Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Desarrollo de una herramienta para el aprendizaje de patrones de diseño software

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Adrián Ferrandis Homsi

Tutor: Vicente Pelechano Ferragud

2020-2021

Resumen

En este TFG se va a desarrollar una herramienta o aplicación web que facilitará el aprendizaje y la puesta en práctica de patrones de diseño orientados a objetos. La herramienta permitirá al profesorado o a cualquier formador la configuración de herramientas, que después los alumnos podrán utilizar a través de una aplicación web para aprender a reconocer, aplicar e implementar patrones de diseño. La herramienta se configurará para ser extensible y ampliable de forma que se pueda crear nuevo contenido para la misma. En su primera versión se dará soporte a un subconjunto de patrones de diseño representativos del libro de Gamma. Utilizando dicha herramienta se podrá dar de alta cualquier patrón de diseño que no esté incluido en la versión inicial de la herramienta.

Palabras clave: Spring, web, patrones de diseño, Java, educación.

Abstract

This TFG will develop a tool or web application that will facilitate the learning and implementation of object-oriented design patterns. The tool will allow teachers or any trainer to configure tools, which can then be used by students through a web application to learn to recognise, apply and implement design patterns. The tool will be configured to be extensible and expandable so that new content can be created for it. In its first version, a subset of representative design patterns from the Gamma book will be supported. Using this tool, any design pattern not included in the initial version of the tool can be added.

Keywords : Spring, web, design patterns, Java, education.

Tabla de contenidos

Índice de ilustraciones

1. Introducción

Un patrón de diseño intenta dar solución a un problema recurrente en la programación. Esta solución ha sido propuesta por especialistas con años de experiencia en la resolución de problemas similares, que han determinado una solución común. Dichas situaciones forman parte del día a día de un profesional dedicado a la programación. Cuando se implementa una solución incorrecta se suele crear una serie de problemas, principalmente un mal mantenimiento del código.

Es relevante destacar que los patrones también son unos facilitadores de la comunicación, ya que, si conocemos el nombre de un patrón de diseño y su función, a la hora de comunicarle o transmitirle a otro especialista que una pieza de código compone dicho patrón. La permitirá hacerse una idea de ante qué estructura de diseño e implementación se encuentra.

Actualmente, los patrones de diseño forman parte del día a día de cualquier programador experto, y cada día cobran más relevancia en el diseño y desarrollo de software. Es por este motivo que, al ser tan frecuentes en el día a día de un profesional, en este trabajo se propone el diseño y desarrollo de una herramienta que facilite el aprendizaje y el uso de estos.

1.1 Motivación

La realización de este trabajo viene motivada para facilitar el aprendizaje de un recurso útil y extendido hoy en día en el sector de la programación como son los patrones de diseño. Crear un código claro y de calidad suele ser una dificultad a la hora de iniciarse en el mundo de la programación. Esto solo puede lograrse realizando un diseño bien estructurado en el cual juegan un papel fundamental los patrones de diseño.

1.2 Objetivos

La realización de este trabajo tiene como objetivo la construcción de una aplicación donde se puedan proporcionar una serie de herramientas software para el aprendizaje

de los patrones de diseño. Además, se quieren desarrollar de forma que su mismo código sirva como ejemplo de una correcta estructuración del código. Además, nos gustaría poder que estas herramientas fueran aplicaciones web para facilitar el aprendizaje y trabajo de patrones.

Por tanto, nuestros objetivos consisten en:

- **Conseguir las bases de una aplicación web que permita el aprendizaje de los patrones de diseño. Esto se hará proporcionando un catálogo de los distintos patrones de diseño.**

- **Proporcionar una herramienta para la recomendación de patrones de diseño que faciliten la resolución de un problema concreto. Se debe guiar al usuario para obtener un patrón como solución a su problema.**

  Por otro lado, la aplicación debería cubrir las siguientes características:

- **Desarrollar un código claro y legible el cual incluya pruebas que validen su funcionamiento.**

- **Construir interfaces gráficas que faciliten el uso a los usuarios.**

- **Realizar un sistema lo más modificable y preparado a cambios posible.**

- **Ofrecer herramientas accesibles en formato web.**

1.3 Impacto esperado

Se espera que con el trabajo realizado se pueda brindar una nueva oportunidad para la implementación de distintas herramientas que puedan ayudar al trabajo con patrones de desarrollo. Esta aplicación está enfocada al ámbito del diseño de software y se espera que sea utilizado por docentes y alumnos. También puede ser usado por profesionales que están ejerciendo la profesión. Por último, pueden existir colaboradores interesados en participar en el futuro añadiendo herramientas a la plataforma, proporcionándoles esta una base para que puedan añadir funcionalidades o (quizás) prestaciones.

En concreto para el:

- **Profesorado**: Se espera que utilicen esta herramienta como administradores. Estos con sus conocimientos podrán modificar el contenido para ir mejorando la herramienta con los resultados del uso de la aplicación
  .
- **Alumnado y Profesionales**: Estos participaran como usuarios de la aplicación. Consumirán recursos que les facilitarán el trabajo de aplicar distintos patrones

de diseño. Además, también mediante sugerencias a los administradores podrán mejorar el contenido o proponer nuevas soluciones a problemas.

Se espera poder llegar a un número de usuarios amplio gracias a la tecnología web, que brinda un mayor alance y accesibilidad.

1.4 Metodología

El proyecto ha constado de varias fases para su construcción:

- **Recogida de información y especificación**: Esta fase comienza con una serie de conversaciones con el tutor de este trabajo de final de grado. A partir de ellas, se extrae la finalidad de la aplicación, tecnología y posibles utilidades. Se elige la plataforma web ya que esta proporciona un mayor alcance y accesibilidad. Posteriormente, con la información recogida se realiza una encuesta para la obtención de información de *feedback* del alumnado de la asignatura Diseño de Software. Con esto contaríamos con información de alumnado/profesorado y con datos sobre la tecnología para poder definir e iniciar el proyecto.

- **Formación**: Era necesario realizar formación para adquirir los conocimientos necesarios para la construcción de la web. Esta fase coexiste con la siguiente, ya que mientras me iba formando empecé a aplicar estos conocimientos a la construcción de la aplicación. Para la formación se utilizó dos cursos de Udemy*: Spring Framework 5: Beginner to Guru* [1] *con* una duración de 57 horas y *Testing Spring Boot: Beginner to Guru* [2] de una duración de 17 horas.

- **Desarrollo**: Se ha implementado la aplicación mediante metodologías ágiles usando Scrum y Kanban. Esto ha sido posible con el apoyo de la herramienta de Worki. Esta utilidad permite crear un ecosistema de trabajo que ayuda a gestionar fácilmente las distintas tareas que se deben abordar en un proyecto.

  A la hora de realizar el trabajo se ha optado por dividirlo en tres Sprints. En cada uno se desarrollará un MVP (Mínimo Producto Viable) [3] . Estos entregables esta centrados en características distintas donde se va desarrollando cada vez una aplicación más completa. Estos se plantean inicialmente con una duración estimada de 28 días, aunque esto podría variar en función de las necesidades. También se realizarán reuniones o se irá informando al dueño del producto, rol que tomará en este caso el tutor del TFG. En el resto de los casos, los otros roles serán tomados por el alumno.

  Se ha empleado para algunas secciones del desarrollo la tecnología de **desarrollo guiado por pruebas** [4], el cual permite construir una especificación que luego validaremos mediante nuestro código. Estas pruebas podrán implementar pruebas de aceptación o pruebas de lógica interna de la aplicación.

Gracias a estas pruebas podemos refactorizar sabiendo que la lógica del código sigue siendo la misma. Además, nos permite tener una serie de pruebas que validen que la aplicación es correcta, ahorrando así trabajo futuro de mantenimiento.

2. Estado del arte

Durante este capítulo del documento se va a exponer los distintos puntos que se deben de conocer para el entendimiento correcto de esta memoria y la aplicación de aprendizaje de patrones. Durante estos puntos analizaremos distintas soluciones aportadas para la solución de este problema. No obstante, no existe ninguna que lo realice en la web de forma interactiva.

2.1 Crítica al estado del arte

Aunque existe una amplia/extensa (elige un término) documentación sobre patrones, no existen aplicaciones que ofrezcan una serie de herramientas que ayuden a la identificación o a la integración de patrones de diseño en el desarrollo de software.

La principal referencia a la hora de estudiar los patrones de diseño son los libros de patrones de diseño. Históricamente se han publicado distintos libros publicados sobre el tema. El libro de referencia en el ámbito de los patrones de diseño es *Design Patterns: Elements of Reusable Object-Oriented Software* [5].En el prefacio de este libro encontramos la siguiente cita *"Design patterns capture solutions that have developed and evolved overtime. Hence they aren't the designs people tend to generate initially. They reflect untold redesign and recoding as developers have struggled for greater reuse and flexibility in their software"* [5] *. Es decir, los patrones son una solución aceptada de forma general pero no son un estándar como podría ser una norma ISO. Además, en este libro también encontramos la definición de patrón de diseño*: "Each design pattern systematically names, explains, and evaluates an important and recurring design in object-oriented systems. Our goal is to capture design experience in a form that people can use effectively." [5].

Necesitamos usar los patrones de forma coherente. Un uso excesivo y

descontrolado de ellos puede suponer un problema. Esto lo explica Erich Gamma, autor del libro mencionado el párrafo anterior en una de sus entrevistas:

> Gamma 2005
>
> Trying to use all the patterns is a bad thing, because you will end up with synthetic designs—speculative designs that have flexibility that no one needs. These days software is too complex. We can't afford to speculate what else it should do. We need to really focus on what it needs. That's why I like refactoring to patterns. People should learn that when they have a particular kind of problem or code smell, as people call it these days, they can go to their patterns toolbox to find a solution. [6]

A la hora de implementar un patrón de diseño no debemos perder de vista el objetivo de la implementación. Este aspecto lo explica Eric Freeman en su libro [7]: “First of all, when you design, solve things in the simplest way possible. Your goal should be simplicity, not "how can I apply a pattern to this problem.”. Esto significa que en lugar de centrarnos en aplicar patrones deberíamos de hacerlo cuando aparezca un problema que requiera de uno. Si no, debemos tomar la decisión más simple para resolver algo. Esto corresponde también con el principio de programación KISS. [8]

Con la popularidad de las nuevas tecnologías, se han desarrollado/propuesto un gran volumen de recursos en línea, que presentan el contenido expuesto en los libros, pero en el contexto de la red no existe una gran revolución respecto a los libros, más allá de la incorporación de la información que proporcionan los libros en formatos audiovisuales accesibles a través de la web.

También debemos remarcar la presencia de los patrones de diseño en la mayoría de los entornos de desarrollo integrados o entornos de trabajo facilitando o incorporando los mismos para aplicarnos fácilmente. Por ejemplo, en el entorno de desarrollo de Eclipse encontramos multitud de opciones de refactorización [9]. No obstante, estas constituyen más una ayuda para la aplicación de un patrón que la propia aplicación de este. Son ayudas que nos permiten la rápida ordenación del código para un patrón propuesto. También podemos encontrar este tipo de ayudas en la refactorización en la mayoría de los entornos. A modo de ejemplo podríamos nombrar también IntelliJ. [10]

Finalmente hay que mencionar que, a pesar de todos los recursos comentados, existe una ausencia de herramientas que permitan trabajar con los patrones más allá de la aplicación de algunas refactorizaciones en los entornos de desarrollo.

2.2 Propuesta

La propuesta de este trabajo es la de una herramienta usando una tecnología web que ofrezca utilidades para aprender los patrones y trabajar con ellos. Esto se llevará a cabo mediante el uso de las herramientas para obtener conocimiento de la información existente del campo presentada en el apartado anterior. Esto se realizará por ejemplo construyendo un ayudante para recomendar un patrón para un problema a través de la información de las aplicaciones comunes de los patrones. A esto le sumamos que sea una aplicación web para hacer esta información accesible.

Pretende innovar en este campo proporcionando una nueva forma de trabajar con patrones de diseño, que pueda ser mejorada de forma iterativa hasta conseguir soluciones más complejas o tomarse como referencia para desarrollar otras aplicaciones similares.

También se proporcionarán recursos, existentes en la actualidad, como información y ejemplos de patrones. Esto se hará mediante la inclusión de contenido que intente aportar otro punto de vista a la explicación del patrón o a su posible uso.

3. Análisis del problema

3.1 Identificación y análisis de las soluciones

3.1.1 Necesidades de la aplicación

La aplicación, al enfrentarse a un reto que se podría considerar novedoso ya que no ha sido muy explorado. Debería de cumplir con las siguientes características:

- **Ofrecer información no ambigua**: No podemos ofrecer información que pueda no ser clara para el usuario, ya que debemos de poder orientarlo de forma que pueda encontrar el patrón que busca para su solución, en caso de poder encontrar uno.

- **Evitar imprecisión en las recomendaciones**: Es preferible no encontrar un patrón que solucione el problema a recomendar uno incorrecto, ya que esto podría causar confusión o frustración al usuario. Además, de llegar a aplicarse podría causar aún más problemas. Podría ser interesante recoger información de cuando no ha sido posible encontrar un patrón para poder guiar al usuario mejor la próxima vez.

- **Mejora y extensión continua**: Debe ser posible ir mejorando las herramientas, tanto facilitando la mejora por código como guiándolo la aplicación por los datos proporcionados por los administradores de esta. Gracias a esto podrá ir mejorándose y convirtiéndose en una mejor aplicación.

3.1.2 Complejidad de la herramienta

Dado que nos encontramos ante una aplicación novedosa, es necesario realizar un análisis de las complejidades a encontrar para poder saber a qué riesgos nos enfrentamos y como debería abordarse la aplicación.

- Precisión de la herramienta: Es complejo plantear una solución concreta. A veces, existen problemas que podrían ser abordados de varias formas o que su solución real es la aplicación de varios patrones. La forma de abordar esto sería dar como solución uno de los patrones que puede solucionar el problema y mostrar dentro del mismo, en un apartado de sinergias las posibles combinaciones con el mismo.

- Manejo de datos: Una aplicación de este volumen tiene mucha información cambiante y sobre la que se debe de ir aprendiendo. Una herramienta de estas características debe ser configurable mediante ficheros de datos además de mediante código para permitir su fácil restructuración. Así se garantizará que funciona de forma correcta.

- Falta de aplicaciones similares que tomar como referencia: Aunque existen multitud de fuentes que informan de los distintos patrones de diseño, hacer herramientas para los mismos tiene cierta dificultad. No se ha encontrado nada público que haga unas funciones planteadas a las del trabajo. Esto por ejemplo hace que, a la hora de plantear preguntas para discriminar entre varios patrones de diseño, no podamos encontrar algo parecido. Esto hará que a la hora de generar una herramienta como un cuestionario se tengan que generar este tipo de preguntas a partir de las características de cada patrón de diseño.

3.1.3 Modificación de la herramienta

Al estar ante un panorama incierto, se debe construir la herramienta pensada de forma que pueda recibir modificaciones en el futuro. Es por ello, que debe de ser construida lo más desacoplada posible de forma interna. También, debería construirse una batería de pruebas unitarias para garantizar que el funcionamiento de la aplicación es el correcto.

Un motivo importante para tener una buena cantidad de pruebas es que la aplicación desarrollada sea de código abierto y pueda ser modificada por una cantidad de desarrolladores indeterminada con el fin de brindar nuevas características. Incluyendo

estas pruebas, garantizaríamos que la funcionalidad sigue permaneciendo de acuerdo con sus requisitos esenciales.

3.1.4 Tratamiento de la información del usuario

Por la herramienta pueden pasar una serie de datos de usuarios que pueden tener carácter sensible. Interesa tener los datos anonimizados lo máximo posible. Esto se debe a tratamos con datos que pueden ser de carácter delicado. Por ejemplo, en caso de exponer un fragmento de código podríamos descubrir una vulnerabilidad en una aplicación.

Esto se va a abordar sin almacenar la información de los usuarios. Esto quiere decir que serán identificados mediante una credencial y nunca se almacenará información, si no resultados. Encontramos los siguientes casos:

- Cuestionarios: Se identificará al usuario por una ID. Mostraremos la pregunta y el resultado final, pero no la traza de preguntas mostradas o datos como la IP de la solicitud u otra información presentada. Tampoco almacenaremos en las cookies del navegador esta información para poder proteger de ataques esta información.

- Herramientas de análisis de código: Aunque estas herramientas no se plantean incluirse en el primer entregable de este proyecto, si se contempla como trabajo futuro. Por este motivo será necesario hacer un análisis adecuado de esta funcionalidad. Para esto se plantea hacer un tratamiento de la información y mostrar los resultados al usuario por pantalla. Al mismo tiempo, se le ofrecerá la posibilidad de enviarlo por email a uno proporcionado. Una vez hecho esto, no se volverá a mostrar estos resultados, y de forma periódica será borrada la información en caso de existir alguna sensible. Así se garantiza que, si por ejemplo un usuario proporciona un proyecto entero, este sea eliminado después de su tratamiento y de mientras no sea accesible.

3.1.5 Solución a Problemas actuales

El problema que tenemos actualmente a la hora de trabajar con patrones es su dificultad de aplicación y que no suelen ser prioritarios en el desarrollo. Pero la realidad hoy en día es que los profesionales informáticos invierten casi un día de trabajo a la semana en solucionar deuda técnica [11]. Esta situación vería reducida su frecuencia en el tiempo si los patrones se estudiaran, entendieran y aplicaran más frecuentemente cuando son necesarios. Esto, acompañado de otras ayudas como documentación o código claro harían que esta situación mejorase notablemente.

Además, como hemos visto en el estado del arte, actualmente no existe una herramienta que nos proporcione de forma interactiva métodos para trabajar, identificar o aplicar los patrones de diseño. Esto se propone llevar a cabo desarrollando una serie de herramientas que permitan ayudar al uso de patrones de diseño.

3.1.6 Encuesta a exalumnos

El proyecto ha tenido dificultades debido a la pandemia ya que no se ha podido encuestar de forma presencial sobre la aplicación para obtener información de herramientas de utilidad para la misma. No obstante, se ha realizado una adaptación en formato de encuesta online, que, aunque más limitada ha proporcionado la información necesaria de requisitos del proyecto.

Con esta encuesta se planea obtener más información acerca de las características que podría tener la aplicación y de sus herramientas. Este cuestionario tuvo como objetivo obtener un mayor conocimiento sobre aquellos aspectos que el usuario final considera más relevantes en la aplicación.

La aplicación desarrollada en este Trabajo de final de grado tiene un fuerte peso en el ámbito educativo ya que será utilizada principalmente para facilitar la formación en el ámbito del diseño de software. Hemos seleccionado como público objetivo los alumnos que estén cursando o que acaben de cursar la rama de Ingeniería del Software. Este grupo está trabando con patrones de diseño o lo ha hecho recientemente. Con el fin de comprender a nivel educativo las posibles dificultades a las que se pueden enfrentar los alumnos de ingeniería informática al aprender patrones.

La encuesta se ha realizado a 33 alumnos de la asignatura de Diseño de Software de la rama de Ingeniería del Software en la cual se trabaja principalmente con patrones de diseño. A estos alumnos se los ha entrevistado cuando estaban a punto de terminar el

grado y ya habían superado esta asignatura. Esto se hace con el fin de ver qué entendimiento alcanzan los alumnos y a qué se enfrentarán en el inicio de su mundo laboral. De esta forma, esta herramienta podrá diseñar soluciones para facilitarles la incorporación en cuanto a la aplicación de patrones y ver qué aspectos pueden ser de utilidad como herramientas para el aprendizaje, ya que se pueden diseñar herramientas para reforzar las posibles carencias o dificultades.

Las preguntas han sido definidas a partir de una serie de reuniones entre el tutor y el alumno de este trabajo. En estas se han ido planteando varias herramientas y utilidades para poder listarlas y preguntarles a los alumnos por ellas. Además, preguntamos de manera informal a algunos alumnos por ideas para esta lista. Después, se decidió incluir al final de la encuesta también un espacio donde el alumnado pudiera escribir algunas sugerencias extra para la aplicación que no se hubieran podido identificar y plasmar en ningún punto anterior.

.

El cuestionario ha sido realizado aplicando una escala LIKERT a excepción de las dos últimas preguntas que son de SI/NO y de respuesta libre. Esto nos da las siguientes posibles respuestas para cada afirmación:

1	2	3	4	5
Totalmente en desacuerdo	En desacuerdo	Ni de acuerdo ni en desacuerdo	De acuerdo	Totalmente de acuerdo

Al principio del cuestionario, se les planteó a los usuarios la dificultad de abordar dos aspectos clave en la aplicación de patrones de diseño. Estos eligieron en la escala LIKERT la dificultad de este.

- **Identificar un patrón a aplicar**: Se preguntó a los usuarios por la dificultad de decidir qué patrón debían utilizar para resolver un problema concreto. La pregunta concreta fue ***"¿Qué grado de dificultad consideras que tiene la tarea de elegir el patrón a aplicar?***".
- **Implementar un patrón**: Esta pregunta se hizo a los alumnos usuarios para entender la dificultad de, una vez determinado el patrón a aplicar, implementarlo e integrarlo en un código existente. La pregunta exacta fue "Implementar patrones en el desarrollo de una aplicación es una tarea:" la cual tiene como respuesta el nivel de dificultad que le encuentran a la hora de hacer esta implementación.

Los alumnos puntuaron del 1 al 5 en la escala presentada anteriormente la dificultad de estos dos tipos de tareas. Como resultado obtuvimos las siguientes puntuaciones:

Dificultad de un concepto [1,5]	Mediana	Media	Suma
Identificar patrón a utilizar	**4**	3,66	121
Implementar un patrón	**3**	3,21	106

Podemos ver que tenemos una mediana de 4, en cuanto a la dificultad de identificación de un patrón. Esto significa que se considera difícil realizar esta tarea. Por otro lado, observamos que la mediana es de 3 en cuanto a su implementación. Hemos tomado como referencia la mediana ya que nos permite obtener un resultado más preciso, ignorando las respuestas que pueden estar más en el límite y que si alteran otros valores.

Ya que se considera que tiene más dificultad su identificación que su implementación, se dará prioridad a herramientas relacionado con ello. No obstante, se recogerá información de herramientas relacionadas con su implementación ya que también son importantes y de ayuda para los usuarios.

Además de estas cuestiones, introdujimos una pregunta donde debían puntuar algunas características que consideraban que la futura aplicación debería incorporar. Se ha ignorado la respuesta de No sabe / No responde a la hora de realizar el cálculo:

Importancia característica [1,5]	Mediana	Media	Suma
Explicación del patrón	**5**	4,51	140
Generación del código del patrón	**4**	3,6	119
Generación automática de test unitarios (JUnit) para probar el patrón	**3**	3,18	105
Quiz para comprobar tus conocimientos sobre patrones	**2**	2,09	69
Wizards o ayudantes que te permitan averiguar qué patrón puede ser más útil para resolver tu problema	**4**	4,03	125

Wizard o ayudante que compruebe si un patrón determinado es adecuado para resolver tu problema	**4**	4	124
Ejercicios didácticos para aprender patrones	**3**	3,03	100
Identificación patrones en el código fuente	**4**	4,13	128

En estas respuestas obtuvimos información para priorizar dentro de la categoría de identificación del patrón y de ayudas en la implementación el orden de importancia de cada característica para el usuario.

Como podemos observar, dentro de la categoría de identificación de patrones de diseño, podemos encontrar como herramientas más solicitadas los ayudantes para identificar si un patrón es válido o encontrar un patrón para un problema concreto. Por ello en este TFG se le dará prioridad a este tipo de utilidades. También podemos encontrar la herramienta de identificación en el código fuente. Dado que el código fuente puede variar dependiendo del lenguaje de programación, se decide dar prioridad a los *wizards* para encontrar un patrón ya que son más generales e independientes del lenguaje de programación. Además, esta se considera una mezcla entre identificación y aplicación ya que estamos en un caso práctico donde ya se está poniendo en uso el conocimiento y no solo buscando una posible solución.

Dentro de la categoría de aplicación de patrones de diseño, encontraríamos como herramienta más valorada la de generación de código del patrón. Además, dentro de la categoría general y como punto más valorado encontramos la explicación del patrón.

Además, se ha realizado esta pregunta ***"¿Estarías interesado en utilizar una aplicación de este tipo si trabajaras habitualmente con patrones?"*** Donde encontramos validada la idea del TFG, teniendo en cuenta la suma de la opción de SI, obtenemos un 90,9% de aceptación y un 9,1% de negación de la idea.

Por último, exponemos la lista de otras funcionalidades que los usuarios han destacado que les gustaría ver. Casi la mitad de las personas han mostrado algunas ideas. A continuación, mostramos algunas consideras interesantes:

1. *Wizard o ayudante para comparar el programa antes y después de usar el patrón.*

2. *Sistema de aportes de la comunidad, para introducir patrones que no hayan sido considerados por el desarrollador, y que puedan ser programados por una*

persona externa al equipo de desarrollo de la aplicación, para ser aprobados o no más tarde.

3. *Aparte de la implementación de patrones, podrías implementar una función que genere un diagrama de clases del patrón implementado en el proyecto.*

4. *Es un problema muy tedioso para nosotros como estudiantes el hecho de aplicar patrones. Creo que todos tuvimos problemas en ver donde se podían aplicar.*

5. *Normalmente tenemos problemas para saber dónde podríamos aplicar un patrón de diseño, es decir, en su identificación. Solemos preguntarnos: "¿Dónde puedo aplicar este patrón?".*

6. *Me gustaría ver ejemplos de aplicación de un patrón y contra ejemplos de donde no hacerlo.*

7. *Me gustaría ver ejemplos de aplicación de patrones en casos reales.*

Además de estos comentarios mostrados, había más que expresaban la dificultad de encontrar un patrón para un problema. Estos se han obviado para evitar repetición en el texto. También, se muestran otros casos que se podrían incorporar a la aplicación, como los ejemplos en aplicaciones reales. Otras opciones quedan fuera del alcance lo que intenta hacer la aplicación, al menos en su planteamiento inicial. No obstante, pueden ser tenidas en cuenta como trabajo futuro.

Conclusiones de la encuesta

La encuesta nos ha servido para identificar que lo que más valora el potencial usuario de este tipo de herramienta. Busca herramientas que ayuden a la detección antes que a la programación. No obstante, la detección automática puede cometer errores y será importante el criterio del usuario a la hora de aceptar la sugerencia o no.

Se dará prioridad a una explicación del patrón y un ayudante que pueda ayudar a dado un problema, elegir que patrón es más correcto. Posteriormente, como trabajo futuro, debería de realizarse el análisis del código donde se podría aplicar un patrón de diseño.

3.2 Solución propuesta

A partir del feedback recibido y nuestros objetivos iniciales, estas son las características de la solución que se desarrollará en la aplicación:

- Aplicación web: Se desarrollará una aplicación en formato web para ser accesible por un gran número de público. Esto permitirá un desarrollo en una tecnología más novedosa y con escasas fronteras para acceder.

- Tecnologías: Se propone Java y Spring como tecnologías principales para el *Backend*. Esto se debe a que el alumno no tiene conocimientos de programación web, pero sí de Spring lo cual le facilita la construcción del sistema. También se ha decidido esto por la amplia gama de herramientas que proporciona Spring. Además, se usarán otras tecnologías como HTML con Bootstrap 5 en el *FrontEnd* o con MongoDB para la base de datos.

- Búsqueda de Patrones: Se ofrecerá un cuestionario que mediante una serie de preguntas nos responderá con una sugerencia de que patrón debería emplearse.

- Panel administrativo: Se proporcionará un panel administrativo mediante el cual se puede configurar el contenido de la web. Este servirá para realizar una mejor gestión del contenido de esta e ir mejorándolo.

- Explicación del patrón: Se ofrecerá una documentación variada de cada patrón con el fin de conocer más de él.

Como resultado queremos obtener una aplicación como mínimo producto viable) con las condiciones mencionadas previamente. Es importante recalcar que quedaría como trabajo futuro la implementación de otras características de interés remarcadas por los alumnos, como ayudas para la implementación.

3.2.1 Listado de usuarios

Alumnos/Estudiantes: Este rol será ejercido por los usuarios que utilicen la herramienta con el objetivo de ampliar sus conocimientos. Su uso será el de emplear las distintas herramientas para obtener la información necesaria para su trabajo con los patrones de diseño.

Se parte de que tienen unos conocimientos básicos del contenido de los patrones de diseño, aunque se ofrecerán ayudas en caso de partir de cero.

Profesorado: Su rol es del de administradores del sitio. Podrán mediante la sugerencia de los usuarios y sus propios conocimientos ir mejorando la información y herramientas de este, de forma totalmente independiente del código en un panel administrativo.

Se supone que son expertos en los patrones, aunque no necesariamente en la implementación del servidor.

3.2.2 Casos de uso

Encontraremos los siguientes casos de uso divididos en las siguientes secciones:

Figura 1 Diagrama general de casos de uso

Acceso a información de la portada

Será necesario la construcción de una portada en la cual se brinde información sobre la página web y se ofrezca el acceso a sus distintos apartados. Entre estos encontraremos una lista de patrones, una página con un cuestionario para encontrar un patrón y una página de auto

CU Presentación de información general

Descripción	**El usuario podrá visualizar en la página principal información y acceder al contenido de la aplicación y sus distintas herramientas.**
Actores	Estudiante
Precondición	Vista de página principal.
Observaciones	En este caso de uso no se tiene en cuenta la opción de búsqueda de patrón, que será probada en otro caso de uso.
Flujo de eventos	1. El actor entra en la página principal 2. El sistema devuelve la visión de dicha página 3. El actor va navegando por los distintos menús 4. El sistema carga correctamente cada vista con sus datos
Postcondición	Desde cualquier menú que se haya terminado la navegación, puede volverse a los otros.

Consulta de* información *de patrones

Al ser una aplicación sobre los patrones de diseño es necesario dedicar una sección de la aplicación a proporcionar contenido individualizado sobre cada patrón.

CU Lista de patrones

Descripción	**El usuario podrá observar una lista con todos los patrones de la aplicación y una breve descripción de ellos.**
Actores	Estudiante
Precondición	Vista de ventana principal.
Observaciones	En caso de no haber ningún patrón dado de alta se mostrará un mensaje. Esto se hace para evitar fallos en caso de no poder cargar los patrones en algún momento.
Flujo de eventos	1. El usuario accede al menú de patrones 2. El sistema devuelve una lista de los distintos patrones

Postcondición	Si se abandona esta lista y se vuelve a acceder el resultado será el mismo.

CU Ficha de patrón

Descripción	**Se presentará una ficha del patrón con información de este, sinergias, un ejemplo del mundo real y referencia a la documentación.**
Actores	Estudiante
Precondición	Vista de patrones.
Observaciones	En caso de no poder acceder a un patrón o no existir se mostrará un mensaje de error notificando esta situación del sistema.
Flujo de eventos	1. El sistema ofrece una lista de patrones para dicha vista 2. El usuario accede a un patrón de diseño concreto 3. El sistema devuelve una vista con información de ese patrón concreto 4. El usuario puede consultar todos los datos del patrón disponibles.
Postcondición	El patrón no debe ser modificado después de ser consultado o perderse alguna información.

CU Se ocultan los campos no existentes

Descripción	**En caso de no existir información de un campo concreto para un patrón de diseño, esta no se presentará visualmente.**
Actores	Estudiante
Precondición	Vista de patrones.
Observaciones	En caso de no existir un patrón registrado en el sistema, se mostrará un mensaje de que este patrón todavía no se ha incluido. Todos los campos menos el nombre y la descripción son opcionales.
Flujo de eventos	1. El sistema ofrece una lista de patrones disponibles. 2. El usuario accede a uno concreto. 3. El sistema encuentra un patrón, pero este tiene campos no presentes. Este procesa la vista para ofrecerla en un estado correcto sin mostrar estos campos como vacíos. 4. El usuario puede consultar la información existente del patrón.

Postcondición	No se debería de modificar el patrón después de la consulta, permaneciendo esos campos sin información.

Realización de cuestionario de recomendación de patrón

Esta es una parte fundamental de la aplicación ya que será la principal herramienta que se brindará al usuario para ser capaz de encontrar el patrón de diseño buscado. Consistirá en un cuestionario con un algoritmo de poda.

CU Las preguntas se ordenan de forma correcta

Descripción	**En caso de existir un orden o estrategia de ordenación para las preguntas, estas serán capaces de aparecer en el orden correcto según las preguntas del usuario.**
Actores	Estudiante
Precondición	Ventana principal
Observaciones	Se tiene en cuenta que se han cargado correctamente las preguntas del patrón. Un cambio de orden no implica que las preguntas se puedan cargar de forma incoherente.
Flujo de eventos	1. El usuario accede al cuestionario 2. El sistema devuelve una lista de preguntas 3. El usuario va contestando a las preguntas 4. El sistema va mostrando la batería de preguntas a mostrar en el orden determinado
Postcondición	El orden de preguntas debe ser siempre el mismo para unas preguntas dadas en el próximo cuestionario, salvo que en futuras iteraciones del producto este sea capaz de mejorar el orden de presentación de forma automática.

CU El cuestionario se mantendrá en el tiempo

Descripción	**Si el usuario quiere pasar momentáneamente el cuestionario para revisar cosas del problema y volver más tarde a él, podrá guardarse él id del cuestionario y volver a él luego.**
Actores	Estudiante
Precondición	Ventana principal
Obversaciones	El cuestionario será identificado por una ID generada aleatoriamente al empezar a realizarlo. El cuestionario podrá volver a ser accedido, aunque se haya finalizado. En este caso, nos dirigirá a la solución de este.

	Cada cuestionario debe de almacenar la fecha en la que se inició, para saber cuánto tiempo lleva empezado.
Flujo de eventos	1. El usuario accede al cuestionario 2. El sistema devuelve una lista de preguntas 3. El usuario va contestando a las preguntas 4. El sistema va sirviendo la respuesta a las preguntas del usuario 5. El usuario abandona el cuestionario 6. El usuario vuelve un tiempo después a acceder a la dirección web que identifica al cuestionario 7. El sistema lo servirá en el estado que se encontraba
Postcondición	El cuestionario debe de permanecer en base de datos por si el usuario quiere continuarlo más tarde, por lo menos durante una semana. Ya que se pretende almacenar las respuestas, puede ofrecerse al usuario una traza de las respuestas que se habían dado.

CU Se eliminarán las preguntas innecesarias

Descripción	**Durante la realización del cuestionario se irán eliminando las preguntas que ya no aporten contenido a la obtención del resultado, reduciendo así el espacio de búsqueda.**
Actores	Estudiante
Precondición	Ventana principal
Flujo de eventos	1. El usuario accede al cuestionario 2. El sistema devuelve una lista de preguntas 3. El usuario va contestando a las preguntas 4. El sistema va sirviendo la respuesta a las preguntas del usuario 5. El usuario da una respuesta a una pregunta que elimina posibles respuestas 6. El sistema elimina las preguntas que ya no pueden guiar a una solución
Postcondición	Esta eliminación de preguntas no puede penalizar gravemente al algoritmo, ya que podemos tener varios usuarios realizando este cuestionario al mismo tiempo.

CU Se deriva al patrón solución

Descripción	**Cuando se llega a una solución se cargará la página del patrón de esta, para que el usuario pueda obtener información del patrón, uniendo así ambas partes de la aplicación. S**
Actores	Estudiante
Precondición	En caso de que no se pueda encontrar una solución se le notificará al usuario de que no existe una solución con las respuestas proporcionadas. Si no existe información del patrón, pero si es una solución posible, debe de mostrarse el nombre del patrón solución como mínimo. Aun así, se recomienda dar de alta como mínimo el nombre del patrón, una descripción y un enlace para obtener información para evitar estos casos.
Observaciones	En caso de no existir
Flujo de eventos	1. El usuario da una respuesta que lleva a una solución 2. El sistema carga la página del patrón y la devuelve 3. El usuario puede consultar la información del patrón concreto
Postcondición	El cuestionario se debe de marcar como finalizado una vez este acabe, y almacenarse su solución.

Control de la aplicación mediante el panel administrativo

Ya que la creación de los datos de la aplicación se ha propuesto la construcción de la misma basada en los datos que se obtengan desde la base de dados. Estos datos se quieren ir modificando desde un panel administrativo en lugar de estar prestablecidos en código o en base de datos. Contaremos con información de patrones, preguntas de los cuestionarios, o configuración del servidor. Es por esto por lo que se construye un panel administrativo para la modificación de esta información. Esto permite que la información se mejore de forma iterativa.

No se contempla el caso de uso de cerrar sesión ya que se considera trivial. En cambio, el de iniciar sesión si por la naturaleza de la configuración de la contraseña.

Por seguridad no se presenta ningún acceso a información de la base de datos como los cuestionarios en este panel. Únicamente a datos como los cuestionarios o las preguntas, es decir, que no pertenezcan a ningún usuario.

CU Se puede acceder al panel administrativo con la contraseña configurada

Descripción	**Existirá una contraseña configurada en el archivo de despliegue del servidor web. Mediante esta, se podrá acceder al panel administrativo**
Actores	Profesorado
Precondición	Haber configurado la contraseña en el archivo de propiedades de la aplicación web.
Observaciones	Esta contraseña solo puede ser configurada desde este archivo, por motivos de seguridad. Esto se debe a que no se quiere exponer la posible modificación de esta con un uso malintencionado. En caso de intentar acceder a una página administrativa o hacer una petición sin estar autentificados no se nos permitirá. Se nos redirigirá a la página de inicio de sesión.
Flujo de eventos	1. El administrador accede a la página de acceso administrativo. 2. El sistema detecta que no tiene la sesión iniciada y le pide las credenciales. 3. El administrador introduce los datos correctos de las credenciales del servidor. 4. El sistema guarda su autentificación.
Postcondición	Una vez iniciada sesión no le hará falta volver a introducir las credenciales para hacer operaciones salvo que cierre la misma.

CU Es posible modificar la información de los patrones

Descripción	**Mediante el panel de patrones se podrán modificar, añadir o cambiar patrones de la aplicación.**
Actores	Profesorado
Precondición	Sesión iniciada
Observaciones	Se contará con un formato para la introducción de datos que hará que el profesorado no deba tener conocimientos de la implementación para la modificación de estos.
Flujo de eventos	1. El administrador accede a la página de configuración de patrones. 2. El sistema devuelve la información de los patrones existente procesada para el usuario. 3. El administrador cambia datos de los patrones y los actualiza. 4. El sistema procesa la información y modifica los datos con la nueva información

Postcondición	El usuario podrá observar de forma directa los cambios en la aplicación

CU Las preguntas del cuestionario se pueden alterar

Descripción	**En el menú de cuestionario se podrán modificar, añadir o cambiar preguntas del cuestionario de la aplicación.**
Actores	Profesorado
Precondición	Sesión iniciada
Observaciones	Se contará con un formato para la introducción de datos que hará que el profesorado no deba tener conocimientos de la implementación para la modificación de estos.
Flujo de eventos	1. El administrador accede a la página de configuración de preguntas. 2. El sistema devuelve la información existente de las preguntas existentes procesada para el usuario. 3. El administrador cambia datos de las preguntas y las actualiza. 4. El sistema procesa la información y modifica los datos con la nueva información.
Postcondición	El usuario podrá observar de forma directa los cambios en la aplicación

CU La información sobre la configuración puede ser comprobada

Descripción	**Se ofrecerá documentación sobre cada parámetro del archivo de configuración.**
Actores	Profesorado
Precondición	Sesión iniciada
Observación	No se expondrá la información real de la configuración como ejemplo. Tampoco modificarla de forma directa o consultar información sensible como la contraseña administrativa u otros datos reales de configuración.
Flujo de eventos	1. El administrador accede a la página de información de la configuración de la aplicación. 2. El sistema devuelve la información existente de las preguntas existentes procesada para el usuario.
Postcondición	Los datos de la configuración permanecerán sin ser modificados.

CU Se pueden acceder a estadísticas de la aplicación

Descripción	**Se ofrecerá una serie de estadísticas de rendimiento y otros de la aplicación**
Actores	Profesorado
Precondición	Sesión iniciada
Observación	También se puede ver en la información de presentación del panel administrativo. La información que mostrar puede ser modificada en el archivo de configuración del servidor.
Flujo de eventos	1. Acceder a la ventana de inicio de administración. 2. El sistema devuelve información sobre las estadísticas de la aplicación.
Postcondición	La información mostrada para las estadísticas no expondrá datos críticos del servidor.

3.3 Plan de trabajo

Para les estimaciones de este trabajo usaremos las horas de programación por persona. Para cada *Sprint* se decidirán unas horas disponibles de programación y se estimada dejando un 20% de horas disponibles para poder cubrir cualquier tiempo que exceda la planificación.

El trabajo se dividirá en tres *Sprints* donde cada uno construirá una mejor versión, pero siempre respetando el principio del mínimo producto viable. Cada uno dispondrá de una estimación de horas de programación, aunque también tendremos en cuenta la duración de otras tareas en el cómputo global de trabajo del *Sprint*. Se realizará un *Sprint* previo a los tres para analizar el trabajo a realizar.

Además, al final de cada *Sprint* revisaremos la metodología utilizada en el trabajo con el fin de mejorarla con los conocimientos adquiridos en procesos de software durante el grado. Esto hace que del Sprint 1 al Sprint 2 se realicen los siguientes cambios:

- Eliminación de las pruebas de aceptación en la fase de programación: Al ser un solo programador, no será necesario pasar dos veces las pruebas como se hace en la metodología normal (en programación y en la fase de pruebas). Se reservarán únicamente para la fase de pruebas

- Unión de fase de diseño y especificación: Se realizarán en una única frase donde se especificarán los requisitos y el posible diseño del software.

Además, cabe destacar que algunos aspectos como la reunión diaria no han sido realizados ya que carecen de finalidad en equipos que contienen un único desarrollador.

En el paso del Sprint 2 al Sprint 3 no se ha realizado ninguna mejora metodológica.

4. Diseño de la solución

4.1 Arquitectura del sistema

La aplicación web es accedida mediante el usuario mediante su navegador. Se basará en la arquitectura de **tres capas**. Este tipo de arquitectura es conocido por su separación entre presentación, lógica y persistencia. En la Figura 2 podemos observar la distribución mencionada.

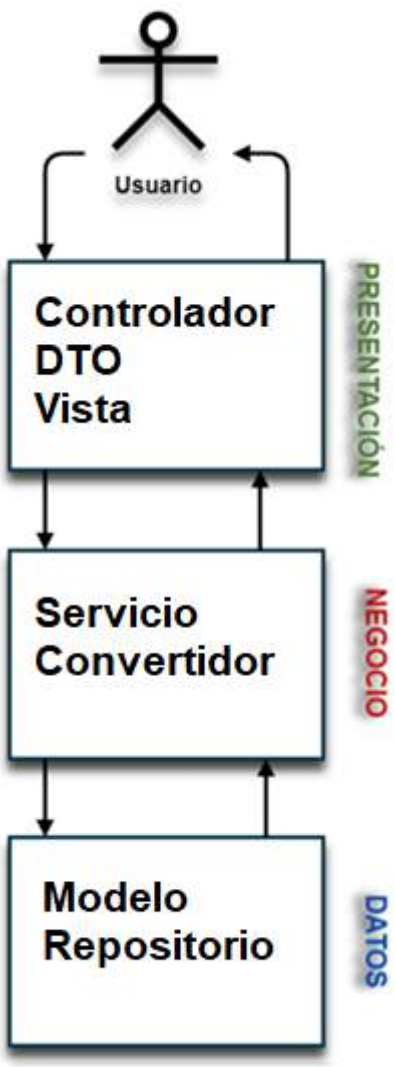

Figura 2 Esquema básico arquitectura

4.1.1 Capa de presentación

En la capa de presentación encontramos todas las interfaces de la aplicación. Esta capa será la encargada de:

- Recibir y procesar las peticiones del usuario: Recibirá las peticiones, y después de procesarlas y validarlas se las enviará a la capa de negocio de una forma que la comprenda.

- Presentar la información proporcionada desde la capa de negocio: Data una respuesta de la capa inferior, será capaz de procesar los resultados y mostrarlos de forma visual de una forma adecuada para el usuario.

En el caso concreto de una aplicación web, en esta capa se gestionará la generación del documento HTML y la traducción de las peticiones HTTP en datos que pueda entender la capa de servicio.

Sobre nuestra aplicación, cobrara sentido a la hora de traducir un objeto que contiene información sobre un patrón a una interfaz donde pueda leerse la misma, personalizándola para cada caso concreto. También se hará uso de esta capa cuando se recoja una entrada del usuario en un ayudante y se envié a la capa inferior para procesarla.

Es necesario conocer los siguientes términos que emplearemos para referirnos a ciertas partes concretas de la capa:

- **DTO**: Este objeto utilizado en presentación y lógica contendrá información independiente de la base de datos. Su finalidad es la transmisión de información de servicio a presentación de forma bidireccional.

- **Vista**: Será la forma de presentar la información al usuario de forma visual con los datos obtenidos. Normalmente será mediante el procesamiento de información de un DTO.

- **Controlador**: Sera el encargado de recibir las peticiones, validarlas y gestionarlas. Este capturará una serie de peticiones HTTP e intentará traducirlas de una forma que sean comprensibles para la lógica de la aplicación mediante objetos DTO. También proporcionaría vistas determinadas para una petición, basándose normalmente en información provista en DTOs para la construcción de esta.

4.1.2 Capa de negocio

Esta capa será la encargada de realizar todas las operaciones lógicas de la aplicación. En concreto realizará estas funciones:

- En caso de recibir una petición de la presentación: Consultar la capa inferior en busca de datos en caso de ser necesario y devolver una respuesta después de ejecutar la lógica correspondiente.

- En caso de tener una tarea programada o recibir una petición ajena a la presentación: Ejecutar la lógica relacionada con la misma.

En nuestra aplicación, esta capa se encargará de traducir datos entre persistencia y lógica de forma que se puedan enviar también a la presentación. Por otro lado, realizará la operación inversa para poder almacenar información. También se encargará de realizar todos los cálculos u operaciones lógicas que se deben realizar.

Esto aplicado sobre la aplicación lo encontraremos en la obtención de la información de un patrón desde la base de datos u obtener una respuesta a una pregunta de un ayudante de recomendación de patrón.

Es de gran relevancia destacar la siguiente nomenclatura que será utilizada para referirse a algunos aspectos de esta capa:

- **Servicio**: Será la clase encargada de gestionar las peticiones del controlador y servirse de la capa inferior para obtener datos del repositorio. Normalmente expondrá una serie de métodos y para ser usados desde la presentación. Además, incorporará una conversión entre base de datos y presentación. Normalmente al trabajar con objetos nos proporcionada una interfaz CRUD (Creación, Lectura, Modificación y Eliminación). Para este caso, se encargará de traducir la petición del controlador en una operación CRUD que pueda entender la base de datos. En otros casos como los ayudantes, contendrá la lógica de estos para procesar su información y dar una respuesta al usuario.

- **Convertidor**: Es la clase encargada de convertir entre presentación y persistencia. Es decir, entre DTO y Modelo (Concepto explicado en el siguiente apartado).

4.1.3 Capa de datos

Por último, encontramos la capa de datos. Esta será la encargada de obtener la información desde la base de datos y de almacenarla. Realizará las siguientes funciones:

- Guardado de datos: Actualizara información con los datos proporcionados.
- Modificación de datos: Con la información presentada, obtendremos una nueva versión modificada de la existente en la base de datos.
- Eliminación: Dada una referencia a un objeto, eliminaremos el objeto o la lista de objetos que correspondan con el mismo.

Específicamente, encontraremos estos aspectos por ejemplo en la gestión de la vida útil de un cuestionario. Primero lo crearemos y almacenaremos con los datos concretos. Mas tarde con la referencia que lo identifica iremos modificándolo y almacenándolo según recibamos respuestas del usuario. Finalmente, pasado un tiempo de la existencia de este lo eliminaremos por creerlo ya no necesario.

En cuanto a la persistencia, debemos de comprender los siguientes conceptos:

- **Modelo**: Es el objeto que da una representación cercana a la información a la base de datos. Esta clase puede ser entendida por la base de datos y utilizada para almacenar esta información o devolverla a un servicio.
- **Repositorio**: Nos referimos con esto a la implementación que ofrece una serie de métodos para trabajar con la base de datos de forma indirecta y sencilla.

4.2 Tecnología utilizada

En este apartado explicaremos el contenido a nivel tecnológico de cada capa mencionada y que otras tecnologías se han utilizado para el proyecto. En la Figura 3 podemos observar a grandes rasgos las principales tecnologías que componen cada capa.

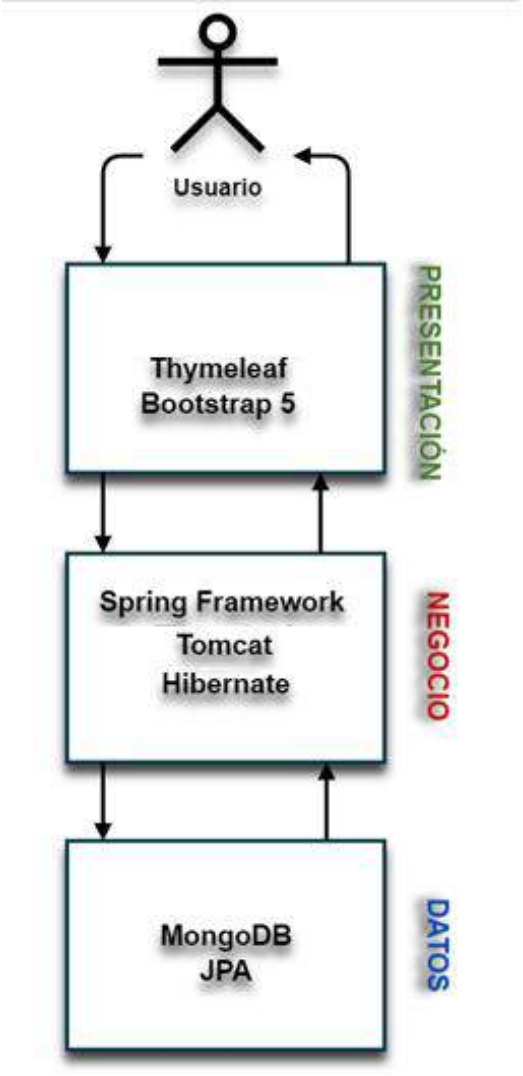

Figura 3 Arquitectura detallada de la aplicación

4.2.1 Capa de presentación

A la hora de implementar la parte gráfica de la aplicación web se ha recurrido a las siguientes soluciones software:

***Thymeleaf**:* Usada para mostrar la información en la interfaz y obtener las entradas del usuario. Es una librería que nos brinda la posibilidad de usar objetos Java para trabajar con la vista, accediendo de forma directa a sus propiedades

Bootstrap 5: Es una utilidad que nos da una serie de estilos CSS que nos permiten la configuración de la vista de forma más sencilla, en base a una guía de parámetros proporcionados por la herramienta. Esto hace que construir interfaces sea más sencillo y rápido. **Prometheus**: Es un cliente que nos permite acceder a las estadísticas del servidor expuestas desde el panel administrativo. Este procesa la información expuesta y nos la devuelve en formatos entendibles para el ser humano. Además, ofrece una serie de utilidades para realizar consultas sobre la información.

4.2.2 Capa de negocio

Para la lógica de la aplicación se han empleado las siguientes tecnologías:

Spring Framework: Hemos empleado Spring para realizar la parte del servidor de la aplicación (*Backend*). Esta decisión se ha tomado por sus utilidades a la hora de construir aplicaciones web. También cabría destacar su gestión de las dependencias, ya que mediante inyección de dependencias permite realizar un sistema más independiente entre sus partes. Por último, querríamos destacar el soporte que incluye nativo para construir esta arquitectura por capas o sus ayudas para la configuración de seguridad de forma nativa.

Mockito y Junit 5: Hemos utilizado estas tecnologías para la realización de una serie de pruebas unitarias para la aplicación, con el fin de validar su comportamiento ante la presencia de cambios.

Modelmapper: Esta librería se ha añadido en etapas finales del desarrollo. Permite realizar transformaciones entre modelo y persistencia de forma más sencilla. **Lombok**: Mediante esta incorporación los objetos pueden ser creados más fácilmente. Por ejemplo, puede encargarse de crear los accesos a sus propiedades de forma automática y sin que sean visibles en el código, proporcionando así un código más claro.

4.2.3 Capa de datos

En cuanto a la persistencia, contamos con las siguientes tecnologías que han sido de gran utilidad para el desarrollo de la solución:

Hibernate: Para solucionar este problema incorporamos Hibernate. Esta solución permite trabajar directamente con objetos de Java para almacenarlos en la base de datos, independientemente de su implementación. Esto es de gran utilidad para no tener que insertar esta información de forma manual.

JPA: Junto a Hibernate como si de una sola herramienta se tratará, usaremos JPA para construir las operaciones a la base de datos de forma simple. Esto se debe a que nos permite construir repositorios con un lenguaje simple y añadirles consultas independientemente de la tecnología de base de datos que se utilice (La misma consulta serviría para MySQL y MongoDB).

MongoDB: Se empleará esta solución para el almacenamiento de datos, al haberse considerado que una base de datos de características no relacionales se acerca más a la velocidad que buscamos a la hora de tratar con datos, donde premiará más esta rapidez y un gran volumen de datos que su consistencia. Para consultar esta información de forma local se ha empleado MongoDB Compass, una utilidad que permite consultar la información de forma visual en estas bases de datos.

No ha sido necesario almacenar información sensible de los usuarios, ya que los cuestionarios son identificados mediante un ID que les representa, permitiendo el acceso y uso de la aplicación sin registro e independiente de los cokies del navegador.

4.2.4 Otras tecnologías

A continuación, exponemos otras tecnologías que han sido claves para desarrollar el proyecto pero que no han tenido impacto en ninguna capa en concreto de las mencionadas, si no en el desarrollo en general:

Git y Github: Se ha utilizado Git para llevar a cabo el control de versiones de la aplicación. Esta tecnología en conjunto a GitHub permite tener un repositorio en la nube donde los cambios pueden ser gestionados y seguidos.

IntelliJ Community: Este entorno de desarrollo se ha utilizado para desarrollar toda la parte de Java de la aplicación, ya que su rama de atajos y herramientas hace del desarrollo una labor más sencilla y rápida.

Visual Studio Code: Este editor de textos ha sido empleado para desarrollar las interfaces de la aplicación, ya que es un editor ligero y que ofrece facilidades para la programación web.

Worki: Se ha utilizado esta herramienta de gestión de metodologías agiles para llevar a cabo un seguimiento de todas las tareas y fases del desarrollo, así como llevar un registro de los tiempos empleados y de la planificación.

Microsoft Office Word: El famoso editor de textos ha sido de gran relevancia para la redacción de la memoria.

Microsoft Office Excel y Google Forms: Juntos han permitido la construcción de la encuesta para los usuarios y el procesado de la información de esta.

Moqups y GenMyModel: Estas herramientas se han utilizado para la generación de todos los gráficos y bocetos vistos en esta memoria.

Maven: Hemos empleado esta tecnología para la gestión automática de las dependencias de librerías de la aplicación.

Teams: Esta aplicación de comunicación es la que ha sido empleada para las conversaciones entre alumno y tutor de este trabajo.

5. Desarrollo de la solución propuesta

Planteamos el desarrollo del trabajo mediante metodologías ágiles Scrum y Kanban. Esto se debe a que permite ir modificando la especificación según se vaya avanzando en el proyecto. Esto se propone ya que el alumno es nuevo en la tecnología y que es una propuesta novedosa según se ha visto en el estado del arte. Esto podría suponer un cambio en los requisitos que sería más fácilmente gestionado con esta metodología.

Durante los distintos *Sprints* se ha ido iterando la metodología, depurándola cada vez más con los conocimientos adquiridos en la rama de Ingeniería del Software para hacerla lo más acertada posible para la solución del problema propuesto. En cada Sprint se desarrolla una entrega de un producto mínimo viable. Encontraremos los siguientes Sprints:

- **Sprint 1**: En este sprint de crea de forma general las bases de la aplicación a utilizar. Eso consiste en un algoritmo de búsqueda de patrón básico, una página de patrones con la información. Además, se construirá la aplicación de modo que este guiada por datos desde la base de datos.

- **Sprint 2**: En este Sprint se realizan grandes mejoras internas de la aplicación para hacer un código más estructurado y mejorar el acceso a datos. Se introducirá JPA y sus repositorios entre otras cosas para mejorar el código. Posteriormente, se realizarán mejoras de interfaz y se añadirán nuevos campos de información a los patrones. Finalmente, se realizarán mejoras del algoritmo de cuestionario. Esto se realiza con el objetivo de hacer un algoritmo más preciso donde se pueda alcanzar un mejor resultado.

- **Sprint 3**: Este Sprint esta caracterizado por añadirle a la aplicación la funcionalidad relacionada con los aspectos del panel administrativo. En este panel los docentes podrán añadir información sobre patrones o modificar las preguntas del algoritmo de búsqueda de la sugerencia del patrón.

Como herramienta de bocetado se utilizarán las herramientas mencionadas en el apartado de Diseño de la solución. Estas herramientas se usarán para la realización de

todos los bocetos que veremos a continuación. En cuanto al diseño, se realizará una estructura basada por capas como se explica en al apartado mencionado.

Por último, las pruebas de aceptación son el resultado de hablar con el dueño del producto (el tutor de este trabajo) los requisitos que debería de tener la aplicación. Estos han sido traducidos en los requisitos explicados anteriormente y especificados de forma más concreta en los que podemos observar ahora como pruebas de aceptación, las cuales detallan un comportamiento concreto del sistema y no un caso de uso de forma más general. Esto se diferencia en que indicarán detalladamente los pasos a seguir y el resultado.

Todos los tiempos y descripciones de pruebas de aceptación han sido documentados mediante la herramienta Worki. Esta ha ayudado a la gestión de los distintos procesos de la gestión de las unidades de trabajo. En la Figura 4, podemos ver el modelo de trabajo TUNE-UP Process utilizado en Worki:

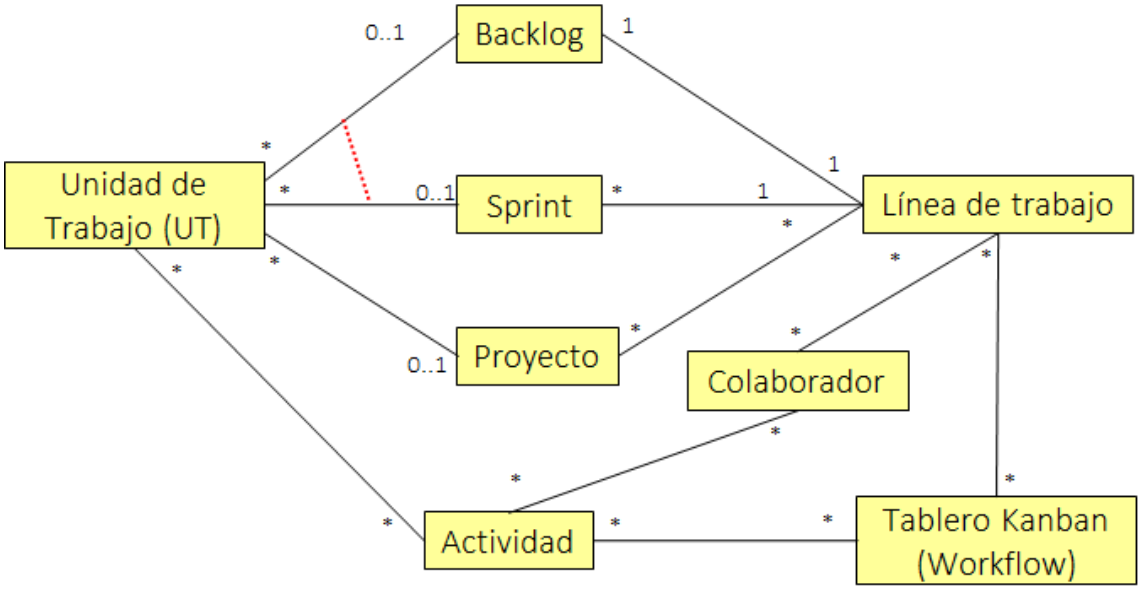

Figura 4 Modelo de trabajo TUNE-UP Process de Worki

5.1 Primer MVP

En esta iteración se propone la construcción de un sistema básico sobre el que basar el resto del trabajo. Se introducirán las funcionalidades básicas:

- Página principal: Se ofrecerá una página desde donde se puedan consultar información general de la aplicación, una lista de patrones y los créditos de la aplicación.

- Ficha del patrón: Se construirá una página donde podamos encontrar información de cada patrón. En ella contaremos con su nombre, una descripción básica y un ejemplo del mundo real.

- Cuestionario: Se realizará un sistema de cuestionario básico desde el cual podemos mediante una serie de preguntas encontrar el patrón adecuado para un problema dado.

Este Sprint será una toma de contacto para las tecnologías en este trabajo por parte del alumno, además de obviamente los conocimientos obtenidos de los cursos consultados.

5.1.1 Página principal

En esta unidad de trabajo hemos planteado la creación de toda la estructura del proyecto interna y la página principal del mismo. En las siguientes figuras vamos a ir mostrando el diseño que se ha propuesto como boceto para los menús.

Como se puede ver en la Figura 5 Boceto de la ventana inicial , hemos propuesto una página principal desde la cual se puede navegar a los distintos menús y es posible encontrar información sobre los contenidos fundamentales de la web.

Bocetos

A continuación, mostramos los distintos bocetos realizados para la interfaz. En la Figura 5 podemos ver el modelo realizado para la página de inicio de la aplicación.

Figura 5 Boceto de la ventana inicial

En la Figura 6 se muestra el menú planteado para los patrones de diseño. Así se presentará la distinta información de cada patrón.

Figura 6 Boceto de la página de Patrones

Por último, en la Figura 7 se podrá observar el menú de créditos, en el cual aparecerá el alumno, el tutor del trabajo de final de grado y la universidad en la cual se ha realizado (La Universidad Politécnica de Valencia).

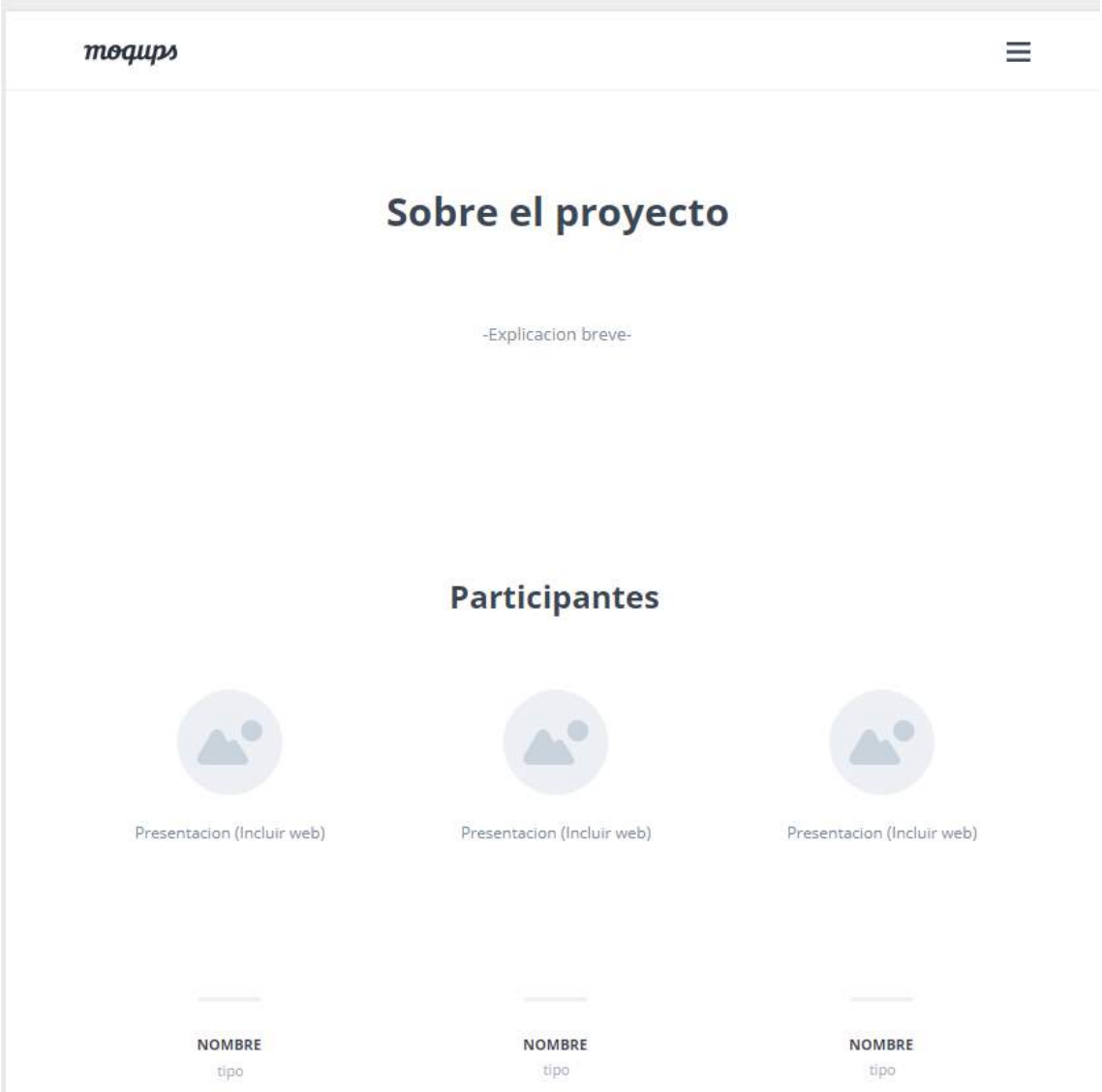

Figura 7 Pagina de créditos

En la Figura 8, Figura 9 y Figura 10 podemos observar la implementación final de los bocetos mostrados en la última versión de la aplicación.

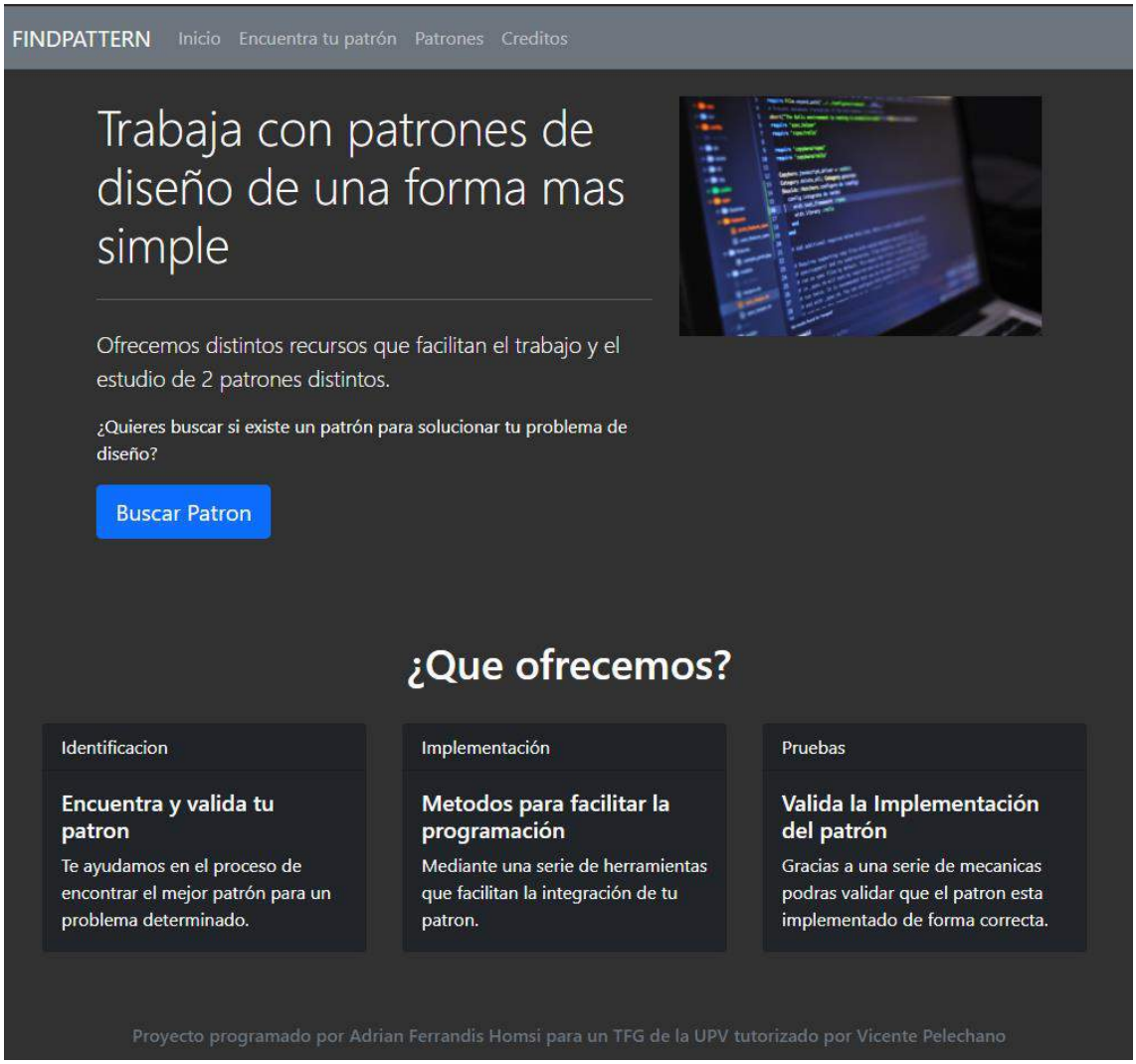

Figura 8 Resultado de la página principal

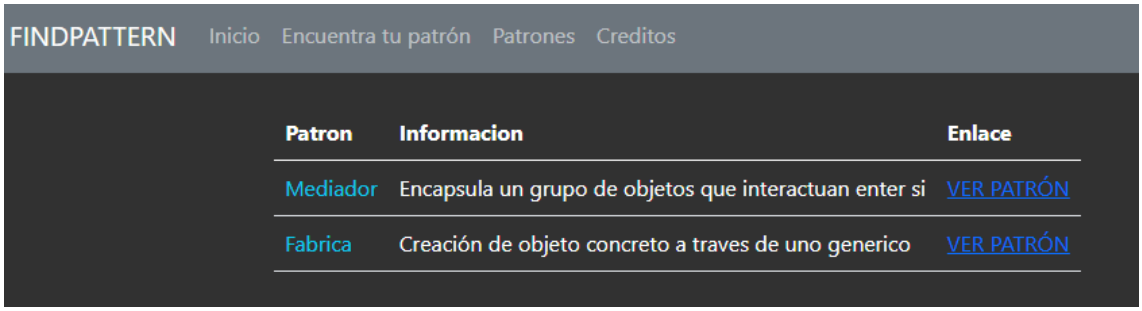

Figura 9 Resultado de la vista de patrones

Figura 10 Resultado de la ventana de créditos

Diseño

Se tendrá que crear tantas interfaces/controladores como ventanas existan. No existe nada de Código así que no se detectan patrones u otras características a seguir.

Se va a seguir una estructura basada en controlador - servicio - persistencia apoyada por un modelo para la aplicación. Es decir, tendremos construiremos la base de una arquitectura basada en capas

Por tanto, estará compuesto de:

- Presentación: Tendremos un controlador de la ventana principal. Este contendrá la lógica de las distintas peticiones de la ventana principal. Ese servirá en un documento HTML los datos proporcionados del servicio gracias a Thymeleaf.

- Lógica: Ofrecerá una serie de llamadas para recibir información de los patrones para mostrarlo en la ventana de patrones de la ventana principal.

- Persistencia: Una serie de consultas para poder obtener información de estos patrones.

Para la ventana principal se contempla únicamente un nombre y una breve descripción del patrón. El patrón concreto se desarrollará en la tarea de ficha del patrón. En esta tarea y las demás de este Sprint no habrá una separación tan grande ente servicio y base de datos a nivel de código como sería deseable. Esto se corrige en la tarea del Sprint 2 con los nuevos conocimientos adquiridos por el alumno.

Pruebas de aceptación

Las pruebas de aceptación se centran en la navegación por las distintas pestañas de la aplicación y la visualización de datos del servicio.

- Prueba aceptación: Se podrá acceder a la web de cada participante

CONDICIONES	**Pestaña de créditos**
PASOS	Accedemos a la web de un participante
RESULTADO	Nos redirecciona a la web del participante
OBSERVACIONES	Preferiblemente será interesante que esto se haga en una nueva pestaña

- Prueba aceptación: Se puede acceder al apartado de créditos

CONDICIONES	**Página principal**
PASOS	Entrar en el apartado de créditos
RESULTADO	Se mostrarán los distintos participantes
OBSERVACIONES	

- Prueba aceptación: Se podrá acceder al apartado de patrones

CONDICIONES	**Página Principal**
PASOS	Acceder a la pestaña de patrones
RESULTADO	Veremos una lista con los distintos patrones

OBSERVACIONES	Que no cargue la lista de patrones correctamente contara como un error

Tiempos

Se puede observar en la Figura 11 como se ha distribuido el a tiempo a la hora de ejecutar los distintos aspectos de la tarea. No se ha encontrado ninguna dificultad, salvo la poca experiencia en la tecnología.

Registrar			1m
Esperar Prioridad			
Especificar Requisitos			49m
Diseñar y estimar			8m
Esperar Sprint			
Programar	15h	15	14h 12m
Aplicar Pruebas de Aceptación			17m

Figura 11 Desglose de tiempos de la ventana principal

5.1.2 Explicación/Ficha del patrón

En esta tarea construiremos para cada patrón mostrado en la sección de patrones de la página web una información. Esta contará de:

- Nombre: El nombre del patrón, el cual funcionará también como su identificador.
- Descripción: Una breve descripción del propósito del patrón
- Explicación: Un ejemplo de uso real del patrón de diseño

- Botón de validación: Existente, aunque deshabilitado por defecto, está pensado para en un futuro presentar una serie de preguntas que validen que el patrón es aplicable para un caso en concreto.

Todos los campos no utilizados por un patrón se ocultarán. También se expandirían en función de la información disponible y el ancho de pantalla.

Si un campo mencionado de los anteriores no existe, no se mostrará por defecto, dejando el espacio disponible para ser ocupado por otra información del patrón.

Bocetos

En la Figura 12 observamos como se propone la distribución original de la información de cada patrón de diseño.

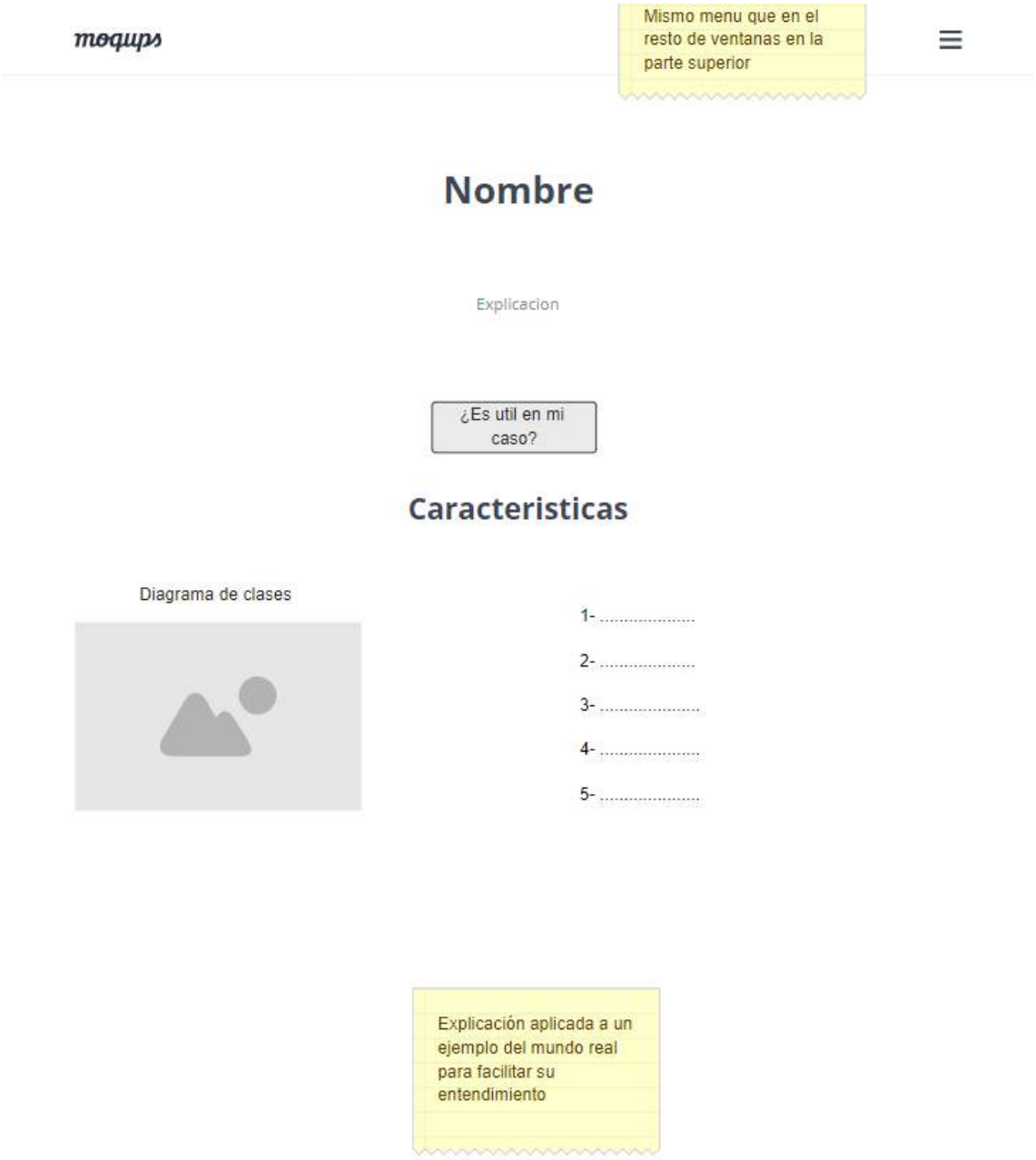

Figura 12 Distribución de la explicación del patrón

Para finalizar este apartado, *mostramos en la Figura 13 y Figura 14 la una* versión final de este contenido en la aplicación. Cabe mencionar que este resultado se puede ver afectado por futuros cambios.

Figura 13 Resultado ficha patrón (1)

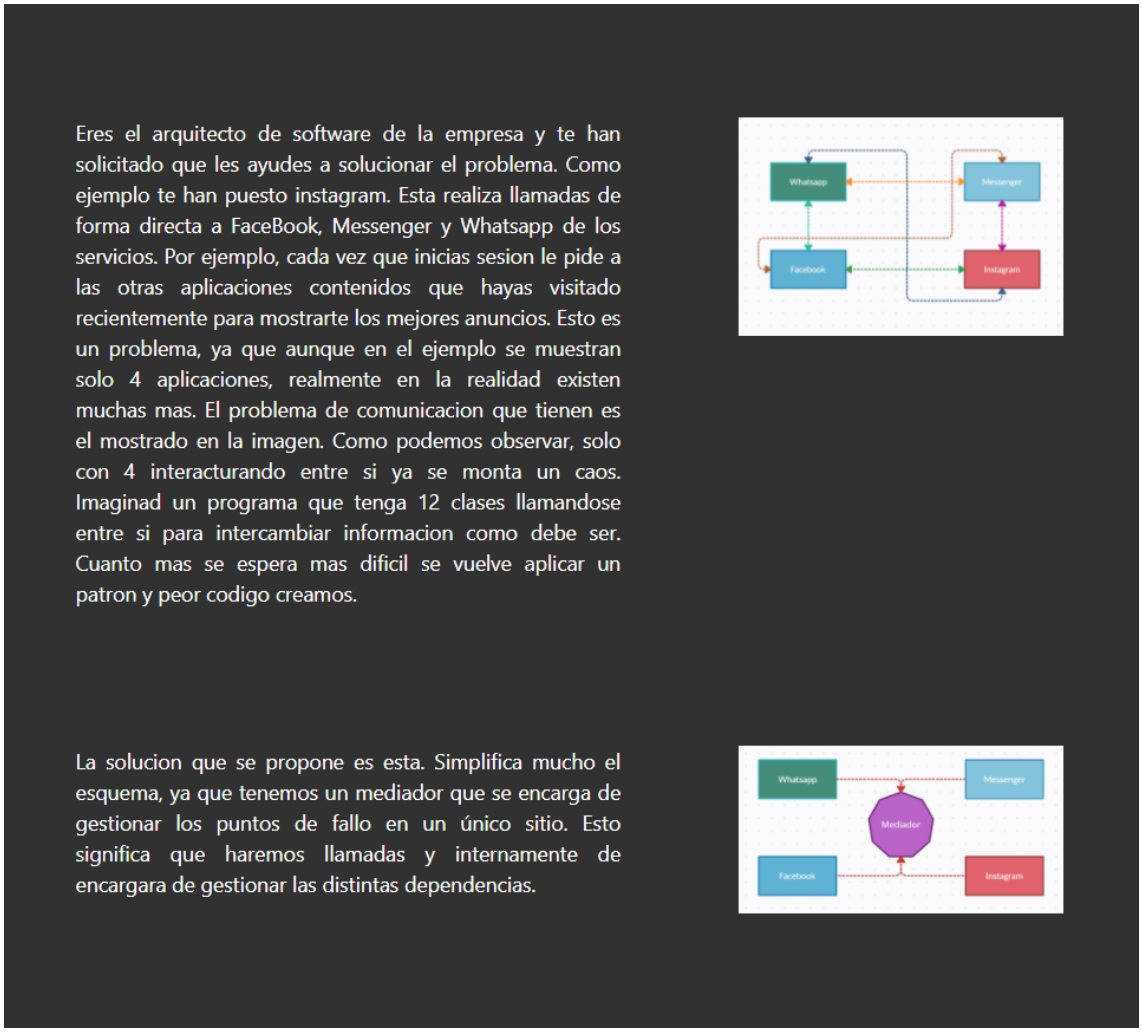

Figura 14 Resultado ficha patrón (2)

Diseño

Se utilizará dentro de la ventana principal, al acceder a cada patrón a donde nos redirige para construir este aspecto. Es importante hacerlo lo más modificable posible de cara a futuros cambios que se quieran introducir en la aplicación.

Se prevé que se tendrá que construir una clase controladora que se encargue de gestionar la carga de información. Esto se hará gracias a una clase de Modelo que almacene toda la información necesaria. además, se sugiere crear una estructura flexible también en este objeto, sugiriendo lo más posible que siga un formato de lista o tupla (texto / imagen) para poder introducir tantas imágenes como sea necesario para apoyar la explicación. Por tanto, tendremos los siguientes apartados:

- Presentación: Un controlador nuevo junto a un contenido HTML que dado un patrón en la URL será capaz de suministrar los datos de este o un error en caso de no encontrarlo.

- Lógica: Partimos de la lógica creada para los patrones, pero ahora le añadimos el acceso a un patrón concreto mediante su información en lugar de una lista de patrones.

- Persistencia: Se incluirán los nuevos campos del patrón para ser almacenados y cargados desde la base de datos.

Pruebas de aceptación

En esta unidad de trabajo tenemos la prueba de aceptación centrada en la visualización de la nueva información.

- Prueba aceptación: Se podrá acceder a la web de cada participante

CONDICIONES	**Ventana Principal**
PASOS	Accedemos a la categoría de Patrones y luego en un patrón concreto
RESULTADO	La información cargara correctamente
OBSERVACIONES	

Tiempos

En esta tarea sobro notablemente más tiempo que el estimado. Esto se justifica por el desconocimiento de la tecnología del alumno ya que no se podía saber con exactitud el esfuerzo a realizar. Podemos observar estos resultados en la Figura 15.

Actividad	Primera estimación	Estimación actual (en horas)	Registrado
Registrar			
Esperar Prioridad			
Especificar Requisitos			35m
Diseñar y estimar			7m
Esperar Sprint			
Programar	10h	10	6h 4m
Aplicar Pruebas de Aceptación			5m

Figura 15 Tiempos de Explicación patrón

5.1.3 Cuestionario para saber qué patrón elegir

El cuestionario nos hará diversas preguntas con el objetivo de encontrar el patrón ideal para la situación que buscamos. Las preguntas serán principalmente utilizando la escala LIKERT con el objetivo de conseguir la mayor precisión en la información obtenida.

Al principio será un algoritmo sencillo. Cada pregunta va asociada a una valoración de un patrón concreto y al final el de mayor puntuación se considera el patrón recomendado para resolver ese problema. Este algoritmo se ira iterando en futuras iteraciones.

Esto se hará, que, si tenemos tres preguntas, la escala LIKERT se evaluara con un número del 1-5, es decir:

Respuesta	**Muy en desacuerdo**	**En desacuerdo**	**Ni de acuerdo ni en desacuerdo**	**De acuerdo**	**Muy de acuerdo**
Valoración	1	2	3	4	5

Esto hará, que hagamos la media de los valores recibidos como el interés de un patrón. Después comparamos las medias de los distintos patrones y los ofrecemos en este

orden de medias como posible solución. Por cómo está estructurado el patrón se realizan todas las preguntas, aunque se puedan descartar algunos patrones por las respuestas obtenidas. Esto se corrige en la futura tarea de mejoras del cuestionario con un nuevo algoritmo.

Bocetos

Se puede observar en la Figura 16 la distribución planteada para el cuestionario. Esta costara de una o varias preguntas y un botón de enviar. Esto nos ira haciendo navegar entre las distintas secciones del cuestionario.

Figura 16 Preguntas del cuestionario

Además, en la Figura 17 podemos ver el resultado final de este diseño al final de los tres Sprints.

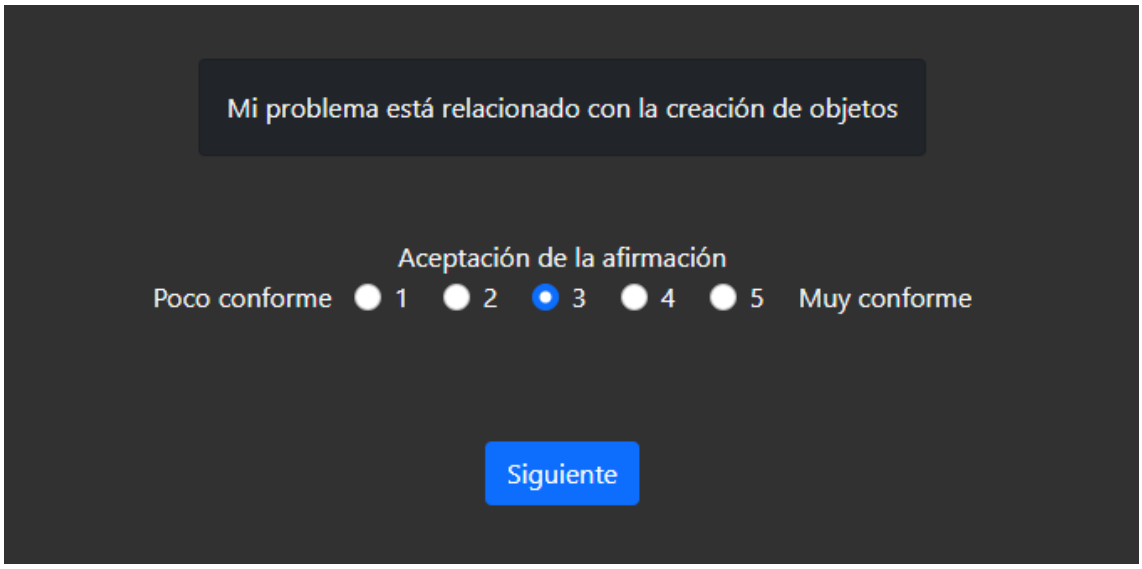

Figura 17 Resultado del cuestionario

Diseño

- Presentación: Contaremos con nuevo un controlador encargado de gestionar la lógica de las peticiones que recibamos sobre el cuestionario. Este actualizará la información al usuario y procesará sus respuestas para saber a qué paso tiene que avanzar el cuestionario. Este controlador, contendrá un objeto el cual tiene dentro métodos para poder actualizar las preguntas. Este objeto serán los datos del modelo estructurados de forma que se pueda acceder más rápido a su contenido. Tendrá toda la lógica para dada una llamada procesada por el controlador poder obtener preguntas o calcular puntuaciones de cada patrón. En un futuro cambio de algoritmo, toda esta lógica será trasladada al servicio para dejar únicamente un DTO como un objeto con datos. Una vez finalizado el cuestionario, tendremos un controlador encargado de presentar los resultados. También contamos con un DTO que tiene el fin especial de una vez finalizado el proceso del cuestionario almacenar los resultados para presentarlos.

- Lógica: Tendremos un nuevo servicio de cuestionario encargado de recibir peticiones sobre la obtención de un cuestionario dada una ID que lo identifica. En caso de existir, nos devuelve el estado de sus preguntas. En caso de no poder encontrarlo, creara uno de nuevo y lo iniciara a los valores por defecto.

- Persistencia: Se incluye todo el almacenamiento de la información de los cuestionarios en base de datos. En estas primeras versiones todavía no se ha incorporado JPA, por eso la información varia más de lo que lo hará en JPA del DTO al modelo. Al mismo tiempo, obtenemos ciertos datos acoplados entre JPA

y DTO que no tendrían por qué ser así, al no contar con objetos distintos. Contaremos en persistencia también con un objeto que almacena las preguntas por defecto del cuestionario. Esto posteriormente en futuras iteraciones será movido a base de datos para hacer la aplicación orientada a datos.

A continuación, mostramos en la Figura 18 el diseño planteado para la resolución de este requisito.

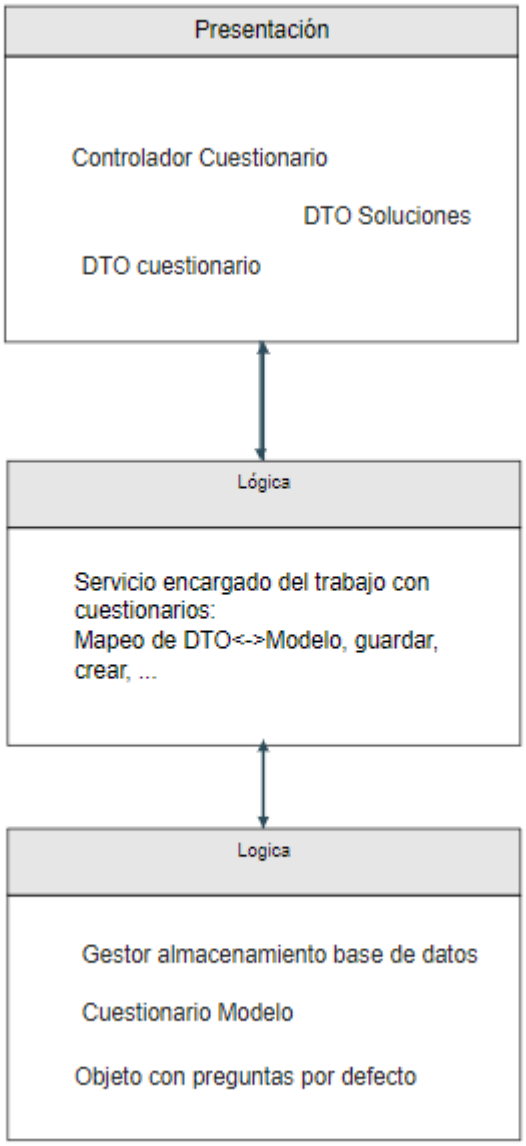

Figura 18 Estructura del cuestionario (primera versión)

Es interesante construir el de mostrar resultados de la forma más general posible, para poder reutilizarlo en otro tipo de búsquedas de patrones. Por ejemplo, puede ser utilizado en la detección automática de código. Esto simplificaría el código futuro.

Cabe destacar también que se presentará los patrones de diseño separados en tres categorías: Creacionales, Estructurales y Comportamiento. Al seleccionar una categoría, empezaremos a recibir preguntas de los distintos patrones de esta. Esto en el futuro cambiará ya que obtendremos preguntas para identificar la categoría del patrón.

Se construyo también una serie de pruebas unitarias para convertidores de modelo a DTO de los cuestionarios y del algoritmo de cuestionario.

A continuación, mostramos una versión de preguntas de patrones creacionales que se incluyó en esta primera iteración para probar el algoritmo:

Factory	No conozco las dependencias y tipos exactos de los objetos con los que va a trabajar mi código Me gustaría dar a los programadores futuros que usen estas clases una forma de extender los componentes internos del mismo Quiero ahorrar recursos del sistema mediante la reutilización de objetos existentes
Builder	Quiero evitar constructores sobresaturados (muchos parámetros o constructores distintos) o con parámetros a nulo Quiero crear distintas representaciones de distintos objetos (ejemplo: Camión rojo y Camión azul)
Abstract Factory	Mi código tiene que funcionar con varias familias de productos relacionados, pero no quiero que dependa de clases concretas de estos productos (Normalmente porque no los conoces de antemano o quieres facilitar la extensibilidad)
Singleton	Busco una forma de llevar un control estricto de las variables globales. Quiero tener una única instancia de una clase en todo el programa, la cual esté disponible para ser utilizada.
Prototype	Me gustaría reducir las subclases que solo se diferencian en la forma en la que se inicializan sus objetos. Quiero clonar objetos sin acoplarlos a otra clase

Pruebas de aceptación

- Prueba aceptación: Se puede encontrar un patrón

CONDICIONES	**Ventana Principal**
PASOS	Accedemos a la pestaña de búsqueda de patrones Completamos el cuestionario para resolver un problema concreto.
RESULTADO	Nos proporciona un patrón como posible solución.
OBSERVACIONES	En caso de no ser posible encontrar una solución debe de informarse al usuario de ello.

Tiempos

A continuación, en la Figura 19, observamos los tiempos invertidos en la construcción de esta funcionalidad. Esta tarea ha sido la más extensa del Sprint debido a su complejidad.

Actividad	Primera estimación	Estimación actual (en horas)	Registrado
Registrar			
Esperar Prioridad			
Especificar Requisitos			41m
Diseñar y estimar			10m
Esperar Sprint			
Programar	20h	20	21h 5m
Aplicar Pruebas de Aceptación			4m

Figura 19 Estimaciones y registros de tiempo de la creación del cuestionario

5.2 Segundo MVP

En esta iteración, se propone a partir del resultado del primer MVP realizar una serie de cambios. Estos cambios parten de la reunión entre alumno (que hace de rol de desarrollador en la metodología ágil) y del tutor de este trabajo (que tiene como rol el del dueño del producto).

Como conclusión de dicha reunión, se plantean remodelaciones en partes fundamentales de la aplicación y nuevos cambios. Gracias a los conocimientos adquiridos por el alumno de la tecnología estos cambios son posibles. A continuación, mostramos una lista de los cambios y mejoras realizadas en la aplicación durante este Sprint:

- Preparación del sistema: Se realizan una serie de cambios en el sistema. Esto se realiza para poder incluir nuevos cambios y mejorar la estructura general, ya que con los nuevos conocimientos del alumno podía ser replanteada a una arquitectura mejor.

- Nueva información de los patrones: Se incluirán nuevos cambios para proporcionar información más variada a cada patrón de diseño si se desea. Esto se hace con el fin de poder ampliar conocimientos sobre el mismo

- Mejora de la interfaz: Se realizarán mejoras de la interfaz ya que el contenido después de estos cambios ya está en su versión final, así que se pueden realizar estas mejoras de cara a la presentación.

- Cambio del algoritmo del cuestionario: Se realizarán cambios en el algoritmo para incluir un algoritmo de poda. Esto nos permite dar una solución más rápida a un problema concreto, en lugar de presentar un volumen grande de preguntas cuando podíamos conocer ya la solución en mitad del cuestionario. Para estos cambios será necesario rediseñar completamente el algoritmo del ayudante de búsqueda de patrón.

5.2.1 Preparación del sistema para soportar los nuevos cambios

Se plantea cambiar la estructura interna de la aplicación. Esto viene justificado por la mejora de conocimientos adquiridos por parte del alumno en la tecnología. Esto es debido a que el código actual esta implementado de forma poco mantenible de cara al futuro. En un TFG de estas características es incluso más grave esta deuda técnica. Por ello, se intenta mejorar el código interno mediante un refactoring de la aplicación para adaptarlo a los patrones y prácticas más frecuentemente usadas en Spring Framework.

Además, también es necesario realizar cambios en algunas funciones de la aplicación para facilitar la integración de nuevo contenido en la aplicación, como el algoritmo de poda que se desarrolla en este Spring. Por otro lado, con estos cambios que se realizaran introduciremos más separación entre las capas y soluciones más simples a algunos problemas. Por ejemplo, eso será realizado incorporando JPA para el tratamiento con la base de datos.

Bocetos

Ya que eran cambios internos de funcionamiento no va a haber ningún cambio de funcionalidad para realizar una maqueta.

Diseño

En cuanto al diseño, se realizarán varios cambios internos relevantes:

- Presentación: Se han modificado o creado DTOs nuevos para darles más independencia de la capa de base de datos. Se ha modificado la capa de presentación para adaptarse a los nuevos DTOs creados, ya que estos están más desacoplados del modelo y han variado algunos campos.

- Lógica: Se incluirá una serie de convertidores entre DTO y modelo. Antes esto no era necesario porque se construirá el objeto DTO a partir de los datos de la base de datos y se utilizaba en todo el proyecto. Esto se hará mediante la construcción de unos convertidores que asignaran cada campo al adecuado.

- Persistencia: Se modificará completamente para incluir JPA como nuevo estándar de acceso a base de datos. Antes se realizaba mediante consultas manuales usando Hibernate, pero gracias a los conocimientos nuevos adquiridos por el alumno esto puede ser modificado. De esta forma, conseguiremos un código más legible y mantenible. Además, esto permitía crear más problemas a la hora de comunicar las capas porque requería un mayor número de transformaciones. Todas las clases de consultas serán cambiadas por repositorios JPA que se llamarán desde los servicios. Además, añadiremos la configuración de JPA para conectarse con MongoDB y construiremos todos los objetos de modelo apropiados.

También se han realizado cambios de nombre de las clases y cambios generales estructurales de la aplicación como restructuración de paquetes, incluir las nuevas librerías, cambiar como se comunican las capas u otros cambios generales.

A lo largo de este MVP y del siguiente, gracias a los conocimientos adquiridos por el alumno, se irán modificando poco a poco los convertidores manuales por unos automáticos utilizados habitualmente en Spring.

Se ha creado una rama en git separada para gestionar estos cambios y una serie de pruebas que validen que se han realizado los cambios más críticos correctamente.

Pruebas de aceptación

Como prueba de aceptación se utilizarán todas las pruebas diseñadas en el Sprint 1. Esto se debe a que no cambia la lógica si no como esta implementada. Estas pruebas deben de ser superadas junto a las pruebas unitarias para garantizar que el sistema es equivalente en funcionamiento al original.

Tiempos

A continuación, en la Figura 20, podemos observar el tiempo invertido en los cambios. Al tener tantos cambios estructurales y aplicar conocimientos nuevos, esta tarea ha sido una de las más largas del MVP. Gracias a ella se ha conseguido dotar a la aplicación de la estructura que se buscaba.

Actividad	Primera estimación	Estimación actual (en horas)	Registrado	Debe estimarse
Registrar			<1m	☐
Esperar Prioridad				☐
Especificar Requisitos			40m	☐
Diseñar y estimar				☐
Esperar Sprint			1m	☐
Programar	25h	25	20h 57m	☑
Aplicar Pruebas de Aceptación			42m	☐

Figura 20 Tiempos de los cambios de lógica de la aplicación

5.2.2 Añadir enlaces de documentación y sinergia en vista de patrones

Se plantea incluir más información de un patrón. Se incluirán dos listas con el siguiente contenido

- Enlaces a páginas web con documentación donde se pueda aprender más sobre el patrón.
- Ideas de sinergia o compatibilidad entre patrones para fomentar el uso de varios en conjunto.

Esta tarea responde a la necesidad de contar con más información sobre un patrón y sus posibles usos. Además, también le da más contenido a una característica que los usuarios consideraron importante.

Bocetos

En la Figura 21 se observa una maqueta de la interfaz del patrón donde se detallan los nuevos cambios y estructura de esta. En ella podemos apreciar la distribución y presentación del contenido.

Figura 21 Maqueta de documentación y sinergias del patrón

El resultado final de este boceto es el mostrado en la Figura 13.

Diseño

Se han tenido que realizar los siguientes cambios en la aplicación web para poder añadir los campos nuevos:

- Presentación: Modificaremos el DTO de Patrón que contiene información sobre sus campos para añadir esta nueva información. Añadiremos dos listas en el para incluirla. También habrá que adaptar la vista para añadir estos dos campos nuevos.

- Lógica: Modificaremos los convertidores para poder añadir los nuevos campos en Modelo y DTO a la hora de convertir.

- Persistencia: Cambiaremos el Modelo para que admita guardar y cargar información de estos nuevos campos.

Pruebas de aceptación

- Prueba aceptación: Será posible ver los nuevos campos para un patrón.

CONDICIONES	**Vista de patrones**
PASOS	Acceder a un patrón con información
RESULTADO	Se podrá observar los nuevos campos del patrón.
OBSERVACIONES	

- Prueba aceptación: Los campos no se ven en caso de tener contenido

CONDICIONES	**Vista de patrones**
PASOS	Acceder a un patrón sin información de sinergias y/o documentación.
RESULTADO	En caso de no contener información esta sección no se mostrará.
OBSERVACIONES	

Tiempos

Como hemos podido ver, al ser una tarea sencilla nos ha llevado poco tiempo. Esto es observable en la Figura 22.

Actividad	Primera estimación	Estimación actual (en horas)	Registrado	Debe estimarse
Registrar				☐
Esperar Prioridad				☐
Especificar Requisitos			20m	☐
Diseñar y estimar				☐
Esperar Sprint				☐
Programar	4h	4	3h 46m	☑
Aplicar Pruebas de Aceptación			31m	☐

Figura 22 Tiempos de la tarea de documentación y sinergias

5.2.3 Mejora de colores y centrado de la interfaz

El propósito de esta tarea es cambiar los colores y centrar los elementos de la interfaz. Estos elementos se quieren centrar de forma escalable, haciendo que se adapten lo máximo posible dentro del diseño actual. Esto también se realiza a raíz de las conclusiones del anterior Sprint sobre cambios de la aplicación.

Bocetos

No se ha maquetado nada ya que no se propone ningún cambio de funcionalidad o ventana nueva. Los colores que utilizar se basarán principalmente en los proporcionados por Bootstrap por defecto, para facilitar la lectura de los distintos contenidos.

Diseño

Únicamente realizaremos cambios a nivel de controlador. Se modificarán los archivos HTML y se incluirán archivos CSS o referencias a estilos de Bootstrap desde el HTML.

Pruebas de aceptación

- Prueba aceptación: No hay ningún elemento principal de la aplicación mal centrado o con colores que dificulten la lectura

CONDICIONES	**Entrar a cualquier ventana que nos ofrece la ventana principal**
PASOS	
RESULTADO	Todos los elementos se verán bien centrados y con colores que faciliten la lectura
OBSERVACIONES	

Tiempos

Finalmente ahorramos algo de tiempo gracias a la tecnología de Bootstrap. Como podemos ver en la Figura 23, al igual que la tarea anterior, resulto una tarea con una duración no muy extensa:

Actividad	Primera estimación	Estimación actual (en horas)	Registrado	Debe estimarse
Registrar				☐
Esperar Prioridad				☐
Especificar Requisitos			30m	☐
Diseñar y estimar				☐
Esperar Sprint				☐
Programar	4h	4	3h 20m	☑
Aplicar Pruebas de Aceptación			17m	☐

Figura 23 Tiempos Mejora Interfaz

5.2.4 Cambio de algoritmo de búsqueda (poda)

Se realiza un cambio en el ayudante de recomendación de cuestionario creado previamente. Esto se realiza como último punto de este MVP. Se reescribe la mayor parte del código con los nuevos conocimientos tecnológicos y con las mejoras obtenidas de la revisión.

Este cambio se plantea como un cambio en las preguntas planteadas y la forma de tratar las mismas. Ahora la aplicación será capaz de hacer preguntas que descarten patrones o marquen un patrón como posible solución a un problema. Esto hará que el proceso sea mucho más rápido.

Las preguntas tendrán un orden para ser mostradas. Esto es interesante porque podemos mostrar primero las preguntas que correspondan a patrones más frecuentes o que eliminen varios patrones de la sugerencia, para así obtener una recomendación más pronto y eficazmente.

Preguntas del cuestionario

Como conclusión de esta tarea, se han formulado el siguiente conjunto de preguntas para el algoritmo de búsqueda del patrón. Estas preguntas irán apareciendo al usuario y guiándole hasta alcanzar la respuesta. Con el fin de hacer una mejor búsqueda todas estas preguntas son distintas de las planteadas en el *Sprint* anterior.

En la Figura 24 se puede observar un gráfico del flujo del algoritmo. Este será explicado posteriormente en más detalle:

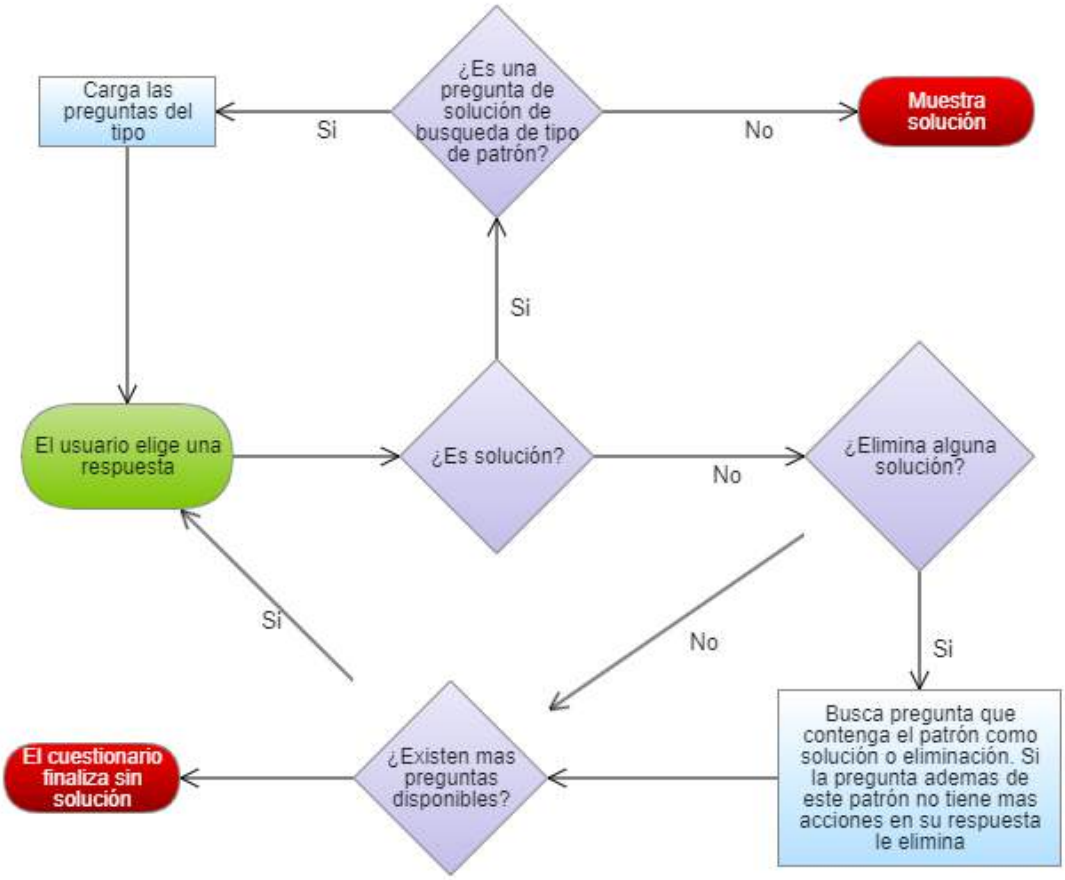

Figura 24 Flujo del algoritmo de sugerencia de patrón

A continuación, explicamos las posibles respuestas a las preguntas y como se responde ante ellas:

- El usuario elige una respuesta que determina el tipo del patrón: Las primeras preguntas mostradas tendrán que ver con la determinación del tipo del patrón. En caso de elegir una respuesta que determine el tipo (creacional, estructural, o comportamiento) se actualizarán las preguntas disponibles para este tipo. Será el primer paso que debe realizar el usuario. Es necesario discriminar el tipo del patrón como primer paso para poder pasar a buscar la solución.

- El usuario elige una respuesta que guía a una solución: El usuario será redireccionado a la página del patrón solución.

- El usuario elige una respuesta que elimina un patrón como solución: El cuestionario busca que preguntas tienen ese patrón en sus respuestas. En caso de contener ese patrón, lo eliminan de sus respuestas (ya sea de solución o de descarte). Si después de esto, la pregunta patrón puede seguir haciendo otras acciones (como eliminar otro patrón no eliminado) la pregunta se mantiene. En caso de que no sea así, esta pregunta es eliminada para acelerar el algoritmo.

- El usuario se queda sin posibles soluciones: Se mostrará un mensaje notificándole al usuario de esta situación y se le anima a volverlo a intentar. Esto se hace si el algoritmo no ha podido determinar una solución para su problema

Además, cabe destacar que en los casos que no encuentra solución, una vez hecho el posible descarte de preguntas, el algoritmo seguirá mostrando preguntas en el orden determinado.

Preguntas para descubrir el tipo de patrón

Leyenda:

Solución

Descartado

Afirmación	Nada satisfecho	Poco satisfecho	Neutral	Muy satisfecho	Totalmente satisfecho
Mi problema está relacionado algoritmos y asignación de responsabilidades	Comportamiento	Comportamiento		Comportamiento	Comportamiento
Mi problema está en cómo se relacionan las distintas clases entre si	Estructural	Estructural		Estructural	Estructural

Mi problema está relacionado con la creación de objetos	Creacional	Creacional		Creacional	Creacional

Preguntas patrones creacionales

Leyenda:

Solución

Descartado

Afirmación	Nada satisfecho	Poco satisfecho	Neutral	Muy satisfecho	Totalmente satisfecho
Tengo un problema con la creación de objetos NUEVOS.	Builder, Factory y Abstract Factory	Builder, Factory y Abstract Factory		Singleton	Singleton Protype
Existe un objeto que es complejo de construir (constructores grandes, mucho código disperso para su creación, …). Se quiere construir paso a paso como si se tratase de ir construyendo un edificio donde se puede personalizar sus componentes.	Builder	Builder		Builder	Builder
Necesito centralizar u ocultar los detalles de la creación de distintos objetos	Factory y Abstract Factory	Factory y Abstract Factory		Builder Singleton Prototype	Builder Singleton Prototype
Es necesario centralizar u ocultar los detalles de creación de un grupo de objetos, pudiendo existir	Abstract Factory Factory	Abstract Factory		Abstract Factory	Abstract Factory

distintos tipos de un mismo objeto. Por ejemplo, si tenemos una fábrica de coches, tener piezas para coches deportivos y piezas para todoterrenos. Entonces ambos tendrán una rueda, un volante, … pero con sus características					
Necesito centralizar u ocultar la creación de un tipo de objeto concreto	Factory y Abstract Factory	Factory		Factory	Factory
Busco una forma de clonar un objeto de forma completa (incluso sus atributos privados)	Prototype	Prototype		Prototype	Prototype
Necesito limitar las instancias de una clase a una única. (Por ejemplo, que solo exista una base de datos)	Singleton	Singleton		Singleton	Singleton

Preguntas patrones estructurales

Leyenda:

Solución

Descartado

Afirmación	Nada satisfecho	Poco satisfecho	Neutral	Muy satisfecho	Totalmente satisfecho
Existen dos objetos que no pueden cooperar entre sí. Por	Adapter	Adapten		Adapter	Adapter

ejemplo, tenemos una clase que trabaja con JSON y otra con XML.					
Tengo un problema complejo que me gustaría dividir en distintas partes más abordables. (Un coche que pueda tener varios colores. En lugar de crear un objeto cada color, creamos un objeto color y lo vinculamos al coche)	Bridge	Bridge		Bridge	Bridge
Me gustaría acceder a un conjunto de objetos mediante una interfaz que facilite su uso (Una biblioteca de procesamiento de ficheros a la que se le quiere dar un acceso sencillo mediante una interfaz en lugar de tener que llamar a varias clases concretas del sistema)	Facade	Facade		Facade	Facade
Necesito tratar un conjunto de objetos relacionados entre sí. Estos podrían ser establecidos como una jerárquica (Por ejemplo, una caja que dentro tenga otra caja que a su vez tenga un artículo)	Composite	Composite		Composite	Composite
Existe un objeto al que me gustaría añadirle funcionalidad (Tenemos algo ya existente y	Decorator	Decorator		Decorator	Decorator

queremos darle una funcionalidad extra en algún caso concreto)					
Tengo problemas de muchos objetos replicados que podrían compartir información (Un explorador de archivos donde existen muchos ficheros con información compartida como su ubicación o el icono que tienen).	Flyweight	Flyweight		Flyweight	Flyweight
Tengo un objeto sobre el cual no se controla el acceso y esto es un problema. Por ejemplo, me gustaría controlar el acceso sobre la base de datos.	Proxy	Proxy		Proxy	Proxy

Bocetos

Al tratarse de mejoras algorítmicas y respetar la misma interfaz, no se han realizado maquetas nuevas.

Diseño

Estos cambios se realizarán con el objetivo de dar a la aplicación un ayudante que pueda resolver de una forma más precisa los problemas del usuario a la hora de buscar un patrón para un problema, ofreciéndole una recomendación de forma más rápida y precisa. También se creará un DTO y un Modelo de la Pregunta, la cual sería encargada de contener información de las preguntas mientras el cuestionario contiene información de este y de su estado.

- Presentación: Se han realizado cambios en los DTO del cuestionario. Además, añadiremos el DTO de pregunta mencionado anteriormente.

- Lógica: Se ha cambiado el convertidor entre el DTO del cuestionario y se han incorporado pruebas para verificarlo. Se ha creado un nuevo servicio de cuestionario encargado de gestionar la lógica de este. Este, dada una respuesta para una pregunta será capaz de hacer una serie de acciones, las cuales explicaremos posteriormente. También se añadirá un servicio de Preguntas para cargar las preguntas.

- Persistencia: Se cambiará el modelo del cuestionario para incluir los nuevos datos de este.

Es decir, contaremos con un servicio y objeto cuestionario, encargado de tener la lógica del algoritmo y los datos de este. Por otro lado, contaremos con una clase y servicio de preguntas, encargadas de gestionar el contenido de las preguntas y sus respuestas independientemente del algoritmo ejecutado. Con esto conseguimos una mayor independencia que antes no existía de cara a cambiar el algoritmo en el futuro.

Pruebas de aceptación

- Prueba aceptación: Las preguntas se ordenan según la estrategia implementada

CONDICIONES	**Vista de búsqueda de patrón**
PASOS	Contestar preguntas del cuestionario
RESULTADO	Se podrá observar las preguntas reordenadas según la estrategia seleccionada
OBSERVACIONES	

- Prueba aceptación: Se llega a una solución de forma correcta

CONDICIONES	**Página de búsqueda de patrón**
PASOS	Se empieza la búsqueda de un patrón concreto
RESULTADO	Se alcanza un resultado esperado
OBSERVACIONES	En caso de elegir un camino sin solución se notificará que no ha sido posible encontrar una.

Tiempos

Como se puede observar en la Figura 25, se ha invertido un gran tiempo en la especificación de requisitos para poder obtener las preguntas necesarias para la búsqueda del patrón.

Actividad	Primera estimación	Estimación actual (en horas)	Registrado
Registrar			
Esperar Prioridad			
Especificar Requisitos			16h 52m
Diseñar y estimar			
Esperar Sprint			
Programar	20h	20	19h 9m
Aplicar Pruebas de Aceptación			1h 12m

Figura 25 Tiempos cambio a algoritmo de poda

5.3 Tercer MVP

Como ultimo Sprint se ha procedido a construir una versión del producto lo más cercana a su salida a producción. Es por esto por lo que se dedica principalmente el trabajo realizado a la construcción de un sistema de administración.

Este panel administrativo tendrá varias partes que deben ser construidas. El trabajo de este se dividirá de la siguiente forma:

- Inicio: Se construirá una pestaña de inicio donde se proporcione información del panel de control y datos de su rendimiento. Además, mientras se construye

esta funcionalidad se establecerán los permisos y las pantallas de inicio y cierre de sesión.

- Configuración: Se presentará una pestaña donde se informa de como configurar el servidor desde el archivo de despliegue de este.

- Patrones: Una pestaña donde se permite modificar los datos de los patrones de la aplicación web. Esto nos permitirá añadir nuevos patrones o modificar y eliminar alguno de los existentes.

- Cuestionario: Esta idea es similar a la presentada de los patrones, pero para las preguntas del cuestionario, permitiendo así ir iterando en su funcionamiento y añadir poco a poco mayor precisión y calidad en las preguntas.

5.3.1 Panel Administrativo - Inicio

Será necesario incorporar un sistema de inicio de sesión administrativo y permisos. Durante la ejecución de esa tarea se tendrá que configurar las credenciales para el inicio de sesión y las páginas o acciones que requieren autentificación por parte del servidor. Las claves se podrán configurar desde las propiedades del servidor. En caso de intentar realizar una acción no permitida nos llevara al panel de inicio de sesión.

También se mostrará una ventana principal donde presentaremos información del panel y datos sobre el estado del servidor.

Bocetos

Podemos observar en Figura 26 el inicio de sesión del panel administrativo. En este se nos pedirá que introduzcamos el usuario y la contraseña.

Figura 26 Inicio de sesión administrativo

También podemos observar en Figura 27 la vista que observaremos desde el menú principal de administración. En esta también se nos brindará información sobre las estadísticas del servidor.

Figura 27 Menú principal administración

En la Figura 28 y Figura 29 la vemos el resultado final de la interfaz de este diseño.

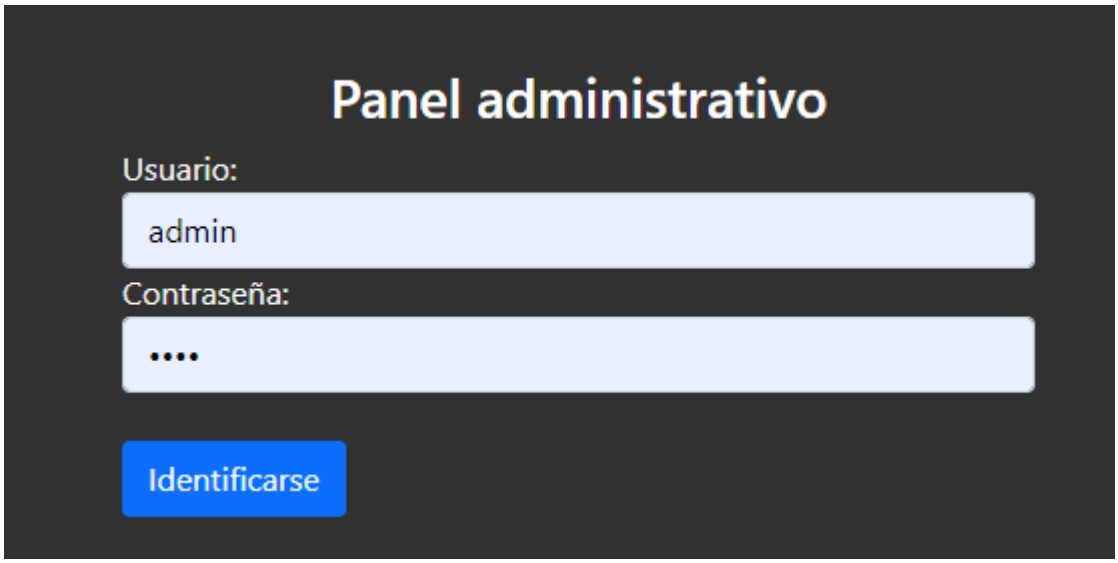

Figura 28 Resultado inicio sesión panel administrativo

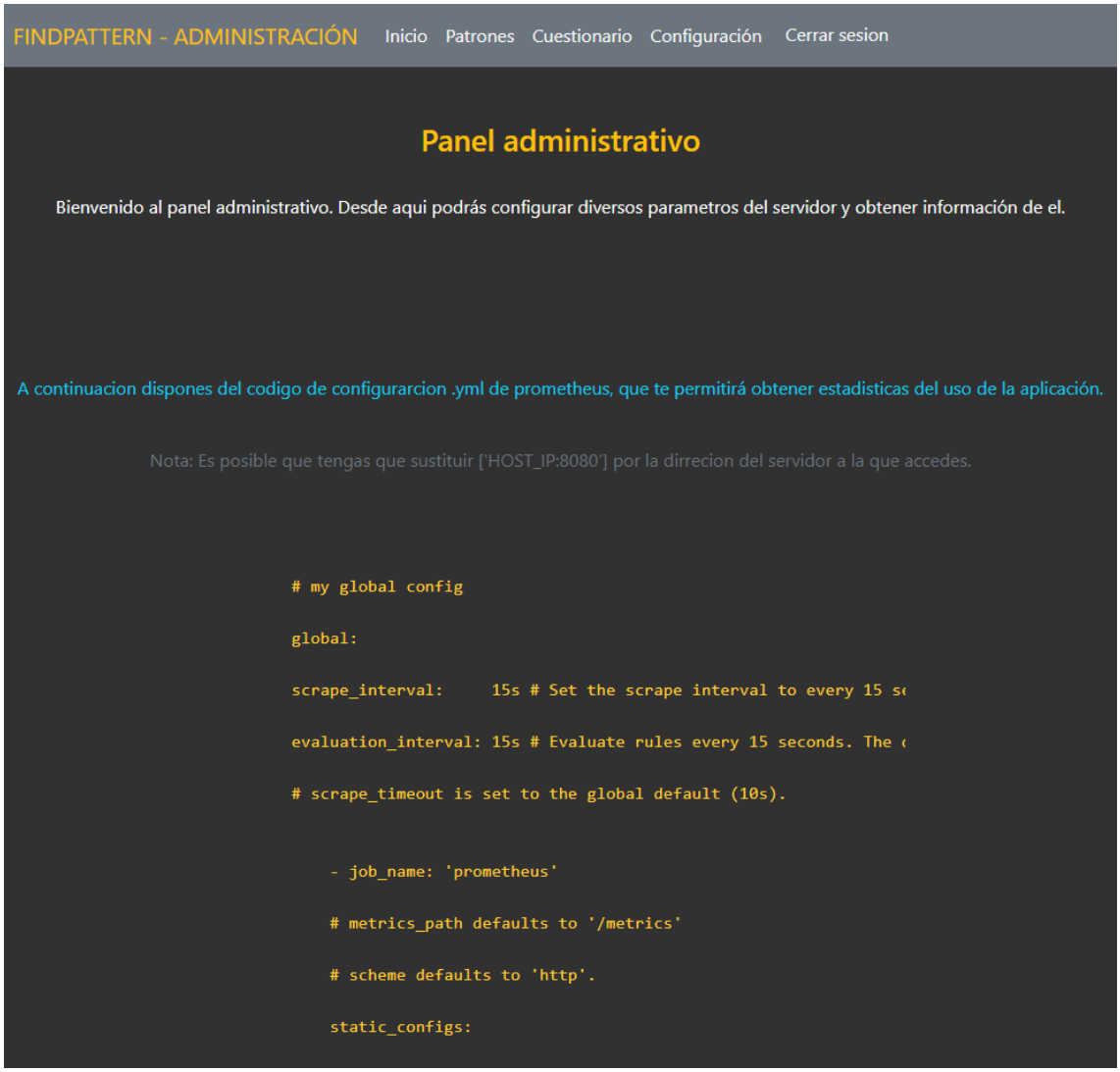

Figura 29 Resultado página principal panel administrativo

Diseño

- Presentación: Será necesario construir un sistema de permisos que solo permita obtener la vista en caso de tener permisos para ello. Esto se hará de forma que todo lo que este en el directorio web /admin/* no se muestre sin permisos. Además, se construirá un nuevo controlador encargado de llevar toda la lógica del panel administrativo. Este controlador se encargará de inicio y cierre de sesión, de la página de inicio de sesión y de la página de inicio. En caso de tener una lógica demasiado complicada, se separará en el futuro en varios controladores. Además, en el panel de inicio del panel administrativo se ofrecerá información sobre como conectar Prometheus con Spring-metrics y obtener así datos del servidor.

- Lógica: Se incluirá la configuración de permisos de la aplicación. Si intentas acceder alguna característica sin permisos, te redirigirá al panel de autentificación. También se añadirá spring-metrics, que será el encargado de procesar todos los datos de rendimiento y estado del servidor.

- Persistencia: Se añadirá al archivo de propiedades la configuración de seguridad. Esta estará compuesta de un nombre de usuario y una contraseña.

Es importante destacar a la hora de realizar esta tarea que será la primera vez que el estudiante se enfrenta a la autentificación en Spring. Por suerte, este ofrece una serie de ayudas por defecto.

Pruebas de aceptación

- Prueba aceptación: Es posible iniciar sesión en el panel administrativo.

CONDICIONES	Ventana de inicio de sesión.
PASOS	Introducir credenciales.
RESULTADO	Se iniciará sesión correctamente y se redirige al panel administrativo.
OBSERVACIONES	

- Prueba aceptación: Es posible consultar información del servidor.

CONDICIONES	**Sesión iniciada administración.**
PASOS	Ir a la pestaña de inicio de administración.
RESULTADO	Se podrá ver información sobre el estado del servidor o como acceder a ella.
OBSERVACIONES	

Tiempos

Podemos ver en la Figura 30 que ha tenido una especificación en la cual se ha invertido un tiempo notorio ya que nos enfrentábamos a una tecnología nueva y nunca

se había trabajado con seguridad. El tiempo de esperar Sprint corresponde también a programación, el cual se ha imputado en una categoría distinta por error.

Actividad	Primera estimación	Estimación actual (en horas)	Registrado	Debe estimarse
Registrar				☐
Esperar Prioridad				☐
Especificar Requisitos			1h 54m	☐
Diseñar y estimar				☐
Esperar Sprint			27m	☐
Programar	8h	8	8h 19m	☑
Aplicar Pruebas de Aceptación			35m	☐

Figura 30 Tiempos inicio panel administrativo

5.3.2 Panel Administrativo - Configuración

Desarrollaremos una explicación visual del archivo de propiedades de donde se pueden configurar distintos aspectos del servidor. La configuración no está expuesta en el panel al ser sensible para el funcionamiento del servidor o ser datos de acceso, por ejemplo, de conexión con la base de datos o de recuperación.

Bocetos

A continuación, en la Figura 31, mostramos un esquema básico de lo que se mostrará. No se ha concretado las propiedades a mostrar ya que durante las otras tareas se va a ir ampliando el contenido del archivo de propiedades del servidor.

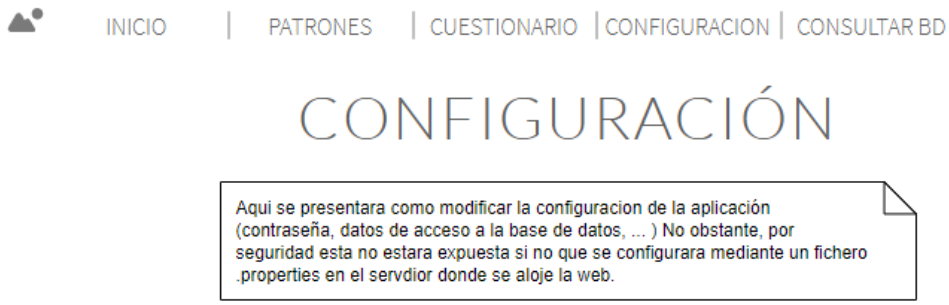

Figura 31 Panel Configuración Administración

En la Figura 32 podemos ver el resultado con todas las propiedades principales explicadas

Figura 32 Resultado configuración panel administrativo

Diseño

- Presentación: Se añadirá al controlador de administración la vista de la ventana de configuración.

- Lógica: Se incluirá la configuración que hace efectiva lo definido por el usuario en el archivo de propiedades. Para ello deberán exponerse ciertas propiedades de la aplicación.

- Persistencia: Se provee al usuario de un archivo de propiedades que puede configurar para hacer que el servidor se comporte como desea.

Pruebas de aceptación

- Prueba aceptación: Se podrá ver una explicación de la configuración del archivo propiedades.

CONDICIONES	**En la vista de configuración de panel administrativo**
PASOS	
RESULTADO	Se visualizará un texto explicativo del archivo y de cómo modificar la configuración.
OBSERVACIONES	

- Prueba aceptación: Se podrá modificar la configuración de la aplicación

CONDICIONES	**Tener disponible el archivo de propiedades**
PASOS	Modificar valores en el archivo
RESULTADO	La web se comportará de acuerdo con los valores modificados
OBSERVACIONES	

Tiempos

En la Figura 33 podemos observar el tiempo invertido. Esta tarea era relativamente sencilla de añadir sobre el contenido de panel administrativo previamente construido.

Actividad	Primera estimación	Estimación actual (en horas)	Registrado	Debe estimarse
Registrar				☐
Esperar Prioridad				☐
Especificar Requisitos			40m	☐
Diseñar y estimar				☐
Esperar Sprint				☐
Programar	4h	4	3h 45m	☑
Aplicar Pruebas de Aceptación			45m	☐

Figura 33 Tiempos de desarrollo de la configuración administrativa

5.3.3 Panel Administrativo - Patrones

.

Es necesario la construcción de un sistema que permita desde un menú de configuración añadir, eliminar o modificar patrones. Esto se hace con la finalidad de que se pueda realizar sin conocimientos técnicos y accediendo de forma remota a la aplicación desplegada.

Inicialmente se planteaba añadir por JSON los distintos patrones, pero finalmente se ha optado por desarrollar una estructura de texto que permite incluir patrones de forma sencilla. Esto facilita la exportación e importación de la información de forma sencilla.

Esta estructura ha sido construida en base a un lenguaje natural, donde un campo suele referirse a un campo de la base de datos. También se traducen entre sí para realizar conversiones, crear listas vacías cuando sea necesario u otros cambios. Esto está realizado de esta forma y no mediante un cuestionario ya que prevén cambios en el cuestionario en las siguientes versiones. Este formato también permite la fácil exportación o importación de datos para pruebas. En cambio, no se ha querido utilizar JSON porque se ha considerado que puede ser demasiado alejado del lenguaje natural o que puse resultar técnico y confuso. Esta técnica tiene previsto no encontrar algunos campos y darles una inicialización por defecto.

Bocetos

A continuación, en la Figura 34, vemos una aproximación del menú de edición. En la sección de 'Edición' se presentará el contenido actual de la aplicación en el formato especificado y se podrá añadir información nueva.

Internamente se realizará una validación del contenido enviado para garantizar que es correcto, antes de insertar nada.

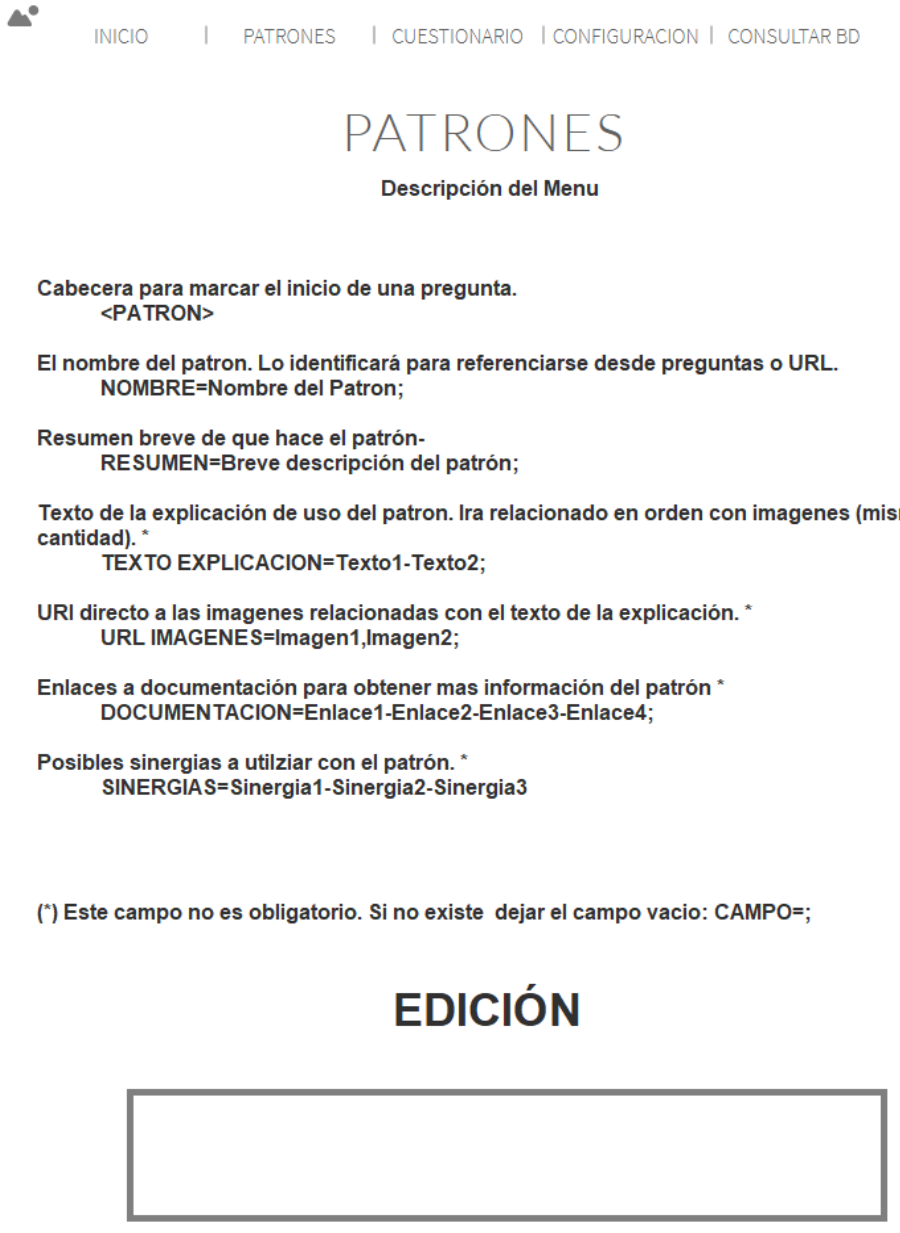

Figura 34 Boceto del panel de patrones de administración

Podemos observar un resultado de esta interfaz al final del trabajo en la Figura 35, la Figura 36 y la Figura 37.

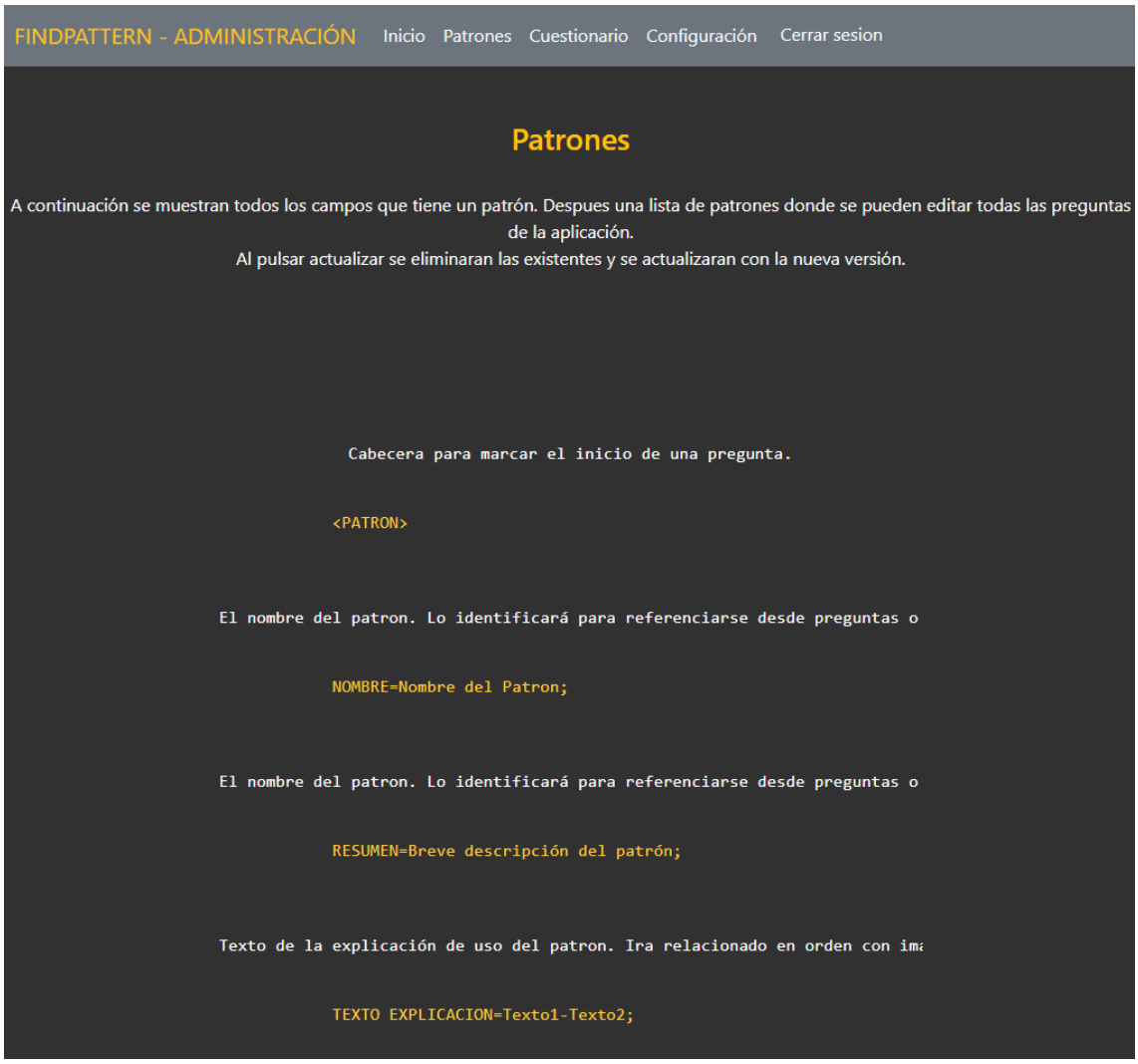

Figura 35 Resultado Patrones administración (1)

```
URl directo a las imagenes relacionadas con el texto de la explicación. *

        URL IMAGENES=Imagen1,Imagen2;


    Enlaces a documentación para obtener mas información del patrón *

        DOCUMENTACION=Enlace1-Enlace2-Enlace3-Enlace4;


            Posibles sinergias a utilziar con el patrón. *

        SINERGIAS=Sinergia1-Sinergia2-Sinergia3



(*) Este campo no es obligatorio. Si no existe  dejar el campo vacio: CAMPO=;
```

Figura 36 Resultado Patrones administración (2)

Figura 37 Resultado Patrones administración (3)

Diseño

En esta sección presentaremos como se ha estructurado este procesamiento de información y seguidamente su formato esperado.

- Presentación: Incluiremos una nueva vista para la modificación de información de los patrones. Se utilizará de forma general el controlador de la administración, salvo que crezca demasiado el número de operaciones a realizar. Actualmente

serán únicamente dos las que se realizarán desde esta vista. Será necesario garantizar la autentificación del usuario para la modificación de datos.

- Lógica: Es interesante hacer el procesado de la información lo más genérico posible, para facilitar la reutilización de contenido en otros paneles. Se construirá una clase con una serie de métodos generales que serán llamados desde otros más específicos, que harán uso de ellos para construir la información en el formato querido. Por ejemplo, un método general que busca un campo que es llamado desde uno especifico que le dice el nombre del campo a buscar.

- Persistencia: El funcionamiento de la inserción de la información y de la presentación intentara procesar el contenido. Si consigue procesar todo lo presentará/actualizará y si no dará un error. No existen resultados donde solo se presenta o actualiza una subsección.

Seguidamente, mostramos una sintaxis resumida de la construcción de un patrón, la cual se encuentra más extensamente explicada en la página de configuración de patrones:

<PATRON>

NOMBRE=Nombre del Patron;

RESUMEN=Breve descripción del patrón;

TEXTO EXPLICACION=Texto1-Texto2;

URl directo a las imágenes relacionadas con el texto de la explicación.

URL IMAGENES=Imagen1,Imagen2;

DOCUMENTACION=Enlace1-Enlace2-Enlace3-Enlace4;

SINERGIAS=Sinergia1-Sinergia2-Sinergia3

Pruebas de aceptación

- Prueba aceptación: Es posible interactuar con la información de patrones que tiene la aplicación.

CONDICIONES	**Acceder desde el panel de administración a la sección de patrones.**
PASOS	Realizar una modificación en la presentación. Pulsar enviar.
RESULTADO	Se ha actualizado la información de la aplicación.
OBSERVACIONES	En caso de inconsistencia se mostrará un error.

Tiempos

En la Figura 38 vemos como la especificación original ha crecido al verse modificado el formato de JSON a un lenguaje creado para la configuración de patrones de una forma más cerca al ser humano (y toda la lógica de transformación interna).

Actividad	Primera estimación	Estimación actual (en horas)	Registrado	Debe estimarse
Registrar				☐
Esperar Prioridad				☐
Especificar Requisitos			1h 32m	☐
Diseñar y estimar				☐
Esperar Sprint				☐
Programar	6h	14	14h 30m	☑
Aplicar Pruebas de Aceptación			2h 48m	☐

Figura 38 Tiempos del panel de administración de patrones

5.3.4 Panel Administrativo - Cuestionario

Se plantea un sistema que permita modificar, añadir o eliminar preguntas al sistema de cuestionario de poda creado anteriormente. Es por ello por lo que implementamos este menú con la posibilidad de hacer dichas modificaciones.

Esta tarea se planteaba añadir también mediante JSON. Se ha optado también por hacerla con un lenguaje como el mencionado en la tarea anterior por los mismos

motivos. Se reutilizará funcionalidad hecha de manera general con el fin de poder hacer el código de mayor calidad.

Bocetos

Podemos observar en Figura 39 el boceto del menú de configuración de cuestionario. En la sección de 'Edición' se presentará el contenido actual de la aplicación en el formato especificado y se podrá añadir información nueva.

Internamente se realizará una validación del contenido enviado para garantizar que es correcto, antes de insertar nada.

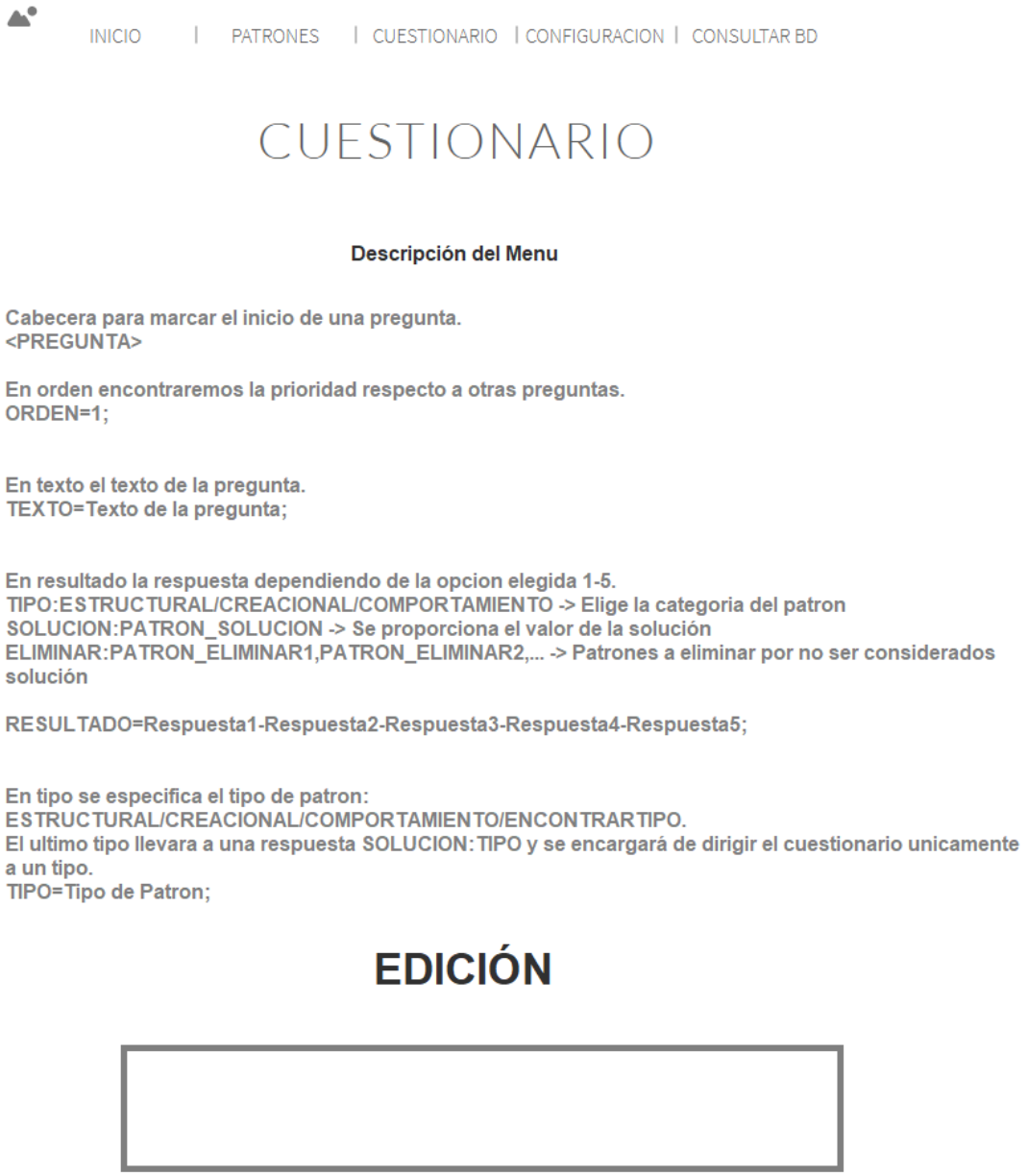

Figura 39 Menú de configuración de preguntas del cuestionario

En la Figura 40, la Figura 41 y la Figura 42 podemos observar los resultados al final de la aplicación de estas interfaces.

Figura 40 Resultado Administración Cuestionario (1)

Figura 41 Resultado Administración Cuestionario (2)

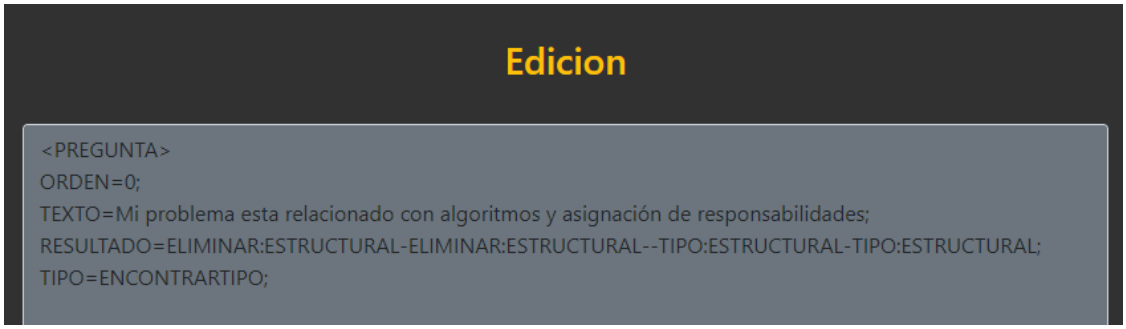

Figura 42 Resultado Administración Cuestionario (3)

Diseño

En esta sección, igual que en la tarea anterior, presentaremos como se ha hecho este procesamiento de información y después su formato esperado.

- Presentación: Se añadirá una nueva vista para la configuración de cuestionario. Aquí también utilizaremos el controlador de la administración, salvo que crezca demasiado el en tamaño. Será necesario garantizar la autentificación del usuario para la modificación de datos.

- Lógica: Interesará mantener el procesado genérico que hemos mencionado en la tarea anterior. Se construirán métodos específicos que hagan uso de estos métodos generales, como un método que construye el objeto de la pregunta en base a llamar a la búsqueda de campos genéricos que le devuelva estos campos procesados o un valor por defecto en caso de no existir.

- Persistencia: El funcionamiento de la inserción de la información y de la presentación intentara procesar el contenido. Si consigue procesar todo lo presentará/actualizará y si no dará un error. No existen resultados donde solo se presenta o actualiza una subsección.

A continuación, mostramos una sintaxis resumida de la construcción de una pregunta, la cual se puede observar en más detalle en el boceto anterior.

<PREGUNTA>

ORDEN=1;

TEXTO=Texto de la pregunta;

RESULTADO=Respuesta1-Respuesta2-Respuesta3-Respuesta4-Respuesta5;

Tipos resultado:

TIPO:ESTRUCTURAL/CREACIONAL/COMPORTAMIENTO -> Elige la categoria del patron

SOLUCION:PATRON_SOLUCION -> Se proporciona el valor de la solución

ELIMINAR:PATRON_ELIMINAR1,PATRON_ELIMINAR2,... -> Patrones a eliminar por no ser considerados solución

TIPO=Tipo de Patron;

Pruebas de aceptación

- Prueba aceptación: Se puede alterar y consultar las preguntas del cuestionario.

CONDICIONES	**Desde el panel administrativo accedemos al apartado de cuestionario.**
PASOS	Realizar una modificación. Presionamos enviar.
RESULTADO	Veremos cómo se ha modificado esta información en la aplicación.
OBSERVACIONES	En caso de inconsistencia se mostrará un error.

Tiempos

Se puede ver en la Figura 43 como la especificación original ha crecido al verse modificado el formato de JSON a un lenguaje creado para la configuración de patrones de una forma más cerca al ser humano (y toda la lógica de transformación interna).

Actividad	Primera estimación	Estimación actual (en horas)	Registrado	Debe estimarse
Registrar				☐
Esperar Prioridad				☐
Especificar Requisitos			1h 11m	☐
Diseñar y estimar				☐
Esperar Sprint				☐
Programar	6h	6	9h 30m	☑
Aplicar Pruebas de Aceptación			1h	☐

Figura 43 Tiempos de la construcción de la administración del cuestionario

6. Ejemplo de uso

En este capítulo, vamos a mostrar un ejemplo de uso del cuestionario desarrollado para encontrar patrones. El cuestionario, al tener cierta complejidad como hemos visto en la Figura 24 , merece ser mostrado en un apartado distinto con el fin de comprender el funcionamiento de este.

A los cuestionaros mostrados en los siguientes apartados se accederá desde la ventana principal, accediendo desde la barra superior en la opción “encontrar patrón” o desde el botón “buscar patrón” (áreas marcadas en naranja).

6.1 Ejemplo de cuestionario con solución

Supongamos que estamos desarrollando una librería para ser utilizada por distintas aplicaciones de la empresa. Estamos finalizando su desarrollo, pero nos surge un requisito especial: Queremos que los usuarios de la librería tengan acceso a las funciones de esta, pero no nos convence la idea de que se puedan referenciar todas las clases necesarias para ello Vamos a intentar encontrar un patrón que pueda ayudarnos a implementar una solución al problema. El cuestionario nos debe ayudar a encontrar el patrón adecuado frente a un problema de implementación concreto. Elegimos y pulsamos la opción “Buscar Patrón”. Al principio, tenemos las siguientes posibles soluciones para la búsqueda del tipo de patrón (explicadas previamente en el punto 5.2.4, en la página 72): Comportamiento, Estructural o Creacional.

Como primera pregunta nos muestra la de Figura 44. En esta pregunta responderemos con la opción “poco conforme” porque nuestro problema no está relacionado con algoritmos o asignación de responsabilidades.

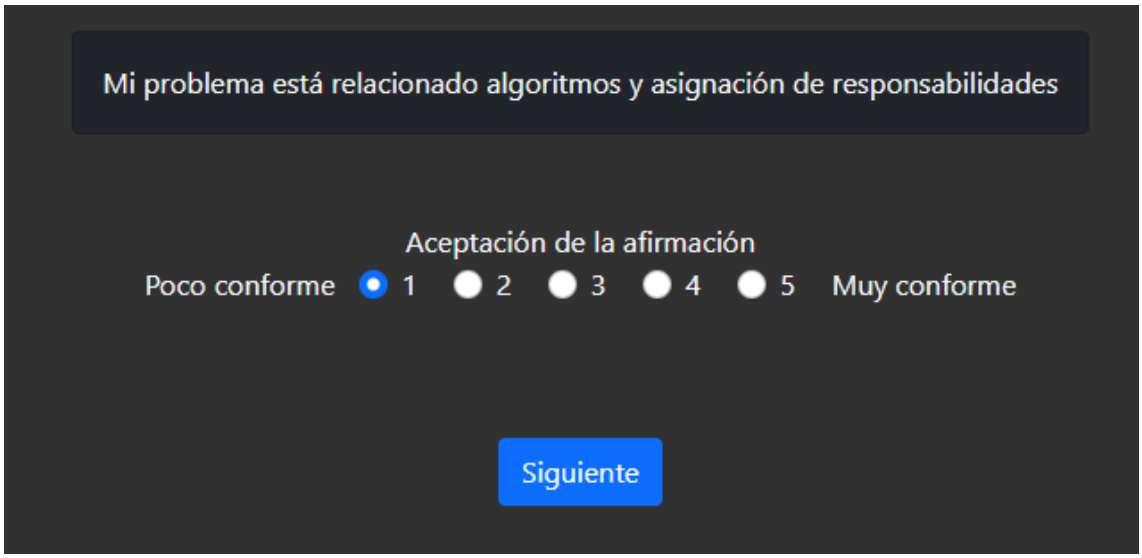

Figura 44 Primera pregunta del cuestionario de ejemplo

Esta respuesta eliminaría como posible solución el conjunto de patrones de comportamiento, tal y como se ha explicado en el apartado 5.2.4 de la página 72. Esto nos acota las posibles soluciones patrones de tipos: Estructurales o Creacionales.

La siguiente pregunta que nos aparece se puede ver en la Figura 45. En esta elegiremos que la opción “muy conformes” ya que buscamos modificar la forma de relacionarse a las clases, creando una nueva forma de relación entre las mismas.

Esta selección de la opción hará que el algoritmo seleccione como solución un tipo de patrón estructural. El sistema ante esta respuesta carga las preguntas necesarias para identificar el tipo de patrón estructural concreto. Estas preguntas podemos consultarlas en el apartado 5.2.4 de la página 72.

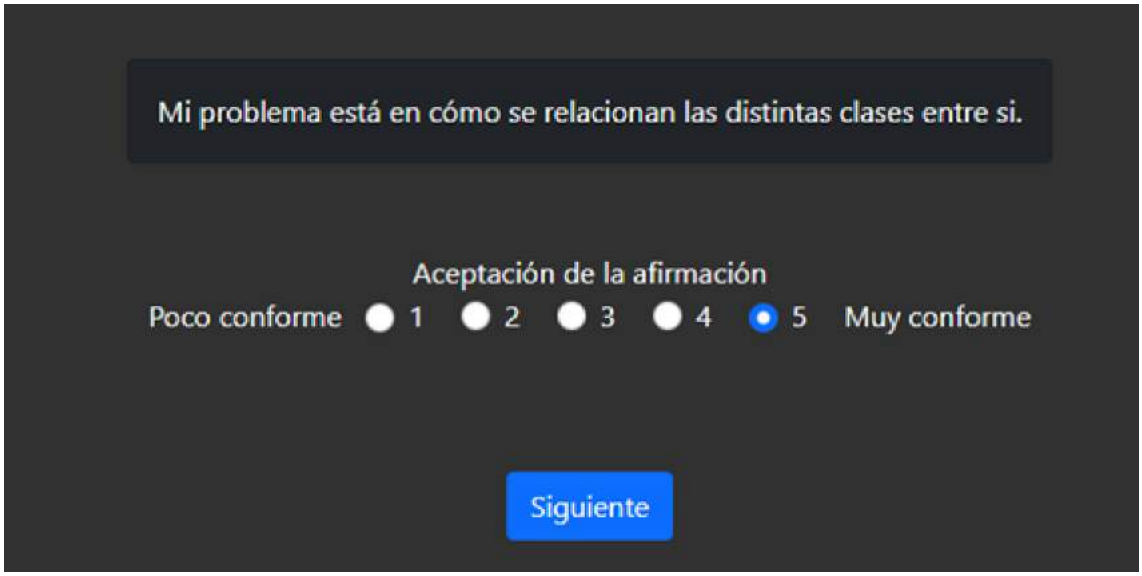

Figura 45 Segunda pregunta del cuestionario de ejemplo

Como posibles soluciones, después de conocer el tipo del patrón, contamos con los siguientes patrones: *Adaptader, Bridge, Facade, Composite, Decorator, Flyweight y Proxy*

La primera pregunta que encontraremos en esta nueva categoría de preguntas será la mostrada en la Figura 46. Esta tendrá una valoración negativa ya que nuestros objetos pueden y deben cooperar entre sí, lo que buscamos es una forma de solucionar el problema de que no queremos exponer todas las clases de la librería para hacerla accesible.

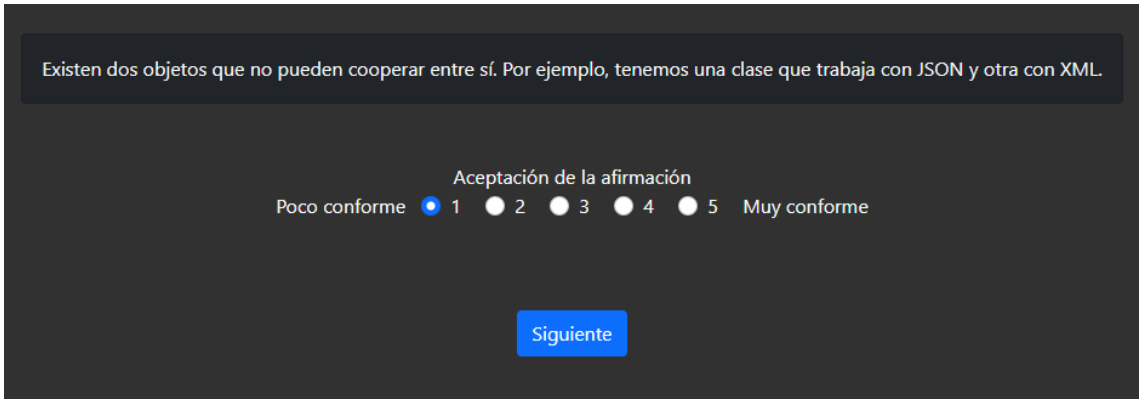

Figura 46 Tercera pregunta del cuestionario de ejemplo

- Esta respuesta nos eliminaría el patrón *Adapter* como posible solución, quedando las siguientes posibles soluciones: *Bridge, Facade, Composite, Decorator, Flyweight y Proxy.*

Una vez eliminada esta solución, el cuestionario carga la siguiente pregunta que tenga como solución o como eliminación un tipo que todavía se encuentre en la lista de posibles soluciones. En este caso cargará la pregunta mostrada en la Figura 47*. Esta tendrá una respuesta de "poco de acuerdo" ya que no buscamos la división, si no quizá algún mecanismo para mejorar su acceso.*

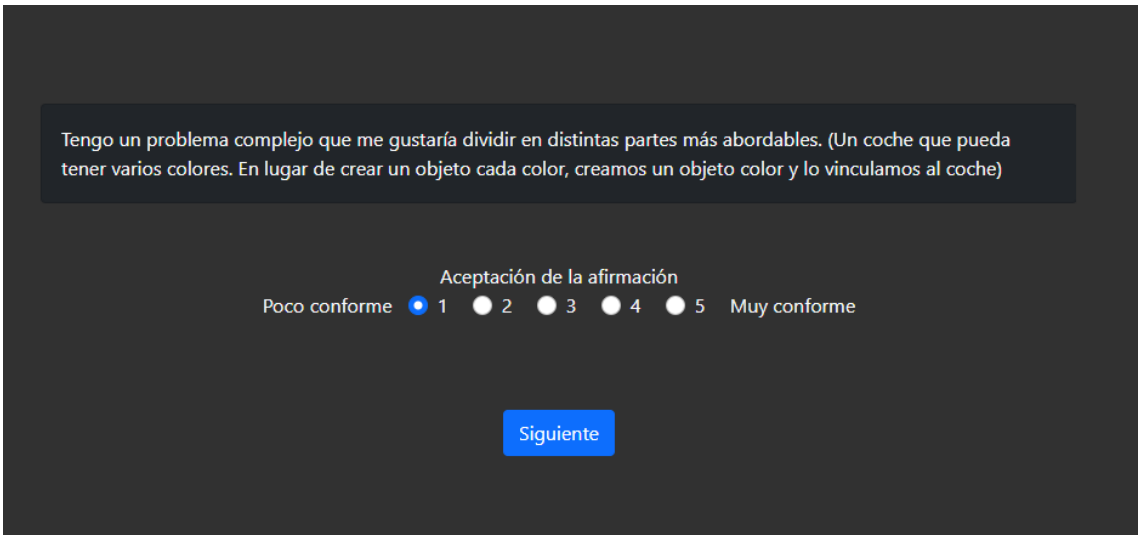

Figura 47 Cuarta pregunta del cuestionario de ejemplo

Esto llevará a la eliminación del patrón Bridge y su solución/eliminación de todas las preguntas que hagan referencia a este. En caso de que exista alguna pregunta que solo tuviera este patrón ya como contenido, la pregunta será eliminada. Si tenía otra solución/eliminación, se elimina Bridge de sus parámetros y se mantiene la pregunta. Este proceso se puede observar más detallado en la Figura 24. Por tanto, las posibles soluciones que todavía podemos encontrar son posibles soluciones: *Facade, Composite, Decorator, Flyweight y Proxy*

Como siguiente pregunta la aplicación nos ofrece la pregunta mostrada en la Figura 48. Esta pregunta tendrá como respuesta "muy de acuerdo", ya que nos ofrece una solución para poder acceder a nuestras clases sin exponerlas directamente. Este patrón puede ayudarnos a implementar una posible solución al problema que tenemos.

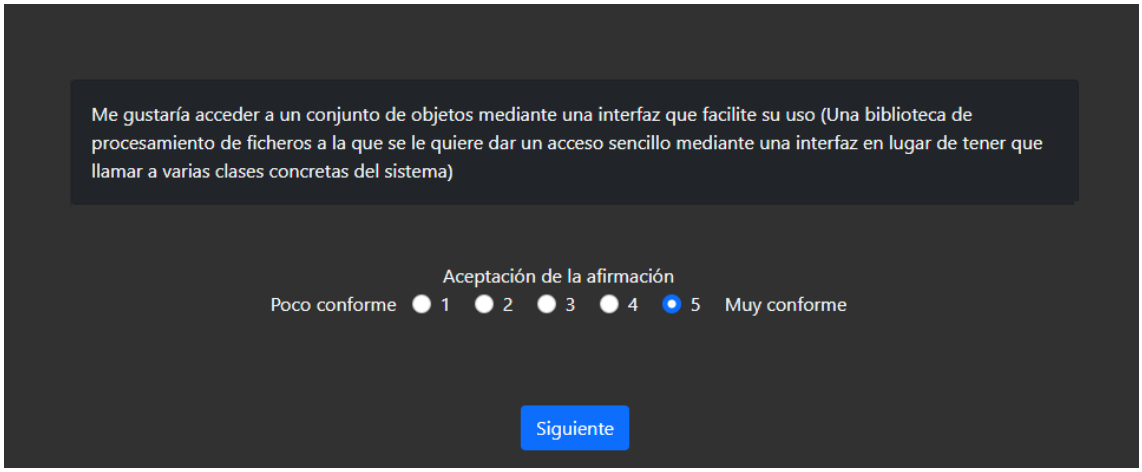

Figura 48 Quinta pregunta del cuestionario de ejemplo

Al marcar esta respuesta, como observamos en el apartado previamente mencionado, encontramos como solución el patrón ***Facade*** (o Fachada en español). Esto nos redirige a la página del patrón (mostrada previamente) donde podemos encontrar todos los datos que tenga disponibles. A modo de ejemplo se muestra un recorte de la página superior del patrón en la Figura 49. Con esto concluimos la ejecución de la herramienta con este patrón, el cual se determina como solución.

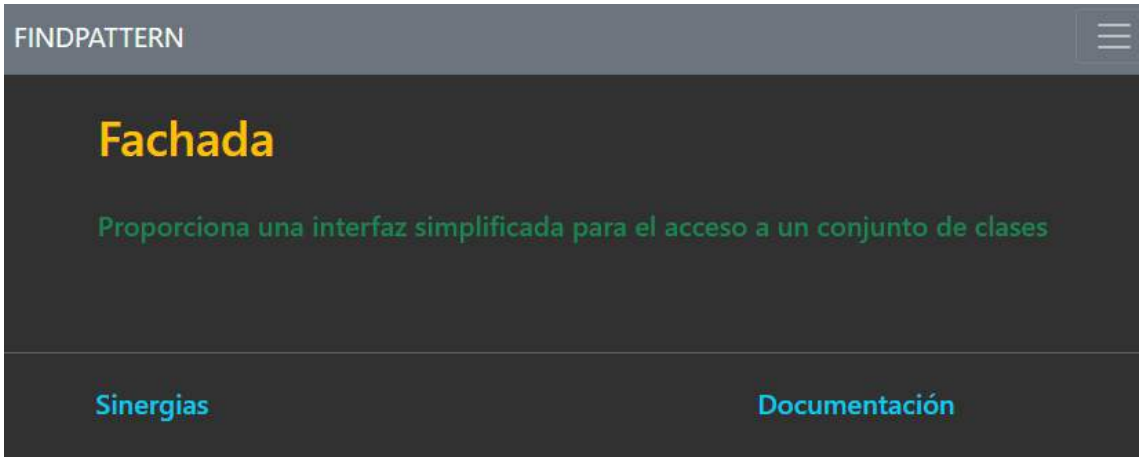

Figura 49 Patrón solución

6.2 Ejemplo de cuestionario sin solución

En caso de finalizar todas las preguntas disponibles sin encontrar una posible solución, la aplicación nos mostrará un mensaje que insta a que volvamos a probar. Esto podría realizarse siguiendo otro camino distinto. En futuras iteraciones, podría modificarse para redirigir de alguna manera por otras preguntas posibles del cuestionario que puedan llevar a la solución o incluir algún tipo de pregunta comodín. Este mensaje de error es mostrado en la Figura 50.

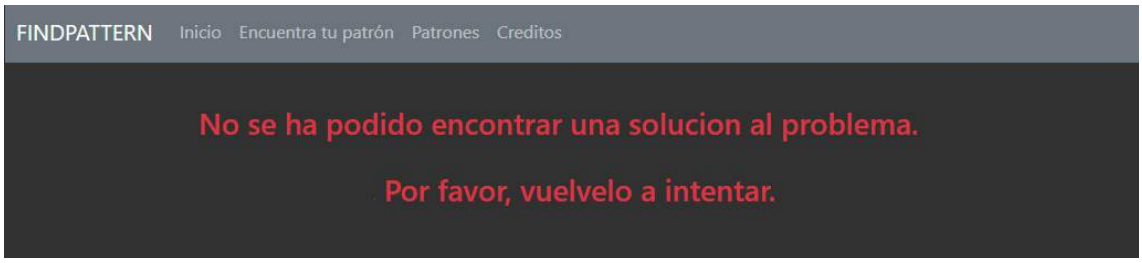

Figura 50 Mensaje de error del cuestionario

7. Conclusiones

En el proyecto se han empleado muchas horas de trabajo y se ha adquirido una notable experiencia tanto en patrones como en las tecnologías empleadas. Ha habido un reto extra causado por el COVID-19 ya que esto supone tener que hacer todas las comunicaciones del trabajo de forma remota.

Observando el resultado final se valora positivamente el mismo, ya que se ha logrado un desarrollo de una aplicación funcional y el alumno ha adquirido amplios conocimientos de la herramienta. En cuanto al *Backend*, al inicio de su desarrollo el alumno no había utilizado tecnologías como Spring y sus conocimientos de desarrollo web se veían limitados a una pequeña aplicación realizada. Sobre el *Frontend no existía experiencia previa. Pero gracias a la información consultada mencionada se ha podido adquirir una amplia base de conocimiento para realizar este trabajo.*

Inicialmente, se pensaba que quizá se podrían abordar más características de las que los usuarios remarcaban como importantes en las preguntas que se les realizaron. No obstante, hemos podido abordar la varias de ellas importantes que pueden suponer una base para trabajos futuros.

En su estado actual, supone un prototipo ideal de lo que se plantea, para que pueda ser llevado a producción y analizar con el paso del tiempo el impacto de los usuarios. Se propone que esta aplicación sea analizada en el futuro curso por los alumnos y que se pueda volver a iterar con las sugerencias de estos.

8. Relación del trabajo con los estudios cursados

Es de gran relevancia el conocimiento adquirido a lo largo de los años que se ha cursado el Grado de Ingeniería Informática para la obtención del conocimiento que ha permitido el desarrollo de este trabajo. En concreto destacamos:

Ingeniería del software y gestión de proyectos: Gracias a estas asignaturas se ha podido adquirir el conocimiento necesario para la gestión y el análisis de un proyecto en sus distintas etapas. Por otro lado, contamos con los métodos para mitigar los errores de planificación o verificar que se está avanzando correctamente por ella. Es gracias a estas asignaturas que se ha podido medir y controlar de forma adecuada el desarrollo de este proyecto. Además, también gracias a ingeniería del software hemos podido saber cómo construir un software de calidad.

Interfaces persona computador: Mediante el conocimiento obtenido a través de esta asignatura contamos con los métodos para crear una interfaz correcta. Asimismo, conocemos la importancia de estas a la hora de ofrecer una buena experiencia. Otra información obtenida es como hacer la interfaz más amigable para el usuario o más fácilmente navegable.

Bases de datos: Por otro lado, a través de esta asignatura conocemos los distintos tipos de bases de datos y la estructura interna de la misma. Esto ha sido útil a la hora de comprender los problemas de almacenamiento de este proyecto.

Rama de ingeniería del software: Cabría destacar que se ha utilizado conocimientos adquiridos de todas las asignaturas de la rama, ya que estas te van enseñando distintos aspectos del desarrollo del software y en este trabajo ha habido todo el flujo de desarrollo del software. Gracias a los conocimientos obtenidos en esta rama ha podido hacerse todos los pasos del software de forma correcta, desde sus primeros requisitos especificados y analizados a el sistema desarrollado que tenemos actualmente. Otro aspecto interesante es la mejora de la metodología ágil empleada, como ha sido mencionado en el apartado del plan de trabajo. Destacaría especialmente la asignatura de Diseño **del software**, ya que en ella es donde se trabajó con patrones de diseño y la motivadora de todo este desarrollo.

Por otro lado, para el desarrollo de este proyecto ha sido crucial la **competencia trasversal** de **análisis y resolución de problemas** por darnos la habilidad para ser capaces de analizar los distintos problemas que han ido apareciendo y saber resolverlos para continuar con el desarrollo. No podríamos no destacar también el uso de la

competencia **de innovación, creatividad y emprendimiento** ya que hemos propuesto una solución creativa a un problema común en el día a día del ingeniero de software.

Tendríamos que destacar la competencia de **aprendizaje permanente**, ya que en este trabajo la mayoría de las tecnologías han sido nuevas para el alumno y ha continuado su aprendizaje. Por último, sería interesante mencionar la competencia de **planificación y gestión del tiempo**, ya que el trabajo al ser extenso comenzó a desarrollarse de una forma planificada y gestionada.

9. Trabajos futuros

Las tareas pendientes que podrían tener relevancia para la aplicación serían las otras dos funcionalidades que los alumnos parecían interesados, pero no ha sido posible abordar en esta primera versión:

- Herramienta para la búsqueda de un patrón en el código: Este ayudante permitiría dado un código la búsqueda de código donde podría aplicarse un patrón de diseño para la mejora de este. Esta herramienta tendría el problema de que cada lenguaje podría tener sus problemas. Es una herramienta que puede resultar realmente compleja de diseñar y requeriría de algoritmos de aprendizaje, ya que se enfrentaría a multitud de casos distintos. Quizá obtener cierta información sobre la aplicación pueda ayudar al análisis de esta. Se propone elegir un subconjunto de patrones y aplicar la búsqueda para estos

- Generación del código del patrón: Dado un código donde queremos aplicar un patrón, esta utilidad se encargaría de la generación del código o parte de el para la aplicación del patrón. Podríamos ayudarnos de una serie de preguntas que pueden ser realizadas al usuario. Al igual que la herramienta anterior, dependería mucho del lenguaje de programación e incluso de la versión de este. Además, quizá la generación de código no sirviera con los estándares de la empresa o del trabajo.

Otro posible trabajo futuro sería ampliar la información presente en la aplicación e ir mejorando el cuestionario y presentación de la información. Consideramos esto como el trabajo futuro más viable ya que se podría ir realizando a partir de las sugerencias de los alumnos y conseguir una herramienta completa antes de seguir explorando hacia otras nuevas.

10. Agradecimientos

Quiero dedicar en especial **a mis padres Teresa y Julián**. Ambos me han enseñado a no rendirme en la búsqueda de mis objetivos por difícil que sea, y es gracias a ellos que puedo estar hoy cursando estos estudios que siempre he querido alcanzar.

También **a mi pareja**, ya que gracias a su apoyo constante durante todo este trabajo he podido esforzarme en el para sacar el mejor resultado posible.

A **mi tutor, Vicente Pelechano**, ya que me ha sabido guiar a alcanzar la aplicación que quería realizar, dejándome libertad creativa dentro de la misma.

No puedo olvidarme de **mis amigos**, que me han estado acompañando durante tantos años hasta llegar a este trabajo.

A **mis compañeros de trabajo** y a mi empresa, ya que gracias a ellos he podido adquirir más conocimientos sobre el mundo de la programación.

Referencias

[1] J. Thompson, «Spring Framework 5: Beginner to Guru,» [En línea]. Available: https://www.udemy.com/course/spring-framework-5-beginner-to-guru/.

[2] J. Thompson, «Testing Spring Boot: Beginner to Guru,» [En línea]. Available: https://www.udemy.com/course/testing-spring-boot-beginner-to-guru/.

[3] Emprenderalia, «¿Qué es el Producto Mínimo Viable (MVP)?,» [En línea]. Available: https://www.emprenderalia.com/que-es-el-mvp-producto-viable-minimo/.

[4] IBM, «Test-driven development,» IBM, [En línea]. Available: https://www.ibm.com/garage/method/practices/code/practice_test_driven_development/. [Último acceso: 2021].

[5] E. Gamma, R. Helm, J. Ralph y V. John, Design Patterns: Elements of Reusable Object-Oriented Software, Addison Wesley, 1994.

[6] E. Gamma, Interviewee, *How to Use Design Patterns.* [Entrevista]. 23 Mayo 2005.

[7] E. Freeman, . E. Robson, B. Bates y . K. Sierra, Head First Design Patterns, O'Reilly Media, Inc., 2004.

[8] dev.to, «The K.I.S.S Principle in Programming,» [En línea]. Available: https://dev.to/kwereutosu/the-k-i-s-s-principle-in-programming-1jfg.

[9] Baeldung, «Refactoring in Eclipse,» 8 Diciembre 2019. [En línea]. Available: https://www.baeldung.com/eclipse-refactoring.

[10] Jetbrains, «Generación de código y refactorización,» [En línea]. Available: https://www.jetbrains.com/es-es/objc/features/refactorings-and-code-generation.html.

[11] P. Rodríguez, «La deuda técnica, un lastre para las tecnológicas: un estudio señala que los informáticos pierden casi un día de trabajo a la semana para solventarlas».