GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

USABILITY ANALYSIS TOOL

Alumna: Anto Cisneros, Diana Yolanda

Director: Villamañe Gironés, Mikel

Curso: 2017-2018

Fecha: 18 de julio de 2018

RESUMEN

La realización del estudio de usabilidad en las aplicaciones se hace cada vez más importante, ya que el éxito de la misma está directamente relacionado con su grado de usabilidad. Por ello se ha implementado una herramienta web llamada Usability Analysis Tool, que sirva de apoyo para atenuar el trabajo que conlleva realizar ese estudio.

El usuario puede hacer uso de esta herramienta web, con la finalidad de obtener resultados numéricos y gráficos de los tres objetivos básicos de la usabilidad, que son: la efectividad, la eficiencia y la satisfacción.

Los resultados que ofrece la herramienta web acerca del objetivo de la efectividad, son las estimaciones de la tasa de éxito muestral, la tasa de éxito poblacional y los intervalos de confianza. Por otro lado, los resultados de la eficiencia son las estimaciones de la mediana del tiempo de ejecución de las tareas, intervalos de confianza y el nivel de pérdida. Y por último los resultados de la satisfacción son las estimaciones del grado de satisfacción, la media de éste y su respectivo intervalo de confianza.

Con los resultados obtenidos, el usuario analista puede identificar aquellas acciones en las que los usuarios que han hecho uso de la aplicación, han tenido problemas al momento de realizarlas; para posteriormente decidir si mejorar o no la aplicación que este evaluando.

ABSTRACT

The study of usability in applications is more and more important, as the success of it is directly related to its usability degree. For this reason, a web tool called Usability Analysis Tool has been implemented, which aim is to be a support to mitigate the work involved in making this study.

The user can make use of this web tool, in order to obtain numerical and graphical results of the three principal objectives of usability, which are: effectiveness, efficiency and satisfaction.

The results about the objective of effectiveness offered by the web tool are the estimations of the sample success rate, the population success rate and confidence intervals. On the other hand, the efficiency results are the estimations of the median of the execution time of the tasks, confidence intervals and loss level. Finally, the results of satisfaction are estimations of the satisfaction degree, its average and its respective confidence interval.

With those results, the analyst user can identify those actions in which users who have made use of the application have had problems when performing them, and then decide whether to improve or not the application that is evaluating.

LABURPENA

Aplikazioak garatzean erabilgarritasunari buruzko azterketa egiteari gero eta garrantzi handiagoa ematen zaio, izan ere, sistemaren arrakasta eta erabilgarritasun maila zuzenean lotuta daude. Horregatik, Usability Analysis Tool izeneko web tresna sortu da. Erabilgarritasun azterketa horiek burutzeko lanak arintzeko intentzioarekin.

Erabiltzaileek web tresna hau erabil dezakete, erabilgarritasunaren oinarrizko hiru helburuen emaitzak kalkulatzeko eta haiei buruz grafikoak lortzeko. Hiru helburu horiek efikazia, efizientzia eta asebetetzea dira.

Efikaziari buruzko web tresnak eskaintzen dituen emaitzak atazen arrakasta-tasa, populazioaren arrakasta-tasa eta konfiantza-tarteak dira. Bestalde, efizientziari buruz atazak burutzeko erabilitako denbora, konfiantza-tarteak eta erabiltzaileen galera-faktorea kalkulatzen dira. Eta, azkenik, erabiltzaileen asebetetze maila kalkulatzen da galdetegi bat erabiliz eta batazbestekoa eta bere konfiantza-tarteak kalkulatzen dira.

Lortutako emaitzekin, analistak erabiltzaileari aplikazioa erabiltzerakoan arazoak eman dizkion atazak identifikatu ditzake, eta ondoren, ebaluatzen ari den aplikazioa hobetzeko erabakiak hartu.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

Este proyecto consiste en realizar una herramienta web que sirva de herramienta de apoyo al análisis de usabilidad de diferentes aplicaciones. En concreto facilita el cálculo de los tres objetivos básicos de la usabilidad. Estos objetivos son la efectividad, la eficiencia y la satisfacción.

La persona responsable de realizar el estudio de usabilidad o analista, podrá hacer uso de esta herramienta web llamada Usability Analysis Tool en la que por medio de acciones que realice con la misma pueda definir las pruebas que considere necesarias para medir los tres objetivos de la usabilidad.

Lo que hoy llamamos usabilidad existe desde siempre en la historia de la tecnología, no con ese nombre, pues no se trataba de un estudio como tal, sino como algo tan simple como usar el sentido común y la experiencia para construir objetos útiles, fáciles y cómodos de manejar, por esa razón, es importante realizar un análisis de usabilidad, antes, durante y al finalizar una aplicación.

Por otro lado, la norma ISO 9241 la define como "*el grado en el que un producto puede ser utilizado por usuarios específicos para conseguir objetivos específicos con efectividad, eficiencia y satisfacción en un determinado contexto de uso*" (ISO 9241-11:1998, 2013).

Estos tres objetivos han ido dando forma un nuevo objeto de estudio que se aplica en distintas disciplinas. A continuación, se hace una breve reseña de lo que se significa cada uno de los objetivos.

La efectividad está principalmente relacionada con la precisión con la que los usuarios utilizan una aplicación para alcanzar objetivos específicos. Esta idea va relacionada con la facilidad de aprendizaje. Por ello la calidad de la solución y la tasa de errores son indicadores de la efectividad (Zaharias, 2004).

La eficiencia e**s** la relación que existe entre la efectividad y el esfuerzo o los recursos empleados para lograr una tarea. En la eficiencia influyen el tiempo de finalización de las tareas y el tiempo de aprendizaje, es decir, la rapidez con la que se aprende a usar la aplicación (Zaharias, 2004).

La satisfacción mide el interés del usuario por el contenido de la aplicación, es decir el grado de satisfacción y la comodidad, con actitudes positivas al utilizar la aplicación para alcanzar objetivos específicos (Zaharias, 2004).

Para hacer el análisis de usabilidad se hacen diferentes pruebas con usuarios, a estos usuarios en este proyecto se les llamará testers. Estas pruebas servirán para saber si la aplicación es lo suficientemente usable. Para ello se pide al tester que realice tareas sencillas y se valora la rapidez y la facilidad con la que realizan dichas tareas. Con los resultados obtenidos de estas pruebas, el analista podrá valorar cuales son las tareas en las que los testers han tenido mayor dificultad y centrar su atención en esa tarea en concreto.

Para finalizar, es importante que el tester sienta que tiene el control y la libertad de uso. Es por esa razón que la presencia de un estudio de usabilidad al realizar una aplicación es de suma importancia, ya que son los testers y no los diseñadores o los desarrolladores, los que determinan si una aplicación es fácil de utilizar.

1.1 Motivaciones para la elección del proyecto

La motivación principal para la elección de este proyecto, es la de desarrollar una herramienta web que sea capaz de medir los objetivos de la usabilidad de otras aplicaciones. Es decir, que el analista pueda realizar estudios de usabilidad con gran facilidad sin importar de que trate la aplicación que desea evaluar, facilitar los cálculos estadísticos que son necesarios para saber si se superan con éxito los objetivos.

Otra de las motivaciones es que la desarrolladora puede planificar el trabajo como ella crea conveniente, dándole así la oportunidad de aprender a gestionar un proyecto de principio a fin, así mismo, de aprender nuevas tecnologías y lenguajes de programación. Lo que le supone un gran reto y un gran avance en el comienzo de su carrera profesional.

2. PLANTEAMIENTO INICIAL

Este apartado se descompone en varios puntos en los que se exponen la forma de desarrollo del proyecto, empezando por los acrónimos y abreviaturas que son los términos más importantes para comprender la documentación. Continuando con la definición de los objetivos que se quieren alcanzar, la arquitectura a utilizar, la estructura y descomposición de las tareas que se van a realizar, la planificación temporal, así como también la gestión de riesgos y la evaluación económica que conlleva la realización de este proyecto.

2.1 Definiciones, Acrónimos y abreviaturas

Back-end: es la parte que se encarga del servidor, es quién interactúa con las bases de datos, es quien manipula los datos, y por lo tanto se encarga de su seguridad y de cubrir posibles vulnerabilidades. "*Un back-end debe procesar grandes cantidades de información de forma fluida y el mejor rendimiento posible, haciendo un uso lo más eficiente de los recursos disponibles"*. (Mollá Sirvent, 2016)

Front-end: es la parte responsable de recolectar los datos de entrada y ajustarlos a las especificaciones que demanda el back-end. En él se trabajan con las tecnologías que se ejecutan del lado del cliente, estas tecnologías se ocupan principalmente del diseño y la estructura de las páginas.

Framework: entorno de desarrollo que permite el uso de librerías, programas y lenguajes, que ayudan al desarrollo y organización de software.

Salt: es un número de dígitos aleatorios, pueden ser letras o números que se agregan al inicio o final del Hash.

Hash: es un código que se obtiene después de aplicar un algoritmo especial a una cadena de texto, en este caso se ha utilizado el algoritmo "*md5*", y a cadena de texto es la contraseña y un Salt que se ha generado de forma aleatoria, esta cadena se encripta y se guarda en la base de datos junto con el Salt.

Rest: deriva de "*Representational State Transfer*", también conocido como servidor API. Es un servicio web que no tiene estado, lo que quiere decir, es que, si se realizan dos llamadas cualesquiera, el servicio pierde todos los datos entre una llamada y otra. Por ejemplo, si se realiza una petición con un nombre y contraseña, no se puede esperar que el servicio recuerde esos datos, se tienen que enviar los datos exactos que el servidor espera recibir para llevar a cabo la petición.

Token: es una firma cifrada que permite al servidor API identificar al usuario, este token es creado en la parte del servidor con una clave que solo se conoce en el servidor. En el token se pueden guardar datos como su email, nombre o cualquier otra información que se crea necesaria.

Usuario: es como se llama al analista que hará uso de Usability Analysis Tool.

Proyecto: es un término que representa a la aplicación a evaluar y que el usuario crea en la herramienta web.

Nodo: representa a una página o un lugar concreto de la aplicación a evaluar.

Path (óptimo o alternativo): un path es el camino que hay que seguir en un orden determinado para realizar una tarea, un path está compuesto por nodos, esos nodos representan las páginas que se tienen que visitar para realizar una tarea. Por consiguiente, una tarea puede tener uno o varios caminos óptimos o alternativos.

Tarea: es una acción que el usuario establece y que posteriormente será realizada por testers. Las tareas tienen paths óptimos y/o alternativos.

Path realizado: es el camino de nodos que ha seguido el tester para realizar la tarea.

Tarea realizada: es la tarea que ha realizado el tester en un tiempo determinado. Las tareas realizadas tienen paths realizados.

Cuestionario: como su propio nombre lo dice es un cuestionario y contiene preguntas, que son establecidas por el usuario.

Grafo de navegación: es una forma gráfica de mostrar las páginas que han visitado los testers para realizar una tarea. Esas páginas están representadas por nodos y el recorrido realizado, por arcos.

Tester: como ya se ha definido en la *Introducción* el tester es la persona que realiza las tareas o cuestionarios que el analista haya preparado para realizar el estudio.

2.2 Descripción del proyecto

Usability Analysis Tool es una herramienta web a la que puede tener acceso cualquier usuario que desee realizar un estudio de usabilidad, y necesite automatizar el proceso para obtener los resultados. En concreto, para medir si la aplicación a analizar cumple con los tres objetivos básicos que se han descrito en la introducción de este proyecto (la efectividad, la eficacia y la satisfacción).

Con Usability Analysis Tool, el usuario se puede registrar e iniciar sesión cómo en cualquier otra web. Cuenta con un área personal, en la que creará un proyecto para cada aplicación que quiera analizar.

Dentro de cada proyecto pueden crear las tareas que considere necesarias para el estudio, así como también un cuestionario y las respuestas obtenidas de los testers.

Una vez tenga configurado el proyecto con todo lo necesario, puede generar un informe detallado para cada objetivo o un informe completo. Cada apartado del informe está compuesto por tablas que han sido generadas mediante cálculos estadísticos. Así mismo dispone de un gráfico para visualizar de una forma más rápida las tareas en los que los testers han tenido mayor dificultad para realizarlas.

Con los resultados obtenidos el usuario puede ver si debería mejorar o no la usabilidad de la aplicación, centrando su atención a aquellas tareas en las que se identifican los problemas.

2.3 Objetivos

Como en todo proyecto es importante trazar los objetivos que se quieren alcanzar, en este caso son tres, y se consideran objetivos principales:

- Medir la efectividad, eficiencia y satisfacción de cualquier aplicación de forma automática, con datos proporcionados por el usuario. Ofreciendo unos resultados comprensibles que permitan identificar los posibles problemas.

- Aprender y adquirir destreza en la elaboración de páginas web, el manejo de nuevos lenguajes de programación y tecnologías.

- Crecimiento personal, ya que con la elaboración de esta memoria se aprende a gestionar un determinado proyecto, e identificar las etapas o fases necesarias antes y después de empezar a desarrollar una herramienta web.

2.4 Arquitectura

En este apartado se destaca la importancia que tiene la arquitectura para organizar la herramienta web de manera que sea lo más eficiente posible, que sea fácil de mantener y de ampliar en un futuro. Ya que de ello depende la rapidez de la respuesta. La comunicación ha de ser buena para ofrecer al usuario un servicio óptimo.

En principio se ha realizado una investigación sobre qué arquitectura aplicar a este proyecto. Concretamente, se ha realizado la comparación entre las arquitecturas SOAP (Simple Object Access Protocol) y REST (Representational State Transfer). Aunque ambas son arquitecturas que ofrecen en apariencia las mismas funcionalidades, no hacen lo mismo.

SOAP es un protocolo para intercambiar mensajes entre redes, generalmente usando HTTP, aunque no está limitado a este protocolo puede utilizar FTP (Protocolo de transferencia de archivos), POP3 (Protocolo de oficina de correo), TCP (Protocolo de control de transmisión), etc.

SOAP sólo soporta XML lo cual en ciertas ocasiones facilita la lectura, y se puede utilizar para formar protocolos más complejos y completos según se necesite, lo que provoca que los mensajes resulten más largos y por ende son más lentos de transferir, esto resulta muy complejo y poco flexible, por lo que se ha decidido utilizar REST.

REST es una arquitectura que sigue unas ciertas reglas, entre ellas: se basa en el sistema cliente servidor, es un modelo de aplicación de capas, utiliza peticiones sin estado, interfaces uniformes que identifiquen los recursos y utiliza la caché para mejorar la eficiencia de la red, por lo que las respuestas pueden ser clasificadas como *cacheable* o *no cacheable*.

Se empezará por definir qué quiere decir sistema **cliente servidor**, en este tipo de sistema el desarrollo del front-end y back-end se hacen por separado teniendo en cuenta la API. Esto facilita los cambios que se quieran realizar. Esta separación entre cliente y servidor es una separación de tipo lógico. Por tanto, REST es para servicios web sencillos.

El servidor es un proveedor de servicios que no gestiona los recursos del cliente, por lo que, el cliente es el que demanda el servicio y tiene que enviar toda la información necesaria al servidor. (de la Torre & González, 2008)

Es un **modelo de aplicación de capas** en el que se utiliza el patrón MVC, de esta forma se tiene el código mejor organizado y si en futuro se desea ampliar la funcionalidad de la herramienta web, será más sencillo. A continuación, se explican las tres capas a emplear:

1. Modelo: es la capa que trabaja con los datos, por consiguiente, contendrá los mecanismos para acceder, crear, eliminar y actualizar la información. Es decir, trabajará directamente con funciones que accederán a la base de datos.

2. Vista: contiene el código que necesita la herramienta web para mostrar al usuario la interfaz. En esta capa, la mayor parte se trabajará con los datos, es decir, las vistas solicitarán los datos a los modelos y en ellas se generará la salida.

3. Controlador: esta capa contiene toda la lógica de negocio, en dónde se aloja el código necesario para responder a las acciones que se solicitan en la aplicación.
   Esta capa es importante porque sirve de enlace entre las vistas y los modelos para implementar las diversas necesidades del desarrollo, su responsabilidad no es manipular directamente los datos, ni mostrar ningún tipo de salida.

En la Figura 1, se muestra cómo se realiza la comunicación y se ven claramente las tres capas.

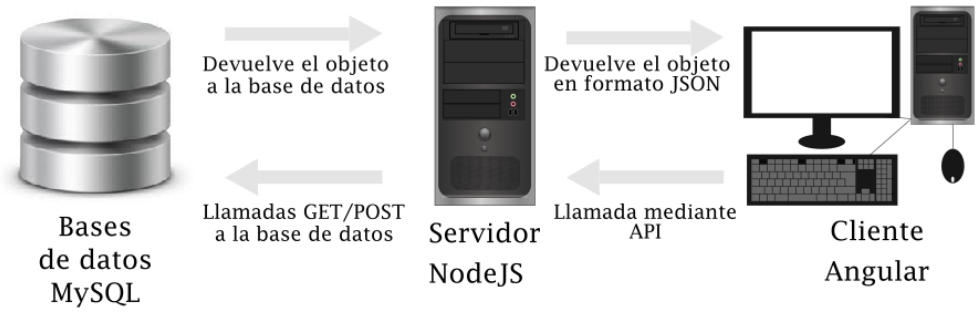


FIGURA 1: CAPAS DE LA COMUNICACIÓN

Peticiones sin estado o también conocido como S*tateless*: se denomina así, porque REST es un estilo de arquitectura para el desarrollo de aplicaciones web, que no guarda los estados de las peticiones, es decir, en cada petición que se realice al servidor, el cliente debe realizar la petición con toda la información necesaria para entenderla y ejecutarla. En conclusión, el servidor no guarda la información de la sesión o el estado entre dos peticiones. Por lo explicado y para solventar esta característica, se hace uso de tokens que serán generados por el servidor y proporcionados al cliente, para que así, el cliente pueda proporcionar esa información al momento de realizar la petición.

En la Figura 2, se muestra cómo se representa la comunicación al realizar el inicio de sesión. Las respuestas que proporcione el servidor pueden estar en formato XML, JSON, etc. En este caso se ha decidido utilizar JSON.

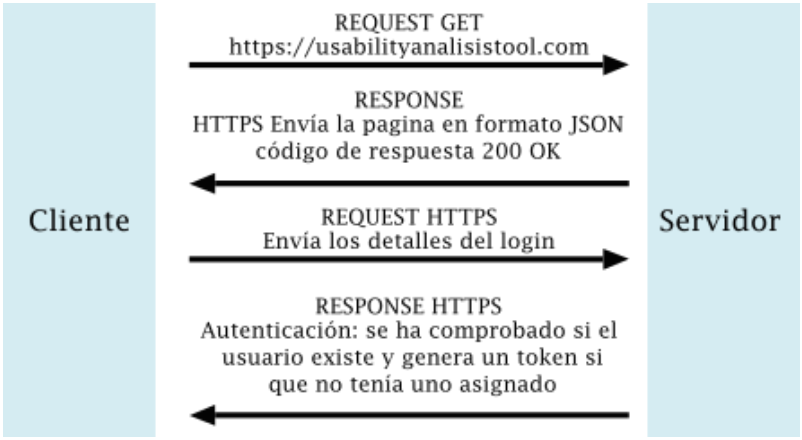


FIGURA 2: INICIO DE SESIÓN

Continuando con las reglas que tiene la arquitectura REST, se define lo que son las **interfaces uniformes**. Las interfaces deben identificar el comportamiento y la semántica sin mirar el cuerpo del mensaje, en la URL se debe declarar que recurso está siendo manipulado. Por ello se basan en mensajes estándar como:

- GET: Para consultar y leer recursos.
- POST: Para crear recursos.
- PUT: Para editar/modificar recursos.
- DELETE: Para eliminar recursos.

2.5 Herramientas

Las herramientas que se describen en este apartado, son necesarias para la elaboración de la documentación, análisis y desarrollo del proyecto.

Ordenador portátil: herramienta fundamental para la realización de este proyecto, en él se tendrán instaladas las herramientas descritas a continuación:

Angular: es un framework necesario para implementar la parte del front-end de la herramienta web. Facilita el desarrollo de aplicaciones web, en la que la navegación entre las diferentes secciones y páginas, además de la carga de datos se realiza de forma dinámica, debido a que realiza peticiones asíncronas al servidor API, todo esto sin necesidad de refrescar la página. El resultado tras utilizar Angular en el front-end es obtener una implementación de la vista, legible y fácil de desarrollar gracias a la reutilización de código. (Google ©, 2010-2018)

Cacoo: es una herramienta web, útil para el desarrollo de gráficos UML, casos de uso, diagramas de flujo, diagramas de secuencia, etc. Estos gráficos son necesarios para el apartado de análisis.

Dropbox: en este proyecto se utilizará como soporte de seguridad, para tener copias de seguridad de la herramienta web y de la documentación.

Express: *"es una infraestructura de aplicaciones web Node.js mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web y móviles".* (Fundación Node.js, s.f.). Node.js es un entorno de ejecución.

Inkscape: es una aplicación de escritorio, útil para realizar algunas de las figuras necesarias para este TFG.

JavaScript: es un lenguaje de programación que normalmente se utiliza para la creación de páginas web. Este lenguaje de programación no necesita ser compilado y se puede probar desde cualquier navegador.

MySQL: es un sistema de gestión de base de datos relacional de código libre, con un gran rendimiento en aplicaciones web.

Microsoft Office: herramienta de texto, útil para la redacción de esta memoria (Microsoft Word), y la presentación del TFG (Power Point).

Mozilla Firefox, Google Chrome: navegadores web de código libre, necesarios para realizar las pruebas de desarrollo. Disponibles de forma gratuita.

NodeJS: *"es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome. Node.js usa un modelo de operaciones E/S sin bloqueo y orientado a eventos, que lo hace liviano y eficiente."* (Node.js Foundation, s.f.). Este entorno es necesario para implementar el back-end de la herramienta web

Postman: es una aplicación de escritorio, que será de gran utilidad en este proyecto para realizar las pruebas de la implementación, sobre todo al principio que no se tiene ninguna interfaz, esto ayudará a la desarrolladora a realizar peticiones HTTP REST sin necesidad de tener desarrollado un cliente, así sabrá todo momento si su implementación va por buen camino.

Visual Studio Code: es un editor de código, multiplataforma que permite debugear y compilar el código que se va desarrollar, similar a Notepad++ o a Sublime Text.

Visual Paradigm: es un software de desarrollo que permite la realización de diagramas UML, en este caso se hace uso de él, para realizar los diagramas de secuencia.

TypeScript: es una extensión de JavaScript, que ofrece un sistema de implementación en base a módulos, clases, interfaces.

2.6 Alcance

En este apartado se dan conocer las funcionalidades o tareas que tiene el proyecto, se explican cada uno de los paquetes de trabajo que se han definido, indicando la duración estimada, descripción, las entradas y las salidas. En este caso es desarrollado por una sola persona por lo que no hace falta hacer mención quién ha realizado que funcionalidad.

2.6.1 Estructura de Descomposición del Trabajo (EDT)

Se ha decidido utilizar la forma de desarrollo por prototipos, esto permite ver el avance del mismo progresivamente. Este tipo de desarrollo permite a cada prototipo tener un poco de independencia. Es una forma de desarrollo a partir de requisitos poco claros o cambiantes, teniendo como producto final un sistema de calidad.

Al trabajar con prototipos se tiene la ventaja de mostrar un adelanto de cómo se va a visualizar la herramienta web. De esta forma el cliente, en este caso el tutor del proyecto, puede tener una idea de qué funcionalidades se van a desarrollar. Por otra parte, se tiene que tener en cuenta que cada apartado se puede modificar de forma independiente, esto será una ventaja o desventaja, dependiendo de que tanto haya que modificar, porque se puede dar el caso de rehacer el prototipo en su totalidad, lo que significaría una inversión de tiempo considerable.

En la Figura 3 se ven cuatro apartados, a continuación, se hace un pequeño resumen de cada apartado:

Documentación: en el que se tratan cuestiones relacionadas con la elaboración de esta memoria, además de la investigación sobre herramientas existentes o similares a Usability Analysis Tool.

Organización y aprendizaje: apartado que trata sobre las herramientas necesarias para desarrollar el proyecto, además del aprendizaje de la desarrolladora de dichas herramientas y nuevos lenguajes de programación.

Captura de requisitos: es básicamente lo relacionado con los casos de uso, modelo de dominio y la transformación de modelo de dominio a bases de datos.

Prototipos: este apartado engloba a los prototipos que se van a realizar. Se divide en siete sub apartados, cada uno de ellos hace referencia a una funcionalidad importante de la herramienta web. Cada prototipo a su vez, se va a dividir en análisis, diseño e implementación y pruebas. A continuación, se hace una breve descripción de cada uno de los prototipos:

- **Registro e inicio de sesión de usuario**: en este prototipo se desarrolla el registro e inicio de sesión del usuario, su funcionalidad es permitir el acceso a los usuarios que se hayan registrado previamente.

- **Creación y configuración del proyecto, tareas, cuestionarios y testers de forma manual**: este prototipo tiene como funcionalidad crear tareas, cuestionarios y testers, además de configurar el proyecto, que no es más que crear los nodos necesarios para la aplicación. Estos objetos se pueden crear de forma manual, es decir el usuario introduce los datos necesarios a través de la interfaz. A su funcionalidad hay que añadirle, que también se puede actualizar y eliminar los objetos que se hayan podido crear.
  Se llama objetos a las tareas, proyectos, cuestionarios, testers, etc.

- **Crear tareas realizadas y responder cuestionario de forma manual**: con este prototipo se pueden crear tareas realizadas y responder cuestionario, ambos son creados de forma manual, ya que el usuario introduce los datos necesarios a través de la interfaz, también tiene como funcionalidad, actualizar y eliminar los objetos que se hayan podido crear.

- **Cargar datos desde fichero**: este prototipo permite al usuario configurar el proyecto, crear tareas, cuestionarios y testers, tareas realizadas y responder cuestionario desde ficheros CSV o JSON, creándose los objetos automáticamente, después de leer el fichero.

- **Resultados en forma de tabla**: este prototipo permite generar tablas con resultados para cada objetivo básico de la usabilidad, y que son calculados a partir de los datos que se han creado con anterioridad con los prototipos de "Crear tareas realizadas y responder cuestionario de forma manual" y "Cargar datos desde fichero".

- **Resultados gráficos**: con este prototipo se obtienen los grafos de navegación de cada tarea realizada. Estos grafos se corresponden con el objetivo de la eficiencia.

- **Informe final con tablas y gráficos**: este prototipo es una fusión de "Resultados en forma de tabla" y "Resultados gráficos". Es el que hace posible ver los resultados de los tres objetivos básicos de la usabilidad incluyendo los grafos de navegación.

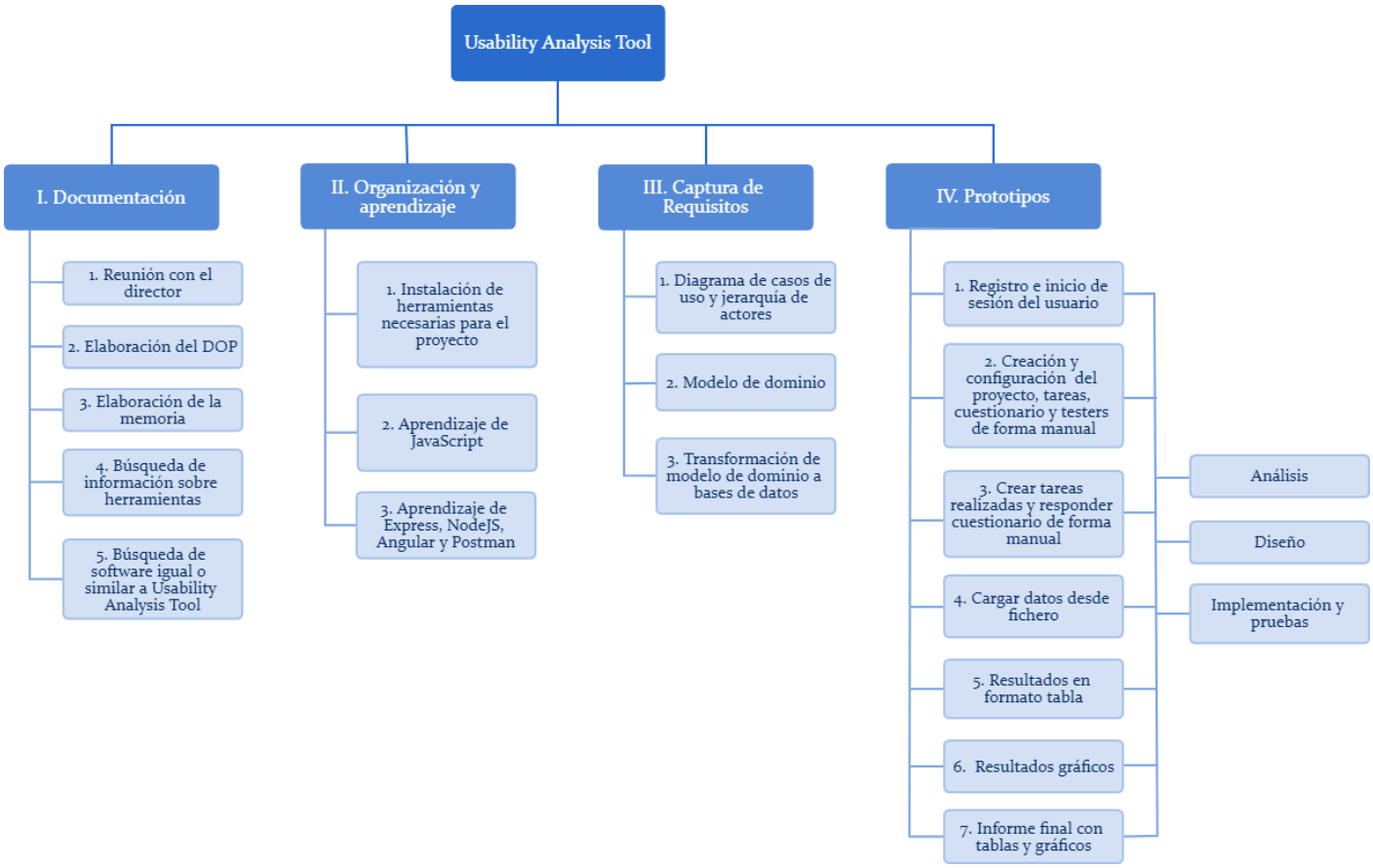


FIGURA 3: ESTRUCTURA DE DESCOMPOSICIÓN DEL TRABAJO

Descripción de los paquetes de trabajo

Los paquetes de trabajo son los que se han mostrado en la Figura 3, en ella se pueden ver cuatro apartados en los que se encuentran los paquetes de trabajo que se describen de forma detallada a continuación:

I. Documentación

En este apartado se hace referencia a la parte previa al desarrollo de la herramienta web, como por ejemplo la reunión con el director del proyecto, la elaboración del documento de objetivos, etc. En las siguientes tablas se indican la duración estimada y la descripción de cada tarea.

TABLA 1: REUNIÓN CON EL DIRECTOR

I.1. Reunión con el director	
Duración estimada	3 horas
Descripción	Reunión con el director para comentar lo se quiere obtener como resultado de este proyecto, hablar sobre los requerimientos necesarios y el progreso a lo largo del desarrollo del proyecto.

TABLA 2: ELABORACIÓN DEL DOP

I.2. Elaboración del documento de objetivos del proyecto (DOP)	
Duración estimada	40 horas
Descripción	Es el documento donde se encuentran los objetivos, el planteamiento y las herramientas necesarias para la realización del proyecto. Así como también el EDT. Básicamente es la primera parte de esta memoria.

TABLA 3: ELABORACIÓN DE LA MEMORIA

I.3. Elaboración de la memoria	
Duración estimada	150 horas
Descripción	En concreto se hace referencia a este documento, en él se describe la forma como se ha planteado realizar este proyecto y el coste económico que conlleva, además del análisis y desarrollo del mismo.

TABLA 4: BÚSQUEDA DE INFORMACIÓN SOBRE HERRAMIENTAS

I.4. Búsqueda de información sobre herramientas necesarias para la elaboración del proyecto	
Duración estimada	12 horas
Descripción	Búsqueda de herramientas que se utilizan en desarrollo web, teniendo en cuenta que se quiere aprender nuevas tecnologías y lenguajes de programación. De todas ellas elegir las que mejor se adapten y sean útiles.

TABLA 5: BÚSQUEDA DE SOFTWARE

I.5. Búsqueda de software igual o similar a Usability Analysis Tool	
Duración estimada	16 horas
Descripción	Investigación acerca de otras aplicaciones iguales o similares a este proyecto. En caso de haber iguales, tener en cuenta lo que ofrecen, que les diferencia de Usability Analysis Tool, además de analizar si aporta algo a este proyecto, y de los puntos débiles que pueda tener y trabajar en ello para no caer en lo mismo.

II. Organización y aprendizaje

En organización y aprendizaje se definen las herramientas y preparación del entorno de desarrollo del proyecto, así como también el tiempo que ha dedicado la desarrolladora a aprender las distintas tecnologías necesarias para este proyecto. En las siguientes tablas se indican la duración estimada y la descripción de cada tarea.

TABLA 6: INSTALACIÓN DE HERRAMIENTAS

II.1. Instalación de herramientas	
Duración estimada	8 horas
Descripción	Como el propio nombre lo dice, en este apartado se instalarán las herramientas resultantes de la investigación realizada.

TABLA 7: APRENDIZAJE DE JAVASCRIPT

II.2. Aprendizaje de JavaScript	
Duración estimada	30 horas
Descripción	El objetivo de esta tarea es ampliar el conocimiento de la desarrolladora en cuanto a JavaScript, mediante tutoriales, artículos o investigar preguntando a personas que tengan experiencia en el con el manejo de Java Script.

TABLA 8: APRENDIZAJE DE EXPRESS, NODEJS, ANGULARJS Y POSTMAN

II.3. Aprendizaje de Express, NodeJS, Angular y Postman	
Duración estimada	20 horas
Descripción	Aprendizaje de cómo realizar el desarrollo del proyecto con estas herramientas, qué estructura utilizar, mediante tutoriales o la experiencia de otras personas.

III. Captura de requisitos

Es importante definir el análisis antes de empezar con el desarrollo. Por ello en este apartado se tiene en cuenta el tiempo que se dedica a la realización de casos de uso, modelo de dominio, y la respectiva transformación de modelo de dominio a bases de datos. En las siguientes tablas se indican la duración estimada y la descripción de cada tarea.

TABLA 9: DIAGRAMA DE CASOS DE USO Y JERARQUÍA DE ACTORES

III.1. Diagrama de casos de uso y jerarquía de actores	
Duración estimada	10 horas
Descripción	Elaboración de los casos de uso para obtener las diferentes acciones que el usuario tendrá disponible en el proyecto. El rol que representa el usuario y la jerarquía de actores. Para todo ello es necesario capturar los requisitos funcionales y no funcionales. Por esto se tienen en cuenta las acciones que el usuario espera encontrar en la herramienta web, es decir, todo lo relativo a las interfaces, como pueden ser: registro e inicio de sesión, creación de proyecto y configuración, creación de tareas y cuestionarios, mostrar los resultados en forma de tabla y gráficos.

TABLA 10: MODELO DE DOMINIO

III.2. Modelo de dominio	
Duración estimada	6 horas
Descripción	A partir de la información obtenida de la captura de requisitos obtenidos del cliente en este caso el tutor. Se trata en la mayor parte de lo posible determinar qué datos se quieren almacenar y las relaciones que se producen entre ellos.

TABLA 11: TRANSFORMACIÓN DE MODELO DE DOMINIO A BD

III.3. Transformación de modelo de dominio a base de datos	
Duración estimada	6 horas
Descripción	Conversión del modelo de dominio a base de datos, con sus respectivas relaciones.

IV. Prototipos

Los prototipos representan las diferentes funcionalidades de la herramienta web e incluirán el back-end y front-end, además del análisis, el diseño, la implementación, las pruebas y documentación para cada uno de ellos. En las siguientes tablas se indican la duración estimada y la descripción de cada prototipo.

TABLA 12: PROTOTIPO DE REGISTRO E INICIO DE SESIÓN

IV.1. Registro e inicio de sesión de usuario	
Duración estimada	30 horas
Descripción	Elaboración de la funcionalidad registro e inicio de sesión. Controlando también el tiempo de la sesión una vez iniciada la sesión. Solo tienen acceso al sistema los usuarios que estén registrados e inicien sesión.

TABLA 13: PROTOTIPO DE CREACIÓN Y CONFIGURACIÓN DE PROYECTO, TAREAS Y TESTERS

IV.2. Creación y configuración del proyecto, tareas, cuestionario y testers de forma manual	
Duración estimada	50 horas
Descripción	Este prototipo permite crear un proyecto, configurarlo, crear las tareas, cuestionario y testers. Se centra sobre todo en la configuración del proyecto. Es importante añadir que estos objetos son creados de forma manual, objeto a objeto, y que no sólo se centra en crear sino también en actualizar y eliminar los diferentes datos que hayan sido creados.

TABLA 14: PROTOTIPO DE CREAR TAREAS REALIZADAS Y RESPONDER CUESTIONARIO

IV.3. Crear tareas realizadas y responder cuestionario de forma manual	
Duración estimada	20 horas
Descripción	Con este prototipo el usuario puede crear las tareas realizadas por los testers, así como también las preguntas que haya respondido del cuestionario, una a una, es decir de forma manual.

TABLA 15: PROTOTIPO DE CARGAR DATOS DESDE FICHERO

IV.4. Cargar datos desde fichero	
Duración estimada	50 horas
Descripción	Este prototipo es una versión más automática que el prototipo anterior ya que el usuario puede cargar un fichero de tipo CSV o JSON que contendrá todos los datos referentes a tareas, cuestionario, testers, tareas realizadas y las respuestas del cuestionario evitando el tener que crear cada uno de ellos.

TABLA 16: PROTOTIPO DE RESULTADOS EN FORMATO TABLA

IV.5. Resultados en formato tabla	
Duración estimada	30 horas
Descripción	Este prototipo realiza todos los cálculos necesarios para poder dar el resultado en formato tabla de cada uno de los objetivos de la usabilidad. Los cálculos se realizan utilizando formulas estadísticas.

TABLA 17: PROTOTIPO DE RESULTADOS GRÁFICOS

IV.6. Resultados gráficos	
Duración estimada	50 horas
Descripción	Este prototipo tiene como funcionalidad crear un grafo de navegación, en donde los nodos son las páginas que hay que visitar en cada tarea, y los arcos el recorrido realizado por los testers.

TABLA 18: PROTOTIPO DE INFORME FINAL

IV.7. Informe final con tablas y gráficos	
Duración estimada	30 horas
Descripción	En este prototipo muestra los resultados en formato de tabla y resultados gráficos. De esta forma el usuario obtiene un informe general de los tres objetivos básicos de la usabilidad.

2.7 Planificación temporal (GANTT)

Como ya se conoce el tiempo que se va a realizar en cada tarea es necesario decidir la distribución del tiempo en función a las tareas que se han visto en el EDT (Figura 3).

En cuanto a las tareas de los prototipos, en el diagrama de Gantt se tiene en cuenta el tiempo para la realización del back-end y del front-end. Para realizar este trabajo se establece que la carga de trabajo es de 25 horas a semanales teniendo en cuenta solo los días laborables.

En la Figura 4 se puede ver la línea del tiempo de los paquetes de trabajo de: documentación, organización y aprendizaje, captura de requisitos y prototipos. Dentro de cada una de ellas sus respectivas tareas. La visualización esta por semanas y se puede ver también que se ha empezado la primera semana de enero y tiene fecha prevista de finalización la semana 20, es decir a mediados de mayo.

También se puede ver que la tarea de "Elaboración de la memoria" se realiza de forma paralela a algunas de las tareas del proyecto, esto se debe a que se va realizando a la vez que las demás tareas.

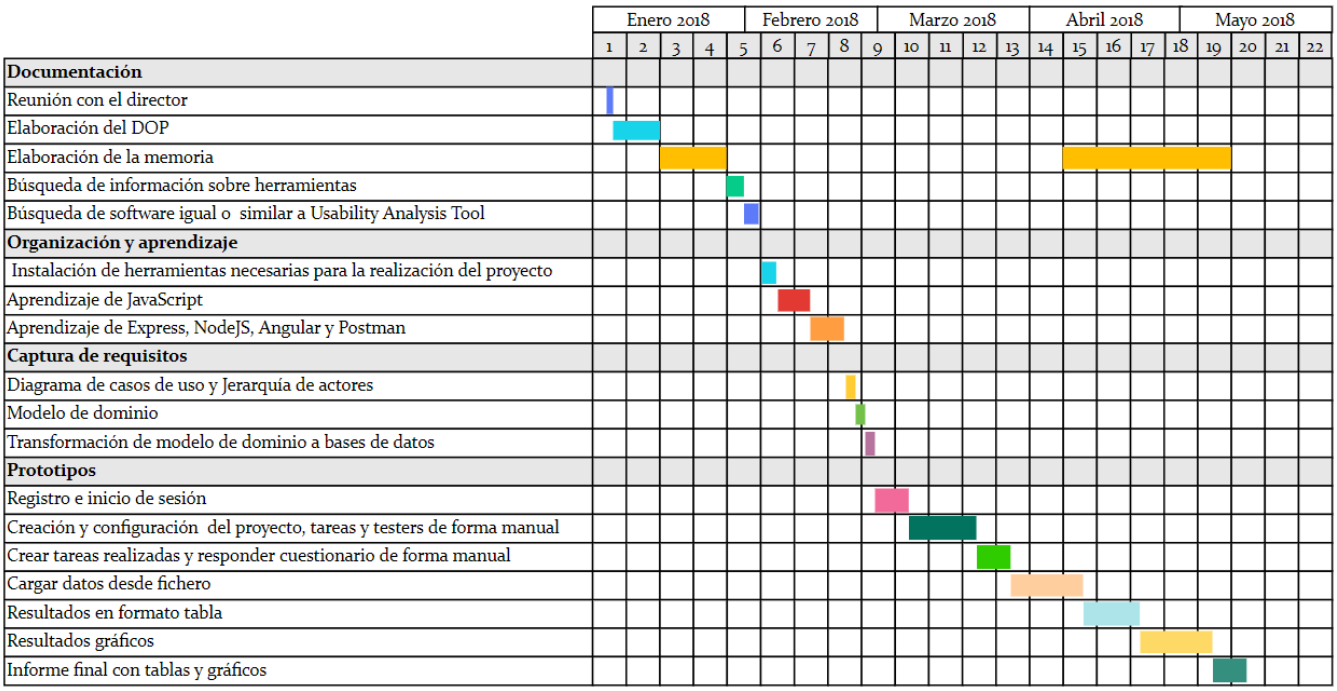


FIGURA 4: DIAGRAMA DE GANTT

En vista de no apreciarse las tareas de forma detallada, se da como solución la Tabla 19, en la que se agrupan por horas, y se puede ver la duración total de la elaboración de este proyecto.

TABLA 19: DURACIÓN TOTAL TAREAS

TAREA	DURACIÓN (horas)
Documentación	
Reunión con el director	3
Elaboración del DOP	40
Elaboración de la memoria	150
Búsqueda de información sobre herramientas	12
Búsqueda de software igual o similar a Usability Analysis Tool	16
Organización y aprendizaje	
Instalación de herramientas necesarias para la realización del proyecto	8
Aprendizaje de JavaScript	30
Aprendizaje de Express, NodeJS, Angular y Postman	20
Captura de requisitos	
Diagrama de casos de uso y Jerarquía de actores	10
Modelo de dominio	6
Transformación de modelo de dominio a bases de datos	6
Prototipos	
Registro e inicio de sesión	30
Creación y configuración del proyecto, tareas y testers de forma manual	50
Crear tareas realizadas y responder cuestionario de forma manual	20
Cargar datos desde fichero	50
Resultados en formato tabla	30
Resultados gráficos	50
Informe final con tablas y gráficos	30
Horas totales	561

2.8 Evaluación económica

Como en la gran mayoría de proyectos de desarrollo existen una serie de costes, en este apartado se realiza el cálculo monetario del valor del proyecto. Para ello se tienen en cuenta los meses invertidos en su realización, mano de obra, además, de las herramientas que son necesarias.

En este proyecto se diferencian dos tipos de costes, los costes directos o variables que tienen que ver directamente con el proyecto, por ejemplo: hardware, software, mano de obra, etc. Y los costes indirectos o fijos que son, por ejemplo, la luz, fotocopias, material de oficina, etc.

2.8.1 Gastos directos o variables

En esta sección se identifican los materiales tangibles e intangibles, como, por ejemplo, la mano de obra directa. Es decir, todo lo que está relacionado directamente con la elaboración del proyecto.

Mano de obra

La mano de obra es un coste directo, por tanto, se tendrán en cuenta las horas que se han empleado en la realización de este proyecto. Según el diagrama Gantt que se ha elaborado, se han necesitado 561 horas en total que vienen a ser unos cinco meses y medio aproximadamente, teniendo en cuenta que se trabaja sólo los días laborables.

Para calcular la mano de obra se tiene en cuenta el sueldo base de un *"Licenciado y titulados 2.º y 3.er ciclo universitario y Analista"* según el BOE es de 1.687,02€/mes (Agencia Estatal Boletín Oficial del Estado, s.f.). Por lo tanto, tras haber calculado 561 horas de mano de obra, el total del gasto es aproximadamente 5.5767,17 €.

Hardware

- Ordenador portátil Sony Vaio, se compró en 2013 por un valor de 890€ con una vida útil estimada de 6 años y un uso estimado hasta día de hoy del 80%. Por lo que la amortización y el gasto correspondiente a la realización del proyecto se calcula de la siguiente forma:

  - Amortización anual: 895€/6 años = 150€
  - Gastos del portátil: [(150€ x 22,5 semanas de duración del proyecto) x 80%]/52 semanas = 52€ aprox.

Software

- Licencia de Windows 10: incluido en el precio del ordenador.
- Licencia Office 2016: 69€.
- Toms Planner para Diagrama Gantt: gratuito.
- Sublime Text: gratuito.
- Visual Code: gratuito.
- Postman: gratuito.
- NodeJS: gratuito.
- Angular: gratuito.
- Cacoo: gratuito.
- Dropbox: gratuito.
- Inkscape: gratuito.

2.8.2 Gastos indirectos

Los gastos indirectos son aquellos que gastos necesarios para el mantenimiento y se producen durante el proceso de elaboración del proyecto, como, por ejemplo: luz, agua, internet, etc. Se considera que estos gastos suponen un 5% de los gastos directos del proyecto.

En la Tabla 20 se pueden ver los gastos directos e indirectos, así como el total de ambos.

TABLA 20: TOTAL GASTOS

Descripción	Coste
Gastos directos	
Mano de obra	5.5767,17 €
Hardware	52,00 €
Software	69,00 €
Gastos indirectos	
Gastos varios	282,17 €
Gastos totales	5.6170,34 €

2.8.3 Retorno de inversión

Dado que es un proyecto de fin de carrera relacionado con la investigación y que por lo tanto no se considera la venta de licencias, ingresos por incluir publicidad o similares, no se contempla el beneficio económico del mismo.

2.9 Gestión de riesgos

Para desarrollar el proyecto con normalidad y ante posibles pérdidas ya sean de información, software, hardware o cualquier otra eventualidad, es necesario tener un plan de contingencia para hacer frente a diversos problemas. Por estos motivos se tiene en cuenta la probabilidad con la que puede ocurrir, en la Tabla 21 se muestran tres valores para medir la probabilidad.

TABLA 21: PROBABILIDAD

Probabilidad	Valor cuantitativo (%)
Improbable	0 – 25%
Poco probable	26 – 65%
Muy probable	66 -100%

Otro aspecto importante a tener en cuenta es el impacto que puede llegar a tener en el caso de producirse, puede haber riesgos que no sean de gran importancia y otros que sí lo sean. Hay que tener en cuenta que los riesgos pueden retrasar el proyecto, entonces de la misma forma que se ha hecho con la probabilidad, en la Tabla 22 se muestran cuatro valores para medir el impacto.

TABLA 22: IMPACTO

Impacto	Valor cuantitativo (días)
Muy alto	Más de 15
Alto	8-15
Medio	3-7
Bajo	1-2
Muy bajo	Menos de 1

Con los criterios que se han considerado importantes para medir la probabilidad y el impacto del daño que puede recaer sobre el proyecto, se va a tener en cuenta la prevención y un plan de contingencia acorde al riesgo a tratar. Estos riesgos se han dividido en tres apartados que se pueden ven en la Tabla 23.

TABLA 23: RIESGOS

Riesgos de análisis y desarrollo
2.9.1 Software y hardware
Error en el cálculo del tiempo de formación en alguna de las tecnologías o lenguajes de programación a utilizar
Fallo en el disco duro
Fallo de la conexión de internet
Desaparición de librería(s) que se han utilizado
Infección por virus
2.9.2 Bases de datos y su seguridad
Sobrecargar el servidor con demasiadas peticiones
Inyección SQL (código malicioso)
Gestión de sesiones
Almacenamiento de datos inseguro
Riesgos generales
2.9.3 Causas personales
Enfermedad
Falta de tiempo por empleo o prácticas

Una vez identificados los riesgos de forma general, se detalla cada una indicando la probabilidad, el impacto, además de la prevención y el plan de contingencia para solucionar las diferentes eventualidades.

En la Tabla 24 se analizan los riesgos de análisis y desarrollo correspondientes a software y hardware.

TABLA 24: RIESGOS DE ANÁLISIS Y DESARROLLO

2.9.1 Software y Hardware	
2.9.1.1 Error en el cálculo del tiempo de formación necesaria en alguna de las tecnologías o lenguajes de programación a utilizar. Es un riesgo muy importante ya que sin los conocimientos necesarios la desarrolladora tiene que invertir más horas, que una persona que ya domina las herramientas necesarias para desarrollar este proyecto.	
Probabilidad	Muy probable, debido a que las tecnologías aplicadas en este proyecto son nuevas para la desarrolladora.
Impacto	Muy alto, porque la desarrolladora tiene que aprender tecnologías nuevas.
Prevención	Formarse por cuenta propia y buscar información acerca de las herramientas a utilizar antes de empezar y no abandonar el proyecto durante tiempo para no olvidar lo aprendido
Plan de contingencia	En caso de que este riesgo se produzca, se buscará ayuda externa ya sea en internet o con personas relacionadas a este tema en concreto o en todo caso se buscará apoyo en el tutor de prácticas.

2.9.1.2 Fallo en el disco duro

Se pueden producir diferentes fallos en el disco, entre ellos los más comunes son:

- Fallos en los sistemas de archivos del disco duro o error humano, se pueden producir por virus o malware, borrado de datos, formateos, etc.
- Fallos producidos por cambios en la temperatura o humedad, subidas y bajadas de tensión.
- Fallos producidos por golpes, fallos de fabricación, desgaste o corrosión, etc.

Estos son sólo algunos ejemplos de fallos que se pueden producir a lo largo del desarrollo del proyecto.

Probabilidad	Poco probable, debido a que el ordenador sólo se utiliza para la realización de este proyecto.
Impacto	Muy Alto, lo cual causa la perdida de todos los datos del ordenador.
Prevención	Cuidar el equipo correctamente. Mantenerlo en condiciones ambientales adecuados. Apagar el equipo cuando no se utilice. Evitar cortes de electricidad. Realizar copias de seguridad cada poco tiempo, una semana como máximo en un disco externo o en Dropbox.
Plan de contingencia	La primera solución es comprar otro disco duro y cambiarlo, si no se sabe cómo hacerlo acudir a un técnico, y cuando se tenga operativo el ordenador (no más de una semana) restaurar la copia de seguridad más reciente que se tenga. Si pasado una semana o un tiempo mayor el problema no está solucionado se optará por comprar un ordenador y restaurar la copia de seguridad más reciente. Durante esa semana que no se puede desarrollar nada relacionado a implementación, se empleará el tiempo de espera en mejorar y seguir desarrollando la documentación del TFG en un ordenador de la universidad.

2.9.1.3 Fallo de la conexión de internet

Se puede dar el caso en el que se pierda la conexión a internet y esto ocasione un poco de retraso, ya que dificultaría el buscar información necesaria para la elaboración del proyecto

Probabilidad	Improbable
Impacto	Bajo, si bien es cierto es una herramienta web, la parte de desarrollo se puede implementar sin necesidad de estar conectado a internet y al ser el impacto bajo, es un problema que no duraría más de dos días.
Prevención	Utilizar internet por cable que nos asegura una mayor velocidad al momento de trabajar. Dar un buen mantenimiento al router y tenerlo en un lugar seguro. Tener contratada una empresa de telefonía fiable que sea capaz de resolver el inconveniente en poco tiempo.
Plan de contingencia	En este caso podemos trasladar nuestro entorno de trabajo a una biblioteca pública con WIFI o a la propia universidad que dispone de internet para sus alumnos.

2.9.1.4 Desaparición de librería(s) que se ha(n) utilizado Se puede dar el caso de que las librerías que están utilizando para este proyecto sean eliminadas de los repositorios o hayan sido corrompidos.	
Probabilidad	Improbable
Impacto	Muy alto, debido a que una parte importante del proyecto hace uso de esa librería
Prevención	En primer lugar y si es posible descargarse la librería a utilizar. Realizar una buena investigación de si esa librería puede desaparecer o no.
Plan de contingencia	En caso de haber descargado la librería utilizarla en local. De lo contrario modificar el código con todo lo que hacía referencia a la librería en cuestión.
2.9.1.5 Infección por virus Ningún ordenador está libre de infectarse por virus a través de internet o por utilizar algún dispositivo infectado en el ordenador en el que se tiene toda la documentación y la implementación por ello es importante tener un plan de prevención y de contingencia.	
Probabilidad	Improbable
Impacto	Bajo
Prevención	Lo primero es realizar copias de seguridad cada poco tiempo, el tiempo máximo entre una copia y otra, es de una semana. Evitar utilizar software desconocidos o poco fiables. Tener actualizado el ordenador, al igual que el antivirus (Avast, es gratuito y es el que se tiene instalado). No tener conectado el disco duro externo en el que se hacen las copias de seguridad al ordenador todo el tiempo ya que el virus podría dañar el disco.
Plan de contingencia	Realizar un análisis profundo al ordenador y eliminar la amenaza. Si el peligro es inminente y se ha perdido datos, restaurar la copia de seguridad más reciente.
2.9.1.6 Planificación incorrecta del tiempo Al ser el primer trabajo que la desarrolladora realiza en solitario, es normal que no se calcule el tiempo correctamente.	
Probabilidad	Muy probable
Impacto	Medio
Prevención	Ceñirse a lo aprendido a lo largo de la carrera y calcular el tiempo lo más preciso posible, buscar información sobre los temas nuevos.
Plan de contingencia	Volver a realizar la planificación del tiempo esta vez teniendo en cuenta la experiencia que ya se ha ganado.

2.9.2 Bases de datos y seguridad

2.9.2.1 Sobrecargar el servidor con demasiadas peticiones

Este riesgo se refiere a que la herramienta web pueda o no soportar una cantidad elevada de conexiones, es decir, haya recibido muchas más visitas de lo esperado. Esto conlleva a un consumo de muchos recursos por parte del servidor.

Probabilidad	Poco probable
Impacto	Medio
Prevención	Implementar funciones que sean de ayuda para mejorar la carga del sistema, siendo capaz de soportar una gran cantidad de usuarios que realizan peticiones al servidor.
Plan de contingencia	Un buen plan de contingencia en este caso es limitar el número de conexiones en un mismo instante de tiempo.

2.9.2.2 Inyección SQL (código malicioso)

Se puede dar el caso en el que personas maliciosas, introduzcan código en la herramienta web para obtener datos y ejecutar operaciones en la base de datos. Esto se debe a que no se han implementado funciones que filtran los datos que los usuarios envían en las peticiones.

Probabilidad	Poco probable
Impacto	Medio
Prevención	Implementar funciones que controlen la introducción de posible código malicioso (caracteres que se suelen utilizar en la base de datos)
Plan de contingencia	Comprobar que datos se han visto afectados. Restaurar la copia de seguridad y revisar las funciones que se han implementado, para encontrar las posibles debilidades y mejorar el código.

2.9.2.3 Gestión de sesiones

Es importante tener en cuenta el tiempo de duración de la sesión. Para evitar posibles robos de cuentas y de datos.

Probabilidad	Poco probable
Impacto	Medio
Prevención	Se determinará un tiempo de sesión y se implementará una función que controle si la sesión ha expirado o no. En el caso de haber terminado el tiempo de conexión, se solicita el inicio de sesión nuevamente.
Plan de contingencia	En caso de ocurrir este hecho se comprobará los datos del usuario y en caso de haber sido corrompidos se restaurará la copia de seguridad.

2.9.2.4 Almacenamiento de datos inseguro Es muy importante garantizar a los usuarios que las contraseñas se encuentran protegidas.	
Probabilidad	Poco probable
Impacto	Medio
Prevención	Cifrar las contraseñas antes de introducirlas a la base de datos. Así garantizamos que las contraseñas no sean legibles, ya que muchas personas utilizan la misma contraseña para diferentes aplicaciones. Realizar copias de seguridad periódicas.
Plan de contingencia	Si este riesgo ocurre se restaurará la copia de seguridad más reciente.

En general en todos los riesgos que estén relacionados con el análisis y el desarrollo, se realizarán copias de seguridad periódicas para garantizar la protección de los datos.

En la Tabla 25 se ven los riesgos generales que están relacionados directamente con la desarrolladora.

TABLA 25: RIESGOS GENERALES

2.9.3 Causas personales	
2.9.3.1 Enfermedad. Puede ocurrir que en cualquier momento y por cualquier circunstancia la desarrolladora tenga algún accidente o alguna enfermedad.	
Probabilidad	Poco probable
Impacto	Muy bajo
Prevención	Mantener un estilo de vida saludable.
Plan de contingencia	Acudir al médico lo antes posible y cumplir con sus recomendaciones.
2.9.3.2 Falta de tiempo por empleo o prácticas Es muy común que los alumnos de cuarto año realicen prácticas en empresa, y que se dé el caso que sean contratados, talvez tengan que trabajar más horas, o que trabajen y estudien a la vez, por lo que en cualquiera de los casos realizar el proyecto es un poco complicado.	
Probabilidad	Probable
Impacto	Alto
Prevención	Compaginar las horas de las prácticas con las del trabajo, y así permitir realizar las dos cosas a la vez.
Plan de contingencia	Priorizar las situaciones, y dedicar el máximo tiempo posible a terminar el TFG.

3. ANTECEDENTES

Antes de empezar a desarrollar este proyecto, se ha realizado una investigación sobre si existen aplicaciones iguales o similares a Usability Analysis Tool. Se han encontrado diversas herramientas para medir la usabilidad con herramientas online gratuitas y de pago. De la misma forma que se ha realizado la investigación antes mencionada, se ha investigado sobre qué herramientas utilizar para el desarrollo de esta aplicación.

3.1 Comparación de herramientas similares al proyecto

De la investigación realizada se ha obtenido como resultado que hay empresas que se dedican a realizar estudios de usabilidad, aunque no de forma gratuita. Por otro lado, lo que más abunda en internet son las páginas que realizan otras formas de medir la usabilidad, y no tanto como medir la usabilidad, sino más bien que tanto éxito tiene una página web o cuales son las zonas más visitadas de una página determinada.

En este apartado se va a mencionar a algunas de las páginas que parecen de mayor interés y muy similares a la idea principal que se tiene para Usability Analysis Tool, que es la de ayudar a medir los objetivos la usabilidad.

Google Analytics [1]

Google analytics no se dedica a medir la usabilidad, sino más bien a medir la cantidad de visitas de un sitio web, esta herramienta se puede utilizar para recolectar información sobre usabilidad y experiencia de usuarios. Y con estos datos de uso se pueden indicar los diversos problemas a los que el usuario se enfrenta al realizar una determinada acción.

También permite analizar búsquedas internas, además de los caminos que recorren los usuarios en la página web a evaluar. Y para hacer más sencillo el análisis de la usabilidad, sugieren instalar la extensión “Page Analytics” que también es de Google, y se encarga de llevar las métricas de Google Analytics a la página que se está evaluando.

Ventaja: que tiene una versión estándar de forma gratuita, y sobre todo es una herramienta que entrega información detallada de cada elemento de la página. Además, permite evaluar una sesión completa.

Desventaja: debido a que se obtiene como resultado una gran cantidad de información, puede resultar confusa. Para sacar partido de esta herramienta se tiene que definir bien los objetivos que se quieren conseguir, ya que se puede invertir más tiempo del necesario y obtener pocos resultados. (Google, s.f.)

[1] https://www.google.com/intl/es_ALL/analytics/features/analysis-tools.html

ClickHeat y CrazyEgg [2] [3]

Estas dos herramientas que se basan en clicks que hace el usuario en la página que se quiere evaluar, para ello CrazyEgg necesita que se inyecte código JavaScript en las páginas que se quieren evaluar, mientras que en el caso de ClickHeat la inyección se hace en el servidor. En ambos casos proporcionan el código a inyectar.

Ambos realizan un seguimiento de los clicks que hace el usuario en cada elemento de la página, y con ello genera mapas de calor con los resultados. Con esos resultados se pueden identificar elementos que son distractores, áreas que necesitan ser destacadas o lugares que carecen de enlaces donde los usuarios hacen click.

En el caso de CrazyEgg muestra los términos más buscados y los sistemas operativos más usados.

Ventaja: CrazyEdgg tiene la ventaja de poder exportar los datos obtenidos en formato Excel y así analizar los datos. Por otro lado, se puede considerar una ventaja o desventaja el hecho pagar una suscripción mensual, que no es muy costosa.

Desventaja: sólo sirve para monitorizar una página y no la sesión completa, además del hecho de que la persona a realizar el análisis tiene que saber algo de programación, para saber dónde poner el código JavaScript y por si surge alguna eventualidad. (CrazyEgg, s.f.)

Por otro lado, ClickHeat tiene la ventaja de ser OpenSource, además de no aumentar la carga del servidor.

Desventaja: al igual que en la anterior herramienta es necesario tener conocimientos de programación. (DugWood, s.f.)

Checkealos [4]

Es una empresa con sede en Sevilla, que tiene diferentes sedes a nivel mundial, y se dedica a realizar pruebas de usabilidad. Esta empresa se asemeja un poco al funcionamiento que se le quiere dar a Usability Analysis Tool, en cuanto al objetivo de satisfacción, ya que el usuario puede crear proyectos y tareas.

Estas tareas consisten en que el usuario cree un proyecto con datos como la URL de la página que se quiere evaluar, y a continuación elegir el rango de edades de los testers, además de si se quiere que sean hombres o mujeres, o los dos, también puede elegir cuantos testers quiere que realicen el cuestionario.

El cuestionario son preguntas que crea el usuario y que luego son resueltos por testers que trabajan con Checkealos. Una vez contestadas las preguntas, los evalúa un experto, hay que tener en cuenta que antes de responder los cuestionarios, la empresa graba la interacción de los testers con la web que se quiere evaluar. El experto determina los puntos débiles del producto, web, plataforma o tienda.

Ofrecen los siguientes resultados:

---

[2] https://www.crazyegg.com/
[3] https://www.dugwood.com/clickheat/index.html
[4] https://www.checkealos.com/en/

- *"Interacción de los usuarios finales con las Aplicaciones Webs de la competencia y la tuya propia + Grabación de Voz con sus comentarios: El formato es Video con grabación de la Pantalla y Voz del usuario. En estos videos se muestra como el usuario navega por las webs de la competencia y la tuya propia, realizando y comentando las tareas previamente definidas por ti".*
- *"Análisis elaborado por nuestros especialistas en UX* (Experiencia de usuario) *sobre cada uno de las interacciones de los usuarios finales y explicación de sus acciones y emociones".*
- *"Informe completo de nuestros especialistas en UX a través de su análisis, las recomendaciones y el resumen global".* (Checkealos.com, s.f.)

Ventajas: la experiencia de un experto que analiza la aplicación web comparando con webs similares, lo que hace que el usuario no invierta el tiempo en realizar el estudio, además de poder escoger ciertas características que quiere de los testers.

Desventajas: hay que pagar por este servicio, y esperar siete días para obtener los resultados. El contacto que se tiene con el experto es vía internet, lo que hace que no haya una interacción cercana con el usuario.

Distintiva [5]

Es una empresa del País Vasco que realiza análisis de usabilidad, además de otros servicios. Se describen como especialistas en analizar la facilidad de uso y satisfacción de los sitios web. Realizan test de usuarios y análisis heurísticos personalizados.

Empiezan con una planificación inicial realizando un esquema de objetivos, en el que definen el problema, los objetivos esenciales y el planteamiento de la investigación. Para continuar con tests comparativos que incluyen una inspección de diseño inicial y el checkList Heurístico, para continuar con el análisis Benchmarking que consiste en un proceso continuo que toma como referencia productos, servicios o procesos de trabajo de empresas líderes para compararlos con la web en cuestión, y posteriormente sugerir mejoras.

También realizan test de usabilidad y análisis visual, que consiste en una inspección de diseño en la que psicólogos y expertos examinan la interfaz de la web, para comprobar si se adecua al cumplimiento de los objetivos del sitio web, como pueden ser el uso eficiente del color, la estructura, textos, tipografías, mecanismos de navegación, etc.

Por otro lado, los tests de usabilidad son pruebas pasivas e interactivas de observación, memoria y reconocimiento. (Distintiva S.Coop., s.f.)

Ventaja: la empresa cuenta con gran experiencia en análisis de usabilidad, e importante cartera de clientes.

Desventaja: no es un servicio gratuito.

[5] http://www.distintiva.com/consultoria.php#usabilidad

UX Check [6]

Es una extensión de Chrome y hace evaluaciones heurísticas (es una forma de evaluar la usabilidad) que se basan en el método de Jakob Nielsen y sus diez principios para el diseño de interacción. Es muy sencillo de utilizar ya que se instala en el navegador, y una vez hecho esto se visita la página que se quiere evaluar, para posteriormente ver qué elementos no cumplen con los principios de Nielsen. La información obtenida se puede exportar en un documento. (Gallello, s.f.)

Ventaja: es gratuita, y de fácil instalación.

Desventajas: no es intuitiva y hay que saber cómo aplicar los principios de Nielsen y el documento que exporta son capturas de pantallas obtenidas de la página que se analiza.

Para finalizar después de esta investigación se ha llegado a la conclusión de que no hay una herramienta gratuita o de pago igual a Usability Analysis Tool.

Con Usability Analysis Tool el usuario tiene el total control de elegir que cuestiones quiere evaluar, y sobre todo con ayuda de la herramienta evita tener que realizar los cálculos manualmente. Lo que no evita es invertir un tiempo razonable en realizar el estudio y realizar las pruebas a los testers.

3.2 Elección de lenguajes y tecnologías

En todo proyecto es importante elegir lenguajes de programación y tecnologías a usar. Dado que si se hace una mala elección lo complicará. Al tener el proyecto separado en back-end y front-end hay que escoger los lenguajes y tecnologías correspondientes para cada apartado.

Para estudiar cada elección, se exponen ventajas y desventajas, de esa forma se puede analizar diferentes alternativas. Aunque la experiencia de la desarrolladora no es mucha, se espera que, con la investigación realizada, se saquen conclusiones que permitan seleccionar la mejor alternativa teniendo en cuenta que se tiene como uno de los objetivos aprender nuevas tecnologías y lenguajes de programación.

3.2.1 Back-end

Es común desarrollar el back-end con lenguajes como PYTHON, JAVA, .NET, PHP, JavaScript entre otros más. Para este proyecto se ha decidido descartar PYTHON y JAVA debido a que la desarrolladora ya conoce estos lenguajes y uno de los objetivos es conocer nuevos lenguajes de programación, por otra parte, están .NET y PHP que son lenguajes nuevos para la desarrolladora, pero exigen tiempo para aprender y conseguir una base muy sólida, lo que permitiría realizar una herramienta web bien estructurada.

Por otra parte, la desarrolladora tiene un poco de conocimiento en JavaScript, entonces haciendo uso de Express, un framework de NodeJS basado en JavaScript se decide utilizar esta tecnología nueva que aporta conocimientos nuevos, además de estar demandado en el mercado laboral.

[6] http://www.uxcheck.co/

Express aporta una estructura minimalista, flexible y robusta, también acepta utilizar verbos como GET, POST, PUT Y DELETE, métodos que han sido descritos en el apartado de arquitectura.

3.2.2 Front-end

A día de hoy la estructura HTML5 y lenguaje CSS3 son los más básicos para realizar una página web. También se tiene XHTML que está dedicado al desarrollo web y se basa en XML. Esta última sería de gran utilidad si se quisiera utilizar herramientas como XSLT que permite hacer transformaciones de documentos, o si se quisiera utilizar MathML que es un lenguaje que permite describir expresiones matemáticas, o por último SMIL o SVG, ambas útiles para describir presentaciones multimedia, en esos casos se sacaría provecho de este lenguaje ya que permite ese tipo de combinaciones.

Como este es un proyecto que no trabaja con esas herramientas, se elige trabajar con HTML5 y CSS3. A continuación, se muestran las ventajas y desventajas (Tabla 26 y Tabla 27) relevantes de ambos lenguajes para el desarrollo de este proyecto:

TABLA 26:VENTAJAS Y DESVENTAJAS DE HTML5

Ventajas	Desventajas
• Facilidad de uso presentando una forma estructurada y agradable. • Comunicación rápida y directa, admitida por todos los navegadores. • El lenguaje es conocido por la desarrolladora. • Forma de trabajo estructurada. • Se despliega con facilidad.	• Es un lenguaje estático. • Cada navegador lo puede interpretar de forma diferente. • En las versiones anteriores el uso de las tablas era más sencillo. • Guarda muchas etiquetas que pueden dificultar la corrección.

TABLA 27: VENTAJAS Y DESVENTAJAS DE CSS3

Ventajas	Desventajas
• Permite estilos y efectos visuales que antes sólo eran posibles por medio de tecnología adicionales. • Permite tener un mayor control de la presentación	• En alguna ocasión el diseño de la página puede verse distinta en cada navegador • No permite crear diseños de tablas complejos.

También es importante elegir un buen framework con el que desarrollar el front-end y en la actualidad hay infinidad de ellos, lo que dificulta al momento de decidir cual usar, la desarrolladora de este proyecto se decanta por tecnologías nuevas, fáciles de utilizar, ágiles y que estén demandadas en el mercado laboral. Por esa razón los frameworks a comparar son los siguientes: *Polymer*, *AngularJS* y *Angular* todos ofrecen servicios de forma más ágil y un nuevo estilo de programación mediante componentes web.

Un componente web es un trozo de código que permite ser reutilizado, por ejemplo, se puede definir un componente header y reutilizarlo en todas las páginas que tenga una aplicación, sin necesidad de repetir el código. Lo único que hay que hacer es poner como etiqueta el nombre de ese componente, en donde se quiera utilizar. De esta forma se consigue modularidad.

A continuación, se describen los 3 frameworks, además de las ventajas y desventajas de cada una.

Polymer [7]

Es un framework que consta de librerías de componentes web que facilitan el lanzar nuevas aplicaciones haciendo uso de dichos componentes, lo que permite reutilizar código. Sus desarrolladores también participan en el proceso de estándares lo que garantiza una buena evolución de la plataforma. (Polymer Authors, 2017)

Antes de empezar a describir AngularJS y Angular es importante mencionar que Polymer hace uso de JavaScript, mientras que los otros dos utilizan TypeScript, lo cual para la desarrolladora es una ventaja, porque es un lenguaje nuevo para ella, además de que este lenguaje facilita la forma de leer el código, su mantenimiento es fácil, evita la confusión y sobrecarga en la toma de decisiones.

Ventajas:

- Permite realizar el desarrollo de componentes propios, gracias a que cuenta con una gran librería de elementos útiles para hacer nuevos componentes.
- Permite trabajar con componentes web.

Desventajas:

- Si se utiliza en fases tempranas del proyecto puede retrasar la salida de la aplicación ya que hay poco soporte.
- No se encuentra mucha información en caso de tener problemas al momento del desarrollo.
- Impone el uso de material design.

AngularJS [8]

"AngularJS es un conjunto de herramientas para construir el marco más adecuado para el desarrollo de su aplicación. Es completamente extensible y funciona bien con otras bibliotecas. Cada característica se puede modificar o reemplazar para adaptarse a su flujo de trabajo de desarrollo único y a sus necesidades." (Google © 2010-2017, s.f.)

Utiliza un sistema en el que vista y modelo están en relación constante, que permite un cambio visual en tiempo real en el modelo y viceversa. Además de evitar que el desarrollador tenga que lograr sincronización entre el modelo y la vista.

Ventajas:

- Permite utilizar TypeScript, además de JavaScript.
- No hace falta que la desarrolladora haga posible la sincronización entre el modelo y la vista.
- Permite trabajar con componentes web.

Desventajas:

- Se sobre exigen los recursos del dispositivo del usuario.
- La primera vez se carga la aplicación entera en el navegador.
- Dificulta el trabajo con web workers.
- El soporte ha disminuido desde la aparición de Angular.

[7] https://www.polymer-project.org/
[8]https://angularjs.org/

Angular [9]

Se trata de un nuevo framework basado en muchas ideas de *AngularJS*, es sencilla y también está basado en componentes web, permite la reutilización de código, es de fácil uso para web y web móvil.

Su velocidad y rendimiento logran llevarlo más allá de web workers (ejecución de secuencias que se ejecutan en segundo plano) y la representación del lado del servidor. Con Angular se pueden crear HTML's simples. (Google ©, 2010-2018)

Ventajas:

- Permite utilizar TypeScript, además de JavaScript.
- Tiene buen rendimiento y funcionamiento en dispositivos móviles.
- Hace uso directo de las propiedades de los elementos y los eventos estándar.
- Permite modularidad.

Desventajas:

- Cuando la capacidad computacional del dispositivo es menor y la red está saturada o hay una mala conexión el rendimiento decae.

Para concluir este apartado de front-end, a pesar de que los tres frameworks ofrecen trabajar con componentes webs, se ha decidido a utilizar Angular debido a que es un framework mucho más estable que AngularJS, permite compatibilidad con otras librerías, se encuentra en pleno en auge y ofrece facilidad de aprendizaje a la vez que cuenta con una gran comunidad que lo avala, lo que facilita solucionar problemas que pudieran surgir.

Además de ofrecer desde el principio como organizar el código y la arquitectura de la herramienta web. Por no mencionar un conjunto de APIs que permiten crear etiquetas HTML personalizadas, reutilizables.

3.2.3 Elección del IDE

Una vez hecha la criba de los lenguajes y tecnologías a utilizar, es importante escoger un entorno de desarrollo que facilite el uso de todo lo explicado en los dos apartados anteriores. Y que además ofrezca rapidez y la creación de la estructura Express.

Se ha realizado una investigación y a pesar de haber muchos se ha dado con dos entornos de desarrollo muy buenos: Visual Studio Code y WebStorm ambos aportan corrección de sintaxis tanto para JavaScript como para TypeScript, son livianos, rápidos, cuentan con un potente depurador del lado del cliente y del servidor, se integran con NodeJS, en general son eficientes y efectivos. Pese a todas estas similitudes se ha decidido elegir Visual Sudio Code, por las siguientes razones:

- Es más rápido y sensible de WebStorm.
- La experiencia de depuración es mejor, además de tener un terminal integrado.
- Es gratuito.
- Hay actualizaciones constantemente.
- Está patrocinado por Microsoft.

[9] https://angular.io/

3.2.4 Base de datos

Basándose en el hecho de que no es necesario recurrir a bases de datos de empresas como Oracle debido a ser un trabajo de investigación. Se comparan dos soluciones: MySQL [10] y PostgreSQL[11].

Ambas ofrecen opciones muy similares:

- Son de código abierto y gratuitos.
- Ofrecen soporte en caso de necesitarlo.
- Son plataformas muy conocidas y utilizadas por grandes compañías.
- Cumplen ACID (atomicidad, consistencia, aislamiento, durabilidad), aunque en el caso de MySQL solo lo cumple cuando usa mecanismos de almacenamiento InnoDB
- PostgreSQL aporta seguridad desde las bases con soporte SSL, mientras que MySQL solo lo ofrece para algunas. versiones.

En general si este proyecto necesitase más capacidad y más diversidad de opciones, se elegiría PostgreSQL, debido a la variedad de opciones que ofrece, como este no es el caso y no se requiere mayor funcionalidad que la que ofrece MySQL, se decide implantarlo como base de datos.

3.3 Métricas de evaluación para los objetivos de la usabilidad

En este apartado se explica cómo se miden los objetivos de la usabilidad, además de las ecuaciones que se van a utilizar en cada objetivo.

Para calcular la efectividad y la eficiencia se tienen en cuenta las tareas que realizan los testers. Estas tareas serán definidas previamente por el usuario y son las tareas más significativas que los usuarios deben poder realizar en la aplicación cuya usabilidad se quiere estudiar. Por ejemplo, si se quiere analizar la usabilidad de una tienda on-line, entre las tareas a definir podrían estar “buscar producto” o “comprar producto”.

De las tareas a estudiar es necesario definir el tiempo que se considera adecuado para su realización. De este modo se podrá analizar si los testers son capaces de realizar las tareas en un tiempo razonable o no.

También se debe definir cada uno de los pasos que tiene que dar el tester para realizar la tarea: las pantallas por las que debe pasar y el número de clicks que debe dar en cada una de esas pantallas. Esta información se denomina el path de la tarea. El usuario puede definir uno o varios paths óptimos, pero también tiene la opción de considerar otras alternativas, a las que se denominan paths alternativos.

Cuando los testers realizan las tareas que les han sido asignadas, además de almacenar las páginas por las que pasan, el número de clicks que dan en cada una de ellas y el tiempo que ha tardado en completar dicha tarea, también se almacena si finalmente fue capaz de resolver la tarea o no.

[10] http://www.oracle.com/us/products/mysql/mysql-wp-top10-webbased-apps-461054.pdf
[11] https://www.postgresql.org/about/

Para calcular la satisfacción es necesario almacenar la información de las respuestas que han dado los testers a las preguntas del cuestionario. Este cuestionario y las preguntas, las tiene que definir previamente el usuario.

3.3.1 Efectividad

La efectividad se calcula en base a las tasas de éxito muestral de una tarea (p) (Tullis & Albert, 2013). La ecuación para este cálculo es la siguiente, Ecuación 1:

$$p = \frac{x}{n}$$

ECUACIÓN 1: TASA DE ÉXITO MUESTRAL

Dónde x representa el número de testers que han completado la tarea correctamente; y n el número total de testers que han realizado la tarea.

Con la tasa de éxito muestral calculada, se puede estimar cual será la tasa de éxito poblacional. Para las tasas de éxito muestral que se encuentren entre 0.5 y 0.9, la estimación de la tasa de éxito poblacional se corresponderá con la tasa de éxito muestral, mientras que para otros valores se utilizará otros estimadores más precisos (Lewis & Sauro, 2006).

Si la tasa de éxito muestral es menor o igual a 0.5, el estimador para la tasa de éxito poblacional es el método de Wilson (Wilson, 1927):

$$Wilson = \frac{(x + z^2_{\frac{\alpha}{2}}/2)}{n + z^2_{\frac{\alpha}{2}}}$$

ECUACIÓN 2: MÉTODO DE WILSON

Donde x es el número de testers que han realizado la tarea correctamente; n es el número total de testers que han realizado la tarea; α es el nivel de significación; y $z^2_{\frac{\alpha}{2}}$ es el valor crítico de la distribución normal estandarizada.

Para las tasas de éxito muestrales mayores a 0.9, el método a utilizar es Laplace (Laplace, 1812) (Lewis & Sauro, 2006):

$$Laplace = \frac{x + 1}{n + 2}$$

ECUACIÓN 3: MÉTODO DE LAPLACE

Donde x es el número de testers que han realizado la tarea correctamente; y n es el número total de testers que han realizado la tarea.

Para poblaciones de muestra inferiores a 150 testers, la mejor forma de calcular la tasa de éxito muestral, es a través de intervalos de confianza, calculados mediante el método ajustado de Wald (Agresti & Coull, 1998):

$$\hat{p} \pm z^2_{\frac{\alpha}{2}} \times \sqrt{\frac{\hat{p} \times (1 - \hat{p})}{n + z^2_{\frac{\alpha}{2}}}}$$

ECUACIÓN 4: MÉTODO DE WALD

Donde $\hat{p}$ representa al estimador de Wilson, calculado con la Ecuación 2; α es el nivel de significación; y $z_{\frac{\alpha}{2}}^2$ es el valor crítico de la distribución normal estandarizada.

Para calcular la efectividad, además de si los testers finalizaron la tarea correctamente o no, se tienen en cuenta 3 niveles de finalización: *abandono*, la tarea no se finalizó con éxito; *con problemas*, la tarea se finalizó correctamente, pero el path realizado no se corresponde con el óptimo o alternativo; *sin problemas*, la tarea se realizó correctamente siguiendo el path óptimo o alternativo.

Con los niveles de finalización que se han definido, se realizan nuevamente los cálculos, para estimar la tasa de éxito muestral, poblacional y sus intervalos de confianza.

3.3.2 Eficiencia

En la eficiencia se tienen en cuenta los tiempos de ejecución de las tareas y el nivel de perdida, además del recorrido que los usuarios han realizado (ese recorrido son los paths realizados).

Tiempos de ejecución

Para establecer el tiempo máximo admisible, se calcula el percentil 95 de los tiempos de ejecución de los testers que finalizaron la tarea y, además puntuaron satisfactoriamente el sistema (Sauro & Kindlund, 2005). Satisfactorio se considera a puntuaciones mayores a 71 puntos en el cuestionario, ver apartado Satisfacción.

En los estudios con menos de 25 testers, para calcular la mediana del tiempo de realización de las tareas, se utiliza la media geométrica (G) (Sauro & Lewis, 2010)

Antes de calcular el percentil 95 es necesario normalizar los tiempos, ya que estos no siguen una distribución normalizada, porque no se distribuyen proporcionalmente a lo largo del tiempo (Sauro & Lewis, 2012). Para normalizar estos tiempos se transforman a través de la función logarítmica. La media geométrica (G) de los tiempos, se calcula como la media aritmética de los algoritmos:

$$G = \frac{\sum_{i=1}^{x} \ln t_i}{x}$$

ECUACIÓN 5: MEDIA GEOMÉTRICA

Donde x representa el número de testers que han finalizado la tarea; y t_i el tiempo que el tester i tardó en realizarla. Después de realizar el cálculo, G se transforma exponencialmente para expresar el resultado en la misma escala en la que se han recogido los tiempos.

Una vez normalizado los tiempos, se calcula los intervalos de confianza por medio de la *t de Student*:

$$\bar{x} \pm t_{n-1,\alpha/2} \times \frac{s}{\sqrt{n}}$$

ECUACIÓN 6: T DE STUDENT

Donde $\bar{x}$ es la media aritmética de la muestra; n es el tamaño de la muestra; $t_{n-1,\alpha/2}$representa al valor crítico de la distribución para n – 1 grados de libertad; α es el valor de significación; y s es la desviación estándar muestral, y se calcula de la siguiente forma:

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

ECUACIÓN 7: DESVIACIÓN ESTÁNDAR

Nivel de pérdida

El nivel de pérdida o lostness se representa en una escala numérica de 0 a 1, lo que significa cuanto se ha perdido el tester durante la realización de una tarea, teniendo en cuenta el path realizado (Smith, 1996).

El nivel de pérdida de un tester se calcula mediante la ecuación:

$$L = \sqrt{\left(\frac{N}{S} - 1\right)^2 + \left(\frac{R}{N} - 1\right)^2}$$

ECUACIÓN 8: NIVEL DE PÉRDIDA

Donde N es el número de páginas distintas que ha visitado el tester para resolver la tarea; S es el número total de páginas visitadas incluyendo las que se han visitado más de una vez; y R es el número de páginas que hay que visitar para resolver la tarea (los paths de la tarea).

El intervalo de confianza del nivel de pérdida, se calcula con la Ecuación 6.

Grafos de navegación

Para identificar las razones por las que los usuarios se pierden al realizar una tarea, se define un tipo de grafo, que permite visualizar los datos de navegación de todos los testers en una tarea determinada, ver el ejemplo de la Figura 5.

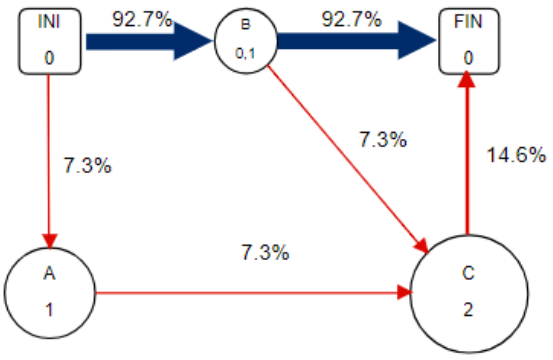


FIGURA 5: GRAFO DE NAVEGACIÓN

En este tipo de grafo los nodos cuadrados indican si es de comienzo y/o finalización, los nodos redondos, son los nodos intermedios. Cada nodo está compuesto por dos etiquetas, el primero es el nombre corto que representa a la página de la aplicación y que está representada

por ese nodo, y el segundo, es el promedio de acciones extra que los testers realizaron en esa página. Las acciones extra, son las acciones que no son necesarias para la realización de la tarea. Para calcular el promedio de las acciones extra, sólo se tiene en cuenta el número de testers que pasaron por dicha página. El tamaño de los nodos está relacionado con el promedio de las acciones extra.

Los arcos del grafo representan el recorrido realizado por los testers en las distintas páginas de la aplicación. El color del arco representa si el paso de una página está contenido en el path óptimo o alternativo de la tarea (las flechas azules) o no (flechas rojas). El grosor del arco y la etiqueta indica el porcentaje de testers que realizaron ese paso.

3.3.3 Satisfacción

El cálculo de la satisfacción de los testers, se realiza mediante la puntuación obtenida de las respuestas del cuestionario SUS utilizando la Ecuación 9.

$$\sum_{i=1,3,5,7,9} (q_i - 1) + \sum_{j=2,4,6,8,10} (5 - q_j)$$

ECUACIÓN 9: ECUACIÓN PARA LA SATISFACCIÓN

Donde q_i representa la valoración dada por el tester a las preguntas i, que representan a las preguntas con connotación positiva, y q_j que representan a las preguntas j con connotación negativa. (Brooke, 1996)

Una vez realizado el cálculo de satisfacción de cada uno de los testers, se calcula la media aritmética de esos resultados. Y se construye un intervalo de confianza con la ecuación.

Para categorizar los resultados obtenidos se utiliza la escala definida por (Bangor, Kortum, & Miller, 2009), en la cual, a partir de 51 puntos, el grado de satisfacción es aceptable, siendo deseable alcanzar 71 puntos, que representa el límite inferior para una buena usabilidad.

Antes de terminar este apartado de Métricas de evaluación para los objetivos de la usabilidad, es importante mencionar que la información de este apartado, ha sido extraída de (Villamañe, 2016).

4. CAPTURA DE REQUISITOS

La captura de requisitos es el primer paso a dar en cualquier aplicación que se quiera realizar, además se puede considerar como uno de los pasos más importantes en el proceso de realización.

Es necesario realizar un buen análisis para hacer una buena elección herramientas necesarias que se adapten a los requerimientos actuales y futuros de la herramienta web. Se empieza por establecer los requisitos funcionales, para posteriormente continuar con los requisitos funcionales, jerarquía de actores, casos de uso y finalizar con el modelo de dominio.

4.1 Requisitos funcionales

Los requisitos funcionales son aquellos que definen las funciones del sistema o sus componentes. En este proyecto son los siguientes:

- La herramienta web debe permitir iniciar sesión y registrarse.
- Sólo los usuarios que hayan iniciado sesión pueden hacer uso de sus funcionalidades.
- Una vez iniciada la sesión el usuario puede crear, modificar o eliminar: proyectos, tareas, testers, cuestionarios, tareas realizadas, respuestas de cuestionarios y configurar un proyecto.
- Obtener resultados sobre los objetivos básicos de la usabilidad, con los datos obtenidos del estudio, y que han sido creados por el usuario.

4.2 Requisitos no funcionales

Los requisitos no funcionales son los que se deben cumplir independientemente de las funcionalidades que se van a implementar en el desarrollo. En este caso se han identificado dos importantes.

- La herramienta web tiene que ser accesible, de forma que cualquier usuario pueda hacer uso de sus funcionalidades sin necesitar un periodo de adaptación.
- El sistema debe de ser estable.

4.3 Jerarquía de actores

En este proyecto se han identificado dos actores, usuario anónimo y usuario registrado, esto se debe a que no se necesita ningún permiso adicional o alguna restricción al acceso de datos, en la Figura 6 se muestra la jerarquía de actores:

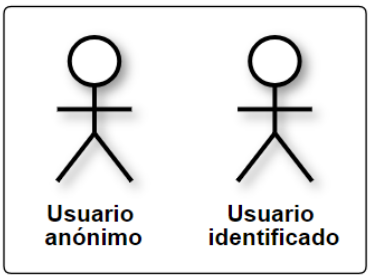


FIGURA 6: JERARQUÍA DE ACTORES

- **Usuario anónimo**: es aquel que llega a la web y puede iniciar sesión o registrarse.
- **Usuario identificado**: es el usuario que hace uso de sus datos de inicio de sesión para iniciar sesión.

4.4 Casos de uso

Los casos de uso son una representación gráfica de las funcionalidades del sistema. Es decir, un caso de uso representa una acción determinada del sistema, mostrando como debe interactuar el sistema con el usuario para conseguir su objetivo. En este proyecto se han definido casos para los dos actores antes mencionados en la jerarquía de actores.

En la Figura 7 se muestran las dos funcionalidades asociadas al actor anónimo y se definen a continuación:

- **Registrarse**: es la funcionalidad que permite al usuario anónimo registrarse en la web, para ello es necesario poner un nombre de usuario y una contraseña.
- **Iniciar sesión**: permite al usuario anónimo iniciar sesión con el usuario y contraseña utilizados al momento de registrarse. Los datos introducidos por el usuario son contrastados con los datos almacenados en la base de datos, y en caso de no existir el usuario se le muestra un mensaje indicando que el usuario no existe. Esta funcionalidad es necesaria para tener acceso al sistema.

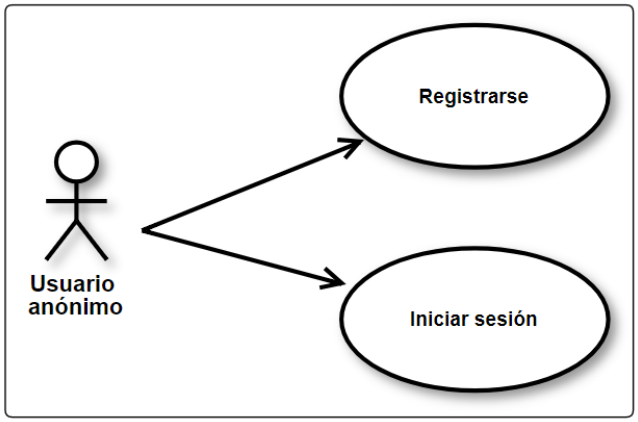


FIGURA 7: CASO DE USO USUARIO ANÓNIMO

A diferencia del usuario anónimo el usuario registrado tiene muchas más funcionalidades, en la Figura 8 se puede ver cuatro de ellas principales: “Crear proyecto”, “Cerrar sesión”, “Eliminar proyecto”, “Ver proyectos”. Los que están con las líneas punteadas son subcasos de uso.

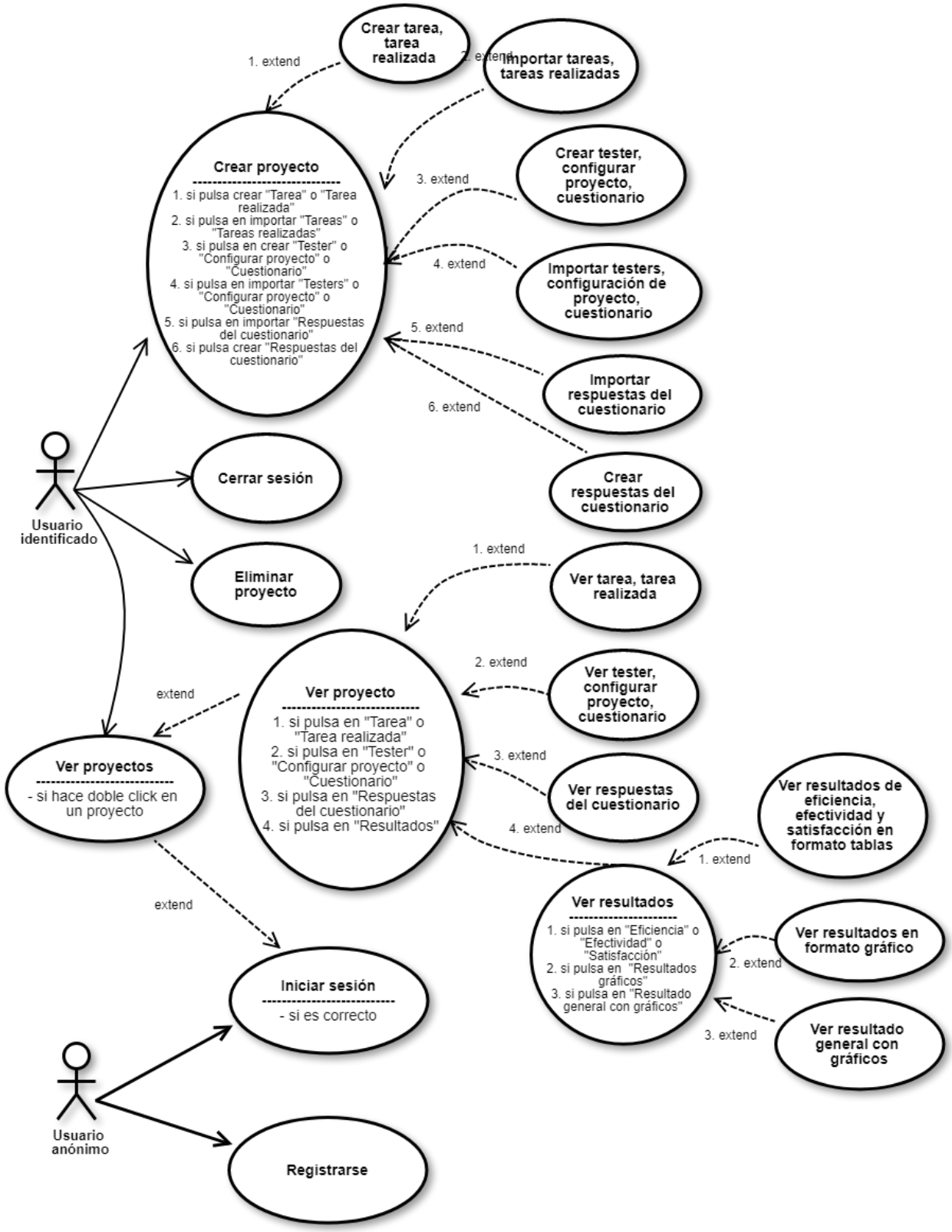


FIGURA 8: CASOS DE USO

Continuando con los casos de uso y sus funcionalidades, en las siguientes líneas se detalla cada uno de ellos, teniendo en cuenta los subcasos de uso que pudieran tener. Todas las funcionalidades descritas a continuación se encuentran disponibles para el usuario identificado.

- **Cerrar sesión**: como en la gran mayoría de webs el usuario identificado puede cerrar sesión esto permite salir de forma segura del sistema. Al ejecutar esta funcionalidad el usuario identificado pasa a ser usuario anónimo.

- **Ver proyectos**: cuando el usuario inicia sesión y todo ha ido correctamente, la primera pantalla que ve es un listado con todos los proyectos que ha creado. Por esa razón extiende de "Iniciar sesión".

    - **Ver proyecto**: en esta funcionalidad se ejecuta cuando el usuario hace doble clic en un proyecto del listado de "Ver proyectos". Lo que hace que tenga disponible las funcionalidades que se describen a continuación:

        - **Ver tarea, tarea realizada**: en este subcaso de uso se han unido dos funcionalidades, debido a que estas son similares. Ver una tarea permite al usuario ver una tarea y los paths óptimos o alternativos que pueda tener, mientras que la tarea realizada permite ver las tareas y los paths que han realizado los testers.

        - **Ver tester, configurar proyecto, cuestionario**: en este subcaso de uso hay tres funcionalidades, la primera, "Ver tester" es una funcionalidad que permite al usuario ver todos los testers que pertenecen al proyecto.
          "Configurar proyecto" es básicamente ver los nodos que se han creado y que representan a las páginas que son visitadas en cada tarea.
          "Ver cuestionario" es visualizar el cuestionario y todas las preguntas asociadas a él.

        - **Ver respuestas del cuestionario**: esta funcionalidad permite visualizar todas las preguntas y sus respectivas respuestas que han sido respondidas por los testers.

        - **Ver resultados**: esta es una de las funcionalidades que representan la razón de ser de este proyecto, ya que con a esta funcionalidad el usuario puede elegir si ver los resultados de forma individual por cada objetivo, o de forma general, que incluye los tres objetivos. Por este motivo este subcaso de uso se divide en los subcasos de uso que se explican a continuación:

            - **Ver resultados de eficiencia, efectividad y satisfacción en formato tablas**: este subcaso de uso al igual que otros vistos anteriormente se ha unido en uno por tener similitudes.
              Ver resultados de eficiencia, efectividad y satisfacción es visualizar los resultados obtenidos de las tareas realizadas y del cuestionario respondido por los testers.

            - **Ver resultados en formato gráfico**: esta funcionalidad está directamente relacionada con uno de los objetivos de la usabilidad, en concreto con la eficiencia, debido a que los resultados gráficos se obtienen de los resultados de la eficiencia. Entonces esta funcionalidad permite visualizar grafos de navegación.

- **Ver resultado general con gráficos**: esta funcionalidad es una mezcla de las dos funcionalidades anteriores que permite mostrar los tres resultados de los objetivos de la usabilidad, con solo hacer click en un botón, estos resultados incluyen los gráficos de cada una de las tareas.

- **Crear proyecto**: al iniciar sesión el usuario identificado además de ver un listado de los proyectos que tiene, puede crear un proyecto haciendo click en el botón "Crear proyecto". Esta funcionalidad como ya se ha descrito permite crear un proyecto y posteriormente seguir creando o no testers, nodos, cuestionarios, tareas, tareas realizadas y respuestas de un cuestionario.
  Para realizar las funcionalidades que se describen a continuación es necesario haber creado un proyecto previamente.

  - **Crear tarea, tarea realizada**: estas dos funcionalidades permiten por un lado crear una tarea y sus path correspondientes, y por otro crear las tareas y los paths que ha realizado un tester. En el caso de las tareas realizadas es requisito obligatorio que los datos de las tareas que se vayan a crear existan.

  - **Importar tareas, tareas realizadas**: con estas funcionalidades el usuario tiene la opción de importar fichero en formato JSON, agilizando así la creación de tareas y tareas realizadas, ya que evita crearlas una a una. Es importante mencionar que el fichero tiene que contener todas las tareas y sus paths correspondientes. Al realizar la importación de las tareas es necesario que los nodos de los paths se hayan creado previamente, lo mismo ocurre con las tareas realizadas y sus path, es necesaria la existencia de los nodos y las tareas.

  - **Crear tester, configurar proyecto, cuestionario**: estas funcionalidades son necesarias para crear testers, configurar un proyecto que no es más que crear los nodos que se utilizan en los paths las tareas y por último crear cuestionarios con las preguntas que desee el usuario. Todo esto se crea de forma manual, una a una.

  - **Importar testers, configurar proyecto, cuestionario**: estas funcionalidades son un poco más avanzadas de las anteriores de "Crear". Ya el que "Crear" limita a hacerlo uno a uno, sin embargo, estas funcionalidades permiten importar desde un fichero en formato CSV los testers, la configuración de proyecto y el cuestionario. Lo que hace que se creen automáticamente. Es necesario un fichero para cada funcionalidad.

  - **Crear respuestas del cuestionario**: esta funcionalidad permite al usuario crear una a una las respuestas que hayan dado los testers al cuestionario. Es importante tener en cuenta que las preguntas tienen que crearse antes de ser contestadas.

  - **Importar respuestas del cuestionario**: esta funcionalidad importa de un fichero de formato CSV, las respuestas que hayan dado los testers a las preguntas de un cuestionario sin necesidad de que el usuario las cree una a una.

- **Eliminar proyecto**: eliminar un proyecto significa borrar un proyecto y todos los datos asociados a él. Es necesario entrar en el proyecto que se quiere eliminar y antes de eliminar el sistema muestra un mensaje de advertencia indicando que se va a eliminar de forma definitiva.

Se pueden encontrar más detalles de los casos y subcasos de uso en el ANEXO I.- CASOS DE USO EXTENDIDOS.

4.5 Modelo de Domino

El modelo de dominio permite comprender y describir las clases más importantes dentro del contexto del sistema. En este apartado se explica de forma detallada el significado de cada una de las entidades y las relaciones que existe entre ellas.

Las entidades y las relaciones que se muestran en la Figura 9 son necesarias para este proyecto.

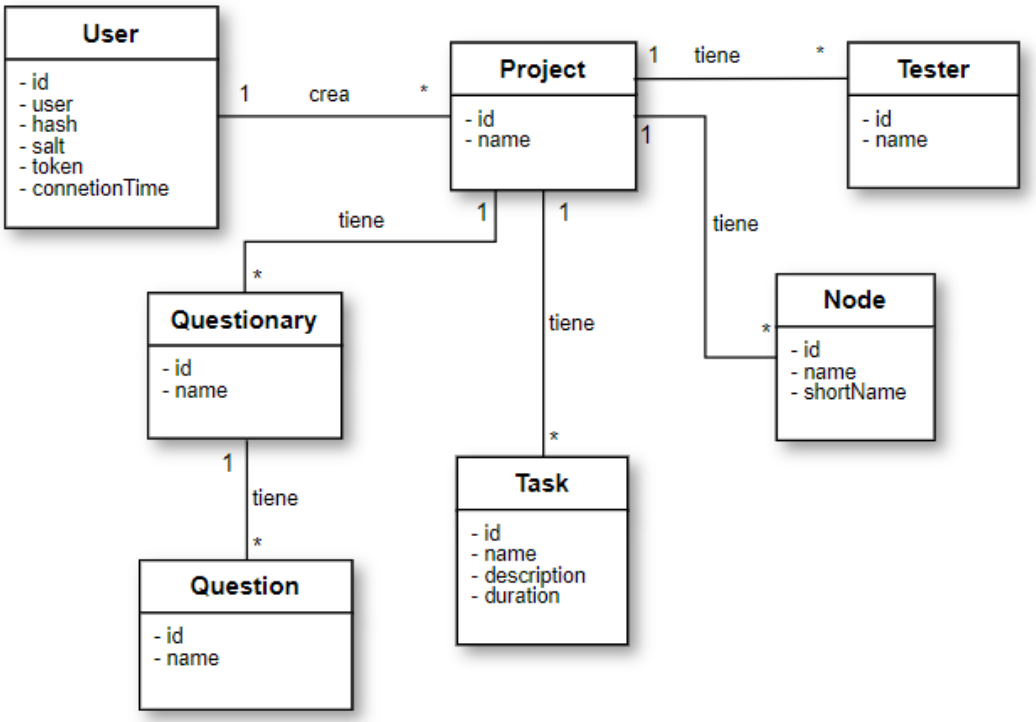


FIGURA 9: MODELO DE DOMINIO (I)

A continuación, se explican cada una de las entidades y las relaciones que se han mostrado en la Figura 9.

- **USER**: el usuario cuenta con un id único, además de con un nombre de usuario, hash, salt, token y el tiempo de conexión para controlar la sesión.
  Relación con project: tiene cardinalidad 1 a N, esto se debe a que un usuario puede crear muchos proyectos y un proyecto le pertenece a un único usuario.

- **PROJECT**: esta entidad es la que engloba a las demás entidades. El proyecto cuenta con un id único y el nombre que se le quiera poner al proyecto.

Relación con tester, node, questionary y task: son relaciones de cardinalidad 1 a N, es decir un proyecto puede tener muchos testers, nodos, cuestionarios y tareas. Y por el contrario un tester, un nodo, un cuestionario y una tarea pertenecen a un proyecto.

- **NODE**: esta entidad representa a las páginas que serán visitadas en la realización de una tarea y que forman parte del proyecto. Un nodo tiene un id único, un nombre y un nombre corto que ha de ser único.

- **TESTER**: es la entidad que representa a la persona que ha realizado una tarea o ha respondido un cuestionario. Cada tester tiene un id único y el nombre del tester.

- **QUESTIONARY**: la entidad questionary contiene preguntas que serán respondidas por los testers. El cuestionario tiene un id único y el nombre que se le quiera dar al cuestionario.
  Relación con question: es una relación de cardinalidad 1 a N, debido a que una pregunta sólo puede pertenecer a un cuestionario, y el cuestionario puede tener muchas preguntas.

- **QUESTION**: la entidad question representa a las preguntas que puede tener un cuestionario. La pregunta tiene un id único y el nombre de la pregunta.

Continuando con el modelo de dominio en la Figura 10 se ven dos relaciones: Tester – Task y Tester – Question

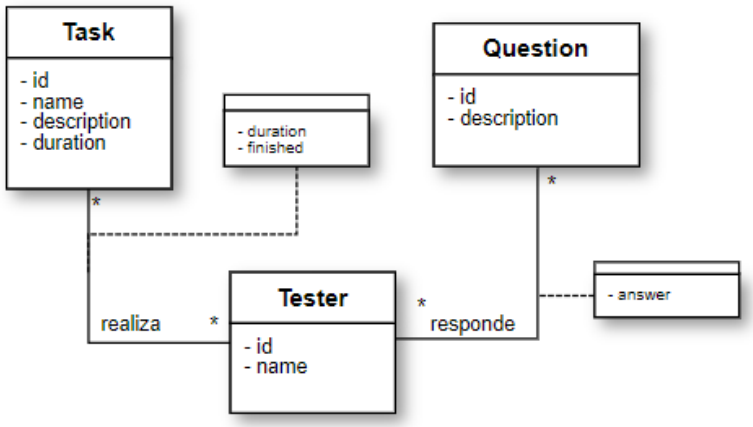


FIGURA 10: MODELO DE DOMINIO (II)

- **RELACIÓN TESTER – TASK**: es una relación de cardinalidad N a M, lo que significa que un tester puede realizar muchas tareas y una tarea puede ser realizada por muchos testers. Esta relación tiene dos atributos: el primero es la duración, que es el tiempo que ha tardado el tester en realizar la tarea y el otro atributo es finalizado, es el que guarda la información de si el tester ha finalizado o no la tarea.

- **RELACIÓN TESTER – QUESTION**: al igual que la relación anterior, es de cardinalidad N a M, esta relación hace referencia las preguntas que han sido respondidas por un tester, o los testers que han respondido una pregunta, siendo necesaria para esta relación guardar la respuesta que se ha dado a esas preguntas.

En la Figura 11 se muestran dos relaciones múltiples que están representadas por las mismas entidades, pero tienen diferente significado, por esa razón se ha decidido tratarlas en un mismo apartado.

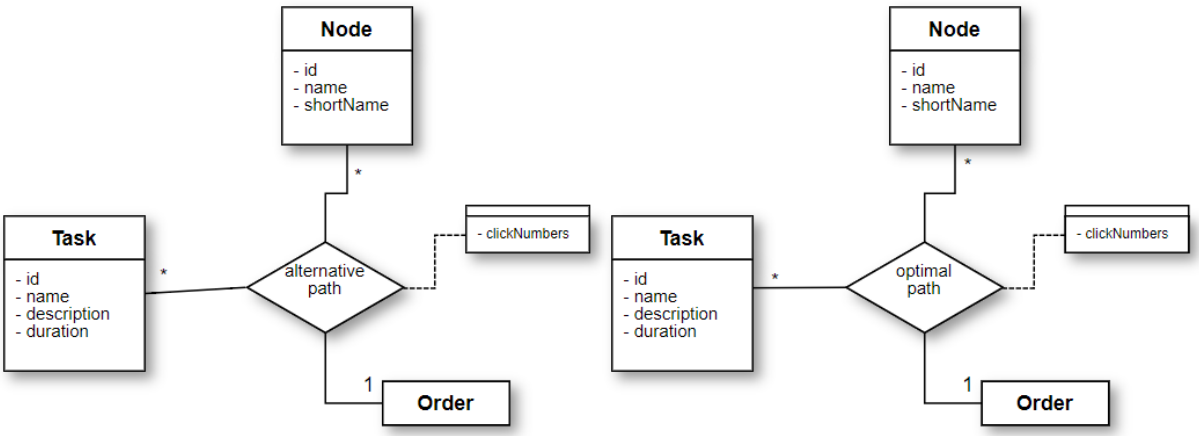


FIGURA 11: MODELO DE DOMINIO (III)

- **RELACIÓN TASK – NODE – ORDER**: estas son dos relaciones que tienen la misma funcionalidad y la misma cardinalidad. Cada una de ellas representa a un "path óptimo" a la izquierda de la Figura 11, o a un "path alternativo" a la derecha de la Figura 11.
  Estas relaciones representan los nodos (entidad Node) por los que hay que pasar para resolver una tarea determinada (entidad Task) y el orden en el que hay que hacerlo (entidad Order). El atributo clickNumbers indica el número de clicks que se debe hacer en cada nodo en cada ocasión que se pasa por él.
  La lógica aplicada a estas relaciones es la siguiente: un nodo en una posición puede estar en muchas tareas. Un nodo en una tarea puede estar en una posición. De esta relación se necesita saber la tarea, el nodo y sobre todo en qué orden hay que seguir el camino, además de los clicks que ha dado en cada nodo.

Por último, en la Figura 12 se muestra una relación muy parecida a la anterior pero que hace referencia al "path realizado".

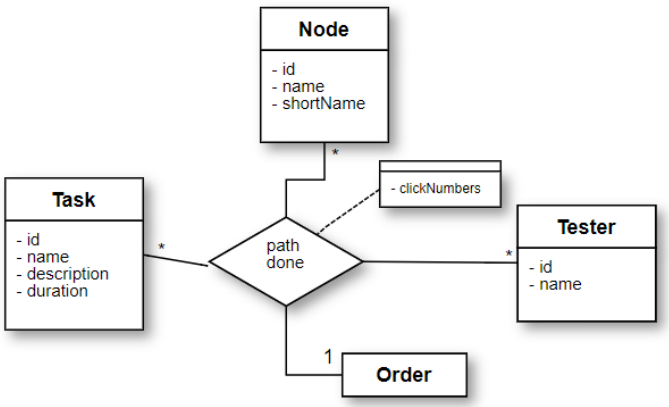


FIGURA 12: MODELO DE DOMINIO (IV)

- **RELACIÓN TASK – NODE – ORDER – TESTER**: como ya sabe de las dos figuras anteriores un path no es más que el camino que hay que recorrer para realizar una tarea. En este caso este path hace referencia al path que ha realizado el tester para resolver una tarea, además de guardar la información de las entidades es necesario guardar el número de clicks que ha hecho en cada nodo.

5. ANÁLISIS Y DISEÑO

En este apartado de análisis y diseño se plantean las pautas de funcionamiento del sistema, se responden las preguntas que engloban el cómo y de qué forma se ha conseguido dar con una solución final que sea eficiente y que satisfaga el planteamiento inicial. Dado que no se trata de una aplicación que se desarrolla orientada a objetos, no se genera un diagrama de clases como tal. Por otro lado, se pueden encontrar detalles sobre diagramas de secuencia en el ANEXO II.- DIAGRAMAS DE SECUENCIA.

Se explican tres apartados, los dos primeros explican las estructuras del back-end y front-end. Y el tercero es el diseño de la base de datos con los diagramas de entidad relación.

Antes de empezar se hace una breve introducción a los primeros apartados referentes al back-end y front-end explicando que son dos estructuras fundamentales en Usability Analysis Tool. Por una parte, se define la estructura del back-end que contiene toda la lógica de negocio del proyecto que cuenta con una arquitectura REST. Por otra parte, la estructura del front-end, trata del cliente y se encarga de la interfaz de la aplicación. En cada apartado se detalla su funcionamiento.

5.1 Estructura del back-end

En este apartado se detallan todos los pormenores de Servidor API, middleware, rutas, controlador y gestor de bases de datos que se muestran en la Figura 13.

Servidor API

El servidor cuenta con diferentes niveles de acceso o capas y se antepone a ellas el *Middleware* (ver apartado *Middleware*) que hace de filtro de seguridad y decide si las peticiones entrantes que se realizan son válidas.

Una vez que la petición ha pasado el filtro de seguridad, llega a la capa de *Routes* o de presentación que determina si el recurso solicitado existe en la API, además de verificar que los datos tengan la estructura correcta, estos datos son enviados a través de la URL o del cuerpo de la petición. Esta capa se encarga de delegar las tareas a *Controllers*.

En caso de que la petición no tenga los parámetros necesarios o no cumpla con lo requerido, la petición es devuelta dejando constancia del estado de la solicitud. En caso contrario el siguiente paso es enviar la petición al nivel inferior, en concreto a *Controllers* el cual se encarga de realizar las llamadas necesarias a *Database Management*, para obtener los datos que ha solicitado el usuario. En *Controllers* se encuentra la lógica y las funciones necesarias para trabajar con los datos, por ejemplo, si se quiere crear un proyecto, analiza si el tipo de datos es correcto, de ser así, prepara los datos para enviarlos a la base de datos a través de *Database Management*. De la misma forma que hace solicitudes para guardar en la base de datos, hace solicitudes de obtención de datos, para luego crear una estructura de datos adecuada para el front-end.

Database Management, tiene como finalidad responder a las solicitudes del *Controllers* además de hacer las peticiones necesarias a la base de datos. En otras palabras, es el que se comunica única y exclusivamente con la base de datos de la aplicación.

Middleware

Middleware es una función que se ejecuta en primera instancia y en base a la respuesta de esa ejecución decide si permite o deniega el acceso al *Servidor API*. Para ello comprueba que el *token* enviado en la cabecera de la petición sea correcto. De esta forma se controlan las solicitudes antes de entrar al sistema.

En la Figura 13 se puede ver de forma más detallada la Arquitectura de este proyecto. El cliente se refiere al front-end. Y servidor API y todo lo que está a su derecha, se ha explicado en el apartado de Servidor API y Middleware.

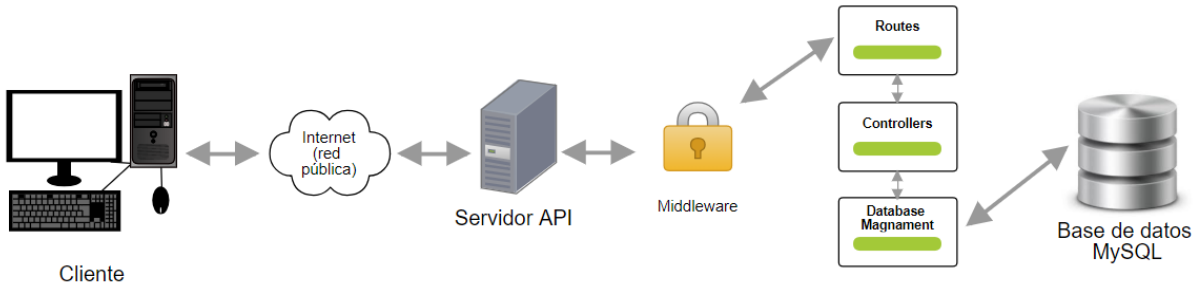


FIGURA 13: ARQUITECTURA USABILITY ANALYSIS TOOL

5.2 Estructura del front-end

La estructura del front-end hace referencia a la parte del cliente. La arquitectura que se utiliza para el front-end es la que se muestra en la Figura 14, en ella se puede ver que la comunicación empieza con HTML y CSS, que en conjunto forman básicamente la interfaz del proyecto, es decir la parte visual de la aplicación.

Los términos que se explican a continuación son necesarios para la correcta comprensión de la Figura 14.

Modules: es uno de los elementos principales de esta arquitectura, ya que permite organizar el código mediante diferentes módulos, y cada uno de estos a su vez agrupan los diferentes componentes que se necesiten. Por ejemplo, en este proyecto se ha decidido tener un módulo llamado *auth* que contiene los componentes para *Login o inicio de sesión* y para el *registro del usuario*. Luego se tiene otro módulo llamado *projectManagement* que tiene componentes para *project, task, questionary, taskDone, questionAnswered, header*, etc. Este último es el encabezado de la interfaz y que lo utilizan todos los componentes que lo necesiten.

Components: o componente, al igual que los módulos es de principal importancia. Este término se encuentra definido en el apartado de Antecedentes. Para recordar un poco esa definición, el componente es una parte de la interfaz y es una parte porque una página puede estar conformada por uno o varios componentes, en concreto un componente es como una etiqueta HTML que se puede crear según la necesidad que tenga la desarrolladora, cada componente puede tener funcionalidad agregada a través de código TypeScript.

Services: o servicios, es una de las piezas fundamentales en el desarrollo del front-end, al igual que los componentes y los módulos. Un servicio es un proveedor de datos, que mantiene la lógica de acceso a ellos. Estos servicios serán consumidos por los componentes, quienes delegarán en los servicios la responsabilidad de acceder a la información. El uso de servicios evitará reescribir código, ya que se puede dar el caso de que dos o más componentes, necesiten acceder a los mismos datos.

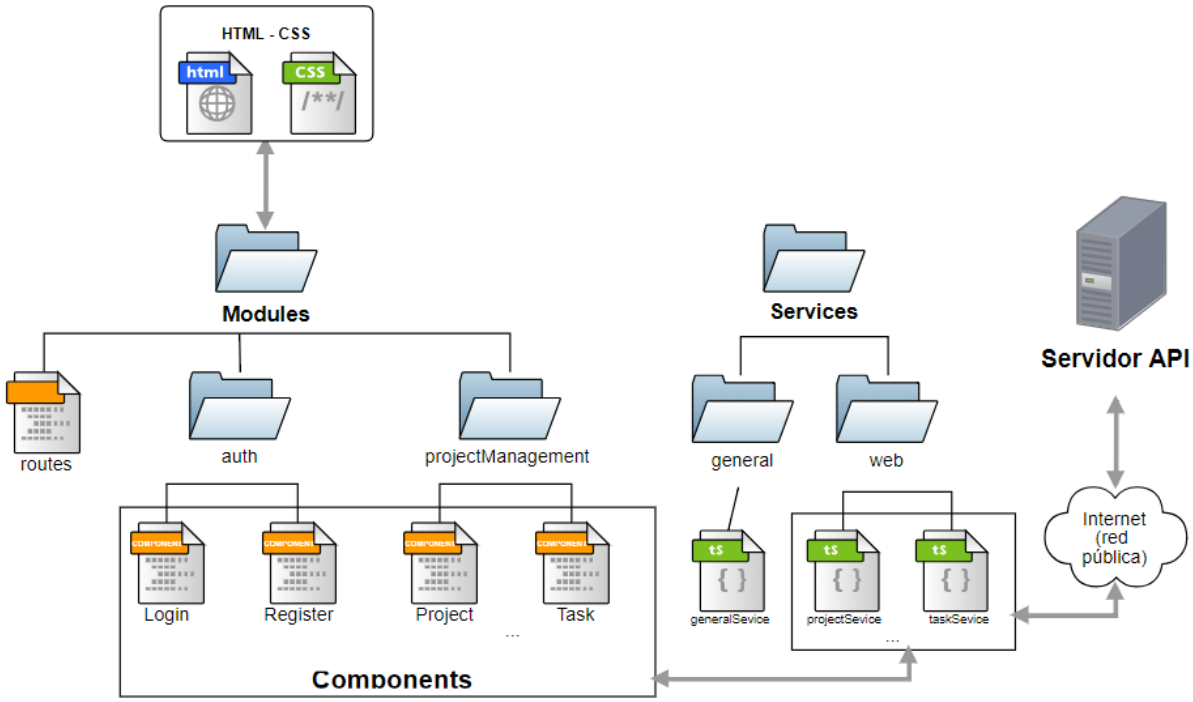


FIGURA 14: ESTRUCTURA FRONT-END

Continuando con el flujo de comunicación, este empieza cuando el usuario hace uso de la herramienta web mediante la interfaz, esta se comunica con *Modules*, que actúa como un gestor de *Components,* además de contar con un fichero en el que guarda las rutas que hacen referencia a dichos componentes.

Para finalizar *Components* se comunica con *Services* y este a su vez con el *Servidor API* a través de URL's. *Services* tiene funciones que dependiendo de lo que el componente necesite, ejecuta una u otra, dependiendo de cuál sea, realiza una determinada llamada al *Servidor API*.

5.3 Diseño del modelo relacional

El diseño del modelo relacional representa las relaciones lógicas existentes en la base de datos. Por ello haciendo uso del modelo de dominio de la aplicación se realiza la transformación del modelo de dominio a base de datos, tal y como se muestra en la Figura 15.

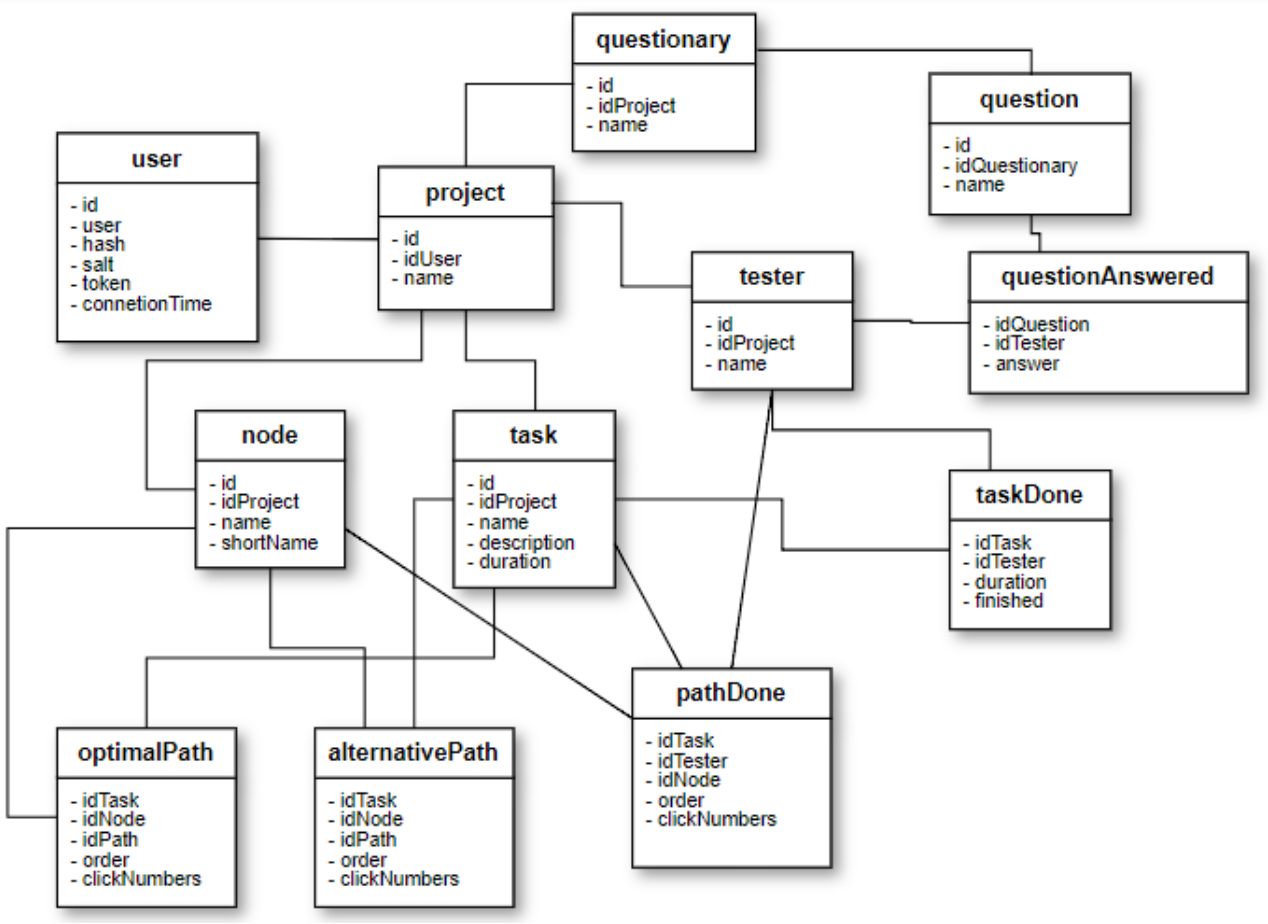


FIGURA 15: DIAGRAMA ENTIDAD - RELACIÓN

En la figura anteriormente mostrada se puede apreciar que hay tablas que no se corresponden con entidades del modelo de dominio, debido a que son relaciones de cardinalidad N a M. Para entender mejor el significado de las tablas nuevas, a continuación se describen cada a una:

- **optimalPath, alternativePath**: la tabla optimalPath es un conjunto de nodos que representan a las páginas que hay que visitar para realizar una tarea.
  Lo mismo ocurre con la tabla alternativePath con la diferencia de que se trata de un path alternativo.

- **pathDone**: la tabla pathDone son un conjunto de nodos que representan a las páginas que ha visitado un tester para una realizar una tarea.

- **taskDone**: esta tabla guarda información referente a las tareas que ha realizado un tester.

- **questionAnswered**: esta tabla guarda las respuestas que ha dado el tester a las preguntas de un cuestionario.

Para finalizar este apartado es importante mencionar que las tablas pathDone, taskDone y questionAnswered son de suma importancia ya que gracias a la información que se guarda en ellas posteriormente se pueden calcular los resultados de los tres objetivos básicos de la usabilidad, que es una de las razones de ser de este proyecto.

6. DESARROLLO

En este apartado de desarrollo se resuelven dudas relacionadas con cómo se ha realizado este trabajo con las herramientas explicadas en el apartado de Antecedentes. La herramienta web está dividida en dos apartados: back-end y front-end, que son los que se describen a continuación.

6.1 Back-end

El back-end contiene toda la parte lógica de la aplicación, es la que se encarga de responder las peticiones que se hacen desde el front-end. A continuación, se describe la estructura y puesta en funcionamiento del back-end.

6.1.1 Estructura de la aplicación y puesta en funcionamiento

Para comenzar este apartado se va a explicar cómo se instalan Express y NodeJS. Ambas herramientas se escogieron en el apartado de Antecedentes y ya se ha justificado la elección. Entonces lo primero que hay que hacer es instalar el IDE, para ello se descarga "*Visual Studio Code" (VSC)* y NodeJS, ambos se instalan de forma convencional.

Una vez instaladas las dos herramientas el siguiente paso es crear la estructura Express, para ello se hace lo siguiente:

1. Se abre VSC y en la terminal que tiene integrada, y se ejecuta el siguiente comando para instalar Express:

```
npm install -g express-generator
```

2. Luego es necesario crear la estructura Express, para ello se ejecuta el siguiente comando:

```
express uat
code .
```

3. Este último comando sirve para abrir el entorno con la estructura creada, que se muestra la Figura 16: Estructura Express básica:

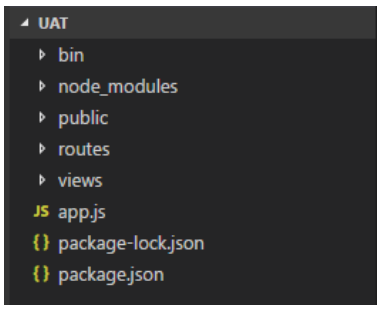


FIGURA 16: ESTRUCTURA EXPRESS BÁSICA

4. Y por último cuando se quiera ejecutar la aplicación es necesario hacer:

```
npm start
```

Llegado a este punto es necesario explicar dicha estructura, ya que para este proyecto no son necesarias todas las carpetas, puesto que esta estructura se utiliza para el back-end, por

tanto, las carpetas “public”, “views”, “routes” y “bin” se eliminan de la estructura, ya que en las dos primeras se guardan imágenes, hojas de estilo (CSS) y todo aquello relacionado con el front-end. Y por otro lado, en la carpeta “routes” se describen las rutas accesibles desde la web, esta carpeta sí que hace falta, pero se cambia de lugar para tener una mejor organización de trabajo y hacerlo, lo más modular posible. Por último, la capeta “bin” es una carpeta de asuntos generales y archivos que contienen funciones que no son necesarias para este proyecto, por lo cual se puede eliminar.

Otra carpeta que llama la atención es node_modules en la que están instaladas todas las librerías que NodeJS ofrece, por lo que es una carpeta que la desarrolladora no modifica en ningún momento.

Una vez explicado porque se han eliminado las carpetas que no hacen falta, se explican los ficheros package.json, package-lock.json y app.js. En package.json se guardan todas las dependencias y los archivos mínimos para poner en marcha la aplicación. Por otro lado package-lock.json es un fichero que se crea automáticamente para cualquier operación en la que npm modifique el árbol de estructura de node_modules o package.json. Por último y no menos importante, app.js es un fichero que contiene todas las rutas disponibles de la herramienta web, se encarga de recibir las rutas y de usarlas.

En la Figura 17 se ven las carpetas y ficheros que han sido creadas por la desarrolladora y que se consideran necesarios para la implementación.

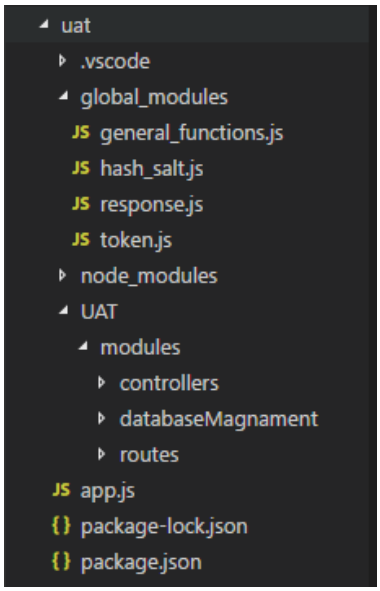


FIGURA 17: ESTRUCTURA DE EXPRESS

Antes de continuar con la definición de la estructura, es importante tener en cuenta que la aplicación hace uso de una base de datos MySQL, y para ello es necesario utilizar un conector que permita realizar la conexión entre el recurso y la base de datos. Esta conexión se consigue instalando la librería *mysql* que está disponible para NodeJS, por ello desde la consola se ejecuta el siguiente comando:

```
npm install mysql
```

Una vez que la librería *mysql* está instalada, se configuran las rutas o recursos de la API. En la figura anterior se puede ver que se han creado nuevos ficheros, los cuales se consideran necesarios para el proyecto, a continuación, se da una explicación de su implementación:

- ***general_functions.js***: contiene funciones que se pueden utilizar desde diversos ficheros con el fin de no reescribir código.
- ***hash_salt.js***: contiene funciones para generar el Salt y Hash necesarios para el registro e inicio de sesión del usuario.
- ***response.js***: en él se configuran los mensajes de confirmación o error con los que responde la aplicación a las peticiones. Además de funciones que comprueban que el código de la respuesta que se va a dar existe.
  Los estados de respuesta que da la aplicación son:
  - **status 200**: se devuelve este estado cuando la petición se ha completado con éxito.
  - **status 201**: este informa sobre el estado de la creación de un recurso.
  - **status 400**: estado que informa que no es capaz de entender la petición del navegador, porque la sintaxis no es correcta.
  - **status 401**: este estado se devuelve cuando el recurso solicitado por el navegador requiere autenticación.
  - **status 404**: se refiere a que el servidor no puede encontrar el recurso solicitado por el navegador y no es posible determinar si esta ausencia es temporal o permanente.
  - **status 500**: se refiere a que la solicitud del navegador no se ha podido completar porque se ha producido un error inesperado en el servidor.

  Dentro de cada uno de esos estados hay códigos con mensajes más específicos, por ejemplo, dentro de status 201 se tienen estos tres mensajes:

```
{status: 201, code: 900, msg: '"Creado: Ok"'},
{status: 201, code: 901, msg: '"Creado:
Nok"'},
{status: 201, code: 902, msg: '"Los siguientes
objetos ya existen en la BD. Los demás se crearon
correctamente"'}
```

- ***token.js***: este fichero tiene funciones que permiten crear el token y encriptarlo.

Continuando con la definición de la estructura, hay un fichero que es muy importante de explicar, se trata de *app.js*, este fichero lo ha creado Express, por lo tanto, es parte de la estructura, pero se ha modificado para que contenga las rutas necesarias. En la siguiente Figura 18 se muestra la jerarquía de las rutas asignadas en dicho fichero de configuración.

```
//Creación de variables para comunicar con las rutas
var route_user = require('./UAT/modules/routes/route_user');
var route_api = require('./UAT/modules/routes/route_api');
var route_project = require('./UAT/modules/routes/route_project');
var route_task = require('./UAT/modules/routes/route_task');
var route_tester = require('./UAT/modules/routes/route_tester');
var route_questionary =
require('./UAT/modules/routes/route_questionary');
var route_task_done = require('./UAT/modules/routes/route_task_done');
var route_question_answered =
require('./UAT/modules/routes/route_question_answered');
var route_upload_file = require('./UAT/modules/routes/route_upload_file');
var route_compute_result =
require('./UAT/modules/routes/route_compute_result');

//Rutas
app.use('/', route_user);
app.use('/api', route_api);
app.use('/api/project/', route_project);
app.use('/api/tasks/', route_task);
app.use('/api/tester/', route_tester);
app.use('/api/questionary/', route_questionary);
app.use('/api/task_done/', route_task_done);
app.use('/api/question_answered/', route_question_answered);
app.use('/api/upload_file/', route_upload_file);
app.use('/api/compute/', route_compute_result);
```

FIGURA 18: DEFINICIÓN DE RUTAS O RECURSOS

En la figura anterior en el apartado de *Rutas,* se encuentran definidas las rutas, las cuales son encapsuladas por su funcionalidad, por ejemplo, todo lo referente a los proyectos tendrá como inicio de ruta */api/project*. Otro ejemplo, todas las operaciones relacionadas con el cálculo de los resultados tendrá como inicio de ruta */api/compute*.

Cada definición de la ruta es acompañada de un segundo parámetro el cual apunta al controlador que contiene la funcionalidad de los recursos. Por ejemplo, *app.use('/api/project', route_project)* tiene como segundo parámetro la variable *route_project*, lo que quiere decir que el contenido de esa variable no es más que un path (se puede ver en la parte superior de la figura que valor toma) que apunta a una zona del sistema, en concreto a un fichero. En conclusión, todas las rutas apuntan a los ficheros que están dentro de *UAT/modules/routes* puesto que es el encargado de gestionar las peticiones entrantes.

Continuando con la figura anterior es importante fijarse en la expresión: *app.use('/api', route_api);*. Si se observan el resto de definiciones de las rutas, estas resultan repetitivas, puesto que la mayoría de ellas parten de */api*. Es en este momento que entra en juego el *Middleware*.

Aunque ya se ha definido en el apartado de *Analisis y diseño,* un Middleware se ejecuta cada vez que se realiza una petición que contiene como raíz */api*. Por ejemplo, en la ruta */api/project*, antes de que la petición pase al recurso solicitado, se filtra en el Middleware comprobando el Token del usuario y su validez, si todo es correcto la petición pasa al siguiente nivel.

Otro fichero importante es *config.js,* que está alojado dentro del módulo databaseManagement y en él está definida la información relativa a la base de datos, como, por ejemplo, el nombre del host donde esta alojada la base de datos, el nombre de usuario de *mysql*, la contraseña y el nombre de la base de datos.

Una vez que las rutas están definidas se explica la comunicación entre los diferentes módulos, los problemas que se han encontrado durante la implementación, mediante el flujo de comunicación que genera el *iniciar sesión*.

Iniciar sesión

Con la descripción de esta funcionalidad se espera que se entienda una parte del funcionamiento de la aplicación. Todo empieza cuando el usuario hace uso de sus credenciales de inicio de sesión, acción que le permite que el sistema le devuelva un Token.

El recuso que permite realizar la identificación en el sistema es */login*, este recurso se encuentra disponible en el módulo *routes* en el fichero *router_user,* y atiende solicitudes con el verbo POST. Además, la petición que ha realizado el cliente debe contener en el cuerpo del mismo, los datos de identificación del usuario, en este caso el nombre de usuario y la contraseña.

Si los datos recibidos en la solicitud son correctos, el módulo *routes* solicita al módulo *controllers* la identificación del usuario enviando los datos y se queda a la espera de recibir una respuesta.

Antes de continuar, es importante mencionar uno de los problemas que se ha encontrado en la implementación, es el tratamiento de la sincronía en las ejecuciones de las tareas, en este proyecto se necesita que la ejecución interna de las tareas sea síncrona, es decir que las ejecuciones de las tareas deben esperar unas a otras. NodeJS ejecuta las tareas de forma asíncrona, lo que significa que, si se ejecuta una tarea el sistema no espera a la finalización de la misma y continúa ejecutando el resto de las tareas que pueda tener pendientes. Esto se debe a que trabaja con un único hilo de ejecución que es el encargado de organizar el flujo de trabajo.

La forma de solventar este inconveniente es trabajar con promesas. Una promesa es un objeto que devuelve un resultado, cuando la operación haya finalizado, este resultado puede ser el valor resultante de la operación (resolve) o un error que se ha producido durante la ejecución (reject).

En las siguientes Figura 19 y Figura 20 se puede ver la implementación necesaria para el uso de promesas. En la Figura 19 se ha implementado la función que está a la espera de recibir la respuesta, el método *then* debe de recibir como parámetro una función que será el resultado de ejecutar la función que se ve en la Figura 20, por otro lado el método *catch* recibe como parámetro el error en caso de haberse producido un error.

```
controller_user.loginUser(user, password)
   .then(function (token) {
      //Correcto: 200 - 800
      var _res = response._r(800, token);
      res.status(_res.status).send(_res);
   })
   .catch(function (error) {
      //Incorrecto: 200 - 801
      var _res = response._r(801);
      res.status(_res.status).send(_res);
   });
```

FIGURA 19: PROMESAS I

En la Figura 20 se ejecuta la función *loginUser* y como se puede ver devuelve una promesa. Aunque las promesas son una buena alternativa para administrar los procesos asíncronos, estas conllevan a utilizar demasiadas veces los métodos, *then* y *catch* lo que puede ocasionar que el código sea difícil de leer. Por este motivo se utilizan las palabras reservadas: async y await que permiten simplificar el manejo de las promesas.

Async es una palabra reservada para declarar una función que devuelve una promesa, mientras que await es usada durante el manejo de una promesa dentro de una función async. En la Figura 20 se puede ver que la función *loginUser,* además de devolver una promesa está a la espera de otra promesa de la función *findUser*. De esta forma se evita el uso de *then* y *catch* y sólo se utiliza cuando se necesite.

```
ControllerUser.loginUser = function (pUser, pPassword) {
   return new Promise(async function (resolve, reject) {
      try {
         let row = await UserModel.findUser(pUser);
         if (row.length != 1) {
                reject(1161); //el usuario no existe
         } else {
            …
            resolve(token);
         }
      } catch (error) {
         console.log(error.message);
         reject(1401)//error en el servidor
      }
   });
}
```

FIGURA 20: PROMESAS II

Continuando con el inicio de sesión, cuando el módulo *routes* ha enviado la petición al módulo *controllers*, éste comprueba que el usuario que solicita el *token* se encuentra registrado en la base de datos. Si el usuario no está registrado devuelve la petición indicando el estado de la solicitud, con anterioridad se han explicado los diferentes estados de los que dispone la aplicación en el fichero *response.js*. Si se valida la existencia del usuario en la base de datos, el sistema comprueba si la contraseña enviada en el cuerpo de la petición es correcta.

Para realizar esta comprobación en primer lugar, se obtienen los datos del usuario de la base de datos, entre ellos se encuentra el salt, que le fue asignado al momento del registro, y el hash, que es la combinación de la contraseña con el salt. En segundo lugar, a la contraseña que se ha enviado en la petición se le concatena el *salt*, para luego hacer una comparación de hashes, con la esperanza de que sean iguales, de ser así la contraseña enviada en la petición es correcta y se puede continuar, de lo contrario se devuelve la petición indicando el estado de la misma.

Una vez comprobada la contraseña y en caso de ser correcto, si el usuario no tiene asignado un *token* se genera uno y se le asigna solicitando al módulo de *databaseManagement* la actualización del token y del tiempo de conexión del usuario. El sistema solo genera *tokens* cuando se inicia sesión.

6.1.2 Acceso a los recursos y seguridad

Para acceder a los recursos del servidor, lo primero que tiene que hacer el usuario es identificarse. Una vez iniciada la sesión, el usuario pasa por una capa de seguridad, esa capa es el token y el tiempo de conexión. Una vez iniciada sesión, el servidor fija un tiempo de conexión y un *token,* cada vez que el usuario que realiza una petición el tiempo se actualiza. Pasado un tiempo de inactividad, el usuario tiene que iniciar sesión nuevamente. No se necesita mayor seguridad por el hecho de no trabajar con datos sensibles.

Token y tiempo de conexión

Un *token*, es un objeto cifrado que contiene información que generalmente es sensible, el cual se utiliza como identificador para acceder los recursos que ofrece una API. La funcionalidad para la creación del *token* y la verificación del tiempo de conexión se encuentra en el fichero *token.js* dentro de la carpeta *global_modules*. Los datos que se guardan en el token son: id y el nombre del usuario, además de un Salt y una frase que sólo se conoce en el lado del servidor, esta información se concatena y se encripta.

La frase es una palabra que está almacenada en el fichero *config.js,* que está alojado dentro del módulo databaseManagement.

La función que verifica si el token existe o no, se encuentra en el fichero *route_api*.js dentro de la carpeta *routes,* que es la misma que hace de *Middleware*. Si la función determina que todo es correcto, la petición continua el flujo de comunicación.

6.1.3 Gestión de métodos

Como se ha explicado en el apartado de arquitectura en este proyecto se hace uso de cuatro verbos y para qué se utilizan. La forma de implementar estos verbos es la que se muestra en la Figura 21. Un detalle a tener en cuenta y que llama la atención el símbolo ':?' de las peticiones *get* y *delete*, esa denotación significa que la *URL* de la petición contiene parámetros. NodeJS los pone en la *query* de la petición. A diferencia de los métodos *post* y *put,* que los parámetros están incluidos en el cuerpo de la petición.

```
route_project.get('/:?', function (req, res) {…
   if (typeof req.query.idUser === 'undefined') {
      …
   }
});

route_project.post('/', function (req, res) {…
   if (typeof req.body.idUser === 'undefined') {
      …
   }
});

route_project.put('/', function (req, res) {…
   if (typeof req.body.idUser === 'undefined') {
      …
   }
});

route_project.delete('/:?', function (req, res) {…
});
```

FIGURA 21: GESTIÓN DE MÉTODOS

La respuesta que da el servidor a una petición es la que se muestra en la Figura 22.

```
{
    "status": 200,
    "code": 800,
    "msg": "Ok",
    "body": {
        "id": 1,
        "user": "Diana",
        "token":
"8c9d5c594a9c8b2d41afe64064bdda07d966cec26c091429089c9511634”
    }
}
```

FIGURA 22: RESPUESTA DEL SERVIDOR

6.1.4 Importar datos desde un fichero

Una de las dificultades al realizar este proyecto es la de importar datos desde un fichero, en concreto al importar las tareas y las tareas realizadas, ya que se cuándo se cargan esos datos, antes de ser guardados hay que comprobar la existencia algunos datos, además de que estos datos necesitan ser guardados en diferentes tablas.

En principio se quería cargar dichos objetos de un fichero CSV que es un formato accesible para todo el mundo, pero resultaba bastante tedioso ya que para leer un fichero CSV se necesita que las cabeceras sean las mismas para todos datos. Lo cual resultaba inviable debido a que en esa información se guardan paths, y esos paths tienen distintos nodos. Para que se tenga una idea se muestra la Figura 23.

```
"Tester1";"Path";"iniciotutor";"vistazotutor";"evaluacióntutor";"rubricamemoria"
"";"Clicks";"1";"1";"1";"0"
"";"Tiempo";"19";"";"";""
"Tester2";"Path";"iniciotutor";"proyectostutor";"vistazotutor";"evaluacióntutor";
"rubricamemoria"
"";"Clicks";"1";"1";"1";"1";"0"
"";"Tiempo";"20";"";"";""
```

FIGURA 23: TAREAS REALIZADAS CSV

Las cabeceras son distintas, entonces para solventar este problema se ha cambiado el formato del fichero, para hacer uso de la funcionalidad *cargar desde fichero*, el usuario tiene que crear un fichero en formato JSON, con la siguiente estructura para las tareas (Figura 24) y tareas realizadas (Figura 25):

```
{
 "tasks": [
    {"name": "TAREA 1",
     "description": "",
     "duration": 60,
     "optimalPath": [
        [{"shortName": "C", "clickNumbers": 1},
         {"shortName": "F", "clickNumbers": 1}, {…}
        ], […]
       ],
       "alternativePath": [[{…}], […]]
    }
}
```

FIGURA 24: TAREAS JSON

```
{
 "tasks": [
   {"name": "TAREA 1",
    "testers": [
      {"name": "TESTER 1",
       "duration": 175,
       "finished": true,
       "pathDone": [
          {"shortName": "C", "clickNumbers": 1},
          {"shortName": "F", "clickNumbers": 1},
          {"shortName": "E", "clickNumbers": 1}, {…}
       ]
      }, {…}
     ]
   }, {…}
  ]
}
```


FIGURA 25: TAREAS REALIZADAS JSON

6.1.5 *Gestión de resultados de usabilidad*

Hasta el momento la información con la que se trabaja en la aplicación es sencilla de gestionar, salvo por la excepción antes descrita y porque surge otra dificultad cuando se quiere calcular los resultados de los objetivos de la usabilidad. Las fórmulas de estos caculos están descritas en el apartado de Antecedentes.

El inconveniente no son las fórmulas, ya que para calcularlas se ha hecho uso de librerías propias de JavaScript, como Math.sqrt() para la raíz cuadrada o Math.round() para redondear cifras, o se han creado funciones que calculen por ejemplo el *percentil 95, la desviación estándar, intervalos de confianza, tasas de éxito muestral y poblacional, método de wilson*. El problema radica en cómo devuelve los resultados la base de datos. Tal vez se piense que es una tarea sencilla de realizar, pero es todo lo contrario ya que se necesita obtener datos de diferentes tablas unificadas en una sola.

Entonces, se ha optado por generar una estructura de datos que sea comprensible para cuando se hagan llamadas desde el front-end, y para cuando se realicen estos cálculos. Esta nueva estructura permite tener organizadas todas las tareas y sus respectivos paths óptimos y alternativos, de una forma ordenada. De esta forma, se evitan excesivas llamadas a la base de datos. Por ello, en el controlador de las tareas y del cuestionario se hacen todas las llamadas necesarias a la base de datos. Y cuando se necesite hacer uso de esa estructura, sólo es necesario hacer la llamada al controlador respectivo.

La estructura de datos que se ha generado es muy parecida, por no decir igual a las que se muestran en la Figura 24 y en la Figura 25, salvo porque se han añadido los ids de los objetos. Los cuales resultan de mucha utilidad al momento de calcular los resultados.

6.2 Front-end

Como ya se ha comentado en el apartado de Antecedentes, para el front-end se ha decidido utilizar Angular como tecnología para implementarlo. A continuación, se describe la estructura y puesta en funcionamiento del front-end.

6.2.1 Estructura de la aplicación y puesta en funcionamiento

El IDE a utilizar para el front-end es el mismo que se utilizado en el back-end, al igual que NodeJS, como ya se tiene el entorno de desarrollo preparado, lo primero que se hace es instalar todo lo necesario para implementar el front-end. Para ello se siguen los siguientes pasos.

1. Se abre Visual Code Studio y en la terminal que tiene integrada se ejecuta el siguiente comando, para instalar Angular:

```
npm install -g @angular/cli
```

2. Una vez instalado Angular se crea la estructura del proyecto ejecutando el siguiente comando:

```
ng new uat-frontend
code .
```

3. Este último comando sirve para abrir el entorno con la estructura creada, que es la que se muestra la Figura 26.

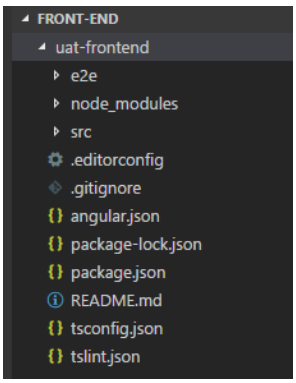


FIGURA 26: ESTRUCTURA ANGULAR BÁSICA

4. Y por último cuando se quiera ejecutar la aplicación es necesario hacer:

```
ng serve
```

En la Figura 27 se puede ver la estructura que se ha creado con Angular, es sencilla y a la vez se puede implementar de una forma muy organizada. Algunos de los ficheros que se ven en esa figura son iguales los del back-end.

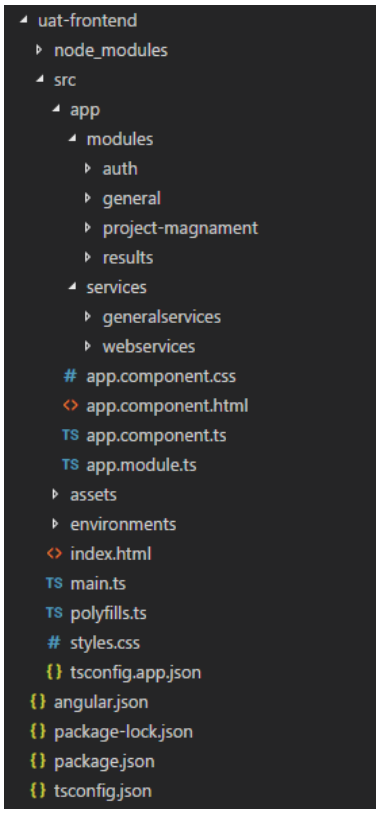


FIGURA 27: ESTRUCTURA DE ANGULAR

En la carpeta assests se colocan todos los elementos estáticos, como pueden ser: archivos png, como el logo, imágenes que se quiera mostrar en la web, etc.

De aquí en adelante se centra la atención en la carpeta src, ya que el resto de ficheros se han explicado en el back-end, los otros los crea Angular automáticamente y son necesarios para su configuración.

En la carpeta *src* se encuentran los módulos y, dentro de cada módulo los componentes; al mismo nivel que los módulos están los servicios, todos estos elementos se han explicado en la arquitectura del front-end.

La desarrolladora ha encontrado un poco de dificultad al empezar a trabajar con Angular, ya que es un estilo nuevo de desarrollo web basado en componentes. Gracias a tutoriales y a la experiencia de compañeros que trabajan con Angular, se ha adquirido la destreza necesaria, y al final no ha resultado tan complicado.

En Angular un componente está compuesto por tres ficheros necesarios para su funcionamiento, para tener una idea de cómo es la estructura de un componente, se explica con un ejemplo la creación del componente project, que representa a la página *proyectos*, en ella se muestran todos los proyectos que tiene el usuario, y también se puede crear un proyecto o al hacer click en un elemento muestra otra página en la se visualiza el contenido del mismo (recordar que un proyecto puede tener tareas, cuestionario, testers, etc). La estructura del componente es la que se muestra en la Figura 28:

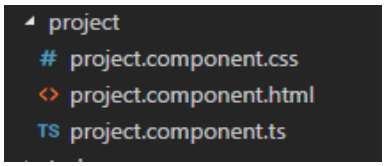


FIGURA 28: ESTRUCTURA PROYECTO

En **project.component.css** se implementa el código necesario para el estilo de la página, como puede ser el tipo de fuente, alineación, color de fondo, etc. Estos estilos sólo se aplican a este componente.

En **project.component.html** se implementa con etiquetas HTML o las que se creen con Angular, las características de la página como pueden ser la cabecera, el pie de página, y el contenido.

Se dice que se implementa con etiquetas que se han creado con Angular, porque como se ha explicado en el apartado de ANÁLISIS Y DISEÑO, un componente es una parte de la página, y en este caso se han creado la cabecera y el pie de página como componentes. Esto permite implementarlo una vez y utilizarlo todas las veces que se quiera, sin necesidad de implementarlo nuevamente. Y en el caso de hacer alguna modificación en estos, solo se hace la modificación en el componente respectivo. Entonces el HTML resultante de project es el que se muestra en la Figura 29, en ella se puede ver como se hace uso de esos componentes, simplemente basta con ponerlas como si fuesen etiquetas de HTML:

```
<app-header [nameUser]="nameUser"> </app-header> //etiqueta de la cabecera
<div class="container">
  <form id="form">
    …
  </form>
</div>
<app-footer> </app-footer>//etiqueta del pie de página
```

FIGURA 29: HTML PROJECT

En **project.component.ts** se implementa con código TypeScript, este fichero es el controlador del componente, es decir en él se implementan las funciones de carga de datos, inicialización de atributos, eventos como: onClick, doubleClick ,etc.

Continuando con el desarrollo del front-end, crear las pantallas de registro e inicio de sesión ha sido lo más sencillo, además de organizar bien la estructura de los objetos. En cuanto a crear pantallas de creación y visualización de objetos ha tenido un poco de complejidad, sobre todo porque se trabaja con tablas. Por otro lado, están las pantallas que además de tablas, muestran grafos. Como se puede intuir la complejidad del desarrollo está en esos dos elementos. En los siguientes apartados se explica lo sucedido y la solución que se ha dado.

6.2.2 Implementación de tablas

Implementar tablas con HTML y Angular no es complicado, ni tampoco el implementar funcionalidades como filtrar u ordenar los elementos. Lo que ocurre es que se genera mucho código que luego resulta tedioso al momento de corregir errores. Entonces en vista de que en este proyecto se utilizan muchas tablas de distintos tamaños, se ha decidido optimizar el código haciendo uso de una librería llamada *ag-grid* [12], que permite tener un código más limpio y legible.

[12] https://www.ag-grid.com/

Esta herramienta facilita la implementación de una tabla, además de contar con funcionalidades como filtrar y ordenar elementos, lo único que hay que hacer para tener disponibles estas dos funcionalidades es, indicar en el objeto que se quiere hacer uso de ellas mediante estos dos atributos: *enableSorting* y *enableFilter*, los cuales se pueden ver en la Figura 30 junto al resto de atributos que se utilizan en este proyecto.

```
<ag-grid-angular
  style="width: 400px; height: 400px;"
  class="ag-theme-balham"
  [enableSorting]="true"
  [enableFilter]="true"
  [rowData]="rowData"
  [columnDefs]="columnDefs"
  (rowDoubleClicked)="rowDoubleClicked($event)">
</ag-grid-angular>
```

FIGURA 30: IMPLEMENTACIÓN TABLA CON AG-GRID

El atributo en el que se definen los elementos de la cabecera es *columnDefs* y en el que se definen los elementos de las filas es *rowData*. Ambos atributos son inicializados en el fichero *component.ts*, con datos que se obtienen del servidor. Por otro lado, tampoco hay que controlar los de eventos de las filas o columnas, ya que esta librería tiene métodos disponibles para cada evento, lo único que hay que hacer es implementar la funcionalidad que se le quiera dar.

Gracias a esta librería, la implementación de las tablas se ha simplificado y se ha ahorrado mucho tiempo de desarrollo.

6.2.3 Implementación de grafos

Los grafos son elementos importantes en este proyecto, ya que permiten ver a primera vista, en qué pasos ha tenido problemas un tester al realizar una tarea. Por ello es importante mostrar un buen grafo.

Implementar un grafo con HTML y Angular significa escribir líneas y líneas de código, algo parecido que con las tablas, solo que para un grafo hay que crear funcionalidades para dibujar los nodos, arcos, la conexión que hay entre ellos, las etiquetas de cada uno de ellos, el diseño, etc. Por eso y porque se quiere optimizar el código, se ha buscado una alternativa, y es utilizar una librería llamada *vis* [13], que es compatible con Angular. Esta librería permite utilizar funcionalidades que ya tiene implementadas, solo es necesario tener claro cuáles son los nodos, como están conectados entre sí y cuál es la escala de cada uno (la escala es el grosor de los arcos o el tamaño de los grafos).

Por ejemplo, para crear un grafo, en el HTML, sólo basta con poner una línea de código, como se puede ver en la Figura 31.

```
<div id="mynetwork"></div>
```

FIGURA 31: HTML GRAFO

[13] http://visjs.org/

Aunque eso no es todo, la verdadera funcionalidad se encuentra en el fichero graphs.component.ts. Para crear un grafo, se necesitan dos arrays, uno para los nodos y otro para los arcos, una vez obtenidos los dos arrays con los datos necesarios (Figura 32), éstas se encapsulan en una variable, llamémosle *data*.

```
// array de nodos
var nodes = [
  {id: 2, label: 'Nodo 2', value: 3},
  {id: 4, label: 'Nodo 4', value: 4},
  {id: 5, label: 'Nodo 5', value: 1},
];

// array de arcos
var edges = [
   {from: 2, to: 8, value: 3, label: '87%'},
   {from: 8, to: 2, value: 3, label: '87%'},
   {from: 2, to: 9, value: 5, label: '92%'},
   {from: 2, to: 10, value: 1, label: '5%'}
];

var data = {
   nodes: nodes,
   edges: edges
}
```

FIGURA 32: ARRAYS DE NODOS Y ARCOS

Por otro lado, se tiene la posibilidad de personalizar los nodos y arcos, escogiendo la forma del nodo, el color, la escala, etc, lo mismo que para los arcos (Figura 33).

```
var options = {
   nodes: {
      shape: 'dot',
      color: 'rgb(255,168,7)'
   },
   edges: {
      arrows: {
         to: {enabled: true, scaleFactor: 1, type: 'arrow'}
      }
   }
};
```

FIGURA 33: PERSONALIZACIÓN DE NODOS Y ARCOS

Para terminar con la creación del grafo, cuando ya se tiene todo preparado hay crearlo como tal, lo cual se hace poniendo las dos líneas de código que se ven en la Figura 33.

```
// para obtener el elemento del HTML
var container = document.getElementById('mynetwork');
// Para crear el grafo
var network = new Network(container, data, options);
```

FIGURA 34: CREACIÓN DEL GRAFO

Aprovechando la ventaja que da Angular de crear componentes para utilizarlas como etiquetas, se ha decidido crear el grafo como un componente, ya que se crea un grafo por cada tarea que haya configurado el usuario. A diferencia de las tablas, que cada una tiene diferente tamaño de columnas y su contenido es distinto, los nodos tienen misma la estructura de datos, con más o menos nodos. Para tener una idea del diseño del grafo, ver la Figura 35

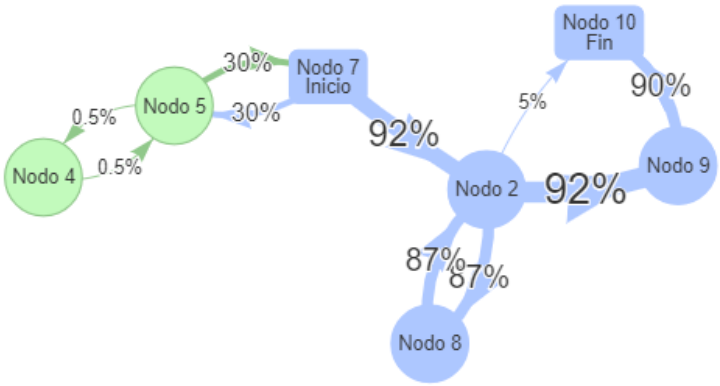


FIGURA 35: GRAFO

6.2.4 Implementación con boostrap

Boostrap es un framework que facilita el diseño de la interfaz, permite crear de forma sencilla webs de diseño adaptable, esto quiere decir, que se ajusta a cualquier dispositivo y tamaño de pantalla, aunque no sólo se ha escogido boostrap por que se ajusta a los tamaños de pantalla, sino también por el diseño que ofrece en elementos de la interfaz.

Por ejemplo, se ha hecho uso de las funcionalidades de este framework para el diseño de los modales que se lanzan al crear una tarea o cualquier objeto. Los modales son las ventanas que se muestran al hacer click en un botón, por ejemplo, el de *Crear tarea*, en este proyecto se usan para la creación de cualquier objeto, o simplemente para visualizarlos, en la Figura 36 se puede ver el ejemplo de un modal.

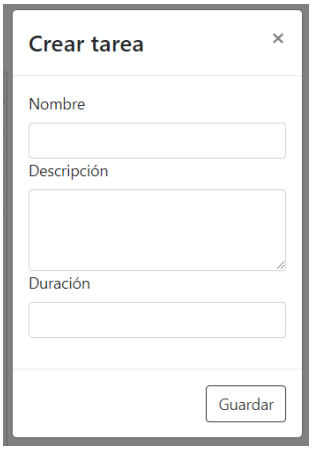


FIGURA 36: MODAL DE CREAR TAREA

Además de utilizar modales, se han utilizado estilos de diseño para los botones, mensajes de alerta, cabecera y pie de página, etc.

7. VERIFICACIÓN Y EVALUACIÓN

En todo proyecto es importante realizar pruebas para comprobar el correcto funcionamiento de la aplicación o sistema. En este proyecto se han realizado pruebas para el back-end y el front-end.

7.1 Pruebas del back-end

Debido al gran volumen de recursos y funcionalidades implementadas, para definir las pruebas del back-end se ha hecho uso de una herramienta llamada Postman, que permite realizar solicitudes, HTTP. Gracias a ella se han podido definir las pruebas y a su vez automatizarlas, lo permite comprobar el estado de todas las funcionalidades implementadas. En la Figura 37 se puede ver la interfaz de Postman con alguna de las pruebas definida. En donde en la parte izquierda, se ven alguna de las pruebas; al centro, el área de configuración de la petición; y a la derecha el resultado de la prueba.

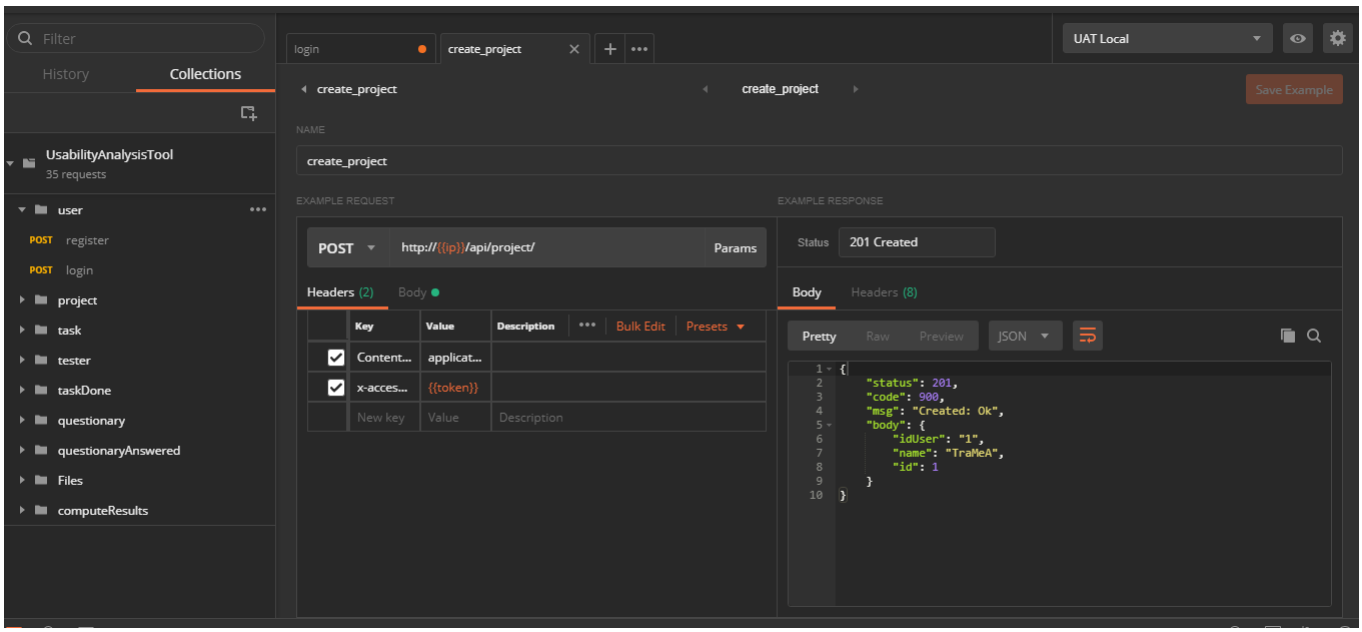


FIGURA 37: INTERFAZ DE POSTMAN

Para hacer uso de esta herramienta se han creado de dos variables globales, una llamada *ip* para guardar una parte de la URL del servidor API (aparece en la Figura 37 en la parte central superior, de color naranja) y otra llamada *token* (esta se define en la cabecera de la petición), para guardar el token que devuelve el servidor al iniciar sesión. Ambas evitan, que por un lado la desarrolladora tenga que escribir *localhost:3000* en todas las peticiones y simplemente poner *{{ip}}*, y por otro lado, que tenga que cambiar el *token* cuando realice una petición que lo requiera, lo único que tiene que hacer en este último caso es cambiar el valor de esa variable en la configuración.

A continuación, se muestran unas tablas que representan a las pruebas realizadas con Postman, además de una descripción, el resultado esperado, el resultado real y un pequeño comentario. Las pruebas están separadas por objetos, puesto que el desarrollo se ha realizado por prototipos y se han realizado a la par que estos.

7.1.1 Pruebas de inicio de registro e inicio de sesión

Estas pruebas consisten en comprobar datos del usuario, al momento de hacer un registro (Tabla 28) e iniciar sesión (Tabla 29), en las pruebas de inicio de sesión se incluyen las pruebas que se hicieron para comprobar si el *Middleware* realizaba correctamente su función.

TABLA 28: PRUEBAS REGISTRO DE USUARIO

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	Nombre de usuario existe en la BD	Mensaje: el usuario ya existe	Alta del usuario	No se tenían en cuenta los nombres del usuario con los de la BD
2	Nombre de usuario no existe en la BD	Creado en la base de datos	OK	--
3	Nombre de usuario, email o contraseña en blanco	Mensaje: error	Si el campo usuario está en blanco: error de registro. Si los campos email o contraseña en blanco: registra al usuario	El front-end comprueba los campos vacíos

TABLA 29: PRUEBAS INICIO DE SESIÓN

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	El usuario existe y la contraseña es correcta	El servidor devuelve un token	Resultado esperado	--
2	El usuario existe y la contraseña es incorrecta.	Mensaje: contraseña incorrecta	Resultado esperado	--
3	El usuario no existe o contraseña incorrecta.	Mensaje: contraseña o usuario incorrecto	OK	--
4	Usuario o contraseña en blanco	Mensaje: contraseña o usuario incorrecto	Resultado esperado	El front-end comprueba los campos vacíos
5	Inicia sesión correctamente y realiza una petición sin el token en la cabecera	Mensaje: los parámetros recibidos no son correctos	Resultado esperado	--
6	Inicia sesión correctamente y realiza una petición con el token en la cabecera y éste es incorrecto	Mensaje: el token no existe	Continua la ejecución y pasa a comprobar el tiempo de conexión	--
7	Inicia sesión correctamente, realiza una petición con el token en la cabecera y este es correcto	Continuar con la ejecución y pasar a comprobar el tiempo de conexión	Resultado esperado	--

8	Inicia sesión correctamente, realiza una petición con el token en la cabecera y éste es correcto, pero el tiempo de conexión ha expirado	Mensaje: el tiempo ha expirado	Acción: accede al servidor, a pesar de haber expirado el tiempo	No se controlaba bien la funcionalidad de trabajar con fechas y horas
9	Inicia sesión correctamente, realiza una petición con un token correcto en la cabecera y el tiempo de conexión no ha expirado	Acción: pasa a la siguiente capa para acceder a las funciones del servidor	Resultado esperado	--

En todas las pruebas que se realizan a continuación se asume que el usuario ha iniciado sesión correctamente. Y para abreviar el término base de datos se utiliza la sigla BD.

7.1.2 Pruebas de proyecto y nodos

En este apartado se comprueban el proyecto (Tabla 30) y los nodos (Tabla 31), en concreto la creación, modificación, eliminación y obtención de los mismos. Para la creación de nodos se necesitan los nombres y nombres cortos de los nodos en una lista de listas.

TABLA 30: PRUEBAS DE PROYECTO

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	Crear un proyecto con id de usuario que no existe en la BD	Mensaje: el usuario no existe	Resultado esperado	--
2	Crear un proyecto con nombre que existe en la BD e id de usuario correcto	Mensaje: existe otro objeto con el mismo nombre	Se crea el objeto	No se hacía la comprobación del nombre nuevo con los de la BD
3	Crear un proyecto con nombre que no existe en la BD e id de usuario que existe en la BD	Mensaje con datos del proyecto creado	Resultado esperado	--
4	Crear un proyecto sin nombre	Mensaje: Nok	Resultado esperado	--
5	Modificar un proyecto que existe	Mensaje: con datos del proyecto modificado	Resultado esperado	--
6	Modificar el nombre de un proyecto por otro nombre que ya existe en la BD	Mensaje: existe un objeto con el mismo nombre	Resultado esperado	--
7	Modificar un proyecto que no existe	Mensaje: el objeto que intenta modificar no existe	Resultado esperado	--

8	Eliminar un proyecto que existe	Mensaje: Ok. Y elimina el proyecto y todo lo relacionado a él	Sólo se eliminan los datos el proyecto, no elimina ningún objeto asociado a él	No se había configurado la BD
9	Eliminar un proyecto que existe	Mensaje: Ok. Y elimina el proyecto y todo lo relacionado a él	Se elimina todo menos las tareas realizadas, paths realizados, preguntas contestadas	No se habían configurado esas tablas en la BD
10	Eliminar un proyecto que no existe	Mensaje: el proyecto no existe	Resultado esperado	--
11	Obtener los proyectos de un usuario que no existe	Mensaje: el proyecto no existe	Resultado esperado	--
12	Obtener los proyectos de un usuario que existe	Mensaje: con todos los nombres de los proyectos que tiene el usuario	Resultado esperado	--
13	Obtener un proyecto que no existe	Mensaje: el objeto no existe	Resultado esperado	--
14	Obtener un proyecto que existe	Mensaje con el nombre del proyecto	Resultado esperado	--

TABLA 31: PRUEBAS DE NODOS

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	Crear nodos con algunos de los nombres o nombres cortos que están duplicados en la lista	Mensaje: hay objetos duplicados	Se crean los nodos en la BD	No se tenían en cuenta los duplicados de la lista antes de guardar en la BD
2	Crear nodos con algunos de los nombres o nombres cortos existen en la BD	Mensaje: algunos nodos ya existen en la BD	Se crean los nodos en la BD	No se tenían en cuenta los nodos que había en la BD
3	Crear nodos a partir de una lista vacía	Mensaje: Nok. Los objetos existen en la base de datos o son incorrectos	Resultado esperado	--
4	Los nodos de la lista son correctos	Mensaje: Ok. Lista de elementos creados	Resultado esperado	--
5	Modificar un nodo que no existe	Mensaje: el objeto no existe	Resultado esperado	--
6	Modificar el nombre o nombre corto con uno que ya existe	Mensaje: existe un objeto con el mismo nombre	Se hace la modificación	No se tenían en cuenta los nodos que habían en la BD

7	Modificar un nombre o nombre corto con uno que no existe	Mensaje: Ok. Con nuevos datos del nodo	Resultado esperado	--
8	Eliminar un nodo que no existe	Mensaje: el objeto no existe	Resultado esperado	--
9	Eliminar un nodo que existe	Mensaje: Ok. Se elimina el nodo y todo lo relacionado a él	Resultado esperado	--
10	Obtener nodos de un proyecto que no existe	Mensaje: Ok. El objeto no existe	Resultado esperado	--
11	Obtener nodos de un proyecto que existe	Mensaje: Ok. Con todos los datos de los todos los nodos	Resultado esperado	--

7.1.3 Pruebas de tarea y paths

En este apartado se exponen las pruebas que se han realizado para las tareas (Tabla 32) y sus respectivos paths (Tabla 33), en concreto la creación, modificación, eliminación y obtención de los mismos. Para la creación de los path se necesita una lista de nodos.

TABLA 32: PRUEBAS DE TAREAS

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	Crear una tarea con un nombre que ya existe en la BD e id del proyecto correcto	Mensaje: existe un objeto con el mismo nombre	Resultado esperado	--
2	Crear una tarea con un nombre que ya existe en la BD e id del proyecto incorrecto	Mensaje: el objeto no existe	Resultado esperado	--
3	Crear una tarea con algunos campos vacíos	Mensaje: Nok. Los objetos existen en la base de datos o son incorrectos	Creaba la tarea en la BD	Sólo se tenía en cuenta el nombre de la tarea
4	Crear una tarea con algunos campos vacíos	Mensaje: Nok. Los objetos existen en la base de datos o son incorrectos	Resultado esperado	Aunque el resultado es el esperado, esta prueba se controla también desde el front-end
5	Crear una tarea con nombre que no existe	Mensaje: Ok. Con datos de la tarea	Resultado esperado	--
6	Modificar una tarea que no existe	Mensaje: el objeto no existe	Resultado esperado	--
7	Modificar el nombre de la tarea con un nombre que ya existe	Mensaje: existe un objeto con el mismo nombre	Resultado esperado	--
8	Modificar una tarea con datos correctos	Mensaje: Ok. Con datos de la tarea	Resultado esperado	--

9	Eliminar una tarea que no existe	Mensaje: el objeto no existe	Resultado esperado	--
10	Eliminar una tarea que existe	Mensaje: Ok. Y se eliminan todos los datos asociados a la tarea, incluidos sus paths	Resultado esperado	--
11	Obtener la(s) tarea(s) de un proyecto que no existe	Mensaje: el objeto no existe	Resultado esperado	--
12	Obtener la(s) tarea(s) de un proyecto que existe	Mensaje con los datos de la(s) tarea(s)	Resultado esperado	Además de la(s) tarea(s), el mensaje contiene los posibles paths que puedan tener

TABLA 33: PRUEBAS DE PATHS

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	Crear path con algunos ids de nodos que no existen	Mensaje: algunos nodos no existen	Guardaba sólo los nodos que existían	Esto no se puede hacer, el orden de la lista es el orden del path. El usuario tiene que corregir la lista para crear el path
2	Crear path con nodos que tienen algunos de los campos vacíos	Mensaje: Nok. Los objetos existen en la base de datos o son incorrectos	Resultado esperado	Aunque el resultado es el esperado, esta prueba se controla también desde el front-end
3	Crear path con una lista vacía que no tiene nodos	Mensaje: Nok. Los objetos existen en la base de datos o son incorrectos	Resultado esperado	Aunque el resultado es el esperado, esta prueba se controla también desde el front-end
4	Crear path correcto con id de tarea que no existe	Mensaje: la tarea no existe	Resultado esperado	--
5	Crear path con los datos correctos	Mensaje con los nodos creados	Resultado esperado	--
6	Modificar path con algunos ids de nodos que no existen	Mensaje: algunos nodos no existen	Guardaba sólo los nodos que existían	Esto no se puede hacer, el orden de la lista es el orden del path. El usuario tiene que corregir la lista para crear el path
7	Modificar un path que no existe	Mensaje: el objeto no existe	Resultado esperado	--

8	Modificar un path con id de tarea que no existe	Mensaje: la tarea no existe	Resultado esperado	--
9	Modificar path con datos correctos	Mensaje con los nodos modificados	Resultado esperado	--
10	Eliminar un path que no existe	Mensaje: algunos nodos no existen	Resultado esperado	--
11	Eliminar un path que existe	Mensaje: Ok	Resultado esperado	--
12	Obtener los paths de una tarea que no existe	Mensaje: la tarea no existe	Resultado esperado	--
13	Obtener los paths de una tarea que existe	Mensaje con todos los path que tiene la tarea	Sólo devuelve el path óptimo	El resultado tiene que incluir paths óptimos y alternativos.

7.1.4 Pruebas de testers

En este apartado se exponen las pruebas que se han realizado para los testers (Tabla 34), en concreto la creación, modificación, eliminación y obtención de los mismos. Para crear testers, es necesario hacerlo con una lista que contenga sus nombres

TABLA 34: PRUEBAS DE TESTERS

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	Crear testers y que algunos de los nombres sean iguales	Mensaje: hay objetos duplicados	Creaba los testers en la BD	No se tenía en cuenta los nombres duplicados de la lista
2	Crear testers a partir de una lista vacía	Mensaje: Nok. Los objetos existen en la base de datos o son incorrectos	Resultado esperado	Aunque el resultado es el esperado, esta prueba se controla también desde el front-end
3	Crear testers con datos correctos	Mensaje con los testers creados	Resultado esperado	--
4	Modificar el nombre de un tester por otro que ya existe	Mensaje: existe un objeto con el mismo nombre	Modificaba el nombre del tester en la BD	No se tenían en cuenta los nombres de la BD
5	Modificar un tester que no existe	Mensaje: el tester no existe	Resultado esperado	--
6	Modificar un tester que existe	Mensaje: con los datos del tester	Resultado esperado	--
7	Eliminar un tester que no existe	Mensaje: el tester no existe	Resultado esperado	--
8	Eliminar un tester que existe	Mensaje: Ok. Se eliminan todos los datos relacionados con el tester.	Resultado esperado	--

9	Obtener testers de un proyecto que no existe	Mensaje: el proyecto no existe	Resultado esperado	--
10	Obtener testers de un proyecto que existe	Mensaje con los testers del proyecto	Resultado esperado	--

7.1.5 Pruebas de cuestionario

En este apartado se exponen las pruebas que se han realizado para el cuestionario (Tabla 35) y las preguntas (Tabla 36), en concreto la creación, modificación, eliminación y obtención de los mismos.

TABLA 35: PRUEBAS DEL CUESTIONARIO

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	Crear un cuestionario con id de proyecto que no existe	Mensaje: el proyecto no existe	Resultado esperado	--
2	Crear un cuestionario que ya existe	Mensaje: existe un objeto con el mismo nombre	Crea el cuestionario en la BD	No se tenían en cuenta los cuestionarios de la BD
3	Crear un cuestionario con datos correctos	Mensaje con los datos del cuestionario	Resultado esperado	--
4	Modificar el nombre del cuestionario con uno que ya existe	Mensaje: existe un objeto con el mismo nombre	Modificaba el nombre del cuestionario	No se tenían en cuenta los nombres de la BD
5	Modificar un cuestionario que no existe	Mensaje: el cuestionario no existe	Resultado esperado	--
6	Modificar un cuestionario con id de proyecto que no existe	Mensaje: el proyecto no existe	Resultado esperado	--
7	Eliminar cuestionario que no existe	Mensaje: el cuestionario no existe	Resultado esperado	--
8	Eliminar cuestionario que existe	Mensaje: Ok. Elimina todos los datos asociados a él	Resultado esperado	--
9	Obtener cuestionarios de un proyecto que no existe	Mensaje: el proyecto no existe	Resultado esperado	--
10	Obtener cuestionarios de un proyecto que existe	Mensaje con los datos de los cuestionarios	Resultado esperado	--

TABLA 36: PRUEBAS DE LAS PREGUNTAS

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	Crear preguntas y que algunos de los nombres sean iguales	Mensaje: hay objetos duplicados	Se crean las preguntas en la BD	No se tenían en cuenta las preguntas de la BD
2	Crear preguntas a partir de una lista vacía	Mensaje: Nok. Los objetos existen en la base de datos o son incorrectos	Resultado esperado	Aunque el resultado es el esperado, esta prueba se controla también desde el front-end
3	Crear preguntas a partir de una lista correcta	Mensaje con las preguntas creadas	Resultado esperado	--
4	Modificar una pregunta que no existe	Mensaje: la pregunta no existe	Resultado esperado	--
5	Modificar una pregunta de un cuestionario que no existe	Mensaje: el cuestionario no existe	Resultado esperado	--
6	Modificar una pregunta que existe	Mensaje con la pregunta modificada	Resultado esperado	No se tiene en cuenta si hay preguntas iguales.
7	Eliminar una pregunta que no existe	Mensaje: la pregunta no existe	Resultado esperado	
8	Eliminar una pregunta de un cuestionario que no existe	Mensaje: el cuestionario no existe	Resultado esperado	--
9	Obtener las preguntas de un cuestionario que no existe	Mensaje: el cuestionario no existe	Resultado esperado	--
10	Obtener las preguntas de un cuestionario que existe	Mensaje con las preguntas del cuestionario	Resultado esperado	--

7.1.6 Pruebas de tareas realizadas

Las tareas realizadas son muy similares a las tareas por el hecho de tener paths, en el caso de las tareas realizadas tienen paths realizados. Estas pruebas que se dividen en dos apartados: tareas realizadas (Tabla 37) y paths realizados (Tabla 38), en ambos se ven funcionalidades de creación, modificación, eliminación y obtención de los mismos

TABLA 37: PRUEBAS DE TAREAS REALIZADAS

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	Crear una tarea realizada en la que el tester ya ha resulto esa tarea	Mensaje: el tester ya ha realizado esa tarea	Guardaba la misma tarea realizada por ese tester	Solo se admite que un tester responda una pregunta una sola vez
2	Crear una tarea realizada en la que el tester o la tarea no existen	Mensaje: el tester o la tarea no existe	Guardaba la tarea realizada sin tener en cuenta la existencia de la tarea o el tester	El mensaje se ha unificado. Son dos casos de prueba distintas. Por otro lado, no se tenía en cuenta existencia de la tarea y el tester
3	Crear una tarea con los datos correctos	Mensaje con las tareas realizadas	Resultado esperado	--
4	Modificar una tarea realizada en la que el tester o la tarea no existen	Mensaje: el tester o la tarea no existe	Modificaba la tarea realizada sin tener en cuenta la existencia de la tarea o el tester	El mensaje se ha unificado. Son dos casos de prueba distintas. Por otro lado, no se tenía en cuenta existencia de la tarea y el tester
5	Modificar una tarea realizada con datos correctos	Mensaje con las tareas realizadas	Resultado esperado	--
6	Eliminar una tarea realizada que no existe	Mensaje: la tarea no existe	Resultado esperado	--
7	Eliminar una tarea realizada que existe	Mensaje: Ok. Solo se elimina la tarea	Resultado esperado	--
8	Obtener las tareas realizadas de una tarea que no existe	Mensaje: la tarea no existe	Resultado esperado	--
9	Obtener las tareas realizadas de una tarea	Mensaje con las tareas realizadas y sus paths correspondientes	El mensaje sólo contenía las tareas realizadas sin los paths	--
10	Obtener las tareas realizadas de un proyecto que no existe	Mensaje: el proyecto no existe	Resultado esperado	--
11	Obtener las tareas realizadas de un proyecto	Mensaje con las tareas realizadas y sus paths correspondientes	El mensaje sólo contenía las tareas realizadas sin los paths	--

TABLA 38: PRUEBAS DE PATHS REALIZADOS

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	Crear un path en la que la tarea, el tester o algunos nodos no existen	Mensaje: algunos nodos, el tester o la tarea no existe	Resultado esperado	El mensaje se ha unificado. Son dos casos de prueba distintas. Por otro lado, no se tenían en cuenta la existencia de la tarea, el tester o los nodos
2	Crear un path con los datos correctos y la lista del path vacía	Mensaje: Nok. Los objetos existen en la base de datos o son incorrectos	Resultado esperado	Aunque el resultado es el esperado, esta prueba se controla también desde el front-end
3	Crear un path con datos correctos y lista de nodos correcta	Mensaje con los paths creados	Resultado esperado	--
4	Modificar un path en la que la tarea, el tester o algunos nodos no existen	Mensaje: algunos nodos, el tester o la tarea no existe	Resultado esperado	El mensaje se ha unificado. Son dos casos de prueba distintas
5	Modificar un path con datos correctos y que la lista del path esté vacía	Mensaje: Nok. Los objetos existen en la base de datos o son incorrectos	Resultado esperado	Aunque el resultado es el esperado, esta prueba se controla también desde el front-end
6	Modificar un path con datos correctos	Mensaje con el path modificado	Resultado esperado	--
7	Eliminar un path que no existe	Mensaje: el path no existe	Resultado esperado	--
8	Eliminar un path que existe	Mensaje: Ok. Sólo se elimina el path	Resultado esperado	--

7.1.7 Pruebas de cuestionario respondido

En este apartado se exponen las pruebas que se han realizado para las respuestas del cuestionario (Tabla 39), en concreto la creación, modificación, eliminación y obtención de las mismas. Las respuestas del cuestionario, son todas las respuestas que han dado los testers a todas las preguntas del mismo.

TABLA 39: PRUEBAS DE CUESTIONARIO RESPONDIDO

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	Crear respuestas en las que la pregunta o el tester no existen	Mensaje: la pregunta o el tester no existe	Resultado esperado	El mensaje se ha unificado. Son dos casos de prueba distintas
2	Crear respuestas con datos correctos	Mensaje con las respuestas creadas	Resultado esperado	--
3	Modificar una respuesta en la que la tarea o el tester no existen	Mensaje: la pregunta o el tester no existe	Resultado esperado	El mensaje se ha unificado. Son dos casos de prueba distintas
4	Modificar la respuesta con datos correctos	Mensaje con la respuesta modificada	Resultado esperado	--
5	Eliminar una respuesta en la que la tarea o el tester no existen	Mensaje: la tarea o el tester no existe	Resultado esperado	El mensaje se ha unificado. Son dos casos de prueba distintas
6	Eliminar la respuesta con datos correctos	Mensaje: Ok. Sólo se elimina la respuesta	Resultado esperado	--
7	Obtener las respuestas de un cuestionario que no existe	Mensaje: el cuestionario no existe	Resultado esperado	--
8	Obtener las respuestas de un cuestionario que existe	Mensaje con las respuestas del cuestionario	Resultado esperado	--

7.1.8 Pruebas de cargar fichero

En estas pruebas de cargar fichero, se han considerado dos pruebas distintas, una para el formato CSV (Tabla 40) que carga los testers, nodos, cuestionario y respuestas del cuestionario; y otra para el formato JSON (Tabla 41), que carga datos de las tareas y tareas realizadas. En ambos casos puede ocurrir que algunos datos no sean correctos, entonces, el usuario tiene la opción de elegir si crear solo aquellos objetos que son correctos o no crear nada. Por defecto esta opción esta deshabilitada, es decir si hay datos incorrectos, no se crea nada. La variable que se va a utilizar para distinguir esta opción es: *guardarCorrectos* = *sí* (se guardan sólo los datos correctos) y *guardarCorrectos* = *no* (no se guarda nada).

Además de elegir si guardar sólo los correctos, tiene que indicar que tipo de datos quiere crear, por ejemplo, si quiere crear nodos tiene que indicarlo en los parámetros del cuerpo de la petición.

TABLA 40: PRUEBAS DE CARGAR FICHERO CSV

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	Cargar un fichero con extensión distinta a CSV	Mensaje: la extensión del fichero es incorrecta	Resultado esperado	--
2	Cargar un fichero cualquiera para leer los datos	Conseguir datos legibles	Los datos que se cargaban contenían caracteres extraños	Sucedía eso porque los ficheros no se guardaban en el formato correcto. La forma correcta de guardar el fichero es UTF-8 (delimitado por comas)
3	Cargar un fichero de nodos o cualquier otro tipo de datos y no tenga la estructura adecuada.	Mensaje: error al importar nodos (o el tipo de datos que sea)	Resultado esperado	Si algunos ficheros no tienen la estructura adecuada para el tipo de datos que se quiera cargar. No se pueden leer los datos del mismo
4	Cargar un fichero en blanco y decir que el tipo de datos es nodo, o cualquiera de los tipos de datos que se puede cargar con CSV	Mensaje: error al importar nodos (o el tipo de datos que sea)	Resultado esperado	--
5	Cargar fichero de testers y que algunos nombres estén duplicados, o existan en la BD. El id del proyecto existe. *guardarCorrectos = sí*	Mensaje con los datos que no están duplicados y que han sido guardados en la BD	Guardaba todos los testers duplicados y no duplicados.	Se comprobaba si había duplicados en el fichero, pero no se tenían en cuenta los nombres de la BD.
6	Cargar fichero de testers y que algunos nombres estén duplicados, o existan en la BD. El id del proyecto existe. *guardarCorrectos = no*	Mensaje con los datos que están duplicados en el fichero y en la BD	Resultado esperado	--

7	Cargar fichero de testers con datos correctos. El id del proyecto no existe. *guardarCorrectos = sí*	Mensaje: el proyecto no existe	Resultado esperado	Aunque tenga *guardarCorrectos = sí,* no se guarda nada, porque lo primero que se comprueba es el proyecto
8	Cargar fichero de nodos y que algunos nombres o nombres cortos estén duplicados, o existan en la BD. El id del proyecto existe. *guardarCorrectos = sí*	Mensaje con los datos que no están duplicados y han sido guardados en la BD	El mensaje contenía los nodos que tenían nombres diferentes pero algunos nombres cortos duplicados	Sólo se tenían en cuenta los nombres de los nodos, los nombres cortos no se comprobaban. Y es necesario tener nombres cortos diferentes porque sirven de códigos.
9	Cargar fichero de nodos y que algunos nombres o nombres cortos estén duplicados, o existan en la BD. El id del proyecto existe. *guardarCorrectos = no*	Mensaje con los datos que están duplicados en el fichero y en la BD	Resultado esperado	--
10	Cargar fichero de nodos con datos correctos. El id del proyecto no existe. *guardarCorrectos = sí*	Mensaje: el proyecto no existe	Resultado esperado	Aunque tenga *guardarCorrectos = sí,* no se guarda nada, porque lo primero que se comprueba es el proyecto
11	Cargar el fichero del cuestionario y que exista un cuestionario con el mismo nombre en la BD. El id del proyecto existe. *guardarCorrectos = sí*	Mensaje: existe un objeto con el mismo nombre	Resultado esperado	Aunque tenga *guardarCorrectos = sí,* no se guarda nada, porque lo primero que se comprueba es el cuestionario
12	Cargar el fichero del cuestionario con datos correctos. El id del proyecto no existe. *guardarCorrectos = no*	Mensaje: el proyecto no existe	Resultado esperado	--

13	Cargar fichero de respuestas del cuestionario y que algunas respuestas estén duplicadas, es decir un tester haya respondido dos veces a la misma pregunta. El id del proyecto existe. *guardarCorrectos = sí*	Mensaje con las respuestas que se hayan guardado correctamente en una lista y en otra las que no	Resultado esperado	--
14	Cargar fichero de respuestas del cuestionario y que la pregunta o el tester no exista. El id del proyecto existe. *guardarCorrectos = sí*	Mensaje con las respuestas que se hayan guardado correctamente en una lista y en otra las que no	Mensaje: el tester o la tarea no existe	Se ha modificado para guardar sólo los datos correctos. Si no existe el tester o la tarea se considera incorrecto y no se guarda.
15	Cargar fichero de respuestas del cuestionario y que la pregunta o el tester no exista, o existan respuestas duplicadas. El id del proyecto existe. *guardarCorrectos = no*	Mensaje con las respuestas incorrectas	Resultado esperado	--
16	Cargar fichero de respuestas con datos correctos y que el id del proyecto no exista. El id del proyecto no existe. *guardarCorrectos = sí*	Mensaje: el proyecto no existe	Resultado esperado	Aunque tenga *guardarCorrectos = sí,* no se guarda nada, porque lo primero que se comprueba es el proyecto

En la siguiente tabla se consideran datos incorrectos, aquellos datos en los que no existan las tareas o los tester, o haya elementos duplicados.

TABLA 41: PRUEBAS DE CARGAR FICHERO JSON

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	Cargar un fichero con extensión distinta a JSON	Mensaje: la extensión del fichero es incorrecta	Resultado esperado	--
2	Cargar un fichero de tareas o tareas realizadas y no tenga la estructura adecuada.	Mensaje: error al importar tareas o tareas realizadas	Resultado esperado	Si algunos ficheros no tienen la estructura adecuada para el tipo de datos que se quiera cargar. No se pueden leer los datos del mismo
3	Cargar un fichero en blanco y decir que el tipo de datos es tareas o tareas realizadas	Mensaje: error al importar tareas o tareas realizadas	Resultado esperado	--
4	Cargar fichero de tareas, o tareas realizadas con datos correctos. El id del proyecto no existe *guardarCorrectos = sí*	Mensaje: el proyecto no existe	Resultado esperado	Aunque tenga *guardarCorrectos = sí,* no se guarda nada, porque lo primero que se comprueba es el proyecto
5	Cargar fichero de tareas, o tareas realizadas con datos incorrectos. El id del proyecto existe *guardarCorrectos = sí*	Mensaje con las tareas o tareas realizadas que se hayan guardado correctamente en una lista y en otra las que no	En el caso de las tareas realizadas se guardaban los datos, aunque no existieran el tester o la tarea. Y en el caso de la tarea se guardaban, aunque no existieran los nodos	Esto es porque no se controlaban la existencia de esos datos en la BD. Solo se controlaban los duplicados.
6	Cargar fichero de tareas, o tareas realizadas con datos incorrectos. El id del proyecto existe *guardarCorrectos = no*	Mensaje con las tareas o tareas realizadas incorrectas	Resultado esperado	--
7	Cargar fichero de tareas, o tareas realizadas con datos correctos y el id del proyecto existe *guardarCorrectos = no*	Mensaje con las tareas o tareas realizadas que se hayan guardado correctamente en una lista y lista de incorrectas tiene que estar vacía	Resultado esperado	--

7.1.9 Pruebas de calcular resultados

Para realizar las pruebas de resultados, hace falta que el usuario indique el nivel de significación para los intervalos de confianza, por ello, para estas pruebas se ha considerado el nivel de significación = 0.05, que proporciona intervalos con un nivel de confianza al 95%. Para comprobar la veracidad de las pruebas se han comparado los resultados obtenidos, con los resultados del estudio de investigación de (Villamañe, 2016), que son los que se muestran en el resultado esperado. Para resumir un poco las respuestas se tienen en cuenta los resultados obtenidos de la tarea *CPUo_A* y el tester *Sujeto 1* de dicho estudio.

Por otro lado, los resultados del grafo para las pruebas de eficiencia no se han comparado con los resultados del estudio, ya que el resultado del estudio es gráfico y el back-end no tiene capacidad de dibujar. Por ello, el grafo se representa a través de dos listas una para los nodos y otra para los arcos, ambos incluyen la escala correspondiente, se ha decido tener en cuenta valores de 0 a 100, para dar los resultados de forma general.

Si se da el caso en el que el usuario quiere calcular los resultados, y algunos datos como tareas, o testers, o nodos, no se han creado, el sistema se muestra un error indicando que faltan datos de configuración para calcular. Por otro lado, no se tiene en cuenta si todos los testers han realizado las tareas o respondido el cuestionario. Es decir, si se tienen 12 testers y sólo se han introducido resultados de 5, el sistema calcula los resultados con ésos 5 testers.

Las pruebas de para calcular la efectividad se muestran en la Tabla 42, las pruebas para la eficiencia en la Tabla 43 y por último para la satisfacción en la Tabla 44.

TABLA 42: PRUEBAS DE CALCULAR EFECTIVIDAD

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	Calcular la tasa de éxito muestral (Ecuación 1)	Tasa de éxito muestral = 0.917	Resultado esperado	
2	Calcular la estimación de éxito poblacional con Wilson (Ecuación 2)	--	--	En este caso no se aplica esta ecuación porque la tasa de éxito muestral es mayor a 0.9, por lo tanto, se aplica Laplace
3	Calcular la estimación de éxito poblacional con Laplace (Ecuación 3)	Tasa de éxito poblacional = 0.857	Resultado esperado	--
4	Calcular intervalos de confianza (Ecuación 4)	Intervalo de confianza: [0.625 - 0.999]	Intervalos de confianza: [0.625 - 1]	En este caso para esta tarea, da el resultado esperado, pero para las otras

				no, y es porque no se tenía en cuenta que cuando el número de testers que realizaron correctamente la tarea es igual a al número de testers total o que cuando el número de testers que realizaron correctamente la tarea es igual a 0, se tiene en cuenta otro nivel de significación.
5	Calcular el porcentaje de tareas completadas éxito por cada tester	Porcentaje d tareas completadas con éxito = 100%	Resultado esperado	--
6	Calcular el nivel de finalización de los testers	Para la tarea *CPUo_A = Con problemas*	Resultado esperado	--
7	Calcular la tasa de éxito muestral (Ecuación 1), teniendo en cuenta el nivel de finalización de los testers	Tasa de éxito muestral = 0.33	Resultado esperado	--
8	Calcular la estimación de éxito poblacional con Wilson (Ecuación 2), teniendo en cuenta el nivel de finalización de los testers	Tasa de éxito poblacional = 0.374	El resultado era distinto	Pasaba lo mismo que en caso de prueba 4 de esta tabla
9	Calcular la estimación de éxito poblacional con Laplace (Ecuación 3), teniendo en cuenta el nivel de finalización de los testers	--	--	En este caso no se aplica esta ecuación porque la tasa de éxito muestral es menor a 0.5, por lo tanto, se aplica Wilson
10	Calcular la estimación de éxito poblacional en la que la tasa de éxito muestral esté entre 0.5 y 0.9	0.583	Resultado esperado	En este caso se ha realizado esta prueba con la tarea *CPU1_A*, que cumple con esos requisitos

11	Calcular intervalos de confianza (Ecuación 4), teniendo en cuenta el nivel de finalización de los testers	Intervalo de confianza: [0.136 – 0.612]	El resultado era distinto	Pasaba lo mismo que en caso de prueba 4 de esta tabla
12	Calcular el porcentaje de tareas completadas con sin problemas por cada tester, teniendo en cuenta el nivel de finalización de los testers	Porcentaje d tareas completadas sin problemas = 50%	El resultado era distinto	Porque al momento de crear las tareas realizadas los datos del estudio no se correspondían con los de la BD

TABLA 43: PRUEBAS DE CALCULAR EFICIENCIA

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	Calcular el percentil 95	Percentil 95 = 187.25	Resultado esperado	Aunque para esta tarea el resultado coincide, no sucede lo mismo con el resto de tareas, las cifras difieren un poco debido a que la fórmula utilizada en este proyecto es la que se utiliza en Excel para el percentil. Se ha hablado con el cliente sobre esta diferencia y está de acuerdo en utilizar dicha fórmula
2	Calcular la media geométrica, sin tener en cuenta el nivel de pérdida (Ecuación 5)	Media geométrica (G) = 148	Resultado esperado	--
3	Calcular los intervalos de confianza, sin tener en cuenta el nivel de pérdida (Ecuación 6, Ecuación 7)	Intervalo de confianza: [118 – 186]	Las cifras son otras	No se calculaba bien la desviación estándar
4	Calcular el nivel de pérdida de los testers (Ecuación 8)	Nivel de pérdida del tester = 0.15	Las cifras son otras	Porque no se calculaba bien el

				valor de N (Ecuación 8), que es el número de páginas distintas que ha visitado el tester para resolver la tarea. Había que tener en cuenta que en el path óptimo o alternativo, los nodos se pueden repetir y se tienen que considerar como páginas distintas
5	Calcular la media, teniendo en cuenta el nivel de pérdida (Ecuación 5)	Media geométrica (G) = 0.13	Las cifras son otras	Debido a lo ocurrido en el caso de prueba 4 de esta tabla los resultados no son lo que se esperaba
6	Calcular los intervalos de confianza, teniendo en cuenta el nivel de pérdida (Ecuación 6, Ecuación 7)	Intervalo de confianza: [0.054 – 0.215]	Las cifras son otras	Debido a lo ocurrido en el caso de prueba 4 de esta tabla los resultados no son lo que se esperaba
7	Calcular nodos y arcos del grafo	Una lista con nodos y otra con arcos, ambos con su respectiva escala	Resultado esperado	--

TABLA 44: PRUEBAS DE CALCULAR SATISFACCIÓN

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	Calcular el nivel de satisfacción de cada tester (Ecuación 9)	Puntuación del tester = 92.5	Resultado esperado	Mal aplicada la Ecuación 9
2	Calcular la media aritmética de la satisfacción de los testers	Media aritmética = 80.21	Resultado esperado	--
3	Calcular intervalo de confianza (Ecuación 6)	Intervalo de confianza: [70.096 – 89.43]	Resultado esperado	Mal calculada la desviación estándar

7.2 Pruebas del front-end

Se han realizado las pruebas del front-end, partiendo de los casos de uso, para comprobar su correcto funcionamiento, se han diseñado pruebas unitarias realizadas directamente sobre la interfaz.

Las páginas de la interfaz sobre todo las vistas de crear, importar, modificar o eliminar elementos, son muy similares para todos los objetos, por lo tanto, se pueden ver pruebas repetidas, pero con distintos objetos. Esto ha servido de ayuda para tratar los datos por igual. Y en caso de identificar una nueva prueba, esta se ha aplicado al resto de objetos.

Para mostrar los errores en la mayoría de ocasiones se muestra un *alert,* que es un cuadro de alerta para informar al usuario de cualquier eventualidad. Por otro lado, también se utilizan mucho los modales, que se describieron en el apartado de desarrollo del front-end.

7.2.1 Pruebas de registro, inicio y cierre de sesión

En este apartado se tratan las pruebas para las pantallas de registro de usuario (Tabla 45), para la pantalla de inicio de sesión (Tabla 46) y para cerrar la sesión (Tabla 47). La página de inicio de sesión es la primera que se muestra en la aplicación, es decir es la página principal. Para visualizar la página de registro es necesario hacer click en el enlace *Registro* de la página principal. Y por último para cerrar sesión lo único que hay que hacer es: hacer click en el botón *Cerrar sesión* que se tiene disponible una vez iniciada la sesión correctamente

TABLA 45: PRUEBAS PÁGINA REGISTRO DE USUARIO

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	Algunos campos requeridos en blanco y pulsa en el botón *Registrarse*	Mensaje: tienes que completar todos los campos requeridos	No muestra mensaje	Error al configurar el alert que muestra el mensaje
2	Si ocurre algún problema por parte del back-end	Mensaje con el error que haya podido ocurrir	Ok	--
3	Se registra al usuario correctamente	Redirigir a la página inicial que es *Iniciar Sesión*	Ok	--

TABLA 46: PRUEBAS PÁGINA INICIO DE SESIÓN

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	El nombre de usuario o contraseña en blanco y pulsa en el botón *Iniciar sesión*	Mensaje: tienes que completar todos los campos	Si nombre de usuario en blanco: no hace nada. Si contraseña en blanco: contraseña incorrecta	No se hacía bien la comprobación del formulario
2	Cualquier error por parte del back-end, como pueden ser: el servidor está caído, el usuario no existe, o la contraseña es incorrecta, etc.	Mensaje indicando de que error se trata	En algunos casos no muestra el error correspondiente	No se controlaban todos los errores
3	Inicia sesión correctamente	Accede a la página de proyectos y se muestran los proyectos que haya podido crear	OK	--

TABLA 47: PRUEBAS CIERRE DE SESIÓN

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	El usuario pulsa en *Cerrar sesión*	Redirigir a la página principal y eliminar los datos de la sesión	Resultado esperado	--

7.2.2 Página proyecto y menú

Estas pruebas se han realizado en la página de proyectos y la página de menú del mismo (Tabla 48), también se tienen en cuenta los modales que pudiera tener. En las pruebas que se realizan se indica cómo acceder a ambos.

TABLA 48: PRUEBAS PÁGINA PROYECTOS Y MENÚ

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	Hace doble click en cualquiera de los proyectos que tiene en la tabla	Redirigir a la página del menú principal del proyecto	Resultado esperado	--
2	Hace click en el botón *Crear proyecto*	Mostrar modal	No se mostraba nada	Error de uso del modal
3	Hace click en el botón *Crear proyecto*. Se muestra el modal, rellena los campos y pulsa en *Guardar*	El nuevo proyecto tiene que añadirse a la tabla	No se añadía a la tabla, pero sí a la BD	No se controlaba la función de añadir nueva fila

4	En la página del menú, hace click en cualquiera de los elementos (configuración, testers, etc.)	Redirigir a la página correspondiente y mostrar sus respectivas tablas. En el caso de los resultados muestra el menú de resultados	En algunos casos no redirigía a la página correspondiente. En los casos en los que sí, muestran las tablas	Error al poner el enlace de la página
5	Hace click en el icono de modificar en alguno de los elementos. Se muestra un modal con los datos del proyecto. Modifica algún dato y hace click en *Guardar*	En la lista se muestra el proyecto modificado, y en la BD tiene que estar modificado	Resultado esperado	--
6	Hace click en el icono de eliminar en alguno de los elementos. Se muestra un modal preguntando *si realmente desea borrar ese ese proyecto*. Hace click en *Aceptar*	El proyecto eliminado no se tiene que mostrar en la lista, y se tiene que eliminar de la BD, junto a todos los datos asociados a él	Se eliminan de la BD, pero no de la lista	Error al hacer uso de la funcionalidad de eliminar fila

7.2.3 Página configurar proyecto

La configuración del proyecto, en este caso son los nodos que se utilizarán durante todo el proyecto, en este apartado se realizan las pruebas necesarias para comprobar la funcionalidad de los elementos de la interfaz, como pueden ser modales, tablas, botones, etc. Las pruebas realizadas para la página de configuración se describen en la Tabla 49. Para acceder a esta página es necesario hacer click en *Configurar* de la página menú del proyecto.

TABLA 49: PRUEBAS PÁGINA CONFIGURACIÓN

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	Hace click en el botón *Crear nodo*	Mostrar modal	Resultado esperado	--
2	Hace click en el botón *Crear nodo*. Se muestra el modal, rellena los campos y pulsa en *Guardar*	El nodo nuevo tiene que añadirse a la tabla y a la BD	Resultado esperado	--
3	Hace click en el botón *Importar nodos*. Se muestra el modal y hace click en *Seleccionar fichero*	Abrir el explorador de ficheros del ordenador	No mostraba el explorador	Mal implementado el input que abre el explorador

4	En el modal de importar hace click en cargar fichero. El fichero contiene nodos repetidos	Mostrar aviso indicando que hay nodos repetidos y si quiere guardar sólo los correctos	Se muestra el aviso y puede hacer click en *SÍ* o *NO*	--
5	En el aviso que se muestra al importar los nodos, hace click en *SÍ*	Actualizar la lista de nodos, con los nodos nuevos	Resultado esperado	--
6	En el aviso que se muestra al importar los nodos, hace click en el botón *NO*	La lista no se modifica	Resultado esperado	--
7	Hace click en el icono de modificar en alguno de los elementos. Se muestra un modal con los datos del nodo. Modifica algún dato y hace click en *Guardar*	En la lista se muestra el nodo modificado, y en la BD tiene que estar modificado	Resultado esperado	--
8	Hace click en el icono de eliminar en alguno de los elementos. Se muestra un modal preguntando *si realmente desea borrar ese nodo*. Hace click en *Aceptar*	El nodo eliminado no se tiene que mostrar en la lista, y se tiene que eliminar de la BD	Se elimina de la BD, pero no de la lista	Error al hacer uso de la funcionalidad de eliminar fila

7.2.4 Página testers

Las pruebas realizadas para la página de testers son las que se describen en la Tabla 50. Para acceder a esta página es necesario hacer click en *Testers* de la página menú del proyecto.

TABLA 50: PRUEBAS PÁGINA TESTERS

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	Hace click en el botón *Crear tester*	Mostrar modal	Resultado esperado	--
2	Hace click en el botón *Crear tester*. Se muestra el modal, rellena los campos y pulsa en *Guardar*	El nuevo tester se tiene que mostrar en la tabla y a la BD	Resultado esperado	--
3	Hace click en el botón *Importar testers.* Se muestra el modal y hace click en *Seleccionar fichero*	Abrir el explorador de ficheros del ordenador	Resultado esperado	--
4	En el modal de importar hace click en cargar	Mostrar aviso indicando que hay tester repetidos y si	Se muestra el aviso y puede hacer click en *SÍ* o *NO*	--

	fichero. El fichero contiene nodos repetidos	quiere guardar sólo los correctos		
5	En el aviso que se muestra al importar los testers, hace click en *SÍ*	Actualizar la lista de testers, con los nodos nuevos	Resultado esperado	--
6	En el aviso que se muestra al importar los testers, hace click en el botón *NO*	La lista no se modifica	Resultado esperado	--
7	Hace click en el icono de modificar en alguno de los elementos. Se muestra un modal con los datos del tester. Modifica algún dato y hace click en *Guardar*	En la lista se muestra el tester modificado, y en la BD tiene que estar modificado	No se muestra el modal	Faltaba configurar el evento que controla esa acción. Al corregir este error modifica correctamente
8	Hace click en el icono de eliminar en alguno de los elementos. Se muestra un modal preguntando *si realmente desea borrar ese tester*. Hace click en *Aceptar*	El tester eliminado no se tiene que mostrar en la lista y se tiene que eliminar de la BD	Resultado esperado	--

7.2.5 Página cuestionarios y preguntas

Las pruebas realizadas para la página de cuestionarios y para la página preguntas, son las que se describen en la Tabla 51. Para acceder a la página cuestionarios es necesario hacer click en Cuestionario de la página menú del proyecto. Y para acceder a la página preguntas, se explica en las pruebas.

TABLA 51: PRUEBAS PÁGINA CUESTIONARIOS Y PREGUNTAS

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	Hace click en *Crear cuestionario*	Mostrar modal	Resultado esperado	--
2	Hace click en *Crear cuestionario*. Se muestra el modal, rellena los campos y pulsa en *Guardar*	El cuestionario nuevo se tiene que mostrar en la tabla y a la BD	Resultado esperado	--
3	Hace click en el icono de modificar en alguno de los elementos. Se muestra un modal con los datos del cuestionario. Modifica algún dato y hace click en *Guardar*	En la lista se muestra el cuestionario modificado, y en la BD tiene que estar modificado	Resultado esperado	--

4	Hace click en el icono de eliminar en alguno de los elementos. Se muestra un modal preguntando *si realmente desea borrar ese cuestionario*. Hace click en *Aceptar*	El cuestionario eliminado no se tiene que mostrar en la lista, ni tampoco las preguntas. El cuestionario y las preguntas son eliminadas de la BD	Resultado esperado	--
5	Hace doble click en cualquiera de los cuestionarios, que tiene en la tabla	Redirigir a la página de preguntas y mostrar una tabla	Resultado esperado	--
6	Hace click en el botón *Crear pregunta*	Mostrar modal	Resultado esperado	--
7	Hace click en el botón *Crear pregunta*. Se muestra el modal, rellena los campos y pulsa en *Guardar*	La nueva pregunta se tiene que mostrar en la tabla y a la BD	Resultado esperado	--
8	Hace click en el botón *Importar preguntas.* Se muestra el modal y hace click en *Seleccionar fichero*	Abrir el explorador de ficheros del ordenador	Resultado esperado	--
9	En el modal de importar hace click en cargar fichero. El fichero contiene nodos repetidos	Mostrar aviso indicando que hay preguntas repetidas y si quiere guardar sólo los correctos	Se muestra el aviso y puede hacer click en *SÍ* o *NO*	--
10	En el aviso que se muestra al importar las preguntas, hace click en *SÍ*	Actualizar la lista de testers, con los nodos nuevos	Resultado esperado	--
11	En el aviso que se muestra al importar los testers hace click en el botón *NO*	La lista no se modifica	Resultado esperado	--
12	Hace click en el icono de modificar en alguno de los elementos. Se muestra un modal con los datos de la pregunta. Modifica algún dato y hace click en *Guardar*	En la lista se muestra la pregunta modificad, y en la BD tiene que estar modificada	Resultado esperado	--

13	Hace click en el icono de eliminar en alguno de los elementos. Se muestra un modal preguntando *si realmente desea borrar esa pregunta*. Hace click en *Aceptar*	La pregunta eliminada no se tiene que mostrar en la lista y se tiene que eliminar de la BD	Resultado esperado	--

7.2.6 Página responder cuestionario

Las pruebas realizadas para la página de responder cuestionario son las que se describen en la Tabla 52. Para acceder a esta página es necesario hacer click en *Responder cuestionario* de la página menú del proyecto.

TABLA 52: PRUEBAS PÁGINA RESPONDER CUESTIONARIO

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	No escoge alguno de los datos para añadir la respuesta y hace click en *Añadir*	Se muestra un modal en el que pone que tiene que completar todos los campos	Resultado esperado	--
	Escoge una pregunta, un tester y la respuesta. Hace click en *Añadir*	La nueva respuesta se añade a la tabla	Resultado esperado	--
2	Hace click en el icono de eliminar de alguna de las respuestas del cuestionario	La respuesta se elimina de la y de la BD	Resultado esperado	--
3	Hace click en el icono de modificar de alguna de las respuestas del cuestionario	Se muestra el modal *Modificar respuesta del cuestionario*	Resultado esperado	--
4	En el modal *Modificar respuesta del cuestionario* modifica la respuesta hace click en *Guardar*	La respuesta se actualiza en la tabla de respuestas del cuestionario, al igual que en la BD	Resultado esperado	--
5	Hace click en el icono de eliminar de alguna de las respuestas	La respuesta se elimina de la y de la BD	Resultado esperado	--
6	Hace click en el botón *Importar* de la página respuestas del cuestionario. Se muestra el modal y hace click en *Seleccionar fichero*	Abrir el explorador de ficheros del ordenador	No mostraba el explorador	--

7	En el modal de importar hace click en cargar fichero. El fichero contiene respuestas repetidas y preguntas que no existen	Mostrar aviso indicando que hay respuestas repetidas y preguntas que no existen y si quiere guardar sólo los correctos	Se muestra el aviso y puede hacer click en *SÍ* o *NO*	--
8	En el aviso que se muestra al importar las respuestas, hace click en *SÍ*	Actualizar la lista de las respuestas	Resultado esperado	--
9	En el aviso que se muestra al importar las respuestas, hace click en el botón *NO*	La lista no se modifica	Resultado esperado	--

7.2.7 Página tareas y tareas realizadas

Las pruebas realizadas para la página de tareas y para la página de paths son las que se describen en la Tabla 53. Y para la página de tareas realizadas y para la página de paths realizados, se describen en la Tabla 54.

Para acceder a la página de tareas es necesario hacer click en *Tareas* y para acceder a la página de tareas realizadas hacer click en *Tareas realizadas* de la página menú del proyecto. Estos dos objetos contienen paths, en las pruebas que se han realizado se indica cómo acceder a las páginas de paths de ambos casos.

TABLA 53: PRUEBAS PÁGINA TAREAS

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	Hace click en el botón *Crear tarea*	Mostrar modal	Resultado esperado	--
2	Hace click en el botón *Crear tarea*. Se muestra el modal, rellena los campos y pulsa en *Guardar*	La tarea nueva tiene que añadirse a la tabla y a la BD	Resultado esperado	--
3	Hace click en el icono de modificar en alguno de los elementos. Se muestra un modal con los datos de la tarea. Modifica algún dato y hace click en *Guardar*	En la lista se muestra la tarea modificada, y en la BD tiene que estar modificada	Resultado esperado	--

4	Hace click en el icono de eliminar en alguno de los elementos. Se muestra un modal preguntando *si realmente desea borrar esa tarea*. Hace click en *Aceptar*	La tarea eliminada no se tiene que mostrar en la lista al igual que sus paths correspondientes, y se tienen que eliminar de la BD también	Resultado esperado	--
5	Hace click en el enlace *Paths* de una de las tareas	Redirigir a la página de paths	Resultado esperado	--
6	Escoge un path del selector que se muestra en la página de paths	Se tiene que mostrar la tabla que contienen los nodos correspondientes a ese path	Cargaba el mismo path todo el tiempo	No se hacia el tratamiento de los diferentes paths
7	Hace click en el botón *Crear path*, en la página paths	Mostrar modal de *Crear paths*	Resultado esperado	--
8	En el modal de *Crear paths* escoge un nodo y escribe el número de clicks y hace click en el botón *Añadir*	El nodo se tiene que agregar al final de tabla	El nodo se agregaba en cualquier posición	Hay que configurar el método addRows de la tabla para añadir al final de la tabla
9	En el modal de *Crear paths* hace click en *Guardar*	El path tiene que estar disponible en la página paths	El path nuevo estaba disponible tras refrescar la página	No se añadía el elemento nuevo al selector
10	Hace click en el botón *Modificar path*, en la página paths	Mostrar modal de *Modificar paths*	Resultado esperado	--
11	En el modal de *Modificar paths* escoge un nodo y escribe el número de clicks y hace click en el botón *Añadir*	El nodo se tiene que agregar al final de tabla	Resultado esperado	--
12	En el modal de *Modificar paths* hace clic en uno de los nodos que ha añadido para mover a otra posición	La fila del nodo se mueve a la posición que desee el usuario	La fila se queda en la misma posición	No se había activado esa opción
13	En el modal de *Modificar paths* hace click en el icono de eliminar	El nodo se elimina de la lista	Resultado esperado	--

14	En el modal de *Modificar paths* hace click en *Guardar*	El path tiene que estar disponible en la página paths y modificado en la BD	Resultado esperado	--
15	Hace click en el botón *Eliminar* de la página paths	Se muestra un modal preguntando si quiere eliminar el path activo	Resultado esperado	--
16	Hace click en el botón SÍ del modal que se muestra al hacer click en *Eliminar* de la página paths	Se elimina el path del selector de paths y de la BD	El nombre del path activo seguía disponible, aunque la tabla estaba vacía	No se eliminaba el elemento nuevo del selector
17	Hace click en el botón *Importar* de la página tareas. Se muestra el modal y hace click en *Seleccionar fichero*	Abrir el explorador de ficheros del ordenador	No mostraba el explorador	--
18	En el modal de importar hace click en cargar fichero. El fichero contiene tareas repetidas y nodos que no existen	Mostrar aviso indicando que hay tareas repetidas y nodos que no existen y si quiere guardar sólo los correctos	Se muestra el aviso y puede hacer click en *SÍ* o *NO*	--
19	En el aviso que se muestra al importar las tareas, hace click en *SÍ*	Actualizar la lista de tareas, con los nodos nuevos	Resultado esperado	--
20	En el aviso que se muestra al importar las tareas, hace click en el botón *NO*	La lista no se modifica	Resultado esperado	--

TABLA 54: PRUEBAS PÁGINA TAREAS REALIZADAS

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	No escoge alguno de los datos para añadir la tarea y hace click en *Añadir*	Se muestra un modal en el que pone que tiene que completar todos los campos	Resultado esperado	--
2	Escoge una tarea, un tester, escribe el número de clicks y si ha finalizado o no. Hace click en *Añadir*	La nueva tarea realizada se añade a la tabla	Resultado esperado	--

3	Hace click en el icono de eliminar de alguna de las tareas realizadas	La tarea realizada se elimina de la tabla al igual que, el path realizado y de la BD	Resultado esperado	--
4	Hace click en el icono de modificar de alguna de las tareas realizadas	Se muestra el modal *Modificar tarea realizada*	Resultado esperado	--
5	En el modal *Modificar tarea realizada* modifica el tiempo o la finalización y hace click en *Guardar*	La tarea se actualiza en la tabla de tareas realizadas y en la BD	Se actualizaba en la tabla al refrescar la página	Se tiene que eliminar esa fila y añadir la nueva.
6	Hace click en el enlace *Path realizada* de una de las tareas realizadas	Redirigir a la página de path realizado	Resultado esperado	--
7	Hace click en el botón *Crear path*, en la página paths	Si tiene un path realizado el botón esta deshabilitado, de lo contrario se muestra el modal *Modificar path realizado*	Probadas las dos opciones: resultado esperado	--
8	En el modal de *Crear path realizado,* escoge un nodo y escribe el número de clicks y hace click en el botón *Añadir*	El nodo se tiene que agregar al final de tabla	Resultado esperado	--
9	En el modal de *Crear path realizado,* no escoge un nodo o no escribe el número de clicks y hace click en el botón *Añadir*	Se muestra un alert indicando que tiene que rellenar los dos campos, no se guarda nada	Resultado esperado	--
10	En el modal de *Crear paths* hace click en *Guardar*	El path tiene que estar disponible en la página paths	Resultado esperado	--
11	Hace click en el botón *Modificar path realizado*, en la página path realizado	Mostrar modal de *Modificar path realizado*	Resultado esperado	--
12	En el modal de *Modificar path realizado* escoge un nodo y escribe el número de clicks y hace click en el botón *Añadir*	El nodo se tiene que agregar al final de tabla	Resultado esperado	--
13	En el modal de *Modificar paths* hace clic en uno de los nodos que ha añadido para mover a otra posición	La fila del nodo se mueve a la posición que desee el usuario	Resultado esperado	--

14	En el modal de *Modificar path realizado* hace click en el icono de eliminar	El nodo se elimina de la lista	Resultado esperado	--
15	En el modal de *Modificar path realizado* hace click en *Guardar*	El path tiene que estar disponible en la página path realizado y modificado en la BD	Resultado esperado	--
16	Hace click en el botón *Eliminar* de la página path realizado	Se muestra un modal preguntando si quiere eliminar el path realizado	Resultado esperado	--
17	Hace click en el botón SÍ del modal que se muestra al hacer click en *Eliminar* de la página path realizado	Se muestra una tabla vacía, y se habilita el botón *Crear path*, además de eliminar el path realizado de la BD	El botón *Crear path* no se habilitaba	Faltaba agregar la condición para habilitar el botón
18	Hace click en el botón *Importar*. Se muestra el modal y hace click en *Seleccionar fichero*	Abrir el explorador de ficheros del ordenador	Resultado esperado	--
19	En el modal de importar hace click en cargar fichero. El fichero contiene tareas realizadas repetidas y algunos nodos que no existen	Mostrar aviso indicando que hay tareas repetidas y nodos que no existen y si quiere guardar sólo los correctos	Se muestra el aviso y puede hacer click en *SÍ* o *NO*	--
20	En el aviso que se muestra al importar las tareas, hace click en *SÍ*	Actualizar la lista de tareas realizadas, con el path realizado	Resultado esperado	--
21	En el aviso que se muestra al importar las tareas, hace click en el botón *NO*	La lista no se modifica	Resultado esperado	--

7.2.8 Página resultados: efectividad, eficiencia y satisfacción

Las pruebas realizadas para la página de resultados son las que se describen en la Tabla 55. Para acceder a esta página es necesario hacer click en *Resultados* de la página menú del proyecto. Este apartado no tiene muchos casos de prueba, puesto que la interfaz es sencilla y basta con hacer click en cualquiera de las opciones para mostrar los resultados.

Por otro lado, como se ha explicado en el apartado de calcular resultados del back-end. Si el usuario

TABLA 55: PRUEBAS PÁGINA RESULTADOS

Código	Descripción	Resultado esperado	Resultado real	Comentario
1	Hace click en *Efectividad* y se tienen las respuestas de todos los testers	Mostrar la página de *Efectividad* con sus respectivas tablas	Resultado esperado	--
2	Hace click en *Eficiencia* y se tienen las respuestas de todos los testers	Mostrar la página de *Eficiencia* con sus respectivas tablas	Resultado esperado	--
3	Hace click en *Satisfacción* con y se tienen las respuestas de todos los testers	Mostrar la página de *Satisfacción* con su respectiva tabla	Resultado esperado	--
4	Hace click en *Eficiencia con grafos* con y se tienen las respuestas de todos los testers	Mostrar la página de *Eficiencia* con sus respectivas tablas y grafos	El grafo no se mostraba correctamente con la escala propuesta por el back-end	Se ha modificado esa escala para adaptarlo la necesidad del proyecto
5	Hace click en *Informe completo con grafos* con y se tienen las respuestas de todos los testers	Mostrar la página Informe completo con grafos, que incluyen las tablas de la efectividad, eficiencia y satisfacción, además de los grafos correspondientes a la eficiencia	Al tener tantas tablas en el resultado, no se organizaba bien en la pantalla	Se ha modificado la interfaz para ordenar las tablas y mostrarlas de otra forma
6	Hace click en cualquiera de las opciones para calcular. Y no se han configurado todos los datos necesarios para el cálculo	Se muestra un modal indicando que no se han configurado todos los datos del proyecto	Resultado esperado	--
7	Hace click en cualquiera de las opciones para calcular. Y se han configurado todos los datos necesarios para el cálculo, pero sólo se han introducido 5 de 12 respuestas de tareas y cuestionario	Realiza los cálculos	Resultado esperado	--

8. CONCLUSIONES Y TRABAJO FUTURO

En este apartado se tratan las conclusiones al finalizar el proyecto, el cómo y porque ha variado la planificación inicial, también se analizan los objetivos propuestos y una crítica personal.

8.1 Conclusiones de gestión

Al hacer una comparación entre la planificación inicial de horas y la planificación real, es importante resaltar que se ha producido una considerable diferencia. Principalmente, por causas personales, al compaginar el trabajo, las prácticas y el desarrollo del proyecto.

En la Figura 38 se puede ver cómo se ha modificado la planificación, sobre todo en el apartado *Prototipos,* la parte izquierda representa a la distribución del tiempo dedicada al back-end, y la de la derecha al front-end. Se ha tomado esa decisión puesto que resulta más cómodo para la desarrolladora, por el hecho de haber utilizado una tecnología nueva, lo cual ha requerido más tiempo en el aprendizaje del que se había planificado.

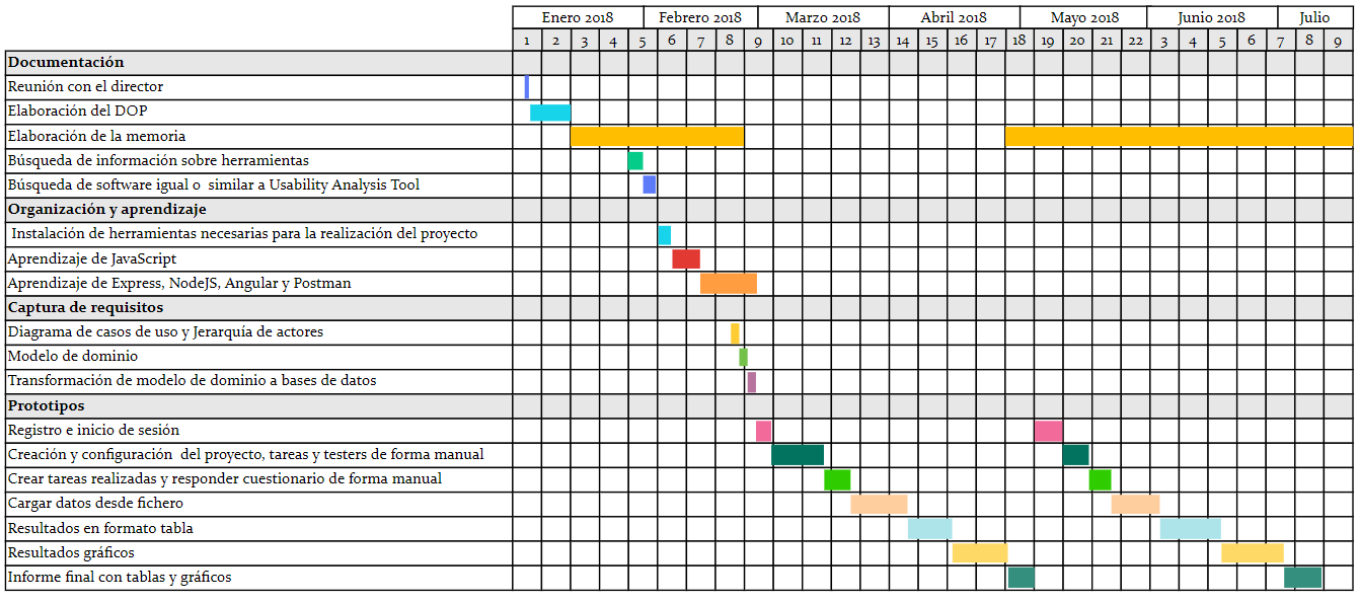


FIGURA 38: DIAGRAMA DE GANTT ACTUALIZADO

Por otro lado, para ver las comparaciones directamente sobre las horas, en la Tabla 56 se muestran las horas que se habían planificado, frente a las que se han empleado realmente. Han sufrido considerables cambios todos los apartados menos el de *Captura de requisitos* que se ha realizado según lo previsto.

TABLA 56: COMPARACIÓN DE HORAS PLANIFICADAS Y REALES

	DURACIÓN (horas planificadas)	**DURACIÓN (horas reales)**
Documentación	221	401
Organización y aprendizaje	58	108
Captura de requisitos	22	22
Prototipos	260	360
Horas totales	561	885

El apartado de *Documentación*, en concreto la *Elaboración de la memoria* la cual ha duplicado las horas que se estimaban, y se debe a que se han documentado todos los casos de prueba realizados, además de los casos de uso extendidos, los diagramas de secuencia, la investigación de las herramientas a utilizar y de software similar a este proyecto. Ha sido tanta la información que ha incrementado las horas.

En el apartado de *Organización y aprendizaje* (*Aprendizaje de Express, NodeJS, Angular y Postman*) se le ha dedicado más tiempo de lo estimado al aprendizaje de esas tecnologías, puesto que son totalmente nuevas para la desarrolladora. Ha sucedido lo mismo al desarrollar los *Prototipos*.

Tras haber incrementado las horas totales dedicadas al proyecto, se recalcula la mano de obra de la evaluación económica, quedando los gastos totales, tal y cómo se muestran en la Tabla 57.

TABLA 57: TOTAL GASTOS ACTUALIZADO

Descripción	**Coste**
Gastos directos	
Mano de obra	8.482,54 €
Hardware	52,00 €
Software	69,00 €
Gastos indirectos	
Gastos varios	430,18 €
Gastos totales	9.033,72 €

8.2 Conclusiones de objetivos

Al finalizar este proyecto, se han analizado los objetivos que se esperaban cumplir a lo largo de la realización del mismo, llegando a la conclusión de que se han cumplido satisfactoriamente.

Se ha conseguido medir los objetivos de la usabilidad, aunque por el camino se han producido problemas al momento de aplicar las diferentes ecuaciones necesarias para el cálculo, al final se lograron solucionar, dando los resultados lo más preciso posible, consiguiendo que la herramienta web realice los cálculos de forma automática con los datos que se le proporcionan, obteniendo unos resultados comprensibles y fáciles de interpretar.

Otro de los objetivos era aprender y adquirir destreza en la elaboración de páginas web, el manejo de nuevos lenguajes de programación y tecnologías, el cual se ha cumplido, gracias a la investigación realizada y que se describe en el apartado de Antecedentes (Elección de lenguajes y tecnologías), se han podido elegir aquellas tecnologías y lenguajes de programación en las que la desarrolladora tenía poca o ninguna experiencia, permitiéndole ampliar sus conocimientos.

El último objetivo es el de crecimiento personal, se ha cumplido, gracias a las pautas marcadas por el tutor del proyecto, que ha servido de guía. Por otra parte, se ha aprendido a gestionar un proyecto desde sus inicios, teniendo en cuenta qué etapas del mismo son necesarias realizar antes de empezar con el desarrollo.

8.3 Conclusiones personales

A nivel personal, se resalta sobre todo el hecho de haber adquirido conocimientos, sobre el desarrollo web implementado con un back-end y front-end, el trabajar con una arquitectura REST, lo cual ha resultado muy cómodo, además de que hoy en día es lo que las empresas demandan en el mercado laboral, por tanto, es una buena oportunidad de adquirir experiencia. Por otro lado, ha sido un gran acierto complementar Angular con boostrap, ag-grid y vis.js, ya que han facilitado la implementación del front-end.

Pese a todas las complicaciones que han surgido durante la realización de este proyecto, se considera que ha supuesto un gran crecimiento personal, y que ha aumentado las capacidades de resolución frente a adversidades de desarrollo. No sólo se ha profundizado en alguno de los lenguajes de programación en los que se tenía algo de conocimiento, sino también en las tecnologías y lenguajes que eran desconocidas para la desarrolladora.

En cuanto al proyecto, en pocas palabras, es una herramienta que resulta de utilidad, ya que agiliza el proceso de calcular los resultados de los objetivos de la usabilidad, el usuario se puede olvidar de hacer recuentos manuales o utilizar hojas de cálculo. Tal vez lo que resulte un poco trabajoso sea la fase de configurar el proyecto, una vez superado esto, lo demás es hacer un click, y obtener los resultados.

8.4 Conclusiones sobre líneas futuras

En cuanto a mejoras para Usability Analysis Tool, se puede considerar el hecho de cargar los datos necesarios para el proyecto en otros formatos, no limitarse a JSON o CSV. Resultaría interesante también, el hecho de poder exportar los resultados obtenidos en forma de hojas de cálculo o PDF en el caso de los resultados que contienen grafos. De esta forma facilita al usuario la utilización de dichos resultados.

9. BIBLIOGRAFÍA

Agencia Estatal Boletín Oficial del Estado. (s.f.). *Artículo 33. Tablas de niveles salariales.* Obtenido de https://www.boe.es/boe/dias/2017/01/18/pdfs/BOE-A-2017-542.pdf

Agresti, A., & Coull, B. (1998). Approximate is Better than «Exact» for Interval Estimation of Binomial Proportions. En *The American Statistician* (págs. 52(2), 119-126).

Bangor, A., Kortum, P., & Miller, J. (2009). Determining what individual SUS scores mean: Adding an adjective rating scale. En *Journal of Usability Studies* (págs. 114-123.).

Brooke, J. (1996). *SUS: A quick and dirty usability scale. En P. W. Jordan, B. Weerdmeester, A. Thomas, & I. L. Mclelland (Eds.), Usability evaluation in industry.* London: Taylor and Francis.

Checkealos.com. (s.f.). *Checkealos User Experience Intelligence*. Obtenido de https://www.checkealos.com/en/

CrazyEgg. (s.f.). *CrazyEgg*. Obtenido de https://www.crazyegg.com/

de la Torre, C., & González, R. (2008). *Arquitectura SOA con tecnología Microsoft.* Krasis Press.

Distintiva S.Coop. (s.f.). *Distintiva Soluciones Tecnológicas Avanzadas*. Obtenido de http://www.distintiva.com/consultoria.php#usabilidad

DugWood. (s.f.). *DugWood*. Obtenido de https://www.dugwood.com/clickheat/index.html

Fundación Node.js. (s.f.). *Express*. Obtenido de http://expressjs.com/es/

Gallello, C. (s.f.). *UX Check*. Obtenido de http://www.uxcheck.co/

Google © 2010-2017. (s.f.). *AngularJS*. Obtenido de https://angularjs.org/

Google ©. (2010-2018). *Angular*. Obtenido de https://angular.io/

Google. (s.f.). *Google Analytics*. Obtenido de https://www.google.com/intl/es_ALL/analytics/features/analysis-tools.html

ISO 9241-11:1998. (23 de 01 de 2013). *Ergonomic requirements for office work with visual display terminals (VDTs) -- Part 11: Guidance on usability*.

Laplace, P. (1812). *Theorie analytique des probabilitités.* Paris, France: Courcier.

Lewis, J., & Sauro, J. (2006). When 100% Really Isn't 100%: Improving the Accuracy of Small-Sample Estimates of Completion Rates. En *Journal of Usability Studies* (págs. 136-150).

Mollá Sirvent, R. (2016). *Estudio y mejora del rendimiento del backend.* Obtenido de http://hdl.handle.net/10045/57068

Node.js Foundation. (s.f.). *Node JS*. Obtenido de https://nodejs.org/es/

Polymer Authors. (2017). *Polymer Project*. Obtenido de https://www.polymer-project.org/

Sauro, J., & Kindlund, E. (2005). How Long Should a Task Take? Identifying Specification Limits for Task Times in Usability Tests. En *Actas de Human Computer Interaction International Conference.* Las Vegas, Nevada, USA: Lawrence Erlbaum Associates.

Sauro, J., & Lewis, J. (2010). Average task times in usability tests: what to report? En *Actas de SIGCHI Conference on Human Factors in Computing Systems* (págs. 2347-2350). ACM Press.

Sauro, J., & Lewis, J. (2012). Quantifying the user experience: practical statistics for user research. Amsterdam. Waltham, MA: Elsevier/Morgan Kaufmann.

Smith, P. (1996). Towards a practical measure of hypertext usability. En *Interacting with Computers* (págs. 365-381).

Tullis, T., & Albert, B. (2013). *Measuring the user experience: collecting, analyzing, and presenting usability metrics (Second edition.). Amsterdam.* Boston: Elsevier/Morgan Kaufmann.

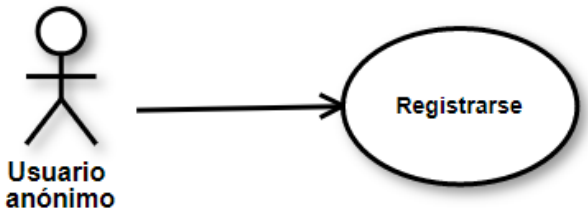
Villamañe, M. (2016). Fase 3: Análisis de datos. En *Análisis y mejora de los marcos actuales de desarrollo y evaluación de los Trabajos Fin de Grado mediante el uso de las TIC* (págs. 166-172).

Wilson, E. B. (1927). Probable Inference, the Law of Succession, and Statistical Inference. Journal of the American Statistical Association.

Zaharias, P. (2004). *Usability and e-Learning: The road towards integration.. ACM eLearn Magazine. 6. 10.1145/998337.998345.* Obtenido de https://www.researchgate.net/publication/200454308_Usability_and_e-Learning_The_road_towards_integration

10. ANEXO I.- CASOS DE USO EXTENDIDOS

10.1 Registrarse



Nombre:	Registrarse

Descripción: permite al usuario registrarse para tener acceso al sistema

Actores: usuario anónimo

Precondiciones: ninguna

Requisitos no funcionales: ninguno

Flujo de eventos:

1. El usuario hace click en *Registrarse* en la página principal de la web. Figura 39
2. Se muestra el formulario con los diferentes campos a completar, los campos que son obligatorios se muestran en color rojo. Figura 40

[Si llena todos los campos requeridos y pulsa en Registrarse]

3a.- Se registra al usuario y se redirige la página al inicio de sesión. Figura 39

[Si no llena alguno o ninguno de los campos requeridos o si el email no tiene el formato de email y pulsa en "Registrarse"]

3b.- No ocurre nada y se siguen mostrando los campos requeridos en rojo. Figura 41

[Si las contraseñas no coinciden]

3b.- Se muestra un mensaje indicando que las contraseñas no coinciden. Figura 42

[Si pulsa en la flecha volver del navegador]

3c.- Se muestra la página principal. Figura 39

Postcondiciones: el usuario queda registrado en la base de datos

Interfaz gráfica:

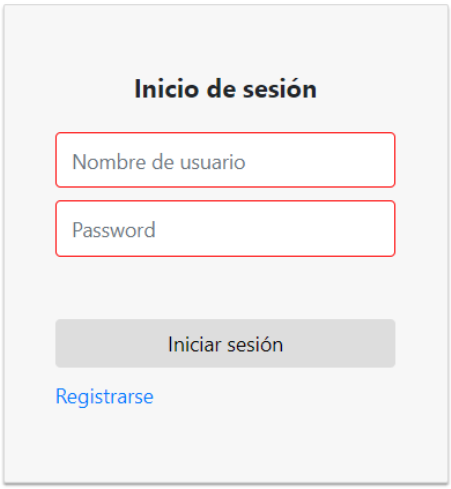


FIGURA 39: PÁGINA PRINCIPAL

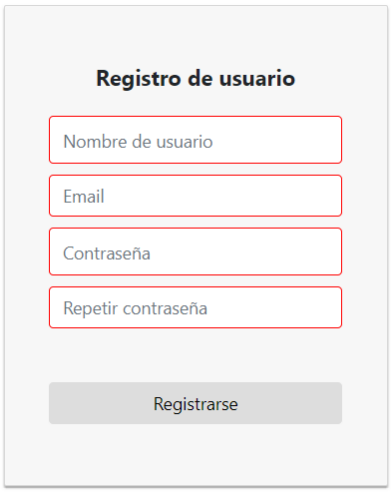


FIGURA 40: REGISTRO USUARIO

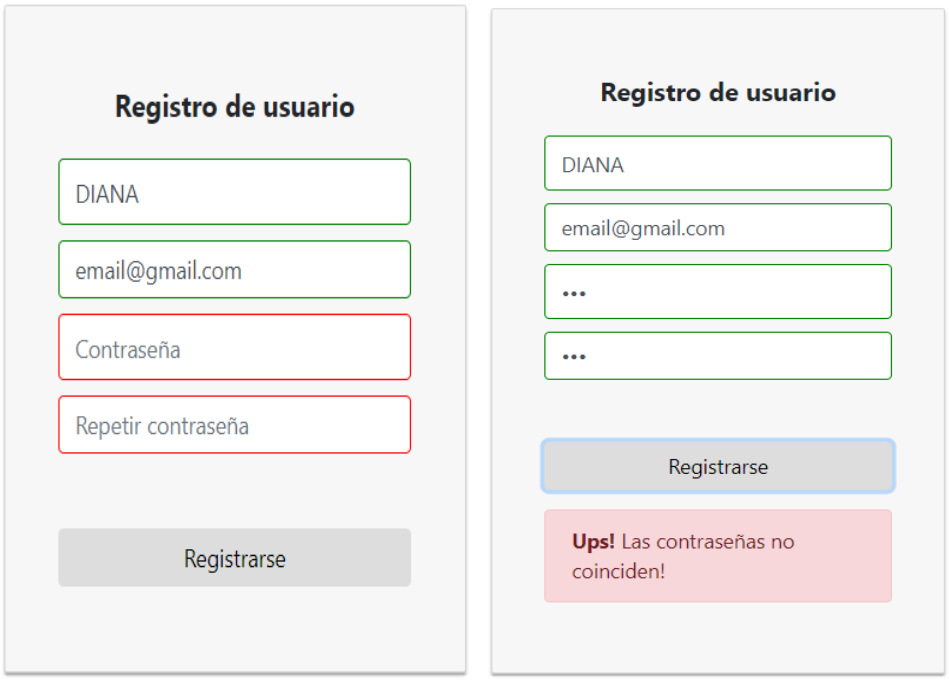
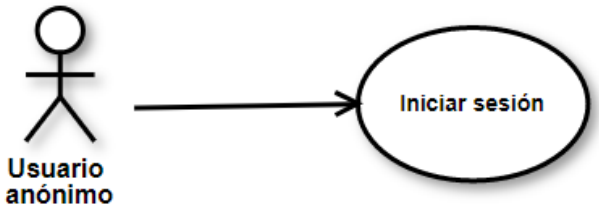


FIGURA 41: REGISTRO DE USUARIO ERROR (IZQUIERDA)
FIGURA 42: REGISTRO DE USUARIO ALERTA (DERECHA)

10.2 Iniciar sesión



Nombre:	Inicio de sesión

Descripción: haciendo uso del nombre de usuario y de la contraseña, permite al usuario acceder al sistema y a todas sus funcionalidades.

Actores: usuario anónimo

Precondiciones: ninguna

Requisitos no funcionales: ninguno

Flujo de eventos:

1. El usuario pone el nombre de usuario y la contraseña y hace click en "Iniciar sesión" en la página principal de la web. Se muestran en rojo los campos que son obligatorios. (Figura 39)

[Si el usuario existe en la base de datos y la contraseña es correcta]

2a.- Accede a la página de proyectos. Figura 43

[Si la contraseña es incorrecta]

2b.- Se muestra un mensaje indicando que la contraseña es incorrecta. Figura 44

[Si el usuario no existe]

2c.- Se muestra un mensaje indicando que el usuario no existe. Figura 44

[Si algún campo está vacío]

2d.- No ocurre nada, se queda en misma página.

Postcondiciones: el usuario inicia sesión y pasa a ser usuario identificado.

Interfaz gráfica:

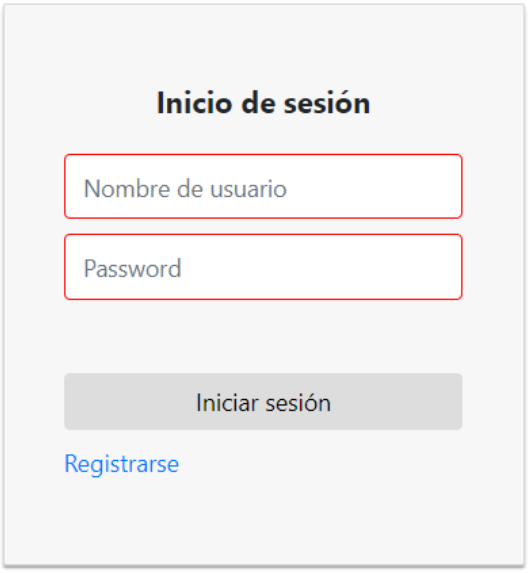


FIGURA 39: PÁGINA PRINCIPAL

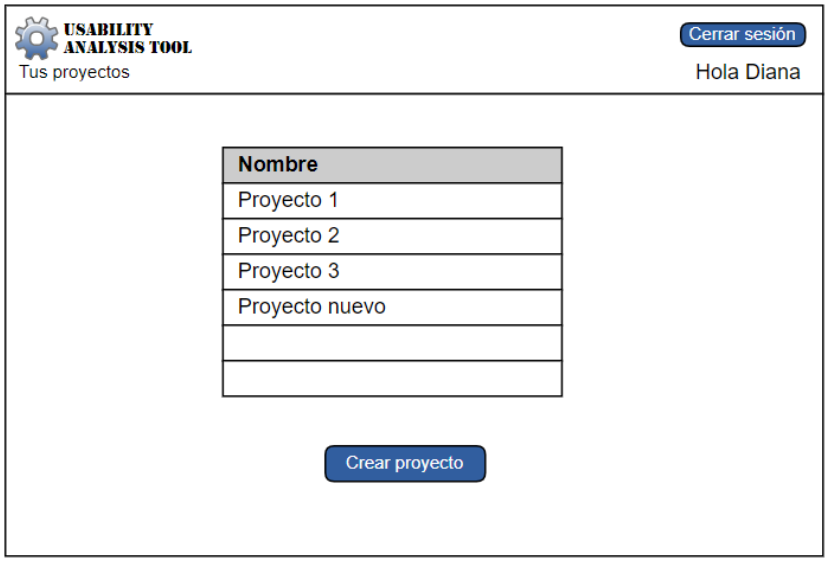
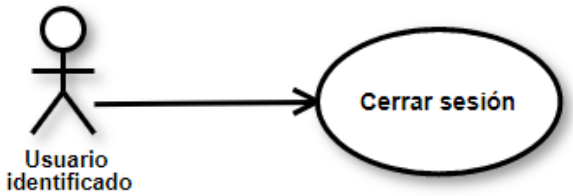


FIGURA 43: PÁGINA PROYECTOS

FIGURA 44: INICIO DE SESIÓN, MENSAJES DE ERROR

10.3 Cerrar sesión



Nombre:	Cerrar sesión

Descripción: permite al usuario cerrar sesión de forma segura.

Actores: usuario identificado

Precondiciones: ninguna

Requisitos no funcionales: ninguno

Flujo de eventos:

1. El usuario hace click en "Cerrar sesión" en la página proyectos de la web.

2a.- Se elimina el token de la sesión y se muestra la página principal.

Postcondiciones: el usuario identificado pasa a ser usuario anónimo

Interfaz gráfica:

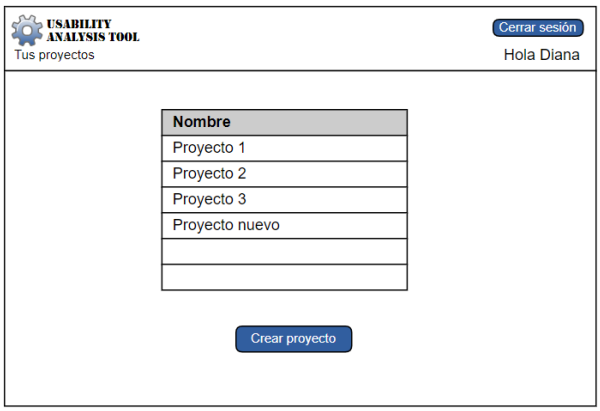


FIGURA 43: PÁGINA PROYECTOS

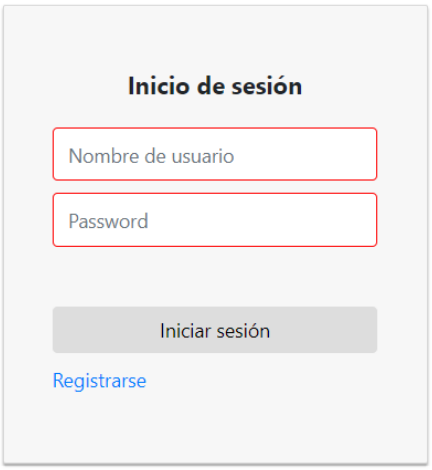
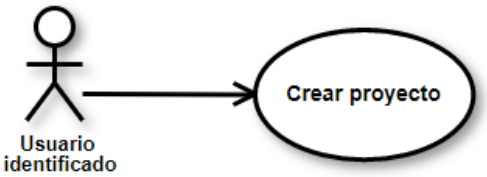


FIGURA 39: PÁGINA PRINCIPAL

10.4 Crear proyecto



Nombre:	Crear proyecto

Descripción: permite al usuario crear un proyecto

Actores: usuario identificado

Precondiciones: ninguna

Requisitos no funcionales: ninguno

Flujo de eventos:

1. El usuario hace click en "Crear proyecto" en la página proyectos de la web. Figura 43
2. Se muestra una ventana en la que el usuario tiene que poner el nombre con el que quiere crear el proyecto. Figura 45

[Si el proyecto se crea correctamente]

3a.- Se muestra la página que se ve en la Figura 46

[Si hay algún problema al momento de crear el proyecto]

3b.- Se muestra un mensaje indicando el error. Figura 47

Postcondiciones: el usuario queda registrado en la base de datos y se actualiza la tabla de proyectos

Interfaz gráfica:

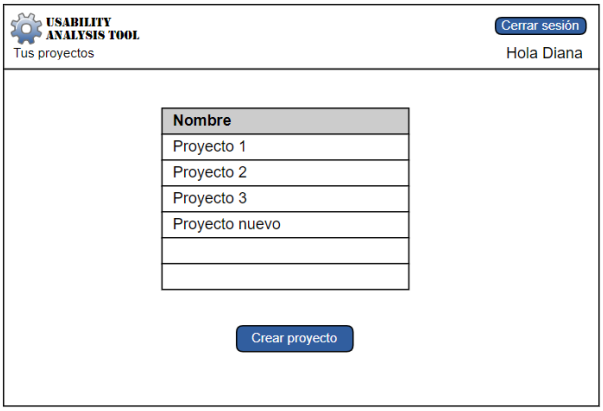


FIGURA 43: PÁGINA PROYECTOS

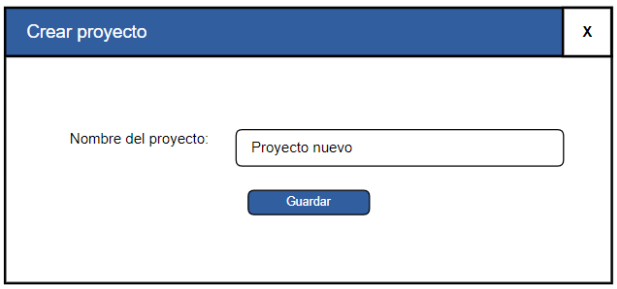


FIGURA 45: CREAR PROYECTO

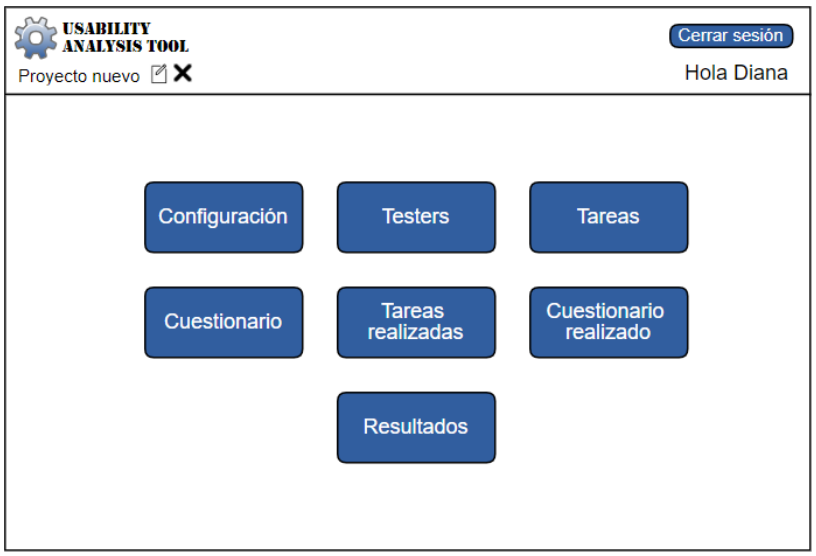


FIGURA 46: VER PROYECTO

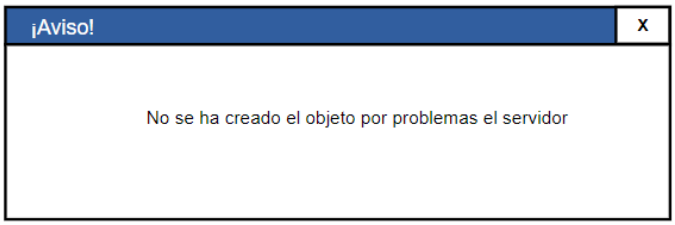


FIGURA 47: VENTANA ERROR AL CREAR

10.5 Crear tester, configurar proyecto, cuestionario

Nombre:	Crear tester, configurar proyecto, cuestionario

Descripción: son tres funcionalidades en una que permiten crear diferentes objetos necesarios para el proyecto.
Crear tester: necesita el nombre del tester.
Configurar proyecto: se necesita el nombre del nodo y el código que es un nombre corto y debe de ser único.
Cuestionario: se necesita un nombre para el cuestionario. Crear un cuestionario implica crear preguntas. Sólo en este caso en el flujo de eventos se realiza un paso adicional.

Actores: usuario identificado

Precondiciones: tener creado un proyecto, sólo en el caso del cuestionario: cuando se crean las preguntas es necesario tener un cuestionario creado.

Requisitos no funcionales: ninguno

Flujo de eventos:

1. El usuario hace click en el proyecto, se muestra la Figura 46.

[Si hace click en "Configuración"]
2.a.- Se muestra la Figura 48
[Si hace click en "Tester"]
2.b.- Se muestra la Figura 49
[Si hace click en "Cuestionario"]
2.c.- Se muestra la Figura 50

3. El usuario hace click en el botón "Crear nodos" o "Crear tester" o "Crear Cuestionario de las figuras antes vistas y se muestran las Figuras: Figura 51, Figura 52 y Figura 53,. Llena los campos que se muestran en esas figuras y pulsa en "Guardar"

[Si el objeto a crear existe]
4.a.- Se muestra la Figura 54
[Si se crea correctamente]
4.b.- Se muestra la Figura 55
[Si ocurre algún error de servidor]
4.c.- Se muestra la Figura 47

5. El usuario hace click en el cuestionario creado. Se muestra la Figura 56, hace click en "Crear pregunta", llena los campos de la Figura 57

[Si el objeto a crear existe]
6.a.- Se muestra la Figura 54
[Si se crea correctamente]
6.b.- Se muestra la Figura 55
[Si ocurre algún error de servidor]
4.c.- Se muestra la Figura 47

Postcondiciones: se crea el objeto correspondiente en la base de datos y se actualiza la tabla en la página correspondiente.

Interfaz gráfica:

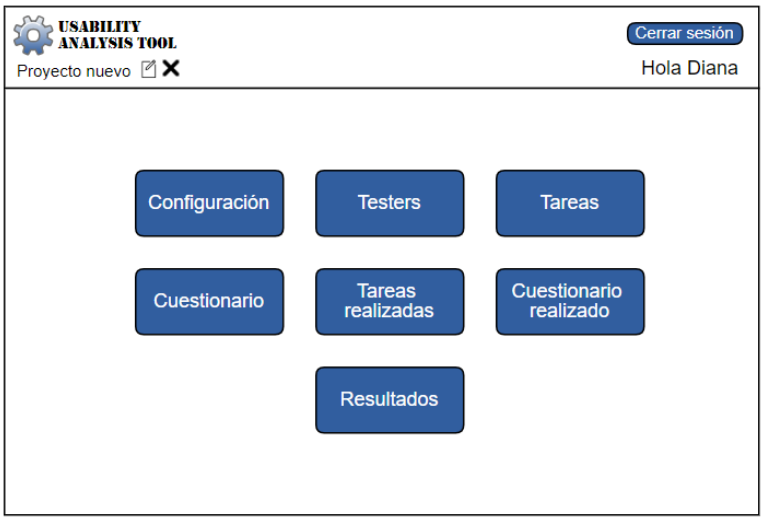


FIGURA 46: VER PROYECTO

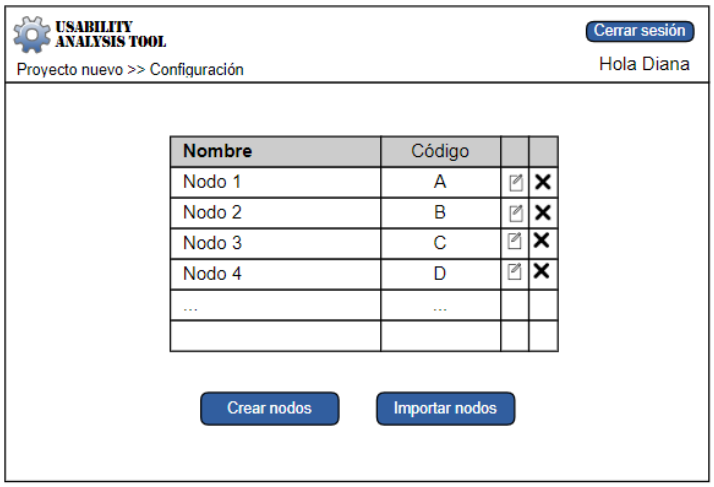


FIGURA 48: PÁGINA CONFIGURACIÓN

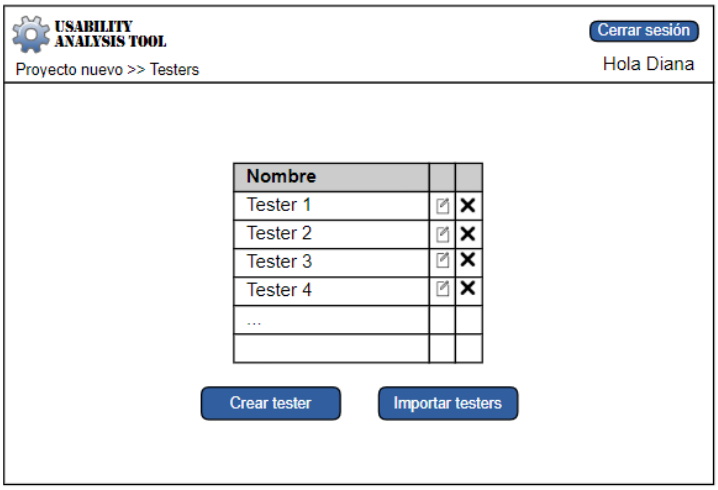


FIGURA 49: PÁGINA TESTERS

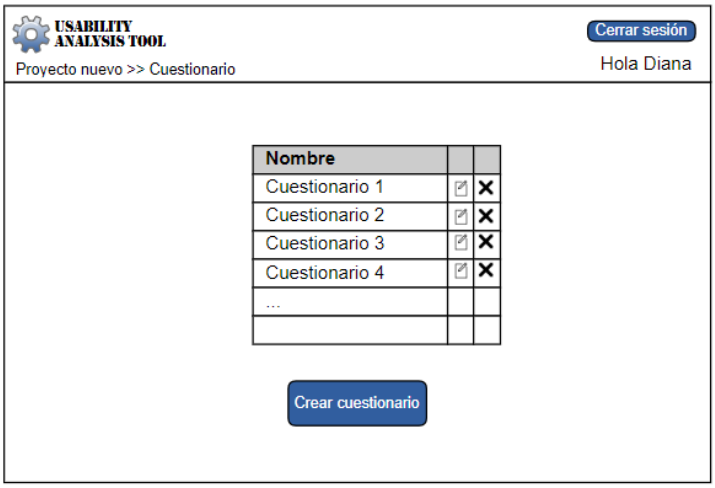


FIGURA 50: PÁGINA CUESTIONARIOS

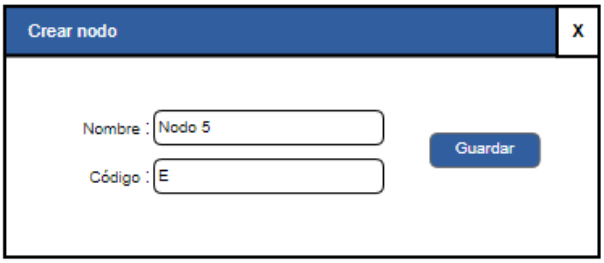


FIGURA 51: VENTANA CREAR NODO

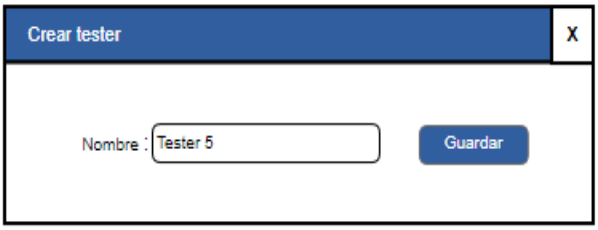


FIGURA 52: VENTANA CREAR TESTER

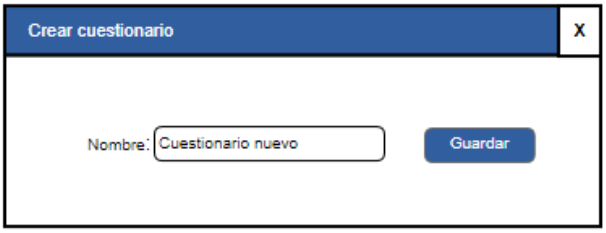


FIGURA 53: VENTANA CREAR CUESTIONARIO

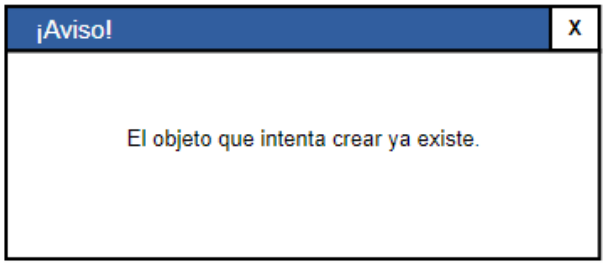


FIGURA 54: VENTANA OBJETO EXISTE

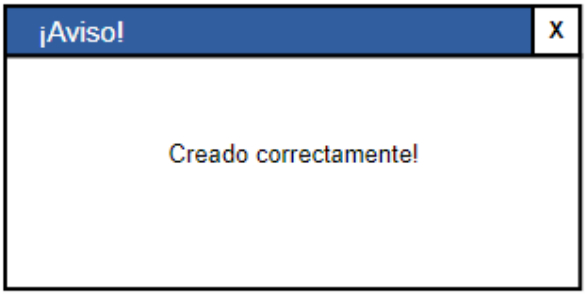


FIGURA 55: VENTANA CREADO CORRECTAMENTE

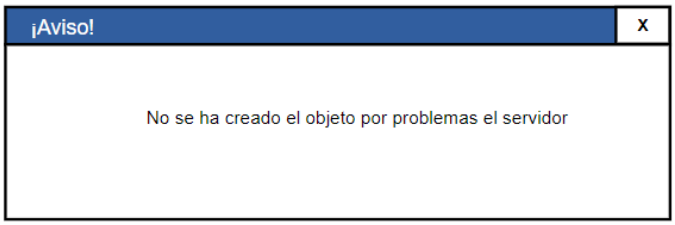


FIGURA 47: VENTANA ERROR AL CREAR

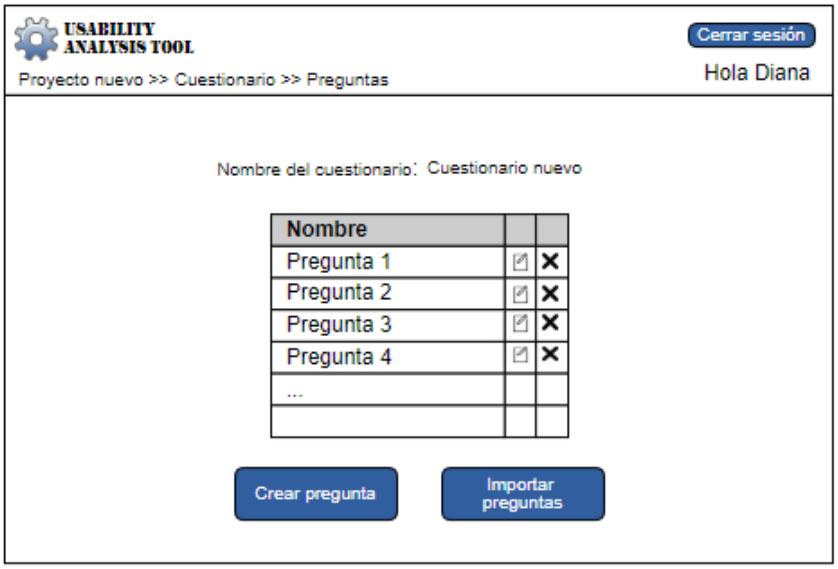


FIGURA 56: PÁGINA PREGUNTAS

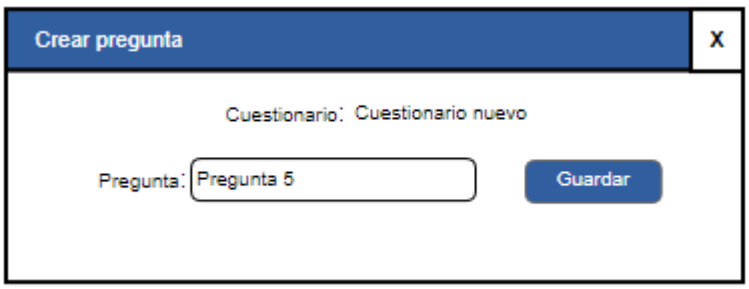


FIGURA 57:VENTANA CREAR PREGUNTA

10.6 Importar testers, configuración de proyecto, cuestionario

Nombre:	Importar testers, configuración de proyecto, cuestionario

Descripción: permite al usuario importar un fichero en formato CSV para crear testers, configurar el proyecto o el cuestionario. Sólo en el caso de importar un cuestionario es necesario crear el cuestionario a través de la interfaz y luego importar las preguntas. Todas estas funcionalidades tienen que tener una estructura específica Ver ANEXO III

Actores: usuario identificado

Precondiciones: tener creado un proyecto y en el caso del cuestionario, crear el cuestionario antes de importar las preguntas.

Requisitos no funcionales: ninguno

Flujo de eventos:

1. El usuario hace click en el proyecto, se muestra la Figura 46 y luego click en "Configuración" o "Tester" o "Cuestionario" y se muestran las figuras: Figura 48, Figura 49 y Figura 50
2. El usuario hace click en "Importar nodos" o "Importar testers" o "Importar cuestionario", se muestra la Figura 58 en la cual carga la ruta correspondiente y hace click en "Importar"

[Si el formato del fichero no es el correcto]

3.a.- Se muestra un mensaje indicando que el formato no es el correcto.

[Si el formato es correcto]

[Si alguno de los objetos a crear, existe en ese proyecto]

Se muestra un mensaje al usuario indicando el error.

[Si todo es correcto]

Se crean los objetos en la base de datos y se indica al usuario que los objetos se han creado correctamente.

Postcondiciones: se crean los objetos correspondientes en la base de datos y se actualiza la tabla en la página correspondiente.

Interfaz gráfica:

FIGURA 46: VER PROYECTO

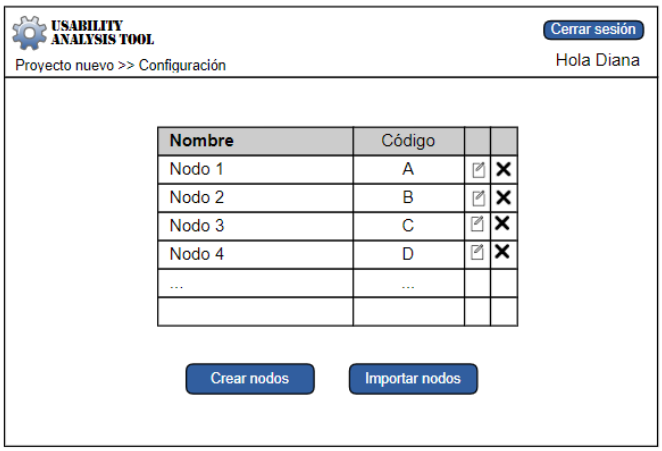


FIGURA 48: PÁGINA CONFIGURACIÓN

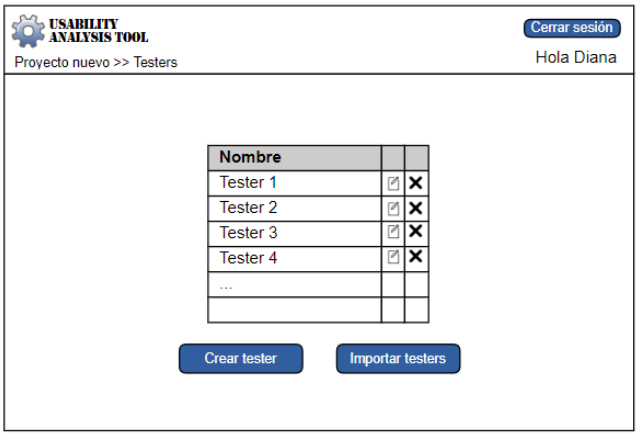


FIGURA 49: PÁGINA TESTERS

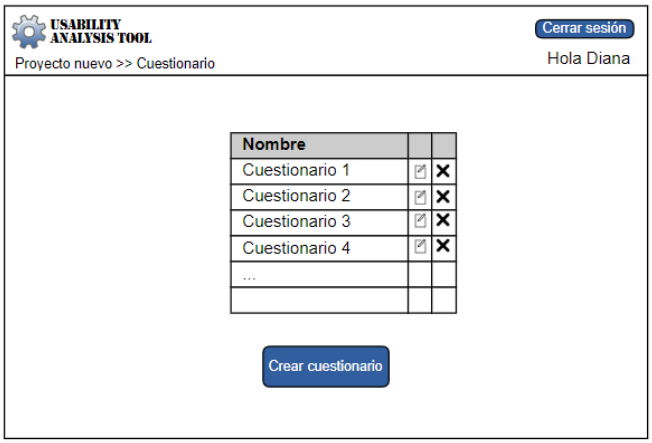


FIGURA 50: PÁGINA CUESTIONARIOS

FIGURA 58: IMPORTAR FICHERO

10.7 Crear tarea, tarea realizada

Nombre:	Crear tareas, tarea realizada

Descripción: Son dos funcionalidades similares, ambas crean objetos de utilidad para el proyecto.
Al crear una tarea también se pueden crear los paths, en el caso de la tarea se pueden crear paths óptimos y alternativos, mientras que en la tarea realizada el path que ha realizado el tester.

Actores: usuario identificado

Precondiciones: Tener creado el proyecto, nodos. Y cuando se trata de una tarea realizada, tener creadas las tareas.

Requisitos no funcionales: ninguno

Flujo de eventos:

1. El usuario hace click en el proyecto, se muestra la Figura 46 y luego click en "Tareas" o "Tareas realizadas" de esa misma figura.

[Si hace click en "Tareas"]

2.a.- Se muestra la Figura 59 y hace click en "Crear tarea", se muestra la Figura 60 llena los campos que se piden y hace click en "Guardar"

[Si la tarea existe o no completa alguno de los campos]

3.a.1- Se le muestra un mensaje indicando el error

[Si todo es correcto]

3.a.2.- Se crea la tarea en la base de datos y se actualiza la tabla de la Figura 59 y hace click en "Añadir path", se le muestra la Figura 61, completa los campos y hace click en "Añadir"

[Si no ha completado el campo "Nº de clicks" o escogido el nodo]

3.a.3.- No se añade el nuevo nodo a la tabla de la Figura 61

[Si completa los campos requeridos]

3.a.4.- Se añade el nuevo nodo a la tabla

3.a.5.- Termina de añadir nodos a la tabla y hace click en "Guardar". Se crean los nodos del path en la base de datos

[Si hace click en "Tareas realizadas"]

2.b.- Se muestra la Figura 62. Completa los datos necesarios para crear la tarea realizada en es misma figura y pulsa en "Añadir"

[Si no ha completado el campo duración o elegido alguno de los campos]

3b.1 Se le muestra un mensaje indicando que ese campo es obligatorio de cumplimentar.

[Si todo es correcto]

3.b.2.- Se crea la tarea en la base de datos y se agrega a la tabla que se muestra en la Figura 62

3.b.1.3.- Hace clic en "Añadir path", se le muestra la Figura 63 y se realizan los mismos pasos de antes mencionados al momento de agregar un path (3.a.2 al 3.a.5).

Postcondiciones: se crean las tareas y tareas realizadas, además de sus paths correspondientes

Interfaz gráfica:

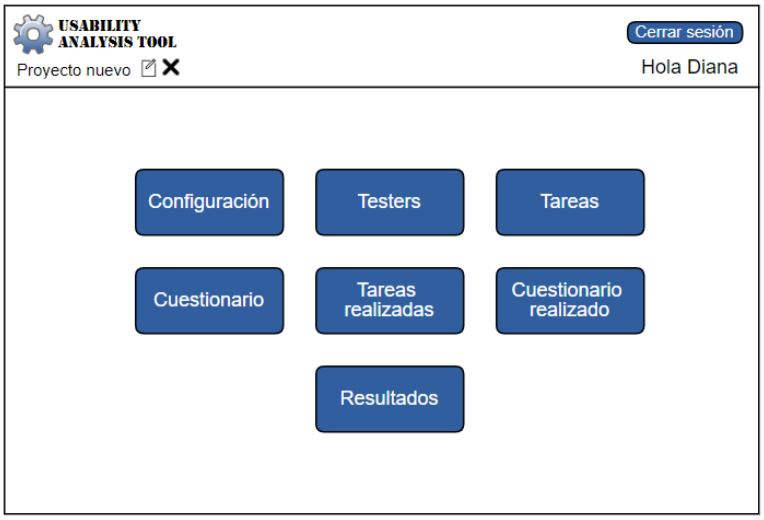


FIGURA 46: VER PROYECTO

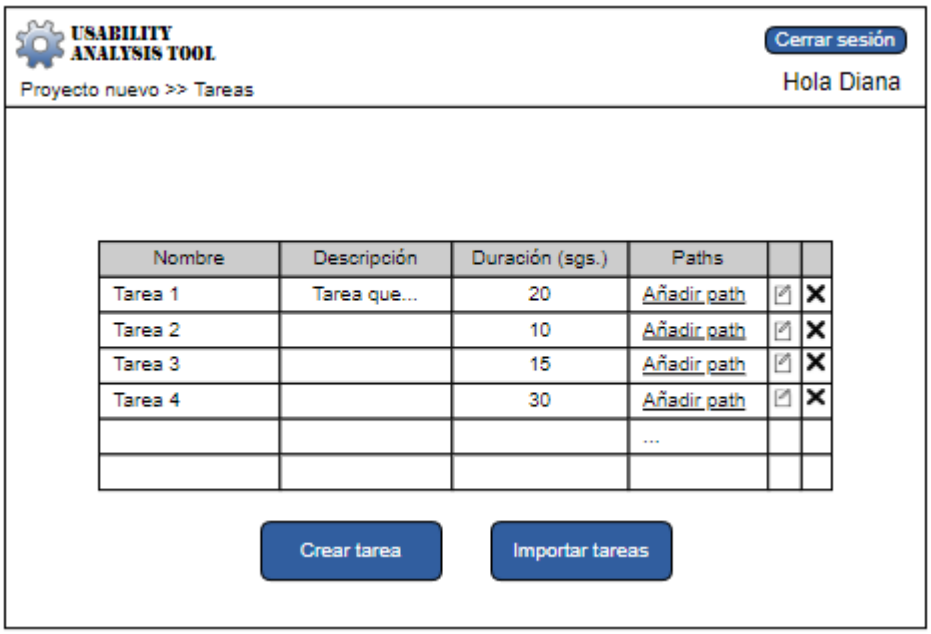


FIGURA 59:PÁGINA TAREAS

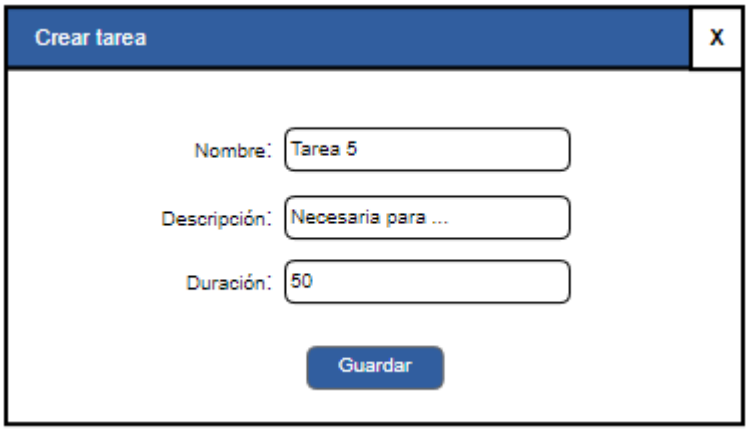


FIGURA 60: VENTANA CREAR TAREA

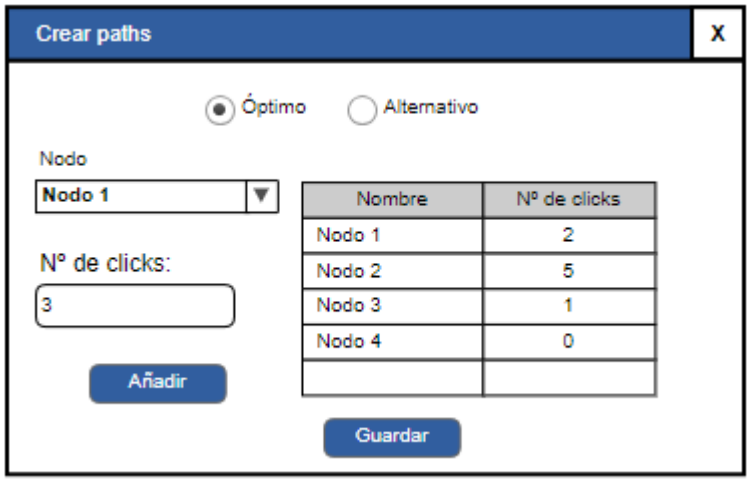


FIGURA 61: CREAR PATH

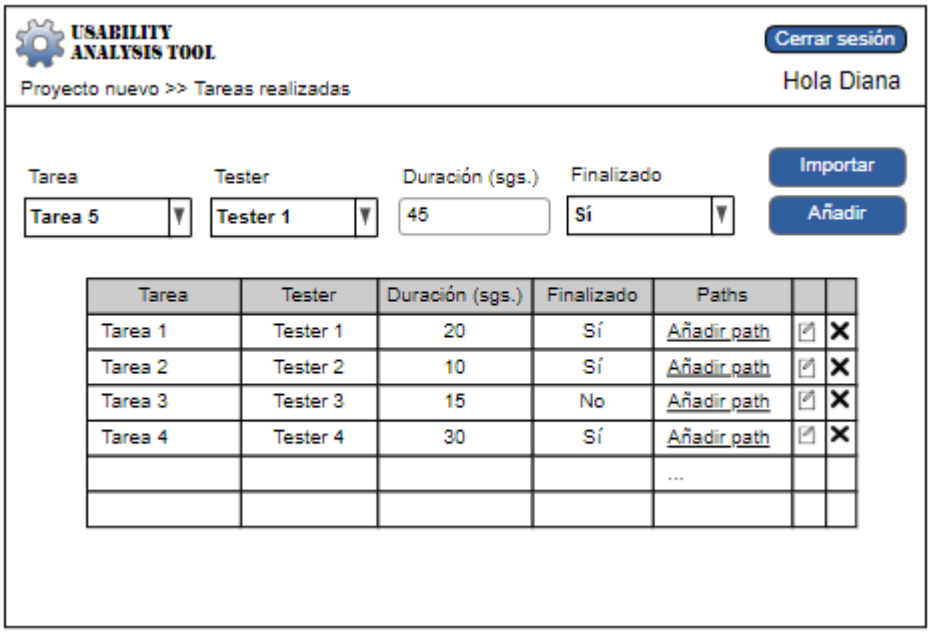


FIGURA 62: PÁGINA TAREAS REALIZADAS

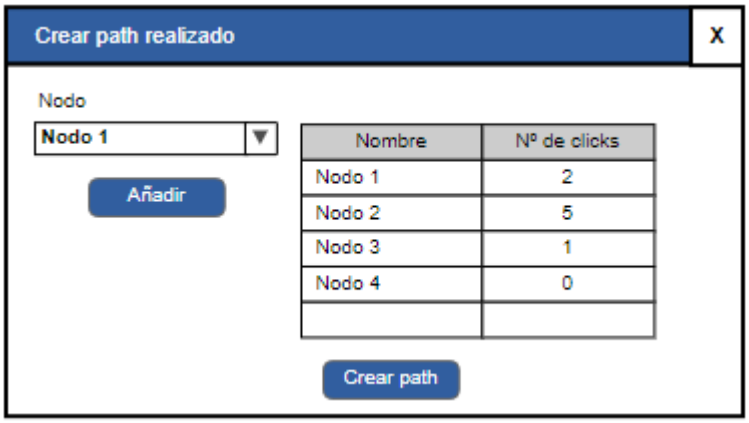


FIGURA 63: VENTANA CREAR PATH REALIZADO

10.8 Importar tareas, tareas realizadas

Nombre:	Importar tareas, tareas realizadas

Descripción: estas funcionalidades permiten importar ficheros en formato JSON. Ambos ficheros tienen que tener una estructura específica. Ver ANEXO III

Actores: usuario identificado

Precondiciones: tener creado el proyecto y los nodos. En caso de tratarse de las tareas realizadas, crear las tareas en primer lugar.

Requisitos no funcionales: ninguno

Flujo de eventos:

1. El usuario hace click en el proyecto, se muestra la Figura 46 y luego click en "Tareas" o "Tareas realizadas" y se muestran las figuras: Figura 59 y Figura 62 Figura 50
2. El usuario hace click en "Importar nodos" o "Importar testers" o "Importar cuestionario", se muestra la Figura 58 en la cual carga la ruta correspondiente y hace click en "Importar"

[Si el formato del fichero no es el correcto]

3.a.- Se muestra un mensaje indicando que el formato no es el correcto.

[Si el formato es correcto]

[Si alguno de los objetos a crear, existe en ese proyecto]

Se muestra un mensaje al usuario indicando el error.

[Si todo es correcto]

Se crean los objetos en la base de datos y se indica al usuario que los objetos se han creado correctamente.

Postcondiciones: se crean las tareas, tareas realizadas y sus correspondientes paths. Se actualizan las respectivas tablas de "Tareas" y "Tareas realizadas"

Interfaz gráfica:

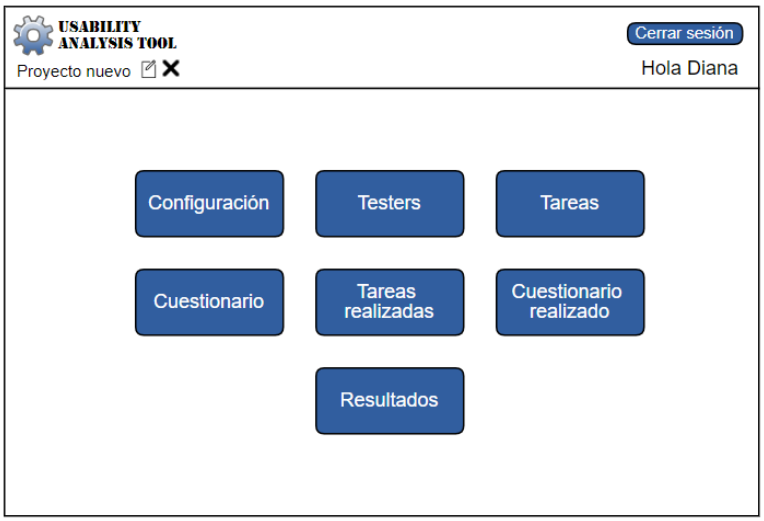


FIGURA 46: VER PROYECTO

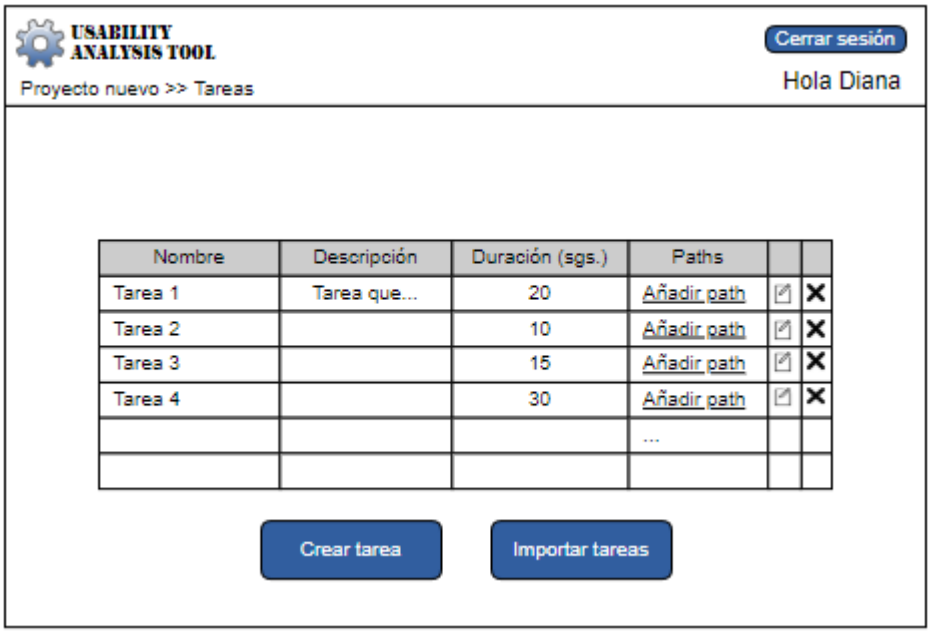


FIGURA 59:PÁGINA TAREAS

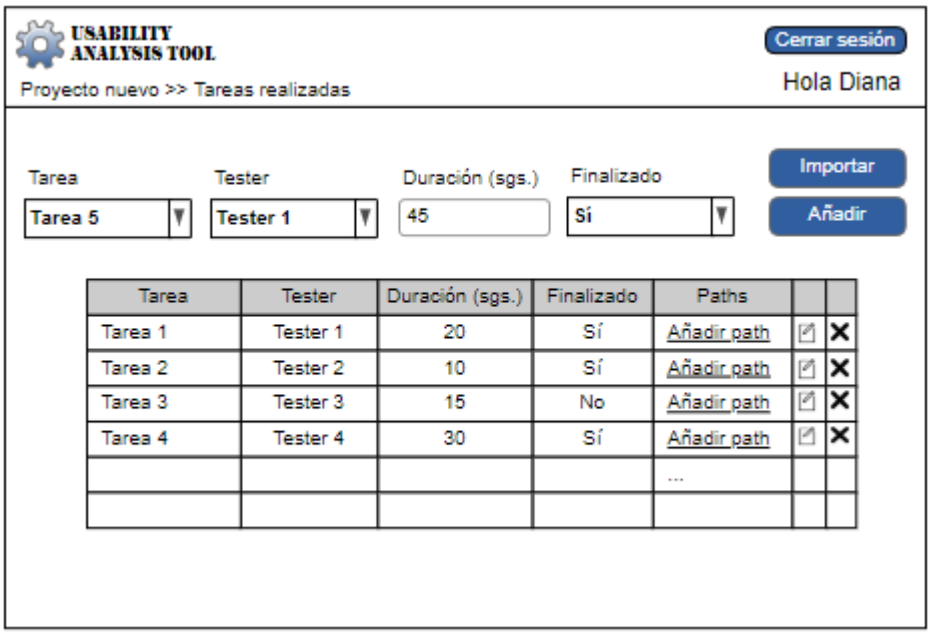
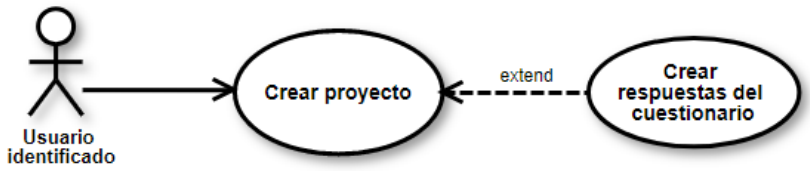


FIGURA 62: PÁGINA TAREAS REALIZADAS

FIGURA 58: IMPORTAR FICHERO

10.9 Crear respuestas del cuestionario



Nombre:	Crear respuestas del cuestionario

Descripción: el usuario puede crear las respuestas que han dado los testers a cada pregunta del cuestionario.

Actores: usuario identificado

Precondiciones: tener creado el proyecto, el cuestionario y sus respectivas preguntas.

Requisitos no funcionales: ninguno

Flujo de eventos:

1. El usuario se encuentra en la página *Responder Cuestionario (*Figura 64*)*. Selecciona los datos necesarios para crear la respuesta y hace click en *Añadir*
   [Si todo es correcto]
   2.a.- La respuesta se añade al final de la tabla de la misma página.
   [Si no ha completado alguno de los campos]
   3.a.- Se muestra un mensaje indicando que todos los campos son requeridos.

Postcondiciones: se crean las respuestas del cuestionario en la base de datos y se actualiza la tabla de "Responder cuestionario"

Interfaz gráfica:

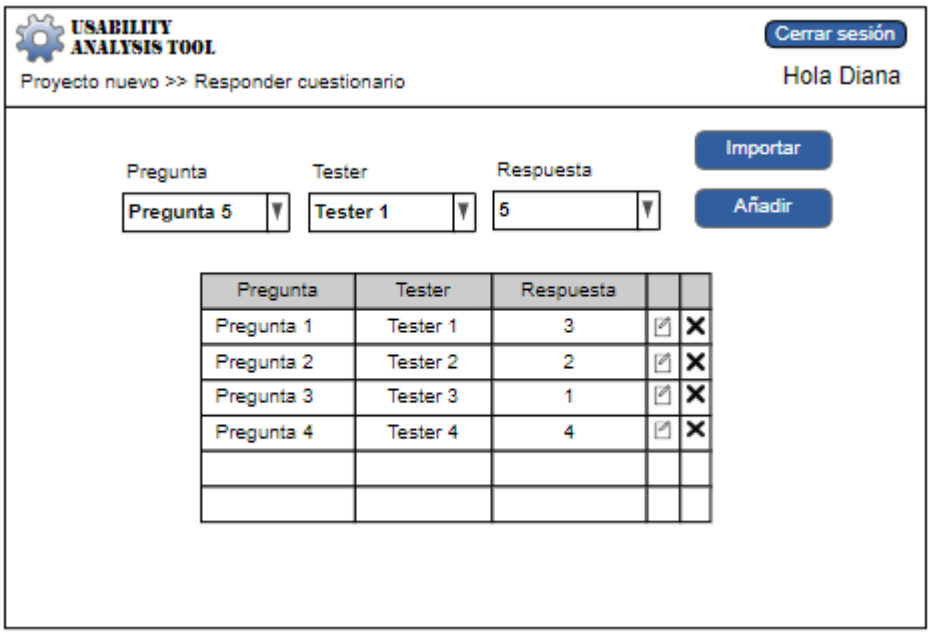
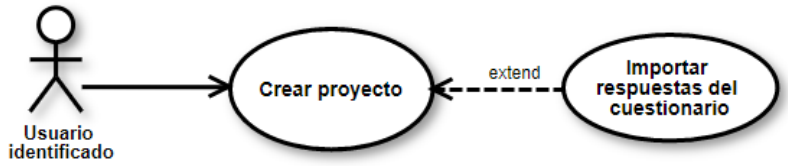


FIGURA 64: PÁGINA RESPONDER CUESTIONARIO

10.10 Importar respuestas del cuestionario



Nombre:	Importar respuestas del cuestionario

Descripción: permite importar las respuestas de un cuestionario de un fichero en formato CSV. Es necesario un formato específico. Ver ANEXO III

Actores: usuario identificado

Precondiciones: tener creado un proyecto, el cuestionario y las preguntas.

Requisitos no funcionales: ninguno

Flujo de eventos:

1. El usuario se encuentra en la página *Responder Cuestionario (*Figura 64*)*. Hace click en *Importar* y se muestra la Figura 58.
   [Si todo es correcto]
   2.a.- La tabla que está en la misma página, se actualiza con los nuevos datos.
   [Si no ha completado alguno de los campos]

3.a.- Se muestra un mensaje indicando el error.

Postcondiciones: se crean las respuestas del cuestionario en la base de datos y se actualiza la tabla de "Responder cuestionario"

Interfaz gráfica:

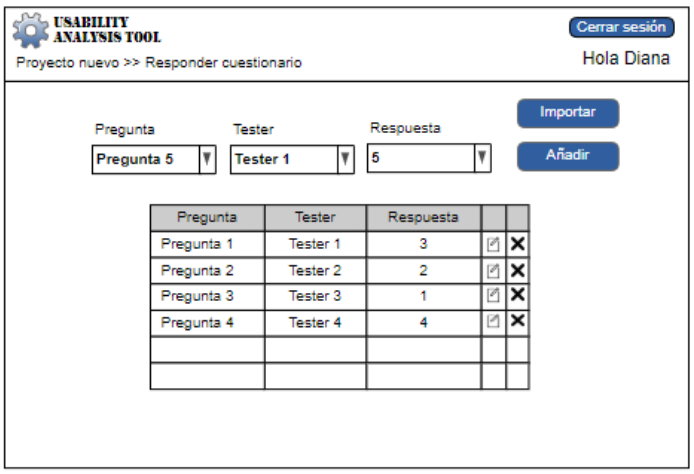
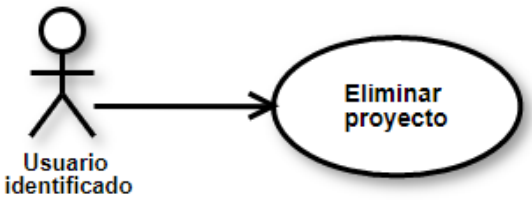


FIGURA 64: PÁGINA RESPONDER CUESTIONARIO

FIGURA 58: IMPORTAR FICHERO

10.11 Eliminar proyecto



Nombre:	Eliminar proyecto

Descripción: permite al usuario eliminar un proyecto, al eliminar un proyecto el usuario elimina toda la información relacionada con ese proyecto.

Actores: usuario identificado

Precondiciones: Tener creado el proyecto que se quiere eliminar

Requisitos no funcionales: ninguno

Flujo de eventos:

1. El usuario hace click en el icono de eliminar en la Figura 46. Se muestra un mensaje indicando si realmente quiere borrar el proyecto. Hace click en sí.
   [Si se elimina correctamente]
   2.a.- El proyecto se elimina de la lista de proyectos.
   [Sino]
   3.a.- Se muestra un mensaje de error.

Postcondiciones: el proyecto se elimina de la base de datos, así como todo lo relacionado con él y por lo tanto ya no es visible en la interfaz

Interfaz gráfica:

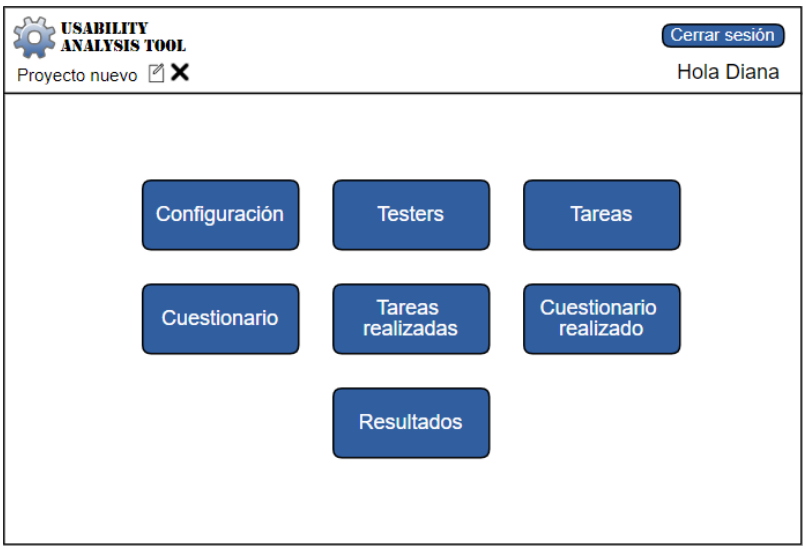


FIGURA 46: VER PROYECTO

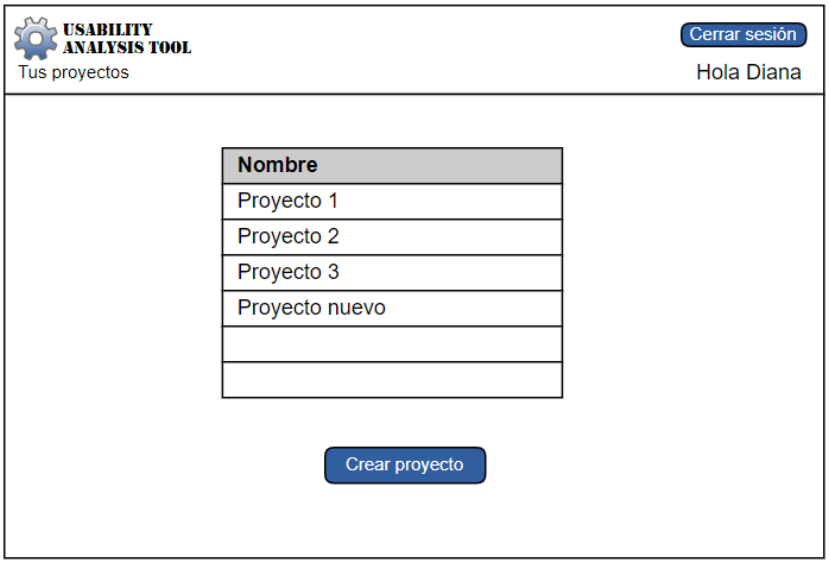
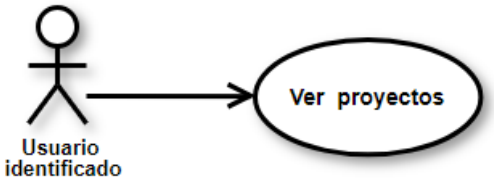


FIGURA 43: PÁGINA PROYECTOS

10.12 Ver proyectos



Nombre:	Ver proyectos

Descripción: cuando el usuario inicia sesión la primera página que se le muestra es la de los proyectos que tiene creados, también se muestran los proyectos al hacer click en el logo de la aplicación.

Actores: usuario identificado

Precondiciones: ninguna

Requisitos no funcionales: ninguno

Flujo de eventos:

1. El usuario ha hecho click en "Iniciar sesión"
   [Si el inicio de sesión es correcto y tiene proyectos creados]
   2.a Se cargan los proyectos que tiene creados en una tabla.
   [Si el inicio de sesión es correcto y no tiene proyectos creados]
   2.a Se muestra una tabla vacía
   3. Si se encuentra en otra página y hace click en el logo de la aplicación
   4.a. Se muestra la Figura 43

Postcondiciones: se muestran los proyectos que el usuario tenga creados.

Interfaz gráfica:

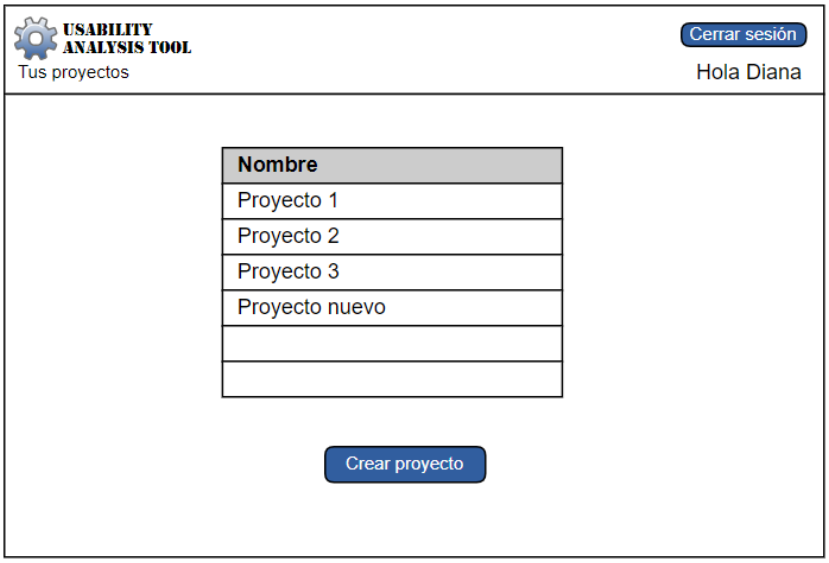
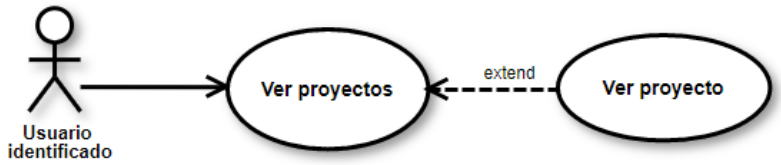


FIGURA 43: PÁGINA PROYECTOS

10.13 Ver proyecto



Nombre:	Ver proyecto

Descripción: Se trata básicamente de visualizar el contenido de un proyecto

Actores: usuario identificado

Precondiciones: ninguna

Requisitos no funcionales: ninguno

Flujo de eventos:

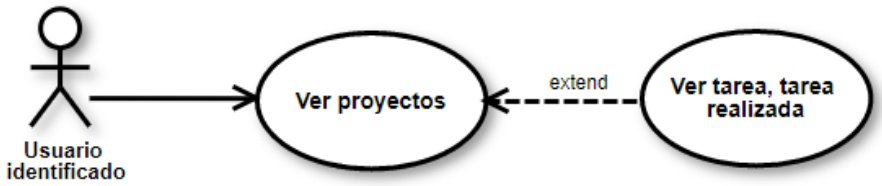
1. El usuario se encuentra en la página proyectos, y hace doble click en uno de ellos, se muestra la Figura 46.

Postcondiciones: se muestra el menú del proyecto

Interfaz gráfica:

FIGURA 46: VER PROYECTO

10.14 Ver tarea, tarea realizada



Nombre:	Ver tarea, tarea realizada

Descripción: permite al usuario visualizar las tareas y/o tareas realizadas.

Actores: usuario identificado

Precondiciones: ninguna

Requisitos no funcionales: ninguno

Flujo de eventos:

1. El usuario hace doble click en el proyecto, se muestra la Figura 46 y luego click en "Tareas" o "Tareas realizadas" de esa misma figura.
   [Si hace click en "Tareas"]
   2.a.-Se muestra la Figura 59.
   [Si hace click en "Tareas Realizadas"]
   2.a.-Se muestra la Figura 62.

Postcondiciones: se muestran las tareas o tareas realizadas que pudiera tener creado el usuario.

Interfaz gráfica:

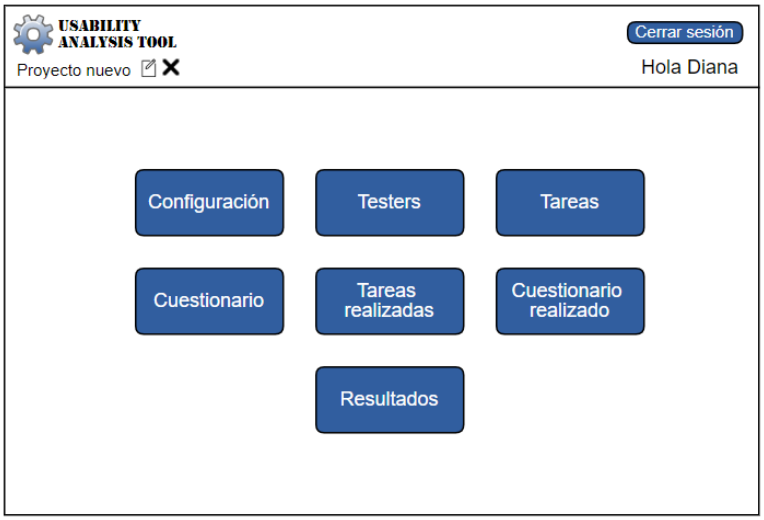


FIGURA 46: VER PROYECTO

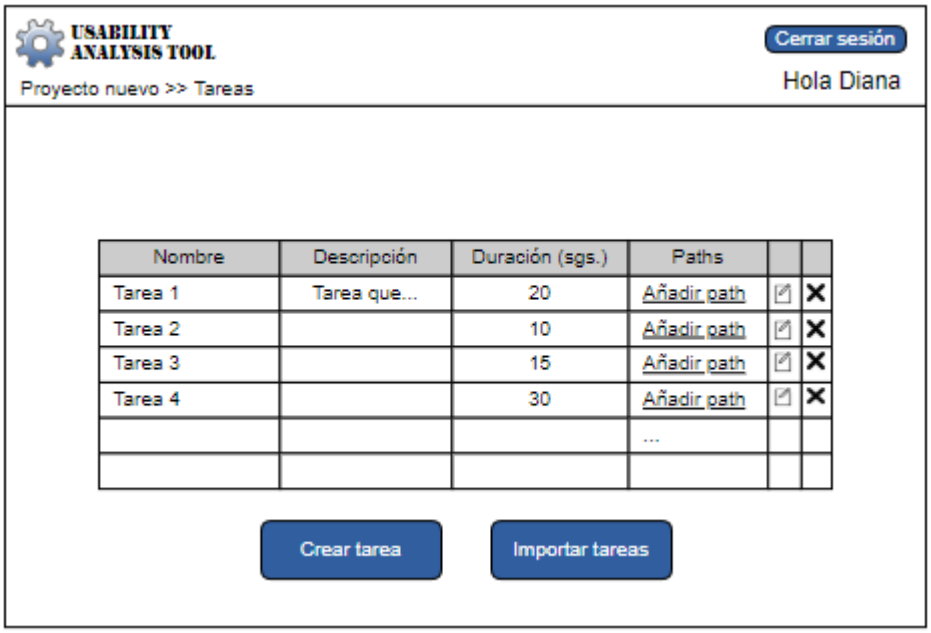


FIGURA 59:PÁGINA TAREAS

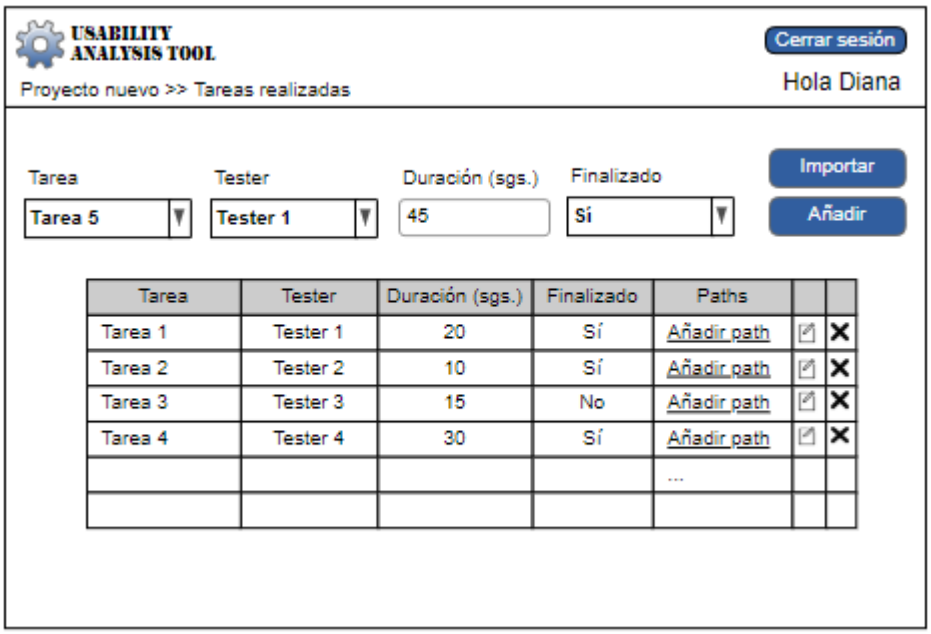
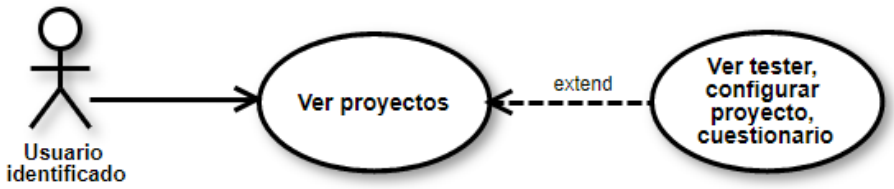


FIGURA 62: PÁGINA TAREAS REALIZADAS

10.15 Ver tester configurar, proyecto, cuestionario



Nombre:	Ver tester, configurar proyecto, cuestionario

Descripción: permite al usuario visualizar los testers, configurar el proyecto o el cuestionario.

Actores: usuario identificado

Precondiciones: ninguna

Requisitos no funcionales: ninguno

Flujo de eventos:

1. El usuario hace doble click en el proyecto, se muestra la Figura 46 y luego click en "Testers" o "Configuración" o "Cuestionario" de esa misma figura.
   [Si hace click en "Testers"]
   2.a.- Se muestra la Figura 49.
   [Si hace click en "Configuración"]
   3.a.- Se muestra la Figura 48.
   [Si hace click en "Cuestionario"]
   4.a.- Se muestra la Figura 50, y si hace doble click sobre un cuestionario, se muestra la Figura 56.

Postcondiciones: se muestran los testers, configuración o cuestionario que pudiera tener creado el usuario.

Interfaz gráfica:

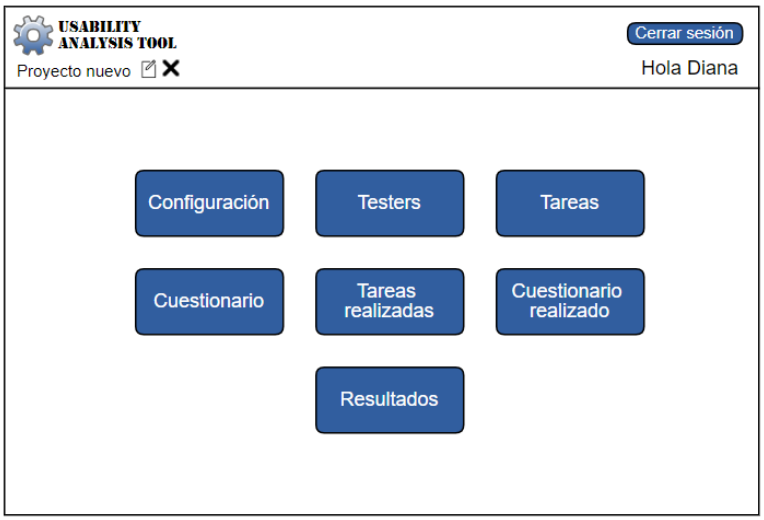


FIGURA 46: VER PROYECTO

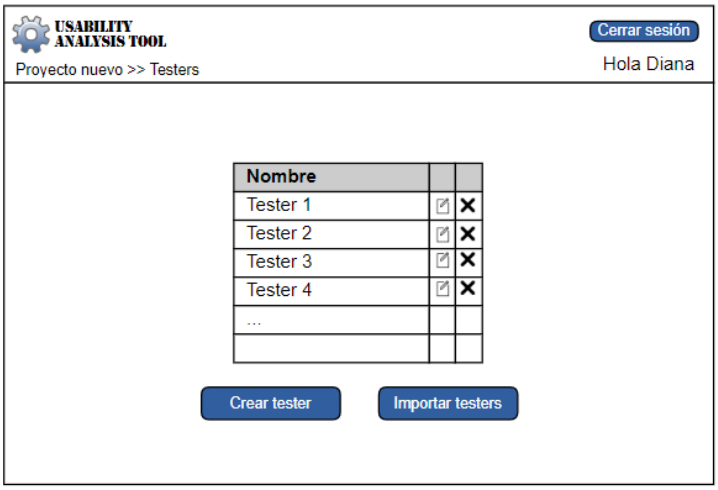


FIGURA 49: PÁGINA TESTERS

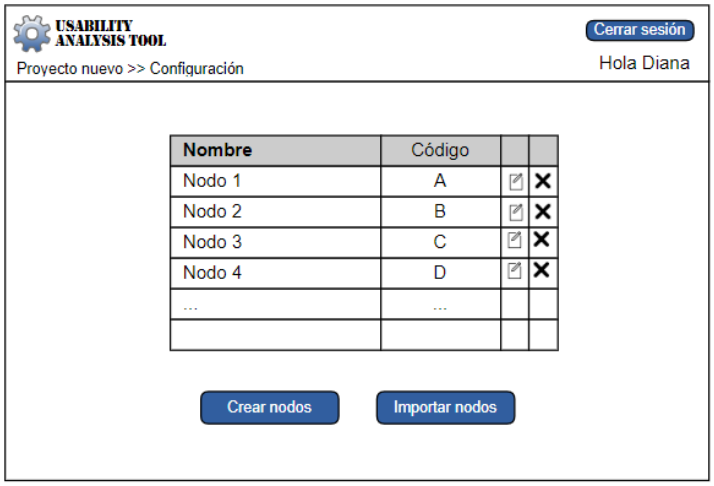


FIGURA 48: PÁGINA CONFIGURACIÓN

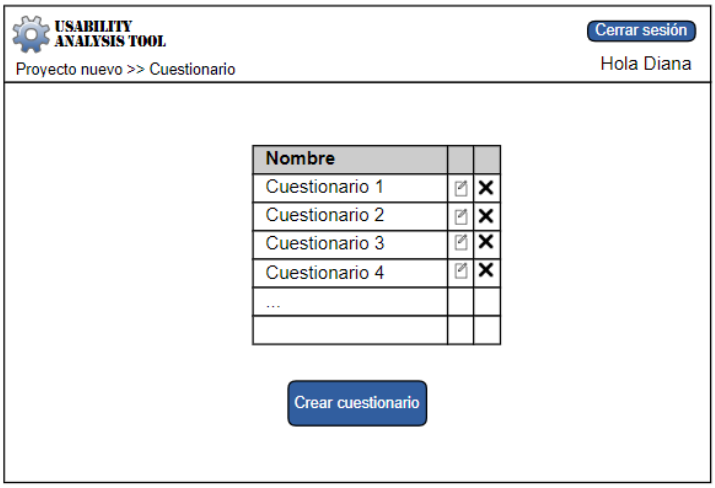


FIGURA 50: PÁGINA CUESTIONARIOS

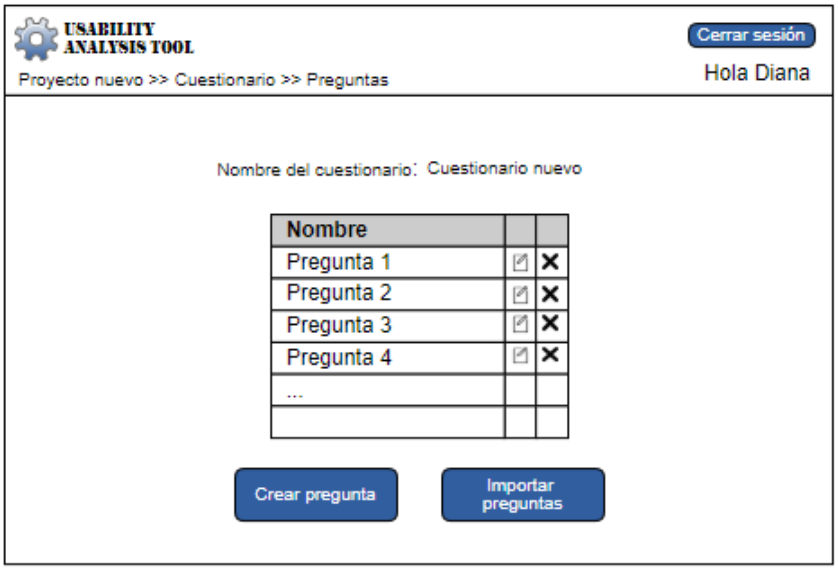


FIGURA 56: PÁGINA PREGUNTAS

10.16 Ver respuestas del cuestionario

Nombre:	Ver respuestas del cuestionario

Descripción: permite al usuario visualizar las respuestas del cuestionario

Actores: usuario identificado

Precondiciones: tener creado el cuestionario y las preguntas

Requisitos no funcionales: ninguno

Flujo de eventos:

1. El usuario hace doble click en el proyecto, se muestra la Figura 46 y luego click en "Cuestionario realizado" de esa misma figura.
   [Si hace click en "Cuestionario realizado"]
   2.a.- Se muestra la Figura 64.

Postcondiciones: se muestran las respuestas del cuestionario, que el usuario haya podido crear.

Interfaz gráfica:

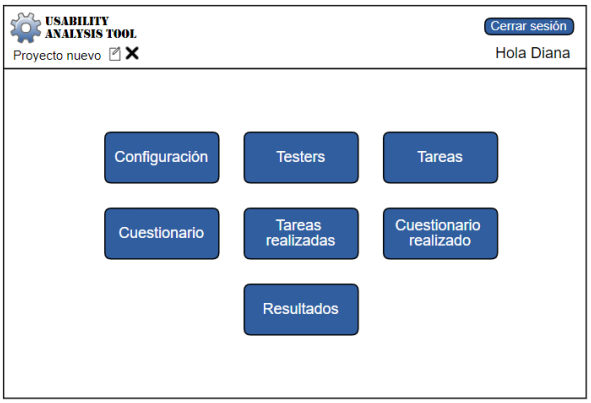


FIGURA 46: VER PROYECTO

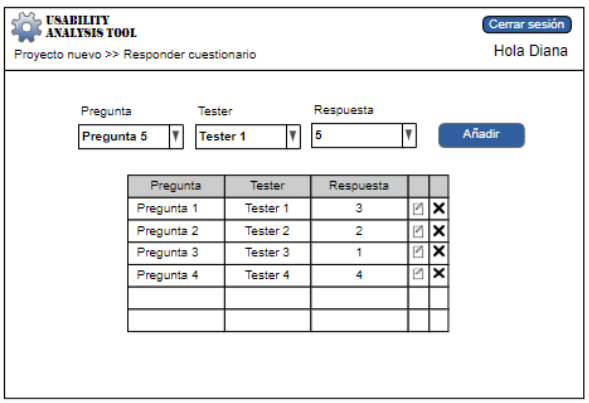


FIGURA 64: PÁGINA RESPONDER CUESTIONARIO

10.17 Ver resultados

Nombre:	Registrarse

Descripción: permite al usuario visualizar los resultados

Actores: usuario identificado

Precondiciones: tener creadas las tareas, tareas realizadas, nodos, testers, preguntas y respuestas del cuestionario.

Requisitos no funcionales: ninguno

Flujo de eventos:

1. El usuario hace doble click en el proyecto, se muestra la Figura 46 y luego click en "Resultados" de esa misma figura.
   [Si hace click en "Resultados"]
   2.a.- Se muestra la Figura 65.

Postcondiciones: el usuario tiene la posibilidad de elegir el resultado que desee

Interfaz gráfica:

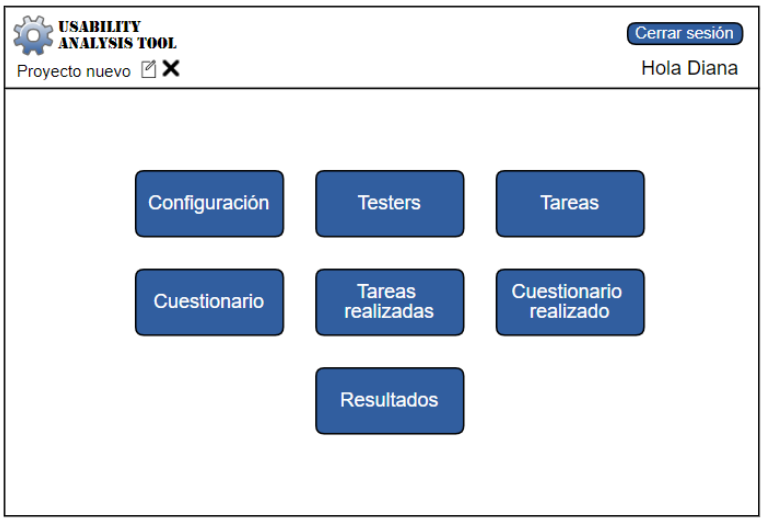
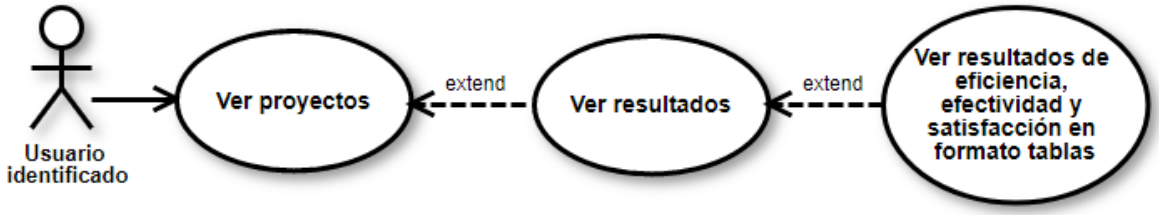


FIGURA 46: VER PROYECTO

FIGURA 65: PANTALLA VER RESULTADOS

10.18 Ver resultados de eficiencia, efectividad y satisfacción en formato tablas



Nombre:	Ver resultados de eficiencia, efectividad y satisfacción en formato tablas

Descripción: con esta funcionalidad el usuario puede ver los resultados obtenidos de los datos del proyecto.

Actores: usuario identificado

Precondiciones: tener creadas las tareas, tareas realizadas, nodos, testers, preguntas y respuestas del cuestionario.

Requisitos no funcionales: ninguno

Flujo de eventos:

1. El usuario ha hecho click en "Resultados", se muestra la Figura 65 y luego, hace click en "Efectividad" o "Eficiencia" o "Satisfacción" de esa misma figura.
   [Si hace click en "Efectividad"]
   2.a.- Se muestra la Figura 66.
   [Si hace click en "Eficiencia"]
   3.a.- Se muestra la Figura 67.
   [Si hace click en "Satisfacción"]
   4.a.- Se muestra la Figura 68.

Postcondiciones: el usuario visualiza los resultados de la opción que haya elegido.

Interfaz gráfica:

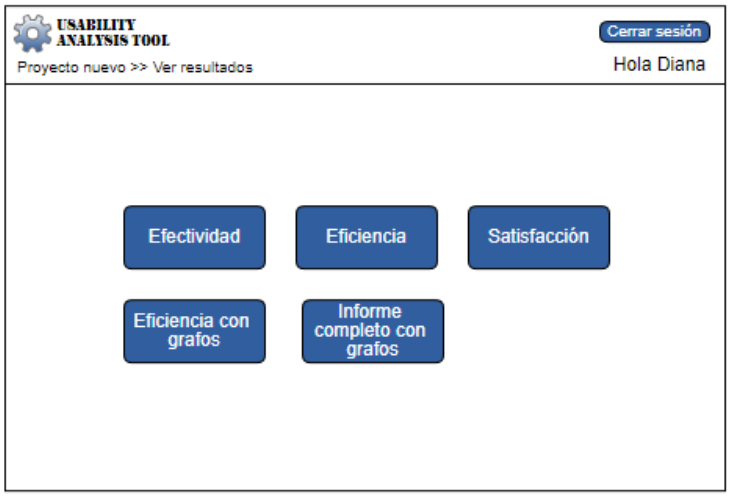


FIGURA 65: PANTALLA VER RESULTADOS

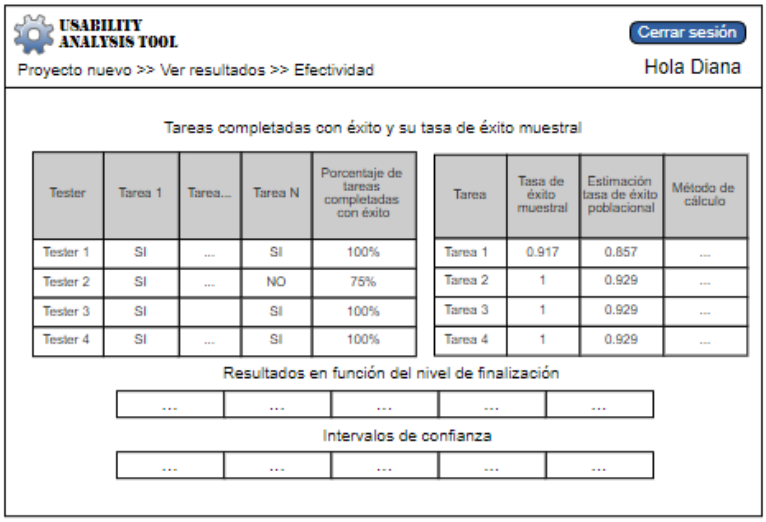


FIGURA 66: PÁGINA EFECTIVIDAD

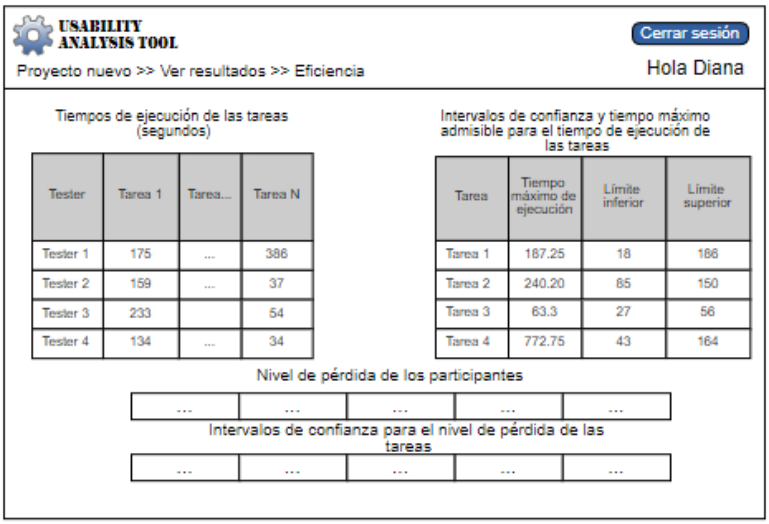


FIGURA 67: PÁGINA EFICIENCIA

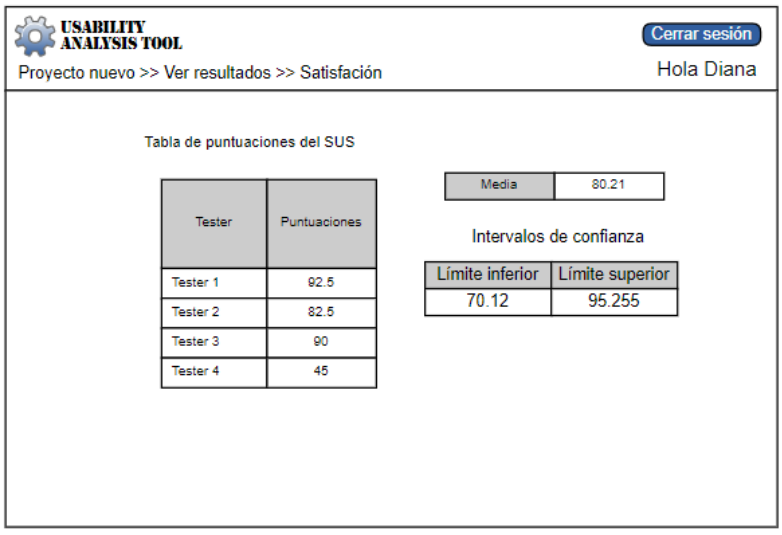
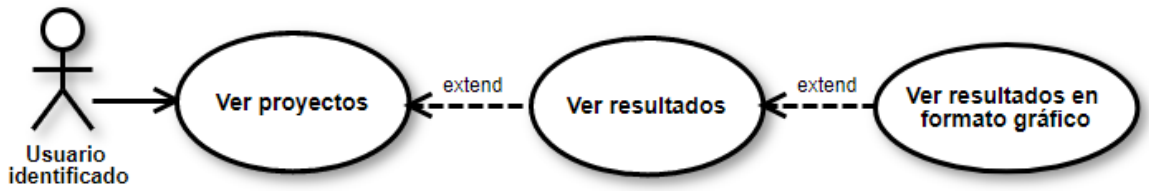


FIGURA 68: PÁGINA SATISFACCIÓN

10.19 Ver resultados en formato gráfico



Nombre:	Registrarse

Descripción: permite visualizar los resultados de la eficiencia con un grafo por cada tarea que tenga el proyecto

Actores: usuario identificado

Precondiciones: tener creadas las tareas, tareas realizadas, nodos, testers, preguntas y respuestas del cuestionario.

Requisitos no funcionales: ninguno

Flujo de eventos:

1. El usuario ha hecho click en "Resultados", se muestra la Figura 65.
   [Si hace click en "Eficiencia con grafos"]
   4.a.- Se muestra la Figura 69.

Postcondiciones: el usuario puede ver los resultados.

Interfaz gráfica:

FIGURA 65: PANTALLA VER RESULTADOS

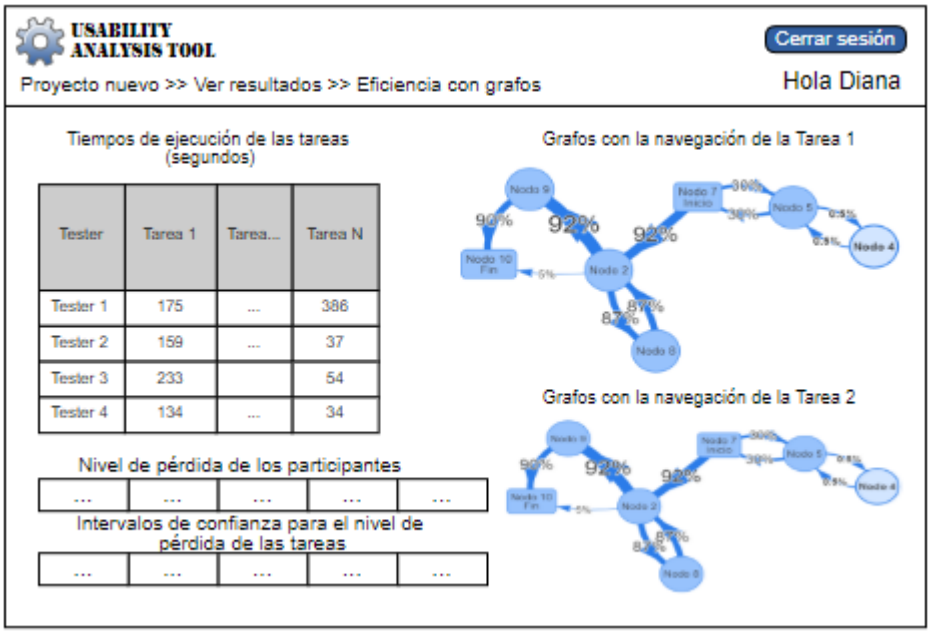
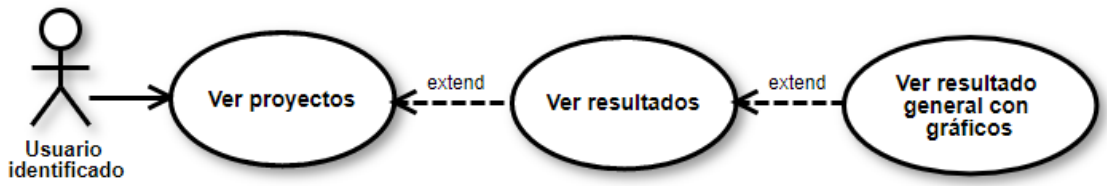


FIGURA 69: PÁGINA EFICIENCIA CON GRAFOS

10.20 Ver resultado general con gráficos



Nombre:	Registrarse

Descripción: permite al usuario visualizar los resultados incluyendo los gráficos

Actores: usuario identificado

Precondiciones: tener creadas las tareas, tareas realizadas, nodos, testers, preguntas y respuestas del cuestionario.

Requisitos no funcionales: ninguno

Flujo de eventos:

1. El usuario ha hecho click en "Resultados", se muestra la Figura 65.
   [Si hace click en "Informe completo con grafos"]
   4.a.- Se muestra Figura 70.

Postcondiciones: el usuario visualiza los resultados de los tres objetivos de la usabilidad incluyendo los gráficos.

Interfaz gráfica:

FIGURA 65: PANTALLA VER RESULTADOS

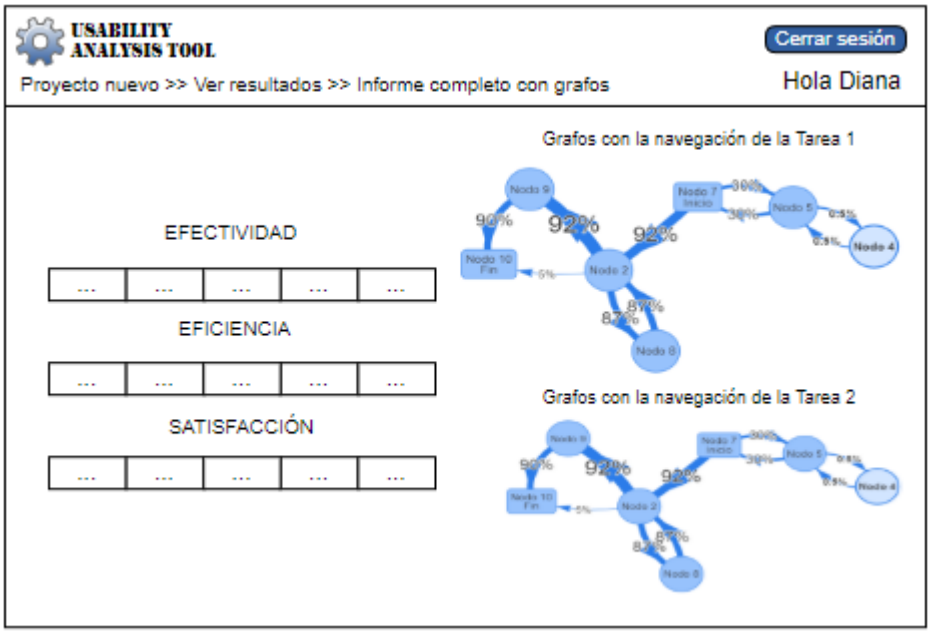
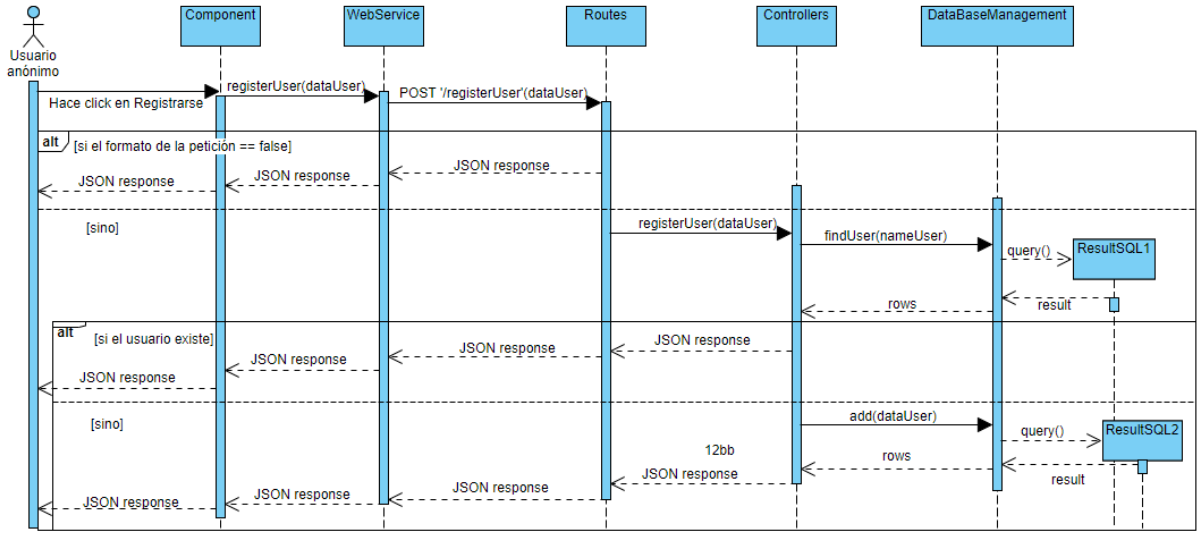


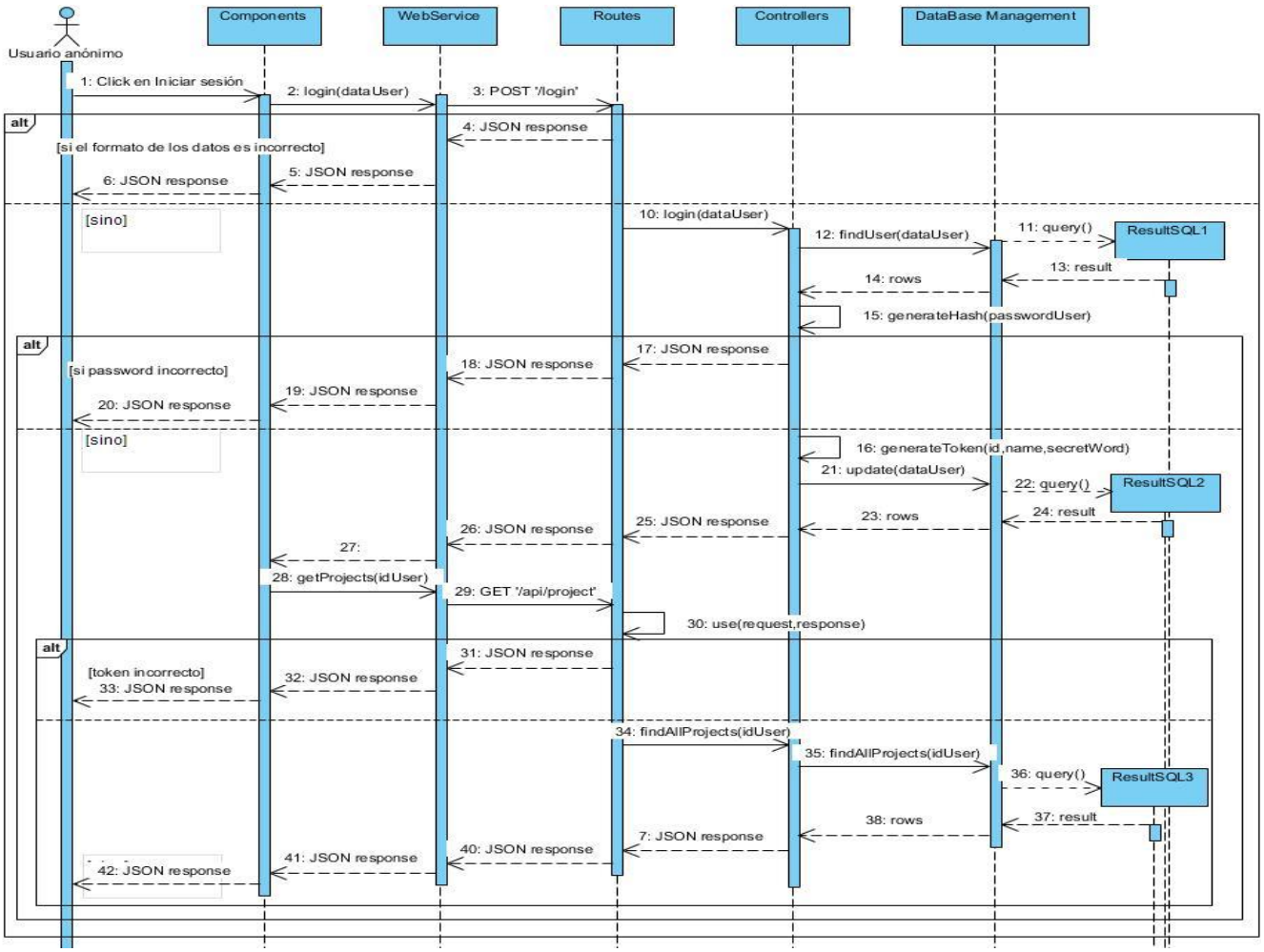
FIGURA 70: PÁGINA INFORME COMPLETO CON GRAFOS

11. ANEXO II.- DIAGRAMAS DE SECUENCIA

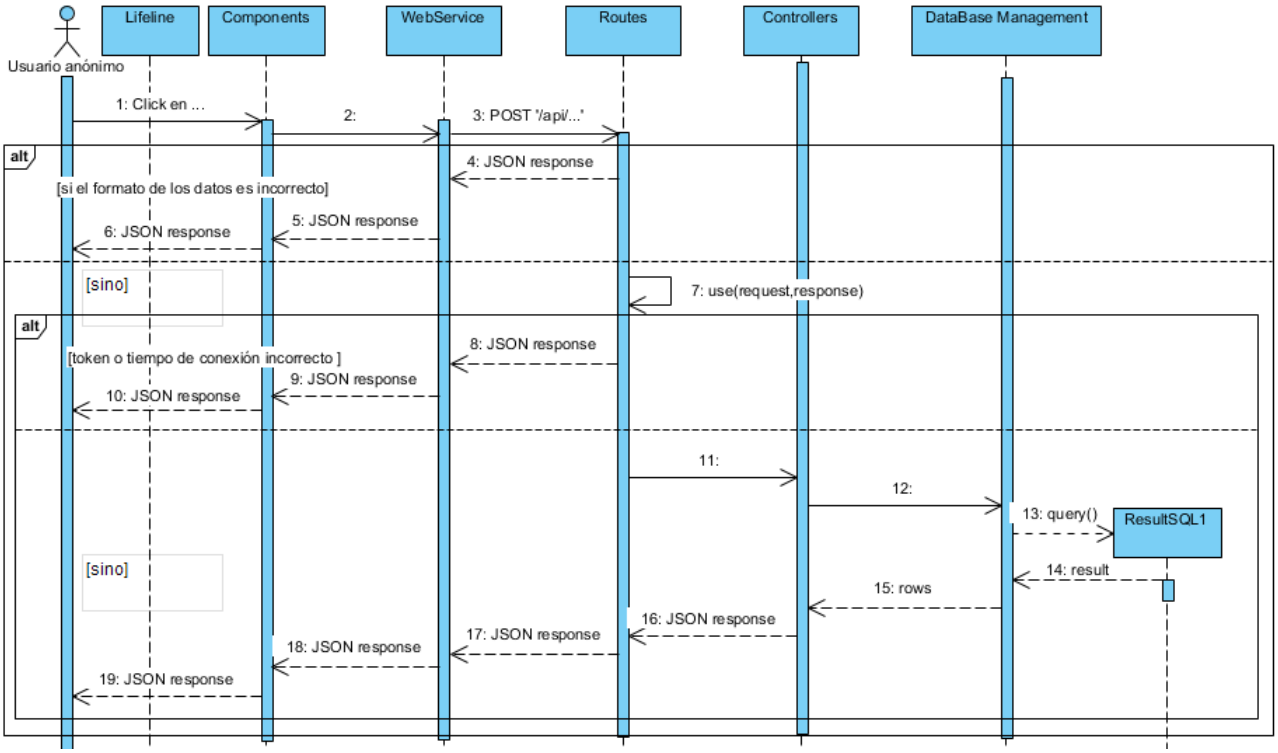
11.1 Registro de usuario



11.2 Iniciar sesión



11.3 Obtener, crear, modificar, eliminar objeto



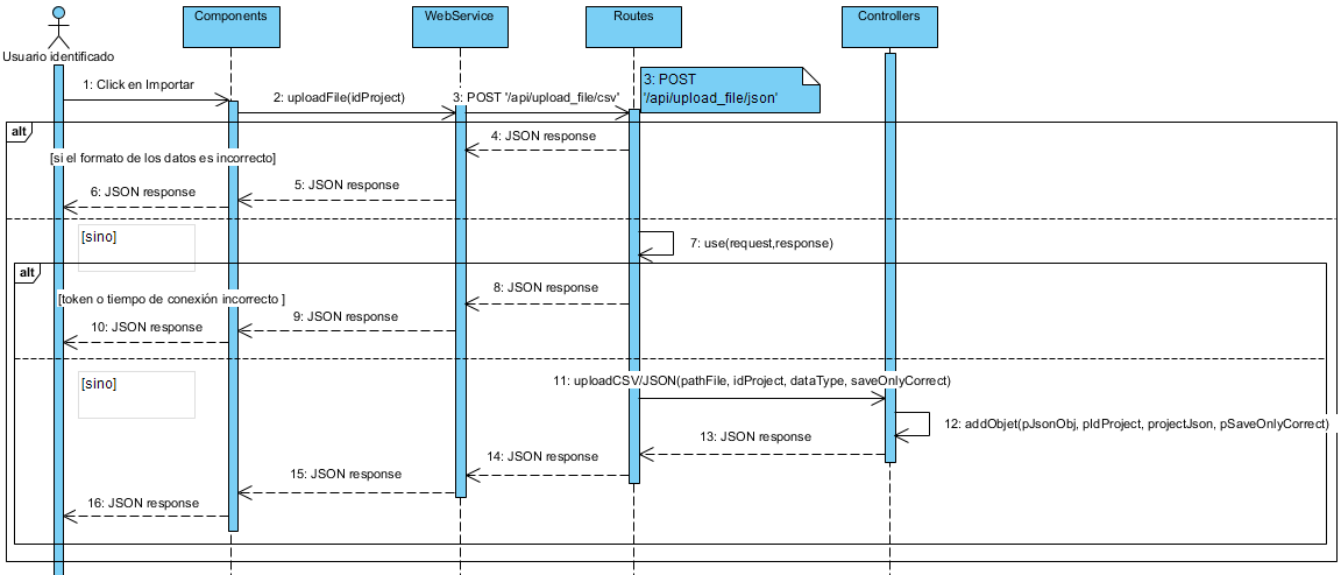
1. Click en Crear, Modificar, Eliminar de cualquiera de las páginas (Proyecto, Tareas, Testers, Configuración, Cuestionarios, Preguntas, Tareas realizadas y Cuestionario realizado), o en el caso de obtener sólo hace falta entrar a la página que se quiera ver.
2. createObject(dataObject)
   updateObject (dataObject)
   deleteObject (idObject)
   getObject(idObject)
3. Para crear POST
   Para modificar PUT
   Para eliminar DELETE
   Para obtener GET
   En todos los casos se accede a los recursos con las siguientes URL's:
   a) '/api/project' → para proyecto.
   b) '/api/project/nodes' → para los nodos.
   c) '/api/task' → para las tareas.
   d) '/api/task/paths' → para los paths de las tareas.
   e) '/api/tester' → para los testers.
   f) '/api/questionary → para el cuestionario.
   g) '/api/ questionary/questions' → para los testers.
   h) '/api/questionAnswered' → para los testers.
   i) '/api/taskDone' → para las tareas realizadas.
   j) '/api/taskDone/pathDone' → para los paths realizados de las tareas realizadas.

11. Se ejecuta el método que sea:
    createObject(dataObject)
    updateObject (dataObject)
    deleteObject (idObject)
    findAllObject(idObject)

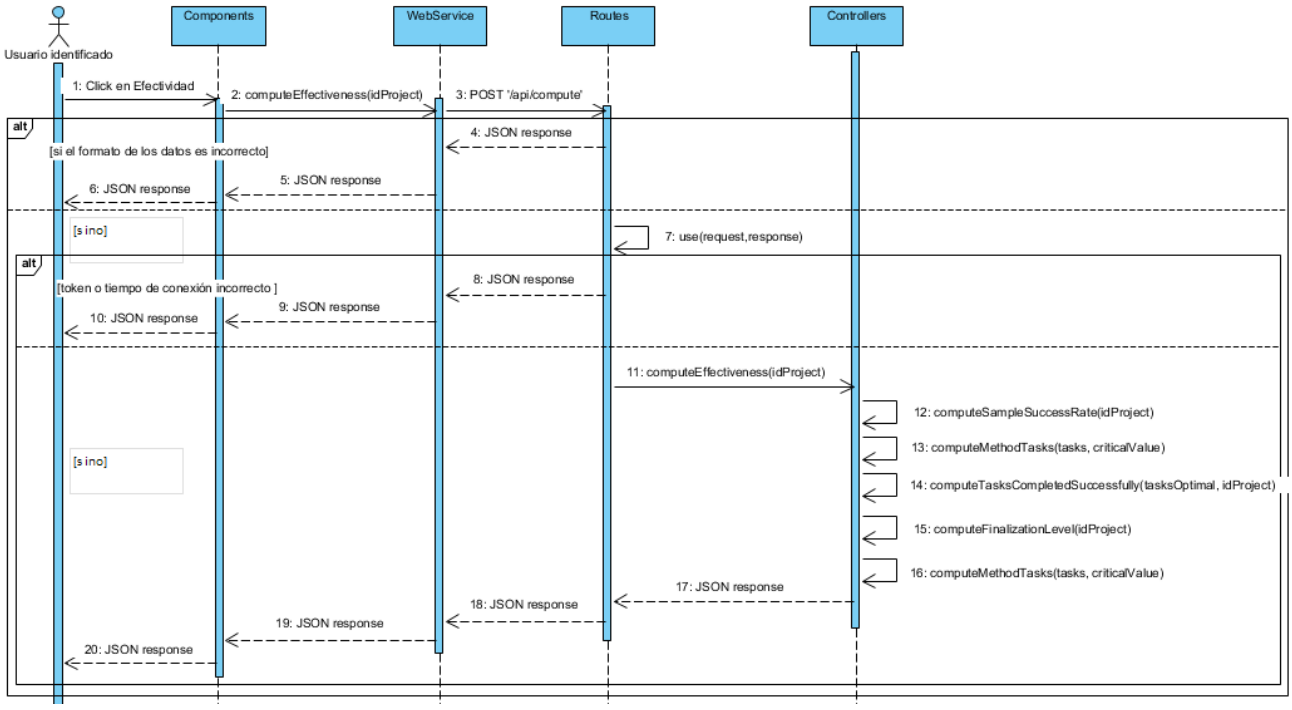
12. Ejecutará un select, update, delete o insert into, según lo que se necesite

En los siguientes diagramas no se ve la comunicación entre *Controllers* y *DatabaseManagement*, puesto que contienen funciones que hacen esa comunicación, único que se hace es llamar a esa función. Para ver esa comunicación habría que ahondar en cada una de ellas, lo cual significaría desarrollar una gran cantidad de diagramas de secuencia.

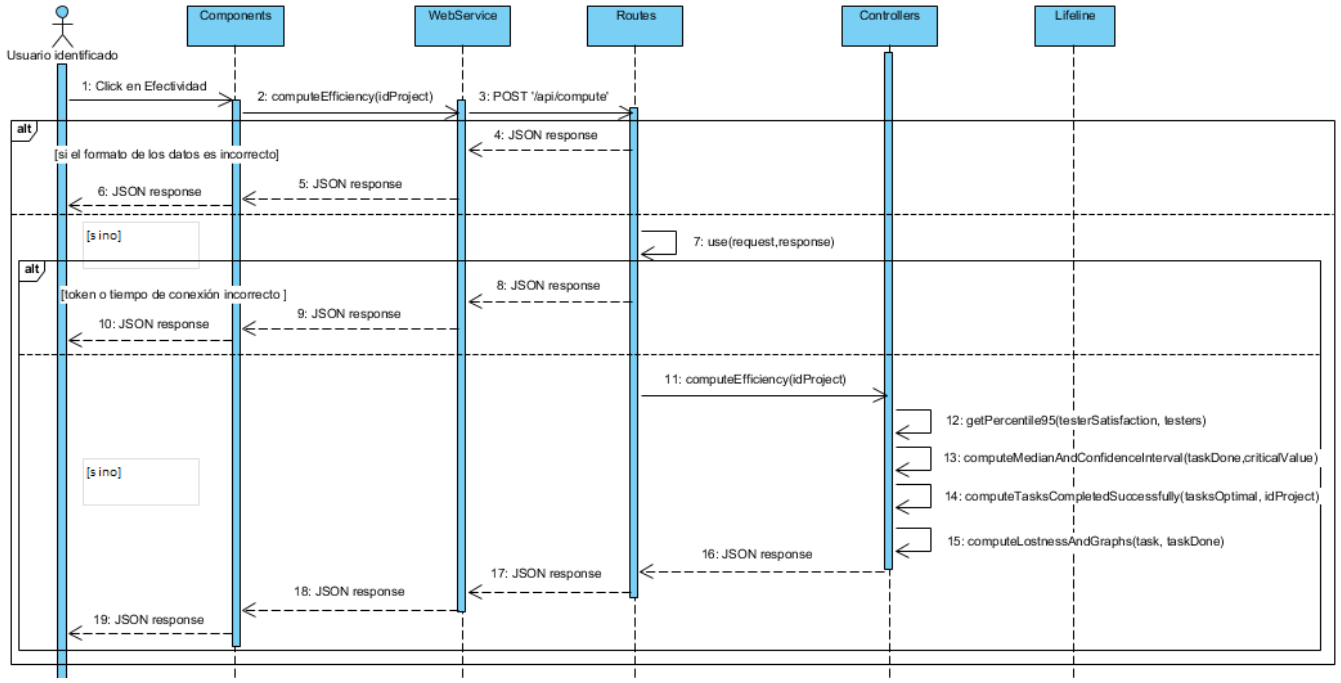
11.4 Cargar fichero



11.5 Calcular la efectividad



11.6 Calcular la eficiencia



11.7 Calcular la satisfacción