mismo que con *Redis*, añadiendo unas dependencias al archivo de configuración *pom.xml* ya se puede hacer uso de esta tecnología dentro de *Spring*. Lo único necesario a parte de las dependencias es tener una instancia ejecutándose de un servidor *RabbitMQ*,

para lo cual hace falta tener instalado el software proporcionado en la página de la oficial de *Rabbit*.

Una vez el proyecto está listo para el uso de *RabbitMQ*, es necesario tener una clase en la cual se van a configurar las colas que se utilizan en la aplicación. En el caso de *Slash and Pair* se tienen dos colas, una se utiliza para la comunicación de que se ha realizado el emparejamiento correctamente, y se tiene otra por separado por la que viajarán los datos de los sensores.

La funcionalidad se separa en los dos proyectos, en el de móvil la funcionalidad se escriben datos en esta cola, para lo cual se ha de definir en qué cola se guardará el dato y enviar este dato, si en el servidor de *RabbitMQ* que se tiene no existe la cola que se está definiendo se crea en ese momento, en el caso de que si que exista se escriben los datos en ella.

En el proyecto del ordenador, existe un objeto que se suscribe a las colas que le interesen, lo hace a las mismas que el servidor de móvil escribe los datos. Para realizar esta acción existen una serie de etiquetas propios de *RabbitMQ* en los cuales se puede mantener la referencia para ser notificado cada vez que se actualicen las colas, como se ha explicado en la descripción del patrón *obverver* en capítulos anteriores.

Integrando WebSockets En el caso de la comunicación entre los dispositivos y los servidores para crear el flujo de datos continuo proveniente de los sensores del móvil, se han utilizado *WebSockets* como ya se ha comentado en apartados anteriores.

Está tecnología se ha de integrar en dos partes de la aplicación, por un lado ha de hacerse uso desde *Spring* y por otro también ha de poder utilizarse en el lado del dispositivo lo cual se realiza por mediante *JavaScript*.

Por la parte de *Spring* se han de añadir las dependencias como en los demás casos. Aparte se ha de realizar una configuración para lo cual se crea una clase en la que se definen los parámetros para el correcto funcionamiento de esta tecnología.

En esta configuración se define que la comunicación mediante los *WebSocket* ha de ser de un usuario con su sesión, para que sus datos no lleguen a otros. Para lo cual existen unos *endpoint* que se definen automáticamente para cada usuario mediante el uso de una función de *Spring*.

Como se realiza la comunicación para diferentes tareas como pueden ser diferentes sensores, se configura un canal para cada uno de ellos, cada cual con sus *endpoints*. Por la parte del cliente se utiliza la mencionada librería de *JavaScript SockJS*, en ella se han de definir los mismos *endpoint* que se han utilizado para la parte de *Spring*, de esta manera quedan definidos los canales de comunicación para el correcto funcionamiento de *Slash and Pair*.

5.6 Diseño de interfaz

Una interfaz de usuario ha de cumplir con una serie de características, desde ella se ha de tener acceso a todas las funciones definidas en los casos de uso de la aplicación, ha de ser intuitiva y ha de tener un diseño usable.

El sistema *Slash and Pair* dispone de una interfaz de usuario para realizar la conexión de los dispositivos, desde la cual se realiza el emparejamiento.

Figura 19: Página de ordenador en la que se muestra el código QR.

En el ordenador se tiene una página que muestra un código QR(figura (19)) además de un botón el cual se ha de clicar en caso de que no funcione la cámara del dispositivo móvil, en ese caso se sustituye el QR por un código como se ve en la figura (20).

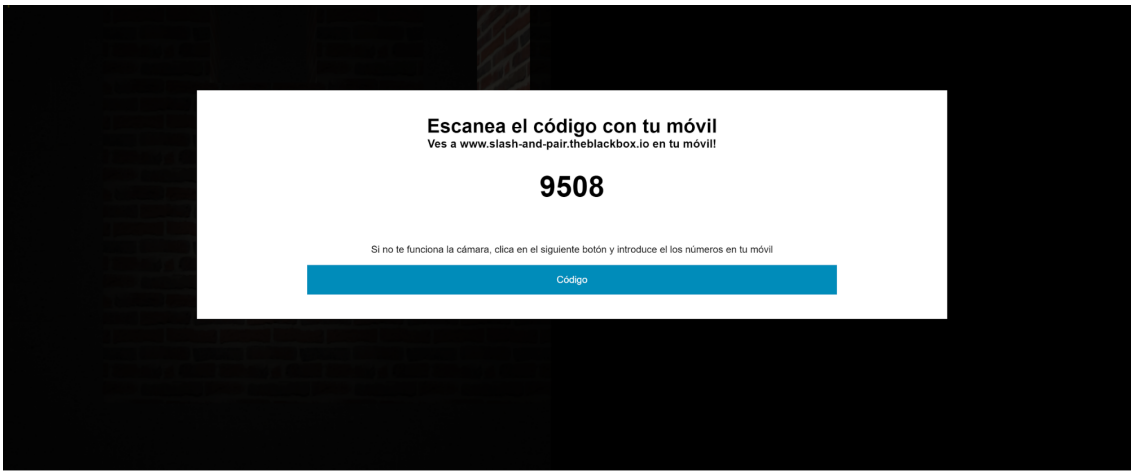

Figura 20: Página del ordenador mostrando el código de dígitos.

En la parte de móvil tenemos una página html en la que se muestran dos botones,

un recuadro en el que se verá la imagen retransmitida por la cámara cuando esta se inicie, instrucciones para ayudar al usuario a realizar las acciones y un cuadro de texto como se muestra en la figura x.

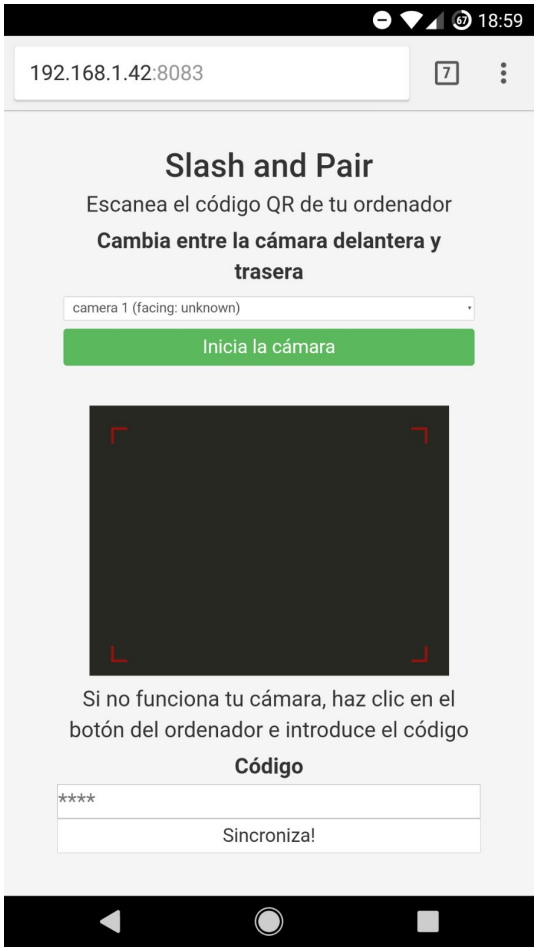

Figura 21: Página de emparejamiento de móvil.

Para empezar el botón en el que se muestra el texto de ?inicia la cámara? tendrá la funcionalidad de iniciar la retransmisión de la cámara por la pantalla del móvil para que el usuario pueda enfocar el código QR, una vez clicado este botón, se encenderá la cámara y el usuario verá en directo las imágenes retransmitidas por ella.

Una vez iniciada la cámara, el usuario enfocará el código QR con el que se realizará el emparejamiento, el cual es transparente al usuario, este no tiene que realizar ninguna acción ya que el navegador le redirige automáticamente a la siguiente página.

Dado el caso de que por su sistema en el móvil o cualquier otro motivo no tenga acceso a la cámara del mismo, dispone de instrucciones de las acciones alternativas que ha de realizar para el emparejamiento. Dichas funciones son, clicar en el botón específico en su ordenador e introducir en el dispositivo móvil el código mostrado en el cuadro de texto.

Tanto el camino habitual en el que se hace uso del escaneo del código QR como en el camino alternativo en el que el usuario ha de introducir el código mostrado por el ordenador son intuitivos para que cualquier usuario pueda ver qué acciones ha de realizar.

Se deja claro que ha de hacer clic en el botón de *play* para accionar la cámara y en caso de que esto no funcione, podrá ver claramente cómo seguir el caso alternativo.

Para la funcionalidad de interacción entre los dispositivos por la parte del ordenador la interfaz es la de la aplicación que haga uso de los datos enviados por el sistema. Por la parte del móvil tenemos una página que da información al usuario de cómo utilizar el sistema o añade elementos mediante los cuales el usuario puede realizar acciones, esto depende de la aplicación que se le den a los datos de los sensores.

En la figura (22), se muestra un ejemplo de la interfaz de usuario en el caso de la interacción.

Figura 22: Página de móvil en horizontal para la interacción entre dispositivos.

En la figura (22) también se muestra un botón mediante el cual el usuario haciendo clic puede mantener el dispositivo despierto sin que se apague la pantalla hasta que cierre la aplicación web.

El funcionamiento de la aplicación hace uso del giroscopio, por lo tanto se requiere que el móvil esté en horizontal por ello disponemos de unas pequeñas instrucciones de cómo ha de utilizar el móvil de forma correcta para que sus acciones tengan el efecto esperado en el ordenador. Por lo que si el usuario tiene el *smartphone* en vertical, se le dan instrucciones para que lo ponga en horizontal(figura (23)), una vez puesto en horizontal, estas instrucciones desaparecen.

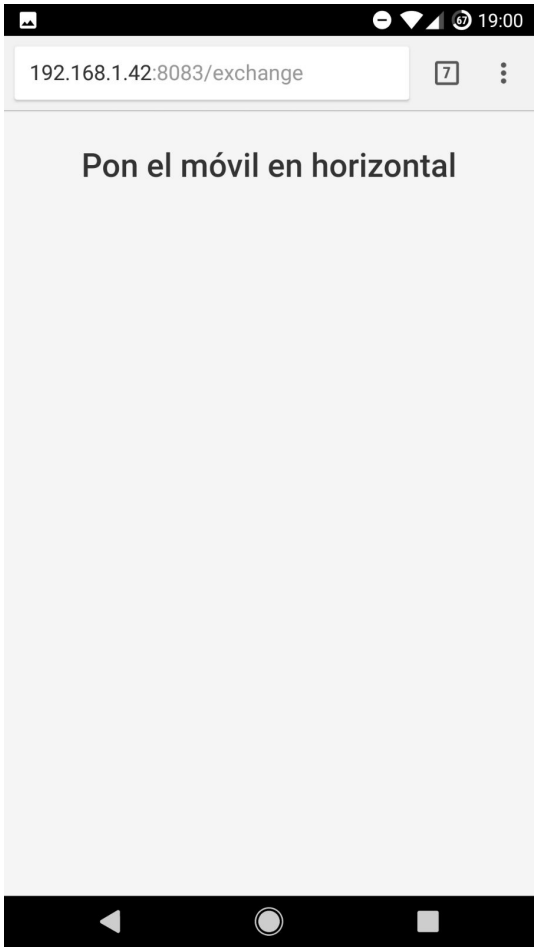

Figura 23: Página de instrucciones para el usuario.

Estas acciones son muy intuitivas ya que el usuario verá que los movimientos que realice con el *smartphone* tienen una reacción en el ordenador, y rápidamente podrá asociar sus acciones al funcionamiento de la aplicación.

5.7 Despliegue

En este apartado se detalla el despliegue de la aplicación en un entorno real comentando todos los pasos seguidos para preparar el servidor y cómo se diseñan *scripts* para siguientes subidas a producción.

Slash and Pair se instala en un servidor en el cual está instalado el sistema operativo *Ubuntu 14.04*.

El primer paso es instalar todas las herramientas software para el funcionamiento del sistema.

Para empezar se necesita la versión de Java en la que se ha escrito *Slash and Pair*, por lo tanto se hace la comprobación de tener la versión 1.8 de Java en el servidor.

Una vez con la versión necesaria de Java, se precisa de la instalación del software *RabbitMQ* y *Redis*.

Para este paso se siguen los comandos de consola necesarios para la instalación, en el anexo se encuentra más detallados los que se han utilizado en el caso de este trabajo.

Una vez realizadas todas las comprobaciones y con todo funcionando correctamente, se pasa a la instalación de *Slash and Pair* en el servidor,para ello lo primero es crear las carpetas necesarias en el servidor que albergarán el código. En el caso de la aplicación se crea una carpeta para *desktop* y otra para móvil.

El siguiente paso es la descarga del código desde *Github* en el repositorio en el que se ha estado trabajando.

Se descomprime el archivo que se acaba de descargar con el código y se procede a la compilación del código mediante comandos de *Maven* los cuales se pueden ver en el anexo.

Con la compilación se generan dos archivos *.jar*, uno con el proyecto de ordenador y otro con el de móvil, los cuales se mueven a sus respectivas carpetas creadas anteriormente.

En este punto se dispone de los archivos necesario para la instalación del sistema y se procede a la configuración del servidor *Apache tomcat*[40], el cual se ha de instalar si no se dispone de él.

En el servidor se configura un virtual *host* para que varias aplicaciones puedan convivir en el mismo. Para ello se configura el nombre de la aplicación con su dominio, se definen los puertos a través de los cuales funcionará todo y se configuran los archivos en los que se guardarán los *logs* de la aplicación.

Con todo el servidor instalado y configurado, se procede a la creación de dos *scripts*, los cuales sirven para arrancar y parar la aplicación en el servidor respectivamente.

En el *script* con el que se arranca la aplicación, se pide que se guarden los *logs* en un archivo, se tiene para que pueda ejecutarse en segundo plano en la terminal y para que cuando esta se cierre, el sistema siga en ejecución.

Se guarda el *pid* de la aplicación en un archivo, para cuando se quiera recuperar, poder hacerlo con más facilidad.

Listing 1: *Script start* de la aplicación

```
nohup java \
        -jar $(HOME)/app/desktop/.jar \
        >> $(HOME)/log/desktop/app_$(date +%Y_%m_%d_%H).log 2>&1 &
echo $! > $(HOME)/pid/desktop.txt
```

El *script* de *stop* simplemente para la ejecución matando el proceso en el que se esté ejecutando.

Listing 2: *Script stop* de la aplicación

```
kill $(cat ${HOME}/pid/desktop.txt)
```

Como último paso en el despliegue de la aplicación, se crea un certificado SSL para que la aplicación disponga de conexión HTTPS para acceder directamente al servidor, de esta manera se obliga al cliente a conectarse por esta vía. Esto es necesario, aparte de para aportar seguridad a la aplicación, para el correcto funcionamiento de la cámara por parte de los dispositivos móviles, ya que desde algunos navegadores si no se tiene esta certificación, no permiten el acceso a esta funcionalidad.

Para ello se hace uso de un software llamado *Certbot*[41], el cual con una serie de comandos, el solo prepara todo lo necesario para que a partir del momento en el que se ejecute, se disponga de conexión mediante HTTPS.

Una vez finalizados todos estos pasos se tiene acceso a la aplicación a través de cualquier navegador y dispositivo mediante HTTPS.

6 Ejemplo de ejecución

Slash and Pair es un sistema en el que se sincronizan dos dispositivos y se envía información de los sensores de un móvil a un ordenador, como se ha explicado a lo largo del documento.

La ejecución al ser una aplicación web ha de ser a través de un navegador. Para realizar las pruebas se han utilizado principalmente dos navegadores en dispositivos móviles, los cuales son *Chrome* y *Safari*.

En un principio para realizar pruebas se implementó una interfaz para la página vista en el ordenador, la cual mostraba la información en tiempo real que le llegaba de los sensores del móvil. El emparejamiento se realiza mediante un código de 4 dígitos mostrado en el ordenador que se tenía que introducir en el *smartphone*.

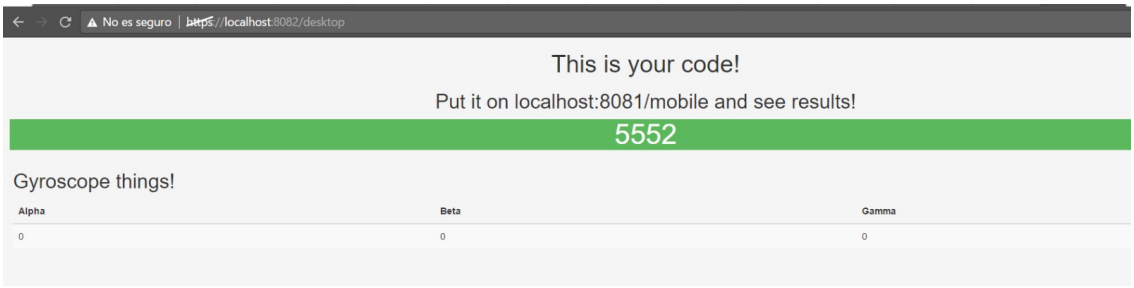

Figura 24: Interfaz de las primeras pruebas del sistema.

Como se observa en la figura (24), se tiene el código y se muestran los valores del giroscopio en tiempo real.

Con esto se tiene una respuesta real del sistema para realizar todas las pruebas, pero no se ve todo el potencial del sistema.

Para un ejemplo real de ejecución se ha realizado una implementación, para adaptar un juego *open source* en *JavaScript* a *Slash and Pair*.

6.1 Realización de la demo

Como demostración del potencial de *Slash and Pair*, se ha realizado la adaptación del sistema a un juego *JavaScript*, el cual aprovecha los datos del sensor de giroscopio.

La primera acción que ha de realizar el usuario, y lo primero que se encuentra, es la página para realizar el emparejamiento como se ha comentado en el diseño de la interfaz.

Una vez realizado el emparejamiento se inicia el juego.

Consiste en un laberinto en el que el usuario tiene que mover una pelota para que ésta salga de él. Para la interacción con la pelota, el usuario debe utilizar las flechas del teclado para dirigir la pelota hacia la salida.

La adaptación del juego a *Slash and Pair* tiene como objetivo es conseguir que la pelota se mueva en reacción a los movimientos que se realicen con el *smartphone*.

Solo dos de los tres ejes son necesarios para esta demostración ya que con estos dos ejes podemos capturar el movimiento de hacia adelante, atrás, derecha e izquierda.

En el juego implementado realmente solo necesitamos saber si ha de ir en una dirección o en otra, para ello se ha definido un rango de ángulos para cada lado, de manera que cuando sobrepasa el valor 10 obtenido de los sensores va en la dirección en la que se esté inclinando el *smartphone*. Para un mejor funcionamiento de la aplicación, se ha reducido el envío de datos desde el giroscopio del dispositivo móvil acotando a que envíe datos en un rango de 5 a 15, de esta manera es suficiente para saber si se sobrepasa o no se llega a 10 en el valor del giroscopio y se puede controlar en qué dirección se mueve la pelota

El funcionamiento es muy intuitivo, lo primero que se ha de hacer una vez está el sistema funcionando, es dejar el móvil en horizontal y plano, como si estuviera encima de una mesa.

A partir de aquí lo único que ha de hacer el usuario es inclinar el móvil en la dirección en la que quiera mover la pelota.

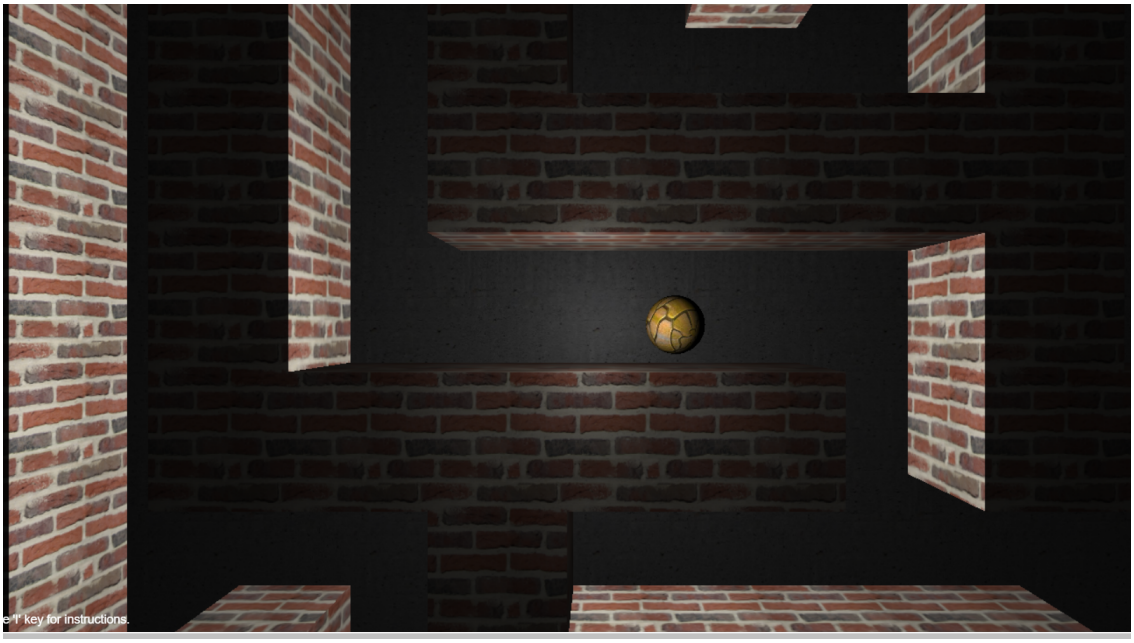

Figura 25: Fotograma del juego en el que se ha adaptado Slash and pair.

La adaptación de Slash and pair, se ha realizado mediante dos funciones dentro del juego, a las que se llaman para iniciar el movimiento o detenerlo, según los datos que le lleguen.

Los datos llegan a través de una librería JavaScript que se ha de añadir a la aplicación de demostración.

Este juego es open source, se puede consultar la evolución del proyecto en el

siguiente enlace: https://github.com/wwwtyro/Astray

El laberinto dispone de niveles que se van generando de forma aleatoria, a cada cual con más dificultad que el anterior.

Como se ha explicado anteriormente, el sistema está desplegado en un servidor Apache, el cual está funcionando y se puede acceder desde la url:

https://slash-and-pair-d.theblackbox.io para el ordenador

https://slash-and-pair.theblackbox.io para el móvil

7 Líneas de futuro

En esta sección se propondrán algunos ejemplos en los que el sistema puede tener continuidad o tomarse como base para otros proyectos que podrían ser perfectamente nuevos trabajos final de carrera.

Slash and Pair es el inicio de un proyecto que tiene tantas posibilidades como la imaginación dé de sí.

Un ejemplo claro de continuidad es uno de los objetivos que se marcó en la idea inicial que se tenía en el proyecto: Realizar un *framework* desde el que poder integrar el sistema en cualquier plataforma o cualquier web. De esta manera sistemas de terceros podrían hacer uso de lo desarrollado en este trabajo con un esfuerzo mínimo de implementación, este punto sería muy interesante ya que puede ser toda una revolución para juegos de navegador por ejemplo, en los que con elementos de los que todo el mundo dispone, sería una experiencia totalmente renovada y podría ser un gran empuje para típicos mini juegos.

Otro trabajo que se podría realizar en un futuro con la base de *Slash and Pair* es el estudio de *User experience*, se pueden analizar las nuevas formas en las que un usuario puede interactuar con la web mediante el sistema. Por poner un ejemplo, se podrían crear nuevas formas de interactuar con redes sociales mediante gestos, o realizar compras en tiendas online?

Se pueden realizar adaptaciones específicas del sistema, por ejemplo como se ha comentado, puede adaptarse para juegos, en los que el *smartphone* pueda ser un potente mando al alcance de cualquiera.

Se puede hacer un estudio para reconocer gestos, con lo cual el sistema tenga unos patrones en los que detecte si el usuario hace un movimiento por ejemplo de cerrar una puerta, de lanzar un objeto?

Mediante estos gestos se podría modificar el sistema para que los datos en vez de ser enviados a un navegador de un ordenador, se envíen por ejemplo a un dispositivo *Arduino* el cual puede estar programado para realizar acciones en un domicilio, o en cualquier otro sitio, por lo que puede ser útil en el internet de las cosas, otra opción puede ser intentar recoger los datos de los sensores desde otros dispositivos como por ejemplo un *smartwatch*.

Se puede implementar una plataforma para realizar un *Software as a service*, en el cual se preparen entornos que puedan utilizar el sistema implementado, y preparar una versión para ser utilizada por terceros.

8 Conclusiones

El último apartado del documento se hace un repaso de todo el sistema comentando las conclusiones a las que se ha ido llegando durante el transcurso del mismo.

A los inicios del proyecto se tiene una ligera idea de lo que ha de ser el sistema a implementar, esta idea es desarrollar un software para la comunicación entre dos dispositivos, un móvil y un ordenador.

Una vez establecido el objetivo se profundiza más en definir todas las funcionalidades con las que debe de cumplir. La primera función de la aplicación es la de emparejar dos dispositivos (un ordenador y un móvil) para que puedan enviar datos entre sí de manera continua. La segunda funcionalidad del proyecto es el envío de datos entre los mencionados dispositivos de manera que el ordenador muestre la interacción de un usuario con el dispositivo móvil.

Con el proyecto definido el proyecto, se ha buscado información de lo que es un software de calidad en la actualidad, para ello primero se estudian los estándares vigentes. Gracias a este estudio se ha observado que el software ha de cumplir con una serie de características para su correcto desarrollo.

A partir de aquí se ha investigado las posibles arquitecturas que puede tener *Slash and Pair* para cumplir con todos sus objetivos. Se llega a la conclusión que existen una serie de patrones de programación que pueden ser muy útiles para resolver el problema planteado, como por ejemplo: el patrón MVC, *Observer* o IoC; patrones que se tienen muy en cuenta en el desarrollo de la aplicación.

Posterior al estudio de los diversos patrones, se pasa a realizar el diseño de las funcionalidades que se han comentado anteriormente, así como el diseño del funcionamiento general de la aplicación. Es en este punto donde se pone en práctica lo aprendido sobre patrones de software. En estos diseños se deciden cosas como la división del proyecto en dos partes, una para móvil y otra para el ordenador, que se han implementado en un equipo de dos personas. Este documento se centra más en la parte del móvil a pesar de que la arquitectura ha sido trabajada para ambos dispositivos.

La aplicación se ha dividido en dos proyectos diferentes, uno para la parte móvil y otro para ordenadores. Esta división tiene mucho sentido. Con el estudio de la escalabilidad de *Slash and Pair* se ha observado que gracias a esto el sistema puede instalarse en diferentes servidores sin perder la comunicación entre los dispositivos aunque cada uno realice la conexión con diferentes servidores, esto está explicado en el apartado de escalabilidad del documento.

En la siguiente fase del proyecto se da inicio a la implementación del código. Aquí está el reto de hacer convivir todas las tecnologías utilizadas ya que no es un asunto trivial. En este punto se ve la importancia de *Spring* como base del proyecto.

Durante la implementación se ha aprendido a manejar multitud de tecnologías partiendo de cero, sin tener ningún proyecto anterior como base, esto ha requerido el estudio del funcionamiento de estas tecnologías para conseguir un software funcional.

Este es un proyecto mediante el cual se puede ver el proceso de creación de una aplicación desde el punto de vista de la arquitectura. En el presente documento se puede encontrar información de la toma de decisiones a lo largo de la implementación de *Slash and Pair* y se finaliza con la puesta en un entorno de producción de dicho sistema dando por respondido el problema inicial de una forma satisfactoria.

Referències

[1] Mora, S. L. (2002). *Programación de aplicaciones web: historia, principios básicos y clientes web*. Editorial Club Universitario.

[2] Nafría, I. (2007). W*eb 2.0: El usuario, el nuevo rey de Internet. Gestión 2000.*

[3] Arroyo-Vázquez, N. (2009). *Web móvil y bibliotecas. El profesional de la información, 18(2)*, 129-136.

[4] Simpson, R. (1 de Noviembre de 2016). *StatCounter*. Recuperado el 5 de Mayo de 2016, de *http://gs.statcounter.com/press/mobile-and-tablet-internet-usage-exceeds-desktop-for-first-time-worldwide*

[5] FDI. (22 de Octubre de 2008). *FDI*. Recuperado el 11 de Mayo de 2017, de *https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf*

[6] Cockburn, A. (2008). Using both incremental and iterative development. *STSC CrossTalk* (*USAF Software Technology Support Center*), 21(5), 27-30.

[7] Trello inc. (2011). *Trello*. Obtenido de *https://trello.com/*

[8] Git. (2009). *Git*. Obtenido de *https://git-scm.com/*

[9] GitHub. (Abril de 2008). Obtenido de *https://github.com/*

[10] Spring. (s.f.). Recuperado el Abril de 2017, de *https://spring.io/guides/gs/spring-boot/*

[11] Spring. (s.f.). Recuperado el Mayo de 2017, de *https://spring.io/guides/gs/messaging-stomp-websocket/*

[12] RabbitMQ. (s.f.). Recuperado el Mayo de 2017, de *https://www.rabbitmq.com/getstarted.html*

[13] Spring. (s.f.). Recuperado el Mayo de 2017, de *https://spring.io/guides/gs/messaging-redis/*

[14] Cavano, J. P., & McCall, J. A. (1978, January). *A framework for the measurement of software quality. In ACM SIGMETRICS Performance Evaluation* Review (Vol. 7, No. 3-4, pp. 133-139). ACM.

[15] *Issco*. (s.f.). Recuperado el Mayo de 2017, de *http://www.issco.unige.ch/en/research/projects/ewg96/node13.html*

[16] iso25000. (s.f.). Recuperado el Mayo de 2017, de *http://iso25000.com*

[17] Fowler, M. (26 de Junio de 2005). *martinfowler*. Recuperado el Junio de 2017, de *https://martinfowler.com/bliki/InversionOfControl.html*

[18] Reenskaug, T. (2003). The Model-View-Controller (MVC) Its Past and Present Trygve Reenskaug, University of Oslo (trygver@ ifi. uio. no).

[19] patron observer

[20] Java. (18 de Mayo de 2014). Obtenido de *https://docs.oracle.com/javase/8/docs/api/*

[21] Spring. (2004). Recuperado el Abril de 2017, de *https://spring.io/*

[22] Johnson, R. (2005). *Professional Java development with the Spring Framework*. Indianapolis.

[23] Spring. (s.f.). Recuperado el Abril de 2017, de *https://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html*

[24] Maven. (2002). Recuperado el Abril de 2017, de *https://maven.apache.org/*

[25] Redis. (2013). Recuperado el Mayo de 2017, de *https://redis.io/*

[26] ZXing. (2007). Recuperado el Junio de 2017, de *https://github.com/zxing/zxing*

[27] András, T. (10 de Junio de 2015). Recuperado el Junio de 2017, de *https://github.com/andrastoth/WebCodeCamJS*

[28] Ayanoglu, E. (2015). Mastering rabbitmq. Place of publication not identified: Packt Publishing.

[29] RabbitMQ. (s.f.). Recuperado el Junio de 2017, de *https://www.rabbitmq.com/amqp-0-9-1-reference.html*

[30] Oracle. (s.f.). Oracle. Recuperado el Mayo de 2017, de *http://www.oracle.com/webfolder/technetwork/tutorials/obe/java/HomeWebsocket/WebsocketH*

[31] SockJS. (s.f.). SockJS GitHub. Recuperado el Mayo de 2017, de *http://sockjs.github.io/sockjs-protocol/sockjs-protocol-0.3.3.html*

[32] Popescu, A. (8 de Noviembre de 2016). *Geolocation API Specification 2nd Edition*. Recuperado el Junio de 2017, de *https://www.w3.org/TR/geolocation-API/*

[33] Tibbett, R. (30 de Mayo de 2017). *DeviceOrientation Event Specification*. Recuperado el Junio de 2017, de *https://www.w3.org/TR/orientation-event/*

[34] Burnett, D. C. (19 de Mayo de 2016). *Media Capture and Streams*. Recuperado el Junio de 2017, de *https://www.w3.org/TR/mediacapture-streams/*

[35] Kostiainen, A. (18 de Octubre de 2016). *Vibration API (Second Edition)*. Obtenido de *https://www.w3.org/TR/vibration/*

[36] Kostiainen, A. (22 de Mayo de 2017). *Ambient Light Sensor*. Obtenido de *https://www.w3.org/TR/ambient-light/*

[37] Kostiainen, A. (19 de Julio de 2016). *Proximity Sensor*. Recuperado el Junio de 2017, de *https://www.w3.org/TR/proximity/*

[38] Kostiainen, A. (7 de Julio de 2016). *Battery Status API*. Recuperado el Junio de 2017, de *https://www.w3.org/TR/battery-status/*

[39] Spring. (s.f.). *Redis*. Recuperado el Mayo de 2017, de *http://projects.spring.io/spring-data-redis/*

[40] Apache. (s.f.). *Tomcat*. Recuperado el Mayo de 2017, de *https://tomcat.apache.org/*

[41] Certbot. (s.f.). *Certbot*. Recuperado el Julio de 2017, de *https://certbot.eff.org/*

[42] Canuise. (s.f.). Canuise. Recuperado el Julio de 2017, de *http://caniuse.com/*

[43] Mozila Firefox. (s.f.). Mozila Firefox. Recuperado el Julio de 2017, de *https://developer.mozilla.org/en-US/docs/Web/API/*

[44] Panji, M. (20 de Noviembre de 2015). *HostPresto!* Recuperado el Julio de 2017, de *https://hostpresto.com/community/tutorials/how-to-install-and-configure-redis-on-ubuntu-14-04/*

[45] RabbitMQ. (s.f.). *Installing on Debian / Ubuntu*. Recuperado el Julio de 2017, de *https://www.rabbitmq.com/install-debian.html*

9 Anexo

9.1 Glosario

Glosario	
Slash and Pair	Aplicación web de comunicación entre dispositivos.
Spring	*Framework* para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java.
RabbitMQ	Software de negociación de mensajes, Implementa el estándar *Advanced Message Queuing Protocol (AMQP).*
Redis	Motor de base de datos en memoria, basado en el almacenamiento en tablas de *hashes* (clave/valor).
WebSocket	Canal de comunicación bidireccional y *full-duplex* sobre un único *socket* TCP.
HTML	*HyperText Markup Language*, hace referencia al lenguaje de marcado para la elaboración de páginas web.
CSS	*Cascading Style sheets* lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado.
JavaScript	Lenguaje de programación interpretado, dialecto del estándar *ECMAScript*. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.
jQuery	Biblioteca multiplataforma de *JavaScript*.
HTTP	*Hypertext Transfer Protocol*, protocolo de comunicación que permite las transferencias de información en la *World Wide Web*.
HTTPS	*Hypertext Transfer Protocol Secure*, protocolo de aplicación basado en el protocolo HTTP, destinado a la transferencia segura de datos de Hipertexto.
URL	*Uniform Resource Locator*, identificador de recursos uniforme
Script	Archivo de órdenes, archivo de procesamiento por lotes.
QR	Módulo para almacenar información en una matriz de puntos o en un código de barras bidimensional.
Localhost	Nombre reservado que tienen todas las computadoras, ratón o dispositivo independientemente de que disponga o no de una tarjeta de red *ethernet*.
Host	Ordenador o otro dispositivo conectado a una red que provee y utiliza servicios de ella.
Móvil/*smartphone*	Teléfono inteligente, teléfono móvil construido sobre una plataforma informática móvil, con mayor capacidad de almacenar datos y realizar actividades, semejante a la de una minicomputadora
Ordenador	Dispositivo en el que se reflejan las interacciones realizadas por el móvil. No ha de ser obligatoriamente un ordenador, puede ser por ejemplo una televisión, tablet o otro móvil.

Framework	Conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.
Hardware	Partes físicas tangibles de un sistema informático.
Software	Equipo lógico o soporte lógico de un sistema informático, que comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas.
API	*Application Programming Interface*, conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software.
Bróker de mensajería	Programa intermediario que traduce los mensajes de un sistema desde un lenguaje a otro.
Bluetooth	Especificación industrial para Redes Inalámbricas de Área Personal que posibilita la transmisión de voz y datos entre diferentes dispositivos.
Wifi	Mecanismo de conexión de dispositivos electrónicos de forma inalámbrica.
Router	Dispositivo que proporciona conectividad a nivel de red o nivel tres en el modelo OSI.
Maven	herramienta de software para la gestión y construcción de proyectos Java.
Testing	Investigaciones empíricas y técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto.
Emparejamiento	Acción de sincronizar dos dispositivos.
Interacción	Comunicación entre dos dispositivos, cuando uno realiza una acción se ve reflejada en el otro.
JSON	*JavaScript Object Notation*, formato de texto ligero para el intercambio de datos.
Sensor de móvil	Partes de un dispositivo móvil que recuperan información de las diferentes acciones que tiene un *smartphone*.

Tabla 21: Definiciones

9.2 Compatibilidad de sensores

Se ha realizado un estudio de la compatibilidad de los sensores con las API de *JavaScript*.[42] [43]

Geolocalización

Navegador	IE	Google Chrome(Android)	Firefox	Safari(IOS)
Versión	15	56	53	10.3

Tabla 22: Compatibilidad del sensor de geolocalización

Giroscopio

Navegador	IE	Google Chrome(Android)	Firefox	Safari(IOS)
Versión	15	56	53	10.3

Tabla 23: Compatibilidad del sensor de giroscopio

Cámara

Navegador	IE	Google Chrome(Android)	Firefox	Safari(IOS)
Versión	15	56	53	-

Tabla 24: Compatibilidad del sensor de cámara

Vibración

Navegador	IE	Google Chrome(Android)	Firefox	Safari(IOS)
Versión	-	56	53	-

Tabla 25: Compatibilidad del sensor de vibración

Sensor de luminosidad

Navegador	IE	Google Chrome(Android)	Firefox	Safari(IOS)
Versión	-	-	53	-

Tabla 26: Compatibilidad del sensor de luminosidad

Sensor de proximidad

Navegador	IE	Google Chrome(Android)	Firefox	Safari(IOS)
Versión	-	-	53	-

Tabla 27: Compatibilidad del sensor de proximidad

Nivel de batería

Navegador	IE	Google Chrome(Android)	Firefox	Safari(IOS)
Versión	-	56	53	-

Tabla 28: Compatibilidad del nivel de batería

9.3 Despliegue de la aplicación

En este apartado del anexo, se detallarán los comandos utilizados y respuestas del servidor.

Como primeros pasos, la instalación del software necesario para el correcto funcionamiento de la aplicación.

Listing 3: Comandos de instalación de *Redis*

```
root@zeus ~ # apt-get -s install redis-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer rec
m4 rlwrap sendmail-base sendmail-cf sensible-mda
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
redis-tools
The following NEW packages will be installed:
redis-server redis-tools
0 upgraded, 2 newly installed, 0 to remove and 244 not upgraded.
Inst redis-tools (2:2.8.4-2 Ubuntu:14.04/trusty [amd64])
Inst redis-server (2:2.8.4-2 Ubuntu:14.04/trusty [amd64])
Conf redis-tools (2:2.8.4-2 Ubuntu:14.04/trusty [amd64])
Conf redis-server (2:2.8.4-2 Ubuntu:14.04/trusty [amd64])

root@zeus ~ # service redis-server status
redis-server is running

root@zeus ~ # redis-cli
127.0.0.1:6379>
```

Con esto tenemos *Redis* instalado y se ha comprobado que se está ejecutando en el servidor.[44]

Listing 4: Comandos de instalación de *RabbitMQ*

```
apt-get install rabbitmq-server
service rabbitmq-server status
Status of node rabbit@zeus ...
[{pid,21477},
{running_applications,[{rabbit,"RabbitMQ","3.2.4"},
{mnesia,"MNESIA  CXC 138 12","4.11"},
{os_mon,"CPO  CXC 138 46","2.2.14"},
{xmerl,"XML parser","1.3.5"},
{sasl,"SASL  CXC 138 11","2.3.4"},
{stdlib,"ERTS  CXC 138 10","1.19.4"},
{kernel,"ERTS  CXC 138 10","2.16.4"}]},
```

```
{os,{unix,linux}},
{erlang_version,"Erlang R16B03 (erts -5.10.4) [source] [64-bit] [smp:8:8]
{memory,[{total,36169408},
{connection_procs,2704},
{queue_procs,5408},
{plugins,0},
{other_proc,13349224},
{mnesia,60720},
{mgmt_db,0},
{msg_index,24368},
{other_ets,760520},
{binary,7216},
{code,16522377},
{atom,594537},
{other_system,4842334}]},
{vm_memory_high_watermark,0.4},
{vm_memory_limit,13437853696},
{disk_free_limit,50000000},
{disk_free,1841409175552},
{file_descriptors,[{total_limit,924},
{total_used,3},
{sockets_limit,829},
{sockets_used,1}]},
{processes,[{limit,1048576},{used,129}]},
{run_queue,0},
{uptime,62}]
...done.

root@zeus /etc/rabbitmq # git --version
git version 1.9.1
root@zeus /etc/rabbitmq # java -version
java version "1.8.0_60"
Java(TM) SE Runtime Environment (build 1.8.0_60-b27)
Java HotSpot(TM) 64-Bit Server VM (build 25.60-b23, mixed mode)
root@zeus /etc/rabbitmq # mvn -v
Apache Maven 3.3.9 (bb52d8502b132ec0a5a3f4c09453c07478323dc5; 2015-11-10
Maven home: /usr/share/maven3
Java version: 1.8.0_60, vendor: Oracle Corporation
Java home: /usr/lib/jvm/java-8-oracle/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "3.19.0-26-generic", arch: "amd64", family: "
```

En este punto se ha instalado *RabbitMQ* y se ha comprobado que está en funcionamiento[45].

Una vez instalado todo el software necesario, se procede a la descarga del código fuente de la aplicación *Slash and Pair* y a su compilado.

Listing 5: Descarga de código fuente de *Slash and Pair*y compilación

```
rm -r app
rm master.zip
rm -r slash-and-pair-master

mkdir -p app/desktop
mkdir -p app/mobile

wget https://github.com/charlyhook/slash-and-pair/archive/master.zip
unzip master.zip

cd slash-and-pair-master
mvn package
cd ..

cp slash-and-pair-master/slash-and-pair-desktop/target/*.jar app/desktop
cp slash-and-pair-master/slash-and-pair-mobile/target/*.jar app/mobile
```

Se pasa a la configuración del servidor para la aplicación.

Listing 6: Configuración del servidor

```
<VirtualHost *:80>
ServerAdmin guillermo@theblackbox.io
ServerName slash-and-pair-d.theblackbox.io
ServerAlias slash-and-pair-d.theblackbox.io
ErrorLog /root/apache2/log/slash-and-pair-d.theblackbox.io/error.log
CustomLog /root/apache2/log/slash-and-pair-d.theblackbox.io/access.log co
ProxyPass / http://localhost:8082/ retry=5 timeout=300
ProxyPassReverse / http://localhost:8082/
ProxyRequests Off
ProxyPreserveHost On

<Proxy *>
Order deny,allow
Allow from all
</Proxy>
<Location />
</Location>
</VirtualHost>
```

La creación de los *scripts* de *start* y *stop* de la aplicación se han definido anteriormente en el apartado de despliegue.

Por último se instalan los certificados necesarios para el acceso por HTTPS a la aplicación mediante la herramienta *Certboot*

Listing 7: Instalacciónde HTTPS con *Certbot*

```
root@zeus ~/conf_apache2/sites-available # vim slash-and-pair.theblackbox
root@zeus ~/conf_apache2/sites-available # a2ensite slash-and-pair.thebla
Enabling site slash-and-pair.theblackbox.io.
To activate the new configuration, you need to run:
service apache2 reload
root@zeus ~/conf_apache2/sites-available # apachectl -t
Syntax OK
root@zeus ~/conf_apache2/sites-available # service apache2 reload
* Reloading web server apache2
```

Si se desea realizar la instalación de *Slash and Pair* en un ordenador corriente, se han de seguir los mismos pasos finalizando en el código de compilación, solo quedaría ejecutar el .jar creado para cada proyecto para poder hacer uso del software introduciendo el enlace *localhost:8082* ejecutar en local el ordenador y el enlace *localhost:8083* ejecutar en local el ordenador.