UNIVERSITAT DE BARCELONA

Treball final de grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

Slash and Pair Desktop: Seguridad en una aplicación web

Autor: Víctor Fernández Coderch

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

Barcelona, 21 de juny de 2017

Abstract

In the current context technology is evolving towards a point where all devices are interconnected. At the moment, the technology consumer has a wide range of tools at his disposal that allow him to use all his gadgets as if it were only one, examples of this are the applications for cars [1] and domotics [2]. This phenomenon is known as the Internet of Things or Web 3.0. The *Slash and Pair* project adds to this trend and aims to enable the user to use his smartphone as an additional peripheral of his desktop computer.

Slash and Pair (S&P) is a project with two very different modules; S&P Desktop and S&P Mobile. Throughout this document the first S&P Desktop module is treated, which is the project that the actions carried out in the smartphone have repercussion within a web page. Comprehensive analysis of system security is also included. The development of this project concludes with a functional system that allows the *pairing* and the sending of data from the Mobile project to the Desktop.

Resum

En el context actual la tecnologia està evolucionant cap a un punt en el qual tots els dispositius estiguin interconnectats. En aquest moment, el consumidor de tecnologia té a la seva disposició un ampli ventall d'eines que li permeten utilitzar tots els seus *gadgets* com si fos un de sol, exemples d'això són les aplicacions per a cotxes [1] i la domòtica [2]. Aquest fenomen es coneix com l'Internet de les coses *(Internet of Things)* o Web 3.0. El projecte *Slash and Pair* es suma a aquesta tendència i té com a objectiu que l'usuari sigui capaç d'utilitzar el seu *smartphone* com un perifèric més del seu ordinador de sobretaula.

Slash and Pair (S&P) és un projecte amb dos mòduls molt diferenciats; S&P Desktop i S&P Mobile. Al llarg d'aquest document es tracta el primer mòdul S&P Desktop, que és el projecte encarregat que les accions realitzades en el *smartphone* tinguin repercussió dins d'una pàgina web. També s'inclou un anàlisi exhaustiu de la seguretat del sistema. El desenvolupament d'aquest projecte conclou amb un sistema funcional que permet realitzar el *pairing* i l'enviament de dades des del projecte Mobile cap a Desktop.

Resumen

En el contexto actual la tecnología está evolucionando hacia un punto en el que todos los dispositivos estén interconectados. En este momento, el consumidor de tecnología tiene a su disposición un amplio abanico de herramientas que le permiten utilizar todos sus *gadgets* como si fuera uno solo, ejemplos de esto son las aplicaciones para coches [1] y la domótica [2]. Este fenómeno se conoce como el Internet de las cosas *(Internet of Things)* o Web 3.0. El proyecto *Slash and Pair* se suma a esta tendencia y tiene como objetivo que el usuario sea capaz de utilizar su *smartphone* como un periférico más de su ordenador de sobremesa.

Slash and Pair (S&P) es un proyecto con dos módulos muy diferenciadas; S&P Desktop y S&P Mobile. A lo largo de este documento se trata el primer módulo S&P Desktop, que es el proyecto encargado de que las acciones realizadas en el *smartphone* tengan repercusión dentro de una página web. También se incluye un análisis exhaustivo de la seguridad del sistema. El desarrollo de este proyecto concluye con un sistema funcional que permite realizar el *pairing* y el envío de datos desde el proyecto Mobile hacía Desktop.

Agradecimientos

Quiero agradecer a Guillermo Blasco la oportunidad que me ha dado ofreciéndome realizar este proyecto, ya que sin él nada de esto hubiera sido posible, por todas las horas de dedicación y su preocupación para que todo saliera bien.

También un especial agradecimiento a los tutores Guillem Palomar y Pablo Martínez, que han dedicado su tiempo para realizar el seguimiento y tutelando casi a diario los progresos que hemos ido realizando.

Por último, agradecer a todos mis familiares y amigos que me han apoyado y animado durante todos los momentos malos y los no tan malos, y en especial a Silvia por estar siempre ahí.

Gracias a todos.

Tabla de contenido

Lista de figuras

Lista de tablas

1 Introducción

1.1 Contexto

Para el desarrollo del proyecto *Slash and Pair* es interesante analizar el porcentaje de hogares que disponen de dispositivos que permiten a sus propietarios acceder a Internet a través de un navegador.

En la actualidad, el uso de ordenadores está ampliamente extendido en los hogares españoles. Según el INE [3] el porcentaje de hogares que disponen de un ordenador en el año 2014 es del 74,8%. Para el año 2015 este valor aumenta hasta el 75.9%. Así como el uso de los ordenadores está ampliamente extendido en los hogares españoles, también es interesante a cuánto asciende el porcentaje de los hogares que disponen de un *smartphone*. Para ello analizamos de nuevo la encuesta sobre el equipamiento de productos de tecnología de la información del INE [3]. Este documento determina que la disposición de teléfonos dentro de los hogares españoles es más alta que la de ordenadores. Se indica, que el 96,4% de los hogares tienen al menos un teléfono móvil en 2014 y este porcentaje aumenta en 2015 hasta el 96.7%. En la figura 1 se resumen todos los datos del INE analizados.

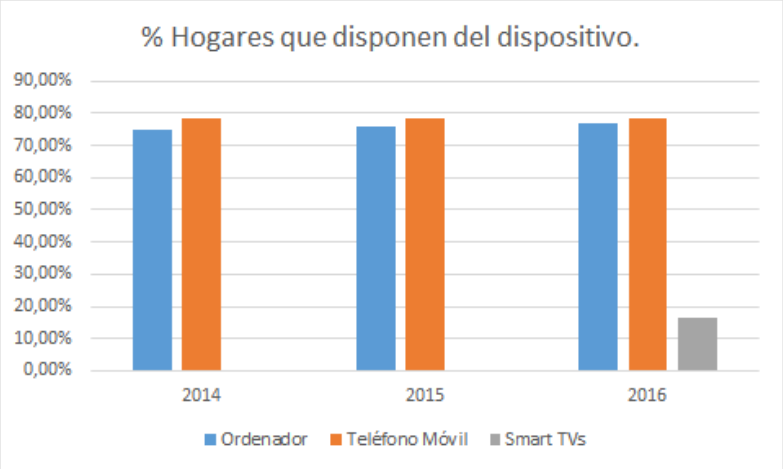

Figura 1: Gráfico resumen de los datos del INE.

Analizando una encuesta más reciente del INE [4], se nota el rápido crecimiento de otro dispositivo que permite la conexión a Internet dentro de los hogares españoles. Estos dispositivos son las SmartTVs. En la encuesta sobre equipamiento y uso de tecnologías de información y comunicación en los hogares de octubre de 2016, ya se incluyen las SmartTVs [5]. Estos modelos más modernos disponen de conexión de Internet y poseen la capacidad de instalar aplicaciones diseñadas específicamente para un televisor. Los televisores más arcaicos, que de un estado de fábrica no incorporan el sistema SmartTV se pueden convertir en un televisor SmartTV a través

de otros dispositivos externos como centros multimedia o reproductores Blu-Ray. Estos dispositivos, según el INE ocupan el 16,3% de los hogares.

Una vez visto el gran número de hogares españoles con capacidad para acceder a Internet a través de un dispositivo Smart o un ordenador, es importante analizar la forma en la que el usuario puede interactuar con estos dispositivos.

Las SmartTV disponen de un mando a distancia con el que navegar por todas las utilidades que disponen, así como los distintos menús. Para utilizar las funcionalidades como los navegadores, es necesario que se despliegue un teclado en la pantalla y el mando permite navegar entre los diferentes caracteres.

Por otro lado, están los ordenadores. Los ordenadores portátiles actualmente disponen de varios dispositivos para introducir datos. Los comunes en todos los modelos dada su necesidad, son el teclado y el touchpad. Un gran número de ellos incluyen cámara y micrófono, y muy pocos incluyen una pantalla táctil. En cuanto a los ordenadores de sobremesa, los dispositivos de entrada son muy similares, sustituyendo el touchpad por un ratón y añadiendo la necesidad de adquirir dispositivos externos para la captación de vídeo y sonido.

Finalmente, tenemos los *smartphones*, según las especificaciones descritas por GSMARENA [6], dentro de las características de un *smartphone* común como el IPhone 7, dispone de los siguientes sensores:

- Touch ID, que permite introducir la huella dactilar para identificar al usuario.
- Barómetro, permite calcular la presión atmosférica.
- Giroscopio de 3 ejes, permite detectar la orientación en los 3 ejes espaciales.
- Acelerómetro, permite detectar la velocidad a la que se desplaza.
- Sensor de proximidad, permite detectar la proximidad del usuario.
- Sensor de luz ambiente, este sensor detecta el brillo emitido por fuentes de luz.

Por último, se incluye una breve descripción de servicios pertenecientes a empresas, que utilizan el *smartphone* para añadir o mejorar funcionalidades de sus respectivas aplicaciones.

El primer ejemplo que hay en la actualidad es Spotify [7]. El servicio que ofrece se llama “Spotify Connect” y, un usuario que disponga de un dispositivo de reproducción con Spotify instalado, o con su versión a través del navegador, puede registrar su dispositivo *smartphone* a través de la aplicación nativa o web para utilizarla como mando a distancia para cambiar su lista de reproducción actual, viajar a través de los menús, o parar la reproducción. Como se cita en la página oficial, es necesario que ambos dispositivos estén conectados a la misma red wifi, lo que es una restricción. También requieren una suscripción premium algunos dispositivos como citan en la página oficial.

El siguiente ejemplo es una aplicación que utilizan muchos millones de personas. WhatsApp Web [8]. WhatsApp, es una aplicación móvil de mensajería que se instala en cualquier *smartphone* de forma nativa.

Una vez instalada la aplicación y con una cuenta activa, es posible utilizar el servicio web que ofrecen de una forma tan sencilla como escaneando un código QR que sincroniza la sesión del navegador a la cuenta de la aplicación móvil. WhatsApp en su página oficial, sólo requiere para su uso que se disponga de una conexión activa y segura.

El último ejemplo del cual se pueden extraer semejanzas con *Slash and Pair*, se llama Join.me [9]. Esta aplicación se puede instalar en un equipo de sobremesa y proporciona un código de acceso que se puede introducir en la aplicación instalada en el *smartphone*, de esta forma permite que el dispositivo Android muestre la pantalla del ordenador remoto.

Hay un número alto de aplicaciones que permiten la conexión entre un ordenador de sobremesa y un *smartphone* para usos de oficina, basadas en mostrar la pantalla del pc y el control remoto de este pc a través del *smartphone*.

1.2 Motivación

La motivación que me lleva a realizar un proyecto como *Slash and Pair* es, en un inicio, la posibilidad de conocer todo el proceso de desarrollo de software que hay detrás de una idea. Este período se inicia con la definición del proyecto de una forma superficial. Se pretende desarrollar: "un software que tiene que realizar una función específica", sin más especificaciones. Esta definición tan vaga requiere un buen análisis, una toma de decisiones, y una planificación lo suficientemente buenas como para llegar a un producto útil. Todo este proceso desde cero por primera vez, supone un reto.

Por otro lado, el proceso de desarrollo de *Slash and Pair* ha requerido la selección y aprendizaje de diversas tecnologías, entre ellas *frameworks* o gestores de proyectos que son herramientas muy comunes en el desarrollo de software.

1.3 Objetivos

En este apartado se describen los objetivos generales del proyecto *Slash and Pair* además de los específicos de *Slash and Pair Desktop*.

1.3.1 Generales

El objetivo principal de *Slash and Pair* es el de desarrollar un software que permita enviar los datos de los sensores de un *smartphone* dentro de un entorno web hacía otro navegador abierto en un ordenador, por lo tanto, los objetivos generales del proyecto *Slash and Pair*, requieren que tanto *Slash and Pair Mobile* como *Slash and Pair Desktop* trabajen de forma conjunta, creando un sistema distribuido, que

permita la interacción entre dos páginas web utilizando un *smartphone* como dispositivo de entrada en la página web correspondiente a *Slash and Pair Mobile* y la captura de dicha información, para posteriormente enviarla al proyecto *Slash and Pair Desktop* y que éste realice acciones en su página web.

1.3.2 Específicos

Los objetivos específicos del proyecto Desktop son los siguientes:

- Realizar la implementación de un servidor que despliegue una URL accesible desde cualquier navegador que permita al usuario:
  - Introducir la URL dada en el navegador y le muestre un código para realizar la sincronización entre dispositivos.
  - Una vez realizada la conexión le muestre al usuario una aplicación web que permita aprovechar algunas de las características que incorporan los *smartphones* para manejarla de una forma diferente.
- Realizar un análisis de la seguridad del sistema analizando las tecnologías utilizadas.

1.4 Estructura del documento

En este punto se resumen brevemente los apartados que incluye la memoria.

En el siguiente apartado, *"Especificaciones y Requisitos de Software"*, se define desde un punto de vista global como es el funcionamiento del software. Se nombrarán las tecnologías utilizadas sin entrar en detalle. Más adelante se profundizará en la descripción de las tecnologías externas utilizadas para el desarrollo del proyecto, se realizará un análisis funcional en cuanto a capacidades y necesidades que cubre el proyecto, para acabar, también se describirán las características de los usuarios y las restricciones que impone el software.

En el apartado que sigue a *"ERS"*, que es *"Metodología y planificación"* se describe todo el proceso de desarrollo del proyecto. Se especifican las fases en las que se estructura el desarrollo del proyecto, en las cuales se han ido realizando y completando las tareas, además también se especifica qué herramientas se han utilizado para el seguimiento del trabajo realizado y la metodología empleada para desarrollar el software.

Posteriormente nos encontramos el apartado de *"Análisis"*, donde se realiza toda la descripción del proyecto utilizando diagramas de casos de uso y las redacciones de estos casos, tanto generales como específicos. Este apartado de análisis incluye un estudio de la seguridad del proyecto, enfocándolo a las herramientas o protocolos empleados. Para cerrar este capítulo, se realizará un estudio enfocado al proyecto *Slash and Pair* utilizando para ello un análisis tipo DAFO [10].

DAFO son las siglas de, debilidades, amenazas, fortalezas y oportunidades. La tabla 1 resume cada una de sus siglas.

Factor	Notación	Descripción
Debilidad	D	Posición desfavorable del proyecto de carácter interno.
Amenaza	A	Situación desfavorable o que puede perjudicar al proyecto de carácter externo.
Fortaleza	F	Posición favorable del proyecto de carácter interno.
Oportunidad	O	Situación favorable que puede proporcionar el entorno al proyecto.

Tabla 1: Resumen siglas análisis DAFO.

La utilidad de esta herramienta es facilitar la toma de decisiones estratégicas, en función del análisis del entorno tanto externo como interno. El objetivo final es consolidar las fortalezas, y reducir las posibles amenazas.

Para concluir la memoria, se expondrá un ejemplo del funcionamiento, capacidades del software y los posibles futuros desarrollos o mantenimiento del mismo.

2 ERS (Especificaciones Requisitos de Software - IEEE 830-1998)

En el apartado anterior se ha resumido brevemente el concepto del proyecto y se ha contextualizado dentro del mundo actual dónde la tecnología va ganando terreno en los hogares. Además, se ha descrito como es la competencia y los objetivos a cubrir en este proyecto.

En esta sección se detalla la especificación de Requerimientos del sistema *Slash and Pair*. Esta especificación de requisitos pretende analizar cuáles pueden ser las necesidades de los usuarios para llegar a construir un software que cumpla esos requerimientos.

El objetivo de un ERS es producir un documento de especificación de requisitos en el que se describa lo que el futuro sistema debe hacer. El análisis de requisitos es una tarea importante, que determina los cimientos de la aplicación. Todos los requerimientos software definidos en este apartado deben aparecer en el producto final.

Para la redacción de este capítulo se han seguido las directrices del IEEE 830-1998 [11] [12].

2.1 Introducción

Se presentarán de forma organizada los requisitos indispensables para desarrollar un sistema web distribuido que permita comunicar a través de navegadores datos entre un *smartphone* y un ordenador.

2.1.1 Propósito

El propósito es dar a conocer las funcionalidades, características y condicionantes. Esta especificación va dirigida a usuarios finales del sistema.

2.2 Descripción general

Slash and Pair es una aplicación que permite realizar el envío de datos unidireccionalmente teniendo como dispositivo de entrada un *smartphone*, y como dispositivo de salida la pantalla de un ordenador portátil dentro de un entorno web. En este proyecto se enviarán los datos correspondientes al sensor del giroscopio y de la pantalla táctil, pero podría recogerse cualquier dato de cualquiera de los sensores descritos anteriormente que pueda contener un *smartphone*. Estos datos se estructurarán dentro de un objeto de tipo JSON que contendrá la información recogida por los sensores determinados en función del caso. La figura 2 muestra como es la jerarquía de la aplicación:

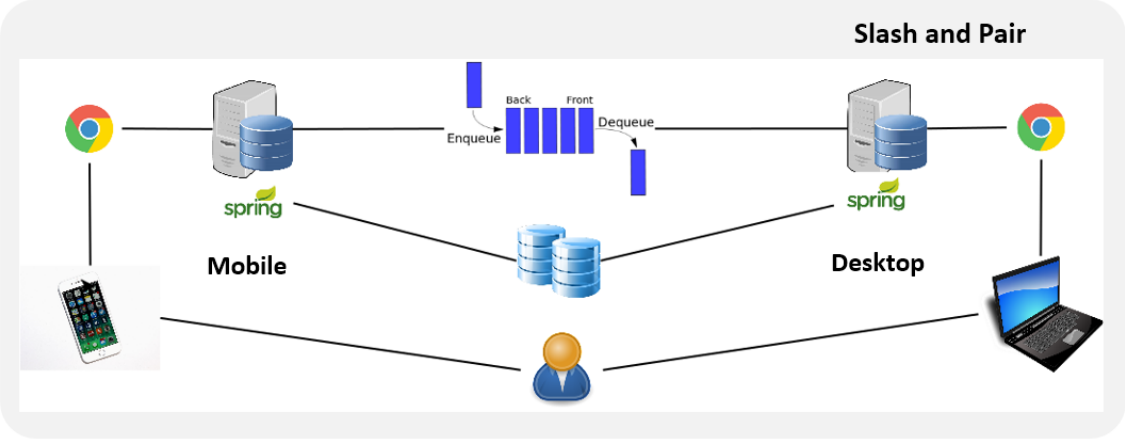

Figura 2: Jerarquía *Slash and Pair*.

El sistema *Slash and Pair* estará preparado para que sea usable por todo tipo de usuarios, sin conocimientos altos en informática. El sistema está preparado para que no requieran conocimientos de ningún tipo para su uso, cualquier persona que disponga de los dispositivos requeridos podrá utilizar el sistema sin problemas, ya que se ha priorizado la facilidad de uso en cuanto a la interfaz, por lo tanto, sólo requerirá habilidades básicas de manejo de un PC y de un *smartphone*.

Slash and Pair consta de dos páginas HTML estáticas distintas, Desktop y Mobile. Estas páginas son proporcionadas por dos servidores Spring distintos.

2.2.1 Perspectiva del producto

El sistema *Slash and Pair Desktop* será un producto diseñado para trabajar en entornos web, lo que permitirá su uso de una forma rápida y eficaz.

El correcto funcionamiento de *Slash and Pair*, depende del trabajo coordinado entre el sistema *Slash and Pair Mobile* y *Slash and Pair Desktop*. Ambos están diseñados para trabajar en un entorno web.

El sistema final requerirá el uso del hardware disponible en un *smartphone* medio, en concreto el sistema de giroscopios y la cámara. En caso de no disponer del giroscopio el software no se podrá utilizar.

La cámara se utilizará para facilitar el uso, ya que ésta sólo se utiliza para establecer la conexión Mobile-Desktop. El usuario podrá realizar la conexión alternativamente utilizando un código provisto por Desktop que tendrá que introducir manualmente a través de las interfície que se le mostrarán en Mobile para realizar la conexión.

2.2.2 Funciones del sistema

El sistema Desktop generará un código de enlace de tipo QR cuando reciba una petición a través del navegador.

En caso de que el usuario no disponga de la cámara o no quiera utilizarla, podrá acceder a un generador de un código alternativo que también le servirá para realizar la conexión con Mobile.

Una vez realizada la conexión, Desktop tratará los datos que reciba desde el proyecto Mobile en función de la aplicación que se haya desplegado dentro del entorno web. El usuario verá cambios en el navegador de Desktop en función de las acciones realizadas en Mobile.

2.2.3 Características de los usuarios

Los tipos de usuarios de *Slash and Pair* se describen por su relación con el sistema. Diferenciamos dos tipos.

El primero de los tipos de usuario será el no logado, que acabará de realizar la conexión a Desktop y no habrá enlazado la sesión con Mobile. Este tipo de usuario podrá convertirse en el usuario logado una vez haya puesto el código de enlace en Mobile.

El usuario logado, podrá aprovechar el potencial de *Slash and Pair Mobile* utilizando su *smartphone* para realizar acciones que afectarán a *Slash and Pair Desktop*.

2.2.4 Restricciones

Para utilizar el software es necesario disponer de dos dispositivos, un *smartphone* y un ordenador con conexión a Internet.

Podemos distinguir dos tipos de restricciones, de hardware y de software.

Las restricciones de hardware se corresponderán con dispositivos *smartphones* que no dispongan del sensor del giroscopio.

En cuanto al software, para dispositivos iOS la cámara no es funcional en *Slash and Pair*, por lo tanto, deberán usar la vía alternativa para la introducción del código de *pairing*.

Es necesario tener el software del navegador utilizado para acceder a Desktop y el sistema operativo actualizados a la última versión por razones de seguridad, ya que la información a la que se accede es sensible.

2.2.5 Suposiciones y dependencias

Se asume que los requisitos aquí descritos son estables. Para los equipos que vayan a ejecutar este software deben cumplir los requisitos antes indicados para garantizar una ejecución correcta del proyecto.

También se asume que el correcto funcionamiento del proyecto tiene lugar cuando Desktop y Mobile trabajan conjuntamente. Uno de los sistemas por sí sólo no tiene funcionalidad alguna, por lo tanto, Desktop es dependiente de Mobile y viceversa.

2.3 Requisitos específicos

En este apartado se detallan los requisitos de software, los elementos aquí detallados deben ser suficientes para describir los comportamientos visibles externamente por los usuarios.

2.3.1 Interfaces externas

Slash and Pair Desktop hace uso de las interfaces externas siguientes: una cola RabbitMQ, una base de datos Redis, WebSockets, y Maven. Estas interfaces son internas del servidor y el usuario no tiene visibilidad de ellas. El usuario de *Slash and Pair* necesitará disponer de 2 navegadores. Uno será necesario para abrir la página correspondiente al proyecto Desktop y el otro para abrir la página correspondiente al proyecto *Slash and Pair Mobile*.

No hay incompatibilidades entre diferentes sistemas operativos que utilicen el sistema ni versiones incompatibles del navegador, aunque se recomienda que se utilice desde versiones debidamente actualizadas.

2.3.2 Restricciones de diseño

Por la propia definición del proyecto, este software está limitado a las acciones que puede realizar un cliente a través de una página web, por lo tanto, el software permite realizar cualquier acción que esté permitida por un navegador web, pero no podría ejecutar funciones de sistema.

Todas las funciones de los *smartphones* para las que no haya API no se podrán utilizar para enviar información a Desktop, por tanto, requerimos de estas interfaces para aumentar los usos que se le pueden dar al sistema.

Por la forma en la que se gestionan los usuarios y la carga de datos que implican el uso de este sistema, el número de usuarios que pueden utilizar el sistema concurrentemente es más bajo que la de un servidor estándar, en la documentación de *Slash and Pair Mobile* se hace un análisis de esta característica.

Debido a la diferencia entre sistemas operativos iOS y Android, la seguridad del sistema flaquea debido a la necesidad de implementar una alternativa a la cámara para realizar el *pairing*. Esta alternativa es la del código de dígitos que, por número de posibilidades es menos segura que una definida dentro de un código QR que, debido a que el usuario no tiene que memorizarla ni escribirla, puede ser todo lo larga y difícil que el sistema pueda implementar.

2.3.3 Requisitos funcionales

Los requisitos funcionales son estrictamente aquellas funciones o acciones que deberá cumplir el sistema una vez esté desarrollado. En este caso se definirán en función del usuario, ya que cada usuario tiene un propósito distinto, para el usuario no autenticado se requiere poder realizar el *pairing* de una forma sencilla y segura,

para el usuario ya Pareado sus requisitos son distintos. Ya que ha de poder utilizar los sensores y visualizar las acciones en Desktop.

En este apartado solo se definirán los requisitos funcionales requeridos por el proyecto *Slash and Pair Desktop*.

Requisitos funcionales de usuario se describen en la tabla 2:

Sincronización:	
REQ01	Cuando un sistema realiza una petición GET a través del navegador el sistema Desktop ha de ser capaz de crear y registrar un ID de sesión personal para el usuario en cuestión y ha de generar un código QR que le representará sólo a él para que pueda ser enlazado con la conexión de otro dispositivo *smartphone*.
REQ02	El usuario ha de poder realizar la conexión con Desktop a pesar de que su *smartphone* no reconozca el código a través de la cámara. Para ello Desktop ha de ser capaz de generar un código numérico que pueda introducir a través del sistema *Slash and Pair Mobile* .
REQ03	El sistema Desktop, al reconocer la petición de autenticación de sesión del usuario ha de ser capaz de mostrar una nueva aplicación.

Tabla 2: Requisitos funcionales para la sincronización.

Requisitos funcionales usuario autenticado se describen en la tabla 3:

Interacción:	
REQ04	*Slash and Pair* implementa una aplicación web la cual se mostrará solamente para un usuario que se haya autenticado. Esta aplicación permite ver los datos enviados a través de Mobile.
REQ05	El usuario ha de poder ver el resultado de sus interacciones desde su *smartphone* hasta su respectivo Desktop.

Tabla 3: Requisitos funcionales para la interacción.

2.3.4 Requisitos de rendimiento

Los requisitos definidos que debe cumplir el proyecto son los siguientes:

- El proyecto debe soportar concurrentemente varios usuarios a la vez.
- La conexión no debe cortarse si el usuario sigue utilizando la aplicación.
- Debe ser viable que con mala conexión se pueda utilizar.
- Se tiene que poder utilizar en la mayor cantidad de dispositivos posible.
- La interfaz debe ser usable y ser intuitiva.

3 Metodología y planificación

En este apartado se describe como se ha llevado a cabo todo el proceso de desarrollo de software.

Inicialmente se definen las herramientas utilizadas para la gestión del equipo y el tipo de proceso desempeñado. Para concluir el apartado se profundiza en cada una de las iteraciones desempeñadas con los resultados y los objetivos a cumplir para la siguiente iteración. Finalmente se mostrará un diagrama de Grannt con el esfuerzo en días que ha requerido cada tarea.

A continuación, se define el tipo de metodología.

3.1 Tipo de metodología

Para estructurar, planificar, controlar y medir el proceso de desarrollo del software se ha utilizado un método enfocado estrictamente al desarrollo de software. Este método utilizado se llama desarrollo iterativo incremental [13], basado en planificar el proyecto en diversos bloques temporales llamados iteraciones. Las iteraciones y todo el desarrollo vienen acompañado de reuniones. Al inicio del proyecto las reuniones son más largas que a medida que se va avanzando, ya que las primeras definiciones del proyecto requieren un mejor análisis.

Todas las iteraciones vienen acompañadas de una reunión inicial, entre todo el equipo donde se sigue el patrón que se explica en este apartado.

Al comenzar el proyecto, hay una fase inicial que solo tiene lugar una única vez y es en la que se definen las líneas generales en el proyecto y se detalla más adelante. Una vez realizada esta fase inicial comienzan las iteraciones.

Las iteraciones descritas en la figura 3 se caracterizan por una serie de fases [14]:

Figura 3: Pasos dentro de la iteración.

Las iteraciones comienzan con la fase de determinar objetivos. Es importante

definir qué objetivos específicos se han de cumplir para dar por finalizada una iteración. Deben ser objetivos alcanzables a corto plazo, concretos, tangibles y deben aportar algún valor al proyecto.

La siguiente fase es de planificación. Una vez se han definido los objetivos, se planifica a lo largo del tiempo en función de la duración de la iteración, para adaptar las horas de trabajo al disponible de los desarrolladores. También es importante realizar un análisis de riesgos y prever que posibles dificultades puede suponer realizar el desarrollo en cuestión.

Una vez definidos todos los puntos anteriores, se inicia la fase de desarrollo, en la que se implementan y se desarrollan todas las tareas definidas en el inicio de la iteración.

Cuando la iteración llega a su final, el equipo se reúne y se evalúan las tareas realizadas, además de las dificultades superadas y se analizan los factores determinantes que han podido generar alguna adversidad de cara a mejorar el proceso para las próximas iteraciones y al finalizar la iteración, el ciclo vuelve a comenzar.

Cada iteración se puede entender como un pequeño proyecto, que proporciona un resultado usable sobre el producto final. De esta manera cada iteración se obtiene una pieza fundamental necesaria para el funcionamiento del proyecto final. Cada iteración requiere realizar todas las tareas designadas y realizarse en el tiempo estimado para que al final del plazo no quede nada crítico por realizar, permiten evolucionar el producto, priorizando los objetivos y los requisitos en función del valor que aportan.

Slash and Pair Desktop ha seguido esta metodología por las ventajas que se listan a continuación:

- Permite gestionar las expectativas de manera regular, lo que supone la capacidad de tomar decisiones trascendentales en cada iteración.
- Buena gestión de los cambios a corto plazo.
- El desarrollo se inicia con los requisitos de alto nivel marcados.
- La finalización de cada iteración permite tomar decisiones. A su vez, una iteración permite conocer el progreso total del proyecto.
- Los objetivos a corto plazo mejoran la productividad.

Para acompañar esta metodología se ha utilizado la herramienta Trello [15]. Esta aplicación ofrece un tablero en el que se pueden incorporar tarjetas con las tareas a realizar. El tablero se ha organizado en listas, en función del estado de las tareas, por realizar, trabajo en progreso, trabajo finalizado y archivadas.

En cada inicio de iteración se definen las actividades a realizar en forma de tarjeta y se colocan en la lista de "por realizar". Dentro de una iteración, cuando se inicia una tarea se coloca su respectiva tarjeta en la lista correspondiente a "trabajo a realizar". Al acabar la iteración todas las tareas definidas como por realizar, deben

colocarse en la lista de "trabajo finalizado". Después de la reunión del final de iteración se valida el trabajo realizado pasándolo a la lista de "archivadas".

3.2 Fases del proyecto

Se pueden distinguir tres fases diferenciadas del proceso de desarrollo de *Slash and Pair Desktop*.

3.2.1 Toma de decisiones

La primera de ellas es previa a la programación. Es la fase de definición del proyecto. En esta fase se expone la idea del proyecto: "Se requiere un software que permita enviar los datos de los sensores de un *smartphone* dentro de un entorno web hacia un ordenador".

Una vez planteado el proyecto a nivel funcional, es preciso desarrollar un storyboard provisional que incluya lo que se espera que el software realice en función de un flujo de pasos, en los que el usuario realice interacciones por la aplicación.

Seguidamente se definen las tecnologías a utilizar. Estas tecnologías deben tener la funcionalidad requerida necesaria para hacer funcionar el software.

Al finalizar la planificación, se tomó la decisión de dividir el proyecto en 2 subproyectos, Desktop y Mobile, cada uno siendo una aplicación ejecutable única e independiente, a pesar de que sea necesaria la comunicación entre ambas.

3.2.2 Desarrollo

El desarrollo se inicia tras haber definido todos los objetivos y especificaciones que el proyecto ha de cumplir. Cuando todo esto ha sucedido, se comienza con la primera iteración.

La primera iteración requería crear un proyecto Java, basado en Spring. El primer objetivo en cuanto a funcionalidad es realizar un proyecto que permita, a través de WebSockets, enviar datos desde el servidor de Desktop al HTML estático que se carga cuando un usuario accede a la URL. Para realizar la conexión con WebSockets, Spring proporciona un paquete tutorial con toda la información y dependencias necesarias para ello [16]. Este se utilizó en primera instancia, modificándolo para ceñirse al requisito concreto.

La primera versión del proyecto Desktop enviaba un dato cuando un temporizador localizado en el servidor lanzaba un evento. Entonces, el JavaScript se encargaba de parsear los datos recibidos a través del WebSocket y los imprimía en formato texto, que, de esta forma, quedaban visibles. Para la primera versión del proyecto, Desktop no cuenta con ninguna relación con su homólogo Mobile. Esta decisión de diseño en próximas iteraciones se modifica.

Una vez finalizada la primera iteración, el objetivo principal era diferenciar qué usuario se conecta con que id y como el servidor, puede reconocerlo como único

para diferenciarlo de los demás usuarios. Para solucionar esta problemática, Spring dispone de un paquete llamado Spring Security [17]. Este paquete permite, de una forma sencilla asignar un identificador de sesión de una manera sencilla. La primera versión del id de sesión consistía en, a través de la petición al servidor, autenticar al usuario con funciones propias de Spring Security.

Dado que, el proyecto ya estaba habilitado para tener varios usuarios de forma concurrentes, surge la necesidad de implementar un sistema de logging para almacenar toda la información proporcionada por el servidor. Esto facilita el desarrollo, por lo tanto, el esfuerzo de la tercera iteración se centró principalmente en utilizar herramientas para logar toda la información que fuera necesaria.

Otro punto a tratar era la distribución de los directorios del proyecto. Dado que los proyectos Desktop y Mobile tienen dependencias comunes, se considera la idea de fusionarlos en una única estructura. La herramienta utilizada para crear la estructura del proyecto, se llama Maven y es un gestor y constructor de proyectos Java [18].

Maven permite crear un proyecto padre, donde se pueden definir las dependencias y configuraciones comunes entre Desktop y Mobile. Esta herramienta es muy útil para que los proyectos compartan versiones y no sufran incompatibilidades a lo largo del desarrollo por utilizar diferentes versiones de herramientas software.

En este momento, también había que tener en cuenta para el desarrollo, es definir el lugar en el cual vamos a guardar las informaciones de los identificadores de usuario. Por lo tanto, en esta iteración se implementó el cliente de Redis en el proyecto [19].

Junto con la implementación de Redis, se define un código por el cual el id de usuario se guarda a través de una clave numérica de cuatro dígitos. Esta clave es la que utilizará el proyecto Mobile para buscar la sesión de usuario.

Para acabar la iteración, se implementa el lugar de intercambio de datos entre proyectos. Se escoge RabbitMQ como intermediario, situándolo entre Desktop y Mobile y se implementa el cliente para Desktop, en resumen, Desktop ya puede seleccionar un usuario, buscarlo en la base de datos y recibir datos de la cola de Rabbit.

En el momento en el que las tecnologias necesarias para poder cumplir los requisitos están integradas en la aplicación, hay que desarrollar las funciones que van a coordinar toda la funcionalidad para que finalmente, Mobile, acabe mandando un dato a Desktop y que éste lo reciba y visualice.

Para cumplir este objetivo, se define la estructura de datos común para los dos proyectos. Esta estructura seleccionada es el JSON y todos los datos enviados se envían en este formato.

Otra decisión que se toma en esta iteración es la de crear dos colas para Rabbit. Una cola de sincronización disponible para cualquier usuario sin autenticar, y otra cola de envío de datos únicamente para usuarios autenticados.

En esta iteración también se detecta que, por defecto, los datos enviados a través de Rabbit se envían a todos los usuarios que utilizan el sistema de forma concurrente,

por lo tanto, se aplica una solucion, que pasa por modificar la petición HTTP para que se incluya en las cabeceras la información del usuario al que va dirigida la información.

La ultima tarea de esta iteración que se define, es la de realizar en el proyecto Desktop un HTML que permita ver los datos que se están enviando a través de Mobile. Por tanto, se añade un HTML que se modifica dinámicamente con los datos enviados a través de Mobile.

Para la siguiente iteración, se comienza la redacción de la memoria además, dado que ya se envían los datos se propuso realizar unas pruebas exhaustivas del sistema, y con ello se detectan varios problemas de funcionamiento del envío de datos. En esta quinta iteración se analizan los posibles errores del sistema y se hace un esfuerzo en ubicar el origen del problema a través de la utilización de trazas y analizando el funcionamiento del sistema.

Al final de la iteración se identifica el problema. Este problema proviene de la conexión a través de redes públicas, por lo tanto, mientras el despliegue de la aplicación se haga de manera local no puede solventarse.

En la sexta iteración, como medida de seguridad se propone implementar un código QR para realizar la autenticación del usuario sustituyendo el anterior código de cuatro dígitos.

En esta iteración, se integra en el servidor Desktop la librería ZXING [20], que permite generar códigos QR utilizando cadenas de caracteres.

Otro objetivo a alcanzar en la iteración es el de utilizar el protocolo HTTPS en lugar del HTTP dado que, el uso de la cámara a través de la aplicación de Mobile requiere una seguridad extra. Para esto se genera un certificado SSL y se configura el proyecto de manera que utilice este protocolo y no permita utilizar el HTTP.

Al finalizar esta iteración, Desktop genera un código de conexión de 32 caracteres alfanuméricos, una solución más segura que un simple código de 4 dígitos y envía este código QR al HTML para que el usuario pueda escanearlo a través de Mobile, todo esto a través de HTTPS.

Para la última iteración, se selecciona una aplicación integrable para un entorno web que permita mostrar el potencial de *Slash and Pair*. Para ello se escoge un juego basado en JavaScript y WebGL que permite utilizar la información de los sensores del Mobile para realizar acciones que tengan un efecto visible en Desktop.

Con la implementación de esta aplicación se finaliza el desarrollo.

3.2.3 Diagrama de Gantt

En la figura 4 se puede ver como han sido repartidas las tareas a lo largo del tiempo para *Slash and Pair*.

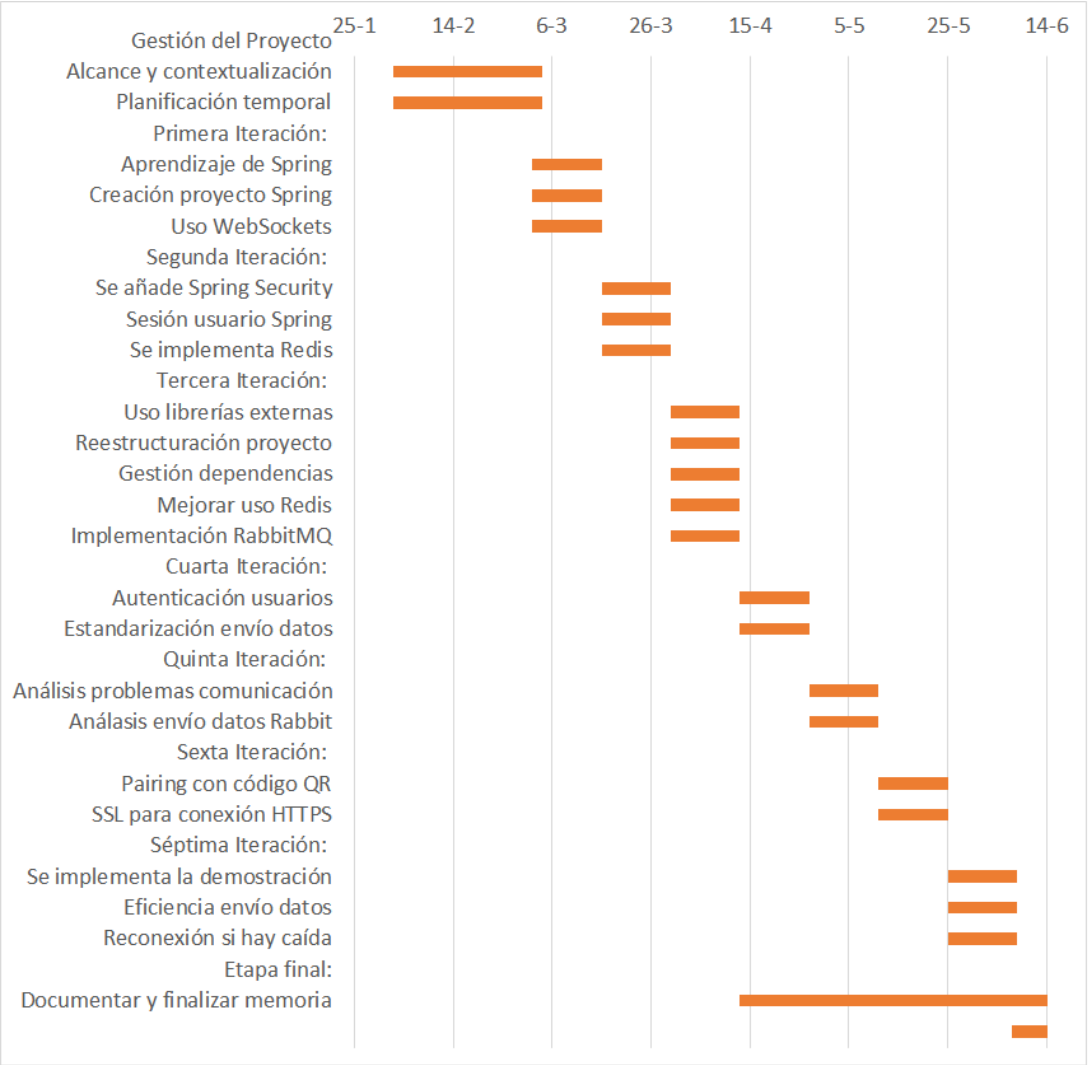

Figura 4: Diagrama de Gantt.

3.2.4 Despliegue aplicación

Una vez se ha finalizado la aplicación, se escoge un entorno Linux para desplegar el servidor. En esta fase se toman otras decisiones como utilizar Apache para generar un *host* virtual además de adaptar el servidor a las necesidades del proyecto. Todo ello está detallado en el apartado de diseño, *"Despliegue aplicación"*.

4 Análisis

En el anterior apartado se ha definido el proceso de creación del software utilizado para crear *Slash and Pair* y se han especificado las iteraciones realizadas para conseguir un proyecto de software que cumple los requisitos descritos al inicio del proceso.

En este apartado se mostrarán los casos de uso la aplicación, tanto a nivel de usuario en el entorno web *Slash and Pair* como concretando los casos de uso de la aplicación específica *Slash and Pair Desktop*.

4.1 Diagrama de casos de uso generales

El diagrama de casos de uso general describe las interacciones que puede realizar el usuario con el sistema, como vemos en la figura 5, el usuario puede realizar dos acciones, autenticarse y realizar interacción a través de Mobile viendo los resultados en Desktop, como se describe a continuación.

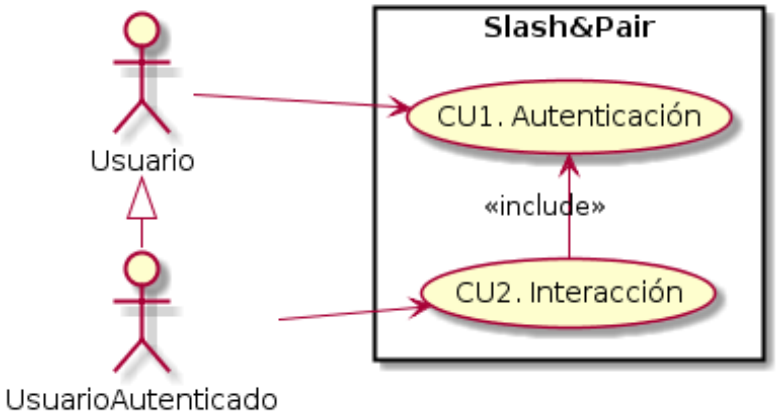

Figura 5: Diagrama de casos de uso de la aplicación.

4.2 Descripciones textuales generales

En este apartado se describe como es el flujo de interacciones entre el actor y el sistema.

Se define una tabla para cada caso de uso del diagrama, la primera de ellas es la tabla 4 que especifica el caso de uso autenticación.

CU1 Autenticación.		
Actor: Usuario.		
Precondición	El usuario dispone de dos dispositivos con conexión a Internet, un *smartphone* y un ordenador. El usuario ha abierto una instancia del navegador en el ordenador accediendo a la URL correspondiente al entorno de Desktop. El usuario ha abierto una instancia del navegador en su *smartphone* accediendo a la URL correspondiente al entorno de Mobile. El usuario no ha realizado el *pairing* anteriormente.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario se conecte por primera vez a *Slash and Pair Desktop* y no haya realizado el *pairing* anteriormente.	
Secuencia normal	Paso:	Acción:
	1	El usuario visualiza el código QR en la pantalla de Desktop.
	2	El usuario visualiza en el *smartphone* un recuadro donde se mostrarán las imágenes captadas por su cámara.
	3	El usuario escanea el código QR con su dispositivo *smartphone* enfocando la cámara hacia la página web Desktop.
	4	El usuario visualiza que la cámara le ha tomado una captura del código cuando se reconozca.
Postcondición	Las dos sesiones web del usuario están sincronizadas para comenzar la interacción.	
Excepciones	Paso:	Acción:
	2	El usuario detecta que la cámara no se enciende cuando se ha conectado a Mobile
		El usuario va a la página correspondiente a Desktop y clica en la opción "Muestra código en dígitos".
		El usuario introduce el código en el recuadro indicado para ello.
		El usuario pulsa el botón sincronización.
	3	Al usuario no se le reconoce el código QR regenerado. Recarga la página para generar un nuevo código QR.
	3	El usuario introduce mal el código numérico en Mobile. Visualizará de nuevo la página para introducir el código.
Comentarios	El código QR generado caducará a la que pase un plazo superior a un determinado tiempo, por tanto, si deja el navegador sin realizar el *pairing* debería refrescar el navegador para regenerar su código.	

Tabla 4: Caso de uso general Autenticación

En la tabla 5 se especifica el caso de uso interacción.

CU2 Interacción.		
Actor: Usuario.		
Precondición	El usuario ha escaneado correctamente el código QR o ha introducido el código que se le ofrece para introducir de forma manual correctamente. El usuario mantiene las sesiones abiertas simultáneamente tanto del navegador de Desktop como de Mobile. El usuario no pierde la conexión de Internet, tampoco cierra ninguno de los dos navegadores.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario haya realizado el *pairing* correctamente.	
Secuencia normal	Paso:	Acción:
	1	El usuario visualiza un aviso que le informa de que se ha realizado correctamente el *pairing*
	2	El usuario acepta el aviso.
	3	El usuario visualiza una nueva página en Desktop, que corresponderá con una aplicación web.
	4	El usuario visualiza una nueva página en Mobile, donde visualizará las instrucciones necesarias para utilizar el sistema.
	5	El usuario sigue las instrucciones, por ejemplo, gira el teléfono 90º a la derecha.
	6	El usuario visualiza el efecto del giro del *smartphone* dentro de la página Desktop.
Postcondición	El usuario puede utilizar la aplicación web *Slash and Pair* hasta que haya cumplido su objetivo.	
Excepciones	Paso:	Acción:
	1	Al usuario no se le muestran correctamente las nuevas páginas web. En este caso debería volver al caso de uso 1.
	5	El dispositivo *smartphone* del usuario no permite utilizar el giroscopio. En este caso no podrá disfrutar del software *Slash and Pair*.
	6	El usuario tiene mucha latencia de conexión, debido a la cantidad de peticiones que generan los movimientos de los sensores no podrá utilizar correctamente el sistema.
Comentarios	El usuario ya ha finalizado el proceso de sincronización y podrá utilizar *Slash and Pair* hasta que cierre la aplicación.	

Tabla 5: Caso de uso general Interacción

4.3 Diagrama de casos de uso específicos

Tras haber conocido el diagrama de casos de uso genéricos de la aplicación, profundizaremos en los casos de uso específicos de la aplicación Desktop, esto incluye una descripción a nivel más técnico de las acciones que realiza el servidor.

4.3.1 Diagrama específico autenticación

Analizaremos los casos de uso correspondientes a la autenticación como se puede visualizar en la figura 6.

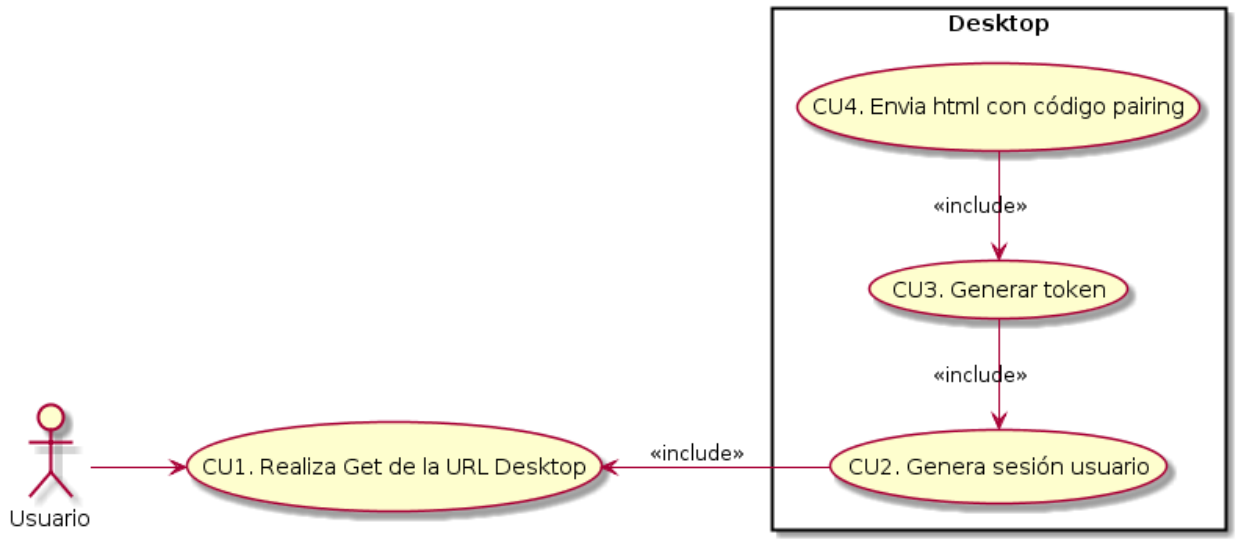

Figura 6: Caso de uso autenticación.

4.3.2 Descripciones textuales especificas caso autenticación

En este apartado se redactarán las casuísticas correspondientes a los casos de uso vistos para la autenticación. En la tabla 6 se describe el caso de uso de abrir la página web correspondiente a Desktop.

CU1 Realiza Get de la URL Desktop.		
Actor: Usuario.		
Precondición	El cliente no ha realizado el *pairing* anteriormente.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario se conecte por primera vez a *Slash and Pair Desktop* y no haya realizado el *pairing* anteriormente.	
Secuencia normal	Paso:	Acción:
	1	El usuario introduce en la URL en el navegador.
	2	El servidor recibe la petición.
Postcondición	El usuario ahora dispone de un código de sesión propio generado por Spring.	
Excepciones	Paso:	Acción:
	1	El usuario detecta que la cámara no se enciende cuando se ha conectado a Mobile
Comentarios	El servidor tiene que realizar los siguientes casos de uso para generar la página que visualizará.	

Tabla 6: Caso de uso específico realizar petición GET a Desktop

En la tabla 7 se describe el caso de uso de generar sesión de usuario.

CU2 Genera sesión usuario.		
Actor: Usuario.		
Precondición	El usuario ha introducido la URL Desktop correctamente en el navegador y el servidor recibe la petición. El usuario no dispone del código de sesión generado.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando se recibe la petición por parte del usuario de registrarlo en el sistema.	
Secuencia normal	Paso	Acción
	1	El sistema comprueba que el usuario no disponga de código de sesión generado. En este caso se avanza al CU3.
Postcondición	El usuario estará preparado para generar su propio *token* de autenticación para realizar el *pairing*.	
Excepciones	Paso	Acción
	1	El servidor busca si el id de sesión ya contiene un *token* generado correspondiente al usuario que ha realizado la petición dentro de la base de datos. En este caso se devuelve y avanza al CU4.
Comentarios	El flujo normal de este caso de uso, es que el usuario no tenga el *token* generado correspondiente a su id de sesión y se le genere uno nuevo en el siguiente caso de uso.	

Tabla 7: Caso de uso específico generar sesión usuario.

En la tabla 8 se especifica el caso de uso correspondiente a la generación del *token*:

CU3 Generar *token*		
Actor: Usuario.		
Precondición	El usuario ha abierto en el navegador por la página de Desktop y no tiene un *token* que corresponda a su id de sesión.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario lance la petición, Spring le haya generado su id de sesión automático y *Slash and Pair Desktop* haya autenticado al usuario.	
Secuencia normal	Paso	Acción
	1	El sistema genera una cadena de caracteres aleatorios que servirán como id de usuario generado por *Slash and Pair*
	2	Se genera un código que servirá para realizar la autenticación. Este código también será una cadena de caracteres aleatorios.
	3	Se crea una estructura de datos, siguiendo la estructura de los datos entendible para la base de datos, donde añadiremos lo generado anteriormente.
Postcondición	El sistema tiene los datos necesarios para que el usuario se pueda autenticar a través de Mobile.	
Excepciones	Paso	Acción
	1	En caso de que el usuario no pueda utilizar la cámara, el código generado será un entero de una longitud entre 4 y 6 cifras.
Comentarios	El siguiente paso es el último antes de que se le envíe al usuario la página web que visualizará para poder realizar la autenticación.	

Tabla 8: Caso de uso específico generar *token*.

En la tabla 9 se define el caso de uso de envíar hacía el navegador del usuario el código para la realización del *pairing*:

CU4 Envía HTML con código *pairing*.		
Actor: Usuario.		
Precondición	El servidor ha generado todos los datos necesarios para realizar el *pairing* de sesiones entre Mobile y Desktop.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando ya tenga los datos necesarios para mostrarle al usuario el código de autenticación.	
Secuencia normal	Paso	Acción
	1	Se monta el HTML por parte del Servidor y se envía al usuario.
	2	El usuario visualiza el HTML correspondiente a su código de *pairing*.
Postcondición	El usuario está preparado para realizar la autenticación a través de Mobile.	
Excepciones	Paso	Acción
	1	El *token* no se genera correctamente, por lo tanto, el usuario deberá volver al CU1.
Comentarios	El sistema está preparado para realizar la interacción.	

Tabla 9: Caso de uso envío del código para realizar el *pairing*.

4.3.3 Diagrama específico interacción

En la figura 7 se analizan las funciones correspondientes al caso de uso interacción.

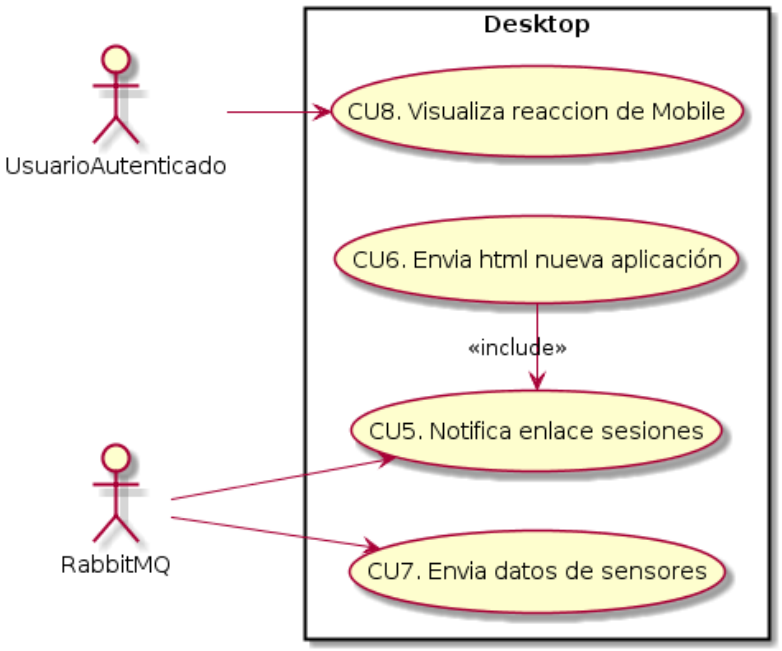

Figura 7: Caso de uso interacción.

4.3.4 Descripciones textuales especificas caso interacción

En este apartado se redactarán las casuísticas correspondientes a los casos de uso vistos para la interacción. En la tabla 10 se detalla el caso de uso de notificación de enlace de sesiones.

<table>
<tr><td colspan="3">CU5 Notifica enlace sesiones.</td></tr>
<tr><td colspan="3">Actor: RabbitMQ.</td></tr>
<tr><td>Precondición</td><td colspan="2">El servidor Desktop ha recibido un evento que proviene desde la aplicación Mobile, esto implica que se ha comprobado que el código introducido es correcto y se enlazan las sesiones, dando al usuario por autenticado. Este evento pasa a través de RabbitMQ, que es el que finalmente notifica a Desktop.</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario haya introducido el código a través de la aplicación Mobile.</td></tr>
<tr><td rowspan="2">Secuencia normal</td><td>Paso</td><td>Acción</td></tr>
<tr><td>1</td><td>El servidor muestra al usuario que se ha realizado correctamente el pairing de sesiones.</td></tr>
<tr><td>Postcondición</td><td colspan="2">El usuario conoce que el proceso ha ido correctamente</td></tr>
<tr><td rowspan="2">Excepciones</td><td>Paso</td><td>Acción</td></tr>
<tr><td>1</td><td>Cualquier excepción en este paso deriva en realizar todo el proceso de autenticación descrita en los casos de uso específicos de autenticación.</td></tr>
<tr><td>Comentarios</td><td colspan="2">El usuario tiene todo el entorno configurado para comenzar a utilizar Slash and Pair.</td></tr>
</table>

Tabla 10: Caso de notificar enlace de sesiones.

En la tabla 11 se especifica el caso de uso de envíar HTML con la nueva aplicación.

CU6 Envía HTML nueva aplicación.		
Actor: Usuario.		
Precondición	El servidor ha autenticado al usuario.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando ya tenga al usuario autenticado y detecte el evento lanzado por RabbitMQ.	
Secuencia normal	Paso	Acción
	1	El usuario visualiza en la pantalla la aplicación web configurada para utilizar *Slash and Pair*.
Postcondición	El usuario puede comenzar a realizar acciones en Mobile las cuales afectarán a Desktop.	
Excepciones	Paso	Acción
	1	Las acciones previamente definidas para la aplicación configurada, pueden no ser utilizables por el tipo de dispositivo *smartphone* utilizado por el usuario, por lo que no podrá utilizar el sistema *Slash and Pair*.
Comentarios	El usuario ha finalizado el proceso de autenticado.	

Tabla 11: Caso de uso específico envío HTML con aplicación

En la tabla 12 se especifica el caso de uso de envío de datos de sensores a la aplicación Desktop.

CU7 Envía datos de sensores.		
Actor: RabbitMQ.		
Precondición	El servidor ha autenticado al usuario.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando ya tenga al usuario autenticado y detecte el evento lanzado por RabbitMQ.	
Secuencia normal	Paso	Acción
	1	EEl usuario realiza una acción en Mobile, que RabbitMQ lanza al server Desktop.
Postcondición	El usuario ve una modificación en su pantalla Desktop, consecuente con la acción realizada en Mobile.	
Excepciones	Paso	Acción
	1	Las acciones previamente definidas para la aplicación configurada, pueden no ser utilizables por el tipo de dispositivo *smartphone* utilizado por el usuario, por lo que no podrá utilizar el sistema *Slash and Pair*.
Comentarios	Si el usuario dispone de una buena conexión a Internet y todos los dispositivos funcionan correctamente, debe notar la acción realizada inmediatamente en Desktop.	

Tabla 12: Caso de uso específico envío datos de sensores

En la tabla 13 se especifica el caso de uso de visualizar la reacción de la acción

realizada en Mobile en la pantalla de Desktop.

CU8 Visualiza reacción de Mobile.		
Actor: Usuario.		
Precondición	El servidor ha autenticado al usuario y ha realizado una acción a través de Mobile.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando ya tenga al usuario autenticado y Mobile detecte alguna acción realizada a través de su dispositivo *smartphone*.	
Secuencia normal	Paso	Acción
	1	El usuario visualiza en la pantalla de Desktop una acción consecuente a la acción realizada en Mobile. Se clica un botón, se mueve un objeto, se pasa una página... Esto dependerá de la aplicación adaptada para usar *Slash and Pair*.
Postcondición	El usuario puede disfrutar del sistema hasta que cumpla su objetivo.	
Excepciones	Paso	Acción
	1	Las acciones previamente definidas para la aplicación configurada, pueden no ser utilizables por el tipo de dispositivo *smartphone* utilizado por el usuario, por lo que no podrá utilizar el sistema *Slash and Pair*.
	2	Al usuario se le desconecta el sistema, con lo cual deberá refrescar la página sin repetir el proceso de autenticación.
Comentarios	El proceso ha finalizado y se espera que el usuario consiga su objetivo satisfactoriamente.	

Tabla 13: Visualizar reacción Mobile

4.4 Análisis de seguridad

Este apartado pretende realizar un análisis de los factores importantes a tener en cuenta en cuanto a la seguridad para el proyecto *Slash and Pair*. Se incluye una breve introducción a la seguridad del software en general, para luego entrar al detalle del proyecto y las tecnologías.

Una vez introducida la materia de seguridad, se analizan las tecnologías que se han utilizado en el proyecto en base a la seguridad, los puntos críticos a tener en cuenta para la tecnología en cuestión.

Por cada punto crítico definido, se describe que solución/propuesta de seguridad se puede tomar para reducir la criticidad y el impacto que puede tener una vulnerabilidad como la analizada.

Una vez definidos los puntos críticos a tener en cuenta, se realiza un análisis DAFO teniendo en cuenta, qué propuestas de seguridad se han tomado para que

el proyecto sea seguro. En función de estos puntos implementados se definirán los puntos que el análisis DAFO contendrá.

4.4.1 Introducción a la seguridad

La ciberseguridad [21] es el área encargada de proteger y asegurar la integridad de cualquier sistema informático, en especial, la información que es contenida por ellos. Para cumplir esta premisa, existen estándares, protocolos, reglas, herramientas y leyes que permiten minimizar los riesgos, ya que un sistema totalmente seguro es una utopía.

El campo de la ciberseguridad incluye hardware, software y los datos que puedan contenerse y gestionarse a través de un sistema informático. En este documento solo se entra a detallar la seguridad a nivel de gestión de la información de los usuarios guardada en la base de datos y la integridad del sistema. No es necesario entrar en la seguridad de los sistemas y equipos informáticos debido a que, en este proyecto, se han utilizado equipos de terceros y esta tarea está incluida en el servicio proporcionado por la empresa dueña del servidor.

En la figura 8 se muestra en porcentaje [22] de los ataques más frecuentes sucedidos en el año 2016. En este documento no se entrará en todos ellos, tan sólo en los más comunes y analizados.

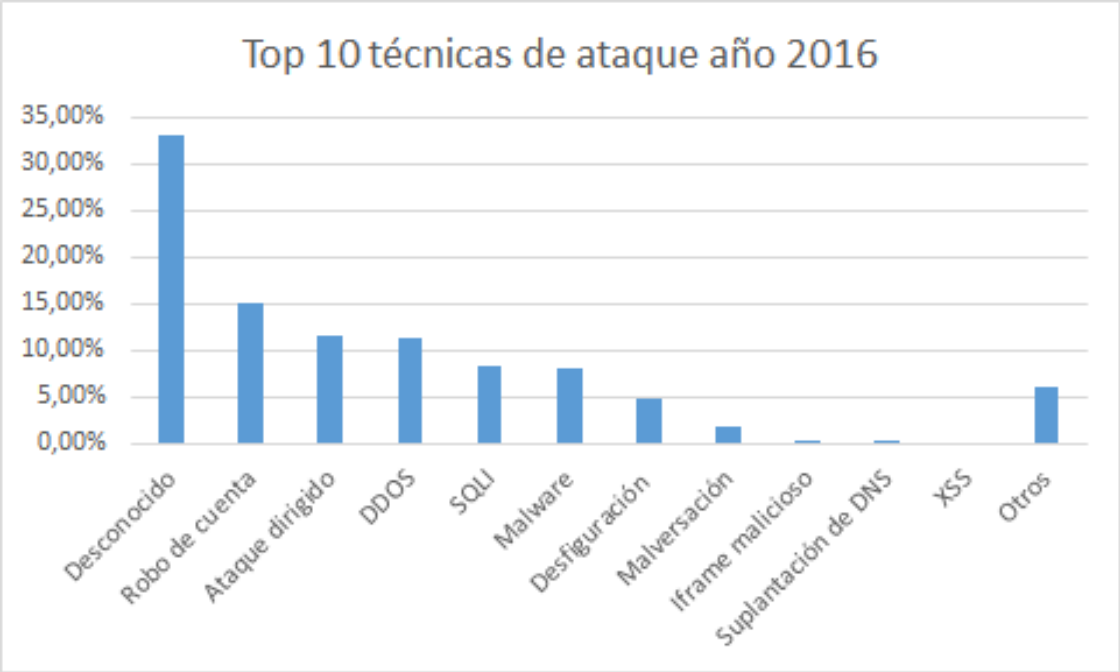

Figura 8: Top 10 ataques sucedidos en el año 2016.

Entonces, nos centraremos en la protección de:

- Los usuarios: Personas que utilizan el software descrito en el documento.

- La información: Los datos que generan los usuarios.
- La integridad del software: Que el servicio proporcione los documentos y archivos con la seguridad que no ha sido modificado o alterado.
- La disponibilidad: Asegurar la accesibilidad y la usabilidad del software.

A continuación, se realiza un breve análisis de los causantes de las amenazas y las amenazas generales que son susceptibles de poner en riesgo las propiedades descritas anteriormente en cualquier sistema informático.

4.4.2 Amenazas

Como amenazas, se entiende todo aquello susceptible de provocar un funcionamiento no deseado de un software. En este apartado se definen los causantes, los tipos de ataques y políticas de seguridad [23].

4.4.2.1 Causantes

En el apartado anterior se ha introducido el concepto de ciberseguridad y contextualizado dentro de *Slash and Pair*. Se detallan los causantes de amenazas que pueden afectar al proyecto:

- Usuarios: Usuarios con más permisos de la cuenta, la no restricción de acciones críticas.
- Programas maliciosos: Programas destinados a perjudicar o utilizar ilícitamente el sistema.
- Errores de programación: Los errores de programación se pueden usar como exploits (Aprovechar una vulnerabilidad para conseguir un comportamiento no deseado del mismo).

Intrusos: Entre los que se incluyen Hackers (experto en vulnerar la seguridad de un sistema y conseguir colarse en él) y Crackers (experto en vulnerar la seguridad de un sistema y conseguir colarse en él con propósitos ilícitos), son personas que consiguen acceder a información no autorizada, ya sea el propio software o datos de usuarios con intención maliciosa.

4.4.2.2 Ataques

En este apartado se describirán los principales ataques:

- XSS Cross Site Scripting:

Esta vulnerabilidad está altamente ligada al lenguaje de *scripting* JavaScript. El objetivo de JavaScript es hacer que el navegador realice tareas del lado del cliente, esto permite modificar la página web para que la página se modifique dinámicamente. JavaScript permite leer datos de las páginas web cargadas en ventanas. Un sitio web malicioso puede interactuar con los contenidos cargados en una página web completamente diferente que no está relacionada con su dominio.

Por lo general estos ataques se caracterizan por la inyección de código HTML y la inserción de comandos maliciosos (código JavaScript) para controlar el navegador web del usuario cuando el usuario está navegando por la página web. Utiliza para ello los datos de entrada no correctamente validados para que se interpreten como código.

- CSRF Cross Site Request Forgery:

Este tipo de ataque es funcionalmente lo contrario que un ataque XSS. Para el XSS el objetivo es el usuario, mientras que en CSRF el objetivo es el servidor web. Los ataques de este tipo engañan al navegador de la víctima presentando una petición a otro sitio donde el usuario está logado, haciendo creer que el usuario ha iniciado una acción y la acción se ejecuta como si el usuario la hubiera iniciado. Este ataque fuerza al navegador web de la víctima, validado en algún servicio (autenticado), a enviar una petición a una aplicación web vulnerable. Por lo tanto, este ataque ejecuta alguna acción en un sitio en el que el usuario nunca tuvo intención de ejecutar.

- DDoS Distributed Denial of Service:

Un servidor web es capaz de soportar cierto número de peticiones o conexiones simultáneas. Si se llevan a cabo muchas peticiones, poco a poco se consumen recursos hasta que el servidor comienza a rechazar las peticiones y en consecuencia comenzará a denegar el servicio.

- LFI Local File Inclusion:

Es una vulnerabilidad web que permite al atacante ejecutar archivos de forma local en el servidor y tener acceso a archivos delicados, de configuración, nombres de usuarios o contraseñas. Esta vulnerabilidad se observa en páginas que incluyen un archivo pasado por GET o POST sin validar el contenido.

4.4.2.3 Protecciones frente a los principales ataques

En este apartado se van a describir algunas medidas a realizar para evitar los posibles ataques, en base a cada uno de ellos:

- XSS Cross Site Scripting:

Una de las precauciones a tomar para evitar ataques XSS es filtrar y validar los valores introducidos por el usuario. Esto implica no permitir caracteres propios de la programación.

Hay otras soluciones como implementar filtros web, diseñar APIs privadas para el intercambio de información dentro de la aplicación y desinfectar las URL buscando elementos peligrosos como *scripts*.

- XSS Cross Site Scripting:

Los mecanismos más utilizados para protegerse contra este tipo de ataques son:

- Validar un *token* secreto:

Consiste en enviar un *token* que sirve para validar si la petición procede de una fuente autorizada. Este código debe ser difícil de descifrar.

Generalmente, en el momento de autenticar al usuario el servidor genera un *token* utilizando valores como el *timestamp* (fecha, hora con minutos y segundos), id de sesión del usuario... Y el servidor lo envía en la cabecera HTTP en un campo *hidden*, para que, cuando el usuario lance una petición viaje este *token* y el servidor pueda verificar si el *token* ha sido generado correctamente y entonces, verificar que la petición no procede de un atacante [24].

- Validar la cabecera HTTP *Referer*:

Esta cabecera permite ver la URL que ha enviado la solicitud.

Se pueden bloquear las peticiones que contengan la cabecera Referer con URL incorrectas.

- Inclusión de cabeceras adicionales:

Se pueden enviar cabeceras personalizadas, ya que el navegador evita que se envíen cabeceras personalizados entre orígenes cruzados. Por lo tanto, se pueden rechazar las peticiones que no vayan acompañadas de la cabecera.

- Inclusión de la cabecera Origen:

Esta cabecera indica desde el lugar el cual se inicia la petición. Proporciona información al sitio web con el fin de aceptar o rechazar las peticiones de diferentes orígenes.

- DDoS Distributed Denial of Service:

Los ataques DDoS no se pueden prevenir, pero se pueden tomar una serie de medidas para hacer más difícil a los atacantes realizar este tipo de ataques. Estas medidas van desde inspeccionar con firewalls las conexiones, como delimitar las peticiones por segundo de una misma IP, otra medida de arquitectura es ubicar los servidores en diferentes centros, con diferentes redes y no haya puntos de acceso únicos [25].

- LFI Local File Inclusion:

Para evitar este ataque, hay que estructurar el servidor de tal manera que tenga el mínimo privilegio posible, limitando el acceso de archivos para solo la carpeta del servidor. Así se evita el acceso a archivos del sistema.

Otras medidas a tomar es evitar que el servidor no utilice ninguna información proporcionada por el usuario para nombres de ficheros como recursos del servidor.

4.4.3 Análisis por tecnologías: La seguridad en el proyecto

En este apartado se procede a analizar tecnología por tecnología, analizando sus características en cuanto a factores de seguridad.

4.4.3.1 Redis

Para analizar la seguridad en cuanto al motor Redis, se hará un breve resumen de los puntos importantes que se deberían tener en cuenta al realizar un análisis básico de seguridad [26] [27]:

- Quién escucha el servicio:

Si no existe autenticación es posible acceder a todos los datos mediante comandos sencillos HTTP y otros comandos similares. En este caso, se utiliza Redis para manejar información sensible como datos de sesión de los usuarios. Manualmente se puede restringir la publicación de la información a un puerto concreto para que no sea visible para toda la red.

- Qué privilegios tiene por defecto el usuario que opere en Redis:

En caso de permitir a un usuario no deseado la realización de comandos críticos, como realizar un "set" modificando una entrada concreta es un déficit de seguridad.

Redis permite la escritura en determinadas rutas (directorios) [28]. Por tanto, es muy importante no ejecutar el cliente como root y desactivar algunos comandos como "config".

En caso de permitir la conexión cliente como root, se podría caer en un estado de denegación de servicio ya que un comando tan sencillo como "shutdown" podría ser invocado por cualquier cliente con esos permisos.

- Autenticación:

Por defecto Redis no incorpora autenticación para su uso, y hay que definirla manualmente mediante sus ficheros ".config".

- Establecer los comandos necesarios como únicos comandos ejecutables:

Este punto es importante para evitar que cualquier usuario pueda ejecutar comandos como "shutdown" y provocar una denegación de servicio. Desde Redis es posible desactivar los comandos que puedan ser un conflicto.

El problema va más allá de una denegación de servicio, ya que hay comandos que permiten eliminar entradas o simplemente toda la información contenida en el sistema.

- Análisis de puertos:

La base de datos ejecuta comandos que permiten realizar la migración de entradas clave/valor de un sistema a otro. El comando "migrate" puede proveer a un atacante información sobre los puertos consultados ya que, un puerto abierto y otro cerrado dan respuestas distintas. Es un factor de seguridad a tener en cuenta.

Como aclaración, el comando "migrate" de Redis permite realizar una copia de una entrada de una instancia de Redis, hacia otra instancia de Redis.

- Cifrado de datos entre cliente y servidor:

Los datos viajan de forma plana entre cliente y servidor, en el caso de Redis, el cliente de Redis y la instancia que lo consume, Spring, están ubicados en la misma máquina, lo que permite restringir la publicación de estos datos a través de la interfaz *"localhost"*. Este punto sólo se puede tratar con herramientas externas como SSH que se pueden utilizar para este caso en concreto.

Redis no incorpora ningún cifrado de los datos, así que cualquiera que pueda conseguir acceso a la red a través de la utilización de un sniffer, podría llegar a obtener toda la información gestionada por este sistema.

- Rendimiento y fuerza bruta en la autenticación:

La base de datos utilizada para este proyecto, está diseñada para utilizarse en entornos de confianza. Esto quiere decir que no se han aplicado medidas de seguridad en cuanto a la definición de accesos y roles. Tampoco para la protección y confidencialidad de los datos almacenados en este sistema.

Redis incluye una nota informativa [29] en la que describe que, debido al alto rendimiento de la base de datos Redis, es posible realizar un ataque por fuerza bruta en paralelo de manera que las claves deberán ser robustas.

A continuación, definiremos los puntos críticos de Rabbit.

4.4.3.2 RabbitMQ

Las vulnerabilidades de las instancias de Rabbit [30] son parecidas a las de cualquier sistema de servidor estándar por el cual circule cualquier tipo de información.

- Fuga de información:

La fuga de información es la situación en la que la aplicación muestra inapropiadamente información sensible, como mensajes en los negociadores de mensajes.

- Administración de sesiones.

Definimos la administración de sesiones como la capacidad de un atacante de interponerse a sí mismos como un usuario registrado del sistema. El sistema debe ser capaz de bloquear los usuarios no válidos.

El sistema RabbitMQ depende de las sesiones de usuario para conocer a qué individuo o sistema que utilice su servicio debe enviar la información, por lo tanto, es uno de los puntos más importantes.

- Autenticación y autorización:

Autenticarse, se entiende como el proceso en el que un usuario se autentifica en un ordenador a través de unas credenciales privadas. Autorización es la capacidad de acceder a recursos concretos, o realizar diferentes tipos de acciones.

La vulnerabilidad de la autenticación y la autorización se define como la utilización de estas credenciales a través de una aplicación que no se asegura de ser inquebrantable, y que las credenciales proporcionadas por externos no sean reproducibles, para evitar el acceso a la información privada. Como medidas a esta problemática las credenciales de autenticación deben ir debidamente encriptadas, cumplir con los estándares de las contraseñas. También utilizar una lista de control de acceso, que permite separa los privilegios y determinar los permisos de acceso apropiados a un determinado objeto.

- Métodos para paliar las vulnerabilidades:

Corregir y prevenir las fugas de información.

La primera de las acciones a tomar es proteger los documentos de los usuarios no autorizados. Dado que, los negociadores de mensajes están estrictamente dedicados al envío de información, hay que tomar medidas de protección. Las líneas generales en cuanto a protección son las siguientes:

- Pedir contraseña por defecto para no permitir los usuarios no autorizados.
- Borrar o cifrar la información de la fuga.
- Encriptar los mensajes entre los puntos de envío y recepción de los mensajes
- Limitar la gestión de la configuración del sistema.

Gestor de sesiones:

Para realizar los intercambios de información, es obligatorio establecer y administrar sesiones de los clientes [31] para prevenir el secuestro de la gestión de la sesión, la mejor solución es transmitirla a través de un protocolo de encriptación.

Uno de los más conocidos es el SSL (Secure Sockets Layer), que es un protocolo criptográfico que proporciona comunicaciones seguras por una red. Por ello, debemos usar SSL para prevenir el secuestro de la gestión de sesión.

Autenticación y autorización.

Este punto está relacionado con la seguridad de la gestión de la sesión. Se debe estar seguro que este sistema utiliza un protocolo seguro. Una de las posibles soluciones es la utilización del SSL para prevenir el secuestro.

Aplicando el control de acceso.

RabbitMQ proporciona una serie de mecanismos y plugins para gestionar todos los problemas de seguridad anteriormente mencionados. Para el acceso de control simplemente se especifican los permisos del usuario limitando sus acciones con el sistema en cuestión.

Cada usuario debe tener sus propios permisos que sólo le permitan realizar las acciones las cuales está autorizado para hacer.

Se pueden gestionar los accesos a través de RabbitMQ utilizando su herramienta de comandos.

4.4.3.3 Video en HTML5

Para incorporar en la página estática que provee Mobile el uso de la cámara es necesario realizar la conexión a través del protocolo HTTPS [32]. En las nuevas versiones del navegador Chrome de Google, se ha desactivado esta característica para origines no seguros, entre otras características.

4.4.3.4 Pairing

Slash and Pair es un sistema que funciona a través de la web, y una de sus características principales es el emparejamiento de dos sesiones de navegación distintas. La sesión se establece en el momento en el que un usuario se conecta utilizando un navegador al servicio web *Slash and Pair*. Como se ha detallado anteriormente, *Slash and Pair* está formado por dos proyectos, que proporcionan dos páginas web distintas, estas páginas generan dos valores de sesión distintos e independientes en el momento en el que algún usuario se conecta.

Este apartado se refiere a las cuestiones relacionadas con el enlace de las sesiones de ambos proyectos a través del sistema *Slash and Pair*. En resumen, el enlace se formaliza en el momento en el que el usuario introduce el código que es generado por Desktop en el entorno de Mobile.

Esta relación de sesiones web genera vulnerabilidades, ya que, si se utiliza un código un atacante puede intentar encontrar ese código para falsear la sesión de un usuario.

Por esta razón, vamos a analizar algunas de las diferentes formas para realizar el emparejamiento a través de un código, ya utilizadas por diferentes sistemas software.

4.4.3.4.1 4 dígitos

Los códigos de 4 dígitos están ampliamente extendidos. En un código de 4 dígitos existen 10.000 posibles combinaciones, si contamos que de los 4 dígitos posibles tenemos una franja entre los valores 0-9.

La complejidad a nivel de implementación no es muy alta. El desarrollador ha de utilizar un generador de números aleatorios. En el caso de *Slash and Pair* que está desarrollado en Java, sería posible realizar esta tarea aprovechando el paquete de Java "Random".

Si seguimos analizando la complejidad en cuanto a la implementación de este tipo de autenticación (ya se ha analizado la complejidad de generar el código), ahora se procede a analizar lo costoso de permitir al usuario introducir el código.

El usuario sólo podrá interactuar a través del navegador, por lo tanto, habrá que utilizar las herramientas provistas por un HTML. Para permitir al usuario introducir un código de 4 cifras se puede utilizar un cuadro "input" de tipo numérico.

En cuanto a la usabilidad, para el usuario es sencillo recordar un código de 4 dígitos para introducirlo en el lado Mobile. La memoria a corto plazo permite recordar una serie de aproximadamente 7 elementos hasta una duración de aproximadamente 30 segundos [33].

Tan sólo es necesario visualizar el cuadro de texto que le permita introducir los dígitos mostrados habiéndolos visualizado previamente en Desktop.

Si analizamos el factor de seguridad, en cuanto a probabilidades es posible utilizando métodos de fuerza bruta encontrar un pin válido ya que son limitadas las combinaciones posibles.

4.4.3.4.2 Sincronización mediante Patrón

Un patrón es una combinación de nodos situados en forma de matriz cuadrada como se muestra en la figura 9. Los nodos se pueden conectar entre ellos utilizando la pantalla táctil. Se desliza el dedo por encima de cada uno de ellos siguiendo una serie de reglas básicas:

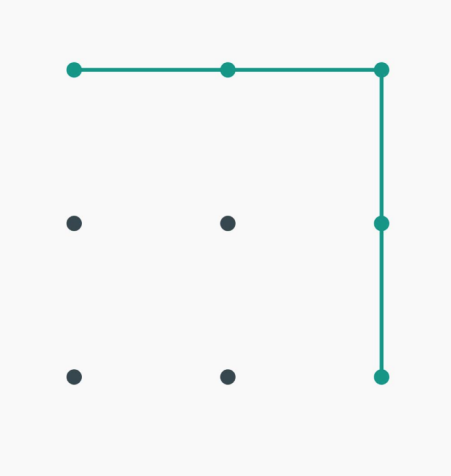

Figura 9: Ejemplo patrón desbloqueo de un *smartphone*.

- Para que el patrón sea válido hay que unir como mínimo 4.
- Un nodo ya marcado no se puede marcar otra vez.
- No se puede atravesar un nodo sin marcarlo.
- Se puede comenzar desde cualquier nodo.
- Se permiten las diagonales.

Utilizando estas reglas descritas, se calcula el número de posibles combinaciones que se pueden utilizar y el resultado es de 389.112 patrones distintos [34].

Para realizar la implementación del patrón dentro de *Slash and Pair*, lo primero que se debe definir es una generación aleatoria del patrón para que sea único de ese usuario. Una vez realizado el patrón aleatorio, hay que generar una matriz que se corresponda al patrón anteriormente generado y generar una imagen, para que se le muestre al usuario a modo de guía del trazo que deberá definir en la web Mobile.

Otro factor a tener en cuenta es como se codifica el patrón, como se estructurarán los datos para poderlos guardar, tratar y comparar. Debido a que la información que contiene el patrón es una secuencia de gestos en los que el orden importa, esta cuestión es importante solventarla.

Finalmente, en el lado Mobile es necesario implementar usando HTML una matriz con los 9 nodos para que el usuario pueda introducir el patrón que ha generado Desktop.

A nivel de usabilidad para los usuarios, a través del software se debería hacer un esfuerzo para que los patrones generados no sean difíciles. Debido a que la generación aleatoria implica que cualquier gesto que cumpla las normas establecidas anteriormente puede ser válido, puede haber una serie de códigos realmente difíciles de reproducir [35]. Por lo tanto, la usabilidad del patrón dependerá de cómo se ha generado y el número de nodos a enlazar y esta característica no es deseable cuando otros sistemas de verificación tienen una usabilidad estándar.

Si analizamos el factor de seguridad, también es posible utilizar fuerza bruta para encontrar un patrón válido y secuestrarlo. Las combinaciones son elevadas respecto al código de 4 dígitos.

4.4.3.4.3 Sincronización mediante código QR

Un código QR (Quick Response Code) es un código de barras bidimensional cuadrado, que puede almacenar datos de forma codificada. Esta matriz se puede leer a través de un dispositivo móvil utilizando lectores específicos. Presenta tres cuadrados en las esquinas que permiten detectar la posición del código al lector como se puede ver en la figura 10.

Figura 10: Ejemplo de código QR.

La implementación del código QR implica dos factores. El sistema Desktop debe poder generar la imagen del código añadiendo la información que sea necesaria para que Mobile pueda descodificarlo e interpretarlo. Por otro lado, Mobile debe ser capaz de capturar el código QR a través de la cámara del *smartphone* del usuario.

El factor de seguridad de un código QR es más elevada comparada con otros métodos de sincronización. Uno de los factores importantes es que la codificación propia del código QR hace que los datos de sincronización no viajen en claro desde el servidor hasta el HTML que verá el usuario. Además, los datos de sincronización son transparentes al usuario.

También es determinante entender, que un código QR permite almacenar 7.089 caracteres numéricos y 4.296 caracteres alfanuméricos [36]. Esta característica eleva las posibles combinaciones de códigos a generar.

El código QR tiene una alta capacidad para incorporar claves de sincronización largas y realmente difíciles de reproducir. Tratar de secuestrarlos a través de fuerza bruta no es viable.

4.4.4 Análisis DAFO

En este apartado se enumeraran los elementos característicos de *Slash and Pair*.

4.4.4.1 Debilidades

- Baja protección ante el XSS.
- Baja protección ante el CSRF.
- Los *tokens* que permiten realizar el *pairing* no caducan actualmente.

4.4.4.2 Fortalezas

- Publicación más restrictiva para Redis y RabbitMQ. La información tan solo es accesible por las aplicaciones que están ubicadas en la misma máquina.
- Buen sistema de log.
- Sistema de sincronización mediante QR seguro.
- El concepto de inyección no existe para Redis bajo circunstancias normales [28].

4.4.4.3 Amenazas

- Muchos usuarios concurrentes pueden hacer caer el servicio.
- El proyecto *Slash and Pair* es especialmente sensible a los ataques de denegación de servicio.

4.4.4.4 Oportunidades

- Uso de herramienta para TLS proporcionada por RabbitMQ para conexiones seguras.
- Aprovechar las características de Redis para blindar el uso de la BBDD.
- Spring proporciona métodos para sanitizar las peticiones para prevenir XSS.
- Spring proporciona protección ante el CSRF utilizando un patrón de *tokens* [37].
- Gracias al diseño del proyecto *Slash and Pair Mobile*, se puede escalar el sistema.

5 Diseño

En el apartado anterior se han detallado los casos de uso de la aplicación *Slash and Pair*, y más específicamente los casos de uso de la aplicación Desktop. Después se ha seguido analizando las principales vulnerabilidades de seguridad repasando cada una de las tecnologías utilizadas en *Slash and Pair Desktop*. Para terminar, se ha ejecutado un análisis DAFO en base a las características y vulnerabilidades detectadas.

También se han analizado en cuanto a la usabilidad, seguridad y complejidad los métodos más comunes y factibles para realizar un *pairing* entre las sesiones de Desktop y Mobile.

Se muestran las tecnologías escogidas para realizar el proyecto, una descripción de la interfaz gráfica, y se hablará del diseño de software.

5.1 Elección de tecnologías

En este apartado se detallarán las tecnologías utilizadas en el proyecto, esto incluye una breve introducción y como se han adaptado y aprovechado sus características en beneficio del proyecto *Slash and Pair*. También incluiremos librerías externas y un breve repaso del protocolo HTTPS.

5.1.1 Spring

Spring es un *framework* de código abierto para el desarrollo de aplicaciones. Es una plataforma Java que provee una infraestructura para el desarrollo de estas aplicaciones en este lenguaje.

El *framework* de Spring consiste en una serie de características organizadas en unos 20 módulos. Estos módulos se organizan de la forma que muestra la figura 11:

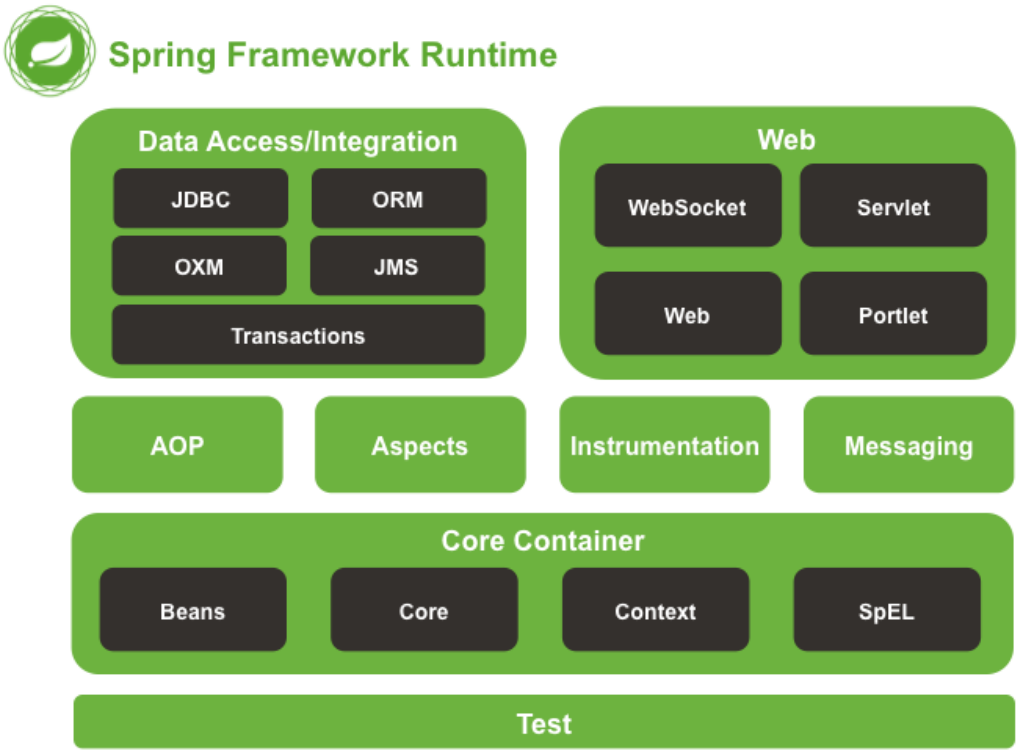

Figura 11: Modulos que componen Spring *Framework*.

El factor a destacar de Spring es su inversión de control, que provee una forma consistente de organizar los objetos Java. Spring contiene un contenedor de inversión de control. Este contenedor se encarga de crear los objetos, inicializarlos, y configurarlos. Estos objetos también son conocidos como "beans". Las configuraciones de los objetos se especifican a través de un fichero XML en las versiones antiguas de Spring o con notaciones Java en las versiones recientes.

Dentro del *framework* Spring cabe destacar un módulo llamado Spring Security.

5.1.1.1 Spring Security

Spring Security [17] proporciona seguridad para aplicaciones basadas en Java específicas para empresas. Este módulo proporciona facilidades para configurar la seguridad además de muchas y diversas características.

El énfasis de este módulo son dos factores, la autenticación y la autorización (control de acceso).

Esta funcionalidad proporcionada para la gestión de los usuarios, es útil para *Slash and Pair* ya que permite identificar a los usuarios de manera interna y gestionar si tienen acceso a las peticiones que realizan.

5.1.2 Maven

Apache Maven [38] es un proyecto de gestión y una herramienta de comprensión. Basada en el concepto de Project Object Model (POM). Maven gestiona la compilación del proyecto, la elaboración de informes y la documentación a partir de una información central.

La meta principal de Maven es permitir a un desarrollador comprender el completo estado del esfuerzo de un desarrollo en el menor tiempo posible por esto, surgen una serie de objetivos:

- Facilitar el proceso de compilación.
- Proveer un sistema de compilación uniforme.
- Proveer información de calidad del proyecto.
- Proveer buenas directrices de programación.
- Permitir la migración transparente a nuevas funcionalidades.

Maven permite a *Slash and Pair* tener un único proyecto, del cual se pueden extraer dos elementos ejecutables, Mobile y Desktop con tan sólo una instrucción de consola.

5.1.3 Redis

Redis [39] es una BSD licenciada *open source*, un contenedor de estructura de datos en memoria utilizado como base de datos, caché y gestor de mensajería. Soporta estructuras de datos como cadenas, *hashes*, listas, *sets* y listas ordenadas con consulta de rangos.

Para *Slash and Pair* es una base de datos compartida entre Desktop y Mobile donde básicamente tienen lugar 2 operaciones, escritura por parte del servidor Desktop y lectura por parte del servidor Mobile.

Spring proporciona un cliente de Redis incluido dentro de sus módulos, por lo tanto, se pueden realizar interacciones a través de Java. A pesar de ello, es necesario instalar Redis en nuestro servidor y utilizando Spring, que incluye todas las dependencias y herramientas que son necesarias para utilizar Redis, tan sólo con incluirlo como una dependencia de Maven es suficiente.

En caso de ser necesario por cuestiones de arquitectura, externalizar Redis, como otros módulos contenidos en el proyecto, se podría realizar una instalación del servidor de Redis en otra máquina distinta que no necesariamente esté ejecutando simultáneamente *Slash and Pair*.

Redis es la elección idónea para *Slash and Pair* por sus características respecto a otras bases de datos. Las ventajas que nos han llevado a escoger Redis son las siguientes:

- Redis trabaja con un sistema de Map (como la clase Map de Java) que, a través de una *key* permite encontrar el contenido identificando únicamente por esa *key*. *Slash and Pair* utiliza como *key* el propio *token* de *pairing* y como contenido a guardar la información correspondiente al usuario.
- *Slash and Pair*, por la información que necesita almacenar no requiere de un esquema complejo de base de datos, cosa que si proporcionan otros sistemas SQL.
- Redis es una base de datos con un alto rendimiento [40].

5.1.4 Introducción a RabbitMQ

RabbitMQ [30] es un software de negociación de mensajes de código abierto. Se puede considerar como un *middleware* de mensajería. Implementa el estándar AMQP. Este estándar se define por sus características de encolamiento y enrutamiento (publicación - suscripción). También estipula el comportamiento del servidor que provee los mensajes como el cliente de la mensajería. Por lo tanto, Rabbit realiza dos acciones que son fundamentales para *Slash and Pair*.

- Realiza la acción de encontrar el destinatario de un mensaje.
- Permite añadir datos en una cola donde Mobile es el productor y Desktop el consumidor.

Estas características son fundamentales para el correcto funcionamiento de *Slash and Pair*

5.1.5 WebSockets

Los WebSockets [41] es una tecnología que permite abrir una sesión de conexión interactiva entre el navegador de un usuario y el servidor. Se pueden enviar mensajes en ambas direcciones asíncronamente.

Esta tecnología permite a *Slash and Pair* abrir canales de comunicación por el cual se transmitirán los datos desde Mobile hasta el servidor Mobile, y desde el servidor Desktop hasta el navegador Desktop.

5.1.6 ZXING

ZXING [42] es una librería escrita en lenguaje Java que permite la generación de códigos QR. Esta librería se ha incorporado al proyecto *Slash and Pair* para poder crear el código QR en función de la necesidad de un sistema de *tokens* sencillos de utilizar.

5.1.7 HTML5

HTML5 [43] es la última versión de HTML. Se ha utilizado esta última versión debido a que se han incluido nuevos elementos multimedia. Esto es fundamental para utilizar la cámara desde Mobile, ya que, se ha incluido una etiqueta propia de HTML llamado "video" que permite inicializar la captura de una forma tan sencilla como incluir ese tag en la página estática.

5.1.8 WebCodeCamJS & WebCodeCamJQuery

WebCodeCamJS & WebCodeCamJQuery [44] es una librería escrita en lenguaje JavaScript que permite realizar la decodificación dinámica de códigos QR. Esta librería implementada en la página de Mobile permite decodificar los códigos QR generados por Desktop para sacar el valor del *token* codificado.

5.1.9 Apache Tomcat

Apache Tomcat [45] es una implementación *open source* de las tecnologias Java Servlet, JavaServer Pages, Java Expression Language y WebSockets de Java. Funciona como un contenedor de servlets y como un servidor web por sí mismo.

5.1.10 Certbot

Certbot [46] es un cliente sencillo, que funciona automáticamente generando y desplegando un certificado SSL/TLS para un servidor web. Este cliente facilita la tarea de generar certificados y como se detallará en el apartado de *"Despliegue"* con unas pocas líneas de código es suficiente para tener la aplicación web bajo el procotolo HTTPS.

5.1.11 NoSleepJS

NoSleepJS [47] es una librería JavaScript que permite mantener la pantalla encendida, evitando así el bloqueo automático que suele ser predeterminado en todos los *smartphones*. Esta librería se implementa en el proyecto *Slash and Pair Mobile*

5.1.12 Análisis del protocolo de comunicación HTTPS

HTTPS [48] son las siglas de *Hypertext Transfer Protocol Secure*. Es un protocolo de aplicación, esto le proporciona la capacidad de acceder a los servicios de intercambio de datos que utilizan otras aplicaciones y servicios. En resumen, es la versión segura de HTTP.

Este protocolo utiliza un cifrado basado en SSL/TLS [50], esto significa que se utilizan certificados digitales para establecer comunicaciones seguras a través de Internet.

5.2 Diseño de la interfaz gráfica

La interfaz gráfica de *Slash and Pair Desktop* permite al usuario realizar las acciones para las cuales está diseñada la aplicación. Es una interfaz simple y sencilla en la que se intenta proporcionar de forma explícita los pasos que ha de ir realizando el usuario para que se el sistema se comporte como se espera.

El diseño de la interfaz en *Slash and Pair Desktop* está formado por dos páginas principales, correspondiéndose a los requisitos de usabilidad generales de la aplicación.

Estas páginas en realidad son el mismo fichero HTML que se modifica a través de funciones JavaScript, es decir, por los eventos que suceden a lo largo del uso de la aplicación.

Para comenzar, en la figura 12 se muestra la primera pantalla que verá el usuario al acceder a Desktop.

Figura 12: Interfaz para la conexión.

En esta página como se muestra en la figura, se visualiza una página estática que contiene un título, el código QR y una breve explicación para que el usuario, en caso de no poder utilizar la cámara desde su *smartphone* tenga la alternativa para que se le genere un código.

La intención de esta pantalla tan básica, es que el usuario visualice el código QR, que tiene un tamaño considerable respecto a todo el contenido intencionadamente, y se dirija automáticamente a la página de Mobile para poner en uso su cámara. Se incluyen instrucciones para facilitar el proceso al usuario.

Se ha tenido en cuenta definir un contraste alto de los elementos legibles en la página para que sea fácil de leer.

Es un diseño en el que el usuario tiene en todo momento constancia del estado del sistema, ya que cuando la única acción que puede realizar es la sincronización,

el usuario es notificado al momento.

La siguiente pantalla consiste en la aplicación que va a utilizar las interacciones realizadas en Mobile para modificar su estado o realizar alguna acción, y esta no forma parte de la aplicación *Slash and Pair*, ya que como se muestra en el apartado *"Ejemplo de ejecución"* la interfaz corresponde a la aplicación que utiliza *Slash and Pair* .

En resumen, se tienen dos capas al iniciar *Slash and Pair Desktop*, la primera de ellas (la capa superior) no permite utilizar la capa de abajo (la capa inferior y la aplicación) hasta que el *pairing* de sesiones no se haya realizado.

5.3 Diseño de software

Slash and Pair es un proyecto escrito en Java que consta básicamente de 3 grandes bloques.

- Slash and Pair Desktop
  - Definición de la página estática que verá el usuario al introducir la URL Desktop, e implementa la funcionalidad del servidor Desktop.
- Slash and Pair Mobile
  - Definición de la página estática que verá el usuario al introducir la URL Mobile, e implementa la funcionalidad del servidor Mobile.
- Slash and Pair Exchange
  - Contiene las clases que definen las estructuras de datos.
  - Contiene clases útiles para la utilización de los códigos QR.
  - Contiene las clases necesarias para generar los objetos Java que gestionan los *tokens*.

Estos tres grandes bloques se engloban en un proyecto Maven único que permite generar los dos ejecutables necesarios para la utilización de *Slash and Pair*.

Una vez desplegada la aplicación, un esquema básico de los componentes que forman *Slash and Pair* es el que se muestra en la figura 13:

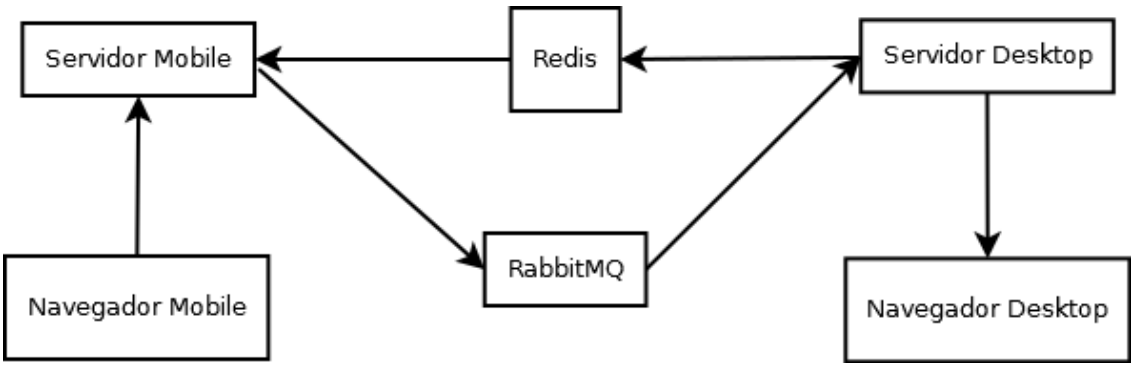

Figura 13: Flujo de datos de la aplicación.

Las flechas indican el flujo de datos de la aplicación. Redis como Rabbit son compartidos entre ambos proyectos.

La gestión de la información es la siguiente durante el uso de la aplicación:

- Desktop genera el *token* y lo guarda en Redis
- Mobile consulta el *token* introducido por el usuario dentro de Redis
- Si hay coincidencia, lanza a través de Rabbit un aviso.
- Comienza el envío de datos desde Navegador Mobile hasta Navegador Desktop

Dentro del proyecto Desktop, se contiene un HTML estático que muestra un código. Este código, se genera en cuanto el usuario se conecta a la URL "slash-and-pair-d,theblackbox.io". Una vez el usuario pulsa el botón acceder del navegador, el servidor a través de Spring reconoce al usuario como *principal* (descripción de un usuario por Spring), y se genera un *token* aleatorio único para este usuario. La sesión del usuario, como el *token*, como también un identificador aleatorio que se utiliza internamente para reconocer a un determinado usuario estarán relacionados por un objeto Map de Java. Este objeto se guarda en la base de datos de Redis. Es la gestión de la información previa a la conexión.

Por el lado de Mobile, cuando el usuario se conecta a la url correspondiente al entorno Mobile "slash-and-pair.theblackbox.io", se le muestra también una página HTML estática. En este caso al usuario se le muestra un cuadro para la cámara y además un input text y dos botones. Por el lado del servidor, lo único que se realiza automáticamente es detectar el *principal* (usuario) por Spring.

En el momento en el que el usuario ha accedido a ambas páginas y las mantiene abiertas simultáneamente, el usuario podrá introducir el código mostrado en Desktop en el HTML provisto por Mobile. En el momento en el que se pulsa el botón, realizamos un post a una URL definida dentro de Mobile "/sync" que, al detectar una petición, nos permitirá realizar la ejecución de un método Java que busca en una base de datos Redis un objeto que corresponda al valor que se ha introducido. Si se encuentra una coincidencia se realiza la conexión entre ambos sistemas.

Si visualizamos las opciones de desarrollador que nos muestran los navegadores, veremos que las variables "jsessionid" se han sincronizado. A pesar de inicialmente haber creado dos sesiones distintas, internamente las gestionamos como una única sesión.

En resumen, la lógica de Mobile es la siguiente:

- Se recoge el usuario provisto por Spring y el *token* y se intenta autenticar.
- Si el código introducido por el usuario existe dentro de Redis se emparejan las sesiones y se autentica al usuario. Si no, se muestra una pantalla de error al usuario.
- El usuario está autenticado, enviamos de Mobile a Desktop una notificación que corresponde a que el usuario actual ha realizado la conexión.

La notificación del emparejado de la sesión de Mobile hacia Desktop se hace a través de una cola RabbitMQ llamada *"pairing"* distinta a la del envío de datos cuyo nombre es "data", y una vez notificado, ambos lados se "suscriben" a la cola de envío de datos.

El siguiente paso es definir como gestiona el sistema el envío de datos. Los datos que se vayan a traspasar de un lado a otro, utilizaran una cola situada en medio de los dos servidores. Es decir, Mobile publicará los datos en la cola y Desktop dispondrá de un *listener* (elemento que está pendiente al suceso de algún evento para realizar alguna función concreta) que lanzará una llamada a un método Java cada vez que se hayan publicado datos nuevos, de esta forma podrá realizar acciones con estos datos recién recibidos.

5.3.1 Uso de Slash and Pair con un juego implementado para navegadores web

Una vez se ha acabado el desarrollo de la plataforma de envío de datos entre el servidor Desktop y el servidor Mobile, es necesario implementar alguna aplicación que permita demostrar el potencial de la aplicación.

Para ello, se ha seleccionado un juego [49] *open source* que permite explotar el giroscopio como medio de introducción de datos.

Este juego es muy simple, consiste en un laberinto, y el usuario a través de las teclas correspondientes a las flechas, puede realizar el movimiento a través del laberinto. *Slash and Pair Desktop* ha modificado el código fuente para permitir que los datos enviados desde Mobile se puedan aprovechar para aplicar la misma funcionalidad que las flechas de dirección.

Dado que, el uso del juego a través del *smartphone* implica tener el dispositivo encendido mucho rato, se ha integrado el uso de la librería *"NoSleepJS"* definida en el apartado de *"Tecnologías"* añadiendo un botón que permite al usuario clicar con el fin de que su pantalla del *smartphone* no se apague. Es importante la integración

de esta librería ya que, el apagado de la pantalla supone una desconexión del canal WebSocket abierto.

5.4 Despliegue de la aplicación en un entorno real

En este apartado se explicará la metodología y las herramientas utilizadas para realizar el despliegue de la aplicación *Slash and Pair* en un servidor Linux.

El primer paso a realizar es instalar todo el software necesario para realizar el despliegue. Como ya se ha comentado a lo largo del documento, la aplicación *Slash and Pair* está escrita en código Java, además requiere del uso de una base de datos Redis y de una cola RabbitMQ. Por lo tanto, se ejecuta el comando que permite saber si el servidor tiene Java instalado:

Script 1: Visualizar versión de Java

```
java -version
java version "1.8.0_60"
```

El servidor está corriendo la versión java 1.8.0_60.

El siguiente paso es realizar la instalación de Redis utilizando el siguiente comando:

Script 2: Instalar Redis

```
apt-get -s install redis-server
```

Instalando Redis se nos añaden dos herramientas, "redis-tools" y "redis-server". Una vez se ha instalado la última versión de Redis, se procede a realizar la instalación de RabbitMQ con el siguiente comando:

Script 3: Instalar RabbitMQ

```
apt-get -s install rabbitmq-server
```

Rabbit, además de su servidor y cliente nos instalará el lenguaje Erlang, que es el lenguaje con el que está escrito su código fuente.

Una vez tenemos disponible en el servidor el software de gestión de la información, se procede a realizar la instalación Apache Maven, que como se ha especificado, es la herramienta con la que vamos a compilar el proyecto. Ejecutamos entonces la siguiente línea de comandos:

Script 4: Instalar Maven

```
apt-get -s install maven
```

Y verificamos que la instalación se ha completado correctamente visualizando la versión de Maven:

Script 5: Versión de Maven

```
mvn -v
Apache Maven 3.3.9
```

```
(bb52d8502b132ec0a5a3f4c09453c07478323dc5;
2015-11-10T17:41:47+01:00)
Maven home: /usr/share/maven3
```

Una vez se tiene todo el software necesario para la compilación listo, debido a que el proyecto se alberga en Github, se comprueba que la versión de Github de la máquina es la última con este comando:

Script 6: Versión de Github

```
git --version
git version 1.9.1
```

Entonces procedemos a crear los directorios para albergar el código:

Script 7: Creación de directorios

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

Nos bajamos el código a través de la URL que nos proporciona el enlace de github:

Script 8: Descargar el proyecto de Github

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

Descomprimimos, y con el comando de Maven generamos el ".jar" ejecutable, finalmente movemos cada ejecutable dentro de las carpetas creadas anteriormente para tenerlo correctamente organizado:

Script 9: Compilación del proyecto con Maven

```
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
```

Llegado este punto, ya están generados los ejecutables y todo el entorno configurado. Ahora se procede a configurar Apache Tomcat y se realiza la instalación:

Script 10: Instalación de Apache Tomcat

```
apt-get install tomcat7
```

Y se crea un *virtual host* para permitir que dentro de este servidor utilizado se pueda albergar más de un sitio web.

La configuración utilizada para desplegar *Slash and Pair* es sencilla. Definimos "ServerName" el nombre de la URL *slash-and-pair-d.theblackbox.io* y como

"ServerAlias" *slash-and-pair-d.theblackbox.io*. Configuramos los ficheros log y realizamos un ProxyPass, que redirigirá la aplicación *Slash and Pair* que va a estar ejecutándose en *localhost* en el puerto 8082.

Una vez Apache está configurado, al introducir en el navegador la dirección definida, Apache redirigirá la petición hacia la aplicación Java que estará corriendo dentro del servidor.

La última configuración de todas se realiza para no permitir que la aplicación se ejecute en "HTTP" si no que solo se ejecute utilizando el protocolo seguro "HTTPS". Para ello se ha utilizado la herramienta Certbot.

Instalamos la herramienta Certbot en el servidor:

Script 11: Instalación Certbot

```
apt-get install software-properties-common
add-apt-repository ppa:certbot/certbot
sudo apt-get update
sudo apt-get install python-certbot-apache
```

Una vez instalado, ejecutamos el comando de certbot:

Script 12: Versión Certbot

```
certbot --apache
```

Y esto generará el certificado que podremos utilizar para la aplicación que esté configurada a través de Apache.

El último paso es desplegar la aplicación y para eso se ha generado un pequeño *script* que es el siguiente:

Script 13: Versión Certbot

```
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 comando *"nohup"* permite ejecutar un comando después de salir de la terminal. Esto permite ejecutar desde la *Shell* la aplicación y tenerla corriendo en segundo plano pudiendo cerrar la consola des de la cual se ha ejecutado el comando.

El comando situado en la línea 4 nos permite sacar en un fichero de texto el *"process id"* que identificará el proceso que está ejecutando la aplicación Desktop, en este caso. Este fichero será útil en el momento en que queramos parar/matar la aplicación utilizando el siguiente comando:

Script 14: Matar proceso

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

Este comando buscará por el pid guardado en el fichero, y matará el proceso.

6 Ejemplos de ejecución

En este apartado se realizará una muestra de cómo es el proceso de ejecución del proyecto *Slash and Pair* para un usuario que disponga de un *smartphone* con cámara y otro sin cámara, adjuntando imágenes conforme se va realizando la prueba.

6.1 Usuario con smartphone con cámara

El usuario se coloca frente al ordenador, y prepara las conexiones a Internet de su *smartphone* y ordenador para poder ejecutar *Slash and Pair*.

Primero se abre el navegador Chrome y se introduce la siguiente URL:

- https://slash-and-pair-d.theblackbox.io

Una vez se ha introducido la URL, el usuario visualizará en su navegador lo que mostramos en la figura 14:

Figura 14: Página Desktop con código QR.

Visualizamos un código QR y un enlace que genera un código en el caso en el que el usuario no quiera utilizar su cámara.

Ahora accedemos a la siguiente URL desde nuestro *smartphone*:

- https://slash-and-pair.theblackbox.io

El usuario visualizará una pantalla como la que se muestra en la figura 15 en su *smartphone*:

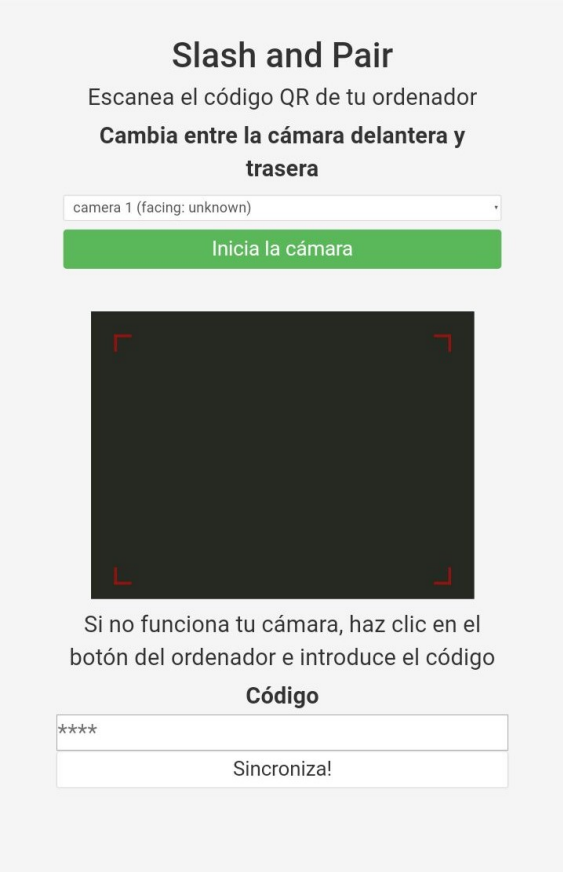

Figura 15: Página Mobile.

En esta pantalla se puede visualizar un cuadro de un color gris oscuro, donde se va a mostrar la imagen de la cámara. También, en la parte superior del cuadro donde se mostrará la cámara, el usuario verá un botón verde, cuyo uso es el de activar la cámara.

Además, se muestra un cuadro de texto preparado para aquellos usuarios que no puedan utilizar su cámara. Entonces, el usuario clica al botón *play* y visualizará la imagen captada por su cámara.

El usuario entonces, enfocará al código QR y, una vez sea escaneado se verá la imagen congelada. Pocos segundos después, le cambiará la pantalla de Desktop y verá un juego que consiste en un laberinto, dónde, con el giroscopio del *smartphone* controlará la bola y el objetivo es salir del laberinto. La pantalla que ve el usuario es como la que se muestra la figura 16.

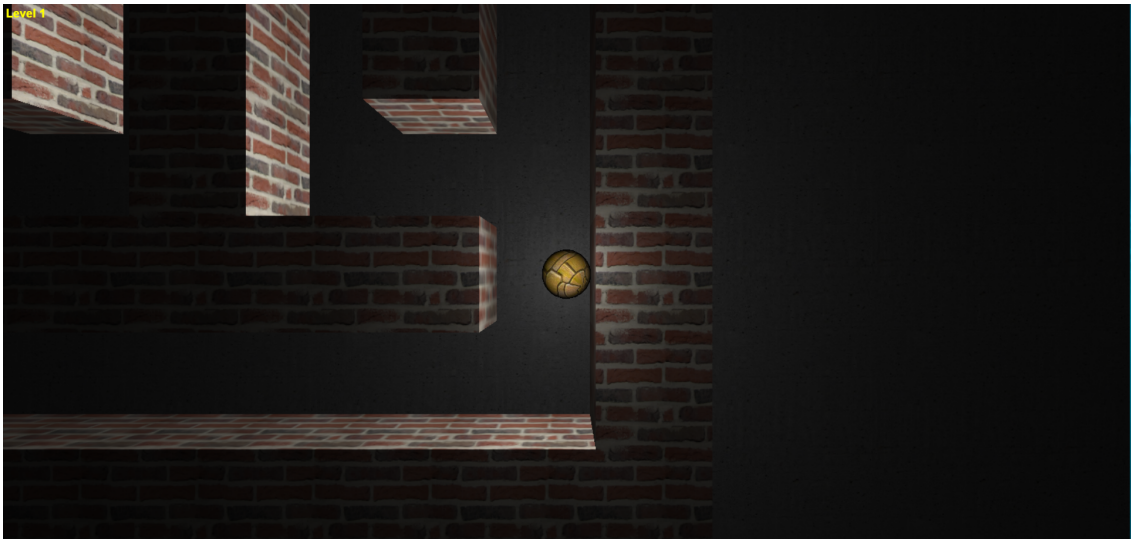

Figura 16: Página Desktop con demostración.

Por último, en su *smartphone* también percibirá que se ha modificado la página y ahora, se le mostrará una pantalla donde el usuario leerá una pequeña instrucción de la acción que debe realizar para poder controlar Desktop debidamente, como se puede visualizar en la figura 17.

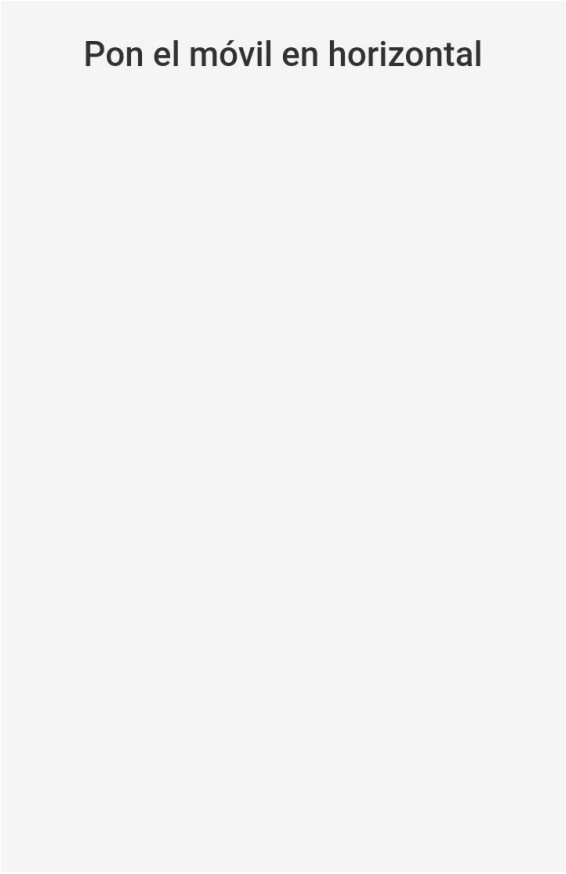

Figura 17: Página Mobile informando al usuario cómo proceder.

El usuario deberá colocar el *smartphone* como se le indica, y podrá, girándolo derecha-izquierda, arriba-abajo, tener el control de la bola instantáneamente, estas acciones posibles se reflejan mediante unas flechas en la pantalla de Mobile que se muestra en el momento en el que el usuario pone el *smartphone* en horizontal, como se muestra en la figura 18:

Figura 18: Página Mobile informando al usuario cómo realizar acciones.

Se ha incluido, un botón que al pulsarlo evita que el *smartphone* bloquee su pantalla.

6.2 Usuario con smartphone sin cámara

El usuario se coloca frente al ordenador, y prepara las conexiones a Internet de su *smartphone* y ordenador para poder ejecutar *Slash and Pair*.

Primero se abre el navegador Chrome y se introduce la siguiente URL:

- https://slash-and-pair-d.theblackbox.io

Una vez se ha introducido la URL, el usuario visualizará en su navegador lo que mostramos en la figura 14.

Visualizamos un código QR y un enlace que generaría un código en el caso en el que el usuario no quiera utilizar su cámara. En este caso el usuario hará clic encima del enlace y se le cambiará de la siguiente forma la pantalla, como se muestra en la figura 19:

Figura 19: Página Desktop con código.

Ahora accedemos a la siguiente URL desde nuestro *smartphone*:

- https://slash-and-pair.theblackbox.io

El usuario visualizará una pantalla como la que se muestra en la figura 15 en su *smartphone*.

En este caso, el usuario no hará clic el botón play para utilizar la cámara si no, que introducirá el código en el cuadro que se visualiza en la parte inferior de la pantalla, pulsando seguidamente el botón "Sincronizar" como se puede ver en la figura 20:

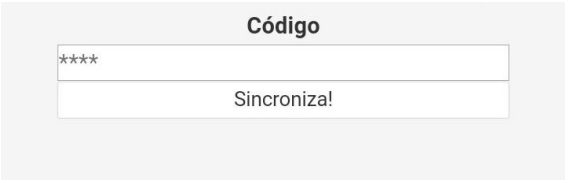

Figura 20: Página Mobile apartado introducción del código.

Por último, en su *smartphone* también percibirá que se ha modificado la página y ahora, se le mostrará una pantalla donde el usuario leerá una pequeña instrucción de la acción que debe realizar para poder controlar Desktop debidamente, como se puede visualizar en la figura 17.

El usuario deberá colocar el *smartphone* como se le indica, y podrá, girándolo derecha-izquierda, arriba-abajo, tener el control de la bola instantáneamente, estas acciones posibles se reflejan mediante unas flechas en la pantalla de Mobile que se muestra en el momento en el que el usuario pone el *smartphone* en horizontal, como se muestra en la figura 18.

Se ha incluido, un botón que al pulsarlo evita que el *smartphone* oscurezca su pantalla.

7 Resumen y conclusiones

Slash and Pair, como proyecto software cumple actualmente todos los requisitos funcionales descritos en la fase de análisis del proyecto, en el apartado *"Requisitos funcionales"*. Estos requisitos incluyen los dos grandes apartados de la funcionalidad de *Slash and Pair*, que son la sicronización y la interacción.

Se ha llegado a la mejor decisión posible para la solución del problema de la sincronización de las dos instancias web por parte de un único usuario. Esta solución empleada es escalable, de manera, que por más usuarios que pueda tener *Slash and Pair* siempre habrá seguridad para que puedan realizar sus sincronizaciones sin tener el miedo, de poder utilizar un código ya utilizado por otro usuario.

Este proyecto ha sido muy útil, ya que ha implicado el aprendizaje de muchas tecnologías y un *framework* muy utilizado en el mundo de la informática.

Todo el proceso de realización del TFG también ha sido útil, para conocer el proceso de desarrollo de un proyecto de software real diseñado para que tenga una especificación compleja y transferible a un entorno real.

Por otro lado, en la definición inicial del proyecto no se contempló la posibilidad de utilizar un código QR para realizar la sincronización entre dispositivos, es por ello que en este apartado se debería haber trabajado más, ya que, utilizar la cámara de un *smartphone* a través de un sitio web no es una problemática trivial, dado que hay muchos problemas de incompatibilidades entre diferentes tipos de navegadores y dispositivos. En la realización de este trabajo no se ha podido disponer de los medios necesarios para realizar las pruebas en una gran cantidad de dispositivos, con distintos sistemas operativos, con distintos navegadores... Esta línea es una línea que se puede reforzar de cara al futuro.

En cuanto a la finalidad del análisis de seguridad, la intencionalidad con la que se ha llevado a cabo, es la de facilitar la implantación de las medidas ya que, su desarrollo se escapa de lo abarcado en el período de realización del TFG. Gracias al análisis realizado, aplicar las medidas de seguridad correspondientes se simplifica. En este proyecto se definen todas las herramientas necesarias para aplicar las medidas requeridas para que todo el esfuerzo realizado en el análisis sea provechoso.

Al final, el proyecto resultante es interesante y satisfactorio ya que mezcla muchas tecnologías que realizan a la perfección su función y, como ha quedado demostrado, realizar una planificación correcta al inicio del desarrollo de este proyecto, como en cualquier otro, permite que el programador, pueda alcanzar sus objetivos de una manera eficiente ya que los cambios rápidos no planificados inicialmente que se van añadiendo a última hora tienen el efecto contrario en el proceso.

Las pruebas realizadas con usuarios, que han dado un buen feedback del uso de la aplicación, destacando, la sencillez que caracteriza el uso de *Slash and Pair*. Esta sensación de sencillez enmascara la dificultad que hay detrás de su desarrollo.

8 Líneas de futuro

En este apartado se hará un repaso a todas las posibilidades que, en caso de continuar el desarrollo del proyecto se podrían abordar aprovechando la funcionalidad de *Slash and Pair*.

La primera opción que se puede abordar es la de redirigir el proyecto para que sea una API.

8.1 API

Una API [51] es una interfaz de programación de aplicaciones, que contiene un conjunto de subrutinas, funciones y procedimientos, que ofrece una librería concreta.

Como ya se ha visto en ejemplos de ejecución, actualmente *Slash and Pair* lo que proporciona es un entorno, donde se puede incrustar cualquier aplicación que funcione sobre un entorno web y adaptarla para que funcione con los sensores del *smartphone*. La idea de la API va más allá. Debido a la dificultad que supone para un desarrollador incorporar todo su trabajo dentro de una aplicación externa, la API de *Slash and Pair* permitirá evitar toda esa dificultad.

La idea de la API es una aplicación web, que funcionará exactamente igual que el proyecto actual pero que en lugar de realizar las acciones sobre el mismo entorno web sobre el que se está ejecutando *Slash and Pair*, la API sólo proporcionaría los datos requeridos por el usuario dentro de una estructura de datos como un JSON que podrían ser utilizados por cualquier web, tan solo realizando peticiones al servidor *Slash and Pair*.

Por lo tanto, en resumen, esta API podría hacer que *Slash and Pair* fuera transparente para el usuario y que, cualquier web, utilizando las funciones que provee este proyecto pueda lanzar peticiones contra una URL que devolverá los datos necesarios para que el uso del sistema siga siendo el mismo, pero mucho más sencillo de integrar.

8.2 User Experience

User Experience (UX) [52] es una filosofía de diseño que tiene por objetivo la creación de productos que resuelvan necesidades concretas de los usuarios, consiguiendo así, una mejor experiencia y satisfacción de uso con el mínimo esfuerzo.

Una vez definido el término *User Experience*, se puede comprender hacia dónde va dirigida está línea, y es que, *Slash and Pair*, proporciona una nueva forma de comunicarse entre el usuario y el ordenador.

Por tanto, el trabajo a realizar consiste en conocer a fondo las disciplinas sobre las cuales puede trasladarse el proyecto y también conocer a los usuarios finales, para ajustar a las necesidades específicas de cada disciplina *Slash and Pair*.

8.3 Conexiones múltiples entre dispositivos

Como se ha ido viendo a lo largo de todo el documento, *Slash and Pair* actualmente conecta un *smartphone* a un ordenador y permite que las acciones realizadas en el *smartphone* lleguen y tengan un efecto sobre el ordenador.

Y, ¿por qué esta limitación? Que sólo pueda realizarse el *pairing* entre dos dispositivos con especificaciones distintas limita las posibilidades del sistema.

Y es que, esta línea propone continuar el proyecto permitiendo una conexión de varios *smartphones*, por ejemplo, para jugar un juego implementado en web con varios jugadores a la vez. También está la conexión de un *smartphone* a varios Desktops, y es que esto permitiría realizar las acciones de un *smartphone* en varios Desktops a la vez.

8.4 Adaptación del sistema a una aplicación nativa

Slash and Pair actualmente se ejecuta sobre la web. Este entorno limita las posibilidades de uso de los sensores del teléfono, ya que, para poder utilizar los sensores a través de la web es necesaria una API que se encargue de ello. En caso de implementarlo de forma nativa permite una mayor interacción entre todos los sensores e información del móvil, mucho más allá de lo que permite una simple web.

8.5 Adaptaciones específicas para webs, juegos

Si se adaptase *Slash and Pair*, específicamente para utilizar una plataforma como Netflix, se podría tener un control más cómodo sobre la aplicación a través del smparthone que por lo general son más usables que los mandos estándar de televisión. También se puede adaptar como un mando utilizando su pantalla táctil como botones y utilizar otros sensores como giroscopio para añadir una mejor experiencia de juego.

8.6 Reconocimiento de movimientos

Con el uso de los sensores que incorpora el *smartphone*, se puede realizar un estudio de los movimientos humanos, como por ejemplo el movimiento de llevarse el *smartphone* a la oreja para llamar, o el de guardárselo en el bolsillo y que en función de los movimientos realizados se realicen acciones correspondientes a esos movimientos.

Referencias

[1] Astrium, DigitalGlobe. (21 de Junio de 2017). www.android.com, https://www.android.com/intl/en_ca/auto/

[2] ASOCIACIÓN ESPAÑOLA DE DOMÓTICA E INMÓTICA. (21 de Junio de 2017). http://www.cedom.es. http://www.cedom.es/sobre-domotica/que-es-domotica

[3] Instituto nacional de estadística. (1 de Octubre de 2015). Encuesta sobre Equipamiento y Uso de Tecnologías de Información y Comunicación en los Hogares. Nota de prensa, http://www.ine.es/prensa/np933.pdf

[4] Instituto nacional de estadística. (3 de Octubre de 2016). Encuesta sobre Equipamiento y Uso de Tecnologías de Información y Comunicación en los Hogares. Nota de prensa, http://www.ine.es/prensa/np991.pdf

[5] Kovach, S. (8 de Diciembre de 2010): (15 de Junio de 2017). Bussinessinsider.com. Obtenido de: http://www.businessinsider.com/what-is-a-smart-tv-2010-12

[6] GSMARENA.COM. (26 de mayo de 2017) gsmarena.com. Obtenido de: http://www.gsmarena.com/apple_iphone_7-8064.php

[7] Spotify AB. (28 de mayo de 2017). spotify.com. Obtenido de https://www.spotify.com/es/connect/?utm_source=wp-picker&utm_medium=web

[8] WhatsApp Inc. (28 de mayo de 2017). whatsapp.com. Obtenido de: https://www.whatsapp.com/faq/es/web/28080003

[9] Xataka. (8 de julio de 2015). xatakamovil.com. Obtenido de: https://www.xatakamovil.com/espacio-sony/13-aplicaciones-para-conectarte-a-tu-ordenador-desde-tu-smartphone

[10] Universidad de Cádiz. (28 de mayo de 2017). www2.uca.es Obtenido de: http://www2.uca.es/serv/dafo/DAFOhelp.html

[11] Universidad Complutense de Madrid. (22 de octubre de 2008). www.fdi.ucm.es Obtenido de: https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf

[12] Universidad de Cantabria. (28 de mayo de 2017). http://www.istr.unican.es Obtenido de: http://www.istr.unican.es/asignaturas/is1/IEEE830/_esp.pdf

[13] Proyectos agiles. (28 de mayo de 2017). proyectosagiles.org. Obtenido de: https://proyectosagiles.org/desarrollo-iterativo-incremental/

[14] Boehm, B. W. (31 de mayo de 2017). Center for Systems and Software Engineering. csse.usc.edu Obtenido de:
http://csse.usc.edu/TECHRPTS/1988/usccse88-500/usccse88-500.pdf

[15] Trello, Inc. (28 de mayo de 2017). trello. Obtenido de:
https://trello.com/

[16] Pivotal Software. (4 de mayo de 2017). Spring by Pivotal. Obtenido de:
https://spring.io/guides/gs/messaging-stomp-websocket/

[17] Pivotal Software Inc. (27 de mayo de 2017). spring.io. Obtenido de:
https://projects.spring.io/spring-security/

[18] The Apache Software Foundation. (7 de junio de 2017). https://maven.apache.org/. Obtenido de:
https://maven.apache.org/what-is-maven.html

[19] Pivotal Software Inc. (28 de mayo de 2017). spring.io. Obtenido de:
http://projects.spring.io/spring-data-redis/

[20] Owen, S. (7 de junio de 2017). https://zxing.org. Obtenido de:
https://zxing.org

[21] Galdámez, P. (29 de mayo de 2017). http://web.iti.upv.es. Obtenido de:
http://web.iti.upv.es/actualidadtic/2003/07/2003-07-seguridad.pdf

[22] HACKMAGEDDON (21 de junio de 2017). http://www.hackmageddon.com. Obtenido de:
http://www.hackmageddon.com/2017/01/19/2016-cyber-attacks-statistics/

[23] Borghello, C. (5 de mayo de 2017). http://www.segu-info.com.ar. Obtenido de:
http://www.segu-info.com.ar/ataques/ataques.htm

[24] OWASP Foundation. (11 de junio de 2017). www.owasp.org. Obtenido de:
https://www.owasp.org/index.php/Cross-Site_Request
Forgery(CSRF)_Prevention_Cheat_Sheet#Encrypted_Token_Pattern

[25] US-CERT. (10 de junio de 2017). www.us-cert.gov. Obtenido de:
https://www.us-cert.gov/sites/default/files/
publications/DDoS%20Quick%20Guide.pdf

[26] Ramos, A. (1 de mayo de 2017). SECURITYBYDEFAULT.COM. Obtenido de:
http://www.securitybydefault.com/2014/02
/12-seguridad-en-redis-fortificacion.html

[27] Ramos, A. (1 de mayo de 2017). SECURITYBYDEFAULT.COM. Obtenido de:
http://www.securitybydefault.com/2014/02
/22-seguridad-en-redis-auditoria.html

[28] Redis labs. (11 de junio de 2017). redis.io. Obtenido de https://redis.io/topics/security

[29] Redis Labs. (15 de mayo de 2017). redis.io. Obtenido de: https://redis.io/commands/auth

[30] Ayanoglu, E., Y. A., & D. N. (2015). Mastering RabbitMQ. Birmingham: PACKT PUBLISHING.

[31] Pivotal Software. (1 de junio de 2017). https://www.rabbitmq.com. Obtenido de: https://www.rabbitmq.com/access-control.html

[32] Google. (05 de junio de 2017). sites.google.com. Obtenido de: https://sites.google.com/a/chromium.org/dev/Home /chromium-security/deprecating-powerful-features-on-insecure-origins

[33] CogniFit. (05 de junio de 2017). www.cognifit.com. Obtenido de: https://www.cognifit.com/es/habilidad-cognitiva/memoria-a-corto-plazo

[34] Beust, C. (05 de junio de 2017). http://beust.com/weblog/. Obtenido de: http://beust.com/weblog2/archives/000497.html

[35] Mehta, D. (5 de junio de 2017). www.quora.com. Obtenido de: https://www.quora.com /What-is-the-most-complex-screen-unlock-pattern-for-Android

[36] DENSO WAVE INCORPORATED. (5 de junio de 2017). www.qrcode.com. Obtenido de: http://www.qrcode.com/en/codes/model12.html

[37] Pivotal Software. (11 de junio de 2017). docs.spring.io. Obtenido de: https://docs.spring.io/spring-security /site/docs/current/reference/html/csrf.html

[38] The Apache Software Foundation. (7 de junio de 2017). https://maven.apache.org/. Obtenido de: https://maven.apache.org/what-is-maven.html

[39] Redis lab. (3 de abril de 2017). Redis. Obtenido de: https://redis.io/topics/introduction

[40] Redis lab. (3 de abril de 2017). Redis. Obtenido de: https://redis.io/topics/benchmarks

[41] Mozilla Foundation. (7 de junio de 2017). developer.mozilla.org. Obtenido de: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API

[42] Owen, S. (7 de junio de 2017). https://zxing.org. Obtenido de: https://zxing.org

[43] W3.CSS. (7 de junio de 2017). www.w3schools.com. Obtenido de: https://www.w3schools.com/html/html5_intro.asp

[44] András, T. (7 de junio de 2017). https://atandrastoth.co.uk. Obtenido de: https://atandrastoth.co.uk/main/pages/plugins/webcodecamjs/

[45] The Apache Software Foundation. (7 de junio de 2017). http://tomcat.apache.org/. Obtenido de: http://tomcat.apache.org/

[46] Electronic frontier foundation (21 de junio de 2017). https://certbot.eff.org. Obtenido de: https://certbot.eff.org/about/

[47] Tibbett, R. (9 de mayo de 2017). https://github.com/. Obtenido de: https://github.com/richtr/NoSleep.js

[48] Heaton, R. (7 de junio de 2017). http://robertheaton.com. Obtenido de: http://robertheaton.com/2014/03/27/how-does-https-actually-work/

[49] Terrell, R. (4 de junio de 2017). https://github.com/wwwtyro. Obtenido de: http://wwwtyro.github.io/

[50] Universidad Nacional Autónoma de México. (4 de junio de 2017). unam.mx. Obtenido de: https://revista.seguridad.unam.mx/node/2157

[51] Clarke, S. (7 de junio de 2017). Measuring API Usability. Obtenido de: http://www.drdobbs.com/windows/measuring-api-usability/184405654

[52] Berry, D. (4 de junio de 2017). The user experience. Obtenido de: https://www.ibm.com/developerworks/library/w-berry/

9 Anexo

9.1 Definiciones

En este apartado se añaden las definiciones de algunos de los términos que se van utilizando a lo largo del trabajo, se encuentran en las tablas 14:

Glosario de términos	
Término:	**Definición:**
Slash and Pair	Nombre del proyecto que consta de dos partes, Slash and Pair Desktop y Slash and Pair Mobile.
Slash and Pair Mobile	Nombre del subproyecto que captará los datos a través de una página web que posteriormente enviará al proyecto Desktop, para que los procese utilice el proyecto Desktop.
Slash and Pair Desktop	Nombre del subproyecto que se analiza en este documento. Este proyecto permitirá visualizar una aplicación manejable desde el navegador de un *smartphone* a través del proyecto Mobile.
HTML	*HyperText Markup Language*. Es el lenguaje marcado para la elaboración de páginas web. Es un estándar que sirve de referencia para crear contenido web.
Spring	Spring es un *framework* para el desarrollo de aplicaciones de código abierto. Comprende una serie de módulos que individualmente aportan un rango de servicios.
JSON	Acrónimo de *JavaScript Object Notation*, es un formato de texto ligero para el intercambio de datos.
Entorno WEB	Entorno web hace referencia a un ambiente de desarrollo o ejecución de programas o servicios en el marco de la web. A estos entornos se accede vía web, a través de Internet. El termino refiere a los entornos que son visibles a través de un navegador.
Hardware	Conjunto de elementos físicas tangibles de un sistema informático, componentes eléctricos, electrónicos o mecánicos.
Software	Soporte lógico de un sistema informático que permiten la realización de tareas específicas. Esto incluye generalmente las aplicaciones informáticas.
LocalHost	Nombre reservado que tienen todos los dispositivos independientemente que dispongan tarjeta de red de ethernet. Se traduce como la siguiente dirección IP: “127.0.0.1”
Host	Ordenador o otro dispositivo conectado a una red que provee y utiliza servicios de ella.
HTTP	Abreviatura de las siglas *Hypertext Transfer Protocol*. Protocolo de comunicación que permite las transferencias de información en la *World Wide Web*.

Petición HTTP	HTTP contiene una serie predefinida de métodos de petición que pueden utilizarse para la comunicación mediante Internet, los indispensables para *Slash and Pair* son los siguientes: GET: Pide una representación del recurso especificado. POST: Envía los datos para que sean procesados por el recurso identificado.
Código QR	Un código QR *(Quick Response Code)* es un módulo que permite almacenar información en una matriz de puntos.
Pairing	*Pairing* se refiere a la acción de realizar una sincronización de sesión entre dos instancias del navegador distintas pertenecientes al mismo usuario a través de un código personal el cual solo podrá disponer el usuario que lo solicite.
Giroscopio	Dispositivo mecánico que sirve para medir, mantener, o cambiar la orientación en el espacio. En el caso de los *smartphones*, permite medir la orientación del dispositivo.
iOS	Sistema operativo móvil de Apple Inc, desarrollado exclusivamente para el iPhone.
Android	Sistema operativo móvil basado en el *kernel* de Linux.
RabbitMQ	Software de negociación de mensajes de código abierto. Se puede catalogar como *middleware* de mensajería e implementa el estándar AMQP *(Advanced Message Queuing Protocol)*. El servidor Rabbit está escrito en Erlang.
Redis	Motor de base de datos en memoria, basado en el almacenamiento en tablas de *hashes* (clave/valor) que puede ser usada como una base de datos durable.
WebSockets	Tecnologia que proporciona un canal de comunicación bidireccional y *full-duplex* (Envía y recibe mensajes de forma simultánea.
SockJS	Librería de JavaScript para navegadores que provee de un objeto WebSocket.
Maven	Herramienta de software para la gestión y construcción de proyectos Java. Tiene un modelo de configuración basado en el formato XML. Se encarga de describir el proyecto de software a construir, sus dependencias de otros módulos, componentes externos y el orden de construcción de los elementos.
API	API *(Application Programming Interface)* es un conjunto de subrutinas, funciones y procedimientos que ofrece cierta librería para ser utilizada por otro software como una capa de abstracción.
Script	Programa simple, almacenado como un archivo de texto plano, que permite interactuar con el sistema operativo o el usuario.

Tabla 14: Definiciones