UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

TFG del Grado en Ingeniería Informática

UBUassistant

Presentado por Carlos González Calatrava
en Universidad de Burgos — 26 de junio de 2018
Tutor: Pedro Renedo Fernández

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

D. Pedro Renedo Fernández, profesor del departamento de Ingeniería Civil, área de lenguajes y sistemas informáticos.

Expone:

Que el alumno D. Carlos González Calatrava, con DNI 71296090T, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado UBUassistant.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 26 de junio de 2018

Vº. Bº. del Tutor:

D. Pedro Renedo Fernández

Resumen

UBUassistant nació el año pasado a través de un TFG bajo la premisa de ayudar a los usuarios cuando acceden por primera vez a una página web, haciendo que la interacción con la misma sea más amigable y mejorando considerablemente la experiencia del usuario.

Actualmente vivimos en una sociedad en la que el gran porcentaje de las personas llevan consigo un *smartphone* con el cual pueden acceder a una gran cantidad de información al instante. El problema radica en que muchas ocasiones, ciertas páginas web no están adaptadas de forma correcta a estos dispositivos, por lo que hace que dificulta la interacción con ellas.

De esta idea surge esta segunda versión de UBUassistant, la cual permite que el asistente sea fácilmente adaptable a otras plataformas como HTML5, iOS, Android, etc.

Descriptores

Asistente virtual, orientación en portales web, razonamiento basado en casos, aprendizaje tutelado, aplicación Android.

Abstract

UBUassistant was born last year in a TFG with the idea to help the users who starts to use a website for first time. With this tool, the user obtains a friendly experience in UBU's website.

Nowadays, we live in a society in which many people carry a smartphone is possible to use it to access to a lot of information instantly. Sometimes, this websites aren't adapted correctly to use it in smartphones so it's difficult to interact with them.

From this idea comes this UBUassistant second version, which allows that the assistant can be adapted to HTML5, iOS, Android, etc.

Keywords

Virtual assistant, orientation in web portals, case based reasoning, tutored learning, Android application.

Índice general

Índice de figuras

Introducción

Para ciertas personas, poder encontrar información desde un *smartphone* les supone una tarea bastante tediosa y complicada, lo cuál les lleva muchas veces a la deseperación y finalmente no acaban encontrando aquello que deseaban encontrar.

La página web de la Universidad de Burgos es una de las mejores adaptadas a los diferentes dispositivos con los que se puede acceder a la misma. Pero como la gran mayoría de páginas web de universidades, esta contiene mucha información, como es lógico, haciendo que para ciertas personas pueda ser algo complicado navegar a través de ella. Esto se debe que a pesar de que la página esté estructurada en cinco menús, la cantidad de submenús que se encuentran dentro de ellos pueden llegar a confundir al usuario cuando a accedido a varios de ellos desde estos dispositivos, haciéndole dudar como ha conseguido entrar hasta el mismo para futuras consultas.

El método propuesto para facilitar la búsqueda de estos usuarios es una aplicación para dispositivos Android, ya que en España el 87,1 % de los dispositivos son Android [1]. De esta forma se pretende facilitar el uso de la web de la Universidad de Burgos al mayor porcentaje posible de personas.

El asistente es el encargado de buscar una respuesta al texto introducido por el usuario dentro de la aplicación. El usuario no tiene que introducir palabras clave para que el asistente funcione correctamente, sino que puede emplear lenguaje natural, ya que los algoritmos del asistente es el encargado de analizar el texto introducido y poder encontrar así la respuesta más adecuada para el usuario.

1.1. Estructura de la memoria

La memoria tiene la siguiente división en apartados:

- **Introducción:** en este apartado se realiza una descripción de una manera breve del problema que se intenta resolver y la solución otorgada. Además incluye subapartados con la estructura de la memoria y el listado de materiales adjuntos.
- **Objetivos del proyecto:** sección donde se explican los objetivos de desarrollar un proyecto de estas características.
- **Conceptos teóricos:** capítulo en el que se abordan los conceptos teóricos necesarios para comprender el resultado final del proyecto.
- **Técnicas y herramientas:** en esta sección se describen las herramientas y las técnicas que se han utilizado para el desarrollo y gestión del proceso del proyecto.
- **Aspectos relevantes del desarrollo:** apartado donde se tratan aquellos aspectos que se consideran destacados en el desarrollo del proyecto.
- **Trabajos relacionados:** capitulo que expone y describe aquellos trabajos que están relacionados con la temática de asistente virtual.
- **Conclusiones y líneas de trabajo futuras:** sección que explica las conclusiones obtenidas tras la realización del proyecto y la funcionalidad que es posible añadir en el futuro.

Además, se proporcionan los siguientes anexos:

- **Plan del proyecto software:** capítulo donde se expone planificación temporal del proyecto y su viabilidad.
- **Especificación de requisitos:** en este apartado se desarrollan los objetivos del software y la especificación de requisitos.
- **Especificación de diseño:** sección que describe el diseño de datos, el diseño procedimental y el diseño arquitectónico.
- **Documentación técnica de programación:** en este capítulo se explica todo lo relacionado con la programación, la estructura de directorios, el manual del programador y las pruebas realizadas.
- **Documentación de usuario:** apartado que realiza un explicación sobre los requisitos de usuarios, la instalación y proporciona un manual de usuario.

1.2. Materiales adjuntos

Los materiales que se adjuntan con la memoria son:

- Aplicación Java servlet UBUassistant.

- Aplicación Android cliente UBUassistant.
- Máquina virtual Ubuntu configurada de la misma manera que está configurado en Microsoft Azure.
- JavaDoc.

Además, los siguientes recursos están accesibles a través de internet:

- Repositorio del proyecto. [2]

Objetivos del proyecto

En este apartado se explica cuáles han sido los objetivos que se han perseguido durante el proyecto.

2.1. Objetivos software

- A nivel de usuario de la aplicación:
  - Disponer de la aplicación en dispositivos móviles que utilicen sistema operativo Android.
  - Dentro de la aplicación Android, poder utilizar la voz como herramienta para poder introducir el texto de las búsquedas.
  - Ampliar el número máximo de vocablos que se utilizan para encontrar los resultados dentro del algoritmo CBR.
- A nivel de gestor y desde el punto de vista tecnológico:
  - Analizar diferentes frameworks de Razonamiento Basado en casos, tal y como se sugirió en la defensa del TFG anterior. En función del resultado del análisis, implementarlo o no.
  - Independizar la capa de presentación de la capa de aplicación de tal forma que se permita implementar el sistema cliente a diferentes dispositivos y tecnologías como HTML5, Android o iOS.
  - Minimizar los posibles ataques de denegación de servicio.
  - Mantener la imagen corporativa.

2.2. Objetivos técnicos

- Aplicar Scrum, en la medida de lo posible, como metodología de desarrollo ágil.

- Realizar la aplicación siguiendo el concepto de Modelo Vista Controlador, separando la interfaz de usuario, el motor de la aplicación y los datos.
- Obtener las respuestas del motor de la aplicación empleando un estándar, haciendo así que la aplicación sea más modular y multiplataforma.
- Acceder a una base de datos MySQL mediante Hibernate y JDBC.
- Servirse de GitHub como sistema de control de versiones.
- Adquirir conocimientos para el desarrollo de aplicaciones en plataforma Android.
- Utilizar una máquina virtual de Microsoft Azure para alojar el servidor de la aplicación, para que el asistente sea accesible con cualquier tipo de conexión a Internet.

2.3. Objetivos personales

- Emplear la mayor cantidad posible de conocimientos adquiridos durante la carrera.
- Ampliar los conocimientos adquiridos durante la carrera utilizando nuevos lenguajes de programación (Android).
- Conseguir una aplicación que facilite las tareas de búsqueda en la página de la Universidad de Burgos.

Conceptos teóricos

Para entender mejor el funcionamiento de este TFG, es necesario conocer el funcionamiento del algoritmo de Razonamiento Basado en Casos (*CBR*).

Un sistema CBR es aquel que trata de simular el mismo comportamiento que realizaría un ser humano cuando tiene ante si una serie de problemas para resolver.

Estos sistemas, al igual que haría una persona, basan sus decisiones en experiencias previas para intentar obtener la mejor respuesta al problema que se ha planteado [3]. Como se sugirió en la defensa del proyecto que se está mejorando, se han estudiado otros framework de razonamiento basado en casos, para considerar un posible cambio. Este estudio de otros frameworks, lo podemos consultar en la sección 5.5.

En el caso de UBUassistant, se está empleando un algoritmo que utiliza una serie de casos base. A partir de estos casos, el sistema puede ir aprendiendo con las diferentes recomendaciones ofrecidas por los usuarios.

Estos conocimientos base, a nivel de algoritmo son conocidos como casos. Los casos empleados para este algoritmo, están formados por un máximo de siete palabras que definen cada uno de los posibles casos que se pueden llegar a dar. El algoritmo, por cada uno de estos casos, tiene asociado una solución, la cual nos devuelve la dirección URL de un apartado dentro de la página de la Universidad de Burgos.

Lo que realiza el algoritmo para poder así encontrar la mejor o las mejores soluciones, es que por cada palabra que ha introducido el usuario, genera una colección con los posibles resultados. En el caso de que haya palabras que coincidan dentro de un mismo caso, estas se unen formando una respuesta única para ambos vocablos. Finalmente, todas las respuestas se combinan, pudiendo así tener tres posibles soluciones:

- **Una única solución:** esto sucede cuando las diferentes palabras del

usuario están dentro de un mismo caso, por lo que se obtiene un único resultado, el cual será mostrado al usuario.

- **Multiples soluciones:** esto ocurre cuando los vocablos que se han introducido pertenecen a diferentes casos, por los que todas las posibles soluciones son mostradas al usuario, para que este elija aquella que mejor se adapta a sus necesidades.
- **Sin solución:** puede ocurrir que el usuario introduzca una serie de palabras que no coincida con ninguno de los casos que posee el algoritmo. En este caso, al usuario se le va a mostrar un mensaje advirtiéndole que no se han encontrado una respuesta que coincida con su solicitud. A parte de esto, se le va a mostrar tres recomendaciones para intentar ayudarle en todo lo posible con la búsqueda.

Finalmente, con el propósito de que el algoritmo vaya creciendo y obteniendo más conocimientos, en el caso de que no haya podido encontrar respuestas, se le muestra tres sugerencias como se ha comentado con anterioridad y en el caso de que se haya seleccionado una de las respuestas correctas, esta se añade con los vocablos introducidos por el usuario dentro de una tabla de la base de datos. Para evitar que se produzca un aprendizaje erróneo por parte del algoritmo, estos resultado que se van añadiendo a la tabla, tienen que ser antes verificados por un administrador de la aplicación.

Para el desarrollo de este algoritmo, se ha empleado *jCOLIBRI 2* [4].

Figura 3.1: Logo de *jCOLIBRI2*

Técnicas y herramientas

4.1. Patrones de diseño

Modelo Vista Controlador

Modelo vista controlador (MVC) es un patrón arquitectónico que nos ayuda a separar los datos, la lógica de negocio y la interfaz de usuario [5].

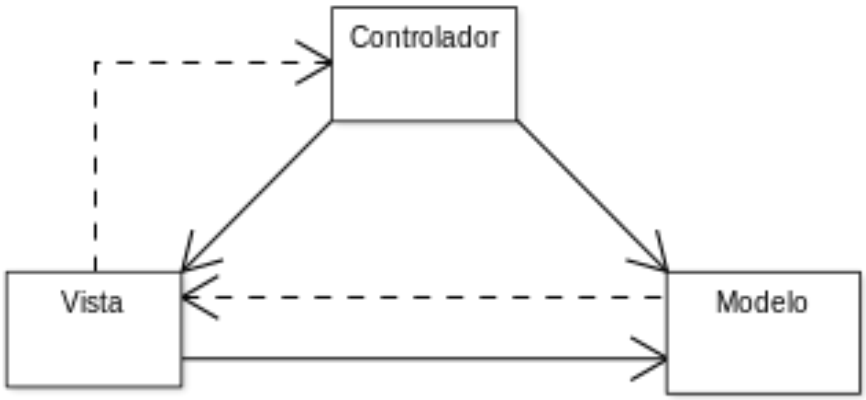

Figura 4.2: Diagrama Modelo Vista Controlador

El significado de cada uno de los tres componentes de este modelo son:

- **Modelo:** representa los datos que utiliza la aplicación.
- **Vista:** Muestra al usuario la información.
- **Controlador:** Controla las interacciones con el usuario.

Estos conceptos pueden ser aplicados a nuestra aplicación Android, obteniendo así un Modelo Vista Controlador muy bien diferenciado y estructurado. Esto lo podemos ver claramente en el siguiente diagrama.

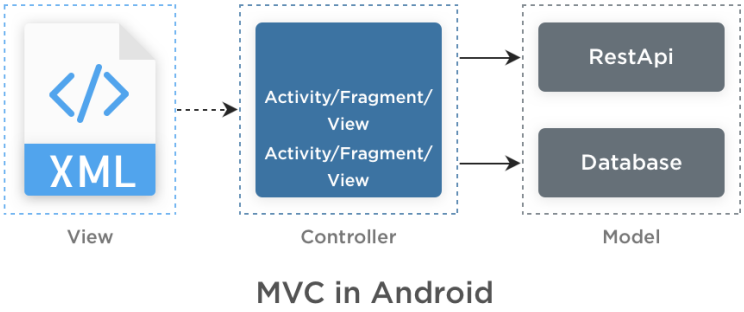

Figura 4.3: Modelo Vista Controlador en Android

4.2. Jersey REST

- Herramientas consideradas: Sockets en Java y Jersey REST.
- Herramienta elegida: API REST.

REST es una interfaz que se encarga de conectar sistemas utilizando conexiones de tipo HTTP y realizar operaciones sobre los datos intercambiado con ficheros de tipo XML o JSON entre otros. Una de las principales características de REST es que puede realizar cuatro operaciones: **POST** (*crear*), **GET**(*leer y consultar*), **PUT**(*editar*) y **DELETE** (*eliminar*) [6].

Se ha decidido utilizar REST como tecnología para conectar el cliente con el servidor, debido a que de esta forma nos garantizamos la adaptabilidad de la conexión del servidor con otras tecnologías y lenguajes de programación.

4.3. Control de versiones

- Herramientas consideradas: Git y Mercurial.
- Herramienta elegida: Git.

Git es un software de control de versiones pensado para proyectos que poseen una gran cantidad de ficheros fuente cuyo propósito es registrar todos los cambios efectuados en dichos ficheros. Además es un software de código libre distribuido bajo licencia *GPL GNU* [7].

La decisión de utilizar *Git* se ha basado en que se ha trabajado con anterioridad con la herramienta, teniendo ya unos conocimientos base sobre ella.

4.4. Hosting del repositorio

- Plataformas consideradas: Bitbucket y Github.
- Plataforma elegida: Github.

GitHub es una plataforma que es utilizada para alojar proyectos, los cuales emplean Git como sistema de control de versiones.

GitHub ha sido elegido frente a BitBucket debido a que es una plataforma que se ha ido utilizando en diferentes asignaturas como Gestión de Proyectos.

4.5. Gestión del repositorio

- Herramientas consideradas: GitKraken y GitDesktop.
- Herramienta elegida: GitKraken.

GitKraken es una aplicación que sirve para gestionar de una forma más sencilla nuestro repositorio de GitHub. Es una herramienta multiplataforma compatible con Windows, Mac y Linux.

La decisión de usar GitKraken frente a GitDesktop está basada en la experiencia personal, ya que previamente había trabajado con ambas herramientas. A parte de esto, la decisión de usar GitKraken está fundamentada en su compatibilidad con sistemas Linux.

4.6. Sistema Operativo

- Sistemas considerados: Windows 10, Ubuntu y Linux Mint.
- Sistema elegido: Linux Mint.

Linux Mint es un sistema operativo Linux que utiliza un núcleo de sistema basado en Debian y Ubuntu.

En un primer momento se planteó si usar Windows o Linux como sistema operativo. Finalmente se decidió usar Linux ya que se consideró que la gestión de recursos la hace de forma más eficiente y emplea menos recursos. Una vez tomada la decisión de usar Linux, se barajaron dos opciones, si usar Ubuntu o Linux Mint. La decisión se tomo en base a la experiencia personal, ya que se había trabajado con anterioridad con ambos sistemas operativos.

4.7. Entorno de desarrollo integrado (IDE) para JAVA

- IDE's considerados: IntelliJ, NetBeans y Eclipse.
- IDE elegidos: Eclipse.

Eclipse es un software que está formado por diferentes herramientas de programación de código abierto multiplataforma y se encuentra bajo la licencia de software libre *Eclipse Public License*. Para el desarrollo de este proceso, se ha empleado la versión EE, la cual permite realizar páginas web basadas en Java en formato JSP [8].

De todos los softwares valorados para emplear como IDE se decidió utilizar Eclipse ya que es una herramienta que resulta familiar debido a que ha sido utilizada en todas aquellas asignaturas del Grado en el que se ha utilizado el lenguaje de programación JAVA.

4.8. Entorno de desarrollo integrado (IDE) para Android

- IDE's considerados: Android Studio, AIDE y Eclipse.
- IDE elegidos: Android Studio.

Android Studio es el IDE oficial para el desarrollo de aplicaciones para Android. Este IDE es el sucesor de Eclipse como IDE oficial de desarrollo, funciona gratuitamente bajo *Licencia Apache 2.0* y está disponible para Windows, macOS y Linux [9].

Esta herramienta ha sido elegida principalmente por que es la herramienta oficial que nos ofrece Google y trae consigo todo lo necesario para desarrollar la aplicación, sin tener que descargar complementos para el software, como pasa en el caso de Eclipse.

4.9. Alojamiento del servidor

- IDE's considerados: Microsoft Azure, Amazon AWS, Google Cloud y servidor local.
- IDE elegidos: Microsoft Azure.

Microsoft Azure es un servicio en la nube que se aloja en los Data Centers de Microsoft. Esta plataforma posee diferentes servicios para aplicaciones, desde procesamiento de datos hasta servicios de comunicación segura [10].

De todas las plataformas estudiadas, se ha decidido utilizar Microsoft Azure, debido a que posee una suscripción gratuita para estudiante, la cual ofrece 100$ de crédito para gastar en servicios de forma gratuita. Debido a que la cuenta de correo electrónico de la Universidad de Burgos es de Microsoft, se ha tenido derecho a emplear este tipo de suscripción.

4.10. Documentación

- Herramientas consideradas: TexMaker, TexStudio, ShareLaTex y OpenOffice.
- Herramienta elegida: TexMaker.

Entre realizar la documentación con LaTex u OpenOffice se decidió emplear LaTex, ya que los resultados que se obtienen son una documentación obtenida con una gran calidad tipográfica.

LaTex es un sistema empleado para la composición de textos escritos con una alta calidad tipográfica tales como artículos académicos, tesis y libros técnicos. Es un software libre bajo la licencia *LPPL* [11].

TexStudio es un editor para LaTex de código abierto y multiplataforma que es una evolución del IDE TexMaker. Este IDE esta bajo la licencia *GNU GPL* [12].

La elección de este editor es que es offline, por lo que no dependemos de una conexión a Internet, y que es una evolución de TexMaker, por lo que funcionalidades añadidas respecto a este IDE.

Aspectos relevantes del desarrollo del proyecto

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto, comentados por los autores del mismo. Aquí se incluyen las diferentes explicaciones tanto del diseño como de la implementación.

5.1. Inicio y fase de análisis

Este proyecto es la consecución del TFG UBUassistant de Daniel Santidrián Alonso en el cual durante la defensa del mismo, surgieron diferentes mejoras para el mismo [13].

Tras valorar las diferentes mejoras con el tutor del proyecto, se decidió continuar con él, debido a que me pareció un proyecto muy interesante tanto a nivel de usuario como a nivel de programación.

5.2. Metodologías aplicadas

En el desarrollo de este proyecto se ha intentado emplear en la medida de los posible la metodología ágil de Scrum.

Debido a que el tamaño del proyecto es pequeño, no se ha podido seguir de forma estricta todas las pautas de la metodología, como las reuniones diarias con todo el equipo. Las pautas que se han seguido durante el proyecto han sido:

- Desarrollo incremental del proyecto mediante sprints.
- Spints de duración semanal en vez de diaria por el tamaño del proyecto.
- Al finalizar el sprint, reuniones para evaluar el proyecto y plantear los pasos a seguir en el siguiente sprint.

Para el desarrollo del servidor del asistente, se realizaron diferentes pruebas de ensayo error hasta que se obtuvieron los datos a través del estandar JSON de la forma deseada.

Al final de cada sesión de trabajo se ha intentado realizar un *commit* en el repositorio del proyecto para mantener así un control sobre los diferentes pasos que se han ido dando.

5.3. Formación

Durante las primeras fases del proyecto, fue necesario aprender el funcionamiento del TFG de Daniel Santidrián Arce. Para esto, se consultó la documentación de dicho proyecto, así como el código fuente del mismo [13].

A parte de esto, se realizó una investigación sobre el funcionamiento genérico de un Sistema de Razonamiento Basado en Casos (*CBR*) como de la utilidad empleada en la versión anterior del TFG, *jCOLIBRI*. Para ello se consultaron los mismos artículos empleados para este propósito por el autor del anterior TFG.

- Razonamiento Basado en Casos: Una visión general (Laura Lozano y Javier Fernández) [14].
- Tutorial jCOLIBRI (J. Recio García, B. Díaz Aguado y P. González Calero) [15].

Una vez entendido el funcionamiento de los algoritmos anteriores, se tuvo que adquirir conocimientos sobre el funcionamiento de API REST, ya que es la técnica utilizada para realizar las peticiones al servidor y obtener las respuestas en forma de fichero JSON. Para ello, se consultó un artículo sobre API REST y un tutorial base sobre JAVA REST, ya que es la librería empleada en el servidor JAVA.

- API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos [6].
- Ejemplo de API REST en Java con JAX-RS y Spring Boot [16].

En el momento en el que se empezaron a dar los primeros pasos sobre la aplicación Android, se tuvo que empezar a adquirir conocimientos sobre este tema. Para ello, se decidió consultar las diferentes guías, tutoriales y documentación que ofrece la propia página de Android Studio, ya que viene en diferentes idiomas y está muy completa [17].

Una vez que las primeras versiones de la aplicación fueron funcionando, se decidió exportar el servidor de JAVA a una máquina virtual en la nube, de esa

forma se puede utilizar la aplicación desde cualquier lugar. Para ello, se decidió emplear una Máquina Virtual de Azure. Para aprender el funcionamiento de estas máquinas virtuales, como las diferentes configuraciones, se siguieron las diferentes guías que se pueden encontrar en la página de Microsoft [18].

Además de todo lo mencionado anteriormente, se fueron consultando diferentes foros de *Stack Overflow* [19] para consultar las diferentes dudas que iban surgiendo a lo largo del proyecto a nivel de programación.

5.4. Adaptación del servidor

Para que el asistente fuera funcional y fácilmente adaptable a otras plataformas, había que modificar la aplicación heredada del TFG anterior, por lo que para ello se realizaron una serie de modificaciones del código.

- Separar la capa de usuario de la de aplicación en los ficheros JSP.
- Obtener las respuestas siguiendo un patrón.

En un primer momento se planteó usar un servidor creado desde cero con JAVA, el cuál iba a estar junto con el resto de ficheros del asistente. Una vez empezado con esta tarea, se empezaron a ver diferentes complicaciones que podrían ir surgiendo, siendo principalmente la conexión con la aplicación Android. Tras ver estos problemas, se optó por buscar más información de como poder hacer conexiones cliente-servidor de forma efectiva. Finalmente se optó por usar API REST para realizar la conexión, ya que nos permitía hacer conexiones tipo *HTTP/HTTPS* de forma sencilla contra un servidor JAVA Tomcat.

A continuación, lo que se realizó fue crear nuestros propios ficheros JSP para el servidor Tomcat. Para ello se utilizó como base los ficheros ya existentes, debido a que cierta parte de ellos se pudo reutilizar para hacer nuestros propios ficheros y obtener así una comunicación correcta con el algoritmo CBR. Este código que se pudo reutilizar, más las diferentes funciones y clases creadas con API REST, nos permitió obtener respuestas por parte del servidor.

Una vez que teníamos el servidor con las respuestas, hubo que cambiar la forma en las que la daba este. Esto se debió a que el servidor obtenía la respuesta en cadenas de texto en formato HTML, para que así se mostrara de forma correcta en una web. Como lo que se iba buscando es que las respuestas fueran fácilmente utilizables por diferentes lenguajes de programación, se decidió que las respuestas tendrían que darse en forma de fichero JSON. La decisión de emplear este tipo de ficheros, es que la gran mayoría de lenguajes orientados a objetos tienen diferentes librerías para poder leer y utilizar este tipo de ficheros.

Para poder hacer esto, se tuvo que modificar a parte de los ficheros JSP, diferentes funciones de JAVA, para poder generar así un fichero JSON que tuviera como mínimo de forma clara el mensaje que da de respuesta. A parte, en los diferentes casos en los que va acompañado de una respuesta con los diferentes enlaces a la página web de la Universidad de Burgos, se cambió para que también esté de forma clara tanto el nombre del apartado que da como respuesta como la dirección URL. Hasta que puedo obtener la respuesta de forma deseada, se realizaron varios intentos de prueba error, utilizando como visor de las respuestas, el navegador web *Mozilla Firefox*, ya que nos muestra de una forma muy intuitiva los datos de un fichero JSON.

5.5. Cambio de framework de razonamiento basado en casos

Una de las propuestas en la defensa del TFG del año pasado fue la búsqueda de un framework de razonamiento basado en casos que fuera más moderno. Debido a este comentario, en las primeras fases del proyecto, se estudiaron los diferentes frameworks de razonamiento basado en casos que son de uso gratuito [20].

Los frameworks estudiados fueron myCBR, jCOLIBRI, CASPIAN, CAT-CBR y CBR Tools. La principal diferencia entre estos frameworks es el año de la última versión de lanzamiento. jCOLIBRI fue el primer framework que se consultó. La última versión que posee es la v2.1, que fue publicada en Septiembre de 2008 [4]. El siguiente framework que se consultó fue CASPIAN, siendo la última versión de dicho framework de 1995 [21]. Después se consultó CBR Tools. Este framework tiene diferentes versiones, siendo la última la v2.4, publicada en Febrero del 2004 [22]. Luego se consultó CAT-CBR. De este framework se pudo saber que su última versión data del año 2002 [23]. Finalmente, se consultó myCBR. Este framework es el más moderno de todos los consultado, además con bastante diferencia, ya que su última versión esta fechada en Mayo de 2015 [24].

Con esta información, automáticamente se descartaron los frameworks que son más antiguos que el empleado en la versión anterior del proyecto. Esta decisión, nos dejo solamente una opción a investigar, que es el framework myCBR.

Antes de empezar a estudiar el funcionamiento de dicho framework, se dedicó el tiempo necesario para la comprensión del funcionamiento del framework jCOLIBRI. Este proceso lo consideramos importante, ya que es un paso muy importante para poder hacer una valoración final del posible cambio.

Después de haber entendido tanto el funcionamiento de jColibri, como la forma de implementación en la versión anterior del proyecto, se comenzó con la

lectura y comprensión de la documentación de myCBR. Durante este proceso, se pudo encontrar diferencias tanto de implementación como de funcionamiento con jCOLIBRI. La principal diferencia que se encontró fue que jCOLIBRI almacena el caso base en una base de datos, siendo la más usual MySQL; mientras que myCBR utiliza un fichero en extensión XML para almacenarlo. Otra de las diferencias, es que la gestión del caso base en jCOLIBRI se puede realizar con los diferentes conectores y aplicaciones para base de datos que existen, mientras que en myCBR hay que utilizar su propia aplicación. Otra de las diferencias, y la que me parece más considerable, es que jCOLIBRI incluye un sistema que te permite la valoración de los diferentes resultados obtenidos, permitiendo que el sistema vaya mejorando con el tiempo, con múltiples usos realizados por los usuarios. En cambio myCBR no permite esto, haciendo que el buen funcionamiento de nuestro sistema dependa al 100 % de como hemos organizado los datos.

Con todo esto, se realizó una valoración de si realizar o no el cambio de framework dentro de este proyecto. Tras una valoración de todo lo comentado con anterioridad junto con el tutor se llegó a la determinación de no realizar el cambio del framework. Esta decisión está basada principalmente en dos motivos:

- **Gestión del aprendizaje:** con myCBR no tenemos una forma de permitir que nuestro algoritmo vaya aprendiendo y por lo tanto que cada vez vaya dando mejores resultados, mientras que lo que se busca es todo lo contrario, que con el uso nuestro algoritmo se vaya refinando.
- **Almacenamiento de los casos:** en myCBR el almacenaje de los diferentes casos se realiza a través de un fichero XML, mientras que en jCOLIBRI se realiza a través de una base de datos SQL. Se consideró que una base de datos SQL es mucho más fácil de manejar con un lenguaje de programación a parte de las mejoras de rendimiento que tiene frente a un fichero XML.

5.6. Aplicación Android

Una vez que el servidor JAVA estaba creado, se empezó por la construcción de la aplicación Android. Para ello se decidió seguir el esquema API-REST.

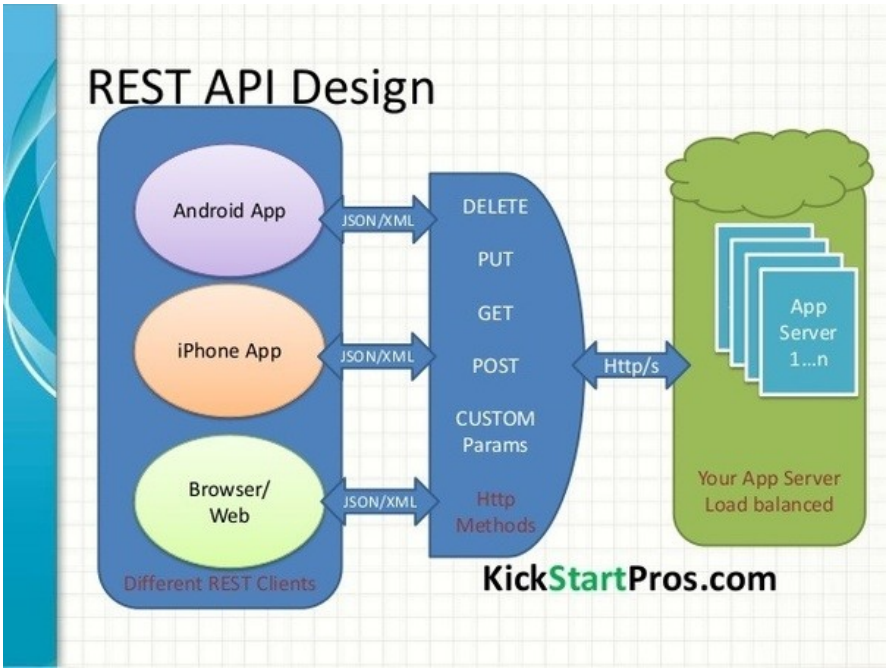

Figura 5.4: Esquema API-REST

Para ello, lo primero que se hizo fue conectar la aplicación al servidor a través de una conexión *HTTP*, obteniendo como respuesta el fichero JSON desde el servidor. Esta parte pertenece a la sección Controlador del Modelo Vista Controlador.

Una vez que ya se tenía la respuesta en nuestra aplicación Android, se empezó a diseñar la forma en la que los datos iban a ser mostrados por la aplicación. En una primera versión se decidió utilizar los *ListView* de Android. Finalmente esto fue descartado, debido a que la apariencia que da no se consideró muy adecuada para una aplicación de este estilo.

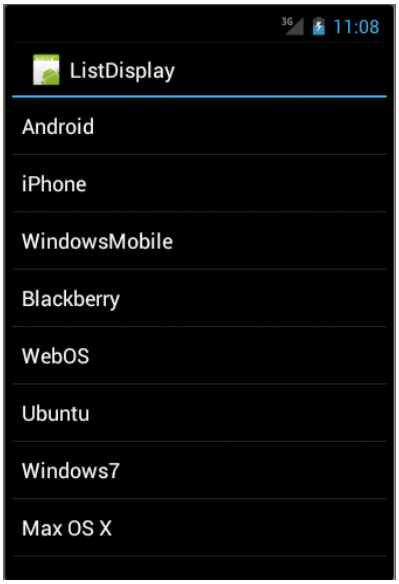

Figura 5.5: Ejemplo de *ListView*

Al final se optó por utilizar los *RecyclerView*, ya que da un aspecto más apropiado a la aplicación. Este tipo de vista es utilizado por diferentes aplicaciones de chat, tales como *Whatsapp* o *Telegram*. Para diferenciar las mensajes enviados por el usuario, de las respuestas del asistente, se optó por seguir el mismo esquema utilizado por el TFG anterior. Los mensajes del usuario van dentro de un recuadro de color blanco con el texto en negro, mientas que las respuestas del asistente van en un recuadro azul, con el texto en blanco. De esta forma, se intenta en mantener todo lo posible la misma imagen que en el TFG anterior.

5.7. Base de datos

Se decidió seguir utilizando la misma base de datos que en el proyecto heredado. Esto se debe a que a pesar de haber modificado el servidor para obtener respuestas de forma más clara, la conexión con la base de datos no se tuvo que modificar, por lo que se sigue empleando *Hibernate* y *JDBC* para conectarse a la base de datos MySQL.

Esta base de datos está formada por un total de siete tablas.

- **casedescription**: esta tabla es la que contiene las diferentes descripciones de los casos. Está compuesta por siete palabras clave junto a la categoría a las que pertenecen.

- **casesolution**: dentro de esta tabla se encuentran las diferentes soluciones para cada uno de los casos.
- **saludos**: tabla en la cual se incluyen los saludos más comunes que pueden ser introducidos por el usuario junto a su respuesta.
- **frases**: en esta tabla nos encontramos con las diferentes frases de respuesta que da el asistente cuando le preguntamos algo.
- **aprendizaje**: esta tabla contiene las palabras que tiene pendiente por aprender el algoritmo junto con la respuesta que tiene que dar.
- **logger**: tabla la cual contiene el log de cada usuario, junto a las palabras que se han buscado, la fecha, la respuesta, la categoría de la búsqueda, el número de búsquedas realizadas, el número de votos y la votación total.
- **administradores**: el contenido de esta tabla son los diferentes nombres de usuarios que tienen derecho de administración, así como la correspondiente contraseña.

5.8. Problemas

Servidor Tomcat

En el momento en el que se intentó montar nuestro propio servidor local de Tomcat nos encontramos con un problema a la hora de importar nuestro proyecto en formato *WAR*. Con otros ejemplos de proyectos descargados en el mismo formato, este mensaje de error no aparecía.

Finalmente y tras consultar diferentes foros, se descubrió que el problema radica en que Tomcat, por defecto trae que el tamaño máximo de las aplicaciones sean de 50MB. Esto se puede solucionar modificando el fichero *web.xml* dentro de la ruta *[CARPETA TOMCAT]/webapps/manager/WEB-INF*. Dentro de este fichero, hay que modificar las siguientes líneas:

```
<max-file-size>52428800</max-file-size>
<max-request-size>52428800</max-request-size>
```

El valor que se modifica va dado en *bytes*, por lo que el valor que introduzcamos, también tiene que serlo.

Hibernate

Tras hacer las primeras modificaciones del código para que la respuesta del servidor se diera a través de API REST, se realizaron diferentes pruebas. Al inicio de estas pruebas, se obtenía un error, el cual no nos dejaba conectar contra la base de datos. Tras realizar diferentes investigaciones durante unos días en diferentes foros, dimos con la solución.

La solución consistía en añadirle un parámetro al fichero de configuración de *Hibernate*, *hibernate.cfg.xml*. El parámetro que se añadió al fichero fue:

```
<property name="javax.persistence.validation.mode">none</property>
```

Máquina virtual de Microsoft Azure

Una vez que la aplicación tuvo bastante funcionalidades de las deseadas y en una de las reuniones semanales con el tutor, nos percatamos en la dificultad de hacer funcionar la aplicación conectada al servidor local creado dentro de la red de la propia Universidad de Burgos. Raíz de esto, se optó por montar una máquina virtual de Microsoft Azure.

El primer problema que se tuvo, es que una vez creada la máquina virtual con Windows 10, no se podía acceder a ella a través de Escritorio Remoto. Tras recorrer diferentes foros, finalmente se descubrió que para poder acceder a la máquina virtual hay que generar primero el *DNS* para poder acceder a ella.

Después de solucionar este problema, y poder acceder a la máquina virtual, se procedió a la instalación de las herramientas necesarias para hacer que el servidor funcionara correctamente. Una vez instalado todo, se intentó conectar nuestra aplicación a dicho servidor, para así poder utilizarla desde cualquier localización y conexión a Internet. El problema surge en este apartado, ya que no conseguíamos conectarnos al servidor.

Tras realizar varias pruebas con los firewall de Microsoft Azure y Windows 10, decidimos hacer una prueba montando una máquina virtual con Ubuntu. Al igual que con anterioridad, nos conectamos a la máquina virtual, esta vez a través de *SSH*, instalamos todo lo necesario y configuramos el firewall correctamente. Después de todo esto, conectamos la aplicación y esta funciona correctamente, por lo que finalmente se quedó con Ubuntu como sistema operativo del servidor.

5.9. Publicación

El servidor de la aplicación se encuentra montado en una máquina virtual de Microsoft Azure como ya se ha comentado en la subsección 5.8.

Dicha máquina virtual monta como sistema operativo Ubuntu Server. Como aplicaciones para hacer funcionar el servidor de la aplicación, lleva las siguientes aplicaciones:

- **Tomcat 9**: es la aplicación encargada de hacer correr nuestro servidor web JAVA.
- **JRE 8**: es el conjunto de herramientas que proporciona Oracle y es necesario para que nuestro servidor JAVA funcione correctamente.

- **MySQL**: base de datos que contiene los datos a los que necesita acceder nuestro servidor JAVA.

Una forma de abaratar costes de dicha aplicación, sería alojar el servidor en una máquina propiedad de la Universidad de Burgos, ahorrándose así el costo del alquiler de la máquina virtual de Microsoft Azure.

Trabajos relacionados

Actualmente nos podemos encontrar con diversas técnicas para crear un asistente virtual. Una de las técnicas que nos podemos encontrar es el uso de inteligencia artificial. Dentro de la inteligencia artificial, una de las técnicas empleadas en estos casos es la de razonamiento basado en casos.

6.1. Artículos

Chatter-bots: Making AI work for your business

Este artículo escrito, proveniente de una página especializada en el razonamiento basado en casos, escrito por Paul White, nos comenta cuales son los pasos que debe de seguir nuestro asistente virtual para que funcione de forma correcta. Los consejos que nos aporta es que nuestro asistente virtual tiene que tener los objetivos muy claros, que al principio empiece con unos conocimientos básicos y que los conocimientos vayan ampliándose progresivamente a lo largo del tiempo.

6.2. Proyectos

TFG - UBUassistant de Daniel Santidrián Alonso

Este TFG es la base de este proyecto, ya que de él está heredado parte del código, sobre todo a aquello a lo referente al sistema de razonamiento basado en casos, utilizando *jCOLIBRI 2* [25].

Cómo crear una aplicación de chat para Android usando Firebase

Este tutorial, realizado por Ashraff Hathibelagal, es la base en la cual se ha basado la aplicación Android. Principalmente se ha empleado como base

el diseño de la aplicación, así como la forma de mostrar los mensajes, aunque posteriormente fue modificada en busca de una estética más propicia para el tipo de aplicación [26].

6.3. Fortalezas y debilidades del proyecto

Fortalezas del proyecto

Las fortalezas que posee este proyecto son:

- Aplicación Android compatible con todos los smartphones que posean versión 4.4 o posterior.
- Interfaz de usuario sencilla e intuitiva.
- Posibilidad de utilizar voz para realizar las búsquedas.
- Facilidad de creación de una interfaz de cliente utilizando otros lenguajes de programación.
- Bajo consumo de datos para realizar las consultas.
- Servidor alojado en la nube, por lo que la aplicación se puede usar desde cualquier localización.

Debilidades del proyecto

Por el contrario, existen algunas debilidades dentro del proyecto:

- La aplicación Android es solo para los usuarios, por lo que hay que seguir empleando la web para administrar el aprendizaje.
- La aplicación siempre necesita conexión a Internet para poder funcionar.

Conclusiones y Líneas de trabajo futuras

7.1. Conclusiones

Tras el desarrollo del proyecto, las conclusiones a las que se ha llegado son las siguientes.

- Finalmente se ha conseguido realizar el proyecto de la forma en la que se había planteado desde un primer momento, cumpliendo todos los objetivos que se habían marcado al comienzo del proyecto.
- Se han utilizado la gran mayoría de los conocimientos adquiridos a lo largo de la carrera.
- Durante el desarrollo del proyecto se han ampliado conocimientos, tales como desarrollo de aplicaciones Android, conexiones cliente-servidor, inteligencia artificial, etc.
- Se ha aprendido a utilizar herramientas de virtualización de servicios en la nube, como las que ofrece Microsoft Azure.
- El proyecto ha sido posible completarlo dentro del periodo estipulado para ello.

7.2. Líneas de trabajo futuras

Este proyecto tiene todavía mucho potencial para crecer aún más, por lo que se puede seguir trabajando en las siguientes líneas:

- Crear una aplicación de administración para dispositivos móviles.
- Permitir que el algoritmo recoja ámbitos más concretos de la página de la Universidad de Burgos.
- Realizar otros clientes para otro tipos de dispositivos.

- Alojar el servidor de la aplicación en un servidor propiedad de la Universidad de Burgos.
- Añadir un nuevo campo a la tabla *caseDescription*. Este campo se tiene que utilizar para tener un texto descriptivo de cada uno de los casos.

Bibliografía

[1] J. A. L. López, "ios crece en españa, todavía con android muy lejos," 2018. [Online]. Available: https://apple5x1.com/ios-crece-espana-android-lejos/

[2] C. G. Calatrava, "Repositorio de UBUassistant en GitHub," 2018, [Internet; Accedido 30-mayo-2018]. [Online]. Available: https://github.com/cgc0045/TFG-UBUassistant

[3] Wikipedia, "Razonamiento basado en casos," 2018, [Internet; Accedido 31-mayo-2018]. [Online]. Available: https://es.wikipedia.org/wiki/Razonamiento_basado_en_casos

[4] J. Recio-García, B. Díaz-Agudo, and P. González-Calero, "Gaia – group of artificial intelligence applications," 2008, [Internet; Accedido 31-mayo-2018]. [Online]. Available: http://gaia.fdi.ucm.es/research/colibri

[5] Wikipedia, "Modelo vista controlador," 2018, [Internet; Accedido 30-mayo-2018]. [Online]. Available: https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador

[6] BBVAOPEN4U, "Api rest: qué es y cuáles son sus ventajas en el desarrollo de proyectos," 2016, [Internet; Accedido 31-mayo-2018]. [Online]. Available: https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos

[7] Wikipedia, "Git," 2018, [Internet; Accedido 30-mayo-2018]. [Online]. Available: https://es.wikipedia.org/wiki/Git

[8] ——, "Eclipse," [Internet; Accedido 30-mayo-2018]. [Online]. Available: https://es.wikipedia.org/wiki/Eclipse_(software)

[9] ——, "Android studio," 2018, [Internet; Accedido 30-mayo-2018]. [Online]. Available: https://es.wikipedia.org/wiki/Android_Studio

[10] ——, "Microsoft azure," 2018, [Internet; Accedido 25-junio-2018]. [Online]. Available: https://es.wikipedia.org/wiki/Microsoft_Azure

[11] ——, "Latex," 2018, [Internet; Accedido 30-mayo-2018]. [Online]. Available: https://es.wikipedia.org/wiki/LaTeX

[12] ——, "Texstudio," 2018, [Internet; Accedido 30-mayo-2018]. [Online]. Available: https://es.wikipedia.org/wiki/TeXstudio

[13] D. S. Alonso, "Tfg del grado en ingeniería informática," 2017, [Internet; Accedido 30-mayo-2018]. [Online]. Available: https://github.com/DanielSantidrian/UBUassistant/raw/master/doc/latex

[14] L. Lozano and J. Fernández, "Razonamiento Basado en Casos:"Una Visión General"," pp. 1–59, 2008. [Online]. Available: http://www.infor.uva.es/{~}calonso/IAI/TrabajoAlumnos/Razonamientobasadoencasos.pdf

[15] J. Recio-García, B. Díaz-Agudo, and P. González-Calero, "jcolibri2 tutorial," pp. 1–110, 2008.

[16] pico.dev, "Ejemplo de api rest en java con jax-rs y spring boot," 2016, [Internet; Accedido 31-mayo-2018]. [Online]. Available: https://picodotdev.github.io/blog-bitix/2016/09/ejemplo-de-api-rest-en-java-con-jax-rs-y-spring-boot/

[17] Google, "Android studio," 2018, [Internet; Accedido 31-mayo-2018]. [Online]. Available: https://developer.android.com/

[18] Microsoft, "Soporte técnico de la comunidad de azure," 2018, [Internet; Accedido 31-mayo-2018]. [Online]. Available: https://azure.microsoft.com/es-es/support/community/

[19] "Stack overflow," 2018, [Internet; Accedido 31-mayo-2018]. [Online]. Available: https://stackoverflow.com/

[20] A. Atanassov and L. Antonov, "Comparative analysis of case based reasoning softwareframeworks jcolibri and mycbr," 2012, [Internet; Accedido 1-junio-2018]. [Online]. Available: https://bit.ly/2tvHULY

[21] U. of Wales, "Caspian," 1995, [Internet; Accedido 1-junio-2018]. [Online]. Available: http://www.aber.ac.uk/~dcswww/Research/mbsg/cbrprojects/getting_caspian.shtml

[22] U. of Edinburgh, "Cbr tools," 2004, [Internet; Accedido 1-junio-2018]. [Online]. Available: http://www.aiai.ed.ac.uk/project/cbr/cbrtools.html

[23] I. d'Investigació en Intel-ligència Artificial, "Cat-cbr," 2002, [Internet; Accedido 1-junio-2018]. [Online]. Available: http://www.iiia.csic.es/Projects/cbr/cat-cbr/

[24] G. R. C. for Artificial Intelligence, "mycbr," 2015, [Internet; Accedido 1-junio-2018]. [Online]. Available: http://www.mycbr-project.net/

[25] D. S. Alonso, "Tfg del grado en ingeniería informática," 2017, [Internet; Accedido 31-mayo-2018]. [Online]. Available: https://github.com/DanielSantidrian/UBUassistant

[26] A. Hathibelagal, "Cómo crear una aplicación de chat para android usando firebase," 2016, [Internet; Accedido 31-mayo-2018]. [Online]. Available: https://code.tutsplus.com/es/tutorials/how-to-create-an-android-chat-app-using-firebase--cms-27397