UNIVERSIDAD DE LA RIOJA

TRABAJO FIN DE ESTUDIOS

Título

Aplicación web de triaje con interacción mediante la voz

Autor/es

IMANOL RODRÍGUEZ SAN MIGUEL

Director/es

ARTURO JAIME ELIZONDO

Facultad

Facultad de Ciencia y Tecnología

Titulación

Grado en Ingeniería Informática

Departamento

MATEMÁTICAS Y COMPUTACIÓN

Curso académico

2020-21

publicaciones.unirioja.es
E-mail: publicaciones@unirioja.es

UNIVERSIDAD DE LA RIOJA

Facultad de Ciencia y Tecnología

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Aplicación web de triaje con interacción mediante la voz

Realizado por:

Imanol Rodríguez San Miguel

Tutelado por:

Arturo Jaime Elizondo

Logroño, Julio, 2021

Resumen

Este proyecto ha sido realizado en la empresa Hiberus Osaba. Desde la misma, me ofrecieron a mí la posibilidad de realizar este trabajo, que consistiría en la investigación de diferentes bots de voz para rellenar formularios en aplicaciones web, teniendo como principal baza Amazon Lex. Esto pretendía ayudar a las personas que escriben lento a máquina.

Para aunar las tecnologías, se ha realizado una aplicación web que representara el proceso de triaje, que es la evaluación que se realiza a una persona que acude a urgencias. Además de la realización de la evaluación, la aplicación también muestra las evaluaciones, los pacientes y las enfermedades guardadas en la aplicación.

La aplicación web se realizaría con Spring e Hibernate, frameworks de Java. Estas tecnologías no las conocía antes de empezar el proyecto, por lo que tuve que realizar una formación que ha ocupado un tiempo considerable del proyecto antes de iniciar el producto.

Como se cuenta en este documento, la consecución de rellenar el formulario con la voz no ha sido posible, aunque se ha realizado un estudio de diferentes tecnologías que facilitará el enfoque de futuros proyectos en los que la empresa intente implementar dicha funcionalidad.

Abstract

This project has been made in Hiberus Osaba. They offered me the possibility to do this task, which consisted in investigating some voice bots to fulfill the forms in web applications. The starting point from the project was Amazon Lex. This goal was thought to help people who type slowly with keyboards.

To try this technology, the company thought about the triage process, which is the process that takes place when a patient attends the emergency room. Besides the evaluation process, the application also shows the evaluations, the patients and the diseases saved in the application.

The web application would be made with Spring and Hibernate, both are Java's frameworks. I did not know these technologies prior to starting the project, therefore I had to learn them before starting the web application implementation.

As it is told in this document, I could not implement the fulfillment of the form with the voice, although a study has been made comparing different technologies which will ease the approach in future projects in which the company tries to implement this feature.

Índice

1. Introducción

En este apartado justificamos la elección del proyecto, los motivos que llevaron a realizarlo, la situación por la que el cliente decidió ponerlo en marcha y sus objetivos principales.

1.1. Contexto

Por desgracia, a todos nos toca acudir a urgencias, ya sea porque somos nosotros los que tenemos algún problema o por algún familiar cercano. En ambos casos, sabemos que el tiempo, en determinadas situaciones, puede marcar la diferencia y, por ello, surgió la idea de esta aplicación web.

Muchas veces, nos atienden personas que no escriben rápido a máquina, y, mediante el control por voz, se podría acelerar el proceso de triaje para conseguir reducir el tiempo dedicado a esta tarea al mínimo posible.

Así, se podría mejorar el proceso de triaje para dedicar el tiempo a lo que verdaderamente importa, que es atender a los pacientes que acuden a urgencias.

Esta es la idea que Javier Virto, director provincial de Hiberus Osaba, me transmitió cuando me dijo que iba a cambiar mi proyecto para el Trabajo de Fin de Grado.

1.2. Motivación

En primera instancia, el proyecto iba a ser un gestor del desempeño de los empleados desarrollado en ASP.NET MVC. Más tarde, se cambió el proyecto al que actualmente nos ocupa. El cambio lo acepté sin dudarlo, ya que había varias características que me atraían más de este proyecto que del anterior.

La primera de ellas era que en la empresa en la que iba a realizar el Trabajo de Fin de Grado se necesitaba personal que desarrollara en Spring e Hibernate y, si realizaba este proyecto era posible que, si lo que yo realizaba satisfacía a la empresa, ocupara esa vacante.

Además, el proyecto iba a ser desarrollado en Java, y no en C#, lo cual me agradaba ya que Java lo considero el lenguaje central de la carrera y el que más conozco, teniendo en cuenta, por supuesto, que la asignatura de Programación de Aplicaciones Web se realizaba en dicho lenguaje.

Por otro lado, desde antes de entrar a este grado he deseado realizar algún proyecto que sirviera para ayudar a la mayor gente posible, y considero que el apartado de la salud es el más importante dentro de una sociedad, por lo que crear un proyecto que sirviera para acelerar el proceso de triaje era algo que me atraía.

Asimismo, así como el otro proyecto era interesante, no veía ningún reto intrínseco, mientras que en este caso tenía que aprender a usar no sólo Spring e Hibernate, sino también encontrar la mejor forma de transcribir voz a texto en tiempo real, conllevaba tanto investigación como aprendizaje de su uso, y esto último podría llegar a no completarse.

En esta memoria, comenzaremos plasmando la planificación del proyecto realizada inicialmente para después llevar a cabo un seguimiento y comprobar si hemos cumplido el objetivo del proyecto en esa estimación de tiempos. La planificación consta del alcance (estudio de alternativas, entregables y EDT), del tiempo de las tareas (cronograma, diagrama de Gantt y de hitos) y de la gestión del proyecto (gestión de comunicación, de recursos, de riesgos, de calidad, de cambios, de interesados y de adquisiciones).

Posteriormente, desarrollaremos la captura y análisis de requisitos y diseño del problema planteado. En ellas aparecerán gráficamente los diseños realizados y los diagramas necesarios para poder comprenderlo mejor.

A continuación, explicaremos aspectos sobre el proceso de implementación seguido, es decir, del desarrollo del proyecto: pasos que hemos ido realizando, problemas que nos han ido apareciendo, cómo los hemos resuelto, etc. Y finalmente, incluiremos el informe de seguimiento que hemos ido realizando en el transcurso del proyecto, las lecciones que hemos aprendido a lo largo de ese tiempo y unas cuantas conclusiones sobre la realización del proyecto. Así como la bibliografía de las páginas y documentación más visitada.

2. Antecedentes

El proyecto se ha realizado en Hiberus Osaba S.L., perteneciente al grupo Hiberus, empresa TIC (Tecnologías de la Información y la Comunicación) cuya sede central está en Zaragoza. Cuenta con más de 1.200 trabajadores que cubren varias áreas de las TIC. La empresa se organiza en varios departamentos:

- **Consultoría**: dónde se realizan proyectos para clientes.
- **Desarrollo y Outsourcing**: se gestiona el talento y se desarrollan proyectos con metodologías ágiles. Este proyecto se enmarca en esta área.
- **Digital**: especializada en varias disciplinas como el desarrollo con CMS (Gestores de Contenidos), el control SEO/SEM (posicionamiento en buscadores) o usabilidad.
- **Soluciones**: desarrollos propios e integración de estos en empresas.
- **Tecnologías diferenciales**: especializado en nuevas tendencias.
- **Sistemas**: seguridad, helpdesk, cloud…

Por tanto, se manejan diferentes tecnologías dependiendo del departamento y de cada equipo. La propuesta de proyecto ha venido precedida de la realización en esta empresa de las dos asignaturas de prácticas externas. Durante las mismas me encomendaron diferentes tareas de programación web en el contexto ASP.NET MVC con C#, JavaScript, jQuery, HTML, CSS y procedimientos almacenados de bases de datos.

Al final de dicho periodo, me ofrecieron la realización de un proyecto interno a la empresa como Trabajo de Fin de Grado (TFG). El objetivo era desarrollar un **gestor del desempeño de los empleados**. Al comienzo de cada ejercicio, cada empleado introduciría la lista de sus objetivos anuales a los que se asociarían recompensas. Objetivos y recompensas deberían pactarse entre trabajador y su evaluador. Durante el año se irían introduciendo los objetivos cumplidos de forma que, a final del ejercicio, se pudiesen valorar los logros y otorgar las recompensas.

Sin embargo, tras dos semanas de trabajo, la empresa decidió priorizar otro tipo de proyecto y me plantearon la posibilidad de cambiarlo. Esta reorientación se debía a que se había

identificado la necesidad de personal formado en **Spring MVC**. La realización de este nuevo proyecto se asociaba a una posible contratación a partir de su finalización.

La idea de este nuevo proyecto fue lanzada por Javier Virto, director provincial de la empresa. A este, le parecía interesante el posible uso de **Amazon Lex** en los proyectos realizados en la empresa y planteó explorar sus posibilidades a corto plazo mediante la realización de un proyecto. Amazon Lex permite crear bots online a los que poder enviar grabaciones de voz o de texto para que las procesen y realicen diferentes tareas. Un ejemplo, que se podría implementar es la petición de una comanda a un restaurante interactuando con el bot en lugar de hacerlo mediante la interfaz de su sitio web.

El objetivo inicial de la empresa fue rellenar un formulario a través de la voz, con ayuda de uno de estos bots. Para ello habría que enviar el audio al bot, recoger los datos textuales que devolviera, y extraer separadamente los fragmentos correspondientes a los distintos campos del formulario. El objetivo era diseñar una solución para un formulario genérico que sirviera para utilizarla en cualquier aplicación. La distribución del texto recogido en los campos del formulario sería responsabilidad de cada aplicación. De esta manera, se estaría desacoplando la interacción con el bot de la aplicación. Se preveía que una solución en esta línea podría mejorar la interacción con los formularios, sobre todo para aquellas personas que escriben más despacio con un teclado, y se podría conseguir una interacción general más agradable con las aplicaciones. Hay que tener en cuenta que la empresa no tenía todavía ninguna experiencia con estas tecnologías. La misión que se encomendaba al proyecto era precisamente aprender, experimentar y documentar los logros conseguidos y los fracasos experimentados. La solución podría servir para mejorar la accesibilidad de las páginas de algunos clientes como gobiernos de comunidades autónomas como la aragonesa.

El contexto elegido para realizar las pruebas sería el proceso de **triaje** de una sala de urgencias. El promotor de la idea tenía una amplia experiencia en el sector sanitario y propuso la realización de una aplicación muy simple que apoyara a este proceso de triaje, que consiste en la evaluación de los pacientes que asisten a estos centros.

Hay dos argumentos favorables al cambio. El primero es que se desarrollaría en Java, lenguaje con el que tengo más experiencia que con C#. El segundo es cambiar el contexto de los recursos humanos por el área sanitaria, que me resulta más atractivo.

Por el contrario, se introducen varios riesgos, algunos difíciles de controlar. El primero es que la empresa no dispone de personal experto en algunas tecnologías, como Amazon Lex o sus alternativas. También hay que tener muy en cuenta que a la empresa le interesa mucho que durante el proyecto adquiera experiencia con Spring e Hibernate, tecnologías muy usadas en la creación de aplicaciones web y de microservicios en Java. Como habían identificado la necesidad de personal formado en ellas, se mostró dispuesta a contar conmigo al finalizar el proyecto. El problema es que no tenía experiencia previa con ellas y, por tanto, el proyecto debería incluir todavía más formación. Sin embargo, tenía interés en ellas desde la realización de la asignatura “Programación de Aplicaciones Web”, una de las que me resultaron más atractivas. Después de valorar todo lo anterior, acabé aceptando el cambio.

En conclusión, el proyecto parte desde cero, sin respaldo por parte de la empresa respecto a algunas de las tecnologías centrales, y realizando una aplicación simple de triaje, sin un cliente real, y donde el objetivo principal es aprender a utilizar bots online que convierten audios de voz a texto.

3. Alcance y objetivos

El objetivo general del proyecto es la implementación de un módulo que envíe ficheros de **audio** a un bot de Amazon Lex, u otra tecnología similar, y recopilar el texto equivalente devuelto por este. También hay que desarrollar una aplicación sencilla, en el contexto del triaje de un servicio de urgencias sanitarias, donde se incorporará el módulo anterior con el propósito de lograr completar un formulario a través de la voz. Se incluye dentro de los objetivos del proyecto la investigación de la tecnología Amazon Lex. El proyecto también exige la utilización de Spring e Hibernate en la aplicación, lo cual conlleva que la formación en ambas tecnologías formará parte del alcance del proyecto.

4. Gestión

4.1. Alcance (requisitos)

4.1.1. Requisitos generales

A continuación, se presentan los requisitos generales de la aplicación:

- El primero y más importante es el proceso de realizar una evaluación a un paciente, tanto de forma **escrita,** como de forma **hablada**. Para una evaluación, la información que se desea saber es: quién realiza la evaluación (el personal de urgencias), quién es evaluado (el paciente), la dolencia del paciente, la prioridad que tiene el paciente según la escala de Manchester[1], la fecha en la que se realiza la evaluación y un identificador que empezará con el Número de la Seguridad Social del Paciente, y tendrá luego un número que irá incrementando para cada evaluación que reciba el paciente. Además de estos datos, que son obligatorios, puede ser interesante almacenar, dependiendo de cada caso, la temperatura, la altura y el peso.

  El paso previamente descrito se separará en dos, el primero de ellos en el que se elegirá el paciente al que se realiza la evaluación y el segundo en el que se introducen el resto de los datos. Para elegir al paciente existirán dos formas, la primera de ella introducir el Número de la Seguridad Social del Paciente y la segunda el nombre y el domicilio. Para esta segunda opción, sería interesante no tener que rellenar ambos campos, si no que a medida que se va rellenando el del nombre nos aparezcan las opciones a elegir.

  En el formulario final, se podrán visualizar las enfermedades de un paciente con la temporalidad correspondiente.

- Además de realizar una evaluación, se desea que se puedan **visualizar** con todos los campos arriba mencionados. La forma de mostrar los datos ha de ser una tabla que permita ordenar por cada uno de los campos. Para mostrar la dolencia, como es un texto que puede ocupar varias líneas de texto, se preferirá que exista un desplegable

[1] La escala de Manchester sirve para asignar una prioridad en las salas de urgencias a cada paciente. Esta escala va desde el 1 hasta el 5, siendo el 1 el que tiene una mayor prioridad y el 5 una menor. Es usada en muchas salas de urgencias y han surgido variantes a nivel de hospitales, pero se considera la escala madre de todas.

y que sólo se muestre al activarlo. Es interesante poder ver las evaluaciones dependiendo de la fecha, por ejemplo, la de las últimas 4 horas, último día y desde siempre.
- Al igual que visualizar las evaluaciones, también sería interesante tener una lista de los pacientes, mostrados, igualmente, en forma de tabla para que su visualización sea sencilla.
- De la misma forma, queremos tener una lista con las enfermedades.
- Por último, toda la funcionalidad ha de estar protegida por medio de una **autenticación** con usuario y contraseña, ya que sólo se desea que el personal de urgencias tenga los permisos de realizarlo.

4.1.2. Requisitos no funcionales

A continuación, se detallan los requisitos no funcionales:

- La aplicación web ha de ser **multiplataforma**: el cliente desea poder usar la aplicación desde cualquier navegador: Firefox, Chrome, Edge, Internet Explorer o Safari.
- Ha de permitir usarse desde **varios dispositivos**: la aplicación web debe ser usable en móviles, tabletas y ordenadores, siendo este último el objetivo principal.

4.1.3. Exclusiones

Debido a que es un proyecto realizado por una sola persona y con un tiempo limitado, máxime teniendo en cuenta que primero hay un periodo de formación y también otro de investigación, se consideran las siguientes exclusiones que podrían mejorar la aplicación. Estas son:

- Inclusión de un área de administración para que usuarios con unos permisos superiores puedan añadir nuevos evaluadores, modificar sus datos o quitarlos, además de realizar las mismas tareas para pacientes.
- Área del paciente para que este pueda ver sus evaluaciones.
- Permitir administración de enfermedades.

4.1.4. EDT

La figura 1 muestra la EDT (estructura de descomposición del trabajo) del proyecto.

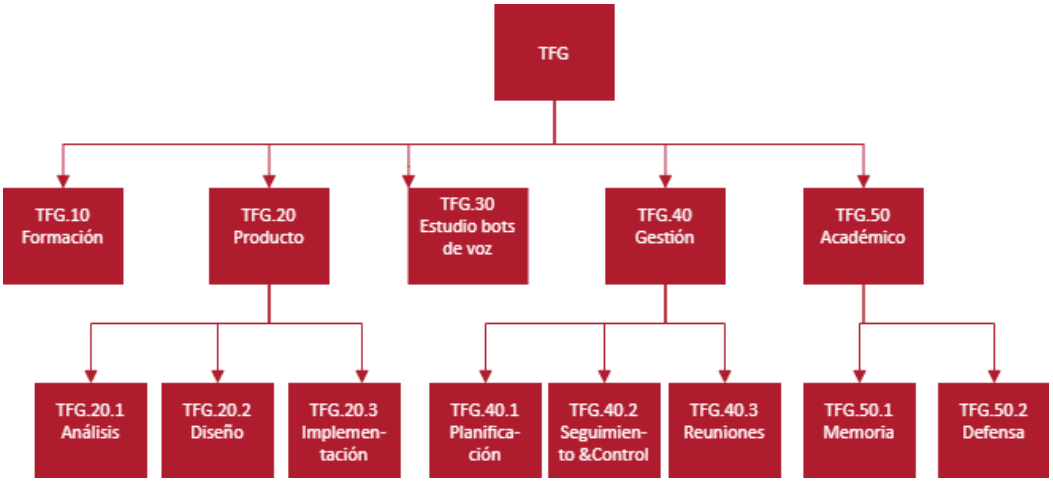

Figura 1. EDT del proyecto.

4.2. Cronograma

En la figura 2 podemos ver los hitos del TFG, que se explican a continuación:

- El inicio se produjo el día 15 de febrero, cuando acudí a la empresa a recoger el ordenador que me prestaron para la totalidad del proyecto.
- El día 26 de febrero se me comunicó que, desde la empresa, se deseaba cambiar el proyecto del TFG. Cambio que acepté y comuniqué al tutor de la universidad. Debido a este cambio, el día 1 de marzo empecé con la formación en Spring e Hibernate.
- El día 11 de marzo tuvimos una reunión sobre el alcance y los requisitos del proyecto.
- Una vez acabé mi formación, inicié la primera de las tres fases del proyecto. Que explicaré en dos apartados diferentes (Metodología e Implementación).
- El día 20 de junio fue el día que cumplí las 300 horas en la empresa y, por lo tanto, cuando acabé el proyecto.
- El depósito de la memoria se realiza entre el 7 y el 9 de julio
- La defensa se realiza el 22 o el 23 de julio.

Hito	Febrero		Marzo					Abril				Mayo					Junio				Julio			
	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24
Inicio TFG	◆																							
Comunicación cambio TFG		◆																						
Aclaración nuevo TFG				◆																				
Inicio formación Spring			◆																					
Inicio fase 1 del proyecto							◆																	
Fin fase 1 del proyecto									◆															
Inicio fase 2 del proyecto										◆														
Fin fase 2 del proyecto													◆											
Inicio fase 3 del proyecto													◆											
Fin fase 3 del proyecto														◆										
Fin estancia en empresa														◆										
Memoria revisar																	◆							
Depósito de memoria																					◆			
Defensa TFG																							◆	

Figura 2. Diagrama de hitos principales del proyecto.

En la figura 3 vemos un diagrama de Gantt explicando los periodos del proyecto. Hay que destacar que la semana 8 coincide con la semana de Pascua, por ello tiene 0 horas.

Periodo ejecución paquete TFG	Febrero		Marzo					Abril				Mayo					Junio				Julio			
	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24
10. Formación																								
20.1. Análisis																								
20.2. Diseño																								
20.3. Implementación																								
30. Investigación bots																								
40.1. Planificación																								
40.2. Segu&control																								
40.3. Reuniones																								
50.1. Memoria																								
50.2. Defensa																								

Figura 3. Cronograma del proyecto.

4.3. Recursos (dedicaciones)

En esta figura 4 vemos un desglose de las horas dedicadas a cada paquete de la EDT. El horario en la empresa era de 5 horas al día, por lo que hacían 25 horas a la semana. La semana 7 de abril coincidió con Semana Santa, por lo que constó de 15 horas únicamente. También, la última semana fue más corta ya que se cumplieron las 300 horas en la empresa el martes de dicha semana.

Dedicación en horas paquete TFG	Febrero		Marzo					Abril				Mayo					Junio				Julio				TOTAL
	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24	
10. Formación			18	20	24	22																			84
20.1. Análisis	18	18	5	3	1	1	1		2	1	1	1	1												53
20.2. Diseño											5	3	3												11
20.3. Implementación							12		20	22	17	17	6	2											96
30. Investigación bots													10	5											15
40.1. Planificación	2	2				1	1		1	1	1	1	1												11
40.2. Segu&control									1	1	1	1	1	1											6
40.3. Reuniones	5	5	2	2		1	1		1				1	1											19
50.1. Memoria												2	2	1	8	8	8	8	8	8	4				57
50.2. Defensa																					5	5	5		15
TOTAL	25	25	25	25	25	25	15	0	25	25	25	25	25	10	8	8	8	8	8	8	9	5	5	0	367

Figura 4. Horas de dedicación a cada paquete por semana redondeadas a horas enteras.

5. Método de desarrollo

No se ha seguido ningún método concreto, como *scrum* o *xp*. En su lugar se ha seguido un proceso iterativo dividido en tres etapas, donde cada una era *horizontal* o *vertical*.

La primera etapa se desarrolló horizontalmente, y se desarrolló primero la base de datos (BD) y después la capa Modelo del patrón MVC.

En cambio, el resto de las etapas se desarrollaron de forma vertical, abordando una por una las características y desarrollando, para cada característica, primero la capa Controlador del patrón MVC y luego la Vista. La investigación sobre el rellenado del formulario se realizó entre las dos últimas etapas.

6. Experimentación tecnológica y formación

Como ya se ha explicado en este proyecto se iban a explorar las posibilidades de interacción desde aplicaciones Java con Amazon Lex o tecnologías alternativas. Además, la realización del proyecto exige la formación en Spring e Hibernate. En este apartado explicaré cómo se realizaron estas tareas.

6.1. Exploración de Amazon Lex

Esta tarea se inició al finalizar la segunda etapa de la aplicación de triaje. Javier Virto organizó una reunión con dos responsables de Amazon que colaboran habitualmente con la empresa para contrastar con ellos si el objetivo perseguido era viable. Yo no pude asistir, pero, como queda reflejado en el acta, los responsables de Amazon concluyeron que no era viable completar un formulario genérico como se había previsto. Por tanto, el objetivo se reorientó a rellenar un campo. El campo más interesante del formulario de la aplicación de triaje era el de **dolencia**, por ser el de mayor longitud y, por tanto, el que podía ahorrar más tiempo de escritura por teclado. Aunque se reducía el alcance, la acción de rellenar un campo de un formulario con un texto potencialmente largo seguía siendo interesante.

Después de invertir varios días investigando, me di cuenta de las dificultades que conlleva el problema y no encontré una solución. El funcionamiento de Amazon Lex es similar al de otros bots como Alexa, Siri o el asistente de Google. Se trata de ir satisfaciendo *intents* (así les llaman desde Amazon) o acciones que el usuario quiere realizar. El primer paso para ello es "despertar" al bot, como puede ser decirle *OK Google* al asistente de dicha compañía, y ahí es dónde estaba el problema, ya que no era capaz de hacer que el bot escuchara la petición.

Así que intenté cambiar el enfoque, y creo que aquí está una de las conclusiones más importantes: **tener claro qué se quiere.** Si sabemos qué es lo que deseamos con nuestra aplicación, obviamente la búsqueda de tecnologías será mucho más satisfactoria. En el caso de la aplicación aquí presentada estaba claro: *crear un formulario que se pueda rellenar con el teclado o con la voz.*

Entonces me pregunté: ¿para qué quiero un bot que pueda procesar lo que se le manda si lo único que quiero es que me devuelva en formato de texto lo que se ha dicho? De esta forma, empecé a buscar formas de transcribir voz a texto, para que, en vez de usar un bot como Amazon Lex, que quizá era demasiado complejo para lo que se quería, simplemente hiciera eso, coger un audio y devolver el texto con las palabras que se dicen en él.

Todos estos problemas, se los comuniqué a la empresa, que estuvieron de acuerdo conmigo en intentar cambiar el enfoque hacia el uso de una tecnología más simple.

Durante la búsqueda llegué a tres alternativas principales: **Amazon Transcribe**, **Google Speech to Text** y una solución creada por una persona en Internet que ponía el **código en su página web**.

El problema con la tercera solución era obvio: copiar el código de una persona en Internet no parece ético ni viable. Además, los ejemplos de uso eran en inglés, con lo no estaba claro si servía para el castellano. Por tanto, decidí descartarlo.

Cuando empecé a estudiar las otras dos opciones, el tiempo del proyecto se estaba agotando. Muy a mi pesar, ya que considero que ésta es una característica central del producto, no se consiguió implementar ninguna, ya que no eran sencillas de integrar en la aplicación que estaba desarrollando.

Una de las soluciones que mucha gente parecía tener clara en los foros de internet era solicitar ayuda de especialistas de Amazon o de Google, dependiendo del caso. Esta solución la consideraba mucha gente en foros puesto que, al final, ahorraba tiempo, que al fin y al cabo es dinero si hablamos de un trabajo real.

Así, esta característica, una de las impulsoras del proyecto, quedaba descartada, al menos de momento. Sin embargo, la empresa sigue interesada en ello y prevé retomar el problema en mi previsible contratación.

6.2. Formación en Spring e Hibernate

La formación comenzó con la búsqueda de un curso que, en el menor número de horas posible, adquiriera un conocimiento suficiente como para realizar la aplicación. Gracias al conocimiento previo de la página Udemy, acudí ahí en búsqueda de un curso con dichas características. Después de un tiempo no muy extenso encontré un curso que me convenció por contenidos y críticas. El curso tenía el nombre de: "Spring & Hibernate for Beginners (includes Spring Boot)[2]".

A lo largo del curso se explicaban varias características de Spring, como Spring MVC, Spring Core, Spring REST, Spring Boot… además de Hibernate y JPA (Java Persistence API).

En cuanto a Hibernate, que podría decirse que es una implementación de JPA, considero que simplifica mucho el mapeo de una clase con respecto a la BD en comparación a cómo lo realizábamos en la asignatura de Programación de Bases de Datos, ya que con simples anotaciones podemos mapear nuestra BD a nuestro modelo de dominio.

El único problema que he tenido con Hibernate es el mapeo de las relaciones muchos a muchos cuando la relación tiene un atributo, ya que, pese a existir la anotación "@ManyToMany" para anotar dichas clases, hay que hacerlas igual que en la BD, es decir, partir la relación en dos y crear una tabla intermedia con claves externas cada una a una tabla de la relación (anotándolas con @OneToMany) y, además, el atributo deseado.

En cuanto a consultas, hay varias opciones para crearlas, pero me sorprendió que JPA es tan potente que, con la creación de interfaces con sus métodos, es capaz de, sin darles ninguna implementación, crear los métodos para que podamos usarlos en otras clases.

Por ejemplo, si en una interfaz de acceso a datos creamos un método que se llame findUserByID (String id), es decir, un método cuyo nombre traducido sería encontrar un usuario por su identificador, pasándole dicho identificador como argumento, no hará falta que lo implementemos en una clase, sino que, gracias a la anotación *@Autowired,* basta con declarar la interfaz en la clase que queramos usar dicho método y usándolo, tendremos el usuario por el identificador que le pasamos por parámetro. De esta forma, la creación de métodos con consultas no muy complejas se puede hacer de forma muy rápida.

En cuanto a consultas más complejas, Hibernate tiene su propio dialecto de SQL (Structured Query Language), que es HQL (Hibernate QL). Al principio es incómodo de usar, ya que se parecen, pero no son exactamente iguales, y deja ver que crear estos dialectos muchas veces lo único que es, es un estorbo al programador. Aun así, una vez acostumbrados el uso es bastante sencillo.

En cuanto a Spring, es una tecnología tan amplia y potente que en el curso se ahonda lo justo como para poder empezar a crear aplicaciones con ella. Considero que la elección del curso

[2] URL del curso: https://www.udemy.com/course/spring-hibernate-tutorial/

fue buena puesto que la parte que más trataba era la parte de programación web (Spring MVC).

Tratar de resumir o de explicar Spring es tremendamente complejo, puesto que es tan grande que resulta inabarcable, aunque podríamos decir que es una capa sobre Java EE (Enterprise Edition) que abstrae el lenguaje de programación y lo potencia para crear aplicaciones más grandes y de una forma más sencilla.

Hablando de Spring MVC, la parte de Spring que he usado para el producto, podríamos decir que sigue la línea de Spring, puesto que los controladores se acortan y son más fáciles de crear, haciendo que tardemos menos a la vez que evitamos errores.

En la asignatura de Programación de Aplicaciones Web (PAW) teníamos métodos que ocupaban más de 100 líneas, y, aquí, se pueden realizar de forma más simple, ocupando mucho menos sin perder claridad. Entre otras, las características que más importantes me parecen para conseguir esto son:

- Mapeo invisible al programador de los argumentos: en un método, si ponemos como argumento una clase, ya sea compleja o simple, Spring la mapeará por nosotros, quitándonos no sólo la creación de la clase sino también de la comprobación de los datos, si es que precisamos de ella.

  Además, no es sólo la comprobación de los datos en el servidor, si utilizamos las etiquetas HTML que ofrece Spring, la petición no se enviará si los datos no pasan la validación que podemos poner a nuestras clases de dominio.

- Inyección de dependencias de forma muy simple: gracias a la anotación @Autowired, podemos desacoplar nuestros módulos de forma muy sencilla, ya que Spring instanciará las clases por nosotros. Haciendo que en una clase declaremos una interfaz, pero utilicemos su implementación de forma transparente.
- Soporta la internacionalización de forma muy sencilla.
- Conversión de tipos
- Además, da libertad con respecto al frontend, podemos usar Angular, ThymeLeaf, HTML puro, JSPs…

Como conclusión con respecto a estas dos tecnologías, considero que son muy avanzadas a la vez que simples de utilizar. Antes, crear una aplicación web como lo hacíamos en PAW era tedioso, aunque divertido, ahora, nos quitamos gran parte de esa carga. Eso sí, la mayoría de los fallos se concentrarán en la configuración de la aplicación web y no en su código.

7. Análisis y diseño

En esta sección cubriremos 3 apartados sobre la aplicación: arquitectura, diseño de datos y la funcionalidad principal.

7.1. Funcionalidad principal

La funcionalidad principal de la aplicación ha de cubrir las siguientes necesidades:

- Autenticarse con las credenciales de cada personal de urgencias.
- Permitir visualizar las evaluaciones en forma de tabla ordenadas de más reciente a más antigua, pero permitiendo ordenar por cualquiera de los otros campos.
- Permitir visualizar los pacientes en forma de tabla.
- Permitir visualizar las enfermedades en forma de tabla.
- Permitir realizar una nueva evaluación, rellenando los campos necesarios para dicha tarea que se han especificado.

7.2. Arquitectura

Pasando a tratar la arquitectura del producto. Esta aplicación web utiliza el conocido patrón Modelo-Vista-Controlador (MVC), repasando las tecnologías desde *abajo* (persistencia), hacia *arriba* (presentación), tenemos:

- **Tomcat** como servidor de la aplicación web.
- **MySQL**: la BD ha sido creada con este Sistema Gestor de Bases de Datos puesto que sabía cómo enlazarlo correctamente con Hibernate y, además, su uso es gratuito.
- **Hibernate**: una de las tecnologías de uso obligado de la aplicación. Este framework de Java se usa para enlazar la BD con el proyecto Java, puesto que lo realiza de una forma sencilla y rápida. Además, su uso con Spring está muy extendido.
- **Spring**: junto a Hibernate, la otra tecnología impuesta para el proyecto. Con Spring, desarrollar una aplicación web MVC es mucho más sencillo de lo que lo era antes, ya que reduce mucho las líneas de código necesarias en las aplicaciones.
- **Bootstrap** (HTML y CSS): pese a que, como he dicho antes, Spring deja a la libertad elegir la forma de maquetar el front-end (podríamos usar ThymeLeaf, por ejemplo), he decidido utilizar Bootstrap ya que lo conozco y usarlo iba a suponer un ahorro de tiempo con respecto a utilizar otras tecnologías.
- **JavaScript, jQuery y AJAX** (Asynchronous JavaScript and XML): utilizados para peticiones al servidor asíncronas, en el caso de AJAX, mostrar las evaluaciones y pacientes con DataTables en el caso de jQuery, y JavaScript para desarrollar pequeños programas que permitan a la aplicación ser más usable.

7.3. Diseño de datos

Durante la primera etapa de desarrollo, diseñe una base de datos (BD) para la aplicación de triaje. Partiendo de los requisitos se diseñó el esquema entidad-relación de la figura 5 y su transformación al modelo relacional aparece en la figura 6.

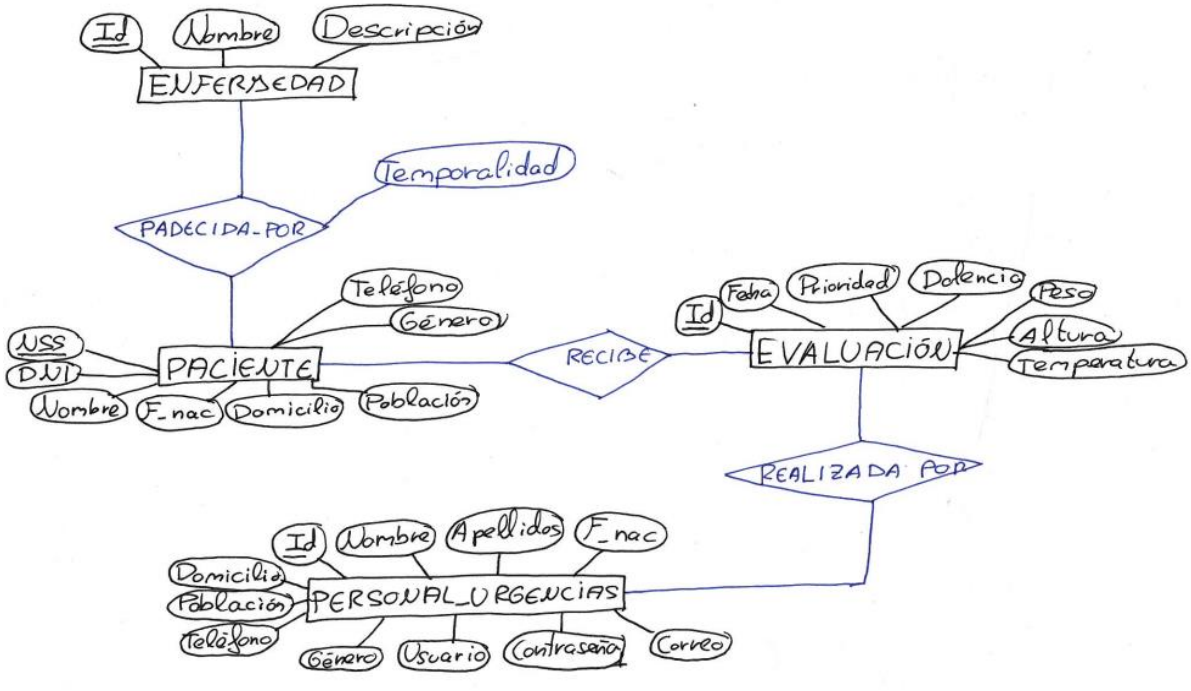

Figura 5. Diagrama Entidad-Relación de los datos de la aplicación

Figura 6. Modelo relacional de los datos de la aplicación

Al realizar el paso de normalización se comprobó de forma trivial que la BD se encuentra ya en Forma normal de Boyce-Codd. Se utilizó MySQL, creando una conexión y dando a la BD un nombre, un usuario y una contraseña. Se crearon las tablas según se muestra en el diagrama de MySQL de la figura 7.

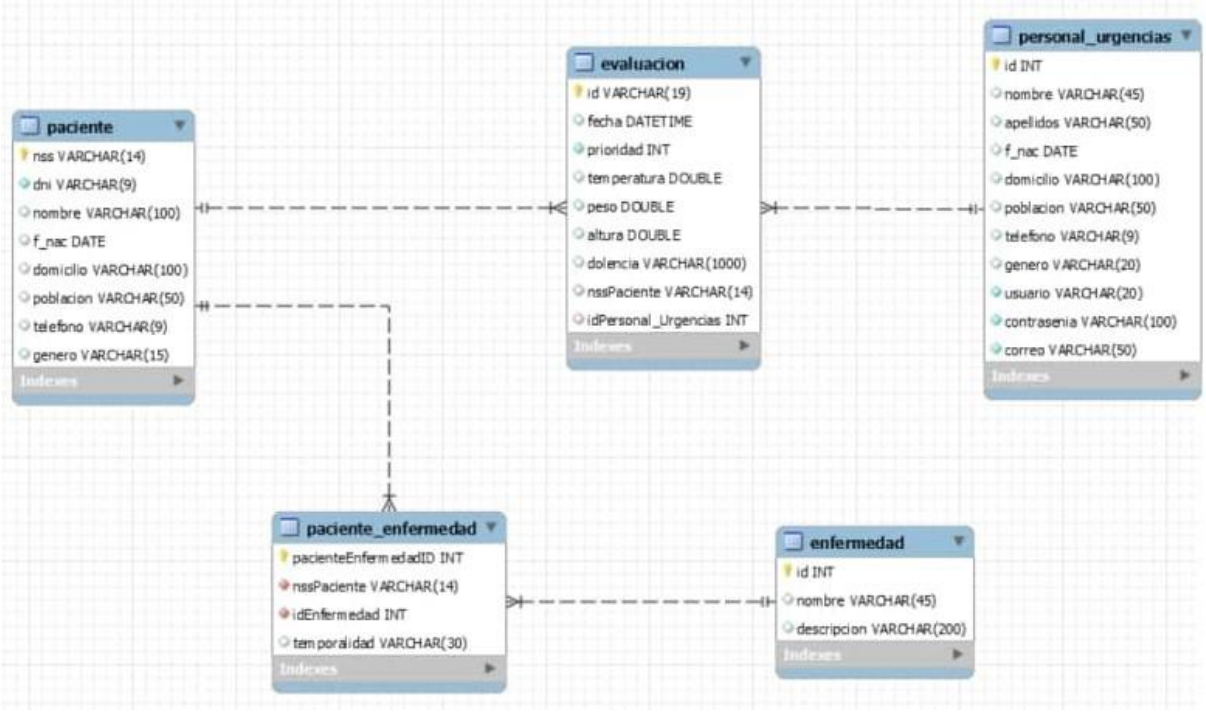

Figura 7. Diagrama generado por MySQL de los datos de la aplicación

8. Implementación

Como ya se ha explicado el desarrollo se dividió en tres etapas, que explicaré a continuación.

8.1. Primera etapa: modelo

Durante esta primera etapa de desarrollo, realicé la BD y la parte de la aplicación web de acceso a la persistencia con Hibernate y JPA.

El paso de creación del modelo de datos y su implementación en MySQL está explicado en el apartado anterior (7.3. Diseño de datos).

Antes de ponerme a programar, tenía que crear unos datos de prueba para comprobar que todo funcionaba correctamente cuando hiciera diferentes consultas a la BD, un proceso muy sencillo de realizar gracias a la sentencia INSERT INTO de SQL.

El siguiente paso en esta etapa era conectar Hibernate y la BD, para ello, había que realizar la configuración necesaria en el fichero **applicationContext.xml.** Lo que tenía que hacer era escribir la URL de la conexión con la BD, junto al usuario y la contraseña de la sesión creada anteriormente. Una vez conectada la aplicación con la BD era el momento de ponerse manos a la obra y crear las clases que correspondían al modelo de dominio, un paso sencillo excepto en un aspecto, que es la relación muchos a muchos que venimos arrastrando desde el diagrama Entidad Relación entre la tabla paciente y la tabla enfermedad. Este problema ha sido explicado anteriormente en el apartado 6.2. (Formación en Spring e Hibernate).

Una vez ya estaban creadas las clases y mapeadas con respecto a la BD, era el momento de probar la conexión, para ello, creé un fichero llamado hibernate.cfg.xml, en el que puse la misma configuración que en el fichero applicationContext.xml. Una vez rellenado dicho fichero, creé una aplicación principal desde la que realizar alguna consulta a la BD, de esta forma, comprobé el enlace entre la aplicación y la BD.

Cuando ya estaba comprobada la conexión, era el momento de crear clases que fueran DTOs (Data Transfer Object), estas fueron:

- PersonalUrgenciasLogin: para usarla cuando una persona perteneciente al personal de urgencias se autentique en el sistema.
- PacienteBúsquedaPorNSS: cuando hacemos una evaluación, antes de pasar al formulario final, el personal de urgencias debe de elegir el paciente al que le está realizando la evaluación, en este caso, se busca según el NSS.
- PacienteBusquedaNombreYDomicilio: al igual que el anterior, sirve para elegir al paciente antes de pasar al formulario final, pero en este caso buscamos el nombre y el domicilio del paciente, se implementará con AJAX.

Después de crear los DTOs, era el momento de crear la parte de acceso a los datos, DAO (Data Access Object), para ello, primero creé una interfaz y luego una clase que las implementara.

Como he comentado en el apartado 6.2. crear estas interfaces era muy sencillo, y las clases que lo implementaran también. En primera instancia podría parecer innecesaria su creación teniendo en cuenta lo dicho en el apartado referenciado, pero consideré el crear las clases que implementaran la interfaz como forma de aprender a usar el framework.

El siguiente paso después de terminar los DAOs, era crear los servicios, que no son más que interfaces y sus implementaciones que en este caso utilizan las interfaces creadas de los DAOs para acceder a la BD. La capa más sencilla de la aplicación.

Una vez creados los accesos a los datos, la primera etapa estaba terminada, por lo que era el momento de pasar a la segunda.

8.2. Segunda etapa: páginas generales

Esta segunda etapa, como ya he comentado en el apartado del método de desarrollo, siguió un proceso incremental de forma vertical, basándose en lo hecho en la anterior etapa. El mismo proceso fue seguido en la tercera etapa: creación de controladores y vistas.

Las páginas que se realizaron durante esta etapa fueron:

- Autenticación del personal de urgencias.
- Página principal de la aplicación.
- Visualización de pacientes.
- Visualización de evaluaciones.
- Visualización de pacientes.

8.2.1. Autenticación del personal de urgencias

La creación de la vista de **autenticación del personal de urgencias** ha sido una de las más convulsas, ya que empecé con la intención de implementar Spring Security para autenticarse y acabé realizando la autenticación de la misma forma que hicimos en la asignatura de Programación de Aplicaciones Web.

El primer paso para la creación de esta funcionalidad fue crear el HTML y el CSS, como en esta página la única funcionalidad disponible ha de ser la de autenticarse, puesto que no se desea que nadie sea capaz de acceder a ninguna funcionalidad si no forma parte del personal de urgencias, la vista es muy sencilla, una barra superior con el logo de la aplicación y sin menú, para no mostrar al usuario sin autenticar la funcionalidad disponible. Además de esto en la parte superior, teníamos un formulario en el cuerpo y un pie de página.

Una vez creada la interfaz, el siguiente paso era dotarla de funcionalidad, como he dicho, primeramente, la idea fue de realizar la autenticación con Spring Security, pero no fue posible debido a que el proyecto que había creado no era compatible con Spring Security, puesto que para ello era necesario tener un proyecto Maven, que no era mi caso. Sopesé la eliminación del proyecto y la creación de uno en Maven, pero como no había mucho tiempo por delante descartamos utilizar esta funcionalidad de Spring y pasamos a utilizar atributos de sesiones.

Para dotar de mayor seguridad a la aplicación, decidí proteger las contraseñas de la BD con MD5, por lo que necesité refactorizarla. Este paso no fue muy complejo, pero sí muy agradecido, porque aumentaba la seguridad de la aplicación.

En cuanto a la programación, fue muy sencilla puesto que lo único que necesitaba hacer era ver si la contraseña y el usuario coincidían con los de la BD y, si era así, poner el objeto personal de urgencias en la sesión para simular la autenticación. La figura 8 contiene la pantalla de autenticación de la aplicación.

Figura 8. Autenticación del personal de urgencias

8.2.2. Página principal de la aplicación

La creación de esta página incluyó la propia página y el menú superior, que iba a ser usado a lo largo de toda la aplicación. La figura 9 muestra esta página.

El menú debía de contener opciones para ver las evaluaciones, los pacientes, realizar una nueva evaluación y cerrar la sesión. En el caso de la opción “Nueva Evaluación” se me pidió que destacara sobre las demás. El menú fue enteramente creado con Bootstrap.

Después, tenemos un pequeño texto dando la bienvenida a la aplicación y, por último, tres componentes que son cartas para ver las evaluaciones, los pacientes y las enfermedades.

Dar funcionalidad a estas opciones fue muy sencillo, puesto que son todo peticiones muy simples de satisfacer.

Figura 9. Página principal de la aplicación

8.2.3. Visualización de pacientes, enfermedades y evaluaciones

Todas estas vistas son muy similares, pues en todas ellas se usa el framework DataTable de jQuery (que es muy potente) para dar forma a la tabla creada por HTML. La única parte compleja fue añadir un botón en la vista de las evaluaciones para mostrar la dolencia del paciente debajo de la fila de la evaluación. Para conseguirlo, utilicé AJAX.

El código detrás de ellas es bastante sencillo, ya que simplemente hay que traer de la BD las correspondientes entradas y luego colocarlas en el JSP recorriendo la lista. Las figuras 10, 11 y 12 muestran estas tres pantallas que son muy similares entre sí.

Figura 10. Visualización de los pacientes de la aplicación

Figura 11. Visualización de las enfermedades de la aplicación

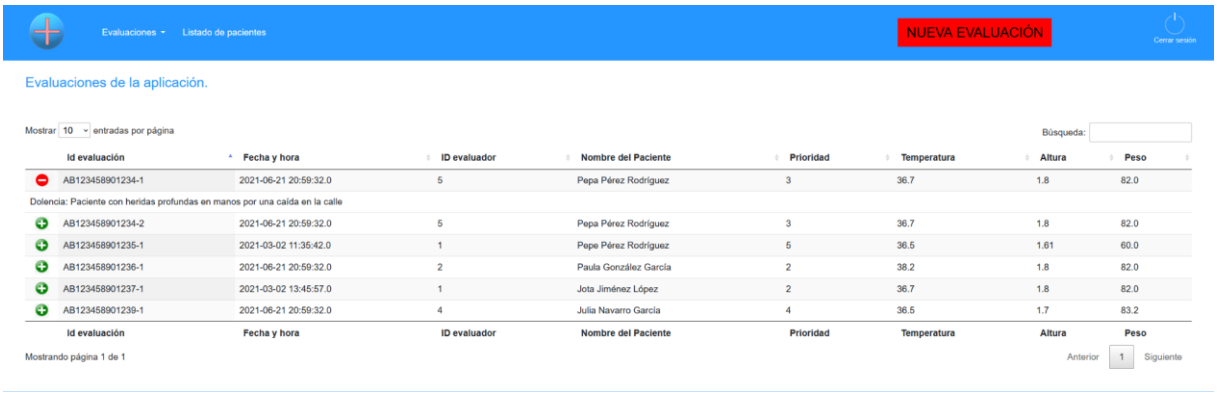

Figura 12. Visualización de las evaluaciones de la aplicación

8.3. Tercera etapa: formulario de nueva evaluación

En esta tercera etapa lo que quedaba por hacer era crear el formulario para realizar una nueva evaluación, para ello, el paso se dividió en dos, primero se introduciría o bien el NSS del paciente, o bien su nombre y su domicilio. Más tarde y una vez elegido el paciente, se pasaría al siguiente y último paso, el de rellenar los distintos campos que el cliente pidió (dolencia, prioridad, altura, peso y temperatura). Aquí era dónde Amazon Lex debería entrar en acción, para permitir rellenar el formulario sin necesidad de tocar el teclado.

Primero, voy a comentar los formularios creados para introducir el cliente y luego el formulario final.

8.3.1. Introducir paciente según su NSS

Este es el formulario que se muestra por defecto cuando un evaluador hace clic en la opción “Nueva Evaluación” del menú, ya que preguntar por el NSS puede ser una forma rápida de rellenar el formulario. Se muestra en la figura 13. Lo único que se hace aquí es introducir el NSS y, en función de si existe o no un paciente con tal NSS en la BD, dar paso al formulario final o devolver a este para que se vuelva a introducir el parámetro.

Figura 13. Introducción del NSS del Paciente

8.3.2. Introducir paciente según su nombre y su domicilio

A urgencias acuden personas que, normalmente, van con prisa porque, valga la redundancia, tienen una urgencia, por lo que es posible que acudan sin su tarjeta de la seguridad social que tiene su NSS, debido a esto, debemos permitir realizar evaluaciones para esos pacientes a los que se les ha olvidado la tarjeta. Esta es la razón de la existencia de esta página.

El usuario deberá introducir el nombre del paciente y, de esta forma, se abrirá debajo un desplegable con el nombre y la dirección del paciente, al hacer clic, se rellenarán los dos campos y se pasará al siguiente formulario. También es posible introducir los campos sin hacer uso de esta funcionalidad, pero creo que es mejor ya que ahorra tiempo, un recurso muy preciado en el proceso de triaje.

Para implementar esta funcionalidad, he hecho uso de AJAX, realizando una petición al servidor según lo que esté escrito en el campo del nombre del paciente. El servidor procesa la petición y devuelve una lista con los pacientes que se adecúan a ese criterio y, por último, el navegador por medio de JavaScript coloca debajo una etiqueta SELECT que, al hacer clic en cualquiera de las entradas, rellena ambos campos con lo escrito en dicha etiqueta. La figura 14 muestra la interfaz que recoge esta funcionalidad.

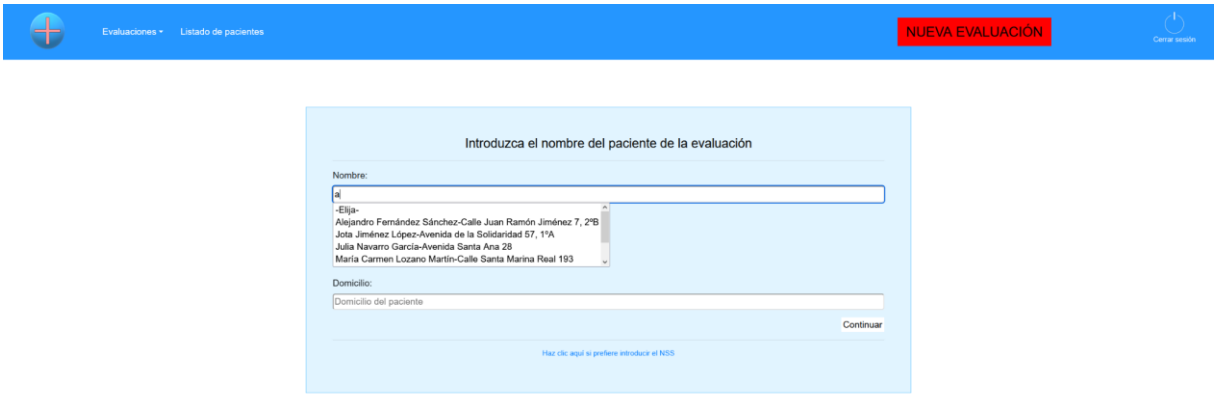

Figura 14. Introducción del nombre y del domicilio del Paciente

8.3.3. Formulario final de la evaluación

En este formulario final del proceso de triaje, el personal de urgencias deberá introducir los campos que faltan para completar una evaluación, que son la prioridad y la dolencia, de forma obligatoria, y el peso, la altura y la temperatura de forma opcional, ya que la fecha será del momento de rellenar el formulario, el id es autogenerado, el personal de urgencias es el que está logueado y el paciente ya ha sido introducido en el paso anterior.

Dar funcionalidad a este formulario no fue muy complicado gracias a Spring e Hibernate, aunque tuve que buscar cómo mostrar las enfermedades del paciente, ya que no encontraba una forma que me convenciera. Así, llegué a poner un enlace debajo del NSS del paciente que abre una ventana modal con sus enfermedades y su temporalidad. Las interfaces correspondientes aparecen en las figuras 15 y 16.

En cuanto a la funcionalidad de voz, como he comentado en el apartado 5.2, no ha sido posible implementarla.

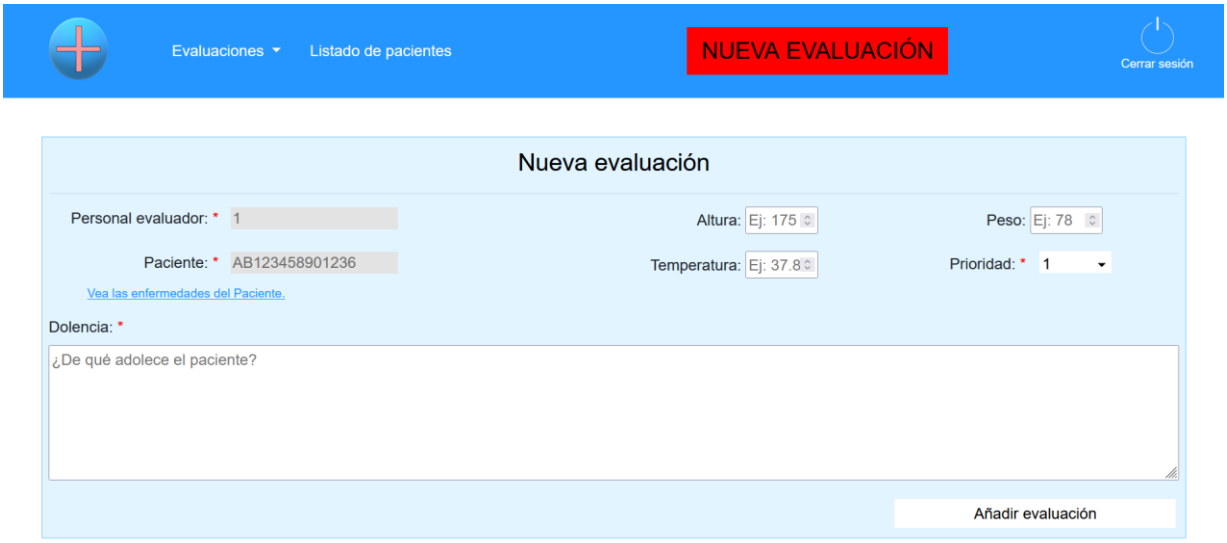

Figura 15. Formulario de nueva evaluación

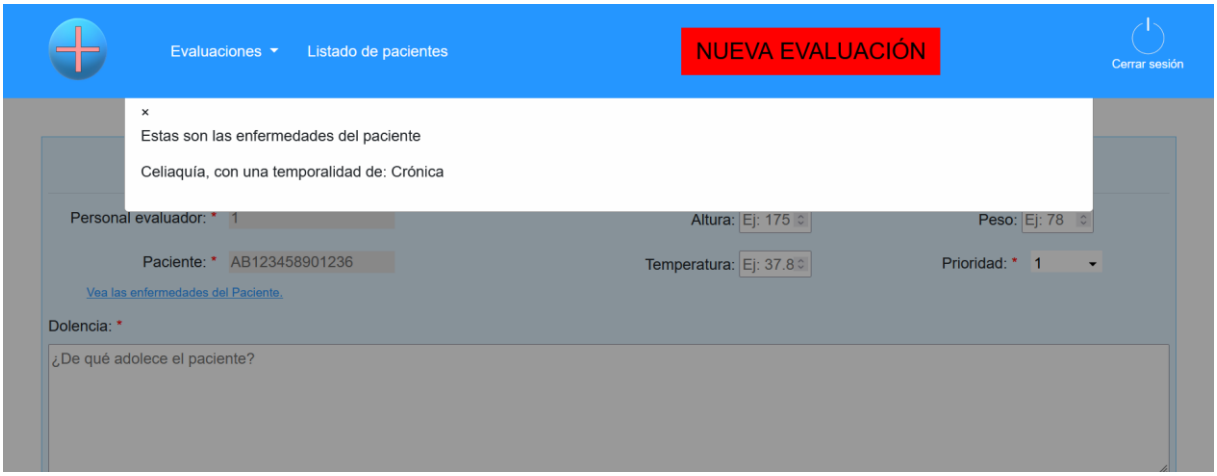

Figura 16. Enfermedades del paciente en el formulario de nueva evaluación

9. Mejoras

Según lo visto hasta aquí, hay varias características que podrían mejorar la aplicación de cara al futuro y a próximas iteraciones. Estas mejoras son:

- Autenticación por medio de Spring Security. Con esta característica de Spring, aumentará de forma notable la seguridad de nuestra aplicación web, un apartado crítico ya que los problemas de salud se consideran muy importantes según la LOPD (Ley Oficial de Protección de Datos).
- Sustitución del apartado de enfermedades por historial clínico. En una reunión con el tutor de la empresa, Iñigo León, se planteó la opción de sustituir el apartado de enfermedades, que es muy tedioso puesto que hay que introducir las enfermedades en la BD y se aleja de lo deseable, por la visualización del historial clínico del paciente. Se consideró que era mejor en todos los aspectos, puesto que ahorraba tiempo, era más cómodo y se visualizaba mejor las posibles complicaciones que podía sufrir un paciente debido a las enfermedades que padecía. No se implementó por falta de tiempo.
- **Por supuesto, la inclusión de la interacción con la voz.**
- Las descritas en exclusiones.

Con estas mejoras, la aplicación web tendría un aspecto diferente ya que sería más completa de lo que es ahora.

10. Conclusiones

10.1. Lecciones aprendidas

A lo largo del proyecto me he enfrentado a varios problemas y a continuación, explico en qué consistían y cómo se pueden resolver o evitar.

1. **La excesiva formación impacta en el alcance.** Trescientas horas parece un esfuerzo considerable como para poder desarrollar un producto interesante. El periodo de prácticas en empresa, previo a la realización de un TFG, da la oportunidad de recibir toda la formación necesaria para realizar un desarrollo con garantías y de cierta calidad. Sin embargo, los cambios de última hora que exigen el cambio a tecnologías desconocidas no están exentos de riesgos. Por una parte, exigen la dedicación de buena parte del tiempo a aprenderlas y comprender todas sus implicaciones. Por ejemplo, dediqué varios días a integrar Spring Security para programar la autenticación y al final tuve que implementarlo de forma manual, es decir, de forma menos segura y menos eficiente. Por otra parte, si alguna de las tecnologías es desconocida para la empresa, entonces estamos asumiendo al menos tres riesgos: primero que no esté disponible en algún momento del desarrollo o no tengamos acceso a alguna parte que nos interese. Segundo, que conlleve costes imprevistos de formación y tercero que acabemos dándonos cuenta de que no es adecuada para las necesidades del proyecto. En nuestro caso, la formación en Spring e Hibernate ha consumido gran cantidad de tiempo. Además, Amazon Lex ha mostrado que no es adecuada para las necesidades

del proyecto. Un consumo tan considerable de tiempo ha impedido reorientar la solución hacia otras posibilidades más acordes con el problema al que nos enfrentábamos.

2. **Las configuraciones**. En los proyectos realizados con tecnologías modernas es muy importante que los ficheros de configuración sean correctos, es decir, que no contengan errores. Ya me encontré con este problema durante las tareas que realicé durante las prácticas previas al proyecto. Para reducir el impacto de este problema es muy importante conocer el objetivo y la implicación de cada uno de los siguientes ficheros de configuración de un sitio web desarrollado con Spring e Hibernate:
    - **web.xml**: sirve para configurar el mapeo de servlets, especificar el *listener o* la localización del applicationContext.xml. En proyectos Maven no es necesario.
    - **pom.xml**: sólo lo encontraremos en proyectos Maven, y se trata de su unidad principal. Contiene información de fuentes, **dependencias**, texto, plugins, versiones… El proyecto realizado no era Maven, pero muchas soluciones encontradas para problemas que surgían pasaban por modificar este fichero (por ejemplo, Spring Security). De haber sabido que con Maven la inyección de dependencias es mucho más sencilla habría adoptado esta solución.
    - **applicationContext.xml**: se definen varias variables esenciales para la aplicación, como los paquetes que escaneará Spring para encontrar nuestros componentes, la conexión a la BD, la localización de los ficheros de vista y su extensión (JSP en mi caso), la factoría de sesiones de Hibernate, adición de soporte para leer recursos web como HTML, CSS y JavaScript…
    - **hibernate.cfg.xml**: en él se indica la clase a utilizar como controlador para conectarnos a la BD, la URL, el usuario y la contraseña de la BD y el número máximo de conexiones. En nuestro proyecto no era necesario, pero lo creé para realizar pruebas de conexión a la BD y probar sentencias HQL.
3. **Diseñar bien la BD**. Parece obvio, pero es importante dedicar tiempo a diseñar bien la BD. Me he encontrado con problemas en la fase de programación cuyo origen era haber cometido errores de diseño y que me han obligado a rehacer la BD.
4. **En Hibernate hay que tener claro qué es *eager* y qué *lazy loading***. Al realizar una consulta (SELECT) a una BD, si tenemos una tabla con una clave foránea hacia otra tabla, podemos cargar los datos de la otra tabla al hacer la consulta (EAGER) o hacerlo más tarde o nunca (LAZY). Al principio, pensando en la eficiencia, ponía siempre LAZY loading menos en puntos en los que era claramente necesario tener los datos de la otra tabla a la vez que se realizaba la consulta. Esto fue un problema debido a que en la realización de una nueva evaluación se pueden ver las enfermedades del paciente. Con LAZY loading, al mostrar la ventana tenía un fallo y no se mostraba nada, así que, para solucionarlo, tuve que utilizar AJAX y realizar otra petición al servidor.
5. **Elegir bien el material formativo**. Esto es una cuestión esencial, sobre todo si buena parte del proyecto exige formación. La calidad del curso que seguí en Udemy era buena. Sin embargo, he encontrado otro de microservicios mucho más enfocado a crear aplicaciones de una forma que me hubiera resultado muy útil. Esto me hubiese ayudado a enfocar el desarrollo de la aplicación de otra forma, con Maven, diseño dirigido por dominio…
6. **Lo primero es entender el problema y después buscar la solución.** Son los pasos típicos en los métodos de resolución de problemas. Es importante tener claro cuál es el problema, definirlo y después buscar soluciones alternativas y elegir la que parece

más apropiada. En nuestro caso partimos de la solución: Amazon Lex, porque parecía resolver un problema. Incluso se fabricó un problema ficticio para resolverlo con esa solución. Sin embargo, el trabajo realizado llevó a concluir que, primero, no resolvía ese problema y, segundo, que había soluciones mejores y más simples al problema. Sin embargo, el avance tecnológico actual y el tratar de utilizar nuevas tecnologías de formas innovadoras parece llevarnos a esta especie de mundo al revés.

10.2. Reflexiones

A lo largo del proyecto, me he dado cuenta de lo que es trabajar en un entorno no controlado. Sí que es verdad que durante la asignatura de Informática móvil y, sobre todo, en Prácticas Externas ya tuve una experiencia similar. Pero en este último caso tenía el apoyo de los compañeros de proyecto a los que, en caso de duda, podía preguntar y me ayudaban a solucionar el problema gracias a su conocimiento y experiencia. En cambio, con el TFG, como es un proyecto individual, esa ayuda no estaba, ya que en la empresa estaban en un periodo de mucho estrés y no he tenido prácticamente ayuda tecnológica, lo que, por un lado, es negativo, ya que el producto no ha sido completo y ha tenido un alcance bastante corto, pero, por otro lado, he aprendido mucho gracias a ello, por ejemplo, ahora busco mejor mis dudas en internet, sé planificarme mejor...

Además, he conseguido poner en acción conocimiento de una gran variedad de asignaturas. De las que podría destacar Diseño de Bases de Datos, Programación de Aplicaciones Web, Programación de Bases de Datos e Informática Móvil, además de todas las asignaturas de programación (Metodología de la Programación, Tecnología de la Programación, Programación Orientada a Objetos…). El conseguir llevar a la práctica conocimientos adquiridos durante estas asignaturas es muy satisfactorio, puesto que siento que los conocimientos adquiridos en la carrera, aunque puedan no tener una aplicación directa, dan unas bases muy importantes a la hora de salir al mundo laboral.

10.3. Conclusiones técnicas y tecnológicas

Como he ido comentando a lo largo del presente documento, se han utilizado varias tecnologías incluyendo MySQL, Hibernate, Spring, HTML, CSS, JavaScript (jQuery y AJAX), Bootstrap y Tomcat. Obviamente no he logrado profundizar mucho en ninguna de ellas, ya que para ello se necesita una experiencia con ellas mucho más dilatada, pero sí que he dado un paso adelante en esa dirección. Voy a comentar lo que entiendo que he aprendido en cada una:

- MySQL: este es el SGBD (Sistema Gestor de Bases de Datos) que más me gusta, puesto que se puede usar de forma gratuita y tengo una experiencia considerable con él. He aprendido a mostrar los diagramas, y he aplicado el conocimiento adquirido en Bases de Datos y en Diseño de Bases de Datos para crear una BD para el proyecto. Diría que ha logrado un nivel de conocimiento medio.
- Hibernate: he partido desde cero y actualmente considero que he logrado conocerlo bastante bien. He mapeado clases con tablas complejas (la relación muchos a muchos con atributos extra, por ejemplo). También he aprendido bastante de HQL, que no es especialmente diferente de SQL, pero cambia ciertos aspectos que hacen que su uso no sea tan sencillo. Además de Hibernate puro, también he utilizado Validator para

implementar restricciones sobre las clases. Considero que he alcanzado un nivel de conocimiento medio.

- Spring es un mundo enorme, y quizá, más que Spring, debería hablar exclusivamente de Spring MVC y Spring Boot, ya que, por ejemplo, no he usado Spring Security (aunque me hubiera gustado). Es un framework que facilita la programación de webs, evitando programar tareas tediosas y dejándole al programador las acciones que contienen mayor complejidad. Considero que tengo un nivel de conocimiento bastante bajo.
- HTML y CSS: los lenguajes por antonomasia de las páginas web. Antes de empezar el proyecto creo que ya tenía un nivel de conocimiento medio. Ahora creo haber alcanzado un nivel medio-alto.
- De JavaScript puro creo que tengo un conocimiento medio, pero si consideramos extensiones como jQuery considero que se reduce a un nivel bajo, puesto que no la he usado demasiado y me parece lo suficientemente distinta a JavaScript como para considerar un nivel de conocimiento diferente. Eso sí, creo tener un nivel medio de conocimiento de AJAX.
- Bootstrap: antes de empezar el TFG, no conocía casi nada, pero gracias a la asignatura de Desarrollo de Interfaces para Usuarios y al proyecto, diría que tengo un buen nivel de conocimiento. Bootstrap facilita notablemente la programación del *frontend.*
- Tomcat: poco lo conocía antes y ahora sólo sé instalarlo y vincularlo al IDE para usarlo.
- Si se desea únicamente recibir una transcripción de un mensaje de voz a texto, no es necesario un bot de procesamiento.

11. Bibliografía

URL del producto en GitHub: https://github.com/imrodrm/triaje[3]

Página principal de Spring: https://spring.io/

Página principal de Hibernate: https://hibernate.org/

Página del curso: https://www.udemy.com/course/spring-hibernate-tutorial/

Stack Overflow (inglés): https://stackoverflow.com/

Stack Overflow (castellano): https://es.stackoverflow.com/

Amazon Lex: https://docs.aws.amazon.com/lex/latest/dg/lex-dg.pdf

Introducción a Amazon Lex: https://docs.aws.amazon.com/en_en/lex/latest/dg/what-is.html

Amazon Transcribe: https://aws.amazon.com/transcribe/

Google Speech to Text: https://cloud.google.com/speech-to-text

Solución de transcripción de voz a texto creada por una persona: https://codingshiksha.com/javascript/javascript-speech-to-text-notes-app-using-web-speech-api-full-project/

Página web de la empresa en la que se realizó este TFG: https://www.hiberus.com/

Página web de la universidad: https://www.unirioja.es/

[3]Los commits están fuera de la fecha de asistencia a la empresa porque hasta que no acabé mi estancia trabajaba localmente.